飞机票售票系统报告

目录

[一、系统概述 3](#_Toc178189275)

[二、需求分析 3](#_Toc178189276)

[2.1 系统功能分析 3](#_Toc178189277)

[2.2 系统数据分析 3](#_Toc178189278)

[2.3 系统非功能分析 3](#_Toc178189279)

[三、系统设计 3](#_Toc178189280)

[3.1 应用程序设计 3](#_Toc178189281)

[3.2 数据库设计 3](#_Toc178189282)

[3.2.1 概念设计 3](#_Toc178189283)

[3.2.2 逻辑设计 3](#_Toc178189284)

[3.2.3 物理设计 3](#_Toc178189285)

[四、系统实现 3](#_Toc178189286)

[4.1 关键技术实现 3](#_Toc178189287)

[4.2 功能实现 3](#_Toc178189288)

[4.2.1注册与登录 3](#_Toc178189289)

[4.2.2用户主页面： 3](#_Toc178189290)

[4.2.3查询页面 3](#_Toc178189291)

[4.2.4购票 3](#_Toc178189292)

[4.2.5查看以及支付订单 3](#_Toc178189293)

[4.2.6退票与改签 3](#_Toc178189294)

[五、系统测试 3](#_Toc178189295)

[六、总结 3](#_Toc178189296)

# 一、系统概述

本系统用于实现售卖飞机票的工作，包括管理员对售票进行管理，以及用户查询购票和退票等操作。

本项目的设计目标旨在设计出具有稳定且可拓展性好的数据库设计、逻辑清晰且高效的数据库操作、简洁直观且稳健性好的操作页面的系统。

该系统对于飞机的航班概念具有不同于一般概念的理解。这里将飞机每日在固定时间的从某机场飞到某机场的一个行为成为“航班”，而将一个航班和某一天联系起来的结果称为“调度”。譬如，每天8:00 - 10:00的从北京大兴机场飞到上海虹桥机场，且航班号为CZ4447的一次行为称为“航班”；而给这个航班附加9月1号的飞行任务，得到的就是“调度”。

# 二、需求分析

## 2.1 系统功能分析

该系统对于实现的功能可分为以下几个模块：航班、调度、用户、乘客、订单、消息。

该系统设计了两个权限：管理员和用户。

管理员应实现的功能：对以上所述的各个模块进行管理，对相关的数据库表进行增删改查操作。

用户应实现的功能如下：

1. 注册、登录以及修改个人信息
2. 查询航班和调度，查询自己的订单、自己绑定的乘客、收到的消息
3. 购票、退票、改签
4. 绑定乘客
5. 支付订单、取消订单
6. 查询余额以及充值

## 2.2 系统数据分析

首先，本系统需要在前端存储用户的id以及token字段用于校验登录状态。该数据需要在前端向后端发送请求时附带在请求头中。

其次，在数据库中还应存储有如下数据：

1. 航班和调度相关数据：

航班（航班编号，航空公司，机型，起始时间，到达时间，起始城市，到达城市，起始机场，到达机场）

调度（航班编号，起始日期，剩余票数，价格，是否准点）

1. 用户和乘客相关数据：

用户（用户编号，用户名，密码，余额，类型）

用户类型（类型）

乘客（用户编号，乘客名，乘客身份证号）

1. 订单、消息等其他数据：

订单（订单号，用户编号 ，乘客身份证号，航班编号，日期，金额，提交时间，是否已支付）

消息（消息号，用户编号，消息内容，提交时间）

## 2.3 系统非功能分析

系统安全性：

1. 用户登录状态检验：

在我们的系统，我们应当始终保持用户处于登录状态，从而使我们的前后端传输正常运行。

我使用了JWT库，用户登录时通过后端生成Token传送到前端，Token中存有用户id，用户权限等信息，用户得到Token后需要将Token存储在前端内存中，然后每次向后端传输信息都需要在请求头中存放Token，后端通过对请求头进行校验就能知道该请求是否合法。

1. 数据库加密

对于密码等敏感信息，我们不能直接将它们存入数据库，而是需要加密后存储。这里使用BCrypt加密，该加密方法难以被破解，安全性高。而且对密码的校验也不是将原密码解密出来实现的，用户信息更为安全。

1. 权限检验

通过自定义注解的方式，实现对不同方法的权限校验。在方法上添加注解，如果前端的请求权限不足或者根本没有权限，那么后端在方法执行前就会将其拦截。

1. 并发处理

通过使用**synchronized**关键字对关键方法加锁，从而实现购票等操作的安全并发操作。

同时可以使用**@Transactional**事务注解对数据库的更新等操作需要添加事务锁，对于并发处理的提交回滚等操作，由**springboot**为我们处理。

