Web 高级编程项目报告

目录

演唱会票务信息管理平台

1 引言

1.1 项目介绍

演唱会票务信息管理平台是一个简化的演唱会票务管理流程的系统。该平台的主要目的是为演唱会主办方提供一个全面而高效的票务信息管理解决方案。通过该平台，演唱会主办方能够实现票务信息修改、票务销售、观众的购买记录等多方面的功能，从而更好地组织演唱会活动。提高了演唱会的票务效率、用户体验以及信息统一管理。

1.2 项目相关技术

1.2.1 Spring boot 框架

演唱会票务信息管理平台采用了 Spring Boot 作为主要的开发框架。Spring Boot 是一个基于 Spring 框架的快速开发框架，通过约定大于配置的原则，简化了 Spring 应用的搭建和开发流程。以下是项目中使用的 Spring Boot 的关键特性：

简化配置： Spring Boot 通过自动配置和默认值减少了对项目的繁琐配置，使开发人员能够更专注于业务逻辑的实现。

内嵌服务器： 集成了常见的内嵌服务器（如 Tomcat），使得部署和运行应用变得更加简便。

快速开发：提供了丰富的开箱即用的功能，例如 Spring Data、Spring Security 等，加速了开发过程。

微服务支持：支持构建和部署微服务架构，使得系统更具弹性和可伸缩性。

1.2.2 Maven 库

Maven 被用作项目的依赖管理工具。它是一个强大的项目管理和构建工具，能够自动化项目的构建过程，并管理项目的依赖关系。在演唱会票务信息管理平台中，Maven 的作用包括：

依赖管理： Maven 负责管理项目所需的所有依赖库，包括 Spring Boot 相关

的库以及其他第三方库。

构建工具： 使用 Maven 进行项目的编译、打包和部署，确保项目的可维护性和可扩展性。

1.2.3 其他技术

除了 Spring Boot 和 Maven，项目还可能使用其他关键技术，例如：

Spring Data JPA： 简化数据访问层的开发，支持对象关系映射（ORM）。

Thymeleaf： 用于处理模板的视图引擎，支持在 HTML 中嵌入动态数据。

在项目中充分利用这些技术，可以帮助提高开发效率、系统的可维护性，并确保系统在性能和安全性方面得到充分考虑。

1. 3 项目开发环境

1.3.1 开发工具

为了有效地开发和维护演唱会票务信息管理平台，我采用了以下开发工具：

IntelliJ IDEA：作为主要的集成开发环境（IDE），提供了强大的代码编辑、调试和构建工具，支持 Spring Boot 项目的快速开发。

版本：IntelliJ IDEA 2023.2.2 (Ultimate Edition)

