##

# Prática 01 - Criando o ambiente e o projeto

##

# Habilitando execução de scripts no PowerShell do Windows

Set-ExecutionPolicy -Scope CurrentUser Unrestricted

# Verificando a versão do Python

py --version

# Setting up a virtual environment

py -m venv venv

# Ativar o ambiente virtual

venv\Scripts\activate.bat

# Instalar o Django

py -m pip install Django

# Verificando versão do Django

python -m django --version

# Colorindo o terminal

py -m pip install "colorama >= 0.4.6"

#Criando o projeto

django-admin startproject mystackoverflow

# Entrando no diretório do projeto

cd mystackoverflow

# Executando o servidor pra ver se está funcionando

python manage.py runserver

# Acessar a url http://127.0.0.1:8000/

# Criando a app para fórum

python manage.py startapp forum

# Abrir arquivo settings.py e configurar

# LANGUAGE\_CODE = 'pt-BR'

# TIME\_ZONE = 'America/Recife'

##

# Prática 02 - Criando os modelos

##

# Executando o primeiro migrate para configurar as tabelas do Django

python manage.py migrate

# Criando os modelos

# No arquivo models.py

class Question(models.Model):

title = models.CharField(max\_length=200, null=False)

details = models.TextField(null=False)

have\_tried = models.TextField()

created\_date = models.DateTimeField("Created on")

# Adicionando o modelo ao projeto

# No arquivo settings.py

INSTALLED\_APPS = [

'forum.apps.ForumConfig',

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

]

# Criando as tabelas do Forum

python manage.py makemigrations forum

# Olhando o SQL gerado

python manage.py sqlmigrate forum 0001

# Atualizando o banco de dados com as mudanças.

python manage.py migrate

# Abrindo o shell do Django e manipulando os objetos.

python manage.py shell

# Importar as classes do modelo

from forum.models import Question

#Listar todas as perguntas

Question.objects.all()

# Criando uma pergunta

# Importando timezone

from django.utils import timezone

q = Question(title="Como ensinar Django?", details="Preciso ensinar Django para o segundo período mas nunca fiz isso direito.", have\_tried="Já tentei aula teórica com slides, mas 2h pareceram pouco.", created\_date=timezone.now())

# Verificando o id

q.id

# Verificando o Título

q.title

# Salvando a pergunta

q.save()

# Verificando o id novamente

q.id

# Verificando os demais atributos

q.details

q.have\_tried

q.created\_date

# Alterando o valor de um atributos e salvando

q.have\_tried="Já tentei aula teórica com slides, mas 2h pareceram pouco. Aparentemente eles precisam de mais tempo com exercícios práticos."

q.save()

# Consultando todas as perguntas da base.

Question.objects.all()

# Adicionando uma representação do objeto e um método ao modelo

# Em models.py

import datetime

from django.utils import timezone

# Na classe Question adicionar os métodos

def \_\_str\_\_(self):

return "[" + str(self.id) + "] " + self.title

def was\_published\_recently(self):

return self.created\_date >= timezone.now() - datetime.timedelta(days=1)

# Inicia novo shell

python manage.py shell

from forum.models import Question

# Listando todos os objetos.

Question.objects.all()

# Obtendo um objeto específico

Question.objects.filter(id=1)

# Filtrando por título

Question.objects.filter(title\_\_startswith="Como")

# Obtendo as perguntas desse ano

from django.utils import timezone

current\_year = timezone.now().year

Question.objects.get(created\_date\_\_year=current\_year)

# Obtendo a pergunta com id=2. Usando o get, apresenta erro (ele espera que o objeto exista). Usando filter retorna a coleção vazia.

Question.objects.get(id=2)

Question.objects.filter(id=2)

#Obtendo a primeira pergunta

q = Question.objects.get(id=1)

# Adicionando dados do usuário

username = models.CharField(max\_length=200, null=False)

# Criando o migrations

python manage.py makemigrations

# Corrigindo o modelo

username = models.CharField(max\_length=200, null=False, default="anonymous")

# Criando o migrations novamente

python manage.py makemigrations

# Aplication o migrations

python manage.py migrate

# Inicia novo shell

python manage.py shell

from forum.models import Question

# Listando todos os objetos.

Question.objects.all()

#Obtendo a primeira pergunta

q = Question.objects.get(id=1)