# 三、系统设计

## 3.1 应用程序设计

一、技术实现

该系统前端采用vue2构建，使用element-ui框架进行页面美化，使用axios与后端进行数据传输。

后端使用springboot，数据库使用Mysql，持久层采用JPA（Java Persistence API），JDK17.0.1。

二、项目结构树

后端结构展示：

─com

└─yrh

└─ff\_ticket

├─config

├─controller

├─converter

├─dao

│ ├─model

│ ├─repository

│ ├─service

│ └─serviceImpl

├─jwt

├─payload

│ ├─dto

│ └─response

├─role

└─util

前端结构展示：

├─assets

│ ├─img

│ └─test

├─components

│ ├─admin

│ │ ├─dialogs

│ │ └─mainViews

│ └─user

│ ├─dialogs

│ ├─mainViews

│ └─userCenterViews

├─router

├─utils

└─views

├─admin

└─user

三、项目详解

1.后端可分为controller层，service层，payload层，dao层。

controller层：后端提供给前端的所有接口

service层：编写各种业务逻辑，其中每个业务都通过接口方式规范

payload层：后端与前端数据传输的规范化层

dao层：用于后端与数据库连接的诸多操作

2.前端可分为assets层，components层，router层，utils层，view层。

assets层：项目图标，图片

components层：项目各种组件

router层：设置路由

utils层：工具类，其中有axios封装，用于与后端通信

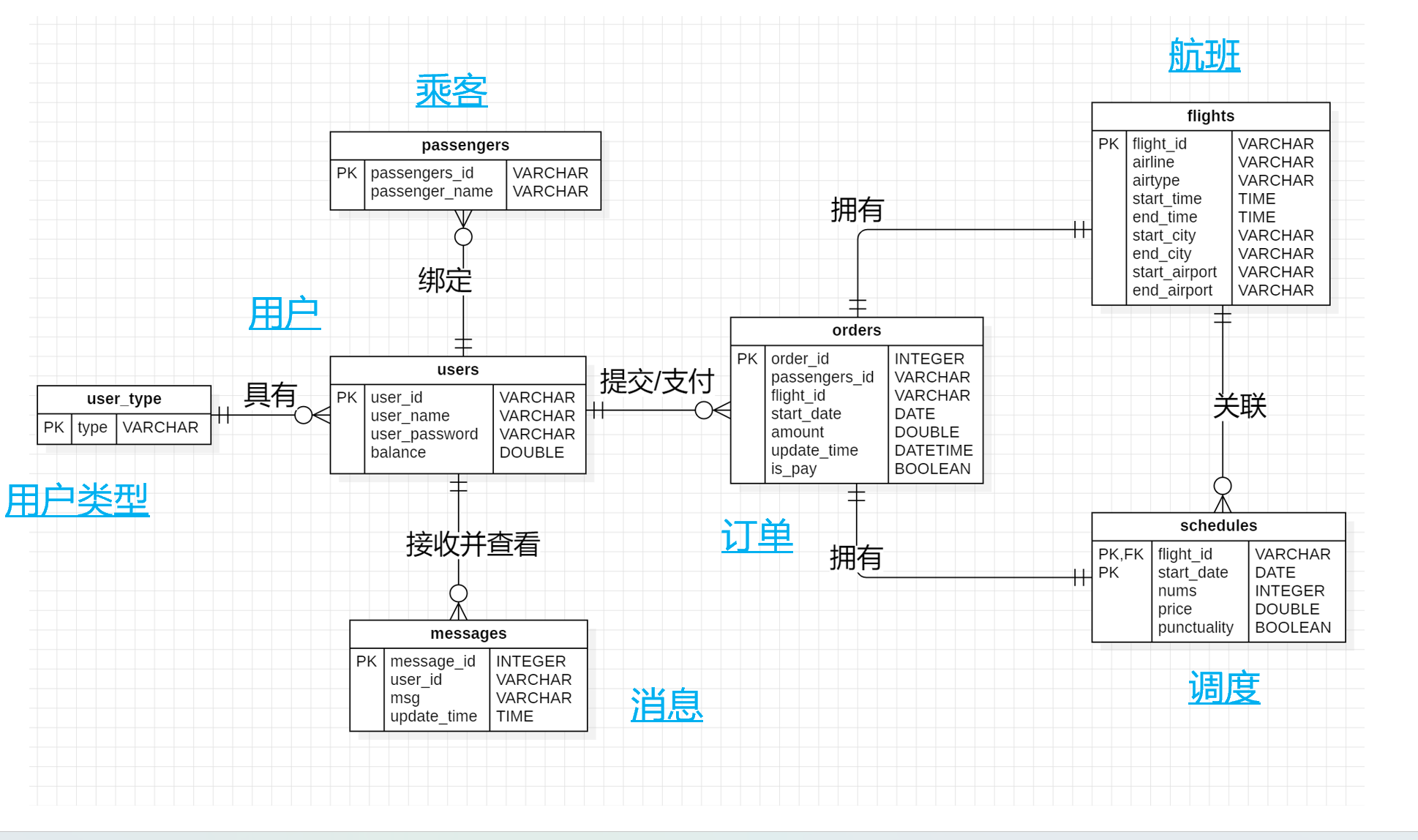
view层：部分页面

## 3.2 数据库设计

### 3.2.1 概念设计

该项目结构存在如下七个实体表，ER图如下。

同时将flights表和schedules表进行自然连接形成视图。



### 3.2.2 逻辑设计

根据ER图可以设计出如下数据库表：

航班（航班编号，航空公司，机型，起始时间，到达时间，起始城市，到达城市，起始机场，到达机场）

调度（航班编号，起始日期，剩余票数，价格，是否准点）

用户（用户编号，用户名，密码，余额，类型）

用户类型（类型）

乘客（用户编号，乘客名，乘客身份证号）

订单（订单号，用户编号 ，乘客身份证号，航班编号，日期，金额，提交时间，是否已支付）

消息 （消息号，用户编号，消息内容，提交时间）

flights( flight\_id , airline , airtype , start\_time , end\_time , start\_city , end\_city , start\_airport , end\_airport )

schedules( flight\_id , start\_date , nums , price , punctuality )

users( user\_id , user\_name , user\_password , balance , type)

passengers( user\_id , passenger\_name , passenger\_id)

orders( order\_id , user\_id , passenger\_id , flight\_id , start\_date , amount , update\_time , is\_pay )

message( message\_id , user\_id , msg , update\_time )

建表语句;

DROP DATABASE IF EXISTS ticket\_project;  
CREATE DATABASE ticket\_project  
 CHARACTER SET utf8mb4  
 COLLATE utf8mb4\_general\_ci;  
  
use ticket\_project;

创建航班表  
DROP TABLE if EXISTS flights;  
CREATE TABLE flights(  
 flight\_id VARCHAR(100) NOT null,  
 airline VARCHAR(100) DEFAULT null,  
 airtype VARCHAR(100) DEFAULT null,  
 start\_time TIME DEFAULT null,  
 end\_time TIME DEFAULT null,  
 start\_city VARCHAR(100) DEFAULT null,  
 end\_city VARCHAR(100) DEFAULT null,  
 start\_airport VARCHAR(100) DEFAULT null,  
 end\_airport VARCHAR(100) DEFAULT null,  
 PRIMARY KEY(flight\_id)  
);

创建调度表  
DROP TABLE IF EXISTS schedules;  
CREATE TABLE schedules(  
 flight\_id VARCHAR(100) not null,  
 start\_date DATE,  
 nums int DEFAULT 0,  
 price DOUBLE DEFAULT 0,  
 punctuality bool DEFAULT TRUE,  
 PRIMARY KEY(flight\_id , start\_date ),  
 FOREIGN KEY(flight\_id) REFERENCES flights(flight\_id)  
);

创建用户类型表  
DROP TABLE IF EXISTS user\_type;  
CREATE TABLE user\_type(  
 type VARCHAR(100) not null,  
 PRIMARY KEY( type )  
);

创建用户表  
DROP TABLE IF EXISTS users;  
CREATE TABLE users(  
 user\_id VARCHAR(100) not null,  
 user\_name VARCHAR(100) not null,  
 user\_password VARCHAR(100) not null,  
 balance DOUBLE DEFAULT 0,  
 type VARCHAR(100) not null,  
 PRIMARY KEY( user\_id ),  
 FOREIGN KEY( type ) REFERENCES user\_type(type)  
);

创建乘客表  
DROP TABLE IF EXISTS passengers;  
CREATE TABLE passengers(  
 passenger\_id VARCHAR(100) not NULL,  
 passenger\_name VARCHAR(100) not null,  
 user\_id VARCHAR(100) not null,  
 PRIMARY KEY( passenger\_id ),  
 FOREIGN KEY( user\_id ) REFERENCES users( user\_id ),  
 INDEX(user\_id)  
);

创建订单表  
DROP TABLE IF EXISTS orders;  
CREATE TABLE orders(  
 order\_id INTEGER not NULL AUTO\_INCREMENT,  
 user\_id VARCHAR(100) not NULL,  
 passenger\_id VARCHAR(100) not null,  
 flight\_id VARCHAR(100) not null,  
 start\_date DATE,  
 amount int,  
 update\_time DATETIME,  
 is\_pay bool DEFAULT FALSE,  
 PRIMARY KEY( order\_id ),  
 FOREIGN KEY( user\_id ) REFERENCES users( user\_id ),  
 FOREIGN KEY( passenger\_id ) REFERENCES passengers( passenger\_id ),  
 FOREIGN KEY( flight\_id ) REFERENCES flights( flight\_id ),  
 INDEX(user\_id)  
);

创建消息表  
DROP TABLE IF EXISTS messages;  
CREATE TABLE messages(  
 message\_id INTEGER not NULL AUTO\_INCREMENT,  
 user\_id VARCHAR(100) not NULL,  
 msg VARCHAR(1000),  
 update\_time DATETIME,  
 PRIMARY KEY( message\_id )  
);

