**Quick Report** – Dúvida- quero estilizar de forma personalizada no framework, mas não atrelado 100% aos frameworks?

Parabéns, se você está lendo esse tutorial com certeza se encontra na parte mais emocionante do projeto. Porém, a mais desafiadora, que é ligar múltiplas tabelas em um único front-end. Isso, pode parecer confuso no início, mas com a prática logo estará apto a criar as suas próprias telas com múltiplos dados.

Vamos atender a primeira dúvida, e nesse suporte vou explicar uma forma de personalização dos seus template, usando a parte “static” do projeto. E também uma técnica de sobrepor os frameworks css.

Então vamos começar.

O projeto base do professor para apoio se encontra no seguinte link:

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

Ele possui apenas o CRUD de escola.

Vamos revisitar agora o modelo de banco de dados construído por você, vamos?

Estrutura clonada

Texto

Descrição gerada automaticamente com confiança média

Vamos focar no curso

Observe que conforme orientação do modelo conceitual de dados, o curso deve ter um id estrangeiro, chamamos de foreign key.

Texto

Descrição gerada automaticamente

Esse é o desafio. Em um mesmo template de curso teremos que ter informações do curso para cadastrar, mas trazer algumas informações da tabela escola.

Como só tem duas informações da escola (id, nome) são os dois necessários para listarmos dentro do select html.

Vamos executar o projeto para verificar o cadastro.

Iniciamos criando um novo ambiente virtual. Não é necessário criar caso já tenha. Esse processo é só caso tenha que clonar a primeira vez, o quer clonar a versão do professoro pela primeira vez, ou perdeu o arquivo em alguma máquina.

Vamos criar a venv.

Tela de celular com letras

Descrição gerada automaticamente

Ative a .venv



Execute o projeto

Texto

Descrição gerada automaticamente

Vou cadastrar a escola.

Interface gráfica do usuário, Texto, Aplicativo, Word

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo, Linha do tempo

Descrição gerada automaticamente

**Parte 1 – CSS personalizado**

Uma dúvida que foi levantada na sala de aula é sobre o visual da aplicação. Se a estrutura ficar dividida conforme se encontra na estrutura do projeto fica tudo tranquilo e fácil a utilização de recursos estáticos.

Vou dar um exemplo simples de um css para formatar um table html.

Texto

Descrição gerada automaticamente

Dentro da static, pasta criada especificamente para trabalhar com o “site”, o front-end puro em suas estilizações bootstrap ou css autoral.

Interface gráfica do usuário

Descrição gerada automaticamente

Crie uma pasta geral, para um css autoral, apesar que no bootstrap posso formatar a tabela também.

Criei um arquivo:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Nele vou criar uma classe css para formatar linhas e bordas de uma tabela.

Criei um css que estiliza a tabela para ficar com a aparência de um carro.

Tela de computador com texto preto sobre fundo branco

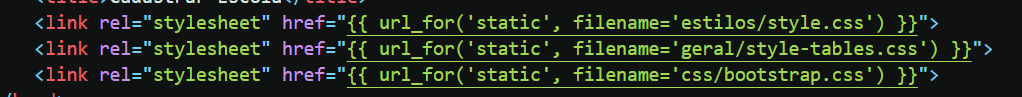
Descrição gerada automaticamente

Agora precisamos linkar esse arquivo na tela de listagem de escola.

Lembre que nosso projeto esta vinculado a uma base.

{% extends "escola/base.html"  %}

De forma bem-organizada dividi em três arquivos css:



Agora temos mais um link que é o geral/style-tables.css

Como ele tem herança de templates automaticamente ele vai utilizar o css da base.

Agora vamos acrescentar a classe para verificar o que aconteceu!

Texto

Descrição gerada automaticamente

Olá, o resultado ficou da seguinte forma:

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

Agora você quer mudar a cor da borda.

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

Tela de celular com texto preto sobre fundo branco

Descrição gerada automaticamente com confiança média

Vou deixar a borda maior:

Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente

Se você quer sobrepor outros css inclusive o css do bootstrap pode usar !important

Tabela

Descrição gerada automaticamente com confiança média

Para ele ficar com as bordas conforme o exemplo acima, tive que inserir o seguinte código no css:

Tela preta com letras brancas

Descrição gerada automaticamente

Assim, você fica a vontade para decidir se usa o css personalizado ou se o css do bootstrap. Apesar do bootstrap ter muitos recursos, mas sei que você gosta de personalizar e deixar do seu jeito.

Espero ter ajudado na dúvida específica de trabalhar com o visual e uma introdução a utilização de herança de templates.

