Курсовой проект

На тему «Разработка модуля отправки отзывов для рекомендательной системы CS»

Выполнил студент
группы АУБП-21-2
Лаврентьев Руслан

Постановка задачи

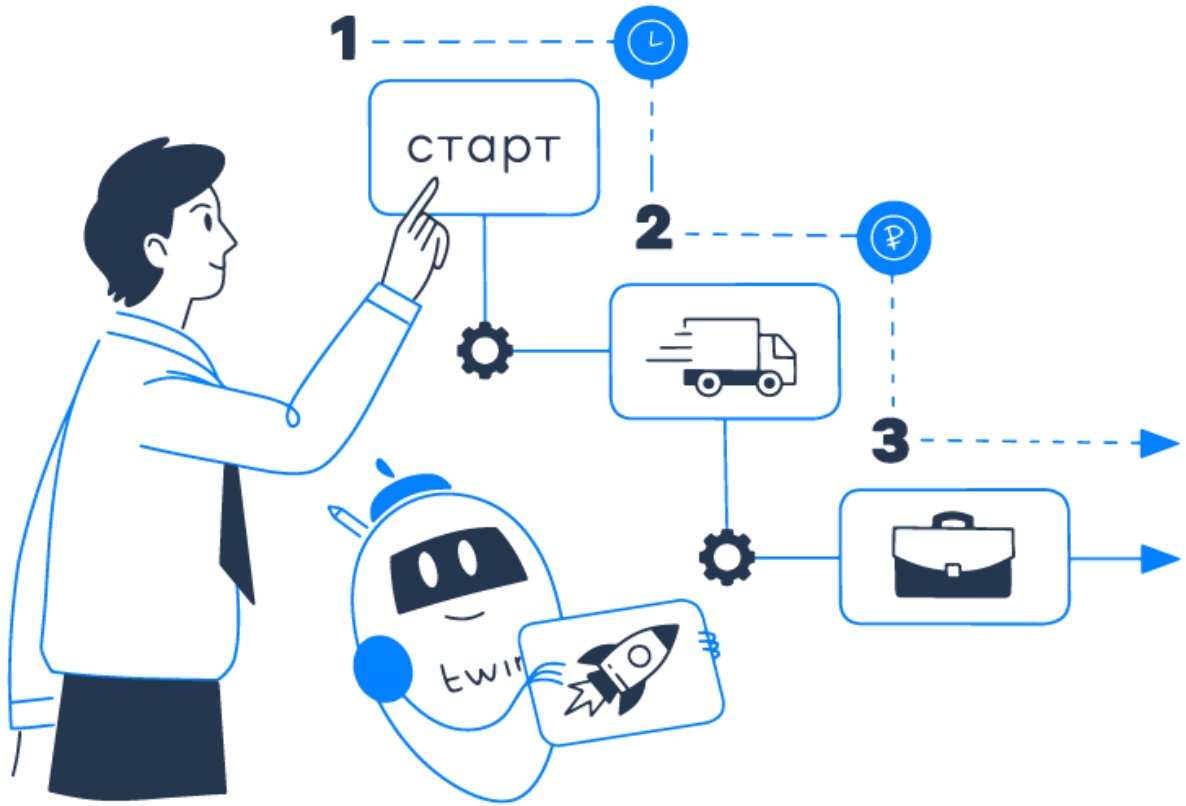
- Целью разработки модуля в рамках системы "Рекомендательная система для игроков CS2" является создание функционала, обеспечивающего сбор, хранение и отображение пользовательских отзывов о работе системы. Это позволит учитывать обратную связь для дальнейшего улучшения качества рекомендаций и удобства системы
- Объектом автоматизации является процесс сбора и отображения пользовательских отзывов о работе рекомендательной системы для игроков CS2.

Существующие решения

- На текущий момент аналогичные системы обратной связи могут быть реализованы через сторонние платформы (например, форумы, социальные сети или сторонние виджеты отзывов). Однако такие подходы не интегрированы непосредственно в систему и не позволяют использовать данные отзывов для улучшения работы алгоритмов.

