**Поволжский Государственный Университет Телекоммуникаций и Информатики**

Кафедра «Программная инженерия»

Сдана на проверку
«__» ____2024 г.

Допустить к защите
«__» _____ 2024 г.
Защищена с оценкой
«___» _____ 2024 г.

# КУРСОВАЯ РАБОТА

По дисциплине: «Прикладное программирование»
На тему: «Разработка клиент-серверного web-приложения Блог цитат»

Пояснительная записка

Студент группы РПИС-12_____ Баженов Е.И. ____210422___
(роспись) (ФИО) (№ зачетной книжки)
Руководитель _____ к.т.н., доц. Ахметшина Э. Г.
(роспись) (ФИО)

Самара 2024

# Рецензия

# Содержание

## Описание предметной области. Актуальность

Веб-приложение "Блог цитат" представляет собой неотъемлемый элемент современного пространства онлайн-коммуникации, предназначенный для обмена и обсуждения высказываний и мыслей. В его основе лежит стремление создать виртуальное сообщество, объединенное общей целью обогащения повседневной жизни цитатами, вдохновением и обменом мудростью.

Центральным аспектом функционала приложения является возможность пользователей динамично делиться цитатами, охватывающими различные сферы: литературы, кинематографа, философских размышлений и персональных наблюдений.

Основной упор делается на стимулирование активного обсуждения. Пользователи вправе выражать свои мнения и реагировать на цитаты соучастников. Это создает интерактивное виртуальное сообщество, предоставляя каждому участнику возможность найти поддержку и взаимопонимание.

Актуальность данного приложения выражается в стремлении современного общества к личностному развитию и обмену ценным опытом. "Блог цитат" становится неисчерпаемым источником вдохновения, где люди могут не только находить смысл в словах других, но и активно участвовать в создании интеллектуального контента, способствуя культурному обогащению в онлайн-среде.

В заключение веб-приложение "Блог цитат" представляет собой уникальную платформу для обмена мыслями и высказываниями, ориентированную на создание виртуального сообщества. Оно предоставляет пользователям возможность делиться цитатами из различных областей, находить вдохновение и стремиться к личному развитию.

## Описание программы. Общие сведения

Приложение – веб-ресурс. Установка не требуется.

Необходимые требования: доступ к интернету.

Язык программирования: Java.

Платформа: Spring Boot.

Среда разработки: Intellij Idea

Объем проекта: 297Кб (вместе с исходным кодом).

Исходный код (классы и код форм):2316 строк

## Функциональное назначение

Веб-приложение "Блог цитат" разработано для управления и обмена цитатами, а также возможность редактировать данные постов, пользователей и реакций.

Основными функциональными возможностями веб-приложения являются:

1. Просмотр, создание, редактирование и удаление пользователей, постов и реакции от лица админа
2. Взаимодействие с постами:
    a. Отображение создателя поста с его цитатой
    b. Добавление постов
    c. Возможность редактировать в посте создателя, цитату и статус поста
    d. Удаление постов
3. Взаимодействие с пользователями:
    a. Просмотр список пользователей
    b. Добавление пользователь
    c. Возможность редактировать имя, пароль и роль пользователя
    d. Удаление пользователя
4. Взаимодействие с реакциями:
    a. Просмотр списка реакций
    b. Добавление разных видов реакций
    c. Редактирование реакций
    d. Удаление реакций
5. Просмотр профиля обычных пользователей:
    a. Просмотр профиля пользователя через ссылку создателя поста
    b. Отображение статистики пользователь
    c. Просмотр опубликованных постов пользователя

В данном проекте используются следующие программные продукты:

1. Spring Boot – удобный и эффективный фреймворк для разработки Java-приложений. Данный фреймворк используется для получения инструментов для создания веб-приложения.

2. Java – это объектно-ориентированный, кроссплатформенный язык программирования нужен для написания серверной части веб-приложения.

3. Spring Data JDBC – платформа в Spring Boot, предоставляющая простой и минималистический способ доступа к базам данных с использованием технологии JDBC (Java Database Connectivity).

4. HTML – это язык, который используется для создания и отображения веб-страницы.

5. JavaScript – язык программирования, который позволяет создавать логику для веб-страниц.

6. Fetch Api – это интерфейс веб-браузера, предоставляющий возможность делать сетевые запросы (HTTP запросы) из JavaScript-кода.

7. MySQL Workbench – это интегрированная среда разработки и администрирования для баз данных MySQL. Используется для создания БД и сервера БД.

# Логическая модель базы данных

## Сущности предметной области

| Сущность | Ключ сущности | Атрибуты |
|---|---|---|
| Пользователи | ID_ Поста | ID_ Пользователя, Никнейм, Пароль, Роль |
| Посты | ID_ Пользователя | ID_ Поста, ID_ Пользователя, Текст поста, Статус |
| Реакции | ID_Реакции | ID_Реакции, Название реакции |

Таблица 1.1 - Сущности

**Связи между сущностями**

| Сущности | Связи |
|---|---|
| Пользователи – Посты | Один пользователь может создать несколько постов, у одного поста может быть один создатель<br>Тип связи: один ко многим. |
| Реакции – Посты | Одна реакция может появиться у нескольких постов, у одного поста может быть несколько реакций<br>Тип связи: многие ко многим. |
| Пользователи – Реакции | Один пользователь может поставить несколько реакций, одна реакция может быть поставлена несколькими пользователями<br>Тип связи: многие ко многим. |

Таблица 1.2 – Связи сущностей

# Физическая модель данных

ER-диаграмма:

