ФГБОУ ВПО Уральский государственный горный университет

Инженерно-экономический факультет

Кафедра информ@тики

Курсовой проект

По дисциплине «Технологии программирования» На тему «Готовый сервис компьютерного клуба»

Выполнил:

Студент гр. АУБП-22-2

Копылов Д. А.

Проверила:

ст. преп. каф. Информатики

Волкова Е.А.

Екатеринбург, 2025г.

Содержание

[Постановка задачи 3](#_Toc199373281)

[Характеристика объекта и проблемной области 3](#_Toc199373282)

[Постановка целей и задач автоматизации/информатизации 3](#_Toc199373283)

[Краткое техническое задание 3](#_Toc199373284)

[Проектирование системы 5](#_Toc199373285)

[Моделирование системы 5](#_Toc199373286)

[Проектирование интерфейса 7](#_Toc199373287)

[Разработка системы 9](#_Toc199373288)

[Выбор средств реализации 9](#_Toc199373289)

[Структура проекта 10](#_Toc199373290)

[Реализация 11](#_Toc199373291)

# Постановка задачи

## Характеристика объекта и проблемной области

Объектом автоматизации является современный компьютерный клуб, предоставляющий услуги по аренде игровых компьютеров и консолей, а также проведение различных мероприятий для посетителей. В условиях роста популярности киберспорта и цифровых развлечений такие клубы сталкиваются с необходимостью повышения эффективности управления бронированиями, оплаты услуг, учёта пользователей и проведения мероприятий.   
  
Проблемная область включает в себя обработку большого количества заявок на бронирование ресурсов клуба, организацию мероприятий и турниров, взаимодействие с пользователями и контроль за выполнением оплат. Ручное ведение таких процессов приводит к ошибкам, снижению качества сервиса и увеличению трудозатрат персонала. Поэтому возникает задача разработки специализированного программного обеспечения для автоматизации бизнес-процессов клуба.

## Постановка целей и задач автоматизации/информатизации

Целью данного курсового проекта является автоматизация ключевых бизнес-процессов современного компьютерного клуба посредством разработки многоуровневого веб-приложения. Основная задача автоматизации — повышение эффективности управления бронированиями игровых ресурсов, проведением мероприятий, учётом пользователей и оплат.   
Для достижения этой цели требуется решить следующие задачи:  
- автоматизировать процесс регистрации и авторизации пользователей с разграничением прав доступа;  
- реализовать интерфейс для бронирования компьютеров и игровых консолей;  
- обеспечить возможность проведения и управления мероприятиями клуба (турниры, события);  
- внедрить учёт оплат и статистику по оплатам;  
- предоставить администраторам инструменты для управления всеми ключевыми сущностями системы (пользователи, ресурсы, бронирования, оплаты);  
- реализовать удобный и интуитивно понятный пользовательский интерфейс.

## Краткое техническое задание

Приложение должно соответствовать следующим требованиям:  
• Пользовательские роли: реализовать разделение прав доступа для «Администратора» и «Пользователя».  
• Регистрация и вход: обеспечить возможность регистрации новых пользователей и входа с валидацией данных.  
• Бронирование ресурсов: пользователь может создавать, просматривать и отменять бронирования компьютеров и консолей на определённую дату и время.  
• Проведение и управление мероприятиями: администратор может создавать события (турниры, встречи), пользователи — записываться на участие.  
• Учёт оплат: реализовать функционал для фиксации оплаты услуг, отображение истории платежей.  
• Статистика: администратор видит статистику по бронированиям, пользователям, оплатам.  
• Требования к интерфейсу: система должна быть реализована в виде веб-приложения с использованием фирменных цветов клуба, основное представление — адаптивные страницы.  
• Безопасность: данные пользователей должны храниться с использованием шифрования паролей; доступ к функциям управления — только для роли администратора.

# Проектирование системы

## Моделирование системы

Система построена по принципам многоуровневой архитектуры (MVC), включает уровни: представления (Thymeleaf), бизнес-логики (Spring Boot контроллеры), доступа к данным (Spring Data JPA).  
В системе реализованы основные бизнес-процессы:  
- Регистрация/авторизация пользователей  
- Бронирование ресурсов (компьютеры, консоли)  
- Управление мероприятиями и участниками  
- Учёт и контроль оплат  
- Управление данными администратором  
  
Диаграмма вариантов использования (Use Case UML) представлена на рис. 2.1 (рекомендуется вставить диаграмму в соответствующее место).

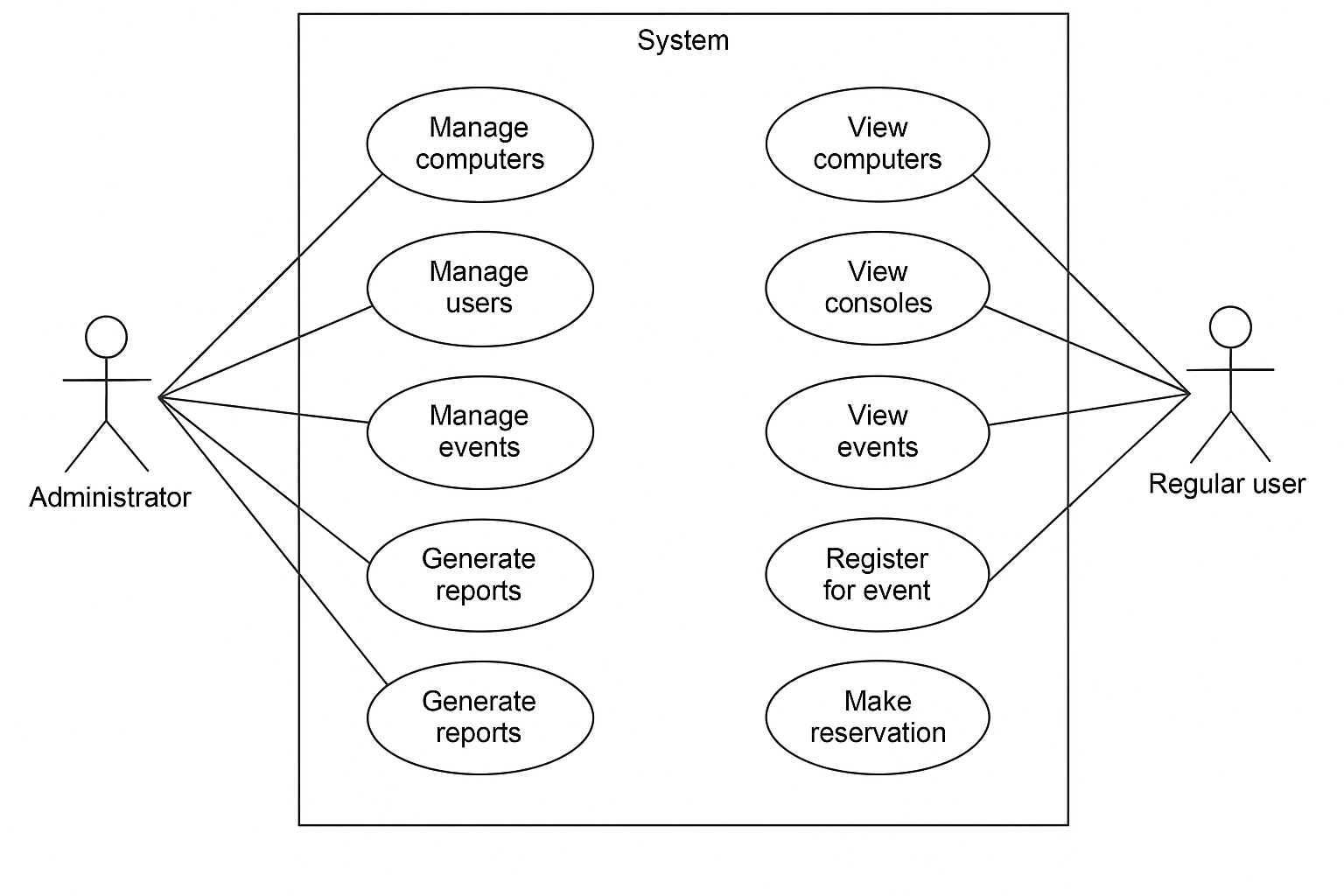


Рисунок UML-диаграмма вариантов использования системы

Для реализации системы используется шаблон MVC (Model-View-Controller), обеспечивающий разделение представления, бизнес-логики и работы с данными

Основные сущности базы данных:  
- User (Пользователь)  
- Role (Роль пользователя)  
- Computer (Компьютер)  
- Console (Игровая консоль)  
- Reservation (Бронирование)  
- Payment (Платёж)  
- Event (Мероприятие)  
- EventParticipant (Участник события)  
- Review (Отзыв)  
  
Структура БД приведена на ER-диаграмме (см. рис. 2.2). Связи между сущностями реализованы через ключи: один-ко-многим (User-Reservation, User-Review), многие-ко-многим (Event-EventParticipant).

 **User** связан с **Reservation** (один-ко-многим).

 **Reservation** связан с **Computer** и **Console** (через resource\_type/resource\_id).

 **User** связан с **Review** (один-ко-многим).

 **Reservation** связан с **Payment** (один-ко-многим).

 **User** и **Event** связаны через промежуточную таблицу **EventParticipant** (многие-ко-многим).

 **User** связан с **Role** (многие пользователей к одной роли).

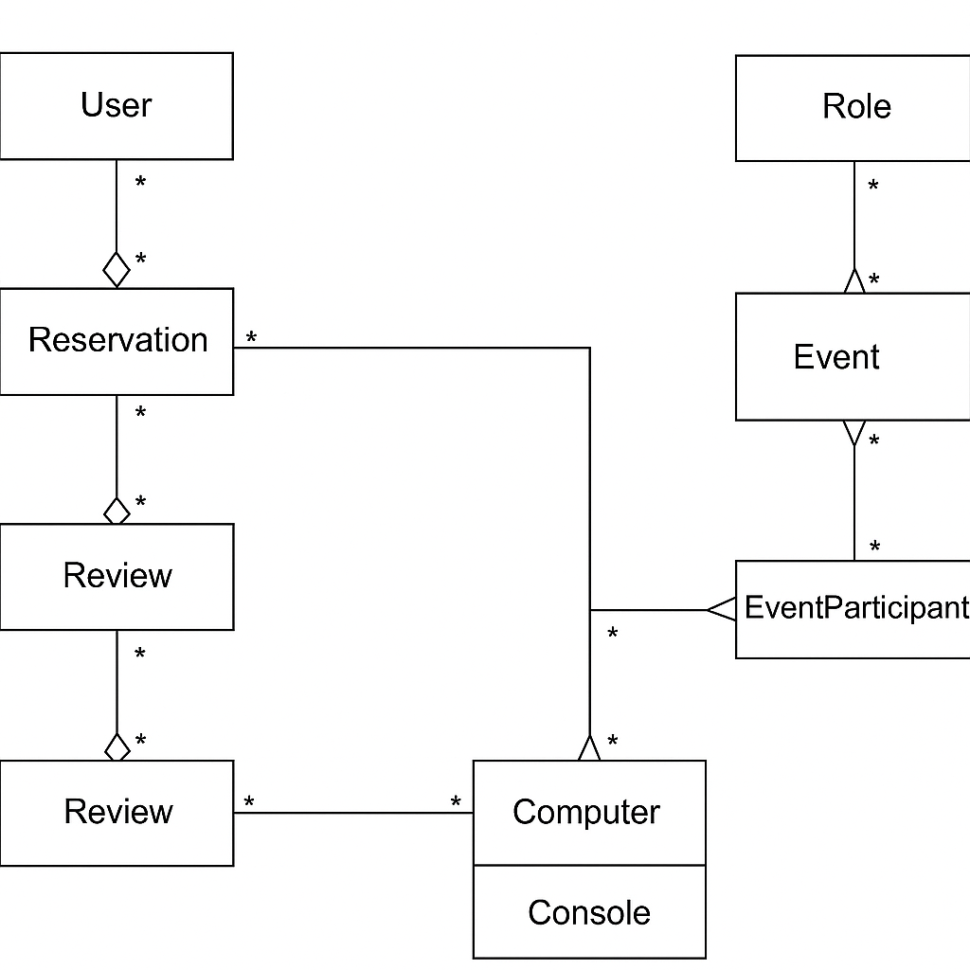


Рисунок ER-диаграмма базы данных

Описание таблиц и ключевых полей:  
User: id, username, password, email, role\_id  
Role: id, name  
Computer: id, name, specs, available  
Console: id, type, available  
Reservation: id, user\_id, resource\_type, resource\_id, start\_time, end\_time, status  
Payment: id, user\_id, reservation\_id, amount, date, method  
Event: id, title, date, description  
EventParticipant: id, event\_id, user\_id  
Review: id, user\_id, text, date, rating

## Проектирование интерфейса

Интерфейс реализован с использованием шаблонов Thymeleaf. Для пользователей доступны страницы:  
- Главная страница  
- Страница бронирования  
- Страница профиля  
- Список событий и регистрация на события  
- Просмотр и оплата бронирований  
  
Для администратора предусмотрены специальные представления для управления.

