Radix

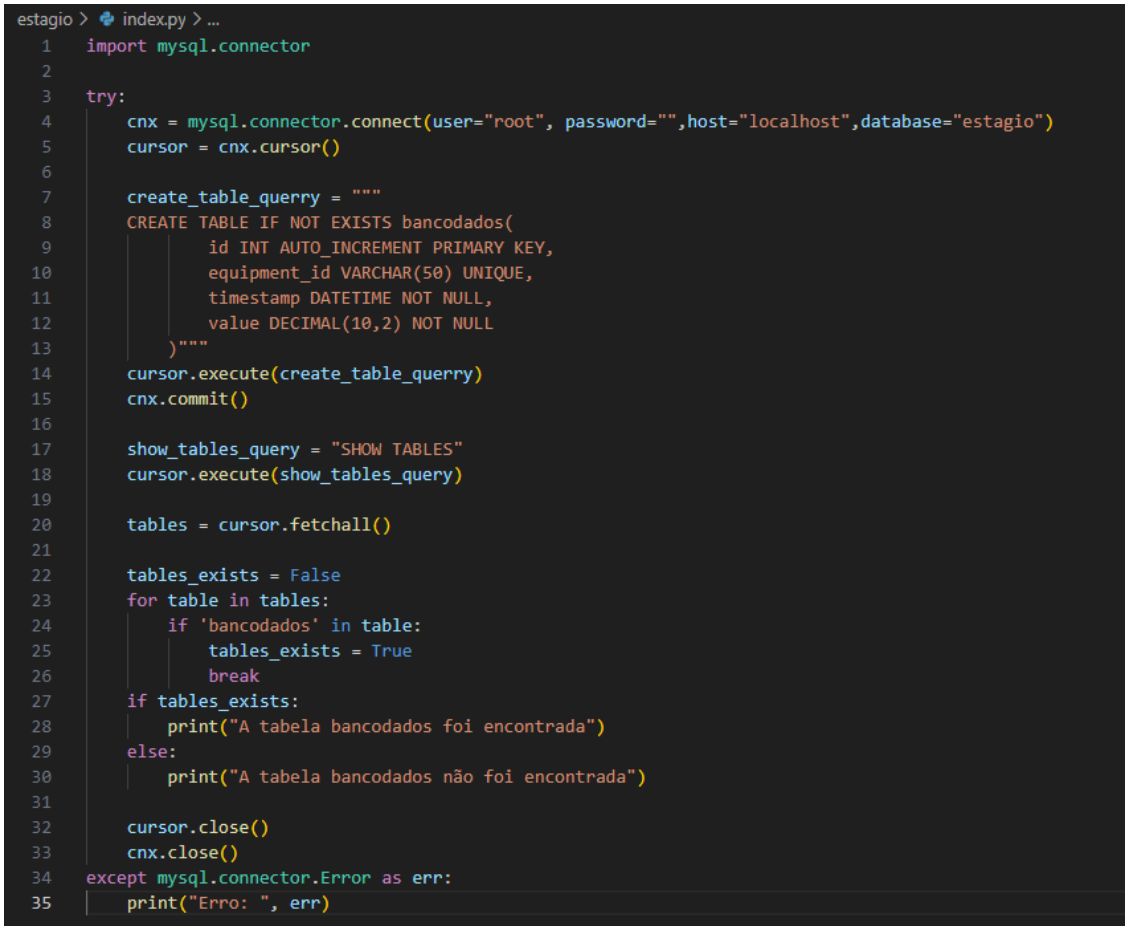
Desafio Técnico
Estágio Desenvolvimento de Software
Documentação

Alexandre Sant’Ana da Costa

1. **Inicialização:**

A criação do Desafio Radix, exigiu conhecimentos envolvendo MySQL, Python, Django e HTML. Necessitando a criação de um Banco de dados, API e otimização conforme as restrições propostas do enunciado.

Para a inicialização do projeto, será necessário executar o arquivo python **index,py** para a criação de uma tabela a um banco de dados MySQL , ele se encontra localizado dentro da pasta estágio.



