UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

FWIOTFURB: UM FRAMEWORK PARA DESENVOLVIMENTO DE APLICAÇÕES IOT

RODRIGO ORTHMANN NIELSON

BLUMENAU
2018

RODRIGO ORTHMANN NIELSON

FWIOTFURB: UM FRAMEWORK PARA DESENVOLVIMENTO DE APLICAÇÕES IOT

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Dalton Solano dos Reis, M.Sc. - Orientador

BLUMENAU
2018

FWIOTFURB: UM FRAMEWORK PARA DESENVOLVIMENTO DE APLICAÇÕES IOT

Por

RODRIGO ORTHMANN NIELSON

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof. Dalton Solano dos Reis, M.Sc. – Orientador, FURB
Membro:	Prof. Miguel Alexandre Wisintainer, M.Sc. – FURB
Membro:	Prof. Marcos Rodrigo Momo, M.Sc. – FURB

Blumenau, 12 de julho de 2018

Dedico este trabalho à família, aos meus amigos e a todos que colaboraram de alguma forma para a conclusão deste trabalho.

AGRADECIMENTOS

A Deus.

Aos meus pais Alberto e Roseli por todo o apoio.

Ao meu orientador Dalton e ao professor Miguel pela confiança e auxílio na elaboração deste trabalho.

À minha amiga Gabriela por toda a ajuda.

Aos meus amigos.

A dualidade compõe o homem, mas há tempos deixamos nosso pior lado ser o predominante.

Sergio Silva Junior

RESUMO

Este trabalho apresenta o desenvolvimento de um *framework* que é capaz de realizar a comunicação com equipamentos eletroeletrônicos residenciais, além de um aplicativo que utiliza o *framework* desenvolvido. Todo o trabalho foi desenvolvido utilizando a IDE Visual Studio Code e o Ionic Framework 3. Foram desenvolvidas duas interfaces de comunicação para o *framework*, sendo elas comunicação via Bluetooth, que utiliza um *plugin* do Cordova, e comunicação via Wi-Fi, que utiliza o protocolo de comunicação Message Queuing Telemetry Transport (MQTT). Além disso, foi possível desenvolver o aplicativo e comunicar com um dispositivo via Bluetooth e com outro dispositivo via MQTT. Também foi realizada a avaliação do *framework* com acadêmicos do curso de Ciência da Computação da FURB, a fim de avaliar a usabilidade do *framework*. A partir da avaliação, foi possível observar bons resultados e confirmar a utilidade do *framework* para desenvolvedores. Por fim, são levantadas as principais contribuições deste trabalho para a comunidade de desenvolvedores.

Palavras-chave: Bluetooth. Ionic Framework. Internet das coisas. MQTT.

ABSTRACT

This paper presents the development of a framework that can perform the communication with residential electrical appliances, in addition to an application that uses the developed framework. Both the framework and the application were developed using the Visual Studio Code IDE and Ionic Framework 3. Two communication interfaces were developed for the framework, the first one is the communication via Bluetooth, which uses a Cordova plugin, and the second one uses Wi-Fi communication, which uses the Message Queuing Telemetry Transport (MQTT) protocol. In addition, it was possible to develop an application that uses the developed framework and communicate with a device via Bluetooth and with another device via MQTT. The framework was also evaluated by students from the course of Computer Science at FURB, in order to evaluate the it's usability. From the evaluation, it was possible to observe good results and confirm the usefulness of the framework for developers. Finally, the main contributions of this work to the community of developers are raised.

Key-words: Bluetooth. Ionic Framework. Internet of things. MQTT.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

FURB – Fundação Universidade Regional de Blumenau

IOT – Internet das coisas

MQTT – Message Queuing Telemetry Transport

RF – Requisitos Funcionais

RNF – Requisitos não Funcionais

SUMÁRIO

1 INTRODUÇÃO

Desde a época da pré-história e até os dias do hoje o homem constrói casas. No início apenas como meio de proteção, e hoje as casas são adaptadas aos diferentes estilos de vida das pessoas, sendo multifuncionais e se mesclando com a tecnologia (DINIZ, 2014). Junto a evolução da própria casa, houve também a evolução dos equipamentos eletrônicos, e uma maior presença desses nas casas. Conforme Fracchetta (2015, p. 1) "com o aparecimento de inúmeros equipamentos eletroeletrônico nas moradias, começa a surgir a necessidade de comanda-los a distância e mesmo monitorar de outro lugar fora da residência."

A automação residencial visa suprir essa necessidade criada pela evolução das casas e a inserção de equipamentos eletroeletrônicos nas mesmas. Por meio da automação residencial, é possível fazer o uso de aplicativos móveis para o controle de equipamentos eletroeletrônicos residenciais, como por exemplo ligar e desligar uma simples lâmpada ou até mesmo controlar um sistema completo de *Home Teather* (ROVERI, 2012).

No mercado atual já existem diversos aplicativos e equipamentos que possibilitam o controle de eletroeletrônicos. Por exemplo o HomeKit da Apple, um *framework* disponível para iOS, que possibilita aplicativos se comunicarem e controlarem equipamentos conectados. Conforme Apple (2017b) um aplicativo que utiliza esse HomeKit é o Home, esse aplicativo possui funcionalidades para adicionar e controlar acessórios (equipamentos certificados pela Apple), além de criar cômodos que agregam vários acessórios e cenas. As cenas possibilitam o controle de vários acessórios simultaneamente, por exemplo "você pode criar uma cena chamada **Chegar em casa** que acende todas as luzes e destranca a porta da frente com um único toque." (APPLE, 2017b, p. 1, grifo do autor).

Contudo, os aplicativos e/ou HomeKits existentes no mercado possuem problemas. Um deles é a falta de flexibilidade, seja pelo aplicativo somente aceitar equipamentos de fabricantes específicos, ou o aplicativo poder ser utilizado em apenas uma plataforma, ou ainda os equipamentos aceitarem somente aplicativos específicos. No caso do HomeKit da Apple, os equipamentos comercializados por empresas precisam seguir especificações definidas pela Apple para poderem ser utilizados no aplicativo e a aplicação funciona somente no sistema operacional da Apple, o iOS (APPLE, 2017a).

Diante do exposto, este trabalho realizou o desenvolvimento de um *framework* multiplataforma que faz a comunicação e o controle de equipamentos eletroeletrônicos, além de um aplicativo que utiliza o *framework* desenvolvido e funciona como um controle remoto

centralizado, trazendo assim uma aplicação capaz de controlar equipamentos eletroeletrônicos residenciais.

1.1 OBJETIVOS

O objetivo deste trabalho é elaborar um *framework* multiplataforma para controle de equipamentos eletroeletrônicos residenciais.

Os objetivos específicos são:

a) desenvolver um aplicativo multiplataforma que utilize o *framework*;
b) controlar através do aplicativo pelo menos um tipo de equipamento;
c) realizar testes de funcionalidade e avaliações de usabilidade do *framework*.

1.2 ESTRUTURA

Este trabalho está organizado em quatro capítulos. Sendo que o primeiro apresenta a introdução do trabalho e seus principais objetivos. O segundo capítulo apresenta a fundamentação teórica necessária para o desenvolvimento deste trabalho, abordando os temas: automação residencial, Ionic Framework, o protocolo MQTT e Internet das Coisas. Além disso, o segundo capítulo também aborda os trabalhos correlatos ao *framework* e aplicativo desenvolvidos. O terceiro capítulo aborda sobre toda a etapa do desenvolvimento do trabalho, apresentando seus requisitos, sua especificação, o processo de implementação a análise dos resultados obtidos. Por fim, o quarto capítulo apresenta as conclusões deste trabalho e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os principais conceitos necessários para a elaboração deste trabalho. A seção 2.1 apresenta os conceitos básicos de automação residencial, a seção 2.2 aborda as funcionalidades do Ionic Framework, a seção 2.3 explica brevemente o funcionamento do protocolo MQTT, a seção 2.4 aborda os conceitos básicos de Internet das Coisas e por fim, a seção 2.5 apresenta os trabalhos correlatos ao *framework* e aplicativo desenvolvidos.

2.1 AUTOMAÇÃO RESIDENCIAL

A automação residencial pode ser entendida como um sistema capaz de controlar automaticamente equipamentos eletroeletrônicos residenciais ou facilitar processos que normalmente seriam realizados por ações humanas (SPIVEY, 2015). Com a constante evolução dos dispositivos móveis, a automação residencial tem mudado significantemente nos últimos anos. Esses dispositivos móveis, tais como tablets e *smartphones*, juntos às redes sem fio Bluetooth e Wi-Fi possibilitam a criação de ambientes cada vez mais inteligentes e automatizados. O uso de aplicativos móveis faz com que tablets e *smartphones* se transformem em verdadeiros controles remotos, capazes de controlar o sistema de automação de uma casa inteira (KYAS, 2013).

Com a implantação de um sistema de automação residencial pode-se obter diversos benefícios, conforme o objetivo do usuário. O primeiro deles é uma maior conveniência e conforto, por exemplo em uma sala de estar, a partir de um aplicativo instalado no celular seria possível ajustar as luzes, ligar o ar-condicionado e a televisão sem a necessidade de sair do sofá. Outro benefício da automação residencial seria na área de segurança, tornando possível o controle e monitoramento de câmeras de segurança, receber alertas de alarmes de segurança ou até mesmo o controle de travas de segurança e portões eletrônicos. Além dos pontos positivos citados, a automação residencial também traz benefícios nas áreas de acessibilidade e economia/controle do consumo de energia elétrica (SPIVEY, 2015).

Conforme Kyas (2013), a automação residencial possui cinco pilares para sua implementação. O primeiro se refere aos dispositivos sob controle, que são equipamentos eletroeletrônicos, como televisores e sistemas de som, que estão conectados a um sistema de automação residencial. O segundo pilar não está presente em todos os sistemas de automação, este é composto por sensores que são capazes de, por exemplo, medir temperatura, humidade do ar ou até mesmo detectar movimento. Além dos sensores, os atuadores também estão presentes nesse pilar, servindo como mecanismos que realizam alguma ação no mundo real,

como por exemplo motores e bombas elétricas. No terceiro pilar estão as redes de controle, que são responsáveis por manter a conectividade entre os dispositivos sob controle, sensores, atuadores e controles remotos. Nas redes de controle existe um destaque para as redes sem fio, como Wi-Fi e Bluetooth, por possuírem uma boa área de alcance e também por serem tecnologias de baixo custo. O próximo pilar é composto pelo controlador, que tem a responsabilidade de fazer o controle dos comandos, dos dados e de acesso das aplicações. O controlador recebe informações dos sensores e os comandos enviados pelos controles remotos. Esses controladores podem ser servidores instalados na própria residência ou até mesmo servidores web na nuvem. Por fim, o último pilar é composto pelos controles remotos, que têm a responsabilidade de enviar comandos para realizar ações. Atualmente os principais controles utilizados são os *tablets* e *smartphones*, por se tratar de dispositivos com uma boa acessibilidade, além da alta conectividade.

Atualmente a automação residencial encontra-se em evolução crescente, trazendo novas tecnologias, além de estar tornando-se cada vez mais possível e acessível para as pessoas. Com a alta conectividade presente nos dias de hoje, não se faz mais necessário adquirir tantos equipamentos para se fazer automação dentro das residências, por conta de muitas pessoas já possuírem uma rede Wi-Fi em suas moradias, além de um *smartphone*, essas que são peças chave para muitos sistemas de automação residencial (SPIVEY, 2015). Certamente para realizar a automação completa de uma casa ainda existe um alto custo, mas dispositivos como por exemplo o Sonoff (Figura 1), facilitam o desenvolvimento e a implantação de sistemas de automação residencial.

Figura 1 - Sonoff

Fonte: Itead (2017).

O Sonoff é um interruptor inteligente que pode ser controlado remotamente através de aplicativos para as plataformas Android e iOS, que devem estar conectados em uma rede Wi-Fi. Este aparelho possibilita ligar ou desligar a passagem de energia elétrica, podendo ser utilizado por exemplo, para controlar lâmpadas (ITEAD, 2017). Mesmo se tratando de um aparelho simples e que não consegue realizar o controle de funções especificas dos equipamentos, este aparelho é uma solução de baixo custo que pode ser utilizado para se dar início à implantação de um sistema de automação residencial, o que demonstra a crescente evolução nos sistemas de automação residencial e também a sua maior acessibilidade nos dias de hoje.

2.2 IONIC FRAMEWORK

O Ionic Framework pode ser definido como um Software Development Kit (SDK) que possibilita o desenvolvimento de aplicações híbridas, ou seja, ele permite o desenvolvimento de aplicações que possuam uma interface Web e possibilita que essas aplicações façam uso de componentes nativos do dispositivo móvel. Esses componentes nativos correspondem, por exemplo à câmera de um dispositivo móvel. Ainda, o Ionic também tem um foco importante na interface e na experiência do usuário, trazendo assim uma interface mais amigável e responsiva, ou seja, ajustável ao tamanho da interface do dispositivo móvel. Além disso, o *framework* é multiplataforma, o que significa que uma mesma aplicação pode ser distribuída e executada em diferentes plataformas, como por exemplo as plataformas Android, iOS, Windows Phone e até mesmo desktop. O Ionic Framework é um SDK *open source* e possui a licença MIT, o que significa que pode ser utilizado em projetos tanto pessoais quanto comerciais gratuitamente (DRIFTY, 2017a).

Para o desenvolvimento da interface, são oferecidos diversos componentes, que incluem botões, alertas, campos de texto, menus, entre diversos outros componentes. Todos eles possuem uma aparência web, pois utilizam tecnologias web (HTML, CSS e Javascript). Esses componentes vêm com uma aparência padrão do Ionic, porém é possível customizá-los de modo a se adequar à necessidade da aplicação que está sendo desenvolvida, ou até mesmo criar os próprios componentes (DRIFTY, 2017b).

Para a utilização de recursos nativos do dispositivo móvel é necessária a utilização do Cordova, que é uma plataforma para desenvolvimento de aplicações nativas, oferecendo o acesso à diversos recursos nativos do dispositivo móvel. O Ionic Native faz o encapsulamento das APIs e *plugins* do Cordova, e serve como uma interface de comunicação. Este utiliza a linguagem de programação TypeScript para se comunicar com as APIs do Cordova,

facilitando assim o desenvolvimento e integração das aplicações (SAINI, 2017). São oferecidos diversos recursos nativos através do Cordova, como por exemplo a comunicação via Bluetooth, este *plugin* se chama Bluetooth Low Energy (BLE) e possibilita o dispositivo móvel buscar dispositivos com que estão com o Bluetooth ativado, se conectar a esses dispositivos, ler e escrever valores e por fim receber notificações. Além disso, também é disponibilizado um *plugin* para comunicação com Arduinos, chamado de Bluetooth Serial, facilitando assim o desenvolvimento de aplicações relacionadas com automação residencial (DRIFTY, 2017b).

Por fim, através do Ionic Framework, também são oferecidos recursos de armazenamento de dados, tanto de armazenamento local, com a utilização do SQLite através do Cordova, quanto na nuvem. A interface para utilização de banco de dados oferecida pelo Ionic possibilita a configuração e utilização de vários bancos de dados simultâneos, podendo definir prioridades para sua utilização. Por exemplo, quando utilizando uma aplicação em um contexto Web, a aplicação preferencialmente irá tentar salvar as informações em um banco de dados na nuvem, e caso este banco esteja indisponível ou o dispositivo móvel esteja sem acesso à internet, serão gravados os dados no banco de dados local e posteriormente sincronizados. Também são disponibilizados componentes para autenticação de usuário, que possibilitam a integração e autenticação com diversas plataformas, como por exemplo Google, Facebook, Twitter, entre outros

2.3 MQ TELEMETRY TRANSPORT (MQTT)

O MQ Telemetry Transport (MQTT) é um protocolo de rede para troca de mensagens entre dispositivos, no formato publicação e inscrição. Por se tratar de um protocolo com baixo consumo de banda e por ser muito leve, seu uso se torna ideal para aplicações de internet das coisas. Além disso, é possível a utilização de usuário e senha na transmissão de pacotes, o que aumenta a sua segurança (YUAN, 2017).

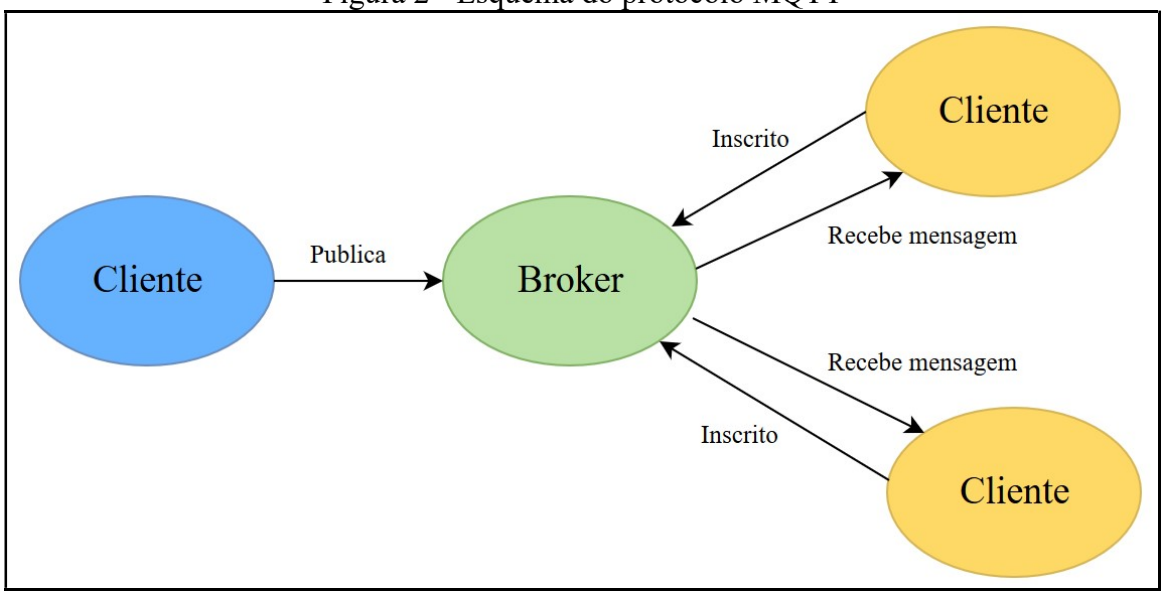
Conforme Yuan (2017):

> O protocolo MQTT define dois tipos de entidades na rede: um message broker e inúmeros clientes. O broker é um servidor que recebe todas as mensagens dos clientes e, em seguida, roteia essas mensagens para os clientes de destino relevantes. Um cliente é qualquer coisa que possa interagir com o broker e receber mensagens. (p. 1).

