

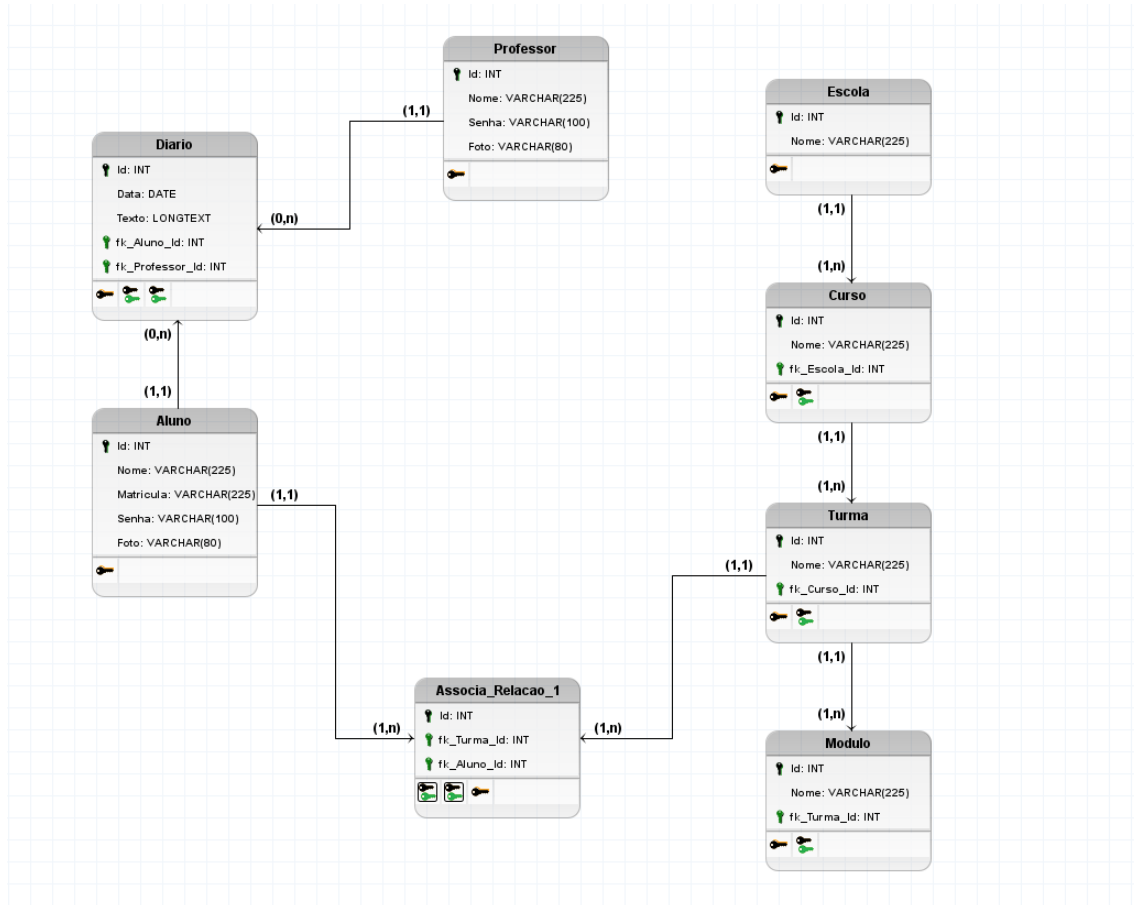
# Aplicação Base

## **Quick Report**

## Sumário

Capítulo 1: Criar modelagem conceitual de dados .....	3
Capítulo 2: Criar um Banco de Dados .....	3
Capítulo 3: Clonar um projeto do github .....	3
Capítulo 4: Ambiente Virtual .....	4
Capítulo 5: config.py .....	4
Capítulo 6: migrations .....	4
Capítulo 7: reverse engineer .....	5
Capítulo 8: Criar novas classes na model .....	5
Classe Model Escola .....	5
Capítulo 9: Template Engine (Jinja 2) .....	7
Capítulo 10: Criar a rota de cadastrar .....	9
Capítulo 10: Criar as rotas do CRUD .....	11

