# ATIVIDADE

## 1 Criação de um Blog Simples

### Passo 1: Configurar o Ambiente

Primeiro, você precisa garantir que o Django está instalado em seu ambiente de desenvolvimento.

1. Criar um ambiente virtual

   ***python -m venv env***

2. Entrar dentro do ambiente virtual:

   ***env/Scripts/activate***

***3.*** Instalar o Django:

   ***pip install django***

***4.*** Criar um novo projeto Django: Agora, crie um novo projeto Django com o seguinte comando

   ***django-admin startproject meu_blog***
   ***cd meu_blog***

***5.*** Criar um novo aplicativo dentro do projeto: Vamos criar um app chamado blog para organizar nosso código.

   ***python manage.py startapp blog***

***6.*** Registrar o aplicativo no settings.py:

   ***1.*** Va no arquivo meu_blog/settings.py

   ***2.*** Localize a lista chamada INSTALLED_APPS, e adicione o ‘blog’ a ele

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'blog',
]
```

## Passo 2: Criar os Modelos

Agora vamos criar o modelo para as postagens do blog.

LEMBREM DA PESQUISA QUE FIZERAM NA AULA PASSADA PARA SABER O QUE É MODEL.

1. Para criar o MODEL:
    1. Dentro da pasta ‘blog’, abriremo o arquivo models.py
    2. Dentro do arquivo models.py, criaremos uma classe que será nosso modelo
        1. Sim uma classe igual fizeram na lista de exercicio
    3. No exemplo faremos uma classe simples que salvara a postagem

```
from django.db import models

class Postagem(models.Model):
  titulo = models.CharField(max_length=200)
  conteudo = models.TextField()
  data_criacao = models.DateTimeField(auto_now_add=True)

  def __str__(self):
    return self.titulo
```

    1. Reparem que nesse modelo, a classe não tem o __init__(), pois para o django não é necessário, pois ao definir os atributos como mostrado acima o django, criara os métodos necessarios para manipular o model
    2. Nessa classe temos os seguintes atributos:
        1. titulo: um campo de texto para o título da postagem.
        2. conteudo: o conteúdo da postagem.
        3. data_criacao: data e hora de criação da postagem (automático).
    4. Criar a tabela no banco de dados: Após definir o modelo, você precisa criar as migrações e aplicá-las no banco de dados:

***python manage.py makemigrations***
***python manage.py migrate***

## Passo 3: Criar o Administrador para o Blog

Agora, que temos o modelo criado, precisamos registrar o modelo no Django Admin para gerenciar as postagens facilmente.

1. Registrar o modelo no admin:
    1. Dentro da pasta blog, abra o arquivo admin.py, e adicione o seguinte código

```
from .models import Postagem

admin.site.register(Postagem)
```

2. Criar um superusuário:
    1. Isso vocês lembram como faz?
3. Rode o servidor
    1. E isso lembram como realizava?
4. Acessar o Admin:
    1. Va para a pagina http://127.0.0.1:8000/admin/ e faça o login com o superusuario criado anteriormente
    2. Ao fazer isso você vera que tera como gerenciar seu blog e suas postagens

## Passo 4: Criar as Views e Templates

Agora vamos exibir as postagens do blog em uma página. Lembram que quando fizemos nossa primeira pagina, usando um Response, printamos apenas uma mensagem, mas estamos trabalhando com paginas web, então, o ideal é criarmos uma página HTML.

Para isso iremos trabalhar com nossas VIEWs e TEMPLATE ( que foi visto na pesquisa )

1. Criar uma view para listar as postagens
2. Para isso iremos trabalhar com a biblioteca *render* do django
3. Abra dentro da pasta ‘blog’ o arquivo views.py

```
from django.shortcuts import render
from .models import Postagem

def lista_postagens(request):
  postagens = Postagem.objects.all().order_by('-data_criacao')
  return render(request, 'blog/lista_postagens.html', {'postagens': postagens})
```

    1. Nesse código, estamos buscando todas as postagens no banco de dados, ordenadas pela data de criação (mais recentes primeiro), e passando essa lista para o template lista_postagens.html.
4. Criar o template para a lista de postagens:
    1. No diretório *blog*, crie uma pasta chamada *templates*
        1. Dentro dela, crie a pasta *blog*
        2. *Crie dentro desta pasta o arquivo lista_postagens.html*

```
<!DOCTYPE html>
<html>
<head>
 <title>Blog</title>
```

```
</head>
<body>
   <h1>Blog</h1>
</body>
</html>
```

2. Configurar a URL: Agora, crie uma URL para a view.
    1. Crie o arquivo blog/urls.py

```
from django.urls import path
from . import views

urlpatterns = [
 path('', views.lista_postagens, name='lista_postagens'),
]
```

    2. No arquivo meu_blog/urls.py

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
 path('admin/', admin.site.urls),
 path('', include('blog.urls')),
]
```

## Passo 5: Testar

1. Inicie o servidor:
2. Acesse a página inicial: Vá para http://127.0.0.1:8000/ e você verá uma pagina que foi a criada no template.
3. Adicionar postagens via Admin: Acesse o painel de administração em http://127.0.0.1:8000/admin/ e crie novas postagens.

## Passo 6:

Repare que era para ser um blog que liste as postagens, entretanto não esta exibindo nada, para isso como faremos:

1. Como na etapa 4.3 passamos {'postagens': postagens}. Significa que a pagina de postagens já esta recebendo as postagens.
2. Para isso na pagina HTML criada adicionaremos uma lista de postagens, usando as TAGS <ul> e <li> do HTML e um for para percorrer todas as postagens, da seguinte maneira:
    1. 
```
<ul>
 {% for postagem in postagens %}
 <li>
 <h2>{{ postagem.titulo }}</h2>
 <p>{{ postagem.conteudo|slice:":200" }}...</p>
```

```
            <p><a href="#">Leia mais</a></p>
        </li>
    {% endfor %}
</ul>
```

# Sabendo como fazer faça as seguintes atividades:

## 1 To-Do List (Lista de Tarefas)

1. Objetivo: Criar uma aplicação de lista de tarefas simples.
2. Requisitos:
    1. Criar um modelo de tarefa com campos como descrição e status (concluída ou não).
    2. Criar uma página para listar as tarefas.

## 2 Ecommerce

1. Pesquise como adiciona imagem ao MODEL e ao TEMPLATE.
2. Crie uma tela inicial de Ecommerce, onde mostra os produtos com sua respectiva imagem, nome e preço.
3. Neste item fazer um template apresentável incluido o CSS. ( Pesquise como adicionar )

## 3 Melhorando os feitos

1. Adicione no projeto de Postagens, uma nova pagina com o detalhe da postagens, e quando o usuario clicar na postagem, direciona para a postagem ( Pesquisar como incluir tal link )
2. Adicione no Ecommerce a possibilidade de ao clicar direciona para a pagina com o detalhe do produto.

Avisos:

- Nao esquecer do clean code
- Não copiar o ecommerce do semestre passado, pode-se basear nele, mas não copiar ( principalmente a parte do js )