Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Отчет по лабораторным работам №7 и №8**

Дисциплина: «Разработка профессиональных приложений»

Связи в БД. CRUD. Регистрация и авторизация

Выполнил:

студент группы ИВТАПбд-21

Карпова А.С.

Проверил:

преподаватель кафедры

«Вычислительная техника»

Исхаков И.И.

Ульяновск, 2023

**Задание по варианту**

Для предложенного варианта заданий необходимо реализовать веб-сайт в среде Django, удовлетворяющий следующим требованиям:

1. Количество таблиц в БД - не менее 5.

2. Должны быть реализованы 3 типа связей: one-to-one, one-to-many, man-to-many.

3. Таблицы создаются с использованием моделей в Django и миграций.

4. Как минимум, для 1 сущности должны быть реализованы CRUD операции (Create, read, update, delete) + представление списка сущностей.

5. Остальные таблицы должны быть доступны для редактирования в административной части сайта.

6. Необходимые пять таблиц должны быть разработаны для моделирования предметной области, заданной вариантом. Необходимо придумать эти таблицы, не забывая про здравый смысл.

7. Добавить минимум 2 роли, соответствующих реальным ролям в выбранной предметной области. Добавить пользователей, обладающих этими ролями.

8. Реализовать средства регистрации пользователей, аутенфикации.

9. Добавить средства авторизации и разграничения прав доступа разным ролям к разным сущностям.

**Описание проекта**

1. Пользователю необходимо зарегистрировать аккаунт на сайте.
2. Пользователю необходимо авторизоваться, после авторизации он попадает на страницу новостей с постами.
3. На странице с 7 лабораторной работой, пользователь может просмотреть все посты.
4. Так же админ может добавлять посты.

**Описание реализации**

В файле views.py происходит обработка запроса на добавление / изменение записи в таблице Likes.

Функция like\_handler() проверяет роль пользователя, если она не "admin", то он будет перенаправлен на главную страницу. Если же роль совпадает с "admin", то происходит проверка метода запроса - POST или GET.

Если метод запроса POST, то создается экземпляр формы LikesForm, которой передаются данные из POST-запроса. Затем происходит проверка формы на валидность и, если данные валидны, вызывается метод save() формы для сохранения данных в базе данных. В случае, если данные не валидны, выводится сообщение о ошибке.

Если же метод запроса GET, то функция возвращает страницу с формой на добавление новой записи.

В файле forms.py находятся модели, которые используются в проекте.

UsersForm: форма для добавления/изменения записи в таблице Users. В качестве полей выступают все поля модели Users, кроме роли. Для поля "Никнейм" добавлен атрибут placeholder, который определяет подсказку для ввода данных.

LikesForm: форма для добавления/изменения записи в таблице Likes. В качестве полей выступают поля "post" и "nickname". Для них также добавлен атрибут placeholder.

PostsForm: форма для добавления/изменения записи в таблице Post. В качестве полей выступают поля "nickname", "header" и "text". Для каждого поля также добавлен атрибут placeholder с подсказкой для ввода данных.

TagsForm: форма для добавления/изменения записи в таблице Tags. В качестве полей выступают все поля модели Tags, кроме "pid". Для поля "Тег" добавлен атрибут placeholder.

RatingForm: форма для добавления/изменения записи в таблице Rating. В качестве полей выступают поля "nickname" и "likes\_sum". Для обоих полей также добавлен атрибут placeholder.

CreateUserForm: форма для создания нового пользователя в системе. Она расширяет стандартную форму UserCreationForm модели User, добавляя два поля - username и password2, которые будут использоваться для создания новой учетной записи.

В файле models.py находятся модели, которые используются в проекте.

Users: модель, представляющая пользователя. Она содержит индекс пользователя, никнейм, тип пользователя ('автор' или нет), администратор ли он, а также пароли. Для поля никнейм используется уникальный индекс, чтобы обеспечить уникальность каждого пользователя в системе.

Rating: модель, представляющая рейтинг пользователя. Она содержит никнейм пользователя и его рейтинг, который изначально равен 0. Для поля никнейм используется связь OneToOneField с моделью Users.