## Capítulo 1: Criar modelagem conceitual de dados



## Capítulo 2: Criar um Banco de Dados

```
Query 1
1 • CREATE DATABASE quickreport;
2 • USE quickreport;
3 • SHOW tables;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Contents: |

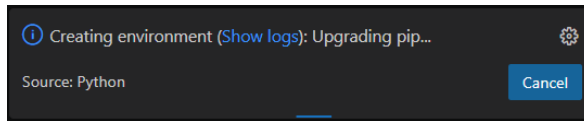
Tables\_in\_quickreport

## Capítulo 3: Clonar um projeto do github

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

## Capítulo 4: Ambiente Virtual

CTRL+SHIFT+P



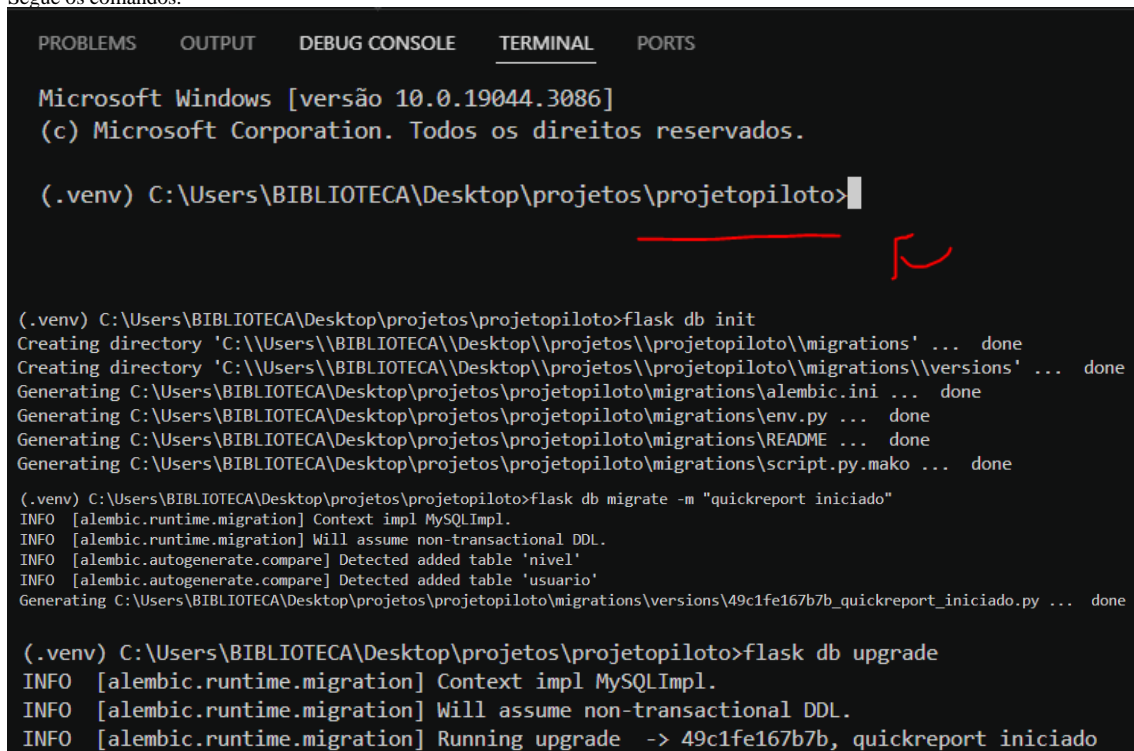
## Capítulo 5: config.py

