

数据库课程设计报告

票务（飞机票）管理系统

年级：2022

班级：菁英班

姓名：叶俊

学号：202200201151

目录

一、系统概述	3
二、需求分析	5
2.1 系统功能分析	5
2.2 系统数据分析	10
2.3 系统非功能分析（可选）	12
三、系统设计	14
3.1 应用程序设计	14
3.2 数据库设计	18
3.2.1 概念设计	18
3.2.2 逻辑设计	21
3.2.3 物理设计	21
四、系统实现	22
4.1 关键技术实现（可选）	22
4.2 功能实现	22
五、系统测试（可选）	52
六、总结	53
6.1 课程设计总结	错误！未定义书签。
6.2 华为 GaussDB for MySQL 数据库使用总结	错误！未定义书签。

一、系统概述

1. 系统背景

随着互联网技术的发展，在线票务管理系统已经成为用户出行和活动参与不可或缺的一部分。携程作为国内知名的旅游与票务平台，通过其强大的信息处理和用户服务能力，为用户提供便捷的购票体验。为了学习和实践数据库管理及系统设计，我们决定仿照携程的票务管理系统，设计一个功能完善的票务管理平台，旨在为用户提供更好的票务查询和购买服务。

2. 系统目标

本票务管理系统的主要目标为：

- 提供用户友好的票务查询和购买界面。
- 实现高效的票务信息管理，包括查询、预订、支付等功能。
- 提供详细的订单管理功能，允许用户查询和管理自己的订单。
- 实现后台管理功能，便于管理员对票务数据进行维护与更新。

3. 系统主要功能

本票务管理系统主要功能模块包括：

用户管理模块

- 。 用户注册、登录及信息管理。
- 。 用户权限控制，区分用户与管理员。

票务售出模块

- 。 支持根据航空公司，出发地、目的地、日期等条件进行查询。
- 。 显示票务信息，包括票种、价格、剩余数量等。

订单管理模块

- 。 提供订单查询与管理功能，用户可以查看自己的历史订单，并且对订单进行退票和改签。

后台管理模块

- 。 管理员可对票务信息进行增、删、改、查操作。
- 。 监控系统运行状态和用户活动（购票等），进行数据分析。

4. 系统用户

本系统主要用户分为两类：

普通用户：他们可以注册登录、查询票务信息、下订单、进行支付、查看自己的订单记录等。

管理员：负责系统的日常管理工作，包括维护票务数据、管理票务信息、订单信息等。

通过本系统的设计与实现，期望为用户提供便捷的票务购置体验，帮助本人深入理解和实践数据库系统的设计与应用

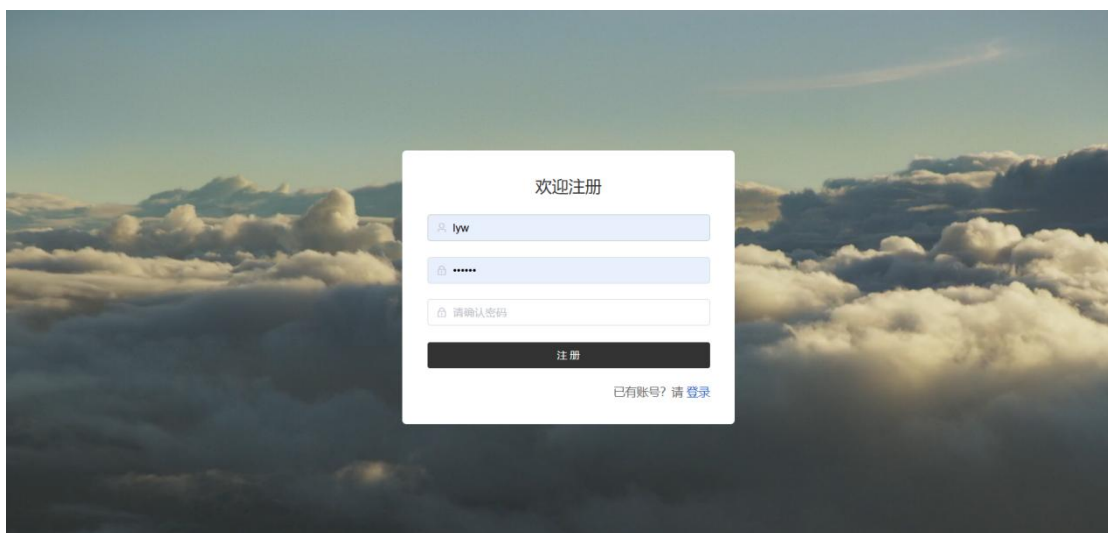
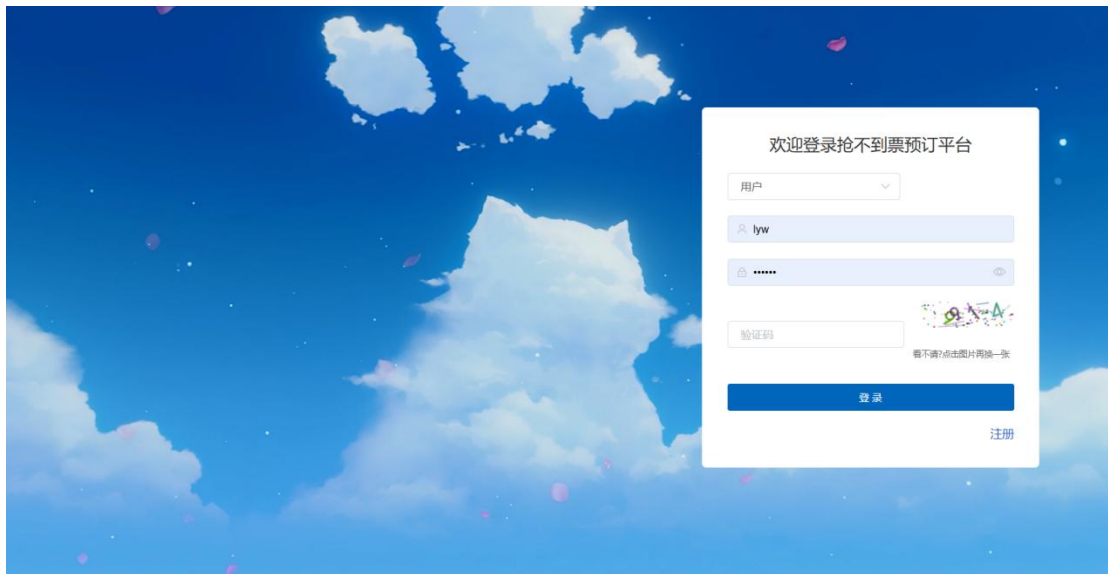
二、需求分析

2.1 系统功能分析

系统功能

1. 登录注册：登录：选择角色为管理员或是用户，输入账号密码以及随机生成的验证码登入，如果看不清验证码，点击图片重新生成。

注册：输入名称，设置密码并确认密码。确认两次输入的密码相同即可完成注册



2. 基于百度 API 实现的功能（附带做的，不是主要功能）：

天气：获取用户所在地的天气（还有温度）并进行展示。

景点推荐，查询与导航：在百度地图上查询附近景点，并实现导航功能



用户:

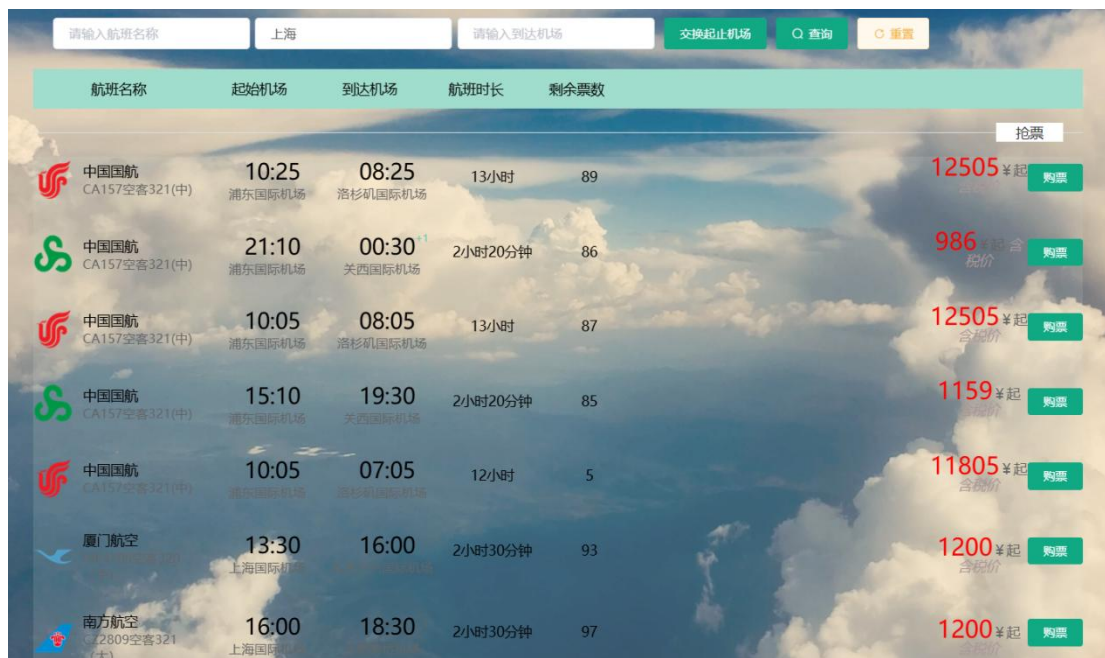
1. 飞机票等票务预定 (以飞机票为例)

