

**UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E
ENGENHARIAS**

DANIEL CARNEIRO BALBINOT

**PROPOSTA E IMPLEMENTAÇÃO DE UMA SOLUÇÃO PARA
AUTOMAÇÃO RESIDENCIAL**

BENTO GONÇALVES

2019

DANIEL CARNEIRO BALBINOT

**PROPOSTA E IMPLEMENTAÇÃO DE UMA SOLUÇÃO PARA
AUTOMAÇÃO RESIDENCIAL**

Trabalho de Conclusão de Curso
apresentado como requisito parcial à
obtenção do título de Bacharel em
Ciência da Computação na Área do
Conhecimento de Ciências Exatas
e Engenharias da Universidade de
Caxias do Sul.

Orientador: Prof. Dr. Ricardo
Vargas Dorneles

BENTO GONÇALVES

2019

DANIEL CARNEIRO BALBINOT

**PROPOSTA E IMPLEMENTAÇÃO DE UMA SOLUÇÃO PARA
AUTOMAÇÃO RESIDENCIAL**

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Aprovado em 26/11/2019

BANCA EXAMINADORA

Prof. Dr. Ricardo Vargas Dorneles
Universidade de Caxias do Sul - UCS

Prof. Me. Alexandre Erasmo Krohn Nascimento
Universidade de Caxias do Sul - UCS

Prof. Me. Joacir Giaretta
Universidade de Caxias do Sul - UCS

RESUMO

A automação residencial, também conhecida como domótica, permite que as ações cotidianas na residência sejam realizadas com mais praticidade, conforto e segurança. Pelo crescimento da *IoT* (*Internet of Things*) ou *Internet* das Coisas, a automação residencial se tornou mais acessível para a população, mas isto inclui a criação de diversos dispositivos que podem possuir protocolos diferentes, necessitando uma central de automação que possa garantir a funcionalidade total destes protocolos. Este trabalho tem por objetivo apresentar uma proposta de solução de uma automação residencial. Para tal, planejou-se uma central de automação que consiga comunicar com os sensores e atuadores de forma sem fio, garantindo a funcionalidade deles. Também foi planejado um aplicativo de *smartphone* para poder ser utilizado como controle remoto da central. Neste aplicativo o usuário poderá realizar diversos tipos de comandos, a fim de facilitar o seu uso, incluindo a integração com comandos de voz. Também foi projetado uma API RESTful para poder realizar a comunicação entre o aplicativo e a central através do protocolo HTTP, juntamente com um servidor que utiliza o protocolo MQTT para realizar a comunicação entre a central e os dispositivos. Para a validação dos objetivos. Serão verificados se todas as funcionalidades propostas foram atingidas, incluindo as funcionalidades do aplicativo e a integridade da central de automação.

Palavras-chaves: Automação residencial, Dispositivos, Sensores, MQTT, HTTP, IoT, Comunicação sem fio

ABSTRACT

Home automation, also known as domotics, allows everyday actions at home to be carried out with more convenience, comfort and safety. By the growth of Internet of Things or IoT, home automation has become more accessible to the population, but this includes the creation of multiple devices that may have different protocols, requiring an automation center that can guarantee the full functionality of these protocols. This paper aims to present a solution proposal for a home automation. To this end, an automation center has been designed that can communicate wirelessly with sensors and actuators, ensuring their functionality. A smartphone application was also planned to be used as a remote control of the switch. In this application the user can perform various types of commands in order to facilitate their use, including integration with voice commands. A RESTful API was also designed to be able to communicate between the application and the switch through the HTTP protocol, along with a server that uses the MQTT protocol to communicate between the switch and the devices. For the validation of objectives. They will verify that all proposed functionality has been achieved, including application functionality and the integrity of the automation center.

Keywords: Home automation, Devices, Sensors, MQTT, HTTP, IoT, Wireless Communication

LISTA DE ILUSTRAÇÕES

Figura 1 – Tipos de redes classificadas por distribuição geográfica	19
Figura 2 – Possibilidades de utilização da rede NFC	21
Figura 3 – Exemplo de componentes da rede BAN	21
Figura 4 – Exemplo de uma rede PAN	22
Figura 5 – Exemplo de uma rede LAN	22
Figura 6 – Exemplo de uma rede CAN	23
Figura 7 – Exemplo de uma rede MAN	24
Figura 8 – Topologia de uma rede WAN	24
Figura 9 – Integração de uma rede 6LoWPAN com a <i>internet</i>	26
Figura 10 – Exemplo de uma arquitetura utilizando ZigBee	27
Figura 11 – Exemplo de uma arquitetura LoRaWAN	28
Figura 12 – Exemplo do funcionamento do MQTT	29
Figura 13 – Estrutura da arquitetura do trabalho	36
Figura 14 – <i>Market Share</i> de servidores nos sites ativos	39
Figura 15 – Módulos da família do chip ESP8266	40
Figura 16 – Integração dos componentes para realização de <i>login</i> no aplicativo	42
Figura 17 – Tela de <i>login</i> dentro do aplicativo	42
Figura 18 – Diagrama de atividades de comando para criação de um registro dentro do aplicativo	43
Figura 19 – Tela de um dispositivo no aplicativo.	43
Figura 20 – Diagrama de atividades de comando para acender a luz através do aplicativo	44
Figura 27 – Diagrama de atividades da execução de um comando de voz através do aplicativo	45
Figura 31 – Consumo de energia demonstrado no aplicativo	46
Figura 32 – Diagrama de atividades da execução de um comando de voz através do aplicativo	46
Figura 21 – Listagens de dispositivos cadastrados no aplicativo.	47
Figura 22 – Criação de um dispositivo no aplicativo.	47
Figura 23 – Edição de um dispositivo no aplicativo.	47
Figura 24 – Listagens de grupos cadastrados no aplicativo.	48
Figura 25 – Criação de um grupo no aplicativo.	48
Figura 26 – Edição de um grupo no aplicativo.	48
Figura 28 – Listagens de comandos cadastrados no aplicativo.	49
Figura 29 – Criação de um comando no aplicativo.	49
Figura 30 – Edição de um comando no aplicativo.	49

Figura 33 – Modelo entidade relacionamento	52
Figura 34 – Tela do aplicativo responsável por editar e cadastrar dispositivos	53
Figura 35 – Tela do aplicativo de criação e edição de comandos	54
Figura 36 – Notificação de alarme no <i>smartphone</i>	55
Figura 37 – Apresentação do <i>MQTT Lens</i>	56
Figura 38 – Ambiente de desenvolvimento <i>Arduino</i>	57
Figura 39 – Placas da família do <i>chip</i> ESP8266	58
Figura 40 – Placa FT232RL	58
Figura 41 – Módulo relé de 1 canal com ESP01	59
Figura 42 – Soldagem do módulo relé	59
Figura 43 – Sensor de fumaça	59
Figura 44 – Sensores de Movimento PIR	59
Figura 45 – Diagrama de pacotes da solução	60

LISTA DE ABREVIATURAS E SIGLAS

IoT	<i>Internet of Things</i>
NFC	<i>Near-Field Communication</i>
BAN	<i>Body Area Network</i>
PAN	<i>Personal Area Network</i>
LAN	<i>Local Area Network</i>
CAN	<i>Campus/Corporate Area Network</i>
MAN	<i>Metropolitan Area Network</i>
WAN	<i>Wide Area Network</i>
6LoWPAN	<i>IPv6 over Low-Power Wireless Personal Area Networks</i>
LoRa	<i>Long Range</i>
LPWAN	<i>Low Power Wide Area Network</i>
OSI	<i>Open Systems Interconnection</i>
MQTT	<i>Message Queuing Telemetry Transport</i>
M2M	<i>Machine to Machine</i>
HTTP	<i>Hypertext Transfer Protocol</i>
URI	<i>Uniform Resource Identifier</i>
CoAP	<i>Constrained Application Protocol</i>
AMQP	<i>Advanced Message Queuing Protocol</i>
MOM	<i>Message-oriented middleware</i>
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
RFID	<i>Radio-frequency identification</i>
UDP	<i>User Datagram Protocol</i>
TCP	<i>Transmission Control Protocol</i>
IP	<i>Internet Protocol</i>

API	<i>Application Programming Interface</i>
REST	<i>Representational State Transfer</i>
HATEOAS	<i>Hypermedia As The Engine Of Application State</i>
TTL	<i>Transistor-transistor logic</i>
LED	<i>Light-emitting diode</i>
OTA	<i>Over-the-Air</i>
PHP	<i>Hypertext Preprocessor</i>

SUMÁRIO

1	INTRODUÇÃO	15
1.1	OBJETIVOS	16
1.2	ESTRUTURA DO TRABALHO	16
2	ALTERNATIVAS DE COMUNICAÇÕES	19
2.1	CLASSIFICAÇÃO DE REDES PARA IOT POR DISTRIBUIÇÃO GEOGRÁFICA	19
2.1.1	Nanonetwork	20
2.1.2	NFC (Near-Field Communication)	20
2.1.3	BAN (Body Area Network)	20
2.1.4	PAN (Personal Area Network)	20
2.1.5	LAN (Local Area Network)	22
2.1.6	CAN (Campus/Corporate Area Network)	23
2.1.7	MAN (Metropolitan Area Network)	23
2.1.8	WAN (Wide Area Network)	23
2.2	REDES SEM FIO X REDES COM FIO	24
2.3	PROTOCOLOS DE CONEXÃO SEM FIO PARA IOT	25
2.3.1	WI-FI	25
2.3.2	BLUETOOTH	26
2.3.3	6LOWPAN	26
2.3.4	ZIGBEE	27
2.3.5	THREAD	27
2.3.6	LORAWAN	28
2.4	PROTOCOLOS DE COMUNICAÇÃO PARA IOT	28
2.4.1	MQTT	29
2.4.2	HTTP	29
2.4.3	CoAP	30
2.4.4	AMQP	30
2.5	API RESTFUL	31
2.6	PHP	32
2.7	Bibliotecas, <i>Frameworks</i> e ferramentas	33
2.7.1	Flutter	33
2.7.2	MQTT Lens	33
2.7.3	Laravel	33
2.7.4	Arduino IDE	34
3	ARQUITETURA DO SISTEMA	35

3.1	REQUISITOS FUNCIONAIS	35
3.2	CONFIGURAÇÕES DO SERVIDOR <i>WEB</i>	38
3.2.1	API RESTFUL	38
3.2.2	APACHE	38
3.2.3	<i>BROKER</i>	39
3.2.4	COMUNICAÇÃO COM O SERVIDOR <i>BROKER</i>	40
3.3	APLICATIVO PARA <i>SMARTPHONES</i>	41
4	IMPLEMENTAÇÃO	51
4.1	BANCO DE DADOS LOCAL	51
4.2	APLICATIVO PARA <i>SMARTPHONE</i>	51
4.3	SERVIDOR MQTT	54
4.4	CENTRAL DE AUTOMAÇÃO E API REST	56
4.5	SENSORES E ATUADORES	57
5	CONSIDERAÇÕES FINAIS	61
5.1	SUGESTÕES DE TRABALHOS FUTUROS	62
	REFERÊNCIAS	63

1 INTRODUÇÃO

Nos dias atuais, estamos vivendo em uma era onde a tecnologia se torna mais presente e está sempre evoluindo, simplificando ações do cotidiano das pessoas. Especificamente no âmbito residencial é possível utilizar esta tecnologia para tornar a residência inteligente, funcional e segura, através da domótica, ou automação residencial.

A palavra "domótica" deriva das palavras *domus* (casa) e robótica (controle automatizado de algo), e é definida como a integração entre mecanismos automáticos em um ambiente, com o objetivo de simplificar e facilitar desde ações simplórias do cotidiano como acender uma lâmpada ou controlar a temperatura, até em ações mais cautelosas, como controlar a segurança da residência através de câmeras ou fechaduras inteligentes.

Juntamente com a domótica, o termo *IoT* (*Internet of Things*) ou *Internet* das Coisas foi se popularizando e possui o objetivo de inserir "coisas" como aparelhos eletrônicos e eletrodomésticos no ambiente da *Internet*. Com isto será aprimorada a relação entre as "coisas" e os humanos, possibilitando a criação e otimização de serviços para melhorar a qualidade de vida e controlar melhor o gasto de recursos (BUYA; DASTJERDI, 2016).

Com esta evolução da tecnologia, a quantidade de aparelhos e controladores que possuem a finalidade de auxiliar nas ações do âmbito residencial está aumentando. Dentro deste contexto, a quantidade de protocolos diferentes que estes componentes utilizam também aumenta, junto com a complexidade de realizar a comunicação entre eles, pois para poder ter uma automação completa é necessário que todos os componentes consigam se comunicar com robustez e segurança, sem causar nenhuma dificuldade para os habitantes da residência. Deve-se facilitar a instalação e a utilização de toda a central de automação, como uma ideia máxima de *Plug and Play*, que neste contexto significa realizar apenas a instalação e o dispositivo já estar integrado na central de automação.

Existem diversas técnicas para se aplicar e controlar a central de automação, como os *softwares* integrados, mas com a crescente popularização dos *smartphones* é cada vez mais utilizado o uso de dispositivos como parte principal do painel de controle da automação, utilizando aplicativos simples, intuitivos e que consigam garantir a segurança contra qualquer pessoa mal-intencionada, podendo também permitir ao usuário visualizar a situação de sua residência em qualquer momento que o mesmo deseja, e em situações de risco notificar o mesmo de qualquer problema que possa ocorrer ou que já esteja ocorrendo.

Para a metodologia deste trabalho, foram realizadas diversas pesquisas sobre os tipos de protocolos de comunicação utilizadas na automatização residencial, considerando uma arquitetura que será composta por uma central, uma aplicação para *smartphone* e os dispositivos dispostos na residência. Também foram levantados os tipos de redes utilizados, as diferenças entre conexão cabeada ou sem fio e as alternativas de comunicação.

Dentro deste contexto, nesse trabalho foi desenvolvida uma central de automação para controlar todos os componentes que serão utilizados. Esta por sua vez está localizada em um servidor *web*, ao qual permitido o acesso remotamente através de um aplicativo de celular, que realiza uma autenticação para garantir que não é alguma pessoa que não deveria ter acesso a esta central para poder liberar o acesso. São guardados automaticamente registros informativos e de segurança para o usuário em um banco de dados local para possibilitar uma maior segurança. Também foi realizado um sistema de notificações para alertar os usuários no caso de alguma situação considerada de risco e permitirá a realização de comandos de voz utilizando o aplicativo para poder controlar a residência.

1.1 OBJETIVOS

Avaliar tecnologias existentes, projetar e implementar um sistema de automação residencial . Com base no objetivo geral, foram elaborados os seguintes objetivos específicos:

1. Desenvolvimento de uma central de automação que permita realizar comunicação remota utilizando *smartphone*
2. Desenvolvimento de integração da central de automação com reconhecimento de voz para a execução dos comandos
3. Desenvolvimento de um sistema de notificações para alertar os habitantes da residência em uma situação considerada de risco

1.2 ESTRUTURA DO TRABALHO

O presente trabalho está organizado da seguinte forma:

- No Capítulo 2 são apresentadas as diferentes formas de comunicação que podem ser utilizadas para se integrar com a central de automação garantindo o seu funcionamento desejado. Entre essas comunicações, irão se destacar as formas que serão utilizadas neste trabalho e os *gateways* que podem ser utilizados para se manter um padrão de comunicação.
- No Capítulo 3 é apresentada a arquitetura da central de automação que foi desenvolvida no trabalho, juntamente com os protocolos definidos para a comunicação e

os dispositivos e objetos residenciais que serão controlados. Também é apresentada como foi a integração dos comandos de voz juntamente com as notificações de alerta para o *smartphone*.

- No Capítulo 4 é apresentado como foi realizado o desenvolvimento de todos os componentes da central de automação, incluindo ferramentas, bibliotecas e *frameworks* utilizadas para facilitar e permitir a integração de todos estes componentes, garantindo uma comunicação funcional, permitindo o funcionamento da central por completo.
- Por fim, no Capítulo 5 são apresentadas as considerações finais do trabalho, incluindo o resultado desta solução proposta no trabalho.