```
12
13 DB = 'quickreport' ←
14 |
15 #connection string
16 SQLALCHEMY_DATABASE_URI=f'mysql://{USERNAME}:{PASSWORD}@{SERVER}/{DB}'
17 #modificação
18 SQLALCHEMY_TRACK_MODIFICATIONS = True
19 #chave secreta - hash (chave criptografada)
20 #entrar em qualquer site que gere hash - colocar o hash
21 #publicar.
22 SECRET_KEY ="8a4dbb9594173ae2747f9704468a89bd"
23
```

## Capítulo 6: migrations

Apague a pasta migrations do projeto piloto

Segue os comandos:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

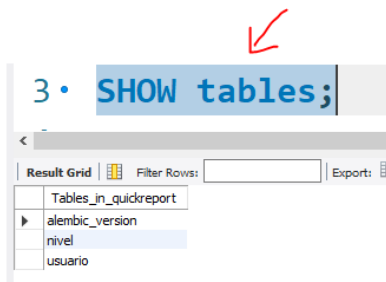
Microsoft Windows [versão 10.0.19044.3086]
(c) Microsoft Corporation. Todos os direitos reservados.

(.venv) C:\Users\BIBLIOTECA\Desktop\projetos\projetopiloto>

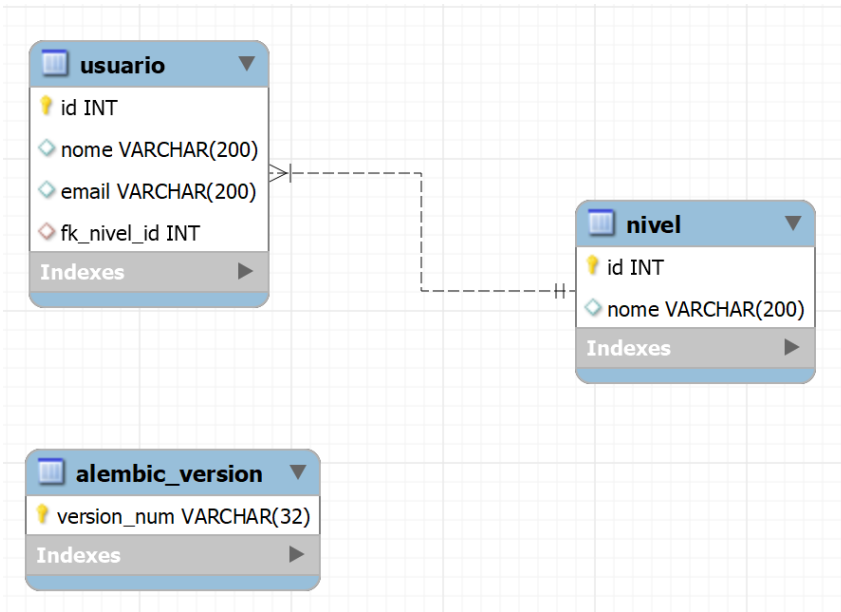
(.venv) C:\Users\BIBLIOTECA\Desktop\projetos\projetopiloto>flask db init
Creating directory 'C:\\Users\\BIBLIOTECA\\Desktop\\projetos\\projetopiloto\\migrations' ... done
Creating directory 'C:\\Users\\BIBLIOTECA\\Desktop\\projetos\\projetopiloto\\migrations\\versions' ... done
Generating C:\\Users\\BIBLIOTECA\\Desktop\\projetos\\projetopiloto\\migrations\\alembic.ini ... done
Generating C:\\Users\\BIBLIOTECA\\Desktop\\projetos\\projetopiloto\\migrations\\env.py ... done
Generating C:\\Users\\BIBLIOTECA\\Desktop\\projetos\\projetopiloto\\migrations\\README ... done
Generating C:\\Users\\BIBLIOTECA\\Desktop\\projetos\\projetopiloto\\migrations\\script.py.mako ... done

(.venv) C:\Users\BIBLIOTECA\Desktop\projetos\projetopiloto>flask db migrate -m "quickreport iniciado"
INFO [alembic.runtime.migration] Context impl MySQLImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
INFO [alembic.autogenerate.compare] Detected added table 'nivel'
INFO [alembic.autogenerate.compare] Detected added table 'usuario'
Generating C:\\Users\\BIBLIOTECA\\Desktop\\projetos\\projetopiloto\\migrations\\versions\\49c1fe167b7b_quickreport_iniciado.py ... done

(.venv) C:\Users\BIBLIOTECA\Desktop\projetos\projetopiloto>flask db upgrade
INFO [alembic.runtime.migration] Context impl MySQLImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
INFO [alembic.runtime.migration] Running upgrade -> 49c1fe167b7b, quickreport iniciado
```



## Capítulo 7: reverse engineer



## Capítulo 8: Criar novas classes na model

### Classe Model Escola



```

1 from app import db
2
3 class Escola(db.Model):
4     __tablename__ = "escola"
5     id = db.Column(db.Integer, primary_key=True, autoincrement=True)
6     nome = db.Column(db.String(255), nullable=False)
  