1.3.2 数据库系统

演唱会票务信息管理平台使用了以下数据库系统来存储和管理数据：

MySQL：作为关系型数据库管理系统，用于存储与演唱会、票务、用户等相关的数据。

版本：Server version: 8.0.35 MySQL Community Server - GPL

1.3.3 依赖管理与构建工具

为了有效管理项目的依赖关系和进行项目构建，选择了以下工具：

Maven：作为项目的构建工具和依赖管理工具，简化了项目的构建过程，并能够自动解决项目所需的各种依赖。

版本：apache-maven-3.6.0

2 需求分析与总体设计

2.1 需求分析

2.1.1 功能需求

演唱会票务信息管理平台主要涉及以下功能模块：注册、登录、票品信息的增删改查、票品的出售、票品的售卖记录查询。

例如：演唱会票务信息的添加

描述：管理员通过系统添加新的演唱会。

主要参与者：管理员

前置条件：管理员已登录系统。

基本流程：

1.管理员进入演唱会管理界面。

2.管理员添加新的演唱会信息，包括演唱会名称、时间、票价等。

3.管理员添加演唱会的座位图。

4.管理员保存演唱会信息。

备选流程：

如果演唱会信息填写不完整，系统拒绝保存并提示管理员完善信息。

2.1.2 性能需求

演唱会票务信息管理平台对性能有以下要求：

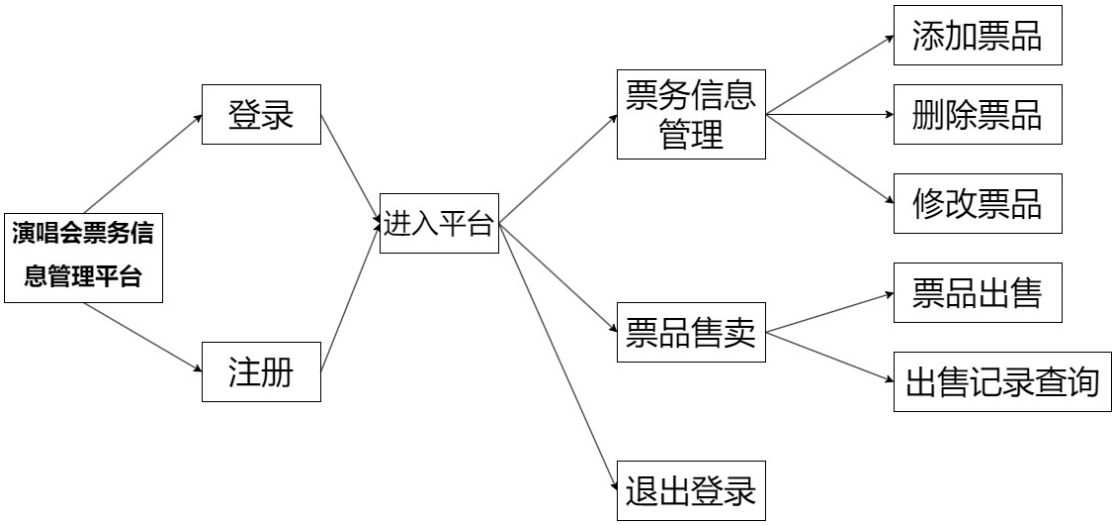
并发性：平台应能支持同时进行多个用户购票操作，确保系统响应迅速，不出现卡顿。

可扩展性：系统设计应具备良好的可扩展性，能够根据实际需求灵活扩展，保证系统的稳定性和可用性。

安全性：对未经登录的用户，拦截通过地址栏跳转系统加棉的请求；对用户信息和交易数据进行加密，确保用户数据的安全性。

2.2 系统架构设计

演唱会票务信息管理平台的主要功能如下图：

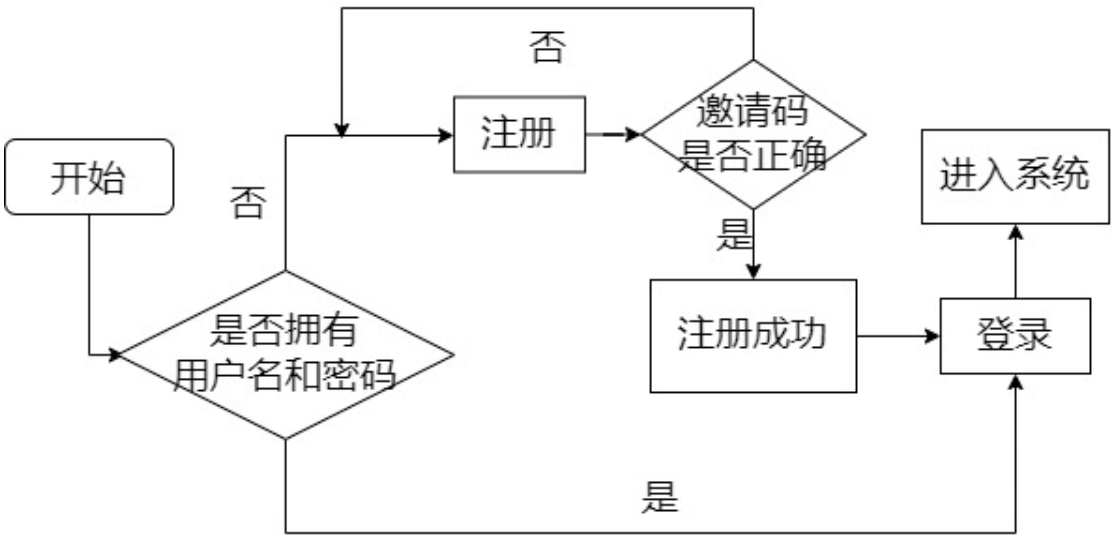


3 系统详细设计

3.1 功能模块设计

3.1.1 系统注册登录

演唱会票务信息管理平台注册登录流程如下



在到达系统登录页面时，首先判断是否拥有用户名和密码，如果拥有可以直接登录进入系统；如果没有用户名和密码，点击注册，填写需要注册的用户名和密码，并填写邀请码，注册成功后登录进入系统。

设计注册时需要填写邀请码的原因是：该平台是面向后台信息管理的系统，

出于实际考虑，不可能面向所有用户自由开放注册，而邀请码由最权威的管理员提供，提供后方可注册，兼顾了安全性。

3.1.2 票品信息录入

管理员登录系统，进入票务信息管理模块，选择录入新的票品信息，填写票品的详细信息，包括票品名称、票价、数量、演唱会名称等，提交信息后，系统保存票品信息。

此功能模块用于管理员向系统录入新的票品信息，确保票品信息的及时更新。

3.1.3 票品信息删除

管理员登录系统，进入票务信息管理模块，找到需要删除的票品信息，通过页面左侧的票品信息列表，找到需要删除票品的序号，在右侧表单中输入该序号，点击删除按钮即可删除该票品信息。

此功能模块用于管理员删除系统中不再有效或需要下架的票品信息。

3.1.4 票品信息修改

管理员登录系统，进入票务信息管理模块，找到需要修改的票品信息，通过页面左侧的票品信息列表，找到需要修改票品信息的序号，在右侧表单中输入该序号，点击查询，下方信息表单会自动填充该票品的信息，修改自己需要修改的部分，点击修改按钮提交信息，系统保存新的票品信息。

此功能模块用于管理员对已有票品信息进行修改，确保信息的准确性和实时性。

3.1.5 票品出售