2 ALTERNATIVAS DE COMUNICAÇÕES

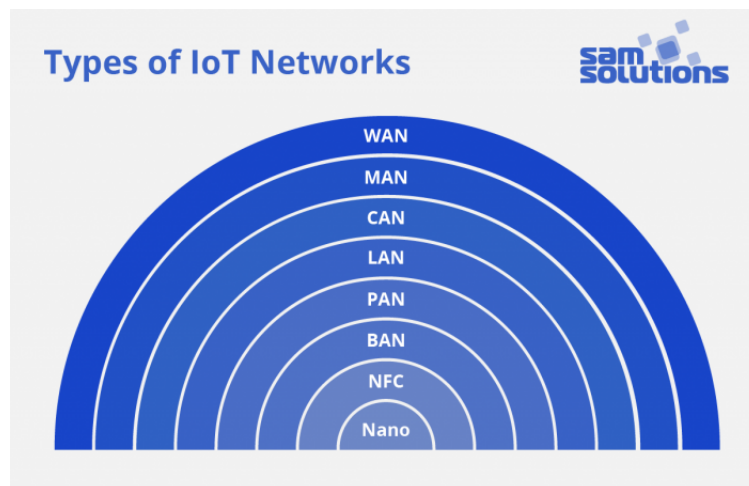
Para poder realizar uma integração completa entre o servidor *web*, o *smartphone* e os componentes utilizados na automação da residência, deve-se considerar os padrões de comunicação de cada um, podendo utilizar *gateways* quando houver padrões heterogêneos, para possibilitar a comunicação com a central com robustez e garantir o funcionamento e desempenho desejado.

Para considerar os padrões de comunicação da central de automação, deve-se analisar o tamanho geográfico da rede de sensores e dispositivos que serão utilizados, pois dependendo da distância da área efetiva, deve-se considerar o tipo de rede e protocolos que serão utilizados para garantir que não ocorra uma falha de comunicação e perda de dados entre os sensores, ou um rendimento ruim, resultando uma insatisfação com os objetivos definidos para a central.

2.1 CLASSIFICAÇÃO DE REDES PARA IOT POR DISTRIBUIÇÃO GEOGRÁFICA

Existem várias redes de conexão para IoT, e o projetista deve considerar a rede que melhor contempla os requisitos de sua central de automação. De acordo com a área de atuação da rede de comunicação e os dispositivos que serão utilizados, é possível identificar qual rede será utilizada. As classificações de redes são definidas de acordo com seu alcance, a Figura 1 demonstra a classificação das redes desde a rede Nano, onde a distância é de milímetros até a rede WAN, que pode encobrir o planeta Terra inteiro.

Figura 1: Tipos de redes classificadas por distribuição geográfica



Fonte: SAKOVICH (2018)

2.1.1 Nanonetwork

A *Nanonetwork* é uma rede em nanoescala entre nanodispositivos, que são dispositivos com características macroscópicas dos materiais que são compostos. Estes dispositivos sofrem de limitações nas capacidades de processamento e gerenciamento de energia, pois os mesmos realizam tarefas simples mas que necessitam de diferentes abordagens de acordo com o objetivo do projetista ao utilizá-los (GALAL; HESSELBACH, 2018). A distância desta rede entre os dispositivos é de alguns milímetros.

A *Nanonetwork* permite expandir as capacidades dos nanodispositivos em termos de complexidade e área de operação, permitindo que os diferentes dispositivos possam coordenar e dividir informações entre eles, o que não seria possível em uma rede comum, devido as dificuldades que estes dispositivos enfrentam (AKYILDIZ, 2010).

2.1.2 NFC (*Near-Field Communication*)

A rede NFC permite que dois dispositivos consigam estabelecer comunicações em uma curta distância de alguns centímetros. Atualmente esta comunicação é bastante utilizada em *smartphones* para realizar pagamentos, através da aproximação deste para um leitor que possui esta tecnologia, ou RFIDs para liberação de acessos em locais físicos que utilizam esta tecnologia. A Distância entre os dispositivos normalmente é de 10 centímetros.

Estes dois dispositivos podem operar em diversos modos, que são distinguidos baseado em qual possui uma fonte de energia própria, sendo considerado como dispositivo ativo, caso contrário será considerado para a aplicação como um dispositivo passivo, e irá depender da aplicação realizar esta autenticação (FINKENZELLER, 2010). A Figura 2 demonstra algumas possibilidades de uso de uma rede NFC.

2.1.3 BAN (*Body Area Network*)

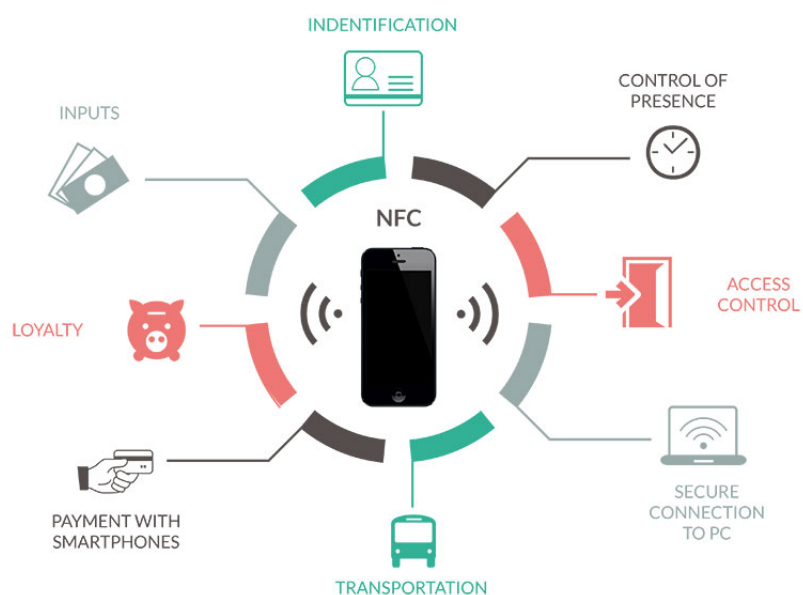
A rede BAN permite conectar dispositivos *wearables* no corpo de uma pessoa, podendo ser "vestida" ou inserida no corpo, através de implantes (NEGRA; JEMILI; BELGHITH, 2016).

Esta rede possui grande aplicação na área da medicina, e os dispositivos possuem o objetivo de medir e monitorar alterações nos sinais vitais do paciente, e também detectar emoções. A Figura 3 demonstra alguns possíveis componentes utilizados em uma rede BAN.

2.1.4 PAN (*Personal Area Network*)

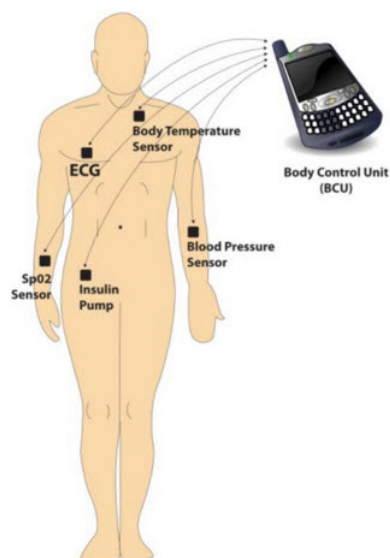
A rede PAN permite que dispositivos eletrônicos perto e junto ao corpo humano possam se comunicar e trocar informações. Uma Rede PAN pode ser cabeada ou sem fio,

Figura 2: Possibilidades de utilização da rede NFC



Fonte: ASIARFID (2018)

Figura 3: Exemplo de componentes da rede BAN



Fonte: ELPROCUS (2018)

a sua área de atuação pode ser de alguns centímetros até aproximadamente 10 metros (MITCHELL, 2019).

Podemos citar como um exemplo de rede PAN um escritório onde se utiliza um *notebook* conectado a um *mouse* e teclado sem fio, realizando uma impressão em uma impressora sem fio. Todos estes equipamentos juntos formam uma rede PAN. A Figura 4 demonstra um exemplo de uma rede PAN.

Figura 4: Exemplo de uma rede PAN



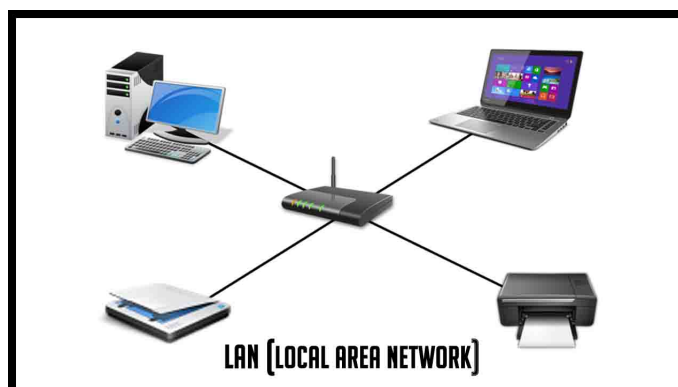
Fonte: KASHYAP (2019)

2.1.5 LAN (*Local Area Network*)

A rede LAN é utilizada para conectar computadores ou dispositivos dentro de um local, podendo ser um escritório, uma casa com vários dispositivos, uma escola, entre outros. As tecnologias utilizadas atualmente podem ser a cabeada, através da *Ethernet*, ou sem fio, através da *Wi-Fi*. A distância de uma rede LAN pode chegar a 1 quilômetro.

A rede LAN funciona como qualquer outra rede de comunicação, possuindo três *hardwares* básicos: um meio de transmissão, um mecanismo para controle de transmissão e uma *interface* para controlar a rede. Em adição a estes elementos, a LAN possui um conjunto de protocolos de software, que controlam a transmissão de informação via *hardware* da rede (CLARK; POGAN; REED, 1978). A Figura 5 demonstra um exemplo de uma rede LAN.

Figura 5: Exemplo de uma rede LAN



Fonte: GUPTA (2018)

2.1.6 CAN (*Campus/Corporate Area Network*)

A rede CAN é uma rede autônoma, normalmente de propriedade de uma universidade ou de uma empresa que possui vários prédios localizados próximos um a outro. É uma rede LAN ou um conjunto de redes LAN a serviço de uma corporação, agência governamental, universidade ou alguma organização parecida (ALI, 2015). A Figura 6 demonstra um exemplo de uma rede CAN.

Figura 6: Exemplo de uma rede CAN



Fonte: REHMAN (2019)

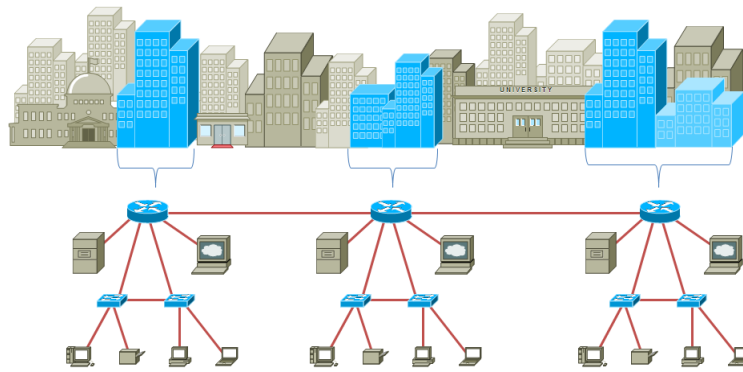
2.1.7 MAN (*Metropolitan Area Network*)

A rede MAN é utilizada para fornecer conexão de alta velocidade para cidades e vilarejos, sendo feita através de múltiplas redes LAN comunicando entre si. Esta rede depende de uma interligação entre equipamentos pertencentes a organizações diferentes. Por este motivo é necessário que exista uma padronização entre os serviços e os equipamentos das companhias e do consumidor final (MOLLENAUER, 1988). A Figura 7 demonstra um exemplo de uma rede MAN.

2.1.8 WAN (*Wide Area Network*)

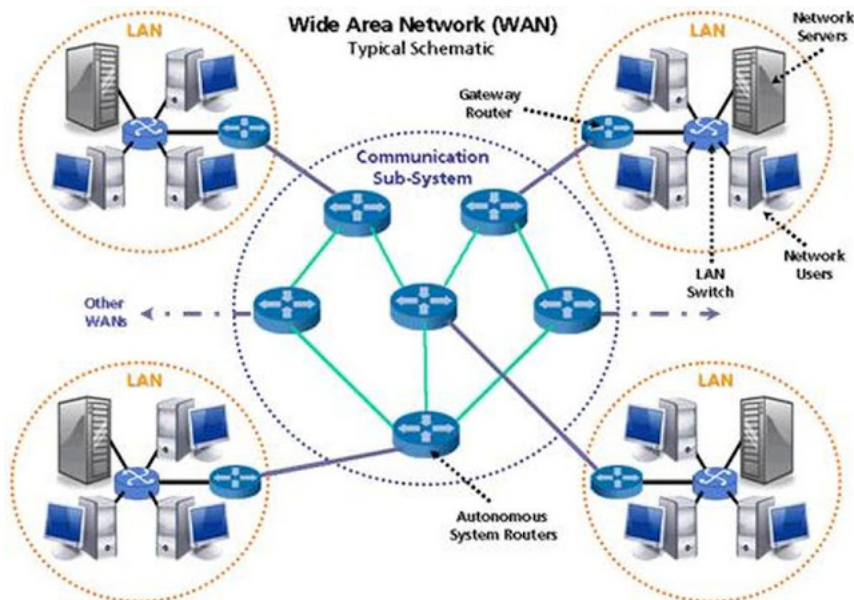
A rede WAN é a de maior alcance, pois realiza a conexão entre outro tipos de redes ao redor do mundo, devendo a mesma suportar qualquer tipo de tecnologia, já que os *hardwares* de uma ponta podem ser diferentes dos *hardwares* da outra ponta. A Figura 8 demonstra uma topologia de uma rede WAN.

Figura 7: Exemplo de uma rede MAN



Fonte: DANVIBOON (2013)

Figura 8: Topologia de uma rede WAN



Fonte: ROUTE-XP (2017)

2.2 REDES SEM FIO X REDES COM FIO

Para a implementação de uma central de automação, é importante decidir se a rede que será utilizada para comunicar a central com os sensores e outros dispositivos irá ser cabeada ou sem fio.

As redes *wired*, ou cabeadas, possuem uma maior segurança dos dados e permitem uma entrega de conexão mais estável. Também possuem uma maior confiabilidade, visto que não ocorrem interferências ou perda de sinais devido a distância. Mas também a instalação de uma rede cabeada é mais cara que uma sem fio, além de ser mais difícil para um usuário comum realizá-la, sendo necessário contratar um profissional. No trabalho do

SCHNEPS-SCHNEPPE et al. (2012), pelo fato de os autores priorizarem a segurança, foi escolhida uma rede cabeada ao invés de uma rede sem fio no experimento. Podemos citar como exemplo de padrão para redes cabeadas a RS-485, que estabiliza as conexões e permite multipontos. E o Modbus como um exemplo de protocolo de comunicações serial.

As redes *wireless*, ou sem fio, estão em alta cada vez mais, devido ao baixo custo dos sensores e dispositivos que utilizam tecnologias sem fio, e a fabricação em massa dos mesmos. Outro ponto positivo das redes sem fio é a questão da instalação, que é facilitada para o usuário comum, evitando a necessidade do mesmo realizar o cabeamento na residência. Além de garantir o mesmo a liberdade de poder realizar a conexão em qualquer local, sem se preocupar com o alcance dos cabos. Como ponto negativo, existe uma insegurança maior dos dados comparada a rede cabeada, e uma vulnerabilidade maior para invasores de distância remota. A rede sem fio está substituindo a cabeada, pois a rede *wired* possui uma instalação mais difícil, além de uma rede cabeada necessitar planejamento para construção (PARK; STHAPIT; PYUN, 2009).

2.3 PROTOCOLOS DE CONEXÃO SEM FIO PARA IOT

Os protocolos de conexão sem fio são muito utilizados atualmente para a automação residencial, são eles que permitem aos sensores e os dispositivos se conectarem com a central de automação, e através destes é possibilitado que o *software* controle o *hardware*, e que este envie informações para a central para serem utilizados ou salvos.

A escolha do protocolo irá depender de aplicações práticas, entre outros fatores como o custo do *chipset*, a confiabilidade da rede, o custo da instalação, etc (LEE et al., 2007).

2.3.1 WI-FI

O protocolo WiFi representa a família de dispositivos que utilizam o protocolo IEEE 802.11, comumente utilizado para a realização da rede WLAN (*Wireless Local Area Network*), a rede local sem fio. Atualmente a maioria dos roteadores utilizam este protocolo, pela facilidade de permitir diversos dispositivos se conectarem simultaneamente, em diversas áreas, como as industriais, comerciais e residenciais.

Os utilizadores desta rede esperam os mesmos serviços e capacidades que a rede cabeada fornece, para isto é necessário analisar fatores que na rede cabeada não é necessário, como a alocação de frequência, a segurança humana, a mobilidade, o rendimento, a segurança dos dados, a interferência e o consumo de energia (CROW et al., 1997).

Pela grande utilização do Wi-Fi, para os projetos de automação residencial este protocolo é bastante utilizado, tanto para a comunicação com os sensores como para

permitir que os computadores da residência possam ser utilizados como *web server*, permitindo o controle remoto destes sensores e de toda a central.

2.3.2 BLUETOOTH

O *Bluetooth* é um protocolo de rede sem fio de curta distância para comunicação direta entre dispositivos fixos e móveis. É operado na arquitetura mestre-escravo, mas com a diferença com relação a outros protocolos que qualquer dispositivo consegue se identificar como mestre. O mestre é quem define as frequências onde se realizarão as trocas de informações e a conexão. Geralmente o dispositivo que inicia a conexão é o mestre deste *link* de dispositivos.