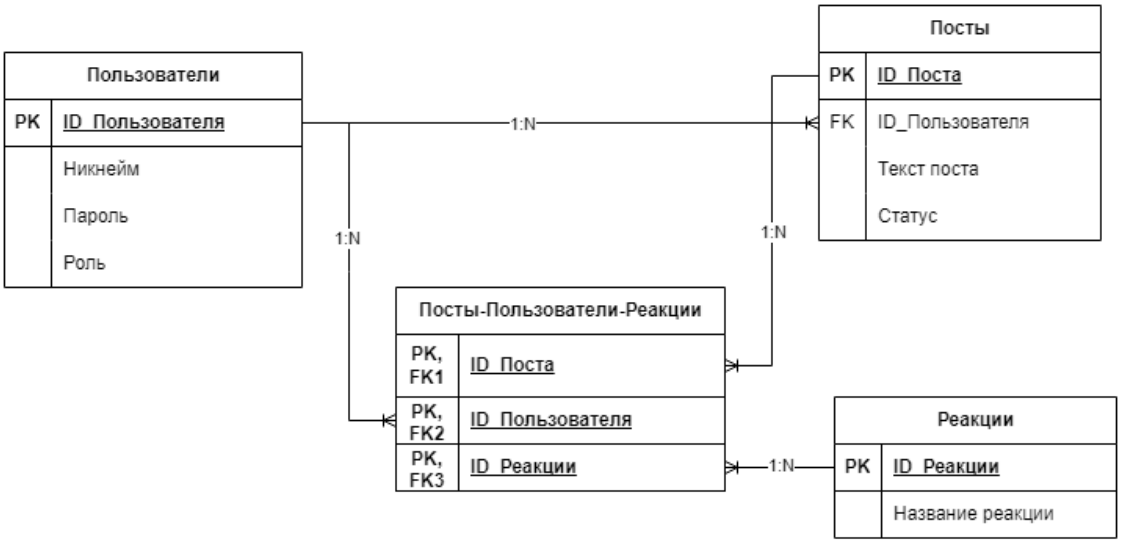


Рис. 1 -ER-модель

# Диаграмма классов

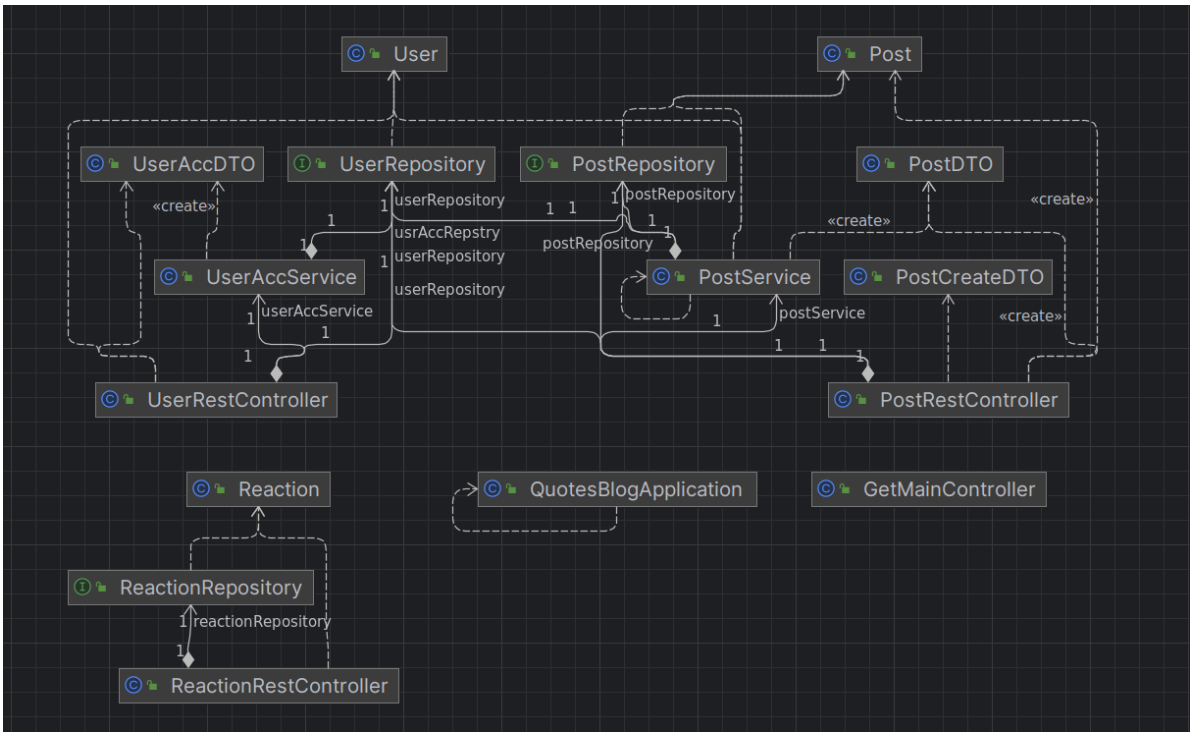


Рис. 2 - Диаграмма классов

# Диаграмма компонентов

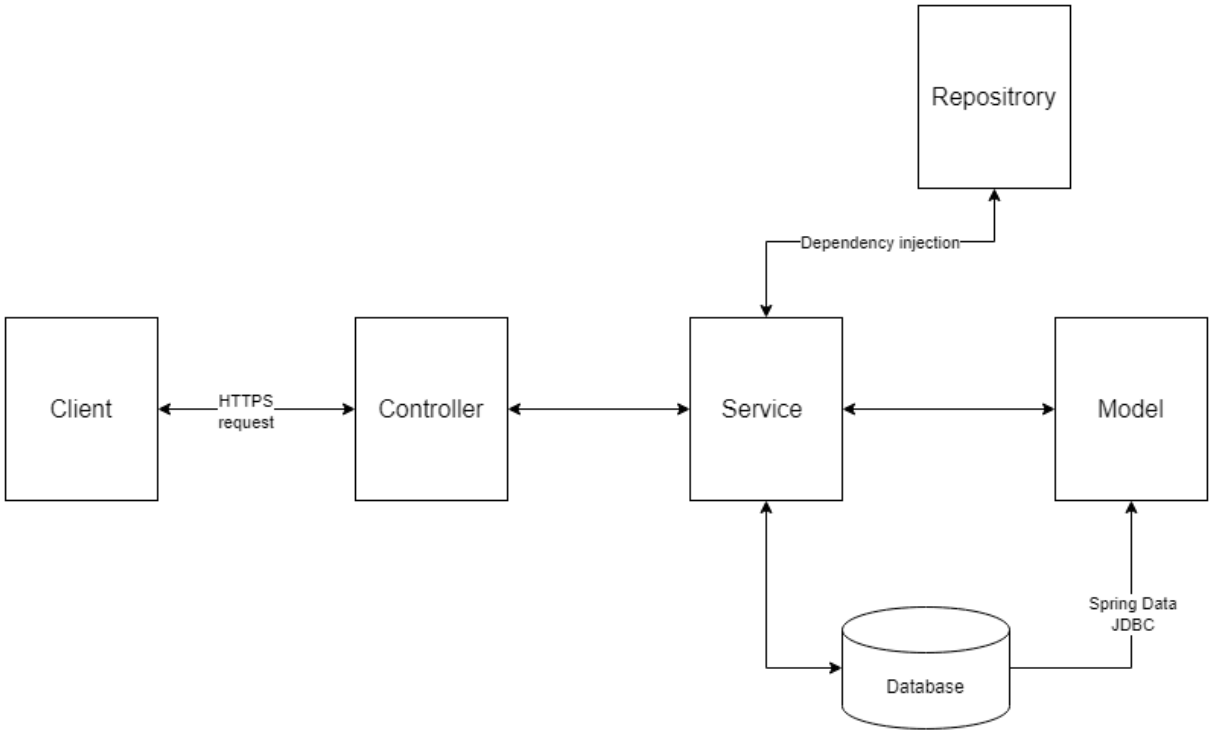


Рис. 3 - Диаграмма компонентов

# Диаграмма вариантов использования

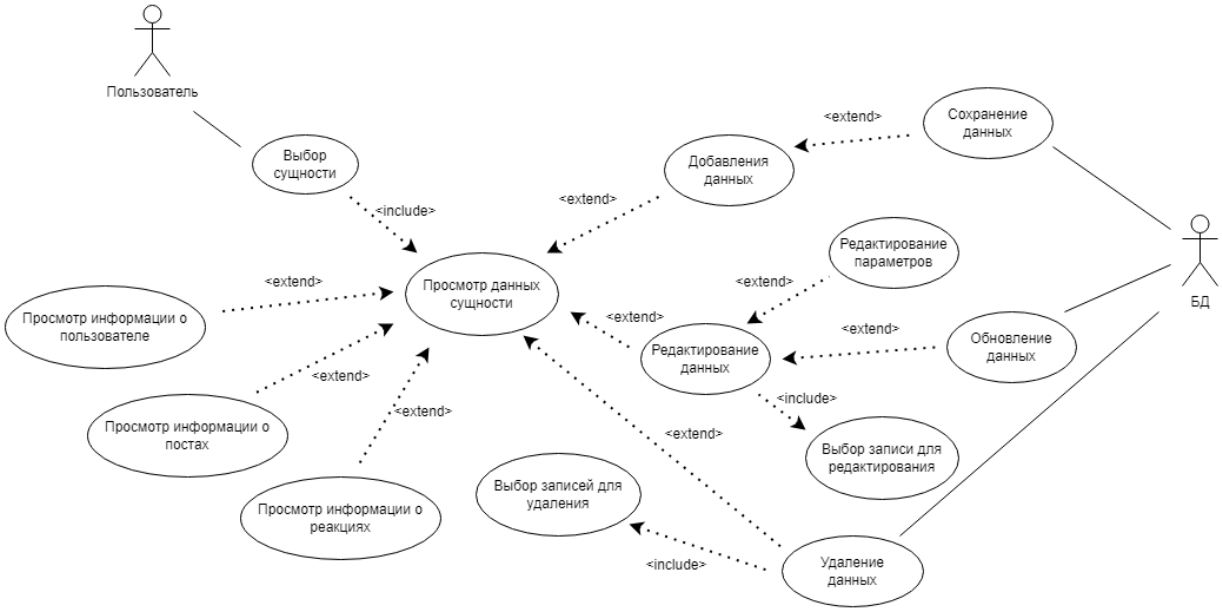


Рис. 4 - Диаграмма вариантов использования

# Диаграмма последовательности

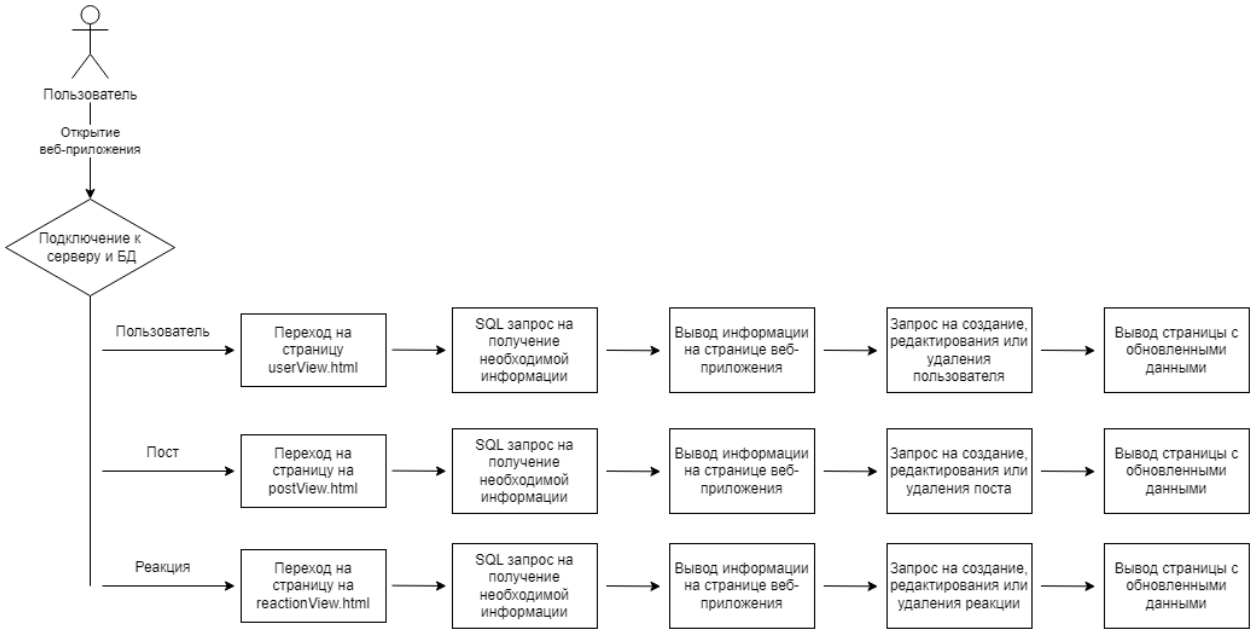


Рис. 5 - Диаграмма последовательности

## Демонстрация работы приложения

При заходе в веб-приложение пользователю отображается шапка и главная страница с опубликованными постами (рис. 6.1). Шапка сайта состоит из логотипа, при нажатии которого будет переводить на главную страницу и кнопки «Создать», «Посмотреть», выводящие список создания и просмотра доступных сущностей в данном приложении.

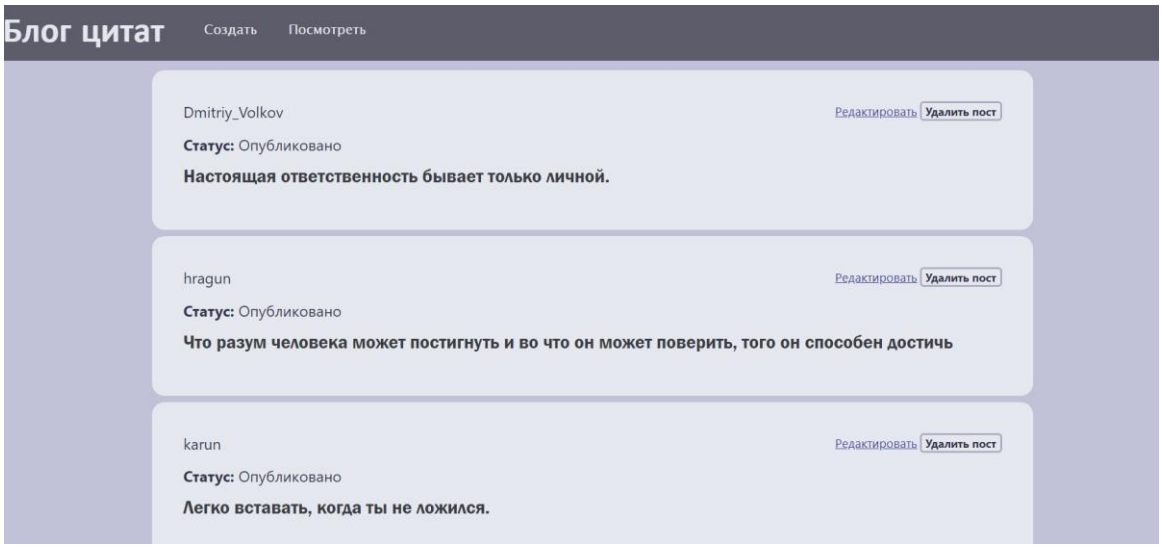


Рис. 6.1 – Главная страница

Отображение сущностей изображены на рис. 6.2–6.4. Пользователи и реакции имеет вид вывода нужных данных с записи, а цитаты реализованы как отображения поста для пользователей.

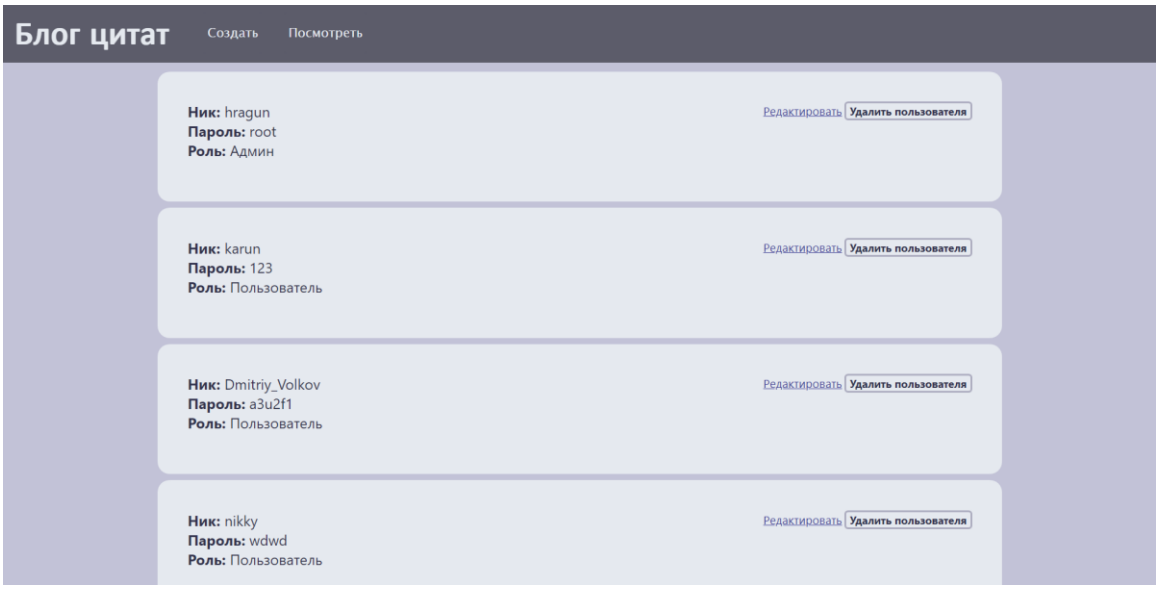


Рис. 6.2 – Отображение пользователей

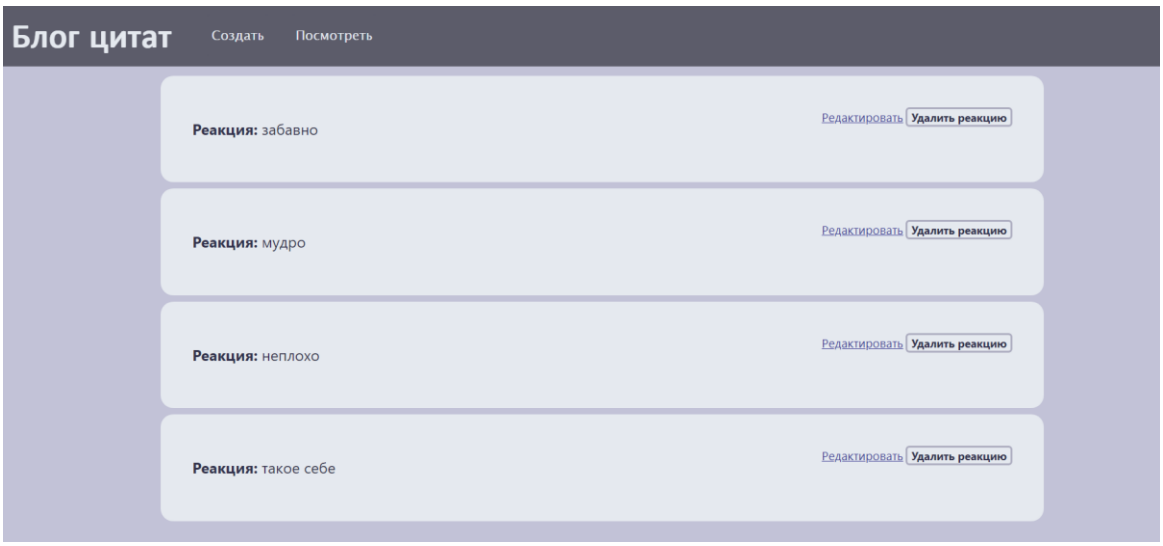


Рис. 6.3 – Отображение реакций

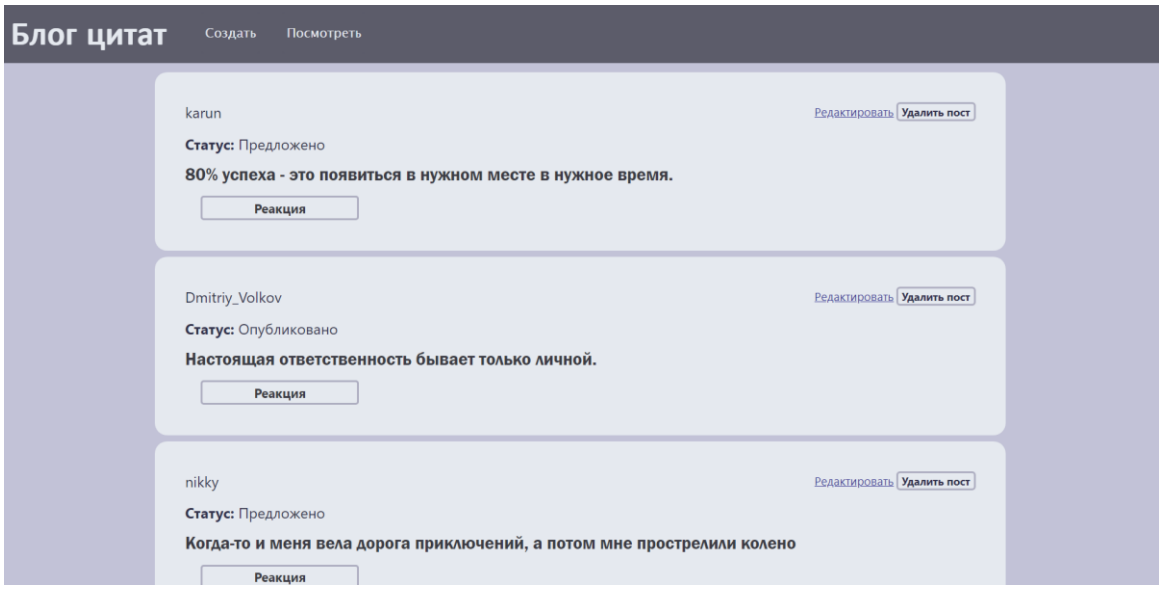


Рис. 6.4 – Отображение постов

При нажатии на «Редактировать» в любой из сущностей пользователь переходит на страницу изменения сущности (рис. 6.5–6.7). При заходе все поля авто заполняются данными той записи, которая была выбрана. После подтверждения изменения пользователь возвращается на обновленный список сущностей. При отказе пользователя возвращают на предыдущую страницу. Если пользователь пытается переписать на существующий ник или реакцию, то будет отображаться ошибка (рис. 6.8–6.9).

