

Equipe:

Alexandre Cícero Araújo Beiruth

João Vitor Gabardo da Cunha

Leandro Cardoso Vieira

Luana Tiemann Halicki Cordeiro

PjBL Fase 4 - Relatório Final

Curitiba, PR

2023

1. WEB

1.1 Introdução

A aplicação Flask é um framework leve e flexível para a construção de aplicativos web em Python. Neste relatório, discutiremos o uso de controladores (blueprints) e visualizações (páginas HTML) em nossa aplicação Flask do nosso produto, um pote para água ou comida para pets, com dispenser de atuação automática. Além disso, abordaremos a implementação de recursos como cadastro, listagem, página inicial e controle de acesso com o Flask-Login.

1.2 Aplicação Flask

Flask é um framework leve e flexível para construir aplicativos web em Python. Ele segue o padrão arquitetural Model-View-Controller (MVC), o qual foi aplicado no projeto, onde os controladores lidam com as solicitações dos usuários, as visualizações exibem os dados e as páginas HTML, e os modelos manipulam a lógica de negócios e a persistência de dados. Flask permite criar aplicativos web de forma rápida e eficiente, fornecendo uma estrutura simples e extensível.

1.3 Controladores (Blueprints)

Em Flask, os controladores são responsáveis por lidar com as solicitações do usuário e fornecer uma resposta apropriada. Os controladores são geralmente implementados usando blueprints. Um blueprint é um objeto Flask que define um conjunto de rotas relacionadas e lógica de manipulação de solicitações. Ele organiza e estrutura o código do aplicativo em módulos reutilizáveis, facilitando o gerenciamento de diferentes partes do aplicativo.

Os blueprints permitem definir rotas, associar funções a essas rotas e especificar como o Flask deve responder às solicitações nessas rotas. Por exemplo, na aplicação um blueprint defini uma rota `"/login"` e a associa a uma função que lida

com o processo de autenticação do usuário. Outro blueprint pode definir uma rota "/" e associá-la a uma função que exibe a página index do pet.

Figura 1 – Código Blueprint

```
from flask import Blueprint, render_template, redirect, url_for, request
from models import Pet

pet = Blueprint("pet", __name__, template_folder='./views/admin/', static_folder='./static/', root_path=".")

@pet.route("/")
def pet_index():
    return render_template("/pet/pet_index.html")
```

Fonte: Arquivo pessoal, 2023.

1.4 Visualizações (Páginas HTML)

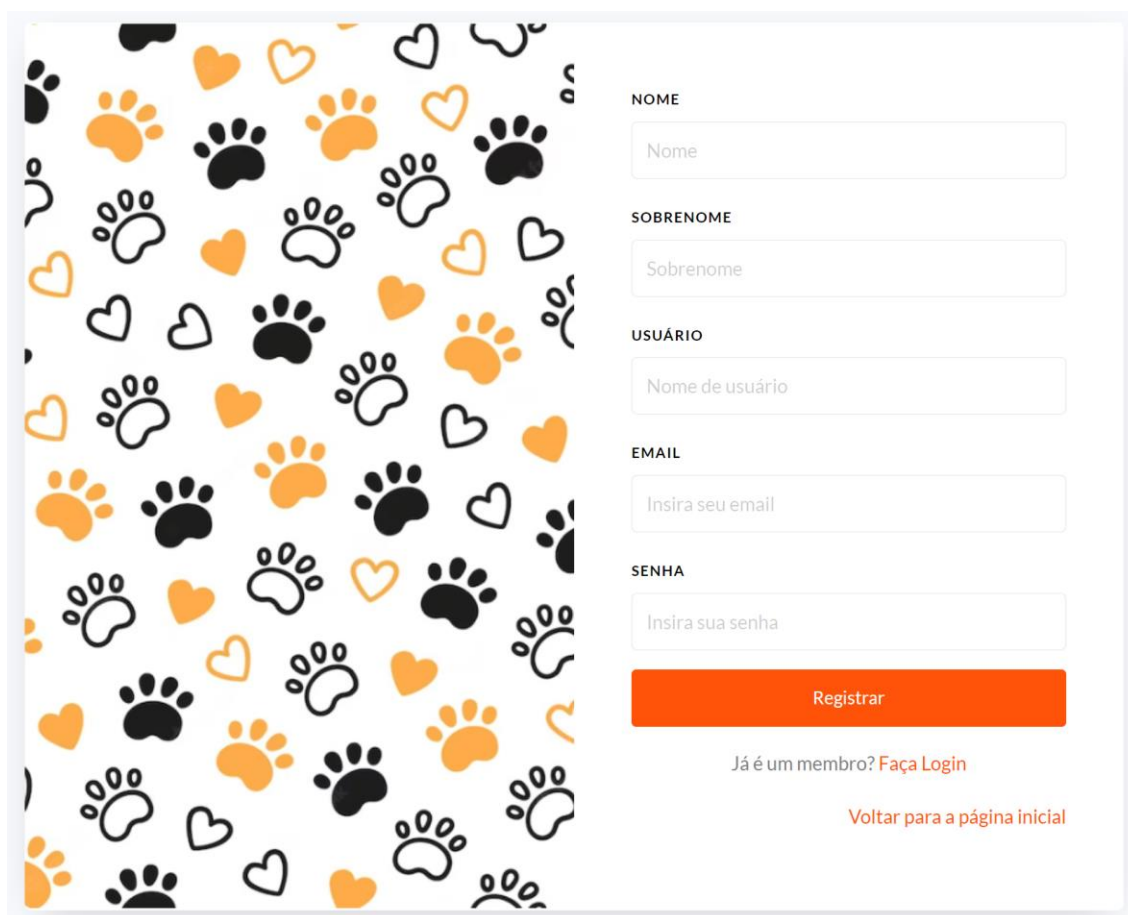
As visualizações em uma aplicação Flask são responsáveis por exibir os dados e as páginas HTML para o usuário. Uma visualização geralmente está associada a uma rota específica e é ativada quando um usuário acessa essa rota. A visualização é responsável por renderizar um modelo de página HTML com os dados necessários e enviá-lo como resposta ao navegador do usuário.

Flask oferece suporte a mecanismos de template como Jinja2, que permitem criar páginas HTML dinâmicas. Os templates Jinja2 permitem inserir variáveis e lógica nas páginas HTML, permitindo a personalização e exibição dinâmica de dados. Isso facilita a separação de preocupações entre a lógica da aplicação e a apresentação dos dados.

Um exemplo de visualização em Flask em nossa aplicação pode ser uma função que consulta um banco de dados para obter informações sobre produtos e, em seguida, renderiza um template Jinja2 com esses dados para exibição na página HTML. O usuário acessa a rota associada a essa visualização e recebe a página HTML resultante com os dados do produto.

Foi desenvolvido ao menos 4 páginas de cadastro e listagem, como por exemplo a página de cadastro do usuário, e a página de listagem de pets. Também, temos uma página inicial com a introdução do produto e página de acionamento e recuperação de dados dos atuadores e sensores do pote e dispenser.

Figura 2 – Página de inicial de cadastro.



The registration form is displayed on a page with a decorative background featuring black and orange paw prints and hearts. The form fields are as follows:

- NOME**: Input field with placeholder text "Nome".
- SOBRENOME**: Input field with placeholder text "Sobrenome".
- USUÁRIO**: Input field with placeholder text "Nome de usuário".
- EMAIL**: Input field with placeholder text "Insira seu email".
- SENHA**: Input field with placeholder text "Insira sua senha".

Below the input fields is an orange button labeled "Registrar". Underneath the button, there is a link "Já é um membro? Faça Login" and a link "Voltar para a página inicial".

Fonte: Arquivo pessoal, 2023.

Figura 3 – Página de cadastro de pet.

Nome	Raça	Idade
Nero	Frajola	2
Avatar	Frajola escuro	2
Ozzy	Gato preto	10
Nome	Raça	Idade

Fonte: Arquivo pessoal, 2023.

Figura 4 – Página sobre nós e produto.

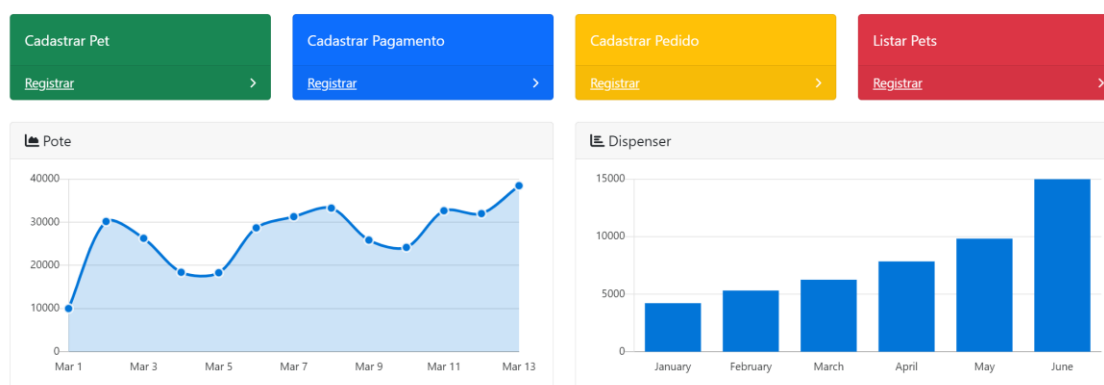


Fonte: Arquivo pessoal, 2023.

Figura 5 – Página de coletas de dados.