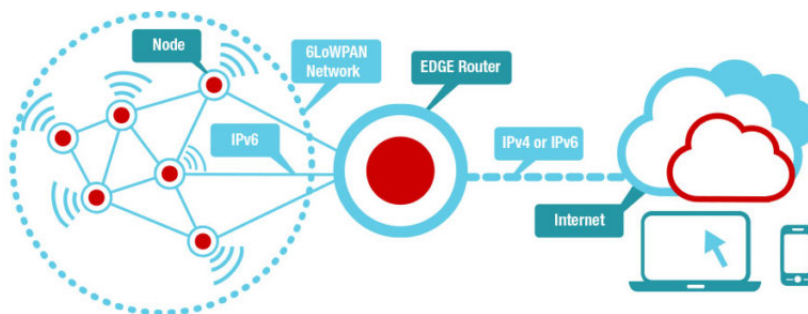
O modelo do *Bluetooth* é uma comunicação *peer-to-peer* baseado na proximidade dos dispositivos, quando o dispositivo mestre detecta um dispositivo escravo pela primeira vez, ele associa um identificador a este e realiza o pareamento, após o qual, a cada vez que o dispositivo escravo for detectado dentro da área de operação a conexão entre eles será realizada automaticamente (MILLER; BISDIKIAN, 2001).

2.3.3 6LOWPAN

A denominação 6LoWPAN é uma sigla para *IPv6 over Low-Power Wireless Personal Area Networks*, ou seja, é um protocolo de comunicação que permite utilizar um protocolo de *internet* sobre dispositivos pequenos de pequena taxa de transferência de dados, utilizados em redes PAN, mais especificamente no padrão IEEE 802.15.4. Pelo fato de se utilizar de protocolos de *internet*, a mesma pode ser utilizada para se conectar com outros tipos de dispositivos através de um *link* de rede de internet como a Wi-fi.

O protocolo 6LoWPAN utiliza conceitos do *IPv6* para criar um conjunto de cabeçalhos que permitem uma codificação eficiente para comprimir grandes endereços e cabeçalhos do *IPv6* em um único cabeçalho que pode ter o tamanho de até 4 *bytes* (MULLIGAN, 2007). A Figura 9 demonstra uma integração do protocolo com a *internet*.

Figura 9: Integração de uma rede 6LoWPAN com a *internet*

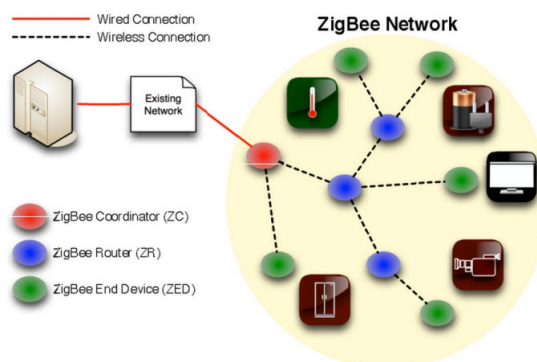


Fonte: ZOLERTIA (2019)

2.3.4 ZIGBEE

O protocolo ZigBee, assim como o protocolo 6LoWPAN, é baseado no padrão IEEE 802.15.4 e tem o seu maior uso em redes PAN, podendo utilizar uma malha (*mesh*) de dispositivos *ZigBee*. Mas entre as diferenças para os outros protocolos está o modo *sleep* dos dispositivos que utilizam o protocolo ZigBee, fazendo com que os dispositivos não precisem estar ativos a todo o momento, aumentando a vida útil da bateria dos mesmos. Mas não são facilmente conectados entre outros protocolos, como o 6LoWPAN. A Figura 10 demonstra um exemplo de uma arquitetura utilizando este protocolo.

Figura 10: Exemplo de uma arquitetura utilizando ZigBee



Fonte: ELPROCUS (2014)

2.3.5 THREAD

O protocolo *Thread* utiliza um outro protocolo, o 6LoWPAN, portanto também utiliza o padrão IEEE 802.15.4 para os dispositivos. O protocolo *Thread* também possibilita a utilização de redes *mesh*, ou redes em malha, uma topologia na qual os nós estão conectados entre si sem possuir hierarquia, permitindo uma maior confiabilidade contra a parada da operação de rede ao ocorrer falhas em um dos nós.

Este protocolo consegue suprir a falta de padrões dos dispositivos, por ser designado a suportar diversos dispositivos para controle residencial e por utilizar o 6LoWPAN, permitindo que o *Thread* seja um protocolo aberto e normalizado (AL-QASEEMI et al., 2016).

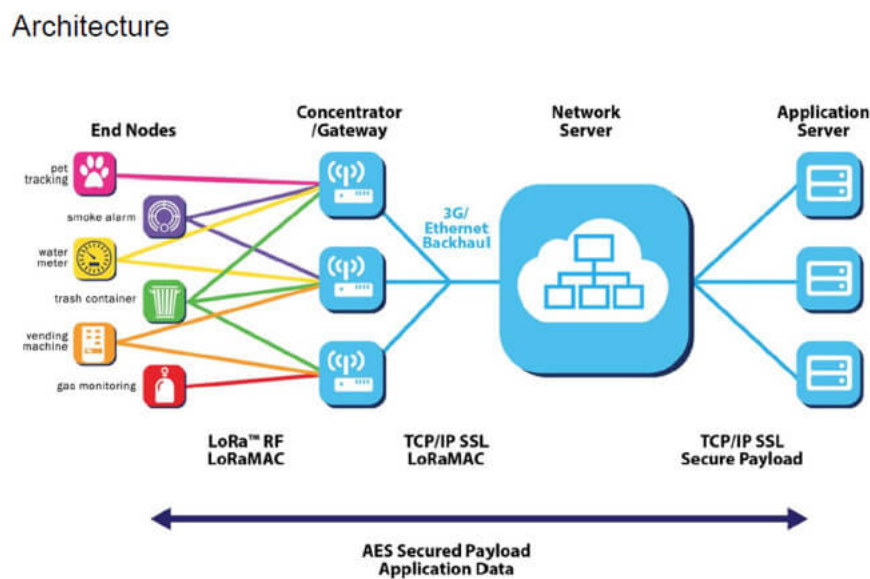
Outra vantagem de utilizar *Thread* é a de que ele utiliza o protocolo UDP para mensagens entre os dispositivos. No contexto de IoT, o UDP é preferível ao TCP pelo custo menor e pela velocidade maior de transmissão.

2.3.6 LORAWAN

O protocolo LoRaWAN é baseado na arquitetura LoRa (*Long Range*). Esta arquitetura é baseada na tecnologia LPWAN (*Low Power Wide Area Network*), que permite conexões de longo alcance, com o seu maior alcance em 45 quilômetros na zona rural, por possuir pouca ou nenhuma interferência para a rede. A definição Low Power é utilizada devido aos dispositivos nesse protocolo possuírem pequena taxa de transferência de dados.

O protocolo LoRaWAN utiliza a topologia *star of stars*, ou seja, os dispositivos e sensores realizam a comunicação para os *Gateways*, que retransmitem as mensagens para o servidor central, geralmente utilizando *Ethernet*, tecnologias móveis como 3G/4G, via satélite ou Wi-Fi. Então o servidor central decodifica os pacotes, realizando as análises necessárias e gerando os pacotes que devem ser enviados aos dispositivos (ADELANTADO et al., 2017). A Figura 11 demonstra um exemplo de uma arquitetura LoRaWAN.

Figura 11: Exemplo de uma arquitetura LoRaWAN



Fonte: FARNELL (2017)

2.4 PROTOCOLOS DE COMUNICAÇÃO PARA IOT

Os protocolos de comunicação, ou protocolos mensageiros são protocolos utilizados na camada de aplicação do modelo OSI (*Open Systems Interconnection*), ou seja, são protocolos que permitem que os dados coletados pelos dispositivos e sensores consigam ser interpretados no nível da aplicação. Para a IoT é fundamental que o *software* consiga interpretar estes dados para o controle da central e registrar as informações coletadas.

Para a escolha do protocolo de comunicação, é de responsabilidade do projetista ou organização responsável analisar o seu projeto de IoT e decidir qual protocolo atenderá melhor às necessidades do projeto.

A IoT não pode depender de apenas um único protocolo que satisfaça todas as necessidades, pois cada protocolo foi construído com um objetivo em mente, sendo necessário investigar os prós e contras de cada protocolo para encontrar os cenários que se encaixam melhor em cada um destes (NAIK, 2017).

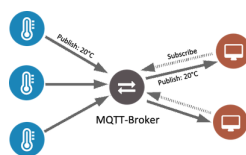
2.4.1 MQTT

O protocolo MQTT (*Message Queuing Telemetry Transport*) é um protocolo de comunicação M2M (*Machine to Machine*), máquina a máquina, ou seja, uma conexão direta entre dois dispositivos. Este protocolo é baseado no método publicação e assinatura.

O MQTT consiste em duas entidades, um servidor de mensagens, denominado "*broker*", que possui tópicos de mensagens e clientes, que podem ser qualquer dispositivo da rede IoT, sendo que o cliente pode ser um publicador ou um assinante. No instante que um cliente publica algo no servidor *broker*, o mesmo redireciona esta mensagem para todos os assinantes do tópico da mensagem enviada.

A maior parte da lógica do protocolo é gerenciada pelo *broker*, resultando em uma implementação extremamente leve por parte do usuário, onde uma implementação completa de um protocolo MQTT-S, uma versão estendida do MQTT, utiliza, em média 12 kB de memória (HUNKELER; TRUONG; STANFORD-CLARK, 2008). A Figura 12 demonstra um exemplo do funcionamento do MQTT.

Figura 12: Exemplo do funcionamento do MQTT



Fonte: FOUNDATION (2014)

2.4.2 HTTP

O protocolo HTTP (*Hypertext Transfer Protocol*) é um protocolo inicialmente projetado para manipulação de mensagens *web* apenas, mas que pode ser incorporado para as redes IoT. Por utilizar o modelo de requisição e resposta, onde um cliente realiza uma requisição contendo um método que é pré-definido no protocolo, a URI (*Uniform Resource Identifier*) e possivelmente um corpo de mensagem para um servidor, que retorna com o

status de sucesso ou erro da resposta, informações do servidor e possivelmente a resposta desejada no corpo da mensagem.

O HTTP, como o nome sugere, é baseado em mensagens de texto e não define o tamanho máximo de sua mensagem ou de seus cabeçalhos, estas características são definidas pelo servidor *web* para o qual serão realizadas as requisições dos clientes.

O protocolo HTTP é genérico e pode ser utilizado para realizar muitas tarefas além de apenas seu uso para o hipertexto através da extensão de seus métodos de requisição, códigos de erros e cabeçalhos (FIELDING et al., 1999).

2.4.3 CoAP

O protocolo CoAP (*Constrained Application Protocol*), também é um protocolo para comunicação M2M, mas diferentemente do MQTT, este protocolo utiliza o modelo de requisição e resposta, de maneira similar ao realizado em comunicações HTTP, mas com a diferença de ser utilizada sobre o protocolo UDP, resultando em uma transmissão mais rápida e leve do que o TCP/IP.

Da mesma forma que o protocolo HTTP realiza, na requisição do CoAP é enviado uma ação, através de um código pré-definido para um URI referente ao tópico utilizado, o qual envia a resposta juntamente com outro código pré-definido para que possa ser interpretado. Diferentemente do HTTP, pelo fato de o CoAP se utilizar do protocolo UDP, as interações entre os dispositivos são realizadas de forma assíncrona.

O objetivo do CoAP não é o de comprimir o HTTP para os dispositivos, mas sim realizar um subconjunto de REST em comum com o HTTP que seja otimizado para aplicações M2M. Apesar do CoAP poder ser utilizado como uma *interface* simplificada do HTTP, o mais importante são as funcionalidades para a comunicação M2M (SHELBY; HARTKE; BORMANN, 2014).

2.4.4 AMQP

O protocolo AMQP (*Advanced Message Queuing Protocol*) é um protocolo aberto padrão para comunicação MOM (*Message-oriented middleware*), a qual suporta envio e recebimento de mensagens em sistemas distribuídos.

O protocolo AMQP possui várias funcionalidades que são relacionadas a mensagens, como enfileiramento confiável de mensagens, método de publicação e assinatura baseado em tópicos, roteamento flexível e transações (NAIK, 2017).

2.5 API RESTFUL

Para a realização dos comandos enviados pelo usuário, a solução deste projeto irá utilizar uma API (*Application Programming Interface*) desenvolvida pelo próprio autor.

A API define um conjunto de rotinas e padrões que são utilizados para o acesso a uma aplicação *Web*. A terminologia REST (*Representational State Transfer*) representa um estilo de arquitetura de software, foi criada por Roy Thomas Fielding, um dos principais autores do protocolo HTTP. Junto com a criação do REST, foram definidos alguns princípios que serão explicados posteriormente. Quando uma API utiliza o estilo de arquitetura REST, a mesma é denominada API RESTful (PAUTASSO; WILDE; ALARCON, 2013).

A arquitetura REST, como mencionado anteriormente, possui alguns princípios a serem seguidas. A primeira delas sendo a separação entre cliente e servidor, definindo que cada uma destas entidades possuem diferentes funções na aplicação. Isto permite que as duas possam ser implementadas de formas diferentes, seja utilizando linguagens de programação ou padrões internos diferentes, desde que seja utilizado os padrões definidos pela aplicação no momento da comunicação do cliente e o servidor. Ao realizar esta separação, a aplicação permite a sua compatibilidade e portabilidade para as mais diversas interfaces, além de permitir a sua alta escalabilidade, esta ultima sendo considerada a mais importante, pois permite que possa ser implementado novos recursos apenas por parte do servidor sem afetar o funcionamento prévio.

O próximo princípio se refere ao modelo de comunicação *Stateless*, neste modelo, o cliente possui por inteiro a responsabilidade de enviar todas as informações necessárias para que o servidor consiga interpretar os dados e executar os comandos que estão sendo requeridos. Com isto é removido a necessidade do servidor armazenar dados do cliente na sessão, com esta particularidade, permite a escalabilidade da aplicação pois a responsabilidade não está contida no servidor, aumentando o número de clientes que o servidor possa atender simultâneamente.

Outro princípio da arquitetura REST é a cacheabilidade dos dados, que permite a definição de quais dados de resposta podem ser armazenados em cache pelo servidor. O cache de dados de resposta pode ajudar a reduzir o tempo de processamento da resposta, aumentar a disponibilidade e a confiabilidade de um aplicativo, além de controlar a carga de um servidor da web. Resumindo, o cache reduz o custo total da *Web*(MASSE, 2011).

O princípio de sistema em camadas permite que *gateways* ou um outro servidor intermediário intercepte a requisição enviada pelo cliente, geralmente isto é aplicado para serviços específicos que o projetista detectou a necessidade dessa interceptação para um melhor rendimento. Esta diretriz também é muito aplicada para o balanceamento da carga, onde um servidor serve como um balanço para distribuir as requisições entre os servidores a fim de evitar uma sobrecarga e garantir um aproveitamento melhor deles. Nesta situação

o cliente não consegue identificar que a sua chamada foi interceptada.

Um princípio opcional é o *Code-in-Demand*, que pode ser traduzido como código sob demanda, este princípio permite que os servidores *Web* possam executar códigos na interface do cliente, estendendo ou customizando a experiência do usuário. Por necessitar que a interface do usuário esteja preparada para entender e executar o código enviado, este princípio é opcional. As tecnologias hospedadas pelo navegador da *Web*, como *applets* Java, JavaScript e Flash, exemplificam este princípio (MASSE, 2011).

Por fim, o último princípio se refere a interface uniforme, ou seja, os componentes do sistema, como os servidores *Web*, os servidores intermediários ou *gateways* e o cliente dependem da uniformidade de suas interfaces para que a comunicação funcione corretamente, caso contrário a comunicação é quebrada. Dentro deste princípio, Fielding definiu quatro subprincípios. O subprincípio da identificação de recursos define que cada componente *Web* seja identificado de forma global, por exemplo utilizando uma URI. O subprincípio da manipulação de recursos através de representações define que deve ser possível manipular os recursos informando o tipo de manipulação ao identificar o recurso desejado, é possível realizar várias manipulações do recurso, como buscar informações, atualizar, criar, deletar, etc. Por exemplo um PUT de atualização ao informar a URI do recurso em um protocolo HTTP. Outro subprincípio define que deve-se possuir mensagens auto-descritivas, onde ao realizar a troca de requisições entre o servidor e o cliente, deve-se possuir dados sobre o recurso e metadados que são informações sobre a representação do recurso. E o último subprincípio é referente a HATEOAS (*Hypermedia As The Engine Of Application State*), este subprincípio informa que os recursos devem ser entregues junto com os *links* relacionados a estes recursos, a presença ou ausência destes *links* facilitam o entendimento do estado atual do recurso requisitado (PAUTASSO; WILDE; ALARCON, 2013).

