Aplicação Base **Quick Report**

Sumário

[Capítulo 1: Criar modelagem conceitual de dados 3](#_Toc169274267)

[Capítulo 2: Criar um Banco de Dados 3](#_Toc169274269)

[Capítulo 3: Clonar um projeto do github 3](#_Toc169274270)

[Capítulo 4: Ambiente Virtual 4](#_Toc169274271)

[Capítulo 5: config.py 4](#_Toc169274273)

[Capítulo 6: migrations 4](#_Toc169274274)

[Capítulo 7: reverse engineer 5](#_Toc169274275)

[Capítulo 8: Criar novas classes na model 5](#_Toc169274276)

[Classe Model Escola 5](#_Toc169274277)

[Capítulo 9: Template Engine (Jinja 2) 7](#_Toc169274278)

[Capítulo 10: Criar a rota de cadastrar 9](#_Toc169274279)

[Capítulo 10: Criar as rotas do CRUD 11](#_Toc169274280)

# Capítulo 1: Criar modelagem conceitual de dados

# Diagrama Descrição gerada automaticamente

# Capítulo 2: Criar um Banco de Dados

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

# Capítulo 3: Clonar um projeto do github

<https://github.com/romulosilvestre/projetopiloto.git>

# Capítulo 4: Ambiente Virtual

CTRL+SHIFT+P

# Interface gráfica do usuário, Aplicativo Descrição gerada automaticamente

# Capítulo 5: config.py

Texto

Descrição gerada automaticamente

# Capítulo 6: migrations

Apague a pasta migrations do projeto piloto

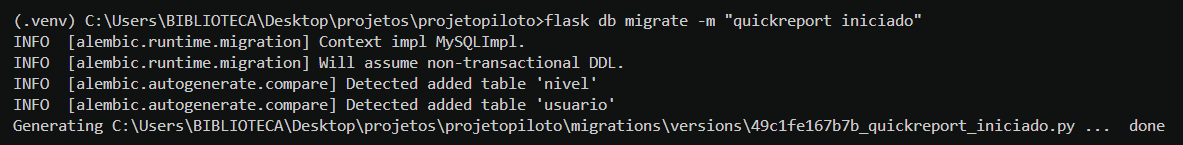
Segue os comandos:

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente



Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

# Capítulo 7: reverse engineer

Diagrama

Descrição gerada automaticamente

# Capítulo 8: Criar novas classes na model

## Classe Model Escola

Interface gráfica do usuário, Aplicativo, Tabela

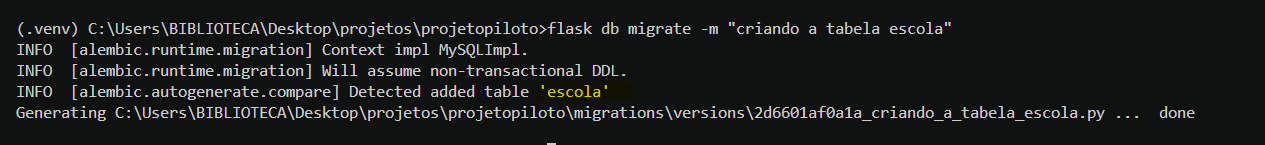
Descrição gerada automaticamente

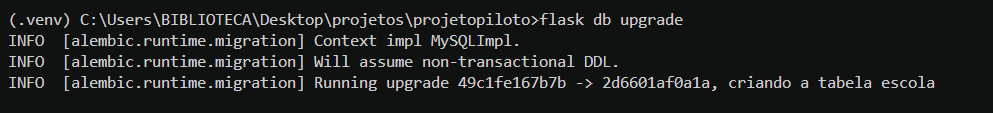
Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente





Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Diagrama

Descrição gerada automaticamente

**A escola tem um relacionamento com curso**

Uma imagem contendo Diagrama

Descrição gerada automaticamente

Diagrama, Tabela

Descrição gerada automaticamente

**Veja a programação da Model**

Texto

Descrição gerada automaticamente

**Um curso está vinculado a apenas uma escola.**

escola é o relacionamento com a tabela Escola, e back\_populates especifica a propriedade correspondente na classe Escola.