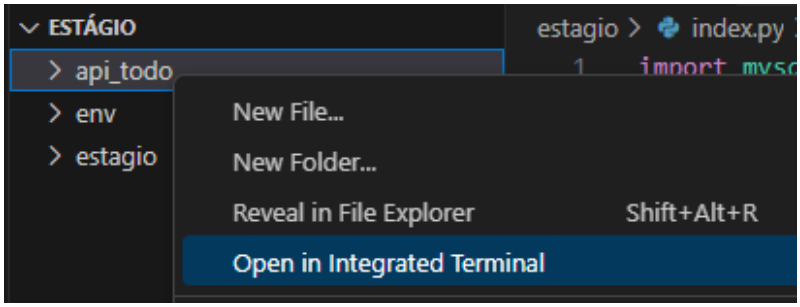
```
estagio > index.py > ...
1   import mysql.connector
2
3   try:
4       cnx = mysql.connector.connect(user="root", password="",host="localhost",database="estagio")
5       cursor = cnx.cursor()
6
7       create_table_querry = """
8       CREATE TABLE IF NOT EXISTS bancodados(
9               id INT AUTO_INCREMENT PRIMARY KEY,
10              equipment_id VARCHAR(50) UNIQUE,
11              timestamp DATETIME NOT NULL,
12              value DECIMAL(10,2) NOT NULL
13          )"""
14      cursor.execute(create_table_querry)
15      cnx.commit()
16
17      show_tables_query = "SHOW TABLES"
18      cursor.execute(show_tables_query)
19
20      tables = cursor.fetchall()
21
22      tables_exists = False
23      for table in tables:
24          if 'bancodados' in table:
25              tables_exists = True
26              break
27      if tables_exists:
28          print("A tabela bancodados foi encontrada")
29      else:
30          print("A tabela bancodados não foi encontrada")
31
32      cursor.close()
33      cnx.close()
34  except mysql.connector.Error as err:
35      print("Erro: ", err)
```

O Script em python possui a função de:

- Conectar ao Banco de Dados;
- Definir a consulta de criação da tabela, caso ela não exista;
- Executar a função de consulta;
- Confirmar alterações no Banco de Dados;
- Consultar a lista de tabelas no Banco de Dados;
- Recuperar as tabelas no Banco de Dados;
- Verificar a existência da tabela e printar caso a encontre ou não;
- Encerrar o cursor e sua conexão com o Banco de Dados;
- Tratar erros que ocorram durante a execução de algum processo no banco de dados.

O usuário deverá configurar seu próprio Banco de Dados a parte após **database = “ “.**

Após isso, programa deverá ser inicializar o servidor por meio da pasta api_todo por meio do terminal com o código: **python manage.py runserver ,** o usuário deverá entrar na api por meio do **http://localhost/** gerado.



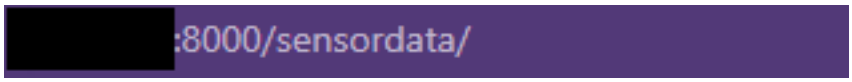

```
PS C:\Users\Alexandre\Desktop\Estágio\api_todo> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 19, 2024 - 15:05:12
Django version 5.0.4, using settings 'api_todo.settings'
Starting development server at http://[redacted]:8000/
Quit the server with CTRL-BREAK.
```

2. **Navegação pela API:**

Ao ser direcionado para a API, ela irá possuir quatro funções com base na resolução da proposta do desafio.

