Professor Rômulo Cesar Silvestre

|  |
| --- |
| Desenvolvimento de Sistemas Web |
| **Área de concentração** |
| TI-SOFTWARE |
| **Competência(a) a ser(em) desenvolvidas** |
| Criar um aplicativo web para avaliar a unidade de competência |
| **Passos** |
| **(1) Contextualização e Mobilização** |
| Professores ministram disciplinas. As disciplinas geralmente fazem parte de um curso. Porém, temos algumas escolas que chamam disciplina de unidade de competência, as famosas UCs.  Temos diversos tipos de cursos em uma escola: qualificação, aperfeiçoamento, técnico, superior entre outros.  Nesse contexto vamos focar no curso técnico. O curso técnico em questão é o curso técnico em informática.  Ele possui uma variedade de unidade de competência. Devido a isso, o software precisa de um SGBD para armazenar os dados. |
| **(2) Atividade de Aprendizagem** |
| Desenvolver um projeto web com o *“microframework flask”* para cadastrar unidade de competência em um banco de dados. |
| **(3) Organização da Atividade de Aprendizagem** |
| **Etapa 1**: preparar ambiente  **Etapa 2:** relembrar fundamentos de ambiente virtual  **Etapa 3:** explicar a utilização do protocolo HTTP  **Etapa 4**: criar a aplicação (estrutura de pastas e design patterns)  **Etapa 5**: preparando o flask  **Etapa 6**: instalar pacotes  **Etapa 7**: conectar com banco de dados  **Etapa 8:** criar um models com migrations  **Etapa 9:** criar rotas  **Etapa 10**: criar rotas com parâmetros  **Etapa 11**: criar rotas com métodos http  **Etapa 12**: criar views  **Etapa 13:** criar templates com html  **Etapa 14**: criar templates com bootstrap  **Etapa 15**: criar template com jinja 2  **Etapa 16**: persistir (inserir) dados no banco de dados |
| **(4) Coordenação e Acompanhamento** |
| A cada etapa deverá ser feita uma entrega e um feedback  **Etapa 1**: rodou um “alô mundo rota”  **Etapa 2:** resolveu problemas de ambientes virtuais?  **Etapa 3:** quais os principais métodos do protocolo HTTP? O que é REST? O que é RESTFull?  **Etapa 4**: o que significa programação em camadas MTV?  **Etapa 5**: rodou a aplicação “alô mundo rota”?  **Etapa 6**: verificar as versões instalados dos pacotes  **Etapa 7**: conseguiu conectar? Roda a aplicação? Rodou?  **Etapa 8:** criou o modelo ? deu um init? deu um migrate? deu um upgrade?  **Etapa 9:** mostre a rota básica funcionando?  **Etapa 10**: mostre a rota com parâmetro funcionando?  **Etapa 11**: teste com postman métodos http diferentes modificando também o tipo de HTTP aceito na rota  **Etapa 12**: você entendeu o que são views?  **Etapa 13:** testar a página .html  **Etapa 14**: testar a página .html com bootstrap  **Etapa 15**: testar a página .html com jinja2 (templa engine)  **Etapa 16**: abra o banco de dados e faça um select \* from na tabela em questão |
| **(5) Análise e Avaliação da Atividade de Aprendizagem;** |
| Verificar cada etapa de 1 a 16 com arguição oral  Realizar um questionário online para verificação de conhecimentos  Verificar o funcionamento do software com funcionalidades básicas de inserção no banco de dados. |
| **(6) Outras Referências** |
| <https://github.com/romulosilvestre/gabaritoflask> |
| **(7) Síntese e Generalização** |
| Utilizando o projeto base e modificado como exemplo, escolha um caso de uso do seu projeto integrador e faça o aplicativo web inserindo dados.  **Recursos necessários:**  Projeto base de exemplo  Diagrama de Caso de Uso  Documento Textual de Caso de Uso  Modelagem Conceitual  Modelo Lógico  Diagrama de Classe  . |

# **Etapa 1**: preparar ambiente

O que é um ambiente virtual?

No python o ambiente virtual é uma instalação que não conflita com as versões que estão instaladas no computador.

