**Пояснительная записка**

**Название проекта**: Соцсеть «Code»

**Исполнители:** Иванова Юлия Александровна, Русанов Андрей Сергеевич, Муратов Богдан Евгеньевич.

**Цель проекта**: Создать социальную сеть для общения пользователей с возможностью публикации постов, обмена сообщениями и управления друзьями.

**Используемые технологии и библиотеки:**

*Backend:*

* Flask
* SQLAlchemy
* pip 24.3.1
* WTForms
* Flask-Login
* Flask\_socketio

*Frontend:*

* HTML/CSS/JavaScrip

**План проекта:**

**1. Регистрация и авторизация пользователей**

1.1. Регистрация

1.2. Авторизация

**2. Базовый шаблон**

2.1. Структура:

Верхняя панель (лого, поиск, уведомления, профиль).

Боковое меню (настройки, чаты, лента, лайки, друзья, выход).

2.2. Поиск пользователей

Поиск по ID → переход на страницу пользователя или 404.

2.3. Запросы в друзья

Отображение входящих запросов.

Кнопки "Принять" / "Отклонить".

**3. Главная страница (лента)**

3.1. Структура:

Кнопка создания поста (форма: описание, картинки, категория).

Лента постов (сортировка по дате / рекомендациям / фильтрам).

Лайки, комментарии, редактирование постов.

3.2. Рекомендации

Приоритет постов из любимых категорий пользователя.

**4. Личная страница**

4.1. Структура:

Информация о пользователе (аватар, фон, имя, фамилия).

Список его постов.

**5. Страница «Друзья»**

5.1. Структура

**6. Страница «Настройки»**

6.1. Редактирование профиля:

Форма с текущими данными (имя, фамилия, аватар, фон, возраст).

**7. Страница «Лайки»**

Отображение понравившихся постов.

**8. Страница «Чаты»**

8.1. Структура:

Список чатов пользователя.

Кнопка создания чата.

Поиск по названию чата.

**9. Страница «Личный чат»**

9.1. Структура:

Информация о чате (аватар, название, участники).

История сообщений (сортировка по дате).

Форма отправки сообщений.

**10. Страница «Комментарии»**

**11. Обработчик страницы «404»**

**12. Скрины приложения**

**13. Структура Базы данных (скрины)**

**14. Используемые функции и классы**

**15. Особенности соцсети**

15.1 Собственное API

15.2 WebSocket

15.3 Рекомендации

База данных

Таблицы:

users – данные пользователей.

posts – посты.

comments – сообщения пользователей.

chats – Чаты комментариев.

private\_chats – личные чаты.

**1. Регистрация и авторизация пользователей**

**1.1 Создание регистрации:**

Представляет собой форму для регистрации пользователей в приложении. Регистрация проходит в три этапа: первый — узнаём у пользователя его имя и фамилию; второй — узнаём его возраст, город и пол; третий — узнаём почту и предлагаем придумать пароль. После окончания регистрации перенаправляем пользователя на страницу авторизации, а данные о нём заносим в таблицу users.

**1.2 Создание авторизации:**

Авторизация состоит из одной формы. В ней всего два поля: для ввода пароля и логина. После того, как пользователь ввёл их необходимо найти его данные в таблице users и проверить совпадают ли пароли. Если найти пользователя не удалось, или не совпал пароль, то об этом нужно сообщить пользователю.

**2. Базовый шаблон**

**2.1 Структура:**

Базовый шаблон состоит из двух частей: верхней панели и меню. В верхней панели расположены логотип и название соцсети, форма для поиска пользователей, кнопка для просмотра уведомлений, кнопка для перехода на личную страницу. Меню содержит 7 кнопок: настройки, чаты, лента, лайки, друзья, фильтры (при нажатии открывается панель с фильтрами, при помощи которой можно сортировать посты), выйти.

**2.2 Поиск пользователей:**

Пользователь вводит в поисковое поле id пользователя, которого хочет найти. Если такой есть, то пользователя нужно перенаправить на его страницу, иначе — на страницу 404.

**2.3 Запросы в друзья:**

При нажатии на кнопку для просмотра уведомлений, открывается окно, в котором будут отображаться все запросы в друзья. Так же там есть кнопки, при помощи которых пользователь может либо принять, либо отклонить запрос в друзья.

**3. Главная страница (лента)**

**3.1 Структура:**

На главной странице есть: кнопка для создания поста (при её нажатии открывается всплывающее окно. В нём расположена форма для создания поста. В ней 3 поля: картинки для поста, описание, категория поста.) и сами посты. Посты отображаются в соответствии с датой их создания, рекомендаций пользователя или фильтров, которые он установил. У каждого поста есть три кнопки: поставить лайк (при нажатии на неё на сервер отправляется обновлённая информация о количестве лайков у поста, эта информация заносится в таблицу posts.), написать коменатрий, изменить/удалить пост (эту кнопку видит только создатель поста.).

**3.2 Рекомендации:**

Если у пользователя есть более 10 лайков, то необходимо найти категорию, которой он ставил лаков больше всего. Посты из этой категории показываются в первую очередь, посты же из самой непопулярной категории у пользователя — в последнею.