# Consultando o usuario

q.username

##

# Prática 03 - Adicionando relacionamentos

##

#Criando a classe Answer

# Arquivo models.py

class Answer(models.Model):

question = models.ForeignKey(Question, on\_delete=models.CASCADE)

text = models.TextField(null=False)

votes = models.IntegerField(default=0)

created\_date = models.DateTimeField("Created on")

username = models.CharField(max\_length=200, null=False, default="anonymous")

def \_\_str\_\_(self):

return "[" + str(self.id) + "] " + self.text

def was\_published\_recently(self):

return self.created\_date >= timezone.now() - datetime.timedelta(days=1)

# Criando o migrations

python manage.py makemigrations

# Aplication o migrations

python manage.py migrate

# Inicia novo shell

python manage.py shell

from forum.models import Answer, Question

#Obtendo a primeira pergunta

q = Question.objects.get(id=1)

# Consultando as respostas da Pergunta

q.answer\_set.all()

from django.utils import timezone

#Criando uma resposta

a = Answer(question=q, text="Te vira", created\_date=timezone.now())

#Salvando a resposta

a.save()

#Verificando a pergunta

a.question

# Verificando as respostas

q.answer\_set.all()

#Criando direto na coleção

q.answer\_set.create(text="Te vira 2", created\_date=timezone.now())

# Recuperando objeto recém-criado

a2 = Answer.objects.get(id=2)

a2.question

# Criando uma terceira resposta

a3 = q.answer\_set.create(text="Te vira 3", created\_date=timezone.now())

# Verificando o id

a3.id

#Consultando todas as respostas

q.answer\_set.all()

# Excluindo o objeto

a3.delete()

#Verificando novamente todas as respostas

q.answer\_set.all()

Contando as respostas

q.answer\_set.count()

# Alterando a quantidade de votos

a2.votes = a2.votes + 1

# Salvando no banco

a2.save()

##

# Prática 04 - Interface de Administração

##

# Criando superuser

python manage.py createsuperuser

# username

stackadmin

# e-mail

rac2@cesar.school

# senha

123456

# repete a senha

123456

# bypass validation

y

# Executar o servidor

python manage.py runserver

# Acessar o endereço

http://127.0.0.1:8000/admin/

Usuario: stackadmin

Senha: 123456

# O model de fórum não está disponível. Vamos adicioná-lo.

# Em forum/admin.py adicionar as linhas abaixo

from django.contrib import admin

from .models import Question

admin.site.register(Question)

# Voltando no navegador e dando F5 a opção vai aparecer.

# Clicar em "Questions" levará a tela com todas as perguntas.

# Clicando numa pergunta específica eu entro na tela de edição.

# Após editar eu posso salvar a pergunta.

# Posso criar uma nova pergunta através da interface

# Posso excluir a pergunta criada.

# Agora vamos adicionar a opção de editar as respostas.

# Em forum/admin.py adicionar as linhas abaixo

from .models import Question, Answer

admin.site.register(Question)

admin.site.register(Answer)

# Voltando no navegador e dando F5 a opção vai aparecer.

# Clicar em "Answers" levará a tela com todas as respostas.

# Clicando numa resposta específica eu entro na tela de edição.

# Após editar eu posso salvar a resposta.

# Posso criar uma nova resposta através da interface, selecionando uma pergunta já existente.

# Posso excluir a resposta criada.

##

# Prática 05 - Criando as primeiras views

##

# No arquivo forum/views.py

from django.http import HttpResponse

from django.shortcuts import render

def index(request):

return HttpResponse("You're looking at all questions.")

def detail(request, question\_id):

return HttpResponse("You're looking at question %s." % question\_id)

def vote(request, answer\_id):

return HttpResponse("You're voting on answer %s." % answer\_id)

# Crie o arquivo urls.py dentro do diretório forum e cole o conteúdo abaixo.

from django.urls import path

from . import views

app\_name = 'forum'

urlpatterns = [

# ex: /forum/

path("", views.index, name="index"),

# ex: /forum/5/

path("<int:question\_id>/", views.detail, name="detail"),

# ex: /forum/5/vote/

path("<int:answer\_id>/vote/", views.vote, name="vote"),

]

# Edite o urls.py do projeto adicionando as linhas abaixo

from django.urls import path, include

path('admin/', admin.site.urls),

path('forum/', include('forum.urls')),

# Agora abrimos o navegador nas URLS http://127.0.0.1:8000/forum/, http://127.0.0.1:8000/forum/5/ e http://127.0.0.1:8000/forum/5/vote/ e verificamos as respostas.