Interface gráfica do usuário

Descrição gerada automaticamente

**Habilidade:**

|  |  |
| --- | --- |
| A | Criar uma pasta na **área de trabalho:** |
| B | Abrir a pasta no Microsoft Visual Studio Code: |
| C | Instalar o ambiente virtual: |
| D | Verificar a pasta criada: |
| E | Abrir a pasta **venv:** |
| F | Mudar o tipo de terminal para **Command Prompt**  **Verifique é necessário mudar de terminal**    Leia no canto inferior direito que tipo de interpretador está selecionado. Caso não seja a venv selecione na pasta quando necessário. |
| G | Ativar o ambiente virtual    Volte para pasta original principal. |

# **Etapa 2**: relembrar fundamentos de ambientes virtuais

Quais problemas têm enfrentado com ambientes virtuais?

O ***objetivo desses ambientes*** é que você possa trabalhar com versões diferentes na mesma máquina.

Qual versão você está utilizando?

Habilidade:

|  |  |
| --- | --- |
| A |  |

# **Etapa 3**: explicar a utilização do protocolo HTTP

O protocolo HTTP tem como base a comunicação entre máquinas. Uma máquina cliente faz requisições para uma outra máquina servidora. Quando a máquina cliente faz um pedido para a máquina servidora chamados de **request.** Agora quando o servidor da a resposta ao cliente chamamos de **response.**

A tabela abaixo mostra os métodos HTTP:

|  |  |
| --- | --- |
| **Método** | **Significado semântico** |
| GET | “Pega” do servidor |
| POST | Incluir “alguma coisa” no servidor |
| PUT | Atualizar “alguma coisa” no servidor |
| DELETE | Apagar “alguma coisa” no servidor |

Nessa comunicação entre cliente e servidor ocorre vários problemas de comunicação, para cada problema temos um código específico:

|  |  |
| --- | --- |
| **Método** | **Significado semântico** |
| 200 | **OK**. Deu tudo certo. |
| 302 | **Found.** O recurso existe no servidor (típico de GET) |
| 401 | **Unauthorized**. Não realizou autenticação para acessar o recurso |
| 404 | **Not Found**. O recurso que você solicitou não existe no servidor. |
| 500 | **Internal Server Error.** O servidor encontrou um erro durante o processamento da requisição. |

Conhecimentos:

|  |  |
| --- | --- |
| A | Os principais métodos HTTP são: GET, POST, PUT, DELETE |
| B | Existem diversos números que possuem significados semânticos no mundo do protocolo HTTP. Cada número representa uma mensagem do que ocorreu na comuniçaõ entre cliente e servidor. |
| C | O padrão REST traz boas práticas de utilização do HTTP |
| D | Quando uma API aplica o padrão REST, chamamos essa API de API RESTFull |

# **Etapa 4**: criar a aplicação (estrutura de pastas e design patterns)

Habilidades:

|  |  |
| --- | --- |
| A | Criar a pasta **app** dentro da pasta **projectscore** |
| B | Dentro da pasta app criar mais três pastas: |
| C | Criar um arquivo de inicialização dentro da pasta app e dentro de cada pasta criada na pasta app. |

**Conhecimentos**

|  |  |
| --- | --- |
| A | Design Patterns são padrões de projetos |
| B | Padrões de projetos em programação é criar pastas para dividir a programação |
| C | Cada pasta fica responsável por uma camada |
| D | Nas linguagens usa-se pastas, pacotes, módulos, a nomenclatura vária, mas o objetivo é o mesmo “separar para organizar” |
| C | Nessa organização as aplicações web podem e devem ser escalonáveis, facilitando a manutenção no futuro |
| E | Ela no início aparenta ter mais trabalho, depois você consegue usufruir da organização. |
| F | No mundo web, o padrão mais conhecido é MVC |
| G | No python, o mais conhecido é MTV usado no framework Django |
| H | MVC ( model -view – controller) |
| I | MTV (model – template – view) |
| J | Model – é o modelo dos dados |
| K | Template – é o html (front-end) da aplicação |
| L | View – trata as requisições da aplicação web (aplicando HTTP e rotas) |
| M | O design patterns tanto MVC como MTV são muito utilizados no mercado de trabalho devido sua aplicabilidade nos frameworks e softwares de grande escala (corporativo) |
| N | Analisar imagem do funcionamento de design pattern:  Como funciona a arquitetura MTV (Django) | by Diandra Andrade e Silva |  Medium |

# **Etapa 5**: preparando o flask

O framework que iremos utilizar é o Flask. Ele é um microframework, devido sua simplicidade sua popularidade cresce a cada dia. É preciso entender que dentro do flask possui outras bibliotecas que o apoiam, cito: Werkzeug e a Jinja2.

