# 互联网购票乘车系统软件体系结构描述文档

| 修改人员 | 日期 | 变更原因 | 版本号 |
| --- | --- | --- | --- |
| 陈泓雨、金语盼、王骏佳、刘子恒 | 2023.5.30 | 初稿（大概框架） | V1.0 |
| 陈泓雨、金语盼、王骏佳、刘子恒 | 2023.6.2 | 完成接口 | V2.0 |
| 陈泓雨、金语盼、王骏佳、刘子恒 | 2023.6.4 | 修改、完善接口 | V2.1 |
| 金语盼 | 2023.6.30 | 根据框架代码修改 | V3.0 |

分工：陈泓雨：1-4.3
金语盼：5.1-5.2 + 6
王骏佳：5.3
刘子恒：5.

# 1. 引言

## 1.1 编制目的

本报告详细完成对互联网购票乘车系统的概要设计，达到详细设计和开发的目的，同时实现和测试人员及用户的沟通。 本报告面向开发人员、测试人员及最终用户而编写，是了解系统的导航。

## 1.2 词汇表

| 词汇名称 | 词汇含义 | 备注 |
| --- | --- | --- |
| ITS | 互联网购票乘车系统 | |

## 1.3 参考资料

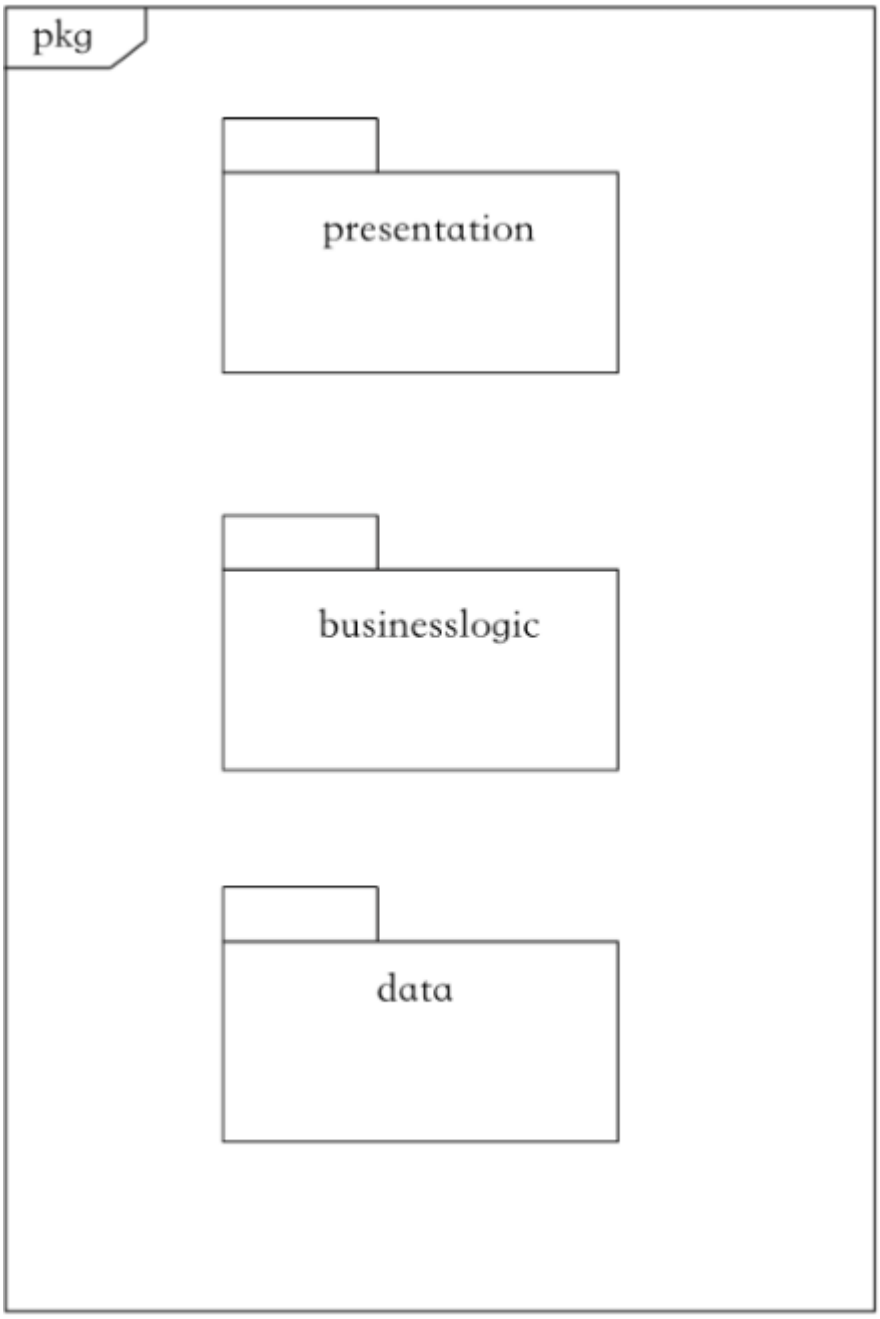
i. [骆斌2012]骆斌，丁二玉，刘钦，软件工程与计算（卷二）：软件开发的技术基础，2012

