Etec
Registro

python™

Flask

Programação Web III

Prof. Diego Max

Etec
Registro

python™

Aula 02: Arquitetura MVC e Request

Controllers no MVC

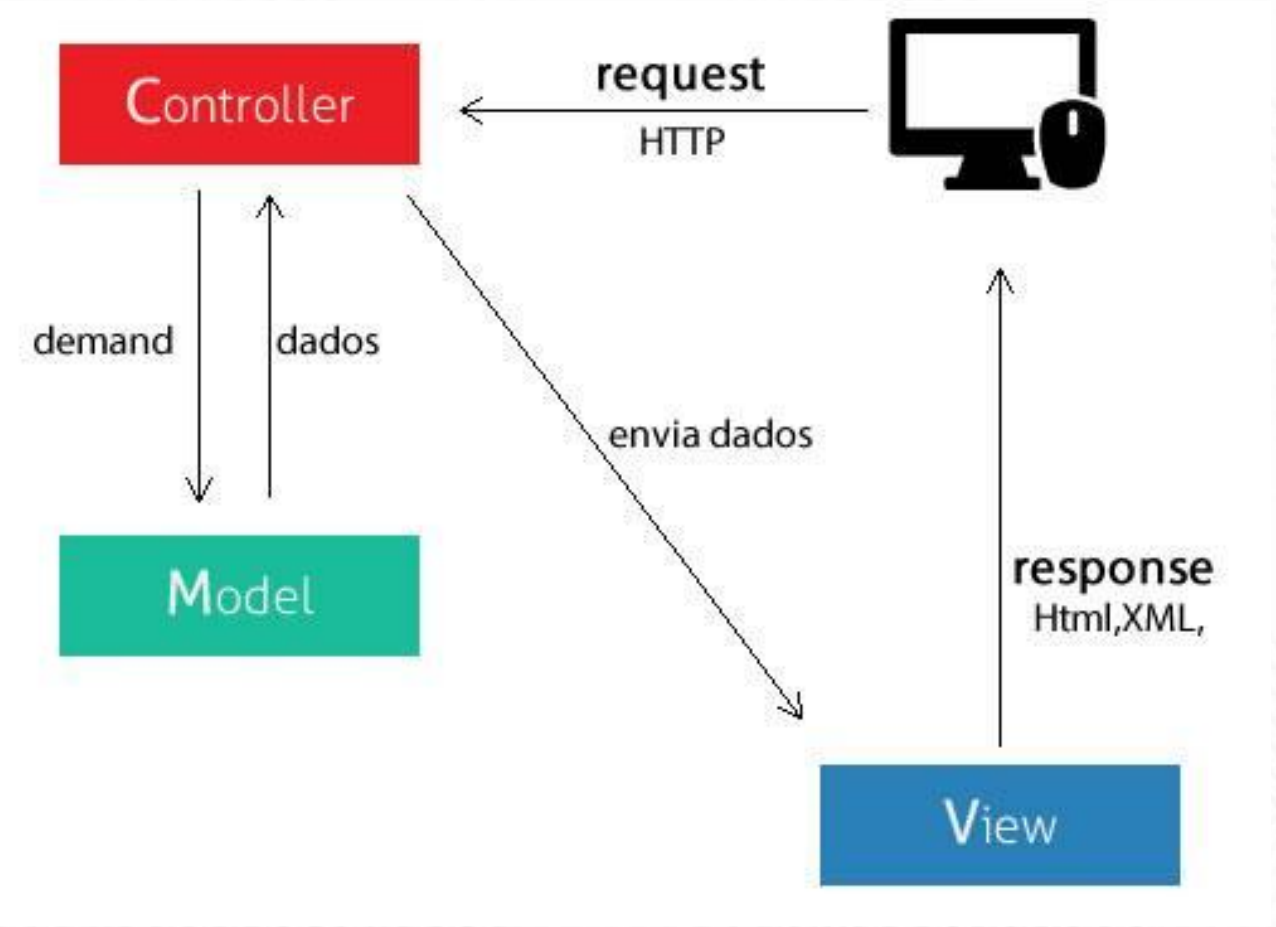
Flask

A arquitetura MVC é nada mais que um padrão de arquitetura de software, separando sua aplicação em 3 camadas.