Цель автоматизации

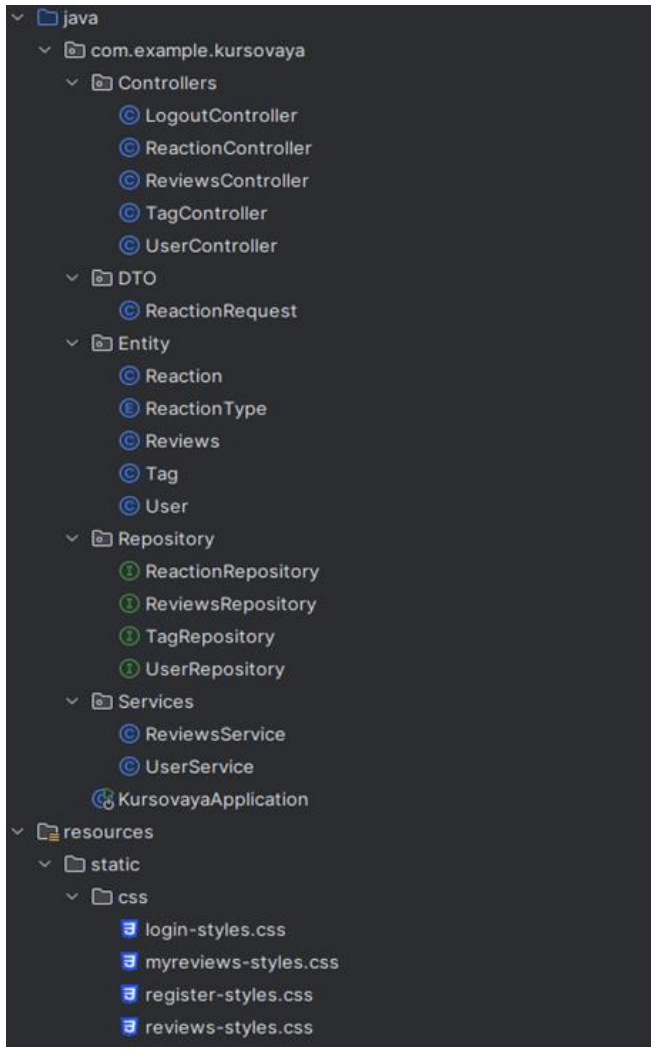
- Основной целью автоматизации является разработка модуля сбора и отображения пользовательских отзывов о работе рекомендательной системы для игроков CS2