Скриншоты интерфейса:

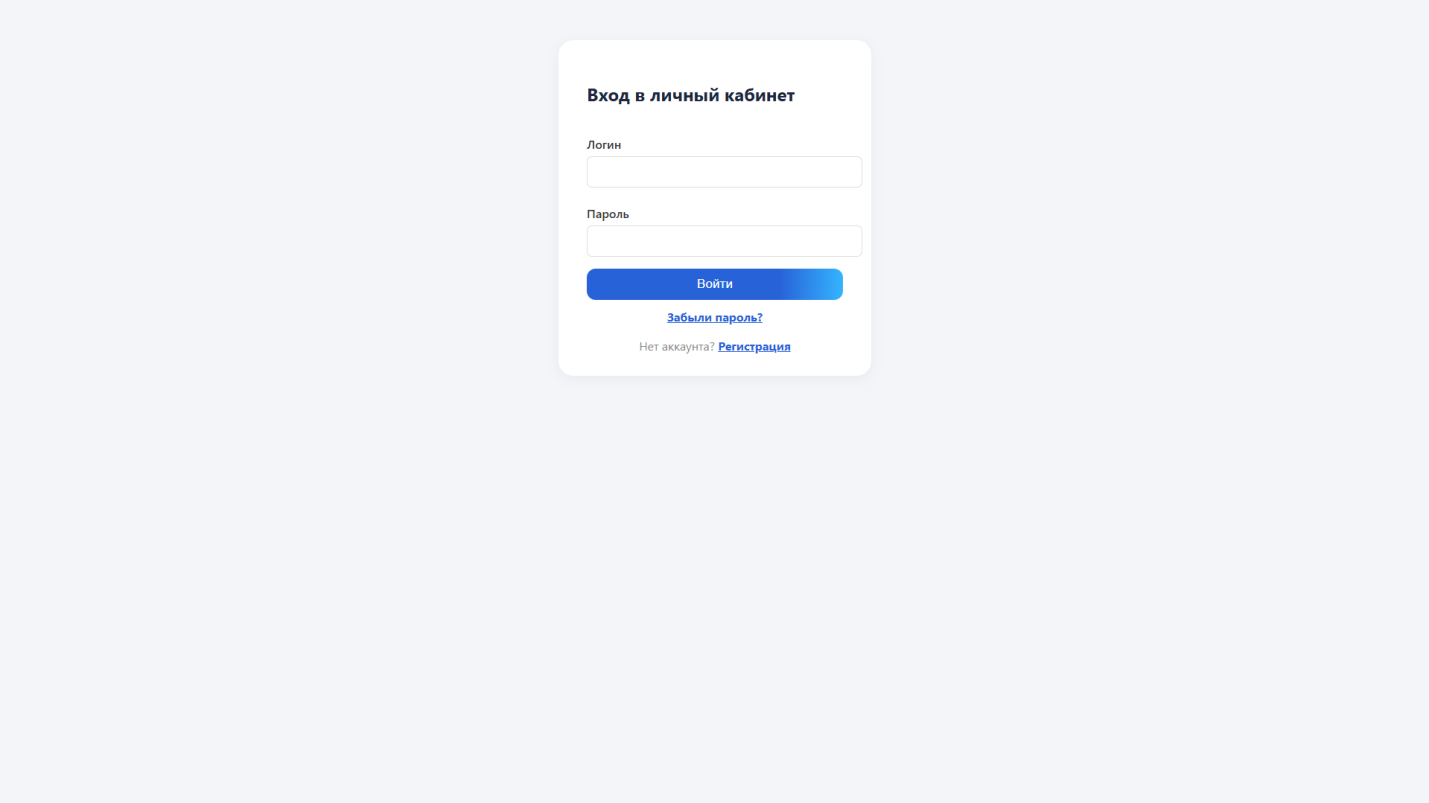


Рисунок Авторизация

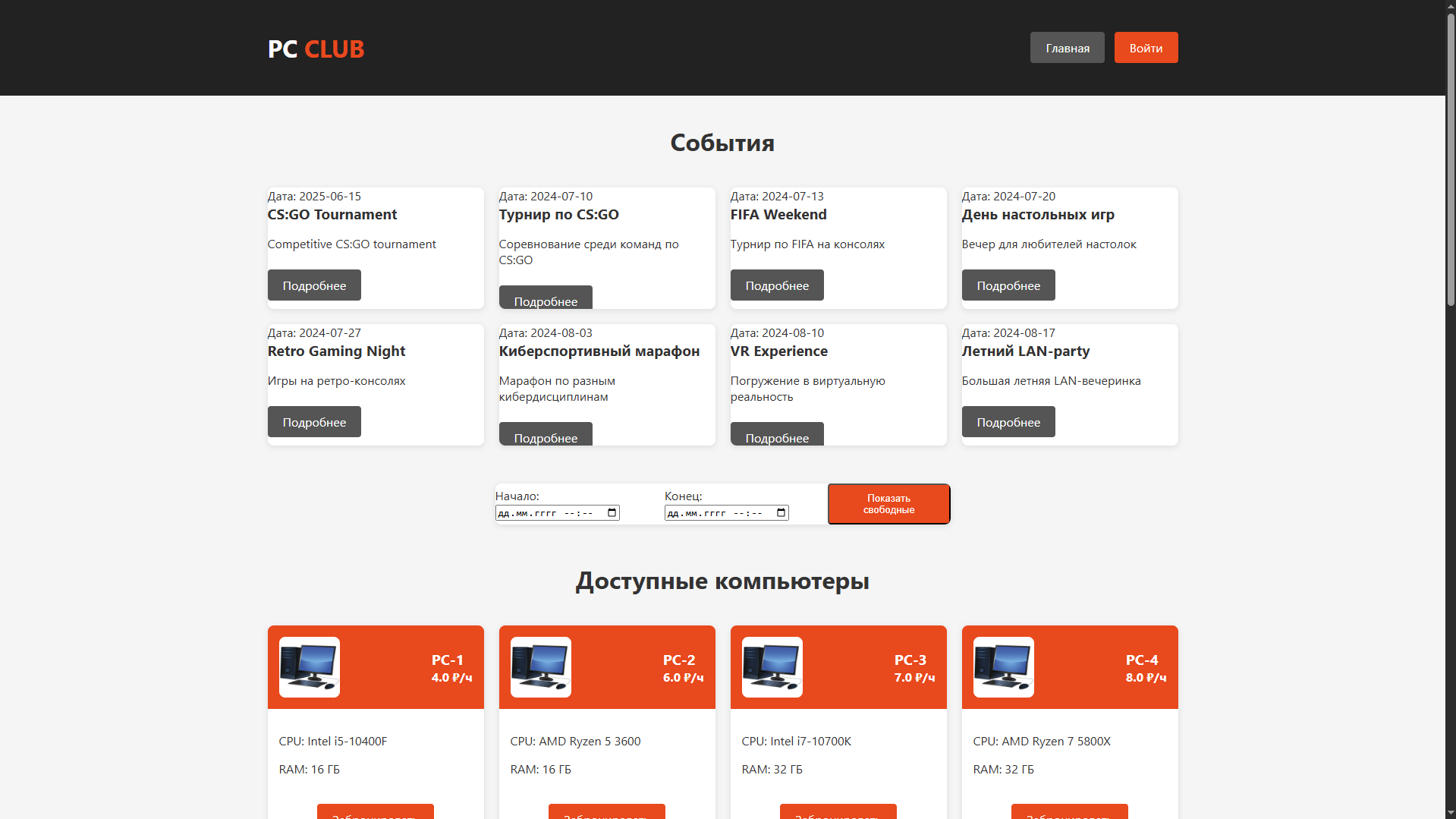


Рисунок Главная страница

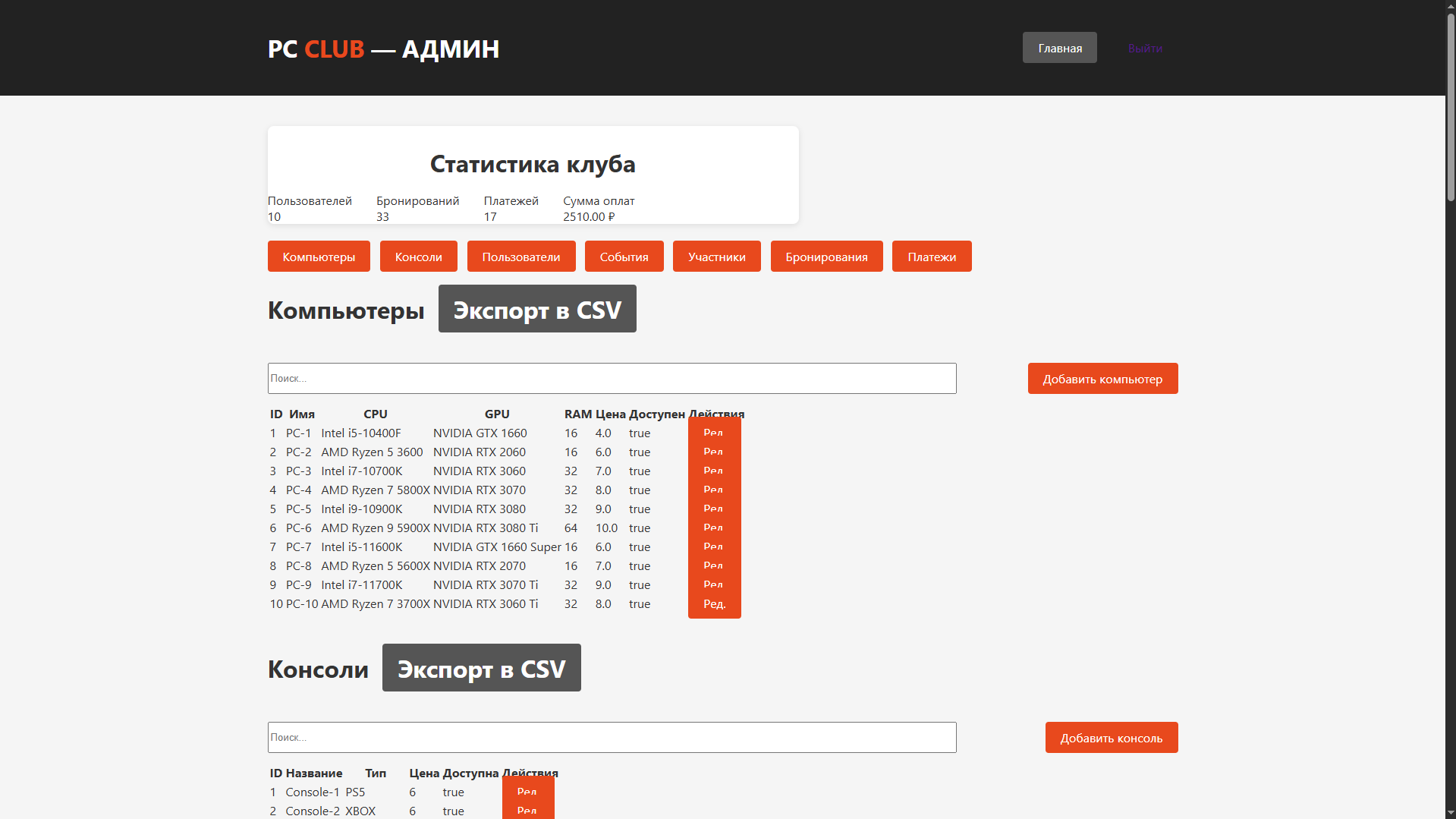


Рисунок Админ-панель

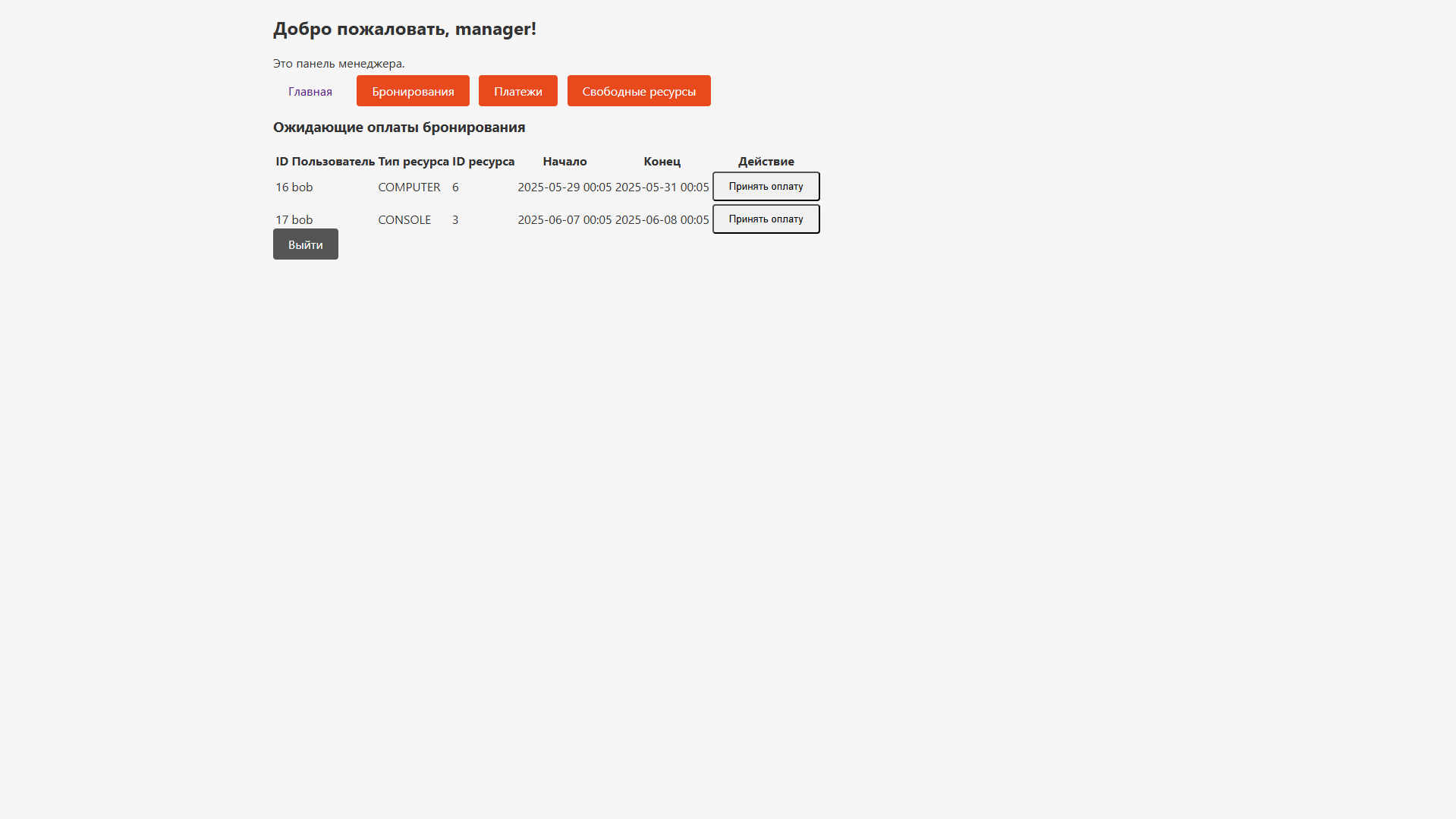


Рисунок Панель менеджера

# Разработка системы

## Выбор средств реализации

В качестве языка программирования выбран Java 21 благодаря его стабильности и широкому распространению для корпоративных решений. Веб-фреймворк — Spring Boot 3.5.0, который обеспечивает быстрое создание безопасных, масштабируемых веб-приложений с поддержкой шаблонов MVC, интеграцией с ORM через Spring Data JPA и гибкой системой валидации. В качестве СУБД применяется реляционная база данных (например, PostgreSQL или MySQL — выбор конкретного движка зависит от окружения), доступ к данным реализован через ORM (JPA/Hibernate). Для шаблонов интерфейса используется Thymeleaf, что обеспечивает генерацию адаптивных HTML-страниц с поддержкой динамического содержимого. Для обмена данными с внешними сервисами и клиентской частью используется формат JSON (через библиотеку Gson).