管理员登录系统，进入票品售卖模块，所有票品信息会显示在页面中，找到顾客需要的票品，点击出售按钮，进入顾客信息登记页面，左侧显示该场演出的信息，右侧需要登记该顾客的姓名、身份证号以及购买数量，总价格在输入数量后会自动填充，点击出售按钮提交信息，票品出售成功，系统保存该次出售记录。

此功能模块用于主办方出售票品，确保出售操作的便捷性和安全性。

3.1.6 售票记录查询

在票品出售界面，点击售票记录查询按钮，即可显示已经出售的所有票品信息。

3.2 数据库设计

总数据库名称为“ticket”，数据库内包含三张表：“home”、“customer”和“user”。

3.2.1 表 home

表 home 是用来存储演唱会票务信息，包含字段 id、show_name、performers、show_time、total_tickets、sold_tickets、remaining_tickets、price；分别代表：序号、演出名称、主要演职人员、演出时间、总票数、已售、余票、价格。

其中，字段 id 设置为主键，且自动递增；show_name、performers、show_time 为 varchar 数据类型，不为空；total_tickets、sold_tickets、remaining_tickets、price 为 int 数据类型，不为空。

```
mysql> show columns from ticket.home;
+-------------------+--------------+------+-----+---------+----------------+
| Field             | Type         | Null | Key | Default | Extra          |
+-------------------+--------------+------+-----+---------+----------------+
| id                | bigint       | NO   | PRI | NULL    | auto_increment |
| show_name         | varchar(255) | YES  |     | NULL    |                |
| performers        | varchar(255) | YES  |     | NULL    |                |
| show_time         | varchar(255) | YES  |     | NULL    |                |
| total_tickets     | int unsigned | YES  |     | NULL    |                |
| sold_tickets      | int unsigned | YES  |     | NULL    |                |
| remaining_tickets | int unsigned | YES  |     | NULL    |                |
| price             | int unsigned | YES  |     | NULL    |                |
+-------------------+--------------+------+-----+---------+----------------+
8 rows in set (0.01 sec)
```