Vamos abrir a classe Escola.

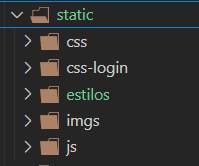
Texto

Descrição gerada automaticamente

# Capítulo 9: Template Engine (Jinja 2)

No desenvolvimento front-end você pode trabalhar com:

* HTML
* CSS
* Java Script
* Bootstrap
* Jinja 2 (template engine)



Em relação aos arquivos externos css, imagens, js, bootstrap etc, devem ficar na pasta static.

A importação da mesma é feita de seguinte forma:

Cria-se um arquivo base para herança de template:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Cadastrar Escola</title>

    <link rel="stylesheet" href="{{ url\_for('static', filename='estilos/style.css') }}">

    <link rel="stylesheet" href="{{ url\_for('static', filename='css/bootstrap.css') }}">

</head>

<body>

    <ul class="nav flex-column">

        <li class="nav-item">

            <a class="nav-link active" aria-current="page" href="#">Escola</a>

        </li>

        <li class="nav-item">

            <a class="nav-link" href="#">Curso</a>

        </li>

        <li class="nav-item">

            <a class="nav-link" href="#">Turma</a>

        </li>

        <li class="nav-item">

            <a class="nav-link disabled" aria-disabled="true">Módulo</a>

        </li>

        <li class="nav-item">

            <a class="nav-link disabled" aria-disabled="true">Aluno</a>

        </li>

        <li class="nav-item">

            <a class="nav-link disabled" aria-disabled="true">Diário</a>

        </li>

    </ul>

    {% block conteudo %}

    {% endblock conteudo %}

</body>

</html>

**Nesse arquivo você deve importar arquivos da pasta static da seguinte forma:**

    <link rel="stylesheet" href="{{ url\_for('static', filename='estilos/style.css') }}">

    <link rel="stylesheet" href="{{ url\_for('static', filename='css/bootstrap.css') }}">

</head>

**Na página principal você deve utilizar herança de template e aplicar recursos do Jinja 2 (template engine)**

{% extends "escola/base.html" %}

{% block conteudo %}

<div class="container col-4">

    <form method="post">

        {{ form.csrf\_token }}

        <div class="form-group">

            <label for="nome">Informe o nome da escola</label>

            {{ form.nome(class="form-control", id="nome") }}

        </div>

        <button type="submit" class="btn btn-primary">Cadastrar</button>

    </form>

</div>

{% if request.method == 'POST' %}

<script>

*// Limpa o campo de texto após o envio do formulário*

    document.getElementById("nome").value = "";

</script>

{% endif %}

{% endblock conteudo %}

# Capítulo 10: Criar a rota de cadastrar

from app import app

from flask import render\_template,redirect,url\_for,request *#renderização*

from app.forms import escola\_form

from app.models import escola\_model

from app import db

@app.route("/cadescola",methods=["POST","GET"])

def cadastrar\_escola():

    form = escola\_form.EscolaForm()

    if form.validate\_on\_submit():

       nome = form.nome.data *#capturando o conteúdo validado*

       escola = escola\_model.Escola(nome=nome)

       try:

*#adicionar na sessão*

          db.session.add(escola)

*#salvar a sessão*

          db.session.commit()

