

Python para WEB

Yuri Kaszubowski Lopes

Especialização em Tecnologia Python para Negócios

15 de Janeiro 2022

A disciplina

- Framework Django
- Backend com Python + Django
 - ▶ Alguma noção de Frontend (HTML, JQuery(Javascript), BootStrap(CSS))
- BD: SQLite (arquivo)
 - ▶ Facilmente configurável para Postgres, MySQL, ...
- Servidor de desenvolvimento do Django
 - ▶ Em produção: Linux, Nginx/Apache, service, uwsgi

Avaliação

- 40 % PI
- 60 % única atividade:
 - ▶ Feita ao longo da disciplina
 - ▶ **Desenvolver individualmente um sistema WEB com Django para o cadastro de séries e reviews**
 - ▶ Este tema não pode ser usado no PI
 - ▶ Arquivo compactado com no máximo 50 MB:
 - ★ Pasta com até 10 imagens do seu site/sistema
 - ★ Pasta com todo o seu projeto (não incluir ambiente com pacotes Python instalados com pip)
 - ★ incluir arquivo requirements.txt
 - ★ Todos os arquivos estáticos necessários (imagem, css, html, js, ...) já prontos para uso
 - ★ Incluir um pdf de página única descrevendo funcionalidades adicionais
 - ▶ 30% da nota da atividade conforme funcionalidades adicionais descritas no pdf com base em conhecimentos adquiridos em estudo independente

Tópicos da disciplina: Django

Principal

- Instalação e Configuração
- Models (Banco de dados)
- Mapeamento de URL
- Views
- Templates

Úteis

- Django Admin
- Arquivos estáticos
- Bootstrap, JQuery e CSS
- Forms

- Avançados

Model View Template (MVT)

- Um padrão de projetos padrão é Model View Controller (MVC)
 - ▶ O **model** são os dados que são mostrados em uma ou mais **views** e o **controller** faz a mediação entre model e view!
- O Django usa MVT:
 - ▶ Models também são responsável por armazenar dados
 - ▶ A view consulta o model e renderiza ele com um template

Model View Template (MVT)

- Geramente em MVC todos os três componentes devem ser escritos em uma mesma linguagem
- Com MVT o template pode ser em outra linguagem
- No Django:
 - ▶ Models e Views em Python
 - ▶ Templates em HTML (uma versão modificada com códigos)
- Um desenvolvedor frontend especializado pode trabalhar com o template
- Um desenvolvedor Python trabalha com models e views

Projeto e aplicações

- Um projeto em Django possui várias aplicações
- Cada aplicação é um conjunto de funcionalidades contendo principalmente:
 - ▶ URLs
 - ▶ Models
 - ▶ Views
 - ▶ Templates

Instalação: Ambiente Python e pacotes

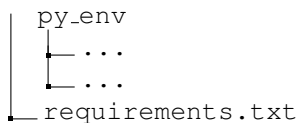
```
$ python -m venv py_env
$ source py_env/bin/activate
$ pip install Django
$ pip install django-autocomplete-light
$ pip install django-crispy-forms
$ pip install -----
$ pip freeze > requirements.txt
```

py_env



Instalação: Criar requirements.txt (opcional)

```
$ pip freeze > requirements.txt
```



- Permite facil setup de colaboradores e deploy na produção.

Instalação: Criar projeto Django

```
$ django-admin startproject server
```

```
py_env
├── ...
├── ...
├── requirements.txt
├── server
│   ├── manage.py
│   └── server
│       ├── asgi.py
│       ├── __init__.py
│       ├── settings.py
│       ├── urls.py
│       └── wsgi.py
```

Instalação: Criar aplicação

```
$ cd server  
$ ./manage.py startapp seriados
```

- Realizar migração
 - ▶ Atualiza banco de dados
 - ▶ Na primeira vez com sqlite cria arquivo `db.sqlite3`

```
$ ./manage.py migrate
```

- Executar servidor em modo desenvolvimento

```
$ ./manage.py runserver
```

Estrutura de diretórios de um projeto

```
├── requirements.txt
└── server
    ├── server # Melhor nome seria config
    │   ├── asgi.py
    │   ├── __init__.py
    │   ├── settings.py
    │   ├── urls.py
    │   └── wsgi.py
    ├── db.sqlite3
    ├── manage.py
    └── seriados
        ├── admin.py
        ├── apps.py
        ├── __init__.py
        ├── migrations
        │   └── __init__.py
        ├── models.py
        ├── tests.py
        └── views.py
```