A camada de interação do usuário (view), a camada de manipulação dos dados (model) e a camada de controle (controller).

O Controller é responsável por receber todas as requisições do usuário. Seus métodos chamados actions são responsáveis por uma página, controlando qual model usar e qual view será mostrado ao usuário.

A imagem ao lado representa o fluxo do MVC em um contexto de Internet, com uma requisição HTTP e resposta em formato HTML.



Controllers no MVC

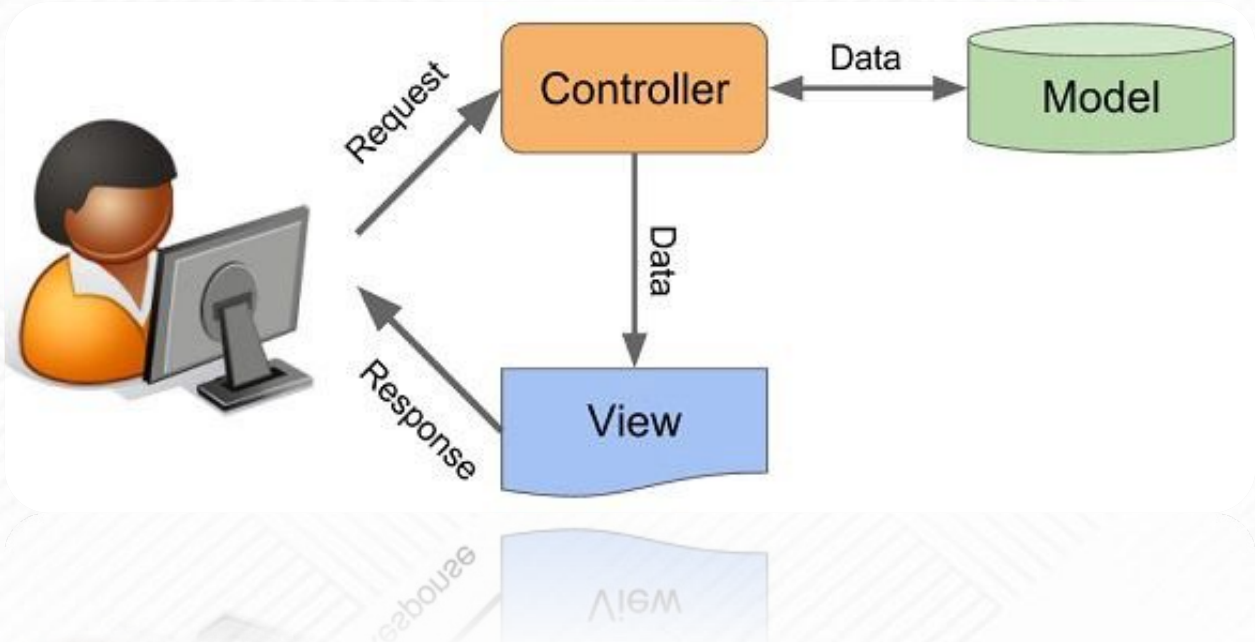
Flask

Resumindo a arquitetura MVC temos:

Models: responsável pela manipulação dos dados;

Views: responsável pela interação com o usuário;

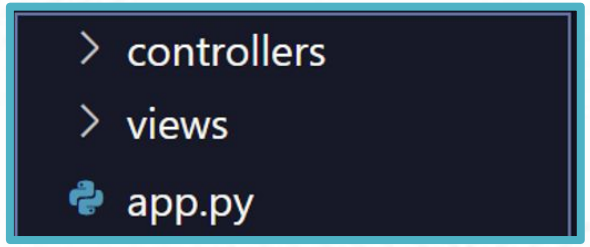
Controllers: camadas de controle.



