Aplicação Mobile React Native para automação residencial utilizando ESP8266

Jefter Franciano Lopes

PROJETO DE CONCLUSÃO DE CURSO

Data de Depósito: 30/11/2021

Assinatura: _____

Aplicação Mobile React Native para automação residencial utilizando ESP8266

Jefter Franciano Lopes

Valter Ribeiro Lima Jr

Monografia apresentada ao Instituto Superior de Educação do UNIFOR/MG, como requisito parcial para obtenção do título de bacharel em Ciência da Computação, sob a orientação do Profº. Valter Ribeiro Lima Jr

UNIFOR-MG – Formiga
30/11/2021

Sumário

Lista de Figuras

Lista de Siglas

API - *Application Programming Interface*
DML - *Data Manipulation Language*
GPIO - *General Purpose Input/Output*
HTTP - *Hypertext Transfer Protocol*
IDE - *Integrated Development Environment*
IoT - *Internet of Things*
IP - *Internet Protocol*
JSON - *Javascript Object Notation*
LED - Diodo Emissor de Luz
SQL - *Structured Query Language*
USB - *Universal Serial Bus*
WIFI - *Wireless Fidelity*

Lista de Tabelas

Resumo

Lopes, J. F. *Aplicação Mobile React Native para automação residencial utilizando ESP8266.* Monografia (Graduação) — Centro Universitário de Formiga – UNIFOR-MG – Formiga, 2021.

Com o crescente avanço tecnológico, e a constante busca por comodidade e praticidade com baixo custo, as pessoas vem procurando cada vez mais por sistemas automatizados, como robôs, eletrodomésticos, casas, dentre outras oportunidades. Atualmente existem vários projetos que trazem equipamentos ou componentes que podem fornecer essa tecnologia. Este projeto apresenta um destes equipamentos mencionados, que é a placa NodeMCU ESP8266 ESP-12 que está hoje entre as mais desejadas e entre as mais utilizadas para projetos de automação industrial e residencial. O projeto apresenta, de forma detalhada, a construção de um sistema de automação residencial baseado na tecnologia ESP8266, usando como equipamento principal a placa NodeMCU ESP8266. O microcontrolador ESP8266 controla alguns atuadores, como *leds*, um portão e um ventilador, localizados em uma maquete. O acionamento é feito através de um aplicativo móvel desenvolvido em React Native, onde serão cadastrados os ambientes da residência bem como os dispositivos a serem acionados.

Palavras-chave: Automação Residencial, ESP8266, microcontrolador, React Native

Capítulo

1

Introdução

1.1 Considerações Iniciais

O termo Tecnologia tem origem no grego ”tekhn” que significa ”técnica” e o sufixo ”logia” que significa ”estudo”. A tecnologia é o resultado da junção entre ciência e engenharia, que envolvem um conjunto de instrumentos, métodos e técnicas a fim de facilitar e resolver problemas. (NEVILLE, 2013)

O avanço tecnológico expandiu de forma acelerada após a Revolução Industrial do século XVIII que provocou diversas transformações no processo produtivo em geral e proporcionou o surgimento de invenções tecnológicas. (NEVILLE, 2013)

Já no século XX, pode-se destacar o avanço nas tecnologias de informações e comunicações, que se deve a evolução das telecomunicações, o uso de computadores, o desenvolvimento e acesso a Internet, e, o surgimento de novos aparelhos eletrônicos. (RODRIGUEZ, 2019)

Em 1994 foi lançado o primeiro aparelho que mesclava a telefonia com a computação, o IBM Simon, desde então os aparelhos evoluíram para o que conhecemos hoje como *smartphones*, que possibilitam muito mais do que só fazer ligações e salvar contatos. (RODRIGUEZ, 2019)

Com os celulares mais modernos a criação e desenvolvimento de diversos aplicativos se tornaram essenciais, os aplicativos disponíveis de forma gratuita ou paga possibilitam um leque de opções aos usuários, que vai desde o lazer ao trabalho, facilitando e organizando a vida de cada usuário.(RODRIGUEZ, 2019)

Para a instalação dos aplicativos, e, funcionamento de alguns, a Internet é essencial. A Internet existe há pouco mais de duas décadas na vida das pessoas, primeiramente ela era uma rede de computadores com conectividade mundial que permitia a conexão entre as universidades, governos e órgãos militares, posteriormente abrangeu o comércio em geral, as residências e outros aparelhos eletrônicos.(FILHO, 2016)

Um novo termo para a Internet vem se destacando, “Internet das Coisas” (ou IoT - sigla para

Internet of Things), nada mais é do que a expansão com limites imensamente maiores do que é conhecido, permitindo que a comunicação entre objetos em rede sejam estendidas.(FILHO, 2016)

Com a expansão da IoT a criação dos sistemas de automação predial e residencial podem se tornar um dos mais promissores. Vale destacar que a IoT não é uma tecnologia, e sim um conjunto de tecnologias e topologias integradas, que operam tanto em macroestruturas ("*cloud*") como em microestruturas ("*fog*").(FILHO, 2016)

1.2 Objetivos do Trabalho

Pensando em como a "Internet das Coisas" está presente no dia a dia e o alto custo que isso representa para quem deseja automatizar sua residência, este projeto tem como objetivo desenvolver um aplicativo para celular que consiga acionar os dispositivos de uma residência.

Um dos princípios que fundamentaram a pesquisa foi o custo-benefício, pois existem diversos benefícios proporcionados ao usuário, tais como: praticidade, conforto e segurança na execução das tarefas com baixo custo de investimento.

O celular envia dados via "Protocolo HTTP" e os dados são recebidos pelo microcontrolador, tomando as ações necessárias inerentes ao controle da residência, o qual realiza a função desejada.

Além disso com o aplicativo será possível enviar comandos a vários microcontroladores e um microcontrolador poderá controlar vários dispositivos, proporcionando baixo custo ao projeto.

1.2.1 Objetivos Específicos

Utilizar a Placa NodeMCU ESP8266, que consegue se conectar ao Wi-Fi de um roteador, e dentro da rede e se comunicar com o aplicativo no dispositivo móvel utilizando o "Protocolo HTTP". O microcontrolador ESP8266 tem um baixo custo e facilidade para escrever códigos utilizando a IDE do Arduino.

Desenvolver um aplicativo móvel utilizando o Framework "React Native" que permite a geração de aplicativos nativos tanto pra Android, quanto para iOS.

Criar a maquete de uma residência, com alguns dispositivos conectados ao microcontrolador para a demonstração das funcionalidades do aplicativo.

1.3 Estrutura da Monografia

O Projeto foi distribuído em 6 capítulos. No Capítulo 1 contêm a introdução ao trabalho juntamente com os objetivos do trabalho. Os termos utilizados, ferramentas, softwares e estudos necessários para o planejamento e desenvolvimento do projeto serão apresentados no Capítulo 2. Alguns trabalhos relacionados a esse, são encontrados no Capítulo 3. No Capítulo 4 são apresentadas as metodologias utilizadas assim como os componentes presentes no projeto. Os resultados obtidos estão no Capítulo 5 assim como toda a evolução do projeto. E, por fim no Capítulo 6 estão a conclusão e os trabalhos futuros.

Capítulo

2

Referencial Teórico

Neste capítulo serão apresentadas as ferramentas, conceitos, softwares e tecnologias que foram utilizadas durante o desenvolvimento do projeto.

2.1 IOT - Internet das Coisas

Atualmente, Internet das Coisas (IoT - *Internet of Things*) é classificado como um conjunto de tecnologias e protocolos que quando associados permitem que objetos se conectem a redes de comunicações sendo identificados e controlados através de uma conexão de rede. O pesquisador Meira (2016) define "coisas" como dispositivos que tem em alguma intensidade, capacidades de computação, comunicação e controle simultaneamente, dando o conceito de Internet das coisas onde as coisas são objetos digitais completos.

Vale ressaltar que o conceito de IoT ainda é muito divergente, sendo assim não há nenhum conceito considerado pacífico ou unânime. Uma característica comum entre as diversas definições de IoT é que todas apontam para o conceito de que os computadores, sensores e objetos interagem entre si e uns com os outros, processando as informações e ou dados em um contexto de hiperconectividade. (MAGRANI, 2018)

2.1.1 Automação Residencial

A definição de Automação Residencial pode ser entendida como um conjunto de serviços desenvolvidos por sistemas tecnológicos integrados que tem como objetivo suprir as necessidades básicas de uma residência, como a segurança, a comunicação, economia e gestão energética, conforto luminotécnico e de temperaturas, dentre outros, proporcionando maior conforto e controle. (MURATORI; Bó, 2015)

É uma rede de hardware, comunicação e interfaces eletrônicas que trabalham para integrar dispositivos uns aos outros via Internet. (ENGENHARIA, 2021b)

A automação pode ser empregada em diversas áreas, desde que estejam conectadas a dispo-

sitivos que se comunicam através de *smartphones*, *tablets* ou computadores desde que estejam conectados via Wi-Fi ou outro meio de comunicação. Dentre elas estão os eletrodomésticos, as luzes, alarmes e câmeras de segurança, portões eletrônicos, sistemas de som e vídeo, cortinas, sistemas de irrigação, dentre outros. (MURATORI; Bó, 2015)

De acordo com Engenharia (2021b) existem três principais elementos de um sistema de automação residencial: sensores, controladores e atuadores.

- Os sensores podem monitorar mudanças na luz do dia, temperatura ou detecção de movimento.
- Os controladores são os dispositivos como computadores, *tablets* e *smarphones* usados para enviar e receber mensagens sobre o status dos recursos automatizados em uma residência.
- O atuadores podem ser interruptores de luz, motores ou válvulas motorizadas que controlam o mecanismo ou função de um sistema de automação residencial. Eles são programados para serem ativados por um comando remoto por meio de um controlador.

(ENGENHARIA, 2021b)

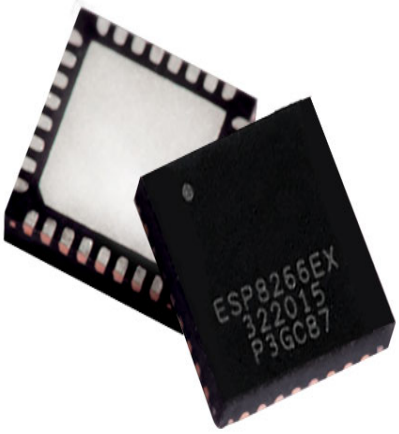
2.1.2 ESP8266

O ESP8266 é um microcontrolador do fabricante chinês Expressif com a capacidade de se conectar a Internet via Wi-Fi, sem a necessidade de nenhum módulo externo. O chip permite a conexão de diversos dispositivos a Internet (ou rede local) como sensores, atuadores, etc. Devido apresentar boa integração com SOCs (chips de sistemas Wi-fi), foi adotado em placas Arduino. (SYSTEMS; TECHNOLOGY, 2015)

Ele fornece capacidade de incorporar recursos de Wi-Fi em outros sistemas ou de funcionar como um sistema autônomo, podendo trabalhar do lado do cliente ou do servidor, com baixo custo e requisitos mínimos de espaço. (SYSTEMS; TECHNOLOGY, 2015)

Para facilitar o uso desse chip, vários fabricantes criaram placas e módulos para desenvolvimento que variam em tamanho, número de pinos ou tipos de conexão. A Figura 2.1 ilustra o modelo ESP8266X. (SYSTEMS; TECHNOLOGY, 2015)

Figura 2.1: Chip ESP8266X



Fonte:(SYSTEMS; TECHNOLOGY, 2015)

Para facilitar o uso do ESP8266 foram desenvolvidos módulos que são integrados em algumas placas de desenvolvimento. Serão usados dois desse módulos no projeto, são eles:

NodeMCU ESP12E

O módulo ESP-12E Wi-Fi foi desenvolvido pela Ai-thinker. e possui o processador ESP8266. É uma placa de desenvolvimento que combina o chip ESP8266, a uma interface usb-serial e um regulador de tensão 3.3V. Sua programação pode ser feita usando LUA ou a IDE do Arduino, utilizando a comunicação via cabo micro-usb. Possui antena embutida e 11 pinos de I/O e conversor analógico-digital. (SYSTEMS; TECHNOLOGY, 2015)

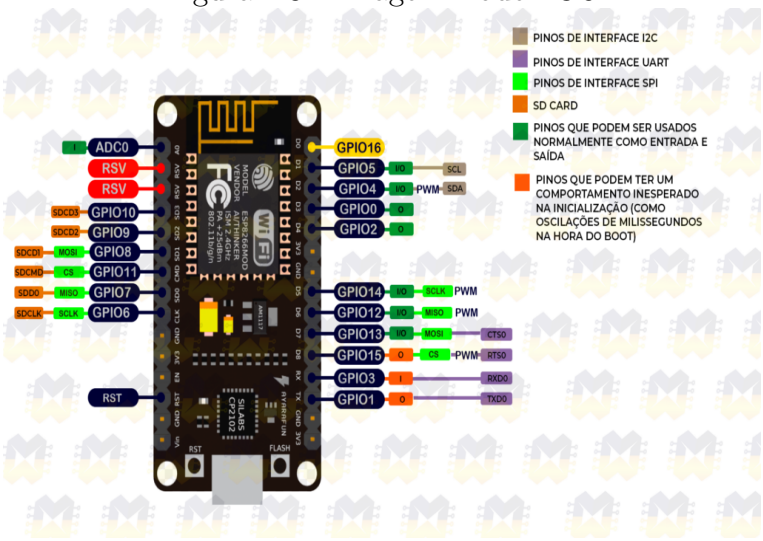
O modelo ESP12E pode ser visto na Figura 2.2, ele é responsável pelo gerenciamento das portas na placa NODE MCU.