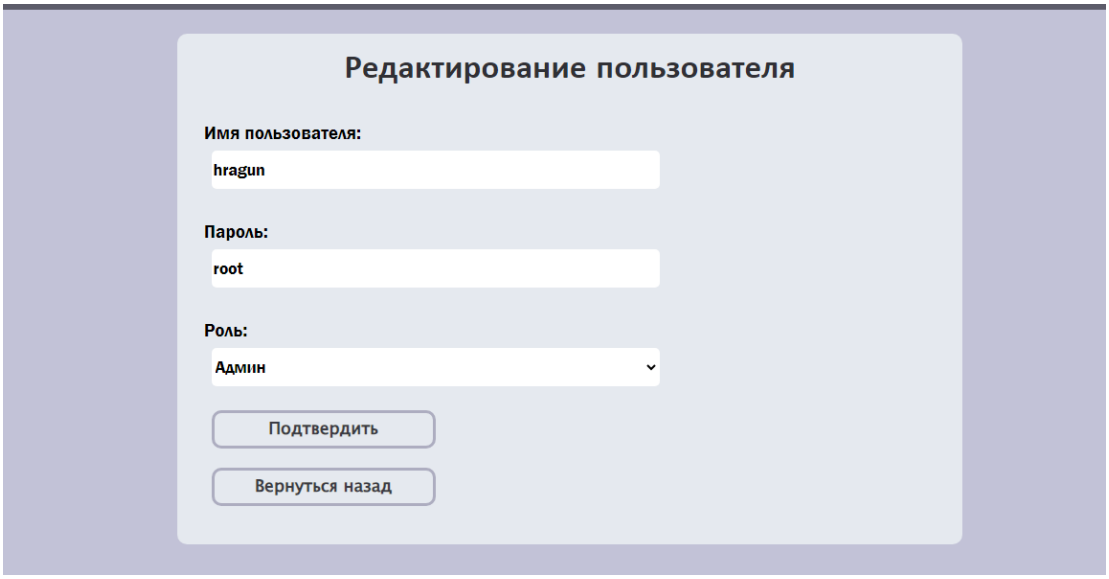


Рис. 6.5 – Редактирование пользователя

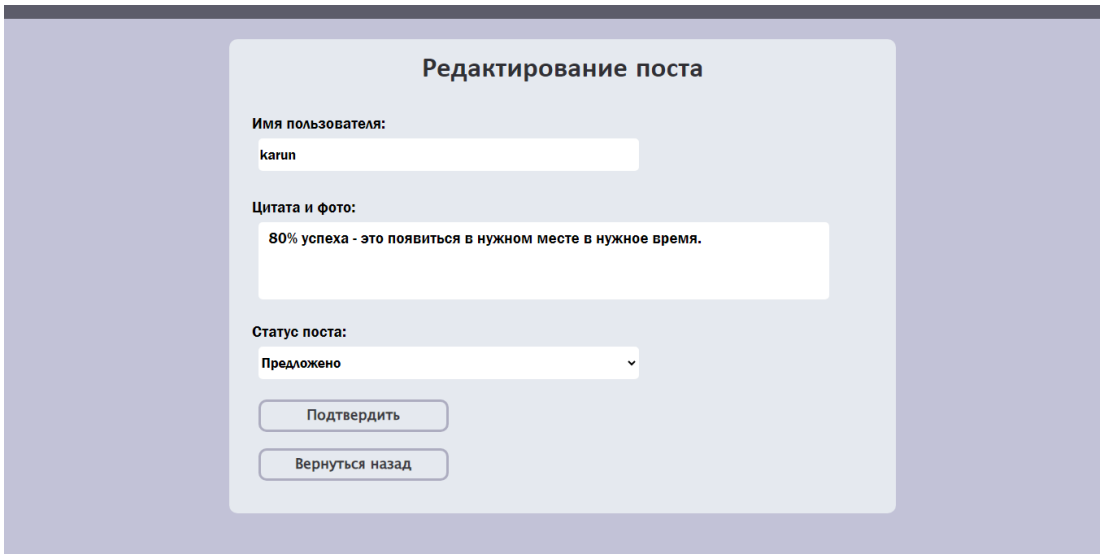


Рис. 6.6 – Редактирование поста

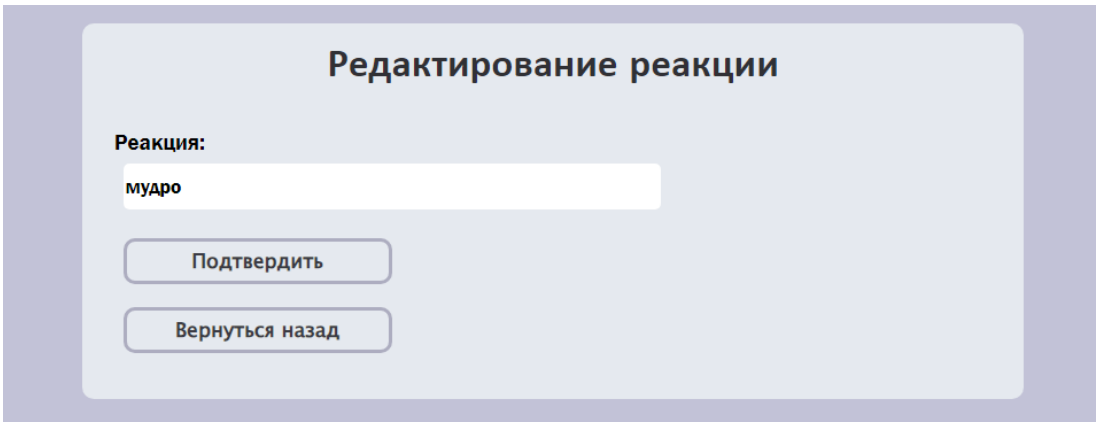


Рис. 6.7 – Редактирование реакции

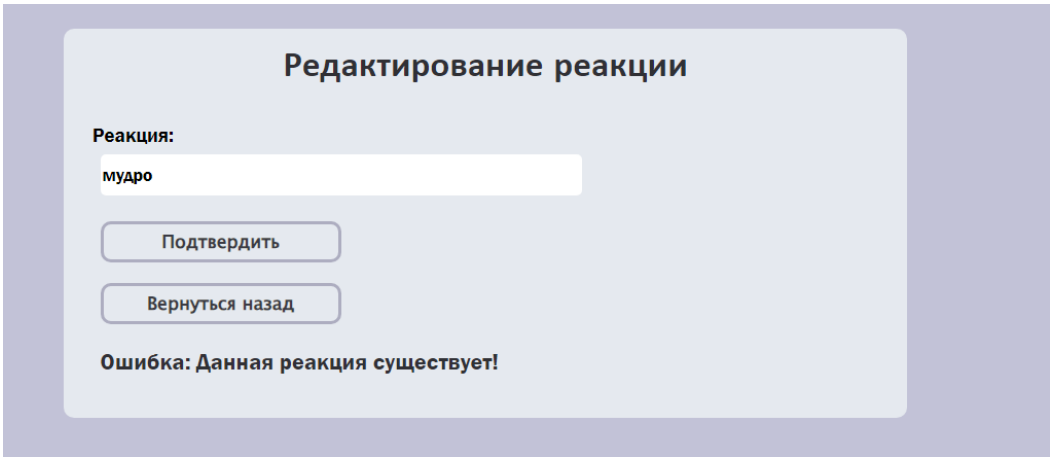


Рис. 6.8 – Ошибка при редактировании реакции

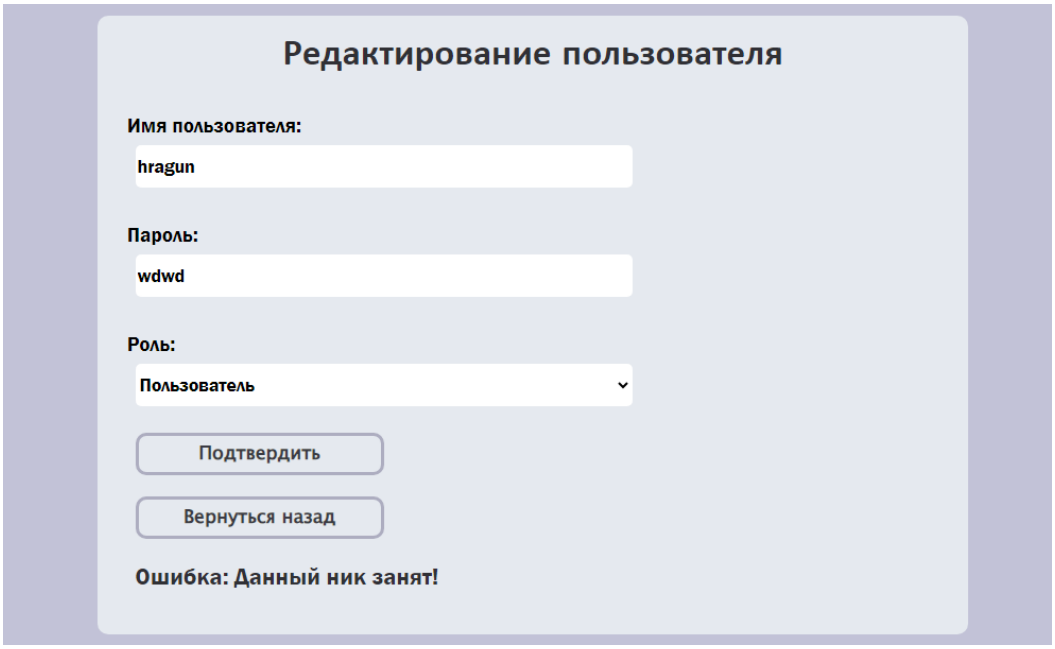


Рис. 6.9 – Ошибка при редактировании пользователя

При нажатии на кнопку «Удалить» в отображении записей пользователю выводится модальное окно с подтверждением удаления (рис. 6.10–6.12). При подтверждении выбранная запись удаляется и перезапускается страницы. При отказе модальное окно пропадает.

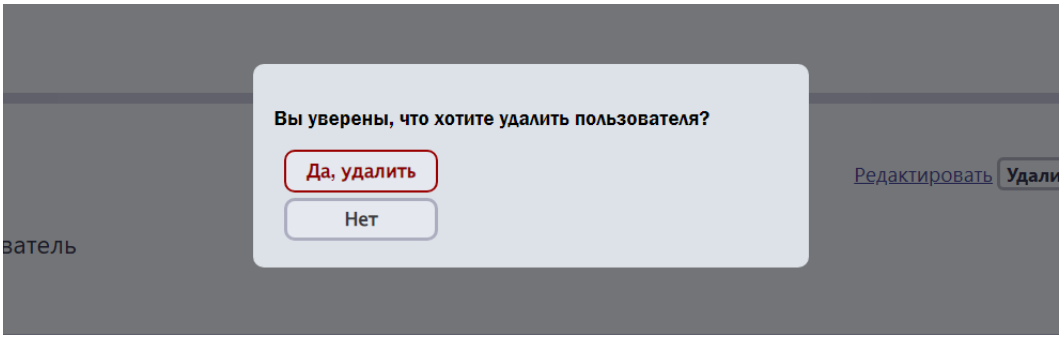


Рис. 6.10 – Удаление пользователя

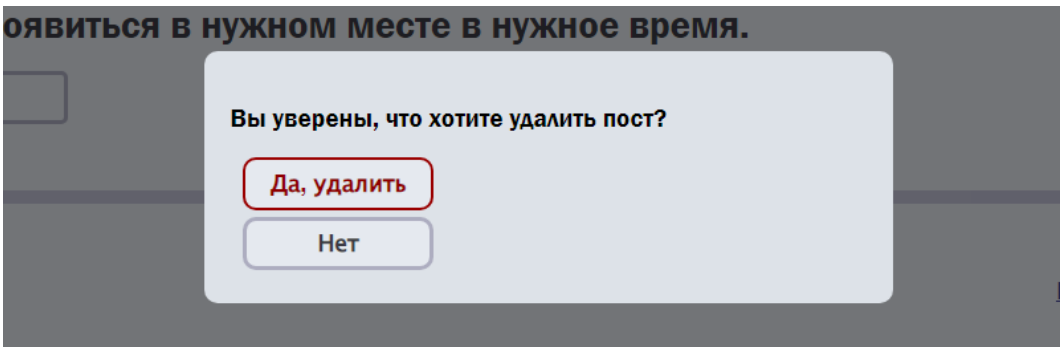


Рис. 6.11 – Удаление поста

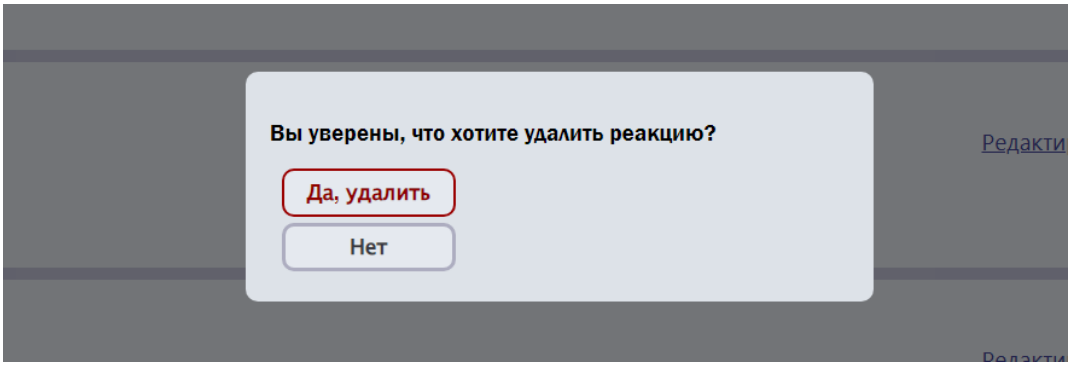


Рис. 6.12 – Удаление реакции

Для добавления новой записи пользователь выбирает в шапке нужную ему сущность, а затем его переводит на другую страницу (рис. 6.13–6.15). При подтверждении в каждой странице создании выводится надпись по том, что запись была добавлена, а в случае, если ник пользователя или название реакции уже существует, то выводится ошибка (рис 6.16–6.17).

