

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.



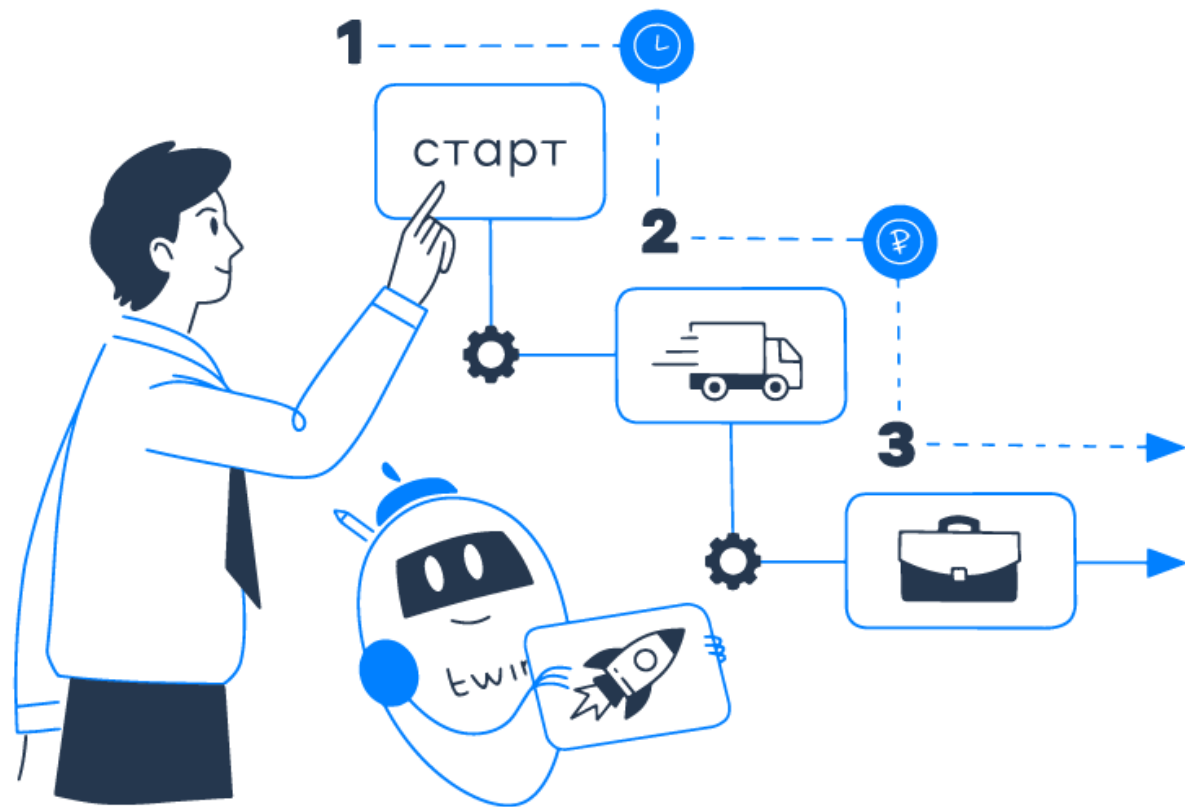
Существующие решения

- 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

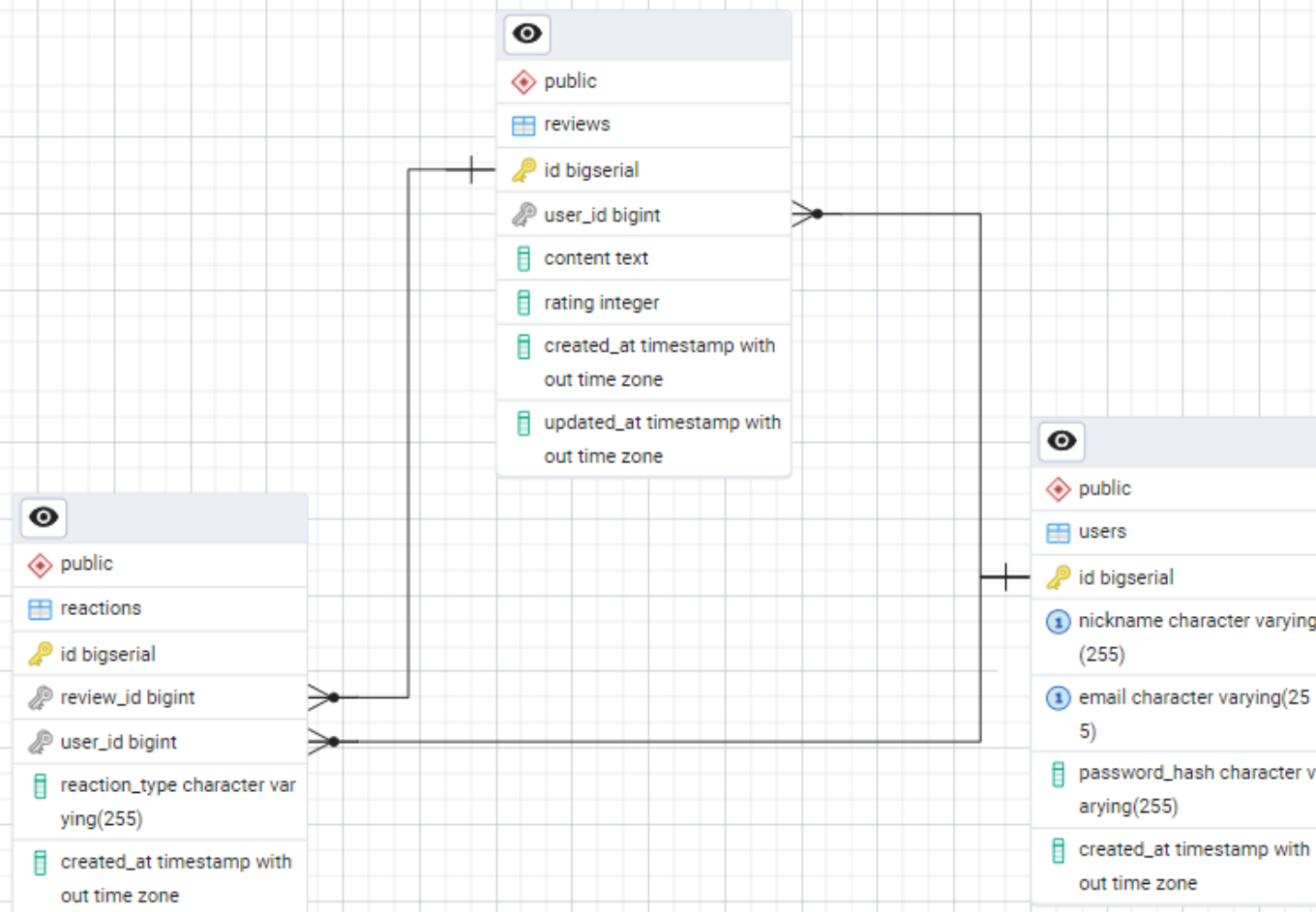
- Для разработки системы был выбран язык программирования Java с использованием фреймворка Spring Boot и вспомогательных библиотек (Spring Data JPA, Thymleaf, Hibernate).



Дерево проекта

```
java
├── com.example.kursovaya
│   ├── Controllers
│   │   ├── LogoutController
│   │   ├── ReactionController
│   │   ├── ReviewsController
│   │   └── UserController
│   ├── DTO
│   │   └── ReactionRequest
│   ├── Entity
│   │   ├── Reaction
│   │   ├── ReactionType
│   │   ├── Reviews
│   │   └── User
│   ├── Repository
│   │   ├── ReactionRepository
│   │   ├── ReviewsRepository
│   │   └── UserRepository
│   ├── Services
│   │   ├── ReviewsService
│   │   ├── UserService
│   │   └── KursovayaApplication
│   └── resources
│       ├── static
│       │   └── css
│       │       ├── login-styles.css
│       │       ├── myreviews-styles.css
│       │       ├── register-styles.css
│       │       └── reviews-styles.css
```

```
images
├── commercial.png
├── delete-icon.png
├── dislike-active.png
├── dislike-icon.png
├── foto.png
├── history.png
├── home.png
├── like-active.png
├── like-icon.png
├── logo.png
├── monik.png
├── review.png
├── review_ava.png
├── star-empty.png
├── star-filled.png
├── tchk.png
├── upload.png
├── js
│   ├── rating.js
│   └── reaction.js
├── s.css
└── templates
    ├── login.html
    ├── myreviews.html
    ├── register.html
    ├── reviews.html
    └── application.properties
```



Листинг программы

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

```
// 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";