A Figura 2 apresenta o esquema de funcionamento do protocolo MQTT. Neste esquema é possível observar que existem dois clientes (em alaranjado) que são inscritos em um determinado tópico do *broker*, sendo que um tópico funciona como um endereço no

broker, para separar as mensagens recebidas. Quando o cliente azul publica uma mensagem no *broker*, todos os clientes que são inscritos no tópico de publicação, recebem a mensagem publicada.

Figura 2 - Esquema do protocolo MQTT



Fonte: elaborado pelo autor.

Um exemplo de aplicação para o protocolo MQTT seria: um cliente que corresponde a um dispositivo que pode ligar e desligar uma lâmpada é inscrito em um determinado tópico; outro cliente que corresponde a um *smartphone* publica uma mensagem neste determinado tópico; quando o cliente (dispositivo) recebe a mensagem, ele acende uma lâmpada.

2.4 INTERNET DAS COISAS

Inicialmente o termo Internet das Coisas (IoT) foi utilizado por Kevin Ashton para descrever um sistema em que objetos do mundo físico poderiam se conectar com a Internet por sensores. Atualmente, o termo IoT se tornou um termo popular para descrever cenários de alta conectividade e a capacidade de objetos computacionais se comunicarem com objetos e dispositivos do dia-a-dia (ROSE; ELDRIDGE; CHAPIN, 2015, p. 12).

A Internet das Coisas pode utilizar diferentes métodos de comunicação, um exemplo disso é a comunicação entre dispositivos. Esse tipo de comunicação é composto por dois dispositivos que se comunicam diretamente, geralmente esse método de comunicação é utilizado para sistemas de automação residencial, sendo um exemplo a utilização da comunicação via Bluetooth entre um dispositivo móvel e uma lâmpada que possua uma interface de comunicação Bluetooth. Outro exemplo de tipo de comunicação seria a comunicação dispositivo-nuvem. Neste método de comunicação existem um ou mais dispositivos conectados na internet, conectados diretamente à um serviço da nuvem. Esses

dispositivos podem enviar ou receber mensagens do serviço em que estão conectados. Um exemplo disso seria um sensor de temperatura e um dispositivo móvel conectados à uma aplicação na nuvem utilizando o protocolo MQTT. Sempre que o sensor detecta alguma mudança de temperatura, ele envia uma mensagem para a aplicação na nuvem, que por sua vez envia esta mensagem para o dispositivo móvel (ROSE; ELDRIDGE; CHAPIN, 2015, p. 13).

2.5 TRABALHOS CORRELATOS

A seguir são apresentados três trabalhos que possuem características similares ao objetivo do trabalho proposto. Na seção 2.5.1 será exposto o trabalho feito por Ciocari (2013), um aplicativo móvel para controle e monitoramento de energia elétrica de equipamentos residenciais. Na seção 2.5.2 será abordado o *framework* Hades Tools (PRADO, 2012). Por fim, a seção 2.5.3 descreve um aplicativo para controle de sala de *Home Theater* (MORIBE, 2013).

2.5.1 Controle e monitoramento do consumo de energia elétrica de equipamentos residenciais via Android

O trabalho desenvolvido por Ciocari (2013) descreve um aplicativo desenvolvido para a plataforma Android, que é capaz de realizar o controle de equipamentos residenciais, além de controlar o consumo de energia elétrica. Para isso o autor propõe o desenvolvimento de um protótipo, desenvolvendo tanto o hardware, quanto o software (aplicativo móvel).

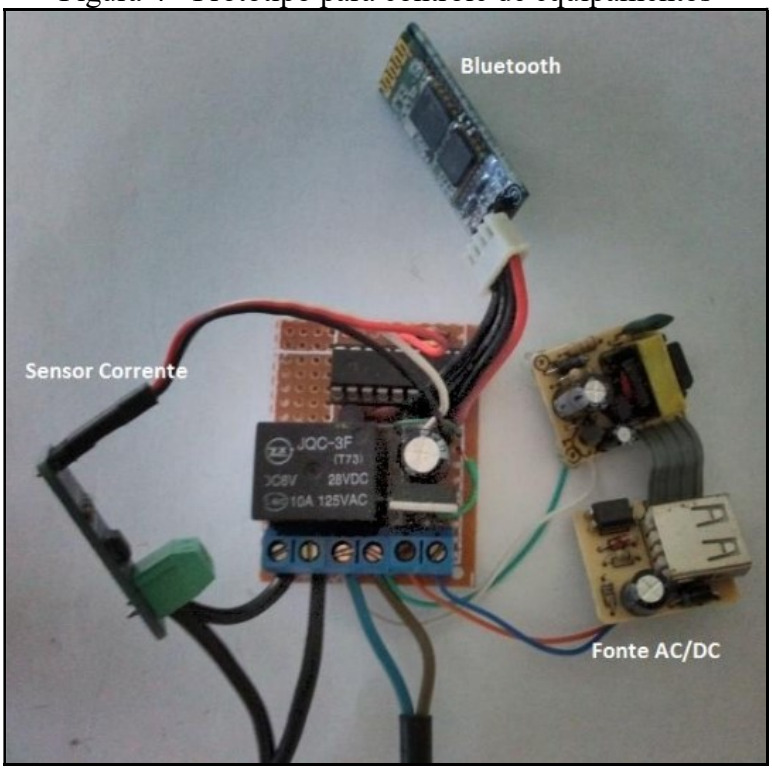
Ciocari (2013) afirma que seu trabalho é justificado pelo fato de os usuários necessitarem de informações sobre o consumo de energia elétrica de seus equipamentos residenciais em tempo real. O software desenvolvido para Android utiliza comunicação via Bluetooth, conseguindo assim detectar equipamentos disponíveis no ambiente. Na tela inicial, são mostradas algumas informações gerais relacionadas ao consumo de energia, além da opção de ligar ou desligar o equipamento (Figura 3).

Figura 3 - Tela principal do aplicativo Android

Fonte: Ciocari (2013).

Em relação ao *hardware* utilizado para o desenvolvimento de seu protótipo, foi utilizado um módulo Bluetooth classe 2, um sensor de corrente do tipo ACS712T ELC-05B por ser um equipamento com alta precisão, um regulador de tensão LM 1117 3,3V, um relê de 5V e 10A, um microcontrolador dsPIC 33FJ12GP201 e uma fonte, além de outros componentes básicos como capacitores, resistores, etc. A Figura 4 apresenta o protótipo desenvolvido.

Figura 4 - Protótipo para controle de equipamentos



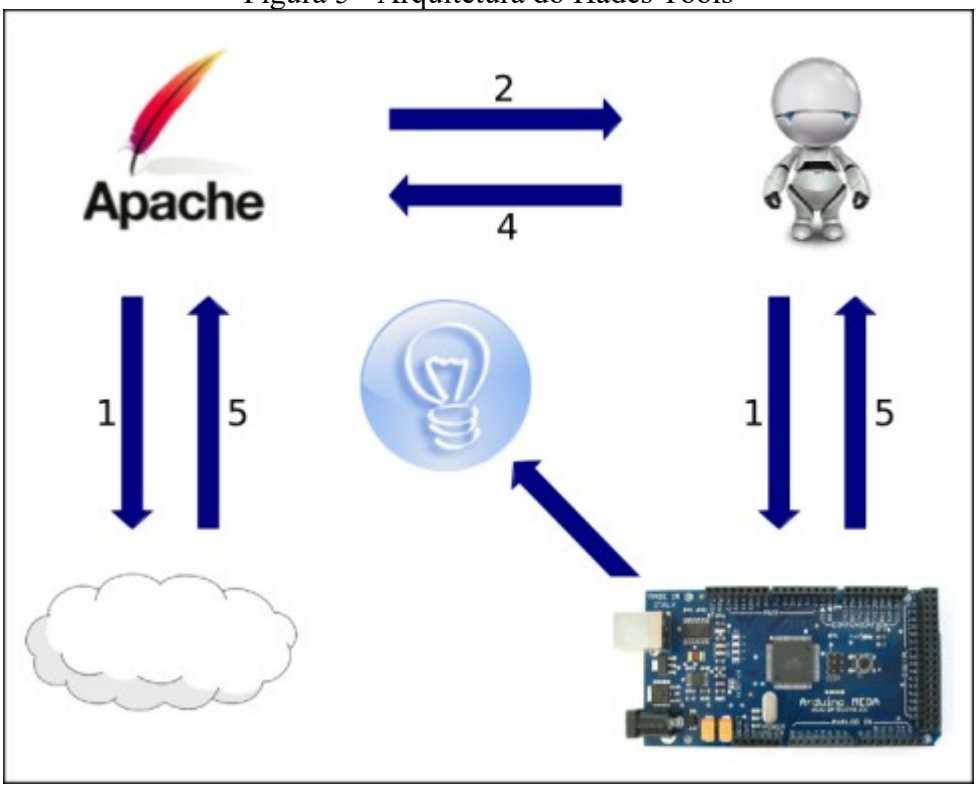
Fonte: Ciocari (2013).

De acordo com Ciocari (2013), o protótipo apresentado cumpriu os objetivos do trabalho, sendo possível monitorar e controlar o consumo de energia elétrica de equipamentos residenciais. Este trabalho apresenta como pontos fortes o bom controle de consumo de energia elétrica, além de possuir uma interface móvel. Por outro lado, a interface móvel foi desenvolvida especificamente para a plataforma Android, não sendo multiplataforma, além de apresentar uma interface não muito amigável ao usuário final.

2.5.2 Framework web para automação

O *framework* desenvolvido por Prado (2012), chamado de Hades Tools, tem como objetivo criar um *framework* para desenvolvimento de aplicações relacionadas a automação residencial, abstraindo o funcionamento dos dispositivos. O funcionamento do *framework* é ilustrado pela Figura 5.

Figura 5 - Arquitetura do Hades Tools



Fonte: Prado (2012).

Conforme Prado (2012), a proposta implementada funciona com um cliente (ilustrado pela nuvem) que se comunica com um servidor web (Apache), podendo enviar e receber mensagens. Este servidor Web é chamado de HadesCGI.

O HadesCGI é responsável por receber as mensagens do cliente e fazer o controle de acesso dos recursos. Após feito o controle de acesso da mensagem do cliente, a mensagem é enviada para o HadesServer (representado pelo robô), que é um segundo servidor web. Este tem a responsabilidade de manter o banco de dados atualizado e centralizar todas as informações, além de ser responsável pela comunicação com os dispositivos de controle. Já o Arduino tem o papel de dispositivo de controle, ficando responsável por enviar mensagens de comando para algum equipamento, como por exemplo um comando para ligar ou desligar uma lâmpada (PRADO, 2012).

Assim, o autor apresenta que grande parte dos objetivos foram alcançados dando destaque a criação do *framework*, agregando vários dispositivos, com a integração de dispositivos com emissores infravermelho e dispositivos ON/OFF (lâmpadas). Neste trabalho foi possível observar a construção de um *framework* que fosse capaz de controlar dispositivos, possibilitando assim o desenvolvimento de aplicações para controle de equipamentos residenciais, sem a preocupação com funcionamento dos equipamentos. Por outro lado, o

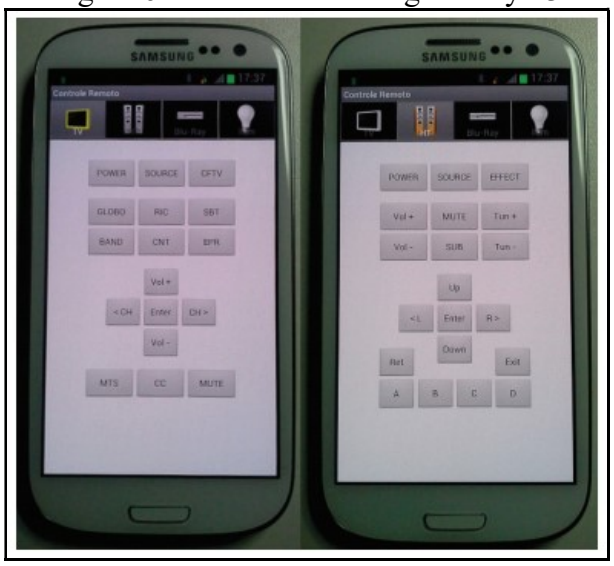
autor não descreveu os testes realizados com o *framework*, além de não mostrar a utilização do *framework* em um aplicativo.

2.5.3 Automação de Sala de Home Theater Utilizando Dispositivos Móveis Baseados em Android

O trabalho proposto por Moribe (2013) tem como objetivo desenvolver "um protótipo para controlar uma sala de *Home Theater* utilizando *smartphone* ou *tablet* baseados em Android". Como requisitos funcionais são apresentados o controle das principais funcionalidades da televisão, *Home Theater*, *DVD/Blu-Ray* e as funcionalidades do controlador de iluminação (MORIBE, 2013).

O autor realizou tanto o desenvolvimento do hardware quanto o software. A aplicação desenvolvida por Moribe (2013) possui uma aba para o controle de cada tipo de equipamento, dividido assim entre televisão, *Home Theater*, DVD/Blu-ray e iluminação (Figura 6).

Figura 6 - Telas no Samsung Galaxy S3



Fonte: Moribe (2013).

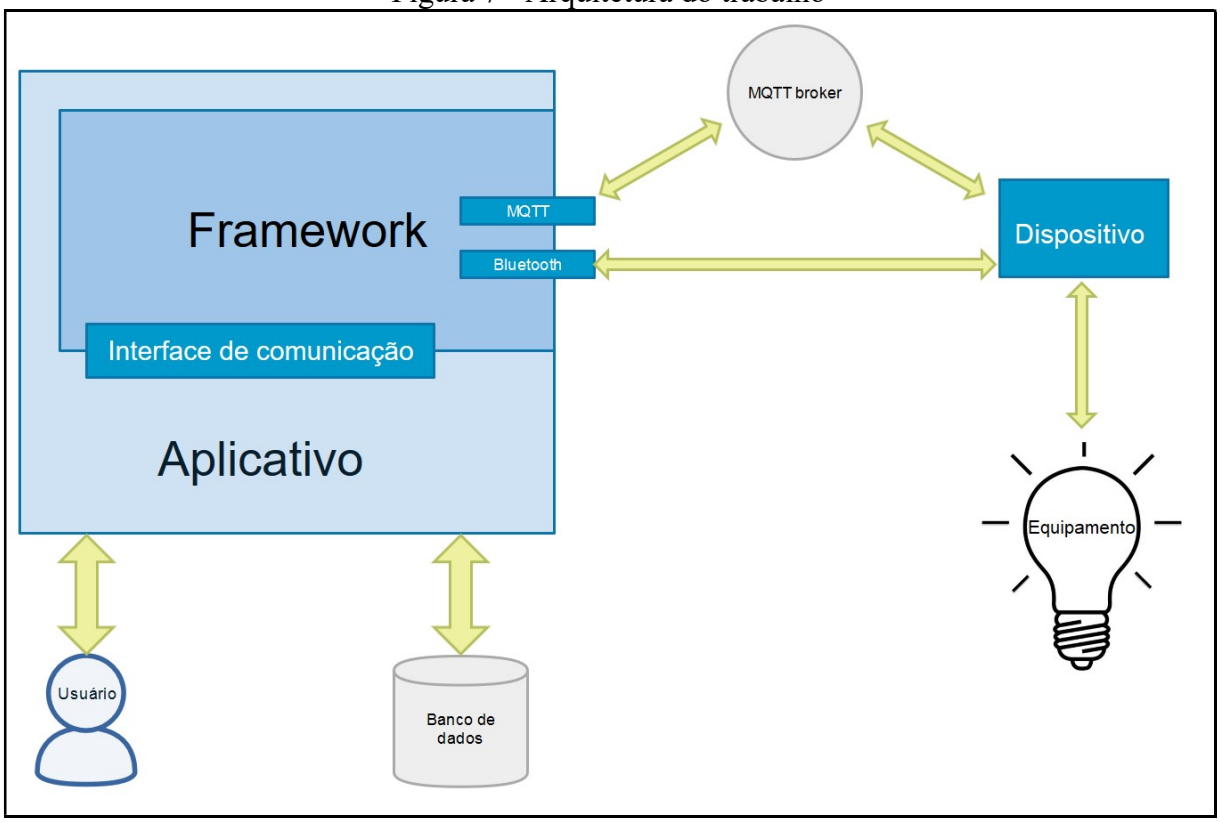
Para a validação do protótipo foram realizados testes em uma sala de *Home Theater*, utilizando um *smartphone* Samsung Galaxy S3 e um *tablet* Galaxy Note 10.1, testando as funcionalidades e comprovando o funcionamento do protótipo (Moribe, 2013). O autor não descreveu exatamente os testes realizados, apenas garante que todas as funcionalidades foram testadas.

Foi possível observar bons resultados no protótipo proposto, por conta da aplicação conseguir controlar o funcionamento de uma sala de *Home Theater.* Sendo assim, o protótipo é uma boa solução pelo fato de ser um aplicativo móvel e conseguir agregar mais de um tipo de dispositivo e suas funcionalidades (MORIBE, 2013).

3 DESENVOLVIMENTO

Esta seção aborda as principais etapas do desenvolvimento tanto do *framework* quanto do aplicativo, estes que estão disponibilizados no Bitbucket (NIELSON, 2018). O trabalho foi desenvolvido em duas etapas: primeiro, o desenvolvimento do *framework*, descrito na seção 3.1. Segundo, o desenvolvimento do aplicativo descrito na seção 3.2. A arquitetura do trabalho desenvolvido pode ser visualizada na Figura 7.

Figura 7 - Arquitetura do trabalho



Fonte: elaborado pelo autor

A arquitetura apresentada possui os elementos essenciais para o funcionamento do sistema como um todo. Primeiramente o *framework*, que funciona como o núcleo da aplicação. Ele possui duas interfaces de comunicação com um dispositivo. A comunicação através da interface MQTT passa por um *broker* antes de chegar ao dispositivo e a interface Bluetooth comunica diretamente com o dispositivo. Um dispositivo é capaz de controlar algum equipamento e fica responsável por receber os comandos das interfaces e realizar ações em cima de um equipamento. Um exemplo de dispositivo seria o Sonoff que pode ligar e desligar um equipamento em que está conectado, como uma lâmpada.

O aplicativo por sua vez tem a responsabilidade de prover uma interface para o usuário final, fornecendo uma tela de controle para os dispositivos. O controle dos dispositivos é feito através das interfaces de comunicação do *framework*, que possui as chamadas para as funções

das interfaces MQTT e Bluetooth que permitem enviar comandos para os dispositivos. Além disso, o aplicativo também fica responsável por fazer o armazenamento dos dados dos usuários, que é feito através do Firebase.