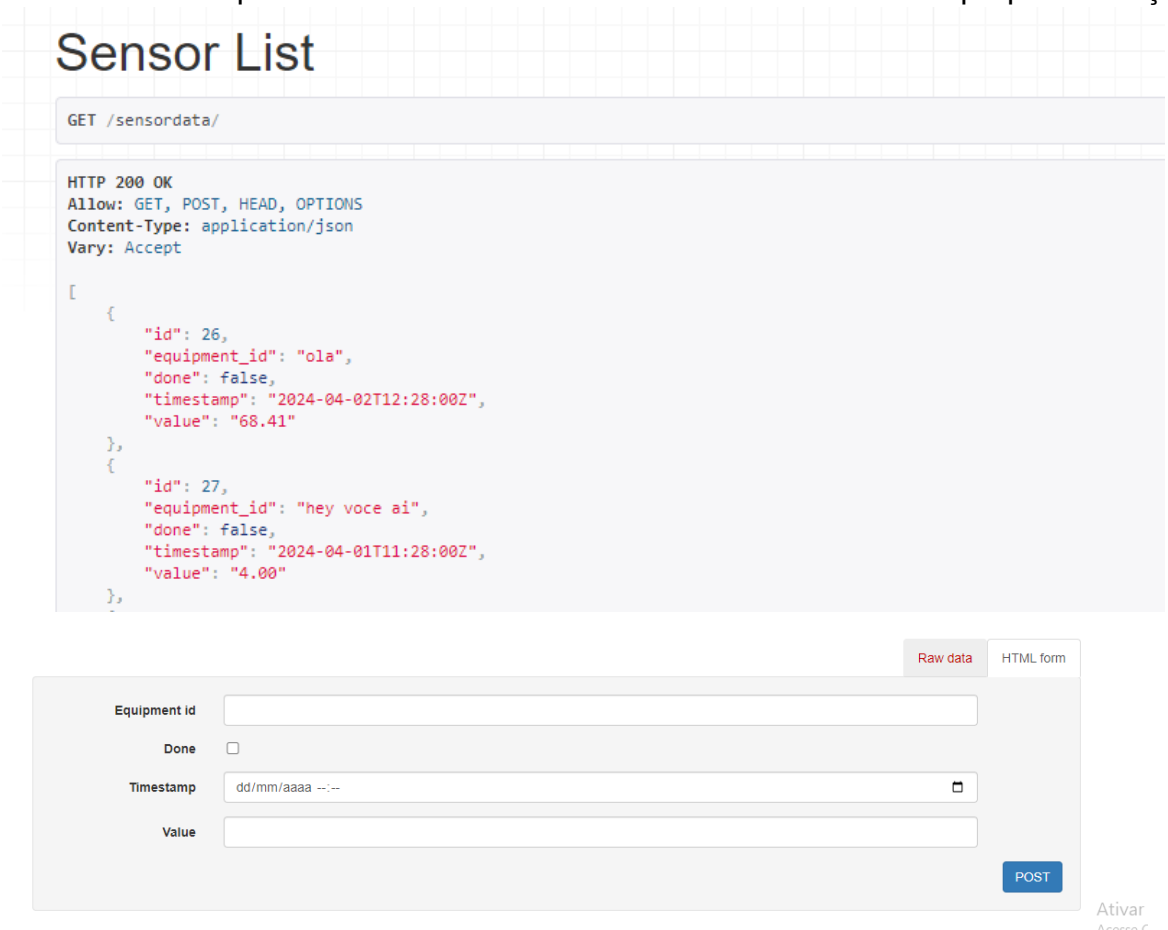
```
1. admin/
2. sensordata/
3. upload-csv/
4. gerar-grafico/
5. ver-grafico/
```

Para acessar alguma das funções na API, será necessário a alteração no URL, seguindo o modelo de **localhost/comando-que-deseja/.**



Sensordata:

Uma função baseada em representar uma representação geral do banco de dados, mostrando as tabelas criadas nele. As Sendo possível adicionar novas tabelas manualmente por meio Raw ou HTML contido na própria função.



Os dados de model na tabela se define pela seguinte estrutura:

{

“Id:”

“equipment_id:”

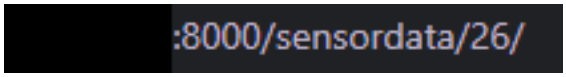
“done:”

“timestamp:”

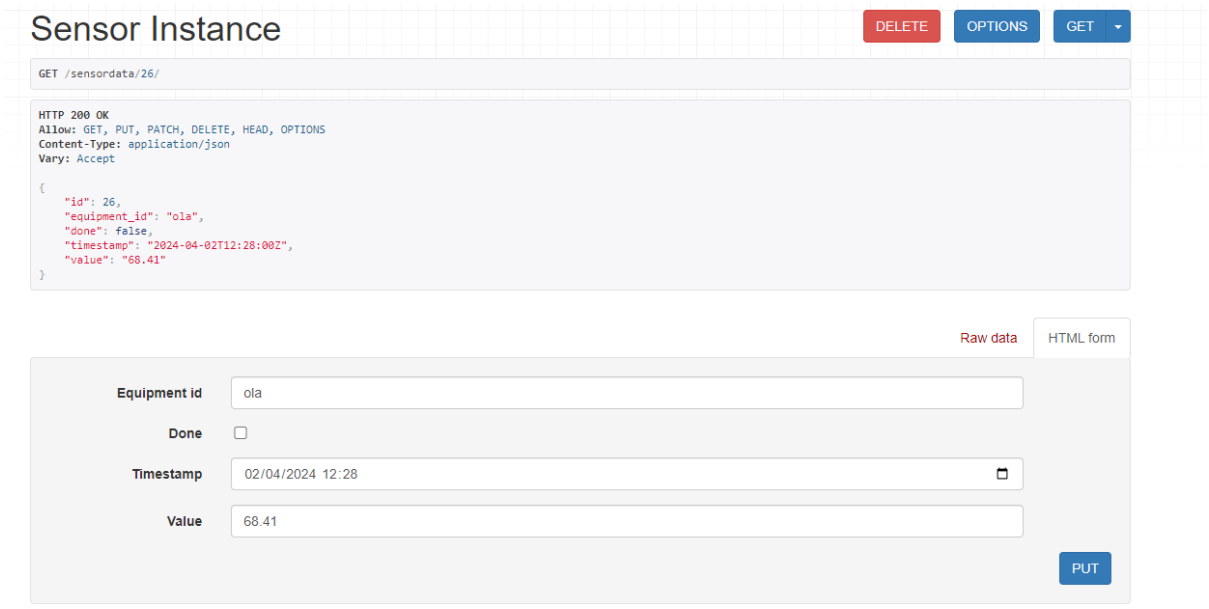
“value:”