Figura 2.2: ESP12E

Fonte: (SYSTEMS; TECHNOLOGY, 2015)

Em relação a pinagem, o NodeMCU tem um total de 30 pinos, os quais permitem conexões com os mais variados módulos e sensores. A Figura 2.3 ilustra os pinos da placa. (OLIVEIRA, 2021)

Figura 2.3: Pinagem NodeMCU



Fonte: (OLIVEIRA, 2021)

Em relação a Figura 2.3, se verifica que:

- há 4 pinos para a alimentação, um pino VIN e três pinos 3.3V.
- Os pinos GND são de aterramento da placa.
- A placa possui 17 pinos GPIO(*General Purpose Input/Output*)[1] que são usados para conexão de módulos, sensores e componentes eletrônicos.
- Os pinos GPIOs 6 e 11 estão conectados a memória flash do NodeMCU, portanto não é recomendado a utilização desses pinos para conexão de módulos e sensores.

(OLIVEIRA, 2021)

Apesar de existirem 17 pinos GPIO, de acordo com os testes realizados pelo Tecnologia (2021) os pinos 6, 7, 8 e 11 não funcionam para entrada e saída, portanto não conseguem se conectar a nenhum módulo, sensor, etc.

2.1.3 IDE do Arduino

Arduíno é uma das ferramentas mais populares utilizada no desenvolvimento de produtos IoT, é conhecido como o primeiro projeto de hardware de código aberto amplamente difundido.(ARDUINO, 2021) A linguagem utilizada em sua programação é a C e C++. Através do uso de bibliotecas o seu ambiente pode se estender, permitindo novas funcionalidades em seu uso. Permite que os usuários façam *upload* dos seus códigos para as placas que são compatíveis com ele. (ARDUINO, 2021)

[1]Um conjunto de pinos responsável por fazer a comunicação de entrada e saída de sinais digitais.

C e C++

A linguagem C foi criada por Dennis Ritchie em 1972 com o intuito de ser usada para desenvolver uma nova versão do sistema operacional Unix que até então utilizava o Assembly. (CASAVELLA, 2018)

É uma linguagem que permite seu uso em praticamente qualquer tipo de projeto, se adaptando em qualquer plataforma, ela permite a criação de sistemas operacionais, aplicativos diversos, drivers e outros controladores de dispositivos, programar microcomputadores, dentre outros. (CASAVELLA, 2018)

Por ser uma linguagem flexível ela é capaz de gerar programas rápidos em tempo de execução, com uma sintaxe simples e eficaz, com instruções de alto nível. Foi através dela que foram desenvolvidas outras linguagens como C++, Java, Objective C, dentre outras. (CASAVELLA, 2018)

Já linguagem de programação C++ foi criada por Bjarne Stroustrup, no início da década de 1980. Foi desenvolvida com base em C. É considerada atualmente uma das linguagens mais populares para programação orientada a objetos. Em 1998 foi padronizada pelo American National Standards Institute (ANSI) e pela International Standards Organization (ISO). (JUNIOR; VIRTUOSO; MARTINS, 2012)

2.2 JavaScript

JavaScript é uma linguagem de programação que foi criada pela Netscape em parceria com a Sun Microsystems, com a finalidade de tornar páginas Web interativas. (SILVA, 2010) Sua primeira versão, denominada JavaScript 1.0, foi lançada em 1995 e implementada em março de 1996 no navegador Netscape Navigator 2.0 quando o mercado ainda era dominado pela Netscape. (SILVA, 2010)

As linguagens de programação como por exemplo PHP, ASP, Java, Python, entre outras, foram desenvolvidas para executar do lado do servidor, isto é, dependem de uma máquina remota onde estão hospedadas as funcionalidades capazes de interpretar e fazer funcionar os programas.(SILVA, 2010)

O JavaScript foi desenvolvido para executar do lado do cliente, ou seja, essa interpretação depende de funcionalidades hospedadas no navegador do usuário. Isso só é possível porque existe um interpretador JavaScript hospedado no navegador. (SILVA, 2010)

A linguagem JavaScript permite a comunicação assíncrona e pode atualizar partes de uma página Web, ou mesmo todo o conteúdo da página. Também pode ser usado para informação de data e hora, efetuar animações, validar *inputs* de formulários, sugerir resultados de acordo com que o utilizador escreve em um *input*, e muito mais. (DIMES; TORRES, 2015)

JavaScript é a linguagem de programação do lado do cliente, mais popular do mundo usada hoje em dia, mas também pode ser usada do lado do servidor. Usando Node.Js, Meteor, Wakanda, MongoDB, entre outros, poderá usá-lo do lado do servidor. (DIMES; TORRES, 2015)

Como visto, JavaScript foi criada para interatividade com páginas Web, porém hoje em dia

é vista em várias outras áreas, como da programação móvel, por exemplo usando a biblioteca React Native que será apresentada na seção a seguir.

2.3 React Native

React Native foi lançado pelo Facebook em 2015 na conferência React.js, considerado o *framework* que revolucionaria a criação de aplicativos móveis. (DANIELSSON, 2016) Quando criado permitia suporte apenas para iOS, sendo atualizado desde então o suporte para o Android, e permanece em expansão até hoje. (DANIELSSON, 2016)

É uma estrutura de software livre que permite a construção de aplicativos usando o React e os recursos nativos da plataforma de aplicativos, através do JavaScript acessa as API's [2] (*Application Programming Interface* - Interface de Programação de Aplicativos) de sua plataforma, desenvolve a aparência e o comportamento de sua IU [3] (Interface de Usuário) , utilizando os pacotes de código reutilizável e aninhada. (DANIELSSON, 2016)

Ele possibilita a criação de aplicativos móveis de forma rápida e eficiente tanto no iOS quanto no Android. Levando em consideração que são duas plataformas distintas em aparência, sensação e recursos, mas baseadas na mesma linguagem e tendo os gráficos renderizados de formas diferentes de acordo com a plataforma de destino e ser componentes nativos reais. (NATIVE, 2021)

O React Native é executado dentro dos aplicativos em uma instância agregada de JavaScriptCore para o iOS, ou de V8 para o Android, renderizando para os componentes específicos da plataforma de nível superior. (NATIVE, 2021)

2.3.1 Expo

Expo é uma ferramenta de desenvolvimento móvel com React Native, utilizada pela facilidade de acesso as API's nativas do dispositivo sem a necessidade de instalar qualquer dependência ou alterar código nativo. (EXPO, 2021)

Com ele ao desenvolver uma aplicação React, se consegue gerar seus aplicativos tanto em Android quanto em IOS. Uma de suas desvantagens é que nem todas as APIs do Android e iOS estão disponíveis. (EXPO, 2021)

2.4 Node.js

Node.js é um ambiente de execução Javascript do lado do servidor. Ou seja, através dele é possível criar aplicações Javascript para ser executadas no servidor, sem a necessidade de um *browser* para executá-las. (NODEJS, 2021)

Ele possui uma execução *single-thread* que não exige resposta a cada requisição, ele possibilita executar várias de conexões simultâneas, porque não aguarda o processamento da resposta, ou seja, é altamente escalável. (SOUZA, 2020a)

[2] É um conjunto de definições e protocolos usado no desenvolvimento e na integração de software de aplicações.
[3] User interface ou interface do usuário: é o conjunto dos controles e canais sensoriais mediante as quais um usuário pode comunicar-se com uma máquina.

Node.js é uma biblioteca usada do lado do servidor para diversas finalidades, principalmente para criar API baseadas em JSON [4] (*Javascript Object Notation*) para conexões com banco de dados (SOUZA, 2020a). Permitindo que os dados sejam enviados para o *front-end* não havendo necessidade de diversas conexões que padronizam as informações facilitando que as informações cheguem ao usuário.(SOUZA, 2020a)

Para a instalação do Node.Js e suas dependências e pacotes, nescessita que se instale antes o gerenciador de pacotes NPM (*Node Package Manager*).

2.4.1 NPM

NPM que significa *Node Package Manager* é um gerenciador de pacotes que faz parte do Node.js. Ele permite instalar, desinstalar e atualizar dependências por meio de instrução na linha de comando. Sempre que é criado um projeto por meio dele, é adicionado um arquivo chamado package.json, que contém a relação dos pacotes instalados. (SOUZA, 2020b)

Sendo assim, quando for necessário realizar alguma alteração, o NPM verifica o arquivo package.json e faz as atualizações necessárias de forma simples e rápida. Graças a isso mantém-se a organização do projeto e de suas dependências, além de evitar erros de configurações ao fazer a instalação de pacotes de forma manual. (SOUZA, 2020b)

2.5 SQLite

O SQLite é um banco de dados relacional de código aberto que implementa um banco de dados SQL[5] (*Structured Query Language* - Linguagem de Consulta Estruturada) não havendo necessidade do uso de um servidor na sua execução. (SILVA et al., 2017) Tem a capacidade de armazenar seus arquivos em sua própria estrutura, sendo assim é considerado funcional em diversas aplicações, principalmente, *websites* menores e sistemas *mobile.*(SILVA et al., 2017) O formato do arquivo de banco de dados é multiplataforma, podendo copiar o banco de dados entre sistemas 32 e 64 *bits*. (SQLITE, 2021)

Sua utilização é recomendada em sistemas que exigem menor complexidade, no qual seus recursos são totalmente aproveitados, o que não acontece em aplicações de grande porte, em que a necessidade de processamento e armazenamento é maior.(SILVA et al., 2017)

Ele possibilita criar e manter diversas tabelas de banco de dados e utiliza comandos SQL. Esses dados são manipulados por meio de comandos DML [6] (*Data Manipulation Language* - Linguagem de manipulação de dados) (INSERT, UPDATE e DELETE) e se consulta utilizando o SELECT. (SILVA et al., 2017) De acordo com Silva et al. (2017) algumas de suas características são:

- O banco de dados é armazenado localmente em um arquivo de extensão ".db"
- Suporta a maior parte do SQL 92

[4] É uma formatação utilizada para estruturar dados em formato de texto e transmiti-los de um sistema para outro, como em aplicações cliente-servidor ou em aplicativos móveis

[5] É a linguagem usada para executar comando em bancos de dados relacionais

[6] É a linguagem que permite aos usuários fazer o acesso aos dados ou manipulá-los, conforme modelo de dados apropriado.

- Não oferece integridade referencial (chaves estrangeiras)
- Suporta o uso de transações (COMMIT, ROLLBACK)
- Não deve ser utilizado nos seguintes casos: aplicações de alta concorrência e sistemas ou aplicações *web* de grande porte

(SILVA et al., 2017)

Não há dúvidas quanto a confiabilidade do SQLite, basta verificar que alguns usuários famosos como Facebook, Adobe, Bosch, Google, Apple, Bentley, entre outros, usam o SQLite em alguns de seus produtos. (SQLITE, 2021)

2.6 Considerações Finais

Nesse capítulo foram apresentadas as ferramentas e tecnologias que foram utilizadas para o desenvolvimento do projeto.

Capítulo

3

Estudo da arte

3.1 Considerações iniciais

Neste capítulo serão apresentados trabalhos desenvolvidos por terceiros que, como este projeto, também são relacionados a automação residencial.

3.2 Sistema de automação residencial de baixo custo utilizando o ESP8266

Neste projeto, o grupo formado por Silva, Mendes e Sales (2021), desenvolveram um sistema de automação residencial (domótica)[1] sem fio, utilizando o microcontrolador embarcado de baixo custo ESP8266, conectado ao Wi-Fi, usando uma página HTML programada no mesmo, para a interface do usuário.

No projeto foi utilizado o microcontrolador ESP8266 modelo 07, pela sua grande quantidade de pinos GPIO disponíveis (16 nessa versão), pelo seu baixo preço de aquisição e também por seu tamanho reduzido.

O usuário acessa página HTML gerada pelo ESP8266, efetua *login* e configura seus dispositivos, os cenários e sua rede através da página HTML.

Depois das configurações feitas o usuário poderá usar qualquer dispositivo(como celular, computador, etc) conectado a rede local para controlar quaisquer dispositivos que possam ser acionados por relés (como lâmpadas, motores, ventiladores, etc.) usando apenas um navegador.

Como observado na Figura 3.1 foram utilizados módulos relé no circuito do projeto para o acionamento desses dispositivos.

Já na Figura 3.2 é apresentada a página inicial HTML usada para a interação com o usuário.

[1]É a tecnologia responsável pela gestão de todos os recursos habitacionais. Esse termo nasceu da fusão da palavra Domus, que significa casa, com a palavra Robótica, que está ligada ao ato de automatizar

(SILVA; MENDES; SALES, 2021)

Figura 3.1: Circuito

Fonte: (SILVA; MENDES; SALES, 2021)

Figura 3.2: Página inicial

Fonte: (SILVA; MENDES; SALES, 2021)

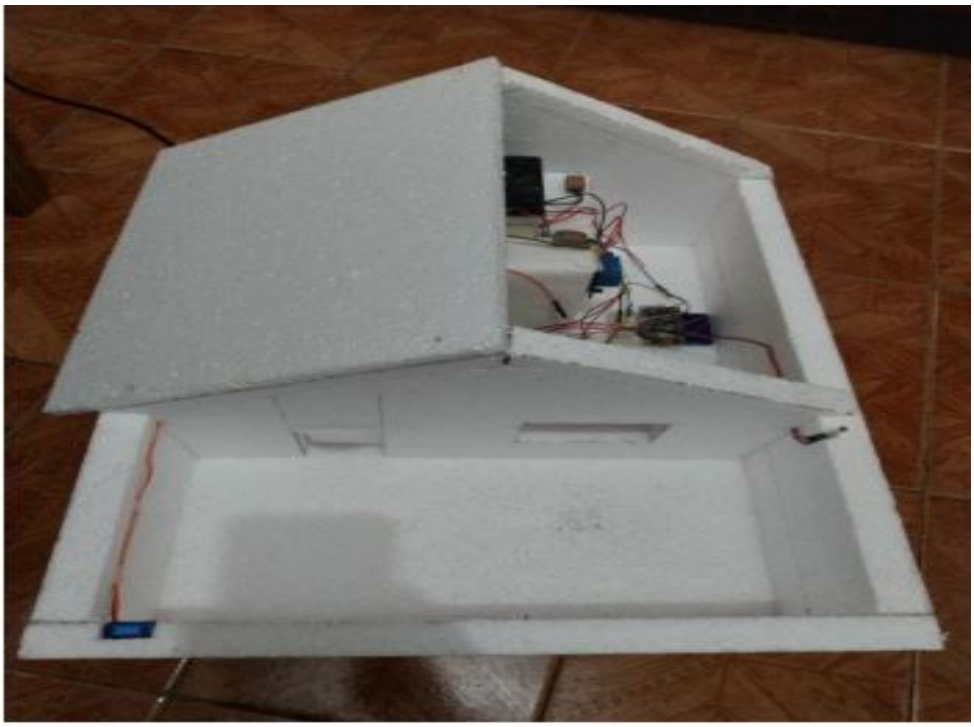
3.3 Automação residencial com NodeMCU

No projeto desenvolvido por Santanna e Calvalcanti (2018) o intuito foi desenvolver um sistema de automação residencial *wireless* que possibilita o controle de todos os dispositivos que estejam configurados e próximos ao microcontrolador ESP8266, com a pretensão de simplificar as rotinas residenciais.