创建航班和调度的视图  
DROP VIEW IF EXISTS flightsandschedules;  
CREATE VIEW flightsandschedules  
(  
 flight\_id , airline , airtype , start\_time , end\_time , start\_city , end\_city , start\_airport , end\_airport , start\_date , nums , price , punctuality  
)  
AS SELECT  
flight\_id , airline , airtype , start\_time , end\_time , start\_city , end\_city , start\_airport , end\_airport , start\_date , nums , price , punctuality  
FROM (flights NATURAL JOIN schedules);

### 3.2.3 物理设计

因为用户需要且只需要知道自己所绑定的乘客，所以在乘客表passengers中给user\_id建立索引。订单表同理，我们也在orders表中给user\_id建立索引。

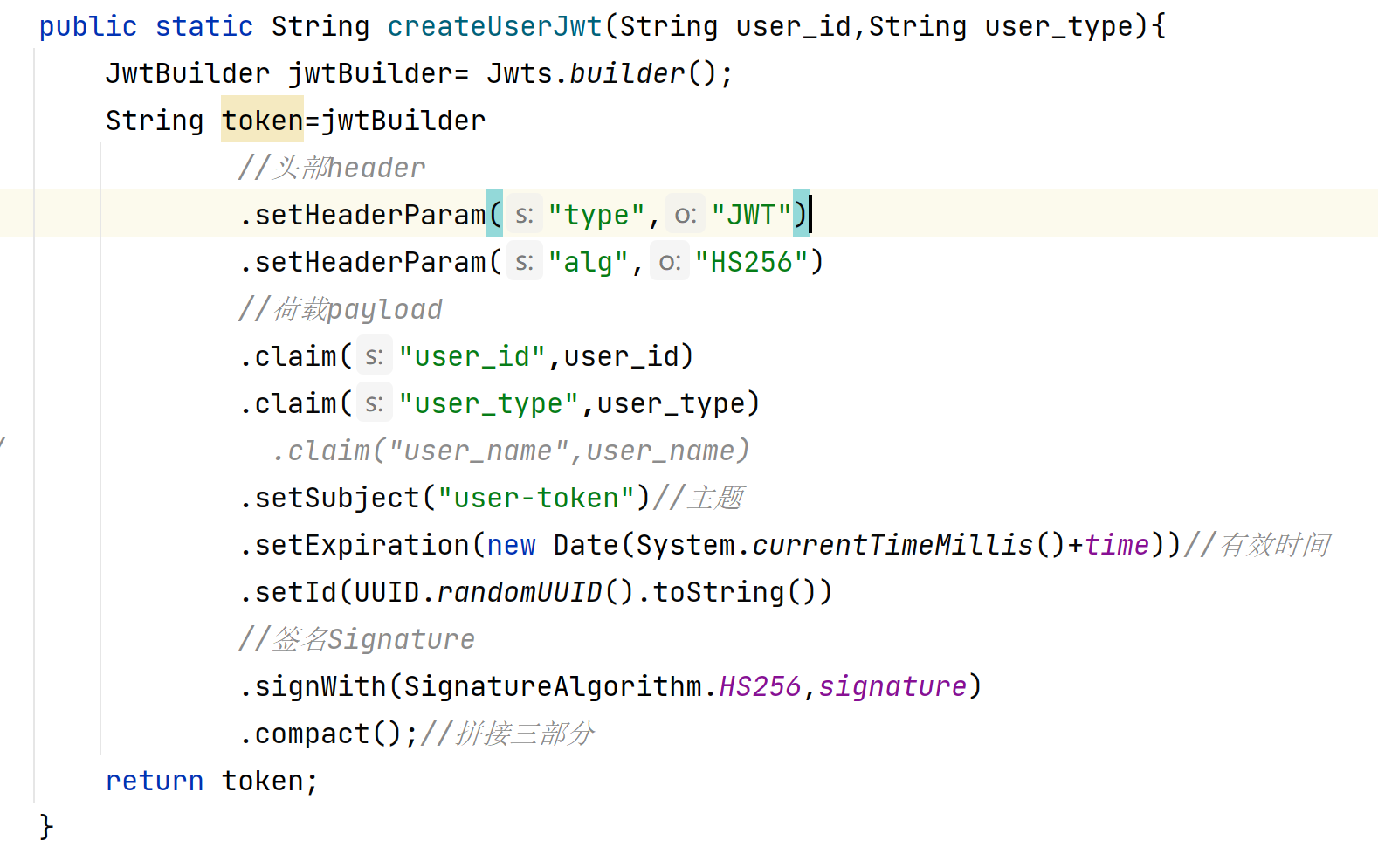
建立索引在建表时即可通过INDEX关键字完成。

