UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

KLEBER LEANDRO REPECKI DOS SANTOS
GABRIEL ADRIOVANE SCHEIDT

PROTÓTIPO PARA MONITORAMENTO DE ESTUFAS DE SECAGEM DE TABACO

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA
2021

KLEBER LEANDRO REPECKI DOS SANTOS

GABRIEL ADRIOVANE SCHEIDT

PROTÓTIPO PARA MONITORAMENTO DE ESTUFAS DE SECAGEM DE TABACO

Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, do Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Augusto Foronda

PONTA GROSSA

2021

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Ponta Grossa
Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

PROTÓTIPO PARA MONITORAMENTO DE ESTUFAS DE SECAGEM DE TABACO

por

KLEBER LEANDRO R. DOS SANTOS
GABRIEL ADRIOVANE SCHEIDT

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 13 de maio de 2021 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. Augusto Foronda
Orientador(a)

_____
Prof. Dr. Itamar Iliuk
Membro titular

_____
Prof. Luiz Rafael Schmitke
Membro titular

_____
Prof. Geraldo Ranthum
Responsável pelo Trabalho de Conclusão de Curso

_____
Prof. Dr. André Pinz Borges
Coordenador do curso

RESUMO

REPECKI DOS SANTOS, Kleber Leandro; ADRIOVANE SCHEIDT, Gabriel. **PROTÓTIPO PARA MONITORAMENTO DE ESTUFAS DE SECAGEM DE TABACO**. 2021. 49 f. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) — Universidade Tecnológica Federal do Paraná, Ponta Grossa, 2021.

Este trabalho tem como intuito desenvolver um protótipo de aplicativo *mobile* para monitoramento de estufas de secagem de tabaco. Existem diversas soluções para monitoramento afim de manter o agricultor informado e auxiliando nas tomadas de decisão. O objetivo principal deste trabalho é desenvolver um protótipo com a utilização do micro computador *Raspberry* com o sensor de temperatura e umidade para armazenar os dados de temperatura e umidade em estufas de tabaco, com o interfaceamento das informações através de dispositivos móveis conectados na rede de *WiFi* local. Pois, compreende-se que, a utilização dessas ferramentas tecnológicas, além de auxiliar no desenvolvimento da produção da agricultura, possibilita o monitoramento e controle mais preciso, reduzindo os custos da produção. Dessa forma, observa-se que é de suma importância ter a tecnologia aliada aos processos da agricultura.

Palavras-chave: Tabaco. Raspberry. Estufa.

ABSTRACT

REPECKI DOS SANTOS, Kleber Leandro; ADRIOVANE SCHEIDT, Gabriel. **..** 2021. 49 p. Undergraduate Thesis (Technologist Degree in Course(s)) — Federal University of Technology — Paraná, Ponta Grossa, 2021.

This work aims to develop a prototype application *mobile* for monitoring tobacco drying greenhouses. There are several solutions for monitoring in order to keep the farmer informed and assisting in decision making. The main objective of this work is to develop a prototype using the micro computer *Raspberry* with the temperature and humidity sensor to store temperature and humidity data in tobacco greenhouses, with the interface of information through mobile devices connected to the local *WiFi* network. Therefore, it is understood that the use of these technological tools, in addition to assisting in the development of agricultural production, enables more accurate monitoring and control, reducing production costs. Thus, it is observed that it is of paramount importance to have technology allied to the processes of agriculture.

Keywords: Tobacco. Raspberry. Stove.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

SIGLAS

API	Application Programming Interface
CSS	Cascading Style Sheets
DHCP	Dynamic Host Configuration Protocol
HDMI	High-Definition Multimedia Interface
HTML	HypertText Markup Language
IoT	Internet of Things
ISO	International Organization for Standardization
MB	Megabyte
MIMO	Multiple Input Multiple Output
NTP	Network Time Protocol
OFDM	Orthogonal Frequency Division Multiplexing
RAM	Random Access Memory
RESTful	Representational State Transfer
SD Card	Secure Digital Card
SDK	Software Development Kit
SO	Sistema Operacional
USB	Universal Serial Bus

SUMÁRIO

1 INTRODUÇÃO

Para o avanço da agricultura de precisão, tem-se buscado cada vez mais tecnologias alternativas para se ter um acompanhamento preciso de cada processo das atividades agronômicas (BERNARDI; INAMASU, 2014). Dessa forma, este trabalho consiste em desenvolver um protótipo para monitorar a secagem das folhas de tabaco com o interfaceamento através de aplicativo mobile.

O processo de secagem e cura do tabaco são subdivididos em 4 fases, nas quais é estabelecido um limite de temperatura e umidade para extrair uma melhor qualidade da folha. As fases descritas são: amarelação, murchamento, secagem da folha/lâmina e secagem do talo (BUDNY, 2016).

Estas fases são controladas através dos dados de temperatura e umidade, juntamente com o aspecto físico do tabaco acompanhado pelo produtor. Estes dados são captados através de um sensor instalado no interior da estufa e mostrado em um *display* na parte externa. Em muitos casos essas estufas ficam em locais remotos e a comunicação com a Internet é limitada, encarecendo assim o emprego de sistemas de monitoramento que utilizam a Internet.

Neste processo produtivo encontra-se presente a agricultura familiar, na qual em muitos casos a produção é pequena (HARTWIG; VENDRAMINI, 2008), inviabilizando assim financeiramente a compra de sistemas complexos de automatização, submetendo esses trabalhadores a realizar trabalhos manuais, com longos períodos de exposição aos agentes químicos do processo (MURAKAMI et al., 2017).

Para resolver estes problemas, observa-se atualmente o surgimento de várias ferramentas, que tem como intuito auxiliar tanto no desenvolvimento dos negócios empresariais quanto nos negócios familiares. Uma dessas ferramentas é o *Raspberry*, um microcomputador com diversas funções que pode auxiliar na estufa de tabaco gerando assim benefícios e bons resultados, além de sua capacidade de permitir a ajuda na agricultura familiar também pode ser uma tecnologia de baixo custo. Outra tecnologia é a rede *WiFi*, que faz o transporte das informações captadas pelos sensores até um servidor para o usuário poder visualizar e analisar as informações, através do desenvolvimento de aplicativos.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Desenvolver uma solução de baixo custo para armazenamento e monitoramento da temperatura e umidade durante o processo de secagem em estufa do tabaco. Solução essa para aplicação em uma rede local com roteador sem acesso a internet.

1.1.2 Objetivos Específicos

Os objetivos específicos são:

- Estudar a aplicabilidade do microcomputador *Raspberry*
- Estudar estratégias para leitura e armazenamento de temperatura e umidade a partir de sensores
- Demonstrar a utilização do micro computador *Raspberry* com o sensor de temperatura e umidade;
- Desenvolver aplicação híbrida para dispositivos móveis para interfaceamento remoto do protótipo;

1.2 JUSTIFICATIVA

A criação de um protótipo de monitoramento de temperatura e umidade, tem com intuito facilitar o processo na agricultura familiar, assim, buscando através do monitoramento remoto minimizar a exposição dos trabalhadores aos agentes químicos do processo, assim como um produto mais acessível financeiramente para os pequenos fumicultores.

Esse processo que necessita de um acompanhamento continuo do agricultor tem maior valor quando utilizado dentro de uma rede local, pois além de termos uma conexão de internet limitada nas regiões mais afastadas da cidade, o agricultor também não terá ação no processo de secagem se estiver a quilômetros de distancia da estufa. Dessa forma acessar as informações via internet não apresenta tanta relevância para a aplicação quanto uma rede local.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 AGRICULTURA DO TABACO

O tabaco é uma planta denominada cientificamente Nicotiana tabacum. Observa-se, que esta planta chegou ao Brasil através de migração de tribos tupis-guaranis.

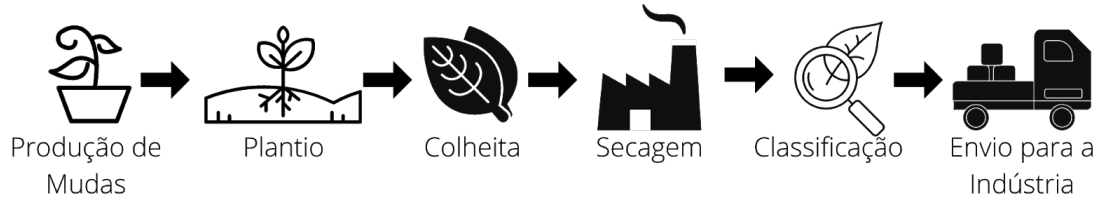
Os dados disponibilizados pela Associação dos Fumicultores do Brasil (AFUBRA, 2019) mostra, que o Brasil é o segundo maior produtor de fumo do mundo, e líder em exportações (BOBATO, 2017). Em 2019, foram exportadas cerca de 651.190 toneladas de Tabaco oriundos da produção brasileira. A região sul é a líder na produção nacional com 664.335 toneladas produzidas no ano de 2019 (AFUBRA, 2019).

