**Папка core, файл urls.py**

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

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

path('', include('work24.urls')),

]

* **path('admin/', admin.site.urls)**: добавляет URL маршрут для административной панели Django, который доступен по пути admin/.
* **path('', include('work24.urls'))**: перенаправляет корневой URL ('') к URL-ам, определенным в work24.urls.

**Папка work24, файл urls.py**

from django.urls import path

from . import views

urlpatterns = [

path("", views.HomeView.as\_view(), name="homepage"),

path("search/", views.PostSearchView.as\_view(), name="post\_search"),

path("<slug:post>/", views.post\_single, name="post\_single"),

]

* **path("", views.HomeView.as\_view(), name="homepage")**: определяет маршрут для главной страницы, которая отображается с использованием класса HomeView.
* **path("search/", views.PostSearchView.as\_view(), name="post\_search")**: определяет маршрут для страницы поиска, отображаемой с использованием класса PostSearchView.
* **path("<slug:post>/", views.post\_single, name="post\_single")**: определяет маршрут для отображения отдельного поста, передавая slug поста в функцию post\_single.

**Папка work24, файл views.py**

from django.shortcuts import get\_object\_or\_404, render

from django.views.generic import ListView

from .forms import PostSearchForm

from .models import Post

class HomeView(ListView):

model = Post

context\_object\_name = "posts"

paginate\_by = 10

def get\_template\_names(self):

if self.request.htmx:

return "work24/components/post-list-elements.html"

return "work24/index.html"

def post\_single(request, post):

post = get\_object\_or\_404(Post, slug=post, status="published")

related = Post.objects.filter(author=post.author)[:5]

return render(

request,

"work24/components/single-post-elements.html",

{"post": post, "related": related},

)

class PostSearchView(HomeView):

form\_class = PostSearchForm

def get\_queryset(self):

form = self.form\_class(self.request.GET)

if form.is\_valid():

return Post.objects.filter(title\_\_icontains=form.cleaned\_data["q"])

def get\_template\_names(self):

if self.request.htmx:

return "work24/components/search-list-elements.html"

return "work24/search.html"

* **HomeView**: класс представления, наследуемый от ListView, отображающий список постов. Использует пагинацию по 10 элементов на страницу. Шаблон зависит от того, использует ли запрос HTMX.
* **post\_single(request, post)**: функция представления, отображающая один пост. Получает пост по slug и статусу "published", а также 5 связанных постов от того же автора. Отображает шаблон single-post-elements.html.
* **PostSearchView**: класс представления для поиска постов. Наследует HomeView, использует форму PostSearchForm. Определяет get\_queryset для фильтрации постов по заголовку, содержащему поисковый запрос. Использует разные шаблоны в зависимости от HTMX.

**Папка work24, файл models.py**

from django.db import models

from django.contrib.auth.models import User

from django.urls import reverse

class Post(models.Model):

options = (("draft", "Draft"), ("published", "published"))

title = models.CharField(max\_length=200)

subtitle = models.CharField(max\_length=100)

slug = models.SlugField(max\_length=250, unique=True)

author = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name="post\_author")

content = models.TextField()

created\_at = models.DateTimeField(auto\_now\_add=True, editable=False)

updated\_at = models.DateTimeField(auto\_now=True)

status = models.CharField(max\_length=10, choices=options, default="draft")

class Meta:

ordering = ("-created\_at",)

def get\_absolute\_url(self):

return reverse("post\_single", args=[self.slug])

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

return self.title

* **Post**: модель, представляющая блог-пост. Имеет поля для заголовка, подзаголовка, URL-slug, автора, содержимого, даты создания и обновления, а также статус публикации. Метод get\_absolute\_url возвращает URL для конкретного поста.

**Папка work24, файл forms.py**

from django import forms

class PostSearchForm(forms.Form):

q = forms.CharField()

* **PostSearchForm**: форма поиска, содержащая одно текстовое поле q для ввода поискового запроса.

**Папка work24, файл factory.py**

import factory

from factory.faker import faker

from django.contrib.auth.models import User

from .models import Post

FAKE = faker.Faker()

class PostFactory(factory.django.DjangoModelFactory):

class Meta:

model = Post

title = factory.Faker("sentence", nb\_words=12)

subtitle = factory.Faker("sentence", nb\_words=12)

slug = factory.Faker("slug")

author = User.objects.get\_or\_create(username="admin")[0]

@factory.lazy\_attribute

def content(self):

x = ""

for \_ in range(0, 5):

x += "\n" + FAKE.paragraph(nb\_sentences=30) + "\n"

return x

status = "published"

* **PostFactory**: фабрика для создания объектов модели Post с использованием библиотеки factory\_boy и faker. Генерирует случайные данные для заголовка, подзаголовка, slug и содержимого. Автором устанавливается пользователь с именем "admin". Статус по умолчанию - "published".