Post: модель, представляющая пост. Она содержит идентификатор поста, никнейм автора, заголовок, текст и количество лайков. Идентификатор поста является первичным ключом модели. Для поля никнейм используется обычное текстовое поле.

Tags: модель, представляющая тег поста. Она содержит индекс, идентификатор поста и тег. Для поля идентификатор поста используется связь с моделью Post. Индекс используется в качестве первичного ключа модели.

Likes: модель, представляющая лайк пользователя. Она содержит индекс, номер поста и никнейм пользователя. Для поля номер поста используется связь с моделью Post. Индекс используется в качестве первичного ключа модели.

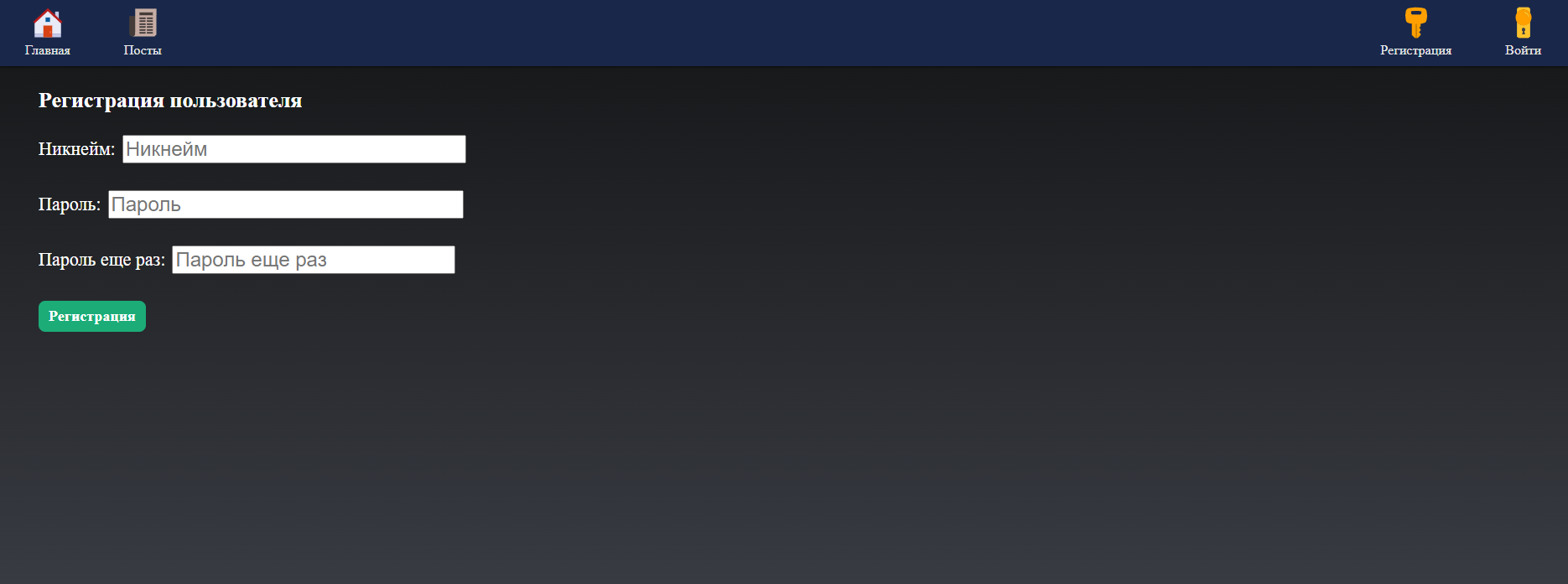
Каждая модель имеет метод get\_absolute\_url, который возвращает URL-адрес для данной модели. Это может быть полезно в случае, если требуется перейти к странице с объектом модели.

В файле admin.py содержится регистрация моделей в административной панели Django.