## Структура проекта

Структура проекта включает в себя следующие основные директории и файлы:

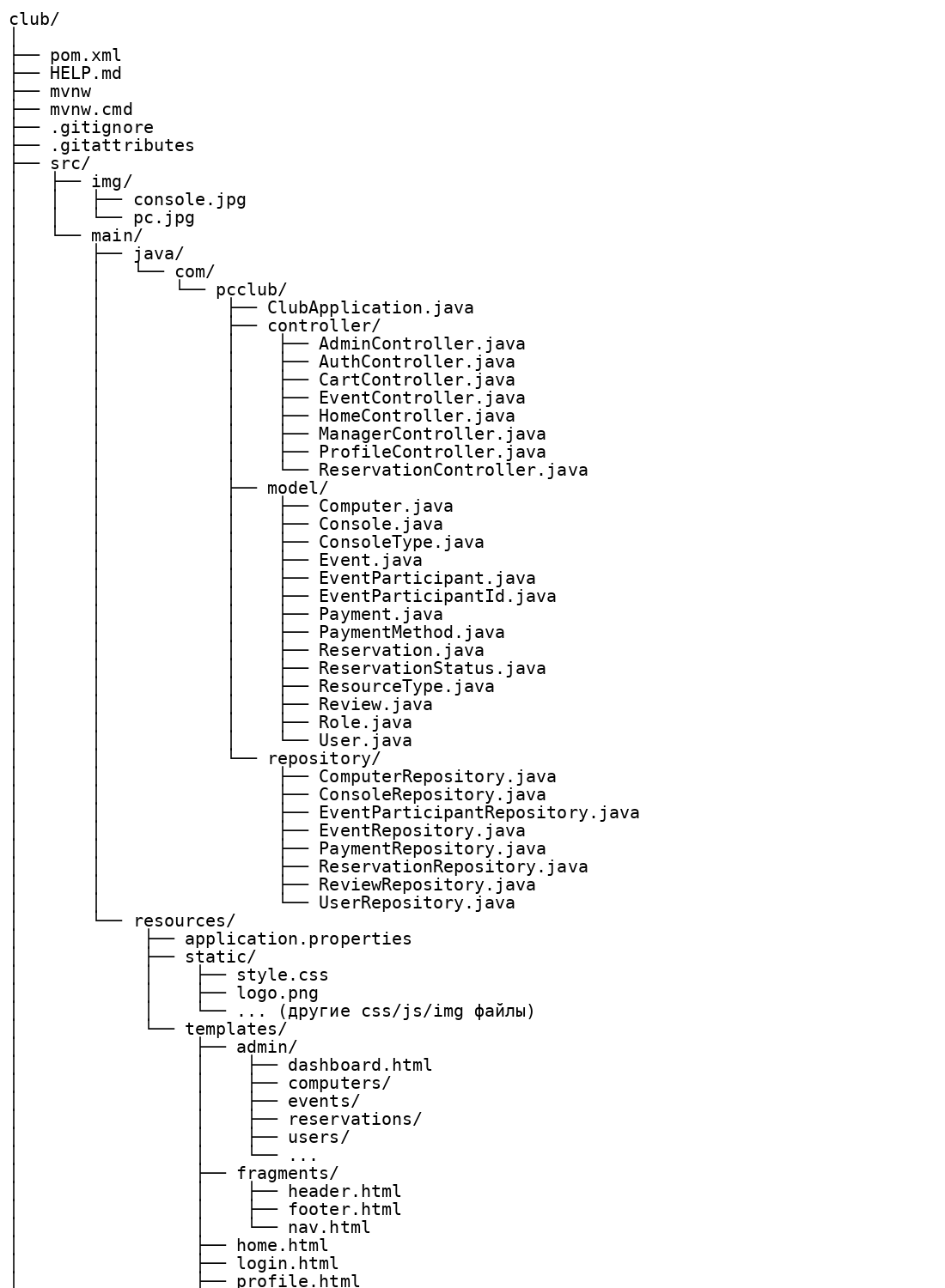


Рисунок дерево структуры проекта

UML-диаграмма классов, построенная средствами среды разработки, иллюстрирует связи между основными сущностями и классами проекта.

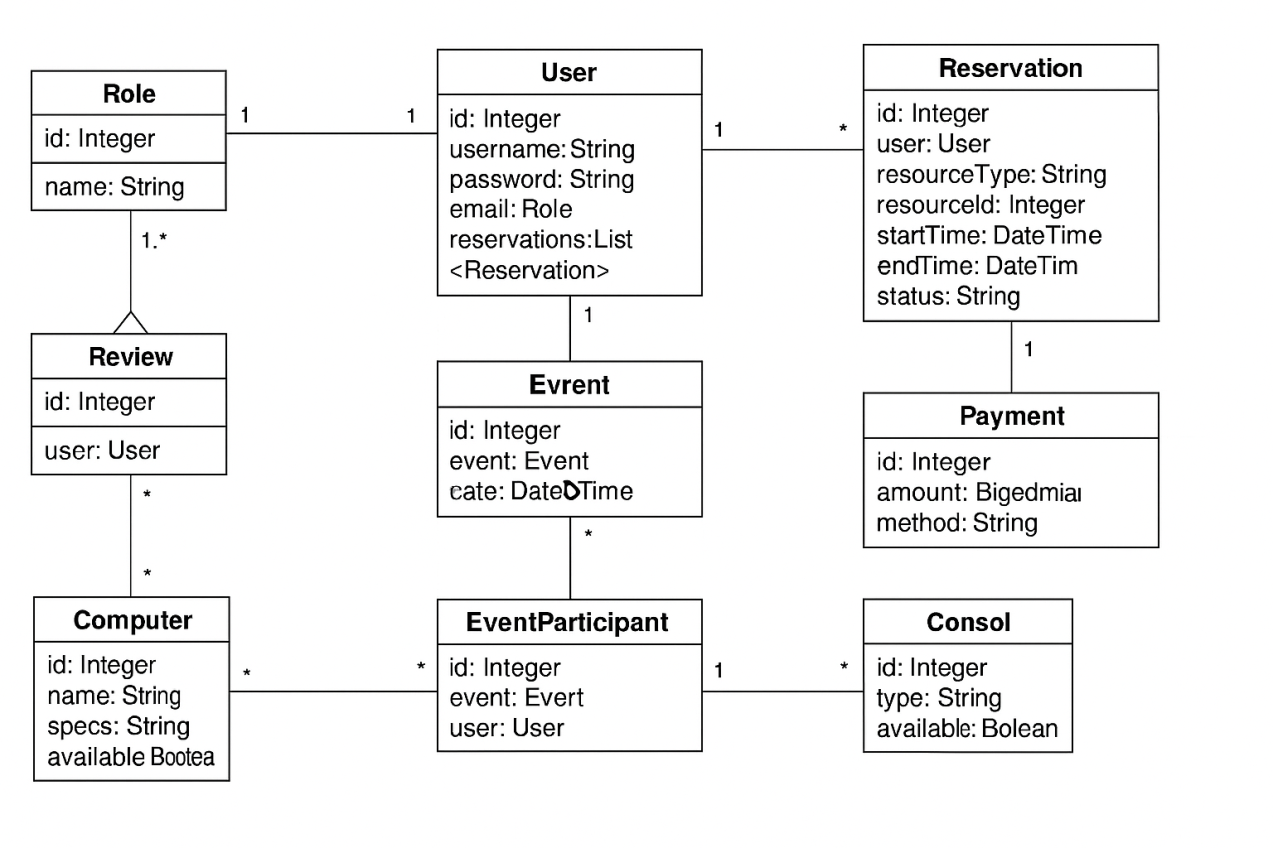


Рисунок UML-диаграмма классов

## Реализация

Ключевые классы приложения реализуют основные функции системы. Ниже приведены примеры листингов:

**Всего:**

* 1 основной класс приложения (ClubApplication.java)
* 8 контроллеров
* 13 моделей (включая вспомогательные классы и enum)
* 8 репозиториев

ClubApplication.java

package com.pcclub;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class ClubApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(ClubApplication.class, args);  
 }  
}

controller/AdminController.java

package com.pcclub.controller;  
  
import com.pcclub.model.\*;  
import com.pcclub.repository.\*;  
import jakarta.servlet.http.HttpSession;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.ModelAttribute;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.ResponseBody;  
import org.springframework.http.ResponseEntity;  
import org.springframework.http.HttpHeaders;  
import org.springframework.http.MediaType;  
import java.nio.charset.StandardCharsets;  
import java.util.stream.Collectors;  
import java.util.List;  
import org.springframework.web.bind.annotation.RequestParam;  
  
@Controller  
public class AdminController {  
  
 @Autowired  
 private ComputerRepository computerRepo;  
 @Autowired private ConsoleRepository consoleRepo;  
 @Autowired private UserRepository userRepo;  
 @Autowired private EventRepository eventRepo;  
 @Autowired private EventParticipantRepository partRepo;  
 @Autowired private ReservationRepository resRepo;  
 @Autowired  
 private PaymentRepository paymentRepo;  
 @Autowired private ReviewRepository reviewRepo;  
  
 @GetMapping("/admin/dashboard")  
 public String adminDashboard(HttpSession session, Model model) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 model.addAttribute("username", session.getAttribute("username"));  
  
 // load all tables  
 model.addAttribute("computers", computerRepo.findAll());  
 model.addAttribute("consoles", consoleRepo.findAll());  
 model.addAttribute("users", userRepo.findAll());  
 model.addAttribute("events", eventRepo.findAll());  
 model.addAttribute("participants", partRepo.findAll());  
 model.addAttribute("reservations", resRepo.findAll());  
 model.addAttribute("payments", paymentRepo.findAll());  
 model.addAttribute("reviews", reviewRepo.findAll());  
  
 // Статистика  
 long userCount = userRepo.count();  
 long reservationCount = resRepo.count();  
 long paymentCount = paymentRepo.count();  
 java.math.BigDecimal totalPayments = paymentRepo.findAll().stream()  
 .map(p -> p.getAmount() == null ? java.math.BigDecimal.ZERO : new java.math.BigDecimal(p.getAmount().toString()))  
 .reduce(java.math.BigDecimal.ZERO, java.math.BigDecimal::add);  
 model.addAttribute("userCount", userCount);  
 model.addAttribute("reservationCount", reservationCount);  
 model.addAttribute("paymentCount", paymentCount);  
 model.addAttribute("totalPayments", totalPayments);  
  
 return "admin/dashboard";  
 }  
  
 // --- Компьютеры ---  
 @GetMapping("/admin/computers/edit/{id}")  
 public String editComputerForm(@PathVariable Integer id, HttpSession session, Model model) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 model.addAttribute("computer", computerRepo.findById(id).orElseThrow());  
 return "admin/computers/edit";  
 }  
 @PostMapping("/admin/computers/edit/{id}")  
 public String updateComputer(@PathVariable Integer id, HttpSession session, @ModelAttribute Computer computer) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 computer.setId(id);  
 computerRepo.save(computer);  
 return "redirect:/admin/dashboard";  
 }  
 @GetMapping("/admin/computers/create")  
 public String createComputerForm(HttpSession session, Model model) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 model.addAttribute("computer", new Computer());  
 return "admin/computers/create";  
 }  
 @PostMapping("/admin/computers/create")  
 public String createComputer(HttpSession session, @ModelAttribute Computer computer) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 computerRepo.save(computer);  
 return "redirect:/admin/dashboard";  
 }  
  
 // --- Консоли ---  
 @GetMapping("/admin/consoles/edit/{id}")  
 public String editConsoleForm(@PathVariable Integer id, HttpSession session, Model model) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 model.addAttribute("console", consoleRepo.findById(id).orElseThrow());  
 return "admin/consoles/edit";  
 }  
 @PostMapping("/admin/consoles/edit/{id}")  
 public String updateConsole(@PathVariable Integer id, HttpSession session, @ModelAttribute Console console) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 console.setId(id);  
 consoleRepo.save(console);  
 return "redirect:/admin/dashboard";  
 }  
 @GetMapping("/admin/consoles/create")  
 public String createConsoleForm(HttpSession session, Model model) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 model.addAttribute("console", new Console());  
 return "admin/consoles/create";  
 }  
 @PostMapping("/admin/consoles/create")  
 public String createConsole(HttpSession session, @ModelAttribute Console console) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 consoleRepo.save(console);  
 return "redirect:/admin/dashboard";  
 }  
  
 // --- Пользователи ---  
 @GetMapping("/admin/users/edit/{id}")  
 public String editUserForm(@PathVariable Integer id, HttpSession session, Model model) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 model.addAttribute("user", userRepo.findById(id).orElseThrow());  
 return "admin/users/edit";  
 }  
 @PostMapping("/admin/users/edit/{id}")  
 public String updateUser(@PathVariable Integer id, HttpSession session, @ModelAttribute User user) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 user.setId(id);  
 userRepo.save(user);  
 return "redirect:/admin/dashboard";  
 }  
 @PostMapping("/admin/users/delete/{id}")  
 public String deleteUser(@PathVariable Integer id, HttpSession session) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 userRepo.deleteById(id);  
 return "redirect:/admin/dashboard";  
 }  
 @GetMapping("/admin/users/create")  
 public String createUserForm(HttpSession session, Model model) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 model.addAttribute("user", new User());  
 return "admin/users/create";  
 }  
 @PostMapping("/admin/users/create")  
 public String createUser(HttpSession session, @ModelAttribute User user) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 userRepo.save(user);  
 return "redirect:/admin/dashboard";  
 }  
  
 // --- События ---  
 @GetMapping("/admin/events/edit/{id}")  
 public String editEventForm(@PathVariable Integer id, HttpSession session, Model model) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 model.addAttribute("event", eventRepo.findById(id).orElseThrow());  
 return "admin/events/edit";  
 }  
 @PostMapping("/admin/events/edit/{id}")  
 public String updateEvent(@PathVariable Integer id, HttpSession session, @ModelAttribute Event event) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 event.setId(id);  
 eventRepo.save(event);  
 return "redirect:/admin/dashboard";  
 }  
 @GetMapping("/admin/events/create")  
 public String createEventForm(HttpSession session, Model model) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 model.addAttribute("event", new Event());  
 return "admin/events/create";  
 }  
 @PostMapping("/admin/events/create")  
 public String createEvent(HttpSession session, @ModelAttribute Event event) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 eventRepo.save(event);  
 return "redirect:/admin/dashboard";  
 }  
  
