

# O padrão MVC e programação no servidor

Disciplina: Desenvolvimento Web

Prof. Dr. Rafael Will M. de Araujo



## 1 O padrão MVC

## 2 O módulo Flask

# O que é MVC?

- É um padrão de arquitetura de software que auxilia no desenvolvimento de aplicações Web.

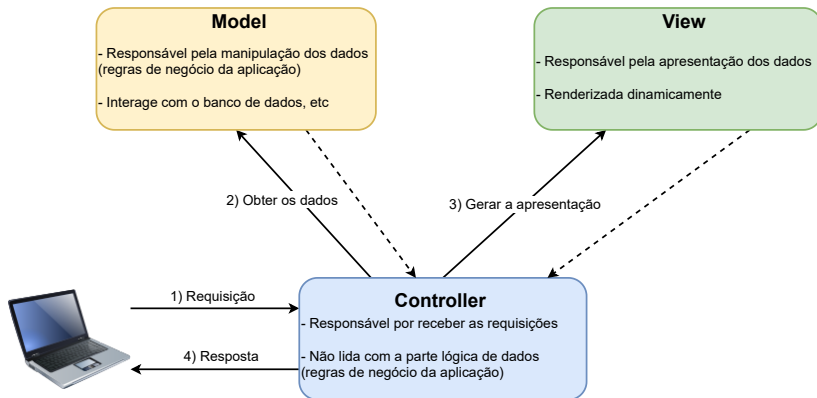
# O que é MVC?

- É um padrão de arquitetura de software que auxilia no desenvolvimento de aplicações Web.
- O padrão arquitetural MVC (*Model-View-Controller*) foi criado por um engenheiro da Xerox, Trygve Reenskaug, em 1979. A ideia era criar um padrão de estrutura de aplicações feitas com a linguagem Smalltalk, criando uma estrutura em camadas para diminuir o acoplamento e aumentar a coesão entre as classes no projeto, facilitando a manutenção do sistema.

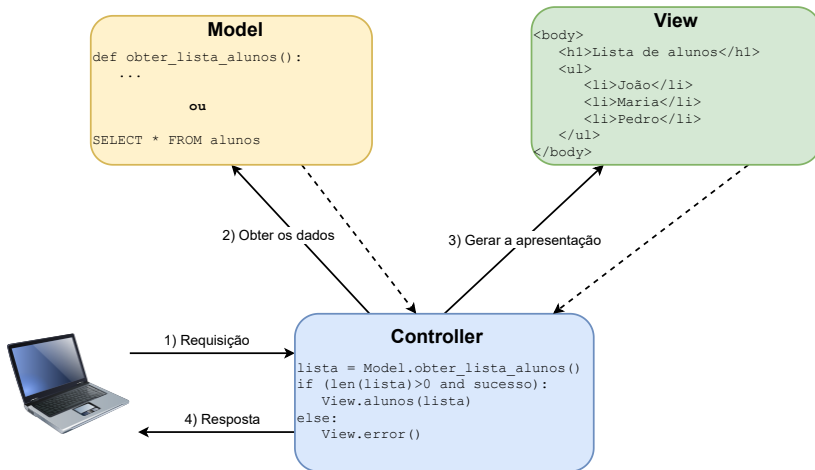
# O que é MVC?

- É um padrão de arquitetura de software que auxilia no desenvolvimento de aplicações Web.
- O padrão arquitetural MVC (*Model-View-Controller*) foi criado por um engenheiro da Xerox, Trygve Reenskaug, em 1979. A ideia era criar um padrão de estrutura de aplicações feitas com a linguagem Smalltalk, criando uma estrutura em camadas para diminuir o acoplamento e aumentar a coesão entre as classes no projeto, facilitando a manutenção do sistema.
- O padrão consiste nas três camadas que dão o seu nome: **Modelo** (*Model*), **Visualização** (*View*) e **Controle** (*Controller*), assim cada camada é responsável por uma responsabilidade no sistema.