3.1 DESENVOLVIMENTO DO FRAMEWORK

A seguir são listados os requisitos do *framework* desenvolvido:

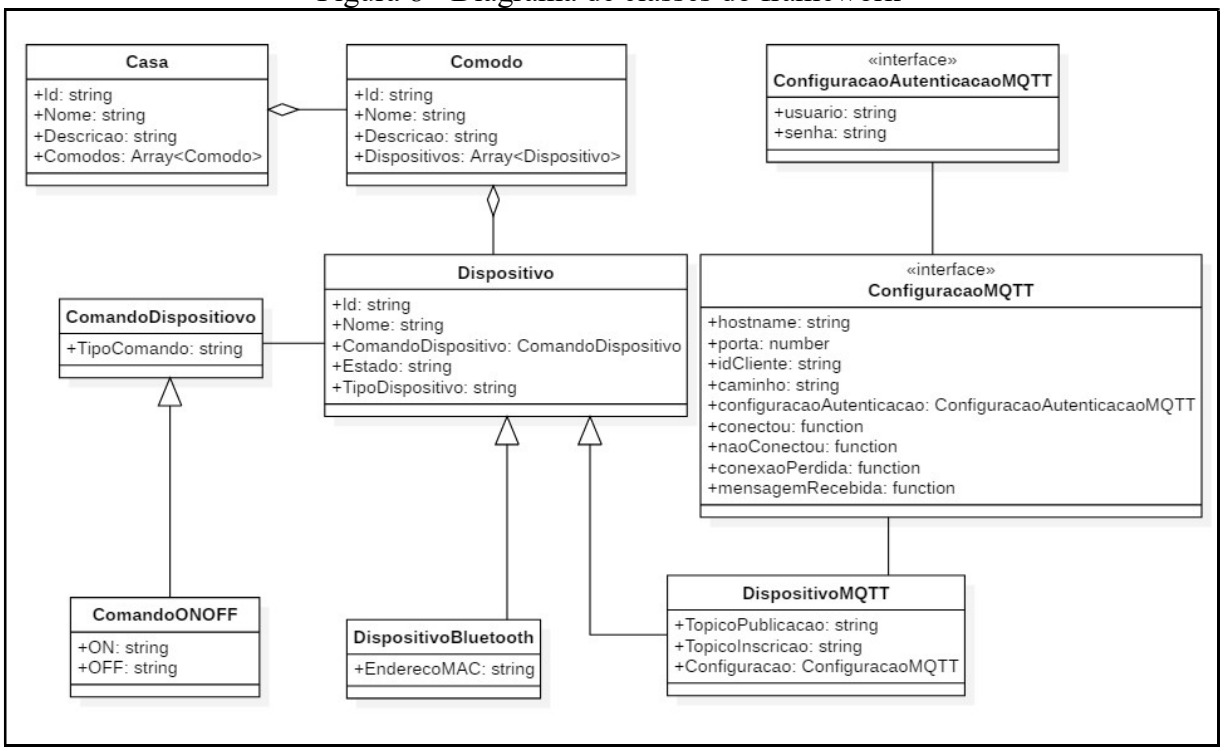
a) permitir a comunicação com dispositivos através do protocolo MQTT (Requisito Funcional - RF);
b) permitir a comunicação com dispositivos via Bluetooth (RF);
c) possuir objetos que simulem a estrutura de uma casa (RF);
d) utilizar a linguagem de programação TypeScript (Requisito Não Funcional - RNF);
e) ser compatível com o Ionic Framework 3 (RNF).

3.1.1 ESPECIFICAÇÃO

Esta seção apresenta a especificação do *framework* desenvolvido. A especificação do *framework* foi feita seguindo a Unified Modeling Language (UML), além disso, foi utilizada a ferramenta StarUML para a elaboração dos diagramas.

A Figura 8 apresenta o diagrama de classes do *framework*. O diagrama permite ter uma visão geral das classes de controle providas pelo *framework*. Essas classes tem o objetivo de trazer objetos que se assemelhem com a estrutura de uma casa real.

Figura 8 - Diagrama de classes do framework



Fonte: elaborado pelo autor.

Conforme o diagrama, pode-se observar que a classe `Casa` funciona como um agregador em que a partir dela podem ser encontrados todos os cômodos e dispositivos. Essa estrutura de classes permite a simulação de um ambiente de uma casa real, tendo em vista que a casa pode agregar vários cômodos. A classe `Comodo` tem o objetivo de separar os dispositivos dentro da casa, fornecendo assim uma forma de melhorar a disposição dos dispositivos dentro da casa.

A classe `Dispositivo` funciona apenas como uma classe base para as especializações de dispositivo, trazendo assim uma abordagem genérica e possibilitando a criação de novos tipos de dispositivo quando necessário. Nesta mesma ideia, a classe `ComandoDispositivo` também traz uma abordagem genérica em relação aos diferentes tipos de comando para um dispositivo. Neste caso foi elaborado apenas a classe `ComandoONOFF` que é utilizada para dispositivos com apenas dois estados.

3.1.2 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas para a implementação do *framework*.

3.1.2.1 Técnicas e ferramentas utilizadas

O *framework* foi desenvolvido utilizando a linguagem de programação TypeScript para ser compatível com o Ionic Framework. Além disso, todo o desenvolvimento foi realizado no editor de código fonte Visual Studio Code, por trazer diversas funcionalidades que agilizam o desenvolvimento, como *autocomplete*, ferramentas de *debug* e integração com o Git (MICROSOFT, 2018).

3.1.2.2 Implementação do framework

O desenvolvimento do *framework* pode ser dividido em duas partes principais: o desenvolvimento da interface MQTT e o desenvolvimento da interface Bluetooth. Em ambos os casos o desenvolvimento foi iniciado dentro de um projeto Ionic e depois de completo, o *framework* foi movido para um projeto isolado. Para o desenvolvimento da interface de comunicação MQTT foi utilizada uma biblioteca desenvolvida em TypeScript que implementa os métodos do protocolo MQTT, essa biblioteca se chama ng2-mqtt. Essa interface foi nomeada `FwMqttProvider` e ela fornece as principais funcionalidades para a comunicação via MQTT. A função utilizada para se conectar a um *broker* MQTT se chama `configurarMQTT`, conforme o Quadro 1.

Quadro 1 – Função que configura um *broker* MQTT

```
public configurarMQTT(configuracao: ConfiguracaoMQTT) {
    this.Cliente = new Paho.MQTT.Client(configuracao.hostname,
configuracao.porta, configuracao.idCliente);

    this.Cliente.onConnectionLost = this.conexaoPerdida;

    this.Cliente.onMessageArrived = configuracao.mensagemRecebida !=
null ? configuracao.mensagemRecebida : this.mensagemRecebida;

    this.conectarCliente({
        configuracaoAutenticacao: configuracao.configuracaoAutenticacao,
        conectou: configuracao.conectou,
        naoConectou: configuracao.naoConectou
    });
}
```

Fonte: elaborado pelo autor.

Essa função recebe as informações necessárias para configurar um *broker* MQTT (*hostname*, porta, usuário e senha) e se conecta ao *broker*, através de outra função, chamada `conectarCliente`. Além disso, ela também possui propriedades que recebem uma função de *call-back* para quando é realizada alguma ação no *broker*, como por exemplo a `onConnectionLost`. A função desta propriedade é chamada quando o *framework* perde a conexão com o *broker*, deste modo, essas propriedades possibilitam um maior controle com relação à conexão do *framework* com o *broker*.

Depois que conectado à um *broker* MQTT, já é possível enviar mensagens para ele. O envio de mensagens é feito através da função `publicar`. Essa função é apresentada no Quadro 2.

Quadro 2 - Função que publica em um *broker* MQTT

```
public publicar(mensagem: string, topico: string, qos?: number): void {
    if (this.Cliente != null && this.Cliente.isConnected()) {
      let msg = new Paho.MQTT.Message(mensagem);
      msg.destinationName = topico;
      if (qos != null)
        msg.qos = qos;
      this.Cliente.send(msg);
    }
  }
```

Fonte: elaborado pelo autor.

A função `publicar` recebe a mensagem a ser publicada, o tópico de publicação e o *Quality of Service* (QoS), sendo que o QoS 0 não garante a entrega da mensagem enviada, o QoS 1 garante que a mensagem enviada será entregue pelo menos uma vez, podendo ser entregue mais de uma e o QoS 2 que garante que a mensagem enviada será entregue apenas uma vez . Com isso, é feita a chamada para a biblioteca ng2-mqtt e é publicada a mensagem para o *broker* MQTT conectado no momento. Além dessas funções, o *framework* também provém outras funções básicas para o uso do MQTT, como `inscrever` e `desinscrever`, que são utilizados para se inscrever e cancelar a inscrição de um determinado tópico; `clienteConectado` que valida se o *framework* está conectado a um *broker* MQTT no momento e `desconectar`, que é utilizado para se desconectar do *broker* atual.

Para o desenvolvimento da interface Bluetooth foi utilizada a biblioteca BluetoothSerial do Cordova. Essa biblioteca fornece as funções básicas para a utilização do Bluetooth em sistemas Android, iOS e Windows Phone. A interface de comunicação Bluetooth foi nomeada FwBluetoothProvider e ela provem os métodos necessários para buscar dispositivos Bluetooth, conectar e enviar mensagens. O Quadro 3 apresenta a função `listarDispositivosPareados`.

Quadro 3 – Função para listar dispositivos pareados

```
listarDispositivosPareados(): Promise<Array<DispositivoBluetooth>> {
    return new Promise((resolve, reject) => {
      this.bluetoothSerial.list().then(
        (dispositivos) => {
          let dispositivosPareados: Array<DispositivoBluetooth> = new
Array<DispositivoBluetooth>();
          dispositivos.forEach(d => {
            dispositivosPareados.push(new DispositivoBluetooth(null,
d.name, null, null, null, d.address));
          });
          resolve(dispositivosPareados);
        }
      );
      (err) => {
        reject();
      };
    })
  }
```

Fonte: elaborado pelo autor.

Essa função tem o objetivo de buscar todos os dispositivos pareados com o dispositivo móvel e retornar uma `Promise` com os dispositivos encontrados. Uma observação é que esta função traz os dispositivos pareados e não os dispositivos conectados com o dispositivo móvel no momento. Os dispositivos encontrados são retornados como objetos `DispositivoBluetooth`. Assim como a função `listarDispositivosPareados`, existe uma função para listar os dispositivos não pareados, essa função é chamada de `listarDispositivosNaoPareados`. Ambas as funções possuem a implementação semelhante, sendo a única diferente que uma retorna os dispositivos pareados e a outra os dispositivos não pareados.

Além das funções de listagem, foram implementadas funções para ativar o Bluetooth do dispositivo móvel, conectar à um dispositivo Bluetooth, validar se o dispositivo móvel está com o Bluetooth conectado, enviar mensagem à um dispositivo Bluetooth. A função para enviar uma mensagem ao dispositivo Bluetooth conectado, se chama `enviarMensagem` e sua implementação é mostrada no Quadro 4.

Quadro 4 - Função que envia uma mensagem ao dispositivo Bluetooth conectado

```
enviarMensagem(mensagem: string, funcaoSucesso?: () => void,
funcaoErro?: () => void) {
    this.bluetoothSerial.write(mensagem.toString())
      .then((success) => funcaoSucesso == undefined ?
console.log('Enviou a mensagem com sucesso' + success) : funcaoSucesso)
      .catch((err) => funcaoErro == undefined ? console.log('Erro ao
enviar mensagem' + err) : funcaoErro);
  }
```

Fonte: elaborado pelo autor.

Esta função é utilizada para enviar uma mensagem ao dispositivo Bluetooth conectado no momento. Ou seja, antes de ser utilizada, o dispositivo móvel deve se conectar a um

dispositivo Bluetooth utilizando a função `conectarDispositivo`. A função `enviarMensagem` também recebe dois parâmetros opcionais chamados `funcaoSucesso` e `funcaoErro`, que são invocados quando a mensagem é enviada com sucesso ao dispositivo Bluetooth e quando ocorre algum tipo de erro ao enviar a mensagem ao dispositivo Bluetooth, respectivamente.

3.1.2.3 Dependências, instalação e utilização

O *framework* desenvolvido foi publicado no NPMJS que é um gerenciador de pacotes JavaScript, sendo que um pacote são vários arquivos JavaScript. O NPMJS possibilita a publicação desses pacotes em seu repositório gratuitamente, assim como possibilita aos usuários realizarem o *download* desses pacotes (NPM, 2018). No caso do *framework* desenvolvido, foi publicado um pacote com o nome de fwiotfurb. O guia para instalação do pacote, bem como a lista de suas dependências e detalhes sobre seu uso estão descritos no Apêndice A.

3.2 DESENVOLVIMENTO DO APLICATIVO

A seguir estão os requisitos funcionais e não funcionais do aplicativo desenvolvido:

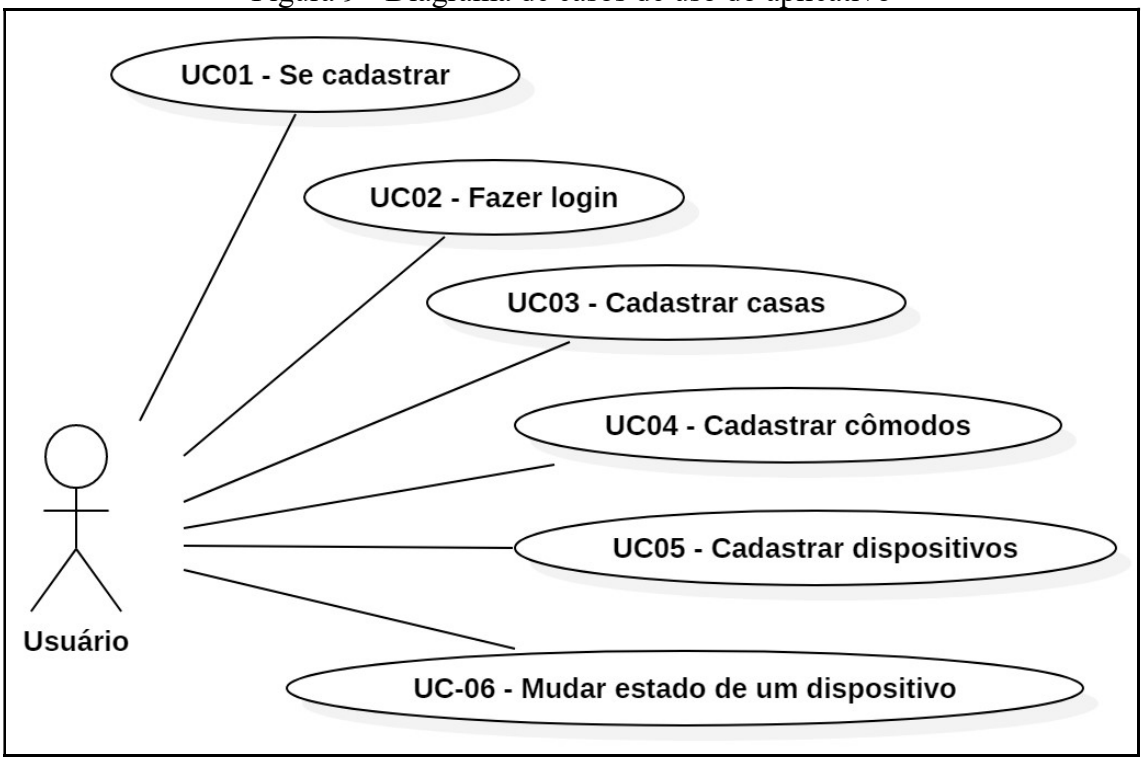
a) permitir o cadastro de diferentes casas (RF);
b) permitir o cadastro de diferentes cômodos (RF):
c) permitir o cadastro de diferentes dispositivos (RF);
d) permitir o controle dos dispositivos cadastrados (RF);
e) possuir autenticação com e-mail e senha (RF);
f) armazenar os dados do usuário no Firebase (RNF);
g) utilizar a linguagem de programação TypeScript (RNF);
h) ser implementado utilizando o Ionic Framework 3 (RNF).

3.2.1 Especificação

Esta seção apresenta a especificação do aplicativo desenvolvido. Assim como a especificação do *framework*, a especificação do aplicativo foi feita seguindo UML e utilizada a ferramenta StarUML para a elaboração dos diagramas.

A Figura 9 apresenta os casos de uso do aplicativo desenvolvido. Para todos os casos de uso existe apenas um ator, que é o usuário final do aplicativo. O `usuário` realiza o próprio cadastro, bem como o cadastro de todas as informações referentes à casa, aos cômodos e aos dispositivos.