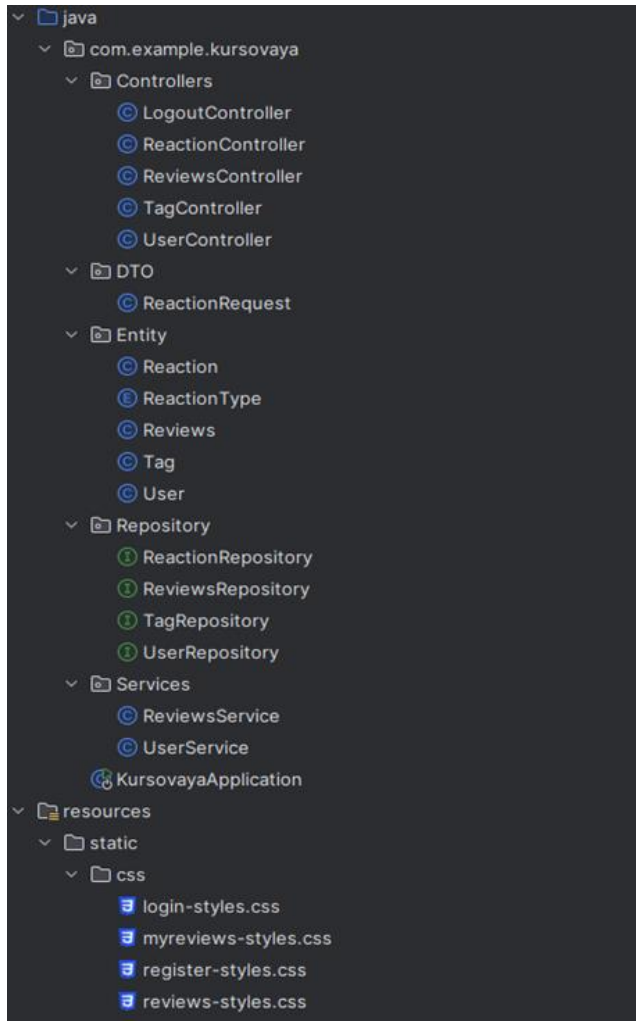


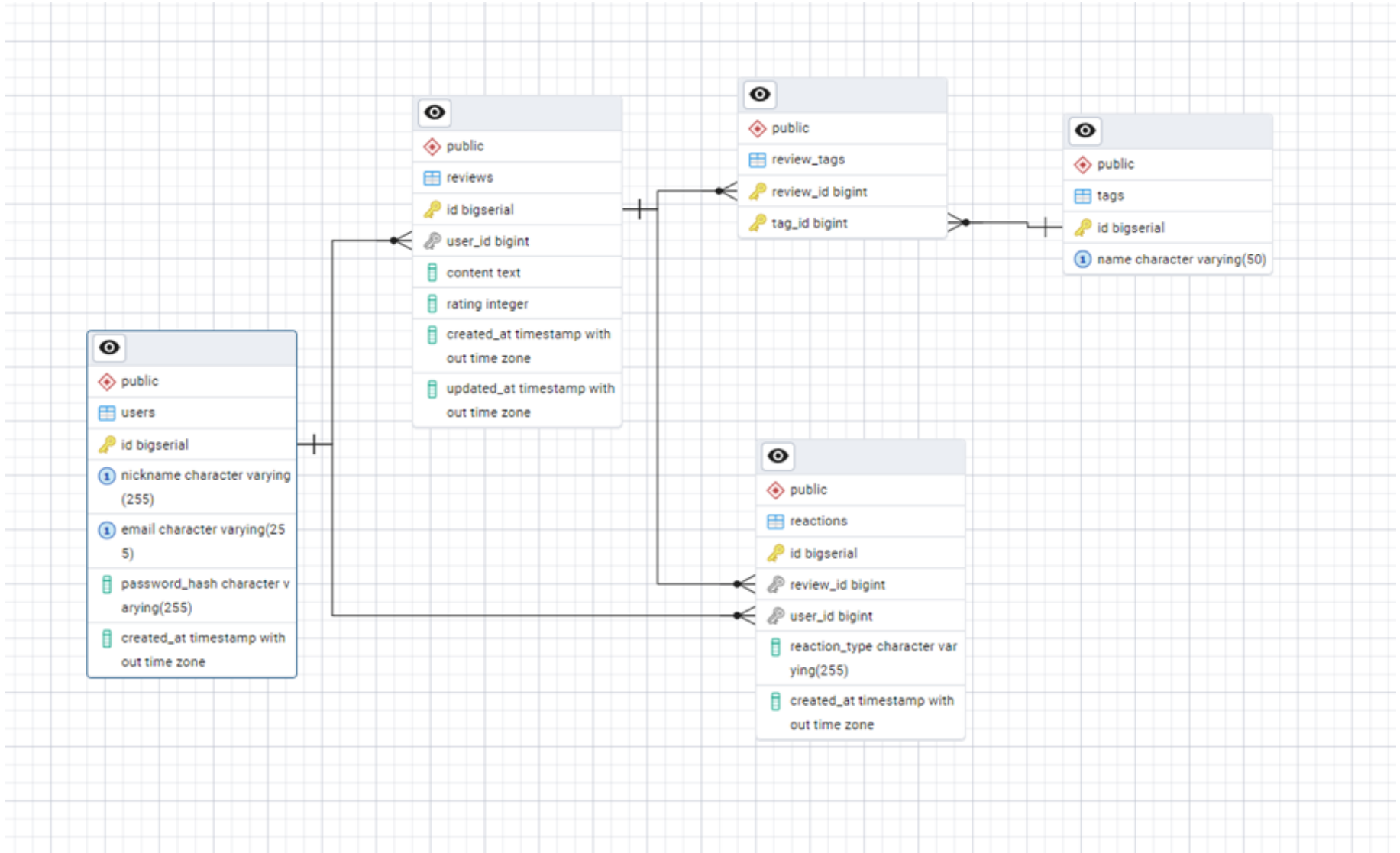
Выбор средства разработки

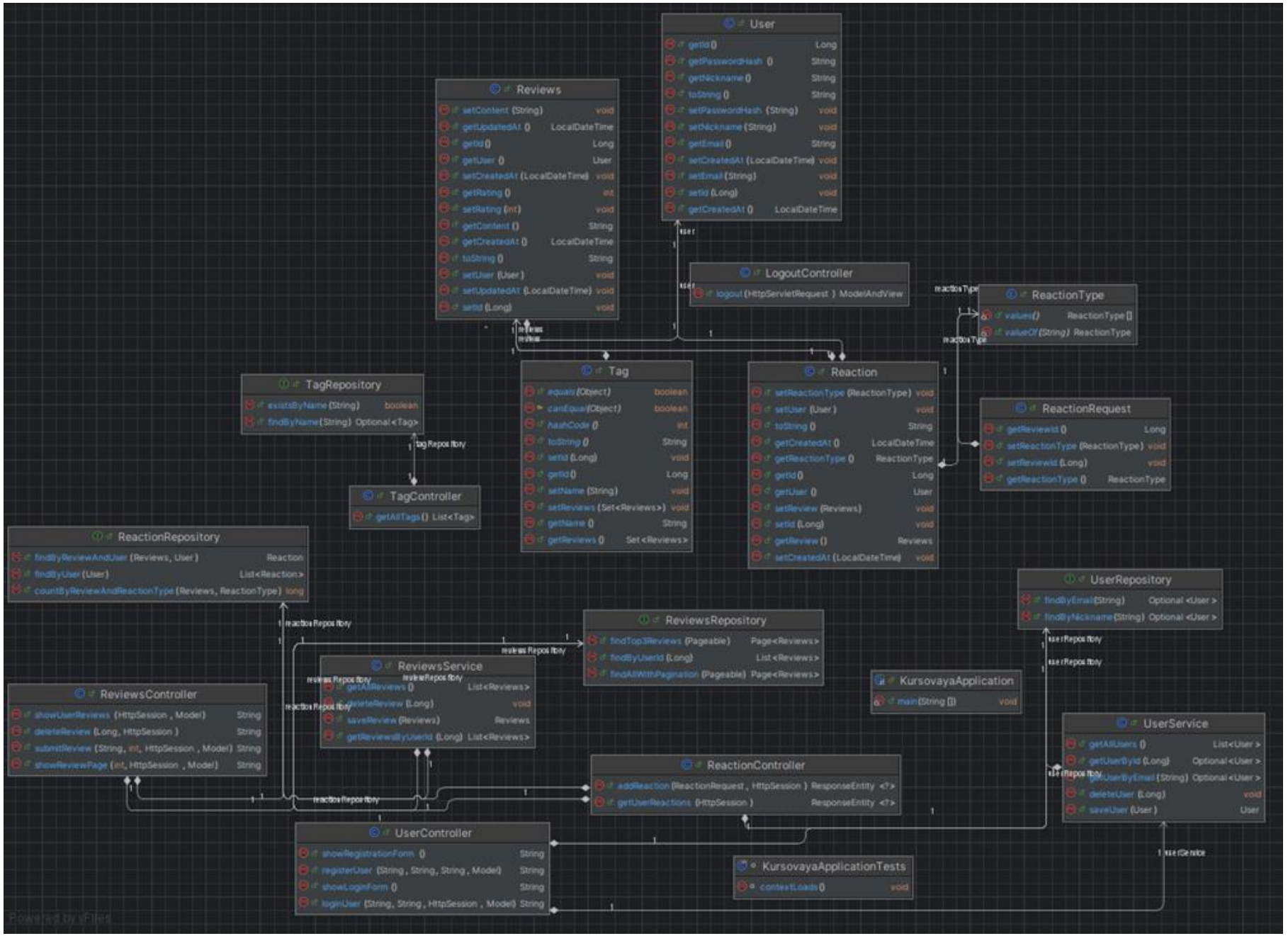
- Для разработки системы был выбран язык программирования Java с использованием фреймворка Spring Boot и вспомогательных библиотек (Spring Data JPA, Thymleaf, Hibernate).