A Werkzeug tem o objetivo de oferecer recursos para o programador utilizar o HTTP.

* Analisar cabeçalho HTTP;
* Facilitar o trabalho com requisições e respostas;
* Depurar interativamente;
* Ter suporte ao Unicode;
* Ter suporte a sessões e cookies;
* Ter um sistema de roteamento integrado.

Por outro lado, o Jinja2 é uma biblioteca que tem como objetivo trabalhar com template.

Com ela você pode:

* Ter herança de template
* Prevenção de XSS
* Sintaxe configurável
* Execução de código Python em páginas HTML

Conhecimentos

|  |  |
| --- | --- |
| A | O que é um framework? |
| B | O que é um template engine? |
| C | O que é jinja2? |
| D | O que é Werkzeug? |
| E | Quais as vantagens de utilizar o Jinja2? |

Habilidades

|  |  |
| --- | --- |
| A | Instalar o Flask (com a venv ativada e posicionado no diretório principal): |
| C | Gerar uma lista dos pacotes instalados juntamente com o flask: |

# Etapa 6: instalar pacotes

Habilidades

|  |  |
| --- | --- |
| A | Instalar o “mysql-connector-python 8.3.0” : |
| B | Dê um pip freeze > requirements.txt |
| C | Instalar: |
| D | Verificar as instalações: |
| E |  |

# **Etapa 7 e 8**: criar models , conectar com banco e realizar migrations

Habilidades

|  |  |
| --- | --- |
| A | Criar um arquivo ***config.py*** no projeto raiz:  Tela de computador com texto preto sobre fundo branco  Descrição gerada automaticamente |
| B | Estudar, corrigir e analisar codificação:  **Lista de Codificação 01:**  DEBUG = True  USERNAME = 'root'  PASSWORD = 'root'  SERVER = 'localhost'  DB = 'projectscore'  SQLALCHEMY\_DATABASE\_URI=f'mysql://{USERNAME}:{PASSWORD}@{SERVER}/{DB}'  SQLALCHEMY\_TRACK\_MODIFICATIONS = True |
| C | Verificar pacotes :  Flask-SQLAlchemy: trabalhar com ORM e Flask  mysql-connector-python : conector com banco  SQLAlchemy: ORM escolhido |
| D | O SQLAlchemy é uma biblioteca de mapeamento objeto-relacional (ORM) que fornece um conjunto completo de comandos para manipular bancos de dados utilizando o Python. |
| E | Frameworks ORM tentam reduzir o tamanho do trabalho que temos ao nos conectar principalmente com bases de dados relacionais. |
| F | ORM é um acrônimo para object-relational mapping – mapeamento objeto-relacional.   * Cada classe acaba sendo interpretada como uma tabela; * Cada linha de uma tabela, bem como seus relacionamentos, é tratada como instância do objeto relacionado à tabela em questão. |
| G | DEBUG = True  DEBUG = False  Faça um teste utilizando os dois e escolha qual o melhor para você |
| H | O que são models? |
| I | Como vimos anteriormente, iremos separar nosso projeto Flask em três camadas principais (Model-Template-View). |
| J | A camada Model tem um papel fundamental nessa arquitetura. |
| K | É ela quem se comunica, através do SQLAlchemy, com o banco de dados para criar os modelos da aplicação diretamente no banco de dados sem a necessidade de usar código SQL. |
| L | Precisamos do Migrations    Assim como o GitHub faz o versionamento do código o Flask-Migrate faz o versionamento do banco. |
| M | Interface gráfica do usuário, Texto, Aplicativo  Descrição gerada automaticamente |
| N |  |
| O |  |
| P | Em models crie um arquivo uc\_model.py: |
| Q | **Segue listagem da codificação:**  from app import db  class UnidadeCompetencia(db.Model):  \_\_tablename\_\_ = "unidadecompetencia"  id = db.Column(db.Integer,primary\_key=True,autoincrement=True)  numero = db.Column(db.Integer)  nome = db.Column(db.String(200))  carga\_horaria = db.Column(db.Integer)  competencia\_geral= db.Column(db.String(255)) |
| R | Programando o **\_\_init\_\_.py** do app |
| S | Verifique a linha do Migrate (importante) |
| R | Texto  Descrição gerada automaticamente |
| S |  |
| T | Crie um banco de dados conforme o **config.py** |
| U | Inicie o migrations |
| V |  |
| X | Avaliando se os dois comandos anteriores deram certo: |
| Z |  |