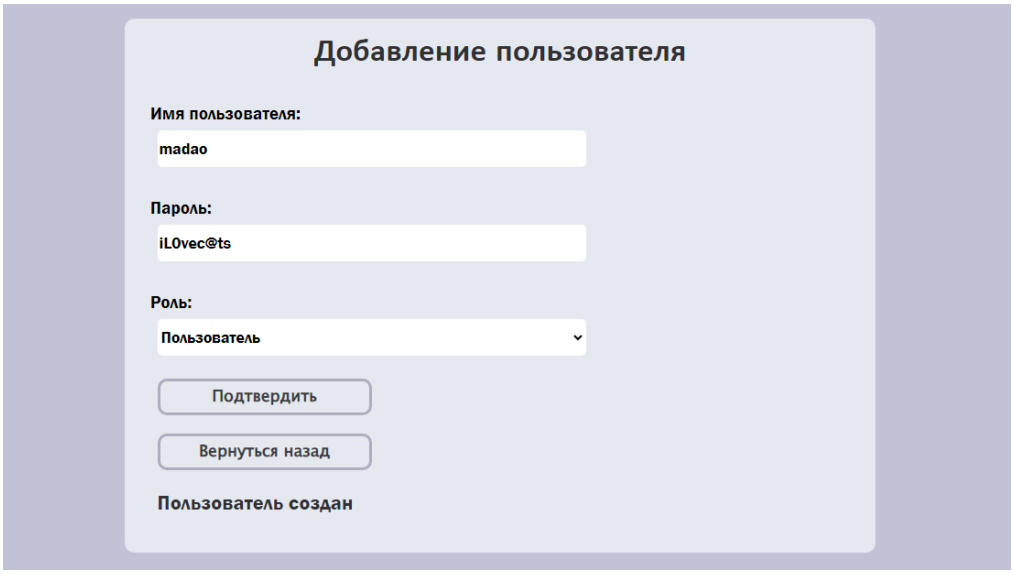


Рис. 6.13 – Добавление пользователя

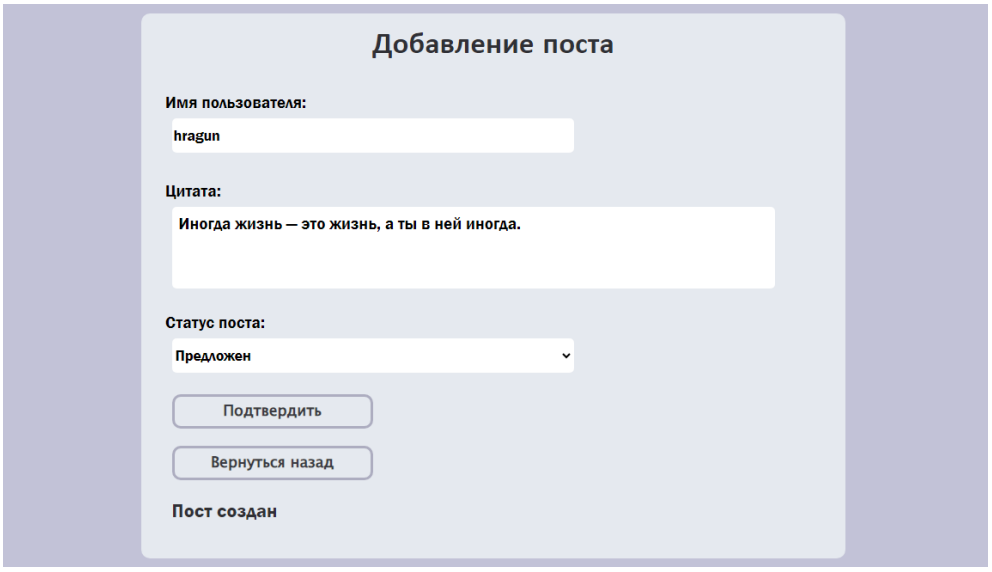


Рис. 6.14 – Добавление поста

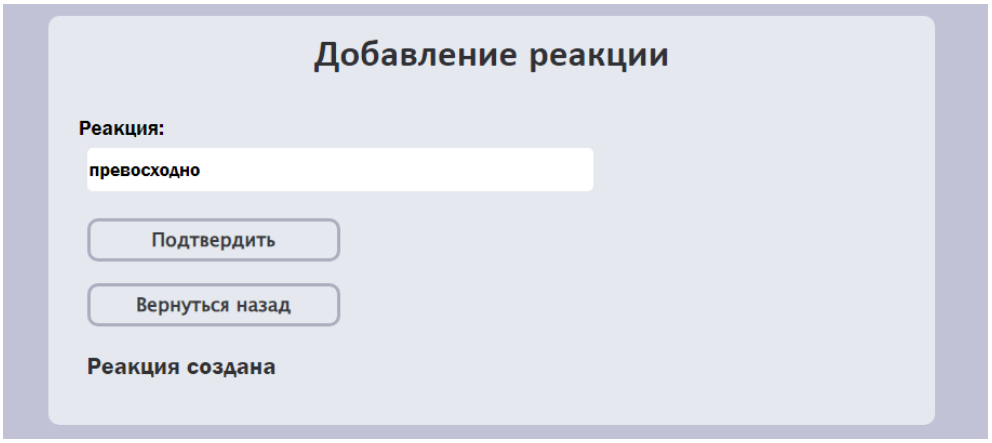


Рис. 6.15 – Добавление реакции

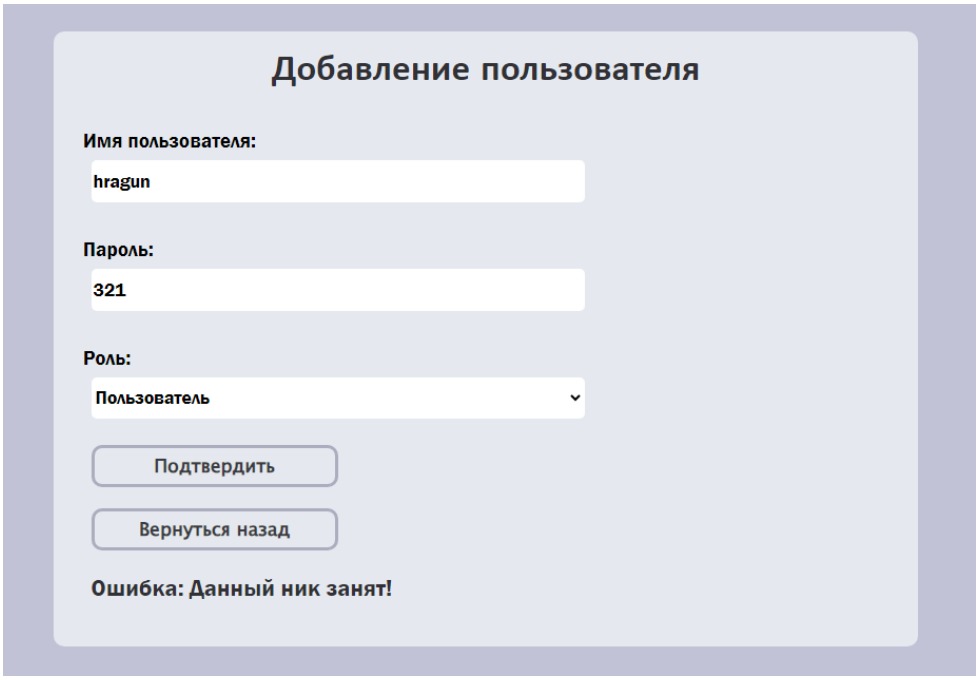


Рис. 6.16 – Ошибка при добавлении пользователя

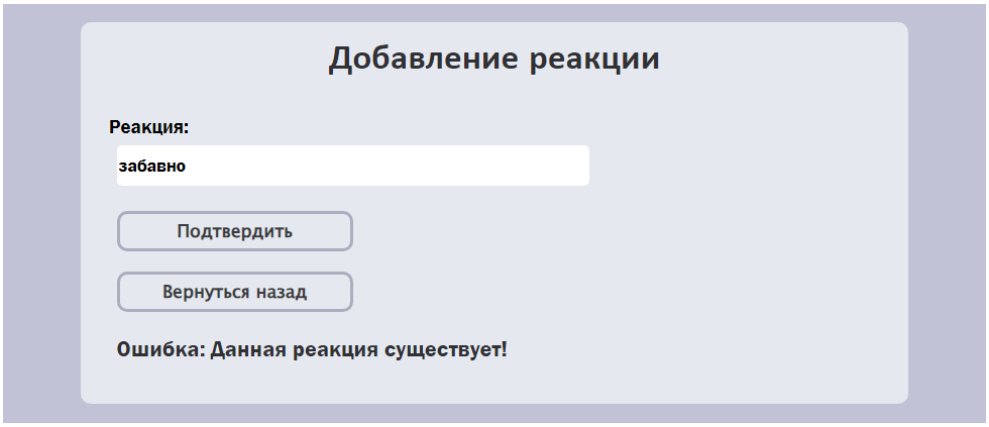


Рис. 6.17 – Ошибка при добавлении реакции

В посте можно зайти на профиль пользователя, в которой будут отображаться его статистика и его созданные посты (рис. 6.18). Данная страница создана для дальнейшей возможности добавление пользователей в приложение, у которых была бы своя страница со своими данными.

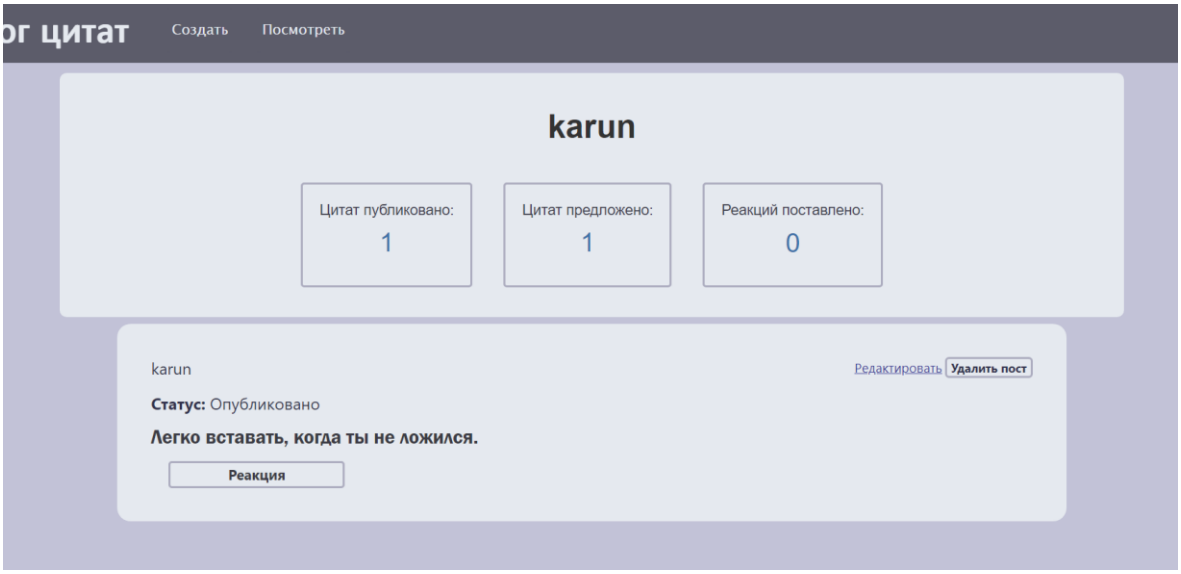


Рис. 6.18 – Профиль пользователя

# Список используемых источников

1. Фелипе Гутьеррес (Ben Frain) - "Spring Boot 2. Лучшие практики для профессионалов", Питер, 2020, 464 стр.
2. Справочник HTML и CSS [Электронный ресурс] – URL: https://hcdev.ru/ (дата обращения 9 01 2024)
3. The Modern JavaScript Tutorial [Электронный ресурс] – URL: https://javascript.info/ (дата обращения 9 01 2024)
4. Официальная документация по Spring Data JDBC [Электронный ресурс] – URL: https://docs.spring.io/spring-data/jdbc/docs/current/api/ (дата обращения 10 01 2024)
5. Использование Fetch – Интерфейсы веб API | MDN [Электронный ресурс] – URL: https://developer.mozilla.org/ru/docs/Web/API/Fetch_API/Using_Fetch (дата обращения 11 01 2024)
6. Руководство по MySQL [Электронный ресурс] – URL: https://metanit.com/sql/mysql/ (дата обращения 11 01 2024)

# Приложение «Блог цитат» - Листинг программного кода

GetMainController.java

```
package com.hragun.quotesblog.controllers;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

@Controller
public class GetMainController {
    //GetMapping
    //user
    @GetMapping("/userCreate")
    public String getUserCreate() {
        return "user/userCreate";
    }
    @GetMapping("/userPage")
    public String getUserPage() {
        return "user/userPage";
    }

    @GetMapping("/userEdit")
    public String getUserEdit() {
        return "user/userEdit";
    }

    @GetMapping("/userView")
    public String getUserView() {
        return "user/userView";
    }

    @GetMapping("/{idUser}")
    public String getAcc(@PathVariable Long idUser, Model model) {
        model.addAttribute("idUser", idUser);
        return "user/userPage";
    }

    //post
    @GetMapping("/postCreate")
    public String getCreateQuote() {
        return "post/postCreate";
    }

    @GetMapping("/")
    public String getQuoteFeed() {
        return "post/postMain";
    }

    @GetMapping("/postView")
    public String getPostView() {
        return "post/postView";
    }

    @GetMapping("/postEdit")
    public String getPostEdit() {
        return "post/postEdit";
    }
    //reaction
    @GetMapping("/reactnCreate")
    public String getReactnCreate() {
```

```
        return "reaction/reactnCreate";
    }
    @GetMapping("/reactnView")
    public String getReactnView() {
        return "reaction/reactnView";
    }
    @GetMapping("/reactionEdit")
    public String getReactionEdit() {
        return "reaction/reactnEdit";
    }
}
```

PostRestController.java

```
package com.hragun.quotesblog.controllers;

import com.hragun.quotesblog.DTO.PostCreateDTO;
import com.hragun.quotesblog.DTO.PostDTO;
import com.hragun.quotesblog.models.Post;
import com.hragun.quotesblog.repository.PostRepository;
import com.hragun.quotesblog.repository.UserRepository;
import com.hragun.quotesblog.services.PostService;
import jakarta.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;


@RestController
public class PostRestController {

    private final PostService postService;
    private final PostRepository postRepository;
    private final UserRepository userRepository;

    @Autowired
    private HttpSession session;
    @Autowired
    public PostRestController(PostService postService, PostRepository
postRepository, UserRepository userRepository) {
        this.postService = postService;
        this.postRepository = postRepository;
        this.userRepository = userRepository;
    }

    //Вывод в профиле
    @GetMapping("/fetch/user/{idUser}")
    public Iterable<PostDTO> getProfilePost(@PathVariable Long idUser) {
        return postService.getAllPosts(idUser);
    }

    @PostMapping("/fetch/post/create")
    public ResponseEntity<String> createPost(@RequestBody PostCreateDTO
postDTO) {
        try {
            Post post = new
Post(userRepository.findIdUserByNick(postDTO.getUsername()),
postDTO.getTextPost(), postDTO.getStatusPost());
            postRepository.save(post);
            return ResponseEntity.ok("Пост успешно создан");
        } catch (Exception e) {
            return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Ошибка при
```

```
создании поста");
        }
    }


    @GetMapping("/fetch/posts/all")
    public Iterable<PostDTO> getAllPosts() {
        return postService.getAllPosts();
    }
    @GetMapping("/fetch/posts/published")
    public Iterable<PostDTO> getPublishedPosts() {
        return postService.getPublishedPosts();
    }
    @PostMapping("/fetch/post/find")
    public ResponseEntity<String> savePost(@RequestBody Post post) {
        session.setAttribute("id", post.getIdPost());
        return ResponseEntity.ok("ID успешно сохранен в сессии");
    }

    @GetMapping("/fetch/post/load")
    public ResponseEntity<PostDTO> loadPost() {
        Long id = (Long) session.getAttribute("id");
        if (id != null) {
            PostDTO loadedReaction = new PostDTO();
            Post post=postRepository.findPostById(id);

loadedReaction.setUsername(userRepository.findNick(post.getIdUser()));
            loadedReaction.setTextPost(post.getTextPost());
            loadedReaction.setStatusPost(post.getStatusPost());
            return ResponseEntity.ok(loadedReaction);
        } else {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(null);
        }
    }

    @PostMapping("/fetch/post/delete")
    public void delPost(@RequestBody Post post) {
        postRepository.delPost(post.getIdPost());
    }

    @PostMapping("/fetch/post/edit")
    public void editPost(@RequestBody PostDTO postDTO) {

postRepository.editPost(postDTO.getIdPost(),userRepository.findIdUserByNick(p
ostDTO.getUsername()),postDTO.getTextPost(),postDTO.getStatusPost());
    }
}
```