3.2.2 表 customer

表 customer 是用来存储购买票品的顾客信息，包含字段 id、name、id_card、show_name、quantity、amount；分别代表：序号、姓名、身份证号、演出名称、数量、总价。

其中，字段 id 设置为主键，且自动递增；name、id_card、show_name 为 varchar 数据类型，不为空；quantity、amount 为 int 数据类型，不为空；home_id、ticket_id 为表 customer 的外键。

```
mysql> show columns from ticket.customer;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| id        | bigint       | NO   | PRI | NULL    | auto_increment |
| name      | varchar(255) | YES  |     | NULL    |                |
| id_card   | varchar(255) | YES  |     | NULL    |                |
| show_name | varchar(255) | YES  |     | NULL    |                |
| quantity  | int          | YES  |     | NULL    |                |
| amount    | int          | YES  |     | NULL    |                |
| home_id   | bigint       | YES  | MUL | NULL    |                |
| ticket_id | bigint       | YES  | MUL | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
8 rows in set (0.00 sec)
```

3.2.3 表 user

表 user 是用来存储整个系统的登录用户信息，包含字段 id、username、password、invitat_code；分别代表：序号、用户名、密码、邀请码

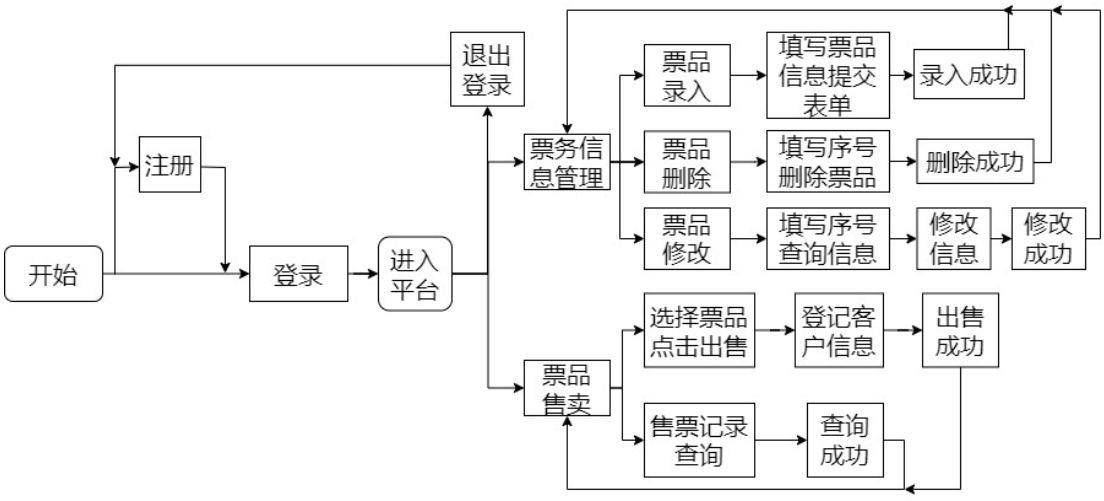
其中，字段 id 设置为主键，且自动递增；username、password 为 varchar 数据类型，不为空。

```
mysql> show columns from ticket.user;
+-----------------+--------------+------+-----+---------+----------------+
| Field           | Type         | Null | Key | Default | Extra          |
+-----------------+--------------+------+-----+---------+----------------+
| id              | bigint       | NO   | PRI | NULL    | auto_increment |
| username        | varchar(255) | YES  |     | NULL    |                |
| password        | varchar(255) | YES  |     | NULL    |                |
| invitation_code | varchar(255) | YES  |     | NULL    |                |
+-----------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

4 系统实现

4.1 系统的操作逻辑

演唱会票务信息管理平台的操作逻辑用流程图表示如下：



4.2 登录功能的实现

4.2.1 登录账户数据验证

登录功能通过在 TicketController 控制类中使用“login”方法实现，首先通过调用“userResponsitory”中的“findByUsernameAndPassword”方法验证数据库中的用户名和密码，登录成功，将用户信息存储在 Session 中；登录失败，返回登录页面。

代码如下：