**4. Личная страница**

**4.1 Структура:**

На личной странице находятся посты, которые делал владелец страницы, также здесь есть блок с информацией о пользователе: аватарка, фон его профиля, имя.

**5. Страница «Друзья»**

**5.1 Структура:**

На этой отображаются друзья пользователя (в том порядке в каком они расположены в базе данных.). Каждого друга можно: удалить из друзей и написать ему. Так же на этой странице присутствует поиск, при помощи которого можно найти друга по имени, если список друзей слишком большой.

**6. Страница «Настройки»**

**6.1 Редактирование профиля:**

На странице настройки есть форма для редактирования информации о пользователе (Имя, Фамилия, аватарка, фон профиля, возраст). По умолчанию в эту форму записаны текущие данные пользователя.

**7. Страница «Лайки»**

На странице лайки отображаются посты, которые понравились пользователю.

**8. Страница «Чаты»**

**8.1 Структура:**

На этой странице отображаются чаты, в которых пользователь является участником (в том порядке в каком они расположены в базе данных.), кнопка для создания чата. Каждый чат можно: удалить и отправить сообщение в него. Так же на этой странице присутствует поиск, при помощи которого можно найти чат по названию, если список чатов слишком большой.

**9. Страница «Личный чат»**

9.1 **Структура:**

На этой странице есть: панель с информацией о чате (аватарка чата, участники, название чата), кнопка для редактирования чата, панель для отображения сообщений, поле для ввода сообщения, кнопка для отправки сообщения. Когда пользователь входит в чат, проверяется является ли он его участником, если да, то он запускается в чат, иначе отправляется на страницу 404. Все сообщения отсортированы по дате их отправки.

**10. Страница «Комментарии»**

На этой странице есть панель для отображения комментариев, поле для ввода комментария, кнопка для отправки комментария. Все комментарии отсортированы по дате их отправки.

**11. Обработчик страницы «404»**

Создан дизайн для страницы 404 в стиле приложения, т.к пользователя иногда сюда нужно будет отправлять.

**12. Скрины приложения**

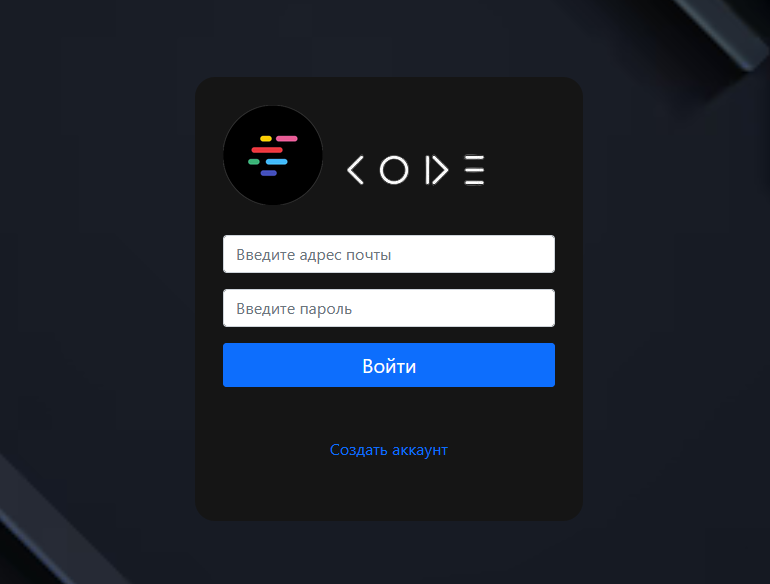
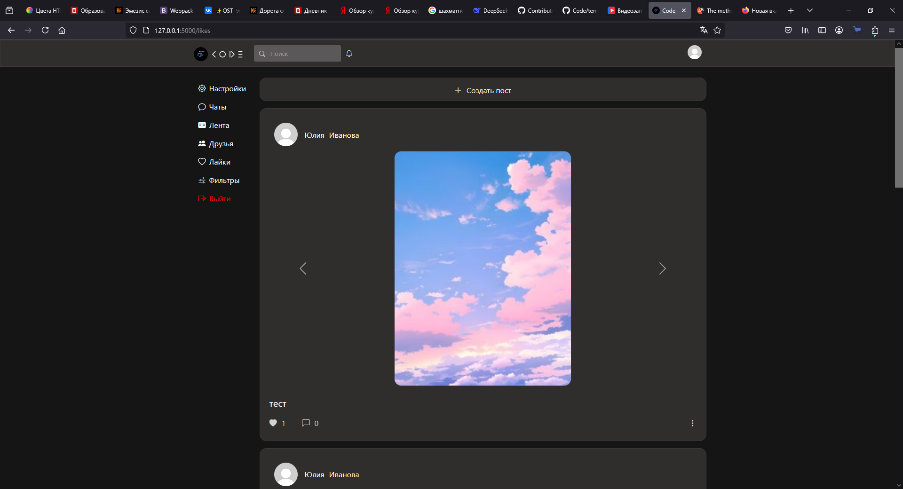
**** 