```

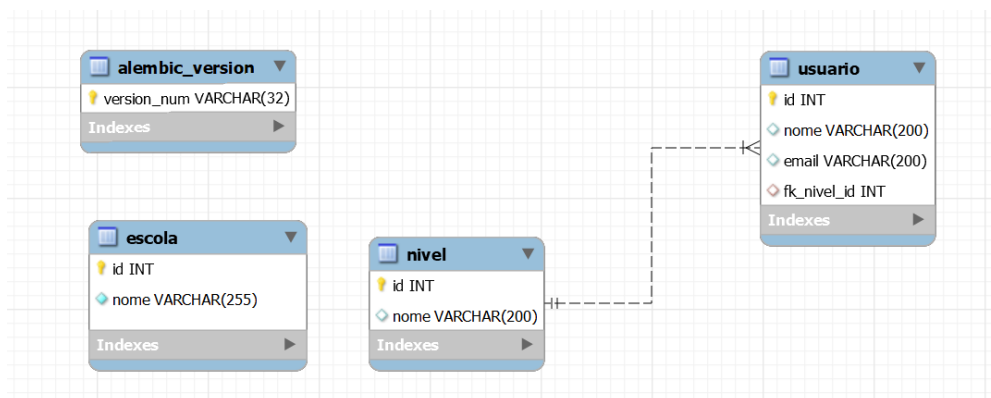
```

22 from app.models import nivel_model
23 from app.models import usuario_model
24 from app.models import escola_model
  
```

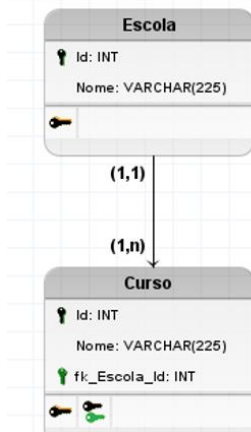
```
(.venv) C:\Users\BIBLIOTECA\Desktop\projetos\projetopiloto>flask db migrate -m "criando a tabela escola"
INFO [alembic.runtime.migration] Context impl MySQLImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
INFO [alembic.autogenerate.compare] Detected added table 'escola'
Generating C:\Users\BIBLIOTECA\Desktop\projetos\projetopiloto\migrations\versions\2d6601af0a1a_criando_a_tabela_escola.py ... done
```

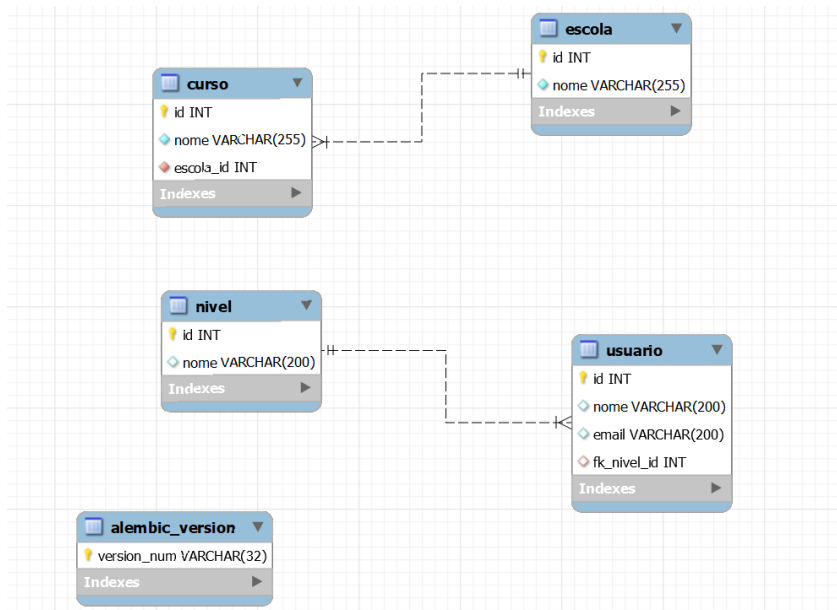
```
(.venv) C:\Users\BIBLIOTECA\Desktop\projetos\projetopiloto>flask db upgrade
INFO [alembic.runtime.migration] Context impl MySQLImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
INFO [alembic.runtime.migration] Running upgrade 49c1fe167b7b -> 2d6601af0a1a, criando a tabela escola
```

The screenshot shows a database management tool interface. On the left, a 'SCHEMAS' pane displays a tree view of the database structure, including 'projectscore', 'projetoabasefinal', and 'quickreport'. Under 'quickreport', there are tables 'alembic\_version', 'escola', 'nivel', and 'usuario'. The 'escola' table is highlighted. On the right, a SQL query editor shows three commands: '1. CREATE DATABASE quickreport;', '2. USE quickreport;', and '3. SHOW tables;'. Below the editor, a 'Result Grid' shows the output of the 'SHOW tables;' command, listing the tables in the 'quickreport' database: 'alembic\_version', 'escola', 'nivel', and 'usuario'.