ReactionRestController.java

```
package com.hragun.quotesblog.controllers;

import com.hragun.quotesblog.models.Reaction;
import com.hragun.quotesblog.repository.ReactionRepository;
import jakarta.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.dao.DataIntegrityViolationException;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;
```

```
@RestController
public class ReactionRestController {
    public final ReactionRepository reactionRepository;
    @Autowired
    private HttpSession session;

    @Autowired
    public ReactionRestController(ReactionRepository reactionRepository) {
        this.reactionRepository = reactionRepository;
    }

    @GetMapping("/fetch/reactions/all")
    public Iterable<Reaction> getAllReactions() {
        return reactionRepository.findAll();
    }

    @PostMapping("/fetch/reaction/create")
    public ResponseEntity<String> createReaction(@RequestBody Reaction
reaction) {
        try {
            reactionRepository.save(reaction);
            return ResponseEntity.ok("Реакция успешно создана");
        }catch (DataIntegrityViolationException e) {
            return ResponseEntity.status(HttpStatus.CONFLICT).body("Такая
реакция существует");
        } catch (Exception e) {
            return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Ошибка при
создании поста");
        }
    }
    @PostMapping("/fetch/reaction/find")
    public ResponseEntity<String> findReaction(@RequestBody Reaction
reaction) {
        session.setAttribute("id", reaction.getIdReaction());
        return ResponseEntity.ok("ID успешно сохранен в сессии");
    }
    @GetMapping("/fetch/reaction/load")
    public ResponseEntity<Reaction> loadReaction() {
        Long id = (Long) session.getAttribute("id");
        if (id != null) {
            Reaction loadedReaction =
reactionRepository.findReactionById(id);
            return ResponseEntity.ok(loadedReaction);
        } else {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(null);
        }
    }
    @PostMapping("/fetch/reaction/delete")
    public void delReaction(@RequestBody Reaction reaction) {
        reactionRepository.delReactn(reaction.getIdReaction());
    }
    @PostMapping("/fetch/reaction/edit")
    public void editReaction(@RequestBody Reaction reaction) {
        reactionRepository.editReactn(reaction.getIdReaction(),
reaction.getNameReaction());
    }
}
```

UserRestController.java

```
package com.hragun.quotesblog.controllers;
```

```java
import com.hragun.quotesblog.DTO.UserAccDTO;
import com.hragun.quotesblog.models.User;
import com.hragun.quotesblog.repository.UserRepository;
import com.hragun.quotesblog.services.UserAccService;
import jakarta.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.dao.DataIntegrityViolationException;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;


@RestController
public class UserRestController {
    private final UserAccService userAccService;
    private final UserRepository userRepository;
    @Autowired
    private HttpSession session;

    @Autowired
    public UserRestController(UserAccService userAccService, UserRepository
userRepository) {
        this.userAccService = userAccService;
        this.userRepository = userRepository;
    }

    @GetMapping("/fetch/users/all")
    public Iterable<User> getAllUsers() {
        return userRepository.findAll();
    }

    //Вывод данных в профиле
    @GetMapping("/fetch/{idUser}/info")
    public UserAccDTO infoUser(@PathVariable Long idUser) {
        return userAccService.getAccInfo(idUser);
    }

    @PostMapping("/fetch/user/create")
    public ResponseEntity<String> createUser(@RequestBody User user) {
        try {
            userRepository.save(user);
            return ResponseEntity.ok("Пользователь успешно создан");
        } catch (DataIntegrityViolationException e) {
            return ResponseEntity.status(HttpStatus.CONFLICT).body("Такой ник
занят");
        } catch (Exception e) {
            return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Ошибка при
создании пользователя");
        }
    }

    @PostMapping("/fetch/user/find")
    public ResponseEntity<String> findUser(@RequestBody User user) {
        session.setAttribute("id", user.getIdUser());
        return ResponseEntity.ok("ID успешно сохранен в сессии");
    }

    @GetMapping("/fetch/user/load")
    public ResponseEntity<User> loadUser() {
        Long id = (Long) session.getAttribute("id");
        if (id != null) {
            User loadedReaction = userRepository.findUserById(id);
```

```
            return ResponseEntity.ok(loadedReaction);
        } else {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(null);
        }
    }

    @PostMapping("/fetch/user/delete")
    public void delUser(@RequestBody User user) {
        userRepository.delUser(user.getIdUser());
    }

    @PostMapping("/fetch/user/edit")
    public ResponseEntity<String> editUser(@RequestBody User user) {
        try{
            userRepository.editUser(user.getIdUser(), user.getUsername(),
user.getPassword(), user.getRoleUser());
            return ResponseEntity.ok("Пользователь успешно изменен");
        }catch (DataIntegrityViolationException e) {
            return ResponseEntity.status(HttpStatus.CONFLICT).body("Такой ник
занят");
        } catch (Exception e) {
            return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Ошибка при
редактировании пользователя");
        }
    }
}
```

PostCreateDTO.java

```
package com.hragun.quotesblog.DTO;

public class PostCreateDTO {
    private String username;
    private String textPost;
    private String statusPost;

    public String getStatusPost() {
        return statusPost;
    }

    public void setStatusPost(String statusPost) {
        this.statusPost = statusPost;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getTextPost() {
        return textPost;
    }

    public void setTextPost(String textPost) {
        this.textPost = textPost;
    }

}
```

PostDTO.java

```
package com.hragun.quotesblog.DTO;

public class PostDTO {
    private Long idPost;
    private Long idUser;
    private String username;
    private String textPost;
    private String statusPost;


    public PostDTO() {
    }



    public Long getIdPost() {
        return idPost;
    }

    public void setIdPost(Long idPost) {
        this.idPost = idPost;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getTextPost() {
        return textPost;
    }

    public void setTextPost(String textPost) {
        this.textPost = textPost;
    }

    public Long getIdUser() {
        return idUser;
    }

    public void setIdUser(Long idUser) {
        this.idUser = idUser;
    }

    public String getStatusPost() {
        return statusPost;
    }

    public void setStatusPost(String statusPost) {
        this.statusPost = statusPost;
    }
}
```

UserAccDTO.java

```
package com.hragun.quotesblog.DTO;

public class UserAccDTO {
    private String nickUser;
```

```
    private int publicPosts;
    private int reactionPosts;
    private int offeredPosts;

    public String getNickUser() {
        return nickUser;
    }

    public void setNickUser(String nickUser) {
        this.nickUser = nickUser;
    }

    public int getReactionPosts() {
        return reactionPosts;
    }

    public void setReactionPosts(int reactionPosts) {
        this.reactionPosts = reactionPosts;
    }

    public int getPublicPosts() {
        return publicPosts;
    }

    public void setPublicPosts(int publicPosts) {
        this.publicPosts = publicPosts;
    }

    public int getOfferedPosts() {
        return offeredPosts;
    }

    public void setOfferedPosts(int offeredPosts) {
        this.offeredPosts = offeredPosts;
    }
}
```

Post.java

```
package com.hragun.quotesblog.models;


import org.springframework.data.annotation.Id;
import org.springframework.data.relational.core.mapping.Column;
import org.springframework.data.relational.core.mapping.Table;


@Table("posts")
public class Post {
    @Id
    @Column("idPost")
    public Long idPost;
    @Column("idUser")
    private Long idUser;
    @Column("TextPost")
    private String textPost;

    @Column("StatusPost")
    private String statusPost;

    public Post() {
    }

    public Post(Long idPost, Long idUser, String textPost, String statusPost)
```

```
{
        this.idPost = idPost;
        this.idUser = idUser;
        this.textPost = textPost;
        this.statusPost = statusPost;
    }
    public Post(Long idUser, String textPost, String statusPost) {
        this.idUser = idUser;
        this.textPost = textPost;
        this.statusPost= statusPost;
    }

    public String getStatusPost() {
        return statusPost;
    }

    public void setStatusPost(String statusPost) {
        this.statusPost = statusPost;
    }

    public Long getIdPost() {
        return idPost;
    }

    public void setIdPost(Long idPost) {
        this.idPost = idPost;
    }

    public Long getIdUser() {
        return idUser;
    }

    public void setIdUser(Long idUser) {
        this.idUser = idUser;
    }

    public String getTextPost() {
        return textPost;
    }

    public void setTextPost(String textPost) {
        this.textPost = textPost;
    }

}
```

Reaction.java

```
package com.hragun.quotesblog.models;

import org.springframework.data.annotation.Id;
import org.springframework.data.relational.core.mapping.Column;
import org.springframework.data.relational.core.mapping.Table;

@Table("reactions")
public class Reaction {
    @Id
    @Column("idReaction")
    private Long idReaction;
    @Column("nameReaction")
    private String nameReaction;

    public Reaction(){
```

```
    }
    public Reaction(Long idReaction, String nameReaction) {
        this.idReaction = idReaction;
        this.nameReaction = nameReaction;
    }

    public Long getIdReaction() {
        return idReaction;
    }

    public void setIdReaction(Long idReaction) {
        this.idReaction = idReaction;
    }

    public String getNameReaction() {
        return nameReaction;
    }

    public void setNameReaction(String nameReaction) {
        this.nameReaction = nameReaction;
    }
}
```

User.java

```
package com.hragun.quotesblog.models;

import org.springframework.data.annotation.Id;
import org.springframework.data.relational.core.mapping.Column;
import org.springframework.data.relational.core.mapping.Table;


@Table("users")
public class User {
    @Id
    @Column("idUser")
    private Long idUser;
    @Column("NickUser")
    private String username;
    @Column("PasswordUser")
    private String password;
    @Column("RoleUser")
    private String roleUser;

    public Long getIdUser() {
        return idUser;
    }

    public void setIdUser(Long idUser) {
        this.idUser = idUser;
    }


    public void setUsername(String username) {
        this.username = username;
    }


    public void setPassword(String password) {
        this.password = password;
    }

    public String getRoleUser() {
        return roleUser;
```

```
    }

    public void setRoleUser(String roleUser) {
        this.roleUser = roleUser;
    }

    public String getPassword() {
        return this.password;
    }

    public String getUsername() {
        return this.username;
    }

    public User(Long idUser, String username, String password, String
roleUser) {
        this.idUser = idUser;
        this.username = username;
        this.password = password;
        this.roleUser = roleUser;
    }

    public User(String username, String password) {
        this.username = username;
        this.password = password;
    }
    public User(String username, String password,String roleUser) {
        this.username = username;
        this.password = password;
        this.roleUser = roleUser;
    }
    public User() {
    }
}
```

PostRepository.java

```
package com.hragun.quotesblog.repository;

import com.hragun.quotesblog.models.Post;
import org.springframework.data.jdbc.repository.query.Modifying;
import org.springframework.data.jdbc.repository.query.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
@Repository
public interface PostRepository extends CrudRepository<Post, Long> {
    @Query("SELECT * FROM quotes_blog.posts p WHERE p.idUser=:idUser AND
p.StatusPost='published'")
    List<Post> findPost(Long idUser);
    @Query("SELECT * FROM quotes_blog.posts ORDER BY idPost DESC")
    List<Post> findAllPosts();

    @Query("SELECT * FROM quotes_blog.posts p WHERE p.statusPost='published'
ORDER BY p.idPost DESC")
    List<Post> findPublishedPosts();

    @Query("SELECT * FROM quotes_blog.posts p WHERE p.idPost=:idPost")
    Post findPostById(Long idPost);

    @Modifying
    @Query("DELETE FROM quotes_blog.posts p WHERE p.idPost = :idPost")
    int delPost(Long idPost);
```

```
    @Modifying
    @Query("UPDATE quotes_blog.posts AS p SET p.idUser=:idUser,
p.TextPost=:TextPost, p.StatusPost=:StatusPost WHERE idPost=:idPost")
    int editPost(Long idPost, Long idUser, String TextPost, String
StatusPost);
}
```

ReactionRepository.java

```
package com.hragun.quotesblog.repository;

import com.hragun.quotesblog.models.Reaction;
import org.springframework.data.jdbc.repository.query.Modifying;
import org.springframework.data.jdbc.repository.query.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface ReactionRepository extends CrudRepository<Reaction, Long> {
    @Query("SELECT * FROM quotes_blog.reactions r WHERE
r.idReaction=:idReaction")
    Reaction findReactionById(Long idReaction);
    @Modifying
    @Query("DELETE FROM quotes_blog.reactions r WHERE r.idReaction =
:idReaction")
    int delReactn(Long idReaction);
    @Modifying
    @Query("UPDATE quotes_blog.reactions AS r SET
r.nameReaction=:nameReaction WHERE r.idReaction =:idReaction")
    int editReactn(Long idReaction, String nameReaction);
}
```

