Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Цифровая кафедра

WEB-Разработка

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ПРОЕКТНОЙ

РАБОТЕ НА ТЕМУ:

«WEB-приложение»

Студент	ИБМ3-62Б	22.06.2023	Б.А. Эдаев
	(Группа)	(Дата)	(И.О. Фамилия)

Москва

2023 г.

СОДЕРЖАНИЕ

Введение 3
1. Приложение «website» 4
2. Приложение «main» 6
3. Приложение «news» 7
4. Приложение «registration» 10
Заключение 13

Введение

Целью проекта является создание функционального и привлекательного новостного веб-сайта на основе Django, который позволит пользователям получать информацию о новостях МГТУ им. Баумана и взаимодействовать с контентом путем добавления, редактирования и удаления новостей.

Проект представляет собой новостной веб-сайт, разработанный с использованием фреймворка Django. Он включает 4 основных приложения: «main», «news», «registration», «website». Приложение main служит главной точкой входа в веб-сайт. Здесь мы предоставляем пользователю информацию о сайте, его функциональности и основные разделы, которые они могут исследовать.

Приложение новостей позволяет нам добавлять, редактировать и удалять новости, а также отображать их на веб-сайте. Каждая новость содержит заголовок, текст, изображение и дату публикации. Пользователи имеют возможность просматривать список новостей, открывать отдельные новости для получения подробной информации.

Приложение регистрации/авторизации пользователей обеспечивает функционал для регистрации новых пользователей и их аутентификации на веб-сайте. Пользователи могут создавать учетные записи, входить в систему с помощью своих учетных данных и получать доступ к дополнительным функциям, таким как редактирование, удаление и добавление новостей.

1. Приложение «website»

Структура всех приложений стандартная от фреймворка Django. Рассмотрим основные составляющие приложения «website».

Первое – файл urls.py – в котором определяются все основные URL маршруты нашего приложения. В начале приложения импортируются необходимые модули. Затем определяются маршруты URL с помощью urlpatterns. В нашем случае они выглядят так:

- path("admin/", admin.site.urls) определяет URL-маршрут для административного интерфейса Django. Все URL, начинающиеся с "admin/", будут обрабатываться модулем admin.site.urls.
- path("", include("main.urls")) включает маршруты из файла urls.py приложения "main".
- path("news/", include("news.urls")) включает маршруты из файла urls.py приложения "news". Все URL, начинающиеся с "news/", будут переданы в обработку модулю news.urls.
- path("registration/", include("registration.urls")) включает маршруты из файла urls.py приложения "registration". Все URL, начинающиеся с "registration/", будут переданы в обработку модулю registration.urls.

В конце списка urlpatterns добавляется маршрут для обработки статических файлов, чтобы они могли быть обслужены во время разработки. Также устанавливается обработчик ошибки 404 (handler404), который указывает на функцию pageNotFound из main.views для отображения пользовательской страницы при ошибке 404. Код приведен на рисунке 1 (см. рис. 1).

```python
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
from main.views import pageNotFound

urlpatterns = [
    path("admin/", admin.site.urls),
    path("", include("main.urls")),
    path("news/", include("news.urls")),
    path("registration/", include("registration.urls")),
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)

handler404 = pageNotFound
```

Второе - файл settings.py содержит основные настройки и конфигурации для Django-приложения:

- Конфигурация базы данных: в данном примере используется база данных SQLite. Она настроена в переменной DATABASES, где указывается путь к файлу базы данных db.sqlite3.

- Конфигурация приложений: в переменной INSTALLED_APPS определены все установленные приложения. В данном случае добавлены приложения "main", "news", "registration", а также стандартные приложения Django, такие как "django.contrib.admin", "django.contrib.auth", "django.contrib.sessions" и другие.

- Конфигурация шаблонов: в переменной TEMPLATES определены настройки для работы с шаблонами. Указан путь к каталогу шаблонов в переменной DIRS, где шаблоны будут искаться в подкаталоге templates приложения.

- Конфигурация статических файлов: в переменной STATIC_URL указывается URL для доступа к статическим файлам, таким как CSS.

2. Приложение «main»

Приложение "main" представляет собой основную часть веб-сайта и содержит представления, URL-пути и конфигурацию приложения. В представлениях приложения "main" определены следующие функции:

- pageNotFound(request, exception): Обрабатывает ошибку страницы, которая не найдена (404) и отображает соответствующий шаблон "main/404.html".
- index(request): Отображает главную страницу. Возвращает шаблон "main/index.html" и передает данные, включающие заголовок "Главная страница" и список значений.
- about(request): Отображает страницу "О нас". Возвращает шаблон "main/about.html".
- contacts(request): Отображает страницу с контактной информацией. Возвращает шаблон "main/contacts.html".

В файле urls.py приложения "main" определены следующие URL-пути:

1. Пустой путь "" соответствует главной странице и связан с представлением index.
2. Путь "about" соответствует странице "О нас" и связан с представлением about.
3. Путь "contacts" соответствует странице с контактами и связан с представлением contacts.

В приложении "main" содержится папка "templates/main/", где хранятся шаблоны HTML для отображения страниц.