A escola tem um relacionamento com curso





### Veja a programação da Model

```

4 class Curso(db.Model):
5     __tablename__ = "curso"
6     id = db.Column(db.Integer, primary_key=True, autoincrement=True)
7     nome = db.Column(db.String(255), nullable=False)
8     escola_id = db.Column(db.Integer, db.ForeignKey('escola.id'), nullab.
9     escola = relationship("Escola", back_populates="cursos")
  
```

### 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.

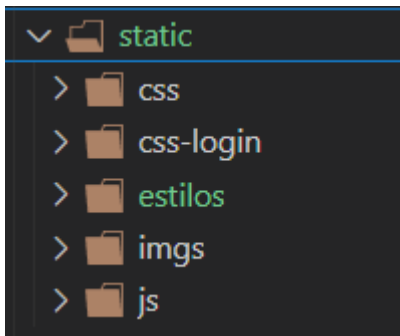
```

2 from sqlalchemy.orm import relationship
3
4 class Escola(db.Model):
5     __tablename__ = "escola"
6     id = db.Column(db.Integer, primary_key=True, autoincrement=True)
7     nome = db.Column(db.String(255), nullable=False)
8     cursos = relationship("Curso", back_populates="escola")
  
```

### 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("nível não cadastrado")

    return render_template("escola/index.html", form=form)

```

Tabela 1: Importações e Configurações Iniciais

Linha de Código	Explicação
<code>`from app import app`</code>	Importa a instância da aplicação Flask ( <code>`app`</code> ) para este arquivo.
<code>`from flask import render_template, redirect, url_for, request`</code>	Importa funções e classes do Flask necessárias para renderizar templates, redirecionar URLs e lidar com requisições HTTP.
<code>`from app.forms import escola_form`</code>	Importa o formulário <code>`EscolaForm`</code> do pacote <code>`forms`</code> dentro do pacote <code>`app`</code> .
<code>`from app.models import escola_model`</code>	Importa o modelo <code>`Escola`</code> do pacote <code>`models`</code> dentro do pacote <code>`app`</code> .
<code>`from app import db`</code>	Importa a instância do banco de dados ( <code>`db`</code> ) criada com SQLAlchemy.

## Parte 1: Definição da Rota e Criação do Formulário

Linha de Código	Explicação
<code>`@app.route("/cadescola", methods=["POST", "GET"])`</code>	Define uma rota para a função <code>`cadastrar_escola`</code> que responde a requisições GET e POST para <code>`/cadescola`</code> .
<code>`def cadastrar_escola():`</code>	Define a função <code>`cadastrar_escola`</code> , que será executada ao acessar <code>`/cadescola`</code> .
<code>`form = escola_form.EscolaForm()`</code>	Instancia o formulário <code>`EscolaForm`</code> para ser utilizado na renderização da página.

## Parte 2: Validação e Captura de Dados do Formulário

Linha de Código	Explicação
<code>`if form.validate_on_submit():`</code>	Verifica se o formulário foi submetido e se os dados são válidos.
<code>`nome = form.nome.data`</code>	Obtém os dados validados do campo <code>`nome`</code> do formulário.

### Parte 3: Criação e Persistência do Objeto no Banco de Dados

Linha de Código	Explicação
<code>`escola = escola_model.Escola(nome=nome)`</code>	Cria uma instância do modelo <code>`Escola`</code> , passando o nome como argumento.
<code>`db.session.add(escola)`</code>	Adiciona a instância <code>`escola`</code> à sessão do banco de dados ( <code>`db.session`</code> ).
<code>`db.session.commit()`</code>	Realiza o commit das mudanças na sessão, persistindo os dados no banco de dados.

### Parte 4: Redirecionamento e Manipulação de Exceções

Linha de Código	Explicação
<code>`if request.method == 'POST':`</code>	Verifica se a requisição é do tipo POST (embora redundante neste contexto).
<code>`return redirect(url_for('listar_niveis'))`</code>	Redireciona o usuário para a função <code>`listar_niveis`</code> após cadastrar a escola.
<code>`except:`</code>	Captura qualquer exceção que ocorra no bloco anterior.
<code>`print("nível não cadastrado")`</code>	Imprime uma mensagem de erro caso ocorra uma exceção.

### Parte 5: Renderização do Template

Linha de Código	Explicação
<code>`return render_template("escola/ index.html", form=form)`</code>	Renderiza o template <code>`index.html`</code> do diretório <code>`escola`</code> , passando o formulário <code>`form`</code> como contexto.

## 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("nível 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 Nível

    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)
```