UserRepository.java

```
package com.hragun.quotesblog.repository;

import com.hragun.quotesblog.models.User;
import org.springframework.data.jdbc.repository.query.Modifying;
import org.springframework.data.jdbc.repository.query.Query;
import org.springframework.data.repository.CrudRepository;

import java.util.List;

public interface UserRepository extends CrudRepository<User, Long> {
    @Query("SELECT * FROM quotes_blog.users WHERE nickUser = :nickUser AND
passwordUser = :passwordUser")
    User findNickAndPassUser(String nickUser, String passwordUser);

    @Query("SELECT * FROM quotes_blog.users WHERE NickUser =:NickUser")
    User findNickUser(String NickUser);

    @Query("SELECT * FROM quotes_blog.users")
    List<User> findAllUsers();
    @Query("SELECT u.idUser FROM quotes_blog.users u WHERE
u.NickUser=:NickUser")
    Long findIdUserByNick(String NickUser);

    @Modifying
    @Query("DELETE FROM quotes_blog.users u WHERE u.idUser = :idUser")
    int delUser(Long idUser);


    @Query("SELECT * FROM quotes_blog.users u WHERE u.idUser=:idUser")
    User findUserById(Long idUser);
```

```
    @Modifying
    @Query("UPDATE quotes_blog.users AS u SET u.NickUser=:username,
u.PasswordUser=:password, u.RoleUser=:roleUser WHERE u.idUser=:idUser")
    int editUser(Long idUser, String username, String password, String
roleUser);

    //Для статистики профиля
    @Query("SELECT u.NickUser FROM quotes_blog.users u WHERE u.idUser =
:idUser")
    String findNick(Long idUser);

    @Query("SELECT COUNT(DISTINCT CASE WHEN p.StatusPost = 'published' THEN
p.idPost ELSE NULL END) AS 'publicPosts' FROM quotes_blog.users u LEFT JOIN
quotes_blog.posts p ON u.idUser = p.idUser WHERE u.idUser=:idUser")
    int findPublishedPosts(Long idUser);

    @Query("SELECT COUNT(DISTINCT r.idReaction) AS 'reactionPosts' FROM
quotes_blog.users u LEFT JOIN quotes_blog.posts_reactions_users r ON u.idUser
= r.idUser WHERE u.idUser=:idUser")
    int findReactionPosts(Long idUser);

    @Query("SELECT COUNT(DISTINCT CASE WHEN p.StatusPost = 'offered' THEN
p.idPost ELSE NULL END) AS 'offeredPosts' FROM quotes_blog.users u LEFT JOIN
quotes_blog.posts p ON u.idUser = p.idUser WHERE u.idUser=:idUser")
    int findOfferedPosts(Long idUser);
}
```

PostService.java

```
package com.hragun.quotesblog.services;

import com.hragun.quotesblog.DTO.PostDTO;
import com.hragun.quotesblog.models.Post;
import com.hragun.quotesblog.models.User;
import com.hragun.quotesblog.repository.PostRepository;
import com.hragun.quotesblog.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;

@Service
public class PostService {

    private final PostRepository postRepository;
    private final UserRepository userRepository;

    @Autowired
    public PostService(PostRepository postRepository, UserRepository
userRepository) {
        this.postRepository = postRepository;
        this.userRepository = userRepository;
    }

    public Iterable<PostDTO> getAllPosts() {
        List<PostDTO> postDtos = new ArrayList<>();
        Iterable<Post> posts = postRepository.findAllPosts();
        createPost(postDtos, posts);
        return postDtos;
    }
    public Iterable<PostDTO> getPublishedPosts() {
        List<PostDTO> postDtos = new ArrayList<>();
        Iterable<Post> posts = postRepository.findPublishedPosts();
```

```
        createPost(postDtos, posts);
        return postDtos;
    }

    public Iterable<PostDTO> getAllPosts(Long idUser) {
        List<PostDTO> postDtos = new ArrayList<>();
        Iterable<Post> posts = postRepository.findPost(idUser);
        createPost(postDtos, posts);
        return postDtos;
    }

    private void createPost(List<PostDTO> postDtos, Iterable<Post> posts) {
        for (Post post : posts) {
            User user =
userRepository.findById(post.getIdUser()).orElse(null);
            PostDTO postDto = new PostDTO();
            postDto.setIdPost(post.getIdPost());
            postDto.setIdUser(post.getIdUser());
            postDto.setUsername(user != null ? user.getUsername() : null);
            postDto.setTextPost(post.getTextPost());
            postDto.setStatusPost(post.getStatusPost());
            postDtos.add(postDto);
        }
    }
}
```

UserAccService.java

```
package com.hragun.quotesblog.services;

import com.hragun.quotesblog.DTO.UserAccDTO;
import com.hragun.quotesblog.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class UserAccService {

    @Autowired
    private UserRepository usrAccRepstry;


    public UserAccDTO getAccInfo(Long idUser){
        UserAccDTO userAccDTO=new UserAccDTO();
        userAccDTO.setNickUser(usrAccRepstry.findNick(idUser));
        userAccDTO.setPublicPosts(usrAccRepstry.findPublishedPosts(idUser));
        userAccDTO.setReactionPosts(usrAccRepstry.findReactionPosts(idUser));
        userAccDTO.setOfferedPosts(usrAccRepstry.findOfferedPosts(idUser));
        return userAccDTO;
    }

}
```

QuotesBlogApplication.java

```
package com.hragun.quotesblog;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class QuotesBlogApplication {

    public static void main(String[] args) {
       SpringApplication.run(QuotesBlogApplication.class, args);
```

```
    }

}
```

header.html

```
<html xmlns:th="http://www.thymeleaf.org">

<body>
    <div th:fragment="headerFragment" class="header-container">
        <header>
            <link rel="stylesheet" href="common/styles/headerStyle.css">
            <a href="/" id="name">Блог цитат</a>
            <div class="dropdown">
                <button>Создать</button>
                <div class="dropdown-content">
                    <a href="/userCreate">Пользователя</a>
                    <a href="/postCreate">Пост</a>
                    <a href="/reactnCreate">Реакцию</a>
                </div>
            </div>
            <div class="dropdown">
                <button>Посмотреть</button>
                <div class="dropdown-content">
                    <a href="/userView">Пользователей</a>
                    <a href="/postView">Посты</a>
                    <a href="/reactnView">Реакции</a>
                </div>
        </header>
</body>

</html>
```

postCreate.html

```
<html xmlns:th="http://www.thymeleaf.org">

<head>
    <title>Предложить цитату</title>
    <link rel="stylesheet" href="../common/styles/headerStyle.css">
    <link rel="stylesheet" href="../common/styles/createAndEditStyle.css">
</head>

<body>
    <div>
    </div>
    <br>
    <div class="create-or-edit">
        <h2>Добавление поста</h2>
        <div>
            <label for="username">Имя пользователя:</label>
            <input type="text" name="username" id="username">
        </div>
        <div>
            <label for="text">Цитата:</label>
            <textarea name="textPost" id="textPost"></textarea>
        </div>
        <div>
            <label for="role">Статус поста:</label>
            <select id="statusPost" name="statusPost">
                <option value="published">Опубликован</option>
                <option value="offered">Предложен</option>
            </select>
        </div>
        <div class="conf">
```

```
            <button onclick="submitForm()"
id="uploadButton">Подтвердить</button>
        </div>
        <div>
            <button id="back-button">Вернуться назад</button>
        </div>
        <div id="msgContainer" class="msgContainer">
            <p id="message"></p>
        </div>
    </div>
    <script src="common/scripts/post/postCreateScript.js"></script>
    <script>
        const backButton = document.getElementById('back-button');
        backButton.addEventListener('click', function () {
            window.history.back();
        });
    </script>
    <th:block th:replace="~{header :: headerFragment}"></th:block>
</body>

</html>
```

postEdit.html

```
<html xmlns:th="http://www.thymeleaf.org">


<head>
    <title>Предложить цитату</title>
    <link rel="stylesheet" href="../common/styles/headerStyle.css">
    <link rel="stylesheet" href="../common/styles/createAndEditStyle.css">
</head>

<body>
    <div th:fragment="headerFragment" class="header-container">
    </div>
    <br>

    <div class="create-or-edit">
        <h2>Редактирование поста</h2>
        <div>
            <label for="username">Имя пользователя:</label>
            <input type="text" name="username" id="username">
        </div>
        <div>
            <label for="text">Цитата и фото:</label>
            <textarea name="textPost" id="textPost"></textarea>
        </div>
        <div>
            <label for="role">Статус поста:</label>
            <select id="statusPost" name="statusPost">
                <option value="published">Опубликовано</option>
                <option value="offered">Предложено</option>
            </select>
        </div>
        <div class="button-container">
            <div class="conf">
                <button onclick="confEdit()">Подтвердить</button>
            </div>
            <div>
                <button id="back-button">Вернуться назад</button>
            </div>
        </div>
    </div>
```

```
    </div>

    <script>
        const backButton = document.getElementById('back-button');
        backButton.addEventListener('click', function () {
            localStorage.removeItem('idPost');
            window.history.back();
        });
    </script>
    <script src="common/scripts/post/postEditScript.js"></script>
    <script src="common/scripts/post/postLoadScript.js"></script>
    <th:block th:replace="~{header :: headerFragment}"></th:block>
</body>

</html>
```

postMain.html

```
<html xmlns:th="http://www.thymeleaf.org">

<head>
    <title>Лента цитат</title>
    <link rel="stylesheet" href="common/styles/postStyle.css">
    <link rel="stylesheet" href="../common/styles/postStyle.css">
</head>

<body>
    <div th:fragment="headerFragment" class="header-container">
    </div>
    <br>
    <div class="posts" id="posts-container"></div>
    <div class="modal" id="deleteAccountModal">
        <div class="modal-content">
            <p id="modal-msg">Вы уверены, что хотите удалить пост?</p>
            <div class="buttons">
                <div class="delete-account">
                    <button onclick="deleteAccount()">Да, удалить</button>
                </div>
                <div class="conf">
                    <button onclick="delCancel()">Нет</button>
                </div>
            </div>
        </div>
    </div>
    <script type="module"
src="common/scripts/post/publishedPostScript.js"></script>
    <script src="common/scripts/post/postEditScript.js"></script>
    <script src="common/scripts/post/postDelScript.js"></script>
    <th:block th:replace="~{header :: headerFragment}"></th:block>
</body>

</html>
```

postView.html

```
<html xmlns:th="http://www.thymeleaf.org">

<head>
    <title>Лента цитат</title>
    <link rel="stylesheet" href="common/styles/postStyle.css">
    <link rel="stylesheet" href="../common/styles/postStyle.css">
</head>

<body>
    <div th:fragment="headerFragment" class="header-container">
```

```
    </div>
    <br>
    <div class="posts" id="posts-container"></div>
    <div class="modal" id="deleteAccountModal">
        <div class="modal-content">
            <p id="modal-msg">Вы уверены, что хотите удалить пост?</p>
            <div class="buttons">
                <div class="delete-account">
                    <button onclick="deleteAccount()">Да, удалить</button>
                </div>
                <div class="conf">
                    <button onclick="delCancel()">Нет</button>
                </div>
            </div>
        </div>
    </div>
    <script type="module"
src="common/scripts/post/allPostScript.js"></script>
    <script src="common/scripts/post/postEditScript.js"></script>
    <script src="common/scripts/post/postDelScript.js"></script>
    <th:block th:replace="~{header :: headerFragment}"></th:block>
</body>

</html>
```

reactionCreate.html

```
<html xmlns:th="http://www.thymeleaf.org">

<head>
    <title>Предложить цитату</title>
    <link rel="stylesheet" href="../common/styles/headerStyle.css">
    <link rel="stylesheet" href="../common/styles/createAndEditStyle.css">
</head>

<body>
    <div>
    </div>
    <br>
    <div class="create-or-edit">
        <h2>Добавление реакции</h2>
        <div>
            <label for="nameReaction">Реакция:</label>
            <input type="text" name="nameReaction" id="nameReaction">
        </div>
        <div>
            <div class="conf">
                <button onclick="submitForm()"
id="uploadButton">Подтвердить</button>
            </div>
            <div>
                <button id="back-button">Вернуться назад</button>
            </div>
            <div id="msgContainer" class="msgContainer">
                <p id="message"></p>
            </div>
        </div>
        <script
src="common/scripts/reaction/reactionCreateScript.js"></script>
        <script>
            const backButton = document.getElementById('back-button');
            backButton.addEventListener('click', function () {
                window.history.back();
            });
```

```
        </script>
    </div>
    <th:block th:replace="~{header :: headerFragment}"></th:block>
</body>

</html>
```

reactionEdit.html

```
<html xmlns:th="http://www.thymeleaf.org">


<head>
    <title>Предложить цитату</title>
    <link rel="stylesheet" href="../common/styles/headerStyle.css">
    <link rel="stylesheet" href="../common/styles/createAndEditStyle.css">
</head>

<body>
    <div th:fragment="headerFragment" class="header-container">
    </div>
    <br>
    <div class="create-or-edit">
        <h2>Редактирование реакции</h2>
        <div>
            <label for="nameReaction">Реакция:</label>
            <input type="text" name="nameReaction" id="nameReaction">
        </div>
        <div>
            <div class="button-container">
                <div class="conf">
                    <button onclick="confEdit()">Подтвердить</button>
                </div>
                <div>
                    <button id="back-button">Вернуться назад</button>
                </div>
            </div>
        </div>
        <div id="msgContainer" class="msgContainer">
            <p id="message"></p>
        </div>
    </div>
    <script>
        const backButton = document.getElementById('back-button');
        backButton.addEventListener('click', function () {
            localStorage.removeItem('idReaction');
            window.history.back();
        });
    </script>
    <script src="common/scripts/reaction/reactionEditScript.js"></script>
    <script src="common/scripts/reaction/reactionLoadScript.js"></script>
    <th:block th:replace="~{header :: headerFragment}"></th:block>
</body>

</html>
```

reactionView.html

```
<html xmlns:th="http://www.thymeleaf.org">

<head>
    <title>Лента реакций</title>
    <link rel="stylesheet" href="common/styles/postStyle.css">
    <link rel="stylesheet" href="../common/styles/postStyle.css">
</head>
```

```
<body>
    <div th:fragment="headerFragment" class="header-container">
    </div>
    <br>
    <div class="posts" id="posts-container"></div>

    <div class="modal" id="deleteAccountModal">
        <div class="modal-content">
            <p id="modal-msg">Вы уверены, что хотите удалить реакцию?</p>
            <div class="buttons">
                <div class="delete-account">
                    <button onclick="deleteAccount()">Да, удалить</button>
                </div>
                <div class="conf">
                    <button onclick="delCancel()">Нет</button>
                </div>
            </div>
        </div>
    </div>
    <script type="module"
src="common/scripts/reaction/allReactionScript.js"></script>
    <script src="common/scripts/reaction/reactionDelScript.js"></script>
    <script src="common/scripts/reaction/reactionEditScript.js"></script>
    <th:block th:replace="~{header :: headerFragment}"></th:block>
</body>

</html>
```