# 2. 产品概述

参考互联网乘车购票系统用例文档和互联网乘车购票系统软件需求规格说明文档中对产品的概括描述。

# 3. 逻辑视角

互联网乘车购票系统中，选择了分层体系结构风格，将系统分为3层（展示层、业务逻辑、数据层）能够很好地示意整个高层抽象。展示层包含GUI页面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。分层体系结构的逻辑视角和逻辑设计方法如图1和图2所示。


pkg
presentation
businesslogic
data

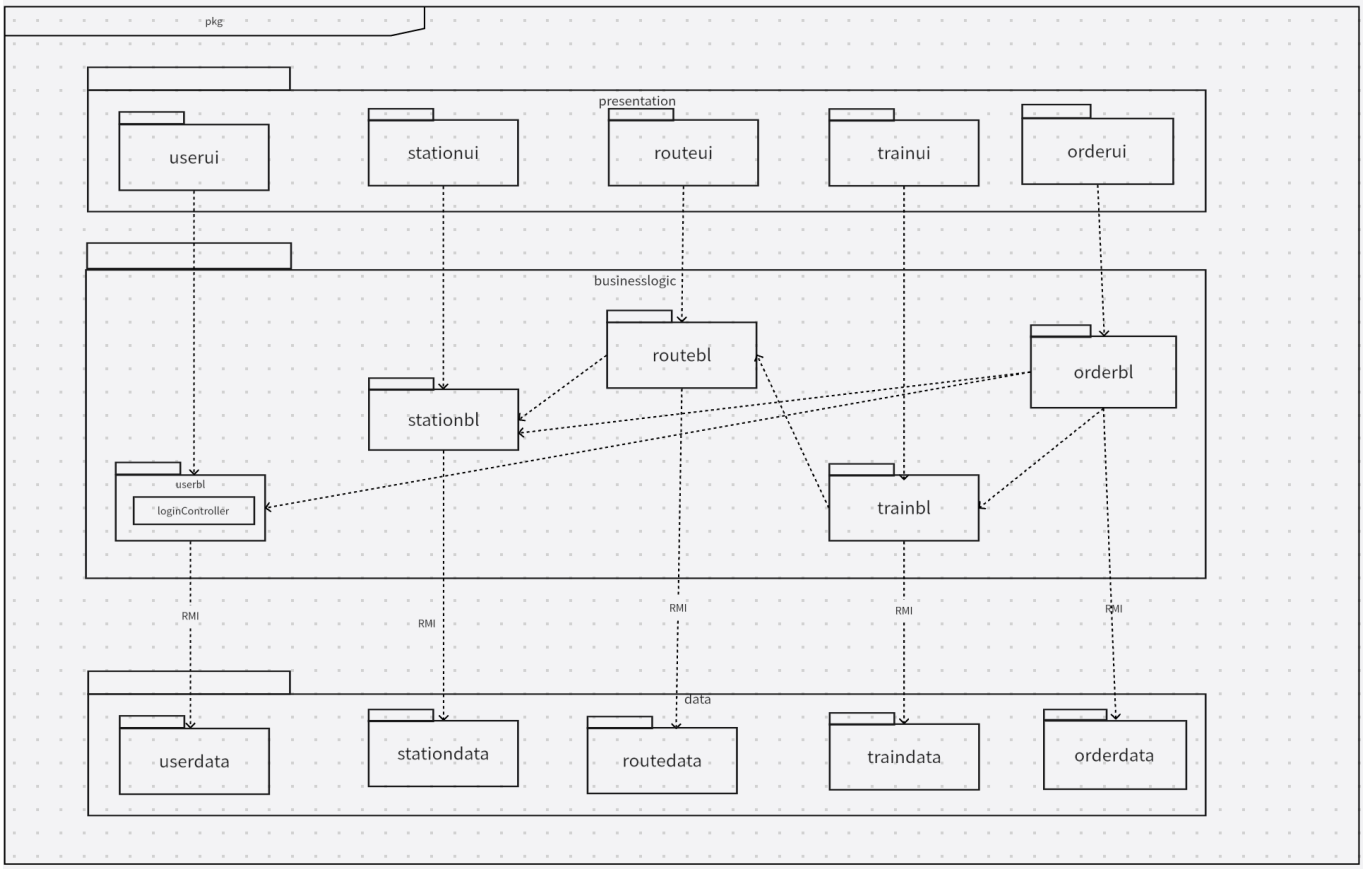



图2 软件体系结构逻辑设计方案

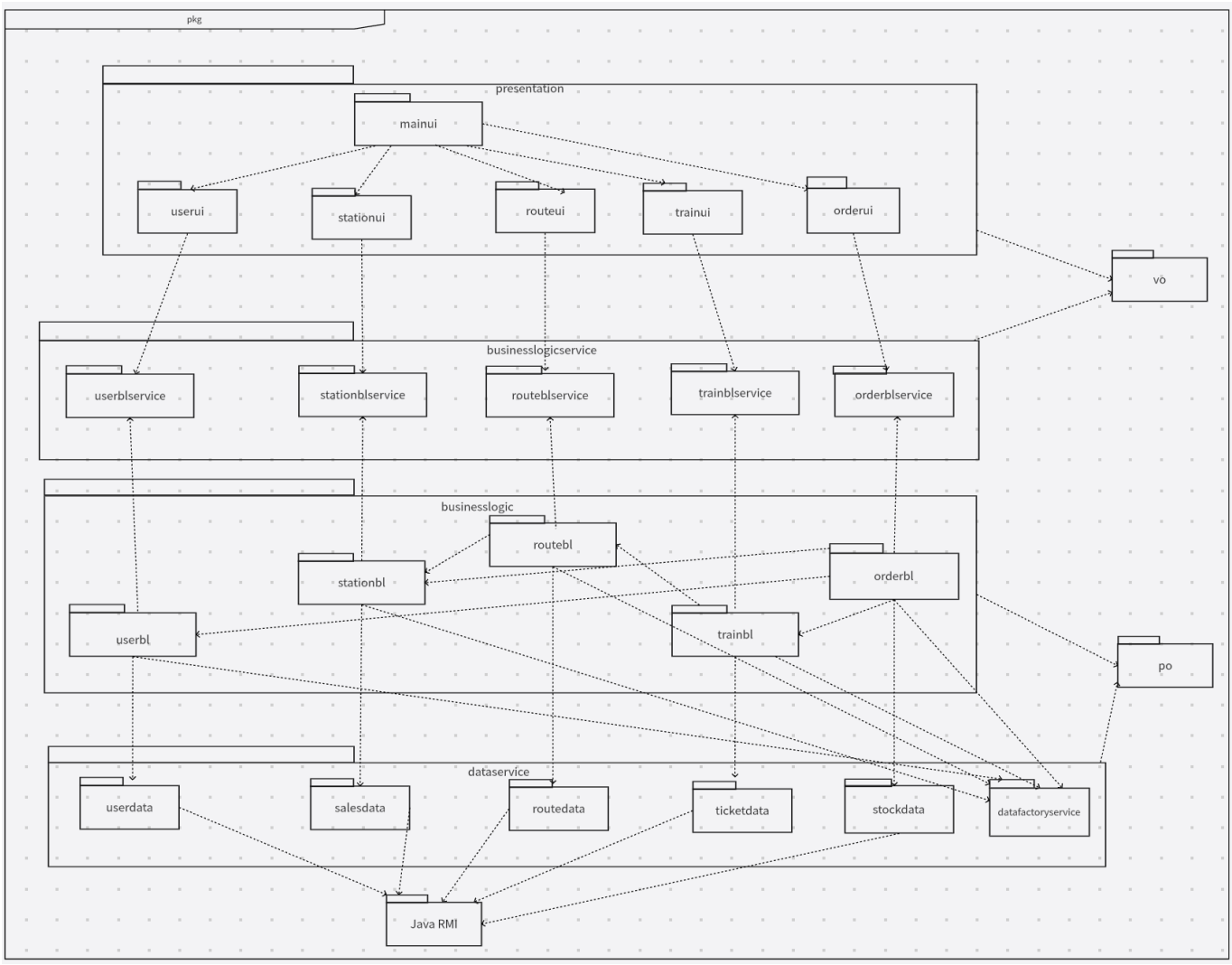
# 4. 组合视角

## 4.1 开发包图

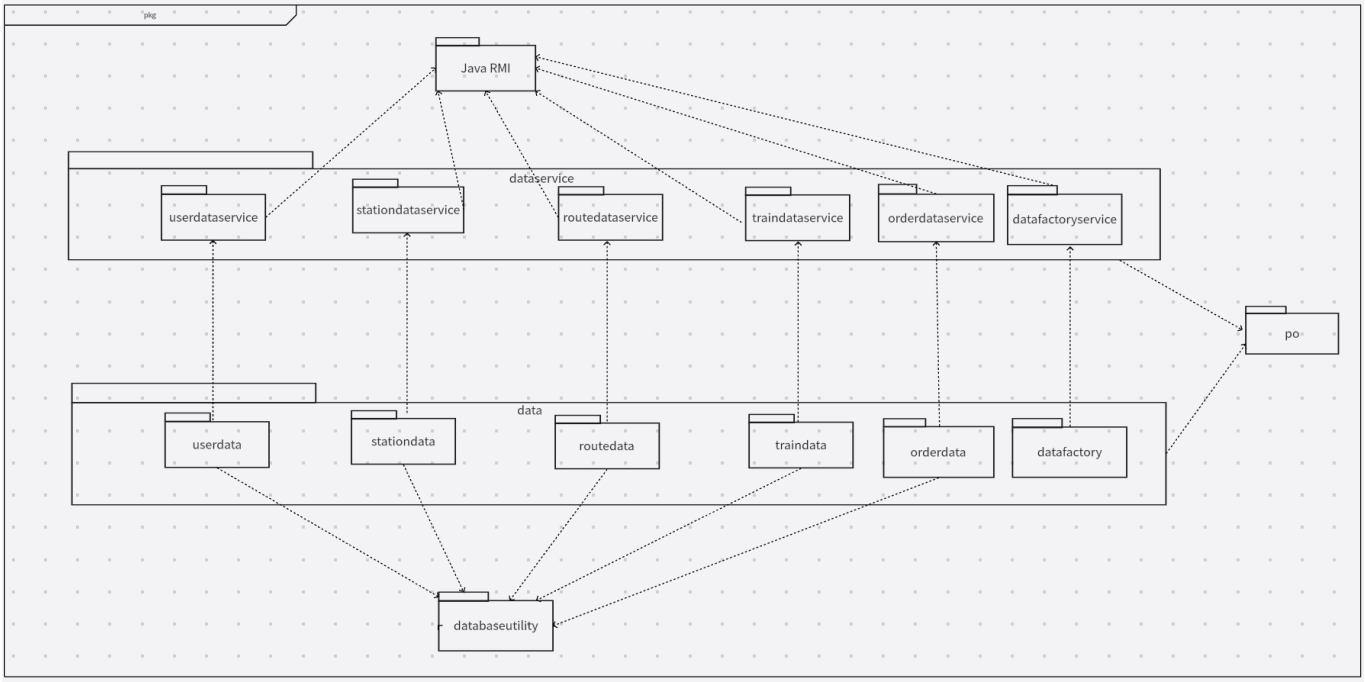
互联网购票乘车系统的最终开发包图设计如下表所示。

| 开发（物理）包 | 依赖的其他开发包 |
| --- | --- |
| mainui | userui, stationui, routeui, trainui, orderui |
| userui | userblservice, 界面类库包, vo |