No entanto, ao analisar o cultivo do tabaco observa-se que sua produção para o mercado tem início efetivo no século XVII, quando Portugal passou a incentivar o seu cultivo no Nordeste Brasileiro, com o objetivo de realizar trocas comerciais com a Europa, visando garantir o fornecimento de trabalho escravo para a economia do açúcar (NARDI, 1996).

Atualmente, o processo de produção do Tabaco ainda consiste em atividades manuais para seu cultivo e colheita. Tendo em vista os dados do Ministério do Desenvolvimento Agrário-MDA, cerca de 97% da produção na cultura do tabaco provêm da agricultura familiar (BOBATO, 2017), o que acaba inviabilizando o uso de práticas mais tecnológicas e mais atualizadas, tendo em vista o alto custo de implementação.

As etapas de produção do tabaco nas propriedades produtoras, consiste em: produção das mudas, transplante para a área de cultivo, processo de desenvolvimento das plantas, colheita, secagem, classificação e envio para a indústria (SOUZA, 2020a).

Figura 1 – Etapas de produção do tabaco



Fonte: Autoria própria (2021).

2.1.1 Processo de secagem do tabaco

O presente trabalho foca na etapa de secagem na qual será implementado o protótipo proposto.

O processo de secagem do tabaco é subdividido em 4 fases: amarelação, murchamento, secagem da lâmina e secagem do talo (BUDNY, 2016). Estas fases são acompanhadas atentamente pelo produtor e podem durar 7 dias ou mais, dependendo do modelo de estufa utilizada na propriedade. A secagem é uma das etapas mais importantes do processo produtivo do tabaco, pois é de demasiada importância para a qualidade final do produto (PROFIGEN,2020).

Cada fase da secagem tem uma determinada faixa de temperatura e Umidade a ser respeitada conforme a Tabela 1 e a Tabela 2. Estes valores servem para trazer uma melhor qualidade do produto e estabelecer uma secagem homogênea da folha e talo. A temperatura segue o padrão de medição em Fahrenheit, já adotado nos equipamentos de controle utilizados atualmente pelos produtores.

Tabela 1 – Faixas de temperatura das etapas de secagem para estufas de grampo

Etapa de secagem	Faixa de Temperatura
Amarelação	90°F a 100°F
Murchamento	101°F a 110°F e 111°F a 120°F
Secagem da Lâmina	121°F a 140°F
Secagem do Talo	141°F a 165°F

Fonte: (BUDNY, 2016).

Tabela 2 – Faixas de umidade relativa das etapas de secagem para estufas de grampo

Etapa de secagem	Faixa de Temperatura	Faixa de Temperatura
Amarelação	90°F	94% a 98%
Amarelação	92°F	93% a 97%
Amarelação	94°F	92% a 96%
Amarelação	96°F, 98°F, 100°F	91% a 95%
Murchamento	102°F	81% a 85%
Murchamento	104°F	78% a 82%
Murchamento	106°F	73% a 77%
Murchamento	108°F	70% a 74%
Murchamento	110°F	68% a 72%
Secagem da Lâmina	112°F	63% a 67%
Secagem da Lâmina	114°F	61% a 65%
Secagem da Lâmina	116°F	59% a 63%
Secagem da Lâmina	118°F	55% a 59%
Secagem da Lâmina	120°F	51% a 55%
Secagem da Lâmina	122°F	48% a 52%
Secagem da Lâmina	124°F	45% a 49%
Secagem da Lâmina	126°F	42% a 46%
Secagem da Lâmina	128°F	39% a 43%
Secagem da Lâmina	130°F	38% a 42%
Secagem da Lâmina	132°F	37% a 41%
Secagem do Talo	134°F	24% a 38%
Secagem do Talo	136°F	32% a 36%
Secagem do Talo	138°F	30% a 34%
Secagem do Talo	140°F	28% a 32%
Secagem do Talo	142°F	26% a 30%
Secagem do Talo	144°F	24% a 28%
Secagem do Talo	146°F	22% a 26%
Secagem do Talo	148°F	20% a 24%
Secagem do Talo	150°F	15% a 22%

Fonte: (BUDNY, 2016).

2.1.2 Tipos de Estufas de Tabaco

A etapa de secagem descrita anteriormente consiste em promover transformações físico-químicas nas folhas colhidas, este processo é seguido de uma extração de água das folhas dentro das estufas. Anteriormente é feito um planejamento de quantidade de plantas ou da área a ser cultivada dependendo da capacidade limite de carga da estufa. Nas estufas do modelo convencional são utilizadas varas ou bastões para acomodar as folhas durante o processo de cura (PROFIGEN, 2020).

Neste modelo de estufa, os ventiladores de entrada, chamados de suspiros, têm sua área um terço menor que a área dos ventiladores de saída. Em razão que o ar

que é admitido na estufa é mais denso e com temperatura inferior à temperatura interna da estufa. Que quando aquece, esse ar se expande, aumenta seu volume e procura a sua saída para cima. Ao subir, o ar encontra uma camada de umidade, a qual é a água liberada pelas folhas. A massa de ar perde temperatura em contato com a umidade, que a carrega para fora da estufa (PROFIGEN, 2020).

Figura 2 – Exemplo de Estufa Convencional

Fonte: (BRASTECC, 2012).

Outro modelo de estufa que vem sendo utilizado nos últimos anos, o qual surgiu em 2012, são as Estufas LL ou de Ar Forçado, as quais utilizam eletro ventiladores localizados acima da fornalha, esses eletro ventiladores trabalham constantemente para facilitar a circulação de ar dentro da estufa, onde a Admissão do ar é feita através de Flaps localizados na parte superior da estufa em cima dos eletromotores que podem ser automatizados. Este modelo pode comportar folhas soltas em grades ou uso de grampos. (MATE, 2018).

O modelo da Figura 3 opera para otimizar a eficiência térmica, com um isolamento que pode reduzir até 25% do consumo da lenha durante o processo de secagem (SOUZA, 2020b). Uma opção que muitos produtores vêm adotando é a conversão da estufa convencional para ar forçado. Onde pode se utilizar a mesma estrutura para acomodar os eletromotores e demais componentes.

Figura 3 – Modelo de Estufa LL / Ar Forçado

Fonte: (SOUZA, 2020b).

2.2 COMPONENTES DE HARDWARE

2.2.1 Micro Computador *Raspberry*

O *Raspberry* PI pode ser definido como um microcomputador com pequenas proporções, assim ao comparar, pode-se dizer que o mesmo tem o tamanho de um cartão de crédito e um custo baixo. O seu desenvolvimento no Reino Unido pela Fundação *Raspberry* Pi, teve como o principal objetivo promover o ensino de desenvolvimento de sistemas nos meios educacionais (UPTON; HALFACREE, 2017).

Apesar de ser um mini microcomputador, o espaço abriga processador, processador gráfico, *slot* para a inserção de cartão micro SD Card, “interface” USB, HDMI e os seus receptivos controladores, podendo ser apresentado como um computador completo já que contém memória RAM, entrada de energia e barramentos de expansão, entre outros detalhes apresentados acima.

Por ter uma grande capacidade de processamento, e a opção de interfaces *plug and play* como teclados, *displays*, sensores e outros componentes aliado a sistemas operacionais robustos e linguagens de desenvolvimento de fácil aprendizagem acabou sendo amplamente utilizado em projetos IoT (UPTON; HALFACREE, 2017).

Entre os microcomputadores *Raspberry* atualmente disponível destacamos o *Raspberry* Pi Zero. Lançado em novembro de 2015, conta com um processador *Broadcom BCM2835 ARMv6* com um núcleo de 1 GHz, memória RAM de 512 MB,

entrada micro USB, uma saída mini-HDMI e porta *GPIO* de 40 pinos. Em 2016, surgiu uma atualização para o modelo com um conector para câmera CSI (RASPBERRY PI, 2020).

No entanto, existem outras versões as quais observa abaixo nos tópicos diferindo os modelos que vão desde *Raspberry* Pi 1 com 700MHz de *clock* de CPU e 256 MB de memória RAM até o mais atual *Raspberry* Pi 4 com 1.5GHz de *clock* de CPU e 8 GB de memória RAM (RASPBERRY PI, 2020).

2.2.2 Sensor de Temperatura e Umidade

Para realizar medições de temperatura e umidade existem diversos sensores no mercado, e para aplicações IoT um modelo muito utilizado são os sensores ASAIR DHT22(AM/CM2302) ou o DTH11.

O sensor DHT11 e o DHT22 são sensores que utilizam um termistor e um sensor capacitivo para medir, de forma simples de usar, mas requer atenção entre duas leituras consecutivas, sendo necessário um intervalo entre uma leitura e outra. Esses sensores de baixo custo e fácil aquisição tem compatibilidade com a porta GPIO de entrada ou saída genérica, digital do *Raspberry*. Como esse sensor é amplamente utilizado em aplicações IoT, se encontra facilmente artigos com exemplos de códigos em Python *python*.