# Etapa 9 – Criar novas rotas

|  |  |
| --- | --- |
| **rota** | **descrição** |
| /listuc | Listar unidades de competências  Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto  Descrição gerada automaticamente |
| /adduc | Adicionar unidade de competência  Interface gráfica do usuário, Texto, Aplicativo  Descrição gerada automaticamente |
| /deluc | Deletar unidade de competência  Interface gráfica do usuário, Texto, Aplicativo  Descrição gerada automaticamente |
| /upuc | Atualizar a unidade de competência  Interface gráfica do usuário, Texto, Aplicativo  Descrição gerada automaticamente |
| /meetuc | Encontrar uma unidade de competência  Interface gráfica do usuário, Texto, Aplicativo  Descrição gerada automaticamente |

Codificação:

**Crie um arquivo:** uc\_view.py

Uma imagem contendo Aplicativo

Descrição gerada automaticamente

*#Nela vou criar os métodos que minha aplicação vai executar*

"""

rota    descrição

/listuc Listar unidades de competências

/adduc  Adicionar unidade de competência

/deluc  Deletar unidade de competência

/upuc   Atualizar a unidade de competência

/meetuc Encontrar uma unidade de competência

"""

from app import app

from flask import render\_template

@app.route("/listuc")

def listar\_uc():

    return render\_template("uncompetencias/uc\_template.html")

@app.route("/adduc")

def adicionar\_uc():

    return "Adicionar unidade de competência"

@app.route("/deluc")

def excluir\_uc():

    return "Excluir unidade de competência"

@app.route("/upuc")

def atualizar\_uc():

    return "Atualizar unidade de competência"

@app.route("/meetuc",defaults={'nome':None},methods={'PUT'})

@app.route("/meetuc/<string:nome>")

def encontrar\_uc(nome):

    if nome:

        return render\_template("ucs/uc\_temp.html",nome\_uc=nome)

    else:

        return f"Navegue pelas UCs de forma personalizada"

@app.route("/numberuc/<int:numero>")

def number\_uc(numero):

    match(numero):

        case 11: return f"UC {numero} -Lógica de Programação"

        case 12: return f"UC {numero} -Informática Básica"

        case \_:return f" Dados inexistente"

código preto e branco

#Nela vou criar os métodos que minha aplicação vai executar

"""

rota descrição

/listuc Listar unidades de competências

/adduc Adicionar unidade de competência

/deluc Deletar unidade de competência

/upuc Atualizar a unidade de competência

/meetuc Encontrar uma unidade de competência

"""

from app import app

from flask import render\_template

@app.route("/listuc")

def listar\_uc():

return render\_template("uncompetencias/uc\_template.html")

@app.route("/adduc")

def adicionar\_uc():

return "Adicionar unidade de competência"

@app.route("/deluc")

def excluir\_uc():

return "Excluir unidade de competência"

@app.route("/upuc")

def atualizar\_uc():

return "Atualizar unidade de competência"

@app.route("/meetuc",defaults={'nome':None},methods={'PUT'})

@app.route("/meetuc/<string:nome>")

def encontrar\_uc(nome):

if nome:

return render\_template("ucs/uc\_temp.html",nome\_uc=nome)

else:

return f"Navegue pelas UCs de forma personalizada"

@app.route("/numberuc/<int:numero>")

def number\_uc(numero):

match(numero):

case 11: return f"UC {numero} -Lógica de Programação"

case 12: return f"UC {numero} -Informática Básica"

case \_:return f" Dados inexistente"

## Estude sobre parâmetros nas rotas:

Texto

Descrição gerada automaticamente

## Programe com lógica dentro das rotas

Texto

Descrição gerada automaticamente

## Teste as rotas

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Site

Descrição gerada automaticamente

Figura 1 - Fortemente tipada (o erro se deu pelo tipo)

## Criando duas rotas para o mesmo método

O aluno pode passar como parâmetro o nome da uc e assim irá mostrar a uc escolhida para estudo. Ou ele não quer escolher uc agora quer apenas navegar no site.

Nesse caso podemos criar duas rotas uma para cada situação.

Vamos usar a rota meetuc

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

Solução:

Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email, Site

Descrição gerada automaticamente

Figura 2 - O cara não quis escolher uma UC agora

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Figura 3 - Nesse caso ele escolheu Java

# Tipos de conversões de rotas

Cada parâmetro de uma rota deve possuir um tipo válido e, assim, evitar que uma rota aceita parâmetros com que o método da view não saiba trabalhar. Sendo assim, na tabela abaixo podemos ver todos os tipos de parâmetros que o Flask suporta:

* **string**: aceita qualquer texto sem barra (padrão);
* **int**: aceita valores positivos inteiros;
* **float**: aceita valores positivos ponto flutuantes;
* **path**: como string, mas aceita barra;
* **uuid**: aceita strings UUID.