userCreate.html

```
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Добавление пользователя</title>
    <link rel="stylesheet" href="../common/styles/createAndEditStyle.css">
</head>
<body>
    <div th:fragment="headerFragment" class="header-container">
    </div>
    <br>
    <div class="create-or-edit">
        <h2>Добавление пользователя</h2>
        <div>
            <label for="username">Имя пользователя:</label>
            <input type="text" name="username" id="username">
        </div>
        <div>
            <label for="password">Пароль:</label>
            <input type="text" name="password" id="password">
        </div>
        <div>
            <label for="role">Роль:</label>
            <select id="role" name="role">
                <option value="user">Пользователь</option>
                <option value="admin">Админ</option>
            </select>
        </div>
        <div class="conf">
            <button type="submit" onclick="submitForm()">Подтвердить</button>
        </div>
        <div>
            <button id="back-button">Вернуться назад</button>
        </div>
        <div id="msgContainer" class="msgContainer">
```

```
            <p id="message"></p>
        </div>
    </div>
    <th:block th:replace="~{header :: headerFragment}"></th:block>
    <script src="common/scripts/user/userCreateScript.js"></script>
    <script>
        const backButton = document.getElementById('back-button');
        backButton.addEventListener('click', function () {
            window.history.back();
        });
    </script>
</body>
</html>
```

userEdit.html

```
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Редактирование пользователя</title>
    <link rel="stylesheet" href="../common/styles/createAndEditStyle.css">
    <link rel="stylesheet" href="../common/styles/headerStyle.css">
</head>
<body>
    <div th:fragment="headerFragment" class="header-container">
    </div>
    <br>
    <div class="create-or-edit">
        <h2>Редактирование пользователя</h2>
        <div>
            <label for="username">Имя пользователя:</label>
            <input type="text" name="username" id="username">
        </div>
        <div>
            <label for="password">Пароль:</label>
            <input type="text" name="password" id="password">
        </div>
        <div>
            <label for="role">Роль:</label>
            <select id="role" name="role">
                <option value="user">Пользователь</option>
                <option value="admin">Админ</option>
            </select>
        </div>
        <div class="button-container">
            <div class="conf">
                <button onclick="confEdit()">Подтвердить</button>
            </div>
            <div>
                <button id="back-button">Вернуться назад</button>
            </div>
        </div>
        <div id="msgContainer" class="msgContainer">
            <p id="message"></p>
        </div>
    </div>
    <script>
        const backButton = document.getElementById('back-button');
        backButton.addEventListener('click', function () {
            localStorage.removeItem('idUser');
            window.history.back();
        });
    </script>
    <script src="common/scripts/user/userEditScript.js"></script>
    <script src="common/scripts/user/userLoadScript.js"></script>
```

```
    <th:block th:replace="~{header :: headerFragment}"></th:block>
</body>
</html>
```

userPage.html

```
<html xmlns:th="http://www.thymeleaf.org">

<head>
    <title>Страница пользователя</title>
    <link rel="stylesheet" href="../common/styles/accPage.Style.css">
    <link rel="stylesheet" href="../common/styles/postStyle.css">
</head>
<body>
    <div th:fragment="headerFragment" class="header-container">
    </div>
    <br>

    <div class="user-info">
        <div class="nick" id="nick">
        </div>
        <div class="status">
            <div class="status-label">Цитат публиковано:
                <div class="status-number" id="status-public"></div>
            </div>
            <div class="status-label">Цитат предложено:
                <div class="status-number" id="status-offered"></div>
            </div>
            <div class="status-label">Реакций поставлено:
                <div class="status-number" id="status-liked"></div>
            </div>
        </div>
    </div>
    <div class="posts" id="posts-container"></div>
    <script type="module"
src="common/scripts/user/userPostScript.js"></script>
    <script type="module"
src="common/scripts/user/accUserScript.js"></script>

    <th:block th:replace="~{header :: headerFragment}"></th:block>
</body>
</html>
```

userView.html

```
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Лента пользователей</title>
    <link rel="stylesheet" href="common/styles/postStyle.css">
    <link rel="stylesheet" href="../common/styles/postStyle.css">
</head>
<body>
    <div th:fragment="headerFragment" class="header-container">
    </div>
    <br>
    <div class="posts" id="posts-container"></div>
    <div class="modal" id="deleteAccountModal">
        <div class="modal-content">
            <p id="modal-msg">Вы уверены, что хотите удалить
пользователя?</p>
            <div class="buttons">
                <div class="delete-account">
                    <button onclick="deleteAccount()">Да, удалить</button>
                </div>
                <div class="conf">
```

```
                    <button onclick="delCancel()">Нет</button>
                </div>
            </div>
        </div>
    </div>
    <script src="common/scripts/user/userEditScript.js"></script>
    <script src="common/scripts/user/userDelScript.js"></script>
    <script type="module"
src="common/scripts/user/allUserScript.js"></script>
    <th:block th:replace="~{header :: headerFragment}"></th:block>
</body>
</html>
```

accPage.Style.css

```
body {
    background-color: #f0f0f5;
    font-family: 'Arial', sans-serif;
    margin: 0;
}

.user-info {
    border-radius: 10px;
    background-color: #e5e9ef;
    margin: 10px auto;
    padding: 20px;
    width: 80%;
}

.nick {
    color: #333;
    font-size: 150%;
    margin-bottom: 10px;
    text-align: center;
    word-wrap: break-word;
}

.status {
    align-items: center;
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
}

.status-label,
.status-number {
    border-radius: 5px;
    margin: 1.5% 1.5%;
    padding: 5%;
    text-align: center;
}

.status-label {
    border: solid #adadc0;
    color: #40404d;
    font-size: 135%;
    height: 100px;
    padding: 1.6%;
}

.status-number {
    color: #3f72a9;
    font-size: 180%;
    text-align: center;
```

```
}

.status-number p {
    color: #3f72a9;
    font-size: 100%;
    text-align: center;
    display: inline;

}
```

createAndEditStyle.css

```
body {
    background-color: #c2c2d7;
    margin: 0;
    margin-top: 80px;
}

#modal-msg {
    color: #3b3b44;
    font-size:160%;
}
.msgContainer p{
    font-size: 140%;
    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
    margin-left: 2%;
    color: #303035
}

.create-or-edit {
    background-color: #e5e9ef;
    border-radius: 10px;
    padding: 20px;
    margin: 20px auto;
    width: 780px;
}

.create-or-edit label {
    display: inline-block;
    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
    font-size: 120%;
    margin-right: 2%;
    padding: 8px;
    text-align: left;
    width: 40%;
}

.create-or-edit div {
    margin-bottom: 20px;
}

.create-or-edit button {
    margin-left: 2%;
    border: solid #adadc0;
    background-color: #e5e9ef;
    border-radius: 10px;
    color: #3b3b44;
    font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande',
'Lucida Sans Unicode', Geneva, Verdana, sans-serif;
    font-size: 110%;
    font-weight: bold;
    height: 40px;
    width: 30%;
}
```

```
.create-or-edit button:hover {
    background-color: #bec2c7;
}

.create-or-edit h2 {
    color: #303035;
    font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-
serif;
    font-weight: bold;
    font-size: 220%;
    text-align: center;
    margin-top: -1%;
    position: relative;
}



.create-or-edit input, .create-or-edit select {

    border-radius: 5px;
    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
    border: none;
    margin-bottom: 5px;
    margin-left: 2%;
    width: 85%;
    height: 4.5%;
    font-size: 110%;
    max-width: 60%;
}


.create-or-edit textarea {
    margin-left: 2%;
    border: none;
    border-radius: 5px;
    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
    font-size: 120%;
    height: 95px;
    padding: 8px;
    width: 90%;
    resize: none;
}
```

headerStyle.css

```
body {
    background-color: #c2c2d7;
    margin: 0;
    margin-top: 80px;
}

#name {
    color: #e5e9ef;
    font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-
serif;
    font-size: 55px;
    font-weight: bold;
    text-decoration: none;
    margin-right: 1%;
}

.header-container {
```

```
    margin-bottom: 85px;
}

header {
    align-items: center;
    background-color: #5c5c6a;
    color: #e5e9ef;
    display: flex;
    height: 60px;
    margin-bottom: 20px;
    padding: 1%;
    position: fixed;
    top: 0;
    width: 100%;
}

.dropdown {
    margin-left: 2%;
    display: inline-block;
    position: relative;
}

.dropdown button {
    margin-left: 0;
    background-color: #5c5c6a;
    border: none;
    border-radius: 3px;
    color: #e5e9ef;
    font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande',
'Lucida Sans Unicode', Geneva, Verdana, sans-serif;
    font-size: 130%;
    height: 110%;
}

.dropdown:hover button {
    background-color: #4c4c57;
}

.dropdown-content {
    border-radius: 5px;
    display: none;
    position: absolute;
}

.dropdown-content a {
    border-radius: 5px;
    color: #e5e9ef;
    display: block;
    font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande',
'Lucida Sans Unicode', Geneva, Verdana, sans-serif;
    font-size: 115%;
    padding: 12px 16px;
    text-decoration: none;
}

.dropdown-content a:hover {
    background-color: #4c4c57;
}

.dropdown:hover .dropdown-content {
    background-color: #5c5c6a;
    display: block;
}
```

postStyle.css

```
.post {
    border-radius: 20px;
    background-color: #e5e9ef;
    padding: 2.6%;
    margin: 10px auto;
    width: 68%;
}

.post-nick {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    font-size: 150%;
    margin-bottom: 1.5%;
}

.post-nick a {
    color: #31314c;
    text-decoration: none;
}

.post-nick a:hover {
    text-decoration: underline;
}

.post-text {
    color: #3b3b44;
    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
    font-size: 180%;
    margin-bottom: 1.5%;
}

.post-img {
    margin: 0 auto;
    margin-bottom: 1.5%;
    overflow: hidden;
}

.post-img img {
    border-radius: inherit;
    display: block;
    max-height: 600px;
    object-fit: contain;
    width: 100%;
}

.buttons button {
    margin-left: 2%;
    border: solid #adadc0;
    background-color: #e5e9ef;
    border-radius: 5px;
    color: #3b3b44;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    font-size: 130%;
    font-weight: bold;
    height: 40px;
    width: 20%;
}

.buttons button:hover {
    background-color: #bec2c7;
}
```

```
.edit {
    float: right;
}

#edit {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    color: #424297;
    font-size: 120%;
}

.del {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #e5e9ef;
    border: solid #adadc0;
    border-radius: 6px;
    font-size: 110%;
    font-weight: bold;
    color: #31314c;
}

.del:hover {
    background-color: #bec2c7;
}


.info,
.title {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    font-size: 150%;
    margin-bottom: 1.5%;
    color: #31314c;

}

.title {
    font-weight: bold;
}

#deleteAccountModal {
    display: none;
}

.modal {
    align-items: center;
    background-color: rgba(0, 0, 0, 0.5);
    height: 100%;
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
}

.modal-content {
    position: relative;
    margin-top: 300px;
    margin-left: auto;
    margin-right: auto;
    width: 480px;
    background-color: #dde2e8;
    padding: 20px;
    border-radius: 10px;
}
```

```
.modal-content p {
    font-size: 120%;
    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;

}

.close-modal {
    position: absolute;
    top: 10px;
    right: 10px;
    font-size: 20px;
    cursor: pointer;
    color: #3b3b44;
    background-color: #e5e9ef;
    border: solid #adadc0;
    border-radius: 10px;
    padding: 8px 12px;
}

.close-modal:hover {
    background-color: #bec2c7;
}

.delete-account button {
    background-color: #e5e9ef;
    border: 2px solid #9a0000;
    border-radius: 10px;
    color: #8f0000;
    font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande',
'Lucida Sans Unicode', Geneva, Verdana, sans-serif;
    font-size: 110%;
    font-weight: bold;
    height: 40px;
    margin-right: 10px;
    width: 30%;
}

.delete-account button:hover {
    background-color: #beb1b1;
}

.conf button {
    margin-left: 2%;
    border: solid #adadc0;
    background-color: #e5e9ef;
    border-radius: 10px;
    color: #3b3b44;
    font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande',
'Lucida Sans Unicode', Geneva, Verdana, sans-serif;
    font-size: 110%;
    font-weight: bold;
    height: 40px;
    width: 30%;
}

.conf button:hover {
    background-color: #bec2c7;
}

.conf,
.delete-account {
    margin-bottom: 1%;
}
```

headerScript.js

```
document.addEventListener("DOMContentLoaded", function () {
    var newElement = document.createElement('div');
    var xhr = new XMLHttpRequest();

    xhr.open('GET', 'header.html', true);

    xhr.onload = function () {
        if (xhr.status >= 200 && xhr.status < 300) {
            newElement.innerHTML = xhr.responseText;
            document.body.appendChild(newElement);
        } else {
            console.error('Ошибка загрузки: ' + xhr.statusText);
        }
    };

    xhr.onerror = function () {
        console.error('Ошибка сети');
    };

    xhr.send();
    console.log("yo");
});
```

allPostScript.js

```
import { displayPosts } from "./postScript.js";

fetch('/fetch/posts/all')
    .then(response => {
        if (!response.ok) {
            throw new Error(`Ошибка HTTP: ${response.status}`);
        }
        return response.json();
    })
    .then(posts => {
        displayPosts(posts);
    })
    .catch(error => {
        console.error('Произошла ошибка:', error);
    });
```

postCreateScript.js

```
function submitForm() {
    let username = document.getElementById("username").value;
    let textPost = document.getElementById("textPost").value;
    let statusPost = document.getElementById("statusPost").value;

    let msgEl = document.getElementById('message');

    let data = {
        username: username,
        textPost: textPost,
        statusPost: statusPost,
    };

    fetch(`/fetch/post/create`, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(data)
    })
```

```
        .then(response => {
            if (!response.ok) {
                throw new Error('Ошибка сети или сервера');
            }
            return response.text();
        })
        .then(data => {
            msgEl.textContent = 'Пост создан';
        })
        .catch(error => {
            msgEl.textContent = 'Ошибка: ' + error.message;

            if (error.message != 'Ошибка сети или сервера.') {
                msgEl.textContent += '. Пользователь не найден ';
            }
        });
}
```