Рисунок 1 Рисунок 2

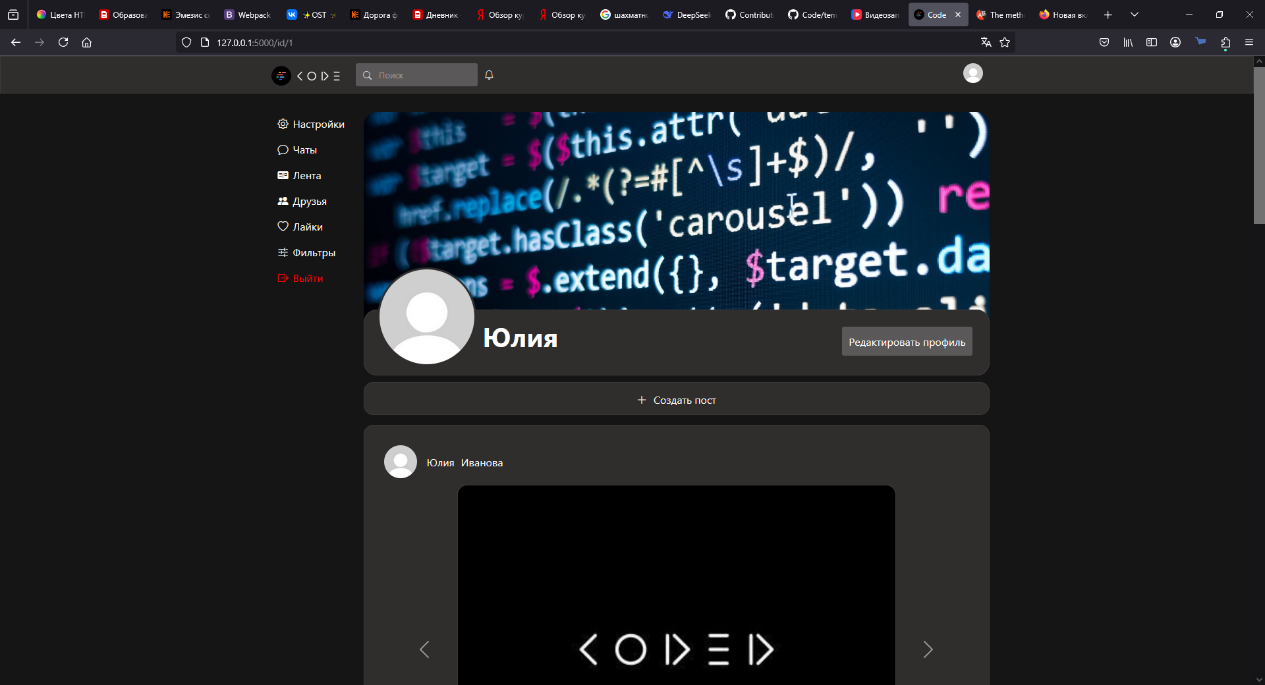


Рисунок 3

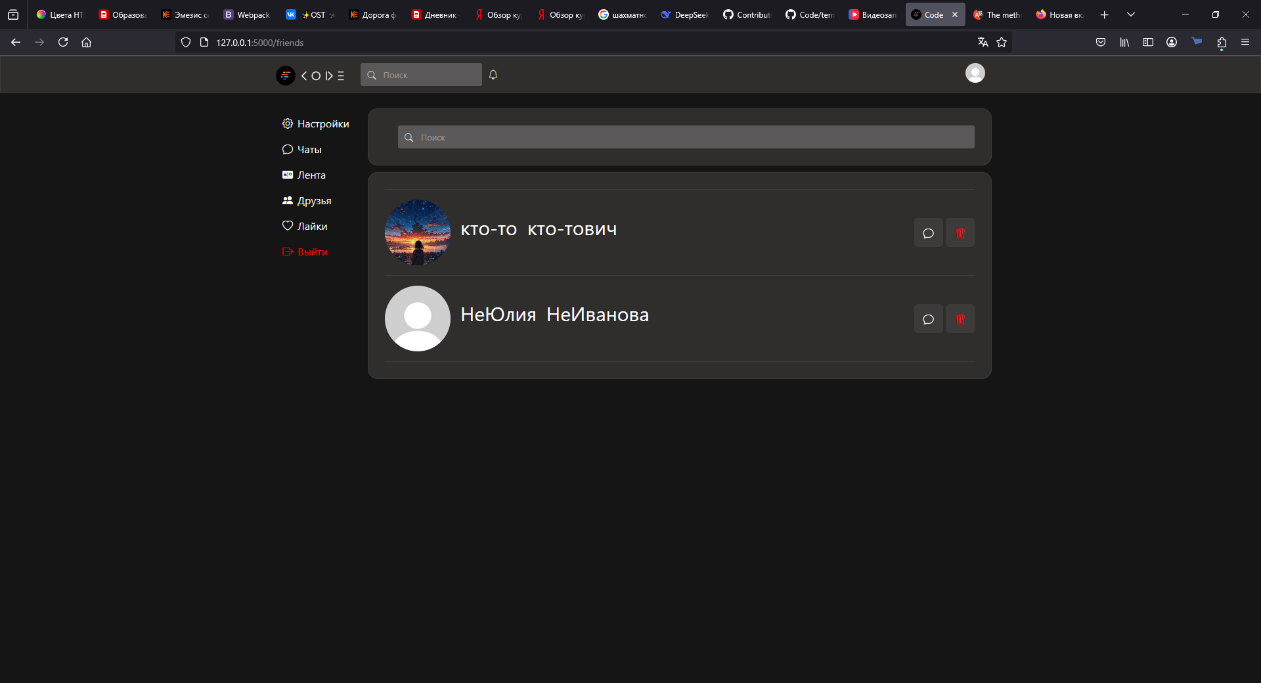