# Pra não dizer que não falei de restringir rotas

Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Site

Descrição gerada automaticamente

Agora fiquei proibido de acessar a rota.

Vamos mudar agora para ele aceitar via GET e POST

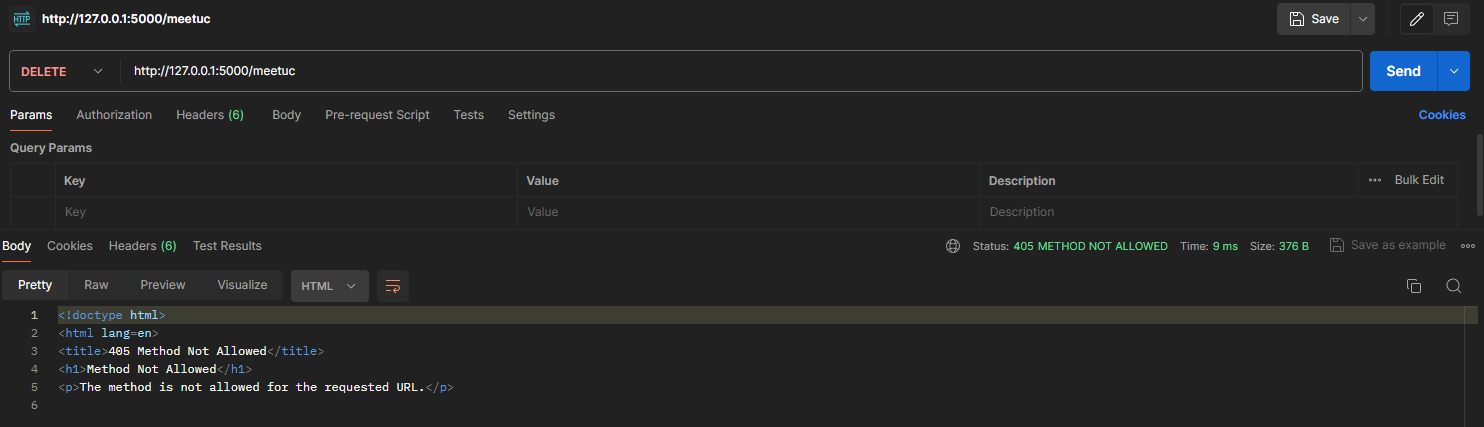
Interface gráfica do usuário, Texto, Aplicativo, Site

Descrição gerada automaticamente

Tela preta com letras brancas

Descrição gerada automaticamente

Testando as rotas no POSTMAN veja que deu o erro pois ele aceita só GET e POST



Se ele usar o get

Tela de celular

Descrição gerada automaticamente

Tela de celular

Descrição gerada automaticamente

Mude para PUT

Texto

Descrição gerada automaticamente

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

Tela de celular

Descrição gerada automaticamente

Figura 4 - deu certo

# Pra não dizer que não falei de Views

# O que são views?

A camada View é responsável por tramitar as informações obtidas pelo model e exibi-las ao usuário (seja através de um template ou não). É ela quem faz o “meio de campo” para que as informações sejam exibidas para quem está acessando a aplicação.

Uma view é composta por uma ou mais ações. Cada ação representa um recurso que a aplicação dispõe.

# O que são templates?

Após todos os dados estiverem prontos para serem exibidos ao usuário, eles são repassados para o template. Esta camada é responsável por exibir as informações para o usuário utilizando páginas HTML.

O Flask utiliza o Jinja2 como sistema de templates. Ele é bem completo e que provê diversas facilidades para a renderização das informações em páginas HTML.

Durante o curso veremos como utilizar os principais recursos do sistema de templates do Flask.

**TEMPLATES**

(mostra os dados para os usuário e interage com o mesmo)

**Views**

(trata as requisições e envia os dados para a camada Templates)

Observe que as informações que quero apresentar para o usuário no código anterior encontra-se dentro da VIEW

Texto

Descrição gerada automaticamente

# Pra não dizer que não falei sobre TEMPLATES

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Figura 5 - Crie uma estrutura do template e um arquivo .html

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Vamos agora para a nossa view tratar a requisição para enviar dados para o template.