2.6 PHP

Uma das principais necessidades ao se desenvolver qualquer *software* é definir a linguagem de programação a ser utilizada, neste projeto foi utilizado a linguagem de código fonte aberto PHP (*Hypertext Preprocessor*).

O PHP é uma linguagem altamente utilizada nos dias atuais, apesar de não ser a linguagem de programação mais moderna para a programação focada no lado do servidor. Esta linguagem é multiplataforma, o que facilita no desenvolvimento em computadores de diferentes sistemas operacionais.

Uma das principais vantagens de se utilizar esta linguagem é o seu suporte a uma ampla variedade de banco de dados. simplificando a utilização de extensões específicas de banco de dados (PHP, 2019). Como por exemplo o MySQL, que foi utilizado neste

trabalho.

2.7 BIBLIOTECAS, FRAMEWORKS E FERRAMENTAS

Ao desenvolver um *software*, é possível contar com algumas bibliotecas e *frameworks* que facilitarão neste processo, agilizando a sua criação e inserindo novas funcionalidades de acordo com o que o desenvolvedor necessitar. Para este trabalho serão utilizados algumas bibliotecas e alguns *frameworks* em diferentes componentes da solução. Para a realização de testes com a finalidade de garantir o funcionamento da solução, foram utilizadas algumas ferramentas durante o processo de desenvolvimento.

2.7.1 Flutter

Entre alguns diversos *frameworks* para desenvolvimento de aplicativos para *smartphones*, podemos citar o *Flutter*, desenvolvido originalmente pela empresa Google.

O *Flutter* permite um rápido desenvolvimento a partir da funcionalidade *hot reload*, onde ao realizar qualquer alteração no código, o aplicativo será atualizado instantaneamente no emulador ou no celular conectado no modo *debug*. E diferentemente de outros *frameworks*, o *Flutter* utiliza a estrutura de *widgets*, onde cada *widget* é um elemento imutável da tela e pode representar diversos tipos de elementos, como de estilo, estrutural, de *layout*, entre outros (FLUTTER, 2019).

O *Flutter* possui um *plugin* de reconhecimento de voz, o *speech_recognition*, onde este foi utilizado

2.7.2 MQTT Lens

O *MQTT Lens* é um *software* que permite analisar a saúde da conexão com o broker *MQTT*, realizar o cadastro em um tópico específico e visualizar as mensagens enviadas ou realizar o envio de uma mensagem para um tópico, simulando o processo que algum sensor ou atuador realiza quando necessita enviar uma informação para a central, ou o processo da central ao enviar um comando para estes sensores. Este *software* é uma extensão do navegador *Google Chrome*, o que facilita bastante a sua instalação, além de seu uso ser bastante simples e intuitivo.

2.7.3 Laravel

Neste trabalho, foi decidido a utilização de uma *framework* para o processo de desenvolvimento no lado do servidor, então foi necessário escolher uma *framework* que utilize o *PHP* e facilite e agilize o desenvolvimento com esta linguagem.

O *Laravel* permite que a programação em *PHP* seja agilizada, devido a sua sintaxe simples e baixa curva de aprendizado. Este *framework* facilita nas partes mais complexas

da aplicação, como na realização de autenticação de usuários, a realização de roteamento que a API ou a página web deve realizar, o tratamento de erros automáticos, entre outros (GARBAR, 2019).

2.7.4 Arduino IDE

Como é necessário realizar a programação diretamente nas placas, foi decidido utilizar a IDE (Ambiente Integral de Desenvolvimento) do Arduino. As placas que utilizam o chip ESP8266 podem ser programadas em diferentes linguagens, como Lua e MicroPython, mas uma das soluções mais utilizadas é utilizar a IDE do Arduino, por ser uma programação mais simples e bem parecida a linguagem C, esta opção foi a escolhida para este trabalho.

Como o chip ESP8266 não é suportado nativamente nesta IDE, para realizar a programação é necessária a instalação de um pacote específico que permite o suporte a este chip. Também dependendo da placa de desenvolvimento que contém este chip, para as placas que não possuem o conversor serial é necessário de um outro dispositivo que converta este serial para FTDI. Sendo assim o código diretamente inserido na memória flash do chip.

3 ARQUITETURA DO SISTEMA

Para a realização de uma central de automação residencial, o projetista deve considerar alguns pontos importantes antes da implementação da mesma, como definir os objetos ou produtos eletrônicos da residência que serão controlados, assim como os microcontroladores e dispositivos para a realização deste controle. Estes serão definidos através de uma análise de requisitos pelo escopo da arquitetura física a ser utilizada no trabalho. A arquitetura do projeto é mostrada na Figura 13.

Além dos pontos mencionados acima, foi necessário realizar uma definição de como realizar a comunicação da central com os dispositivos, através da escolha da rede de comunicação de acordo com a área de operação desejada. Foram definidos os protocolos de conexão que serão utilizados, preferencialmente sem fio para poder manter uma liberdade de movimentação e permitir que novos controladores sejam inseridos sem dificuldades. No caso de ocorrer uma heterogeneização entre os protocolos de comunicação, podem ser analisadas possibilidades de gateways para poder manter um padrão com a central.

Na parte de software, foram analisados os protocolos de comunicação, ou protocolos de aplicação, para poder realizar os registros dos dados, analisar os dados recebidos e enviar comandos para os controladores e dispositivos. Também foi desenvolvido uma aplicação para smartphones com o objetivo de autenticar e permitir o acesso e o controle remotamente, sendo assim necessário a hospedagem de um servidor próprio. Por fim foi realizada a integração do aplicativo com os comandos de voz a fim de facilitar o envio de comandos do usuário.

3.1 REQUISITOS FUNCIONAIS

Para este trabalho, assim como em um projeto de software, foram levantadas as possibilidades de automação de acordo com a arquitetura da residência juntamente com os objetos e produtos eletrônicos que a mesma possui. Destas foram escolhidas as automações realizadas na execução do trabalho.

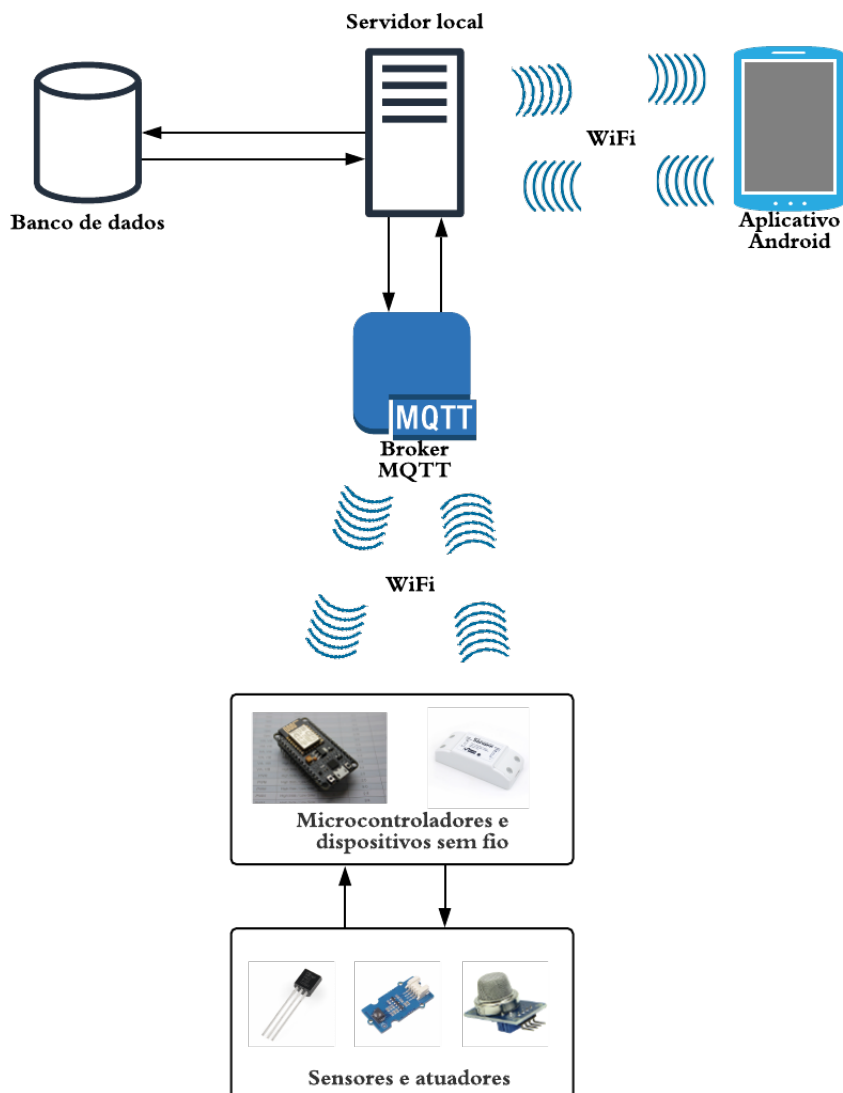
Podemos citar como exemplos de automatização residencial : controlar a iluminação, permitir acesso aos cômodos através das fechaduras eletrônicas, utilizar e controlar câmeras de segurança inteligentes, abrir e fechar as cortinas e persianas das janelas remotamente, controlar a climatização, manipular os televisores, sistema de som e projetores de uma sala de entretenimento, controlar a irrigação e definir os alarmes de segurança.

Estes itens residenciais são apenas uma parcela do que a domótica permite automatizar. Já que, dependendo da arquitetura da residência e do conhecimento técnico do projetista, vários outros podem ser automatizados. Além disto, a domótica pode ser progra-

mável, possibilitando a realização de comandos que facilitem o cotidiano de seus usuários, integrando o controle dos dispositivos de acordo com as funções estabelecidas. Podemos citar como exemplo um desligamento das luzes do quarto, da televisão e o fechamento das persianas quando o usuário desejar dormir, ou diminuir a temperatura e ligar a televisão ou mudar para uma playlist customizada para exercícios da academia no momento que o usuário for praticar exercícios físicos.

Para este trabalho, foram levantados alguns requisitos para a central de automação, que são de suma importância para validar se os objetivos levantados para este projeto foram alcançados.

Figura 13: Estrutura da arquitetura do trabalho



Fonte: o autor (2019)

- *RF1 - Controlar a iluminação*
- *RF2 - Controlar a climatização*
- *RF3 - Realizar um alarme de intrusão*
- *RF4 - Realizar um sistema de notificação de vazamento de gás*
- *RF5 - Controlar e acionar cortinas inteligentes*
- *RF6 - Medir o consumo de energia*

Para o requisito RF1, é permitido para o usuário controlar a iluminação individualmente por cada lâmpada ou agrupada por cômodos, através do aplicativo que foi desenvolvido para smartphones. Também é possível realizar comandos como programar acendimento e apagamento das lâmpadas em horários definidos, para permitir, por exemplo, que o usuário defina uma hora de dormir para controlar o sono, e o mesmo perceba a partir do apagamento das lâmpadas do cômodo que geralmente fica nos horários noturnos.

Para o requisito RF2, é permitido ao usuário controlar a climatização dos ar-condicionados da residência, através do aplicativo, para facilitar o processo de controlar diferentes números de equipamentos.

Para o requisito RF3, foi realizado um alarme de intrusão através de sensores de movimento que serão instalados na residência em pontos estratégicos para a segurança. Na situação em que o sensor detecta um intruso, a central é notificada e guardará o registro do horário em que esta situação ocorreu nos logs do sistema. E ela também notificará o aplicativo para avisar o usuário sobre esta situação de risco.

Para o requisito RF4, foi realizado um sistema de notificação para o usuário sobre vazamentos de gás de cozinha. Para isso, serão instalados sensores de detecção de gás em cima do fogão. Assim que o sensor detectar um nível alto de gás, o usuário é notificado através do aplicativo para que possa verificar esta situação.

Para o requisito RF5, é permitido ao usuário controlar ou acionar as cortinas inteligentes de forma remota através do aplicativo, garantindo que o usuário consiga programar os horários em que as mesmas serão acionadas para a abertura ou fechamento. Com o uso dessa funcionalidade podemos citar a programação do horário para acionar quando for hora de acordar.

Para o requisito RF6, é medido o consumo de energia através de sensores na corrente elétrica da residência. Estes dados são coletados e transmitidos para a central, que guarda os registros no banco de dados integrado a central.

3.2 CONFIGURAÇÕES DO SERVIDOR WEB

Para a realização da comunicação entre o aplicativo desenvolvido, a central e os dispositivos que fazem parte da automação residencial, foi utilizado o protocolo Wi-Fi. Este protocolo foi escolhido por permitir a integração de hardwares que são facilmente encontrados e já são utilizados pela população, como os modems que permitem a conexão sem fio que utilizam o protocolo 802.11.

Para o protocolo de aplicação, foi utilizado o MQTT, este escolhido por ser um protocolo leve e de baixo consumo, além de garantir uma comunicação indireta com o servidor auxiliando no aumento de segurança e utilizar o modelo de publish/subscribe. Pela escolha deste protocolo foi necessário utilizar um servidor para envio e recebimento de mensagens, o broker, como mencionado no capítulo anterior. Também foi necessário a implementação dos tópicos para que os dispositivos consigam receber as mensagens enviadas para os tópicos que estarão inscritos. Estas mensagens serão enviadas através da API RESTful que foi desenvolvida para este trabalho. Para esta API, a comunicação com a central é realizada através do protocolo HTTP, por se tratar de uma API RESTful, a mesma está compatível com o comportamento deste protocolo, para este desenvolvimento foi necessário utilizar boas práticas de desenvolvimento para deixá-la robusta e segura, garantindo uma melhor segurança contra invasores e tornando a sua utilização mais prática para ser mais compatível com o aplicativo para smartphones.

Para guardar os registros de logs dos dispositivos e da aplicação, é necessário armazená-los em um banco de dados. Para este projeto foi escolhido o banco de dados relacional MySQL devido a confiabilidade e a consistência que um banco de dados relacional fornece em comparação a um banco não-relacional. A utilização de um banco de dados relacional permite que seja facilitada o relacionamento entre os sensores e atuadores com os cômodos que estão ou com os grupos que serão utilizados para os comandos personalizados criados no aplicativo. O MySQL também foi utilizado para permitir a vinculação entre os dispositivos e os cômodos, assim como os comandos. além disso este mesmo servidor é utilizado para guardar os comandos de voz convertidos em texto.

3.2.1 API RESTFUL

Para permitir que este modelo de comunicação seja funcional, eficiente e padronizado, foi utilizada uma API RESTful, que garante o controle dos dados para a realização de análises e registros nos logs.

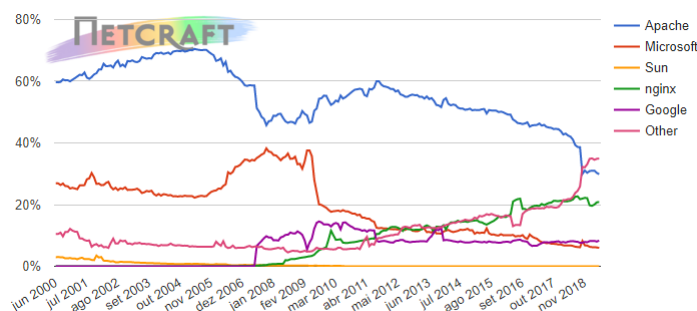
3.2.2 APACHE

Pelo fato de ter sido utilizado o protocolo HTTP neste trabalho, é necessário utilizar um servidor web que seja compatível. Por isto para este trabalho foi utilizado o Apache Web

Server, a escolha deste servidor é devido a alguns motivos. O primeiro motivo é que este servidor é o mais utilizado para aplicações que utilizam este protocolo hoje em dia, como demonstrado na Figura 14. Outro motivo é que este servidor é de código aberto, permitindo que o projetista ou o desenvolvedor do software consiga alterar as configurações para aumentar o desempenho do servidor. O Apache é um servidor Web de alto desempenho e com vários recursos, superior a muitos outros servidores Web baseados em UNIX em termos de funcionalidade, eficiência e velocidade (HU; NANDA; YANG, 1999). A Linguagem de programação utilizada neste servidor web foi o PHP.

Para a segurança desta API, foi utilizada um plugin do framework Laravel, denominada Passport, onde é implementada diretamente no servidor uma autenticação OAuth 2.0, nesta autenticação nenhuma senha ou dado sensível do usuário é enviada ao servidor nas requisições, para realizar a autenticação são utilizados tokens de autenticação identificando o usuário, este token expira ao longo do tempo para garantir uma maior segurança, sendo necessário uma atualização que é feita automática pelo servidor ao verificar que o token está expirado e realmente é o usuário que está realizando a requisição.