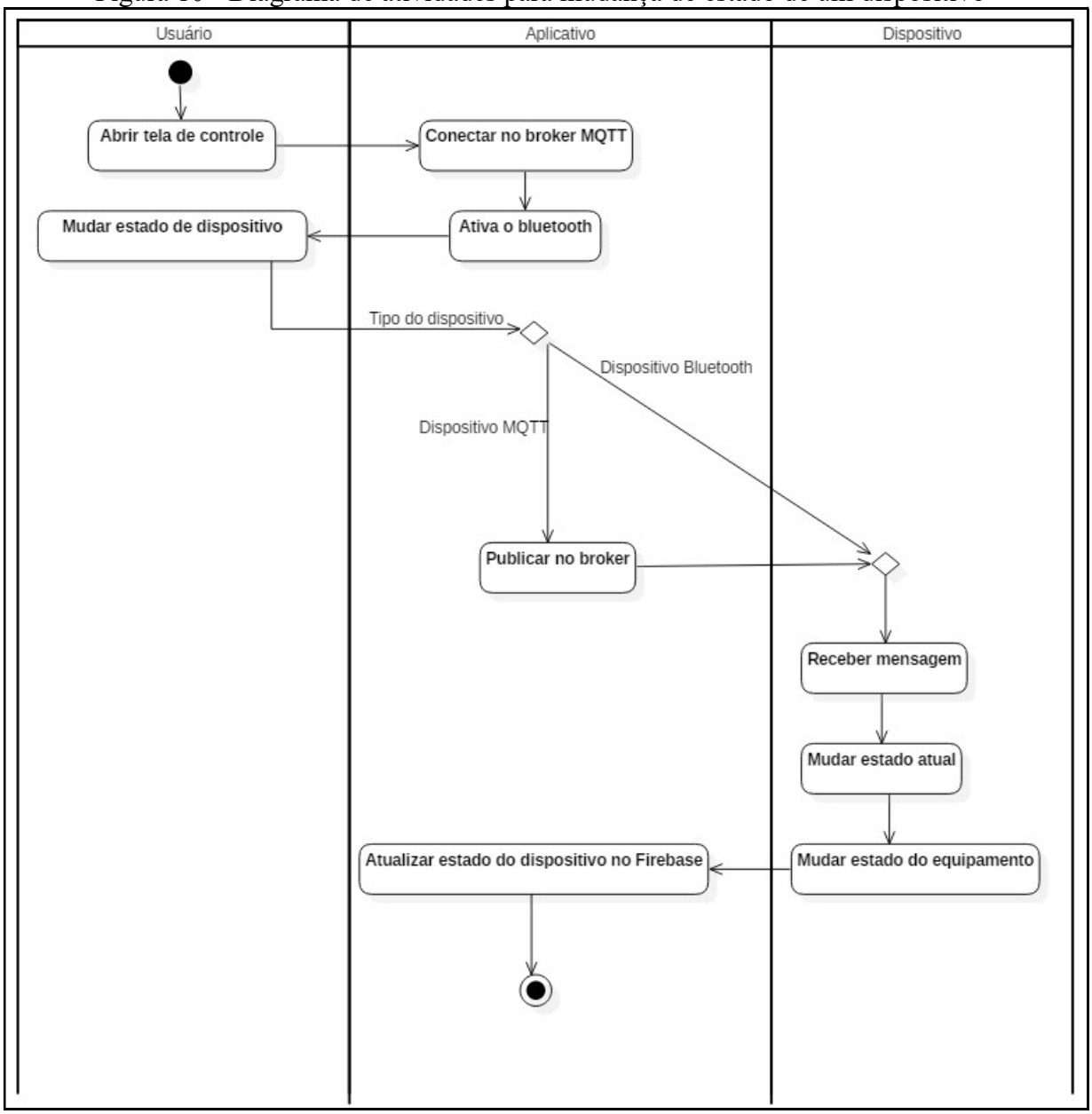
Figura 9 - Diagrama de casos de uso do aplicativo



Fonte: elaborado pelo autor

O principal caso de uso do aplicativo é o UC-06 – Mudar estado de um dispositivo, pois ele tem a responsabilidade de acionar os dispositivos cadastrados no aplicativo. A Figura 10 apresenta o diagrama de atividades para este caso de uso.

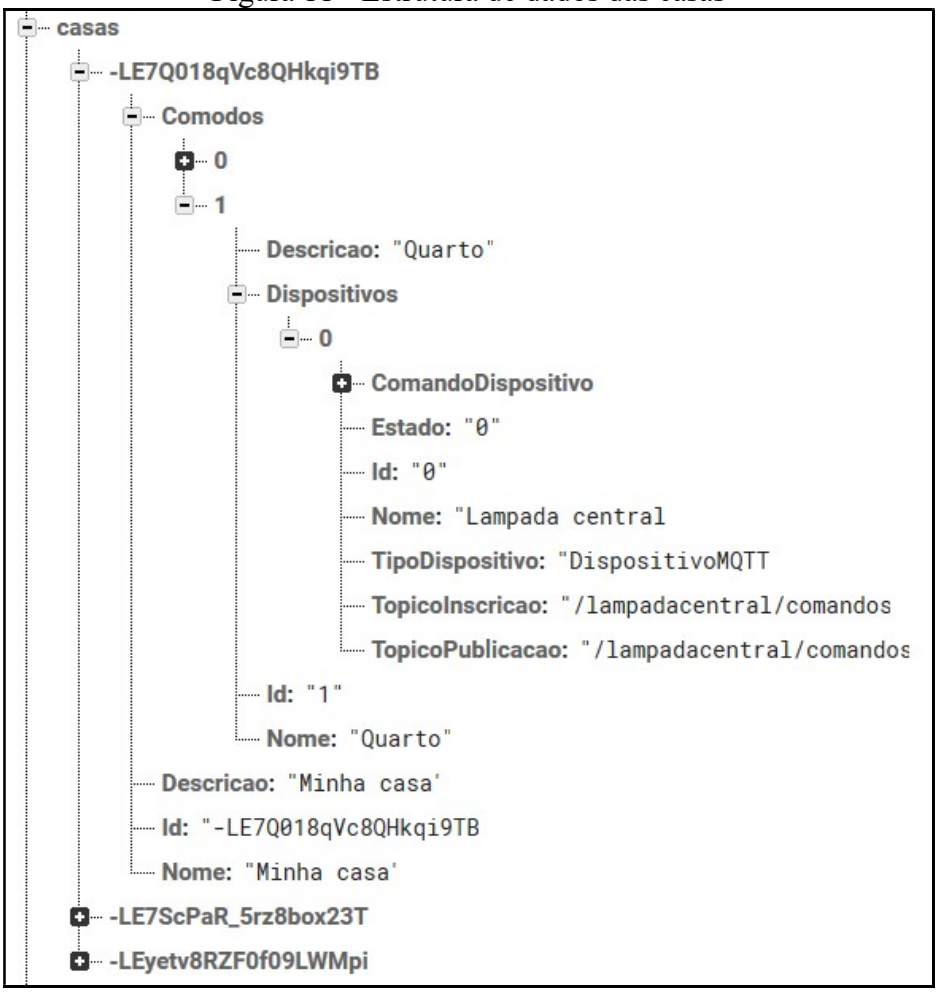
Figura 10 - Diagrama de atividades para mudança de estado de um dispositivo



Fonte: elaborado pelo autor.

Os dados dos usuários do aplicativo são armazenados no Firebase em formato JSON. A estrutura foi dividida em dois nós principais, `casas` e `usuarios`. Sendo que `casas` é uma lista de todas as casas cadastradas no aplicativo, independentemente do usuário, e `usuarios` são todos os usuários cadastrados no aplicativo. O nó casas corresponde a um `Array` de `casas`, este objeto `Casa` é o mesmo objeto fornecido pelo *framework* desenvolvido. A estrutura de dados do nó `casas` pode ser visualizado na Figura 11.

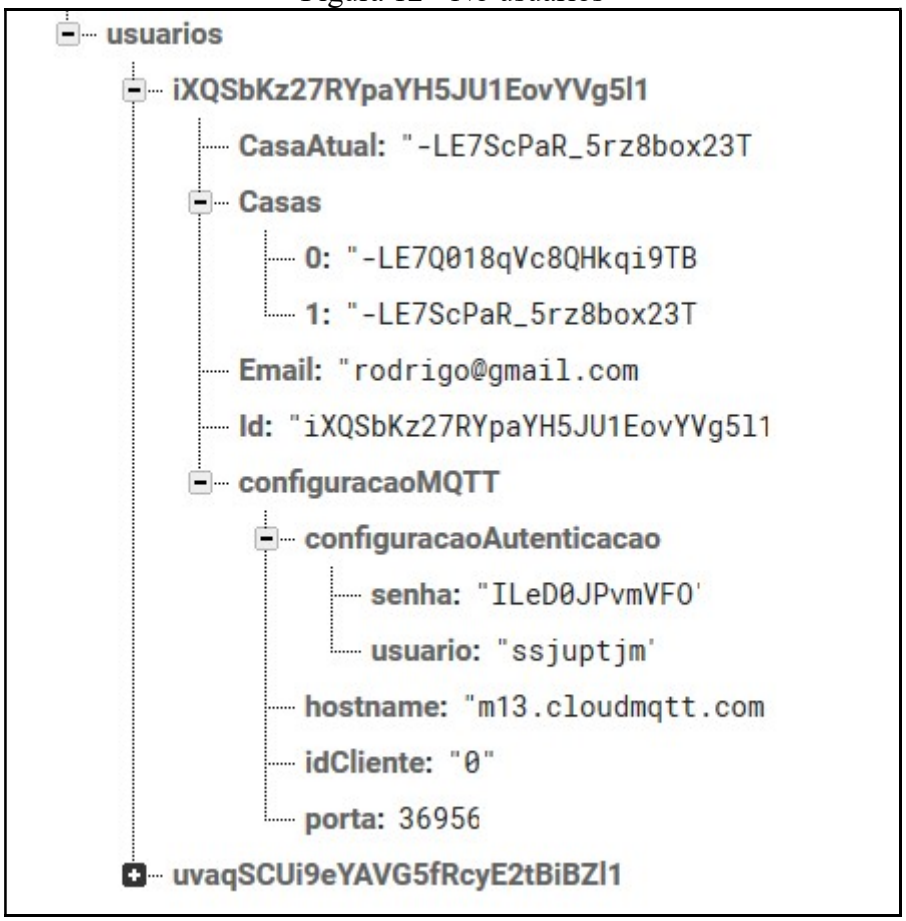
Figura 11 - Estrutura de dados das casas



Fonte: elaborado pelo autor.

O outro nó principal do aplicativo, `usuarios` contém uma lista de usuários cadastrados no aplicativo, diferente do nó `casas`, este não usa uma classe provinda do *framework*. Para isso, foi criado uma classe `usuario` que contém as informações relativas as suas credenciais, bem como referências para todas as suas casas, a sua configuração MQTT e uma referência para a casa atual do usuário. Essa estrutura pode ser visualizada na Figura 12.

Figura 12 - Nó usuarios



Fonte: elaborado pelo autor.

3.2.2 Implementação

A seguir são mostradas as técnicas e ferramentas utilizadas para a implementação do aplicativo, além da operacionalidade da implementação.

3.2.2.1 Técnicas e ferramentas utilizadas

O aplicativo foi desenvolvido utilizando o Ionic Framework. Além disso, do mesmo modo que o *framework*, o aplicativo foi desenvolvido utilizando o Visual Studio Code e a linguagem de programação TypeScript.

3.2.2.2 Implementação do aplicativo

O aplicativo desenvolvido possui autenticação com o Firebase, toda a sua implementação foi feita utilizando o pacote `AngularFireAuth`, sendo este um pacote que provém funções para realizar a autenticação do usuário e sua implementação foi feita na classe `AutenticacaoProvider`. Assim que é feito o login, o aplicativo valida se o usuário autenticado já está salvo na lista de usuários do Firebase através da função

`validarNovoUsuario`. Caso o usuário ainda não esteja salvo no Firebase, serão salvos os seus dados, além de ser cadastrada automaticamente uma `Casa` para este novo `Usuario`. A classe responsável por esta funcionalidade é a `UsuariosFirebaseProvider` e a função pode ser visualizada no Quadro 5.

Quadro 5 - Validação de um novo usuário

```
private validarNovoUsuario() {
this.auth.adicionarInscricao(this.Usuario.valueChanges().subscribe(usuario
=> {
      if (usuario == null) {
        this.adicionarNovoUsuario();
      } else {
        this.UsuarioObj = usuario;
      }
      this.CasaUsuarioAtual = this.db.object<Casa>("casas/" +
this.UsuarioObj.CasaAtual);
    }));
  }
```

Fonte: elaborado pelo autor.

A principal classe do aplicativo se chama `TelaControlePage`, pois é nela em que ocorre as principais atividades do aplicativo. Esta classe é responsável por fornecer uma tela de controle para o usuário, trazendo assim uma lista com todos os dispositivos cadastrados, além de um botão que possibilita ligar/desligar um determinado dispositivo. No `constructor` desta classe são chamadas duas funções de inicialização do aplicativo. A primeira função é a `inicializacaoMqtt` (Quadro 6), que é responsável por validar se o aplicativo já está conectado a um *broker* MQTT, caso o aplicativo ainda não esteja conectado ao *broker* MQTT, ele faz essa conexão.

Quadro 6 - Inicialização do MQTT

```
private inicializacaoMqtt() {
    this.auth.adicionarInscricao(this.configMQTT.ObterConfiguracao()
.subscribe(configuracao => {
      this.configuracaoMQTT = configuracao;
      if (this.configuracaoMQTT != null) {
        this.loading = this.loadingCtrl.create({
          content: 'Conectando ao MQTT'
        });
        if (!this.fwMqtt.clienteConectado()) {
          this.loading.present();
          this.fwMqtt.configurarMQTT(this.configuracaoMQTT);
          this.timeoutConexaoMQTT();
        }
      }
    }));
  }
```

Fonte: elaborado pelo autor.

A segunda função de inicialização do aplicativo é a `inicializacaoCasa` (Quadro 7), que tem a responsabilidade de buscar a casa atual do usuário no Firebase e criar uma lista com os estados de cada um dos dispositivos desta `Casa`.

Quadro 7 - Inicialização da casa

```
private inicializacaoCasa() {
    this.auth.adicionarInscricao(this.dbUsuarios.obterCasaAtual()
.subscribe(casa => {
      this.CasaAtual = casa;
      this.ListaComodos = new Array<Comodo>();
      this.casaEstados = new Array<Array<boolean>>();
      if (casa != null && casa.Comodos != null) {
        this.atualizarListaEstadosCasa();
      }
      this.loadingConfig.dismiss();
    }));
  }
```

Fonte: elaborado pelo autor.

Depois que feita a inicialização `TelaControlePage` é possível realizar a mudança de estado dos dispositivos. Essa mudança de estados é feita através da função `mudarEstado`, que recebe como parâmetro o `Dispositivo` que irá mudar de estado e o `Comodo` em que ele está inserido. Esta função irá validar o tipo do `Dispositivo` e chamar a função de mudança de estado correspondente ao tipo do dispositivo (Quadro 8).

Quadro 8 - Mudança de estado

```
mudarEstado(dispositivo: DispositivoMQTT | DispositivoBluetooth, comodo:
Comodo) {
    let estadoCasaEstados =
this.obterEstadoDispositivoCasaEstados(dispositivo, comodo);
    let comandoDisp = dispositivo.ComandoDispositivo as ComandoONOFF;
    if (estadoCasaEstados && dispositivo.Estado == comandoDisp.OFF ||
!estadoCasaEstados && dispositivo.Estado == comandoDisp.ON) {
      if (dispositivo.TipoDispositivo == "DispositivoMQTT") {
        this.mudarEstadoDispositivoMqtt(dispositivo as DispositivoMQTT,
comandoDisp);
      } else if (dispositivo.TipoDispositivo == "DispositivoBluetooth") {
        this.mudarEstadoDispositivoBluetooth(dispositivo as
DispositivoBluetooth, comandoDisp);
      }
    }
  }
```

Fonte: elaborado pelo autor.

Para dispositivos MQTT é chamada a função `mudarEstadoDispositivoMqtt`, que irá receber um `DispositivoMQTT` e um `ComandoONOFF`. Esta função publica no tópico configurado no `DispositivoMQTT` o `ComandoONOFF`, atualiza a referência estado do dispositivo e por fim atualiza o estado do dispositivo no Firebase. Esta função é apresentada no Quadro 9.

Quadro 9 - Mudanca de estado de dispositivo MQTT

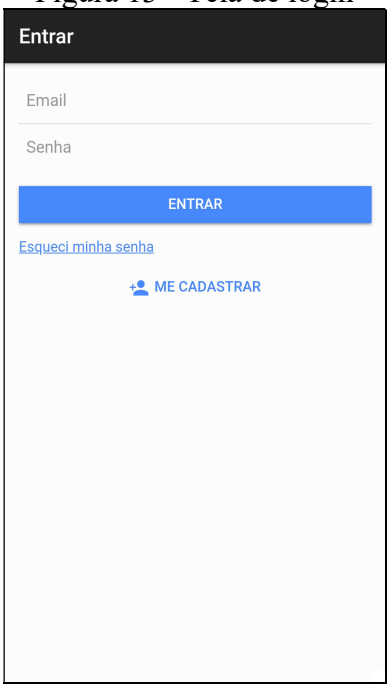
```
mudarEstadoDispositivoMqtt(dispositivoMqtt: DispositivoMQTT, comandoONOFF:
ComandoONOFF) {
    if (dispositivoMqtt.Estado == comandoONOFF.OFF) {
      this.fwMqtt.publicar(comandoONOFF.ON,
dispositivoMqtt.TopicoPublicacao);
      dispositivoMqtt.Estado = comandoONOFF.ON;
    } else {
      this.fwMqtt.publicar(comandoONOFF.OFF,
dispositivoMqtt.TopicoPublicacao);
      dispositivoMqtt.Estado = comandoONOFF.OFF;
    }
    this.casaDb.atualizarDadosCasa(this.CasaAtual);
  }
```

Fonte: elaborado pelo autor.

3.2.2.3 Operacionalidade da implementação

Nesta seção será apresentada a operacionalidade da implementação do aplicativo, demonstrando as suas principais telas e funções. A Figura 13 apresenta a tela de *login* do aplicativo, nela o usuário pode realizar o *login* através de um e-mail e senha já cadastrados. Caso o usuário não esteja cadastrado, ele pode se cadastrar através do botão `ME CADASTRAR`.

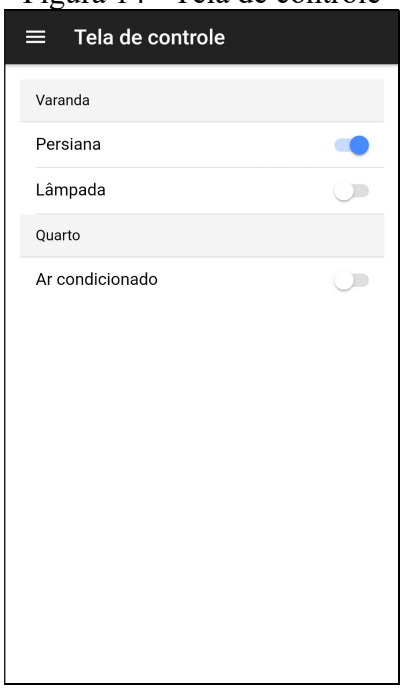
Figura 13 - Tela de login



Fonte: elaborado pelo autor.

Depois que realizado o login, é apresentada a `Tela de Controle` (Figura 14). Nessa tela são exibidos todos os dispositivos já cadastrados para este usuário, bem como os respectivos cômodos dos dispositivos.

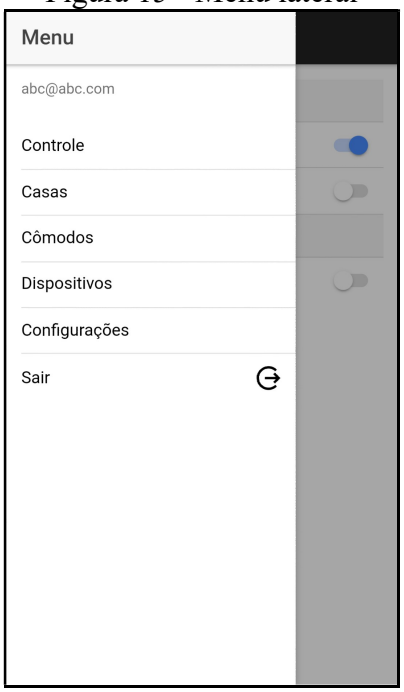
Figura 14 - Tela de controle



Fonte: elaborado pelo autor.