Entretanto, mesmo com a similaridade na pinagem e aparência, apresentam diferentes características que são as seguintes:

As características de cada sensor são as seguintes:

- Sensor DHT11- Um baixo custo comparado com o DHT22, uma tensão de alimentação de 3V a 5V, 2.5mA de corrente máxima durante a conversão, a medição de umidade entre 20% e 80%, com 5% de precisão. Ótimo para medir temperaturas entre 0 e 50ºC, com ± 2ºC de precisão, taxa de amostragem de até 1Hz e as dimensões: 15.5mm x 12mm x 5.5mm com os 4 pinos de 0.1"de espaçamento entre eles (AOSONG, 2020).
- Sensor DHT22- Possui um baixo custo, com a tensão de alimentação de 3V a 5V, 2.5mA de corrente máxima durante a conversão, a medição de umidade entre 0% e 100%, com 2% a 5% de precisão, ótimo para medir temperaturas entre -40 e 125°C, com 0,5°C de precisão, e a taxa de amostragem de até

0,5Hz, e as dimensões: 15.1mm x 25mm x 7.7mm com 4 pinos de 0.1"de espaçamento entre eles (AOSONG, 2020).

Diante disso, observa-se que o DHT22 melhor se enquadra nos requisitos do projeto, pois seu range de temperatura entre -40 e 125°C e de umidade entre 0% e 100% atendem as necessidades do projeto, e sua faixa de atualização de 0,5Hz é mais que suficiente, para a aplicação que fará 1 leitura a cada 10 segundos.

2.2.3 Outros módulos necessários

2.2.3.1 Módulo de relógio

O microcomputador *Raspberry* não possui integrado baterias, e por esse motivo causa um atraso na data e hora quando desligado e por padrão o sistema operacional sempre corrigi esse atraso de data e hora através de servidores NTP na Internet, mas para casos em que o *Raspberry* não está conectado na Internet não é possível fazer essa correção, resultando em uma data e hora errada no microcomputador *Raspberry*.

Para resolver essa questão faz-se necessário instalar fisicamente no *Raspberry* um Módulo Real Time Clock Rtc Ds3231 At24c32 com bateria para ser utilizado de referência pelo sistema operacional nos casos que não for possível conectar a um servidor NTP externo, dessa forma garante maior confiabilidade na data e hora do dispositivo, por não gerar o atraso de data e hora quando o dispositivo estiver desligado.

Figura 4 – Módulo Real Time Clock Rtc Ds3231 At24c32

Fonte: (MAXIM, 2015).

2.2.3.2 Adaptador Micro USB Ethernet

Na versão do microcomputador *Raspberry PI Zero*, não contém conexão ethernet RJ45, e para esse caso é necessário utilizar um módulo adaptador de micro USB para Ethernet RJ45.

O adaptador habilita uma porta ethernet a partir da conexão micro USB, e sua instalação na maioria dos Sistemas operacionais acontece de maneira automática (FLIP FLOP, 2020).

Figura 5 – Adaptador Micro USB Ethernet

Fonte: (FLIP FLOP, 2020).

2.3 COMPONENTES DE SOFTWARE

2.3.1 Sistema Operacional

O sistema operacional é o software executado em um computador que gerencia os recursos do mesmo, fazendo a interface entre o computador e o usuário. Quando adquirido o microcomputador *Raspberry PI*, é necessário instalar um sistema operacional. O fabricante do *Raspberry PI* recomenda o Sistema Operacional *Raspberry PI* OS (32-bit), também chamado de Raspbian, desenvolvido sobre o SO Debian, que está atualmente na versão 5.4 lançada em 20 de agosto de 2020 e o tamanho da ISO é 2531 MB (RASPBERRY PI, 2020).

Existem outros sistemas operacionais como o Windows IoT Core criado especialmente para o *Raspberry PI* como uma plataforma de desenvolvimento para programadores e codificadores. Tem ênfase na segurança, conectividade, criação e integração na nuvem. Também há o sistema operacional Ubuntu projetado para aplica-

tivos IoT, baseado em Linux de fácil uso com o Ubuntu Snappy Core em seu *Raspberry PI*.

Já o *Open Source Media Center* é um Kodi OS independente, simples, de código aberto, com a capacidade de reproduzir virtualmente qualquer formato de mídia, sua interface de usuário moderna e minimalista é totalmente personalizável por conta das diversas imagens incorporadas que ele oferece.

O Sistema Operacional Raspbian que vai ser usado no projeto, vem com um pacote de programas úteis para aplicações com IoT, por exemplo, o Scratch, Python 3 (IDLE), Thonny Python IDE, Flask, entre outros, melhorando assim o desenvolvimento educacional (RASPBERRY PI, 2020).

2.4 BANCO DE DADOS

O MySQL é um banco de dados relacional onde em sua versão 5.7, é possível observar o lançamento de novas funcionalidades, como o *Geographic Information System* (GIS). Já a MariaDB é um banco de dados avançado que disponibiliza mais recursos do que o MySQL, portanto, estes dois sistemas de gerenciamento de banco de dados têm suas diferenças, tais como (BARTHOLOMEW, 2012):

- MariaDB é totalmente licenciada com GPL, enquanto o MySQL tem uma abordagem de dupla licença;
- Cada uma lida com pools de *thread* de forma diferenciada;
- MariaDB suporta mais mecanismos de armazenamento;
- MariaDB oferece melhor desempenho em alguns cenários.

Para armazenar os dados, o *Raspberry* utiliza um banco de dados relacional, gestionado pelo SGBD *open source* MariaDB. Para realizar a instalação do MariaDB é necessário executar o comando: "sudo apt-get install mariadb-server -y". Após realizado essa instalação, é necessário instalar a biblioteca de conexão com o banco de dados MariaDB da linguagem Python. Basta executar o comando “pip3 install mariadb”, e posteriormente fazer a importação e conexão com o banco de dados na aplicação de acordo com a documentação do MariaDB (MARIADB, 2020).

2.5 API

Para que os dispositivos conectados na rede local do *Raspberry PI* possam acessar os dados armazenados de leitura da estufa, é necessário configurar o *Raspberry PI* para funcionar como um Servidor Web no qual será utilizado o padrão *Web services* RESTful.

O padrão *Web services* RESTful é um padrão de Transferência Representacional de Estado, onde são feitas requisições através de uma URL para o servidor e o mesmo executa uma ação ou devolve uma resposta formatada principalmente em HTML, XML ou JSON. Essa resposta pode ser a consulta de uma informação, *feedback* de uma alteração, criação ou exclusão de informação, entre outras aplicabilidades. Quando usado com HTTP, os principais métodos de requisição são: GET, HEAD, POST, PUT, PATCH, DELETE, CONNECT, OPTIONS e TRACE (FIELDING, 2014).

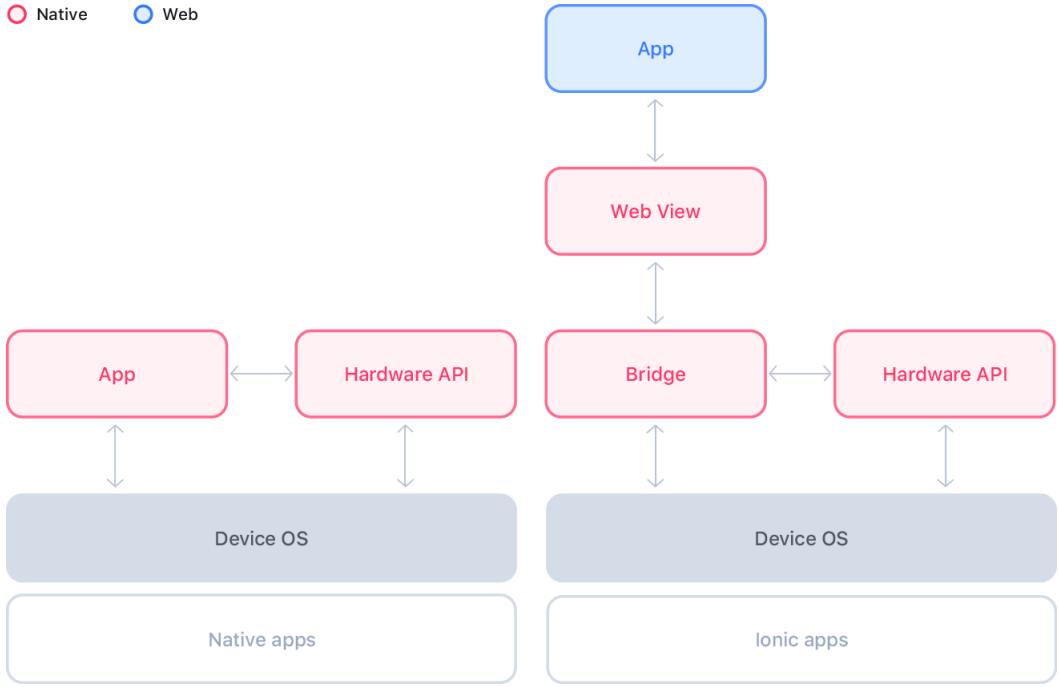
Para configurar essa API RESTful no *Raspberry PI* pode-se utilizar o *framework Flask*. Esse *framework* é escrito com a linguagem Python, e sua estrutura e aplicação é de simples desenvolvimento, pois após instalar as bibliotecas com o comando “pip install Flask” basta somente fazer a importação e adicionar as rotas.