Рисунок 4

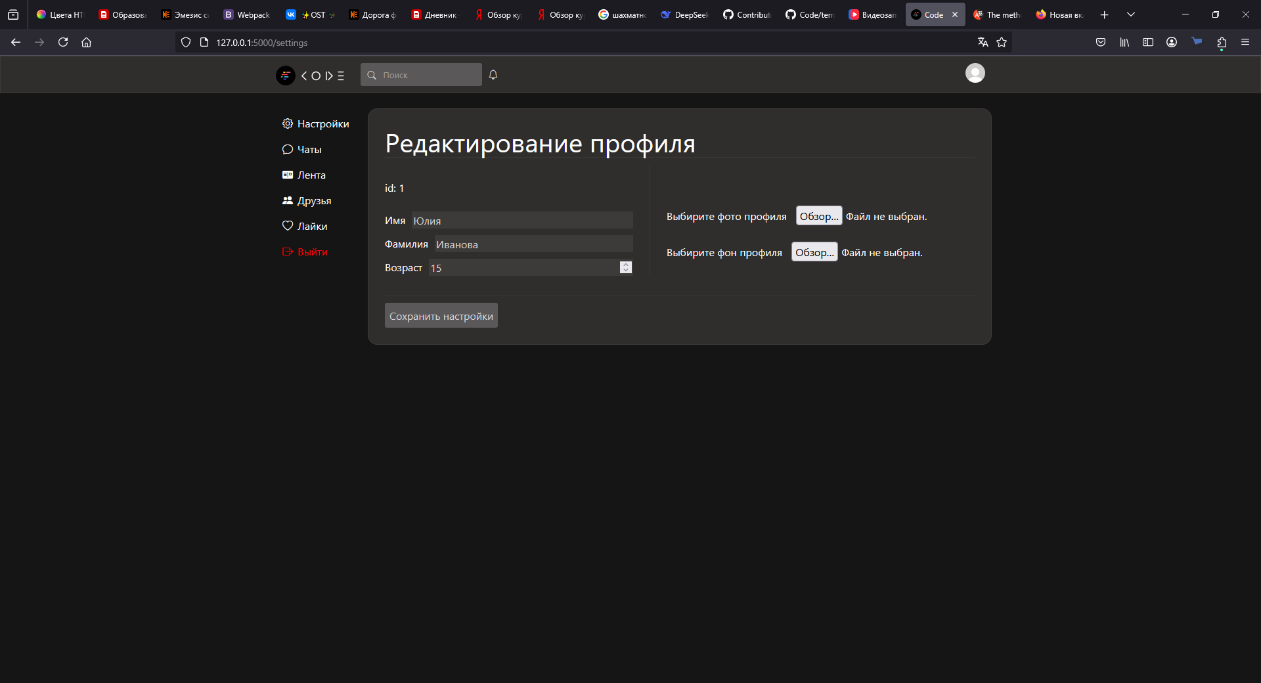


Рисунок 5

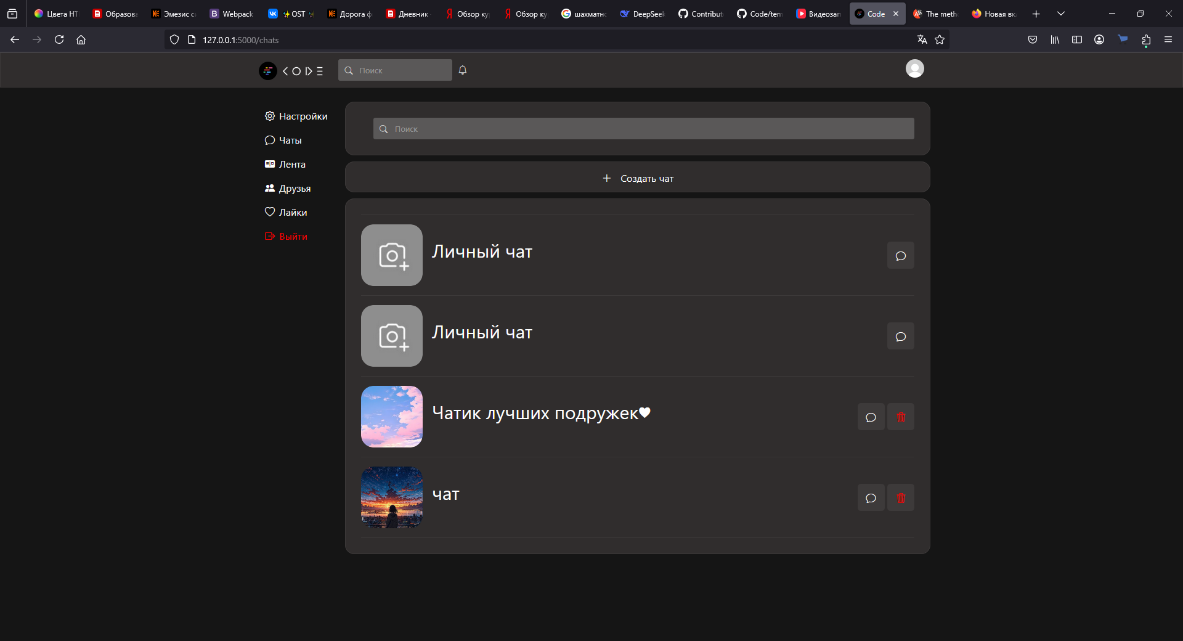


Рисунок 6