1. 查询功能:

可查询航班的航空公司以及飞机型号, 起始时间及起始机场, 航班时长, 剩余票数以及航班售价。

对于直达航班, 通过航空公司名称或者起止机场名称或起止城市名称对航班进行查询 (这里以起飞城市为上海为例, 有

浦东国际和上海国际两个机场），可以交换输入的起止机场（这里本来想做个框显示与输入关键词匹配的信息，但是时间不够，全用来修 bug 了）



航班名称	起始机场	到达机场	航班时长	剩余票数	抢票
中国国航 CA157空客321(中)	10:25 浦东国际机场	08:25 洛杉矶国际机场	13小时	89	12505元起 购票
中国国航 CA157空客321(中)	21:10 浦东国际机场	00:30 ¹ 关西国际机场	2小时20分钟	86	986元起 含税价 购票
中国国航 CA157空客321(中)	10:05 浦东国际机场	08:05 洛杉矶国际机场	13小时	87	12505元起 含税价 购票
中国国航 CA157空客321(中)	15:10 浦东国际机场	19:30 关西国际机场	2小时20分钟	85	1159元起 含税价 购票
中国国航 CA157空客321(中)	10:05 浦东国际机场	07:05 洛杉矶国际机场	12小时	5	11805元起 含税价 购票
厦门航空 MF802空客320	13:30 上海国际机场	16:00 关西国际机场	2小时30分钟	93	1200元起 含税价 购票
南方航空 CZ2809空客321(大)	16:00 上海国际机场	18:30 关西国际机场	2小时30分钟	97	1200元起 含税价 购票

对于中转航班，与以上类似的同时，要注意的是把时间冲突的航班去掉，就是第一程的到达时间必须晚于第二程的出发时间，我这里实现的是同城换乘，在后续展示。

2. 购票：

只能购买有余票的航班，而且购买数大于余票数时不能进行购买，且不能购买当天及之前日期的票，购买中转航班时只要有一程不满足即无法购买

3. 历史订单查询和管理：

可查询购买订单的时间，订单编号，航班的具体信息以及订

单状态，可对订单进行退票，删除和改签，但是无法对已经起飞或已经退票的订单进行退票改签操作

4. 出行攻略：可发布攻略（包括图文表格等等），并对自己或者他人的攻略进行查询，评论，收藏，不可重复收藏，对自己的攻略可增删改查，实现了显示攻略的点击量和收藏量和评论量，可在个人中心查看自己发布的攻略和收藏的攻略

5. 个人信息的修改和查看：可在个人中心对除余额外的信息进行修改，包括密码，姓名，邮箱，电话等等，可以对余额进行充值

管理员：

1. 查看销售额度数据统计图：可在管理员界面的首页查看基于 echarts 图形库实现的对各种票务销售额度及其占比的饼状图等各种统计图

2. 对各种信息的增删改查：

对票务信息（例如航班信息和飞机订单等），攻略信息，用户信息的增删改查



3. 个人信息修改:

对管理员本人的信息（如电话，密码，名称等）的修改

2.2 系统数据分析

描述系统对于数据的需求、系统中需要保存哪些数据以及数据之间的联系。

（一）登录系统时，需要一个用户账户或管理员账户

①需要一个表保存用户信息

②需要一个表保存管理员信息

（二）在查询航班时，页面上需要显示指定航空公司或者指定起始点的航班，需要显示航班的起始地，航空公司，时间，价格等相关信息。点击某航班后的预订按钮，进入选票界面。

①需要一个表来保存航班相关信息。包括起止地、航空公司、时间、含税价，起始地点的时区（用来计算处理时差问题）等

（三）在预订机票时，形成一个订单，订单包括所预订的飞机票的飞机票 ID，所属的航空公司，订单时间，订单金额，支付状态（已支付或已退款），订单所属的用户的用户 ID 等。

①需要一个表来保存订单相关信息。

联系：一种飞机票可能属于多份订单（中转订单视作购买了多张直达票，相当于多个订单，所以不是多对多）。

一个用户可能拥有多份订单。

一个乘客可能使用多份订单。

（四）查看别人发布的攻略时，需要显示攻略的发布人，标题，内容（图片，文字，表格等等），发布时间，以及显示点击量，评论量，收藏量。需要显示评论内容。

①需要一个表来保存攻略的相关信息

②需要一个表来保存评论的相关信息。包括发表用户 id，攻略 id，以及评论内容和时间。

③需要一个表来保存每个用户的收藏情况。包括用户 id 和对应被收藏的攻略的 id

联系：一个用户可以发表多个评论。

一个用户可以发表多个攻略。

一个用户可以收藏多个攻略，一个攻略可以被多个用户收藏。

一个攻略可以有条评论。

（五）处理中转航班问题时，需要显示多个航班的信息，例如飞机型号，中转站，总价格，各个航班各自的余票数。

①需要一个表来存放中转的各个航班的信息。（这里用临时表来存放，没有放在数据库中）

2.3 系统非功能分析（可选）

描述系统的性能、安全性、可用性等。

系统性能：合理建立索引，加快了查询数据的速度。

安全性：考虑到了多种异常情况，例如，

1. 已经购买的航班如果被管理员取消，其余额会进行退还，
2. 进行退票改签操作要确保飞机尚未起飞，且未退票
3. 登录时密码验证码不能为空且输入正确，注册时两次输入秘密均需一致，
4. 购票时金额需要小于等于余额，余票为零或不够无法进行购买，购票只能购买当日第二天起的票，
5. 对于已经收藏的攻略，无法重复收藏

6. 长时间不进行操作 token 失效，需要重新登录

7. 查询时，输入航班名称和起止城市时只能输入汉字，无法输入字母，标点，数字，空格等字符

两次输入的密码不一致

验证码不正确!

请输入验证码

✕ 您已经收藏过该攻略，请勿重复收藏

✕ 您的航班已经起飞，无法退票

✕ 您的航班已经起飞，无法改签

✕ 账号或密码错误

✕ 用户不存在

前端采用 js 进行数据校验，非法的输入和请求不能成功传到后端，且设置弹框提醒用户输入正确合法的输入。

可用性：前端采用简洁且功能明确的界面，易于用户使用。

三、系统设计

3.1 应用程序设计

按照 2.1 小节，描述系统的架构、前后端采用的技术、前后端的关系；描述系统的模块、模块之间的关系。

系统架构

本系统采用前后端分离的架构模式，前端使用 Vue 3 框架，后端则基于 Spring Boot 3 框架，并且利用 Spring Data JPA 简化数据库操作。这种架构模式使得前后端开发能够独立进行，提高了开发效率和系统的可维护性。

前端技术栈

- 框架：Vue 3.x，提供响应式的数据绑定和组件化的开发模式，增强用户交互体验。
- HTML：用于定义网页的结构和内容。
- CSS：控制网页的布局和样式，提升用户体验。
- TypeScript：为 JavaScript 添加类型系统和面向对象特性，有助于在开发阶段捕获潜在错误，提高代码的可读性和可维护性。
- 路由管理：Vue Router，用于前端路由管理，实现单页面应用（SPA）的流畅导航。

- UI 组件库：根据项目需求选择 Element UI 组件库，以简化开发过程并确保一致的用户界面设计。
- 数据可视化：使用 ECharts 库进行数据可视化，增强系统数据展示效果。

后端技术栈

- 框架：Spring Boot 3.x，提供自动配置，简化部署和开发流程。
- 数据访问：Spring Data JPA，用于简化数据库访问层，支持标准化的 CRUD 操作。
- 安全性：Spring Security，提供安全认证和授权功能，确保用户数据的安全性。
- 构建工具：Apache Maven 3.8.4，用于项目构建和依赖管理。
- JDK：JDK 1.8，提供 Java 运行环境进行后端服务的开发和运行。

前后端关系

前端通过基于 axios 封装的自定义函数发送 HTTP 请求 (如 GET、POST、PUT、DELETE 等)

后端通过 SpringBoot 框架使用 RestController 以及 Request/Put/GetMapping 注解指定处理器类内的方法处理前端请求的路径。前后端之间使用 JSON 格式交换数据，前端将请求的数据序列化为 JSON 字符串，并通过 HTTP 请求发送给后端。后端处理请求后，将结果也序列化为 JSON 字符串，并包含在 HTTP 响应体中返回给前端。

用户登录时，后端会生成一个 JWT（JSON Web Token）并返回给前端，前端在后续请求中需携带此 Token 进行身份验证。

系统模块

系统主要模块可划分为用户模块和管理员模块，各模块的具体功能如下：

1. 用户模块：

- 登录注册：
 - 登录：用户可选择角色（管理员或普通用户），输入账号、密码及验证码进行登录。
 - 注册：用户输入姓名、设置密码并确认密码，您需要确保两次输入的密码相同。
- 票务预定：