Página Inicial

Dashboard



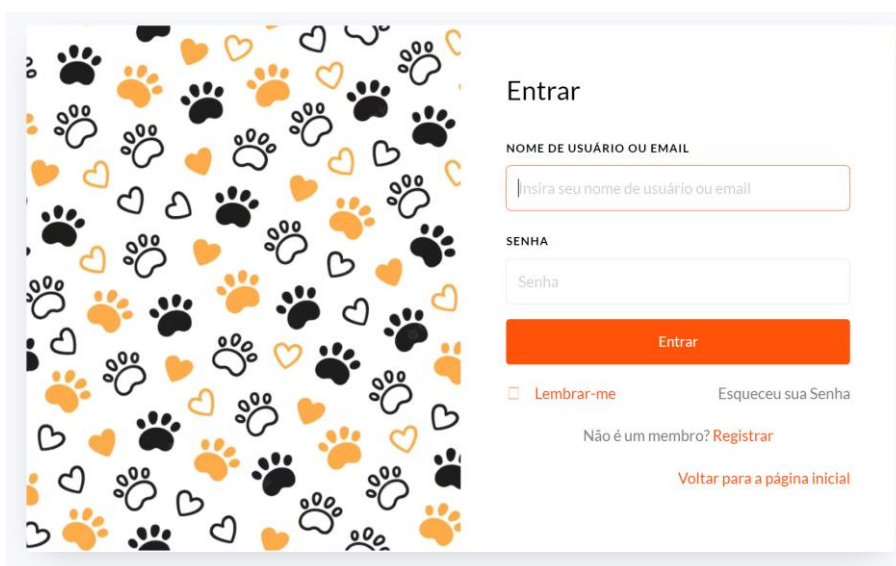
Fonte: Arquivo pessoal, 2023.

1.5 Controle de Acesso com o Flask-Login

O Flask-Login é uma extensão do Flask que simplifica a implementação de autenticação e controle de acesso em aplicativos web. É possível integrar essa extensão à nossa aplicação Flask para gerenciar o acesso de usuários às diferentes partes do sistema.

É recomendado a utilização do Flask-Login para implementar o controle de acesso na aplicação. Com isso, é possível restringir o acesso a certas páginas ou funcionalidades apenas a usuários autenticados. Com isso, definimos uma página de login que utiliza o Flask-Login para autenticar os usuários e conceder acesso às páginas de cadastro e listagem dos atuadores e sensores somente aos usuários autenticados.

Figura 6 – Página de acesso.



A imagem mostra a interface de login de um sistema web. O layout é dividido em duas seções principais: uma decorativa à esquerda e uma funcional à direita. A seção decorativa possui um fundo branco com uma repetição de ícones de patinhas em tons de laranja e preto, além de corações brancos. A seção funcional, intitulada "Entrar", contém os seguintes elementos:

- Um formulário para "NOME DE USUÁRIO OU EMAIL" com o placeholder "Insira seu nome de usuário ou email".
- Um formulário para "SENHA" com o placeholder "Senha".
- Um botão de login laranja com o texto "Entrar".
- Uma opção "Lembrar-me" com uma caixa de seleção desmarcada.
- Um link "Esqueceu sua Senha" em azul.
- Um link "Não é um membro? Registrar" em azul.
- Um link "Voltar para a página inicial" em laranja.

Fonte: Arquivo pessoal, 2023.

1.6 Conclusão:

Neste relatório, discutimos a aplicação Flask e destacamos a importância dos controladores (blueprints) e das visualizações (páginas HTML) na estruturação de um aplicativo web Flask. Propusemos a criação de pelo menos quatro páginas de cadastro e quatro páginas de listagem, além da implementação de uma página

inicial com informações do projeto e uma página para acionamento e recuperação de dados dos atuadores e sensores.

Também mencionamos a necessidade de implementar o controle de acesso utilizando o Flask-Login, garantindo que apenas usuários autenticados tenham acesso às funcionalidades restritas.

Ao seguir as diretrizes apresentadas neste relatório, será possível construir uma aplicação Flask robusta, modular e com recursos essenciais para a interação com atuadores e sensores, além de fornecer um controle de acesso eficiente para os usuários.

2. BANCO DE DADOS

2.1 Introdução

Neste relatório, abordaremos o uso de modelos SQLAlchemy para mapear objetos Python a tabelas de banco de dados e as operações básicas de CRUD (Create, Read, Update, Delete) para inserir, selecionar, editar e excluir dados do banco de dados. Além disso, discutiremos a modelagem conceitual, lógica e física do banco de dados.

2.2 Modelos SQLAlchemy

Os modelos SQLAlchemy são classes em Python que representam as tabelas do banco de dados. Eles fornecem uma interface para interagir com o banco de dados, permitindo a criação, leitura, atualização e exclusão de registros.

Por exemplo, em nosso projeto o modelo SQLAlchemy chamado "conta" para representar a tabela de contas cadastradas em um banco de dados. A definição desse modelo incluiria campos como "id", "nome" e "e-mail", que correspondem às colunas da tabela. Cada instância dessa classe representa um registro na tabela de conta. Foram feitos um total de 8 modelos, incluindo conta, pote, dispenser, pets, produto, compra, pagamento e endereço.

Figura 7 – Código modelo SQLAlchemy

```
from models.db import db
from flask_login import UserMixin

class Conta(UserMixin, db.Model):
    __tablename__ = "conta"
    id = db.Column('id', db.Integer(), primary_key=True)
    nome = db.Column(db.String(30), nullable=False, unique=False)
    usuario = db.Column(db.String(30), nullable=False, unique=True)
    email = db.Column(db.String(30), nullable=False, unique=True)
    senha = db.Column(db.String(1024), nullable=False)

    def get_conta():
        conta = Conta.query\
            .add_columns(Conta.id, Conta.nome, Conta.usuario, Conta.email, Conta.senha).all()
        return conta

    def save_conta(nome, usuario, email, senha):
        conta = Conta(nome=nome, usuario=usuario, email=email, senha=senha)
        db.session.add(conta)
        db.session.commit()

    def delete_conta(id):
        conta = Conta.query.filter(Conta.id == id).first()
        Conta.query.filter_by(actuator_type="X").delete()
        conta.delete()

    def update_conta(data):
        Conta.query.filter_by(id=data['id'])\
            .update(dict(nome=data['nome'], usuario=data['usuario'], email=data['email'],
                        senha=data['senha']))
        db.session.commit()
```

Fonte: Arquivo pessoal, 2023.

2.3 Operações CRUD no Banco de Dados

a. Inserção (Create):

Para inserir dados no banco de dados, utilizamos o método "add" do objeto de sessão do SQLAlchemy. Podemos criar uma instância do modelo desejado e adicionar essa instância à sessão antes de confirmar as alterações.

Figura 8 – Função salvar conta

```
def save_conta(nome, usuario, email, senha):  
    conta = Conta(nome=nome, usuario=usuario, email=email, senha=senha)  
    db.session.add(conta)  
    db.session.commit()
```

Fonte: Arquivo pessoal, 2023.

b. Seleção (Read):

Para selecionar dados do banco de dados, utilizamos métodos de consulta como "all" para obter todos os registros de uma tabela ou "filter_by" para filtrar registros com base em determinados critérios.

Figura 8 – Função selecionar

```
def get_conta():  
    conta = Conta.query\  
        .add_columns(Conta.id, Conta.nome, Conta.usuario, Conta.email, Conta.senha).all()  
    return conta
```

Fonte: Arquivo pessoal, 2023.

c. Atualização (Update):

Para atualizar dados no banco de dados, primeiro selecionamos o registro desejado e, em seguida, atualizamos os campos relevantes antes de confirmar as alterações.

Figura 9 – Função editar/atualizar pet

```
def edit_pet(pet_id):
    pet_upt = Pet.query.get(pet_id)

    if request.method == "POST":
        novo_nome = request.form.get("novo_nome")
        novo_raca = request.form.get("novo_raca")
        novo_idade = request.form.get("novo_idade")
        novo_tipo = request.form.get("novo_tipo")

        pet_upt.update_pet(novo_nome, novo_raca, novo_idade, novo_tipo)

        return redirect(url_for("admin.pet.view_pet"))
    else:
        return render_template("/pet/edit_pet.html", pet_upt=pet_upt)
```

Fonte: Arquivo pessoal, 2023.

d. Exclusão (Delete):

Para excluir dados do banco de dados, selecionamos o registro que desejamos remover e usamos o método "delete" para removê-lo permanentemente.

Figura 9 – Função editar/atualizar pet

```
def delete_pet(pet_id):
    pet_del = Pet.get_pet_by_id(pet_id)
    if pet_del:
        pet_del.delete_pet()
        flash('Employee deleted successfully!')
    else:
        flash('Employee not found!')
    return redirect(url_for('admin.pet.view_pet'))
```

Fonte: Arquivo pessoal, 2023.