 // --- Участники событий ---  
 @GetMapping("/admin/event-participants/edit/{eventId}/{userId}")  
 public String editParticipantForm(@PathVariable Integer eventId, @PathVariable Integer userId, HttpSession session, Model model) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 EventParticipantId pid = new EventParticipantId(eventId, userId);  
 model.addAttribute("participant", partRepo.findById(pid).orElseThrow());  
 return "admin/event-participants/edit";  
 }  
 @PostMapping("/admin/event-participants/edit/{eventId}/{userId}")  
 public String updateParticipant(@PathVariable Integer eventId, @PathVariable Integer userId, HttpSession session, @ModelAttribute EventParticipant participant) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 EventParticipantId pid = new EventParticipantId(eventId, userId);  
 participant.setId(pid);  
 partRepo.save(participant);  
 return "redirect:/admin/dashboard";  
 }  
 @GetMapping("/admin/participants/create")  
 public String createParticipantForm(HttpSession session, Model model) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 model.addAttribute("participant", new EventParticipant());  
 model.addAttribute("events", eventRepo.findAll());  
 model.addAttribute("users", userRepo.findAll());  
 return "admin/event-participants/create";  
 }  
 @PostMapping("/admin/participants/create")  
 public String createParticipant(HttpSession session, @ModelAttribute EventParticipant participant) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 partRepo.save(participant);  
 return "redirect:/admin/dashboard";  
 }  
  
 // --- Бронирования ---  
 @GetMapping("/admin/reservations/edit/{id}")  
 public String editReservationForm(@PathVariable Integer id, HttpSession session, Model model) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 model.addAttribute("reservation", resRepo.findById(id).orElseThrow());  
 return "admin/reservations/edit";  
 }  
 @PostMapping("/admin/reservations/edit/{id}")  
 public String updateReservation(@PathVariable Integer id, HttpSession session, @ModelAttribute Reservation reservation) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 reservation.setId(id);  
 resRepo.save(reservation);  
 return "redirect:/admin/dashboard";  
 }  
 @GetMapping("/admin/reservations/create")  
 public String createReservationForm(HttpSession session, Model model) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 model.addAttribute("reservation", new Reservation());  
 model.addAttribute("users", userRepo.findAll());  
 model.addAttribute("computers", computerRepo.findAll());  
 model.addAttribute("consoles", consoleRepo.findAll());  
 return "admin/reservations/create";  
 }  
 @PostMapping("/admin/reservations/create")  
 public String createReservation(HttpSession session, @ModelAttribute Reservation reservation, Model model) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 resRepo.save(reservation);  
 return "redirect:/admin/dashboard";  
 }  
  
 // --- Платежи ---  
 @GetMapping("/admin/payments/edit/{id}")  
 public String editPaymentForm(@PathVariable Integer id, HttpSession session, Model model) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 model.addAttribute("payment", paymentRepo.findById(id).orElseThrow());  
 return "admin/payments/edit";  
 }  
 @PostMapping("/admin/payments/edit/{id}")  
 public String updatePayment(@PathVariable Integer id, HttpSession session, @ModelAttribute Payment payment) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 payment.setId(id);  
 paymentRepo.save(payment);  
 return "redirect:/admin/dashboard";  
 }  
 @GetMapping("/admin/payments/create")  
 public String createPaymentForm(HttpSession session, Model model) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 model.addAttribute("payment", new Payment());  
 model.addAttribute("reservations", resRepo.findAll());  
 return "admin/payments/create";  
 }  
 @PostMapping("/admin/payments/create")  
 public String createPayment(HttpSession session, @ModelAttribute Payment payment) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 paymentRepo.save(payment);  
 return "redirect:/admin/dashboard";  
 }  
  
 // --- Экспорт ---  
 @GetMapping("/admin/export/users")  
 @ResponseBody  
 public ResponseEntity<byte[]> exportUsersCsv(HttpSession session) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return ResponseEntity.status(403).build();  
 }  
 StringBuilder sb = new StringBuilder();  
 sb.append("ID,Username,Full Name,Email,Role\n");  
 for (User u : userRepo.findAll()) {  
 sb.append(u.getId()).append(",")  
 .append(u.getUsername()).append(",")  
 .append(u.getFullName() != null ? u.getFullName() : "").append(",")  
 .append(u.getEmail() != null ? u.getEmail() : "").append(",")  
 .append(u.getRole()).append("\n");  
 }  
 byte[] csvBytes = sb.toString().getBytes(StandardCharsets.UTF\_8);  
 HttpHeaders headers = new HttpHeaders();  
 headers.setContentType(MediaType.parseMediaType("text/csv"));  
 headers.set(HttpHeaders.CONTENT\_DISPOSITION, "attachment; filename=users.csv");  
 return ResponseEntity.ok().headers(headers).body(csvBytes);  
 }  
 @GetMapping("/admin/export/reservations")  
 @ResponseBody  
 public ResponseEntity<byte[]> exportReservationsCsv(HttpSession session) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return ResponseEntity.status(403).build();  
 }  
 StringBuilder sb = new StringBuilder();  
 sb.append("ID,UserID,ResourceType,ResourceID,StartTime,EndTime,Status\n");  
 for (Reservation r : resRepo.findAll()) {  
 sb.append(r.getId()).append(",")  
 .append(r.getUser() != null ? r.getUser().getId() : "").append(",")  
 .append(r.getResourceType()).append(",")  
 .append(r.getResourceId()).append(",")  
 .append(r.getStartTime()).append(",")  
 .append(r.getEndTime()).append(",")  
 .append(r.getStatus()).append("\n");  
 }  
 byte[] csvBytes = sb.toString().getBytes(StandardCharsets.UTF\_8);  
 HttpHeaders headers = new HttpHeaders();  
 headers.setContentType(MediaType.parseMediaType("text/csv"));  
 headers.set(HttpHeaders.CONTENT\_DISPOSITION, "attachment; filename=reservations.csv");  
 return ResponseEntity.ok().headers(headers).body(csvBytes);  
 }  
 @GetMapping("/admin/export/payments")  
 @ResponseBody  
 public ResponseEntity<byte[]> exportPaymentsCsv(HttpSession session) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return ResponseEntity.status(403).build();  
 }  
 StringBuilder sb = new StringBuilder();  
 sb.append("ID,ReservationID,Amount,PaidAt,Method\n");  
 for (Payment p : paymentRepo.findAll()) {  
 sb.append(p.getId()).append(",")  
 .append(p.getReservation() != null ? p.getReservation().getId() : "").append(",")  
 .append(p.getAmount()).append(",")  
 .append(p.getPaidAt()).append(",")  
 .append(p.getMethod()).append("\n");  
 }  
 byte[] csvBytes = sb.toString().getBytes(StandardCharsets.UTF\_8);  
 HttpHeaders headers = new HttpHeaders();  
 headers.setContentType(MediaType.parseMediaType("text/csv"));  
 headers.set(HttpHeaders.CONTENT\_DISPOSITION, "attachment; filename=payments.csv");  
 return ResponseEntity.ok().headers(headers).body(csvBytes);  
 }  
 @GetMapping("/admin/export/participants")  
 @ResponseBody  
 public ResponseEntity<byte[]> exportParticipantsCsv(HttpSession session) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return ResponseEntity.status(403).build();  
 }  
 StringBuilder sb = new StringBuilder();  
 sb.append("EventID,UserID,RegisteredAt,Result\n");  
 for (EventParticipant p : partRepo.findAll()) {  
 sb.append(p.getId() != null ? p.getId().getEventId() : "").append(",")  
 .append(p.getId() != null ? p.getId().getUserId() : "").append(",")  
 .append(p.getRegisteredAt()).append(",")  
 .append(p.getResult() != null ? p.getResult() : "").append("\n");  
 }  
 byte[] csvBytes = sb.toString().getBytes(StandardCharsets.UTF\_8);  
 HttpHeaders headers = new HttpHeaders();  
 headers.setContentType(MediaType.parseMediaType("text/csv"));  
 headers.set(HttpHeaders.CONTENT\_DISPOSITION, "attachment; filename=participants.csv");  
 return ResponseEntity.ok().headers(headers).body(csvBytes);  
 }  
 @GetMapping("/admin/export/computers")  
 @ResponseBody  
 public ResponseEntity<byte[]> exportComputersCsv(HttpSession session) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return ResponseEntity.status(403).build();  
 }  
 StringBuilder sb = new StringBuilder();  
 sb.append("ID,Name,CPU,GPU,RAM,Price,Available\n");  
 for (Computer c : computerRepo.findAll()) {  
 sb.append(c.getId()).append(",")  
 .append(c.getName()).append(",")  
 .append(c.getCpuModel() != null ? c.getCpuModel() : "").append(",")  
 .append(c.getGpuModel() != null ? c.getGpuModel() : "").append(",")  
 .append(c.getRamGb() != null ? c.getRamGb() : "").append(",")  
 .append(c.getPrice() != null ? c.getPrice() : "").append(",")  
 .append(c.getIsAvailable() != null ? c.getIsAvailable() : "").append("\n");  
 }  
 byte[] csvBytes = sb.toString().getBytes(StandardCharsets.UTF\_8);  
 HttpHeaders headers = new HttpHeaders();  
 headers.setContentType(MediaType.parseMediaType("text/csv"));  
 headers.set(HttpHeaders.CONTENT\_DISPOSITION, "attachment; filename=computers.csv");  
 return ResponseEntity.ok().headers(headers).body(csvBytes);  
 }  
 @GetMapping("/admin/export/consoles")  
 @ResponseBody  
 public ResponseEntity<byte[]> exportConsolesCsv(HttpSession session) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return ResponseEntity.status(403).build();  
 }  
 StringBuilder sb = new StringBuilder();  
 sb.append("ID,Name,Type,Manufacturer,Price,Available\n");  
 for (Console co : consoleRepo.findAll()) {  
 sb.append(co.getId()).append(",")  
 .append(co.getName() != null ? co.getName() : "").append(",")  
 .append(co.getType() != null ? co.getType() : "").append(",")  
 .append(co.getManufacturer() != null ? co.getManufacturer() : "").append(",")  
 .append(co.getPrice() != null ? co.getPrice() : "").append(",")  
 .append(co.getIsAvailable() != null ? co.getIsAvailable() : "").append("\n");  
 }  
 byte[] csvBytes = sb.toString().getBytes(StandardCharsets.UTF\_8);  
 HttpHeaders headers = new HttpHeaders();  
 headers.setContentType(MediaType.parseMediaType("text/csv"));  
 headers.set(HttpHeaders.CONTENT\_DISPOSITION, "attachment; filename=consoles.csv");  
 return ResponseEntity.ok().headers(headers).body(csvBytes);  
 }  
  
 @PostMapping("/admin/reviews/delete/{id}")  
 public String deleteReview(@PathVariable Integer id) {  
 reviewRepo.deleteById(id);  
 return "redirect:/admin/dashboard#reviews";  
 }  
  
 @GetMapping("/admin/reviews/create")  
 public String createReviewForm(HttpSession session, Model model) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 model.addAttribute("review", new Review());  
 model.addAttribute("users", userRepo.findAll());  
 model.addAttribute("computers", computerRepo.findAll());  
 model.addAttribute("consoles", consoleRepo.findAll());  
 return "admin/reviews/create";  
 }  
 @PostMapping("/admin/reviews/create")  
 public String createReview(HttpSession session, @ModelAttribute Review review) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 reviewRepo.save(review);  
 return "redirect:/admin/dashboard#reviews";  
 }  
 @GetMapping("/admin/reviews/edit/{id}")  
 public String editReviewForm(@PathVariable Integer id, HttpSession session, Model model) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 model.addAttribute("review", reviewRepo.findById(id).orElseThrow());  
 model.addAttribute("users", userRepo.findAll());  
 model.addAttribute("computers", computerRepo.findAll());  
 model.addAttribute("consoles", consoleRepo.findAll());  
 return "admin/reviews/edit";  
 }  
 @PostMapping("/admin/reviews/edit/{id}")  
 public String updateReview(@PathVariable Integer id, HttpSession session, @ModelAttribute Review review) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 review.setId(id);  
 reviewRepo.save(review);  
 return "redirect:/admin/dashboard#reviews";  
 }  
 @GetMapping("/admin/export/reviews")  
 @ResponseBody  
 public ResponseEntity<byte[]> exportReviewsCsv(HttpSession session) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return ResponseEntity.status(403).build();  
 }  
 StringBuilder sb = new StringBuilder();  
 sb.append("ID,User,ResourceType,ResourceId,Rating,Text,CreatedAt\n");  
 for (Review r : reviewRepo.findAll()) {  
 sb.append(r.getId()).append(",")  
 .append(r.getUser() != null ? r.getUser().getUsername() : "").append(",")  
 .append(r.getResourceType()).append(",")  
 .append(r.getResourceId()).append(",")  
 .append(r.getRating()).append(",")  
 .append(r.getText() != null ? r.getText().replaceAll(",", " ") : "").append(",")  
 .append(r.getCreatedAt()).append("\n");  
 }  
 byte[] csvBytes = sb.toString().getBytes(StandardCharsets.UTF\_8);  
 HttpHeaders headers = new HttpHeaders();  
 headers.setContentType(MediaType.parseMediaType("text/csv"));  
 headers.set(HttpHeaders.CONTENT\_DISPOSITION, "attachment; filename=reviews.csv");  
 return ResponseEntity.ok().headers(headers).body(csvBytes);  
 }  
 @GetMapping("/admin/reviews/search")  
 public String searchReviews(@RequestParam String query, HttpSession session, Model model) {  
 if (!"ADMIN".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 List<Review> reviews = reviewRepo.findAll().stream().filter(r ->  
 (r.getText() != null && r.getText().toLowerCase().contains(query.toLowerCase())) ||  
 (r.getUser() != null && r.getUser().getUsername() != null && r.getUser().getUsername().toLowerCase().contains(query.toLowerCase())) ||  
 (r.getResourceType() != null && r.getResourceType().name().toLowerCase().contains(query.toLowerCase())) ||  
 (r.getResourceId() != null && r.getResourceId().toString().contains(query)) ||  
 (Integer.toString(r.getRating()).contains(query)) ||  
 (r.getCreatedAt() != null && r.getCreatedAt().toString().contains(query))  
 ).toList();  
 model.addAttribute("reviews", reviews);  
 model.addAttribute("searchQuery", query);  
 return "admin/dashboard";  
 }  
}

controller/AuthController.java

package com.pcclub.controller;  
  
import com.pcclub.model.User;  
import com.pcclub.service.UserService;  
import jakarta.servlet.http.HttpSession;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.\*;  
  
import java.security.SecureRandom;  
  