# Agora vamos alterar a view para que ela faça alguma coisa.

# No arquivo views.py editar a view index()

from .models import Question

def index(request):

latest\_question\_list = Question.objects.order\_by("-created\_date")

output = ", ".join([str(q) for q in latest\_question\_list])

return HttpResponse(output)

# Acessa a URL http://127.0.0.1:8000/forum/

# Agora vamos modificar as outras duas views

# No arquivo models.py, na classe Question

def detail\_string(self):

return "id: " + str(self.id) + "; title: " + self.title + "; details: " + self.details + "; have\_tried: " + self.have\_tried + "; created\_date: " + str(self.created\_date) + "; username: " + self.username

# No arquivo views.py

from .models import Question, Answer

def detail(request, question\_id):

try:

question = Question.objects.get(pk=question\_id)

except Question.DoesNotExist:

raise Http404("Question does not exist")

return HttpResponse(question.detail\_string() + "; answers: {" + ", ".join([str(a) for a in question.answer\_set.all()]) + "}")

def vote(request, answer\_id):

try:

answer = Answer.objects.get(pk=answer\_id)

except Answer.DoesNotExist:

raise Http404("Answer does not exist")

return HttpResponse(str(answer) + "; votes: " + str(answer.votes))

# Acessa a URL http://127.0.0.1:8000/forum/1/

# Acessa a URL http://127.0.0.1:8000/forum/5/

# Acessa a URL http://127.0.0.1:8000/forum/1/vote/

# Agora vamos organizar nossas views em ClassViews

# No arquivo views.py no lugar do método view()

from django.views import View

class MainView(View):

def get(self, request):

latest\_question\_list = Question.objects.order\_by("-created\_date")

output = ", ".join([str(q) for q in latest\_question\_list])

return HttpResponse(output)

class QuestionView(View):

def get(self, request, question\_id):

try:

question = Question.objects.get(pk=question\_id)

except Question.DoesNotExist:

raise Http404("Question does not exist")

return HttpResponse(question.detail\_string() + "; answers: {" + ", ".join([str(a) for a in question.answer\_set.all()]) + "}")

class VoteView(View):

def get(self, request, answer\_id):

try:

answer = Answer.objects.get(pk=answer\_id)

except Answer.DoesNotExist:

raise Http404("Answer does not exist")

return HttpResponse(str(answer) + "; votes: " + str(answer.votes))

# No arquivo urls.py

# ex: /forum/

path("", views.MainView.as\_view(), name="index"),

# ex: /forum/5/

path("<int:question\_id>/", views.QuestionView.as\_view(), name="detail"),

# ex: /forum/5/vote/

path("<int:answer\_id>/vote/", views.VoteView.as\_view(), name="vote"),

##

# Prática 06 - Criando os templates

##

# Criar o diretório "templates\forum" dentro da app "forum"

# Criar o arquivo "base.html" dentro do diretório "templates\forum"

<html>

<head>

<title>Simple StackOverflow</title>

</head>

<body>

{% block content %}

{% endblock %}

</body>

</html>

# Criar o arquivo index.html dentro do diretório "templates\forum"

{% extends "forum/base.html" %}

{% block content %}

{% if questions %}

<h1>Últimas perguntas</h1>

<ul>

{% for question in questions %}

<li><a href="{% url 'forum:detail' question.id %}">{{ question.title }}</a></li>

{% endfor %}

</ul>

{% else %}

<p>Nenhuma pergunta disponível.</p>

{% endif %}

{% endblock %}

# Alterar em views.py

class MainView(View):

def get(self, request):

latest\_question\_list = Question.objects.order\_by("-created\_date")

context = {'questions' : latest\_question\_list}

return render(request, 'forum/index.html', context)

# Acessa a URL http://127.0.0.1:8000/forum/

# Agora vamos criar a view para os detalhes da pergunta

# Criar o arquivo detail.html dentro do diretório "templates\forum"

{% extends "forum/base.html" %}

{% block content %}

<h1>{{ question.title }}</h1>

<p>{{ question.details }}</p>

<p>{{ question.have\_tried }}</p>

<p>{{ question.created\_date | date:"d M Y" }}</p>

<ul>

{% for answer in question.answer\_set.all %}

<li>{{ answer.text }}: {{ answer.votes }} votes</li>

{% endfor %}

</ul>

{% endblock %}

# Alterar em views.py

class QuestionView(View):

def get(self, request, question\_id):

try:

question = Question.objects.get(pk=question\_id)

except Question.DoesNotExist:

raise Http404("Question does not exist")

context = {'question' : question}

return render(request, 'forum/detail.html', context)

# Acessa a URL http://127.0.0.1:8000/forum/ e navegar para os detalhes da pergunta

# Aparentemente não precisamos de uma visão para visualizar os votos de cada resposta, pois eles já aparecem nos detalhes.