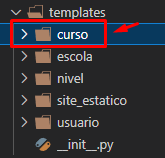
Assim, na camada estática você esta livre para estilizar usando! important. Infinitas possibilidades para estilização de front-end surgem independente do framework utilizado.

Espero ter ajudado nesse suporte escrito!

**A disposição!**

**Parte 2 – Criando a tela curso trabalhando com múltiplas tabelas do banco de dados**

Crie uma pasta chamada curso



**No template curso na página index.html vamos criar uma página que exibe 10 vezes uma <h1>, veja também explicações sobre os delimitadores:**

Texto

Descrição gerada automaticamente

**Para essa página funcionar vamos criar uma rota especifica para curso:**

Texto

Descrição gerada automaticamente

**Assim, temos uma rota que nos auxilia no cadastro de curso. Por enquanto o objetivo é só abrir a tela index.html e ver se os delimitadores do jinja 2 está funcionando corretamente.**

**Não esqueça de trazer para \_\_init\_\_.py o que você acabou de criar:**

Texto

Descrição gerada automaticamente

**Vamos rodar a aplicação:**

Uma imagem contendo Texto

Descrição gerada automaticamente

**Como esse exemplo foi apenas didático vou comentar o código ok.**

Texto

Descrição gerada automaticamente

**Pronto o loop sumiu**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Agora que você sabe comentar e programar no template usando jinja 2.**

**Vamos importar a base da escola, reaproveitando o arquivo base principal , vamos:**

Texto

Descrição gerada automaticamente

**Então, aqui estou herdando o html da outra pasta.**

**Vamos executar a aplicação?**

**Ele deu um erro**

Interface gráfica do usuário, Texto, Site

Descrição gerada automaticamente

**Isso acontece porque temos que voltar o diretório para usar outra pasta como herança:**



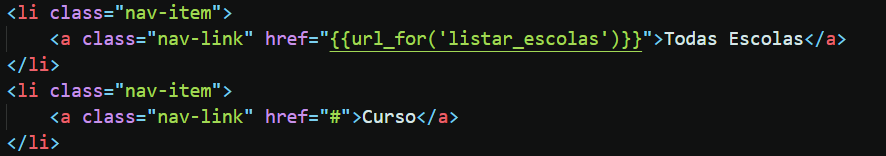
**E é necessário usar “ “.**

Lembre-se, o template engine converte os códigos jinja 2 para o que o navegador realmente entende: HTML, CSS e JS.

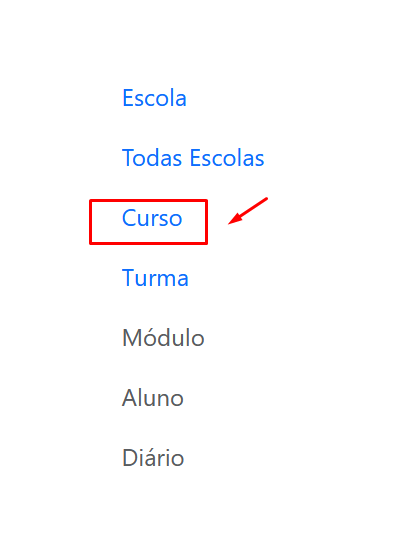
Veja que não tem curso. Olha só que legal a velocidade de atualização usando Jinja.

Você vai atualizar em apenas um lugar e ele vai atualizar em todos. Vamos?

Abaixo de todas as escolas vou criar um link:



Agora atualize!!!



Todas as páginas receberam a atualização.

Vamos para todas as escolas por exemplo.

Tabela

Descrição gerada automaticamente com confiança baixa

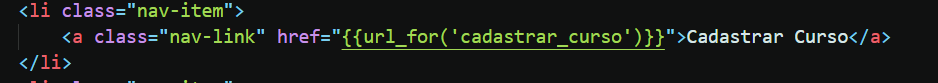
Veja que já foi atualizado automaticamente.

Agora que você aprendeu a usar o Jinja 2 com uma introdução de herança de template. Vamos ajustar a rota.

Vamos?

Entre no template escola/base.html

Atualize o texto e a url do hiperlink :



**Atualize e veja o resultado!**

**Estou em listar escolas e vou clicar curso.**

Tabela

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**Assim, foi direcionado para a tela de cadastro de curso.**

Agora vamos colocar um select dinâmico que busque os dados de escolas de forma personalizada para mostrar os dados em uma select html. Vamos?