2.6 FRAMEWORK IONIC

O *Framework* Ionic é um *open source* SDK que foi criado em 2013, utilizando PhoneGap e AngularJS (GOIS, 2016). Os aplicativos desenvolvidos com Ionic são renderizados com a tecnologia de *Web View*, no qual utilizam um *browser* em tela cheia para executar um código com HTML, CSS e JavaScript, e integrando atualmente com os *frameworks* Angular, Vue.js, e React (IONIC, 2020).

Nas aplicações desenvolvidas com a linguagem de programação nativa do dispositivo é possível acessar diretamente os recursos como por exemplo câmera, microfone, auto falantes entre outros. Por outro lado o *framework* Ionic precisa do *framework* Cordova para realizar essa ponte com os recursos nativos, e para melhor exemplificar a Figura 6 faz a comparação de uma aplicação com linguagem nativa do dispositivo com aplicações desenvolvidas com Ionic no qual o *framework* Cordova é responsável pelo "Bridge"do fluxograma (LIMA et al., 2018).

Figura 6 – Esquema Ionic para acessar recursos do dispositivo



Fonte: (IONIC, 2020).

2.7 FRAMEWORK FLASK

Como o sistema operacional Raspbian é desenvolvido sobre o Linux Debian, sua linguagem nativa é o Python, facilitando assim o desenvolvimento, pois não é preciso configurar o sistema para rodar outras linguagens.

Flask é um microframeworks Python para se construir sistemas web, e seu núcleo simples não precisa de ferramentas ou bibliotecas particulares para execução, dessa forma com poucas linhas de código é possível ter uma aplicação web funcional.

Como o Flask tem aplicabilidade no ambiente web, é possível desenvolver através dele APIs para serem consumidas por outros programas, e também páginas Web com seu mecanismo de template padrão Jinja.

Segue abaixo um exemplo de código com framework Python:

Figura 7 – Exemplo de código Python com framework Flask

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

Fonte: (FLASK).

3 COMPONENTE DE COMUNICAÇÃO

3.1 REDE WIFI

O padrão Wifi (IEEE 802.11) trata-se de uma solução de comunicação sem fio, sendo a qual é amplamente conhecida e utilizada, se fazendo presente nas residências, pontos turísticos, hospitais, espaços comerciais, etc. Esse padrão define um conjunto de padrões de transmissão e codificação e permite uma taxa de comunicação de até 450 Mbps na versão IEEE 802.11n (VANZELLA, 2018), o qual possui multiplexação ortogonal por divisão de frequência (OFDM) usando múltiplas entradas/saídas múltiplas (MIMO) e ligação de canal, podendo trabalhar tanto em bandas de 2,4Ghz tanto em 5Ghz (INTEL, 2021).

4 DESENVOLVIMENTO

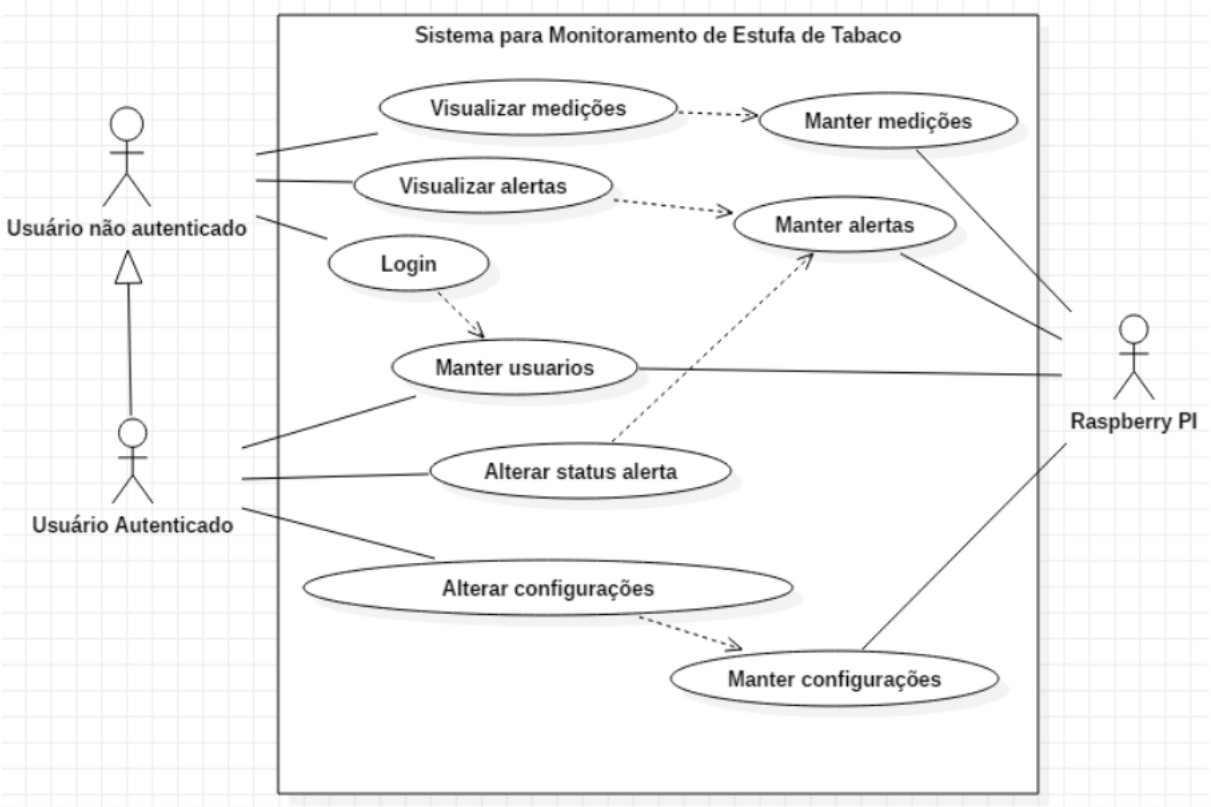
Neste capítulo será apresentado o desenvolvimento do protótipo do projeto e sua aplicação prática, detalhando a montagem do hardware, topologia da rede e os softwares desenvolvidos para as regras de negócio.

4.1 PERSPECTIVA DO PRODUTO

4.1.1 Diagrama de Contexto

Para melhor representação do projeto segue abaixo o diagrama Use Case no qual evidencia os atores e suas atribuições:

Figura 8 – Diagrama Use Case



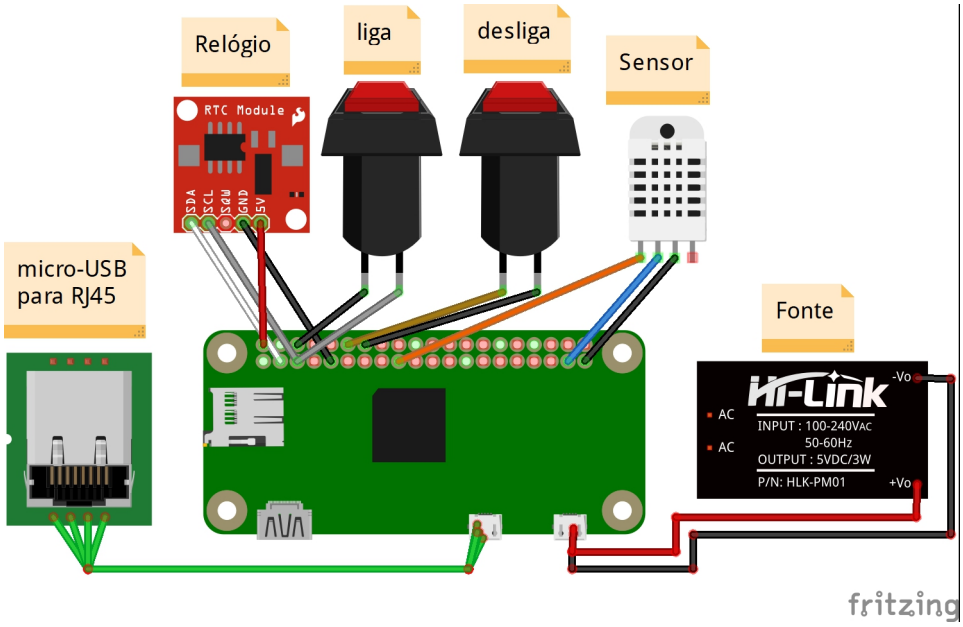
Fonte: autoria própria (2021).

4.2 HARDWARE

4.2.1 Esquema de Ligação

A Figura 6 demostra o esquema de ligação dos componentes de hardware do protótipo.