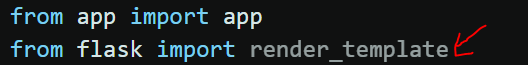
Vou escolher esse aqui

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Primeiro você tem que importar

from flask import **render\_template**



Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Interface gráfica do usuário

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente com confiança média

# Pra não falar que não falei de inspecionar

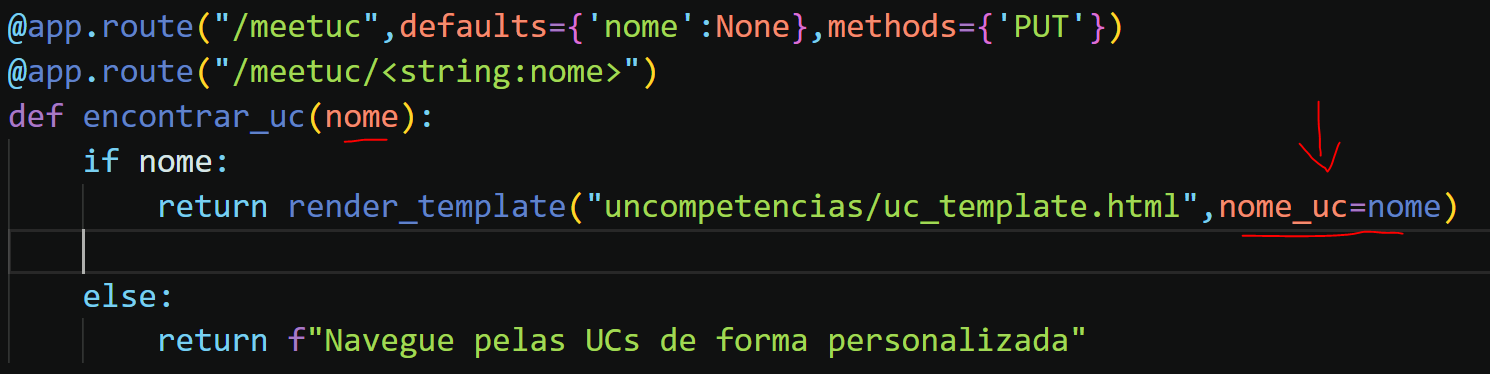
Inspecione os códigos com seus alunos

# Pra não dizer que não falei de passar valores

Vamos usar o seguinte método

Texto

Descrição gerada automaticamente



Interface gráfica do usuário, Texto, Site

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

# Para não dizer que não falei de Jinja 2

Texto

Descrição gerada automaticamente

Tela preta com letras brancas

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

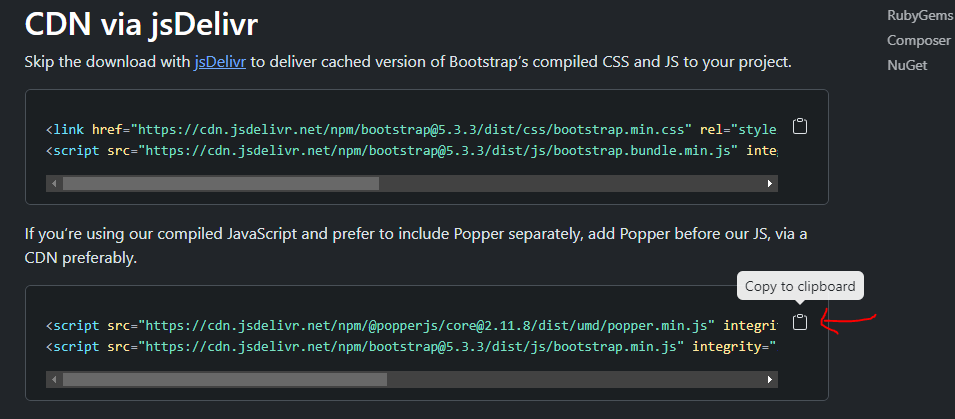
Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

# Pra não dizer que não falei de Bootstrap

Framework CSS com fontes, estilos e grids personalizados e é muito utilizado no mercado

Podemos inserir via CDN



Primeiros passos baixar os arquivos download

Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Figura 6 – descompacte

Crie uma pasta static

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Vamos colocar os dois arquivos bootstrap

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

# Anexo I - Os tipos na Linguagem Python

Tela de celular com texto branco sobre fundo preto

Descrição gerada automaticamente