# 四、系统实现

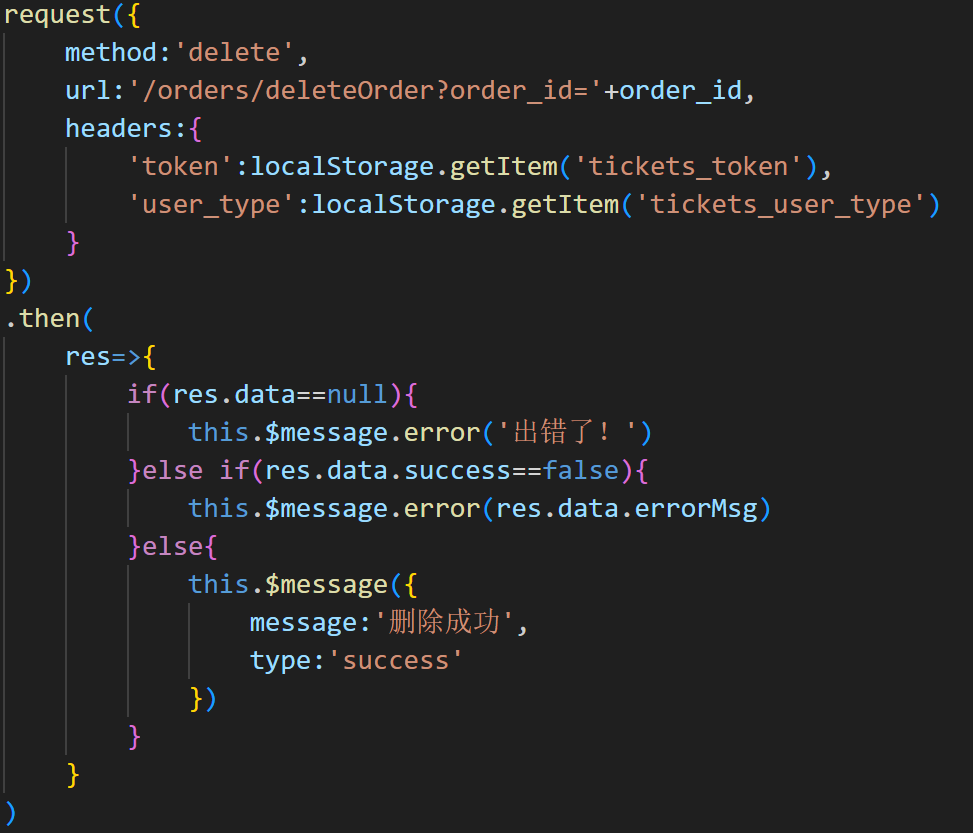
## 4.1 关键技术实现

1．token校验

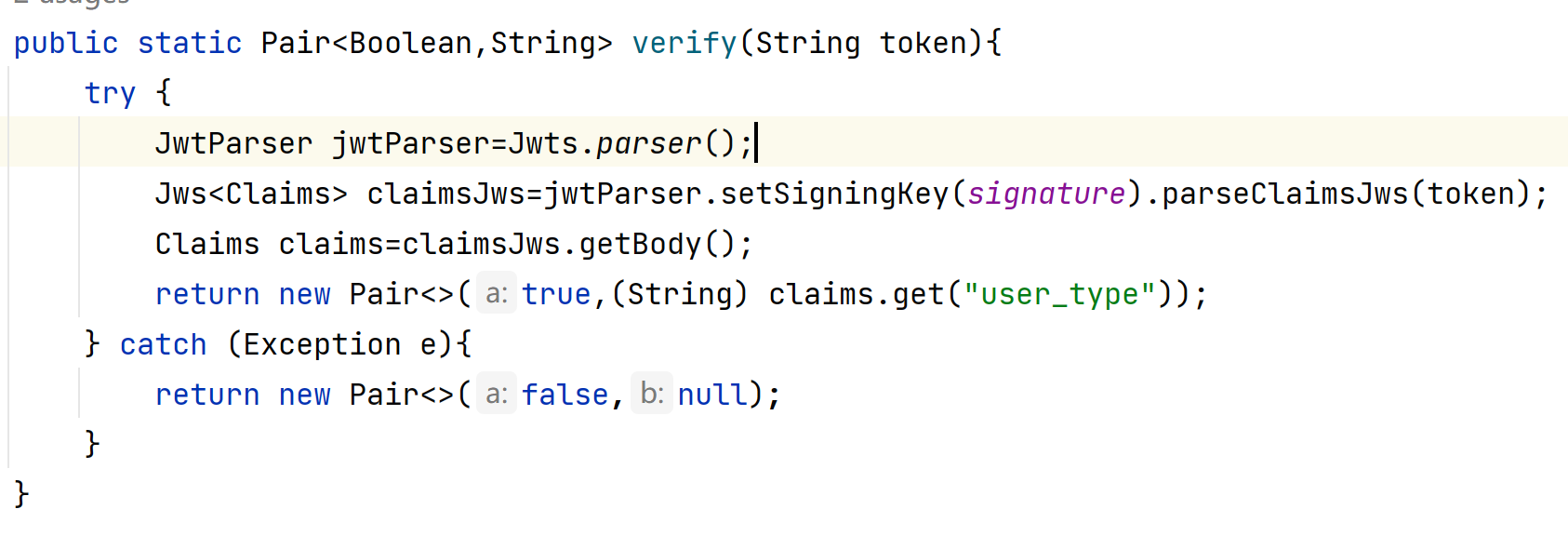
后端可以通过用户id，用户权限生成token字段，代码如下：



同时，前端向后端所发出的所有的request都要在请求头中存放token，如：



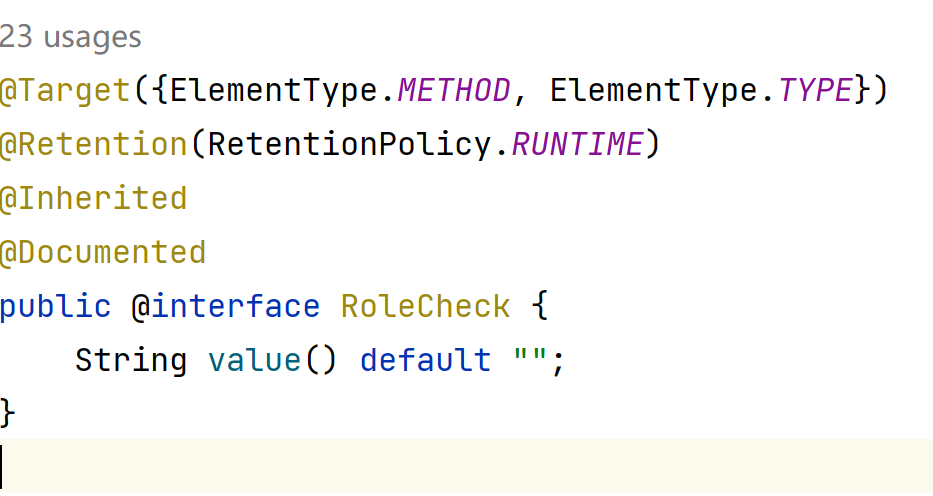
后端需要对前端的token进行校验：



2．权限校验

通过自定义注解，我们可以实现对权限的拦截。

自定义注解：



拦截层：



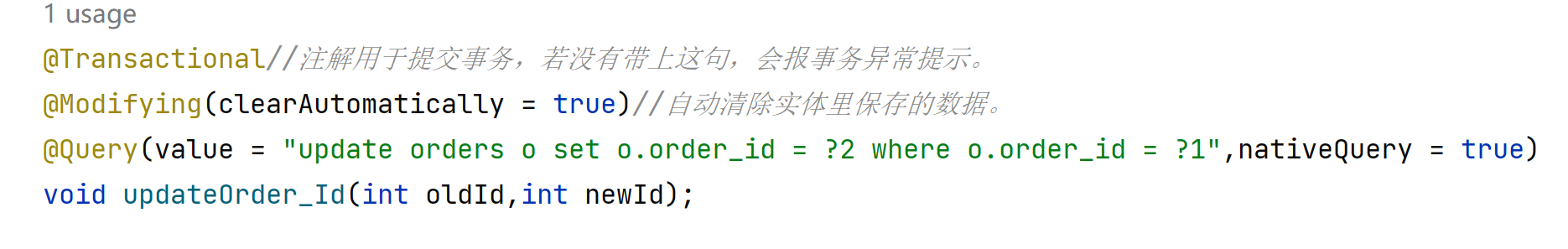
3．并发处理

对于需要一步完成的操作，我们可以加上关键字synchronized，如：



同时对于数据库更新等操作，使用@Transactional

注解构建一个事务，如下：



4．数据库加密

采用BCrypt加密,我们可以通过导入hutool包中的类快速实现，具体实现如下：



在涉及密码操作时，我们将会将明码转化成密文后存入数据库，校验密码时取出密文，直接使用方法校验即可。

Bcrypt加密在哈希时引入了盐值，使得相同的明文密码在每次加密时都会生成产生不同的密文。由于哈希函数是单向的，即使攻击者获得了存储的哈希值，也无法轻易还原出原始密码。