Figura 9 – Esquema de ligação



Fonte: autoria própria (2021).

4.2.2 Topologia da Rede

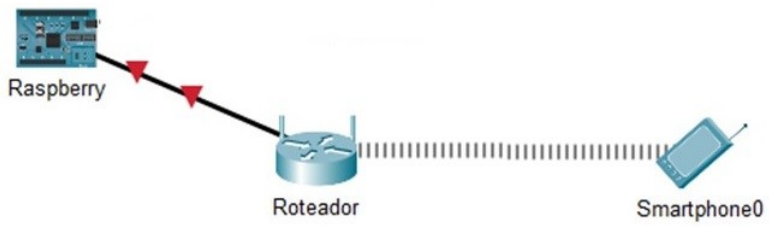
Para transmitir os dados dos sensores, é necessário conectar ao *Raspberry PI* um adaptador de micro USB para Ethernet RJ45, através dessa opção de conexão de rede ethernet foi pensado em uma topologias de rede de simples configuração, para conectar o Microcomputador *Raspberry PI* em um roteador para realizar o roteamento do DHCP dos dispositivos que vão ser conectados.

A topologia utiliza um roteador para distribuir os IP por DHCP para os dispositivos conectados na rede, e o *Raspberry PI* pode ser conectado no roteador através do cabo Ethernet limitado a 100m de comprimento de acordo com a Figura 10.

Como todos os dispositivos estarão conectados na mesma rede, o roteador por padrão tem as portas abertas para que os dispositivos possam se comunicar entre si,

mas em alguns casos é necessário também habilitar a opção “zona desmilitarizada” para que os dispositivos conectados via cabo possam se comunicar com os dispositivos conectados na rede sem fio.

Figura 10 – Topologia 1: Exemplo de topologia de comunicação com o *Topologia 1*



Fonte: autoria própria (2021).

4.3 SOFTWARE

4.3.1 Sistema Operacional

Foi utilizado o Sistema Operacional Raspbian Kernel version: 5.4 recomendado pelo Raspberry Pi Foundation.

Para fazer a instalação do Sistema Operacional no cartão de memória do *Raspberry PI* foi necessário inicialmente formatar o cartão microSD com o software SD Card Formatter, para na sequência transferir os dados do arquivo .ISO do SO com o programa balenaEtcher.

Ao concluir esses passos o cartão está pronto para ser inserido no microcomputador *Raspberry PI* para ser iniciado o Sistema Operacional.

4.3.2 Banco de Dados

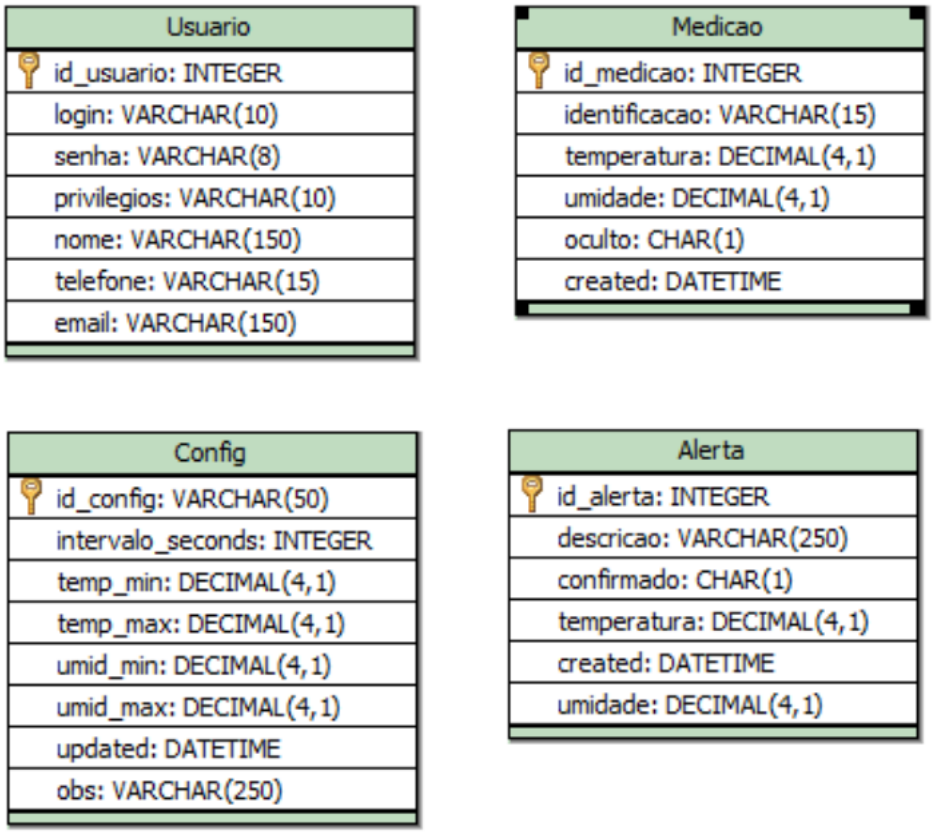
Para suprir as necessidades da regra de negócio do projeto foi necessário modelar o banco de dados com 4 tabelas:

- Tabela Usuário: Com informações de login e senha do usuário, e outros dados que não estamos armazenando no momento, mais prevemos que seja

necessário no futuro armazenar;

- Tabela Medição: Para armazenar os parâmetros medidos com data e hora;
- Tabela Alerta: Para armazenar os momentos que as medições estavam fora do ideal para o processo;
- Tabela Config: Para armazenar a configuração de alertas e funcionamento do dispositivo.

Figura 11 – Diagrama do modelo lógico do banco de dados



Fonte: autoria própria (2021).

4.3.3 Algoritmos

Foi necessário desenvolver 3 aplicações que são responsáveis por: Leitura Sensor, API e Aplicativo Mobile.

4.3.3.1 Leitura do Sensor

Código desenvolvido em Python para realizar a leitura do sensor e armazenar no banco de dados. Nesse algoritmo existe um loop que é executado em um intervalo de tempo de segundos, que está especificado em um registro armazenado na tabela Config do banco de dados, e em cada loop é realizado a leitura da temperatura e umidade, armazena na tabela Medição, e em paralelo é verificado se as medições estão dentro dos padrões, caso contrário é realizado um insert na tabela Alerta.

4.3.3.2 API

Código desenvolvido em Python com o Framework Flask para ser consumido pelo App Mobile, foi utilizado também na mesma estrutura o framework Jinja para montar páginas HTML com Javascript para acessar e modificar os dados armazenados no *Raspberry PI* através do navegador, para situações em que o dispositivo não comporte ou não tenha o App Mobile instalado.

Na sequência segue a relação de algumas requisições de API desenvolvidas para serem consumidas pelo App Mobile:

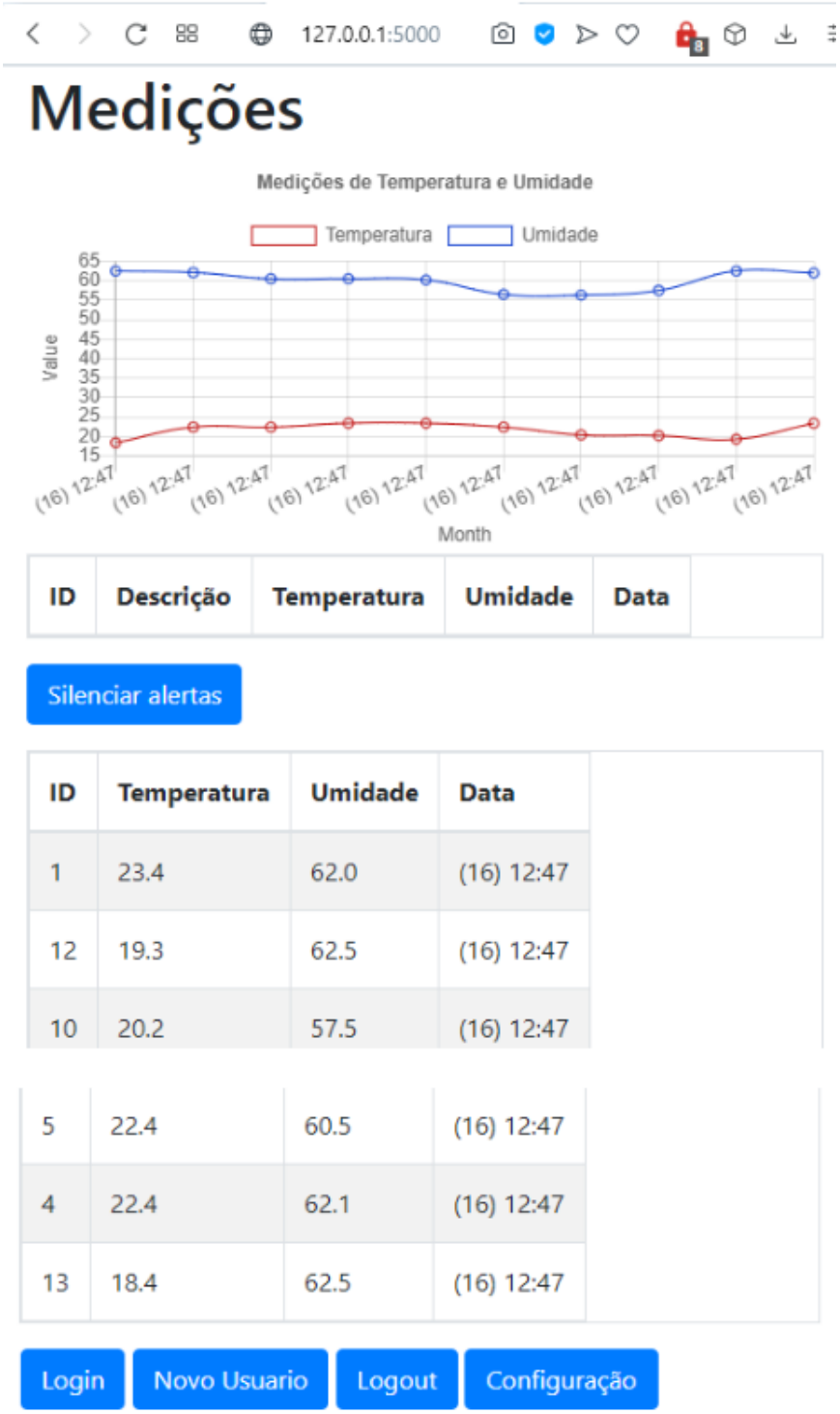
- Update data do Sistema (/updatedatasistema): Para atualizar a data do Microcomputador *Raspberry PI*;
- Valida Busca do dispositivo (/scan): Retorna IP e hora do dispositivo para o App Mobile validar que está com o IP correto do *Raspberry PI*;
- Última Medição (/medicao): Retorna a última medição de temperatura e umidade armazenada no banco de dados;
- Array de Medições (/medicoes): Retorna array de medições a partir da data inicial e final;
- Array de Alertas (/alertas): Retorna array de Alertas pendentes.