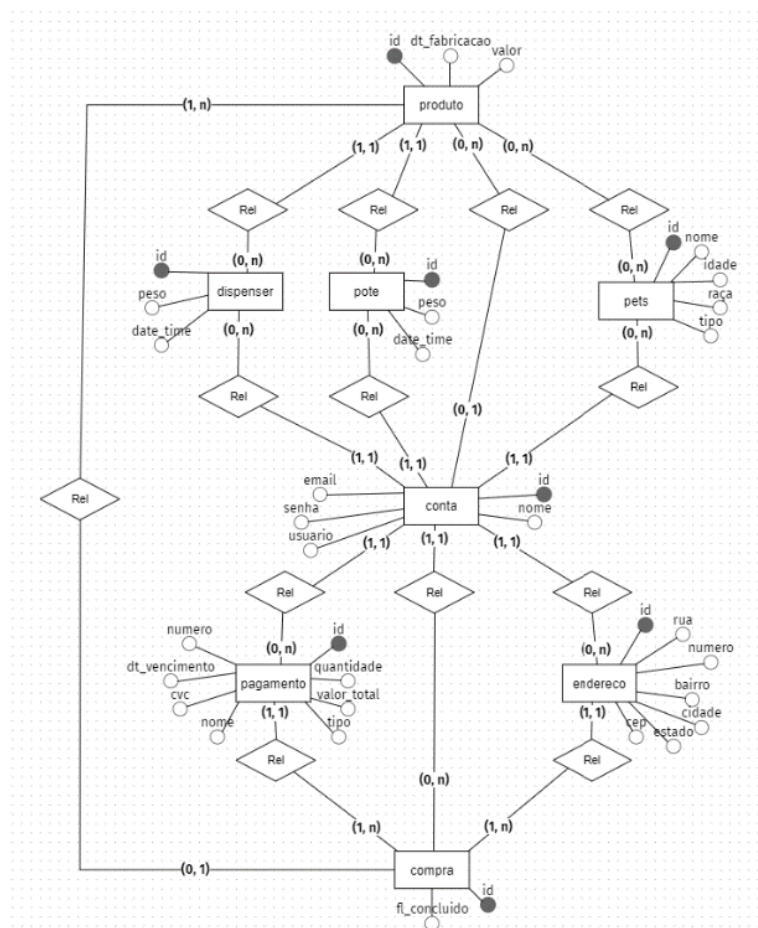
2.4 Modelagem Conceitual, Lógica e Física do Banco de Dados

A modelagem conceitual envolve a criação de um diagrama entidade-relacionamento (ER) que representa as entidades (tabelas) e os relacionamentos entre elas. A modelagem lógica inclui a tradução do diagrama ER para um modelo relacional,

especificando chaves primárias, chaves estrangeiras e restrições. Já a modelagem física envolve a implementação do modelo lógico no banco de dados escolhido. Usamos o brModelo para a criação dos modelos.

Essas etapas envolvem a análise dos requisitos do sistema, a identificação das entidades e relacionamentos, e a definição dos esquemas e estruturas do banco de dados. Decidimos por fazer o projeto focado no usuário nas operações de compra do produto e visualização dos dados dos sensores, optando por não abranger muitas partes dedicadas a administradores, portanto fizemos as seguintes modelagens conceitual, lógica e física respectivamente para nosso projeto:

Figura 10 – Banco de dados



Fonte: Arquivo pessoal, 2023.

Figura 12 – Banco de dados

```

CREATE TABLE conta
(
  id INT PRIMARY KEY,
  nome INT,
  email INT,
  senha INT,
  usuario INT,
);

CREATE TABLE pagamento
(
  id INT PRIMARY KEY,
  tipo INT,
  numero INT,
  dt_vencimento INT,
  cvc INT,
  nome INT,
  quantidade INT,
  valor_total INT,
  idconta INT,
);

CREATE TABLE compra
(
  id INT PRIMARY KEY,
  fl_concluido INT,
  idpagamento INT,
  idendereço INT,
  idconta INT,
);

CREATE TABLE endereço
(
  rua INT,
  bairro INT,
  numero INT,
  cidade INT,
  estado INT,
  id INT PRIMARY KEY,
  cep INT,
  idconta INT,
);

CREATE TABLE pets
(
  id INT PRIMARY KEY,
  nome INT,
  raça INT,
  tipo INT,
  idade INT,
  idconta INT,
);

CREATE TABLE produto
(
  dt_fabricacao INT,
  id INT PRIMARY KEY,
  valor INT,
  idconta INT,
  idcompra INT,
);

CREATE TABLE dispenser
(
  id INT PRIMARY KEY,
  peso INT,
  date_time INT,
  idconta INT,
  idproduto INT,
);

CREATE TABLE pote
(
  id INT PRIMARY KEY,
  peso INT,
  date_time INT,
  idconta INT,
  idproduto INT,
);

CREATE TABLE produto_pets
(
  id INT PRIMARY KEY,
  id INT PRIMARY KEY,
);

ALTER TABLE pagamento ADD FOREIGN KEY(idconta) REFERENCES conta (idconta)
ALTER TABLE compra ADD FOREIGN KEY(idpagamento) REFERENCES pagamento (idpagamento)
ALTER TABLE compra ADD FOREIGN KEY(idendereço) REFERENCES endereço (idendereço)
ALTER TABLE compra ADD FOREIGN KEY(idconta) REFERENCES conta (idconta)
ALTER TABLE endereço ADD FOREIGN KEY(idconta) REFERENCES conta (idconta)
ALTER TABLE pets ADD FOREIGN KEY(idconta) REFERENCES conta (idconta)
ALTER TABLE produto ADD FOREIGN KEY(idconta) REFERENCES conta (idconta)
ALTER TABLE produto ADD FOREIGN KEY(idcompra) REFERENCES compra (idcompra)
ALTER TABLE dispenser ADD FOREIGN KEY(idconta) REFERENCES conta (idconta)
ALTER TABLE dispenser ADD FOREIGN KEY(idproduto) REFERENCES produto (idproduto)
ALTER TABLE pote ADD FOREIGN KEY(idconta) REFERENCES conta (idconta)
ALTER TABLE pote ADD FOREIGN KEY(idproduto) REFERENCES produto (idproduto)
ALTER TABLE produto_pets ADD FOREIGN KEY(id) REFERENCES pets (id)
ALTER TABLE produto_pets ADD FOREIGN KEY(id) REFERENCES produto (id)

```

Fonte: Arquivo pessoal, 2023.

2.5 Conclusão

Neste relatório, discutimos o uso de modelos SQLAlchemy e as operações CRUD para inserir, selecionar, editar e excluir dados do banco de dados. Essas operações são essenciais para a interação efetiva com o banco de dados em nosso aplicativo Flask. Além disso, mencionamos a importância da modelagem conceitual, lógica e física do banco de dados como etapas fundamentais na preparação para a implementação dos modelos SQLAlchemy e das operações CRUD.

3. HARDWARE

3.1 Introdução

Neste relatório, abordaremos o hardware utilizado no projeto, incluindo sensores, atuadores e o microcontrolador ESP32. Além disso, discutiremos a esquemática do hardware e os tópicos MQTT utilizados, fornecendo explicações sobre suas funcionalidades.

3.2 Objetivo

O objetivo é apresentar os hardwares utilizados em nosso projeto. O hardware utilizado no projeto inclui:

Microcontrolador: ESP32

Sensor: dois Sensores de Peso

Atuador: Servo Motor

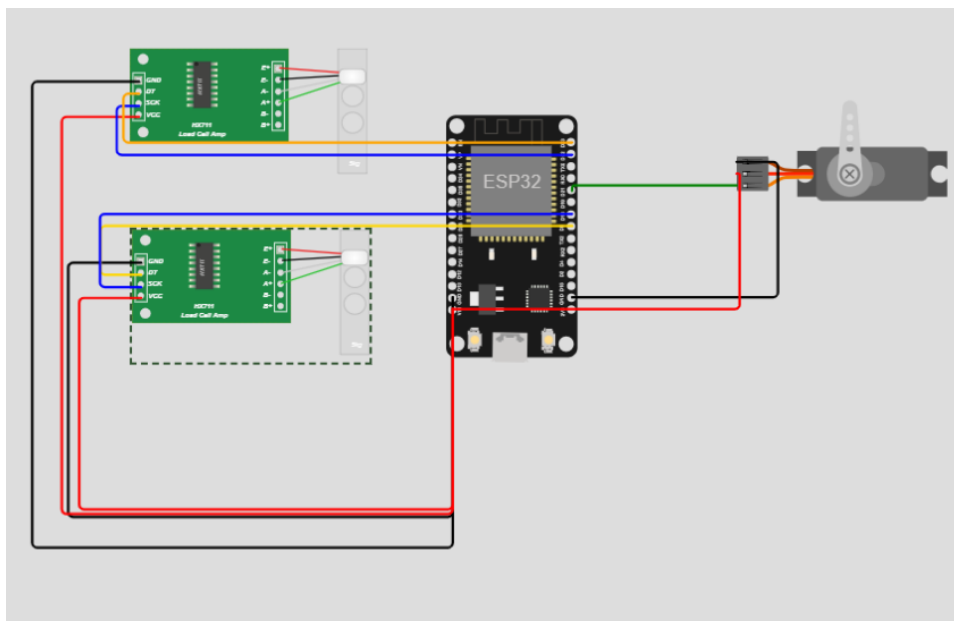
O ESP32 é responsável pelo controle e processamento do projeto, enquanto o sensor de peso é utilizado para medir a carga ou a força exercida sobre ele tanto no pote quanto no dispenser. O servo motor, por sua vez, é o atuador que realiza o movimento mecânico controlado para abrir o dispenser quando o pote estiver quase vazio.

Link do vídeo de apresentação: <https://youtu.be/EZtlELqz1vY>

3.3 Esquemática do Hardware

A esquemática do hardware descreve como os componentes estão conectados eletronicamente. É uma representação visual dos circuitos e conexões utilizados no projeto. A esquemática detalha as conexões entre o microcontrolador (ESP32), os sensores de peso e o servo motor.

Figura 13 – Simulação no Wokwi dos hardwares utilizados.



Fonte: Arquivo pessoal, 2023.

É importante observar as conexões corretas dos pinos do microcontrolador com os componentes, garantindo que os sinais sejam corretamente transmitidos e recebidos. Foi usado a plataforma Wokwi, para criar a esquemática do hardware.