3. Приложение «news»

Приложение "news" отвечает за управление новостями, включая добавление, удаление и редактирование новостей, а также предоставление соответствующих шаблонов для отображения новостей.

- URL-пути:

Файл urls.py содержит определения URL-путей, которые связывают URL с соответствующими представлениями (views):

1. "" - путь к домашней странице новостей. Он обрабатывается функцией news_home из views.py.
2. "create" - путь для создания новости. Он обрабатывается функцией create из views.py.
3. "<int:pk>" - путь для просмотра отдельной новости с определенным идентификатором (pk). Он обрабатывается классом NewsDetailView из views.py.
4. "<int:pk>/update" - путь для редактирования новости с определенным идентификатором (pk). Он обрабатывается классом NewsUpdateView из views.py.
5. "<int:pk>/delete" - путь для удаления новости с определенным идентификатором (pk). Он обрабатывается классом NewsDeleteView из views.py.

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

urlpatterns = [
  path("", views.news_home, name="news_home"),
  path("create", views.create, name="create"),
  path("<int:pk>", views.NewsDetailView.as_view(), name="news-detail"),
  path("<int:pk>/update", views.NewsUpdateView.as_view(), name="news-update"),
  path("<int:pk>/delete", views.NewsDeleteView.as_view(), name="news-delete"),
]
```

- Представления (views):

В файле views.py определены функции и классы-представления, которые обрабатывают запросы от пользователей и взаимодействуют с моделями и формами для выполнения соответствующих действий.

Функция news_home обрабатывает запрос для отображения списка новостей. Она извлекает все новости из модели Articles и передает их в шаблон news/news_home.html в виде переменной news.

Класс NewsDetailView является представлением для отображения отдельной новости. Он использует модель Articles, шаблон news/details_view.html и предоставляет объект новости в качестве переменной article.

Класс NewsUpdateView представляет представление для редактирования новости. Он использует модель Articles, форму ArticlesForm, шаблон news/create.html и требует аутентификации пользователя (LoginRequiredMixin) для доступа. При успешном редактировании новости, пользователь будет перенаправлен на страницу просмотра отредактированной новости.

Класс NewsDeleteView является представлением для удаления новости. Он использует модель Articles, шаблон news/news-delete.html и также требует аутентификации пользователя. После успешного удаления новости, пользователь будет перенаправлен на страницу списка новостей.

- Модели:

В файле models.py определена модель Articles, которая представляет новости. Модель содержит поля, такие как title, anons, full_text и date, которые хранят информацию о заголовке новости, анонсе, содержании и дате публикации соответственно.

Модель также определяет методы __str__, возвращающий заголовок новости, и get_absolute_url, возвращающий URL-адрес для просмотра отдельной новости.

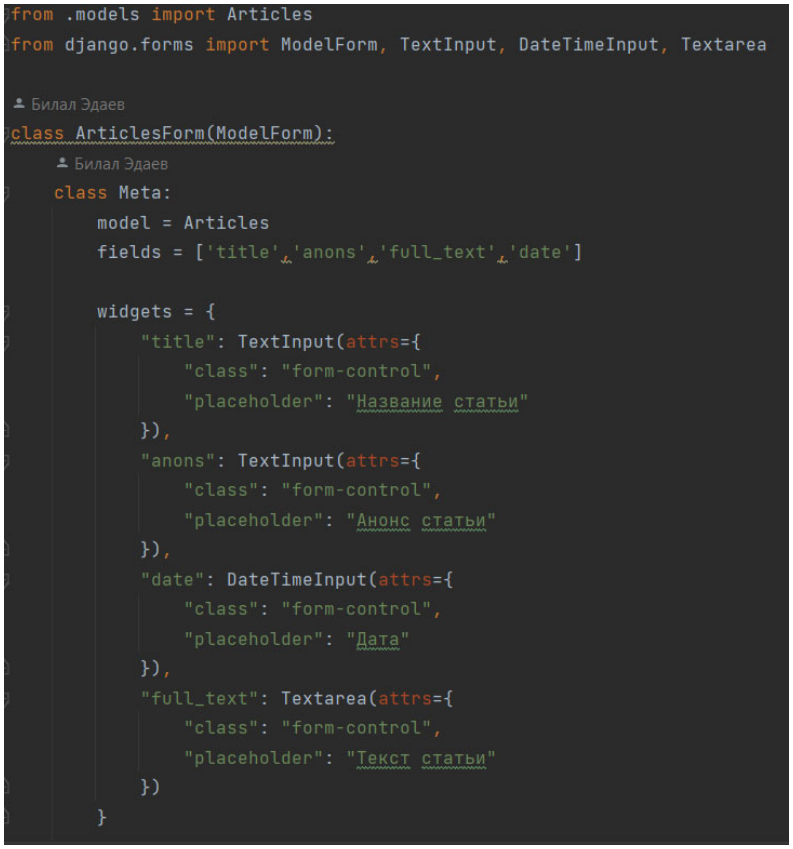
- Формы:

В файле forms.py определена форма ArticlesForm для создания и

редактирования новостей.

Форма ArticlesForm основана на модели Articles и содержит поля title, anons, full_text и date.

Для каждого поля формы определены виджеты, которые определяют, как поле будет отображаться на странице, используя атрибуты attrs.