4.3.3.3 Interface API

Para permitir a visualização das medições do *Raspberry PI* sem a necessidade de um App mobile, foi desenvolvido através do framework Jinja uma interface para ser interpretado por navegadores. Para acessar essa interface é necessário primeiramente

estar conectado na mesma rede e informar o número de IP do *Raspberry PI* na rede com a porta 5000. Na Figura 12 segue a interface da Tela Inicial, com histórico de medições e botões para login, Novo Usuário e Configuração dos alertas. Para as opções de configuração dos alertas e novos usuários é necessário que esteja autenticado.

Figura 12 – Interface da Tela Inicial do template da API com framework Jinja com a escala de temperatura em Célcios



Fonte: autoria própria (2021).

4.3.3.4 Aplicação Mobile

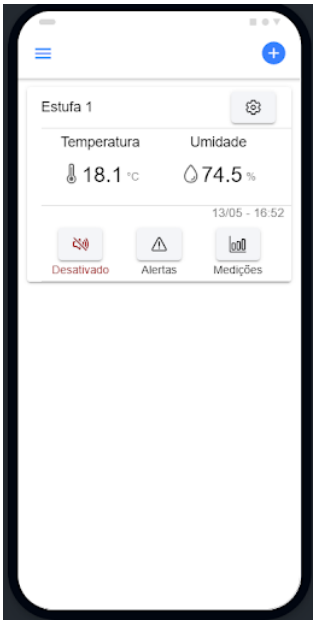
Código desenvolvido em Typescript com Frameworks Angular, Ionic e Cordova. Aplicação responsável em interfacear as informações armazenadas no Banco de dados do *Raspberry PI*. Através do Cordova foi desenvolvido a aplicação para rodar em background, com o objetivo de consultar as medições da estufa a cada 60 segundos, e alertar com sinal sonoro, vibração e push notification medições que estejam ocorrendo fora dos padrões de mínimo/máxima pré estabelecidos para o processo.

Através do App Mobile é possível ser alertado quando perder a conexão com o *Raspberry PI* sinalizando uma possível queda de energia e também quando as medições de temperatura e umidade estiverem fora dos limites estabelecidos na tela Config do App Mobile ou da página configuração acessada via navegador.

4.3.3.5 Interface

- Tela Inicial: Interface com a relação de estufas cadastradas no dispositivo, nessa tela é possível acompanhar a temperatura e umidade das estufas, ativar e desativar o alerta sonoro e navegar para as telas de cadastrar nova, medições, alertas e configuração da estufa;

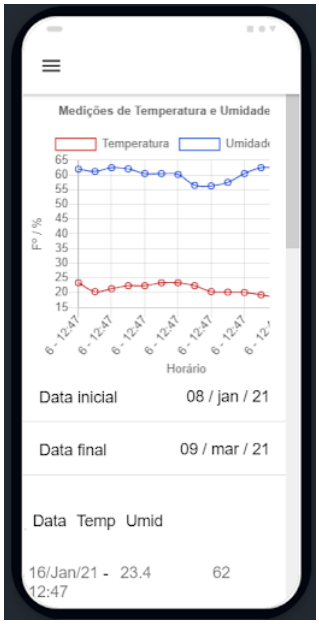
Figura 13 – Interface Inicial



Fonte: autoria própria (2021).

- Tela Medições: Interface com gráfico e tabela do histórico de medições, com opção de filtro para selecionar intervalo de tempo da medição;

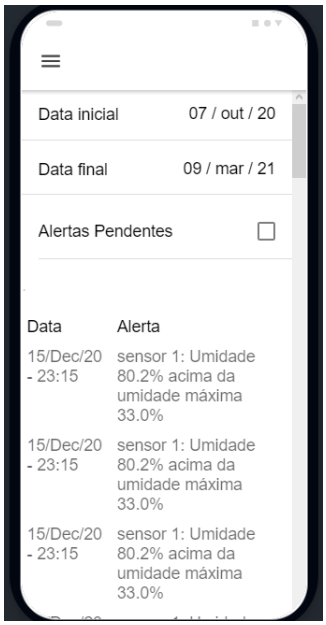
Figura 14 – Interface de histórico de medições



Fonte: autoria própria (2021).

- Tela Histórico de Alertas: Interface com relação de alertas referentes a temperatura e ou umidade que estava fora dos parâmetros pré estabelecidos. Possui filtro para intervalo de data e diferenciação de alertas não visualizados;

Figura 15 – Interface de histórico de alertas



Fonte: autoria própria (2021).

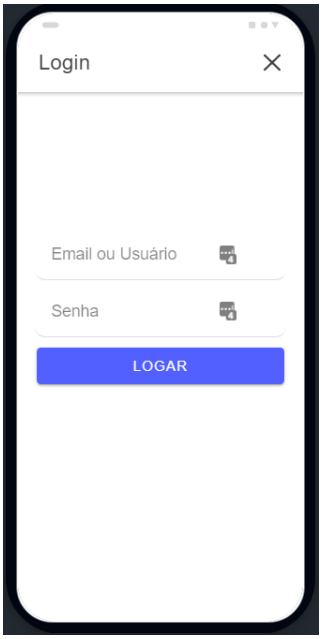
- Tela configuração de estufa: Interface com resumo das configurações da estufa no dispositivo, contando com número do IP da estufa, e opções para configurar IP da estufa, verificar e corrigir atraso de data e hora do *Raspberry PI*, login ou logout, ativar ou desativar os alertas, configuração dos alertas e excluir estufa do dispositivo;

Figura 16 – Interface de configurações da estufa no dispositivo

Fonte: autoria própria (2021).

- Tela Login: Para visualizar as medições da estufa não é necessário estar logado, mas para fazer alterações na configuração e alerta da estufa, é necessário logar com usuário e senha, que estão pré cadastrados no banco de dados do *Raspberry PI*. Ao se autenticar é enviado um token para ser armazenado no local storage do dispositivo, e em cada requisição de alteração de alerta do *Raspberry PI* é necessário que esse token seja enviado junto na requisição;

Figura 17 – Interface de login



Fonte: autoria própria (2021).

- Tela Configuração de alertas: Interface para escolher a escala de temperatura entre Célcios e Fahrenheit, e para parametrizar os valores de mínimo e máximo de temperatura e umidade que a estufa deve operar, caso a estufa opere fora desses parâmetros será emitido um alerta no banco de dados e também será emitido um alerta sonoro no dispositivo mobile se assim estiver configurado;

Figura 18 – Interface de configuração dos parâmetros de alertas, e escala de temperatura.

Fonte: autoria própria (2021).