##

# Prática 07 - Criando os templates com formulários

##

# Para esta prática criaremos as funcionalidades de criar uma pergunta, criar uma resposta e votar numa resposta. Vamos fazer todo esse desenvolvimento com o servidor do django em execução.

# No terminal digite o comando abaixo.

python manage.py runserver

# O primeiro passo será criar a página HTML para criação de uma pergunta. Para isso crie no diretório "mystackoverflow\templates\forum" o arquivo "create\_question.html" e cole o conteúdo abaixo

{% extends "forum/base.html" %}

{% block content %}

<form method="post">

{% csrf\_token %}

<p><label for="title">Escreva sua pergunta:</label>

<input type="text" name="title" id="title" size="50" maxlength="200">

</p>

<p><label for="details">Detalhe o problema:<br />

<textarea rows="10" cols="40" id="details" name="details"></textarea>

</p>

<p><label for="have\_tried">Escreva o que você já tentou:<br />

<textarea rows="10" cols="40" id="have\_tried" name="have\_tried"></textarea>

</p>

<input type="submit" name="dopost" value="Salvar" />

</form>

{% endblock %}

# Agora vamos criar a view no arquivo "views.py".

class CreateQuestionView(View):

def get(self, request):

return render(request, 'forum/create\_question.html')

# Vamos adicionar a URL dessa view no arquivo "forum\urls.py"

app\_name = 'forum'

urlpatterns = [

# ex: /forum/

path("", views.MainView.as\_view(), name="index"),

# ex: /forum/5/

path("<int:question\_id>/", views.QuestionView.as\_view(), name="detail"),

# ex: /forum/5/vote/

path("<int:answer\_id>/vote/", views.VoteView.as\_view(), name="vote"),

# ex: /forum/create/

path("create/", views.CreateQuestionView.as\_view(), name="create\_question"),

# Acesse a URL http://127.0.0.1:8000/forum/create/

# Agora precisamos escrever o método 'post' da 'CreateQuestionView' para que possamos receber os dados do formulário. No arquivo "views.py", na classe 'CreateQuestionView' adicione o método abaixo.

def post(self, request):

if request.user.is\_authenticated:

username = request.user.username

else:

username = 'anonymous'

title = request.POST.get('title')

details = request.POST.get('details')

have\_tried = request.POST.get('have\_tried')

created\_date = timezone.now()

question = Question(title=title, details=details, have\_tried=have\_tried, created\_date=created\_date, username=username)

question.save()

return redirect(reverse('forum:detail', args=[question.id]))

# Volte para o navegador, preencha os dados da pergunta e submeta. A pergunta deve aparecer na lista de perguntas da aplicação.

# Agora vamos adicionar na página inicial a opção de criar uma nova pergunta. No arquivo "index.html", na linha 14 (após o {% endif %} e antes do {% endblock %}) adicione a linha abaixo.

<a href="{% url 'forum:create\_question' %}">Nova pergunta</a>

# Acesse a URL http://127.0.0.1:8000/forum/ e clique na nova opção "Nova pergunta".

# Agora vamos criar a página HTML para criação de uma resposta. Para isso vamos primeiro adaptar o template de detalhar uma pergunta e permitir seu reuso, uma vez que seria interessante para o usuário ver os detalhes da pergunta enquanto está escrevendo sua resposta.

# No arquivo "templates\forum\detail.html" faça a mudança abaixo.

...

<p>Criado por {{ question.username }}</p>

{% block answers %}

<ul>

{% for answer in question.answer\_set.all %}

<li>{{ answer.text }}: <b>{{ answer.votes }} votes</b></li>

{% endfor %}

</ul>

{% endblock %}

{% endblock %}

# Agora vamos criar no diretório "mystackoverflow\templates\forum" o arquivo "create\_answer.html" e colar o conteúdo abaixo. Atenção para o fato de estarmos reusando o template "detail.html" e sobrescrevendo apenas o bloco "answers" substituindo as respostas por um formulário que recebe uma resposta.