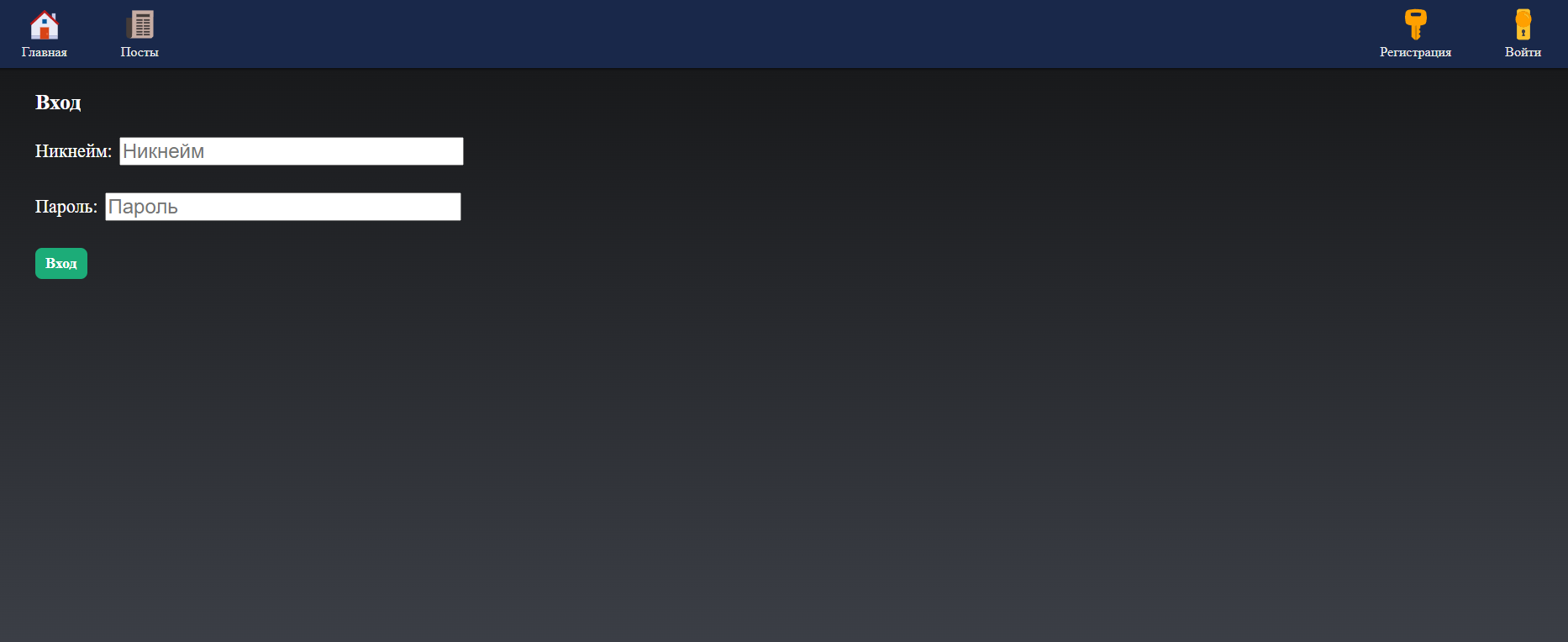
Функция admin.site.register() используется для регистрации каждой модели в административной панели. Передается один аргумент - модель, которую необходимо зарегистрировать.

Таким образом, для каждой из моделей Users, Likes, Rating, Post и Tags вызывается функция admin.site.register(), чтобы зарегистрировать модели в административной панели Django. После этого эти модели будут доступны для редактирования и управления в административной панели Django.