O usuário pode realizar a navegação entre as diferentes telas através do botão superior esquerdo, que irá apresentar um menu de navegação. Este menu de navegação possui atalhos para a tela de `controle`, `casas`, `cômodos`, `dispositivos` e `configurações`, além de um botão para `sair` do aplicativo (Figura 15).

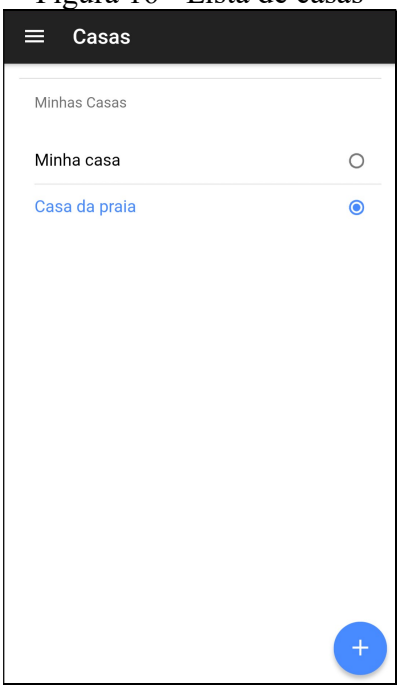
Figura 15 - Menu lateral



Fonte: elaborado pelo autor.

A Figura 16 apresenta a tela `Casas`, que corresponde a uma lista de todas as casas cadastradas para este usuário, além de mostrar a casa que o usuário está no momento. Nesta tela o usuário pode realizar o cadastro de novas casas, além de selecionar outra casa atual.

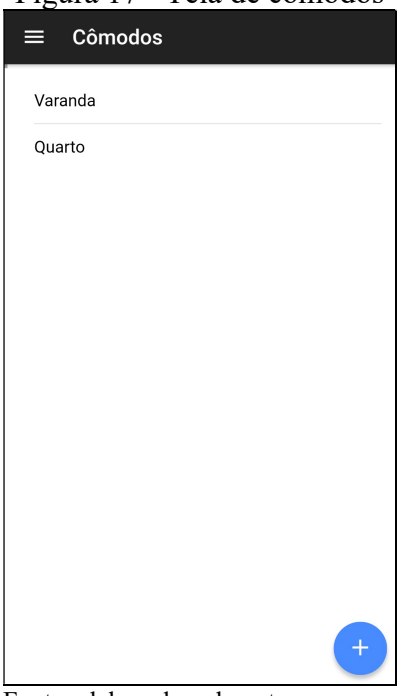
Figura 16 - Lista de casas



Fonte: elaborado pelo autor.

Além da lista de `Casas`, o usuário também consegue visualizar uma lista de cômodos da casa atual, através da tela de `Comodos` (Figura 17). Além da lista de cômodos da casa atual, esta tela também fornece um botão para cadastro de novos cômodos.

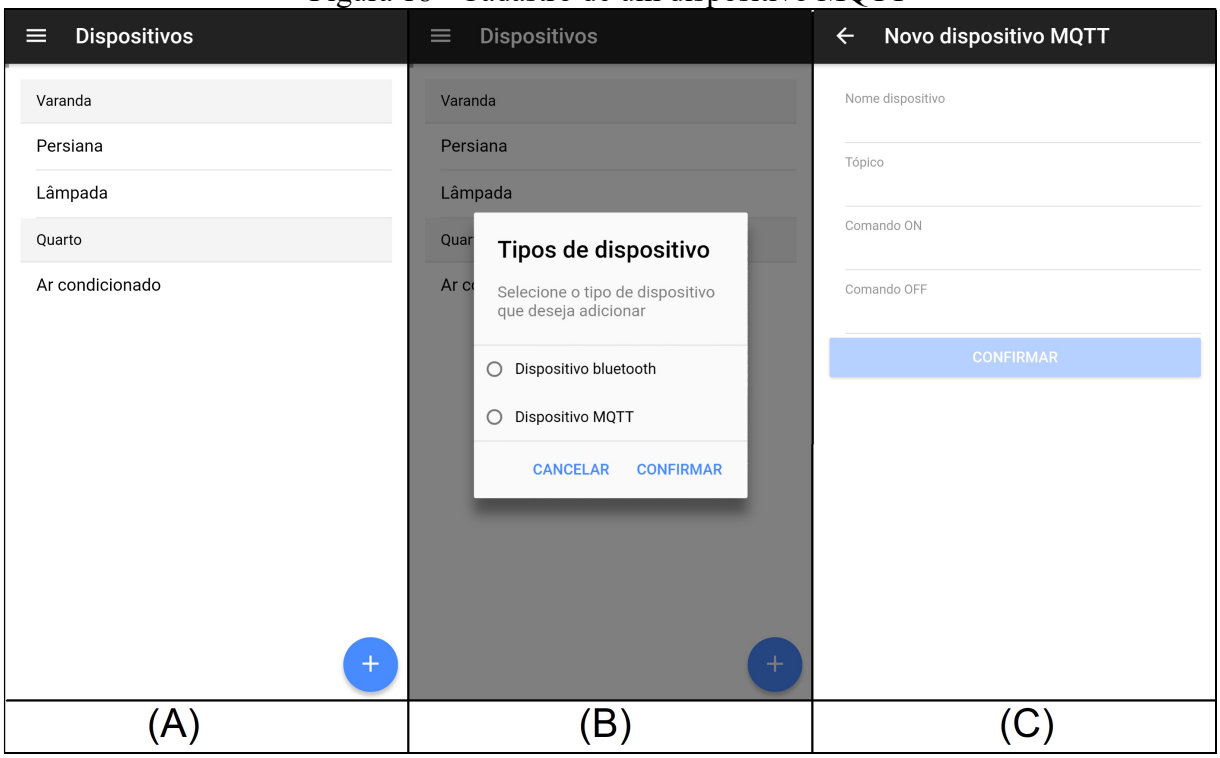
Figura 17 - Tela de cômodos



Fonte: elaborado pelo autor.

Para o cadastro de dispositivos, o usuário pode acessar a tela `Dispositivos`. Nela serão exibidos todos os dispositivos e seus respectivos cômodos (Figura 18(A)). Ao clicar no botão inferior direito, será aberto um *prompt* para o usuário selecionar o tipo de dispositivos que deseja cadastrar (Figura 18(B)). Depois de selecionado o tipo do dispositivo, o usuário será redirecionado para a tela especifica de cadastro daquele tipo de dispositivos. Por exemplo, caso selecionado o tipo Dispositivo MQTT, o usuário será redirecionado para a tela `Novo dispositivo MQTT` (Figura 18(C)). Depois que confirmado o cadastro do dispositivo, o usuário é redirecionado para a tela com a lista dos dispositivos.

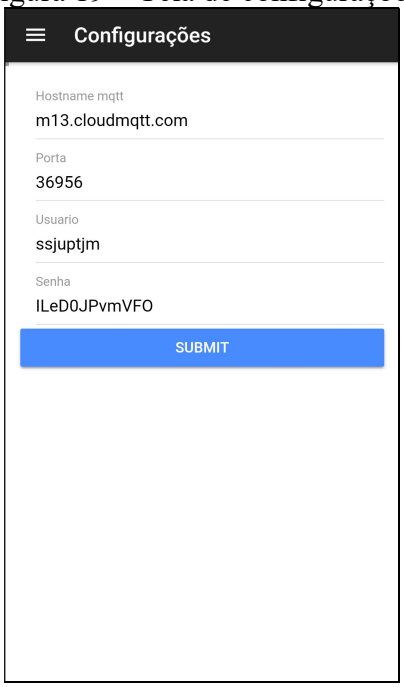
Figura 18 - Cadastro de um dispositivo MQTT



Fonte: elaborado pelo autor.

Por fim, também é fornecida uma tela para o usuário realizar as configurações do *broker* MQTT (Figura 19). Nela são preenchidos os dados básicos para o funcionamento de um dispositivo MQTT.

Figura 19 – Tela de configurações



Fonte: elaborado pelo autor.

3.3 ANÁLISE DOS RESULTADOS

Nesta seção são apresentados os testes de funcionalidade realizados com o *framework*, testes e uma avaliação de usabilidade do *framework* e uma comparação entre o trabalho desenvolvido e os trabalhos correlatos.

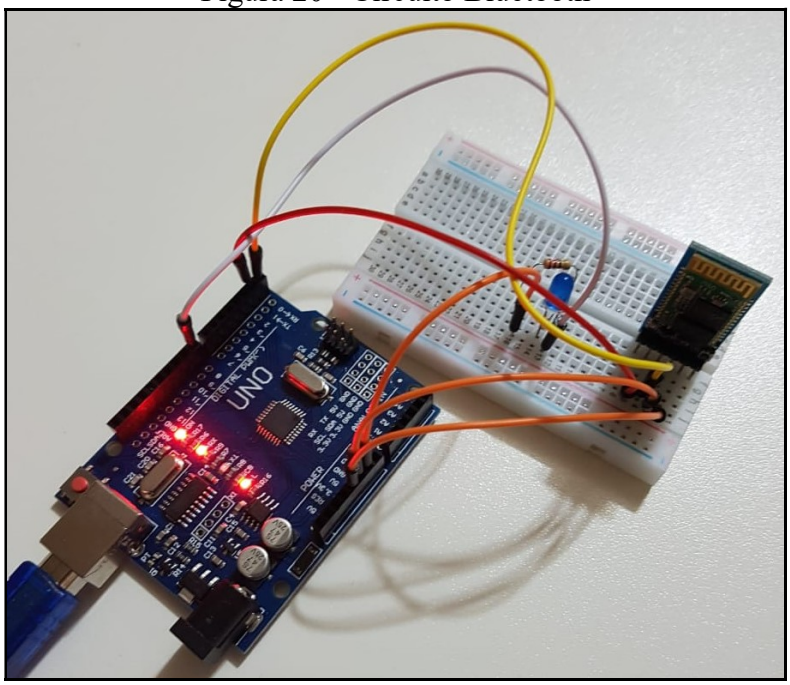
3.3.1 Testes de funcionalidade

Os testes de funcionalidade do *framework* foram divididos em duas partes: primeiro foram realizados os testes da interface Bluetooth e depois foram realizados os testes da interface MQTT.

3.3.1.1 Testes da interface Bluetooth

Para a realização dos testes com a interface Bluetooth, foi necessário elaborar um circuito contendo um módulo de comunicação Bluetooth, pois o *plugin* utilizado para a comunicação via Bluetooth só realiza a comunicação entre dispositivos Android/iOS e Arduino, ou seja, não é possível realizar a troca de mensagens entre dois dispositivos Android ou iOS. O circuito desenvolvido (Figura 20) é composto por um módulo Bluetooth RS232 HC-05, um módulo Arduino, um resistor e um LED azul. O módulo HC-05 é um módulo Bluetooth classe 2, que possibilita o envio de mensagens através de uma interface serial. Além disso, foi desenvolvido um pequeno *software* que liga o módulo Bluetooth e sempre que recebe uma mensagem, dependendo de seu valor, liga ou desliga o LED.

Figura 20 - Circuito Bluetooth



Fonte: elaborado pelo autor.

Os testes realizados com o Bluetooth consistiram em ativar o Bluetooth do dispositivo móvel, listar e conectar a um dispositivo não pareado, enviar mensagens ao dispositivo conectado e por fim listar os dispositivos pareados. O *framework* desenvolvido conseguiu realizar todas essas funcionalidades utilizando um dispositivo móvel com sistema operacional Android. Foi possível ligar o Bluetooth, listar os dispositivos não pareados, conectar ao módulo Bluetooth e enviar mensagens que acendiam e desligavam o LED. Contudo, houveram problemas utilizando um dispositivo móvel iOS, pois o dispositivo móvel não conseguia encontrar o módulo Bluetooth, por conta dos dispositivos iOS não conseguirem comunicar com dispositivos Bluetooth classe 2. Além disso, foi observado que quando realizado o envio de muitas mensagens seguidas para o módulo Bluetooth ele parava de responder.

3.3.1.2 Testes da interface MQTT

Os testes da interface MQTT foram feitos utilizando um dispositivo Sonoff Basic. Primeiramente foi necessário alterar o *firmware* do dispositivo, para ser utilizado um *firmware* customizado e que seja possível configurar o *broker* MQTT a ser utilizado. Primeiramente foi feito o *download* do *firmware* customizado Sonoff-Tasmota (ARENDS, 2018). Depois disso, foram feitas alterações neste *firmware* para o Sonoff se conectar em uma rede específica e utilizar um *broker* MQTT específico. Em seguida foi feito o *upload* deste *firmware* utilizando um conversor USB Serial TTL PL2303hx para fazer a conexão entre o

Sonoff e o computador. Por fim, foi conectada uma lâmpada ao Sonoff e ligado à tomada (Figura 21).

Figura 21 - Circuito com Sonoff

Fonte: elaborado pelo autor.

Depois deste procedimento foram iniciados os testes com a interface MQTT. Os testes foram realizados utilizando um *smartphone* com sistema operacional Android e consistiram em conectar a um *broker* MQTT e enviar mensagens para o tópico em que o Sonoff estava inscrito. Todas essas funcionalidades funcionaram corretamente, sendo possível enviar mensagens para o Sonoff independente da rede Wi-Fi em que o dispositivo móvel estava conectado. Vale observar que, caso o *broker* MQTT esteja hospedado em algum site na Web, o tempo de resposta do *broker* pode influenciar no acionamento do Sonoff. Nos testes realizados, o envio de mensagens para o *broker* e a ativação do Sonoff aconteciam em um pequeno intervalo de tempo.

3.3.2 Testes de usabilidade

A fim de testar a usabilidade do *framework* desenvolvido, foi solicitado para três acadêmicos de ciência da computação desenvolverem uma simples aplicação utilizando o *framework*. A sua avaliação foi dividida em seis etapas, sendo elas a avaliação dos conhecimentos dos acadêmicos, avaliação da instalação do *framework*, avaliação da interface MQTT, avaliação da interface Bluetooth, avaliação das estruturas providas pelo *framework* e a avaliação geral do *framework*.

3.3.2.1 Avaliação dos conhecimentos dos acadêmicos

Inicialmente foram avaliados os conhecimentos dos acadêmicos em relação as tecnologias necessárias para o desenvolvimento de aplicações utilizando o *framework*. Os resultados obtidos a partir dessa avaliação são apresentados na Tabela 1.

Tabela 1 - Avaliação do conhecimento das tecnologias

Como você avalia o seu conhecimento em relação ao Ionic Framework?	0% Não conheço o Ionic Framework. 0% Conheço, mas nunca implementei nada. 100% Conheço e já desenvolvi pequenas aplicações. 0% Conheço bem, já desenvolvi aplicações complexas.
Como você avalia o seu conhecimento em relação ao MQTT?	0% Não conheço o MQTT. 0% Conheço o conceito, mas nunca utilizei. 100% Conheço e já utilizei.
Como você avalia o seu conhecimento em relação ao plugin BluetoothSerial do Ionic?	0% Não conheço. 66.7% Conheço, mas nunca utilizei. 33.3% Conheço e já utilizei.
Como você avalia o seu conhecimento em relação à aplicações para IoT?	0% Nunca desenvolvi nada para IoT. 66.7% Já desenvolvi pequenas aplicações. 33.3% Já desenvolvi aplicações complexas.

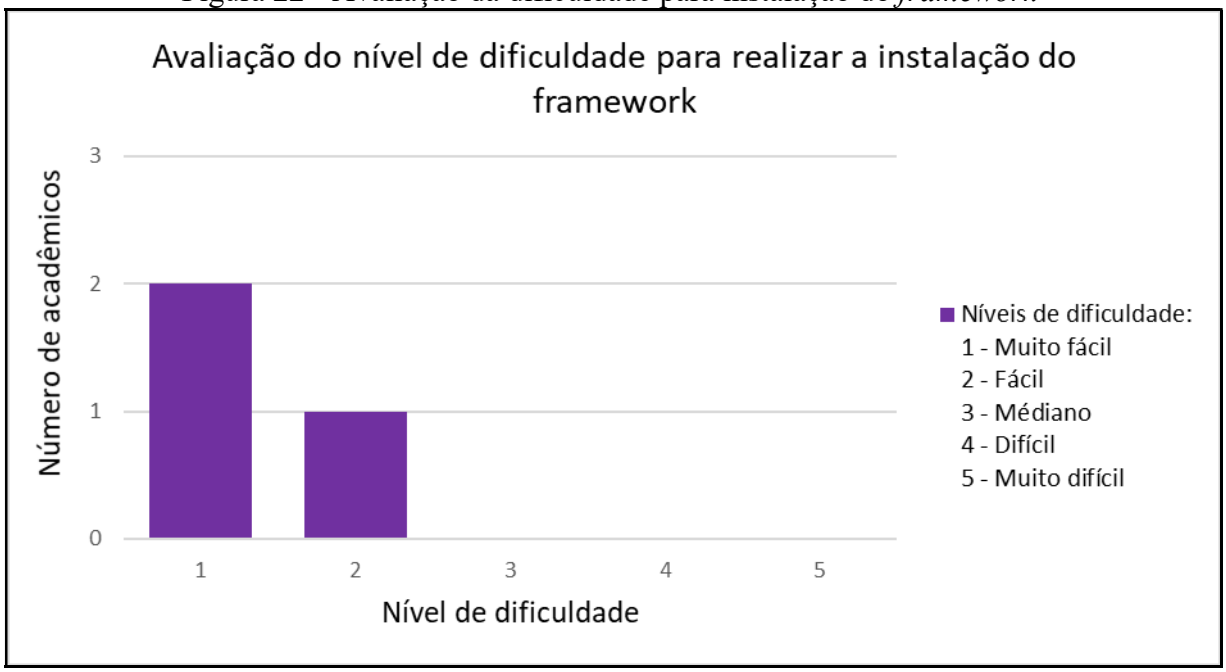
Fonte: elaborado pelo autor.

A partir dos dados apresentados, pode-se observar que todos os acadêmicos possuem os conhecimentos básicos necessários para a utilização do *framework*. É necessária esta avaliação pois caso o acadêmico não possuísse os conhecimentos básicos necessários, isto iria gerar impactos negativos no momento da utilização do *framework*.