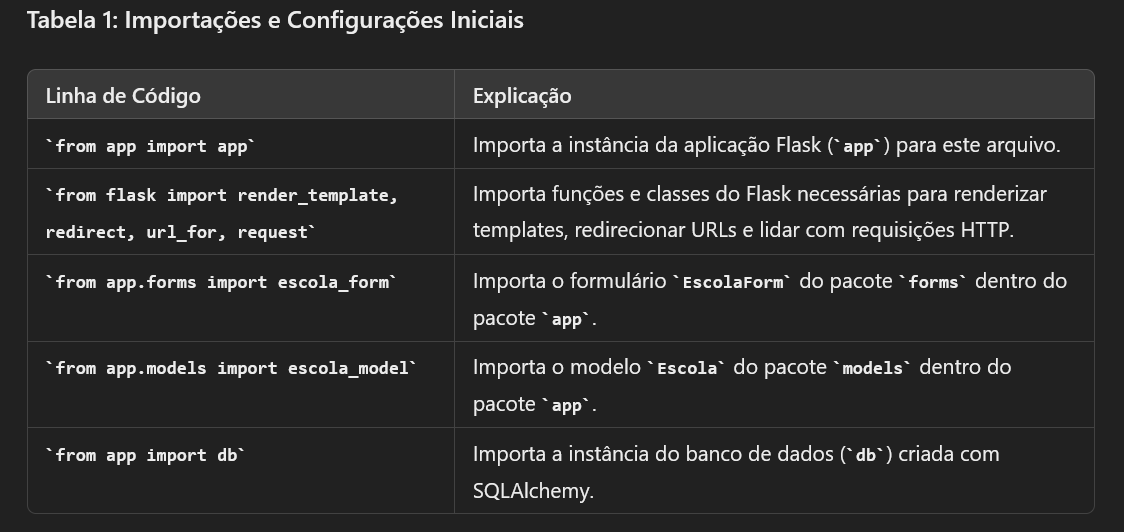
          if request.method == 'POST':

           return redirect(url\_for('listar\_niveis'))

       except:

         print("nivel não cadastrado")

    return render\_template("escola/index.html",form=form)



Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Site

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

# Capítulo 10: Criar as rotas do CRUD

from app import app

from flask import render\_template,redirect,url\_for,request *#renderização*

from app.forms import escola\_form

from app.models import escola\_model

from app import db

@app.route("/cadescola",methods=["POST","GET"])

def cadastrar\_escola():

    form = escola\_form.EscolaForm()

    if form.validate\_on\_submit():

       nome = form.nome.data *#capturando o conteúdo validado*

       telefone = form.telefone.data

       escola = escola\_model.Escola(nome=nome,telefone=telefone)

       try:

*#adicionar na sessão*

          db.session.add(escola)

*#salvar a sessão*

          db.session.commit()

          if request.method == 'POST':

           return redirect(url\_for('listar\_escolas'))

       except:

         print("nivel não cadastrado")

    return render\_template("escola/index.html",form=form,Editar=False)

@app.route("/listarescolas")

def listar\_escolas():

    escolas= escola\_model.Escola.query.all()  *# Consulta todos os registros na escola*

    return render\_template("escola/lista\_escola.html", escolas=escolas)

@app.route("/listaescola/<int:id>")

def listar\_escola(id):

    escola = escola\_model.Escola.query.filter\_by(id=id).first()  *# Consulta todos os registros na tabela Nivel*

    return render\_template("escola/lista\_escola\_id.html",escola=escola)

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

def editar\_escola(id):

   escola = escola\_model.Escola.query.filter\_by(id=id).first()

*# vamos agora criar nossa escola de formulário*

   form = escola\_form.EscolaForm(obj=escola)

*# verificar se todos os dados estão ok*

   if form.validate\_on\_submit():

      nome = form.nome.data

      telefone = form.telefone.data

      try:

        escola.nome = nome

        escola.telefone = telefone

        db.session.commit()

        return redirect(url\_for("listar\_escolas"))

      except:

        print("a escola não foi editado")

   return render\_template("escola/index.html",form=form,editar=True)

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

def remover\_escola(id):

     escola= escola\_model.Escola.query.filter\_by(id=id).first()

*# vamos indicar que o usuário clicou no botão remover*

*# importe request*

     if request.method == "POST":

        try:

             db.session.delete(escola)

             db.session.commit()

             return redirect(url\_for("listar\_escolas"))

        except:

             print("erro ao deletar escola")

     return render\_template("escola/remover\_escola.html",escola=escola)