Os autores optaram pela utilização do módulo ESP8266 NodeMCU - ESP12 para realizar a comunicação com os sensores e controladores. Também foram utilizados como sensores o sensor de luminosidade e o de temperatura além de dois servo motores que abrem a porta e o portão. A lampada está ligada a um módulo relé que acionado irá ligá-la. O sensor de temperatura irá acionar uma Pastilha Peltier que tem o objetivo de climatizar uma pequena área da casa. Todos esses dispositivos foram utilizados em uma maquete feita pelos autores para a demostração das funcionalidades como visto na Figura 3.3.

Na placa NodeMCU foi desenvolvido um servidor *Web* em linguagem HTML para a interação com o usuário. Este servidor pode ser acessado por qualquer dispositivo conectado a mesma rede e com acesso a um navegador. Ao acessar o usuário interage com a página HTML vista na Figura 3.4 visualizando a temperatura retornada pelo sensor, além de ter acesso a ligar/desligar a lâmpada e abrir/fechar o portão. (SANTANNA; CALVALCANTI, 2018)

Figura 3.3: Maquete



Fonte: (SANTANNA; CALVALCANTI, 2018)

Figura 3.4: Página HTML

Fonte: (SANTANNA; CALVALCANTI, 2018)

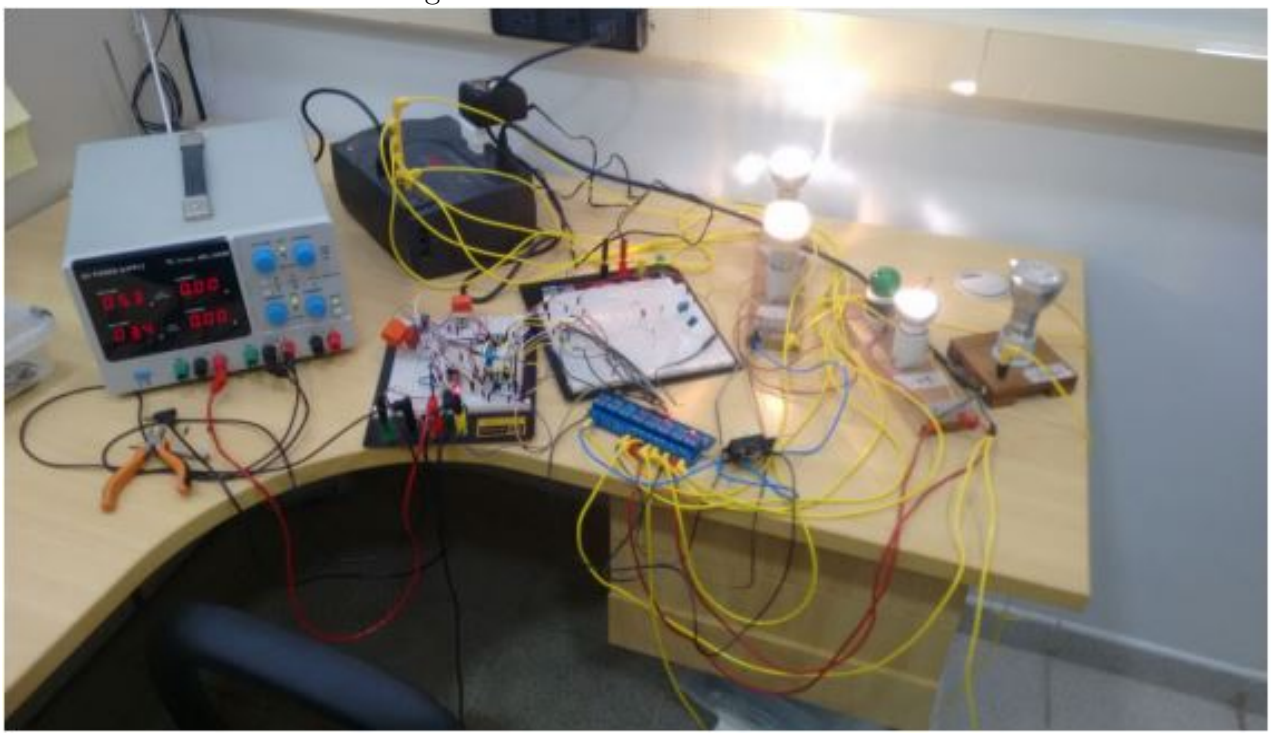
3.4 Uso do microcontrolador ESP8266 para automação residencial

O projeto desenvolvido por Oliveira (2017) foi um sistema de automação residencial com ESP8266, com o intuito principal de diminuir os custos dessa tecnologia, por isso o uso do microcontrolador ESP8266, devido ao seu baixo custo.

No projeto o ESP8266 é utilizado para ligar lâmpadas 110v, e não lâmpadas *led* 5v (como os colocados em maquetes), portanto há a necessidade de se usar módulos relé para acionar as mesmas. Os pinos GPIO's do ESP são ligados aos relés, que estão conectados a um conversor de nível lógico, pois o ESP utiliza apenas 3.3v e o relé é acionado a partir de 5v. O ESP aciona o relé via Wi-Fi que liga/desliga as lâmpadas(lembrando que as lâmpadas precisam de uma tensão de 110v). As lâmpadas além de ligadas ao módulo relé, também estão ligadas a um interruptor, que graças a ligação em paralelo pode acionar as lâmpadas também. O circuito pode ser visto na Figura 3.5.

O ESP está trabalhando em modo servidor, com uma página *Web* sendo exibida ao se conectar com ele via rede Wi-Fi. Ao se conectar, o usuário se depara com botões que conseguem acionar as lâmpadas. Portanto qualquer dispositivo que esteja na mesma rede do microcontrolador consegue controlar as lâmpadas por um navegador acessando a página da Figura 3.6. (OLIVEIRA, 2017)

Figura 3.5: Circuito no Protoboard



Fonte: (OLIVEIRA, 2017)

Figura 3.6: Página HTML

Fonte: (OLIVEIRA, 2017)

3.5 Considerações Finais

Os trabalhos citados neste capítulo foram de grande ajuda para a realização desse projeto, principalmente para verificar os recursos *Web* do microcontrolador ESP8266 e suas funcionalidades como servidor, além de como utilizá-lo na IDE do Arduino.

Capítulo

4

Metodologia

4.1 Considerações Iniciais

Neste capítulo serão apresentadas algumas técnicas e métodos para o desenvolvimento da aplicação móvel se comunicando com o módulo NodeMCU ESP8266 ESP12E, juntamente com os circuitos e diagramas utilizados.

O projeto proposto se trata de um aplicativo móvel desenvolvido em React Native no qual o usuário cadastra os ambientes do local onde será utilizado, além dos dispositivos que deseja acionar pelo aplicativo.

Esses dispositivos deverão estar conectados a uma placa ESP8266, pois ela se conecta a rede Wi-Fi local, onde assim pode receber requisições do aplicativo móvel. A placa ESP8266 interpreta essa requisição e assim aciona o dispositivo solicitado.

Para a demonstração do projeto foi feita uma maquete com alguns dispositivos que serão acionados pelo aplicativo. Mais detalhes serão apresentados nas próximas seções deste capítulo.

4.2 Backend

O *backend* da aplicação foi feito utilizando Node.js. Ele está executando uma API em um servidor na rede local pelo IP 192.168.2.115 na porta 3333, onde a aplicação móvel envia requisições para o servidor node que interpreta e devolve a resposta utilizando JSON.

O Node executa todo o processo não visível ao usuário, ou seja, interpreta os comandos enviados pelo aplicativo realizando as ações de banco de dados como inclusões e alterações e consultas. Também interpreta se o dispositivo está ligado ou desligado e envia comandos para o ESP8266.

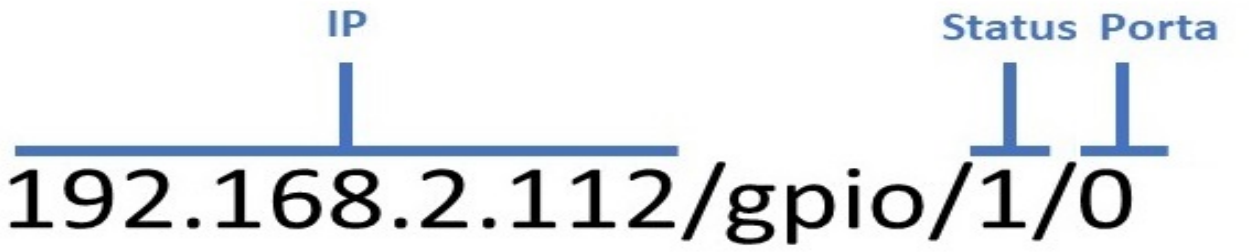
Os comando são enviados ao ESP via protocolo HTTP (*Hypertext Transfer Protocol* - Proto-

colo de Transferência de Hipertexto)[1] por meio de um endereço IP(*Internet Protocol* - protocolo de Internet) [2] do dispositivo cadastrado na tabela de dispositivos do banco de dados.

É enviada para o ESP uma requisição GET [3] com o IP, a palavra 'GPIO', a porta do ESP e o *status* (0 (zero) para desligado e 1 (um) para ligado).

Pode ser visto por exemplo o comando para acionar o *led* da sala na Figura 4.1.

Figura 4.1: Comando backend



Fonte: Próprio Autor

Na estrutura do projeto foi criada uma pasta principal "*src*", e dentro a subpasta "*controlers*" a qual armazena toda a regra de negócio. Nela localizam-se as funções que interpretam e gravam informações no banco de dados ou retornam informações ao aplicativo. Dentro da mesma pasta está a subpasta "*database*", que se localizam as configurações do banco de dados e o próprio arquivo do banco de dados (lembrando que o SQLite salva o banco de dados em um arquivo físico).

Na mesma pasta "*src*" existem 2 arquivos, o "*index.js*" que é onde o *backend* é iniciado, e o arquivo "*routes.js*" que é onde são definidas as rotas da aplicação Node.

4.2.1 Banco de Dados

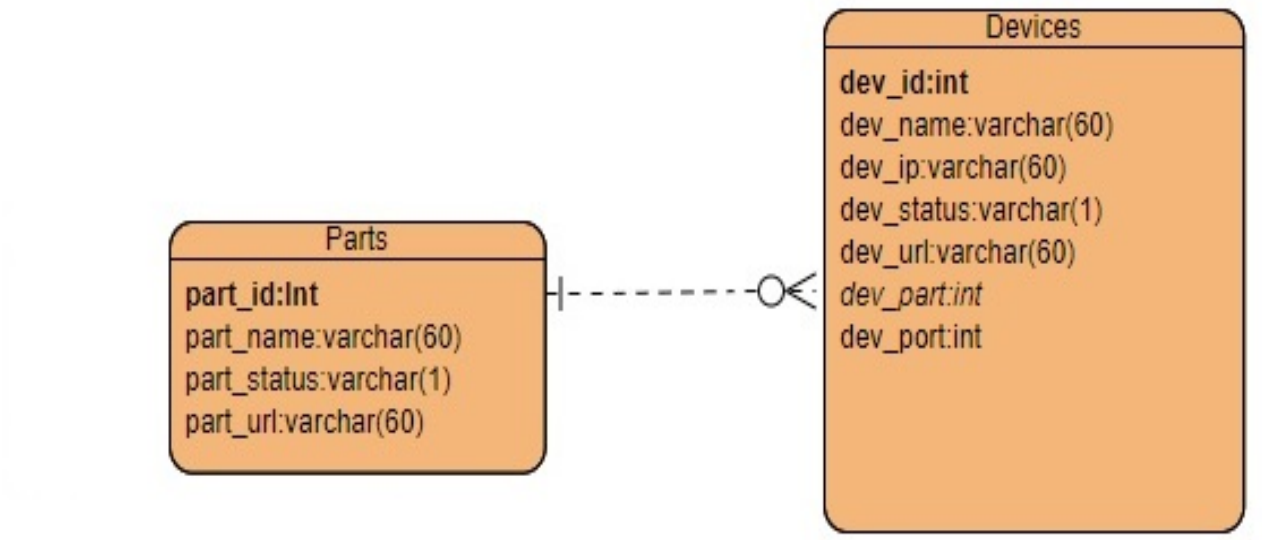
O banco de dados utilizado foi o SQLite, nele foi necessário criar apenas duas tabelas, uma para o cadastro dos ambientes e a outra para o cadastro dos dispositivos. A modelagem do banco de dados é apresentada na Figura 4.2.