{% extends "forum/detail.html" %}

{% block answers %}

<h1>Nova resposta</h1>

<form method="post">

{% csrf\_token %}

<p><label for="text">Escreva sua Resposta:</label><br />

<textarea rows="10" cols="40" id="text" name="text"></textarea>

</p>

<input type="submit" name="dopost" value="Salvar" />

</form>

{% endblock %}

# Agora vamos criar a view no arquivo "views.py". O método 'get' tem que recuperar a 'question' que será respondida.

class CreateAnswerView(View):

def get(self, request, question\_id):

try:

question = Question.objects.get(pk=question\_id)

except Question.DoesNotExist:

raise Http404("Question does not exist")

context = {'question' : question}

return render(request, 'forum/create\_answer.html', context)

# Vamos adicionar a URL dessa view no arquivo "forum\urls.py"

app\_name = 'forum'

urlpatterns = [

# ex: /forum/

path("", views.MainView.as\_view(), name="index"),

# ex: /forum/5/

path("<int:question\_id>/", views.QuestionView.as\_view(), name="detail"),

# ex: /forum/5/vote/

path("<int:answer\_id>/vote/", views.VoteView.as\_view(), name="vote"),

# ex: /forum/create/

path("create/", views.CreateQuestionView.as\_view(), name="create\_question"),

# ex: /forum/5/answer/

path("<int:question\_id>/answer/", views.CreateAnswerView.as\_view(), name="create\_answer"),

# Acesse a URL http://127.0.0.1:8000/forum/1/answer/

# Agora precisamos escrever o método 'post' da 'CreateAnswerView' para que possamos receber os dados do formulário. No arquivo "views.py", na classe 'CreateAnswerView' adicione o método abaixo. Nesse método recuperamos a question da URL, criamos uma nova "answer" para ela e em seguida redirecionamos o usuário para os detalhes da 'question'.

def post(self, request, question\_id):

try:

question = Question.objects.get(pk=question\_id)

except Question.DoesNotExist:

raise Http404("Question does not exist")

if request.user.is\_authenticated:

username = request.user.username

else:

username = 'anonymous'

text = request.POST.get('text')

created\_date = timezone.now()

question.answer\_set.create(text=text, created\_date=created\_date, username=username)

return redirect(reverse('forum:detail', args=[question.id]))

# Volte para o navegador, preencha os dados da resposta e submeta. A resposta deve aparecer na lista de respostas da pergunta.

# Agora vamos adicionar na página de detalhes da pergunta a opção de criar uma nova resposta. No arquivo "detail.html", na linha 15 (antes do {% endblock %}) adicione a linha abaixo.

<a href="{% url 'forum:create\_answer' question.id %}">Nova resposta</a>

# Acesse a URL http://127.0.0.1:8000/forum/, clique numa pergunta e veja a opção de adicionar nova resposta.

# Agora vamos adicionar a opção de votar numa resposta. Já vimos que essa funcionalidade não precisa de uma tela específica. Nesse caso, começaremos criando a view que receberá o voto.

# No arquivo "views.py" altere o código da 'VoteView' para o código abaixo.

class VoteView(View):

def post(self, request, answer\_id):

try:

answer = Answer.objects.get(pk=answer\_id)

except Answer.DoesNotExist:

raise Http404("Answer does not exist")

answer.votes += 1

answer.save()

return redirect(reverse('forum:detail', args=[answer.question.id]))

# Veja que não precisaremos do método 'get', pois a única ação da view será alterar a quantidade de votos de uma pergunta.

# Como essa view já está mapeada no 'forum\urls.py', podemos adicionar o botão para votar na resposta. Para isso abra o template "detail.html" e altere a linha 12, que passará a ficar assim.

<li>{{ answer.text }}: <b>{{ answer.votes }} votes</b> <form action="{% url 'forum:vote' answer.id %}" method="post">{% csrf\_token %}<input type="submit" name="dopost" value="Votar" /></form></li>

# Acesse a URL http://127.0.0.1:8000/forum/, clique numa pergunta que já tenha respostas e veja a opção de votar numa resposta.

# Agora vamos adicionar uma barra de menu para nossa aplicação. Abra o template 'base.html' e coloque o código abaixo logo após a linha 5 (<body>).

<span><a href="{% url 'forum:index' %}">Home</a> | <a href="{% url 'forum:create\_question' %}">Criar pergunta</a></span>

# Abra o navegador na URL http://127.0.0.1:8000/forum/ e note que todas as página possuem o menu que acabamos de criar.