3.3.2.2 Avaliação da instalação do framework

A Figura 22 apresenta os resultados obtidos em relação ao nível de dificuldade para a instalação do *framework*. A partir dos resultados apresentados na Figura 22, é possível constatar que não houveram dificuldades para a instalação do *framework*, podendo assim se afirmar que o *framework* desenvolvido possui uma instalação fácil e prática. Deste modo, desenvolvedores que utilizarem o *framework* possivelmente não enfrentarão problemas ao tentar instalar o *framework*.

Figura 22 - Avaliação da dificuldade para instalação do *framework*



Fonte: elaborado pelo autor.

3.3.2.3 Avaliação da interface MQTT

A Tabela 2 apresenta a avaliação da utilização da interface de comunicação MQTT. A partir de seus dados, é possível afirmar que todos os acadêmicos conseguiram utilizar a interface MQTT e a mesma funcionou corretamente.

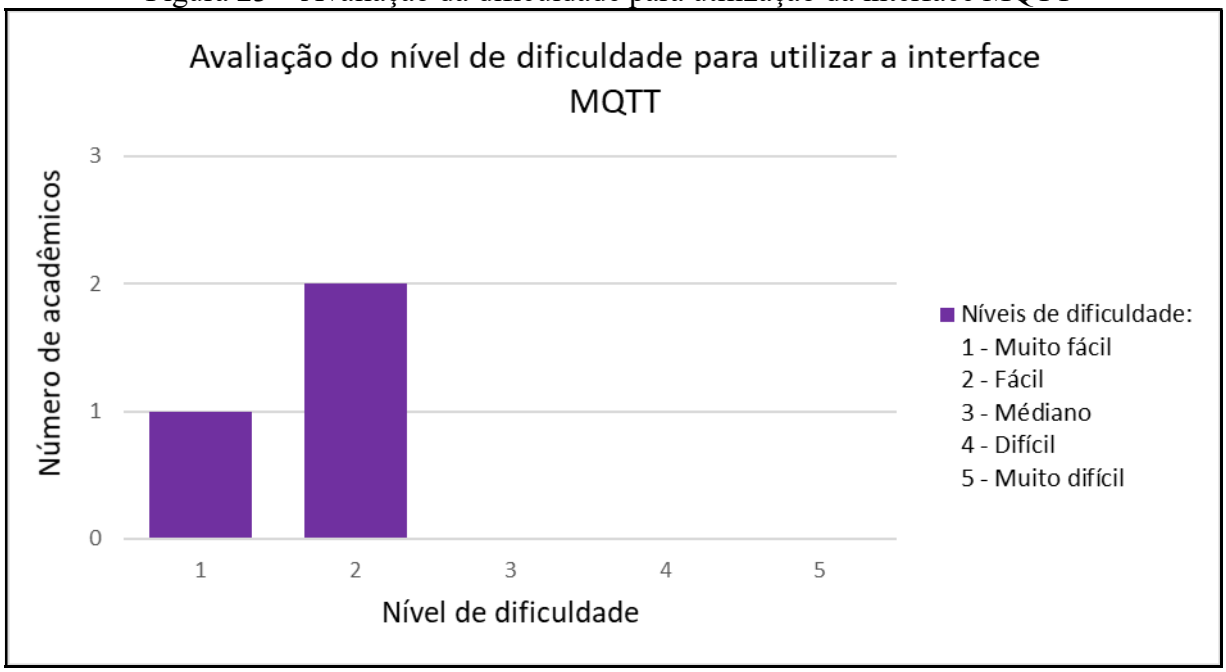
Tabela 2 - Avaliação da utilização interface de comunicação MQTT

Você conseguiu utilizar a interface de comunicação MQTT? (FwMqttProvider)?	100% Sim. 0% Não.
A interface de comunicação MQTT funcionou corretamente? Foi possível enviar comandos para o broker MQTT?	100% Sim. 0% Não.

Fonte: elaborado pelo autor.

A seguir, a Figura 23 apresenta as o nível de dificuldade para a utilização da interface MQTT. A partir dos dados apresentados pela figura, é possível afirmar que que todos os acadêmicos conseguiram utilizar a interface MQTT sem problemas, sendo que sua utilização foi considerada pelo menos fácil por todos os acadêmicos.

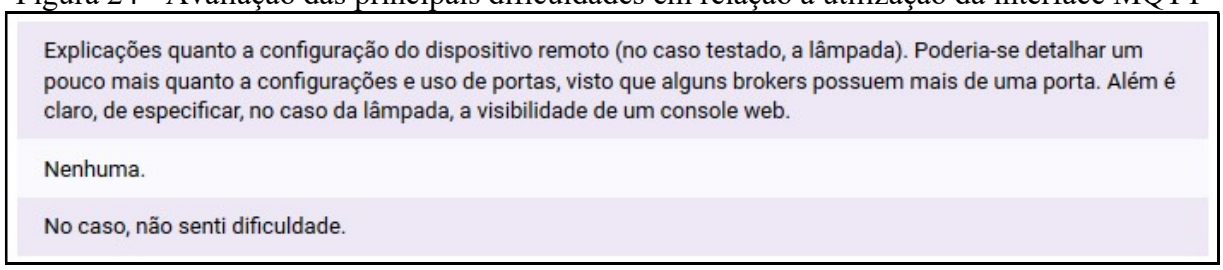
Figura 23 – Avaliação da dificuldade para utilização da interface MQTT



Fonte: elaborado pelo autor.

A Figura 24 apresenta a avaliação das principais dificuldades em relação a utilização da interface MQTT. A partir dela, é possível observar que apenas um acadêmico levantou uma dificuldade, que seria em relação a documentação do funcionamento do *broker* MQTT. Sendo assim, é necessário elaborar uma documentação mais detalhada em relação ao funcionamento do MQTT.

Figura 24 - Avaliação das principais dificuldades em relação a utilização da interface MQTT



Fonte: elaborado pelo autor.

3.3.2.4 Avaliação da interface Bluetooth

A Tabela 3 apresenta a avaliação do uso da interface Bluetooth.

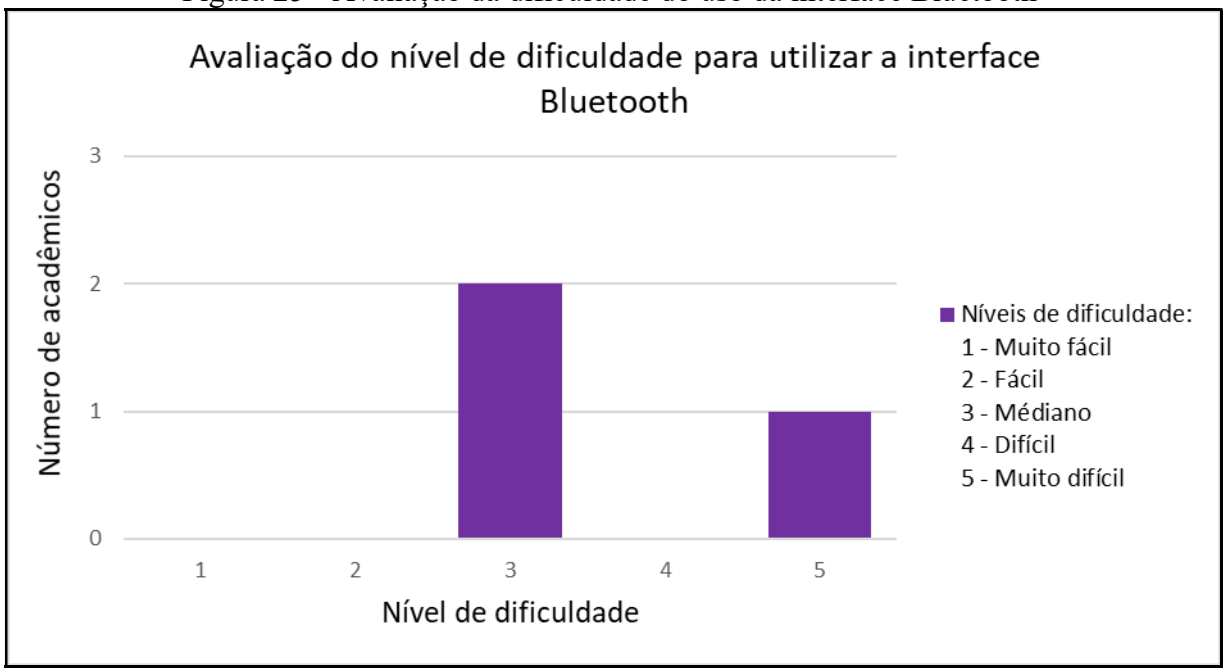
Tabela 3 - Avaliação do uso da interface Bluetooth

Você conseguiu utilizar a interface de comunicação Bluetooth?	0% Sim. 33.3% Não. 33.3% Não utilizei. 33.3% Problemas com hardware, o smarthphone é antigo, ao ligar o bluetooth, o celular reiniciava.
A interface de comunicação funcionou corretamente?	0% Sim. 33.3% Não. 33.3% Não utilizei. 33.3% Problemas com hardware, o smarthphone é antigo, ao ligar o bluetooth, o celular reiniciava.
Foi possível listar os dispositivos bluetooth?	0% Sim. 33.3% Não. 33.3% Não utilizei a listagem de dispositivos. 33.3% Problemas com hardware, o smarthphone é antigo, ao ligar o bluetooth, o celular reiniciava.
Você conseguiu enviar comandos para algum dispositivo bluetooth?	0% Sim. 33.3% Não. 33.3% Não utilizei. 33.3% Problemas com hardware, o smarthphone é antigo, ao ligar o bluetooth, o celular reiniciava.

Fonte: elaborado pelo autor.

A partir dos dados da tabela, é possível observar que nenhum dos acadêmicos conseguiu utilizar a interface Bluetooth corretamente, sendo que o primeiro acadêmico não utilizou a interface Bluetooth, o segundo acadêmico não conseguiu utilizar e o terceiro acadêmico teve problemas em relação ao *hardware* utilizado. Deste modo, é possível observar que a interface apresentou problemas quanto à sua utilização. Um dos problemas levantados pelo terceiro acadêmico foi a incompatibilidade com um *hardware* mais antigo, para isso, seria necessário avaliar a utilização de um outro *plugin* que fosse compatível com dispositivos móveis mais antigos. Já o segundo acadêmico, informou que o Ionic apresentou mensagens como “Não é possível se conectar ao dispositivo” e que o *plugin* Bluetooth não estava instalado corretamente, esses problemas podem acontecer quando executado o *framework* em um dispositivo sem Bluetooth, como por exemplo executar localmente no computador de desenvolvimento. Para resolver este problema, seria necessário realizar um tratamento do dispositivo que está utilizando o *framework*, a fim de prevenir a ativação do Bluetooth caso o dispositivo não possua esse tipo de comunicação. A Figura 25 apresenta a avaliação da dificuldade para utilização da interface Bluetooth. Nela é possível observar que todos os acadêmicos tiveram um certo nível de dificuldade para a utilização da interface, sendo que o segundo acadêmico considerou a utilização como muito difícil. Isso mostra que a interface Bluetooth possui uma utilização mais complicada, trazendo a necessidade de uma melhor documentação ou até mesmo algum tutorial explicando detalhadamente a sua utilização.

Figura 25 - Avaliação da dificuldade do uso da interface Bluetooth

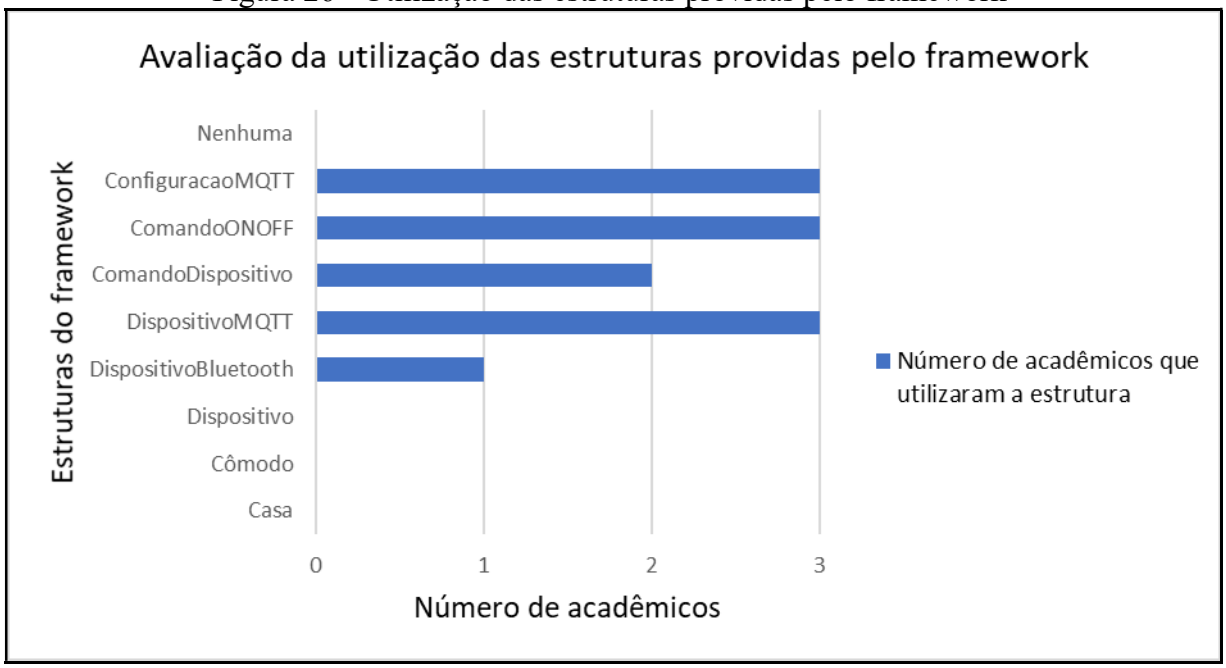


Fonte: elaborado pelo autor.

3.3.2.5 Avaliação das estruturas providas pelo *framework*

Essa avaliação teve o objetivo de levantar principalmente a utilidade das estruturas providas pelo *framework*, como estruturas de `Dispositivos`, `Casa`, `Cômodos`, etc. A Figura 26 apresenta quais foram as estruturas providas pelo *framework* que foram utilizadas pelos acadêmicos. A partir dela, pode-se observar que os acadêmicos utilizarem boa parte das estruturas providas pelo *framework*, em especial as que eram necessárias para utilização das interfaces Bluetooth e MQTT.

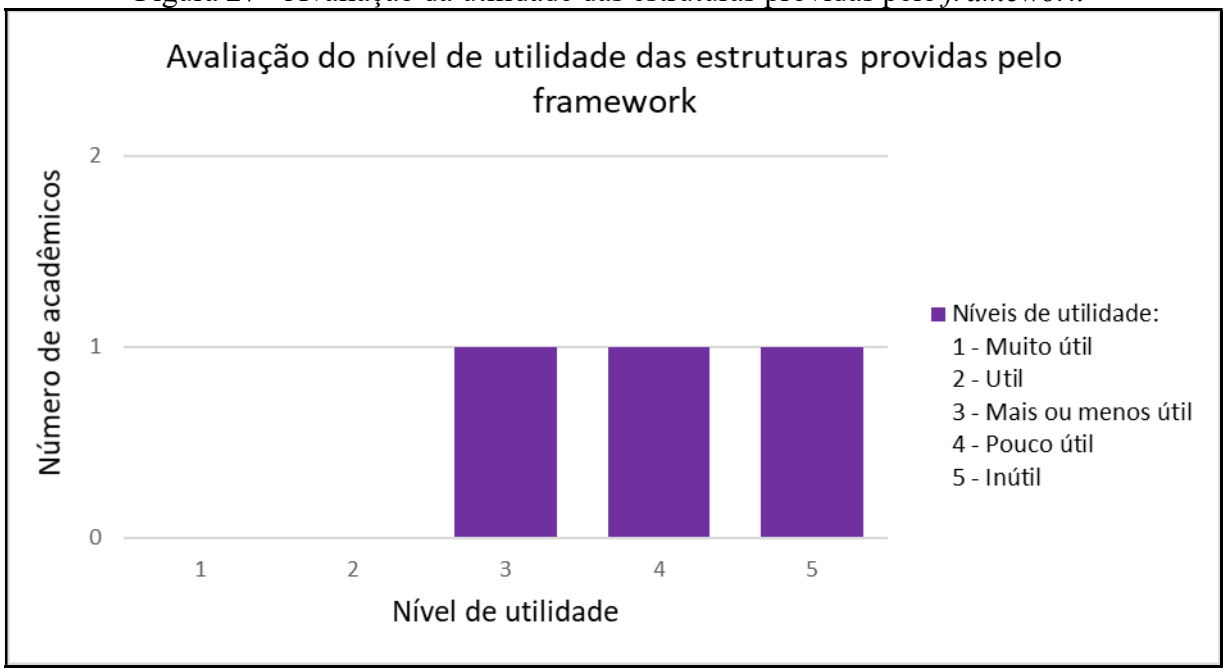
Figura 26 - Utilização das estruturas providas pelo framework



Fonte: elaborado pelo autor.

A seguir, a Figura 27 apresenta a avaliação da utilidade das estruturas providas pelo *framework*. A partir dela, é possível observar que todos os acadêmicos considerarem pelo menos as estruturas como uteis, sendo que um deles considerou muito útil. Essas avaliações são importantes para o *framework*, pois essas estruturas tem o objetivo de facilitar e diminuir a necessidade de criação de estruturas adicionais para a utilização do *framework*. Além disso, por conta da boa avaliação obtida em relação a utilidade dessas estruturas, deve ser avaliada a possibilidade de implementação de novas estruturas.