[1] é um protocolo de transferência que possibilita que as pessoas que inserem a URL do seu site na Web possam ver os conteúdos e dados que nele existem

[2] é um endereço exclusivo que identifica um dispositivo na Internet ou em uma rede local

[3] requisição em que os parâmetros são passados na própria URL

Figura 4.2: Modelagem Banco de Dados



Fonte: Próprio Autor

Tabela Parts

Como mostrado na Figura 4.2, a tabela *Parts* que seriam os ambientes da casa conta com 4 campos:

- *part_id*: é a chave primária da tabela;
- *part_name*: é o nome do ambiente;
- *part_status*: se preenche com ativo ou inativo, caso não queira que o ambiente seja mostrado;
- *part_url*: de acordo com a url escolhida muda o ícone do botão mostrado;

Tabela Devices

Já a outra tabela *Devices* também mostrada na Figura 4.5 são os dispositivos que serão acionados e possui sete campos:

- *dev_id*: é a chave primária da tabela;
- *dev_name*: é o nome do Dispositivo;
- *dev_ip*: é o IP que o ESP8266 ligado ao dispositivo recebeu ao se conectar a rede;

- *dev_status*: é como o dispositivo se encontra, ligado/desligado;
- *dev_url*: de acordo com a url escolhida muda o ícone do botão mostrado;
- *dev_part*: é uma chave estrangeira da tabela *Parts*, representa qual ambiente esse dispositivo pertence.
- *dev_port*: se preenche com qual porta o dispositivo está cadastrado.

4.3 Aplicativo Mobile

O aplicativo móvel tem a principal função de se comunicar com o ESP8266 via Wi-Fi e enviar comandos a ele. Para que essa comunicação seja realizada, os dispositivos devem estar conectados a mesma rede que o celular com o aplicativo.

Foi escolhido desenvolver um aplicativo móvel no lugar de desenvolver uma página web diretamente no ESP8266, pela facilidade de acesso que se tem hoje aos *smartphones*. Também pelo fato de que assim pode se conectar o aplicativo móvel a vários microcontroladores atraves do seu IP cadastrado.

O usuário acessa o aplicativo, cadastra primeiro seus ambientes e dispositivos. Os ambientes são uma maneira de agrupar os dispositivos, facilitando assim encontrá-los quando precisar acioná-los.

Ao cadastrar o dispositivo o usuário informa o IP do ESP8266 em que o dispositivo está conectado e a porta em que esse ele está ligada. Com essas informações já é possível enviar requisições para que o ESP8266 interprete.

4.3.1 Casos de Uso

Para um melhor entendimento da interação do usuário com o aplicativo móvel, foi desenvolvido um diagrama de casos de uso. Na Figura 4.3 pode-se verificar este caso de uso da aplicação.

Figura 4.3: Casos de uso Aplicativo

Fonte: Próprio autor

Nesse caso de uso o ator é o usuário que deseja acionar seus dispositivos. Ele possui as seguintes funções: Cadastrar Ambiente, Listar Ambientes, Editar Ambientes, Cadastrar Dispositivo, Listar Dispositivos, Editar Dispositivos, Excluir Dispositivo, Mudar Estado do Portão, Mudar Estado da Lâmpada e Mudar Estado do Ventilador.

O caso de uso "Cadastrar Ambiente" está descrito na Tabela 4.1. Neste caso o usuário irá inserir os dados do ambiente no banco de dados.

Tabela 4.1: Cadastrar Ambiente

Descrição	UC01 - Cadastrar Ambiente
Autor	Usuário do aplicativo
Fluxo principal	1 - Acessar o aplicativo. 2 - Escolher a opção Novo ambiente. 3 - Inserir os dados e confirmar.
Fluxo Alternativo	Inserir campos obrigatórios em brancos ou incorretos retorna mensagem de erro.
Precondições	Não se aplica.

O caso de uso "Cadastrar Dispositivo" está descrito na Tabela 4.2. Nele o usuário irá inserir os dados do dispositivo no banco de dados.

Tabela 4.2: Cadastrar Dispositivo

Descrição	UC04 - Cadastrar Dispositivo
Autor	Usuário do aplicativo
Fluxo principal	1 - Acessar o aplicativo. 2 - Escolher a opção Novo Dispositivo. 3 - Inserir os dados e confirmar.
Fluxo Alternativo	Inserir campos obrigatórios em brancos ou incorretos retorna mensagem de erro.
Precondições	Possuir algum ambiente cadastrado anteriormente.

Já o caso de uso "Listar Ambientes" está descrito na Tabela 4.3. Este é o caso que o usuário visualiza a lista de ambientes do banco de dados.

Tabela 4.3: Listar Ambientes

Descrição	UC02 - Listar Ambientes
Autor	Usuário do aplicativo
Fluxo principal	1 - Acessar o aplicativo. 2 - Escolher a opção Listar ambientes. 3 - Visualizar a lista de todos os ambientes gravados no banco de dados.
Fluxo Alternativo	Ao clicar em "Minha casa" irá aparecer a lista apenas com ambientes ativos.
Precondições	Não se aplica.

No caso de uso "Editar ambiente" que está descrito na Tabela 4.4, Neste caso o usuário edita os dados do ambiente no banco de dados. O usuário deverá primeiro listar os ambientes para poder escolher qual ambiente irá alterar.

Tabela 4.4: Editar Ambientes

Descrição	UC03 - Editar Ambientes
Autor	Usuário do aplicativo
Fluxo principal	1 - Acessar o aplicativo. 2 - Escolher a opção Listar ambientes. 3 - Clicar no botão Editar. 4 - Editar os dados e confirmar.
Fluxo Alternativo	Inserir campos obrigatórios em brancos ou incorretos retorna menagem de erro.
Precondições	Possuir algum ambiente cadastrado.

O caso de uso "Listar Dispositivos" está descrito na Tabela 4.5. Este é o caso que o usuário visualiza a lista de dispositivos do banco de dados. Para selecionar a lista onde há um filtro de ambientes, o usuário deverá primeiramente passar pela lista de ambientes, para poder escolher o ambiente a ser filtrado, para listar somente os dispositivos do ambiente selecionado.

Tabela 4.5: Listar Dispositivos

Descrição	UC05 - Listar Dispositivos
Autor	Usuário do aplicativo
Fluxo principal	1 - Acessar o aplicativo. 2 - Escolher a opção Listar Dispositivos. 3 - Visualizar a lista de todos os dispositivos gravados no banco de dados.
Fluxo Alternativo	Clicar em "Minha casa" e escolher o ambiente do dispositivo, a lista aparecerá apenas os dispositivos do ambiente selecionado.
Precondições	Não se aplica.

O caso de uso "Editar Dispositivos" está descrito na Tabela 4.6. Neste caso o usuário edita os dados do dispositivo no banco de dados. O usuário deverá primeiro listar os dispositivos para poder escolher qual dispositivo irá alterar.

Tabela 4.6: Editar Dispositivos

Descrição	UC06 - Editar Dispositivos
Autor	Usuário do aplicativo
Fluxo principal	1 - Acessar o aplicativo. 2 - Escolher a opção Listar Dispositivos. 3 - Clicar no botão Editar. 4 - Editar os dados e confirmar.
Fluxo Alternativo	Inserir campos obrigatórios em brancos ou incorretos retorna mensagem de erro.
Precondições	Ter algum dispositivo cadastrado.

Já o caso de uso "Excluir Dispositivos" está descrito na Tabela 4.7. Neste caso o usuário exclui o registro do dispositivo no banco de dados. O usuário deverá primeiro listar os dispositivos

para poder escolher qual dispositivo irá excluir.

Tabela 4.7: Excluir Dispositivos

Descrição	UC07 - Excluir Dispositivos
Autor	Usuário do aplicativo
Fluxo principal	1 - Acessar o aplicativo. 2 - Escolher a opção Listar ambientes. 3 - Clicar no botão Excluir. 4 - Ao aparecer a pergunta clicar em ”sim”.
Fluxo Alternativo	Ao aparecer a pergunta clicar em ”não”, assim a exclusão é cancelada.
Precondições	Ter algum dispositivo cadastrado.

No caso de uso ”Mudar estado do Portão” que está descrito na Tabela 4.8. Este é o caso em que o usuário consegue abrir/fechar o portão. Antes de mudar o estado do portão, o usuário deve listar os dispositivos para poder selecioná-lo na lista de dispositivos para poder mudar seu estado.

Tabela 4.8: Mudar estado do Portão

Descrição	UC08 - Mudar estado do Portão
Autor	Usuário do aplicativo
Fluxo principal	1 - Acessar o aplicativo 2 - Escolher a opção Minha Casa. 3 - Escolher o ambiente em que o dispositivo esteja. 4 - Clicar no botão do portão.
Fluxo Alternativo	Clicar em minha casa sem ambiente cadastrado
Precondições	1 - Ter cadastrado anteriormente o ambiente e o dispositivo. 2 - O dispositivo estar conectado na mesma rede.

No caso de uso ”Mudar estado da Lâmpada” que está descrito na Tabela 4.9. Este é o caso em que o usuário consegue ligar/desligar a lâmpada. Antes de mudar o estado da lâmpada, o usuário deve listar os dispositivos para poder selecioná-la na lista de dispositivos para poder mudar seu estado.

Tabela 4.9: Mudar estado da Lâmpada

Descrição	UC09 - Mudar estado da Lâmpada
Autor	Usuário do aplicativo
Fluxo principal	1 - Acessar o aplicativo 2 - Escolher a opção Minha Casa. 3 - Escolher o ambiente em que o dispositivo esteja. 4 - Clicar no botão da lâmpada.
Fluxo Alternativo	Clicar em minha casa sem ambiente cadastrado
Precondições	1 - Ter cadastrado anteriormente o ambiente e o dispositivo. 2 - O dispositivo estar conectado na mesma rede.

No caso de uso "Mudar estado do Ventilador" que está descrito na Tabela 4.10. Este é o caso em que o usuário consegue ligar/desligar o ventilador. Antes de mudar o estado do ventilador, o usuário deve listar os dispositivos para poder selecioná-lo na lista de dispositivos para poder mudar seu estado.

Tabela 4.10: Mudar estado da Ventilador

Descrição	UC08 - Mudar estado da Ventilador
Autor	Usuário do aplicativo
Fluxo principal	1 - Acessar o aplicativo 2 - Escolher a opção Minha Casa. 3 - Escolher o ambiente em que o dispositivo esteja. 4 - Clicar no botão do ventilador.
Fluxo Alternativo	Clicar em minha casa sem ambiente cadastrado
Precondições	1 - Ter cadastrado anteriormente o ambiente e o dispositivo. 2 - O dispositivo estar conectado na mesma rede.

4.3.2 Desenvolvimento

O aplicativo foi desenvolvido em React Native, que tem a grande vantagem de gerar códigos tanto pra Android quanto pra iOS de forma nativa.

Foi utilizado o VSCode (Visual Studio Code) [4] como editor de texto para o desenvolvimento.

Para se instalar tanto o React Native, quanto suas dependências e pacotes utilizados, foi utilizado o gerenciador de pacotes NPM. Com o NPM e o expo instalados, ao realizar o comando no terminal "*expo init nomedoprojeto*" já são criadas as pastas e arquivos utilizadas na estrutura do projeto.

O React Native usa todos os arquivos de código na extensão ".js"(semelhantes ao Node.js), já que são todos arquivos Javascript, até mesmo os de estilização. A pasta central se chama "*src*", onde estão os arquivos do projeto. Ela separa os arquivos em três subpastas. A "*assets*" que é a subpasta onde se localiza as imagens e ícones. Na subpasta "*Pages*" localizam-se as telas. E a subpasta "*services*" que é onde se localiza o arquivo de configuração do *backend*. Também existem dois arquivos que não estão em subpastas, o arquivo "*generalStyle.js*" que é uma estilização padrão de todas as telas, e o arquivo "*routes.js*" que é o arquivo onde são configuradas as rotas das telas.

4.4 Materiais

Nessa seção serão mostrados os materiais utilizados na maquete criada para demonstrar como o aplicativo móvel atua em um ambiente IoT. No projeto foram utilizados os seguintes componentes: Módulo Wi-Fi ESP8266 NodeMCU ESP12, fonte de alimentação 5v, Ventilador 5v, *leds* de alto brilho na cor branca, *jumpers*, módulo relé 2 canais e um micro servo 5v. O módulo NodeMCU ESP12 já foi descrito no Capítulo 2 deste projeto.

Quanto aos *leds*, foram ultilizados os de 5mm como mostrado na Figura 4.4.

[4] o Visual Studio Code é um editor de código-fonte desenvolvido pela Microsoft para Windows, Linux e macOS

Figura 4.4: Led alto brilho

Fonte: (VOLTRIZ, 2021)

- Tensão de alimentação: 3 a 3.3V DC;
- Luminosidade: 20.000 MCD;
- Corrente máxima: 20mA;
- Ângulo de abertura: 25°;
- Diâmetro do LED: 5mm;
- Comprimento: 37mm;
- Peso unidade: 0,3g;

Foi usado o módulo relé 2 canais (ilustrado na Figura 4.5) para o auxiliar no acionamento do Ventilador. Com o módulo relé é possível acionar cargas de até 220v, como lâmpadas, motores, etc. Ou seja em um ambiente real, deverão ser usados relés para acionar qualquer um dos dispositivos, pois a carga pra acioná-los seria sempre de 110v ou 220v.

Figura 4.5: Módulo relé 2 canais

Fonte: (FLOP, 2021)

- Tensão de operação: 5 VDC;
- Permite controlar cargas de até 220V AC;
- Corrente nominal: 71,4 mA;
- Pinagem: Normal Aberto, Normal Fechado e Comum;
- Tensão de saída: (28 VDC a 10A) ou (250VAC a 10A) ou (125VAC a 15A);
- Dimensões: 50 mm x 37 mm x 18 mm;
- Peso: 30g;

O relé foi utilizado para ligar o ventilador (Figura 4.6) com tensão de 5v e tamanho 145x15x89mm, que foi usado para exemplificar um ventilador de teto em uma residência.

Figura 4.6: Ventilador 5v

Fonte: (LIVRE, 2021)

Foi usado como motor do portão um micro servo motor SG90 ilustrado na Figura 4.7.