- 查询航班：用户可查询航班的航空公司、飞机型号、起始时间、起始机场、航班时长、剩余票数及售价。
- 购票：只能购买有余票的航班，且必须在有效购票时间内。
- 历史订单管理：查询和管理订单，包括退票、删除和改签。
- 出行攻略：用户可发布和查询攻略，评论和收藏攻略。
- 个人信息管理：用户可修改个人信息（除余额外）并进行余额充值。

2. 管理员模块：

- 数据统计：管理员可查看基于 ECharts 实现的票务销售额度统计图。
- 信息管理：管理员对票务信息、攻略信息和用户信息的增删改查操作。
- 个人信息管理：管理员可修改自己的个人信息（如电话、密码和姓名）。

模块之间的关系

- 用户模块与安全模块：用户模块依赖安全模块提供

的身份认证与授权。

- 票务模块：与订单管理模块紧密关联，支持用户进行航班查询和订单管理。

- 数据可视化模块：从票务销售数据中提取数据生成可视化报表，供管理员查看。

- 攻略模块：支持用户发布、查询、评论和收藏攻略，促进用户交流。

3.2 数据库设计

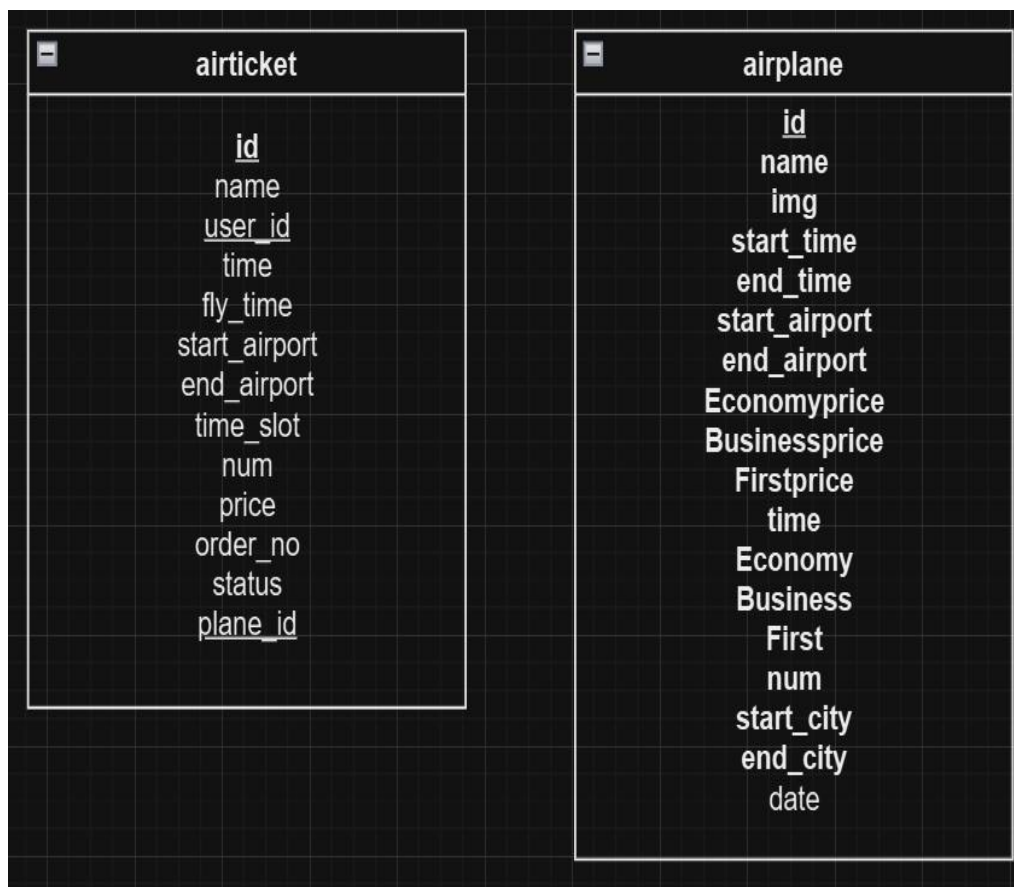
按照 2.2 小节，进行数据库设计。

3.2.1 概念设计

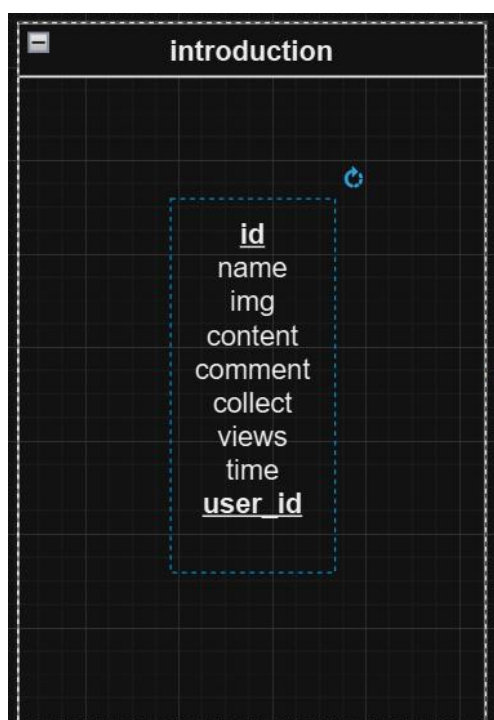
用户：



飞机票订单和飞机航班详细信息：



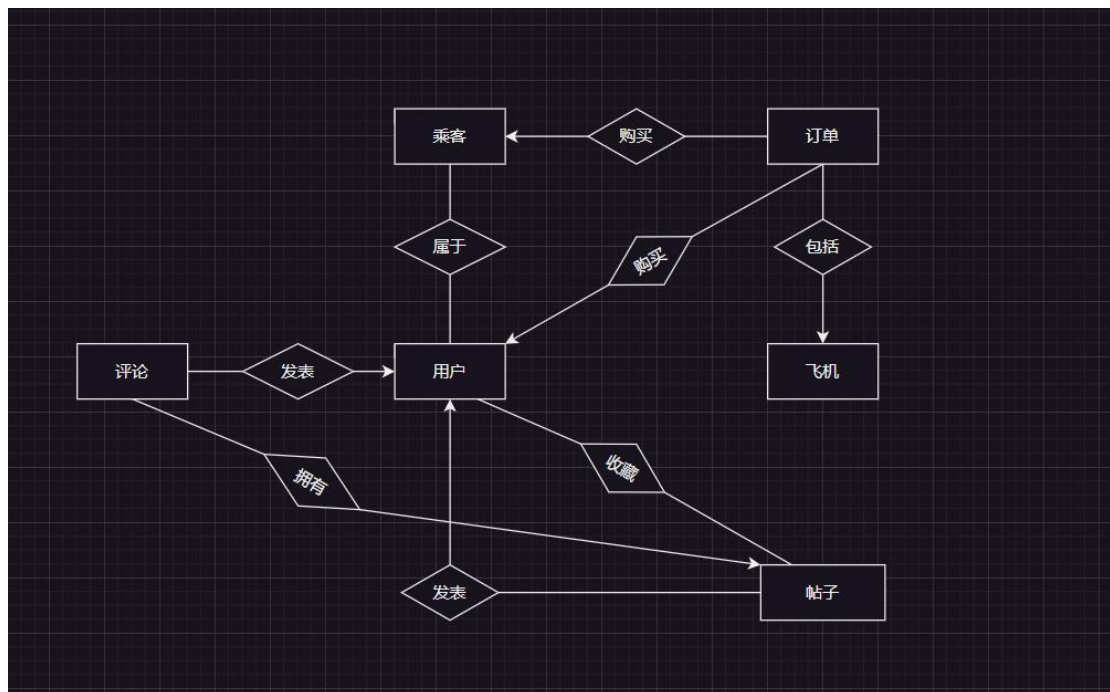
攻略：



评论：



总体 ER 图（以飞机票为例，其他类似）：



建立 ER 模型。

3.2.2 逻辑设计

将 ER 模型转换为关系模式。

用户（用户 ID，性别，用户名，用户密码，头像地址，姓名，role（注：角色类型，登陆时区分用户管理员），电话，邮箱，账户余额）

用户-乘客（用户 ID，乘客 ID）

乘客（乘客 ID，姓名）

飞机订单（订单 ID，航空公司名称，用户 ID，下单时间，起飞时间，起飞机场，目的机场，购票数量，价格，订单编号，订单状态，航班 ID，乘客 ID）

飞机航班（航班 ID，航空公司名称，图标地址，起飞时间，到达时间，起飞机场，目的机场，价格，耗时，余票数，起飞城市，到达城市，起飞时区，到达时区）

评论（评论 ID，用户 ID，攻略 ID，评论内容，评论时间）

收藏（攻略 ID，用户 ID）

攻略（攻略 ID，攻略标题，展示图片，具体图文内容，评论数，点击数，收藏数，发布时间，用户 ID）

3.2.3 物理设计

建立索引。

四、系统实现

4.1 关键技术实现（可选）

框架：前后端分离，前端使用 vue 框架，数据库使用 mybatis 作为持久层框架，

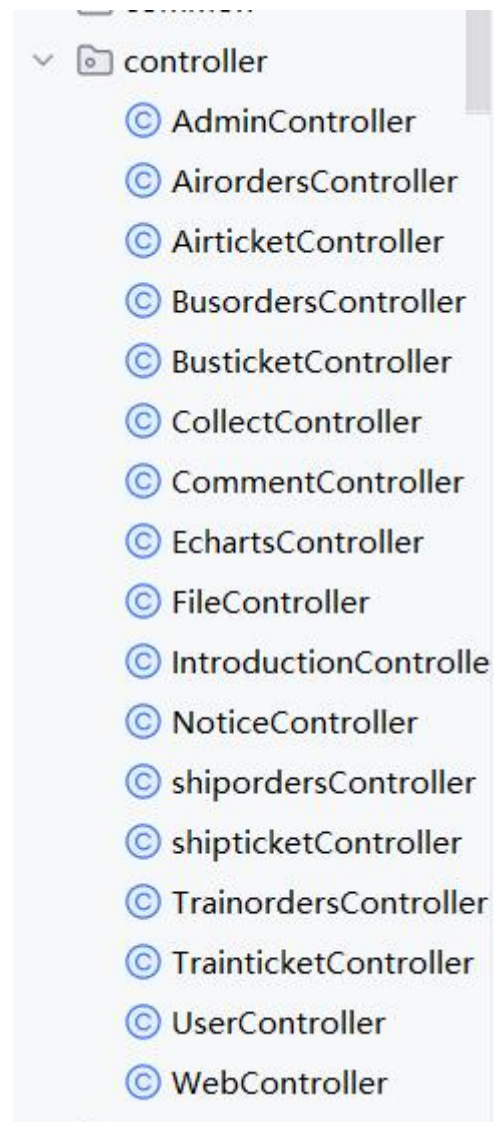
后端 java 使用 spring 框架，并加入 springMvc 架构，对请求和响应进行

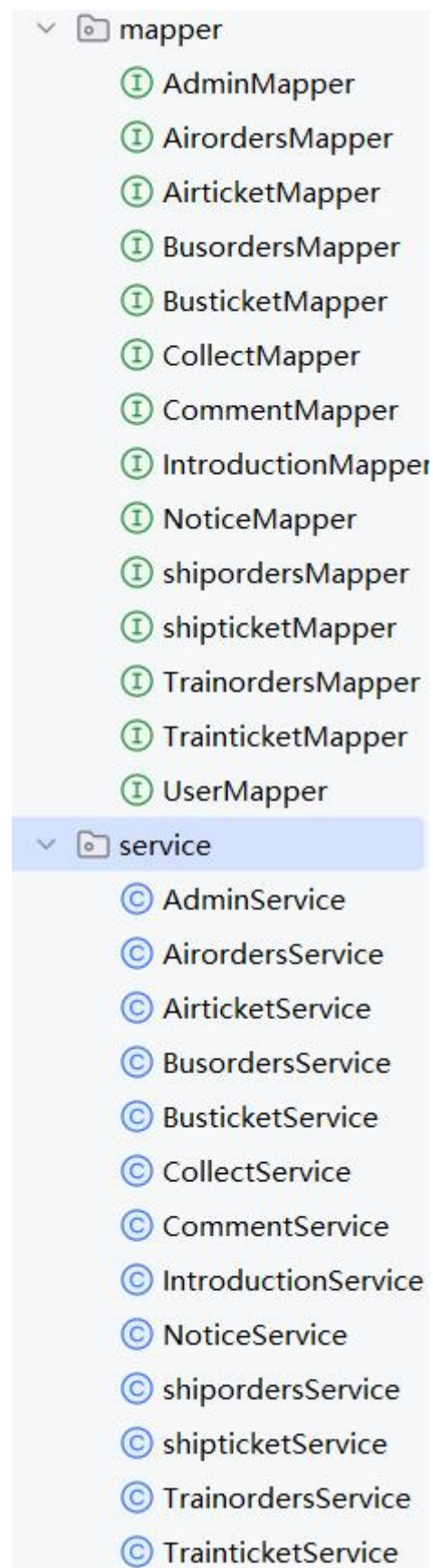
统一处理。同时使用 maven 进行项目管理。

MVC 架构分层的主要作用是解耦。采用分层架构的好处，普遍接受的是系

统分层有利于系统的维护，系统的扩展。就是增强系统的可维护性和可扩

展性。





- (1) .Controller 层接收前台数据和返回页面请求信息。
- (2) .service 层接受 controller 层信息，用于业务处理和逻辑判断。Service 用于处理业务逻辑，会调用 mapper 层的 API；
- (3) .mapper 层用于和数据库交互，想要访问数据库并且操作，只能通过 mapper 层向数据库发送 sql 语句，将这些结果通过接口传给 service 层，对数据库进行数据持久化操作

插件：由于使用了 maven 进行项目管理，一些插件的使用可以通过在 pom.xml 文件中配置来实现，比如：Spring Boot Maven Plugin 插件,这个插件可以用来构建可执行的 JAR 包，增强 Spring Boot 项目的构建能力。

```
<!-->
<plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
    <configuration>
        <fork>true</fork>
    </configuration>
</plugin>
```

Maven Compiler Plugin 插件，指定 Java 版本以编译项目的源代码。

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <configuration>
    <source>8</source>
    <target>8</target>
    <encoding>UTF-8</encoding>
  </configuration>
</plugin>
```

导入一些依赖，就可以使用相应的工具。

比如：PageHelper，用于在 MyBatis SQL 中进行分页操作，提供简单易用的分页功能。

```
<dependency>
  <groupId>com.github.pagehelper</groupId>
  <artifactId>pagehelper-spring-boot-starter</artifactId>
  <version>1.4.6</version>
  <exclusions>
    <exclusion>
      <groupId>org.mybatis</groupId>
      <artifactId>mybatis</artifactId>
    </exclusion>
  </exclusions>
</dependency>
```

Hutool, Hutool 是一个 Java 工具库, 提供了许多实用工具类, 包括字符串处理、日期处理、文件操作等, 极大地简化了常见的开发任务。

```
<dependency>
  <groupId>cn.hutool</groupId>
  <artifactId>hutool-all</artifactId>
  <version>5.8.18</version>
</dependency>