Link do simulador: <https://wokwi.com/projects/367482184806022145>

3.4 Tópicos MQTT Utilizados – Parte 1

Primeiro código a ser comentado será o sketch.io, o qual representa a conexão entre os sensores utilizados no projeto, gerado na plataforma wokwi.com, simulando os potes(dispositivos) no Esp32.

Foi utilizado dois sensores de peso HX711 e um servo motor, além da conexão via Esp32.

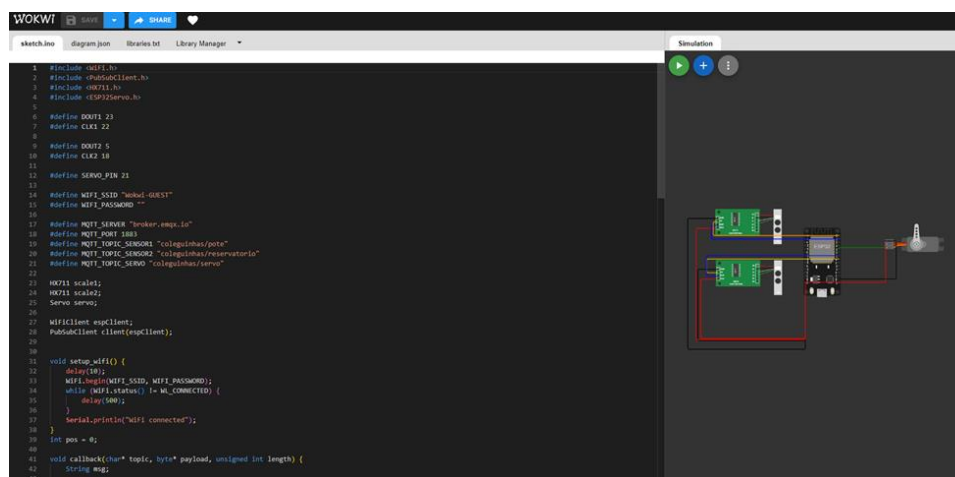
A função `setup_wifi` estabelece a conexão WiFi, após a conexão void call-back é ativado, recebendo comando para ligar o servo motor em 180 graus e aguarda 5 segundos, após término da contagem ele volta a posição inicial (90 graus).

A função void reconnect, utilizada no caso de perdas de conexão com o servidor MQTT, caso a conexão seja estabelecida aparecerá uma mensagem de sucesso e irá subscrever ao tópico “coleguinhas/servo”.

Na função encontra-se uma parte do código que anexa o servo motor ao pino especificado com os limites de pulso, e a inicialização dos sensores de peso 1 e 2, além das configurações do servidor e porta MQTT, e por fim a configuração da função call-back para mensagens recebidas.

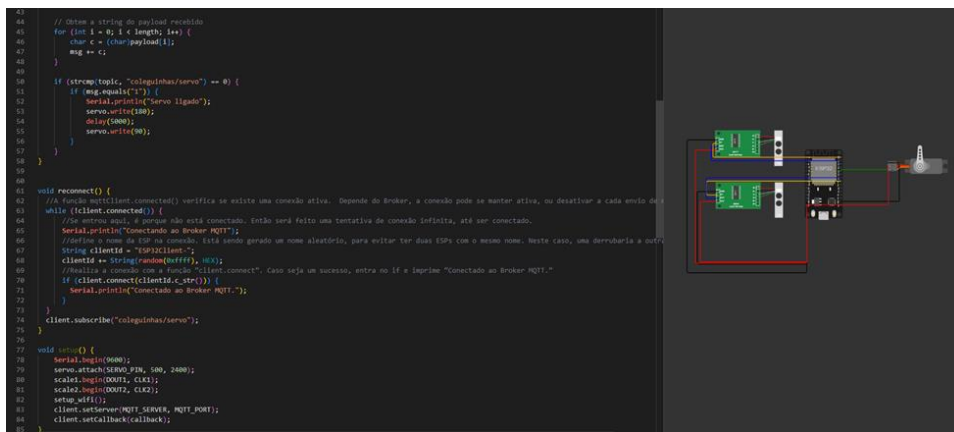
Finalizando temos a função void loop, que possui os parâmetros de reconexão com servidor MQTT, caso seja perdida, cliente.loop mantém a comunicação com servidor MQTT, as variáveis float weight1 e 2 possuem característica de obter o peso do sensor em gramas, os Serial.println imprime os pesos apresentado pelos sensores 1 e 2, além do cliente.publish publicar o peso dos sensores 1 e 2, e por final delay(5000) que aguarda 5 segundos para repetição do loop.

Figura 14 – Primeira parte do código dos dispositivos e atuadores.



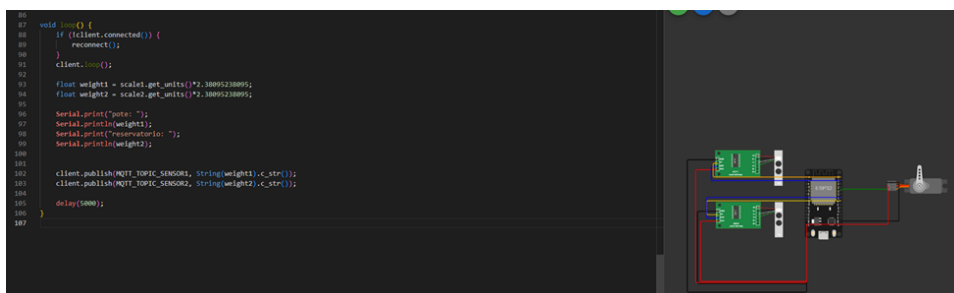
Fonte: Arquivo pessoal, 2023.

Figura 15 – Segunda parte do código dos dispositivos e atuadores.



Fonte: Arquivo pessoal, 2023.

Figura 16 – Segunda parte do código dos dispositivos e atuadores.



Fonte: Arquivo pessoal, 2023.

3.5 Tópicos MQTT Utilizados – Parte 2

Segundo código a ser comentado será o 17eserva17r.py, o qual representa a controle entre os sensores utilizados no projeto, código foi gerado na IDE Visual Studio.

Neste código existe dois tipos de potes diferentes, pote comum para ração animal e outro para 17reserva-la.

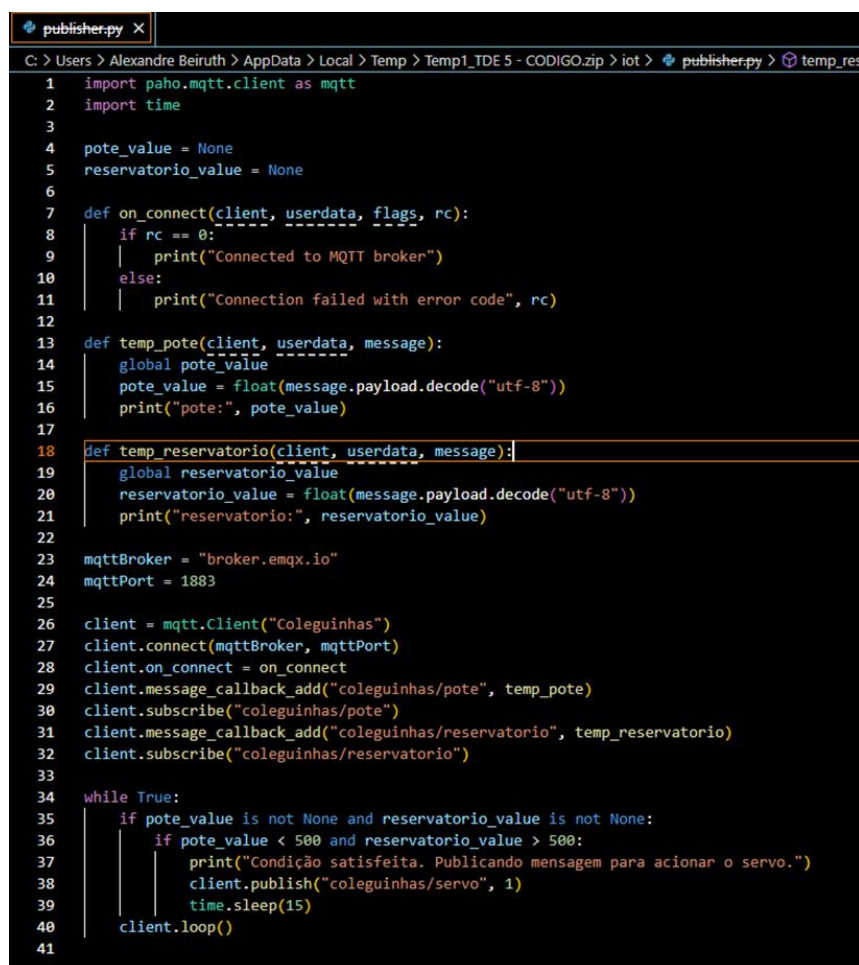
No código abaixo apresenta função que define se a conexão foi feita com broker do MQTT e se houve.

Em seguida temos duas funções que converte o payload em número de ponto flutuante tanto para pote como para o reservatório, pote_value e reservatório_value, e os imprime.

Dando continuidade possuímos no código, broker.emqx.io, que representa o endereço do broker MQTT e sua respectiva porta. Em serie temos criação do cliente MQTT, conexão ao broker MQTT, configuração do call-back para conexão e para mensagens recebidas no tópico coleguinhas/pote, subscrição do tópico coleguinhas/pote assim para mensagens recebidas do tópico coleguinhas/reservatório e subscrição coleguinhas/reservatório.

Por fim temos um laço de repetição, While True, onde verifica a veracidade dos valores do pote e reservatório recebidos, confirmação da condição especificada, True, e publica uma mensagem no tópico coleguinhas/servo no qual aguarda 15 segundos para seu acionamento, finalizando com cliente.loop que mantém a comunicação com o broker MQTT e processa os eventos seguintes.