Nome diretório config (opcional)

```
├── requirements.txt
└── server
    ├── config
    │   ├── asgi.py
    │   ├── __init__.py
    │   ├── settings.py
    │   ├── urls.py
    │   └── wsgi.py
    ├── db.sqlite3
    ├── manage.py
    └── seriados
        ├── admin.py
        ├── apps.py
        ├── __init__.py
        ├── migrations
        │   └── __init__.py
        ├── models.py
        ├── tests.py
        └── views.py
```

Adicione a nova aplicação (app) nas configurações

server/config/settings.py

```
1 ...
2
3 INSTALLED_APPS = [
4     'django.contrib.admin',
5     'django.contrib.auth',
6     'django.contrib.contenttypes',
7     'django.contrib.sessions',
8     'django.contrib.messages',
9     'django.contrib.staticfiles',
10
11     'seriados', # ADICIONE ESSA LINHA
12 ]
13
14 ...
```

URLs

- Crie um arquivo de URL para a aplicação `seriados`
 - ▶ Arquivo `server/seriados/urls.py`

server/seriados/urls.py

```
1 from django.urls import path, re_path, include, register_converter
2
3 app_name = 'seriados'
4
5 urlpatterns = []
```

- Adicione este arquivo no arquivo de urls da configuração do projeto:

server/config/urls.py

```
1 from django.contrib import admin
2 from django.urls import path, include # ADICIONE 'include'
3
4 urlpatterns = [
5     path('admin/', admin.site.urls),
6     path('seriados/', include('seriados.urls')), # ADICIONE ESTA LINHA
7 ]
```

Utilidades

- Setup do ambiente por Colaborador/Deploy

```
$ python -m venv py_env  
$ source py_env/bin/activate  
$ pip install -r requirements.txt  
$ ./manage.py migrate
```

- Habilitando o Ambiente Python
 - ▶ Toda vez que for trabalhar no seu projeto
 - ▶ Se fechar o terminal

```
$ source py_env/bin/activate  
$ ./manage.py runserver
```


Modelos

- Modelos definem dados
- Camada de abstração para um BD SQL
 - ▶ Acesso por um Object Relational Mapper (ORM)
 - ▶ Define tabelas (Models), campos e suas relações com Python
 - ▶ Gera SQL (criação, alteração, consulta) a partir de definições Python
 - ▶ Pode trocar o BD e o código Python é o mesmo
 - ★ Algumas exceções!

Modelos

- Partimos de:

server/seriados/model.py

```
1 from django.db import models
2
3 # Create your models here.
```

Modelos

- Para:

server/seriados/model.py

```
1 from django.db import models
2
3
4 class Serie(models.Model):
5     nome = models.CharField(max_length=70)
6
7     def __str__(self):
8         return self.nome
9
10
11 class Temporada(models.Model):
12     numero = models.IntegerField()
13     serie = models.ForeignKey(Serie, on_delete=models.CASCADE)
14
15
16 class Episodio(models.Model):
17     data = models.DateField()
18     titulo = models.CharField(max_length=200)
19     temporada = models.ForeignKey(Temporada, on_delete=models.CASCADE)
20
21     def __str__(self):
22         return self.titulo
```

Dica para organização de modelos

- Se você tiver vários modelos é possível utilizar vários arquivos para separar os models em grupos
 - 1 Remova o arquivo `model.py` da sua aplicação
 - 2 Crie um diretório (pasta) `model`
 - 3 Dentro de `model` crie os vários arquivos com modelos
 - 4 Dentro de `model` crie um arquivo `__init__.py`
 - 5 Em `__init__.py` importe os modelos dos outros arquivos
 - ★ Você pode usar `from .principal import *`

Instalar modelos

- Execute:

```
$ ./manage.py makemigrations
$ ./manage.py migrate
```

- `makemigrations` cria os arquivos de migração com base nos seus modelos
 - ▶ Estes arquivos gerados automaticamente são parte do seu código fonte. Ao utilizar alguma ferramenta de versionamento de código (e.g. git) eles deverão ser incluídos
- `migrate` aplica as alterações dos arquivos de migração no banco de dados

API para acessar dados

```
$ ./manage.py shell
```

- Você pode agora acessar os modelos

```
1 from seriados.models import Serie, Temporada, Episodio
2
3 Serie.objects.all()
4
5 nova = Serie(nome='Doctor Who')
6 nova.save()
7 print(nova.id)
8 print(nova.nome)
9
10 Serie.objects.all()
```

Interface Admin

- O Django proporciona uma interface de administração pronta (É uma aplicação plugin, o Django chama de **contrib**, são contribuições)
 - ▶ Ela já vem habilitada por padrão!
- Só precisamos definir quais modelos o admin vai usar
 - ▶ Edite o arquivo `admin.py` de cada aplicação

server/seriados/admin.py

```
1 from django.contrib import admin
2
3 from . import models
4
5 # Register your models here.
6
7 admin.site.register(models.Serie)
8 admin.site.register(models.Temporada)
9 admin.site.register(models.Episodio)
```