Arquitetura MVC - Controllers

Flask

Seguindo a arquitetura MVC, criaremos agora nosso primeiro controller. Para isso, é necessário criar uma pasta com o nome “controllers” dentro da pasta do seu projeto.



Dentro da pasta controllers criaremos o arquivo “routes.py”. Esse arquivo será nosso controle de rotas, onde irão ficar todas as rotas do nosso projeto.

No arquivo “routes.py” iremos importar o módulo render_template e criar uma função chamada “init_app()” com o parâmetro “(app)”.

```
from flask import render_template
def init_app(app):
```

Arquitetura MVC - Controllers

Flask

Após isso, basta copiarmos os códigos com as rotas que estavam no arquivo “app.py” e passar para a função “init_app” do arquivo “routes.py”, conforme abaixo:

```
from flask import render_template

def init_app(app):
    @app.route('/')
    def home():
        return render_template('index.html’)

    @app.route('/games')
    def games():
        games = {'Título' : 'CS-GO', 'Ano' : 2012, 'Categoria' : 'FPS Online'}
        jogadores = ['Pedro', 'João', 'Marcos', 'Maria']
        return render_template('games.html',games=games,jogadores=jogadores)
```

Arquitetura MVC - Controllers

Agora o arquivo “app.py” ficará assim:

```
from flask import Flask, render_template


app = Flask(__name__, template_folder='views')


if __name__ == '__main__':
    app.run(host='localhost', port=5000, debug=True)
```

O que precisamos fazer agora é importar o arquivo “routes.py” que está dentro de “controllers” para nosso arquivo “app.py”, conforme o código a seguir:

```
from flask import Flask, render_template
from controllers import routes
```

Arquitetura MVC - Controllers

Flask

Após isso chamaremos a função “init_app()” passando como argumento o app, conforme a seguir:

```
from flask import Flask, render_template
from controllers import routes

app = Flask(__name__, template_folder='views')
routes.init_app(app)

if __name__ == '__main__':
    app.run(host='localhost', port=5000, debug=True)
```

Pronto! Agora já temos nossas rotas separadas em um controller sendo devidamente importadas no nosso arquivo principal “app.py”.

Adicionando dados em listas com Request

Flask

Agora começaremos a inserir dados de forma dinâmica em nossa página.

Esses dados virão de um formulário que será preenchido pelo usuário e serão adicionados em uma lista do Python para ser exibida na página. O primeiro passo é criar nosso formulário na página “games.html”, conforme o código a seguir:

```
    </ol>
    <hr>
    <h4>Entrar no jogo</h4>
    <form>
        <label for="jogador">Jogador:</label>
        <input type="text" name="jogador">
        <input type="submit" value="Entrar">
    </form>
</body>
</html>
```

Adicionando dados em listas com Request

Criado nosso formulário, no arquivo “routes.py” iremos fazer a importação da biblioteca Request.

```
from flask import render_template, request
```

O Python Requests é uma biblioteca de código aberto em Python que simplifica o processo de fazer requisições HTTP.

Com essa biblioteca, os desenvolvedores podem facilmente enviar solicitações HTTP para servidores da web e receber as respostas correspondentes.

Em suma, a biblioteca Requests no Python serve para fazer solicitações e requisições em uma base ou conjunto de dados. Assim, é essencial para estabelecer a interação entre uma base e uma aplicação em Python.

A biblioteca é formada por quatro principais métodos: Get, Post, Patch e Delete.

Adicionando dados em listas com Request

O próximo passo é apagarmos nossa lista “jogadores” que está dentro de um escopo local da função “games()” e cria-la vazia, agora no escopo global da aplicação, conforme o código a seguir:

```
from flask import render_template, request


jogadores = []
```

Feito isso, na rota “/games” iremos informar os métodos que essa rota poderá receber, no caso os métodos GET e POST.

```
    @app.route('/games', methods=['GET', 'POST'])
    def games():
        games = {'Título' : 'CS-GO',
                 'Ano' : 2012,
                 'Categoria' : 'FPS Online'}
        return render_template('games.html',
                               games=games,jogadores=jogadores)
```

Adicionando dados em listas com Request

No nosso formulário HTML iremos definir agora, duas propriedades, a action e o method.