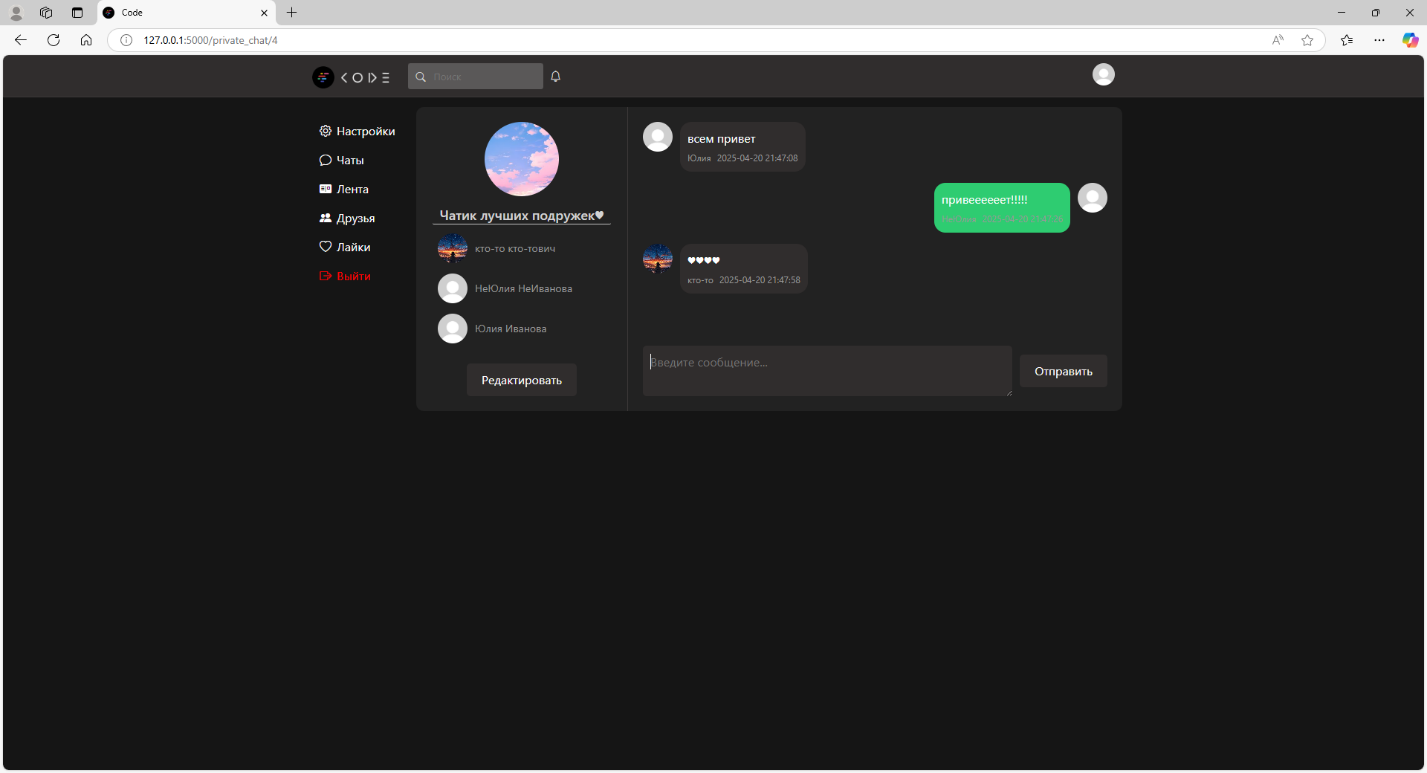


Рисунок 7

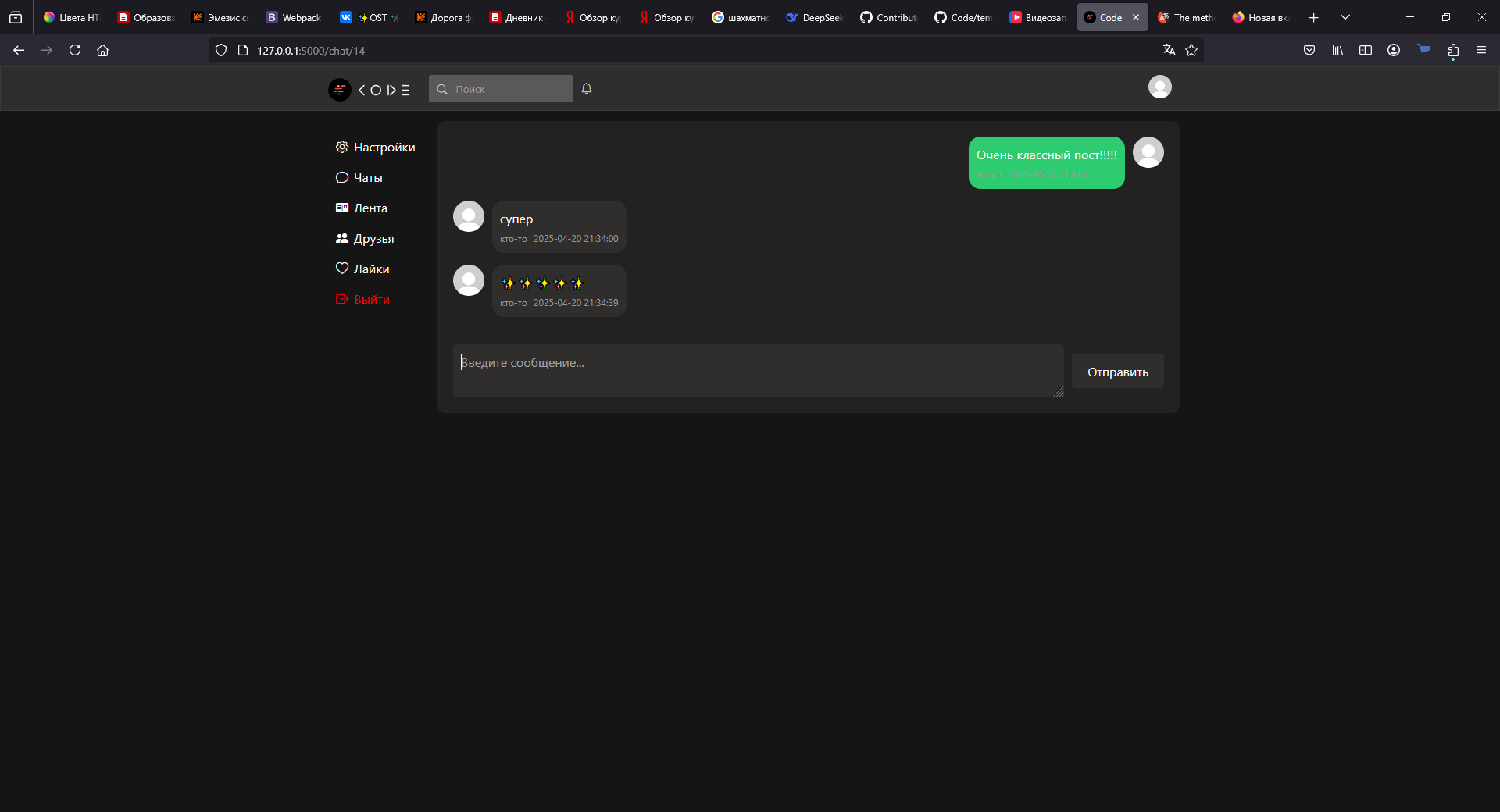


Рисунок 8