Figura 14: Market Share de servidores nos sites ativos



Fonte: NETCRAFT (2019)

3.2.3 BROKER

Além do servidor Apache para o protocolo HTTP, é necessário também a utilização de um servidor que utilize o protocolo MQTT, com o fim de garantir que o modelo publish/subscribe (publicação e assinatura) funcione corretamente para a troca de mensagens entre a central de automação e os dispositivos.

Neste trabalho foi utilizado o Eclipse Mosquitto, um servidor open-source (código aberto) de mensagens que implementa o protocolo MQTT nas versões 5.0, 3.1.1 e 3.0. O Eclipse Mosquitto fornece uma implementação leve do protocolo MQTT que pode ser utilizado em diversas situações, desde uma máquina de potência total até máquinas embutidas e dispositivos de baixa potência, isto é de grande valia pois normalmente os sensores e

dispositivos podem ser pequenos e de baixa potência, incluindo as máquinas embarcadas que podem estar rodando o servidor (FOUNDATION, 2019).

Normalmente, a implementação do Mosquitto possui atualmente um executável de 120kB que consome em média 3MB de memória RAM com 1000 clientes conectados, além de aceitar conexões de aplicações MQTT, o Mosquitto permite se conectar com outros servidores que utilizam este protocolo, incluindo outras instâncias do Mosquitto (FOUNDATION, 2019).

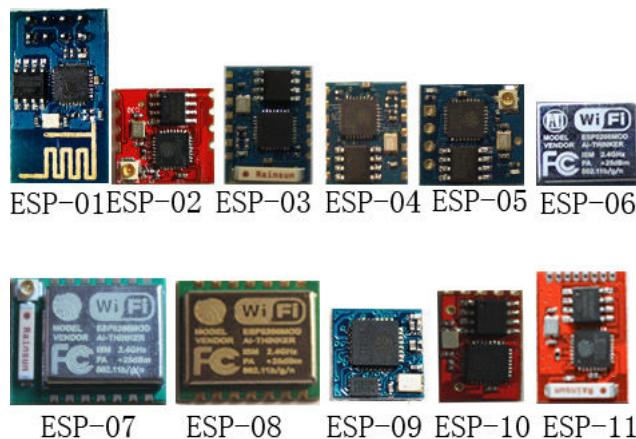
3.2.4 COMUNICAÇÃO COM O SERVIDOR BROKER

Para a comunicação dos dispositivos e atuadores que não possuem a comunicação sem fio já integrada, serão utilizados microcontroladores que possuem essa comunicação para permitir a integração com a central de automação. Neste trabalho, para a comunicação sem fio foi utilizado a família do chip ESP8266, devido ao seu baixo custo e a atuação com o protocolo Wi-Fi.

A família deste chip é bem extensa, e cada um dos módulos que integram o chip são mais apropriadas que outras para certas ações, variando entre o número de portas, a dimensão da placa, o tamanho da memória flash, entre outros. A Figura 15 apresenta alguns dos módulos da família deste chip.

A placa de desenvolvimento NodeMCU utilizada para a realização de algumas funcionalidades desta domótica, esta placa foi escolhida devido a existência de bibliotecas externas já prontas com a finalidade de comunicação com o broker, isto facilitará a parte de programação para a utilização do protocolo MQTT.

Figura 15: Módulos da família do chip ESP8266



Fonte: INSTRUCTABLES (2019)

3.3 APLICATIVO PARA SMARTPHONES

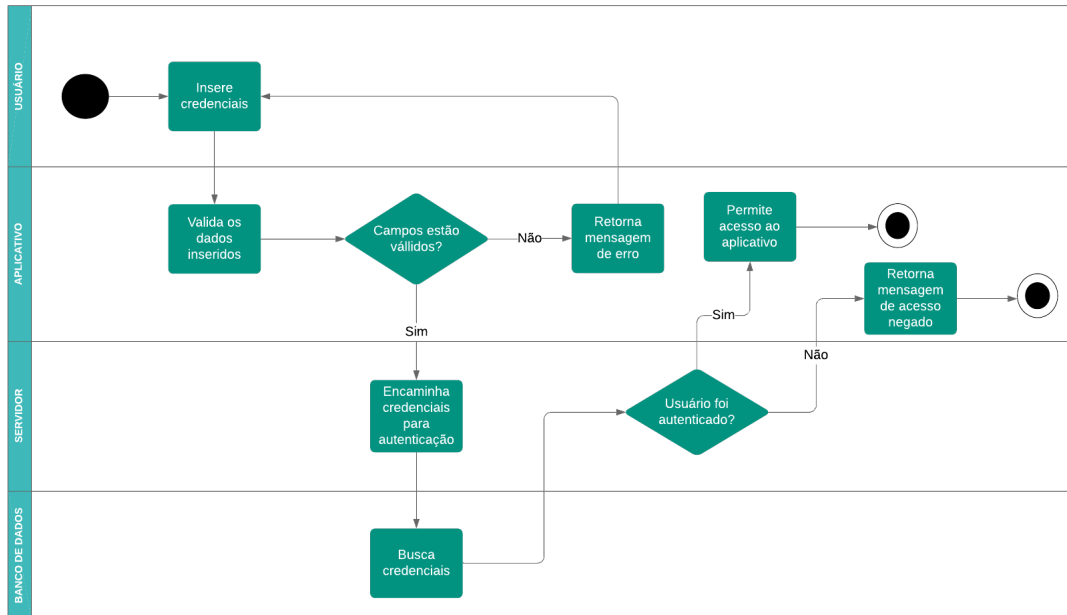
A grande utilização de Smartphones e Tablets nos dias atuais faz-se um grande incentivo para a realização de aplicativos com as mais diversas finalidades, como os aplicativos de relacionamentos, redes sociais, jogos eletrônicos entre outros. Como atualmente o smartphone está muito presente no cotidiano das pessoas, foi decidido a criação de um aplicativo para o controle da central de automação deste trabalho. Outro motivo é a possibilidade para realização de testes a fim de verificar se os objetivos foram alcançados.

Neste trabalho, o aplicativo é utilizado para comunicar-se com a API RESTful, implementada sobre o protocolo HTTP, para o controle da central de automação. É utilizada a API para evitar que ocorra a comunicação diretamente com o servidor, permitindo controlar e limitar o escopo que a aplicação utilizará, aumentando a segurança da informação e evitando algum acesso não permitido do servidor. A segurança para este é de extrema importância, a fim de evitar que alguém com más intenções consiga controlar os dispositivos e sensores. A 16 demonstra, em forma de diagrama de atividades, como é realizada a integração entre os componentes da central para a realização do login, enquanto a 17 demonstra como aparecerá na tela do aplicativo para o usuário.

Outra funcionalidade para o aplicativo é a de permitir gravar os comandos ao vincular com o comando de voz, no momento que o usuário definir o comando através do microfone do smartphone. Este comando de voz é interpretado por uma aplicação externa que converte a voz em texto, sendo gravada no banco de dados. No momento que o usuário realiza este comando através da voz pelo aplicativo, é realizado uma busca do comando no banco de dados e o mesmo é executado.

Para permitir o controle do usuário sobre os dispositivos e sensores, o aplicativo é utilizado para realizar esta conexão, através de requisições para a API. No momento que o usuário realiza o comando, o aplicativo envia para a API as informações necessárias para que o servidor consiga interpretar o que foi enviado e realizar o comando. Após isto, o servidor publica para o broker do MQTT no tópico respectivo ao comando enviado. Isto permite que os dispositivos que estão inscritos no tópico possam interpretar o payload da mensagem para a realização da ação desejada, retornando o status de sucesso ou de falha. Para o processo de criação ou edição de um registro, é necessário que todos os componentes da central consigam se comunicar para realizar esta alteração em todo o sistema, por isso é importante a validação dos dados no aplicativo para que não seja fornecida nenhuma informação que possa ser incompatível com o banco de dados, pois como o banco de dados é relacional, é necessário garantir a compatibilidade entre os tipos de dados. A Figura 18 demonstra em forma de diagrama de atividades como as entidades da central de automação se integram para a realização de uma criação de um registro dentro do aplicativo.

O Quadro 1 demonstra como é a listagem dos dispositivos já criados no aplicativo, a

Figura 16: Integração dos componentes para realização de *login* no aplicativo

Fonte: o autor (2019)

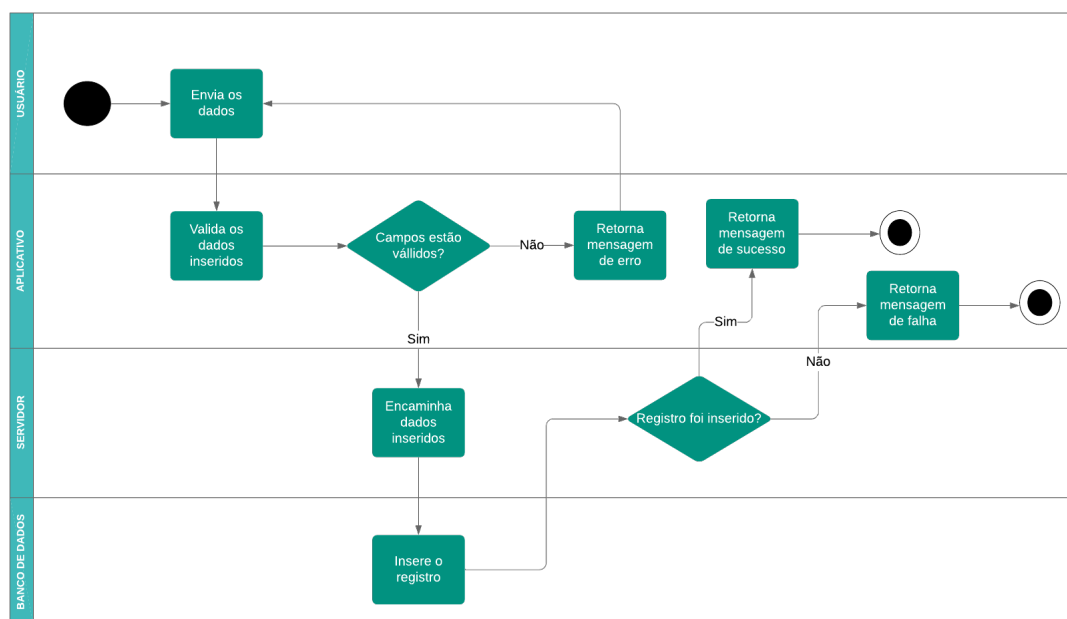
Figura 17: Tela de *login* dentro do aplicativo

Fonte: o autor (2019)

criação de um registro de um dispositivo e a edição deste registro, respectivamente. A partir destes registros, também é possível realizar a vinculação com os grupos e comandos que podem ser cadastrados no aplicativo. Mas também é possível realizar o uso do dispositivo de forma individual, sem esta necessidade da vinculação, é fornecida esta funcionalidade para o usuário possuir uma maior flexibilidade na utilização do aplicativo, a Figura 19 mostra como é apresentada a tela do dispositivo para o usuário no aplicativo. Ao realizar

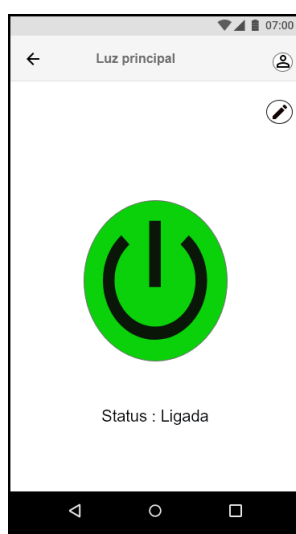
uma ação com o dispositivo a partir do aplicativo, existe a necessidade que toda a central de automação esteja se comunicando corretamente para que esta função seja realizada. A Figura 20 demonstra em forma de diagrama de atividades como é a integração dos componentes da central de automação para a realização da execução de um acendimento de uma luz.

Figura 18: Diagrama de atividades de comando para criação de um registro dentro do aplicativo



Fonte: o autor (2019)

Figura 19: Tela de um dispositivo no aplicativo.



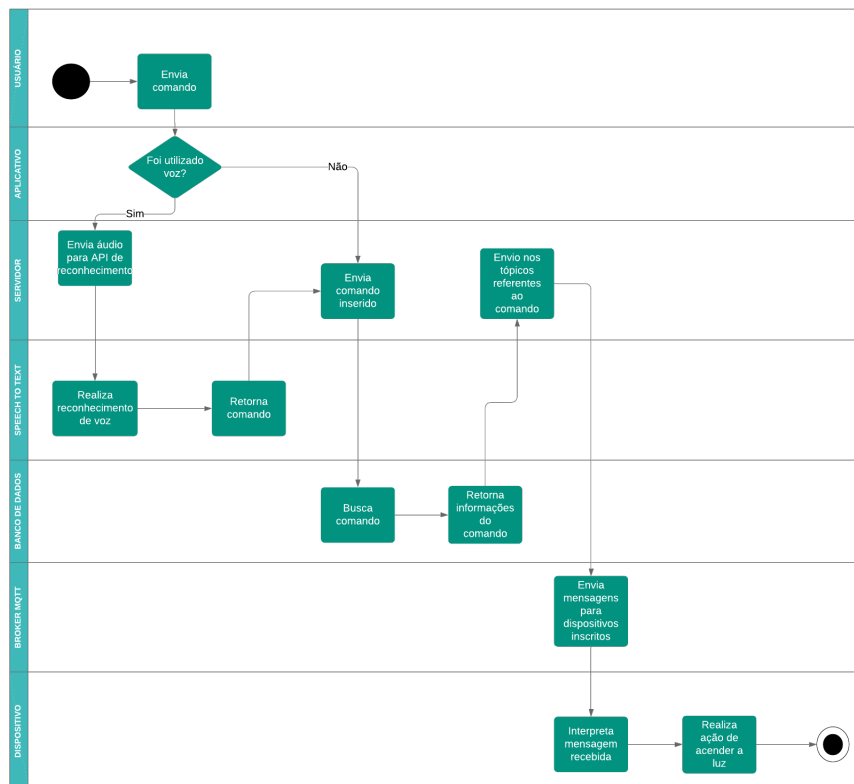
Fonte: o autor (2019)

Para este trabalho, também é permitido a realização de criação de grupos, onde os mesmos podem ser considerados como os cômodos da residência ou um agrupamento de dispositivos de acordo com o tipo do mesmo, como um grupo das luzes da residência ou de todos os ares-condicionados. Estes grupos podem ser utilizados em conjunto com os comandos cadastrados no aplicativo, esta vinculação facilitaria alguns comandos específicos, que utilizam diversos cômodos da casa. O Quadro 2 demonstra como é a listagem dos grupos já criados no aplicativo, a criação de um registro de um grupo e a edição deste registro, respectivamente.

No aplicativo, é possível a criação de comandos que pré-definem os estados que devem estar os dispositivos e sensores e as ações que serão realizadas por estes, isto é salvo no banco de dados e é permitido para o usuário vincular este comando a um comando de voz realizado pelo próprio usuário e convertido para texto. O usuário pode, se preferir, apenas salvar o comando para que seja ativado de forma manual, sem a utilização deste vínculo com o comando de voz. Também é permitido realizar a programação de recorrência da execução ou programar uma data específica para que este comando seja executado.