Figura 4.7: Servo

Fonte: (PINOUT, 2020)

- Peso: 9 g;
- Dimensão: 22,2 x 11,8 x 31 mm aprox.;
- Corrente nominal: 71,4 mA;
- Torque de parada: 1,8 kgf · cm;
- Velocidade de operação: 0,1 s / 60 graus;
- Tensão de operação: 4,8 V (5 V);

Figura 4.8: *Power Bank*

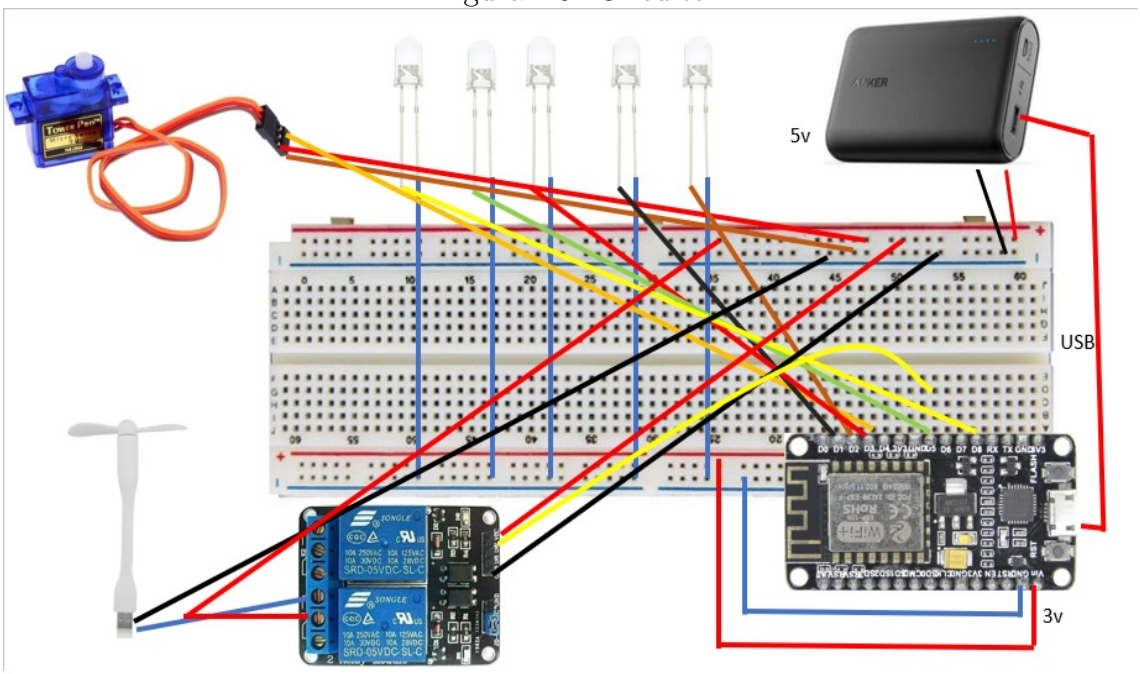
Fonte: Próprio Autor

A Figura 4.8 mostra a *Power Bank*, que foi utilizada como fonte de 5v para alimentar o circuito do projeito.

4.4.1 Circuito

Os materias citados serão ligados ao ESP8266 por um circuito (demonstrado na Figura 4.9), então o *backend* da aplicação envia comandos ao ESP8266 via protocolo HTTP que então os interpreta e aciona os dispositivos da maquete.

Figura 4.9: Circuito



Fonte: Próprio Autor

Custos aproximado dos Materias

Tabela 4.11: Tabela de custos

Material	Custo
Led alto brilho	R$ 2,00
Módulo relé 2 canais	R$ 20,00
Ventilador 5v	R$ 17,00
Protoboard	R$ 20,00
PowerBank	R$ 50,00
Modulo Wi-Fi ESP12e	R$ 55,00
Cabo Usb	R$ 8,00
Servo Motor Tower Pro	R$ 20,00

Como visto no circuito da Figura 4.9, o ESP8266 está ligado a um *protoboard* [5] pelo lado dos seus GPIO de entrada/saída para facilitar que os mesmos sejam ligados aos respectivos dispositivos. Seu pino de 3v e o GND (Malha de terra) estão ligados na trilha de baixo ligando o GND de todos os *leds*. O *led* ligado na porta D3 é o responsável pela luz da Sala, ligado na porta D1 está o *led* da luz do quarto, na D2 o *led* da luz da cozinha, na D5 o *led* da luz da garagem e, por último na D8 o *led* referente a luz do banheiro.

Também nota-se uma bateria de 5v alimentando a trilha de cima com 5v e GND. Ligado nessa trilha encontra-se o VCC (Tensão em corrente contínua.) e o GND do servo motor, o GND do ventilador e o VCC e GND do circuito do módulo relé além do VCC de alimentação

[5]Ou matriz de contato é uma placa com diversos furos e conexões condutoras verticais e horizontais para a montagem de circuitos elétricos experimentais. Seu uso tem a vantagem de dispensar a soldagem

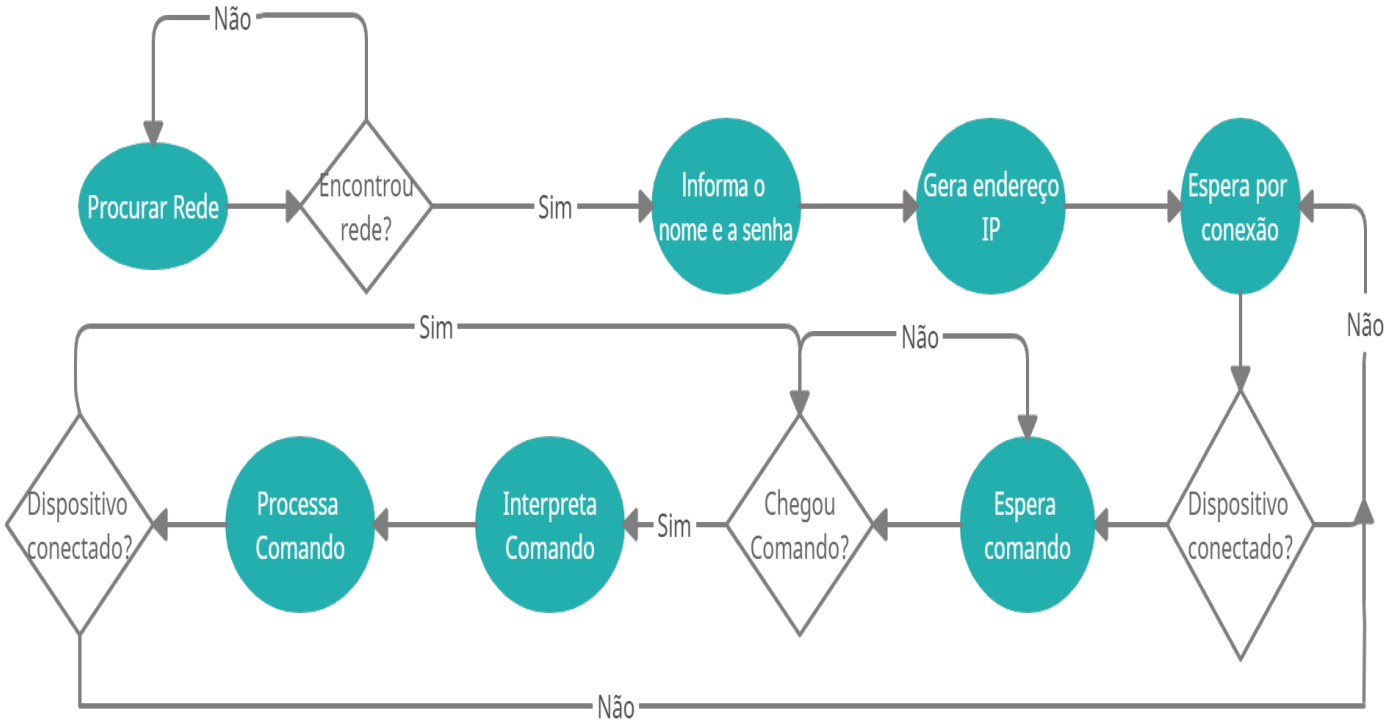
do relé. O servo motor é acionado pela porta D4. Já a porta D7 aciona o módulo relé. Quando acionado o módulo relé, o mesmo aciona o ventilador, por essa razão o 5V do ventilador está ligado na porta de comunicação do módulo relé.

Todos eles estão fixados na maquete em cada ambiente respectivo.

4.5 Algoritimo usado na placa ESP8266

Nesta Figura 4.10, é representado o algoritmo em forma de fluxograma responsável pelo controle dos dispositivos pelo ESP8266.

Figura 4.10: Diagrama do Fluxo do Algoritmo do ESP8266



Fonte: Próprio Autor

Na Figura 4.10 observa-se que o algoritmo se inicia em um *loop* procurando rede e saindo dele somente quando encontra, informando usuário e senha e assim que loga na rede gera um endereço IP para que o usuário se conecte por ele. Posteriormente ele fica em outro *loop* esperando que algum dispositivo se conecte e então permanece esperando por um comando desse dispositivo. Assim que recebe este comando ele interpreta e processa o comando (acionando os dispositivos, como luz, portão, ventilador, etc.) e retorna a parte verifica se o dispositivo está conectado.

4.5.1 Declaração de bibliotecas e variáveis

O código do ESP8266 foi escrito pela IDE do Arduino. Na IDE a primeira coisa a se fazer é importar as bibliotecas necessárias para a execução do código. A principal biblioteca para a comunicação do ESP8266 com a rede Wi-Fi é a ESP8266WiFi.h.

ESP8266WiFi.h é uma biblioteca *open source* criada para a integração do Wi-Fi do ESP. Responsável por realizar a configuração da placa como cliente, servidor e cliente/servidor.

Também foi utilizada a biblioteca "*Servo.h*", que é a biblioteca que permite que o módulo controle os servomotores, permitindo que sejam definidos ângulos entre 0 e 180 graus.

A biblioteca "*Crescer.h*" desenvolvida pela Engenharia (2021a) também foi inserida no código para utilizar a sua função de temporizador. Essa função foi utilizada na parte onde se busca uma requisição do cliente, para evitar de ficar preso na requisição por mais de cinco segundos.

Foram criadas variáveis do tipo "*Int*" para cada um dos dispositivos da maquete, para que possa visualizar qual porta GPIO do ESP foi utilizada para cada um deles no código.

4.5.2 Setup

O ESP8266 foi configurado no projeto como servidor, pois recebe comandos do cliente e interpreta para que possa acionar os dispositivos. No trecho de código da Figura 4.11 podemos verificar onde o ESP se comunica ao Wi-Fi pela função "*Wifi.begin(ssid, password)*", utilizando as variáveis "*ssid*" e "*password*" sendo elas o nome e a senha respectivamente. Também configura o IP pela função "*WiFi.config(ip,ip1,ip2)*". Sendo essas variáveis abastecidas com as configurações do roteador.

A vantagem de se configurar o IP no código, é que caso exista mais de um ESP8266 na mesma rede, pode-se configurar um IP diferente para cada ESP, evitando assim um conflito de IP na rede.

Figura 4.11: Código *Setup*

```
 //conecta no wifi
 WiFi.begin(ssid, password);

//aguarda o sucesso da conexao
 while(WiFi.status()!= WL_CONNECTED){
   delay(500);
   Serial.println(".");
 }

//Setup depois da conexao
 WiFi.config(ip, ip1, ip2);
 Serial.println("");
 Serial.println("WiFi Conectado!");

//iniciando o ESP como servidor
server.begin();
Serial.println("Servidor iniciado");

//Imprime o IP Conectado
Serial.println(WiFi.localIP());
```

Fonte: Próprio Autor

Além da configuração Wi-Fi do ESP8266, também é configurado no "*Setup*" todas as portas citadas no circuito como "*OUTPUT*", ou seja serão portas de saída. E iniciando todas as portas com o estado zero (Ou seja desligado), exceto a porta do módulo relé que deverá ser invertida, pois o relé com estado zero aciona o dispositivo.

4.5.3 Funções Portão

Fora do Setup foram criadas duas funções referentes ao servo motor, uma para abrir e outra pra fechar o portão, vista no código da Figura 4.12.

Figura 4.12: Código Portão

```
  void fechar(){
  for (pos = 0; pos <= 90; pos += 1) { //funcao que fecha o portao
    myservo.write(pos);
    delay(15);
  }
}

void abrir(){
  for (pos = 90; pos >= 0; pos -= 1) { //funcao que abre o portao
    myservo.write(pos);
    delay(1 5);
  }
}
```

Fonte: Próprio Autor

O portão é controlado pela variável "*pos*". Para fechar o portão ela é incrementada a cada quinze milisegundos de zero até chegar nos noventa. E para abrir, a variável é decrementada de noventa a zero. Ou seja o portão chega somente a um ângulo de noventa graus.

4.5.4 Loop

O procedimento "*loop*" é onde se concentram as regras de decisão. O código verifica se foi enviada alguma requisição pelo cliente, caso não, ele continua em um loop até chegar alguma requisição.

Caso encontre alguma comunicação ele tem cinco segundos para receber essa mensagem, caso contrário volta ao início para verificar novamente o cliente. Essa verificação é feita pela variável "*tempCom*".

Nota-se no código da Figura 4.13 que a variável "*req*" armazena a requisição enviada pelo usuário. De acordo com a requisição, o algoritimo busca partes dessa dela e armazena em outras variáveis a porta e o *status*. Também verifica se foram enviados os caracteres "/gpio/" logo após o IP. Caso não tenha sido enviado, retorna requisição inválida e volta ao início do *loop*.