@Controller  
public class AuthController {  
  
 @Autowired  
 private UserService userService;  
  
 // Форма входа  
 @GetMapping("/login")  
 public String loginForm() {  
 return "login";  
 }  
  
 // Обработка входа  
 @PostMapping("/login")  
 public String loginSubmit(  
 @RequestParam String username,  
 @RequestParam String password,  
 HttpSession session,  
 Model model) {  
 User user = userService.authenticate(username, password);  
 if (user == null) {  
 model.addAttribute("errorMsg", "Неверное имя пользователя или пароль");  
 return "login";  
 }  
 session.setAttribute("user", user);  
 session.setAttribute("username", user.getUsername());  
 session.setAttribute("userId", user.getId());  
 session.setAttribute("role", user.getRole().toString());  
 return "redirect:/";  
 }  
  
 // Форма регистрации  
 @GetMapping("/register")  
 public String registerForm(Model model) {  
 model.addAttribute("user", new User());  
 return "register";  
 }  
  
 // Обработка регистрации  
 @PostMapping("/register")  
 public String registerSubmit(  
 @ModelAttribute User user,  
 Model model) {  
 if (userService.existsByUsername(user.getUsername())) {  
 model.addAttribute("errorMsg", "Имя пользователя уже занято");  
 return "register";  
 }  
 if (userService.existsByEmail(user.getEmail())) {  
 model.addAttribute("errorMsg", "Email уже зарегистрирован");  
 return "register";  
 }  
 userService.register(user);  
 model.addAttribute("msg", "Регистрация успешна. Пожалуйста, войдите.");  
 return "login";  
 }  
  
 @GetMapping("/logout")  
 public String logout(HttpSession session) {  
 session.invalidate();  
 return "redirect:/";  
 }  
  
 @GetMapping("/forgot-password")  
 public String forgotPasswordForm() {  
 return "forgot-password";  
 }  
  
 @PostMapping("/forgot-password")  
 public String forgotPassword(@RequestParam String email, Model model) {  
 User user = userService.findByEmail(email);  
 if (user == null) {  
 model.addAttribute("message", "Пользователь с таким email не найден");  
 return "forgot-password";  
 }  
 // Генерируем новый пароль  
 String newPassword = generatePassword(8);  
 user.setPassword(newPassword);  
 userService.update(user);  
 model.addAttribute("message", "Ваш новый пароль: " + newPassword);  
 return "forgot-password";  
 }  
  
 private String generatePassword(int length) {  
 String chars = "ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnpqrstuvwxyz23456789";  
 SecureRandom rnd = new SecureRandom();  
 StringBuilder sb = new StringBuilder(length);  
 for (int i = 0; i < length; i++) {  
 sb.append(chars.charAt(rnd.nextInt(chars.length())));  
 }  
 return sb.toString();  
 }  
}

controller/CartController.java

package com.pcclub.controller;  
  
import jakarta.servlet.http.HttpSession;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.jdbc.core.JdbcTemplate;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.\*;  
  
import java.sql.Timestamp;  
import java.time.Duration;  
import java.time.LocalDateTime;  
import java.util.List;  
import java.util.Map;  
  
@Controller  
public class CartController {  
  
 @Autowired  
 private JdbcTemplate jdbc;  
  
 // Показ корзины  
 @GetMapping("/cart")  
 public String viewCart(HttpSession session, Model model) {  
 String username = (String) session.getAttribute("username");  
 if (username == null) {  
 return "redirect:/login";  
 }  
 Integer userId = jdbc.queryForObject(  
 "SELECT id FROM users WHERE username = ?",  
 Integer.class,  
 username  
 );  
 if (userId == null) {  
 return "redirect:/login";  
 }  
  
 // Получаем все PENDING-брони  
 List<Map<String,Object>> list = jdbc.queryForList(  
 "SELECT r.id,\n" +  
 " r.resource\_type,\n" +  
 " r.start\_time,\n" +  
 " r.end\_time,\n" +  
 " r.status,\n" +  
 " COALESCE(c.name, co.name) AS resourceName,\n" +  
 " COALESCE(c.price, co.price) AS resourcePrice\n" +  
 " FROM reservations r\n" +  
 " LEFT JOIN computers c ON r.resource\_type='COMPUTER' AND r.resource\_id=c.id\n" +  
 " LEFT JOIN consoles co ON r.resource\_type='CONSOLE' AND r.resource\_id=co.id\n" +  
 " WHERE r.user\_id = ?\n" +  
 " AND r.status = 'PENDING'",  
 userId  
 );  
  
 // Считаем длительность и стоимость каждой брони  
 for (Map<String,Object> r : list) {  
 LocalDateTime st = (LocalDateTime) r.get("start\_time");  
 LocalDateTime en = (LocalDateTime) r.get("end\_time");  
 long hours = Duration.between(st, en).toHours();  
 double price = ((Number) r.get("resourcePrice")).doubleValue();  
 r.put("hours", hours);  
 r.put("cost", price \* hours);  
 }  
  
 double total = list.stream()  
 .mapToDouble(r -> ((Number) r.get("cost")).doubleValue())  
 .sum();  
  
 model.addAttribute("cartItems", list);  
 model.addAttribute("cartEmpty", list.isEmpty());  
 model.addAttribute("totalPrice", total);  
 return "cart";  
 }  
  
 // Отмена одной брони  
 @PostMapping("/cart/cancel/{reservationId}")  
 public String cancelReservation(  
 @PathVariable("reservationId") Integer reservationId,  
 HttpSession session  
 ) {  
 String username = (String) session.getAttribute("username");  
 if (username == null) {  
 return "redirect:/login";  
 }  
 Integer userId = jdbc.queryForObject(  
 "SELECT id FROM users WHERE username = ?",  
 Integer.class,  
 username  
 );  
 jdbc.update(  
 "UPDATE reservations " +  
 " SET status = 'CANCELLED' " +  
 " WHERE id = ? AND user\_id = ?",  
 reservationId, userId  
 );  
 return "redirect:/cart";  
 }  
  
 // Оплата всех PENDING-бронирований  
 @PostMapping("/cart/checkout")  
 public String checkout(HttpSession session) {  
 String username = (String) session.getAttribute("username");  
 if (username == null) {  
 return "redirect:/login";  
 }  
 // Получаем id пользователя  
 Integer userId = jdbc.queryForObject(  
 "SELECT id FROM users WHERE username = ?",  
 Integer.class,  
 username  
 );  
 // 1) Сначала получаем все PENDING-бронирования и считаем стоимость  
 List<Map<String,Object>> pending = jdbc.queryForList(  
 "SELECT r.id AS reservationId,\n" +  
 " COALESCE(c.price, co.price) AS resourcePrice,\n" +  
 " r.start\_time, r.end\_time\n" +  
 " FROM reservations r\n" +  
 " LEFT JOIN computers c ON r.resource\_type='COMPUTER' AND r.resource\_id=c.id\n" +  
 " LEFT JOIN consoles co ON r.resource\_type='CONSOLE' AND r.resource\_id=co.id\n" +  
 " WHERE r.user\_id = ?\n" +  
 " AND r.status = 'PENDING'",  
 userId  
 );  
 for (Map<String,Object> r : pending) {  
 LocalDateTime st = (LocalDateTime) r.get("start\_time");  
 LocalDateTime en = (LocalDateTime) r.get("end\_time");  
 long hours = Duration.between(st, en).toHours();  
 double price = ((Number) r.get("resourcePrice")).doubleValue();  
 double cost = price \* hours;  
 int reservationId = ((Number) r.get("reservationId")).intValue();  
 // 2) Вставляем запись об оплате  
 jdbc.update(  
 "INSERT INTO payments (reservation\_id, amount, paid\_at, method)\n" +  
 "VALUES (?, ?, ?, ?)",  
 reservationId,  
 cost,  
 Timestamp.valueOf(LocalDateTime.now()),  
 "CARD"  
 );  
 }  
 // 3) Обновляем статус бронирований на CONFIRMED  
 jdbc.update(  
 "UPDATE reservations\n" +  
 " SET status = 'CONFIRMED'\n" +  
 " WHERE user\_id = ? AND status = 'PENDING'",  
 userId  
 );  
 return "redirect:/cart";  
 }  
  
 @PostMapping("/cart/checkout-balance")  
 public String checkoutBalance(HttpSession session, Model model) {  
 String username = (String) session.getAttribute("username");  
 if (username == null) {  
 return "redirect:/login";  
 }  
 Integer userId = jdbc.queryForObject(  
 "SELECT id FROM users WHERE username = ?",  
 Integer.class,  
 username  
 );  
 // Получаем все PENDING-бронирования и считаем стоимость  
 List<Map<String,Object>> pending = jdbc.queryForList(  
 "SELECT r.id AS reservationId,\n" +  
 " COALESCE(c.price, co.price) AS resourcePrice,\n" +  
 " r.start\_time, r.end\_time\n" +  
 " FROM reservations r\n" +  
 " LEFT JOIN computers c ON r.resource\_type='COMPUTER' AND r.resource\_id=c.id\n" +  
 " LEFT JOIN consoles co ON r.resource\_type='CONSOLE' AND r.resource\_id=co.id\n" +  
 " WHERE r.user\_id = ?\n" +  
 " AND r.status = 'PENDING'",  
 userId  
 );  
 double total = 0.0;  
 for (Map<String,Object> r : pending) {  
 LocalDateTime st = (LocalDateTime) r.get("start\_time");  
 LocalDateTime en = (LocalDateTime) r.get("end\_time");  
 long hours = java.time.Duration.between(st, en).toHours();  
 double price = ((Number) r.get("resourcePrice")).doubleValue();  
 double cost = price \* hours;  
 r.put("cost", cost);  
 total += cost;  
 }  
 // Проверяем баланс  
 Double balance = jdbc.queryForObject("SELECT balance FROM users WHERE id = ?", Double.class, userId);  
 if (balance == null || balance < total) {  
 model.addAttribute("errorMsg", "Недостаточно средств на балансе для оплаты заказа.");  
 return viewCart(session, model);  
 }  
 // Списываем средства  
 jdbc.update("UPDATE users SET balance = balance - ? WHERE id = ?", total, userId);  
 // Создаём платежи и подтверждаем бронирования  
 for (Map<String,Object> r : pending) {  
 double cost = ((Number) r.get("cost")).doubleValue();  
 int reservationId = ((Number) r.get("reservationId")).intValue();  
 jdbc.update(  
 "INSERT INTO payments (reservation\_id, amount, paid\_at, method)\n" +  
 "VALUES (?, ?, ?, ?)",  
 reservationId,  
 cost,  
 Timestamp.valueOf(LocalDateTime.now()),  
 "BALANCE"  
 );  
 }  
 jdbc.update(  
 "UPDATE reservations\n" +  
 " SET status = 'CONFIRMED'\n" +  
 " WHERE user\_id = ? AND status = 'PENDING'",  
 userId  
 );  
 return "redirect:/cart";  
 }  
}

controller/EventController.java

// src/main/java/com/pcclub/controller/EventController.java  
package com.pcclub.controller;  
  
import com.pcclub.model.Event;  
import com.pcclub.model.User;  
import com.pcclub.service.UserService;  
import com.pcclub.service.EventService;  
import com.pcclub.service.EventParticipantService;  
import jakarta.servlet.http.HttpSession;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.\*;  
import org.springframework.web.servlet.mvc.support.RedirectAttributes;  
  
import java.security.Principal;  
  
@Controller  
public class EventController {  
 @Autowired  
 private UserService userService;  
 @Autowired private EventService eventService;  
 @Autowired private EventParticipantService partService;  
 @GetMapping("/events/{id}")  
 public String eventDetails(@PathVariable int id, Model model, Principal principal) {  
 Event event = eventService.getEventById(id);  
 model.addAttribute("event", event);  
  
 // Защита от NullPointerException, если пользователь не авторизован  
 if (principal != null) {  
 model.addAttribute("user", userService.findByUsername(principal.getName()));  
 }  
 return "event-details"; // <-- имя thymeleaf-шаблона в src/main/resources/templates/  
 }  
  
 // этот метод не обязателен, но можно сделать отдельную страницу /events  
 @GetMapping("/events")  
 public String eventsPage(Model model) {  
 model.addAttribute("events", eventService.getUpcomingEvents());  
 return "events";  
 }  
  
 @PostMapping("/events/register/{id}")  
 public String registerEvent(@PathVariable("id") Integer id, HttpSession session, org.springframework.web.servlet.mvc.support.RedirectAttributes redirectAttributes) {  
 User user = (User) session.getAttribute("user");  
 if (user == null) {  
 return "redirect:/login";  
 }  
 boolean alreadyRegistered = partService.isRegistered(user.getId(), id);  
 if (alreadyRegistered) {  
 redirectAttributes.addFlashAttribute("message", "Вы уже записаны на это событие!");  
 return "redirect:/";  
 } else {  
 partService.register(user.getId(), id);  
 redirectAttributes.addFlashAttribute("message", "Вы успешно записались на событие!");  
 return "redirect:/";  
 }  
 }  
}

controller/HomeController.java

package com.pcclub.controller;  
  
import com.pcclub.model.Computer;  
import com.pcclub.model.Console;  
import com.pcclub.model.Event;  
import com.pcclub.service.ComputerService;  
import com.pcclub.service.ConsoleService;  
import com.pcclub.service.EventService;  
import com.pcclub.repository.ReviewRepository;  
import com.pcclub.model.Review;  
import jakarta.servlet.http.HttpSession;  
import org.springframework.format.annotation.DateTimeFormat;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.data.domain.PageRequest;  
import org.springframework.data.domain.Sort;  
  
import java.time.LocalDateTime;  
import java.util.List;  
import java.util.stream.Collectors;  
  
@Controller  
public class HomeController {  
  
 private final ComputerService computerService;  
 private final ConsoleService consoleService;  
 private final EventService eventService;  
 private final ReviewRepository reviewRepository;  
  
 public HomeController(  
 ComputerService computerService,  
 ConsoleService consoleService,  
 EventService eventService,  
 ReviewRepository reviewRepository  
 ) {  
 this.computerService = computerService;  
 this.consoleService = consoleService;  
 this.eventService = eventService;  
 this.reviewRepository = reviewRepository;  
 }  
  