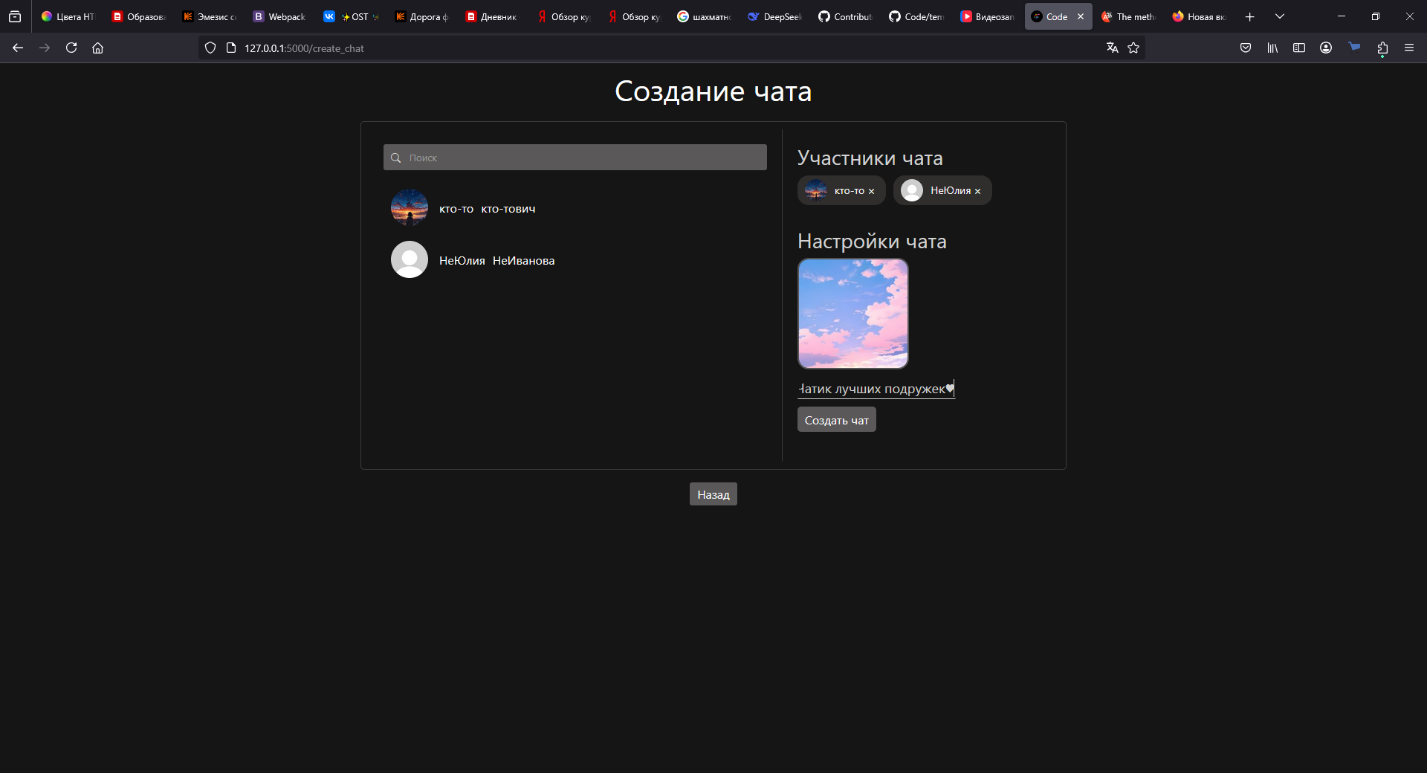
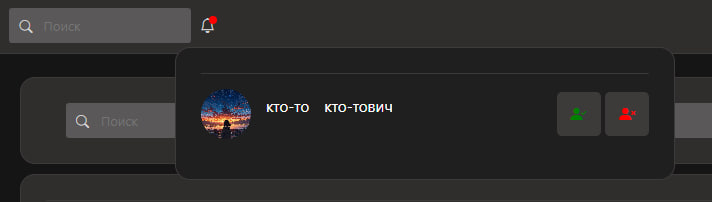
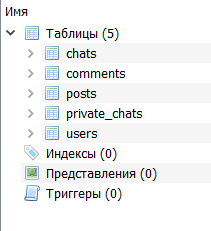
 

Рисунок 9 Создание чата

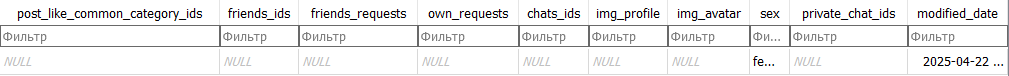
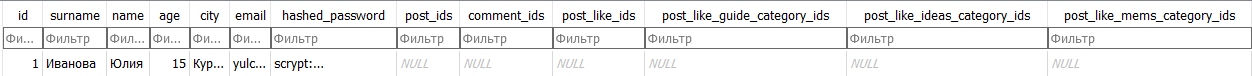