```
from .models import Articles
from django.forms import ModelForm, TextInput, DateTimeInput, Textarea

Билал Эдаев
class ArticlesForm(ModelForm):
    Билал Эдаев
    class Meta:
        model = Articles
        fields = ['title','anons','full_text','date']

        widgets = {
            "title": TextInput(attrs={
                "class": "form-control",
                "placeholder": "Название статьи"
            }),
            "anons": TextInput(attrs={
                "class": "form-control",
                "placeholder": "Анонс статьи"
            }),
            "date": DateTimeInput(attrs={
                "class": "form-control",
                "placeholder": "Дата"
            }),
            "full_text": Textarea(attrs={
                "class": "form-control",
                "placeholder": "Текст статьи"
            })
        }
```

В представлениях NewsUpdateView и NewsDeleteView указаны URL-адреса для перенаправления на страницу входа (login_url) в случае, если пользователь не аутентифицирован.

- Декоратор @login_required применен к созданию новостей только для авторизованных пользователей.

4. Приложение «registration»

Приложение "registration" отвечает за регистрацию, вход и выход пользователя. Вот подробное описание основных компонентов приложения:

- Модели:

В файле models.py определена модель CustomUser, которая является расширением модели AbstractUser из django.contrib.auth.models.

Модель CustomUser добавляет дополнительное поле email типа EmailField и связи groups и user_permissions с моделями Group и Permission соответственно.

Метод __str__ возвращает имя пользователя (username).

- URL-пути:

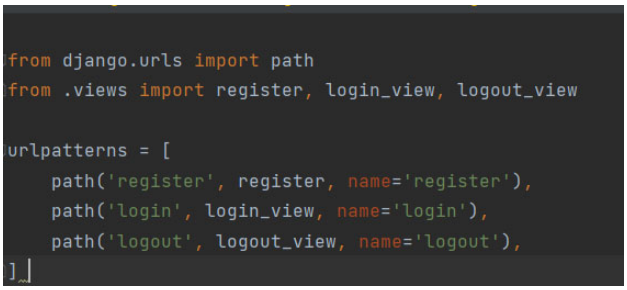
Файл urls.py содержит определения URL-путей, которые связывают URL с соответствующими представлениями (views).

В данном случае определены следующие пути:

1. "register" - путь для регистрации нового пользователя. Он обрабатывается функцией register из views.py.

2. "login" - путь для входа пользователя. Он обрабатывается функцией login_view из views.py.

3. "logout" - путь для выхода пользователя. Он обрабатывается функцией logout_view из views.py.



```python
from django.urls import path
from .views import register, login_view, logout_view

urlpatterns = [
    path('register', register, name='register'),
    path('login', login_view, name='login'),
    path('logout', logout_view, name='logout'),
]
```

- Представления (views):

В файле views.py определены функции, которые обрабатывают запросы от пользователей и взаимодействуют с формами и моделями для выполнения соответствующих действий.

Функция register обрабатывает запрос для регистрации нового пользователя. Если метод запроса - POST и форма RegistrationForm валидна, новый пользователь сохраняется, и пользователь перенаправляется на страницу входа (login). В противном случае форма регистрации передается в шаблон registration/register.html для отображения.

Функция login_view обрабатывает запрос для входа пользователя. Если метод запроса - POST и форма LoginForm валидна, пользователь аутентифицируется и перенаправляется на домашнюю страницу (home). В противном случае пользователю отображается сообщение об ошибке авторизации.

Функция logout_view обрабатывает запрос для выхода пользователя. После выхода пользователя он перенаправляется на домашнюю страницу.

- Формы:

В файле forms.py определены формы RegistrationForm и LoginForm для

регистрации и входа пользователей соответственно.

Форма RegistrationForm основана на модели User из django.contrib.auth.models и содержит поля "username", "email", "password1" и "password2". Она требует обязательное заполнение поля email.

Форма LoginForm является наследником AuthenticationForm из django.contrib.auth.forms и содержит поля "username" и "password". Поле "username" установлено как обязательное и имеет фокус при открытии страницы.

Заключение

В результате разработки новостного веб-сайта с панелью управления на основе Django было создано два основных приложения: "news" и "registration".

Приложение "news" позволяет добавлять, редактировать и удалять новости. Оно содержит модель Articles для представления новостей с соответствующими полями, такими как заголовок, анонс, содержание и дата публикации. В приложении определены URL-пути для отображения списка новостей, создания новости, просмотра, редактирования и удаления конкретной новости. Также были разработаны шаблоны для отображения новостей и форма ArticlesForm для создания и редактирования новостей.

Приложение "registration" обеспечивает функциональность регистрации, входа и выхода пользователя. Оно содержит модель CustomUser, которая расширяет стандартную модель пользователя AbstractUser и добавляет поле электронной почты. Пользователи могут зарегистрироваться через форму RegistrationForm, а затем входить на сайт, используя форму LoginForm. Также были определены URL-пути для регистрации, входа и выхода пользователя.