 @GetMapping("/")  
 public String home(  
 Model model,  
 HttpSession session,  
 @RequestParam(name = "startTime", required = false)  
 @DateTimeFormat(iso = DateTimeFormat.ISO.DATE\_TIME)  
 LocalDateTime startTime,  
 @RequestParam(name = "endTime", required = false)  
 @DateTimeFormat(iso = DateTimeFormat.ISO.DATE\_TIME)  
 LocalDateTime endTime  
 ) {  
 List<Computer> computers = (startTime != null && endTime != null)  
 ? computerService.getAvailableBetween(startTime, endTime)  
 : computerService.getAvailableComputers();  
  
 List<Console> consoles = (startTime != null && endTime != null)  
 ? consoleService.getAvailableBetween(startTime, endTime)  
 : consoleService.getAvailableConsoles();  
  
 // Загружаем все события  
 List<Event> events = eventService.getAllEvents();  
  
 List<Review> latestReviews = reviewRepository.findAll(PageRequest.of(0, 5, Sort.by(Sort.Direction.DESC, "createdAt"))).getContent();  
 List<String> latestReviewNames = latestReviews.stream().map(r -> {  
 if (r.getResourceType() == com.pcclub.model.ResourceType.COMPUTER) {  
 Computer c = computerService.getComputerById(r.getResourceId());  
 return c != null ? c.getName() : "ПК #" + r.getResourceId();  
 } else if (r.getResourceType() == com.pcclub.model.ResourceType.CONSOLE) {  
 Console co = consoleService.getConsoleById(r.getResourceId());  
 return co != null ? co.getName() : "Консоль #" + r.getResourceId();  
 } else {  
 return "#" + r.getResourceId();  
 }  
 }).collect(Collectors.toList());  
  
 model.addAttribute("computers", computers);  
 model.addAttribute("consoles", consoles);  
 model.addAttribute("events", events);  
 model.addAttribute("session", session);  
 model.addAttribute("startTime", startTime);  
 model.addAttribute("endTime", endTime);  
 model.addAttribute("latestReviews", latestReviews);  
 model.addAttribute("latestReviewNames", latestReviewNames);  
  
 // Сохраняем интервалы бронирования в сессии (если нужны)  
 session.setAttribute("bookingStart", startTime);  
 session.setAttribute("bookingEnd", endTime);  
  
 return "home";  
 }  
}

controller/ManagerController.java

package com.pcclub.controller;  
  
import jakarta.servlet.http.HttpSession;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import com.pcclub.model.Reservation;  
import com.pcclub.model.ReservationStatus;  
import com.pcclub.model.Payment;  
import com.pcclub.model.PaymentMethod;  
import com.pcclub.repository.ReservationRepository;  
import com.pcclub.repository.PaymentRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.transaction.annotation.Transactional;  
import java.math.BigDecimal;  
import java.time.Duration;  
import java.time.LocalDateTime;  
import java.util.List;  
import com.pcclub.repository.ComputerRepository;  
import com.pcclub.repository.ConsoleRepository;  
import org.springframework.format.annotation.DateTimeFormat;  
import com.pcclub.model.User;  
import com.pcclub.model.Computer;  
import com.pcclub.model.Console;  
import java.util.stream.Collectors;  
import java.time.LocalDate;  
  
@Controller  
public class ManagerController {  
 @Autowired  
 private ReservationRepository reservationRepository;  
 @Autowired  
 private PaymentRepository paymentRepository;  
 @Autowired  
 private ComputerRepository computerRepository;  
 @Autowired  
 private ConsoleRepository consoleRepository;  
  
 @GetMapping("/manager/dashboard")  
 public String managerDashboard(HttpSession session, Model model) {  
 if (!"MANAGER".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 model.addAttribute("username", session.getAttribute("username"));  
 // Получаем все PENDING-брони  
 List<Reservation> pendingReservations = reservationRepository.findAll().stream()  
 .filter(r -> r.getStatus() == ReservationStatus.PENDING)  
 .toList();  
 model.addAttribute("pendingReservations", pendingReservations);  
 return "manager/dashboard";  
 }  
  
 @PostMapping("/manager/confirm-payment/{reservationId}")  
 @Transactional  
 public String confirmPayment(@PathVariable Integer reservationId) {  
 Reservation reservation = reservationRepository.findById(reservationId).orElse(null);  
 if (reservation == null || reservation.getStatus() != ReservationStatus.PENDING) {  
 return "redirect:/manager/dashboard?error";  
 }  
 // Считаем стоимость  
 long hours = Duration.between(reservation.getStartTime(), reservation.getEndTime()).toHours();  
 BigDecimal pricePerHour = BigDecimal.ZERO;  
 if (reservation.getResourceType().name().equals("COMPUTER")) {  
 var computer = computerRepository.findById(reservation.getResourceId()).orElse(null);  
 if (computer != null) pricePerHour = BigDecimal.valueOf(computer.getPrice());  
 } else if (reservation.getResourceType().name().equals("CONSOLE")) {  
 var console = consoleRepository.findById(reservation.getResourceId()).orElse(null);  
 if (console != null && console.getPrice() != null) pricePerHour = new BigDecimal(console.getPrice());  
 }  
 BigDecimal total = pricePerHour.multiply(BigDecimal.valueOf(hours));  
 // Обновляем статус  
 reservation.setStatus(ReservationStatus.CONFIRMED);  
 reservationRepository.save(reservation);  
 // Создаём платёж  
 Payment payment = new Payment(reservation, total, LocalDateTime.now(), PaymentMethod.CASH);  
 paymentRepository.save(payment);  
 return "redirect:/manager/dashboard";  
 }  
  
 // (A) Все бронирования  
 @GetMapping("/manager/reservations")  
 public String allReservations(HttpSession session, Model model) {  
 if (!"MANAGER".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 List<Reservation> allReservations = reservationRepository.findAll();  
 model.addAttribute("allReservations", allReservations);  
 return "manager/reservations";  
 }  
  
 @PostMapping("/manager/cancel-reservation/{reservationId}")  
 @Transactional  
 public String cancelReservation(@PathVariable Integer reservationId) {  
 Reservation reservation = reservationRepository.findById(reservationId).orElse(null);  
 if (reservation == null) return "redirect:/manager/reservations?error";  
 if (reservation.getStatus() == ReservationStatus.PENDING || reservation.getStatus() == ReservationStatus.CONFIRMED) {  
 reservation.setStatus(ReservationStatus.CANCELLED);  
 reservationRepository.save(reservation);  
 }  
 return "redirect:/manager/reservations";  
 }  
  
 // (B) История оплат с фильтрацией  
 @GetMapping("/manager/payments")  
 public String paymentsHistory(HttpSession session, Model model,  
 @RequestParam(value = "from", required = false) @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate from,  
 @RequestParam(value = "to", required = false) @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate to,  
 @RequestParam(value = "method", required = false) String method) {  
 if (!"MANAGER".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 List<Payment> payments = paymentRepository.findAll();  
 if (from != null) {  
 payments = payments.stream().filter(p -> !p.getPaidAt().toLocalDate().isBefore(from)).collect(Collectors.toList());  
 }  
 if (to != null) {  
 payments = payments.stream().filter(p -> !p.getPaidAt().toLocalDate().isAfter(to)).collect(Collectors.toList());  
 }  
 if (method != null && !method.isEmpty()) {  
 payments = payments.stream().filter(p -> p.getMethod().name().equalsIgnoreCase(method)).collect(Collectors.toList());  
 }  
 model.addAttribute("payments", payments);  
 model.addAttribute("from", from);  
 model.addAttribute("to", to);  
 model.addAttribute("method", method);  
 return "manager/payments";  
 }  
  
 // (C) Свободные ресурсы  
 @GetMapping("/manager/available-resources")  
 public String availableResources(HttpSession session, Model model) {  
 if (!"MANAGER".equals(session.getAttribute("role"))) {  
 return "redirect:/login?error";  
 }  
 List<Computer> computers = computerRepository.findByIsAvailableTrue();  
 List<Console> consoles = consoleRepository.findByIsAvailableTrue();  
 model.addAttribute("computers", computers);  
 model.addAttribute("consoles", consoles);  
 return "manager/available-resources";  
 }  
}

controller/ProfileController.java

package com.pcclub.controller;  
  
import com.pcclub.dto.ReservationView;  
import com.pcclub.model.User;  
import com.pcclub.model.Reservation;  
import com.pcclub.model.Event;  
import com.pcclub.service.ComputerService;  
import com.pcclub.service.ConsoleService;  
import com.pcclub.service.UserService;  
import com.pcclub.service.EventParticipantService;  
import jakarta.servlet.http.HttpSession;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.\*;  
  
import java.time.LocalDateTime;  
import java.util.List;  
import java.math.BigDecimal;  
  
@Controller  
public class ProfileController {  
 @Autowired  
 private ComputerService computerService;  
 @Autowired  
 private ConsoleService consoleService;  
  
  
 @Autowired  
 private UserService userService;  
 @Autowired  
 private EventParticipantService epService;  
  
 // Показ профиля  
 @GetMapping("/profile")  
 public String getProfile(Model model, HttpSession session) {  
 Integer userId = (Integer) session.getAttribute("userId");  
 if (userId == null) {  
 return "redirect:/login";  
 }  
 User user = userService.findById(userId);  
 List<Reservation> reservations = userService.getUserReservations(userId);  
 LocalDateTime now = LocalDateTime.now();  
 List<ReservationView> rvList = reservations.stream().map(r -> {  
 String name = switch (r.getResourceType()) {  
 case COMPUTER -> computerService.getComputerById(r.getResourceId()).getName();  
 case CONSOLE -> consoleService.getConsoleById(r.getResourceId()).getName();  
 };  
 return new ReservationView(  
 r.getId(),  
 r.getResourceType(),  
 r.getResourceId(),  
 name,  
 r.getStartTime(),  
 r.getEndTime(),  
 r.getStatus()  
 );  
 }).toList();  
 List<ReservationView> futureReservations = rvList.stream()  
 .filter(r -> r.getStatus() == com.pcclub.model.ReservationStatus.CONFIRMED &&  
 (r.getStartTime().isAfter(now) ||  
 (r.getStartTime().isBefore(now) || r.getStartTime().isEqual(now)) && r.getEndTime().isAfter(now)))  
 .sorted((a, b) -> a.getStartTime().compareTo(b.getStartTime()))  
 .toList();  
 List<ReservationView> pastReservations = rvList.stream()  
 .filter(r -> !(r.getStatus() == com.pcclub.model.ReservationStatus.CONFIRMED &&  
 (r.getStartTime().isAfter(now) ||  
 (r.getStartTime().isBefore(now) || r.getStartTime().isEqual(now)) && r.getEndTime().isAfter(now))))  
 .sorted((a, b) -> b.getStartTime().compareTo(a.getStartTime()))  
 .toList();  
 List<Event> events = epService.getEventsForUser(userId);  
 model.addAttribute("user", user);  
 model.addAttribute("futureReservations", futureReservations);  
 model.addAttribute("pastReservations", pastReservations);  
 model.addAttribute("now", now);  
 model.addAttribute("events", events);  
 return "profile";  
 }  
  
 // Сохранение изменений  
 @PostMapping("/profile")  
 public String updateProfile(@ModelAttribute("user") User form, HttpSession session) {  
 Integer userId = (Integer) session.getAttribute("userId");  
 if (userId == null) {  
 return "redirect:/login";  
 }  
  
 // 1) подгружаем «живую» сущность из БД  
 User existing = userService.findById(userId);  
  
 // 2) в неё копируем только те поля, что редактирует пользователь  
 existing.setFullName(form.getFullName());  
 existing.setEmail(form.getEmail());  
 // (если у вас есть смена пароля, обработайте его отдельно)  
  
 // 3) сохраняем  
 userService.update(existing);  
  
 return "redirect:/profile";  
 }  
  
 @PostMapping("/profile/cancel/{id}")  
 public String cancelReservation(@PathVariable Integer id, HttpSession session) {  
 Integer userId = (Integer) session.getAttribute("userId");  
 if (userId == null) return "redirect:/login";  
 Reservation r = userService.getUserReservations(userId).stream()  
 .filter(res -> res.getId().equals(id))  
 .findFirst().orElse(null);  
 if (r == null) return "redirect:/profile?error";  
 LocalDateTime now = LocalDateTime.now();  
 if (r.getStatus() == com.pcclub.model.ReservationStatus.CONFIRMED &&  
 r.getStartTime().isAfter(now) &&  
 java.time.Duration.between(now, r.getStartTime()).toHours() > 1) {  
 r.setStatus(com.pcclub.model.ReservationStatus.CANCELLED);  
 userService.updateReservation(r);  
 // Возврат денег на баланс  
 User user = userService.findById(userId);  
 BigDecimal pricePerHour = BigDecimal.ZERO;  
 if (r.getResourceType() == com.pcclub.model.ResourceType.COMPUTER) {  
 pricePerHour = BigDecimal.valueOf(computerService.getComputerById(r.getResourceId()).getPrice());  
 } else if (r.getResourceType() == com.pcclub.model.ResourceType.CONSOLE) {  
 pricePerHour = BigDecimal.valueOf(consoleService.getConsoleById(r.getResourceId()).getPrice());  
 }  
 long hours = java.time.Duration.between(r.getStartTime(), r.getEndTime()).toHours();  
 BigDecimal total = pricePerHour.multiply(BigDecimal.valueOf(hours));  
 user.setBalance(user.getBalance().add(total));  
 userService.update(user);  
 }  
 return "redirect:/profile";  
 }  
}

controller/ReservationController.java

package com.pcclub.controller;  
  
import com.pcclub.model.ReservationStatus;  
import com.pcclub.model.User;  
import jakarta.servlet.http.HttpSession;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.ResponseEntity;  
import org.springframework.jdbc.core.JdbcTemplate;  
import org.springframework.web.bind.annotation.\*;  
  
import java.time.LocalDateTime;  
import java.util.List;  
import java.util.Map;  
  
@RestController  
@RequestMapping("/api/reservations")  
public class ReservationController {  
  
 @Autowired  
 private JdbcTemplate jdbc;  
  