Figura 27 - Avaliação da utilidade das estruturas providas pelo *framework*

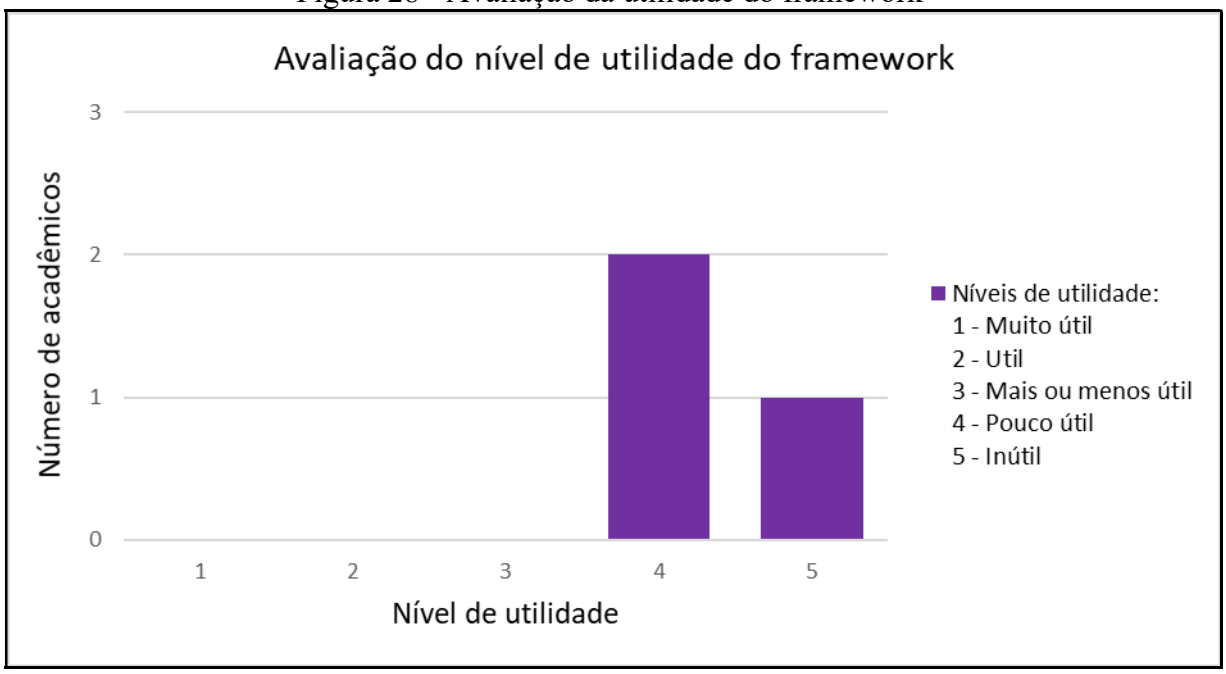


Fonte: elaborado pelo autor.

3.3.2.6 Avaliação geral do *framework*

Por fim, foi realizada a avaliação geral do *framework*, que teve o objetivo de avaliar a utilidade do *framework* de modo geral, bem como avaliar a possibilidade de os acadêmicos utilizarem este *framework* no desenvolvimento de uma aplicação futura. A Figura 28 apresenta a avaliação da utilidade do *framework*, a partir dela é possível afirmar que o *framework* foi considerado bastante útil para os acadêmicos, confirmando assim a sua usabilidade para o desenvolvimento de aplicações para IoT.

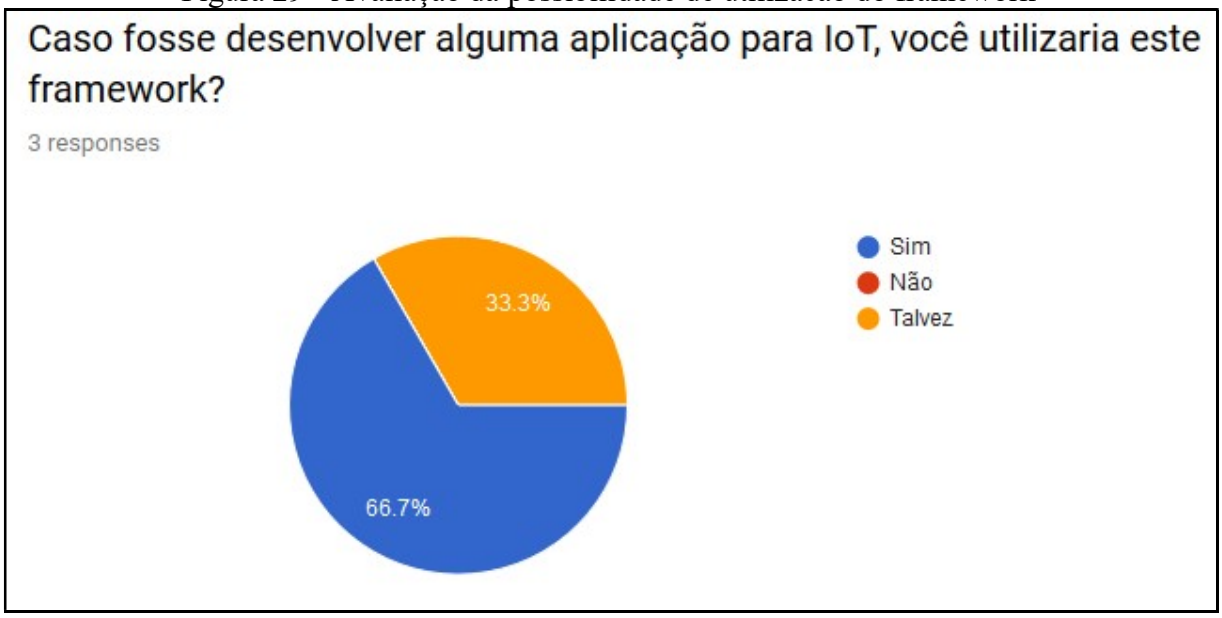
Figura 28 - Avaliação da utilidade do framework



Fonte: elaborado pelo autor.

A Figura 29, que apresenta a possibilidade do acadêmico utilizar este framework para o desenvolvimento de aplicações IoT, que reforça a afirmação da utilidade e usabilidade o *framework* desenvolvido.

Figura 29 - Avaliação da possibilidade de utilizacao do framework



Fonte: elaborado pelo autor.

3.3.3 Comparação com os trabalhos correlatos

O Quadro 10 apresenta as principais características dos trabalhos correlatos e do trabalho desenvolvido. A partir deste quadro, é possível observar que todos os trabalhos apresentados consegues controlar diversos equipamentos, com as funcionalidades de ligar e desligar. Porém o trabalho desenvolvido por Ciocari (2013), é o único que não consegue controlar funções especificas dos equipamentos. Vale observar que o trabalho desenvolvido consegue, através das interfaces MQTT e Bluetooth, controlar diferentes tipos de funções dos equipamentos, por conta da mensagem a ser enviada poder ser customizada. Além disso, vale destacar que apenas o trabalho desenvolvido é multiplataforma, podendo ser utilizado tanto em iOS quanto Android.

Quadro 10 - Comparação entre os trabalhos correlatos

Correlatos/Caracteríticas	Ciocari (2013)	Prado (2012)	Moribe (2013)	Trabalho desenvolvido
Interface móvel	Sim	Não	Sim	Sim
Multiplataforma	Não	Não	Não	Sim
Controla (liga/desliga) diversos equipamentos	Sim	Sim	Sim	Sim
Controle funções especificas dos equipamentos	Não	Sim	Sim	Sim
Tipo de comunicação utilizada	Bluetooth	Wi-Fi	Wi-Fi	Bluetooth, Wi-Fi (MQTT)

Fonte: elaborado pelo autor.

Todos os trabalhos correlatos possuem apenas um tipo de comunicação, sendo ela Bluetooth ou Wi-Fi, diferente do trabalho desenvolvido que consegue utilizar tanto

comunicação via Bluetooth, quanto comunicação via MQTT. Por fim, observa-se que todos os trabalhos, exceto o de Prado (2012) possuem interface móvel.

4 CONCLUSÕES

Este trabalho descreveu o desenvolvimento de um *framework*, cujo objetivo principal era realizar a comunicação com equipamentos eletroeletrônicos residenciais. Durante o desenvolvimento do trabalho foi levantada a necessidade de trazer estruturas adicionais que auxiliassem a utilização do *framework*, sendo assim, foi desenvolvido um *framework* multiplataforma que provém estruturas de comunicação via MQTT e Bluetooth, além de estruturas que podem ser utilizadas para uma melhor organização da aplicação, como estruturas que simulam casas, cômodos e dispositivos. Além do desenvolvimento do *framework*, este trabalho também teve como objetivo desenvolver uma aplicação que utilizasse o *framework* e conseguir realizar a comunicação com pelo menos um tipo de equipamento. Tais objetivos foram atingidos, sendo que foi possível desenvolver um aplicativo que utiliza as duas interfaces de comunicação providas pelo *framework*, além de realizar a comunicação com o dispositivo Sonoff, através da interface de comunicação MQTT.

Também foi realizada uma avaliação do *framework* desenvolvido com três acadêmicos de ciência da computação da FURB. Para isso, os acadêmicos realizaram o desenvolvimento de uma pequena aplicação para IoT utilizando o *framework* desenvolvido. Ao final do desenvolvimento, os acadêmicos responderam um questionário a fim de avaliar todas as funções e estruturas providas pelo *framework*. A partir dos resultados dessa avaliação, foi possível confirmar a funcionalidade e utilidade do *framework*. Além disso, foi realizada uma comparação com o trabalho desenvolvido e os trabalhos correlatos, destacando assim o trabalho desenvolvido que conseguiu realizar a comunicação com dispositivos através de duas interfaces de comunicação, Bluetooth e MQTT, diferente dos trabalhos correlatos que são limitados apenas à comunicação Wi-Fi ou Bluetooth.

A tecnologias utilizadas para a produção deste trabalho supriram as principais necessidades durante o desenvolvimento. O protocolo de comunicação MQTT funcionou corretamente, não havendo problemas em sua utilização. Já o *plugin* Bluetooth funcionou satisfatoriamente, foi possível se comunicar com um módulo Bluetooth classe 2 utilizando um dispositivo móvel com sistema operacional Android. Contudo, foi possível observar limitações em relação a este *plugin*, por conta de ele não funcionar em dispositivos móveis mais antigos e/ou com especificações de *hardware* inferiores.

Por fim, este trabalho traz contribuições para o desenvolvimento tecnológico, em específico para a comunidade de desenvolvedores, por conta do *framework* desenvolvido

poder ser utilizado por qualquer desenvolvedor, o que facilita o desenvolvimento de aplicações para IoT, por conta de trazer interfaces de comunicação MQTT e Bluetooth, dispensando a sua implementação, além de estruturas adicionais que auxiliam o controle e organização da aplicação.

4.1 EXTENSÕES

Como extensões para o *framework* desenvolvido, sugere-se:

a) implementar outras interfaces de comunicação, por exemplo uma interface de comunicação via infravermelho;
b) avaliar a implementação de outro *plugin* Bluetooth, a fim de resolver problemas de compatibilidade com dispositivos móveis mais antigos e possibilitar a comunicação com dispositivos Bluetooth Low Energy (BLE);
c) implementar novas estruturas de comando;
d) implementar funcionalidades de cenas;
e) implementar comandos de voz para o envio de mensagens;
f) implementar a comunicação com *beacons*.

REFERÊNCIAS

APPLE. **HomeKit Accessory Protocol Especification (Non-Commerciapl Version) - Support - Apple Developer.** California, 2017a. Disponível em: <https://developer.apple.com/support/homekit-accessory-protocol/>. Acesso em 10 set. 2017.

_____. **Usar o app Casa no iPhone, iPad e iPod touch – Suporte da Apple**. California, 2017b. Disponível em <https://support.apple.com/pt-br/HT204893>. Acesso em 12 set. 2017.

ARENDS, Theo. **Sonoff-Tasmota**. [S.l.], 2018. Disponível em <https://github.com/arendst/Sonoff-Tasmota>. Acesso em 30 jun. 2018.

CIOCARI, Leonardo. **Controle e monitoramento do consumo de energia elétrica de equipamentos residenciais via Android**. 2013. 70 f. Monografia (Pós-Graduação em Desenvolvimento de Produtos Eletrônicos) – Instituto Federal De Educação, Ciência e Tecnologia de Santa Catarina, Florianópolis.

DINIZ, Marisa Fonseca. **A evolução da habitação**. [S.l.], 2014. Disponível em <https://marisadiniz.wordpress.com/2014/07/16/a-evolucao-da-habitacao/>. Acesso em 03 set. 2017.

DRIFTY. **Build Amazing Native Apps and Progressive Web Apps with Ionic Framework and Angular**. [S.l.], 2017a. Disponível em <https://ionicframework.com/>. Acesso em 12 set. 2017.

_____. **Ionic Documentation**. [S.l.], 2017b. Disponível em <https://ionicframework.com/docs/>. Acesso em 12 set. 2017.

FRACCHETTA, Alexandre. **Automação residencial já é uma realidade**. [S.l.], 2015. Disponível em <http://www.forumdaconstrucao.com.br/conteudo.php?a=11&Cod=1832>. Acesso em 27 ago. 2017.

ITEAD. **Sonoff Basic**: Wifi Remote Control Smart Switch. [S.l], 2017. Disponível em < https://www.itead.cc/sonoff-wifi-wireless-switch.html>. Acesso em 20 out. 2017.

KYAS, Othmar. **How to Smart Home**: A Step by Step Guide Using Internet, Z-Wave, KNX & OpenRemote. Alemanha: Key Concepts Press, 2013.

MICROSOFT. **Visual Studio Code - Code Editing Redefined.** Seattle, 2018. Disponível em: <https://code.visualstudio.com/>. Acesso em 12 jun. 2018.

MORIBE, Sérgio. **Automação de sala de Home Theater utilizando dispositivos móveis baseados em Android**. 2013. 54 f. Monografia de especialização (Especialista em Tecnologia Java e Desenvolvimento para Dispositivos Móveis) – Universidade Tecnológica Federal do Paraná, Curitiba.

NIELSON, Rodrigo Orthmann. **FWIOTFURB**. Santa Catarina, 2018. Disponível em: <https://bitbucket.org/gcgfurb/rodrigoorthmannnielson/>. Acesso em 2 jul. 2018.

NPM, Inc**. About npm**. [S,l], 2018. Disponível em <https://www.npmjs.com/about>. Acesso em 17 jun. 2018.

PRADO, Caio V. D. **Framework Web para Automação**. 2012. 52 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Vila Velha, Vila Velha.

ROSE, Karen; ELDRIDGE, Scott; CHAPIN, Lyman. **The Internet of Things**: An Overview. Geneva, Suíça: The Internet Society (ISOC), 2015

ROVERI, Michael Rubens. **Automação Residencial**, 2012. 87 f. Trabalho de conclusão de curso (Tecnólogo em Redes de Computadores) – Faculdade Politec, Santa Bárbara d'Oeste.

SAINI, Gaurav. **Hybrid Mobile Development with Ionic**: Build high performance hybrid applications with HTML, CSS, and JavaScript. Reino Unido: Packt Publishing Ltd. 2017.

SPIVEY, Dwight. **Home Automation for Dummies**: a Wiley Brand. Estados Unidos: John Wiley & Sons, Inc., 2015.

YUAN, Michael. **Conhecendo o MQTT**. [S, l], 2017. Disponível em <https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html>. Acesso em 18 jun. 2018.

APÊNDICE A – Guia de instalação e utilização do framework desenvolvido

As Figuras de 30 a 33 mostram o detalhamento da instalação e utilização do *framework* desenvolvido.

Figura 30 - Guia de instalação e utilização do framework parte 1

O fwiotfurb é um framework para Ionic 2+ que visa facilitar a comunicação com dispositivos para IoT, possuindo interfaces de comunicação como bluetooth e mqtt.

Pré-requisitos

- Possuir o node.js instalado
- Conhecimento básico de Ionic 2+
- Android SDK (caso utilize android)

Preparação rápida de um projeto

1. Caso não possua o Ionic e Cordova instalados, instalar globalmente

```
npm install -g ionic cordova
```

2. Iniciar um novo projeto Ionic

```
ionic start MeuAPPIot blank
```

3. Navegar para a pasta do projeto

```
cd MeuAPPIot
```

4. Adicionar o plugin bluetooth do Cordova

```
ionic cordova plugin add cordova-plugin-bluetooth-serial
```

5. Adicionar o pacote bluetooth do Cordova

```
npm install --save @ionic-native/bluetooth-serial
```

6. Instalar o pacote fwiotfurb

```
npm install fwiotfurb
```

Importar os providers

Tanto para o ambiente Mqtt quanto Bluetooth, é necessário importar os seus providers no app.module.ts

```
import { FwMqttProvider, FwBluetoothProvider } from 'fwiotfurb';
import { BluetoothSerial } from '@ionic-native/bluetooth-serial';

...

providers: [
    ...
    BluetoothSerial,
    FwBluetoothProvider,
    FwMqttProvider,
    ...
]
```

Ambiente MQTT

Para informações sobre o que é e como funciona o MQTT: Site oficial, Artigo IBM.

Para a utilização do MQTT, é necessário um broker. Um broker gratuido disponibilizado na web é o CloudMQTT.

Para criação de conta e de uma instância no CloudMQTT, acessar o tutorial no youtube

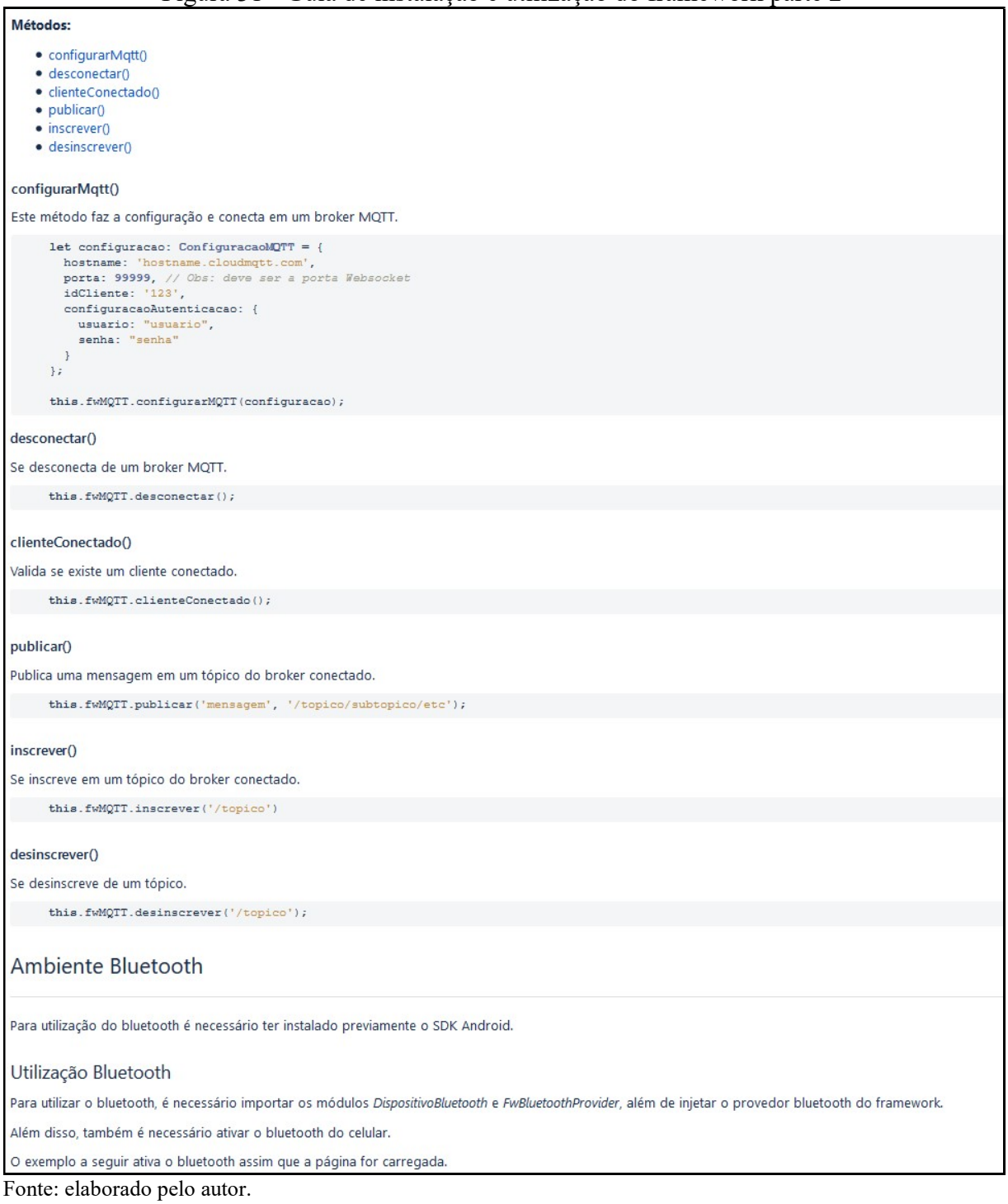
Utilização do MQTT

Para utilizar o MQTT, é necessário importar os módulos *FwMqttProvider*, *ConfiguracaoMQTT* e *Paho*, além de injetar o provedor MQTT do framework.

```
import { FwMqttProvider, ConfiguracaoMQTT } from 'fwiotfurb';
import { Paho } from 'ng2-mqtt/mqttws31';

...

export class HomePage {

constructor
    (
    public navCtrl: NavController,
    public fwMQTT: FwMqttProvider
    ) {}

...
```

Fonte: elaborado pelo autor.