## 4.2 功能实现

### 4.2.1注册与登录

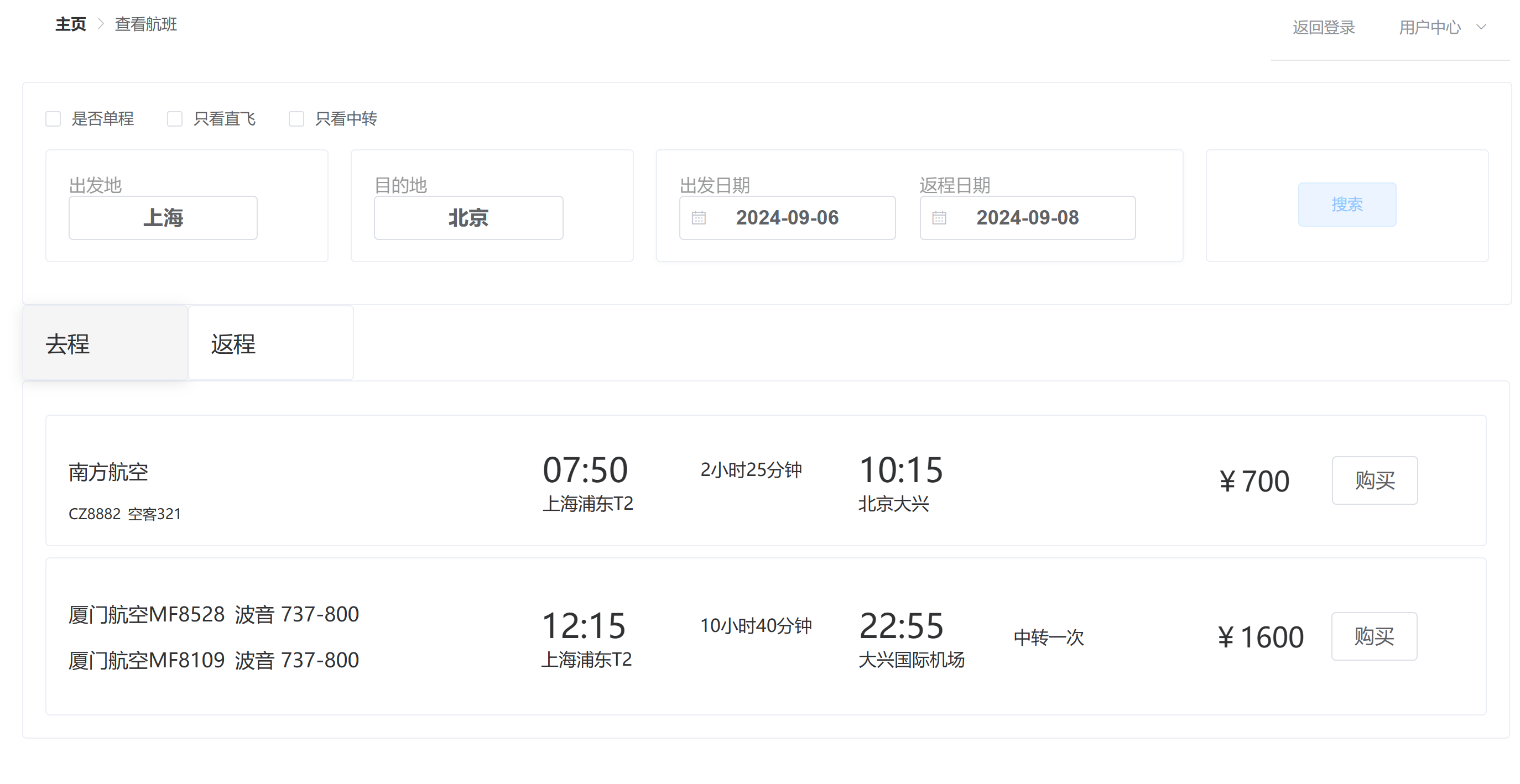


### 4.2.2用户首页面：



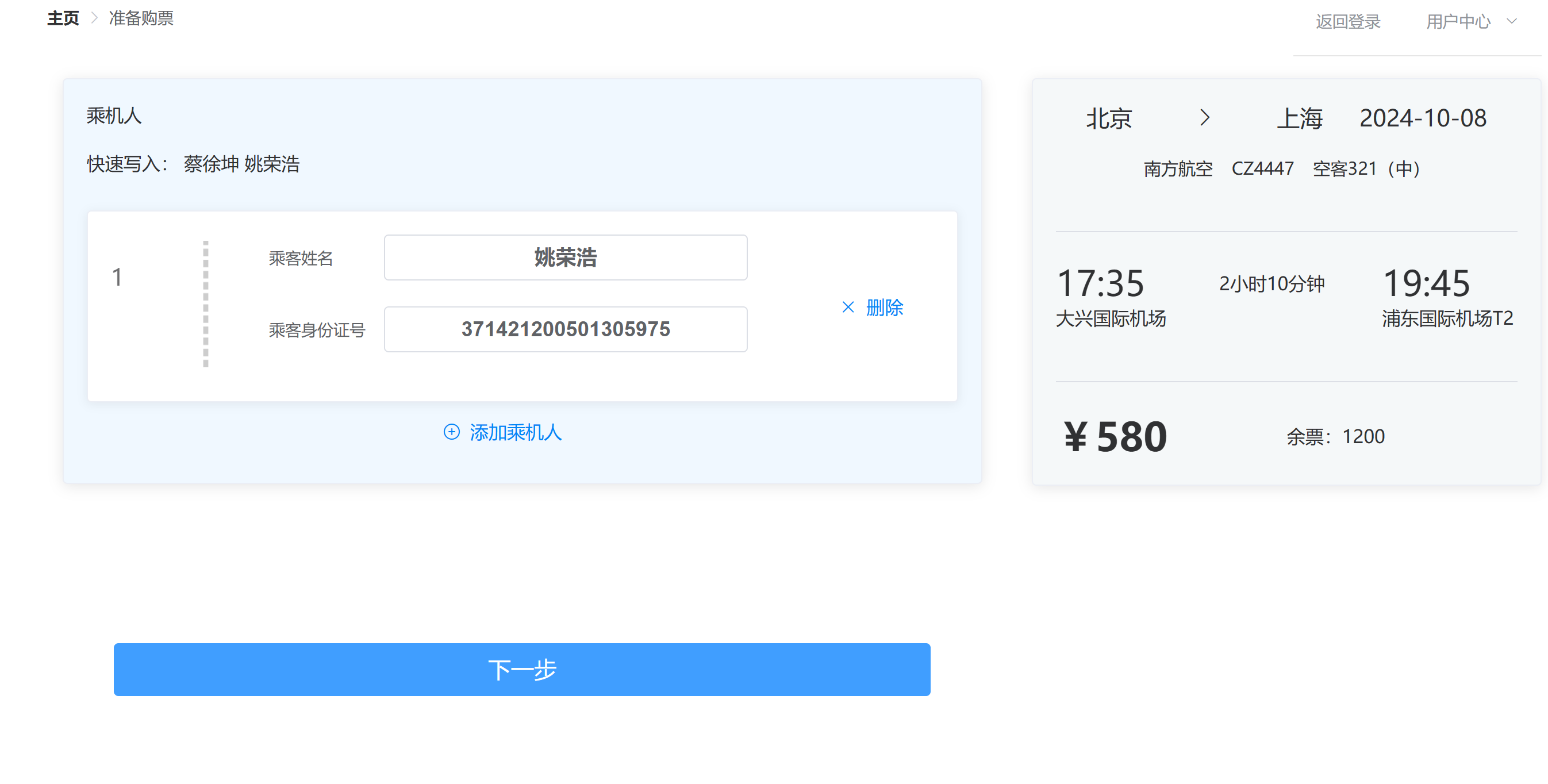
在这个页面可以进行航班的搜索，通过输入出发地，目的地，出发日期，返程日期，以及其他可选项，点击搜索后用户将跳转到对应的航班展示页面。

### 4.2.3查询页面

在键入搜索选项后，我们就可以查询出当天的航班，如图：

用户可以选择各种选项进行查询，可以选择查看单程以及查看返程，可以选择直飞或者中转。用户搜索出航班后就可以在下方的航班卡片展示，可以查看到航班的各种信息，并进行购票操作。