Figura 17 – Segunda parte do código dos dispositivos e atuadores.



```

1  import paho.mqtt.client as mqtt
2  import time
3
4  pote_value = None
5  reservatorio_value = None
6
7  def on_connect(client, userdata, flags, rc):
8      if rc == 0:
9          print("Connected to MQTT broker")
10     else:
11         print("Connection failed with error code", rc)
12
13     def temp_pote(client, userdata, message):
14         global pote_value
15         pote_value = float(message.payload.decode("utf-8"))
16         print("pote:", pote_value)
17
18     def temp_reservatorio(client, userdata, message):
19         global reservatorio_value
20         reservatorio_value = float(message.payload.decode("utf-8"))
21         print("reservatorio:", reservatorio_value)
22
23     mqttBroker = "broker.emqx.io"
24     mqttPort = 1883
25
26     client = mqtt.Client("Coleguinhas")
27     client.connect(mqttBroker, mqttPort)
28     client.on_connect = on_connect
29     client.message_callback_add("coleguinhas/pote", temp_pote)
30     client.subscribe("coleguinhas/pote")
31     client.message_callback_add("coleguinhas/reservatorio", temp_reservatorio)
32     client.subscribe("coleguinhas/reservatorio")
33
34     while True:
35         if pote_value is not None and reservatorio_value is not None:
36             if pote_value < 500 and reservatorio_value > 500:
37                 print("Condição satisfeita. Publicando mensagem para acionar o servo.")
38                 client.publish("coleguinhas/servo", 1)
39                 time.sleep(15)
40     client.loop()
41

```

Fonte: Arquivo pessoal, 2023.

3.6 Conclusão

Neste relatório, abordamos o hardware utilizado, incluindo o microcontrolador ESP32, o sensor de peso e o servo motor. Também discutimos a importância da esquemática do hardware para garantir as conexões corretas dos componentes. Além disso, explicamos a utilização de tópicos MQTT e propusemos exemplos de tópicos relacionados à leitura do sensor de peso e ao controle do atuador.

Link do vídeo de apresentação PjBL 3: <https://youtu.be/EZtlELqz1vY>

4. INTEGRAÇÃO

Relatório: Integração dos Tópicos Principais da Disciplina - Web, Banco de Dados e Hardware

Introdução:

Neste relatório, abordaremos a integração dos três tópicos principais da disciplina: Web, Banco de Dados e Hardware. Exploraremos como a comunicação entre essas áreas pode ser realizada por meio da Internet das Coisas (IoT) usando o framework Flask-Mqtt. Em particular, discutiremos como os dados dos sensores e atuadores podem ser processados pela aplicação Flask, sendo salvos no banco de dados. Além disso, vamos descrever como a aplicação Flask pode enviar solicitações ao hardware por meio de tópicos específicos.

4.1. Integração dos Tópicos Principais:

A disciplina abordou três tópicos principais: Web, Banco de Dados e Hardware. A integração dessas áreas é fundamental para criar sistemas completos e funcionais. A Web permite a interação dos usuários com o sistema, o Banco de Dados armazena e recupera informações relevantes, e o Hardware permite a comunicação com o mundo físico, capturando dados dos sensores e controlando atuadores.

4.2. Comunicação IoT com o Flask-Mqtt:

A Internet das Coisas (IoT) desempenha um papel crucial na integração dos tópicos abordados. O Flask-Mqtt é um framework que permite a comunicação entre dispositivos IoT e uma aplicação Flask. Vamos explorar como essa comunicação pode ser estabelecida para processar dados dos sensores e atuadores, além de controlar o hardware.

4.3 Processamento de Dados:

Os dados dos sensores podem ser enviados para a aplicação Flask por meio do Flask-Mqtt. A aplicação Flask pode se inscrever em tópicos MQTT específicos para receber os dados dos sensores. Uma vez recebidos, esses dados podem ser processados pela aplicação Flask para extrair informações relevantes. Por exemplo, se estivermos monitorando a temperatura em um ambiente, a aplicação Flask pode processar os dados recebidos para determinar se a temperatura está acima de um limite predefinido e tomar ações apropriadas.

4.4 Armazenamento no Banco de Dados:

Após o processamento dos dados dos sensores, a aplicação Flask pode salvar as informações relevantes no banco de dados. Isso permite que os dados sejam armazenados de forma estruturada e possam ser consultados posteriormente. O banco de dados pode ser projetado para armazenar dados históricos, permitindo análises futuras e tomada de decisões baseada em dados.

4.5 Controle do Hardware:

Além de processar dados dos sensores, a aplicação Flask pode usar o Flask-Mqtt para enviar solicitações ao hardware por meio de tópicos MQTT específicos. Por exemplo, se estivermos controlando um sistema de irrigação, a aplicação Flask pode enviar uma solicitação para ligar ou desligar a válvula de irrigação enviando uma mensagem ao tópico MQTT correspondente. Essa mensagem será recebida pelo hardware, que executará a ação solicitada.

4.6 Conclusão:

A integração dos tópicos principais da disciplina - Web, Banco de Dados e Hardware - pode ser realizada por meio da comunicação IoT usando o Flask-Mqtt. Essa integração permite o processamento de dados dos sensores e atuadores pela aplicação Flask, o armazenamento de informações relevantes no banco de dados e o controle do hardware por meio de solicitações enviadas por tópicos MQTT. Essa abordagem proporciona um sistema completo e funcional, permitindo a criação de aplicações poderosas e interativas.

5. TDE 1 - METODOLOGIA ÁGIL NO DESENVOLVIMENTO DE PROJETO

5.1 Introdução

Este relatório apresenta a aplicação da metodologia ágil no desenvolvimento de um projeto de média/longa duração, com o objetivo de aprimorar as habilidades dos estudantes na condução de projetos, proporcionando maior rapidez, flexibilidade e eficiência aos processos e à conclusão das tarefas. No contexto atual, onde a agilidade e a flexibilidade são essenciais para o sucesso dos projetos, a aplicação de metodologias ágeis tem se destacado como uma abordagem eficaz para otimizar processos e alcançar melhores resultados. Para isso, foi escolhida a metodologia ágil Scrum, e a ferramenta Trello foi utilizada para suportar a sua implementação.

5.2 Conceitos e Princípios

5.2.1 Metodologia ágil Scrum

A metodologia ágil Scrum é uma abordagem de desenvolvimento de projetos que se baseia em princípios como interação contínua, colaboração, adaptação, entregas incrementais e auto-organização. Diferentemente das abordagens tradicionais, como o modelo cascata, o Scrum valoriza a flexibilidade, a adaptação às mudanças e a entrega de valor ao cliente em curtos ciclos.

O Trello é uma ferramenta de gerenciamento de projetos baseada em Kanban, que se encaixa perfeitamente na abordagem ágil do Scrum. Ele permite a criação de quadros, listas e cartões, facilitando a organização das tarefas, atribuição de responsabilidades, definição de prazos e acompanhamento do progresso em tempo real. Além disso, possui integrações com outras ferramentas populares.

5.2.2 Razões da escolha do Trello

Optamos pelo Trello devido a uma série de razões que o tornam adequado para o nosso projeto. Primeiramente, sua interface intuitiva e fácil de usar facilita o acesso e

a adoção por todos os membros da equipe, independentemente de seu nível de experiência em gerenciamento de projetos.

Outro fator que pesou em nossa escolha foi a capacidade de personalização do Trello. Podemos adaptar os quadros, listas e cartões de acordo com as necessidades específicas do nosso projeto, bem como adicionar etiquetas, anexar arquivos e fazer comentários, o que promove a colaboração e a documentação centralizada das atividades realizadas, além de ser uma interface conhecida por boa parte da equipe.

5.3 Etapas do Projeto e Definição de Metas e Prazos

Com base na metodologia ágil do Trello, definimos as seguintes etapas para o desenvolvimento do projeto:

5.3.1 Etapa 1: Planejamento

Nesta etapa, a equipe se reuniu para discutir os objetivos do projeto, identificar as principais tarefas e definir as metas iniciais. Utilizamos o quadro do Trello para criar listas como "To Do" (a fazer), "Doing" (em andamento) e "Done" (concluído), onde as tarefas foram distribuídas e acompanhadas.

Decidimos que o objetivo do projeto é criar um pote com dispenser para pets automatizado, utilizando sensores de peso, um microcontrolador eESP32 e um servo motor, então dividimos as tarefas de acordo com os requisitos do projeto estabelecidos pelo professor na primeira, segunda e terceira fase de entrega.

5.3.2 Etapa 2: Desenvolvimento da Primeira Entrega

Na etapa de desenvolvimento, a equipe dividiu as tarefas em cartões individuais, atribuindo-os aos membros do grupo. Cada cartão continha uma descrição clara da tarefa e qualquer dependência identificada. As reuniões semanais ocorreram para avaliar o progresso e identificar possíveis problemas ou obstáculos a serem resolvidos.

Para a primeira entrega, foi decidido tudo sobre o projeto, como ideia de aplicação e Framework de Design, também foi entregue a página inicial, de login, de cadastro, e interface de IoT. Ficou faltando a página de gerenciamento de usuários.

5.3.3 Etapa 3: Desenvolvimento da Segunda Entrega

Na segunda entrega, deveríamos ter entregado o site com todas as páginas necessárias e com o banco de dados funcional, incluindo o Flask-Login, páginas de cadastro, visualização e atualização de registros, e a criação de todos os modelos para a criação do banco de dados com métodos de inserção, atualização, exclusão e listagem de registros usando o SQLAlchemy, mas entregamos somente a modelagem conceitual, lógica e física do banco de dados.