O quadro 3 demonstra como é a listagem dos comandos já criados no aplicativo, a criação de um registro de um comando e a edição deste registro, respectivamente. Já a

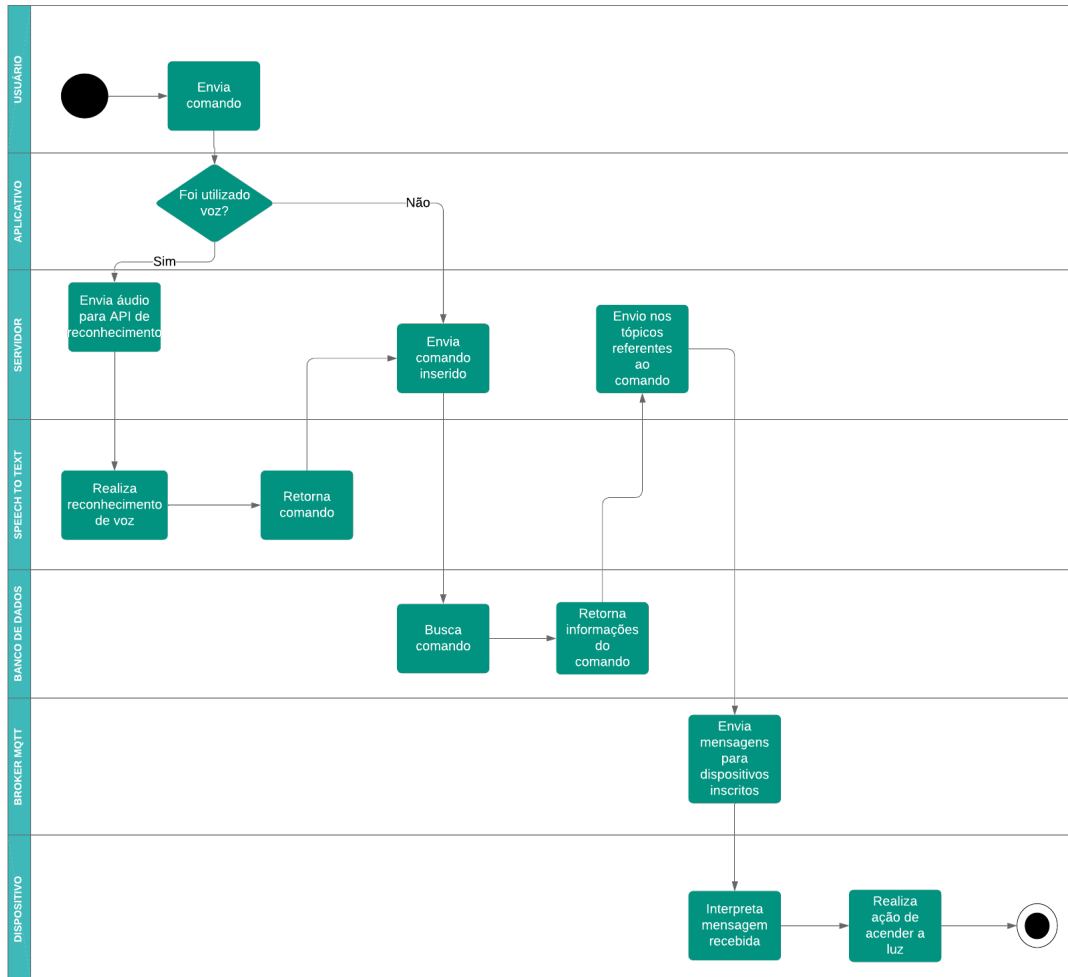
Figura 20: Diagrama de atividades de comando para acender a luz através do aplicativo



Fonte: o autor (2019)

Figura 27 demonstra, em forma de um diagrama de atividades como é a integração dos componentes da central para a execução do comando de automatização que está vinculado a um comando de voz.

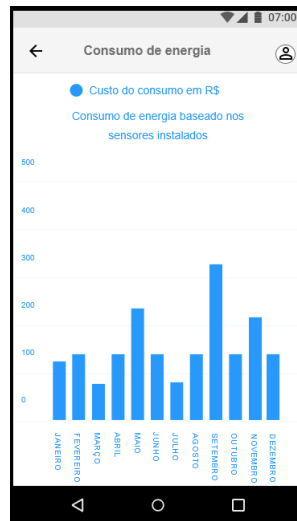
Figura 27: Diagrama de atividades da execução de um comando de voz através do aplicativo



Fonte: o autor (2019)

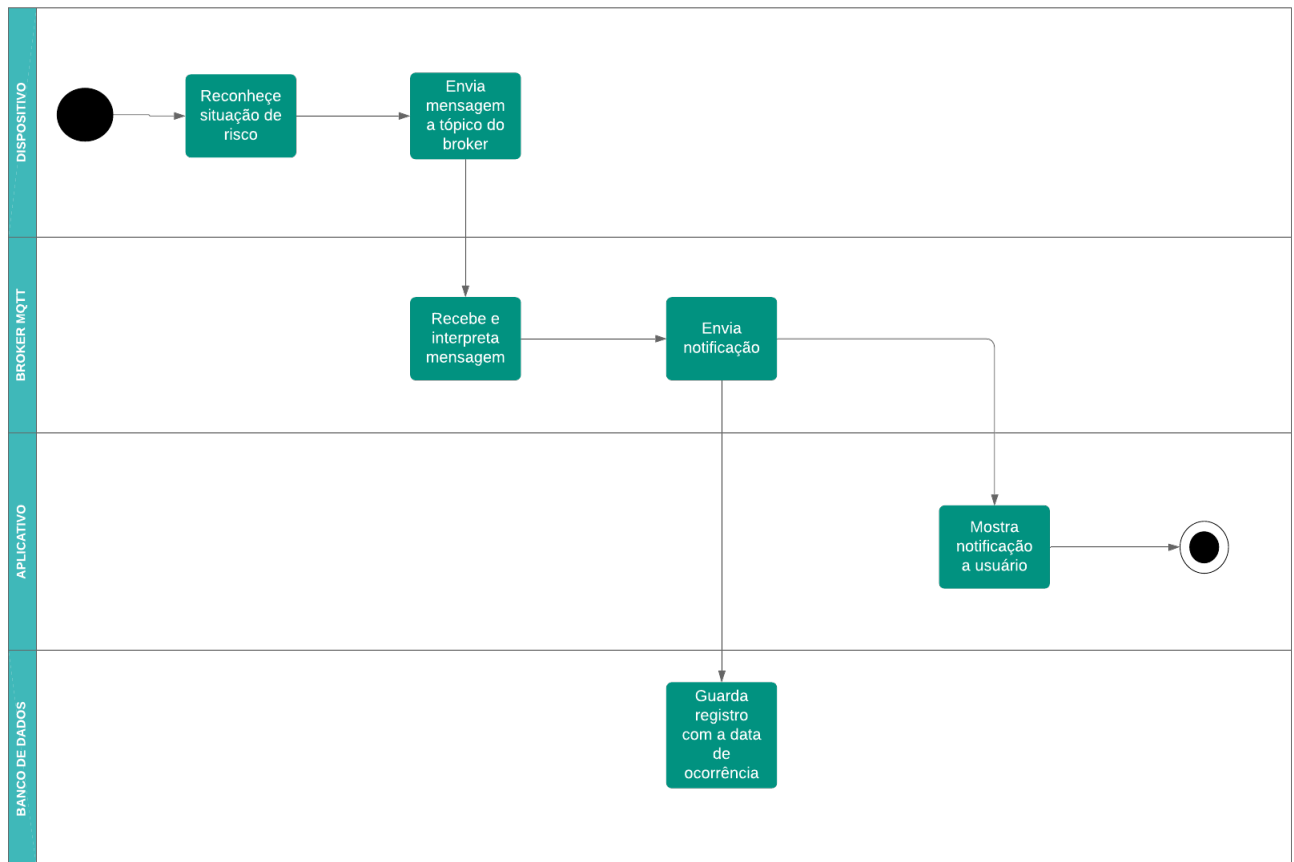
Também é importante a informação sobre o consumo de energia da residência, para isto foram simulados sensores que consigam medir e transmitir a informação calculada para a central, com estes dados guardados no banco de dados é possível mostrar ao usuário estas informações. A Figura 31 demonstra como é apresentada no aplicativo o consumo de energia que é registrado pelos sensores inseridos na residência. Para garantir a segurança dos usuários, foi instalado um sensor de gás para identificar se está ocorrendo um vazamento de gás de cozinha, a Figura 32 demonstra como é a integração dos componentes da central ao ocorrer uma situação considerada de risco para o usuário, esta integração é a mesma utilizada para o dispositivo de sensor de movimento, quando o mesmo identificar um intruso na residência.

Figura 31: Consumo de energia demonstrado no aplicativo

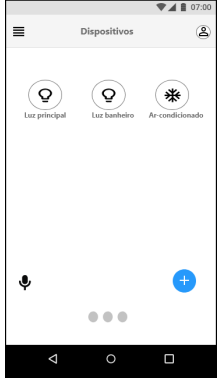
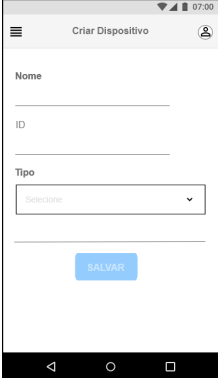
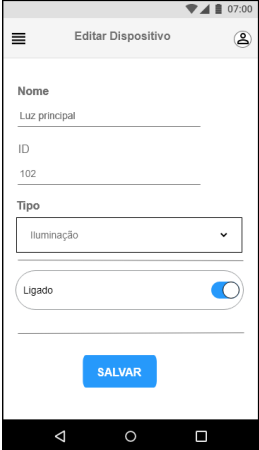


Fonte: o autor (2019)

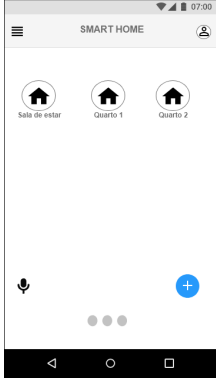
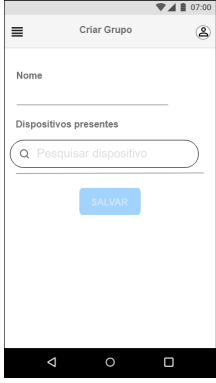
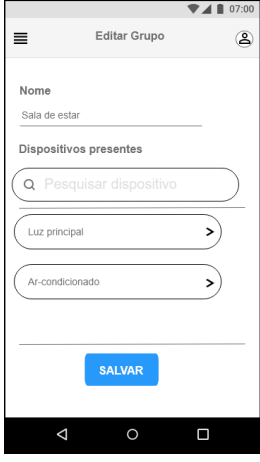
Figura 32: Diagrama de atividades da execução de um comando de voz através do aplicativo



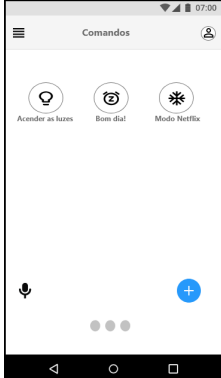
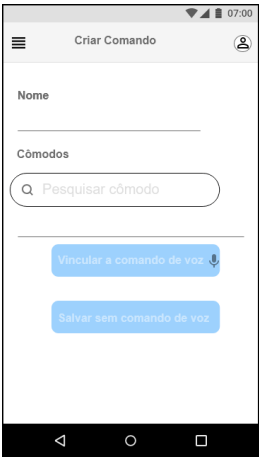

Fonte: o autor (2019)

Telas	Descrição
<p data-bbox="308 286 796 360">Figura 21: Listagens de dispositivos cadastrados no aplicativo.</p>  <p data-bbox="403 790 699 826">Fonte: o autor (2019)</p>	<p data-bbox="826 501 1321 611">Nesta tela é demonstrada como estarão listados os dispositivos já cadastrados no aplicativo</p>
<p data-bbox="308 869 796 943">Figura 22: Criação de um dispositivo no aplicativo.</p>  <p data-bbox="403 1373 699 1408">Fonte: o autor (2019)</p>	<p data-bbox="826 1084 1321 1193">Nesta tela é demonstrada a realização da criação de um novo dispositivo no aplicativo</p>
<p data-bbox="308 1451 796 1525">Figura 23: Edição de um dispositivo no aplicativo.</p>  <p data-bbox="403 2022 699 2058">Fonte: o autor (2019)</p>	<p data-bbox="826 1700 1321 1809">Nesta tela é demonstrada a realização da edição de um dispositivo já cadastrado no aplicativo</p>

Quadro 1: Dispositivos no aplicativo

Telas	Descrição
<p data-bbox="233 286 722 360">Figura 24: Listagens de grupos cadastrados no aplicativo.</p>  <p data-bbox="331 790 619 824">Fonte: o autor (2019)</p>	<p data-bbox="751 501 1241 611">Nesta tela é demonstrada como estarão listados os grupos já cadastrados no aplicativo</p>
<p data-bbox="233 869 722 943">Figura 25: Criação de um grupo no aplicativo.</p>  <p data-bbox="331 1373 619 1406">Fonte: o autor (2019)</p>	<p data-bbox="751 1084 1241 1193">Nesta tela é demonstrada a realização da criação de um novo grupo no aplicativo</p>
<p data-bbox="233 1451 722 1525">Figura 26: Edição de um grupo no aplicativo.</p>  <p data-bbox="331 2033 619 2067">Fonte: o autor (2019)</p>	<p data-bbox="751 1702 1241 1812">Nesta tela é demonstrada a realização da edição de um grupo já cadastrado no aplicativo</p>

Quadro 2: Grupos no aplicativo

Telas	Descrição
<p>Figura 28: Listagens de comandos cadastrados no aplicativo.</p>  <p>Fonte: o autor (2019)</p>	<p>Nesta tela é demonstrada como é realizada a listagem dos comandos já cadastrados no aplicativo</p>
<p>Figura 29: Criação de um comando no aplicativo.</p>  <p>Fonte: o autor (2019)</p>	<p>Nesta tela é demonstrada como é realizada a criação de um novo comando no aplicativo</p>
<p>Figura 30: Edição de um comando no aplicativo.</p>  <p>Fonte: o autor (2019)</p>	<p>Nesta tela é demonstrada como é realizada a edição de um comando já cadastrado no aplicativo</p>

4 IMPLEMENTAÇÃO

Para a solução desenvolvida neste trabalho, foram utilizadas algumas ferramentas e frameworks de programação existentes. Destaca-se que as ferramentas utilizadas neste trabalho são distribuídas sob licença de software livre. Estes softwares foram utilizados para permitir a integração entre o aplicativo para smartphones, a API REST que permite realizar a execução dos comandos requisitados pelo usuário e o servidor da central de automação.

Na solução apresentada neste capítulo, foram utilizados alguns sensores e módulos que permitem realizar os comandos diretamente na arquitetura da residência. Para esta solução, foi realizada a implementação em um cômodo da residência do autor, com a finalidade de cumprir os requisitos levantados pelo trabalho que diz respeito a comunicação do aplicativo com a central, e posteriormente a comunicação da central com os dispositivos e atuadores. Estes sensores serão descritos neste capítulo.

4.1 BANCO DE DADOS LOCAL

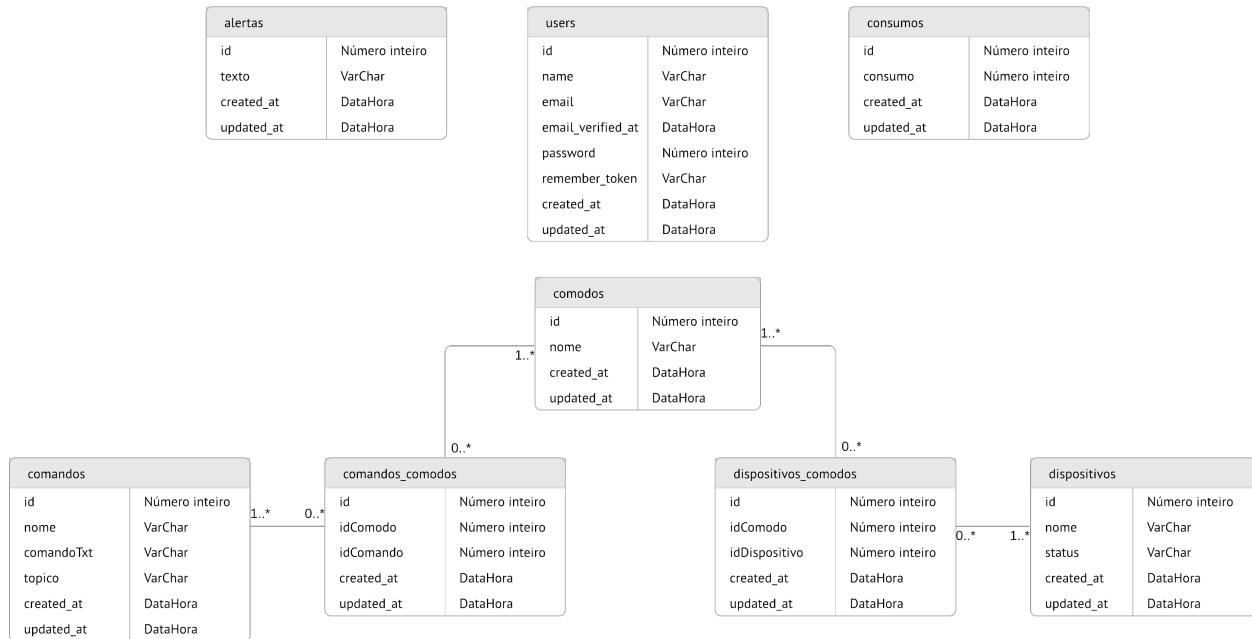
Para permitir gravar as entidades que são os objetos manipulados na parte de programação, foi utilizado um banco de dados relacional MySQL, os registros gravados neste banco permitem que sejam criados objetos manipuláveis no código fonte da central de automação. No banco de dados foram criadas tabelas com os atributos de cada entidade. A Figura 33 representa o modelo entidade relacionamento do banco de dados local.

4.2 APLICATIVO PARA SMARTPHONE

Durante a realização do projeto para o aplicativo de smartphone, foi decidido utilizar apenas a plataforma de celulares móveis Android, por este motivo foi utilizado para programar o aplicativo o ambiente de desenvolvimento Android Studio. Este ambiente é o ambiente oficial para desenvolvimento de aplicativos Android e possui várias vantagens ao utilizá-la, como um emulador de um smartphone com diversos recursos, a possibilidade de poder programar para todas as versões da plataforma, alguns frameworks que facilitam no desenvolvimento e apuração do código, entre outros (ANDROID, 2019).