Figura 31 - Guia de instalação e utilização do framework parte 2



Métodos:

- configurarMqtt()
- desconectar()
- clienteConectado()
- publicar()
- inscrever()
- desinscrever()

configurarMqtt()

Este método faz a configuração e conecta em um broker MQTT.

```
let configuracao: ConfiguracaoMQTT = {
  hostname: 'hostname.cloudmqtt.com',
  porta: 99999, // Obs: deve ser a porta Websocket
  idCliente: '123',
  configuracaoAutenticacao: {
    usuario: "usuario",
    senha: "senha"
  }
};

this.fwMQTT.configurarMQTT(configuracao);
```

desconectar()

Se desconecta de um broker MQTT.

```
this.fwMQTT.desconectar();
```

clienteConectado()

Valida se existe um cliente conectado.

```
this.fwMQTT.clienteConectado();
```

publicar()

Publica uma mensagem em um tópico do broker conectado.

```
this.fwMQTT.publicar('mensagem', '/topico/subtopico/etc');
```

inscrever()

Se inscreve em um tópico do broker conectado.

```
this.fwMQTT.inscrever('/topico')
```

desinscrever()

Se desinscreve de um tópico.

```
this.fwMQTT.desinscrever('/topico');
```

Ambiente Bluetooth

Para utilização do bluetooth é necessário ter instalado previamente o SDK Android.

Utilização Bluetooth

Para utilizar o bluetooth, é necessário importar os módulos *DispositivoBluetooth* e *FwBluetoothProvider*, além de injetar o provedor bluetooth do framework.

Além disso, também é necessário ativar o bluetooth do celular.

O exemplo a seguir ativa o bluetooth assim que a página for carregada.

Fonte: elaborado pelo autor.

Figura 32 - Guia de instalação e utilização do framework parte 3

```
import { DispositivoBluetooth, FwBluetoothProvider } from 'fwiotfurb';

...

export class HomePage {

    constructor(
        private fwBluetooth: FwBluetoothProvider,
        private platform: Platform
    ) {
        this.platform.ready().then(() => fwBluetooth.ativarBluetooth());
    }
...
```

Métodos:

- ativarBluetooth()
- dispositivoConectado()
- conectarDispositivo()
- enviarMensagem()
- conectaEnviaMensagemDispositivo()
- listarDispositivosPareados()
- listarDispositivosNaoPareados()

ativarBluetooth()

Ativa o bluetooh do celular.

```
fwBluetooth.ativarBluetooth();
```

dispositivoConectado()

Valida se existe algum dispositivo conectado.

```
this.fwBluetooth.dispositivoConectado()
    .then((conectado) => console.log('Dispositivo conectado'))
    .catch((desconectado) => console.log('Dispositivo desconectado'));
```

conectarDispositivo()

Conecta em um dispositivo a partir do seu endereço MAC.

```
this.fwBluetooth.conectarDispositivo('enderecoMAC');
```

enviarMensagem()

Envia uma mensagem pro dispositivo conectado.

```
this.fwBluetooth.enviarMensagem('mensagem');
```

conectaEnviaMensagemDispositivo()

Conecta em um dispositivo e envia uma mensagem.

```
this.fwBluetooth.conectaEnviaMensagemDispositivo("mensagem", 'enderecoMac');
```

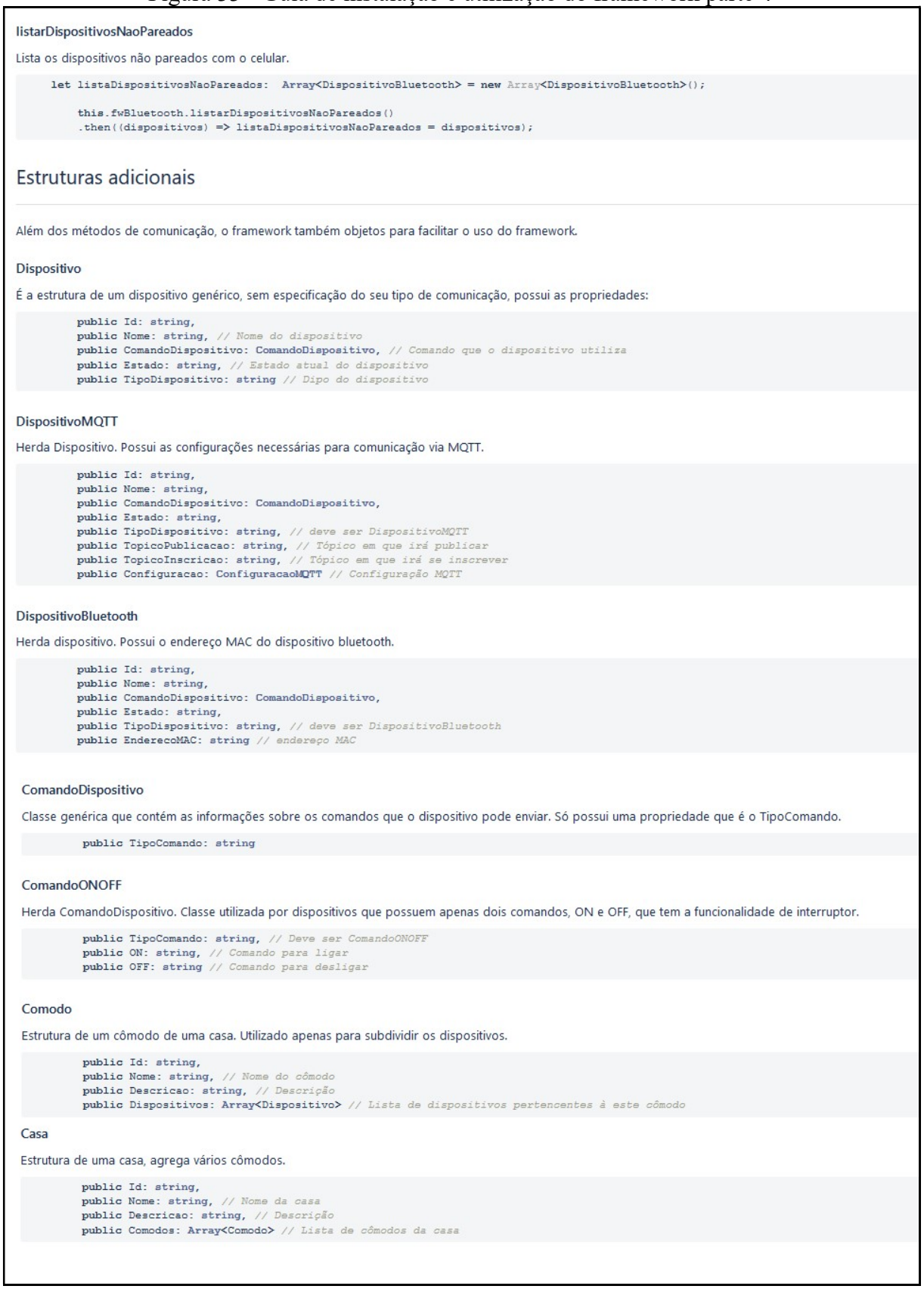
listarDispositivosPareados()

Lista os dispositivos pareados com o celular.

```
let listaDispositivosPareados: Array<DispositivoBluetooth> = new Array<DispositivoBluetooth>();

this.fwBluetooth.listarDispositivosPareados()
  .then((dispositivos) => {
    listaDispositivosPareados = dispositivos;
});
```

Fonte: elaborado pelo autor.

Figura 33 - Guia de instalação e utilização do framework parte 4



listarDispositivosNaoPareados

Lista os dispositivos não pareados com o celular.

```
let listaDispositivosNaoPareados:  Array<DispositivoBluetooth> = new Array<DispositivoBluetooth>();

    this.fwBluetooth.listarDispositivosNaoPareados()
    .then((dispositivos) => listaDispositivosNaoPareados = dispositivos);
```

Estruturas adicionais

Além dos métodos de comunicação, o framework também objetos para facilitar o uso do framework.

Dispositivo

É a estrutura de um dispositivo genérico, sem especificação do seu tipo de comunicação, possui as propriedades:

```
public Id: string,
public Nome: string, // Nome do dispositivo
public ComandoDispositivo: ComandoDispositivo, // Comando que o dispositivo utiliza
public Estado: string, // Estado atual do dispositivo
public TipoDispositivo: string // Dipo do dispositivo
```

DispositivoMQTT

Herda Dispositivo. Possui as configurações necessárias para comunicação via MQTT.

```
public Id: string,
public Nome: string,
public ComandoDispositivo: ComandoDispositivo,
public Estado: string,
public TipoDispositivo: string, // deve ser DispositivoMQTT
public TopicoPublicacao: string, // Tópico em que irá publicar
public TopicoInscricao: string, // Tópico em que irá se inscrever
public Configuracao: ConfiguracaoMQTT // Configuração MQTT
```

DispositivoBluetooth

Herda dispositivo. Possui o endereço MAC do dispositivo bluetooth.

```
public Id: string,
public Nome: string,
public ComandoDispositivo: ComandoDispositivo,
public Estado: string,
public TipoDispositivo: string, // deve ser DispositivoBluetooth
public EnderecoMAC: string // endereço MAC
```

ComandoDispositivo

Classe genérica que contém as informações sobre os comandos que o dispositivo pode enviar. Só possui uma propriedade que é o TipoComando.

```
public TipoComando: string
```

ComandoONOFF

Herda ComandoDispositivo. Classe utilizada por dispositivos que possuem apenas dois comandos, ON e OFF, que tem a funcionalidade de interruptor.

```
public TipoComando: string, // Deve ser ComandoONOFF
public ON: string, // Comando para ligar
public OFF: string // Comando para desligar
```

Comodo

Estrutura de um cômodo de uma casa. Utilizado apenas para subdividir os dispositivos.

```
public Id: string,
public Nome: string, // Nome do cômodo
public Descricao: string, // Descrição
public Dispositivos: Array<Dispositivo> // Lista de dispositivos pertencentes à este cômodo
```

Casa

Estrutura de uma casa, agrega vários cômodos.

```
public Id: string,
public Nome: string, // Nome da casa
public Descricao: string, // Descrição
public Comodos: Array<Comodo> // Lista de cômodos da casa
```

Fonte: elaborado pelo autor.

APÊNDICE B – Questionário de usabilidade do framework desenvolvido

Os Quadros 11 a 16 apresentam o questionário realizado com os acadêmicos de Ciência da Computação da FURB para avaliar a usabilidade do *framework* desenvolvido.

Quadro 11 - Questionário do conhecimento das tecnologias

AVALIAÇÃO DO CONHECIMENTO DAS TECNOLOGIAS

Esta seção tem o objetivo de avaliar o conhecimento do usuário em relação as tecnologias utilizadas no framework.

1. **Como você avalia o seu conhecimento em relação ao Ionic Framework?**
   ( ) Não conheco o Ionic Framework.
   ( ) Conheço, mas nunca implementei nada.
   ( ) Conheço e já desenvolvi pequenas aplicações.
   ( ) Conheço bem, já desenvolvi aplicações complexas.

2. **Como você avalia o seu conhecimento em relação ao MQTT?**
   ( ) Não conheço o MQTT.
   ( ) Conheço o conceito, mas nunca utilizei.
   ( ) Conheço e já utilizei.

3. **Como você avalia o seu conhecimento em relação ao plugin BluetoothSerial do Ionic?**
   ( ) Não conheço.
   ( ) Conheço, mas nunca utilizei.
   ( ) Conheço e já utilizei.

4. **Como você avalia o seu conhecimento em relação à aplicações para IoT?**
   ( ) Nunca desenvolvi nada para IoT.
   ( ) Já desenvolvi pequenas aplicações.
   ( ) Já desenvolvi aplicações complexas.

Fonte: elaborado pelo autor.

Quadro 12 - Questionário da instalação do framework

AVALIAÇÃO DA INSTALAÇÃO DO FRAMEWORK

Esta seção tem o objetivo de avaliar como foi o processo de instalação do framework.

1. **Em uma escala de 1 a 5, qual foi o nível de dificuldade para realizar a instalação do framework?**

	1	2	3	4	5	
Muito fácil	( )	( )	( )	( )	( )	Muito difícil

2. **Qual foi a principal dificuldade para a instalação do framework?** _____
   _____

Fonte: elaborado pelo autor.

Quadro 13 - Questionário da interface de comunicação MQTT

AVALIAÇÃO DA INTERFACE DE COMUNICAÇÃO MQTT (FWMQTTPROVIDER)

Esta seção tem o objetivo de avaliar a funcionalidade e usabilidade do da interface de comunicação MQTT.

1. **Você conseguiu utilizar a interface de comunicação MQTT? (FwMqttProvider)**
   ( ) Sim.
   ( ) Não.
   Outro. _____

2. **A interface de comunicação MQTT funcionou corretamente? Foi possível enviar comandos para o broker MQTT?**
   ( ) Sim.
   ( ) Não.
   Outro. _____

3. **Em uma escala de 1 à 5, qual o nível de dificuldade para a utilização da interface de comunicação MQTT?**

	1	2	3	4	5	
Muito fácil	( )	( )	( )	( )	( )	Muito difícil

4. **Qual foi a principal dificuldade na utilização da interface de comunicação MQTT?**
   _____

Fonte: elaborado pelo autor.

Quadro 14 - Questionário da interface de comunicação Bluetooth

AVALIAÇÃO DA INTERFACE DE COMUNICAÇÃO BLUETOOTH (FWBLUETOOTHPROVIDER)

Esta seção tem o objetivo de avaliar a funcionalidade e usabilidade do da interface de comunicação Bluetooth.

1. **Você conseguiu utilizar a interface de comunicação Bluetooth?**
   ( ) Sim.
   ( ) Não.
   Outro. _____

2. **A interface de comunicação funcionou corretamente?**
   ( ) Sim.
   ( ) Não.
   Outro. _____

3. **Foi possível listar os dispositivos bluetooth?**
   ( ) Sim.
   ( ) Não.
   Outro. _____

4. **Você conseguiu enviar comandos para algum dispositivo bluetooth?**
   ( ) Sim.
   ( ) Não.
   Outro. _____

5. **Em uma escala de 1 à 5, qual o nível de dificuldade para a utilização da interface de comunicação Bluetooth?**

	1	2	3	4	5	
Muito fácil	( )	( )	( )	( )	( )	Muito difícil

6. **Qual foi a principal dificuldade na utilização da interface de comunicação Bluetooth?**
   _____

Fonte: elaborado pelo autor.

Quadro 15 - Questionário dos objetos providos pelo framework

AVALIAÇÃO DOS OBJETOS PROVIDOS PELO FRAMEWORK (CASA, DISPOSITIVO, CÔMODO, ETC)

Esta seção tem como objetivo avaliar a utilidade dos objetos providos pelo framework.

1. **Você utilizou alguma estrutura provida pelo framework? Quais?**
   ( ) Casa.
   ( ) Cômodo.
   ( ) Dispositivo.
   ( ) DispositivoBluetooth.
   ( ) DispositivoMQTT.
   ( ) ComandoDispositivo.
   ( ) ComandoONOFF.
   ( ) ConfiguracaoMQTT.
   ( ) Nenhuma.

2. **Em uma escala de 1 à 5, como você avalia a utilidade dessas estruturas?**
   1 2 3 4 5
   Inútil ( ) ( ) ( ) ( ) ( ) Muito útil

3. **Em uma escala de 1 à 5, qual o nível de dificuldade para utilização dessas estruturas?**
   1 2 3 4 5
   Muito fácil ( ) ( ) ( ) ( ) ( ) Muito difícil

4. **Você possui alguma sugestão de estrutura adicional ou melhoria? _____**
   _____

Fonte: elaborado pelo autor.

Quadro 16 - Questionário da avaliação geral do framework

AVALIAÇÃO GERAL DO FRAMEWORK

Esta seção tem o objetivo de fazer uma avaliação geral do framework.

1. **Em uma escala de 1 à 5, o quão útil você considera este framework?**

	1	2	3	4	5	
Inútil	( )	( )	( )	( )	( )	Muito útil

2. **Caso fosse desenvolver alguma aplicação para IoT, você utilizaria este framework?**
   ( ) Sim.
   ( ) Não.
   ( ) Talvez.

3. **De modo geral, quais foram as principais dificuldades para a utilização do framework?**
   _____

4. **Você recomendaria este framework para outro desenvolvedor?**
   ( ) Sim.
   ( ) Não.
   ( ) Talvez.

5. **Você possui alguma sugestão de melhoria? Qual?** _____
   _____

Fonte: elaborado pelo autor.