Figura 4.13: Código Loop 01

```
  //le e armazena o que foi enviado pelo cliente
  String req = client.readStringUntil('\r');
  Serial.println(req);
  client.flush();

  //buscando porta de acordo com a req
  String aux = req.substring(12,14);
  Serial.print("PortaAux: ");
  Serial.println(aux);
  port = atoi(aux.c_str());
  Serial.print("Porta: ");
  Serial.println(port);

  //buscando o status de acordo com a req
  aux = req.substring(10,11);
  Serial.print("Status: ");
  int val = atoi(aux.c_str());
  Serial.println(val);

  //buscando o gpio de acordo com o req
  aux = req.substring(4,10);
  Serial.print("End: ");
  Serial.println(aux);

  //caso nao tenha enviado /gpio/ nao executa o comando
  if(aux != "/gpio/"){
    Serial.println("requisicao invalida");
    client.stop();
    return;
   }
}
```

Fonte: Próprio Autor

Logo após essa verificação e de armazenar o *status* e a porta, é feita outra verificação nessa porta para a tomada de decisão de qual dispositivo acionar. Visto no código da Figura 4.14

Figura 4.14: Código Loop 02

```
  //Verifica qual dispositivo acionar de acordo com a porta
  if (port != -1){
      if (port == portao){
          if (val == 0){
            fechar();
          }else{
            abrir();
          }
      }else if(port == vent) {
          digitalWrite(port,!val);
      }
      else{
        digitalWrite(port,val);
      }
  }
}
```

Fonte: Próprio Autor

É verificado se a porta é referente ao portão, caso positivo se verifica a variável "*val*" onde é armazenado o *status*. Caso o valor for zero irá chamar a função que fecha o portão, caso contrário irá executar a função de abrir.

Também é verificado se a porta é referente ao ventilador, caso positivo tem que se inverter a variável "*val*", pois como dito antes o ventilador é acionado pelo relé que tem estado invertido.

Caso nenhuma das condições anteriormente citadas forem atendidas, a condição a ser atendida é para o *led*, então basta enviar o *status* e a porta para a função "*digitalWrite*" para ligá-lo ou desligá-lo.

Feito isso o o algoritmo retorna para o início do *loop* para aguardar outra requisição do cliente.

4.6 Considerações Finais

Este capítulo abordou o planejamento e metodologia utilizada para realizar o desenvolvimento do aplicativo e do algoritmo do ESP8266, utilizando diagramas e imagens para visualizar, modelar e documentar o projeto da aplicação. Com a conclusão deste planejamento se tornou mais simples e fácil o processo de desenvolvimento prático.

Capítulo

5

Resultados

5.1 Considerações Iniciais

Este capítulo apresentará os resultados obtidos com o desenvolvimento do projeto, exibindo imagens das telas do aplicativo móvel e também imagens da maquete usada para demostrar o projeto funcionando.

Os resultados foram satisfatórios, o aplicativo móvel se comunica com a placa ESP8266 sem interferências e com ótimo desempenho para o acionamento dos dispositivos.

5.2 Telas do Aplicativo

O aplicativo não necessitou de uma tela de *login*, visto que para o funcionamento do mesmo ele tenha que estar logado na mesma rede do ESP8266, com isso o *login* é o próprio usuário e senha do roteador Wi-Fi. Diante disso a tela inicial é apresentada na Figura 5.1.

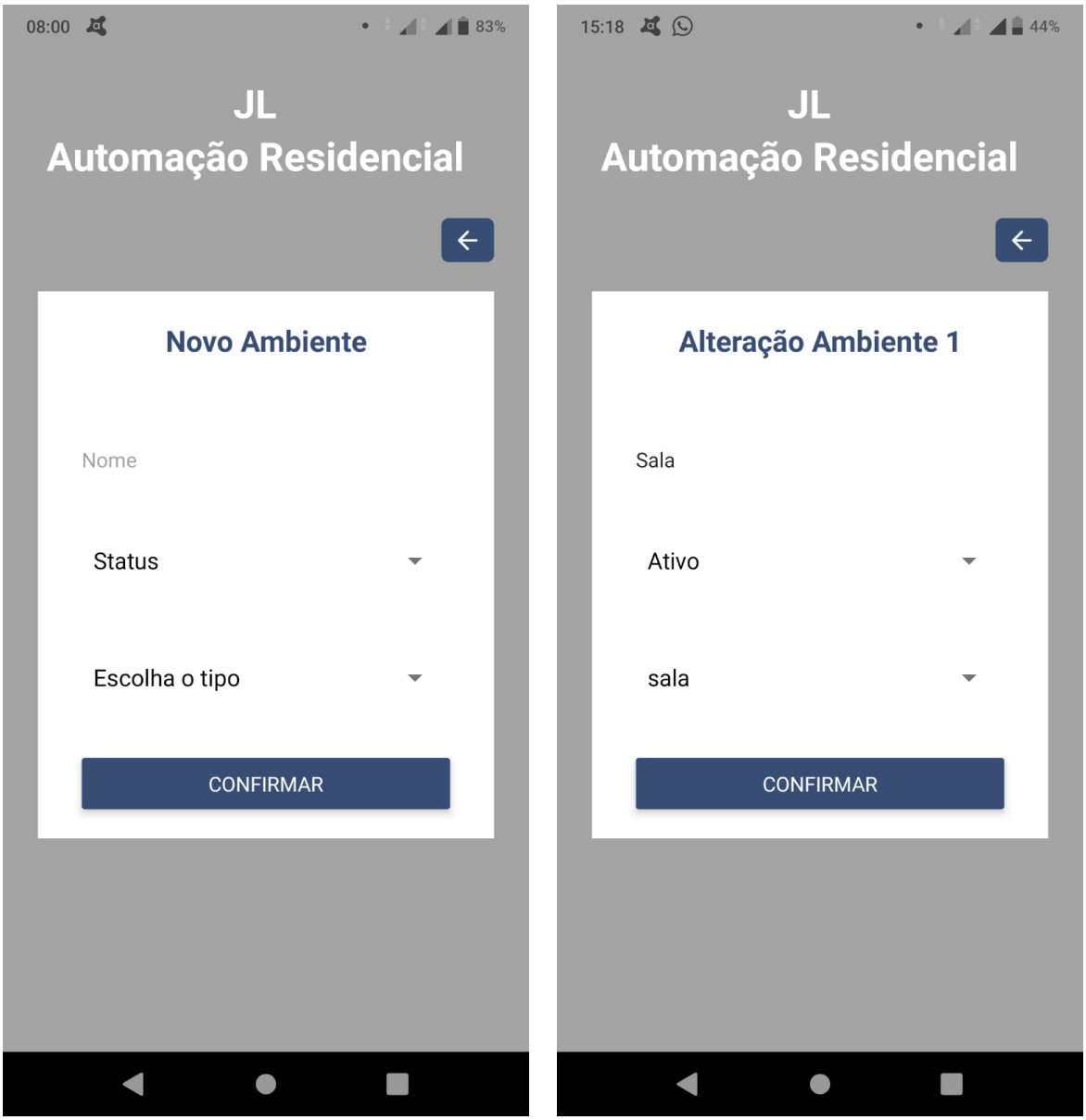
Figura 5.1: Tela inicial do aplicativo

A tela inicial do aplicativo direciona para as demais funções do aplicativo. Nela encontram-se 4 botões menores, na parte superior, que levam para outras telas.

Os botões ”Novo Ambiente” e ”Novo Dispositivo” levam para a tela onde se cadastram novos ambientes e dispositivos, mostrados nas Figuras 5.2(a) e 5.3(a) respectivamente.

Já os botões ”Listar Ambiente” e ”Listar Dispositivo” levam para a tela de listagem dos mesmos, e são mostrados nas Figuras 5.4(a) e 5.5(a) respectivamente.

Além do mais, há um botão central ”Minha Casa” que leva para outra tela de escolha de ambientes que pode ser vista na Figura 5.4(b), onde se filtra somente os ambientes com *status* ativo.



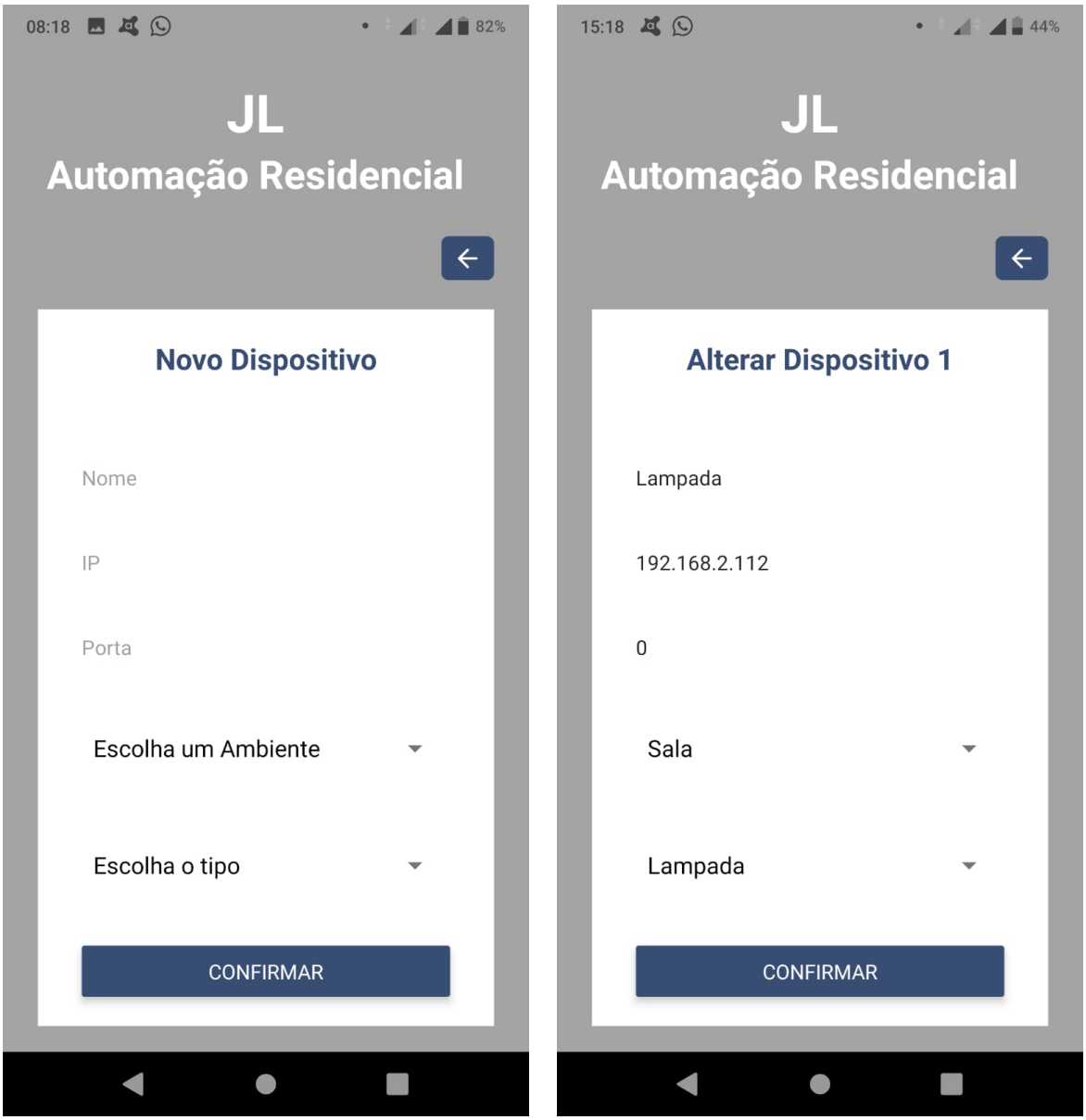
(a) Tela Novo Ambiente

(b) Tela Alteração Ambiente

Figura 5.2: Telas de Inserir e Editar Ambientes

Na Figura 5.2(a) mostra a tela onde se cadastra o ambiente. Ela implementa o caso de uso descrito na Tabela 4.1. Nesta tela o usuário preenche o nome, o *status* (será ativo ou inativo) e o tipo é uma caixa de seleção com opções como sala, cozinha, etc. Ao confirmar o ambiente é gravado no banco de dados.

A tela de "Editar Ambientes" pode ser vista na Figura 5.2(b) e foi descrita na Tabela 4.4. Ela é bem semelhante a tela de "Novo Ambiente", porém mostra o "*ID*" do ambiente logo após o nome da tela. Ela já inicia com as informações do ambiente escolhido preenchidas. Após todas as alterações realizadas, a operação é concluída clicando em Confirmar.



(a) Tela Novo Dispositivo

(b) Tela Alteração Dispositivo

Figura 5.3: Telas de Inserir e Editar Dispositivos

A Figura 5.3(a) ilustra o Cadastro de Dispositivos, que é descrito no caso de uso da Tabela 4.2. O usuário preenche os dados do dispositivo como nome, IP (O IP que o ESP8266 utilizou ao se conectar ao Wi-Fi), a porta (porta que o dispositivo está conectada no ESP8266), qual ambiente está localizado e o tipo do dispositivo (lampada, portão, etc).

A tela de Editar Dispositivos pode ser vista na Figura 5.3(b) e teve seu caso de uso descrito na Tabela 4.6. Ela é semelhante a tela de "Novo Dispositivo", porém mostra o "*ID*" do dispositivo logo após o nome da tela. Ela já inicia com as informações do dispositivo escolhido preenchidas. A operação é concluída clicando em Confirmar.