Дерево проекта






public
users
id bigserial
nickname character varying (255)
email character varying(25 5)
password_hash character v arying(255)
created_at timestamp with out time zone
public
reviews
id bigserial
user_id bigint
content text
rating integer
created_at timestamp with out time zone
updated_at timestamp with out time zone
public
review_tags
review_id bigint
tag_id bigint
public
tags
id bigserial
name character varying(50)
public
reactions
id bigserial
review_id bigint
user_id bigint
reaction_type character var ying(255)
created_at timestamp with out time zone


User
getId() Long
getPasswordHash() String
getNickname() String
toString() String
setPasswordHash(String) void
setNickname(String) void
getEmail() String
setCreatedAt(LocalDateTime) void
setEmail(String) void
setId(Long) void
getCreatedAt() LocalDateTime
Reviews
setContent(String) void
getUpdatedAt() LocalDateTime
getId() Long
getUser() User
setCreatedAt(LocalDateTime) void
getRating() int
setRating(int) void
getContent() String
getCreatedAt() LocalDateTime
toString() String
setUser(User) void
setUpdatedAt(LocalDateTime) void
setId(Long) void
LogoutController
logout(HttpServletRequest) ModelAndView
ReactionType
values() ReactionType[]
valueOf(String) ReactionType
Tag
equals(Object) boolean
canEqual(Object) boolean
hashCode() int
toString() String
setId(Long) void
getId() Long
setName(String) void
setReviews(Set<Reviews>) void
getName() String
getReviews() Set<Reviews>
Reaction
setReactionType(ReactionType) void
setUser(User) void
toString() String
getCreatedAt() LocalDateTime
getReactionType() ReactionType
getId() Long
getUser() User
setReview(Reviews) void
setId(Long) void
getReview() Reviews
setCreatedAt(LocalDateTime) void
ReactionRequest
getReviewId() Long
setReactionType(ReactionType) void
setReviewId(Long) void
getReactionType() ReactionType
TagRepository
existsByName(String) boolean
findByName(String) Optional<Tag>
TagController
getAllTags() List<Tag>
ReactionRepository
findByReviewAndUser(Reviews, User) Reaction
findByUser(User) List<Reaction>
countByReviewAndReactionType(Reviews, ReactionType) long
ReviewsRepository
findTop3Reviews(Pageable) Page<Reviews>
findByUserId(Long) List<Reviews>
findAllWithPagination(Pageable) Page<Reviews>
UserRepository
findByEmail(String) Optional<User>
findByNickname(String) Optional<User>
ReviewsService
getAllReviews() List<Reviews>
deleteReview(Long) void
saveReview(Reviews) Reviews
getReviewsByUserId(Long) List<Reviews>
KursovayaApplication
main(String[]) void
ReviewsController
showUserReviews(HttpSession, Model) String
deleteReview(Long, HttpSession) String
submitReview(String, int, HttpSession, Model) String
showReviewPage(int, HttpSession, Model) String
UserService
getAllUsers() List<User>
getUserById(Long) Optional<User>
getUserByEmail(String) Optional<User>
deleteUser(Long) void
saveUser(User) User
ReactionController
addReaction(ReactionRequest, HttpSession) ResponseEntity<?>
getUserReactions(HttpSession) ResponseEntity<?>
UserController
showRegistrationForm() String
registerUser(String, String, String, Model) String
showLoginForm() String
loginUser(String, String, HttpSession, Model) String
KursovayaApplicationTests
contextLoads() void
Powered by yFiles