Integrado ao Android Studio, para facilitar o desenvolvimento do aplicativo, foi utilizado um framework de desenvolvimento criada pela empresa Google, o Flutter. Este framework foi escolhido entre outros por alguns motivos, o principal sendo que o Flutter foi desenvolvido e ainda é utilizado pela Google, demonstrando que é um framework confiável e de qualidade, devido a relevância e importância desta empresa, outro motivo é que com

Figura 33: Modelo entidade relacionamento



Fonte: o autor (2019)

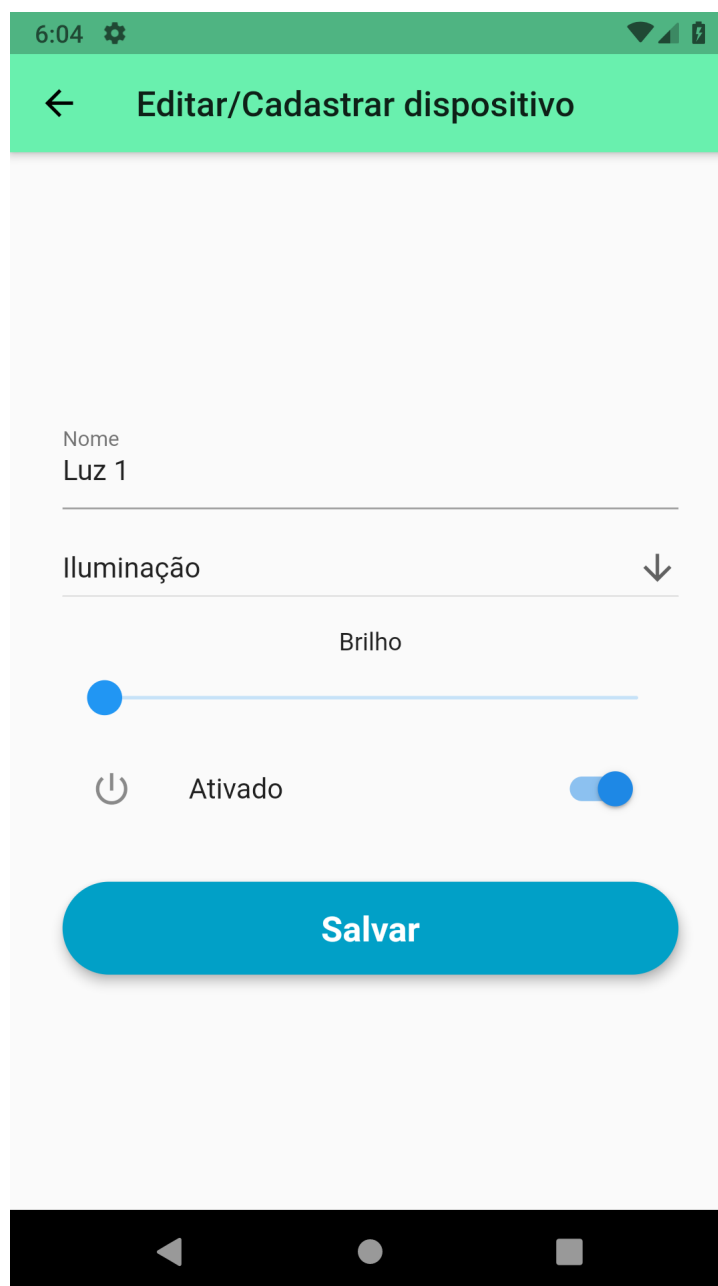
apenas um código, é possível conseguir fazer rodar tanto na plataforma Android quanto na plataforma iOS, que é a plataforma dos dispositivos da empresa Apple.

O aplicativo permite que sejam criados e editados os registros dos dispositivos, permitindo a comunicação entre estes e a central seja funcional e opere normalmente. Um dos exemplos disto é a possibilidade de controlar o brilho de uma lâmpada de forma automática, este exemplo foi utilizado realizando uma lâmpada LED e enviando comandos diretamente para uma placa NodeMCU com a porcentagem de brilho selecionada no aplicativo pelo usuário. Esta mesma tela também serve para realizar o acionamento e desligamento dos dispositivos, como os utilizados no cômodo de simulação do autor, esta tela está demonstrada na Figura 34.

Outra funcionalidade do aplicativo, devido a seu requisito funcional, é a de permitir cadastrar comandos que possam ser ativados através de um comando de voz. Quando o usuário acessar o ícone de microfone que aparece em baixo da tela do smartphone, abrirá uma nova tela onde o mesmo possa realizar um comando de voz. Se este comando, após convertido para texto através de uma biblioteca do próprio framework Flutter, for encontrado nos registros de comando no banco de dados, este comando será executado e as informações para os sensores e atuadores serão enviados através do servidor MQTT. A tela de vinculação de cômodos e sensores com o comando, e a possibilidade de vincular com um comando de voz está demonstrada na Figura 35.

Após realizada a programação dos sensores que notificam alertas considerados de

Figura 34: Tela do aplicativo responsável por editar e cadastrar dispositivos



Fonte: o autor (2019)

perigo, como um sensor de fumaça alertando um nível alto de fumaça no cômodo ou um sensor de movimento alertando que foi detectado uma movimentação indevida em um cômodo, e estes sensores forem registrados devidamente no aplicativo. Ao receber uma notificação destes sensores sobre uma situação de perigo, o aplicativo enviará para os seus usuários uma notificação de alerta, e ao acessar este alerta é possível verificar qual dispositivo que enviou esta notificação e o horário que foi identificado esta situação. A Figura 36 demonstra esta situação ocorrendo em um smartphone.

Figura 35: Tela do aplicativo de criação e edição de comandos

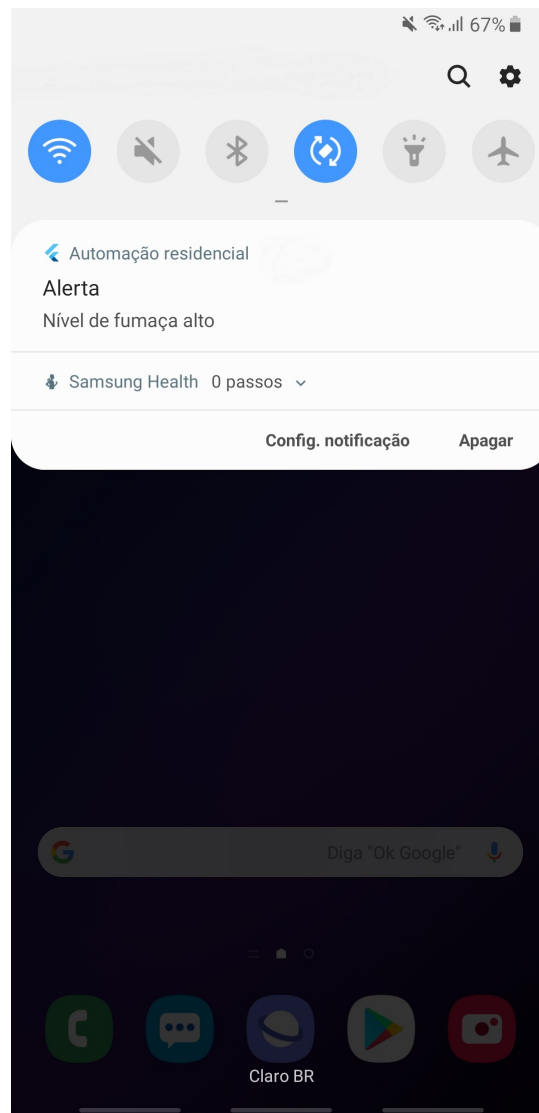


Fonte: o autor (2019)

4.3 SERVIDOR MQTT

Para o servidor MQTT, que é responsável por enviar e receber as mensagens nos tópicos entre os publishers e os subscribers, foi utilizado o servidor de código aberto Eclipse Mosquitto, a estrutura dos tópicos é feita de acordo com a entidade que está sendo utilizada ou requisitada pelo usuário ou pelos sensores e atuadores, mas também foi desenvolvido de uma forma que não gaste muitos recursos desnecessariamente.

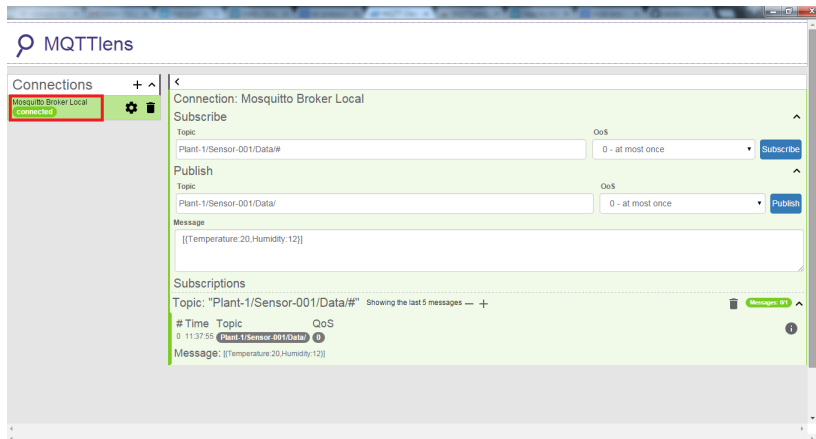
Os tópicos no servidor são separados para cada dispositivo, juntamente com o seu

Figura 36: Notificação de alarme no *smartphone*

Fonte: o autor (2019)

ID no banco de dados, para garantir que esteja sendo enviado para o mesmo sensor que foi requisitado, quando um comando for requisitado pelo usuário, a informação de cada ação do dispositivo será enviada no mesmo tópico que seria enviado caso o usuário selecionasse o dispositivo individualmente, isto evita criar tópicos distintos que realizariam a mesma ação.

Durante o desenvolvimento, para poder testar e avaliar o funcionamento do servidor com os dispositivos, foi utilizado o software MQTT Lens. Com este software foi possível facilitar a verificação do funcionamento da comunicação entre o servidor e os sensores que estavam inscritos nos tópicos de teste. A Figura 37 demonstra como é apresentada a tela ao utilizador do software.

Figura 37: Apresentação do *MQTT Lens*

Fonte: SEESIVA (2015)

4.4 CENTRAL DE AUTOMAÇÃO E API REST

Para o desenvolvimento da central de automação e da API REST, foi utilizado o framework Laravel, um framework web de código aberto para a linguagem de programação PHP. Este framework foi escolhido pois o autor já possuía uma experiência prévia o utilizando, inclusive em projetos de maior escala e mais robustos. Além de ser um dos frameworks mais populares na comunidade do GitHub.

Como o Laravel é um framework web, a API REST foi desenvolvida sobre este mesmo framework, pois com ele a criação de entidades, modelos e propriedades foram facilitadas, isto inclui também realizar a inserção das tabelas juntamente com seus atributos de forma automática. Com este framework, foi garantida a autenticação e validação de usuários que podem acessar e enviar comandos para a central de automação.

Para a API REST, as rotas geradas permitem que o usuário consiga listar, cadastrar, manipular e deletar as entidades, apenas controlando pela aplicação de smartphones.

Para a comunicação com o servidor MQTT, foi necessário instalar alguns pacotes de arquivos adicionais relacionados ao Eclipse Mosquitto no dispositivo da central, o qual utiliza o sistema operacional Fedora, baseado no Linux. Para que estes pacotes que foram instalados consigam ser utilizados no código-fonte da central, foi necessário alterar o arquivo responsável pela configuração do PHP, incluindo a extensão responsável pela classe do PHP(mosquitto.so). Após isto, foi possível incluir a classe MosquittoClient, que é a responsável por realizar a comunicação com o servidor MQTT, incluindo inscrições em tópicos e envio de mensagens.

4.5 SENSORES E ATUADORES

Para realizar a integração entre os comandos enviados da central através do MQTT para a arquitetura do cômodo de teste, foram utilizados diversos sensores conectados diretamente no cômodo ou em um protótipo para demonstrar sua funcionalidade, pois para alguns dos requisitos da central, a sua funcionalidade foi feita em forma de protótipo para simular a sua funcionalidade.

Para a programação das placas que utilizarão a rede sem-fio, foi utilizado o ambiente de desenvolvimento de código aberto Arduino, que originalmente é utilizado para programar placas Arduino, mas através de uma biblioteca disponibilizada pela comunidade é possível realizar esta programação nas placas que utilizam o chip ESP8266. A programação nestas placas é separada em duas funções principais, a Setup que é carregada apenas uma vez quando é ligada a placa e ela busca o código salvo na memória flash integrada, e a função Loop que é executada repetidamente durante o ciclo de vida da placa. A Figura 38 demonstra como é apresentado este ambiente de programação para o desenvolvedor.

Figura 38: Ambiente de desenvolvimento *Arduino*



Fonte: MICROSOFT (2019)

Para a comunicação com a rede sem-fio, através do protocolo Wi-Fi, foram utilizados módulos e placas que utilizam o chip ESP8266, a placa ESP-01 que é mais compacta e possui menos portas digitais, e a placa de desenvolvimento NodeMCU como demonstrado na Figura 39.

Figura 39: Placas da família do *chip* ESP8266

Fonte: o autor (2019)

Diferentemente do NodeMCU, a placa ESP-01 não possui comunicação serial integrada no módulo, a comunicação serial serve para realizar o envio do código para o módulo, no caso do NodeMCU para realizar este envio é necessário apenas inserir um cabo USB que permita enviar e receber dados, e após realizado o comando no ambiente de desenvolvimento Arduino o código está gravado na memória do chip, realizando as funções Setup e Loop. Então para realizar o envio do código para o módulo ESP-01, é necessário um conversor de serial para TTL(Transistor-transistor logic), por isso foi utilizada a placa FT232RL, demonstrada na Figura 40.

Figura 40: Placa FT232RL



Fonte: o autor (2019)

Pelo fato da placa ESP-01 ser mais compacta que a placa NodeMCU, foi decidido usar em alguns locais, como em alguns pontos da rede elétrica, o módulo relé de um canal que utiliza a placa ESP-01, demonstrado na Figura 41. Para controlar este módulo, é necessário definir a porta digital 0 no modo de saída, e ativar e desativar ela de acordo

Figura 41: Módulo relé de 1 canal com ESP01



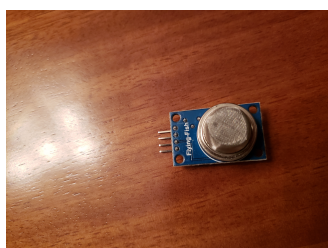
Fonte: o autor (2019)

Figura 42: Soldagem do módulo relé



Fonte: o autor (2019)

Figura 43: Sensor de fumaça



Fonte: o autor (2019)

Figura 44: Sensores de Movimento PIR



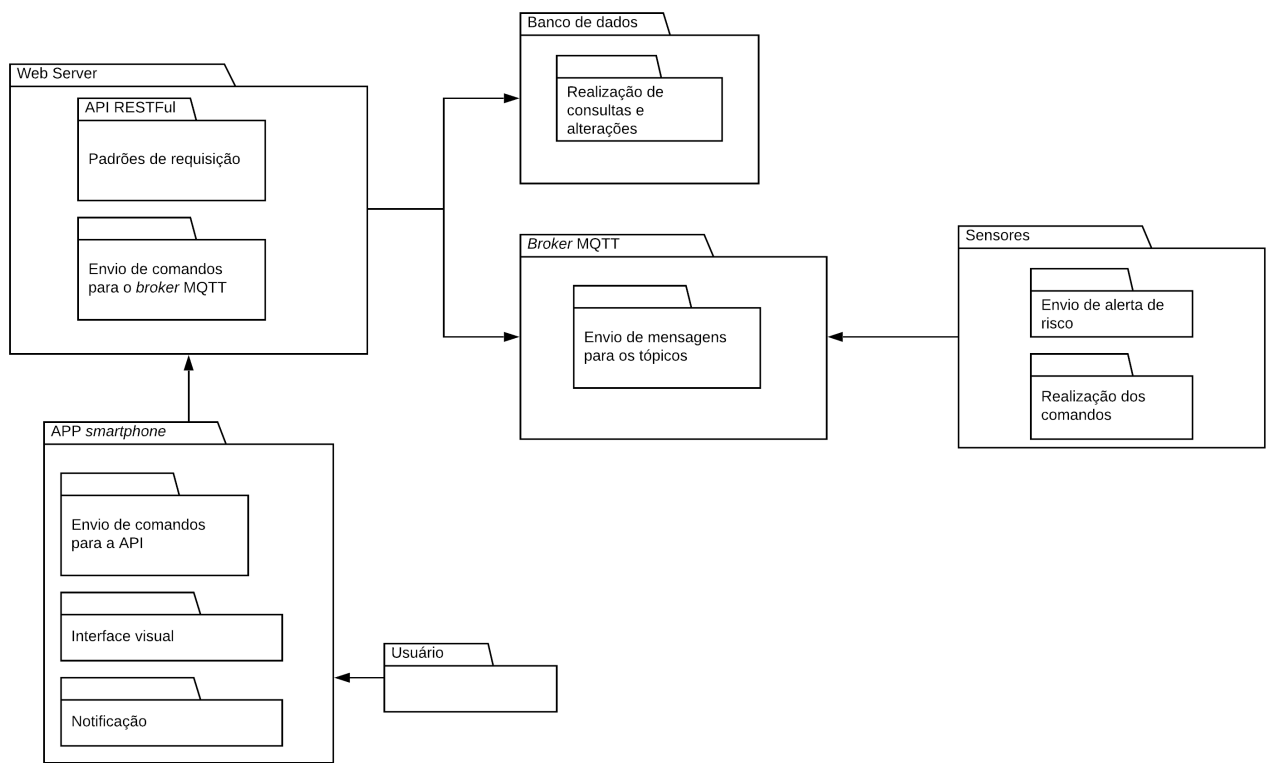
Fonte: o autor (2019)

com a lógica do programa, neste caso a requisição do usuário. Pelo fato do módulo ser construído para funcionar nativamente com a placa ESP-01S, e não a placa ESP-01, foi necessário realizar algumas soldagens diretamente neste módulo para permitir que a placa não entre em modo de programação ao invés de modo de operação, que roda o programa que está carregado na memória, esta soldagem está demonstrada na Figura 42.