### 4.2.4购票



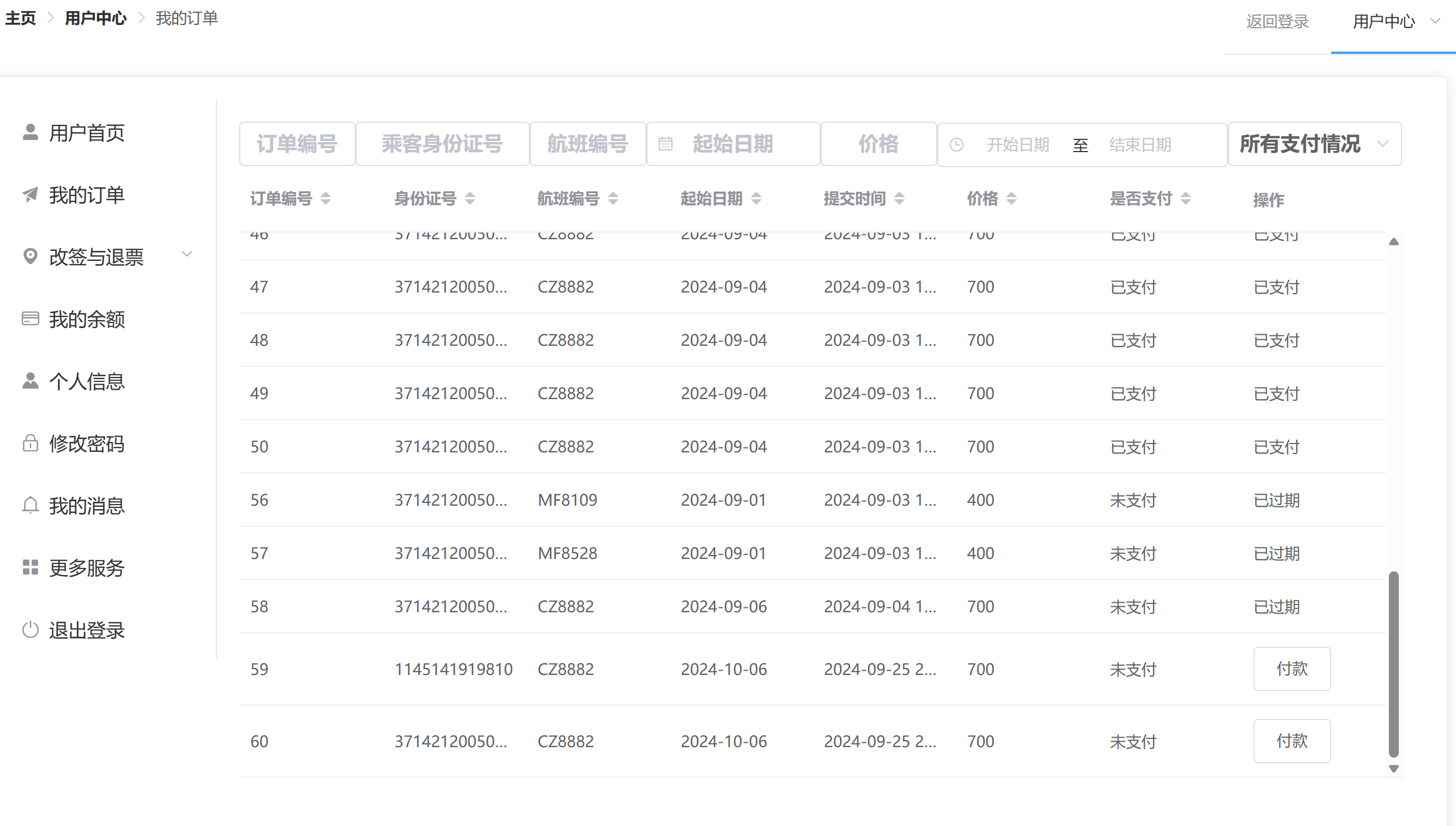
在购票页面用户可以进一步查看到航班的信息，同时为自己已经绑定的乘客购票，确认购票后将提交订单，用户可取支付订单从而完成购票。