```

Java JWT, 用于创建和验证 JSON Web Tokens (JWT), 方便在分布式系统中进行用户身份验证和信息传递。

```
<dependency>
  <groupId>com.auth0</groupId>
  <artifactId>java-jwt</artifactId>
  <version>4.3.0</version>
</dependency>
```

JWT 拦截器的实现代码如下:

```

package com.example.common.config;

import ...

/**
 * jwt拦截器
 */
@Component
public class JwtInterceptor implements HandlerInterceptor {

    private static final Logger log = LoggerFactory.getLogger(JwtInterceptor.class); no usages

    @Resource 1 usage
    private AdminService adminService;
    @Resource 1 usage
    private UserService userService;

    @Override no usages
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) {
        // 1. 从http请求的header中获取token
        String token = request.getHeader(Constants.TOKEN);
        if (ObjectUtil.isEmpty(token)) {
            // 如果没拿到, 从参数里再拿一次
            token = request.getParameter(Constants.TOKEN);
        }
        // 2. 开始执行认证
        if (ObjectUtil.isEmpty(token)) {
            throw new CustomException(ResultCodeEnum.TOKEN_INVALID_ERROR);
        }
        Account account = null;
        try {
            // 解析token获取存储的数据
            String userRole = JWT.decode(token).getAudience().get(0);
            String userId = userRole.split( regex: "-" )[0];
            String role = userRole.split( regex: "-" )[1];
            // 根据userId查询数据库
            if (RoleEnum.ADMIN.name().equals(role)) {
                account = adminService.selectById(Integer.valueOf(userId));
            }
            if (RoleEnum.USER.name().equals(role)) {
                account = userService.selectById(Integer.valueOf(userId));
            }
        } catch (Exception e) {
            throw new CustomException(ResultCodeEnum.TOKEN_CHECK_ERROR);
        }
        if (ObjectUtil.isNull(account)) {
            throw new CustomException(ResultCodeEnum.USER_NOT_EXIST_ERROR);
        }
        try {
            // 用户密码加签验证 token
            JWTVerifier jwtVerifier = JWT.require(Algorithm.HMAC256(account.getPassword())).build();
            jwtVerifier.verify(token); // 验证token
        } catch (JWTVerificationException e) {
            throw new CustomException(ResultCodeEnum.TOKEN_CHECK_ERROR);
        }
        return true;
    }
}

```

4.2 功能实现

按照系统设计，描述系统的主要功能是如何实现的，包括关键源代码分析和主要模块运行截图。

1. 中转功能：先基于原来的机票信息定义一个能存储多程机票信息的实体--transferairticket，

在 xml 文件中，新增一个方法，基于原有的查询方法稍微做出变化：将两个一样的原来存放机票信息的表链接起来，连接的条件是**第一的表的终点城市和第二个表的起点城市相等**，查询的返回类型改为 transferairticket。其他层也相应新增方法。（本来想的是新建一个表来存放中转信息，但是这样做类似于临时表，方便一点），

这样每需要显示中转信息时都会进行一次这样的查询，并将信息传到前端展示出来，就算直达的机票信息有变，查询到的中转机票信息也会相应改变。

还有一点是购买中转票相当于会买多张直达票，可以对每一程机票分别进行退票改签等操作。

要注意的是中转的航班不能有时间冲突，**第一程到达时间需要比第二程起飞时间早一个小时以上。**

源代码：1. 前端发送请求，并将航空公司，起止城市等信息传递给后端