Para os sensores de alertas, que notificarão o aplicativo assim que for detectado uma situação de risco, foram utilizados sensores de movimentos PIR e detecção de fumaça MQ-02, apresentados nas Figuras 44 e 43, respectivamente.

Na implementação teste, no cômodo do autor, foram utilizados 3 módulos relés juntamente com 3 placas ESP-01 com a finalidade de possuir uma implementação física que não seja apenas baseado em protótipos, como a utilizada para os sensores de fumaça e o de movimento, ou simulada como foi realizada a inserção dos dados de consumo de energia. Mas apesar de serem apenas protótipos ou simulações, é possível reproduzir as funcionalidades de acordo com seus requisitos levantados no trabalho. Cada uma destas 3 placas instaladas diretamente na rede elétrica da residência do autor possui um dispositivo registrado no banco de dados e é controlada pelo aplicativo. Uma destas placas é responsável por controlar o acionamento e desligamento do ar-condicionado do cômodo, enquanto as outras placas possuem a responsabilidade de acionar e desligar pontos de luzes do cômodo, cada uma destas com um ponto diferente. A imagem 45 representa um diagrama de pacotes representando toda a solução realizada neste trabalho

Figura 45: Diagrama de pacotes da solução



Fonte: o autor (2019)

5 CONSIDERAÇÕES FINAIS

Neste trabalho foi desenvolvida e implementada- uma solução de automação residencial, utilizando um dos cômodos da residência do autor como ambiente teste para validação dos requisitos. No conjunto da solução estão presentes um broker MQTT, responsável por enviar e receber mensagens entre a central e os sensores e atuadores da automação. Um aplicativo para smartphones da plataforma Android, responsável por permitir o controle remoto da automação para o usuário. Um servidor próprio responsável por hospedar a aplicação API REST, que é a aplicação no qual o aplicativo envia as requisições solicitadas pelo usuário. E por fim um banco de dados relacional MySQL, responsável por guardar os registros das entidades que são cadastradas via aplicativo. Destaca-se que em todos estes componentes da solução, foram utilizados softwares e bibliotecas distribuídas sobre licença livre.

Para a realização da solução de automação residencial, foram realizadas algumas pesquisas com diversas finalidades para decidir o que foi utilizado na solução. Uma das pesquisas se refere às classificações de redes para IoT referente a distribuição geográfica que a solução irá atuar. Outra pesquisa se refere à escolha entre uma solução de rede cabeada ou sem fio, esta segundo sendo a escolhida para a simulação. A partir da escolha de uma rede sem fio, foram pesquisados os diversos protocolos de comunicação sem fio para a IoT, utilizando como objetos de pesquisa os protocolos mais utilizados em automações atualmente, e como é conciliado protocolos diferentes em uma mesma solução de automação. Por fim foi realizado uma pesquisa referente a protocolos de comunicação utilizados na IoT, que são os responsáveis por realizar a comunicação entre a central e os dispositivos.

Pelo objetivo deste trabalho demandar uma integração completa e funcional entre os componentes da solução, as operações realizadas sobre o cômodo da residência do autor representam uma simulação adequada de uma automação residencial por completo, pois ao acrescentar mais cômodos ou dispositivos, a eficiência da solução não irá ser afetada, visto que não existe uma dependência entre a quantidade de cômodos ou dispositivos registrados para o funcionamento da solução.

Para o desenvolvimento dos componentes que necessitam de código-fonte para programar o seu funcionamento, foi decidido realizar a utilização de frameworks e bibliotecas de software livre, a fim de permitir um desenvolvimento mais ágil e garantindo que os softwares criados para a solução sejam estáveis e confiáveis, visto que a realização de uma central segura é um dos requisitos do trabalho.

Para comprovar o funcionamento da central de automação e a comunicação entre os seus componentes, foi implementado diretamente na rede elétrica de um cômodo do

autor algumas placas ESP-01 integradas a módulos relé. As placas foram programadas previamente, antes de sua instalação na rede elétrica junto aos módulos. Após alimentadas estas placas realizaram a conexão ao servidor MQTT e a residência inscrição aos seus respectivos tópicos para permitir a comunicação com a central. Um destes dispositivos controlava o acionamento e desligamento do ar-condicionado do cômodo, enquanto os restantes controlavam o acionamento e desligamento de pontos de luzes diferentes.

A simulação no cômodo da residência do autor demonstrou que a solução de automação residencial desenvolvida neste trabalho conseguiu integrar os seus componentes, e permitiu realizar uma comunicação homogênea entre os diversos protocolos utilizados. Para alguns requisitos foram utilizadas simulações que representam as funcionalidades necessárias para cumpri-los, utilizando um LED acendendo e apagando, demonstrando que a comunicação entre os atuadores e a central está sendo realizada com sucesso.

5.1 SUGESTÕES DE TRABALHOS FUTUROS

Como trabalhos futuros sugere-se:

- *Uma solução de automação que envolva a residência com todos os seus cômodos. Neste trabalho não foi aplicado devido a limitações de tempo.*
- *Comparação dos resultados obtidos ao criar uma solução de automação residencial utilizando diferentes protocolos.*
- *Uma solução utilizando a programação Over-The-Air (OTA) para atualização de códigos.*

REFERÊNCIAS

ADELANTADO, F. et al. *Understanding the limits of LoRaWAN*. IEEE Communications Magazine, Institute of Electrical and Electronics Engineers (IEEE), v. 55, n. 9, p. 34–40, 2017. Disponível em: <<https://doi.org/10.1109/mcom.2017.1600613>>.

AKYILDIZ, I. F. *Nanonetworks: A new frontier in communications*. In: 2010 International Conference on Wireless Information Networks and Systems (WINSYS). [S.l.: s.n.], 2010. p. IS-5–IS-5.

AL-QASEEMI, S. A. et al. *IoT architecture challenges and issues: Lack of standardization*. In: 2016 Future Technologies Conference (FTC). IEEE, 2016. Disponível em: <<https://doi.org/10.1109/ftc.2016.7821686>>.

ALI, M. N. *Design and implementation of a secure campus network*. Journal of Surface Engineered Materials and Advanced Technology, v. 5, 07 2015.

ANDROID. *Conheça o Android Studio*. 2019. Disponível em: <<https://developer.android.com/studio/intro>>. Acesso em: 10 nov. 2019.

ASIARFID. *NFC Facts Applications That Will Broaden Your Mind*. 2018. Disponível em: <<http://www.asiarfid.com/rfid-journal/nfc-facts-and-applications-that-will-broaden-your-mind.html>>. Acesso em: 04 mai. 2019.

BUYYA, R.; DASTJERDI, A. V. *Internet of Things. Principles and Paradigms*. [S.l.]: Morgan Kaufmann, 2016.

CLARK, D. D.; POGGAN, K. T.; REED, D. P. *An introduction to local area networks*. Proceedings of the IEEE, v. 66, n. 11, p. 1497–1517, Nov 1978. ISSN 0018-9219.

CROW, B. et al. *IEEE 802.11 wireless local area networks*. IEEE Communications Magazine, Institute of Electrical and Electronics Engineers (IEEE), v. 35, n. 9, p. 116–126, set. 1997. Disponível em: <<https://doi.org/10.1109/35.620533>>.

DANVIBOON, P. *What is MAN network?* 2013. Disponível em: <<https://ttnetwork.wordpress.com/2013/11/29/what-is-man-network>>. Acesso em: 04 mai. 2019.

ELPROCUS. *ZigBee Wireless Technology Architecture and Applications*. 2014. Disponível em: <<https://www.elprocus.com/what-is-zigbee-technology-architecture-and-its-applications>>. Acesso em: 18 mai. 2019.

ELPROCUS. *Sensor Network – Know all about BAN Body Area Network*. 2018. Disponível em: <<https://www.elprocus.com/ban-body-area-network>>. Acesso em: 04 mai. 2019.

FARNELL. *LoRaWAN - 101 : An Overview on one of the trendiest IoT technologies presently!* 2017. Disponível em: <<https://uk.farnell.com/lorawan-101>>. Acesso em: 18 mai. 2019.

FIELDING, R. et al. Hypertext Transfer Protocol – HTTP/1.1. [S.l.], 1999. Disponível em: <<https://doi.org/10.17487/rfc2616>>.

FINKENZELLER, K. RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication. Wiley, 2010. ISBN 0470695064. Disponível em: <<https://www.amazon.com/RFID-Handbook-Fundamentals-Identification-Communication/dp/0470695064?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimb05-20&linkCode=sm2&camp=2025&creative=165953&creativeASIN=0470695064>>.

FLUTTER. Technical overview. 2019. Disponível em: <<https://flutter.dev/docs/resources/technical-overview>>. Acesso em: 10 nov. 2019.

FOUNDATION, E. MQTT 101 – How to Get Started with the lightweight IoT Protocol. 2014. Disponível em: <https://www.eclipse.org/community/eclipse_newsletter/2014/october/article2.php>. Acesso em: 04 mai. 2019.

FOUNDATION, E. Eclipse Mosquitto. 2019. Disponível em: <<https://projects.eclipse.org/projects/technology.mosquitto>>. Acesso em: 08 jun. 2019.

GALAL, A.; HESSELBACH, X. Nano-networks communication architecture: Modeling and functions. Nano Communication Networks, v. 17, p. 45 – 62, 2018. ISSN 1878-7789. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1878778918300164>>.

GARBAR, D. The Top 10 Advantages Of Using Laravel PHP Framework. 2019. Disponível em: <<https://belitsoft.com/blog/10-benefits-using-laravel-php-framework>>. Acesso em: 10 nov. 2019.

GUPTA, A. DIFFERENCE BETWEEN LAN CAN MAN AND WAN IN TABULAR FORM. 2018. Disponível em: <<http://www.learnabhi.com/difference-between-lan-can-man-and-wan>>. Acesso em: 04 mai. 2019.

HU, Y.; NANDA, A.; YANG, Q. Measurement, analysis and performance improvement of the apache web server. In: 1999 IEEE International Performance, Computing and Communications Conference (Cat. No.99CH36305). IEEE, 1999. Disponível em: <<https://doi.org/10.1109/pccc.1999.749447>>.

HUNKELER, U.; TRUONG, H. L.; STANFORD-CLARK, A. MQTT-s — a publish/subscribe protocol for wireless sensor networks. In: 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08). IEEE, 2008. Disponível em: <<https://doi.org/10.1109/comswa.2008.4554519>>.

INSTRUCTABLES. ESP8266 a Complete Beginners Guide (IOT). 2019. Disponível em: <<https://www.instructables.com/id/ESP8266-a-Complete-Beginners-Guide-IOT>>. Acesso em: 22 jun. 2019.

KASHYAP, P. Types of Network | Different Computer Networks. 2019. Disponível em: <<https://technicalknowledge.in/types-of-network/>>. Acesso em: 04 mai. 2019.

LEE, J.-S. et al. A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi. Industrial electronics society, v. 5, p. 46–51, 2007.

MASSE, M. REST API Design Rulebook. O'Reilly Media, 2011. ISBN 1449310508. Disponível em: <<https://www.amazon.com/REST-Design-Rulebook-Mark-Masse/dp/1449310508?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=sm2&camp=2025&creative=165953&creativeASIN=1449310508>>.

MICROSOFT. Uploading Board Code and Arduino IDE. 2019. Disponível em: <<https://support.office.com/en-us/article/uploading-board-code-and-arduino-ide-a9723765-1314-49e0-a69b-bb5c3e1f628d>>. Acesso em: 11 nov. 2019.

MILLER, B. A.; BISDIKIAN, C. Bluetooth Revealed: The Insider's Guide to an Open Specification for Global Wireless Communication. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001. ISBN 0-13-090294-2.

MITCHELL, B. Overview of a Personal Area Network (PAN). 2019. Disponível em: <<https://www.lifewire.com/definition-of-pan-817889>>. Acesso em: 04 mai. 2019.

MOLLENAUER, J. Standards for metropolitan area networks. IEEE Communications Magazine, Institute of Electrical and Electronics Engineers (IEEE), v. 26, n. 4, p. 15–19, abr 1988. Disponível em: <<http://dx.doi.org/10.1109/35.442>>.

MULLIGAN, G. The 6lowpan architecture. In: Proceedings of the 4th workshop on Embedded networked sensors - EmNets '07. ACM Press, 2007. Disponível em: <<https://doi.org/10.1145/1278972.1278992>>.

NAIK, N. Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In: 2017 IEEE International Systems Engineering Symposium (ISSE). IEEE, 2017. Disponível em: <<https://doi.org/10.1109/syseng.2017.8088251>>.

NEGRA, R.; JEMILI, I.; BELGHITH, A. Wireless body area networks: Applications and technologies. Procedia Computer Science, Elsevier, v. 83, p. 1274–1281, 2016.

NETCRAFT. May 2019 Web Server Survey. 2019. Disponível em: <<https://news.netcraft.com/archives/2019/05/10/may-2019-web-server-survey.html>>. Acesso em: 08 jun. 2019.

PARK, Y. T.; STHAPIT, P.; PYUN, J.-Y. Smart digital door lock for the home automation. In: TENCON 2009 - 2009 IEEE Region 10 Conference. IEEE, 2009. Disponível em: <<https://doi.org/10.1109/tencon.2009.5396038>>.

PAUTASSO, C.; WILDE, E.; ALARCON, R. REST: Advanced Research Topics and Practical Applications. Springer New York, 2013. (SpringerLink : Bücher). ISBN 9781461492993. Disponível em: <<https://books.google.com.br/books?id=jhS4BAAAQBAJ>>.

PHP. O que o PHP pode fazer? 2019. Disponível em: <https://www.php.net/manual/pt_BR/intro-whatcando.php>. Acesso em: 11 nov. 2019.

REHMAN, J. Advantages and disadvantages of campus area network (CAN). 2019. Disponível em: <<http://www.itrelease.com/2019/04/advantages-and-disadvantages-of-campus-area-network-can>>. Acesso em: 04 mai. 2019.

ROUTE-XP. Introduction to Routers and WAN networks. 2017. Disponível em: <<http://www.routexp.com/2017/12/introduction-to-routers-and-wan-networks.html>>. Acesso em: 04 mai. 2019.

SAKOVICH, N. Internet of Things (IoT) Protocols and Connectivity Options: An Overview. 2018. Disponível em: <<https://www.sam-solutions.com/blog/internet-of-things-iot-protocols-and-connectivity-options-an-overview>>.

SCHNEPS-SCHNEPPE, M. et al. Wired smart home: Energy metering, security, and emergency issues. In: 2012 IV International Congress on Ultra Modern Telecommunications and Control Systems. *IEEE*, 2012. Disponível em: <<https://doi.org/10.1109/icumt.2012.6459700>>.

SEESIVA. Installing and using MQTT Lens with Mosquitto. 2015. Disponível em: <<https://sivatechworld.wordpress.com/2015/08/01/installing-and-using-mqtt-lens-with-mosquitto/>>. Acesso em: 10 nov. 2019.

SHELBY, Z.; HARTKE, K.; BORMANN, C. The Constrained Application Protocol (CoAP). [S.l.], 2014. Disponível em: <<https://doi.org/10.17487/rfc7252>>.

ZOLERTIA. WHAT IS 6LOWPAN AND WHY SHOULD I TRY IT IN MY IOT PROJECT? 2019. Disponível em: <<https://zolertia.io/6lowpan-iot-protocol>>. Acesso em: 18 mai. 2019.