 // 1) Создание новой брони со статусом PENDING  
 @PostMapping("/create")  
 public ResponseEntity<?> create(@RequestBody Map<String,String> p, HttpSession session) {  
 User user = (User) session.getAttribute("user");  
 if (user == null) {  
 return ResponseEntity.status(401).body(Map.of("error", "Unauthorized"));  
 }  
 String type = p.get("resourceType");  
 int id = Integer.parseInt(p.get("resourceId"));  
 LocalDateTime start = LocalDateTime.parse(p.get("startTime"));  
 LocalDateTime end = LocalDateTime.parse(p.get("endTime"));  
  
 jdbc.update(  
 "INSERT INTO reservations(user\_id, resource\_type, resource\_id, start\_time, end\_time, status) " +  
 "VALUES(?,?,?,?,?,?)",  
 user.getId(), type, id, start, end, ReservationStatus.PENDING.name()  
 );  
 return ResponseEntity.ok(Map.of("status","PENDING"));  
 }  
  
 // 2) Получение списка «PENDING»-бронирований для текущего пользователя  
 @GetMapping("/pending")  
 public List<Map<String,Object>> pending(HttpSession session) {  
 User user = (User) session.getAttribute("user");  
 if (user == null) return List.of();  
  
 return jdbc.queryForList(  
 "SELECT r.id, r.resource\_type, r.resource\_id, r.start\_time, r.end\_time, r.status, " +  
 " COALESCE(c.name, co.name) AS resourceName, " +  
 " COALESCE(c.price, co.price) AS resourcePrice " +  
 " FROM reservations r " +  
 " LEFT JOIN computers c ON r.resource\_type='COMPUTER' AND r.resource\_id=c.id " +  
 " LEFT JOIN consoles co ON r.resource\_type='CONSOLE' AND r.resource\_id=co.id " +  
 " WHERE r.user\_id=? AND r.status='PENDING'",  
 user.getId()  
 );  
 }  
}

controller/ResourceController.java

package com.pcclub.controller;  
  
import org.springframework.web.bind.annotation.\*;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.jdbc.core.JdbcTemplate;  
import java.util.\*;  
  
@RestController  
@RequestMapping("/api/resources")  
public class ResourceController {  
  
 @Autowired  
 private JdbcTemplate jdbcTemplate;  
  
 @GetMapping  
 public Map<String, Object> getResources() {  
 List<Map<String, Object>> computers = jdbcTemplate.queryForList(  
 "SELECT \* FROM computers WHERE is\_available = 1"  
 );  
  
 List<Map<String, Object>> consoles = jdbcTemplate.queryForList(  
 "SELECT \* FROM consoles WHERE is\_available = 1"  
 );  
  
 Map<String, Object> result = new HashMap<>();  
 result.put("computers", computers);  
 result.put("consoles", consoles);  
  
 return result;  
 }  
}

controller/ResourceServlet.java

package com.pcclub.controller;  
  
import com.google.gson.Gson;  
import javax.servlet.\*;  
import javax.servlet.http.\*;  
import javax.servlet.annotation.\*;  
import java.io.\*;  
import java.sql.\*;  
import java.util.\*;  
  
@WebServlet("/api/resources")  
public class ResourceServlet extends HttpServlet {  
  
 // Подключение к БД — укажи свои параметры!  
 private Connection getConnection() throws SQLException {  
 String url = "jdbc:mysql://localhost:3306/computer\_club";  
 String user = "root";  
 String pass = "root";  
 return DriverManager.getConnection(url, user, pass);  
 }  
  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {  
 response.setContentType("application/json; charset=UTF-8");  
  
 List<Map<String, Object>> computers = new ArrayList<>();  
 List<Map<String, Object>> consoles = new ArrayList<>();  
  
 try (Connection conn = getConnection()) {  
 // ПК  
 Statement stmt = conn.createStatement();  
 ResultSet rsPC = stmt.executeQuery("SELECT \* FROM computers WHERE is\_available = 1");  
 while (rsPC.next()) {  
 Map<String, Object> pc = new HashMap<>();  
 pc.put("name", rsPC.getString("name"));  
 pc.put("cpu", rsPC.getString("cpu\_model"));  
 pc.put("gpu", rsPC.getString("gpu\_model"));  
 pc.put("ram", rsPC.getInt("ram\_gb"));  
 pc.put("price", rsPC.getDouble("price"));  
 computers.add(pc);  
 }  
 rsPC.close();  
  
 // Консоли  
 ResultSet rsConsole = stmt.executeQuery("SELECT \* FROM consoles WHERE is\_available = 1");  
 while (rsConsole.next()) {  
 Map<String, Object> console = new HashMap<>();  
 console.put("name", rsConsole.getString("name"));  
 console.put("type", rsConsole.getString("type"));  
 consoles.add(console);  
 }  
 rsConsole.close();  
 stmt.close();  
 } catch (SQLException e) {  
 response.setStatus(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR);  
 response.getWriter().write("{\"error\":\"DB error\"}");  
 return;  
 }  
  
 Map<String, Object> result = new HashMap<>();  
 result.put("computers", computers);  
 result.put("consoles", consoles);  
  
 String json = new Gson().toJson(result);  
 response.getWriter().write(json);  
 }  
}

controller/ReviewController.java

package com.pcclub.controller;  
  
import com.pcclub.model.Review;  
import com.pcclub.model.ResourceType;  
import com.pcclub.model.User;  
import com.pcclub.service.ReviewService;  
import com.pcclub.service.UserService;  
import com.pcclub.service.ComputerService;  
import com.pcclub.service.ConsoleService;  
import jakarta.servlet.http.HttpSession;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.\*;  
import java.util.List;  
  
@Controller  
public class ReviewController {  
 @Autowired  
 private ReviewService reviewService;  
 @Autowired  
 private UserService userService;  
 @Autowired  
 private ComputerService computerService;  
 @Autowired  
 private ConsoleService consoleService;  
  
 @GetMapping("/reviews/{type}/{id}")  
 public String reviewsForResource(@PathVariable("type") ResourceType type,  
 @PathVariable("id") Integer resourceId,  
 Model model) {  
 List<Review> reviews = reviewService.getReviewsForResource(type, resourceId);  
 double avg = reviews.stream().mapToInt(Review::getRating).average().orElse(0.0);  
 List<String> reviewNames = reviews == null ? java.util.Collections.emptyList() : reviews.stream().map(r -> {  
 if (r.getResourceType() == com.pcclub.model.ResourceType.COMPUTER) {  
 var c = computerService.getComputerById(r.getResourceId());  
 return c != null ? c.getName() : "ПК #" + r.getResourceId();  
 } else if (r.getResourceType() == com.pcclub.model.ResourceType.CONSOLE) {  
 var co = consoleService.getConsoleById(r.getResourceId());  
 return co != null ? co.getName() : "Консоль #" + r.getResourceId();  
 } else {  
 return "#" + r.getResourceId();  
 }  
 }).collect(java.util.stream.Collectors.toList());  
 model.addAttribute("reviews", reviews);  
 model.addAttribute("avgRating", avg);  
 model.addAttribute("resourceType", type);  
 model.addAttribute("resourceId", resourceId);  
 model.addAttribute("reviewNames", reviewNames);  
 return "reviews";  
 }  
  
 @PostMapping("/reviews/add")  
 public String addReview(@ModelAttribute Review review, HttpSession session, Model model) {  
 Integer userId = (Integer) session.getAttribute("userId");  
 if (userId == null) return "redirect:/login";  
 User user = userService.findById(userId);  
 review.setUser(user);  
 reviewService.save(review);  
 return "redirect:/reviews/" + review.getResourceType() + "/" + review.getResourceId();  
 }  
  
 @GetMapping("/reviews")  
 public String allReviews(Model model) {  
 List<Review> reviews = reviewService.getAllReviews();  
 double avg = reviews.stream().mapToInt(Review::getRating).average().orElse(0.0);  
 List<String> reviewNames = reviews == null ? java.util.Collections.emptyList() : reviews.stream().map(r -> {  
 if (r.getResourceType() == com.pcclub.model.ResourceType.COMPUTER) {  
 var c = computerService.getComputerById(r.getResourceId());  
 return c != null ? c.getName() : "ПК #" + r.getResourceId();  
 } else if (r.getResourceType() == com.pcclub.model.ResourceType.CONSOLE) {  
 var co = consoleService.getConsoleById(r.getResourceId());  
 return co != null ? co.getName() : "Консоль #" + r.getResourceId();  
 } else {  
 return "#" + r.getResourceId();  
 }  
 }).collect(java.util.stream.Collectors.toList());  
 model.addAttribute("reviews", reviews);  
 model.addAttribute("avgRating", avg);  
 model.addAttribute("resourceType", null);  
 model.addAttribute("resourceId", null);  
 model.addAttribute("reviewNames", reviewNames);  
 return "reviews";  
 }  
}

controller/ScheduleController.java

package com.pcclub.controller;  
  
import com.pcclub.model.Reservation;  
import com.pcclub.model.ResourceType;  
import com.pcclub.service.UserService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import java.util.List;  
  
@Controller  
public class ScheduleController {  
 @Autowired  
 private UserService userService;  
  
 @GetMapping("/schedule/{type}/{id}")  
 public String resourceSchedule(@PathVariable("type") ResourceType type,  
 @PathVariable("id") Integer resourceId,  
 Model model) {  
 List<Reservation> reservations = userService.getReservationsForResource(type, resourceId);  
 model.addAttribute("reservations", reservations);  
 model.addAttribute("type", type);  
 model.addAttribute("resourceId", resourceId);  
 return "schedule";  
 }  
}

model/Computer.java

package com.pcclub.model;  
  
import jakarta.persistence.\*;  
  
@Entity  
@Table(name = "computers")  
public class Computer {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Integer id;  
 private String name;  
 @Column(name="cpu\_model")  
 private String cpuModel;  
 @Column(name="gpu\_model")  
 private String gpuModel;  
 @Column(name="ram\_gb")  
 private Integer ramGb;  
 @Column(name="is\_available")  
 private Boolean isAvailable;  
 private Double price;  
  
 // Getters and setters  
 public Integer getId() { return id; }  
 public void setId(Integer id) { this.id = id; }  
 public String getName() { return name; }  
 public void setName(String name) { this.name = name; }  
 public String getCpuModel() { return cpuModel; }  
 public void setCpuModel(String cpuModel) { this.cpuModel = cpuModel; }  
 public String getGpuModel() { return gpuModel; }  
 public void setGpuModel(String gpuModel) { this.gpuModel = gpuModel; }  
 public Integer getRamGb() { return ramGb; }  
 public void setRamGb(Integer ramGb) { this.ramGb = ramGb; }  
 public Boolean getIsAvailable() { return isAvailable; }  
 public void setIsAvailable(Boolean isAvailable) { this.isAvailable = isAvailable; }  
 public Double getPrice() { return price; }  
 public void setPrice(Double price) { this.price = price; }  
}

model/Console.java

package com.pcclub.model;  
  
import jakarta.persistence.\*;  
  
@Entity  
@Table(name = "consoles")  
public class Console {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Integer id;  
  
 private String name;  
 private Integer price;  
 private String manufacturer;  
  
 @Column(name = "type")  
 @Enumerated(EnumType.STRING)  
 private ConsoleType type;  
  
 @Column(name = "is\_available")  
 private Boolean isAvailable;  
  
 // Getters and setters  
 public Integer getId() { return id; }  
 public void setId(Integer id) { this.id = id; }  
  
 public String getName() { return name; }  
 public void setName(String name) { this.name = name; }  
  
 public ConsoleType getType() { return type; }  
 public void setType(ConsoleType type) { this.type = type; }  
  
 public Boolean getIsAvailable() { return isAvailable; }  
 public void setIsAvailable(Boolean isAvailable) { this.isAvailable = isAvailable; }  
  
 public Integer getPrice() {  
 return price;  
 }  
 public void setPrice(Integer price) {  
 this.price = price;  
 }  
  
 public String getManufacturer() {  
 return manufacturer;  
 }  
  
 public void setManufacturer(String manufacturer) {  
 this.manufacturer = manufacturer;  
 }  
}

model/ConsoleType.java

package com.pcclub.model;  
  
public enum ConsoleType {  
 PS5,  
 XBOX,  
 NINTENDO  
}

model/Event.java

// src/main/java/com/pcclub/model/Event.java  
package com.pcclub.model;  
  
import jakarta.persistence.\*;  
import java.time.LocalDate;  
  
@Entity  
@Table(name = "events")  
public class Event {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Integer id;  
 @Column(columnDefinition = "TEXT")  
 private String fulldescription;  
  
 private String title;  
  
 @Column(columnDefinition = "TEXT")  
 private String description;  
  
 @Column(name = "start\_date")  
 private LocalDate startDate;  
  
 @Column(name = "end\_date")  
 private LocalDate endDate;  
  
 public Integer getId() {  
 return id;  
 }  
  
 public void setId(Integer id) {  
 this.id = id;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
 public String getFulldescription() {  
 return fulldescription;  
 }  
  
 public void setFulldescription(String fulldescription) {  
 this.fulldescription = fulldescription;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 public void setDescription(String description) {  
 this.description = description;  
 }  
  
 public LocalDate getStartDate() {  
 return startDate;  
 }  
  
 public void setStartDate(LocalDate startDate) {  
 this.startDate = startDate;  
 }  
  
 public LocalDate getEndDate() {  
 return endDate;  
 }  
  
 public void setEndDate(LocalDate endDate) {  
 this.endDate = endDate;  
 }  
}

model/EventParticipant.java

// src/main/java/com/pcclub/model/EventParticipant.java  
package com.pcclub.model;  
  
import jakarta.persistence.\*;  
import java.time.LocalDateTime;  
  
@Entity  
@Table(name = "event\_participants")  
public class EventParticipant {  
  
 @EmbeddedId  
 private EventParticipantId id;  
  
 @ManyToOne  
 @MapsId("eventId")  
 @JoinColumn(name = "event\_id", nullable = false)  
 private Event event;  
  
 @ManyToOne  
 @MapsId("userId")  
 @JoinColumn(name = "user\_id", nullable = false)  
 private User user;  
  
 @Column(name = "registered\_at", nullable = false)  
 private LocalDateTime registeredAt;  
  
 @Column(name = "result")  
 private String result;  
  
 public EventParticipant() {  
 this.registeredAt = LocalDateTime.now();  
 }  
  
 public EventParticipant(EventParticipantId id, Event event, User user) {  
 this.id = id;  
 this.event = event;  
 this.user = user;  
 this.registeredAt = LocalDateTime.now();  
 }  
  
 public EventParticipantId getId() {  
 return id;  
 }  
  
 public void setId(EventParticipantId id) {  
 this.id = id;  
 }  
  
 public Event getEvent() {  
 return event;  
 }  
  
 public void setEvent(Event event) {  
 this.event = event;  
 }  
  
 public User getUser() {  
 return user;  
 }  
  
 public void setUser(User user) {  
 this.user = user;  
 }  
  
 public LocalDateTime getRegisteredAt() {  
 return registeredAt;  
 }  
  
 public void setRegisteredAt(LocalDateTime registeredAt) {  
 this.registeredAt = registeredAt;  
 }  
  
 public String getResult() {  
 return result;  
 }  
  
 public void setResult(String result) {  
 this.result = result;  
 }  
}

model/EventParticipantId.java

// src/main/java/com/pcclub/model/EventParticipantId.java  
package com.pcclub.model;  
  
import jakarta.persistence.Embeddable;  
import java.io.Serializable;  
import java.util.Objects;  
  