**Vamos criar um form curso:**

Texto

Descrição gerada automaticamente

**Importe na rota**

Texto

Descrição gerada automaticamente

Na linha: **from app import app**

**Importação do app**

**=============================================================**

Na linha: **from flask import render\_template**

**Importação flask,** importo o render\_template para renderizar a página web

==============================================================

Na linha: **from app.models.escola\_model import Escola**

**Faço uma importação direta como você gostou na aula**

**=============================================================**

Na linha: **from app.forms.curso\_form import CursoForm**

**Faço a importação da camada form**

**=============================================================**

**Na linha: @app.route("/cadcurso",methods=["POST","GET"])**

**Programo a rota**

**=============================================================**

**def cadastrar\_curso():**

**form\_curso = CursoForm()**

**Programo a instância do objeto form\_curso da classe CursoForm( )**

**# Carregar os níveis do banco de dados**

**escolas = Escola.query.all()**

**==================================================================**

**# Formatar os níveis para o formato necessário para o campo de seleção**

**escolas\_choices = [(escola.id, escola.nome) for escola in escolas]**

**Vamos a explicação detalhada linha a linha:**

1. Criamos uma variável escolas\_choices.
2. Essa variável recebe os dados da tabela escola de forma personalizada
3. Para personalizar tivemos que criar um for onde a variável escola percorre escolas (que a lista que veio do banco)
4. Criamos duas outras variáveis escola.id e escola.nome que representa os dois campos que virão do banco. Assim, conseguimos personalizar.
5. Armazenamos essa personalização dentro da variável escolas\_choices.

**==================================================================**

**form\_curso.escola\_id.choices = escolas\_choices**

**Vamos a explicação detalhada linha a linha:**

1. A variável escolas\_choices possui a personalização lembra (id, nome) para carregar o select html lá no template curso/index.html
2. Então, para que o form\_curso entenda precisamos , pegar a variável que criamos para a chave estrangeira escola\_id.
3. form.curso.escola\_id , assim acessamos o form e a variável responsável pelo relacionamento.
4. Form\_curso.escola\_id.choices : choices é uma propriedade do SelectField conforme documentação do mesmo.
5. Armazenamos essa personalização dentro da variável escolas\_choices.

Interface gráfica do usuário, Texto, Aplicativo, Word

Descrição gerada automaticamente

**if form\_curso.validate\_on\_submit():**

**pass**

**Vamos a explicação detalhada linha a linha:**

1. Aqui eu faço uma verificação condicional se o formulário foi enviado, no caso o formulário form\_curso.
2. Nesse caso coloquei “pass”, porque meu objetivo agora é apenas mostrar como carregar o SelectField do WtForms e também no Select HTML que esta no template curso/index.html.

============================================================

**Por último, e não menos importante o retorno do formulário como parâmetro da função render\_template ( envio do template, envio do formulário)**

**return render\_template("curso/index.html",form\_curso=form\_curso)**

**Para rodar aplicação inicialmente tire a herança de template para testar:**

*<!--Como a escola é nossa primeira entidade, vou considera-la como base para as outras-->*

*<!--Considerações sobre delimitadores do Jinja 2-->*

{#

    Pode utilizar esses delimitadores para comentar nos templates

    Outros delimitadores:

    {%   %}  : usado para utilizar métodos python nos templates

    {{   }}  : serve para concatenar variáveis aos templates

#}

{# vamos dar um exemplo simples: #}

{#

{% for i in range(10): %}

    <h1>estou no loop {{ i }}</h1>

{% endfor%}

#}

<h1>oi</h1>

<div class="form-group">

    <label>Nível</label>

    <select id="escola\_id" name="escola\_id" class="form-control">

        {% for id, nome in form\_curso.escola\_id.choices %}

            <option value="{{ id }}">{{ nome }}</option>

        {% endfor %}

    </select>

</div>

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Para ficar com o visual elegante coloquei o boostrap e no container na classe coloquei col-4

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Segue a codificação do template, ficou grande pois tem outras explicações didáticas que fiz pra você.

<!--Como a escola é nossa primeira entidade, vou considera-la como base para as outras-->

<!--Considerações sobre delimitadores do Jinja 2-->

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

{#

Pode utilizar esses delimitadores para comentar nos templates

Outros delimitadores:

{% %} : usado para utilizar métodos python nos templates

{{ }} : serve para concatenar variáveis aos templates

#}

{# vamos dar um exemplo simples: #}

{#

{% for i in range(10): %}

<h1>estou no loop {{ i }}</h1>

{% endfor%}

#}

{% block conteudo %}

<div class="container col-4">

<div class="form-group">

<label>Nível</label>

<select id="escola\_id" name="escola\_id" class="form-control">

{% for id, nome in form\_curso.escola\_id.choices %}

<option value="{{ id }}">{{ nome }}</option>

{% endfor %}

</select>

</div>

{% endblock conteudo %}

**Está marcado de amarelo os códigos que alterei para melhorar o visual herando templates.**