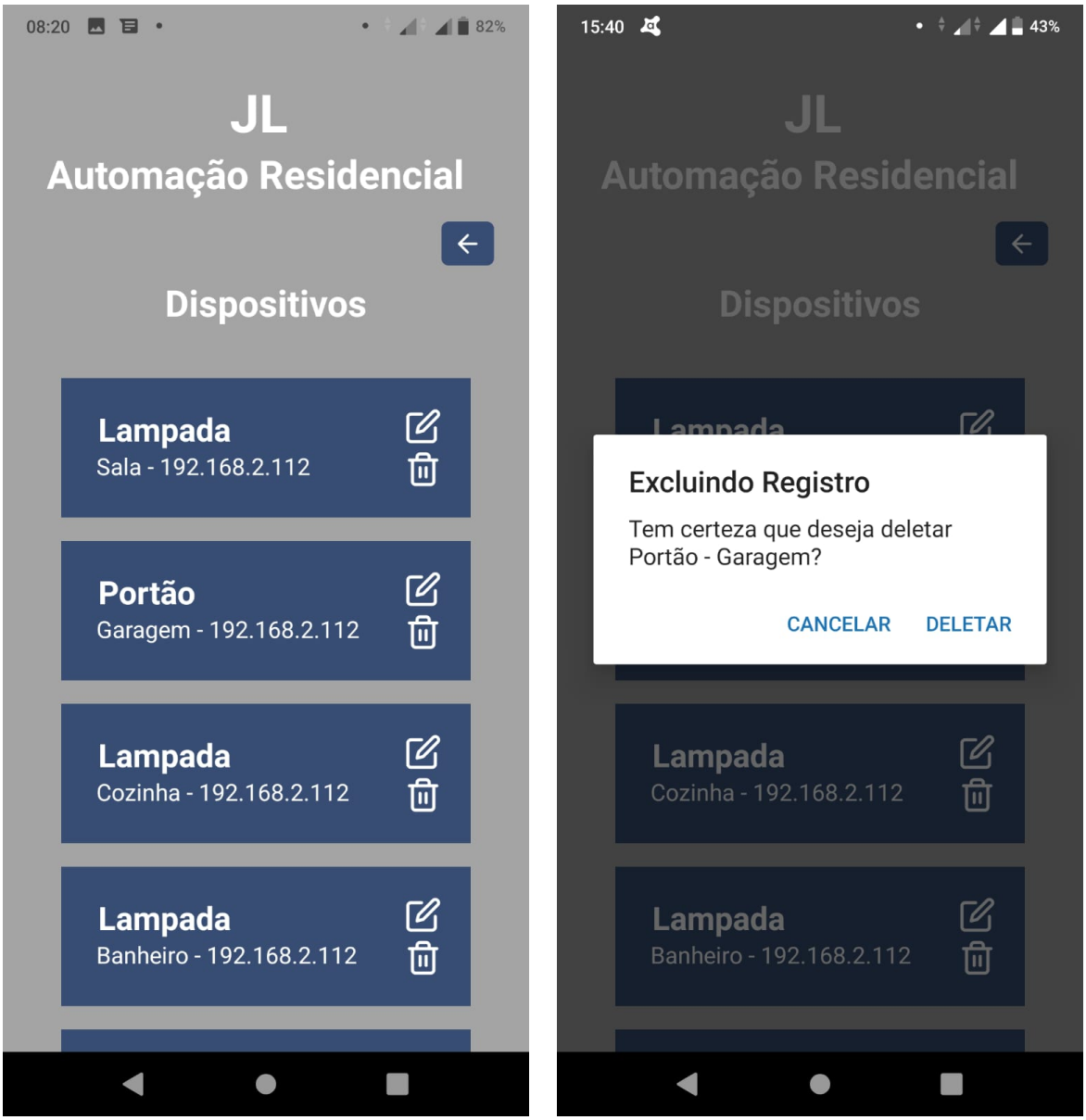
(a) Listagem de Ambientes

(b) Minha Casa

Figura 5.4: Telas de Listar Ambientes e Minha Casa

Na Figura 5.4(a) é mostrada a Listagem de Ambientes. Ela é descrita na Tabela 4.3 do caso de uso. A tela possui a listagem dos ambientes mostrando apenas o nome, e ao lado um botão que irá direcionar para a tela de Edição dos Ambientes.

Ao clicar no botão "Minha Casa" da tela inicial o usuário é direcionado para a tela que mostra os ambientes ativos da casa, ilustrada na Figura 5.4(b). Ela também é descrita pela Tabela 4.3 do caso de uso, pois também se trata de uma lista de ambientes. Esta tela mostra todos os ambientes com *status* "Ativos" da casa. O propósito é que ao clicar em algum ambiente, o usuário é direcionado para outra tela com todos os dispositivos daquele ambiente. De acordo com o tipo do ambiente os ícones dos botões são modificados.

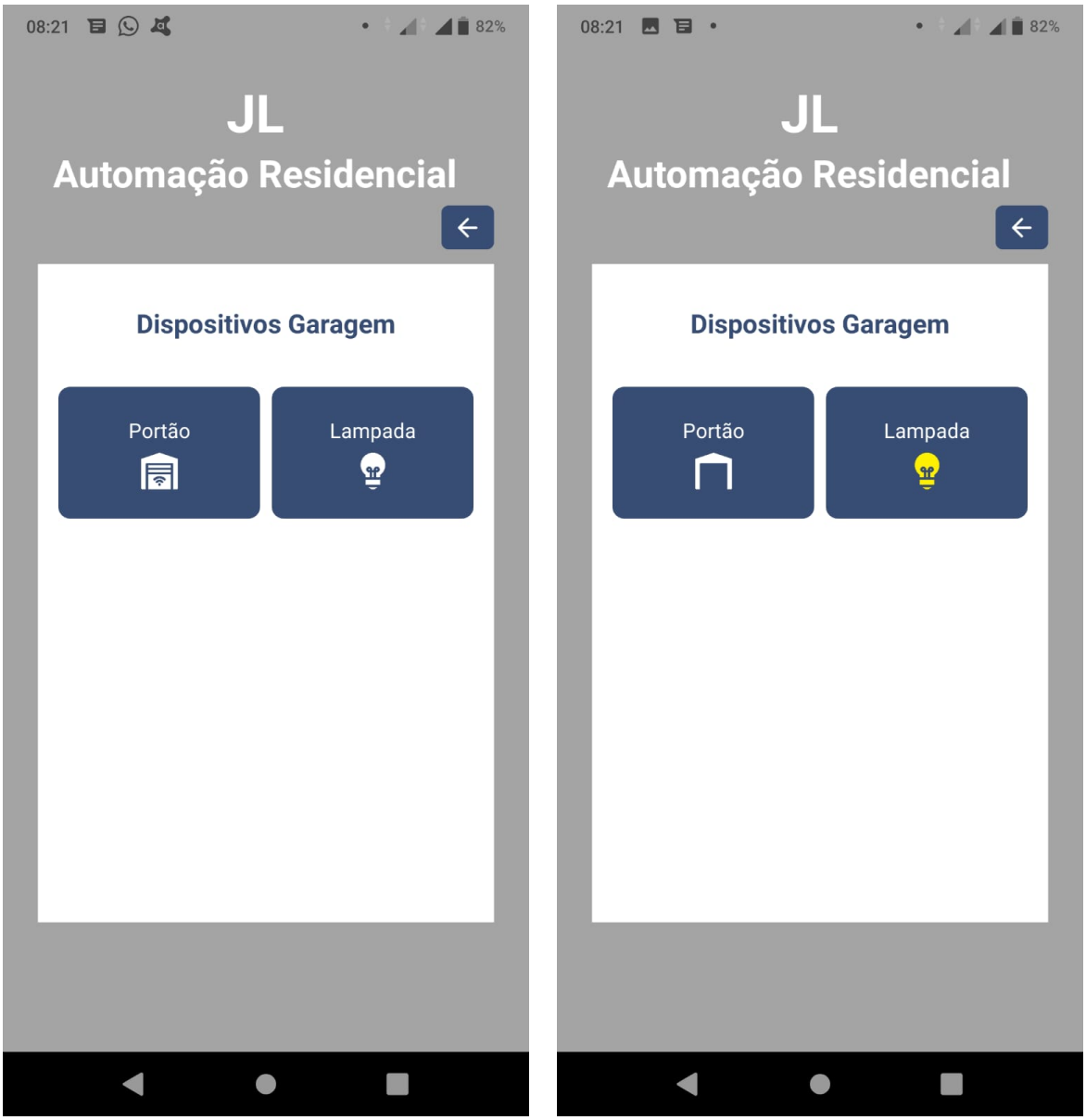


(a) Listagem de Dispositivos (b) Excluir Dispositivo

Figura 5.5: Telas de Listar e Excluir Dispositivos

A Figura 5.5(a) ilustra a tela de listagem de dispositivos que foi descrita no caso de uso da Tabela 4.5. Nesta tela são exibidas as informações do dispositivo, como o nome, o ambiente em que ele se encontra e o IP do ESP8266 que ele está conectado. Além de dois botões ao lado direito de cada dispositivo que é usado para editar e excluir o dispositivo.

Como já citado, para excluir dispositivos basta ir a tela de listagem de dispositivos e clicar no ícone de ”lixeira” (ao lado direito do IP do dispositivo). Ao clicar irá aparecer uma mensagem (vista na Figura 5.5(b)). O usuário poderá clicar em ”Deletar” para excluir o registro, ou ”Cancelar” para interromper a operação. Essa exclusão foi descrita no caso de uso da Tabela 4.7.



(a) Dispositivos do Ambiente (b) Dispositivos do Ambiente (Ligados)

Figura 5.6: Telas de mudança de estados

Na tela ilustrada na Figura 5.6(a), são exibidos todos os dispositivos do ambiente escolhido (a garagem no caso do exemplo da Figura 5.6(a)). Esta tela foi descrita na Tabela 4.5 do caso de uso. Ela é uma listagem de dispositivos, porém com um filtro de ambiente.

Os ícones são modificados de acordo com o tipo do dispositivo e com o seu *status*. Ao clicar em algum dispositivo o usuário pode acioná-lo (desde que ele esteja configurado no ESP8266). Ao acioná-lo o ícone do botão é alterado, como visto na Figura 5.6(b).

Na Figura 5.6(b) nota-se a mesma tela da Figura 5.6(a). Porém os dois dispositivos estão com o *status* ligado (ou seja preenchidos com valor 1 o campo *dev_status* no banco de dados). Sendo assim quando o dispositivo é ligado/desligado, ou no caso do portão, aberto/fechado, o ícone do botão muda para que o usuário visualize no próprio aplicativo como o estado do seu dispositivo (lampada, portão, etc.).

Ao clicar no dispositivo, o usuário modifica o estado do mesmo. Esta ação é descrita nos

casos de uso das Tabelas 4.8, 4.9 e 4.10.

5.3 Maquete

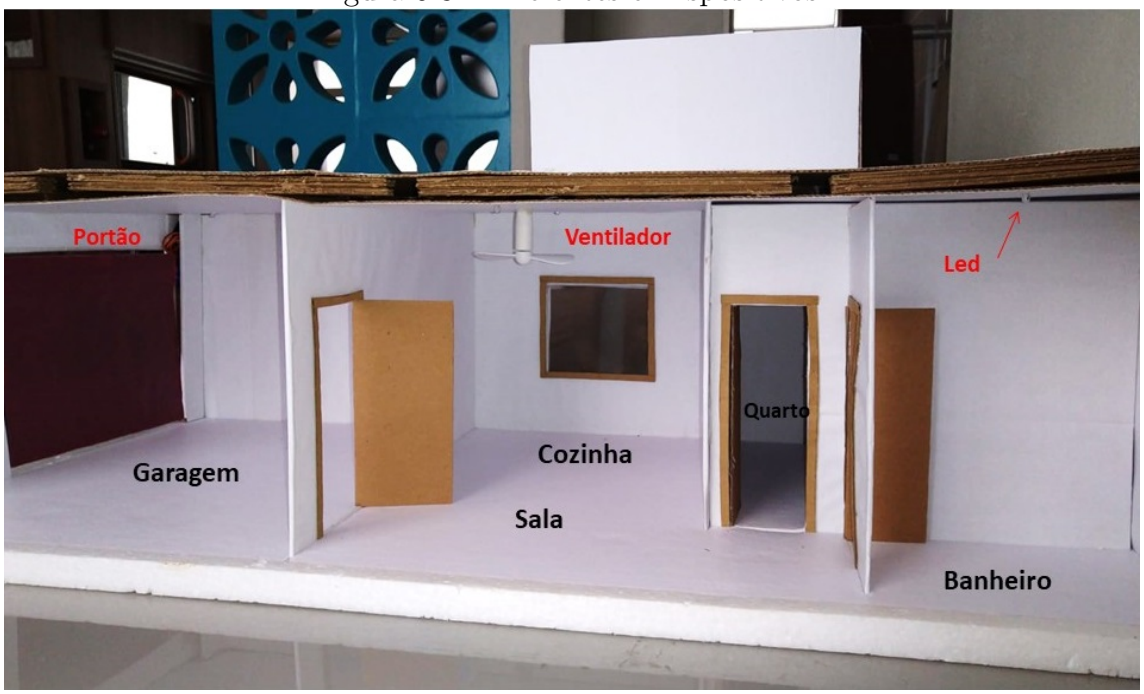
Para demonstração do projeto foi criada uma maquete. Como visto na Figura 5.7 a maquete representa uma casa, foi feita com o chão de isopor e as paredes, portas e portão de papelão. Em cima da casa existe uma caixa, nela está o ESP8266 juntamente com o *protoboard* e a bateria de 5V, e os fios que ligam os dispositivos passam entre o teto da casa para não ficarem expostos.