# Fluxo em uma aplicação MVC



# Fluxo em uma aplicação MVC



1 O padrão MVC

2 O módulo Flask



# O que é o Flask?

- O Flask é um micro framework escrito em Python para a construção de aplicações web. O micro vem do fato de que o framework em si é bem pequeno, concentrando-se apenas nas tarefas mais importantes, deixando pontos de extensão para que novas capacidades sejam acopladas a ele de outras maneiras.
- Pela sua simplicidade, o Flask está pronto para ser usado nas mais diversas aplicações em produção, com amplas capacidades de escalabilidade. Usado por grandes empresas de tecnologia como Airbnb, Uber, Reddit, Mozilla, Netflix, dentre outras.
  - ▷ Há uma lista mais completa aqui: <https://github.com/rochacbruno/flask-powered>

# Instalação do Flask (Windows)

- Para instalar o Flask no Windows, utilize o seguinte comando no *cmd* ou no shell do VSCode:

## Instalação do Flask no Windows

```
py -m pip install -U flask
```

- Após executar o comando acima, verifique se a instalação foi realizada corretamente com o seguinte comando:

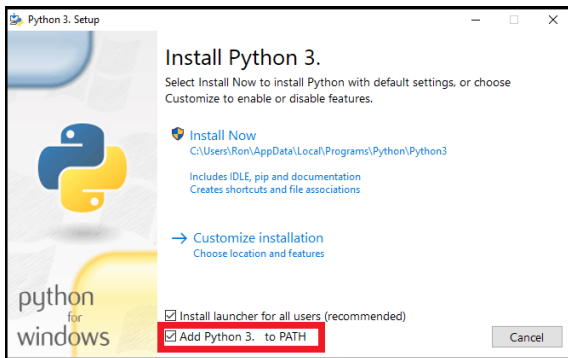
## Verificando a instalação no Windows

```
py -m flask --version
```

A saída deve ser uma versão válida do Flask.

# Instalação do Flask (Windows)

- Caso você tenha problemas para instalar, verifique se:
  - ▷ 1) você possui permissão de administrador no computador;
  - ▷ 2) você marcou a opção **"Add Python 3.x to PATH"** quando instalou o Python.
- Recomendação: tente reinstalar o Python marcando a opção da figura abaixo (quase sempre funciona!)



# Instalação do Flask (Linux/Mac)

- Para instalar o Flask no Linux ou Mac, utilize o seguinte comando no *terminal* ou no shell do VSCode:

## Instalação do Flask no Linux/Mac

```
python3 -m pip install -U flask
```

## Verificando a instalação no Linux/Mac

```
python3 -m flask --version
```

# A primeira aplicação em Flask

## Primeira aplicação

```
from flask import Flask

app = Flask(__name__)

app.run()
```

- A variável `app` é um objeto que representa a nossa aplicação em Flask.
  - ▷ A variável `__name__` representa o nome do módulo. A aplicação precisa saber onde ela está localizada, e utilizar a variável `__name__` é uma maneira conveniente de fazer isso.
  - ▷ Mas também poderíamos utilizar uma string qualquer para ser o nome da nossa aplicação.
- Note que a seguinte mensagem será exibida ao executar o código:
  - ▷ \* Running on `http://127.0.0.1:5000/` (Press CTRL+C to quit)
  - ▷ `127.0.0.1` é um endereço IP especial que aponta para a nossa próprio computador, enquanto que `5000` é a porta na qual o servidor HTTP criado escuta as requisições HTTP.
  - ▷ Experimente também trocar o endereço `127.0.0.1` por `localhost`. Lembre-se de manter o número da porta!
- Teste abrir o navegador no endereço indicado acima.

# Primeiros passos com Flask

- Note que a aplicação resultou numa página com erro 404. Isso acontece porque não configuramos nenhuma rota na nossa aplicação, isto é, não existe nenhum recurso para responder à requisição.
- Podemos configurar rotas na nossa aplicação utilizando a construção a seguir:

## Hello World no Flask

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def ola():
    return "Hello World no Flask"

app.run()
```

- Aqui você vai notar que tivemos que utilizar uma função em Python com um decorador, e o conteúdo gerado (e retornado) por essa função será recebido como **resposta** da requisição do navegador.
- Acesse novamente o endereço indicado e veja o que acontece.

# Primeiros passos com Flask

- Como a aplicação Flask é uma aplicação HTTP, precisamos de um simples controle que responda a uma requisição de GET para testarmos. Controles no Flask são funções que retornam respostas.
- Para que o Flask saiba de qual caminho (*path*) da URL responderá esse controle, podemos registrar essa informação usando um decorador (decorator) específico, o `@app.route`

## Adicionando rotas

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    return 'Essa é a página principal'

@app.route('/ola')
def ola():
    return 'Bem-vindo à página "ola"'

@app.route('/outra_pagina')
def uma_funcao_qualquer():
    return 'Essa é outra página qualquer da nossa aplicação'

app.run()
```

# Resposta com HTML

- O que acontece se adicionarmos *tags* HTML às nossas strings?