### 4.2.5用户中心页面

用户中心页面给用户提供了若干操作。

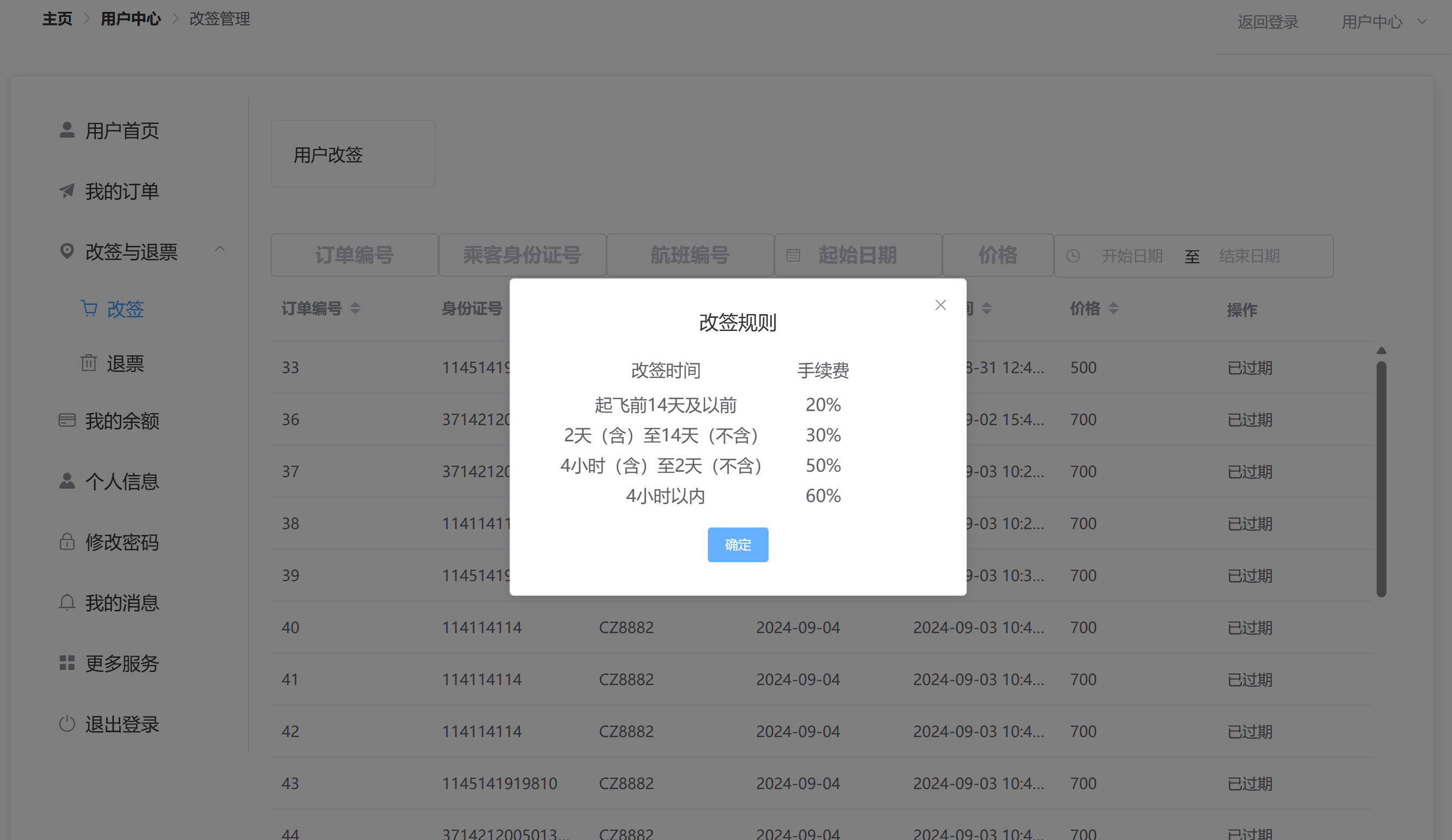
1.查看并支付订单



订单操作位于用户中心页面的子页面，在这里用户可以查看自己已经提交的订单并支付订单。上方的过滤器可以快速查询到符合用户需求的订单。

2. 退票与改签

只要遵守改签以及退票规则，用户就可以退票和改签。用户可以在改签和退票页面查看自己已经支付的订单并对其改签或者退票。

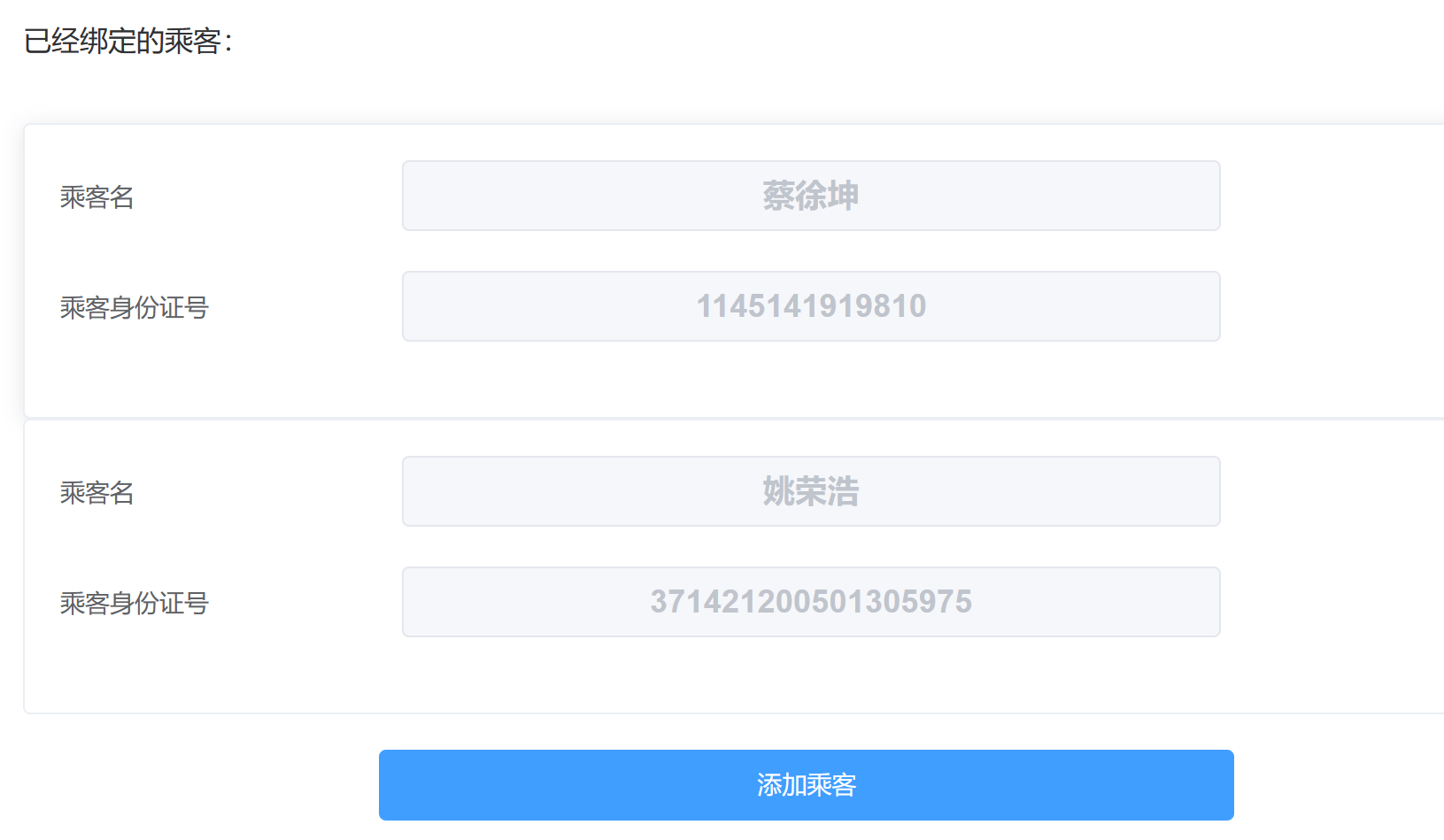


改签时用户可以选择同一个航班的不同调度时间，然后根据规则退款，并提交一个新调度的订单。退款时用户将直接按照退款规则退款，并取消订单。

3．查看并修改个人信息



同时可以查看并修改自己已经绑定的乘客。



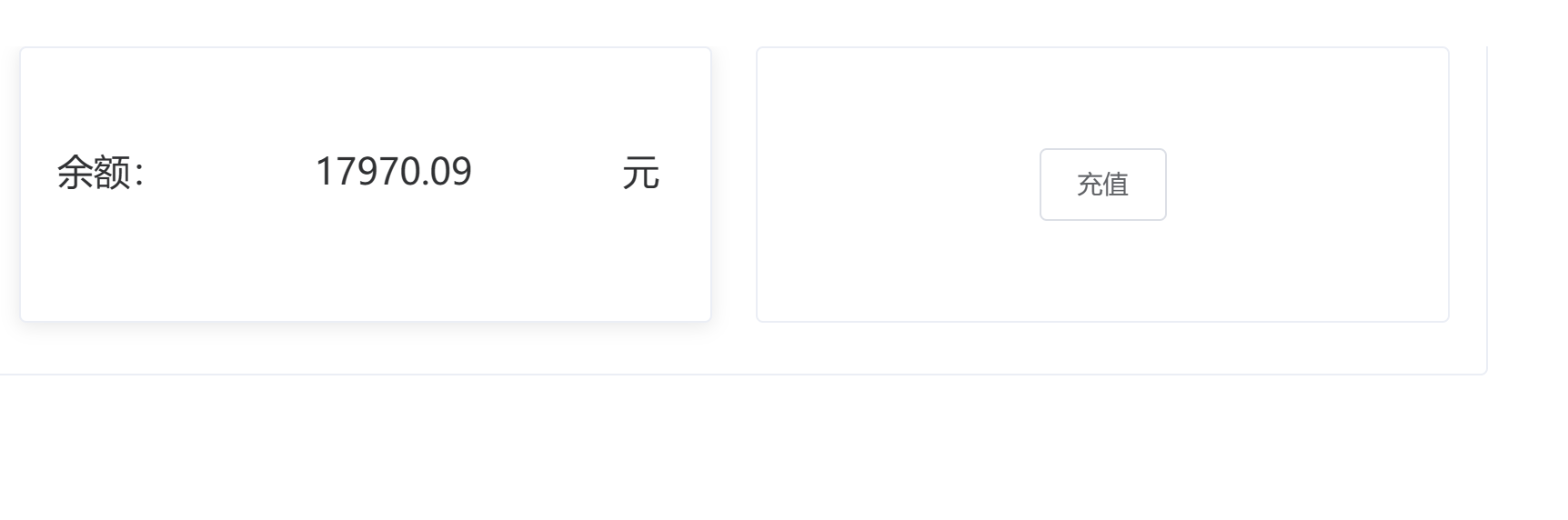
1. 查看消息

用户可以在消息一栏查看自己收到的消息，包括航班取消消息、系统通知等等。



6其他操作

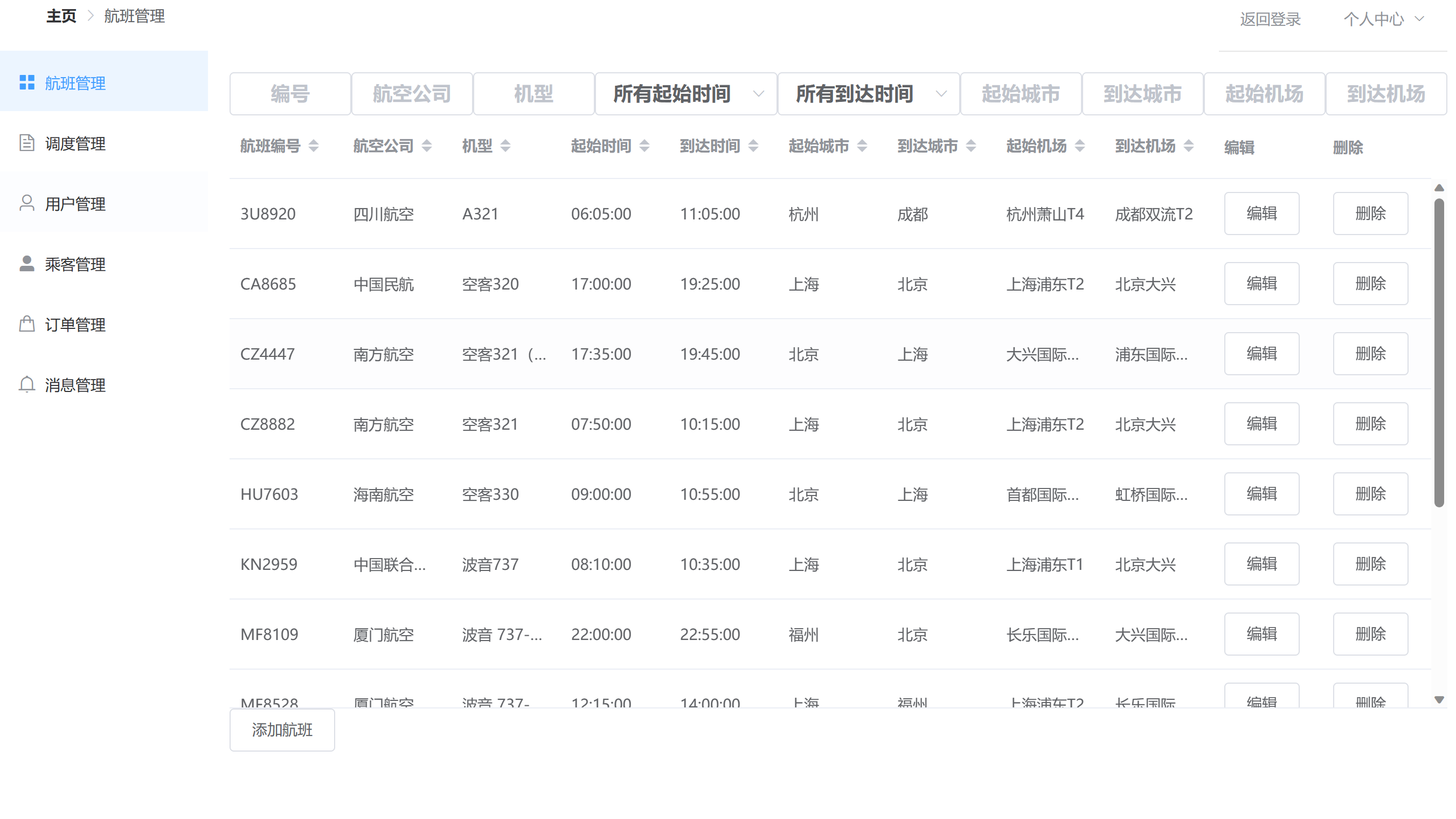
用户还可以进行修改密码，查看余额并充值等操作。





### 4.2.6 管理员页面

管理员具有自己独有的页面与权限，负责对系统中的数据进行各种操作，因为系统中的数据事实上存储于各个数据库表中，所以管理员对数据进行操作实际上就是对数据库表进行增删改查操作。



在管理员页面中，我们为管理员提供了各种管理接口，包括：航班管理、调度管理、用户管理、乘客管理、订单管理、消息管理。这些管理操作事实上对应于数据库的各个表，管理员可以在前端用可视化的方式对数据库表进行增删改查。

管理员对表也不能完全随意的操作。因为我们建表时对部分表与表的数据之间添加了外键约束，所以管理员无法在保留有外键时删除某一列。为了系统的安全性，管理员必须手动删除存在约束的内容。