Листинг программы

```java
// Метод для отображения отзывов текущего пользователя
@GetMapping("/myreviews")
public String showUserReviews(
        HttpSession session,
        Model model) {
    User loggedInUser = (User) session.getAttribute( s: "user"); // Получение текущего пользователя из сессии
    if (loggedInUser == null) {
        return "redirect:/login"; // Перенаправление на страницу входа, если пользователь не авторизован
    }
    List<Reviews> userReviews = reviewsRepository.findByUserId(loggedInUser.getId());  // Получение отзывов пользователя из базы данны
    model.addAttribute( attributeName: "user", loggedInUser);
    model.addAttribute( attributeName: "userReviews", userReviews);
    return "myreviews";
}
```

```java
// Метод для отображения страницы с отзывами
@GetMapping("/reviews")
public String showReviewPage(
        @RequestParam(value = "page", required = false, defaultValue = "0") int page,
        HttpSession session,
        Model model) {
    User loggedInUser = (User) session.getAttribute( s: "user");   // Получение текущего пользователя из сессии

    if (loggedInUser == null) {
        return "redirect:/login"; // Перенаправление на страницу входа, если пользователь не авторизован
    }

    Pageable topThreePageable = PageRequest.of( pageNumber: 0,  pageSize: 3, Sort.by(Sort.Direction.DESC,  ...properties: "createdAt"));  // Получение топ-3 отзывов (по дате создания, по убыванию)
    List<Reviews> topThreeReviews = reviewsRepository.findTop3Reviews(topThreePageable).getContent();
    Pageable pageable = PageRequest.of(page,  pageSize: 5, Sort.by(Sort.Direction.DESC,  ...properties: "createdAt")); // Получение пагинированного списка отзывов (по 5 отзывов на страницу)
    Page<Reviews> paginatedReviews = reviewsRepository.findAllWithPagination(pageable);

    model.addAttribute( attributeName: "user", loggedInUser);
    model.addAttribute( attributeName: "topReviews", topThreeReviews);
    model.addAttribute( attributeName: "paginatedReviews", paginatedReviews.getContent());
    model.addAttribute( attributeName: "currentPage", page);
    model.addAttribute( attributeName: "totalPages", paginatedReviews.getTotalPages());

    return "reviews";
}
```

```java
// Метод для добавления нового отзыва
@PostMapping("/reviews")
public String submitReview(
        @RequestParam String content,
        @RequestParam int rating,
        @RequestParam(required = false) String selectedTag,
        HttpSession session,
        Model model) {
    User loggedInUser = (User) session.getAttribute( s: "user"); // Получение текущего пользователя из сессии
    if (loggedInUser == null) {
        return "redirect:/login"; // Перенаправление на страницу входа, если пользователь не авторизован
    }
    if (rating < 1 || rating > 5) { // Проверка корректности рейтинга
        model.addAttribute( attributeName: "error",  attributeValue: "Rating must be between 1 and 5!");
        return "reviews";
    }
    Reviews review = new Reviews(); // Создание нового отзыва
    review.setUser(loggedInUser);
    review.setContent(content);
    review.setRating(rating);
    review.setCreatedAt(LocalDateTime.now());
    review.setUpdatedAt(LocalDateTime.now());
    if (review.getTags() == null) {
        review.setTags(new HashSet<>());
    }

    if (selectedTag != null && !selectedTag.isEmpty()) {
        Tag tag = tagRepository.findByName(selectedTag)
                .orElseGet(() -> {
                    Tag newTag = new Tag();
                    newTag.setName(selectedTag);
                    return tagRepository.save(newTag);
                });
        review.getTags().add(tag); // Связываем отзыв с тегом
    }
```

