

Experiência Criativa: Criando Soluções Computacionais

RA3

Módulo 3: Manipulação de dados
com Flask SQLAlchemy :
Manipulação de registros.

Andrey Cabral
andrey.cabral@pucpr.br

PUC

AGENDA

- Arquitetura MVC
 - Model (Modelo).
 - View (Visualização).
 - Controller (Controladores).
- Flask SQLAlchemy.
 - Criação de modelos e Tabelas.
 - Criação da estrutura do Banco de Dados.
 - Inserção de dados.
 - Realização de consultas.
 - Atualização de registros.

Exemplo 5 – Atualizar registros no banco

Sensores:

Sensor	Marca	Modelo	Tópico	Unidade	Ativo?		
2	1	1	1	1	Ativo	Editar	Deletar

Adicionar Sensor

Retornar

```
@sensor_.route('/edit_sensor')
def edit_sensor():
    id = request.args.get('id', None)
    sensor = Sensor.get_single_sensor(id)
    return render_template("update_sensor.html", sensor = sensor)
```

- ❑ Criar um método chamado edit_sensor no sensors_controller.py

Exemplo 5 – Atualizar registros no banco

```
def get_single_sensor(id):  
    sensor = Sensor.query.filter(Sensor.devices_id == id).first()  
    if sensor is not None:  
  
        sensor = Sensor.query.filter(Sensor.devices_id == id)\  
            .join(Device).add_columns(Device.id, Device.name, Device.brand,  
                Device.model, Device.is_active, Sensor.topic, Sensor.unit).first()  
  
    return [sensor]
```

- ❑ Criar método para obter os dados do sensor selecionado para edição.

Exemplo 5 – Atualizar registros no banco

Nome:

Marca:

Modelo:

Unidade de Medida:

Tópico MQTT:

☒ Ativo?

Alterar

```
@sensor_.route('/update_sensor', methods=['POST'])
def update_sensor():
    id = request.form.get("id")
    name = request.form.get("name")
    brand = request.form.get("brand")
    model = request.form.get("model")
    topic = request.form.get("topic")
    unit = request.form.get("unit")
    is_active = True if request.form.get("is_active") == "on" else False

    sensors = Sensor.update_sensor(id, name, brand, model, topic, unit, is_active)

    return render_template("sensors.html", sensors = sensors)
```

- ❑ Criar um método chamado update_sensor no sensors_controller.py

Exemplo 5 – Atualizar registros no banco

```
def update_sensor(id,name, brand, model, topic, unit, is_active):  
    device = Device.query.filter(Device.id == id).first()  
    sensor = Sensor.query.filter(Sensor.devices_id == id).first()  
    if device is not None:  
        device.name = name  
        device.brand = brand  
        device.model = model  
        sensor.topic = topic  
        sensor.unit = unit  
        device.is_active = is_active  
        db.session.commit()  
    return Sensor.get_sensors()
```

- ❑ No modelo sensor, verificar presença do device no banco.
- ❑ Alterar com db.commit()

Exercício 5 – PBL3



- ❑ Criar funções para permitir edição de atuadores.

Exemplo 6 – Deletar registros no banco

- ❑ Para excluir registros o SQLAlchemy possui o método delete:
- ❑ Criar método de delete no modelo sensors.py

```
def delete_sensor(id):  
    device = Device.query.filter(Device.id == id).first()  
    sensor = Sensor.query.filter(Sensor.devices_id == id).first()  
  
    db.session.delete(sensor)  
    db.session.delete(device)  
    db.session.commit()  
  
    return Sensor.get_sensors()
```


Exemplo 6 – Deletar registros no banco

- ❑ Criar método `del_sensor` para deletar sensores e devices no controlador de sensores.

```
@sensor_.route('/del_sensor', methods=['GET'])  
def del_sensor():  
    id = request.args.get('id', None)  
    sensors = Sensor.delete_sensor(id)  
    return render_template("sensors.html", sensors = sensors)
```

Sensores:

Sensor	Marca	Modelo	Tópico	Unidade	Ativo?	
1	1	1	1	1	Ativo	Editar Deletar

Adicionar Sensor

Retornar

Exercício 6 – PBL3

- ❑ Criar funções para permitir exclusão de atuadores.

Exemplo 7 - Relacionamento 1:N

```
from models.db import db
from models.iot.sensors import Sensor
from datetime import datetime

class Read(db.Model):
    __tablename__ = 'read'
    id = db.Column('id', db.Integer, nullable = False, primary_key=True)
    read_datetime = db.Column(db.DateTime(), nullable = False)
    sensors_id = db.Column(db.Integer, db.ForeignKey(Sensor.id), nullable = False)
    value = db.Column(db.Float, nullable = True)
```

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
read_datetime	DATETIME	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
sensors_id	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
value	FLOAT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

- ❑ Criar modelo read.py.
- ❑ Adicionar classe no arquivo __init__.py

```
# __init__.py
...
from models.iot.read import Read
```

Exemplo 7 – Tabela Read

- ❑ Criar arquivo blueprint reads_controller.py para controlar chamadas do front e acesso ao banco.

```
#reads_controller.py
from flask import Blueprint, request
from models.iot.read import Read

read = Blueprint("read",__name__, template_folder="views")
```

- ❑ Registre o blueprint reads_controller no app_controller.py

```
app.register_blueprint(read, url_prefix='/')
```