删除调度操作来说则是一个特例，因为删除一个调度本质上就是将一个调度取消，从现实中来说就是某天某次航班取消飞行任务了，这种操作是很常见的，而且管理员也不能一个一个的去删除买了这个调度的订单。于是删除调度会先删除所有的选了这个调度的订单，并给对应用户发一条航班取消的消息，以及为用户退款。

具体方法如下：



值得注意的是，因为这些操作事实上是要按顺序依次且同时完成的，如果在并行操作时有两个用户操作同时进入该方法，很有可能会导致数据库错误，因此我们要使用关键字synchrionzed给这个方法加锁。

# 五、系统测试

向数据库插入若干航班和调度后，我们就可以让用户进行航班搜索等各种操作。



对于航班中转操作，我对数据库进行一次中转的等值连接就可以拿出存在中转的航班。如下：



对于其他操作，前端可以通过访问后端接口进行测试，这些操作都定义在了可视化页面里面。值得注意的是，因为后端接口存在权限检验，所以我们如果不附带管理员的token的话是无法调用后端的管理员权限接口的。

# 六、总结

该项目作为飞机票管理系统，我们要实现飞机票售票的基本逻辑，要考虑到管理员和用户各自的权限，要能够对数据库进行合理的选择和设计。

该项目前端使用vue2进行构建，vue2作为一个非常好用的js框架，简化了html、css、js这1传统三件套的项目构建，可以让我们更为快速的设计出自己想要的页面。

后端使用springboot框架，使用JPA进行数据库联系，让我们可以使用原生sql语言对数据库进行操作。同时，将表对象化的一一映射关系使我们进行逻辑设计时更加方便。

本次项目其实有一个缺陷，在我选择框架设计数据库时，我还没有深入了解关系型数据库和非关系型数据库的区别，事实上，对于这样一种高并发的项目设计，使用非关系数据库是更加合理的。在持久层使用redis应该是更加合理的选择。