Com a função “url_for()” indicamos para qual a função o formulário irá apontar quando for enviado, no nosso caso para a função “games()”, sendo essa função a responsável por tratar os dados recebidos. Já o método será o POST.

```
    <form action="{{url_for('games')}}", 
method="POST">
        <label for="jogador">Jogador:</label>
        <input type="text" name="jogador">
        <input type="submit" value="Entrar">
    </form>
    <br>
```

Agora iremos tratar a requisição que chegará para a função “games()”.

Adicionando dados em listas com Request

Flask

Primeiro iremos verificar se o método recebido é POST com a linha if request.method == 'POST'. Em seguida, verificamos se o request.form.get('jogador') existe.

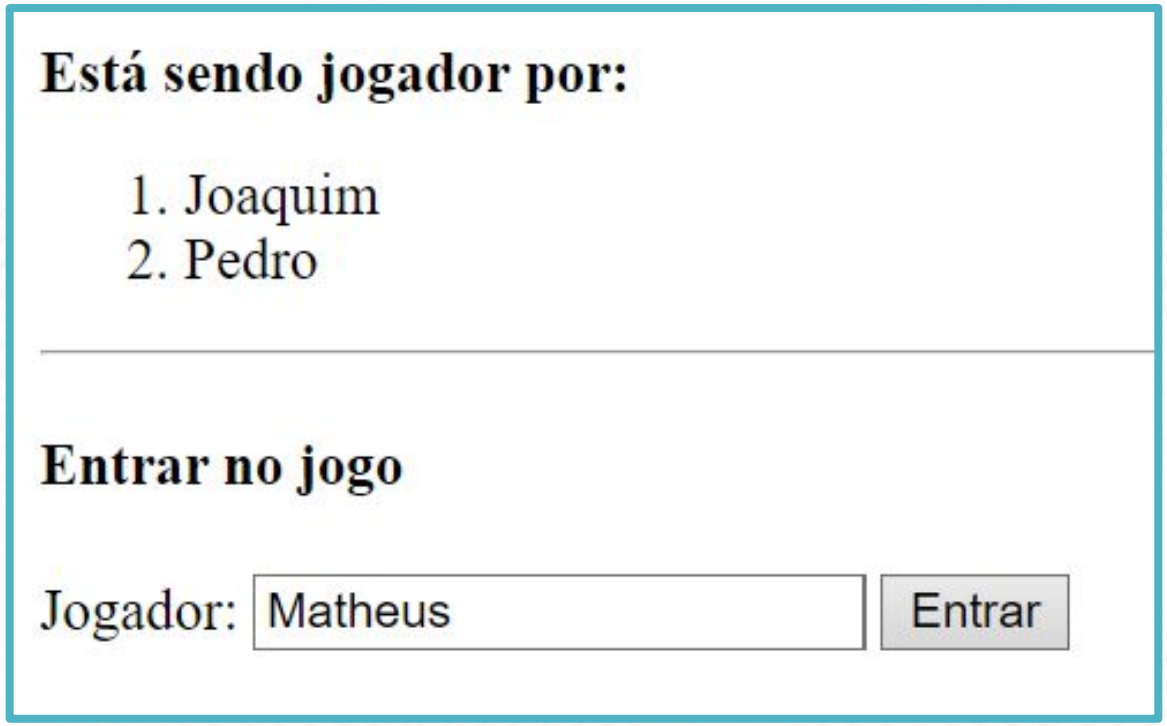
O request.form.get é responsável por receber os dados que foram enviado pelo formulário. O argumento 'jogador' é o name do campo que definimos no formulário. Satisfeito essas duas condições, iremos adicionar o dado a lista "jogadores", através da linha jogadores.append(request.form.get('jogador')).