Nessa etapa, enfrentamos desafios relacionados ao planejamento e à organização das tarefas, o que resultou na não conclusão de grande parte dos requisitos. Esse contratempo nos ensinou a importância de uma adequada distribuição das responsabilidades e a necessidade de um planejamento mais detalhado para evitar atrasos e garantir entregas completas.

5.3.4 Etapa 4: Desenvolvimento da Terceira Entrega

Na terceira e última entrega, entregamos os requisitos não cumpridos na segunda entrega junto a toda parte de hardware e comunicação, incluindo o esquemático do hardware e todos os tópicos MQTT utilizados explicado em formato de relatório, o código da ESP32 e o código Python realizando a comunicação e controle, e um vídeo de no máximo 5 minutos apresentando o trabalho.

5.3.5 Etapa 5: Revisão e Feedback

Após a conclusão de cada tarefa, ocorreu uma revisão interna para avaliar a qualidade do trabalho e garantir que atendesse aos requisitos definidos, e se caso não foi, quais foram os motivos. A equipe também buscou feedback de colegas e professores, permitindo melhorias contínuas e ajustes nos próximos passos.

5.3.6 Etapa 6: Entrega Final e Apresentação

A etapa final do projeto foi a entrega e a apresentação dos resultados. Foi entregue os arquivos do site concluído, todas as entregas PjBL, os relatórios completos incluindo os TDEs e o acesso a plataforma Trello.

5.4 Documentação e Reuniões de Acompanhamento

Durante a execução do projeto, foram realizadas reuniões semanais ou com base na necessidade, para avaliar o andamento das tarefas, identificar problemas e definir ações para solucioná-los. Essas reuniões foram registradas no Trello por meio de comentários nos cartões correspondentes, garantindo a documentação das decisões tomadas.

Os registros das reuniões semanais serviram como base para as conversas de acompanhamento com os professores. Nesses encontros, apresentamos a evolução do projeto, destacamos os principais desafios enfrentados e buscamos orientações para a resolução de problemas identificados.

5.5 Resultados e Considerações Finais

Ao final do projeto, nossa equipe obteve resultados positivos na apresentação final do trabalho. A aplicação da metodologia ágil do Trello permitiu um melhor controle das tarefas, maior visibilidade do progresso e maior capacidade de resposta a mudanças. A colaboração entre os membros da equipe foi aprimorada, promovendo a comunicação e o compartilhamento de conhecimentos.

Em termos de aprendizado, ganhamos experiência na aplicação de metodologias ágeis, desenvolvimento de projetos em equipe e resolução colaborativa de problemas. Reconhecemos a importância de se adaptar às mudanças e priorizar a entrega de valor ao cliente.

Em suma, a aplicação da metodologia ágil do Trello no desenvolvimento do nosso projeto trouxe benefícios significativos em termos de rapidez, flexibilidade e

eficiência. Aprendemos a importância da comunicação e da colaboração em equipe, bem como a necessidade de adaptação contínua. Essas habilidades serão valiosas em nossa trajetória acadêmica e profissional.

6. TDE 2 – Design de Aplicações Web

6.1 Introdução

Os frameworks de design front-end desempenham um papel crucial na criação de aplicações web com interfaces atraentes e responsivas. Eles oferecem um conjunto de ferramentas e componentes pré-construídos que ajudam os desenvolvedores a economizarem tempo, melhorar a produtividade e garantir uma experiência consistente para os usuários.

A escolha da ferramenta mais adequada para o propósito específico das aplicações web é fundamental para o sucesso do projeto. Diversos fatores devem ser considerados, como requisitos funcionais e de design, experiência da equipe de desenvolvimento, documentação e suporte da comunidade, curva de aprendizado e compatibilidade com as tecnologias utilizadas.

6.2 Objetivo

Nosso objetivo neste projeto foi replicar os frameworks de front-end sugeridos durante as aulas de Experiencia Criativa, ao qual se adequaram para o nosso propósito de um site de vendas do nosso dispositivo para pets, incluindo a conexão entre o site e o dispositivo citado.

6.3 Principais características dos frameworks de front-end.

Os frameworks de design front-end possuem diversas características, que torna uma ferramenta de suma importância para os desenvolvedores.

Primeira característica é a responsividade, atualmente com diversas dimensões de telas e dispositivos, é importante que as aplicações web possuam essas características para sua sobrevivência no meio digital. Os frameworks possuem criação de layouts, grids flexíveis, sistema de grades ou componentes, garantindo aplicações visualmente agradáveis e útil nos diversos tipos de dispositivos eletrônicos.

Segunda característica é a compatibilidade e suporte, os frameworks são criados a partir da compatibilidade com os navegadores modernos, isso garante que a interface gráfica funcione corretamente nas diferentes plataformas existentes. Com isso, facilita a criação de comunidades ativas que oferecem suporte técnicos, atualizações e documentação extensa, suprimindo problemas e facilitando os aprendizados sobre esses frameworks.

Terceira característica são as bibliotecas, maior parte dos frameworks oferecem bibliotecas para uso e isso amplia várias possibilidades de uso e funcionalidade. As bibliotecas oferecem campos de formulários, tipografias, ícones, grids e outros elementos comumente utilizados em web. Com essa ferramenta disponível acelera o desenvolvimento das aplicações e permite que os desenvolvedores aproveitem os componentes sem a preocupação da criação do zero.

Apesar das três características citadas acima, existem mais opções, vantagens e características dos frameworks, porém cada ferramenta vai se adaptar conforme a necessidade do desenvolvedor.

6.4 Tipo de ferramentas diferentes de frameworks

Além dos softwares mais utilizados no mercado como HTML, CSS e JavaScript, existem outras ferramentas e tecnologias que podem ser utilizadas em conjunto com esses frameworks de design front-end para aplicação web.

Pré-processadores CSS, como Stylus, Less e Sass, que são linguagens que estendem a funcionalidade do CSS padrão. Essas extensões oferecem recursos variados como, funções, importações de arquivos, mixins e variáveis. Permitindo uma estilização mais avançada e organizada do CSS padrão.

Os compiladores e bundlers, exemplo Parcel, Gulp e Webpack, são ferramentas que automatizam tarefas de front-end e podem ser utilizados para compilar arquivos CSS e Javascript, otimizando o tamanho dos arquivos, processamento de imagens e gerenciamento de dependências e mais.

Pré-processadores de HTML, exemplo Pug (antigo Jade) e Haml, são frameworks que permitem escrever códigos em HTML de maneira concisa e legível. Oferecem

recursos de alinhamento de elementos, mixins e variáveis, que ajudam a simplificar a criação e manutenção das marcações do HTML.

6.5 Necessidade do Projeto

Nosso projeto necessita das ferramentas de framework pois se trata de uma página comercial, onde oferece produto online, com página principal do produto, pagamento, registro de cliente e pet, e ainda uma página dedicada as informações transferidas do disposto para o usuário. Sem a utilização dos frameworks seria inviável a comercialização do produto criado. Portanto é de extrema importância a utilização dos frameworks citados acima.

6.6 Escolha de um Framework

BootStrap foi o framework de design escolhido para o nosso projeto pois foi ministrado durante todo semestre decorrido.

6.7 Documentação do Framework

Bootstrap é um framework de front-end popular utilizado com abrangência por desenvolvedores. Oferece vasto recurso e componentes, que podem ser usados para diversos tipos de sites.

O Bootstrap possui responsividade que facilita a criação de layouts que se adaptam automaticamente a diferentes dispositivos e telas de diversos tamanhos, essencial para site de venda como nosso projeto, pois o usuário pode acessá-lo a partir de dispositivos moveis, computadores e desktops.

O framework ainda possui uma ampla variação de componentes, cards, barras de navegação, formulários, modais entre outros. Os componentes podem ajudar na integração dos sites, permitindo a criação de interfaces atraentes e funcionais.

Sua documentação é abrangente detalhada e bem-organizada, isso inclui guia de implementação, uso, e informações sobre componentes e recursos disponíveis. Isto

possibilita o aprendizado e a utilização do framework, permitindo que o nosso site de venda utilize de forma eficiente os recursos oferecidos pelo framework.

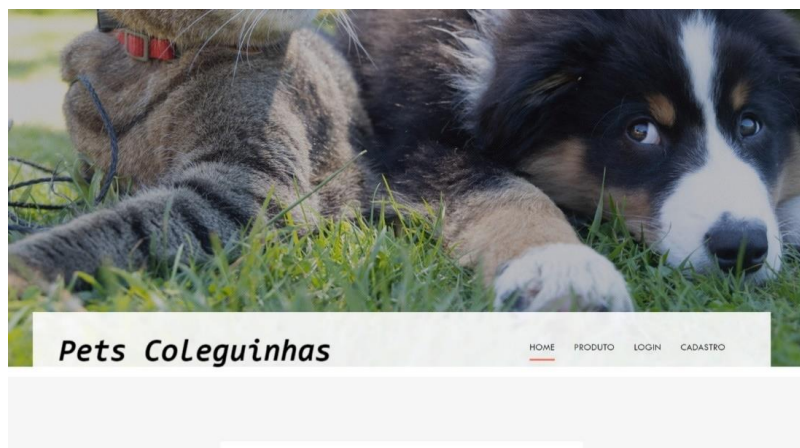
Figura 18 – Framework Bootstrap

```
<!--
header-img start
===== -->
<section id="hero-area">
  
</section>

<!--
Header start
===== -->
<nav id="navigation">
  <div class="container">
    <div class="row">
      <div class="col-md-12">
        <div class="block">
          <nav class="navbar navbar-default">
            <div class="container-fluid">
              <!-- Brand and toggle get grouped for better mobile display -->
              <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
                  data-target="#menu">
                  <span class="sr-only">Toggle navigation</span>
                  <span class="icon-bar"></span>
                  <span class="icon-bar"></span>
                  <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="#">
                  
                </a>
              </div>
              <!-- Collect the nav links, forms, and other content for toggling -->
              <div class="collapse navbar-collapse" id="menu">
                <ul class="nav navbar-nav navbar-right" id="top-nav">
                  <li><a href="#about-us">Home</a></li>
                  <li><a href="#price">produto</a></li>
                  <li><a href="/auth/login">Login</a></li>
                  <li><a href="/auth/signup">Cadastro</a></li>
                </ul>
              </div><!-- /.navbar-collapse -->
            </div><!-- /.container-fluid -->
          </nav>
        </div>
      </div><!-- .col-md-12 close -->
    </div><!-- .row close -->
  </div><!-- .container close -->
</nav><!-- header close -->
```

Fonte: Arquivo pessoal, 2023.