```
@Autowired
private UserRepository userRepository;
@PostMapping("/login")
public String login(String username, String password, HttpSession session, Model model) {
    // 查询数据库中的用户信息
    user user = userRepository.findByUsernameAndPassword(username, password);

    if (user != null) {
        // 登录成功，将用户信息存储在Session中
        session.setAttribute( s: "user", user);
        return "redirect:/index"; // 重定向到index页面
    } else {
        // 登录失败，返回登录页面并显示错误消息
        model.addAttribute( s: "ERROR",  o: "用户名或密码无效");
        return "login";
    }
}
```

```
4 usages
public interface UserRepository extends JpaRepository<user, Long> {
    1 usage
    user findByUsernameAndPassword(String username, String password);
}
```

4.2.2 登录拦截器

在用户使用表单“login”登录时，如果仅仅是 4.2.1 的代码，当在浏览器地址栏输入例如 http://localhost:8080/index/ticket-page 的地址，就可以跳过登录界面达到直接访问 ticket-page 的效果，平台的安全性得不到保障，因此，使用“LoginInterceptor”拦截器来拦截未登录状态下除注册功能外的所有请求。

如果在登录状态下，session 中会存储登录成功的信息，因此，拦截器的主要思想就是通过验证 session 中的信息来达到拦截的效果。

代码如下：

```
@Configuration
public class LoginInterceptor implements HandlerInterceptor {

    no usages
    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
            throws Exception {
        // 在请求处理之前进行拦截
        if (request.getSession().getAttribute( s: "user") == null) {
            // 用户未登录，重定向到登录页面
            response.sendRedirect( s: request.getContextPath() + "/login");
            return false;
        }
        // 用户已登录，继续处理请求
        return true;
    }

}
```

```
@Configuration
public class WebConfig implements WebMvcConfigurer {

    no usages
    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        // 注册拦截器，并设置拦截的路径
        registry.addInterceptor(new LoginInterceptor()).addPathPatterns("/index/**");
    }
}
```

4.3 票品录入功能的实现

在 TicketController 控制类中使用“showAddTicketForm”方法显示添加表单；使用“addConcert”方法调用 ConcertService 中的“saveConcert”方法将表单中的数据添加到数据库。

代码如下：

```
@GetMapping("/index/show-list/add-ticket")
public String showAddTicketForm(Model model) {
    //显示添加票务的表单页面
    model.addAttribute( s: "home", new home());
    return "add-ticket";
}
@PostMapping("/add")
public String addConcert(@ModelAttribute home home) {
    //处理实际的添加操作
    concertService.saveConcert(home);
    return "redirect:/index/success";
}
```

```
1 usage
public void saveConcert(home home) {
    //保存表home所有数据
    concertRepository.save(home);
}
```

4.4 票品修改功能的实现

首先，使用“showReviseTicketForm”方法显示修改表单，reviseTicket 模型显示用序号查到的源数据，updatedTicket 模型绑定修改后的数据。

代码如下：

```
@GetMapping("/index/show-list/revise-ticket")
public String showReviseTicketForm(Model model) {
    //跳转到revise-ticket

    // 显示表 home 的所有内容
    model.addAttribute( s: "homes", concertService.getAllConcerts());
    //显示用序号查到的源数据
    model.addAttribute( s: "reviseTicket", new home());
    //绑定修改后的数据
    model.addAttribute( s: "updatedTicket", new home());
    return "revise-ticket";
}
```

其次，使用“findTicketById”方法将查询到需要修改的票品信息填充到表单中，为用户的修改数据提供便利性。

代码如下：

