铁路售票系统项目介绍

陈治杰曾灵杰刘子鸣

2025 年 12 月 16 日

项目需求

本项目旨在构建一个功能完善的铁路售票系统（类似 12306），支持用户购票、退票、改签以及管理员对车次、车站、时刻表的管理。

核心功能模块

- **用户模块**：用户注册、登录、个人信息管理。
- **票务模块**：车票查询、预订、支付、退票、改签。
- **车次管理**：列车信息管理、车厢管理、座位管理。
- **时刻表管理**：列车时刻表制定、经停站管理。
- **运营管理**：车站管理、员工管理、检票（进站/出站）。
- **交易平台**：二手票交易或票务转让功能（`ticket_listing`）。

技术栈

- **后端**: Java 21, Spring Boot 3.3.5, Spring Data JPA, Spring Security, MySQL.
- **前端**: Vue.js, TypeScript, Vite.
- **数据库**: MySQL 8.0+.

数据库设计 - 基础数据表

系统采用关系型数据库 MySQL，主要包含以下核心数据表：

- **station (车站表)**：存储车站的基本信息（代码、名称、城市）。
- **train (列车表)**：存储列车的基本信息（车次号、列车类型）。
- **coach (车厢表)**：定义列车的车厢组成、类型及座位数。
- **seat (座位表)**：定义车厢内的具体座位信息。

数据库设计 - 调度与时刻表

- **train_schedule (列车时刻表)**：定义某次列车在具体日期的发车计划及状态。
- **schedule_stop (经停站表)**：定义列车时刻表的具体停靠站点、到达及出发时间。

数据库设计 - 订单与交易

- **user_account (用户表)**：存储用户信息及角色。
- **booking_order (订单表)**：记录用户的购票订单及状态（待支付、已支付、已取消等）。
- **seat_allocation (座位分配表)**：管理具体时刻表的座位占用情况，防止超卖。
- **ticket (车票表)**：生成的实际乘车凭证，关联订单、座位及乘客。
- **payment (支付表)**：记录支付流水。
- **ticket_listing (票务挂牌表)**：用于票务转让或二手交易。

数据库设计 - 运营与检票

- **employee (员工表)**：车站工作人员信息。
- **checkin (检票记录表)**：记录乘客进出站的检票日志。

数据库模式 (Database Schema) - 基础信息

```sql
CREATE TABLE station (
  station_id INT PRIMARY KEY AUTO_INCREMENT,
  code VARCHAR(16) NOT NULL UNIQUE,
  name VARCHAR(128) NOT NULL,
  city VARCHAR(128) NOT NULL
);

CREATE TABLE train (
  train_id INT PRIMARY KEY AUTO_INCREMENT,
  train_no VARCHAR(32) NOT NULL UNIQUE,
  train_type VARCHAR(32) NOT NULL
);
```

数据库模式 (Database Schema) - 时刻表

```sql
CREATE TABLE train_schedule (
  schedule_id INT PRIMARY KEY AUTO_INCREMENT,
  train_id INT NOT NULL,
  depart_date DATE NOT NULL,
  status ENUM('PLANNED', 'OPEN', 'DEPARTED', 'ARRIVED', '
    CANCELLED') NOT NULL DEFAULT 'PLANNED',
  CONSTRAINT fk_schedule_train FOREIGN KEY (train_id)
    REFERENCES train (train_id)
);
```

接口文档

后端提供 RESTful API 供前端调用，主要接口如下：

认证模块 (Auth)

- `POST /api/auth/login`: 用户登录。
- `POST /api/auth/register`: 用户注册。

预订模块 (Booking)

- `GET /api/bookings/user/{userId}`: 获取指定用户的订单列表。
- `POST /api/bookings/reserve`: 发起座位预订请求，创建订单。
- `POST /api/bookings/{id}/cancel`: 取消指定订单。

接口文档 (续)

车次与时刻表 (Schedule & Train)

- `GET /api/schedules/search`: 根据出发地、目的地和日期查询车次。
- `GET /api/trains/{id}`: 获取列车详情。

车站管理 (Station)

- `GET /api/stations`: 获取所有车站列表。
- `POST /api/stations`: 新增车站（管理员）。

订单服务 (BookingService)

BookingService 负责处理订单的核心业务逻辑。

```java
@Service
@RequiredArgsConstructor
@Transactional
public class BookingService {
    private final BookingOrderRepository bookingOrderRepository
    ;

    public BookingOrder createOrder(BookingOrder order) {
        order.setStatus(OrderStatus.PENDING);
        return bookingOrderRepository.save(order);
    }

    public void cancelOrder(Integer orderId) {
        // ... implementation details
    }
}
```

订单控制器 (BookingController)

```java
@RestController
@RequestMapping("/api/bookings")
@RequiredArgsConstructor
public class BookingController {
    private final BookingService bookingService;

    @PostMapping("/reserve")
    public ResponseEntity<BookingOrder> reserveSeats(@Valid
    @RequestBody ReserveSeatsRequest request) {
        BookingOrder order = new BookingOrder();
        return ResponseEntity.ok(bookingService.createOrder(
    order));
    }
}
```

时刻表服务 (ScheduleService)

```java
@Service
public class ScheduleService {
    public void createSchedule(CreateScheduleRequest request) {
        // Create Schedule
        TrainSchedule schedule = new TrainSchedule();
        // ...
        trainScheduleRepository.save(schedule);

        // Create Stops, Coaches, Seats
        // ...
    }
}
```

票务交易服务 (TradingService)

```java
@Service
public class TradingService {
    @Transactional
    public void createListing(Integer userId, Integer orderId,
    BigDecimal price) {
        // Check ownership and status
        // Create listing
    }

    @Transactional
    public void buyListing(Integer buyerId, Integer listingId)
    {
        // Transfer ownership
        // Update ticket passenger name
    }
}
```

选座与预订页面 (Booking.vue)

```
// Load seats and group by coach
const seatsByCoach = computed(() => {
  const groups: Record<string, SeatAllocation[]> = {}
  allAllocations.value.forEach(a => {
    const coachNo = a.seat.coach.coachNo
    if (!groups[coachNo]) {
      groups[coachNo] = []
    }
    groups[coachNo].push(a)
  })
  return groups
})
```

交易平台页面 (TradingPlatform.vue)

```
const handleBuy = async (listing: TicketListing) => {
  if (!userStore.user) {
    ElMessage.warning('Please login first');
    return;
  }
  // Confirm and buy logic
  await buyListing(listing.listingId, userStore.user.id);
};
```

谢谢观看！

Q & A