```
this.$request.get('/airticket/selecttransfer',{  
  params: {
```

```

        name: this.name,

        startAirport: this.startAirport,

        transferAirport: this.transferAirport,

        transferCity: this.transferCity,

        endAirport: this.endAirport,

        startCity: this.startCity,

        endCity: this.endCity,

    }

    }).then(res => {

        if (res.code === '200') {

            this.transAirticketData = res.data

        } else {

            this.$message.error(res.msg)

        }

    })

}

```

controller 处理，调用 service，service 调用 mapper 方法

Controller:

```

@GetMapping("/selecttransfer")

public Result selecttransfer(Airticket airticket) {

    List<TransAirticket> list = airticketService.selecttransfer(airticket);

```

```
        return Result.success(list);
    }
}
```

```
Service: public List<TransAirticket> selecttransfer(Airticket airticket) {

    List<TransAirticket> transAirtickets = airticketMapper.selecttransfer(airticket);

    // airticketMapper.addRentalItem(transAirtickets);

    for (TransAirticket dbTransAirticket : transAirtickets) {

        // 处理 name 属性

        String name = dbTransAirticket.getName();

        if (name != null && name.split(" ").length >= 2) {

            String[] nameParts = name.split(" ");

            dbTransAirticket.setLeft(nameParts[0]);

            dbTransAirticket.setRight(nameParts[1]);

        }

        // 处理 name2 属性

        String name2 = dbTransAirticket.getName2();

        if (name2 != null && name2.split(" ").length >= 2) {

            String[] name2Parts = name2.split(" ");

            dbTransAirticket.setLeft2(name2Parts[0]);

            dbTransAirticket.setRight2(name2Parts[1]);

        }

        //插入数据库
    }
}
```

```

        String time1 = dbTransAirticket.getTransferTime1();

        String time2 = dbTransAirticket.getTransferTime2();

        if (!TimeUtils.compareDate2(time1, time2, "HH:mm:ss")) {

            deleteById(dbTransAirticket.getId());

        }

    }

    return transAirtickets;

}

```

Service 中的 name 处理是为了界面美观将信息分开来处理展示，如下



Mapper:

```
List<TransAirticket> selecttransfer(Airticket airticket);
```

关键代码位于 xml 文件: 这里对于传来的信息进行比对查询, 并在 where 语句中对第一程终点时间和第二程起飞时间进行比对以处理时间冲突问题

```
<select id="selecttransfer" resultType="com.example.entity.TransAirticket">
```

```
SELECT
```

```
t1.id AS id1,
```


t2.id AS id2,

t1.name AS name,

t1.img AS img,

t1.start_time AS start_time,

t1.end_time AS transfer_time1,

t2.start_time AS transfer_time2,

t1.start_airport AS start_airport,

t1.price AS price,

t1.time,

t1.num AS num,

t1.start_city AS start_city,

t2.start_airport AS transfer_airport,

t2.start_city AS transfer_city,

t2.end_city AS end_city,

t2.end_time AS end_time,

t2.end_airport AS end_airport,

t2.name AS name2,

t2.img AS img2,

t2.price AS price2,

t2.time AS time2,

t2.num AS num2,

t1.uit1,

```

t2.uit2

FROM

airticket t1

JOIN

airticket t2 ON t1.end_city = t2.start_city

<where>

    <if test="id != null"> AND t1.id = #{id} or t2.id = #{id}</if>

    <if test="name != null"> AND t1.name LIKE CONCAT('%', #{name}, '%') or
t2.name LIKE CONCAT('%', #{name}, '%')</if>

    <if test="startAirport != null"> AND t1.start_airport LIKE CONCAT('%',
#{startAirport}, '%') </if>

    <if test="endAirport != null"> AND t2.end_airport LIKE CONCAT('%',
#{endAirport}, '%') </if>

    <if test="startCity != null"> and t1.start_city like concat('%', #{startCity}, '%')

</if>

    <if test="endCity != null"> and t2.end_city like concat('%', #{endCity}, '%') </if>

    AND STR_TO_DATE(t1.end_time, '%H:%i:%s') < STR_TO_DATE(t2.start_time,
'%H:%i:%s')

</where>

</select>

```

功能展示：



2. 退票功能:

前端先将订单的 ID 传给后端，在表中找到相应 ID 订单进行操作

1. 先将状态为“已购买”的机票的起飞时间与当前时间进行比对，如果飞机已经起飞，则无法进行操作，
2. 否则将对应订单的状态变为“已退票”，并将票钱退回，将余票数增加

源代码:

主要流程仍是前端传数据，后端接收，相应的 controller 调用 service 方法进行处理，故此处后端只分析关键代码。前端仍然是调用函数传回数据，发送请求，并判断返回结果是否正确：

```
returnTicket(row) {
  this.$request.post('/airorders/returnTicket', row).then(res => {
    if (res.code === '200') {
      this.$message.success('退票成功')
      this.load(1)
    } else {
```

```

        this.$message.error(res.msg)
    }
}
}

```

后端关键代码在 service 层，代码分析如下：

```

public void returnTicket(Airorders airorders) throws ParseException {

    // 1. 判断当前的时间是否已经过了起飞的时间

    String now = DateUtil.now();

    String flyTime = airorders.getFlyTime();

    if (!TimeUtils.compareDate(flyTime, now, "yyyy-MM-dd HH:mm:ss")) {

        throw new CustomException("-1", "您的航班已经起飞，无法退票");

    }

    // 2. 退票

    airorders.setStatus("已退票");

    airordersMapper.updateById(airorders);

    // 3. 退钱

    User user = userMapper.selectById(airorders.getUserId());

    user.setAccount(user.getAccount() + airorders.getNum() * airorders.getPrice());

    userMapper.updateById(user);

    // 4. 票数返还

    Airticket airticket = airticketMapper.selectById(airorders.getTicketId());

    airticket.setNum(airticket.getNum() + airorders.getNum());
}

```

```

    airticketMapper.updateById(airticket);
}

```

其他代码及结构与上述分析基本类似，故不重复展示，只展示关键代码，飞机订单（airorders）xml 文件如下，实现的是基础的增删改查功能，其他飞机航班 xml 文件等基本类似，故不再展示

```

<sql id="Base_Column_List">

```

```

    id,name,user_id,time,fly_time,start_airport,end_airport,time_slot,num,price,order_no,s
    tatus,ticket_id

```