## Adicionando HTML

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    resposta = '<h1>Essa é a página principal</h1>'
    resposta += '<p><a href="ola">Clique aqui</a> para ir à página "ola"</p>'
    return resposta

@app.route('/ola')
def ola():
    resposta = '<h1>Bem-vindo à página "ola"</h1>'
    resposta += '<p>Página de exemplo do slide</p>'
    return resposta

app.run()
```

- Elas serão interpretadas pelo navegador!!
- Ou seja, numa aplicação em Flask o HTML será representado como strings em Python.
  - ▷ Mas calma, veremos mais adiante uma forma muito mais elegante de trabalhar com HTML no Flask!



# Alguns parâmetros adicionais da aplicação

- Note que a cada modificação, tivemos que parar a aplicação e reiniciá-la.
- Para evitar fazer isso o tempo todo durante a etapa de desenvolvimento, podemos utilizar o parâmetro nomeado `debug=True` no método `app.run()`:

## Aplicação em modo *debug*

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    resposta = '<h1>Essa é a página principal</h1>'
    resposta += '<p><a href="ola">Clique aqui</a> para ir à página "ola"</p>'
    return resposta

@app.route('/ola')
def ola():
    resposta = '<h1>Bem-vindo à página "ola"</h1>'
    resposta += '<p>Página de exemplo do slide</p>'
    return resposta

app.run(debug=True)
```

# Alguns parâmetros adicionais da aplicação

- A porta também pode ser alterada com o parâmetro `port=NUMERO_DA_PORTA`:

## Aplicação em modo *debug*

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    resposta = '<h1>Essa é a página principal</h1>'
    resposta += '<p><a href="ola">Clique aqui</a> para ir à página "ola"</p>'
    return resposta

@app.route('/ola')
def ola():
    resposta = '<h1>Bem-vindo à página "ola"</h1>'
    resposta += '<p>Página de exemplo do slide</p>'
    return resposta

app.run(debug=True, port=8080)
```

# Parametrização de rotas

- Uma característica interessante do Flask é que podemos adicionar parâmetros às nossas rotas. Essa técnica é conhecida como *path parameter*.

## Exemplo de rotas com parâmetros (*path parameters*)

```
from flask import Flask

app = Flask(__name__)

@app.route('/curso/<nome_curso>')
def curso(nome_curso):
    resposta = 'O curso passado por parâmetro é: ' + nome_curso
    return resposta

@app.route('/exemplo/<int:ano>')
def exemplo(ano):
    resposta = 'O ano informado é: ' + str(ano)
    return resposta

@app.route('/curso/<nome_curso>/<int:numero_turma>')
def outro_exemplo(nome_curso, numero_turma):
    resposta = 'O nome do curso e o numero da turma é: ' + nome_curso + ' ' + str(numero_turma)
    return resposta

app.run(debug=True)
```

- Note que cada um dos *path parameters* são colocados entre os símbolos <>, e as funções (*controllers*) de cada rota adicionamos um parâmetro (na mesma ordem) para cada *path parameter*.

# Parametrização de rotas

- Por padrão esses parâmetros são interpretados como strings, mas se quisermos forçar um tipo de dado específico, podemos colocar o tipo na frente do nome, separado por dois pontos:
  - ▷ `<int:param>` - para inteiros positivos
  - ▷ `<float:param>` - para decimais positivos
  - ▷ `<path:param>` - igual a string, mas aceita barras no meio
  - ▷ `<uuid:param>` - aceita objetos do tipo **UUID**

# Criação de templates

- Como vimos anteriormente, uma forma (nada produtiva) de trabalhar com HTML no Flask é através de strings no Python.
- Felizmente, podemos trabalhar com arquivos HTML como fazíamos antes, através do conceito de *templates*.
- Para isso, vamos criar um projeto no VSCode (organizado numa pasta inicialmente vazia). Neste projeto (pasta), vamos criar 3 coisas:
  - ▷ Uma pasta vazia chamada: *templates*;
  - ▷ Um arquivo *app.py* que representa a nossa aplicação;
  - ▷ Um arquivo *pagina.html* que representa uma página HTML que desejamos renderizar utilizando o Flask.

## Exibindo uma página através de *templates*