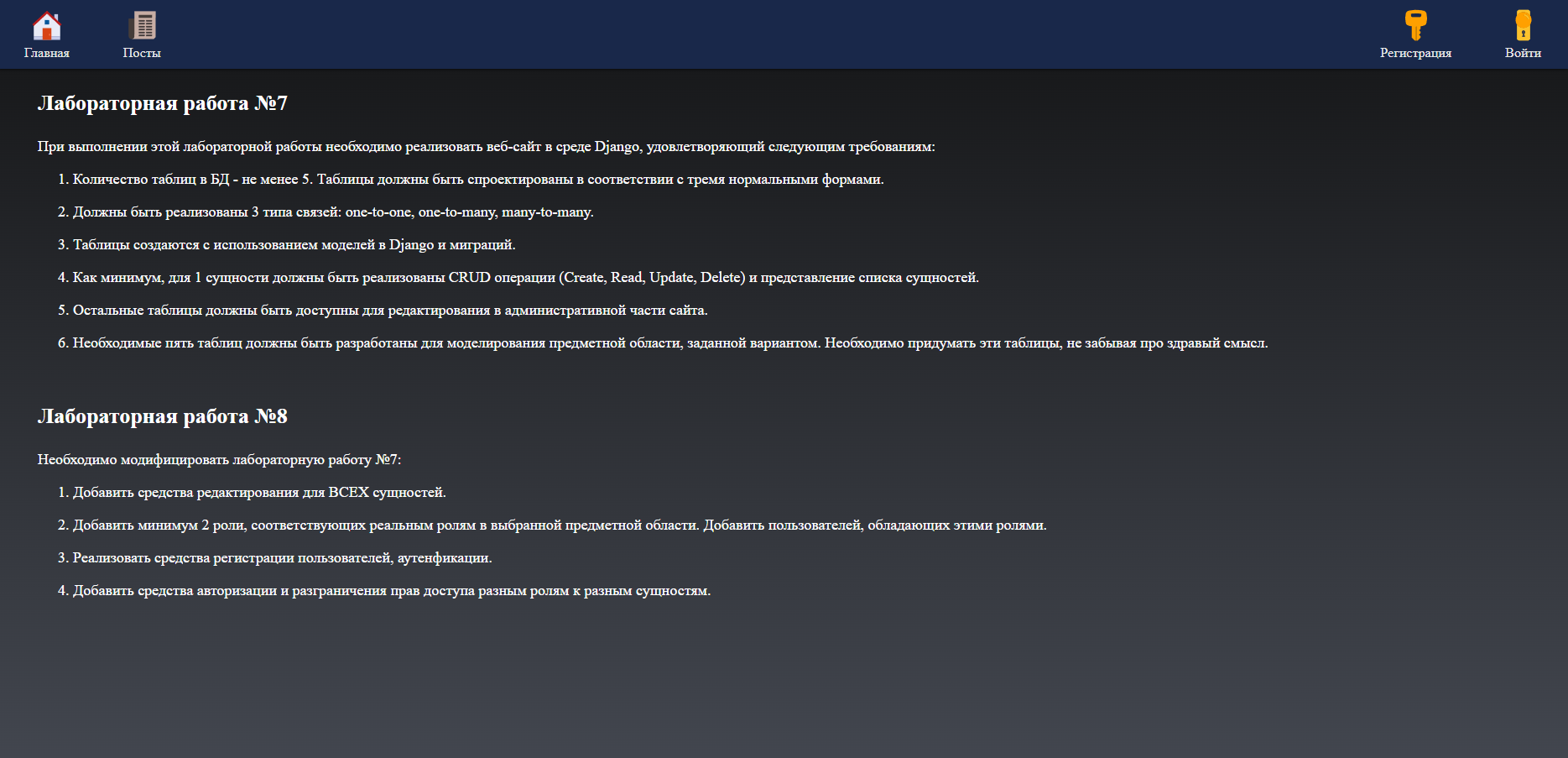
**Тестирование web приложения**



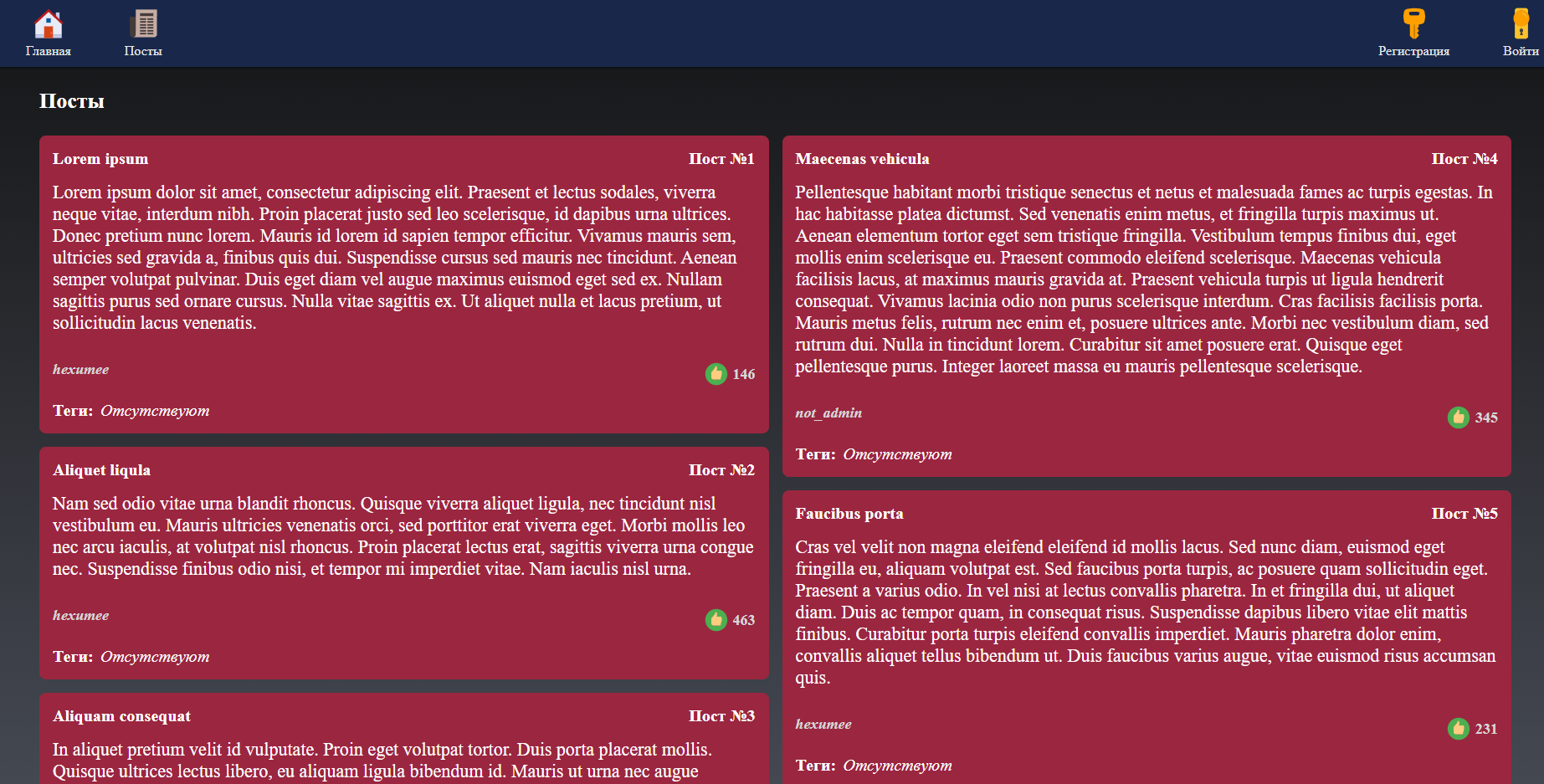
*Рис. 1.1 Страница с регистрацией аккаунта*

**

*Рис. 1.2 Страница с авторизацией*

**

*Рис. 1.3 Главная страница*

**

*Рис 1.4 Страница с постами*

**Памятка для разработчика:**

1. Скачать проект на официальном **Github** разработчиков по ссылке ниже
2. Запустить проект в PyCharm или любой другой удобной IDLE
3. Открыть терминал
4. Установить Django командой «pip install Django»
5. Перейти в директорию «app» командой «cd .\app\»
6. Запустить проект командой «py manage.py runserver»
7. После запуска откройте **url** сайта <http://127.0.0.1:8000/>