Заявка в друзья

**13. Структура Базы данных (скрины)**



Users



Comments



Posts



Private\_chats



Chats



**14. Используемые функции и классы**

**allowed\_file** - проверяет допустимость расширения загружаемого файла.  
**form** (/create\_accaunt) - обрабатывает регистрацию новых пользователей.  
**main** (/main) - отображает главную страницу с лентой постов.  
**likes** (/likes) - показывает посты, лайкнутые текущим пользователем.  
**friends** (/friends) - отображает список друзей.  
**login** (/) - авторизует пользователя.  
**logout** (/logout) - выполняет выход из аккаунта.  
**id** (/id/<Clientid>) - показывает профиль пользователя и его посты.  
**create\_chat** (/create\_chat) - отображает страницу создания чата с друзьями.  
**edit\_chat** (/edit\_private\_chat/<chat\_id>) - позволяет редактировать существующий приватный чат.  
**private\_chat** (/private\_chat/<id>) - отображает приватный чат между пользователями.  
**handle\_join\_chat** (WebSocket) - обрабатывает подключение пользователя к чату.  
**handle\_message** (WebSocket) - обрабатывает отправку сообщений в чате.  
**chat** (/chat/<id>) - отображает публичный чат поста.  
**chats** (/chats) - показывает список всех чатов пользователя.  
**edit\_post** (/edit\_post/<post\_id>) – редактирует пост.  
**settings** (/settings) – отображает настройки профиля.  
**page\_not\_found** – отображает страницу 404.  
**custom\_sorted\_posts** – сортирует посты по фильтру.  
**sort\_posts\_by\_user\_likes** – рекомендации.  
**LikeResource.post** - обрабатывает лайки постов, обновляя счетчики лайков пользователя и поста.  
**ChatResource.post** - создает новый чат.  
**ChatResource.put** - обновляет данные существующего чата (название, аватар, участников).  
**ChatResource.delete** - удаляет чат и обновляет списки чатов у участников.  
**MessageResource.post** - создает новое сообщение в чате и обновляет связанные данные.  
**AddFriendResource.post** - отправляет запрос на дружбу, обновляя списки заявок у обоих пользователей.  
**RejectFriendRequestResource.post** - отклоняет входящий запрос на дружбу, удаляя его из списков заявок.  
**AcceptFriendRequestResource.post** - принимает запрос на дружбу: добавляет пользователей в друзья, создает приватный чат и очищает заявки.  
**RemoveFriendResource.delete** - удаляет пользователя из друзей, включая удаление связанного приватного чата.  
**DeletePostResource.delete** - удаляет пост и все связанные с ним данные (лайки, комментарии) у всех пользователей.  
**class Chat** – создаёт таблицу chats.  
**class Comments** – создаёт таблицу comments.  
**class Posts** – создаёт таблицу posts.  
**class PrivateChat** – создаёт таблицу private\_chats.  
**class User** – создаёт таблицу users.  
**class CreatePostForm** –форма для создания поста. **class EditPostForm** – форма для редактирования поста.  
**class SettingsForm**  - форма для редактирования профиля.  
**class LoginForm** – форма для авторизации.  
**class Registration** – форма для регистрации.  
**class SerchUserForm** – форма для поиска пользователей.  
**updateURL (js)** - обновляет URL с параметром фильтра и перезагружает страницу.  
**DOMContentLoaded (js)** - при загрузке страницы устанавливает выбранный фильтр из URL.  
**filterButton click (js)** - показывает/скрывает блок фильтров при клике.  
**buttonsAccept (js)** - обработчики для кнопок принятия заявок в друзья.  
**buttonsReject (js)** - обработчики для кнопок отклонения заявок в друзья.  
**bell click (js)** - управляет отображением уведомлений о заявках.  
**WebSocket chat (js)** - реализует чат через WebSocket:

* Подключение к чату
* Получение сообщений
* Отправка сообщений

**buttons delete (js)** - обработчики удаления приватных чатов.  
**friends search (js)** - динамический поиск друзей по введенному тексту.  
**friendsList click (js)** - обработчик выбора друзей для создания чата (добавляет/удаляет из списка участников)  
**chatAvatarInput change (js)** - предпросмотр выбранного аватара чата перед загрузкой  
**createChat (js)** - асинхронная функция отправки данных нового чата на сервер  
**createChatButton click (js)** - собирает данные формы и вызывает createChat()  
**lookImg (js)** - открывает увеличенное изображение с затемненным фоном (модальное окно)  
**cntLikesUp (js)** - обработчик клика по лайку: меняет иконку, счетчик и отправляет на сервер  
**addPanelSee (js)** - показывает/скрывает панель действий (редактирование/удаление) для поста  
**document.onclick (js)** - глобальный обработчик для скрытия всех открытых панелей действий  
**set\_contaner\_2 (js)** - валидация и переход на 2-й шаг формы:

* Проверяет обязательные поля (имя/фамилия)
* Подсвечивает ошибки красной рамкой
* Меняет размеры контейнера и отображает следующий блок

**set\_contaner\_3 (js)** - валидация и переход на 3-й шаг формы:

* Проверяет возраст (число > 0)
* Проверяет заполненность полей (город/пол)
* Активирует required для полей пароля/email
* Меняет отображение блоков

**15. Особенности соцсети**

**15.1 Собственное API:**

У соцсети Code есть собственное API. Вот запросы, которые оно обрабатывает:

* /api/like – ставит лайк посту (вся необходимая информация отправляется через js)
* /api/chats/create – создаёт чат
* /api/chats/edit/<int:chat\_id> - редактирует данные о чате
* /delete\_private\_chat/<chat\_id> - удаляет чат
* /api/send-message – отправляет сообщение
* /add\_friend/<int:serch\_user\_id> - добавляет пользователя в друзья
* /accept\_request/<user\_id> - принимает заявку в друзья
* /reject\_request/<user\_id> - отклоняет заявку в друзья
* /delete\_friend/<friend\_id> - удаляет пользователя из друзей
* /del\_post/<post\_id> - удаляет пост

**15.2 WebSocket:**

**Социальная сеть Code использует технологию WebSocket для организации мгновенного обмена сообщениями, что позволяет пользователям видеть новые сообщения в чатах без задержек и перезагрузки страницы, значительно улучшая удобство общения.**

**15.3 Рекомендации:**