- Acesse <http://127.0.0.1:8000/admin> no seu navegador

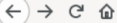
Super usuário para o admin

- Para acessar o **admin** precisamos de um usuário.
- Execute:

```
$ ./manage.py createsuperuser
```

- E siga as instruções

Interface Admin

127.0.0.1:8000/admin/

Django administration

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups	+ Add	✎ Change
Users	+ Add	✎ Change

SERIADOS

Episodios	+ Add	✎ Change
Series	+ Add	✎ Change
Temporadas	+ Add	✎ Change

Recent actions

My actions

None available

Interface Admin

The screenshot shows the Django administration interface in a web browser. The address bar displays the URL `127.0.0.1:8000/admin/seriados/serie/`. The page header is a dark blue bar with the text "Django administration" in yellow and "WELCOME, ADMIN. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)". Below the header is a light blue breadcrumb trail: "Home > Seriados > Series".

On the left is a sidebar menu. It starts with a search bar "Start typing to filter...". Below it are sections: "AUTHENTICATION AND AUTHORIZATION" containing "Groups" and "Users" (each with a green "+ Add" link), and "SERIADOS" containing "Episodios", "Series" (highlighted in yellow), and "Temporadas" (each with a green "+ Add" link). A double-left arrow "«" is at the bottom of the sidebar.

The main content area is titled "Select serie to change" and includes an "ADD SERIE +" button. Below the title is an "Action:" label, a dropdown menu showing "-----", a "Go" button, and the text "0 of 1 selected". A list of items follows: a checkbox next to "SERIE" and a checkbox next to "Doctor Who". Below the list, it says "1 serie".

Interface Admin

← → ↺ 🏠

🔒 ⓘ 127.0.0.1:8000/admin/auth/user/

📄 ... 🏠 ⭐

🔍 📖 👤

☰

Django administration

WELCOME, ADMIN [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home · Authentication and Authorization · Users

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

SERIADOS

Episodios + Add

Series + Add

Temporadas + Add

«

Select user to change

🔍

Search

Action:

⌵

 Go 0 of 1 selected

<input type="checkbox"/>	USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
<input type="checkbox"/>	admin	yurikazuba@gmail.com			🟢

1 user

ADD USER +

FILTER

By staff status

All

Yes

No

By superuser status

All

Yes

No

By active

All

Yes

No

⏪ ⏩ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹ ⏺ ⏻ ⏼ ⏽ ⏾ ⏿ ⏿ 🔍 ↺

View

server/seriados/views.py

```
1 from django.shortcuts import render
2
3 from .models import Serie
4
5 def series_lista(request):
6     lista_series = Serie.objects.all()
7     context = {'lista_series': lista_series}
8     return render(request, 'series_lista.html', context)
```

Template

- Crie o diretório `server/seriados/templates`
- Nesta pasta, crie os arquivos de template
 - ▶ `base.html`: Um esqueleto básico para muitos dos nossos templates
 - ▶ `series_lista.html`: O template usado pela nossa view `series_lista`

Template

server/seriados/templates/base.html

```
{% load static %}
{% load i18n %}

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Python para Negócios</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <p>Bem vindo</p>
    {% block content %}
      <p>Não há nenhum conteúdo aqui!</p>
    {% endblock %}
  </body>
</html>
```

Template

server/seriados/templates/series_lista.html

```
{% extends "base.html" %}

{% block content %}
<h1>Lista de Séries</h1>

{% for series in lista_series %}
    <p>{{ series.nome }}</p>
{% endfor %}
{% endblock %}
```

Maapeamento de URL

- Vamos mapear um endereço URL para nossa view

server/seriados/urls.py

```
1 from django.urls import path, re_path, include, register_converter
2
3 from . import views
4
5 app_name = 'seriados'
6
7 urlpatterns = [
8     path('series/', views.series_lista, name='series_lista'),
9 ]
```

- Acesse <http://127.0.0.1:8000/seriados/series/>

Atividade

- Lembre-se que será um única entrega no final!
- Incremente seu site semanalmente
- **Não deixe acumular**
- O projeto é o mesmo para todos
 - ▶ Tire dúvidas com colegas
 - ▶ **Não compartilhem código**
 - ★ Comandos e questões de instalação sem problemas copiar e colar

Para essa semana

- Instalação do ambiente
- Criação do projeto e app
- Crie todos os modelos na aplicação
- Crie urls, views e templates para listar
 - ▶ Sugestão, um único template para todas as listas
 - ★ Passe detalhes como texto pelo `context` do `render`

Python para WEB

Yuri Kaszubowski Lopes

Especialização em Tecnologia Python para Negócios

15 de Janeiro 2022