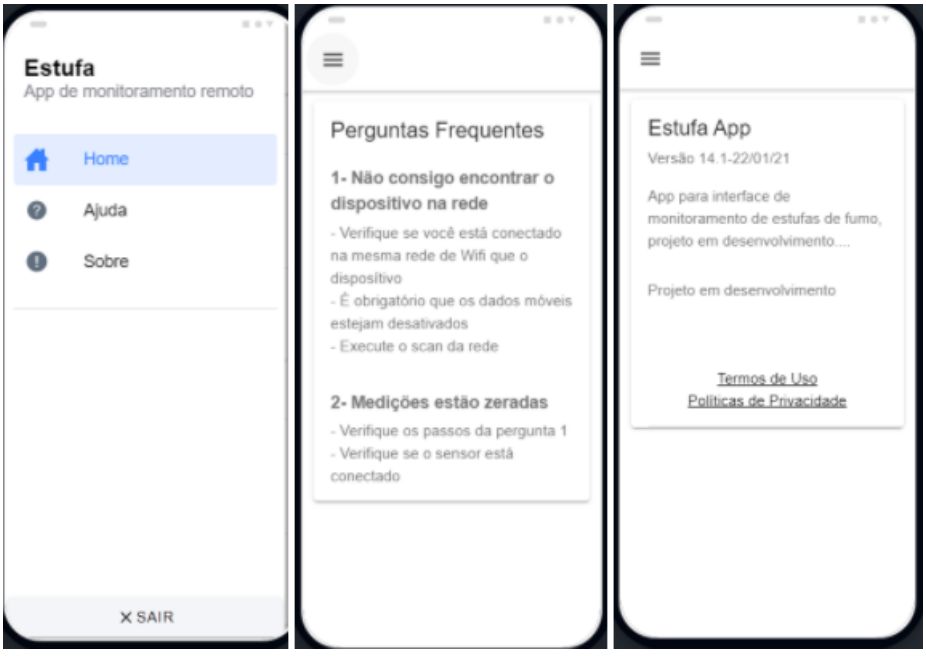
- Tela configuração IP do *Raspberry PI* no dispositivo mobile: Para que o dispositivo consiga se comunicar com o *Raspberry PI* faz se necessário que primeiramente estejam conectados na mesma rede. O IP do *Raspberry PI* deve ser preenchido no campo IP manualmente ou automaticamente através da busca na rede. Por padrão está sendo utilizado a porta 5000, essa porta pode ser alterada caso seja montada uma rede com vários roteadores, no qual é necessário fazer redirecionamento de portas nos roteadores. Essa interface pode ser chamada para alterar o IP de uma estufa já cadastrada no App Mobile e também para inserir uma nova estufa no App Mobile;

Figura 19 – Interface de configuração de IP

Fonte: autoria própria (2021).

- Menu lateral: Através do menu lateral existem os links para voltar para Tela Inicial, Tela Ajuda com perguntas frequentes pertinentes a configuração, Tela Sobre no qual descreve o App com versão, desenvolvedores e link para termos de Uso e Políticas de Privacidade obrigatório de acordo com a LGPD (Lei Geral de Proteção de Dados) e no final o botão Sair para finalizar corretamente o App Mobile;

Figura 20 – Interface Menu Lateral, Tela Ajuda e Tela Sobre



Fonte: autoria própria (2021).

4.4 SEGURANÇA

Para melhorar a segurança da aplicação foi desenvolvido algumas estratégias:

- Sistema Operacional: Como o SO não possui uma interface gráfica, só é possível acessar via Terminal informando no controle de acesso nativo do SO um usuário e senha. Como camada de segurança física o acesso via rede SSH desse terminal vai estar desativada, portanto somente vai ser possível acessá-lo plugando um teclado e monitor no microcomputador *Raspberry PI*;
- Banco de dados: Todas as consultas e alterações do banco de dados são executados pelas aplicações que estão rodando no microcomputador *Raspberry PI*, e não é possível acessar diretamente o banco de dados de forma remotamente;
- Rede Local: Como o objetivo da aplicação é rodar em um local sem acesso a internet. Somente será acessível aos dispositivos que estiverem no raio de alcance do sinal Wireless dos roteadores. E esses roteadores por sua vez também possuem senha de acesso;
- Aplicação: Para acessar as medições armazenadas no dispositivo, é possível via navegador informando o endereço IP local ou através do App Mobile. O acesso dessas informações são controlados através da aplicação de API, no qual para visualizar as medições está com acesso livre, mas para alterar as configurações de medição do equipamento faz se necessário logar com usuário e senha armazenados no Banco de Dados do dispositivo;

5 CONSIDERAÇÕES FINAIS

5.1 EVOLUÇÃO

Durante a fase de testes do protótipo em uma estufa de tabaco em produção, percebeu-se uma série de necessidades não levantadas inicialmente, entre essas melhorias foram:

Opção de adicionar várias estufas no mesmo App mobile, pois no local de teste do protótipo possui 3 estufas, e para a nossa solução seria necessário 1 *Rasperry PI* para cada estufa, e consequentemente um único App precisaria visualizar as 3 estufas.

Implantação de bateria para os momentos que ocorrerem falta de energia da concessionária, pois além de não ser possível a medição nesse período sem energia, o *Rasperry PI* não estava conectado a Internet e acumulou um atraso na data e hora, não apresentando confiabilidade no horário da medição.

Simplificação na configuração da estufa no App Mobile, como o nível de conhecimento de informática no usuário é bem limitado, foi necessário que o fluxo de configuração da estufa no App Mobile fosse extremamente simplificado, mas tendo também a opção de configurações personalizadas se assim o usuário desejar.

5.2 CONCLUSÃO

Diante do projeto proposto, observa-se que microcomputador *Rasperry PI Zero* atende as necessidades de processamento do projeto, e ainda se apresenta como uma alternativa financeiramente acessível para o monitoramento da temperatura e umidade de estufas de tabaco. E o conjunto das regras de negócio tratadas através dos algorítimos desenvolvidos com *Framework Ionic*, *Framework Flask* junto com o banco de dados MariaDB e conectados à rede local, atenderam as necessidades da regra de negócio.

Dessa forma, compreende-se que é de suma importância ter a tecnologia aliada aos negócios, pois, a mesma tem como intuito possibilitar o desenvolvimento de todo o processo na agricultura familiar, assim, buscando monitorar o processo de forma que diminui a exposição dos trabalhadores aos agentes químicos, como também torna-se o produto mais acessível financeiramente para os pequenos fumicultores e cooperando

com a saúde do mesmo.

5.3 TRABALHOS FUTUROS

Para conseguir agregar valor ao produto, faz se necessário adicionar uma regra com contatores para acionamento de motor de ventilação e abertura e fechamento de portinhola, equipamentos esses com acionamento automatizado em estufas mais modernas, minimizando a necessidade de intervenção humana no processo para controlar as faixas de temperatura e umidade da estufa.

REFERÊNCIAS

AFUBRA. **Fumicultura Mundial, Fumicultura no Brasil**. 2019. Disponível em: http://www.afubra.com.br.

AOSONG. **Especifícação técnica de sensores de temperatura e umidade**. 2020. Disponível em: http://www.aosong.com/en/products-22.html.

BARTHOLOMEW, D. Mariadb vs. mysql. **SkySQL Ab**, rozero.webcindario.com, 2012. Disponível em: http://rozero.webcindario.com/disciplinas/fbmg/abd3/MariaDB_vs_MySQL.pdf.

BERNARDI, A. C. d. C.; INAMASU, R. Y. Agricultura de Precisão. **Revista Embrapa**, Revista Embrapa, 2014. Disponível em: https://www.alice.cnptia.embrapa.br/bitstream/doc/1003476/1/cap1.pdf.

BOBATO, Z. L. A Fumicultura Como Parte Integrante Da Formação Socioespacial Sul-Brasileira: (Des) Caminhos E Perspectivas Futuras Para Os Produtores E A Região. Setor de Ciëncias da Terra, Universidade Federal do Paraná, p. 110–135, 2017.

BRASTECC, M. e. E. **Estufa Convencional**. 2012. Disponível em: http://brastecc.com.br/produto/estufa-convencional/.

BUDNY, I. E. C. **Manual de Instruções: Aparelhos controladores de Temperatura B-400 e B-400USB**. [S.l.: s.n.], 2016. P. 1–26.

FIELDING, R. **Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content**. 2014. Disponível em: https://tools.ietf.org/html/rfc7231#section-4.

FLIP FLOP, C. E. **Acessórios, Raspberry Pi, REF. 4MDC2**. 2020. Disponível em: https://www.filipeflop.com/produto/adaptador-micro-usb-ethernet/.

GOIS, A. **Ionic Framework: Construa aplicativos para todas as plataformas mobile**. 1. ed. [S.l.]: Casa do Código, 2016. v. 1, p. 5–11. ISBN 978-85-5519-288-3.

HARTWIG, M.; VENDRAMINI, C. R. TRABALHO COLETIVO NA AGRICULTURA FAMILIAR INTEGRADA AO CAPITAL AGROINDUSTRIAL DE FUMO. **Trabalho Necessario**, Periódicos UFF, v. 6, p. 1–10, 2008. Disponível em: periodicos.uff.br/trabalhonecessario/article/view/4628.

