**Importação de Bibliotecas**

Significado de pontos na importação de pacotes:

* Um ponto: Usado quando os arquivos que serão importados estão dentro do mesmo pacote
* Dois pontos: Usado quando os arquivos que serão importados estão em pacotes distintos. Ex:(pasta atual controllers) : **from ..models.user import User**

Usa-se “as” na importação de bibliotecas para renomear o nome da biblioteca com o nome que você desejar.

Ex:

**import** **fibo** **as** **fib #Agora para chamar a biblioteca durante a execução do script, usa o nome fib!**

Passos para que o projeto em MVC seja iniciado corretamente:

1. Verificar se todas as pastas têm arquivo \_\_init\_\_.py
2. Verificar se dentro dos \_\_init\_\_.py estão sendo mencionados os arquivos que estejam dentro da mesma pasta que ele. **Ex:**  
   Interface gráfica do usuário

   Descrição gerada automaticamente com confiança média
3. Verificar se startou o banco de dados. Python (nome do banco.sql)
4. Verificar se startou o app.py, arquivo geral que não está dentro de nenhuma pasta

Atenção: Toda edição realizada no banco de dados coloca-se o commit no final. Para salvar sua modificação!!!

**Resumo sobre SQLAIchemy**

* **O create\_engine** – cria o seu banco de dados e estabelece conexão com um já existente. Ex:

**Texto

Descrição gerada automaticamente**

* **Sqlalchemy.orm** – É uma biblioteca que permite mapear classes Python para tabelas em um banco de dados relacional, e objetos Python para registros (linhas) dessas tabelas.
* **Session.query(nome da tabela)** – permite realizar uma consulta de dados na tabela indicada.

Funções do query():

1. **.all()** - Se quiser consultar todos os valores da tabela
2. **.first()** – Se quiser consultar apenas um valor
3. **.filter\_by()** – Filtra os dados de acordo com o parâmetro indicado. Ex: 

**Exemplo usando sqlalchemy**

rom flask import Flask, request, render\_template, \

    redirect, url\_for, flash

from sqlalchemy import create\_engine, Column, String, Integer

from sqlalchemy.orm import sessionmaker, declarative\_base

app = Flask(\_\_name\_\_)

db = create\_engine("sqlite:///meubanco.db")

Session = sessionmaker(bind=db)

session = Session()

Base = declarative\_base()

class Usuario(Base):

    \_\_tablename\_\_ = "usuarios"

    id = Column("id", Integer, primary\_key=True, autoincrement=True)

    email = Column("email", String)

    senha = Column("senha", String)

    def \_\_init\_\_(self,email, senha):

        self.email = email

        self.senha = senha

Base.metadata.create\_all(bind=db)

@app.route('/')

def index():

    usuarios = session.query(Usuario).all()

    return render\_template('pages/index.html', usuarios = usuarios)

@app.route('/create', methods=['POST', 'GET'])

def create():

    if request.method == 'POST':

        email = request.form['email']

        senha= request.form['password']

        usuario = Usuario(email, senha)

        session.add(usuario)

        session.commit()

        if not email:

            flash('Email é obrigatório')

        else:

            return redirect(url\_for('index'))

    return render\_template('pages/create.html')

@app.route('/<int:id>/edit', methods=['POST', 'GET'])

def edit(id):

    usuario = session.query(Usuario).filter\_by(id=id).first()

    if not usuario:

        return redirect(url\_for('error', message="Usuário não encontrado"))

    if request.method == 'POST':

        email = request.form['email']

        usuario.email = email

        session.commit()

        return redirect(url\_for('index'))

    return render\_template('pages/edit.html', usuario = usuario)

@app.route('/error')

def error():

    error = request.args.get('message')

    return render\_template('errors/error.html', message=error)