@Embeddable  
public class EventParticipantId implements Serializable {  
 private Integer eventId;  
 private Integer userId;  
  
 public EventParticipantId() {}  
  
 public EventParticipantId(Integer eventId, Integer userId) {  
 this.eventId = eventId;  
 this.userId = userId;  
 }  
  
 public Integer getEventId() {  
 return eventId;  
 }  
  
 public void setEventId(Integer eventId) {  
 this.eventId = eventId;  
 }  
  
 public Integer getUserId() {  
 return userId;  
 }  
  
 public void setUserId(Integer userId) {  
 this.userId = userId;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 EventParticipantId that = (EventParticipantId) o;  
 return Objects.equals(eventId, that.eventId) &&  
 Objects.equals(userId, that.userId);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(eventId, userId);  
 }  
}

model/Payment.java

package com.pcclub.model;  
  
import jakarta.persistence.\*;  
import java.math.BigDecimal;  
import java.time.LocalDateTime;  
  
@Entity  
@Table(name = "payments")  
public class Payment {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Integer id;  
  
 @ManyToOne(fetch = FetchType.LAZY)  
 @JoinColumn(name = "reservation\_id", nullable = false)  
 private Reservation reservation;  
  
 @Column(nullable = false)  
 private BigDecimal amount;  
  
 @Column(name = "paid\_at", nullable = false)  
 private LocalDateTime paidAt;  
  
 @Enumerated(EnumType.STRING)  
 @Column(nullable = false, columnDefinition = "enum('CASH','CARD')")  
 private PaymentMethod method;  
  
 // Constructors  
 public Payment() {}  
  
 public Payment(Reservation reservation, BigDecimal amount, LocalDateTime paidAt, PaymentMethod method) {  
 this.reservation = reservation;  
 this.amount = amount;  
 this.paidAt = paidAt;  
 this.method = method;  
 }  
  
 // Getters and Setters  
 public Integer getId() {  
 return id;  
 }  
  
 public void setId(Integer id) {  
 this.id = id;  
 }  
  
 public Reservation getReservation() {  
 return reservation;  
 }  
  
 public void setReservation(Reservation reservation) {  
 this.reservation = reservation;  
 }  
  
 public BigDecimal getAmount() {  
 return amount;  
 }  
  
 public void setAmount(BigDecimal amount) {  
 this.amount = amount;  
 }  
  
 public LocalDateTime getPaidAt() {  
 return paidAt;  
 }  
  
 public void setPaidAt(LocalDateTime paidAt) {  
 this.paidAt = paidAt;  
 }  
  
 public PaymentMethod getMethod() {  
 return method;  
 }  
  
 public void setMethod(PaymentMethod method) {  
 this.method = method;  
 }  
}

model/PaymentMethod.java

// src/main/java/com/pcclub/model/PaymentMethod.java  
package com.pcclub.model;  
  
public enum PaymentMethod {  
 CASH, CARD, BALANCE  
}

model/Reservation.java

package com.pcclub.model;  
  
import jakarta.persistence.\*;  
import java.time.LocalDateTime;  
  
@Entity  
@Table(name = "reservations")  
public class Reservation {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Integer id;  
 @ManyToOne(fetch = FetchType.LAZY)  
 @JoinColumn(name = "resource\_id", insertable = false, updatable = false)  
 private Computer computer;  
  
 @ManyToOne(fetch = FetchType.LAZY)  
 @JoinColumn(name = "resource\_id", insertable = false, updatable = false)  
 private Console console;  
  
 @ManyToOne(fetch = FetchType.LAZY)  
 @JoinColumn(name = "user\_id", nullable = false)  
 private User user;  
  
 public User getUser() {  
 return user;  
 }  
  
 public void setUser(User user) {  
 this.user = user;  
 }  
 @Enumerated(EnumType.STRING)  
 @Column(name = "resource\_type", nullable = false)  
 private ResourceType resourceType;  
  
 @Column(name = "resource\_id", nullable = false)  
 private Integer resourceId;  
  
 @Column(name = "start\_time", nullable = false)  
 private LocalDateTime startTime;  
  
 @Column(name = "end\_time", nullable = false)  
 private LocalDateTime endTime;  
  
 @Enumerated(EnumType.STRING)  
 @Column(nullable = false)  
 private ReservationStatus status;  
  
 public Reservation() {  
 }  
  
 public Integer getId() {  
 return id;  
 }  
  
 public void setId(Integer id) {  
 this.id = id;  
 }  
  
 public ResourceType getResourceType() {  
 return resourceType;  
 }  
  
 public void setResourceType(ResourceType resourceType) {  
 this.resourceType = resourceType;  
 }  
  
 public Integer getResourceId() {  
 return resourceId;  
 }  
  
 public void setResourceId(Integer resourceId) {  
 this.resourceId = resourceId;  
 }  
  
 public LocalDateTime getStartTime() {  
 return startTime;  
 }  
  
 public void setStartTime(LocalDateTime startTime) {  
 this.startTime = startTime;  
 }  
  
 public LocalDateTime getEndTime() {  
 return endTime;  
 }  
  
 public void setEndTime(LocalDateTime endTime) {  
 this.endTime = endTime;  
 }  
  
 public ReservationStatus getStatus() {  
 return status;  
 }  
  
 public void setStatus(ReservationStatus status) {  
 this.status = status;  
 }  
}

model/ReservationStatus.java

package com.pcclub.model;  
  
public enum ReservationStatus {  
 PENDING,  
 CONFIRMED,  
 CANCELLED  
}

model/ResourceType.java

package com.pcclub.model;  
  
public enum ResourceType {  
 COMPUTER,  
 CONSOLE  
}

model/Review.java

package com.pcclub.model;  
  
import jakarta.persistence.\*;  
import java.time.LocalDateTime;  
  
@Entity  
@Table(name = "reviews")  
public class Review {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Integer id;  
  
 @ManyToOne(fetch = FetchType.LAZY)  
 @JoinColumn(name = "user\_id", nullable = false)  
 private User user;  
  
 @Enumerated(EnumType.STRING)  
 @Column(name = "resource\_type", nullable = false)  
 private ResourceType resourceType;  
  
 @Column(name = "resource\_id", nullable = false)  
 private Integer resourceId;  
  
 @Column(nullable = false)  
 private int rating; // 1-5  
  
 @Column(columnDefinition = "TEXT")  
 private String text;  
  
 @Column(nullable = false)  
 private LocalDateTime createdAt = LocalDateTime.now();  
  
 // геттеры и сеттеры  
 public Integer getId() { return id; }  
 public void setId(Integer id) { this.id = id; }  
 public User getUser() { return user; }  
 public void setUser(User user) { this.user = user; }  
 public ResourceType getResourceType() { return resourceType; }  
 public void setResourceType(ResourceType resourceType) { this.resourceType = resourceType; }  
 public Integer getResourceId() { return resourceId; }  
 public void setResourceId(Integer resourceId) { this.resourceId = resourceId; }  
 public int getRating() { return rating; }  
 public void setRating(int rating) { this.rating = rating; }  
 public String getText() { return text; }  
 public void setText(String text) { this.text = text; }  
 public LocalDateTime getCreatedAt() { return createdAt; }  
 public void setCreatedAt(LocalDateTime createdAt) { this.createdAt = createdAt; }  
}

model/Role.java

package com.pcclub.model;  
  
public enum Role {  
 ADMIN, MANAGER, CLIENT  
}

model/User.java

package com.pcclub.model;  
  
import jakarta.persistence.\*;  
import java.time.LocalDateTime;  
  
@Entity  
@Table(name = "users")  
public class User {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Integer id;  
  
 @Column(unique = true, nullable = false)  
 private String username;  
  
 @Column(nullable = false)  
 private String password;  
  
 @Column(name = "full\_name", nullable = false)  
 private String fullName;  
  
 @Column(unique = true, nullable = false)  
 private String email;  
  
 @Enumerated(EnumType.STRING)  
 @Column(nullable = false)  
 private Role role;  
  
 @Column(name = "created\_at", nullable = false, updatable = false, insertable = false)  
 private LocalDateTime createdAt;  
  
 @Column(nullable = false)  
 private java.math.BigDecimal balance = java.math.BigDecimal.ZERO;  
  
 // ======= getters & setters =======  
 public Integer getId() { return id; }  
 public void setId(Integer id) { this.id = id; }  
 public String getUsername() { return username; }  
 public void setUsername(String username) { this.username = username; }  
 public String getPassword() { return password; }  
 public void setPassword(String password) { this.password = password; }  
 public String getFullName() { return fullName; }  
 public void setFullName(String fullName) { this.fullName = fullName; }  
 public String getEmail() { return email; }  
 public void setEmail(String email) { this.email = email; }  
 public Role getRole() { return role; }  
 public void setRole(Role role) { this.role = role; }  
 public LocalDateTime getCreatedAt() { return createdAt; }  
 public void setCreatedAt(LocalDateTime createdAt) { this.createdAt = createdAt; }  
 public java.math.BigDecimal getBalance() { return balance; }  
 public void setBalance(java.math.BigDecimal balance) { this.balance = balance; }  
}

repository/ComputerRepository.java

package com.pcclub.repository;  
  
import com.pcclub.model.Computer;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.query.Param;  
  
import java.time.LocalDateTime;  
import java.util.List;  
  
public interface ComputerRepository extends JpaRepository<Computer, Integer> {  
 List<Computer> findByIsAvailableTrue();  
 @Query("""  
 SELECT c FROM Computer c  
 WHERE c.isAvailable = true  
 AND c.id NOT IN (  
 SELECT r.resourceId FROM Reservation r  
 WHERE r.resourceType = com.pcclub.model.ResourceType.COMPUTER  
 AND r.status <> com.pcclub.model.ReservationStatus.CANCELLED  
 AND r.startTime < :end  
 AND r.endTime > :start  
 )  
 """)  
 List<Computer> findAvailableBetween(  
 @Param("start") LocalDateTime start,  
 @Param("end") LocalDateTime end  
 );  
}

repository/ConsoleRepository.java

package com.pcclub.repository;  
  
import com.pcclub.model.Console;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.query.Param;  
  
import java.time.LocalDateTime;  
import java.util.List;  
  
public interface ConsoleRepository extends JpaRepository<Console, Integer> {  
  
 // Метод для получения всех консолей, помеченных как доступные  
 List<Console> findByIsAvailableTrue();  
  
 // Метод для фильтрации по доступности в заданном интервале  
 @Query("""  
 SELECT c FROM Console c  
 WHERE c.isAvailable = true  
 AND c.id NOT IN (  
 SELECT r.resourceId FROM Reservation r  
 WHERE r.resourceType = com.pcclub.model.ResourceType.CONSOLE  
 AND r.status <> com.pcclub.model.ReservationStatus.CANCELLED  
 AND r.startTime < :end  
 AND r.endTime > :start  
 )  
 """)  
 List<Console> findAvailableBetween(  
 @Param("start") LocalDateTime start,  
 @Param("end") LocalDateTime end  
 );  
}

repository/EventParticipantRepository.java

// src/main/java/com/pcclub/repository/EventParticipantRepository.java  
package com.pcclub.repository;  
  
import com.pcclub.model.EventParticipant;  
import com.pcclub.model.EventParticipantId;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
import java.util.List;  
  
public interface EventParticipantRepository extends JpaRepository<EventParticipant, EventParticipantId> {  
 boolean existsByUserIdAndEventId(int userId, int eventId);  
 List<EventParticipant> findByIdUserId(int userId);  
}

repository/EventRepository.java

// src/main/java/com/pcclub/repository/EventRepository.java  
package com.pcclub.repository;  
  
import com.pcclub.model.Event;  
import org.springframework.data.jpa.repository.JpaRepository;  
import java.time.LocalDate;  
import java.util.List;  
  
public interface EventRepository extends JpaRepository<Event, Integer> {  
 // все активные (ближайшие) события  
 List<Event> findByEndDateGreaterThanEqualOrderByStartDate(LocalDate date);  
}

repository/PaymentRepository.java

// src/main/java/com/pcclub/repository/PaymentRepository.java  
package com.pcclub.repository;  
  
import com.pcclub.model.Payment;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface PaymentRepository extends JpaRepository<Payment, Integer> {  
}

repository/ReservationRepository.java

package com.pcclub.repository;  
  
import com.pcclub.model.Reservation;  
import com.pcclub.model.ResourceType;  
import com.pcclub.model.User;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.query.Param;  
  
import java.time.LocalDateTime;  
import java.util.List;  
  
public interface ReservationRepository extends JpaRepository<Reservation, Integer> {  
  
 @Query("""  
 SELECT r.resourceId  
 FROM Reservation r  
 WHERE r.resourceType = :type  
 AND r.status <> com.pcclub.model.ReservationStatus.CANCELLED  
 AND r.startTime < :end  
 AND r.endTime > :start  
 """)  
 List<Integer> findBookedResourceIds(  
 @Param("type") ResourceType type,  
 @Param("start") LocalDateTime start,  
 @Param("end") LocalDateTime end  
 );  
 List<Reservation> findByUserId(int userId);  
 List<Reservation> findByUser(User user);  
}

repository/ReviewRepository.java

package com.pcclub.repository;  
  
import com.pcclub.model.Review;  
import com.pcclub.model.ResourceType;  
import org.springframework.data.jpa.repository.JpaRepository;  
import java.util.List;  
  
public interface ReviewRepository extends JpaRepository<Review, Integer> {  
 List<Review> findByResourceTypeAndResourceId(ResourceType type, Integer resourceId);  
 List<Review> findByUserId(Integer userId);  
}

repository/UserRepository.java

package com.pcclub.repository;  
  
import com.pcclub.model.User;  
import org.springframework.data.jpa.repository.JpaRepository;  
import java.util.Optional;  
  
public interface UserRepository extends JpaRepository<User, Integer> {  
 Optional<User> findByUsernameAndPassword(String username, String password);  
 Optional<User> findByUsername(String username);  
 Optional<User> findByEmail(String email);  
}