}

Caso o usuário deseje realizar a alteração ou exclusão de algum dado da tabela, ele deverá acrescentar ao URL o ID gerado na estrutura, seguindo o modelo de l**ocalhost/sensordata/id/ .**



Após isso, o usuário será redirecionado para a função de edição ou exclusão de tabelas no Banco de Dados.

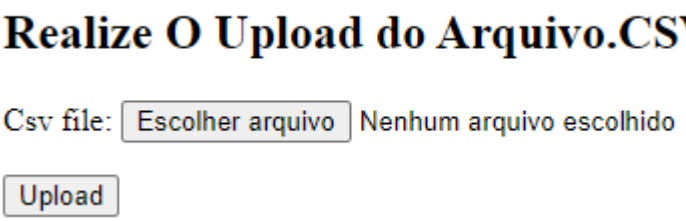


Para realizar a execução da API de sensordata, foi necessário adotar um Viewset com base em Router, para gerar as urls necessárias para executar da melhor forma para prezar pela otimização e praticidade do código.

O código utiliza das funções ‘GET’, ‘POST’, ‘PUT’ e ‘DELETE’ abaixo como base para estruturas da API.

Upload-csv

Trata-se de uma URL que possui como objetivo receber um arquivo CSV, decodificá-lo e adicioná-lo ao banco de dados em tempo real. Para o usuário ser redirecionado para essa função, deverá executar uma mudança na URL: **Localhost/upload-csv/ .**



Após o envio do arquivo.csv, o usuário será redirecionado para a URL sensordata, para que possa visualizar a adição dos dados em tempo real na tabela do banco de dados. Utilizando como base os critérios de ‘equipmentId’ , ‘timestamp’ e ‘value’ com cada um sendo definido através da leitura da tabela em CSV.

```
def upload_csv(request):
    if request.method == 'POST':
        form = CSVUploadForm(request.POST, request.FILES)
        if form.is_valid():
            csv_file= request.FILES['csv_file']
            decoded_file= csv_file.read().decode('utf-8').splitlines()
            csv_reader= csv.DictReader(decoded_file)
            for row in csv_reader:
                SensorData.objects.create(
                    equipment_id=row['equipmentId'],
                    timestamp=row['timestamp'],
                    value=row['value']
                )
            return redirect ('sensordata-list')
    else:
        form = CSVUploadForm()
    return render(request, 'upload_csv.html', {'form': form})
```

O código deverá realizar as seguintes funções:

- Requisitar o método ‘Post’;
- Realizar uma instância de formulário de upload CSV;
- Obter o arquivo CSV;
- Criar um leitor de CSV com base na decodificação;
- Criar um objetivo com base nos dados de cada linha do CSV;
- Redirecionar para a lista do Banco de Dados;
- Caso o método de requisição não seja ‘Post’, deverá ser criado um forms;
- Renderizar a página html do csv.

Gerar-Grafico:

Função realizada através do uso da biblioteca python chamada mathplotlib para a criação de gráficos, busca utilizar o cálculo de tempo e média de valores de sensores para gerar três tipos de gráficos: diários, semanais e mensais.