Figura 19 – Utilização Bootstrap na página inicial do site



Fonte: Arquivo pessoal, 2023.

Figura 20 – Utilização Bootstrap no site



Fonte: Arquivo pessoal, 2023.

6.8 Conclusão sobre benefícios e desafios do uso do Framework.

Segundo nossas experiências utilizando a ferramenta Bootstrap, podemos concluir que seus benefícios incluem, uma abrangência de recursos pré-construídos, exemplo estilos e componentes, que podem ser reutilizados, assim acelerando o desenvolvimento, e reduzindo a necessidade do uso de códigos repetidos. Sua consistência visual que consistente de estilos e componentes visuais, que garante uma visibilidade coesa em todo aplicativo. Ajudando na criação da identidade visual

do site, favorecendo uma boa experiência para o usuário final. Por fim, responsividade que se adapta automaticamente aos diversos dispositivos que iremos utilizar na aplicação do nosso projeto final.

Os desafios encontrados durante o uso do Bootstrap são, a sobrecarga de código, que por ser um framework robusto e incluir recursos que as vezes não são necessários para o nosso projeto específico. Gerando um código volumoso e as vezes possibilitando uma sobrecarga no desempenho do aplicativo.

7. TDE 3 - Utilização de Bancos de Dados NoSQL em Aplicações de IoT

7.1 Introdução

NoSQL ou Not Only SQL, é um banco de dados que se difere dos bancos de dados relacionais padrões existentes no setor de softwares. NoSQL foi desenvolvido para lidar com algumas limitações encontradas nos bancos de dados relacionais, exemplo desempenho, flexibilidade e escalabilidade.

Algumas das diferenças entre NoSQL e bancos de dados relacionais encontra-se nos modelos de dados flexíveis, primeiro ponto de diferença, os modelos relacionais seguem um molde estruturado, com esquemas e tabelas predefinidas. Entretanto o NoSQL oferece modelos mais flexíveis, pares chave-valor, grafos, colunas largas e documentos. Assim permitindo que armazenem dados não estruturados ou semiestruturados sem a necessidade de uma estrutura rígida.

Segundo ponto de diferença, escalabilidade horizontal, NoSQL é projetado para ser escalável horizontalmente, isso significa que pode lidar com um grande volume de dados, distribuindo o armazenamento e processamento através de vários nós ou servidores. Porém os bancos de dados relacionais são adequados para escalabilidade vertical, onde seu desempenho evolui por aumento de recurso de hardware em um único servidor.

Terceiro ponto de diferença, NoSQL são otimizados para leitura e gravação de dados em grande escala. Como na maioria dos casos eles não possuem esquemas rígidos, eles podem realizar operações de leitura/gravação de mais eficiente em

contraposição aos bancos relacionais, especificamente em cenários de alto tráfego ou cargas de trabalho distribuídas. Porém, em casos de consulta complexa que exija junções ou operações relacionais avançadas, banco de dados relacionais possuem um maior desempenho.

7.2 Diferenças entre NoSQL e bancos de dados relacionais

Além da existência das diferenças entre NoSQL e os bancos de dados relacionais, existe algumas vantagens que podem ser pontuadas, como: O desempenho de larga escala, onde NoSQL é projetado para alto desempenho em cenários de leitura/gravação intensiva e distribuída.

Sua escalabilidade, no qual NoSQL são altamente escaláveis e podem trabalhar com grandes volumes de banco de dados e cargas de trabalho distribuído.

Flexibilidade do esquema, que lhe permite a inserção de dados sem a necessidade de um esquema predefinido, o que é útil para aplicativos em que a sua estrutura de dados pode variar com decorrer do tempo. No entanto o NoSQL possui suas desvantagens como, menor consistência transacional, ao qual o NoSQL sacrifica a consistência transacional forte para escalabilidade e do desempenho, o que se torna uma dificuldade para aplicativos que exigem garantia de integridade de dados. Além disso podemos destacar, menor maturidade do ecossistema, no qual seu pouco tempo no mercado atual lhe dá um desvantagens pois possui menos ferramentas, recursos e suporte disponíveis em comparação aos outros bancos de dados relacionais.

Por fim, possui menos suporte para consulta complexas, em comparação novamente aos bancos de dados relacionais, o NoSQL oferece menos suporte para consultas complexas que envolva junções e operações relacionais.

7.3 Tipos de bancos de dados NoSQL

Existem vários tipos de bancos de dados NoSQL, cada qual com sua característica e uso específico, aqui descreverei alguns mais utilizados no mercado. Banco de dados de grafos, onde esse banco de dados é projetado para armazenar e consultar dados

de específico de garfos, onde os nós representam entidade e as arestas os relacionamentos. Os mais populares são, Amazon Neptune, Neo4j e ArangoDB.

Outro exemplo de banco de dados são os series temporais, onde são especialmente projetados para lidar com dados de series temporais, como os sensores de IoT que geram dados em intervalos regulares ao longo do tempo. São otimizados para consultas eficazes em dados cronológicos e geralmente oferecem recursos como compressão de dados e agregação em tempo real. Exemplos mais usados são Amazon Timestream, InfluxDB e o TimescaleDB.

Por fim, outro exemplo de NoSQL, são os bancos de dados de documentos, esses bancos armazenam dados dos formatos XML, JSON ou BSON, essa documentação pode ser grava em formato hierárquico e autocontidos, permitindo uma estrutura flexível, exemplo mais comuns são o MongoDB, Elasticsearch e Couchbase.

7.4 Utilização de bancos de dados NoSQL em aplicações de IoT

Os bancos de dados NoSQL são utilizados usualmente nas aplicações de IoT (Internet das coisas) devidos suas características que favorecem as necessidades desse ambiente. Aqui cinco pontos que favorecem sua utilização em IoT.

Primeiro ponto, modelagem flexível de dados, nas características de IoT os dados podem ser heterogêneos e variar em formato, estrutura e esquema. Banco de dados NoSQL voltados para documentos são flexíveis para lidar com essa variação de dados sem a necessidade de um esquema rígido, assim permitindo o armazenamento e consulta de forma dinâmica e adaptável.

Segundo ponto, o armazenamento de dados de sensores geralmente envolve uma quantidade grande dados gerados e em tempo real. NoSQL, como os de series temporais são adequados para armazenar e consultar esses dados. Oferecendo recursos de compressão, agregação e indexação otimizados para dados cronológicos, permitindo que dados dos sensores sejam armazenados e consulados de maneira escalável.

Terceiro ponto, escalabilidade horizontal, a grande escala e natureza distribuída das aplicações de IoT exige o armazenamento e processamento seja dimensionados

horizontalmente. NoSQL é projetado para essa escalabilidade horizontal, que favorece os grandes volumes de dados distribuídos em diversos servidores e nós.

Quarto ponto, o processamento de eventos complexos que as aplicações em IoT envolve em sua análise em tempo real, os bancos de dados NoSQL podem ser integrados a plataforma desses processamentos de eventos (event streaming) e sistemas de análise em tempo real, permitindo a detecção de padrões, tomadas de decisões automatizadas e respostas imediatas a eventos de destaque.

Quinto ponto, baixa latência e alta disponibilidade, na qual as aplicações de IoT exigem respostas rápidas e em tempo real, além da alta disponibilidade para lidar com eventos críticos. Os bancos de NoSQL são próprios para essa leitura e gravação de dados em grande escala e podem oferecer tempo de resposta rápido em comparação aos demais bancos de dados relacionais. Além da sua capacidade de replicação e distribuição permite alta disponibilidade e tolerância a falhas.

7.5 Desafios e limitações na utilização de bancos de dados NoSQL em IoT

Apesar dos bancos de dados NoSQL serem utilizados amplamente em aplicações de IoT, há permanência de alguns desafios e limitações na sua utilização. Irei citar abaixo alguns exemplos.

Exemplo primeiro, consistência dos dados que o NoSQL oferece são modelos eventuais, significa que pode haver atraso na propagação das atualizações entre os nós distribuídos, que leva a inconsistência temporais dos dados, que problemática crítica na captação de dados em tempo real.

Segundo exemplo, são as consultas complexas, onde os dados do NoSQL têm limitações em relações essa atribuição, como junções e operações relacionais avançadas. Desafio na análise de dados de IoT onde pode se exigir a correlação de informações de diferentes fontes.

Terceiro exemplo, a necessidade de uma modelagem cuidadosa, devido a flexibilidade dos bancos de dados de NoSQL, é de extrema cautela o planejamento na modelagem dos dados. O formato e a estrutura dos dados devem ser projetados de forma eficiente para atender as necessidades das consultas e análises

realizadas. A modelagem feita de forma inadequada pode levar a ineficiência e dificuldades das consultas posteriormente.