```
@PostMapping("/revise")
public String findTicketById(@ModelAttribute home reviseTicket, Model model) {
    // 用序号查找票品信息
    home existingTicket = concertService.findTicketById(reviseTicket.getId());
    // 显示表 home 的所有内容
    model.addAttribute( s: "homes", concertService.getAllConcerts());
    // 将原数据显示在修改表单中
    model.addAttribute( s: "reviseTicket", existingTicket);
    model.addAttribute( s: "updatedTicket", existingTicket);

    return "revise-ticket";
}
```

在用户修改数据后，使用“updateTicket”方法调用 ConcertService 中的“updateTicket”方法将修改后的数据更新到数据库。

代码如下：

```
@PostMapping("/update")
public String updateTicket(@ModelAttribute home updatedTicket) {
    // 更新数据库中的数据
    concertService.updateTicket(updatedTicket);
    return "redirect:/index/success";
}
```

4.5 票品出售功能的实现

在 TicketController 控制类中使用“showSellTicket”方法显示添加表单，将

数据库中演唱会的所有信息显示在表单，并在每一条数据后添加出售按钮，用于出售该场演唱会的票品。

代码如下：

```
@GetMapping("/index/sell-ticket")
public String showSellTicket(Model model) {
    //跳转到sell-ticket

    // 显示表 home 的所有内容
    List<home> homes = concertService.getAllConcerts();
    model.addAttribute( s: "homes", homes);
    return "sell-ticket";
}
```

```
<a class="a2" th:href="@{'/index/sell-ticket/customer-information/' + ${home.id}}">出售</a>
```

在点击出售按钮后，会跳转到登记顾客信息的页面，页面中会显示该场演出的名称、总票数、已售票数、未售票数和价格，并且提供登记顾客的姓名和身份证号。

代码如下：

```
@GetMapping("/index/sell-ticket/customer-information/{homeId}")
public String sellTicketForm(Model model, @PathVariable Long homeId) {
    //显示点击出售后的表单
    home home = concertService.findTicketById(homeId);
    List<String> showNames;
    showNames = concertService.getAllShowNames();

    model.addAttribute( s: "home", home);
    model.addAttribute( s: "customer", new customer());
    model.addAttribute( s: "showNames", showNames);

    return "customer-information";
}
```

在服务类“ConcertService”中，还需定义“sellTicket”方法，处理出售逻辑，例如出售一定数量的票，数据库中的票品信息需要相应的更新，以达到实时数据更新的目的。

代码如下：

```java
public void sellTicket(Long homeId,customer customer) {
    // 获取相应的 home 对象
    home home = concertRepository.findById(homeId).orElse( other: null);

    if (home != null) {
        // 点击确认出售后，已售+数量，未售-数量
        int quantity = customer.getQuantity();
        home.setSoldTickets(home.getSoldTickets() + quantity);
        home.setRemainingTickets(home.getRemainingTickets() - quantity);

        // 保存更新后的 home 对象
        concertRepository.save(home);

        // 保存表 customer 的数据
        customer.setHome(home);
        customerRepository.save(customer);