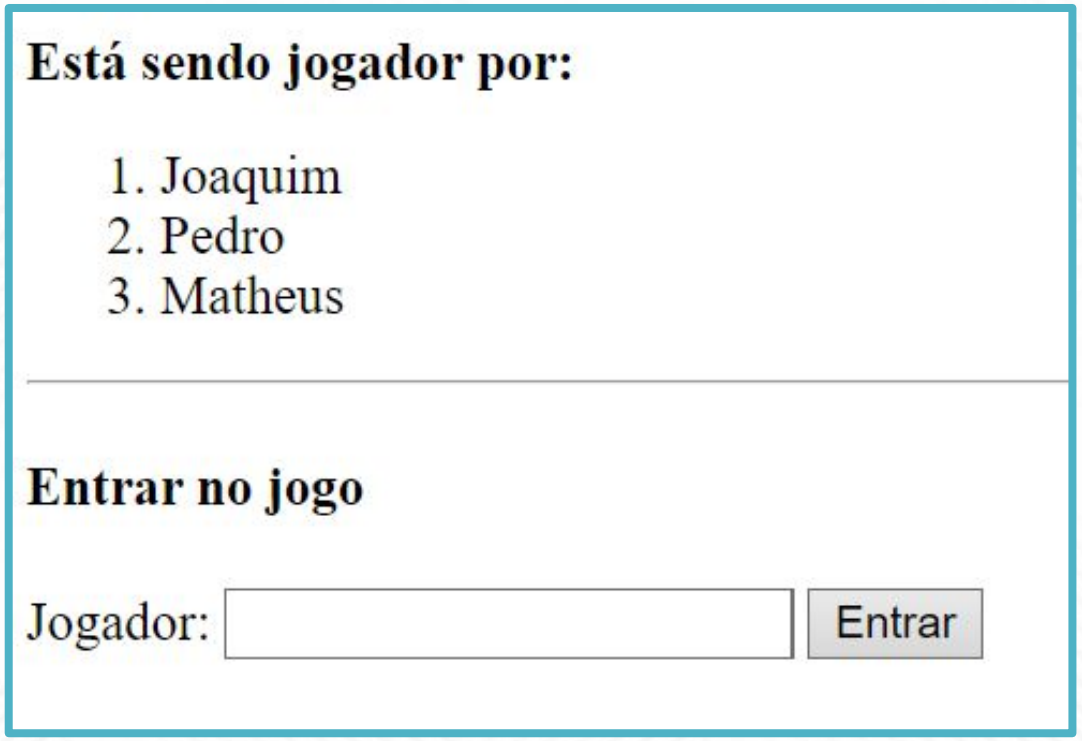
```
@app.route('/games', methods=['GET', 'POST'])
    def games():
        games = {'Título' : 'CS-GO', 'Ano' : 2012, 'Categoria' : 'FPS Online'}

        if request.method == 'POST':
            if request.form.get('jogador'):
                jogadores.append(request.form.get('jogador'))
        return render_template('games.html', games=games, jogadores=jogadores)
```

Adicionando dados em listas com Request

Flask

Agora nossa aplicação já consegue receber dados enviados através do formulário e exibi-los na página





Adicionando dados em dicionários com Request

Iremos agora inserir dados na página através da estrutura de dados dicionário. O primeiro passo é criarmos um dicionário com o nome "gamelist" com escopo global, conforme abaixo:

```
from flask import render_template, request


jogadores = []
gamelist = [{'Título' : 'CS-GO', 'Ano' : 2012, 'Categoria' : 'FPS Online'}]
```

Esse dicionário contém apenas os dados do primeiro game (o game mais jogado), os outros serão adicionados no dicionário através de um formulário. Na função "games()", o antigo dicionário que tinha sido criado receberá agora apenas o primeiro item do dicionário "gamelist", conforme código abaixo:

```
    @app.route('/games', methods=['GET', 'POST'])
    def games():
        game = gamelist[0]
```