```
from flask import Flask
from flask import render_template

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('pagina.html')

app.run(debug=True)
```

# Criação de templates

- Como vimos anteriormente, uma forma (nada produtiva) de trabalhar com HTML no Flask é através de strings no Python.
- Felizmente, podemos trabalhar com arquivos HTML como fazíamos antes, através do conceito de *templates*.
- Para isso, vamos criar um projeto no VSCode (organizado numa pasta inicialmente vazia). Neste projeto (pasta), vamos criar 3 coisas:
  - ▷ Uma pasta vazia chamada: *templates*;
  - ▷ Um arquivo *app.py* que representa a nossa aplicação;
  - ▷ Um arquivo *pagina.html* que representa uma página HTML que desejamos renderizar utilizando o Flask.

## Exibindo uma página através de *templates*

```
from flask import Flask
from flask import render_template

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('pagina.html')

app.run(debug=True)
```

- Note que vamos utilizar uma função do Flask chamada `render_template` para que o Flask leia o HTML, interprete-o e o devolva na resposta para o Cliente.

# Criação de templates

- Como vimos anteriormente, uma forma (nada produtiva) de trabalhar com HTML no Flask é através de strings no Python.
- Felizmente, podemos trabalhar com arquivos HTML como fazíamos antes, através do conceito de *templates*.
- Para isso, vamos criar um projeto no VSCode (organizado numa pasta inicialmente vazia). Neste projeto (pasta), vamos criar 3 coisas:
  - ▷ Uma pasta vazia chamada: *templates*;
  - ▷ Um arquivo *app.py* que representa a nossa aplicação;
  - ▷ Um arquivo *pagina.html* que representa uma página HTML que desejamos renderizar utilizando o Flask.

## Exibindo uma página através de *templates*

```
from flask import Flask
from flask import render_template

app = Flask(__name__)

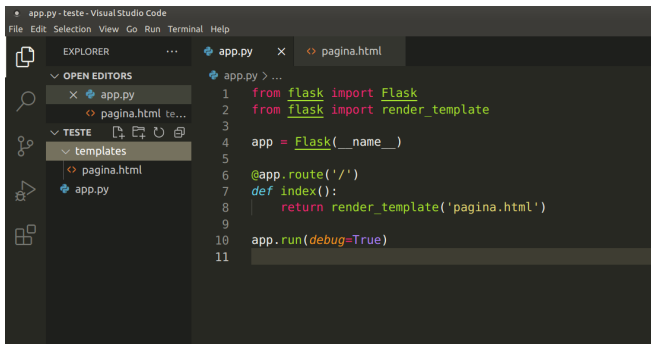
@app.route('/')
def index():
    return render_template('pagina.html')

app.run(debug=True)
```

- Note que vamos utilizar uma função do Flask chamada `render_template` para que o Flask leia o HTML, interprete-o e o devolva na resposta para o Cliente.
- Note também que não precisamos especificar a pasta *templates* informar o arquivo *pagina.html*. Isso acontece porque o Flask utiliza a pasta *templates* por padrão para procurar esses arquivos HTML.

# Criação de templates

- Estrutura do projeto criado dentro do VSCode:



The screenshot shows the Visual Studio Code interface with the following details:

- Explorer Panel:** Shows the project structure with folders `TESTE` and `templates`. The `templates` folder contains `pagina.html`. The `app.py` file is also visible.
- Open Editors:** Lists `app.py` and `pagina.html`.
- Code Editor:** Displays the content of `app.py`, which is a Flask application script.

```
1 from flask import Flask
2 from flask import render_template
3
4 app = Flask(__name__)
5
6 @app.route('/')
7 def index():
8     return render_template('pagina.html')
9
10 app.run(debug=True)
11
```