Exemplo 7 - Read

- Adicione a função de salvar dados no banco no modelo read.py.

```
def save_read(topic, value):  
    sensor = Sensor.query.filter(Sensor.topic == topic).first()  
    device = Device.query.filter(Device.id == sensor.devices_id).first()  
    if (sensor is not None) and (device.is_active==True):  
        read = Read( read_datetime = datetime.now(), sensors_id = sensor.id, value = float(value) )  
        db.session.add(read)  
        db.session.commit()
```

Exemplo 7 - Read

- No arquivo app_controller.py flask_mqtt.

```
import json
from flask_mqtt import Mqtt
...
app.config['MQTT_BROKER_URL'] = 'mqtt-dashboard.com'
app.config['MQTT_BROKER_PORT'] = 1883

app.config['MQTT_USERNAME'] = '' # Set this item when you need to verify username and password
app.config['MQTT_PASSWORD'] = '' # Set this item when you need to verify username and password
app.config['MQTT_KEEPALIVE'] = 5000 # Set KeepAlive time in seconds
app.config['MQTT_TLS_ENABLED'] = False # If your broker supports TLS, set it True

mqtt_client= Mqtt()
mqtt_client.init_app(app)

topic_subscribe = "/aula_flask/"
```

Exemplo 7 - Read

- No arquivo `app_controller.py`, criar método de conexão e subscribe.

```
@mqtt_client.on_connect()
def handle_connect(client, userdata, flags, rc):
    if rc == 0:
        print('Broker Connected successfully')
        mqtt_client.subscribe(topic_subscribe) # subscribe topic
    else:
        print('Bad connection. Code:', rc)
```

Exemplo 7 - Read

- No arquivo app_controller.py, adicionar função para escrita no banco.

```
@mqtt_client.on_message()
def handle_mqtt_message(client, userdata, message):
    if(message.topic==topic_subscribe):
        js = json.loads(message.payload.decode())
        try:
            with app.app_context():
                Read.save_read(js["sensor"],js["valor"])
        except:
            pass
```

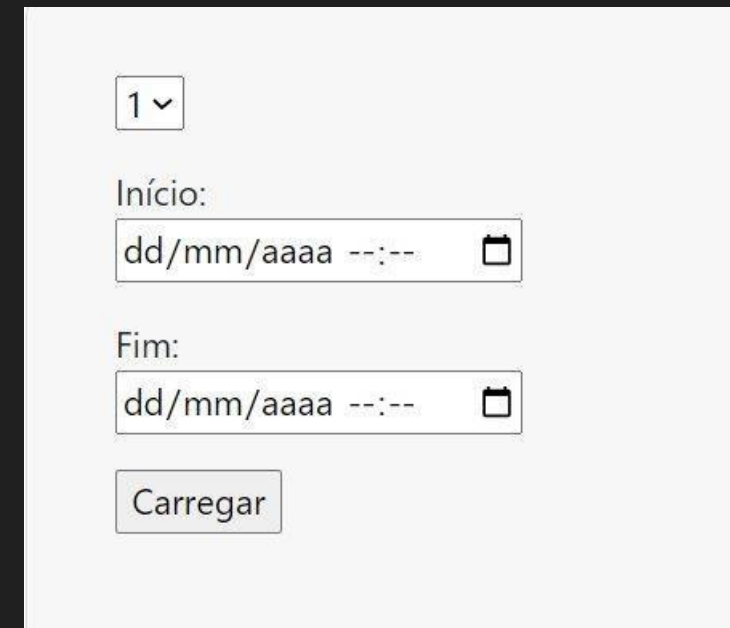

Exercício 7 – PBL3

- ❑ Criar modelo write para salvar dados dos atuadores.
- ❑ Criar classe Write.
- ❑ Criar controle write_controller.py.
- ❑ Salvar dados de atuação na tabela write.

Exemplo 8 – Mostrar dados históricos

- No controle `reads_controller.py`, adicionar o método `history_read`. Ele irá repassar os sensores para uma tela de seleção de dados históricos.

```
@read.route("/history_read")
def history_read():
    sensors = Sensor.get_sensors()
    read = {}
    return render_template("history_read.html", sensors = sensors, read = read)
```



1 ▾

Início:
dd/mm/aaaa --:--

Fim:
dd/mm/aaaa --:--

Carregar

Exemplo 8 – Mostrar dados históricos

- Adicionar método `get_read` para exibir valores na tela.

```
@read.route("/get_read", methods=['POST'])
def get_read():
    if request.method == 'POST':
        id = request.form['id']
        start = request.form['start']
        end = request.form['end']
        read = Read.get_read(id, start, end)

        sensors = Sensor.get_sensors()
        return render_template("history_read.html", sensors = sensors, read = read)
```

Exemplo 8 – Mostrar dados históricos

- Adicionar método `get_read` no modelo `read.py`.

1

Início:
dd/mm/aaaa --:--

Fim:
dd/mm/aaaa --:--

Carregar

Sensores:

Sensor	Valor	Horário
2	73.0	2024-04-20 20:00:25
2	73.0	2024-04-20 20:00:25

```
def get_read(device_id, start, end):  
    sensor = Sensor.query.filter(Sensor.devices_id == device_id).first()  
    read = Read.query.filter(Read.sensors_id == sensor.id,  
                             Read.read_datetime > start,  
                             Read.read_datetime < end).all()  
  
    return read
```


Exercício 8 – PBL3



- ❑ Criar função para exibir dados históricos dos atuadores.



- <https://flask-sqlalchemy.palletsprojects.com/en/3.0.x/>
- <https://jinja.palletsprojects.com/en/3.1.x/>
- <https://flask.palletsprojects.com/en/2.2.x/>