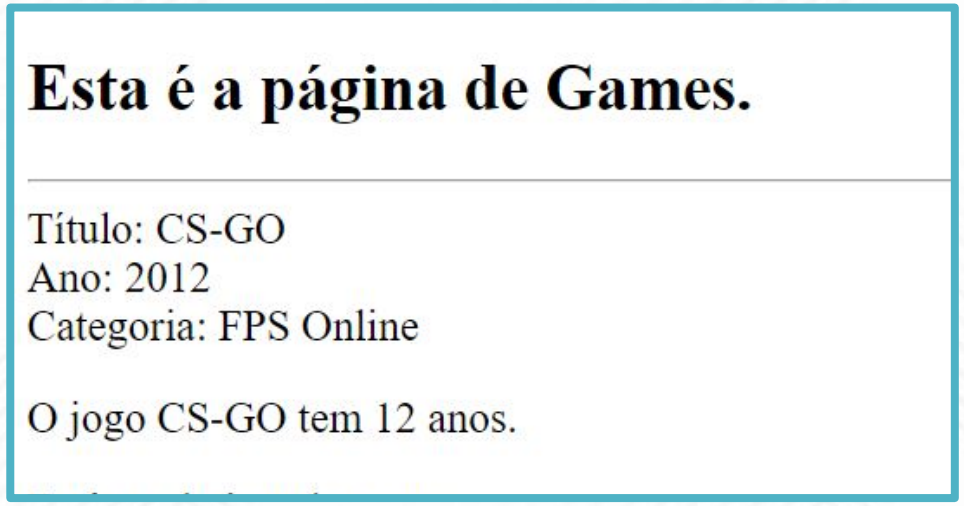
Note também que mudamos seu nome de "games" para "game".

Adicionando dados em dicionários com Request

Na função return render_templete() o nome da variável também deve ser alterada de “games” para “game”:

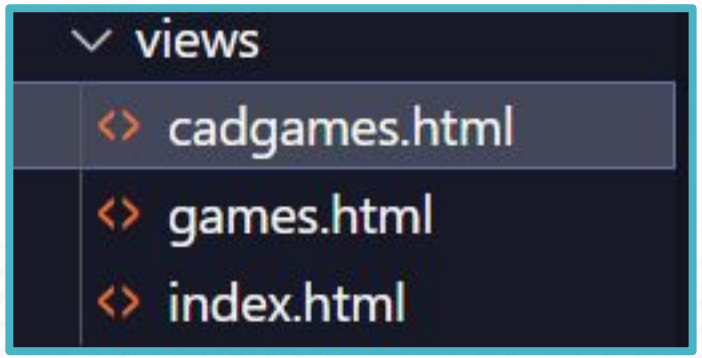
```
        if request.method == 'POST':
            if request.form.get('jogador'):
                jogadores.append(request.form.get('jogador'))
        return render_template('games.html',game=game, jogadores=jogadores)
```

Assim na página “games.html”, continuamos com o mesmo resultado:



Adicionando dados em dicionários com Request

Agora iremos criar nossa página de cadastro de games, para isso criaremos um arquivo com o nome "cadgames.html" em nossa pasta de views.



No arquivo "routes.py", adicionaremos uma nova rota "/cadgames", que permitirá receber os métodos GET e POST. Na função "cadgames()" iremos renderizar a página "cadgames.html" e passar o dicionário "gamelist" como variável para a página, conforme código a seguir:

```
@app.route('/cadgames', methods=['GET', 'POST'])
def cadgames():
    return render_template('cadgames.html',
                           gamelist=gamelist)
```