    }
```

5 系统测试

5.1 测试用例

1. 注册一个测试账户，用户名：test，密码：12345678，并用此账户登录演唱会票务信息管理平台。
2. 录入一场演唱会的数据：

   演出名称：演唱会_test；

   主要演职人员：演员_test；

   演出时间：2023 年 12 月 31 日；

   总票数：26000；

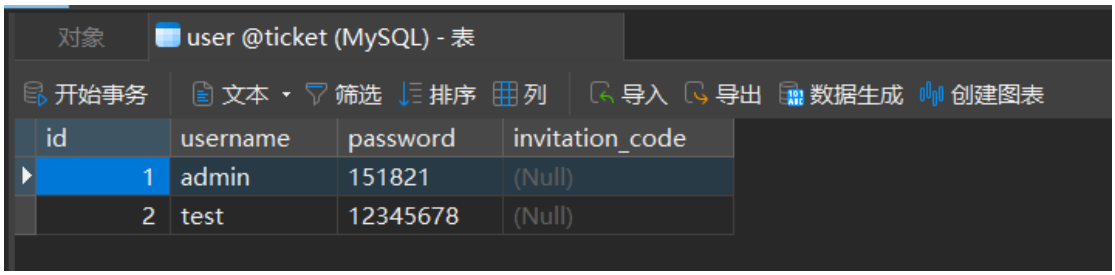
   已售：0；

   余票：26000；

   价格：1000
3. 修改此次演唱会的演出时间为 2024 年 1 月 1 日，票价为 1300 元。
4. 向顾客：姓名：小红；身份证号：610610122000010112 34，出售此次演唱会的票 3 张。
5. 查询此次出售记录。
6. 删除演唱会_test 的信息。

5.2 测试结果

1. 注册并登录平台



2. 录入演唱会数据

6 号数据即为测试录入

票品列表

序号	演出名称	主要演职人员	演出时间	总票数	已售	余票	价格
1	薛之谦天外来物巡回演唱会-澳门站	薛之谦	2023年12月8日-10日	30000	1200	28800	1717
2	周杰伦2024巡回演唱会-福州站	周杰伦	2024年5月16日	30000	5	29995	2000
3	林俊杰世界巡回演唱会-南宁站	林俊杰	2023年12月9日-10日	25000	15000	10000	1820
4	邓紫棋I AM GLORIA 演唱会-泉州站	邓紫棋	2024年1月20日	35000	15006	1994	1500
5	演唱会5	演员5	2023年10月30日	10	7	3	425
6	演唱会_test	演员_test	2023年12月31日	26000	0	26000	1000

3. 修改演唱会数据

票品信息修改

按序号查询: 6 查找

序号: 6

演出名称: 演唱会_test

主要演职人员: 演员_test

演出时间: 2023年12月31日

总票数: 26000

已售: 0

余票: 26000

价格: 1000

修改

票品信息修改

按序号查询: 6 查找

序号: 6

演出名称: 演唱会_test

主要演职人员: 演员_test

演出时间: 2024年1月1日

总票数: 26000

已售: 0

余票: 26000

价格: 1300

修改

票品列表

序号	演出名称	主要演职人员	演出时间	总票数	已售	余票	价格
1	薛之谦天外来物巡回演唱会-澳门站	薛之谦	2023年12月8日-10日	30000	1200	28800	1717
2	周杰伦2024巡回演唱会-福州站	周杰伦	2024年5月16日	30000	5	29995	2000
3	林俊杰世界巡回演唱会-南宁站	林俊杰	2023年12月9日-10日	25000	15000	10000	1820
4	邓紫棋I AM GLORIA 演唱会-泉州站	邓紫棋	2024年1月20日	35000	15006	1994	1500
5	演唱会5	演员5	2023年10月30日	10	7	3	425
6	演唱会_test	演员_test	2024年1月1日	26000	0	26000	1300

4. 向顾客售票

序号	演出名称	主要演职人员	演出时间	总票数	已售	余票	价格	操作
1	薛之谦天外来物巡回演唱会-澳门站	薛之谦	2023年12月8日-10日	30000	1200	28800	1717	出售
2	周杰伦2024巡回演唱会-福州站	周杰伦	2024年5月16日	30000	5	29995	2000	出售
3	林俊杰世界巡回演唱会-南宁站	林俊杰	2023年12月9日-10日	25000	15000	10000	1820	出售
4	邓紫棋I AM GLORIA 演唱会-泉州站	邓紫棋	2024年1月20日	35000	15006	1994	1500	出售
5	演唱会5	演员5	2023年10月30日	10	7	3	425	出售