INTEL. **Diferentes protocolos e taxas de dados de Wi-Fi**. 2021. Disponível em: https://www.intel.com.br/content/www/br/pt/support/articles/000005725/wireless/legacy-intel-wireless-products.html.

IONIC. **Ionic Framework Documentação**. 2020. Disponível em: https://ionicframework.com/docs.

LIMA, A. S. A. et al. **Aplicativo colaborativo para alerta de vulnerabilidade a alagamentos e enchentes no Vale do Itajaí**. [S.l.]: WORKSHOP DE COMPUTAÇÃO APLICADA À GESTÃO DO MEIO AMBIENTE E RECURSOS NATURAIS (WCAMA), 2018. v. 1, p. 5–11. ISBN 2595-6124. DOI: https://doi.org/10.5753/wcama.2018.2933.

MARIADB. **MariaDB SGBD Documentação**. 2020. Disponível em: https://mariadb.com/kb/pt-br/installing-mariadb-deb-files/.

MATE, F. do. **Secador de carga contínua garante tabaco de qualidade**. 2018. Disponível em: https://folhadomate.com/noticias/secador-de-carga-continua-garante-tabaco-de-qualidade/.

MAXIM, I. P. I. **Datasheets: DS1307**. [S.l.: s.n.], 2015. P. 1–14. Disponível em: https://datasheets.maximintegrated.com/en/ds/DS1307.pdf.

MURAKAMI, Y. et al. Intoxicação crônica por agrotóxicos em fumicultores. **Saúde Pública**, Scielo, v. 41, p. 563–576, 2017. Disponível em: www.scielosp.org/article/sdeb/2017.v41n113/563-576/pt/.

NARDI, J. B. **O fumo brasileiro no período colonial: lavoura, comércio e administração**. [S.l.: s.n.], 1996. v. 1, p. 150–162. ISBN 8511131175, 9788511131178.

PROFIGEN, d. B. **Curar do tabaco**. 2020. Disponível em: https://profigen.com.br/informacoes/aprenda-a-cultivar/cura-100.

RASPBERRY PI, F. **Raspberry Pi Documentation**. 2020. Disponível em: https://www.raspberrypi.org/documentation/.

SOUZA, C. Tabaco: Maior exportadora do Brasil, Souza Cruz produz um dos melhores tabacos do mundo. Souza Cruz, 2020. Disponível em: http://www.souzacruz.com.br/group/sites/SOU_AG6LVH.nsf/vwPagesWebLive/DO9YAEUN#.

SOUZA, C. Tecnologias de Cura do Tabaco. Souza Cruz, 2020. Disponível em: https://www.produtorsouzacruz.com.br/informacoes-tecnicas/colheita-e-cura/tecnologias-de-cura.

UPTON, E.; HALFACREE, G. **Raspberry PI: Guia do Usuário**. [S.l.: s.n.], 2017. v. 4, p. 8–22.

VANZELLA, A. Sistema de Detecção de Queda e Monitoramento da Frequência Cardíaca utilizando ESP8266 e protocolo MQTT. **Universidade Estadual do Oeste do Paraná**, Universidade Estadual do Oeste do Paraná, p. 16–17, 2018. Disponível em: http://tede.unioeste.br/bitstream/tede/4119/5/Alanna_Vanzella_2018.pdf.

APÊNDICE A — PARTE DO CÓDIGO FONTE DE LEITURA DO SENSOR

Código 1: Loop de leitura do sensor DHT22

```
while True:
    try:
        temperature_c = dhtDevice.temperature
        temperature_f =  temperature_c * (9 / 5) + 32
        humidity = dhtDevice.humidity
        add_medicao("sensor 1",
                    temperature_f,
                    humidity,
                    oculto)
        verificarAlertas("sensor 1",
                    temperature_f,
                    humidity)
    except RuntimeError as error:
        print('erro12', error.args[0])
        time.sleep(10)
        continue
    except Exception as error:
        print('erro23', error)
        time.sleep(30)
        continue

    oculto = oculto+1
    if oculto > 3 :
        oculto = 0

    time.sleep(config.intervalo_seconds)
```

Fonte: autoria própria (2021).

APÊNDICE B — PARTE DO CÓDIGO FONTE DA API

Código 2: Requisição GET de lista de medições

```python
@app.route('/medicoes', methods=['GET'])
def medicoes():
    # print(request.args['datainicial']) #'2020-12-15'
    # print(request.args['datafinal']) # '2020-12-25'
    try:
        conn = mariadb.connect(
            user=user,
            password=password,
            host=host,
            port=port,
            database=database)
        cur = conn.cursor()
        ocultopesq = request.args['oculto'].split(',')
        if len(ocultopesq) != 4:
            ocultopesq = [0, 1, 2, 3]
        cur.execute("SELECT identificacao, temperatura, umidade, created"
            " FROM Medicao WHERE created >= ? and created <= ? and"
            " (oculto = ? or oculto = ? or oculto = ? or oculto = ?)"
            " ORDER BY created DESC LIMIT 50",
            (request.args['datainicial'],
            request.args['datafinal'],
            ocultopesq[0],
            ocultopesq[1],
            ocultopesq[2],
            ocultopesq[3]))
        retornoBD = []
        for identificacao, temperatura, umidade, created in cur:
            retornoBD.append(
                {'Sensor': identificacao,
                'Temperatura': float(temperatura),
                 'Umidade': float(umidade),
                 'Data': f"{created}"})
        cur.close()
        conn.close()
    except mariadb.Error as e:
        print(f"Erro Mariadb: {e}")
        return jsonify({'erro': 'Erro no Banco de Dados'})
        sys.exit(1)
    return jsonify(retornoBD)
```

Fonte: autoria própria (2021).

Código 3: Requisição GET de lista de medições

```python
@app.route('/apisalvarconfig', methods=['POST', ])
def apisalvarconfig():
    try:
        if 'token' in request.json:
            return_token = auth.verify_token(request.json['token'])
            if 'autenticado' in return_token:
                print('autenticado')
            else:
                return jsonify({'erro': 'Necessario estar logado!'})
        else:
            return jsonify({'erro': 'Necessario estar logado!'})
        if request.json['config']['intervalo_seconds'] < 60:
            intervalo = 60
        else:
            intervalo = request.json['config']['intervalo_seconds']
        temp_min = request.json['config']['temp_min']
        temp_max = request.json['config']['temp_max']
        umid_min = request.json['config']['umid_min']
        umid_max = request.json['config']['umid_max']
        obs = request.json['config']['obs']
        etapa = request.json['config']['etapa']
    except:
        return jsonify({'erro': '´Parametros invalidos'})

    try:
        updateConfig(
                etapa,
                intervalo,
                temp_min,
                temp_max,
                umid_min,
                umid_max,
                obs
            )

    except:
        return jsonify({'erro': 'Erro no Banco de Dados'})

    return jsonify({'retorno': f"salvo"})
```

Fonte: autoria própria (2021).

APÊNDICE C — PARTE DO CÓDIGO FONTE APP MOBILE

Código 4: Execução em background para atualizar a medição a cada 60 segundos

```
public timerSubsc
interval(ip: string, guarda: string) {
    const source = timer(30000, 60000);
    this.timerSubsc = source.pipe(takeUntil(this.end)).subscribe(t => {
      if( ((t+1) % 9) == 0 ){
        this.buscaConfig(ip)
      }
      this.buscarMedicao(ip, guarda)
    });
}
```

Fonte: autoria própria (2021).

Código 5: Interface Tela Inicial

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-menu-button ></ion-menu-button>
    </ion-buttons>
    <ion-buttons slot="end">
      <ion-button color="primary" (click)="startInsert()" >
        <ion-icon name="add-circle" style="font-size: 32px;"></ion-icon>
      </ion-button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>
<ion-content [fullscreen]="true" *ngIf="configModulo.modulos">
  <ion-refresher slot="fixed" (ionRefresh)="doRefresh($event)">
    <ion-refresher-content
      pullingIcon="chevron-down-circle-outline"
      refreshingSpinner="circles">
     </ion-refresher-content>
  </ion-refresher>
  <ion-row *ngFor="let modulo of configModulo.modulos; let i = index">
    <app-card-estufa [modulo]="modulo" [posicao]="i"></app-card-estufa>
  </ion-row>
</ion-content>
```

Fonte: Autoria própria

Listing 1: *

Código 6: Loop de busca na rede

```
async SyncScanRede() {
    this.dispositivosIp = []
    this.scanDispositivo = true
    var ipArray = this.formIP.value.ip.split('.')
    var ip = ipArray[0]+'.'+ipArray[1]+'.'+ipArray[2]
    this.indexScan = 0
    for (this.indexScan; this.indexScan < 254; this.indexScan++) {
      this.folderService.validaDispositivo(
        ip + '.' + this.indexScan+ ':' + this.formIP.value.porta)
          .then( r => {
          if(r){
            console.log('if', r)
            this.dispositivosIp.push(r)
          }
        })
    }
    setTimeout(() => {
      this.scanDispositivo = false;
    },
    50000);
  }
```

Fonte: autoria própria (2021).