Quarto exemplo, a curva de aprendizado, no qual os bancos de NoSQL podem ter um aprendizado mais íngreme em relação aos bancos de dados relacionais, especificamente para os desenvolvedores de SQL e modelagem relacional. A compressão e adaptação aos modelos de dados NoSQL e a seleção adequada dos bancos de dados exigem tempo de aprendizado e experiência sobre essa ferramenta.

Quinto exemplo, ecossistema e as ferramentas em evolução dos bancos de dados NoSQL ainda estão em constante evolução, isso pode apresentar falta ou falhas em ferramentas, bibliotecas e suporte em comparação aos bancos de dados relacionais. W por fim, o sexto exemplo, a manutenção e o gerenciamento desse tipo específico de banco de dados NoSQL podem ser complexos, especialmente em ambientes distribuídos. O monitoramento, backup, implantação e recuperação de falhas podem ser de extrema dificuldades exigindo uma infraestrutura robusta, cara e requerente de conhecimento especializado em operações de banco de dados distribuídos.

Os bancos de dados NoSQL surgiram como alternativas aos bancos de dados relacionais, oferecendo flexibilidade, escalabilidade e desempenho em aplicações de IoT. Suas características, como modelagem flexível de dados, escalabilidade horizontal e baixa latência, os tornam adequados para lidar com os desafios enfrentados no contexto da IoT. No entanto, é importante estar ciente das limitações e desafios associados ao uso de bancos de dados NoSQL em aplicações de IoT.

A consistência dos dados, as consultas complexas, a necessidade de modelagem cuidadosa, a curva de aprendizado, a evolução do ecossistema e as demandas de manutenção e gerenciamento são fatores a serem considerados ao adotar essas soluções. Apesar das dificuldades, os benefícios dos bancos de dados NoSQL, como escalabilidade, flexibilidade e desempenho em larga escala, tornam-nos uma opção atraente para armazenar e processar os dados gerados pela IoT.

Com um planejamento adequado e compreensão das características e limitações dos bancos de dados NoSQL, é possível aproveitar ao máximo seu potencial e obter sucesso em aplicações de IoT.

8. TDE 4 – Entendendo o projeto final e seus requisitos de Hardware

8.1 Introdução

Este TDE consiste no estudo de sensores e atuadores que podem ser utilizados para a resolução do problema proposto, e desta forma definir os componentes a serem utilizados.

8.2 Objetivo

Nossa proposta de *IoT* (sensores e atuadores) foi idealizada para medição, em gramas e litros, armazenamento e análise em tempo real dos dados recebidos dos recipientes, estes serão produzidos especificamente para alimentação e hidratação de pets domésticos, ao qual o cliente possuirá todas as informações cedidas por esse dispositivo no seu aplicativo mobile.

8.3 Pesquisa de sensores e atuadores;

De acordo com as nossas pesquisas iniciais os sensores e atuadores possíveis para utilização nesse dispositivo idealizado por nos foram, o *ESP32*, *Servo Motor*, *Hx711 Load Cell* (5kg). O *ESP32* é um microcontrolador de baixo custo e de alto desempenho projetado para conectar dispositivos *IoT*, amplamente utilizado por sua conectividade Wi-Fi e Bluetooth integrada, além do amplo recursos e baixo consumo de energia, ponto chave para construção do nosso dispositivo.

O *Servo Motor* é um dispositivo eletromecânico que é usado para controlar a posição angular precisa de um eixo, ele possui a capacidade girar para uma posição especificada com base no comando recebido de um controlador, útil para o acesso da liberação da alimentação ou líquidos para os pets/clientes do nosso projeto.

Por fim, *Hx711 Load Cell* (5kg), sensor que é usado para medir forças ou pesos aplicados a ele, o mesmo amplifica o sinal da célula de carga e converte-o em um sinal digital, tornando a leitura e processamento pelos microcontroladores de fácil acesso.

8.4 Funcionamento Geral

Como verificado, os equipamentos pesquisados e definidos para nosso projeto, possuem uma vasta gama de código, bibliotecas e parâmetros, devido a sua utilização frequente e temporal no mercado computacional e robótico, podemos citar algumas fontes, tanto nas línguas portuguesa e inglesa, que possuem bibliotecas voltadas para aplicação em diversas linguagem de programação ou sistemas operacionais, como *Windows*, *Mac OS* e *Linux*, o site www.arduinolibraries.info/architectures/esp32, <https://github.com/espressif/arduino-esp32> e <https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/>, possuem informações e soluções adequadas para a utilização do *ESP32* como exemplo.

O servo motor também possui sites que demonstram e esclarecem dúvidas sobre a sua utilização, como <https://esp32io.com/tutorials/esp32-servo-motor>, <https://dronebotworkshop.com/esp32-servo/> e <https://rntlab.com/question/unit-9-esp32-control-servo-motor-library-problem/>.

O *Hx711 Load Cell* (5kg), sensor de peso utilizado no projeto, oferece fóruns e sites que apresenta códigos e manual para sua utilização, <https://www.instructables.com/Arduino-Scale-With-5kg-Load-Cell-and-HX711-Amplifi/>, <https://github.com/RobTillaart/HX711> e <https://www.cytron.io/p-5kg-load-cell-with-hx711-amplifier>.

8.5 Custos e Viabilidade

Foi verificado os custos destes equipamentos selecionados e identificados em sites que vendem produtos para robótica e computação, como o site Casa da Robotica, AliExpress, Mercado Livre. O *Hx711 Load Cell* (5kg), dependendo do combo o oferecido (inclui módulos, células e tipo de material), pode chegar em torno de R\$ 38,0 a R\$ 8,0. *ESP32* com *Wi-Fi*, câmera e *Bluetooth* pode chegar em torno de R\$ 69,0 a R\$ 18,75. E por fim, *Servo Motor*, pode chegar a R\$ 21,99 a R\$ 52,14 nas lojas pesquisadas.

8.6 Diferentes sensores e atuadores

Os diferentes sensores e atuadores que foram verificados devidos a sua mesma função, porém com custo-benefício foram o para o modulo de pesagem *Adafruit HX711 Breakout Board* que custa 14.64 Euros, para memória externa *SanDisk Ultra microSDHC 1TB* custa R\$ 696,93e substituindo o *ESP32*, o uso do *Nordic Semiconductor nRF24L01+* custa R\$22,65. Portanto avaliando os custos além disso a funcionalidade e aplicabilidade voltada para o uso, nossos dispositivos iniciais ainda se tornam menos onerosos que os equipamentos similares pesquisados.

9. TDE 5 – Realizando a integração do Hardware, MQTT e Flask

9.1 Introdução

A integração entre os três elementos (hardware, MQTT e Flask), permite controlar e monitorar dispositivos em tempo real a partir de interfaces web, facilitando a visualização dos dados coletados de sensores, além dos controles dos dispositivos atuadores e a interação com sistema remotamente.

9.2 Objetivo

O objetivo é apresentar a integração do Hardware, MQTT e Flask em nosso projeto acadêmico e simulação do dispositivo.

9.3 Referências

Site pessoal do grupo Trello <<https://trello.com/w/experienciacriativa11>>

Documento Flasq-MQTT: Disponível em:

<<https://flask-mqtt.readthedocs.io/en/latest/>

Flask-MQTT 1.1.1, Disponível em:

<<https://pypi.org/project/Flask-MQTT/>

Welcome to Flask — Flask Documentation (1.1.x). Disponível em:

<<https://flask.palletsprojects.com/>>.

Flask-Login — Flask-Login 0.7.0 documentation. Disponível em: <<https://flask-login.readthedocs.io/>>. Acesso em: 16 jun. 2023.

Flask-SQLAlchemy — Flask-SQLAlchemy Documentation (2.x). Disponível em:

<<https://flask-sqlalchemy.palletsprojects.com/>>.

SQLAlchemy Documentation — SQLAlchemy 2.0 Documentation. Disponível em:

<<https://docs.sqlalchemy.org/>>. Acesso em: 16 jun. 2023.

Documento Flasq-MQTT: Disponível em:

<<https://flasq-mqtt.readthedocs.io/en/latest/>

Flask-MQTT 1.1.1, Disponível em:

<<https://pypi.org/project/Flask-MQTT/>

SCRUM.ORG. **What is Scrum?** Disponível em: <<https://www.scrum.org/learning-series/what-is-scrum>>.

JOINER, B. **Tutorial para implementar o Scrum e o Trello na sua equipe.**

Disponível em: <<https://blog.trello.com/br/tutorial-scrum>>. Acesso em: 7 jun. 2023.

WIKIPEDIA CONTRIBUTORS. **Trello.** Disponível em:

<<https://en.wikipedia.org/wiki/Trello>>.

Documento Bootstrap: Disponível em:

<<https://getbootstrap.com/docs/>>

BootstrapBay, Disponível em:

<<https://bootstrapbay.com/blog/>>

Medium – Bootstrap Tag, Disponível em:

<<https://medium.com/tag/bootstrap>>

George Reese. "The NoSQL Handbook: A Thorough Introduction to the World of NoSQL Databases". Publisher: O'Reilly Media, 2012.

Pramod J. Sadalage and Martin Fowler. "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence". Publisher: Addison-Wesley Professional, 2012

MongoDB. "What is NoSQL?". Disponível em: <https://www.mongodb.com/nosql-explained>. Acesso em 14 de junho de 2023.

esp32io.com, Servo Motor, Disponível em:

<<https://esp32io.com/tutorials/esp32-servo-motor>>

Randomnerdtutorials.com, Esp32, Disponível em:

<<https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/>>