O usuário será redirecionado a partir da URL **localhost/gerar-grafico/** e irá inicialmente para uma página intermediária que irá puxar os dados já inseridos no banco de dados e solicitar a geração de gráficos, para em seguida ser redirecionado para a função ‘ver-grafico’ para permitir a visualização dos gráficos.

```python
def calcular_media(periodo):
    agora= timezone.now()
    inicio= datetime(1980, 1, 1)
    periodos= {···
    if periodo not in periodos:···

    inicio = periodos[periodo]['inicio']
    registro = periodos[periodo]['registro']

    sensor_dados = SensorData.objects.filter(timestamp__gte=inicio).annotate(···
    return sensor_dados

def gerar_grafico(request):
    sensores = SensorData.objects.values_list('equipment_id', flat=True).distinct()
    context = {'sensores': sensores}

    # Processo de cálculo e geração de média
    dados_diarios = calcular_media('diário')
    arquivo_grafico_diario = gerar_grafico_medias('diário', dados_diarios)

    dados_semanais = calcular_media('semanal')
    arquivo_grafico_semanal = gerar_grafico_medias('semanal', dados_semanais)

    dados_mensais = calcular_media('mensal')
    arquivo_grafico_mensal = gerar_grafico_medias('mensal', dados_mensais)

    context['arquivo_grafico_diario'] = arquivo_grafico_diario
    context['arquivo_grafico_semanal'] = arquivo_grafico_semanal
    context['arquivo_grafico_mensal'] = arquivo_grafico_mensal

    return render(request, 'carregar_graficos.html', context)
```

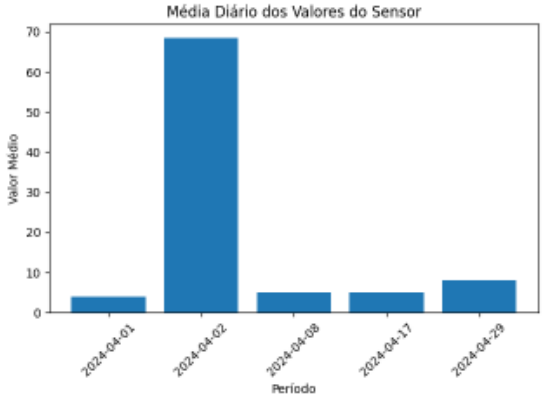
```python
def gerar_grafico_medias(periodo, dados):
    labels = [d['periodo_registro'].strftime('%Y-%m-%d') for d in dados]
    valores = [d['media'] for d in dados]

    plt.bar(labels, valores)
    plt.xlabel('Período')
    plt.ylabel('Valor Médio')
    plt.title(f'Média {periodo.capitalize()} dos Valores do Sensor')
    plt.xticks(rotation=45)
    plt.tight_layout()

    # Salvar o gráfico como .png
    nome_arquivo = f'media{periodo}.png'
    caminho_arquivo_static = os.path.join(settings.STATIC_ROOT, nome_arquivo)
    plt.savefig(caminho_arquivo_static)

    # Salvar o gráfico na pasta staticfiles
    caminho_arquivo_staticfiles = os.path.join(settings.STATICFILES_DIRS[0], nome_arquivo)
    plt.savefig(caminho_arquivo_staticfiles)
    plt.close()

    return nome_arquivo

def ver_grafico(request):
    return render(request, 'sensor_media.html')
```

Ver-grafico:

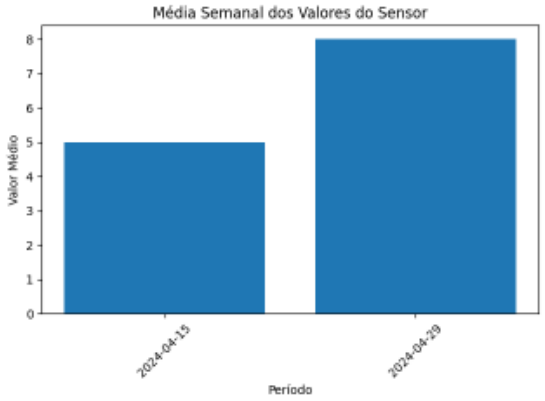
Possui a função de recursividade para a visualização de um gráfico já existente, caso o usuário deseje visualizar aquele que foi anteriormente gerado sem que haja atualizações. Podendo ser visualizado a partir da URL **localhost/ver-grafico/**

Médias dos Valores do Sensor

Média Diária



Média Semanal



Média Mensal