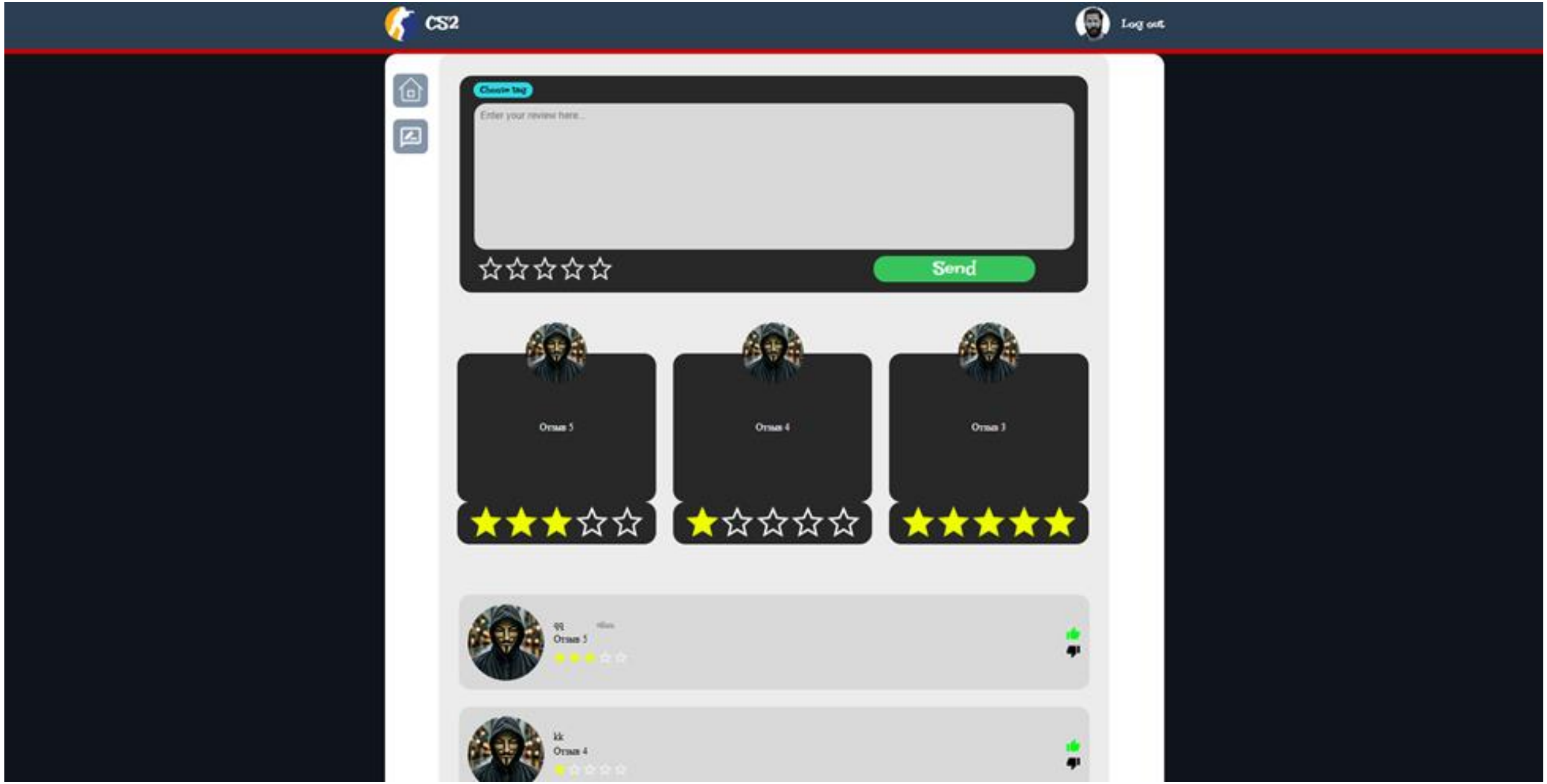
```java
// Метод для удаления отзыва
@PostMapping("/reviews/delete/{id}")
public String deleteReview(@PathVariable Long id, HttpSession session) {

    User loggedInUser = (User) session.getAttribute( s: "user"); // Получение текущего пользователя из сессии
    if (loggedInUser == null) {
        return "redirect:/login"; // Перенаправление на страницу входа, если пользователь не авторизован
    }
    Reviews review = reviewsRepository.findById(id) // Получение отзыва по его ID или выброс исключения, если он не найден
            .orElseThrow(() -> new IllegalArgumentException("Отзыв с ID " + id + " не найден."));

    if (!review.getUser().getId().equals(loggedInUser.getId())) { // Проверка, является ли пользователь владельцем отзыва
        throw new SecurityException("Вы не можете удалить чужой отзыв.");
    }
    reviewsRepository.delete(review);  // Удаление отзыва из базы данных
    return "redirect:/myreviews";
}
```

```java
// Метод для получения всех реакций текущего пользователя
@GetMapping
public ResponseEntity<?> getUserReactions(HttpSession session) {
    User loggedInUser = (User) session.getAttribute( s: "user");  // Получение текущего пользователя из сессии
    if (loggedInUser == null) {
        return ResponseEntity.status(401).body(Map.of( k1: "message",  v1: "User not authenticated")); // Возвращение ошибки 401, если пользователь не авторизован
    }
    List<Reaction> reactions = reactionRepository.findByUser(loggedInUser); // Получение списка реакций пользователя
    Map<Long, String> userReactions = reactions.stream()  // Преобразование списка реакций в Map, где ключ - ID отзыва, значение - тип реакции
            .collect(Collectors.toMap(
                    reaction -> reaction.getReview().getId(),
                    reaction -> reaction.getReactionType().name()
            ));
    return ResponseEntity.ok(userReactions); // Возвращение списка реакций пользователя