}
```

```
// Метод для добавления нового отзыва
@PostMapping("/reviews")
public String submitReview(
    @RequestParam String content,
    @RequestParam int rating,
    HttpSession session,
    Model model) {
    User loggedInUser = (User) session.getAttribute("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("error", "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(LocalDate.now());
    review.setUpdatedAt(LocalDate.now());

    reviewsRepository.save(review); // Сохранение отзыва в базе данных
    model.addAttribute("message", "Review submitted successfully!");
    return "redirect:/reviews";
}
```

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

    User loggedInUser = (User) session.getAttribute("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";
}
```

```
// Метод для получения всех реакций текущего пользователя
@GetMapping
public ResponseEntity<?> getUserReactions(HttpSession session) {
    User loggedInUser = (User) session.getAttribute("user"); // Получение текущего пользователя из сессии
    if (loggedInUser == null) {
        return ResponseEntity.status(401).body(Map.of("message", "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
}
```

```

// Метод для добавления или обновления реакции
@PostMapping
public ResponseEntity<?> addReaction(@RequestBody ReactionRequest request, HttpSession session) {
    User loggedInUser = (User) session.getAttribute("user"); // Получение текущего пользователя из сессии
    if (loggedInUser == null) {
        return ResponseEntity.status(401).body(Map.of("message", "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();
}

```



Enter your review here...



Send



Жесть!



Классно!



Круто!



kek
Жесть!



www
Классно!





qq

Крыто!



qq

qwe