Figura 5.7: Maquete

Fonte: Próprio Autor

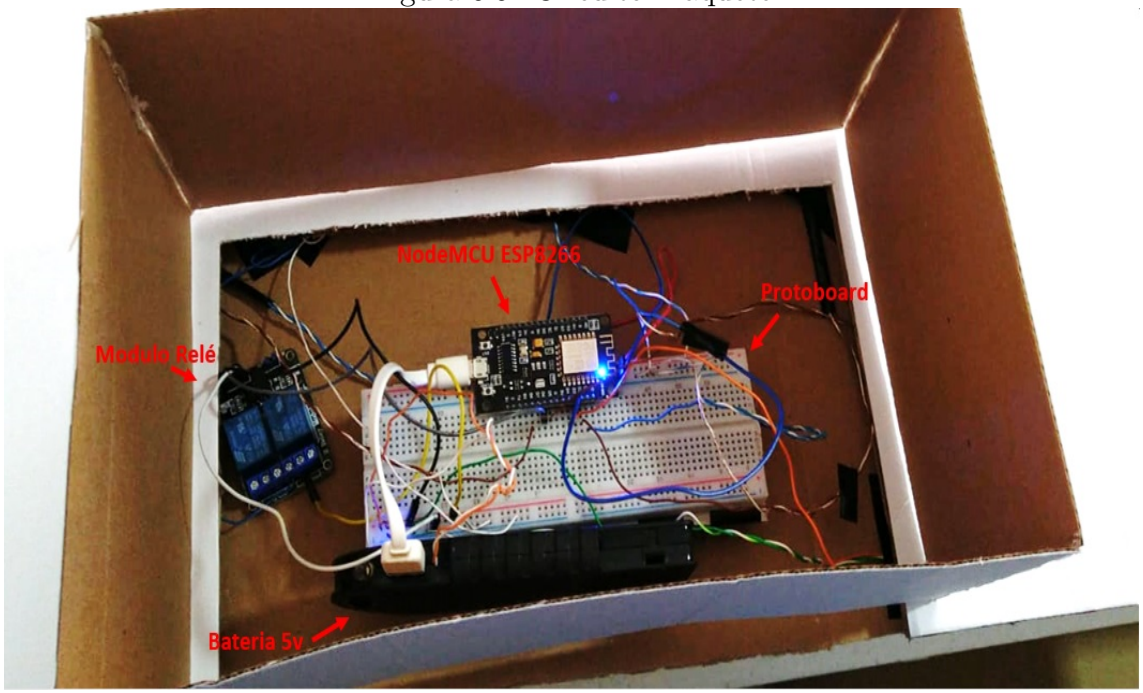
Nota-se na Figura 5.8 a frente da maquete. A casa foi feita com a frente sem paredes para facilitar a apresentação dos dispositivos. A maquete possui 5 ambientes. Todos eles possuem *leds* que serão acionados. Na garagem o portão também será acionado, e na sala além do *led* será acionado também o ventilador.

Figura 5.8: Ambientes e Dispositivos



Fonte: Próprio Autor

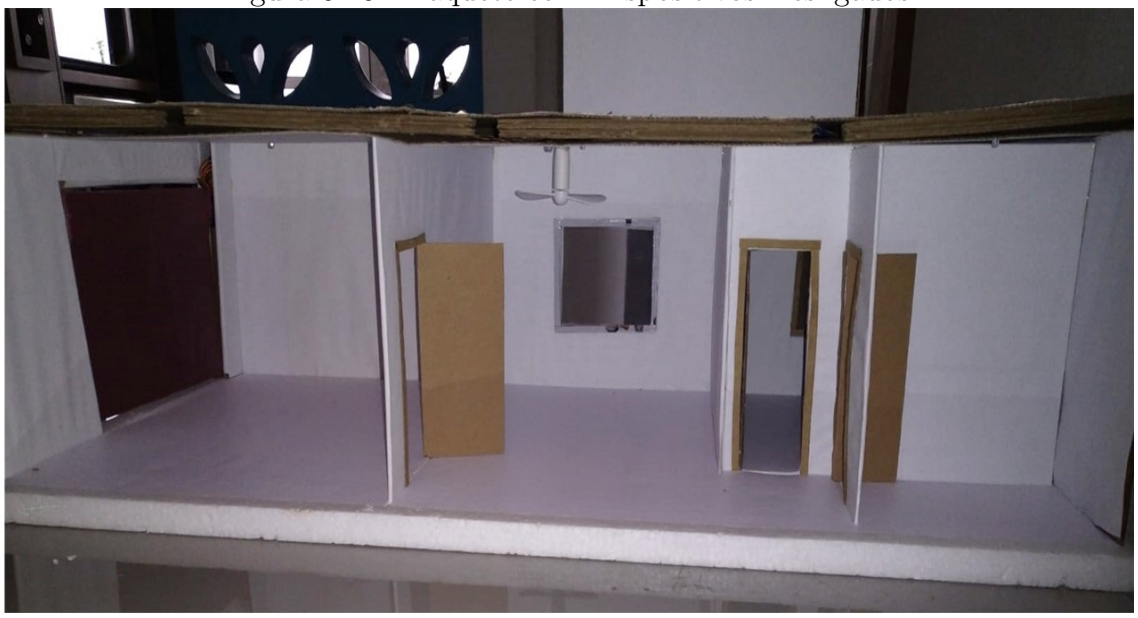
Figura 5.9: Circuito Maquete



Fonte: Próprio Autor

Na Figura 5.9 pode-se ver o circuito demonstrado no Capítulo 4. Como dito anteriormente, este circuito se localiza na caixa logo acima do teto da maquete. Os fios deslocam-se através do interior do teto, saindo do *protoboard* para os dispositivos da casa. Todos os fios estão ligados ao ESP8266 pelo *protoboard*. A bateria de 5v é a fonte de alimentação do circuito.

Figura 5.10: Maquete com Dispositivos Desligados



Fonte: Próprio Autor

Figura 5.11: Maquete com Dispositivos Ligados

Fonte: Próprio Autor

A Figura 5.10 apresenta a maquete com todos os dispostitivos desligados. Já na Figura 5.11 se nota o portão aberto, os *leds* da garagem, da cozinha e do banheiro acesos e o ventilador ligado, todos acionados pelo aplicativo.

5.4 Considerações Finais

Este capítulo apresentou os resultados do aplicativo móvel e da maquete. O resultado foi satisfatório, tudo testado e todos os dispositivos foram acionados pelo celular através do aplicativo.

Capítulo

6

Conclusões

6.1 Considerações Finais

Nota-se o surgimento de várias aplicações contendo, ou baseadas, na família ESP8266. Entre as várias áreas de aplicação, a maior parte envolve automação residencial que, encaixa-se diretamente com o motivo pelo qual os equipamentos ESP8266 estão sendo fabricados, neste caso, aplicações IoT. O projeto proposto se encaixa na mesma visão, ao demonstrar o funcionamento da placa NodeMCU ESP8266, sendo utilizada para automação via Intranet. É visível que se fazem necessários conhecimentos prévios de algumas linguagens de programação, por exemplo JavaScript e a linguagem de programação da IDE Arduino, que é derivação da linguagem C.

Também se torna necessário o conhecimento em circuitos, microcontroladores e atuadores. A demonstração do projeto foi realizada em um circuito de 5v por ser demonstrado em uma maquete. Caso seja utilizado em uma residência real, os atuadores da mesma seriam alimentados com 110v ou 220v, portando seria nescessário um módulo relé para acionar todos os dispositivos nesta voltagem.

Sendo assim, o projeto se encerra com o objetivo alcançado, pois foi possível testar a funcionalidade da placa NodeMCU, bem como criar um aplicativo móvel se comunicando com a mesma. Como existem várias formas de aplicação para cada uma das várias áreas em que se pode utilizar o ESP8266, compreende-se que este projeto é apenas uma demonstração, dentre muitas possibilidades que esta tecnologia pode oferecer.

6.2 Trabalhos Futuros

Como abordado na seção anterior, há uma infinidade de formas de aplicação para a automação residencial com o microcontrolador ESP8266. Uma maneira de agregar funcionalidades a esse projeto é o desenvolvimento de alguns recursos do ESP8266, em que ele busque informações

de alguns sensores, processe essas informações e envie comandos ou mensagens ao aplicativo móvel, como por exemplo:

- Utilizar um sensor de temperatura para ler a temperatura do ambiente e mostrar no aplicativo.
- Usar um sensor de movimento para acionar um alarme e enviar notificações para o celular.
- Utilizar um sensor de umidade para ler a umidade do solo, e caso a umidade estiver abaixo do normal, enviar uma notificação para o aplicativo que o sistema de irrigação foi acionado.
- Acrescentar uma assistente Virtual, para que o usuário consiga acionar os dispositivos da maquete por comando de voz.

Referências Bibliográficas

ARDUINO, G. C. e D. *Sobre Arduino*. 2021. Disponível em: <https://www.arduino.cc/en-/Main/AboutUs>.

CASAVELLA, E. *O que é Linguagem C?* 2018. Artigo. Disponível em: <http://linguagemc-.com.br/o-que-e-linguagem-c/>.

DANIELSSON, W. React native application development - a comparison between native android and react native. *Article*, 2016. Disponível em: <https://www.diva-portal.org/smash-/get/diva2:998793/FULLTEXT02.pdf>.

DIMES, T.; TORRES, P. *JavaScript: Um Guia para Aprender a Linguagem de Programação JavaScript*. Babelcube Incorporated, 2015. ISBN 9781507124048. Disponível em: <https:/-/books.google.com.br/books?id=WzbfCgAAQBAJ>.

ENGENHARIA, C. 2021. Disponível em: <https://www.crescerengenharia.com/>.

ENGENHARIA, T. *Automação residencial: quais os tipos e como funciona?* 2021. Disponível em: <https://thorusengenharia.com.br/automacao-residencial-quais-os-tipos-e-como-funciona-/>.

EXPO, G. C. e D. *Documentação Expo*. 2021. Disponível em: <https://docs.expo.dev/>.

FILHO, M. F. *Internet das Coisas (Internet of Things)*. [S.l.: s.n.], 2016. ISBN 9788550601113.

FLOP, F. 2021. Disponível em: <https://www.filipeflop.com/produto/modulo-rele-5v-2-canais/>.

JUNIOR, M. A. C.; VIRTUOSO, G. H. F.; MARTINS, P. J. *Propriedades Desejáveis a uma Linguagem de Programação: Uma Análise Comparativa entre as Linguagens C, C++ e Java*. 2012. Disponível em: <http://periodicos.unesc.net/sulcomp/article/view/796>.

LIVRE, M. 2021. Disponível em: <https://produto.mercadolivre.com-.br/MLB-1659123359-mini-ventilador-portatil-haste-cabo-flexivel-usb-5v-1w-_JM?searchVariation=64191560735#searchVariation=64191560735&position=17&search_layout=stack&ty
b4b1-4e1c-a77f-cde34ee6ae2a>.

MAGRANI, E. *A internet das coisas*. Editora FGV, 2018. ISBN 9788522520060. Disponível em: <https://books.google.com.br/books?id=qYtlDwAAQBAJ>.

MEIRA, S. *Sinais do futuro imediato, internet das coisas*. 2016. Artigo. Disponível em: <https://silvio.meira.com/silvio/sinais-do-futuro-imediato-1-internet-das-coisas/>.

MURATORI, J. R.; Bó, P. H. D. Automação residencial: histórico, definições e conceitos. *Article*, 2015. Disponível em: <http://www.osetoreletrico.com.br/wp-content/uploads/2011-/04/Ed62_fasc_automacao_capI.pdf>.

NATIVE, R. *React Native Aprenda uma vez, escreva em qualquer lugar.* 2021. Disponível em: <http://reactnative.dev/>.

NEVILLE, A. *Tecnologia do Concreto - 2ed*. Bookman Editora, 2013. ISBN 9788582600726. Disponível em: <https://books.google.com.br/books?id=cqY5AgAAQBAJ>.

NODEJS, G. criador e D. 2021. Disponível em: <https://nodejs.org/en/about/>.

OLIVEIRA, G. *Descomplicando a pinagem do NodeMCU*. 2021. Disponível em: <https:-//blogmasterwalkershop.com.br/embarcados/esp8266/descomplicando-a-pinagem-do-nodemcu>.

OLIVEIRA, R. R. Uso do microcontrolador esp8266 para automação residencial. 2017. Disponível em: <http://repositorio.poli.ufrj.br/monografias/monopoli10019583.pdf>.

PINOUT. 2020. Disponível em: <http://www.datasheetcafe.com/sg90-datasheet-pdf-9-g-micro-servo/>.

RODRIGUEZ, M. *A História dos Aplicativos – Quem usa e quem vive de desenvolver*. 2019. Disponível em: <https://tribunadaimprensalivre.com/a-historia-dos-aplicativos-quem-usa-e-quem-vive-de-desenvolver/>.

SANTANNA, B.; CALVALCANTI, L. Automação residencial com no-demcu. 2018. Disponível em: <https://app.uff.br/riuff/bitstream/1/8063/1-/TCC%20%20%20Bernardo%20harduim%20e%20Luiz%20Vinicius.pdf>.

SILVA, C. O. R.; MENDES, D.; SALES, A. Sistema de automação residencial de baixo custo utilizando o esp8266. 2021. Disponível em: <http://prpi.ifce.edu.br/nl/_lib/file/doc1526-Trabalho/PEVPI_RF.pdf>.

SILVA, M. *JavaScript - Guia do Programador: Guia completo das funcionalidades de linguagem JavaScript*. Novatec Editora, 2010. ISBN 9788575222485. Disponível em: <https://books.google.com.br/books?id=BB9WDQAAQBAJ>.

SILVA, P. et al. *SQLite para Dispositivos Móveis*. 2017. Disponível em: <https://web-.archive.org/web/20180423181816id_/http://revistaeletronica.unicruz.edu.br/index.php-/revistaeletronica/article/viewFile/5410/1143>.

SOUZA, I. de. *Saiba o que é Node.js, como ele funciona e como usá-lo no seu site*. 2020. Disponível em: <https://rockcontent.com/br/blog/node-js/>.

SOUZA, I. de. *Saiba o que é NPM (Node Package Manager) e como instalar*. 2020. Disponível em: <https://rockcontent.com/br/blog/npm/#1>.

SQLITE, G. C. e D. *Sobre SQLite*. 2021. Disponível em: <https://www.sqlite.org/about-.html>.

SYSTEMS, E.; TECHNOLOGY, S. A. Esp-12e wifi module. *Documentação*, 2015. Disponível em: <https://cdn.datasheetspdf.com/pdf-down/E/S/P/ESP-12E-AI-Thinker.pdf>.

TECNOLOGIA, F. K. *NodeMCU ESP8266: Detalhes e Pinagem*. 2021. Disponível em: <https://www.fernandok.com/2018/05/nodemcu-esp8266-detalhes-e-pinagem.html>.

VOLTRIZ. 2021. Disponível em: <https://www.voltriz.com.br/produto/led-branco-5mm-alto-brilho/>.