6	演唱会_test	演员_test	2024年1月1日	26000	3	25997	1300	出售

5. 查询出售记录

序号	姓名	身份证号	演出名称	数量	金额
1	张佳瑶	6106166166	周杰伦2024巡回演唱会-福州站	5	10000
2	韩飞	610623200303151812	邓紫棋I AM GLORIA 演唱会-泉州站	3	4500
3	韩飞	610623200303151812	演唱会5	2	850
4	张三	60623200001021234	邓紫棋I AM GLORIA 演唱会-泉州站	3	4500
5	小红	6106101220000101234	演唱会_test	3	3900

6. 删除演唱会信息

6 总结与展望

6.1 项目开发概括

本次项目开发的是“唱会票务信息管理平台”，从 2023 年 11 月 28 日开始搭建项目环境到 2023 年 12 月 18 日完成项目报告，共计耗时 20 天，我个人利用课余时间以及周末完成了整个项目的开发，期间也遇到了很多的问题，在我的努力下，大部分问题也得到了解决，我对自己项目的满意度达到了 80%。

“演唱会票务信息管理平台”是基于 Spring boot 框架和 Maven 库开发的，开发环境我用的是 IDEA 2023，计算机操作系统为 Windows 10 专业版。

在开发项目的整个过程中，我主要借助的解决问题的工具是 Chat GPT-3.5，包括前期项目框架的搭建，中期一些操作逻辑的详解和项目 bug 的解决，后期前端页面的美化。

6.2 开发遇到的问题

在我的开发过程遇到了大量的问题，大部分问题已经得到解决，但还有一少

部分在互联网的帮助下依然无法得到解决。

1. 在 pom.xml 文件中引入依赖时，需要注意 maven 库中各个依赖的版本，在引入一些存在问题的版本时，部分代码会报错，我们尽可能引入那些下载量高的版本，存在问题较少
2. 在设计数据库表的时候，字段名称、字段类型等基础设置一定要综合自己的整个系统考虑设计，否则后期如果出现字段需要更改的问题，工作量会很大，例如主类中的字段类型都需要更改。
3. 在初步完成登录功能时，发现可以通过地址栏输入其他页面路径跳过登陆界面，最后通过拦截器拦截了在未登录状态下除登陆注册页面以外的所有页面。
4. 由于我的出售功能需要引用演唱会信息的表数据，所以在外键设置上下了很大的功夫，例如：

在最后两列“删除时”、“更新时”的类型应该更改为 CASCADE 而不是默认的 RESTRICT，因为默认的设置会在我引用表 home 的数据后，无法通过票品信息删除功能删除演唱会信息，修改后，便可在删除演唱会信息时同步删除该演唱会的售票记录而不报错。

5. 在前端页面中，我使用了 Thymeleaf 视图引擎在 HTML 页面中嵌入了动态数据，因此在 html 标签中应加入“xmlns:th=http://www.thymeleaf.org”，表示在 HTML 文档中使用 Thymeleaf 的语法时引入此命名空间。
6. 演唱会票品信息数据存储在表 home 中，它的 id 字段为主键，字段类型为 bigint 类型，且自动递增，因此有一个问题，在使用票品信息删除功能删除了某条数据后，其对应的序号便不在有效，下次插入数据就会出现序号不连续的问题，这个问题我查了大量的资料，但都是说这是数据库的数据碎片化，因此我到现在都没有解决。

6.3 项目的改进空间

1. 在设计登录功能时，可以使用 Spring Security，使整个系统的安全性能够得到保证，我自己也尝试使用这个登录验证框架，但很多代码我都无法理解，调试多次也没有解决，因此便弃用了，如果有机会，可以使用此框架进行登录验证。
2. 注册时用户名和密码的安全性与很大的隐患，可以使用一些加密算法对用户名和密码进行加密，而不是我简单的 varchar 类型。
3. 由于我个人对前端页面的代码不是很熟悉，所以整个前端页面的美观性不是很好，也是值得改进的一个部分。
4. 在系统功能中，由于时间问题，我只实现了依靠序号查询信息，我觉得可以添加一些查询功能，例如用每个字段查询，可以提供单独的界面，提高系统数据与用户的交互性。