postDelScript.js

```
let delAlertBtn = document.getElementById("deleteAccountModal");
let idPost;

function delAlert(button) {
    let postBlock = button.closest('.post');

    if (postBlock) {
        idPost = postBlock.getAttribute('data-id');
        delAlertBtn.style.display = "block";
    } else {
        console.error('Post block not found');
    }

    delAlertBtn.style.display = "block";
}
function deleteAccount() {
    let data = {
        idPost: idPost,
    };
    fetch(`/fetch/post/delete`, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(data)
    })
    location.reload ()
}
function delCancel() {
    delAlertBtn.style.display = "none";
}
```

postEditScript.js

```
let delAlertEdit = document.getElementById("deleteAccountModal");
function getEdit(a_href) {
    let postBlock = a_href.closest('.post');

    if (postBlock) {
        const idPost = postBlock.getAttribute('data-id');
        localStorage.setItem('idPost', idPost);
    } else {
        console.error('Post block not found');
    }
```

```
    delAlertEdit.style.display = "none";
}
function confEdit() {
    const idPost = localStorage.getItem('idPost');
    let data = {
        idPost: idPost,
        username: document.getElementById("username").value,
        textPost: document.getElementById("textPost").value,
        statusPost: document.getElementById("statusPost").value
    };
    fetch(`/fetch/post/edit`, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(data)
    })
        .then(response => {
            if (!response.ok) {
                throw new Error('Ошибка сети или сервера');
            }
            return response.text();
        })
    window.location.href = '/postView';
    localStorage.removeItem('idPost');
}
```

postLoadScript.js

```
function getData() {
    let idPost = localStorage.getItem('idPost');
    let data = {
        idPost: idPost,
    };
    return data;
}

async function fetchData() {
    try {
        await fetch(`/fetch/post/find`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(getData())
        });
        const response = await fetch('/fetch/post/load');
        if (!response.ok) {
            throw new Error(`Ошибка HTTP: ${response.status}`);
        }
        const post = await response.json();

        document.getElementById("username").value = post.username;
        document.getElementById("textPost").value = post.textPost;
        document.getElementById("statusPost").value = post.statusPost;
    } catch (error) {
        console.error('Произошла ошибка:', error);
    }
}

fetchData();
```

postScript.js

```
export function displayPosts(posts) {
    const postsContainer = document.getElementById('posts-container');

    posts.forEach(post => {
        const postElement = document.createElement('div');
        postElement.className = 'post';
        postElement.setAttribute('data-id', `${post.idPost}`);
        let status;
        if(post.statusPost === "published"){
            status ="Опубликовано";
        }else{
            status ="Предложено";
        }

        postElement.innerHTML = `
            <div class="edit">
                <a id="edit" href="/postEdit"
onclick="getEdit(this)">Редактировать</a>
                <button class="del" onclick="delAlert(this)">Удалить
пост</button>
            </div>
            <div class="post-nick">
                <a href="/${post.idUser}">${post.username}</a>
            </div>
            <div>
            <span class="title">Статус:</span><span class="info"> ${status}
</span>
            </div>
            <br>
            <div class="post-text">
                <span>${post.textPost}</span>
            </div>
        `;

        postsContainer.appendChild(postElement);
    });
}
```

publishedPostScript.js

```
import { displayPosts } from "./postScript.js";

fetch('/fetch/posts/published')
    .then(response => {
        if (!response.ok) {
            throw new Error(`Ошибка HTTP: ${response.status}`);
        }
        return response.json();
    })
    .then(posts => {
        displayPosts(posts);
    })
    .catch(error => {
        console.error('Произошла ошибка:', error);
    });
```

allReactionScript.js

```
import {displayReactions } from "./reactionScript.js";

fetch('/fetch/reactions/all')
    .then(response => {
        if (!response.ok) {
            throw new Error(`Ошибка HTTP: ${response.status}`);
        }
```

```
        return response.json();
    })
    .then(reactions => {
        displayReactions(reactions);
    })
    .catch(error => {
        console.error('Произошла ошибка:', error);
    });
```

reactionCreateScript.js

```
function submitForm() {
    let nameReaction = document.getElementById("nameReaction").value;

    let msgEl = document.getElementById('message');

    let data = {
        nameReaction: nameReaction,
    };

    console.log(data+ " ");

    fetch(`/fetch/reaction/create`, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(data)
    })
    .then(response => {
        if (!response.ok) {
            throw new Error('Ошибка сети или сервера');
        }
        return response.text();
    })
    .then(data => {
        msgEl.textContent = 'Реакция создана';
    })
    .catch(error => {
        msgEl.textContent = 'Ошибка: ' + error.message;

        if (error.message != 'Ошибка сети или сервера.') {
            msgEl.textContent = 'Ошибка: Данная реакция существует!';
        } else {
            msgEl.textContent += ' Ошибка сервера: ' + error;
        }
    });
}
```

reactionDelScript.js

```
let delAlertBtn = document.getElementById("deleteAccountModal");
let idReaction;

function delAlert(button) {
    let postBlock = button.closest('.post');

    if (postBlock) {
        idReaction = postBlock.getAttribute('data-id');
        delAlertEdit.style.display = "block";
    } else {
        console.error('Post block not found');
    }

    delAlertEdit.style.display = "block";
```

```
}
function deleteAccount() {
    let data = {
        idReaction: idReaction,
    };
    fetch(`/fetch/reaction/delete`, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(data)
    })
    location.reload ()
}
function delCancel() {
    delAlertEdit.style.display = "none";
}
```

reactionEditScript.js

```
let delAlertEdit = document.getElementById("deleteAccountModal");
function getEdit(a_href) {
    let postBlock = a_href.closest('.post');

    if (postBlock) {
        const idReaction = postBlock.getAttribute('data-id');
        localStorage.setItem('idReaction', idReaction);
    } else {
        console.error('Post block not found');
    }

    delAlertEdit.style.display = "none";
}
function confEdit() {
    const idReaction = localStorage.getItem('idReaction');
    const nameReaction = document.getElementById("nameReaction").value

    let msgEl = document.getElementById('message');

    let data = {
        idReaction: idReaction,
        nameReaction: nameReaction
    };
    fetch(`/fetch/reaction/edit`, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(data)
    })
        .then(response => {
            if (!response.ok) {
                throw new Error('Ошибка сети или сервера');
            }
            window.location.href = '/reactnView';
            localStorage.removeItem('idReaction');
            return response.text();
        })
        .catch(error => {
            msgEl.textContent = 'Ошибка: ' + error.message;

            if (error.message != 'Ошибка сети или сервера.') {
                msgEl.textContent = 'Ошибка: Данная реакция существует!';
            }
```

```
            document.getElementById('errorContainer').style.display =
'block';
        });
}
```

reactionLoadScript.js

```
function getData() {
    let idReaction = localStorage.getItem('idReaction');
    let data = {
        idReaction: idReaction,
    };
    return data;
}

async function fetchData() {
    try {
        // Выполняем POST-запрос для сохранения idReaction
        await fetch(`/fetch/reaction/find`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(getData())
        });

        // Выполняем GET-запрос для загрузки реакции
        const response = await fetch('/fetch/reaction/load');
        if (!response.ok) {
            throw new Error(`Ошибка HTTP: ${response.status}`);
        }

        // Получаем данные реакции
        const reaction = await response.json();

        // Устанавливаем значение в HTML-элемент
        document.getElementById("nameReaction").value =
reaction.nameReaction;
    } catch (error) {
        console.error('Произошла ошибка:', error);
    }
}

// Вызываем функцию
fetchData();
```

reactionScript.js

```
export function displayReactions(reactions) {
    const postsContainer = document.getElementById('posts-container');

    reactions.forEach(reaction => {
        const postElement = document.createElement('div');
        postElement.className = 'post';
        postElement.setAttribute('data-id', `${reaction.idReaction}`);
        let status;
        if(reaction.statusPost === "published"){
            status ="Опубликовано";
        }else{
            status ="Предложено";
        }
```

```
        postElement.innerHTML = `
        <div class="edit">
            <a id="edit" href="/reactionEdit" onclick="getEdit(this)"
>Редактировать</a>
            <button onclick="delAlert(this)" class="del">Удалить
реакцию</button>
        </div>
        <br>
        <div>
            <span class="title">Реакция:</span><span class="info">
${reaction.nameReaction}</span>
        </div>
        <div class="post-img">
        </div>
        `;

        postsContainer.appendChild(postElement);
    });
}
```

accUserScript.js

```
const currentUrl = window.location.href;
const url = new URL(currentUrl);
const userId = url.pathname.split('/').pop();
const apiUrl = `/fetch/${userId}/info`;

fetch(apiUrl)
    .then(response => {
        if (!response.ok) {
            throw new Error(`Ошибка HTTP: ${response.status}`);
        }
        return response.json();
    })
    .then(info => {
        const postsPublic = document.getElementById('status-public');
        const postsLiked = document.getElementById('status-liked');
        const postsOffered = document.getElementById('status-offered');
        const nickUser=document.getElementById('nick');
        nickUser.innerHTML=`<h1>${info.nickUser}</h1>`;
        postsPublic.innerHTML = `<p>${info.publicPosts}</p>`;
        postsLiked.innerHTML = `<p>${info.reactionPosts}</p>`;
        postsOffered.innerHTML = `<p>${info.offeredPosts}</p>`;
    })
    .catch(error => {
        console.error('Произошла ошибка:', error);
    });
```

allUserScript.js

```
import { displayUsers } from "./userScript.js";

fetch('/fetch/users/all')
    .then(response => {
        if (!response.ok) {
            throw new Error(`Ошибка HTTP: ${response.status}`);
        }
        return response.json();
    })
    .then(users => {
        displayUsers(users);
    })
    .catch(error => {
```

```
        console.error('Произошла ошибка:', error);
    });
```

userCreateScript.js

```
function submitForm() {
    let username = document.getElementById("username").value;
    let password = document.getElementById("password").value;
    let roleUser = document.getElementById("role").value;

    let msgEl = document.getElementById('message');

    let data = {
        username: username,
        password: password,
        roleUser: roleUser
    };

    fetch(`/fetch/user/create`, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(data)
    })
    .then(response => {
        if (!response.ok) {
            throw new Error('Ошибка сети или сервера');
        }
        return response.text();
    })
    .then(data => {
        msgEl.textContent = 'Пользователь создан';
    })
    .catch(error => {
        msgEl.textContent = 'Ошибка: ' + error.message;

        if (error.message != 'Ошибка сети или сервера.') {
            msgEl.textContent = 'Ошибка: Данный ник занят!';
        } else {
            msgEl.textContent += ' Ошибка сервера: ' + error;
        }

        document.getElementById('errorContainer').style.display = 'block';
    });
}
```

userDelScript.js

```
let delAlertBtn = document.getElementById("deleteAccountModal");
let idUser;

function delAlert(button) {
    let postBlock = button.closest('.post');

    if (postBlock) {
        idUser = postBlock.getAttribute('data-id');
        delAlertBtn.style.display = "block";
    } else {
        console.error('Post block not found');
    }

    delAlertBtn.style.display = "block";
}
```

```
function deleteAccount() {
    let data = {
        idUser: idUser,
    };
    fetch(`/fetch/user/delete`, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(data)
    })
    location.reload ()
}
function delCancel() {
    delAlertBtn.style.display = "none";
}
```

userEditScript.js

```
let delAlertEdit = document.getElementById("deleteAccountModal");
function getEdit(a_href) {
    let postBlock = a_href.closest('.post');

    if (postBlock) {
        const idUser = postBlock.getAttribute('data-id');
        localStorage.setItem('idUser', idUser);
    } else {
        console.error('Post block not found');
    }

    delAlertEdit.style.display = "none";
}
function confEdit() {
    const idUser = localStorage.getItem('idUser');
    const username = document.getElementById("username").value;
    const password = document.getElementById("password").value;
    const roleUser =document.getElementById("role").value;

    let msgEl = document.getElementById('message');

    let data = {
        idUser: idUser,
        username: username,
        password: password,
        roleUser: roleUser
    };
    fetch(`/fetch/user/edit`, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(data)
    })
        .then(response => {
            if (!response.ok) {
                throw new Error('Ошибка сети или сервера');
            }
            window.location.href = '/userView';
            localStorage.removeItem('idUser');
            return response.text();
        })
        .catch(error => {
            msgEl.textContent = 'Ошибка: ' + error.message;
```

```
            if (error.message != 'Ошибка сети или сервера.') {
                msgEl.textContent = 'Ошибка: Данный ник занят!';
            }

            document.getElementById('errorContainer').style.display =
'block';
        });
}
```

userLoadScript.js

```
function getData() {
    let idUser = localStorage.getItem('idUser');
    let data = {
        idUser: idUser,
    };
    return data;
}

async function fetchData() {
    try {
        await fetch(`/fetch/user/find`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(getData())
        });
        const response = await fetch('/fetch/user/load');
        if (!response.ok) {
            throw new Error(`Ошибка HTTP: ${response.status}`);
        }
        const user = await response.json();
        document.getElementById("username").value = user.username;
        document.getElementById("password").value = user.password;
        document.getElementById("role").value = user.roleUser;
    } catch (error) {
        console.error('Произошла ошибка:', error);
    }
}
fetchData();
```

userPostScript.js

```
import { displayPosts } from "../post/postScript.js";

const currentUrl = window.location.href;
const url = new URL(currentUrl);
const userId = url.pathname.split('/').pop();
const apiUrl = `/fetch/user/${userId}`;

fetch(apiUrl)
    .then(response => {
        if (!response.ok) {
            throw new Error(`Ошибка HTTP: ${response.status}`);
        }
        return response.json();
    })
    .then(posts => {
        displayPosts(posts);
    })
    .catch(error => {
        console.error('Произошла ошибка:', error);
    });
```

userScript.js

```
export function displayUsers(users) {
    const postsContainer = document.getElementById('posts-container');

    users.forEach(user => {
        const postElement = document.createElement('div');
        postElement.className = 'post';
        postElement.setAttribute('data-id', `${user.idUser}`);
        let role;
        if (user.roleUser === "admin") {
            role = "Админ";
        } else {
            role = "Пользователь";
        }

        postElement.innerHTML = `
            <div class="edit">
            <a id="edit" href="/userEdit"
onclick="getEdit(this)">Редактировать</a>
            <button class="del" onclick="delAlert(this)">Удалить
пользователя</button>
            </div>
            <div>
                <span class="title">Ник:</span><span class="info">
${user.username}</span>
            </div>
            <div >
                <span class="title">Пароль:</span><span class="info">
${user.password}</span>
            </div>
            <div>
                <span class="title">Роль:</span><span class="info">
${role}</span>
            </div>
            <div class="post-img">
            </div>
        `;

        postsContainer.appendChild(postElement);
    });
}
```