```

</sql>

```

```

<select id="selectAll" resultType="com.example.entity.Airorders">

```

```

    select airorders.*, user.name as userName

```

```

    from airorders

```

```

    left join user on airorders.user_id = user.id

```

```

    <where>

```

```

        <if test="id != null"> and id = #{id}</if>

```

```

        <if test="orderNo != null"> and order_no = #{orderNo}</if>

```

```

        <if test="userId != null"> and user_id = #{userId}</if>

```

```

    </where>

```

```

    order by id desc

```

```

</select>

```

```
<select id="selectById" resultType="com.example.entity.Airorders">
```

```
    select
```

```
    <include refid="Base_Column_List" />
```

```
    from airorders
```

```
    where id = #{id}
```

```
</select>
```

```
<delete id="deleteById">
```

```
    delete from airorders
```

```
    where id = #{id}
```

```
</delete>
```

```
<insert id="insert" parameterType="com.example.entity.Airorders"
```

```
useGeneratedKeys="true">
```

```
    insert into airorders
```

```
    <trim prefix="(" suffix=")" suffixOverrides=",">
```

```
        <if test="id != null">id,</if>
```

```
        <if test="name != null">name,</if>
```

```
        <if test="userId != null">user_id,</if>
```

```
        <if test="time != null">time,</if>
```

```
        <if test="flyTime != null">fly_time,</if>
```

```

    <if test="startAirport != null">start_airport,</if>

    <if test="endAirport != null">end_airport,</if>

    <if test="timeSlot != null">time_slot,</if>

    <if test="num != null">num,</if>

    <if test="price != null">price,</if>

    <if test="orderNo != null">order_no,</if>

    <if test="status != null">status,</if>

    <if test="ticketId != null">ticket_id,</if>

  </trim>

  <trim prefix="values (" suffix=")" suffixOverrides=",">

    <if test="id != null">#{id},</if>

    <if test="name != null">#{name},</if>

    <if test="userId != null">#{userId},</if>

    <if test="time != null">#{time},</if>

    <if test="flyTime != null">#{flyTime},</if>

    <if test="startAirport != null">#{startAirport},</if>

    <if test="endAirport != null">#{endAirport},</if>

    <if test="timeSlot != null">#{timeSlot},</if>

    <if test="num != null">#{num},</if>

    <if test="price != null">#{price},</if>

    <if test="orderNo != null">#{orderNo},</if>

    <if test="status != null">#{status},</if>

```

```

        <if test="ticketId != null">#{ticketId},</if>

    </trim>

</insert>

<update id="updateById" parameterType="com.example.entity.Airorders">

    update airorders

    <set>

        <if test="name != null">

            name = #{name},

        </if>

        <if test="flyTime != null">

            fly_time = #{flyTime},

        </if>

        <if test="startAirport != null">

            start_airport = #{startAirport},

        </if>

        <if test="endAirport != null">

            end_airport = #{endAirport},

        </if>

        <if test="timeSlot != null">

            time_slot = #{timeSlot},

        </if>

```



```

<if test="price != null">

    price = #{price},

</if>

<if test="status != null">

    status = #{status},

</if>

<if test="ticketId != null">

    ticket_id = #{ticketId},

</if>

</set>

where id = #{id}

</update>

```

功能展示：初始账户余额



The image shows a user interface for a member's account. At the top, there is a cartoon character with the text "耶! 耶!" (Yay! Yay!). To the right, it says "尊敬的会员:" (Respected Member:), followed by the name "刘宇为" (Liu Yuwei). Below this, it says "欢迎回来!" (Welcome back!) and "今天抢到票了嘛?" (Did you get tickets today?). There are three buttons: "充值" (Recharge), "修改密码" (Change password), and "修改个人信息" (Change personal information).

Below the main content, there is a section titled "会员信息" (Member Information) with a table showing the following data:

用户名	手机号	邮箱
lyw	02502502502	lyw@xm.com
备注	余额	
学校	54097	

初始订单状态

航班名称	用户姓名	下单时间	起始机场	到达机场	起飞时间	航程时长	机票数量	机票价格	订单状态	操作
中国国航 CA157空客321(中)	刘宇为	2024-10-19 21:16:32	浦东国际机场	洛杉矶国际机场	2024-10-20 10:25:00	13小时	1	12505	已购买	退票 改签 删除
上海航空 FM9096空客320 (中)	刘宇为	2024-10-12 21:17:02	上海国际机场	合肥新桥机场	2024-10-13 13:00:00	2小时	1	1000	已退票	退票 改签 删除
四川航空 3U8198波音777 (中)	刘宇为	2024-10-12 21:17:02	重庆国际机场	上海国际机场	2024-10-13 08:00:00	2小时15分钟	1	1200	已退票	退票 改签 删除
中国国航 CA157空客321(中)	刘宇为	2024-10-12 18:16:53	浦东国际机场	关西国际机场	2024-10-13 21:10:00	2小时20分钟	1	986	已退票	退票 改签 删除
中国国航 CA157空客321(中)	刘宇为	2024-10-12 18:16:12	浦东国际机场	洛杉矶国际机场	2024-10-13 10:25:00	13小时	1	12505	已退票	退票 改签 删除

退票后订单及余额

退票成功										
航班名称	用户姓名	下单时间	起始机场	到达机场	起飞时间	航程时长	机票数量	机票价格	订单状态	操作
中国国航 CA157空客321(中)	刘宇为	2024-10-19 21:16:32	浦东国际机场	洛杉矶国际机场	2024-10-20 10:25:00	13小时	1	12505	已退票	退票 改签 删除
上海航空 FM9096空客320 (中)	刘宇为	2024-10-12 21:17:02	上海国际机场	合肥新桥机场	2024-10-13 13:00:00	2小时	1	1000	已退票	退票 改签 删除
四川航空 3U8198波音777 (中)	刘宇为	2024-10-12 21:17:02	重庆国际机场	上海国际机场	2024-10-13 08:00:00	2小时15分钟	1	1200	已退票	退票 改签 删除
中国国航 CA157空客321(中)	刘宇为	2024-10-12 18:16:53	浦东国际机场	关西国际机场	2024-10-13 21:10:00	2小时20分钟	1	986	已退票	退票 改签 删除
中国国航 CA157空客321(中)	刘宇为	2024-10-12 18:16:12	浦东国际机场	洛杉矶国际机场	2024-10-13 10:25:00	13小时	1	12505	已退票	退票 改签 删除



尊敬的会员:

刘宇为

欢迎回来! 今天抢到票了嘛?

[充值](#)
[修改密码](#)
[修改个人信息](#)

会员信息

用户名	手机号	邮箱
lyw	02502502502	lyw@xm.com
备注	余额	
学校	66602	

3. 改签功能:

改签分为两步:

第一步:先处理改签操作, 获取可选的航班, 并打开对话框。

在这一步中还是先将起飞时间与当前时间进行对比, 确

保飞机没有起飞。

然后查询出前端传回的航班 ID 的具体信息，并查询所有航班，再筛选出可以改签的航班条件包括：

起始城市与待改签航班的起始城市相同。

目的城市与待改签航班的目的城市相同。

当前航班的余票数量大于等于用户的订票数量。

过滤掉当前的航班 ID（即不能选择同一航班进行改签）。

最后将符合条件的航班收集到一个列表中并返回。

第一步前端关键代码（还是发送请求，传回数据）：

```
changeTicket(row) {  
  
  this.$request.post('/airticket/getChange', row).then(res => {  
  
    if (res.code === '200') {  
  
      this.changes = res.data  
  
      this.form = JSON.parse(JSON.stringify(row))  
  
      this.fromVisible = true  
  
    } else {  
  
      this.$message.error(res.msg)  
  
    }  
  
  })  
  
},
```

后端关键代码, 在 service 层, xml 中的增删改查功能与上述类似故不再展示:

```
public List<Airticket> getChange(Airorders airorders) throws ParseException {  
  
    // 1. 判断当前的时间是否已经过了起飞的时间  
  
    String now = DateUtil.now();  
  
    String flyTime = airorders.getFlyTime();  
  
    if (!TimeUtils.compareDate(flyTime, now, "yyyy-MM-dd HH:mm:ss")) {  
  
        throw new CustomException("-1", "您的航班已经起飞, 无法改签");  
  
    }  
  
    // 查询出待改签的航班  
  
    Integer ticketId = airorders.getTicketId();  
  
    Airticket airticket = airticketMapper.selectById(ticketId);  
  
    // 查询所有的航班  
  
    List<Airticket> airtickets = airticketMapper.selectAll(new Airticket());  
  
    // 筛选可改签的航班  
  
    return airtickets.stream()  
  
        .filter(x -> x.getStartCity().equals(airticket.getStartCity())  
  
            && x.getEndCity().equals(airticket.getEndCity())  
  
            && x.getNum() >= airorders.getNum()  
  
            && !x.getId().equals(ticketId))  
  
        .collect(Collectors.toList());  
}
```

```
}  
  
}
```

第一步 功能展示：

航班名称	用户姓名	下单时间	起始机场	到达机场	起飞时间	航班时长	机票数量	机票价格	订单状态	操作
中国国航 CA157空客321(中)	刘宇	2024-10-19 23:47:09	浦东国际机场	洛杉矶国际机场	2024-10-20 10:25:05	13小时	1	12505	已购买	<button>退票</button> <button>改签</button> <button>删除</button>
南方航空 CZ2809空客321 (大)	刘宇	2024-10-19 23:13:10	上海国际机场	合肥新桥机场	2024-10-20 16:00:00	2小时30分钟	1	1200	已购买	<button>退票</button> <button>改签</button> <button>删除</button>
吉祥航空 HO1196波音737 (中)	刘宇	2024-10-19 23:13:10	青岛国际机场	上海国际机场	2024-10-20 10:30:00	2小时30分钟	1	1300	已购买	<button>退票</button> <button>改签</button> <button>删除</button>
中国国航 CA157空客321(中)	刘宇	2024-10-19 23:12:16	浦东国际机场	洛杉矶国际机场	2024-10-20 10:05:00	13小时	1	12505	已退票	<button>退票</button> <button>改签</button> <button>删除</button>
吉祥航空 HO1196波音737 (中)	刘宇	2024-10-19 23:08:05	青岛国际机场	上海国际机场	2024-10-20 10:30:00	2小时30分钟	1	1300	已购买	<button>退票</button> <button>改签</button> <button>删除</button>

对浦东国际机场到洛杉矶国际机场的机票进行改签

航班名称	用户姓名
中国国航 CA157空客321(中)	刘宇
南方航空 CZ2809空客321 (大)	刘宇
吉祥航空 HO1196波音737 (中)	刘宇
中国国航 CA157空客321(中)	刘宇

改签信息

出发时间

2024-10-21

选择航班

请选择航班

中国国航 CA157空客321(中)~上海国际机场-洛杉矶国际机场~15:50:00-23:50:00

中国国航 CA157空客321(中)~浦东国际机场-洛杉矶国际机场~10:05:00-23:05:00

中国国航 CA157空客321(中)~浦东国际机场-洛杉矶国际机场~10:05:00-22:05:00

确定

操作

退票

改签

删除

退票

改签

删除

退票

改签

删除

退票

改签

删除

第二步：确认改签，前端发送改签请求，完成改签。

先对比用户所填改签时间与当前时间，只能改签为当日第二天起的票，

然后检查是否需要退补差价，如果需要补差价，检查余额是否充足，若不足则抛出异常提醒用户进行充值，充足则扣除余额，若需要退钱，则将多出的钱退回用户余额中

最后修改订单各项信息和改签与被改签两个航班的余票数

源代码：前端发送请求，这里的 begin 是存放的用户选择的改签日期，ticketid 是新航班的 id:

```

doChange() {

    this.form.ticketId = this.ticketId

    this.form.begin = this.begin

    console.log(this.form)

    this.$request.post('/airorders/change', this.form).then(res => {

        if (res.code === '200') {

            this.$message.success('改签成功')

            this.fromVisible = false

            this.load(1)

        } else {

            this.$message.error(res.msg)

        }

    })

}

```

后端关键代码，位于 service 层：

```

public void change(Airorders airorders) throws ParseException {

    // 获取今天的时间

    String today = DateUtil.format(new Date(), "yyyy-MM-dd");

    if (TimeUtils.compareDate(today, airorders.getBegin(), "yyyy-MM-dd")) {

        throw new CustomException("-1", "平台限定最早只能改签第二天的票");

    }

    // 计算一下是否需要补差价或者退差价

```

```

User user = userMapper.selectById(airorders.getUserId());

double price = airorders.getNum() * airorders.getPrice();

Integer ticketId = airorders.getTicketId();

Airticket airticket = airticketMapper.selectById(ticketId);

double nowPrice = airticket.getPrice() * airorders.getNum();

if (price < nowPrice) {

    double gap = nowPrice - price;

    if (user.getAccount() < gap) {

        throw new CustomException("-1", "您的余额不足，请到个人中心充值");

    } else {

        // 用户补差价

        user.setAccount(user.getAccount() - gap);

        userMapper.updateById(user);

    }

}

if (price > nowPrice) {

    double gap = price - nowPrice;

    // 平台退差价

    user.setAccount(user.getAccount() + gap);

    userMapper.updateById(user);

}

// 更新一下改签后的订单信息

```

```

        airorders.setName(airticket.getName());

        airorders.setFlyTime(airorders.getBegin() + " " + airticket.getStartTime());

        airorders.setStartAirport(airticket.getStartAirport());

        airorders.setEndAirport(airticket.getEndAirport());

        airorders.setTimeSlot(airticket.getTime());

        airorders.setPrice(airticket.getPrice());

        airordersMapper.updateById(airorders);
    }
}

```

功能展示：



改签后，各项信息均已改变

航班名称	用户姓名	下单时间	起始机场	到达机场	起飞时间	航班时长	机票数量	机票价格	订单状态	操作
中国国航 CA157空客321(中)	刘宇	2024-10-19 23:47:09	上海国际机场	洛杉矶国际机场	2024-10-21 15:50:00	8小时	1	10000	已购买	<button>退票</button> <button>改签</button> <button>删除</button>
南方航空 CZ2809空客321 (大)	刘宇	2024-10-19 23:13:10	上海国际机场	合肥新桥机场	2024-10-20 16:00:00	2小时30分钟	1	1200	已购买	<button>退票</button> <button>改签</button> <button>删除</button>
吉祥航空 HO1196波音737 (中)	刘宇	2024-10-19 23:13:10	青岛国际机场	上海国际机场	2024-10-20 10:30:00	2小时30分钟	1	1300	已购买	<button>退票</button> <button>改签</button> <button>删除</button>

改签成的航班票数由 20 变成 19，被改签的航班票数恢复成 83





4. 处理国际航班时差:

这里对于这个功能的实现是实现的显示每个起止城市所在地的时间，为实现这个功能首先要在航班信息中设置起飞时区和目的时区两个属性，再根据这两个属性和起止时间进行时差处理，









具体关键源代码位于前端，如下，展示的是对目的时间的处理，起飞时间类似不重复展示，

1. 先将 `endTime` 字符串(格式应为 "HH:mm")通过 `split(':')` 方法拆分为小时和分钟两个部分，并使用 `map(Number)` 将它们转换为数字类型。此 `hour` 和 `minutes` 将分别存储小时和分钟的数值。
2. 然后生成一个新的 `Date` 对象(后续将基于这个对象来设置新的时间)
3. 然后用 `hours` 减去起始城市的时区与目的城市的时区的时差，需要注意的是这里也要判断处理完时差 `hours` 后是否超过了 24，超过了先减去 24 然后要在时间右角上显示 +1 代表这是第二天(代码中是 `bool` 变量 `plus` 为 `true` 就显示出来)，

4. 最后更新小时和分钟并进行格式化处理。

```
updatedEndTime(item) {  
  
  if ( item.endTime ) {  
  
    const [hours, minutes] = item.endTime.split(':').map(Number);  
  
    const date = new Date();  
  
    let adjustedHours = hours - item.uit1 + item.uit2;  
  
    console.log(adjustedHours);  
  
    if (adjustedHours >= 24) {  
  
      adjustedHours =adjustedHours-24;  
  
      item.plus = true;  
  
    } else {  
  
      item.plus = false;  
  
    }  
  
    date.setHours(adjustedHours);  
  
    date.setMinutes(minutes);  
  