}
```

```java
// Метод для добавления или обновления реакции
@PostMapping
public ResponseEntity<?> addReaction(@RequestBody ReactionRequest request, HttpSession session) {
    User loggedInUser = (User) session.getAttribute( s: "user"); // Получение текущего пользователя из сессии
    if (loggedInUser == null) {
        return ResponseEntity.status(401).body(Map.of( k1: "message",  v1: "User not authenticated")); // Возвращение ошибки 401, если пользователь не авторизован
    }
    Reviews review = reviewsRepository.findById(request.getReviewId()) // Получение отзыва по ID или выброс исключения, если отзыв не найден
            .orElseThrow(() -> new RuntimeException("Review not found"));
    Reaction existingReaction = reactionRepository.findByReviewAndUser(review, loggedInUser);  // Проверка, существует ли уже реакция на данный отзыв от текущего пользователя
    if (existingReaction != null) {
        if (existingReaction.getReactionType() == request.getReactionType()) { // Если тип реакции совпадает с новым запросом, удаляем существующую реакцию
            reactionRepository.delete(existingReaction);
            return ResponseEntity.ok().build();
        } else {
            existingReaction.setReactionType(request.getReactionType()); // Если тип реакции отличается, обновляем её
            reactionRepository.save(existingReaction);
            return ResponseEntity.ok().build();
        }
    }
    Reaction reaction = new Reaction(); // Если реакция отсутствует, создаем новую
    reaction.setReview(review);
    reaction.setUser(loggedInUser);
    reaction.setReactionType(request.getReactionType());
    reactionRepository.save(reaction); // Сохраняем новую реакцию в базе данных
    return ResponseEntity.ok().build();
}
```


CS2
Log out
Choose tag
Enter your review here...
Send
Отзыв 5
Отзыв 4
Отзыв 3
qq
Отзыв 5
kk
Отзыв 4

CS2

Log out

99

Отзыв 5