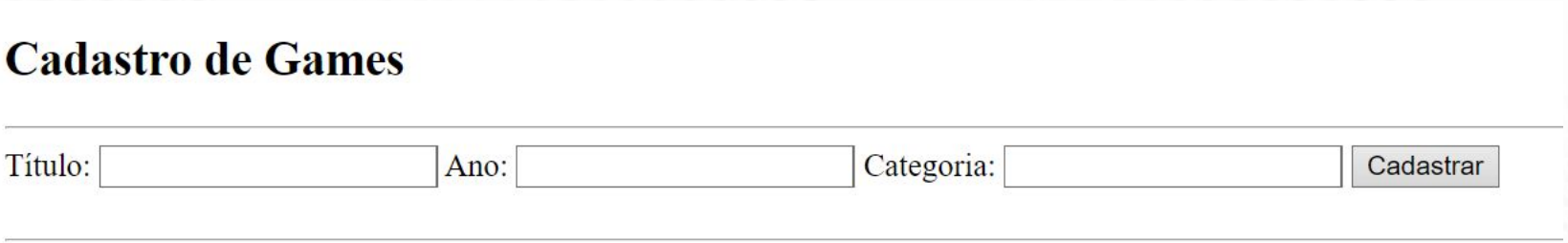
Adicionando dados em dicionários com Request

Feito isso, criaremos agora o conteúdo da página “cadgames.html”, ela terá um formulário de cadastro, esse formulário apontará para função “cadgames()” e terá o método POST, conforme código a seguir:

```
<title>Cadastro de Games</title>
</head>
<body>
    <h2>Cadastro de Games</h2>
    <hr>
    <form action="{{url_for('cadgames')}}", method="POST">
        <label for="titulo">Título:</label>
        <input type="text" name="titulo" required>
        <label for="ano">Ano:</label>
        <input type="text" name="ano" required>
        <label for="categoria">Categoria:</label>
        <input type="text" name="categoria" required>
        <input type="submit" value="Cadastrar">
    </form>
    <br>
    <hr>
```

Adicionando dados em dicionários com Request

O resultado da página será esse:

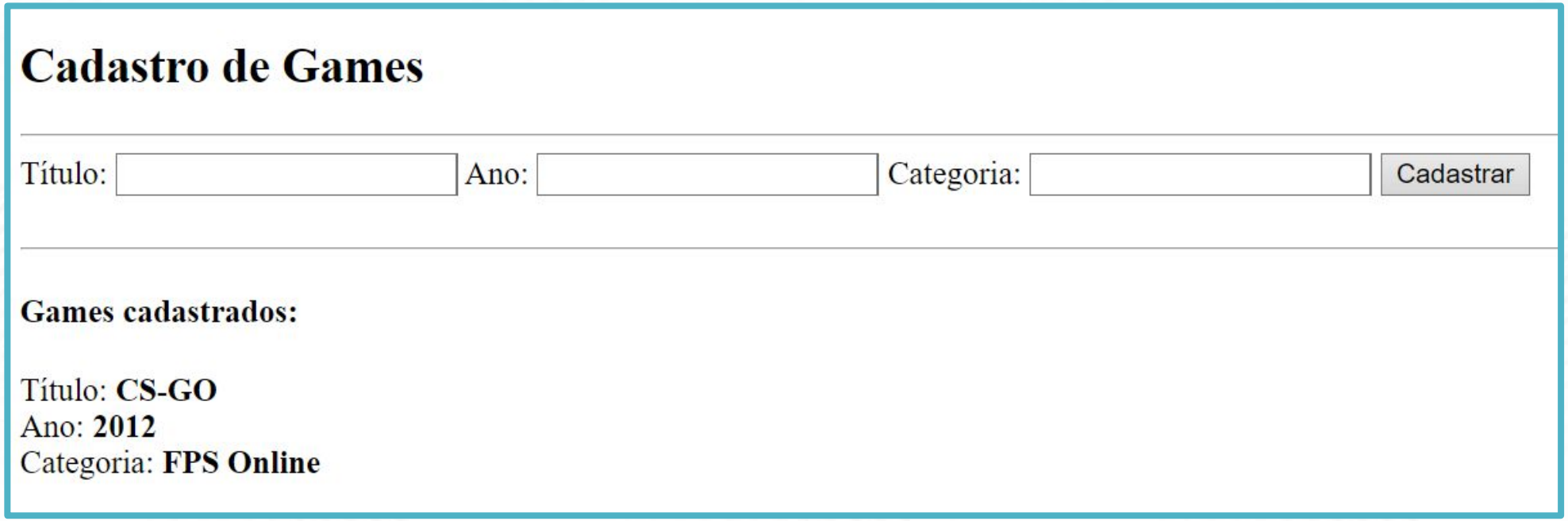


Agora, logo abaixo do formulário que acabamos de criar, iremos incluir o código que irá listar os novos games cadastrados, dentro de uma estrutura de repetição for, conforme código a seguir:

```
    <hr>
    <h4>Games cadastrados:</h4>
    {% for game in gamelist %}
    Título: <strong>{{game.Título}}</strong> <br>
    Ano: <strong>{{game.Ano}}</strong> <br>
    Categoria: <strong>{{game.Categoria}}</strong> <br><br>
    {% endfor %}
</body>
</html>
```

Adicionando dados em dicionários com Request

Acessando agora a rota “localhost:5000/cadgames”, temos o seguinte resultado:



O último passo agora é configurar a rota para receber os dados que serão enviados do formulário.

Adicionando dados em dicionários com Request

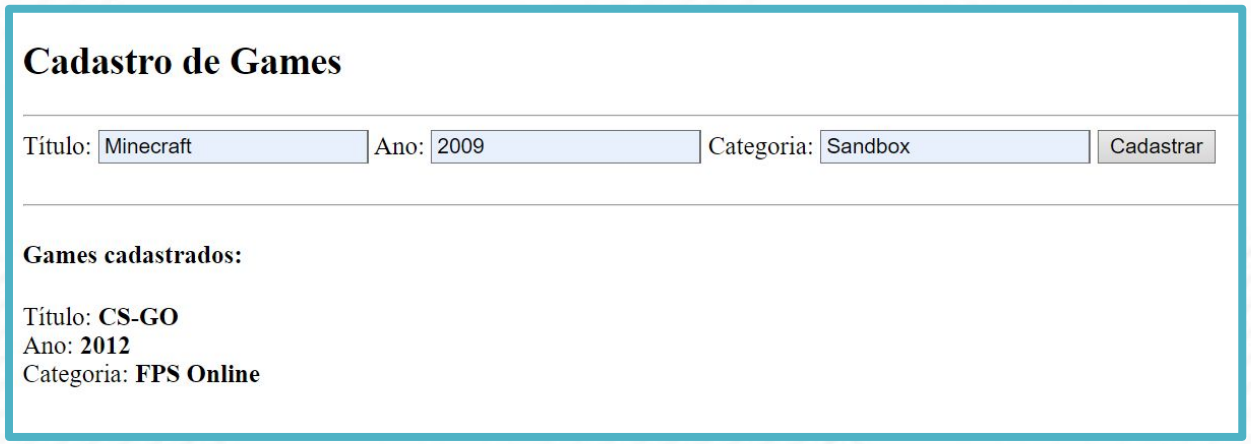
Flask

Para isso, no início da função “cadgames()” incluiremos as condições com as validações e por fim, adicionaremos os dados recebidos ao dicionário “gamelist”, conforme código a seguir:

```
@app.route('/cadgames', methods=['GET', 'POST'])
def cadgames():
    if request.method == 'POST':
        if request.form.get('titulo') and
        request.form.get('ano') and
        request.form.get('categoria'):
            gamelist.append({'Título' :
            request.form.get('titulo'), 'Ano' :
            request.form.get('ano'), 'Categoria' :
            request.form.get('categoria')})

    return render_template('cadgames.html’,gamelist=gamelist)
```

Adicionando dados em dicionários com Request

Feito isso, nossa aplicação já consegue cadastrar os dados de novos games adicionar ao dicionário de games e mostrar na página:



Etec
Registro

python™

Flask

Programação Web III

Prof. Diego Max