    return date.toTimeString().slice(0, 5); // 格式化为 HH:mm  
  
  }  
  
}
```

功能展示：

	中国国航 CA157空客321(中)	15:50 上海国际机场	08:50 洛杉矶国际机场	8小时	20	10000 含税价	购票
	中国国航 CA157空客321(中)	10:25 浦东国际机场	08:25 洛杉矶国际机场	13小时	82	12505 含税价	购票
	中国国航 CA157空客321(中)	21:10 浦东国际机场	00:30 ⁻¹ 关西国际机场	2小时20分钟	86	986 含税价	购票
	中国国航 CA157空客321(中)	10:05 浦东国际机场	08:05 洛杉矶国际机场	13小时	90	12505 含税价	购票
	中国国航 CA157空客321(中)	15:10 浦东国际机场	19:30 关西国际机场	2小时20分钟	85	1159 含税价	购票
	中国国航 CA157空客321(中)	10:05 浦东国际机场	07:05 洛杉矶国际机场	12小时	6	11805 含税价	购票
	厦门航空 MF4706空客320 (大)	13:30 上海国际机场	16:00 北京大兴国际机场	2小时30分钟	93	1200 含税价	购票
	南方航空 CZ2809空客321 (大)	16:00 上海国际机场	18:30 关西国际机场	2小时30分钟	93	1200 含税价	购票

1	四川航空 3U8198波音777	http://localhost:9090/fil	08:00:00	10:15:00	重庆国际机场	上海国际机场	1200.00	2小时15分钟	0	重庆
2	山东航空 SC7660波音777	http://localhost:9090/fil	10:00:00	13:00:00	烟台国际机场	上海国际机场	1300.00	3小时	99	烟台
3	东方航空 MU6639波音73	http://localhost:9090/fil	13:00:00	15:00:00	大连国际机场	上海浦东机场	1400.00	2小时	100	大连
4	吉祥航空 HO1196波音73	http://localhost:9090/fil	10:30:00	13:00:00	青岛国际机场	上海国际机场	1300.00	2小时30分钟	94	青岛
5	上海航空 FM9096空客320	http://localhost:9090/fil	13:00:00	15:00:00	上海国际机场	合肥新桥机场	1000.00	2小时	98	上海
6	南方航空 CZ2809空客321	http://localhost:9090/fil	16:00:00	18:30:00	上海国际机场	合肥新桥机场	1200.00	2小时30分钟	93	上海
7	厦门航空 MF4706空客320	http://localhost:9090/fil	13:30:00	16:00:00	上海国际机场	北京大兴国际机场	1200.00	2小时30分钟	93	上海
8	中国国航 CA157空客321	http://localhost:9090/fil	10:05:00	22:05:00	浦东国际机场	洛杉矶国际机场	11805.00	12小时	6	上海
9	中国国航 CA157空客321	http://localhost:9090/fil	15:10:00	18:30:00	浦东国际机场	关西国际机场	1159.00	2小时20分钟	85	上海
10	中国国航 CA157空客321	http://localhost:9090/fil	10:05:00	23:05:00	浦东国际机场	洛杉矶国际机场	12505.00	13小时	90	上海
11	中国国航 CA157空客321	http://localhost:9090/fil	21:10:00	23:30:00	浦东国际机场	关西国际机场	986.00	2小时20分钟	86	上海
12	中国国航 CA157空客321	http://localhost:9090/fil	10:25:05	23:25:00	浦东国际机场	洛杉矶国际机场	12505.00	13小时	82	上海
13	中国国航 CA157空客321	http://localhost:9090/fil	15:50:00	23:50:00	上海国际机场	洛杉矶国际机场	10000.00	8小时	20	上海

5. 一城多机场

数据库的航班信息中每个航班的起止机场与起止城市相对应，当查询起止城市时，该城市对应的所有机场的航班信息就都会显示。

源代码：主要体现在数据库设计上，查询功能与上述展示的xml 文件类似，不重复展示

功能演示：查询起飞城市为上海的结果如下

请输入航班名称	上海	请输入到达机场	交换起止机场	Q 查询	O 重置
航班名称	起始机场	到达机场	航班时长	剩余票数	
中国国航 CA157空客321(中)	15:50 上海国际机场	08:50 洛杉矶国际机场	8小时	20	10000¥起 含税价 购票
中国国航 CA157空客321(中)	10:25 浦东国际机场	08:25 洛杉矶国际机场	13小时	82	12505¥起 含税价 购票
中国国航 CA157空客321(中)	21:10 浦东国际机场	00:30 关西国际机场	2小时20分钟	86	986¥起 含税价 购票
中国国航 CA157空客321(中)	10:05 浦东国际机场	08:05 洛杉矶国际机场	13小时	90	12505¥起 含税价 购票
中国国航 CA157空客321(中)	15:10 浦东国际机场	19:30 关西国际机场	2小时20分钟	85	1159¥起 含税价 购票
中国国航 CA157空客321(中)	10:05 浦东国际机场	07:05 洛杉矶国际机场	12小时	6	11805¥起 含税价 购票
厦门航空 MF801空客321(中)	13:30 上海浦东国际机场	16:00 洛杉矶国际机场	2小时30分钟	93	1200¥起 含税价 购票
南方航空 CZ330空客321(中)	16:00 上海浦东国际机场	18:30 洛杉矶国际机场	2小时30分钟	93	1200¥起 含税价 购票

除此之外还有实现的一些其他功能但不是主要功能（像各种增删改查，还有用百度地图 API 实现的各种功能，管理员端可以查看票务售卖详情等等，在录屏中有展示）

五、系统测试（可选）

本章内容可选。

设计测试用例、记录测试用例的执行情况，

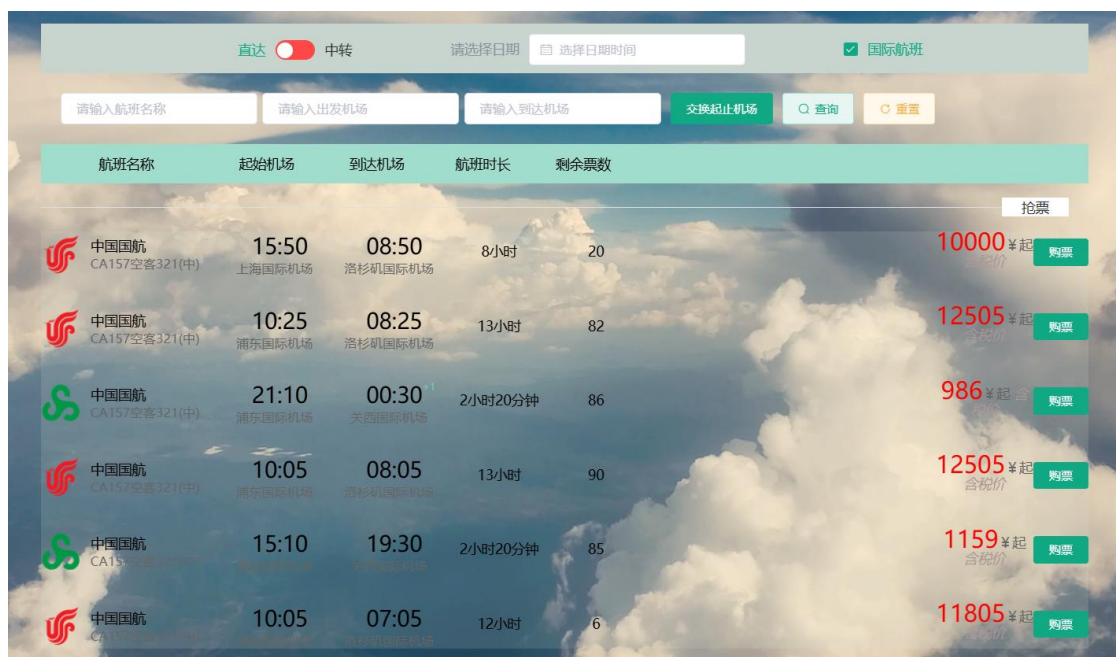
中转功能：查询重庆到洛杉矶的中转航班，

请输入航班名称	重庆	洛杉矶	交换起止机场	Q 查询	O 重置
航班名称	起始机场	到达机场	一程余票数	二程余票数	
四川航空 3U8198波音777 (中) CA157空客321(中)	08:00 重庆国际机场	中转站 08:25 浦东国际机场	0	82	13705¥起 含税价 购票
四川航空 3U8198波音777 (中)	08:00 重庆国际机场	中转站 08:50 上海浦东国际机场	0	20	11200¥起 含税价 购票

四川航空 3U8198波音777	http://localhost:9090/file	08:00:00	10:15:00	重庆国际机场	上海国际机场	1200.00	2小时15分钟	0
山东航空 SC7660波音777	http://localhost:9090/file	10:00:00	13:00:00	烟台国际机场	上海国际机场	1300.00	3小时	99
东方航空 MU6639波音73	http://localhost:9090/file	13:00:00	15:00:00	大连国际机场	上海浦东机场	1400.00	2小时	100
吉祥航空 HO1196波音73	http://localhost:9090/file	10:30:00	13:00:00	青岛国际机场	上海国际机场	1300.00	2小时30分钟	94
上海航空 FM9096空客321	http://localhost:9090/file	13:00:00	15:00:00	上海国际机场	合肥新桥机场	1000.00	2小时	98
南方航空 CZ2809空客321	http://localhost:9090/file	16:00:00	18:30:00	上海国际机场	合肥新桥机场	1200.00	2小时30分钟	93
厦门航空 MF4706空客321	http://localhost:9090/file	13:30:00	16:00:00	上海国际机场	北京大兴国际机场	1200.00	2小时30分钟	93
中国国航 CA157空客321	http://localhost:9090/file	10:05:00	22:05:00	浦东国际机场	洛杉矶国际机场	11805.00	12小时	6
中国国航 CA157空客321	http://localhost:9090/file	15:10:00	18:30:00	浦东国际机场	关西国际机场	1159.00	2小时20分钟	85
中国国航 CA157空客321	http://localhost:9090/file	10:05:00	23:05:00	浦东国际机场	洛杉矶国际机场	12505.00	13小时	90
中国国航 CA157空客321	http://localhost:9090/file	21:10:00	23:30:00	浦东国际机场	关西国际机场	986.00	2小时20分钟	86
中国国航 CA157空客321	http://localhost:9090/file	10:25:05	23:25:00	浦东国际机场	洛杉矶国际机场	12505.00	13小时	82
中国国航 CA157空客321	http://localhost:9090/file	15:50:00	23:50:00	上海国际机场	洛杉矶国际机场	10000.00	8小时	20

可以看出到洛杉矶的机场只有倒数一二个可以中转，其他从上海去洛杉矶的机票起飞时间都是 10:05，与 10:15 才到的第一程有时间冲突

只显示国际航班的功能



其余测试详见录屏

六、总结

对于数据库课程设计进行总结，包括经验和教训以及个

人在数据库课程设计中的感悟等。

完成了这个花费了我足足两个月的时间来制作的售票系统后，心里有许多话想说，这次课设是我第一次独立完成的项目从数据库的设计到后端实现各种功能，再到前端各种界面绘制真的让我学到了很多，也许我做的课设还是没有别人好，这也是我认真尽全力完成的，也许有一天以后在面试时可以将这个课设展示出来。总体而言，老师 ppt 上要求的功能像是改签，退票，国际航班时差等等，都有尽力去实现，也加入了许多自己的东西。

一开始制作时想着参照一个售票系统来完成可能会显得比较专业，于是就照着携程的网站来制作，结果可能自己照着携程制作的功能有些偏题，当时也是只顾着想多做些功能了，没来得及抬头看路，可能做了许多无用的功能。

这个课设让我对数据库的设计的了解提升了很多，而且让我了解到有些功能的实现虽然也要储存信息但是不一定要存到数据库中，就像我的中转功能的实现可以用类似于临时表来做，而且也让我熟悉了 `spring boot + vue` 框架的开发，也学会了许多快捷键，就像 `alt+ins` 可以快速设置 `get` 和 `set` 方法，`ctrl+r` 可以快速实现查找和全局替换，我所制作的火车票和汽车票等等都是用这个方法加上复制粘贴和稍作改动完成的简陋版本，而且学会了利用各大厂商提供的

API 来实现功能这个方法，就像项目中用百度地图 API 实现的天气报道功能以及搜索景点和查看地图等各项功能（一个制作网约车的同学告诉我这个不难做就能实现很多功能就试着做了一下，结果展示前天晚上熬夜做的导致展示时出了点错误）以及学会了如何制作**动态的网页背景**，也学会了使用 `console.log` **查看数据情况进行调试的方法**。

教训方面，对于这个课设我的制作其实也有许多不足，一个是未能完成火车票等的换乘等相应功能，只有基础的购票，查询，改签，退票等功能，还有测试时一定要进行多组数据测试，不然就会像我展示的时候一样出现问题。而且一定要**考虑细致**，不然会出现许多很离谱的错误，就像我一开始制作的改签功能忘记对票数进行相应变动，就会导致改签完后买的是另一张票，减少的还是原来的票。