| userblservice | |
| userbl | userblservice, userdataservice,po |
| userdataservice | Java RMI, po |
| userdata | databaseutility, po, userdataservice |
| stationui | stationblservice, 界面类库包, vo |
| stationblservice | |
| stationbl | stationblservice, stationdataservice, po |
| stationdataservice | Java RMI, po |
| stationdata | databaseutility, po, stationdataservice |
| routeui | routeblservice, 界面类库包, vo |
| routeblservice | |
| routebl | routeblservice, routedataservice, po, stationbl |
| routedataservice | Java RMI, po |
| routedata | databaseutility, po, routedataservice |
| trainui | trainblservice, 界面类库包, vo |
| trainblservice | |
| trainbl | trainblservice, traindataservice, po, routebl |
| traindataservice | Java RMI, po |
| traindata | databaseutility, po, traindataservice |
| orderui | orderblservice, 界面类库包, vo |
| orderblservice | |
| orderbl | orderblservice, orderdataservice, po, userbl, trainbl, stationbl |
| orderdataservice | Java RMI, po |
| orderdata | databaseutility, po, orderdataservice |
| vo | |
| po | |
| utilitybl | |
| 界面类库包 | |
| Java RMI | |
| databaseutility | JDBC |

互联网购票乘车系统浏览器端开发包图如下图所示，服务器端开发包图如下下图所示。



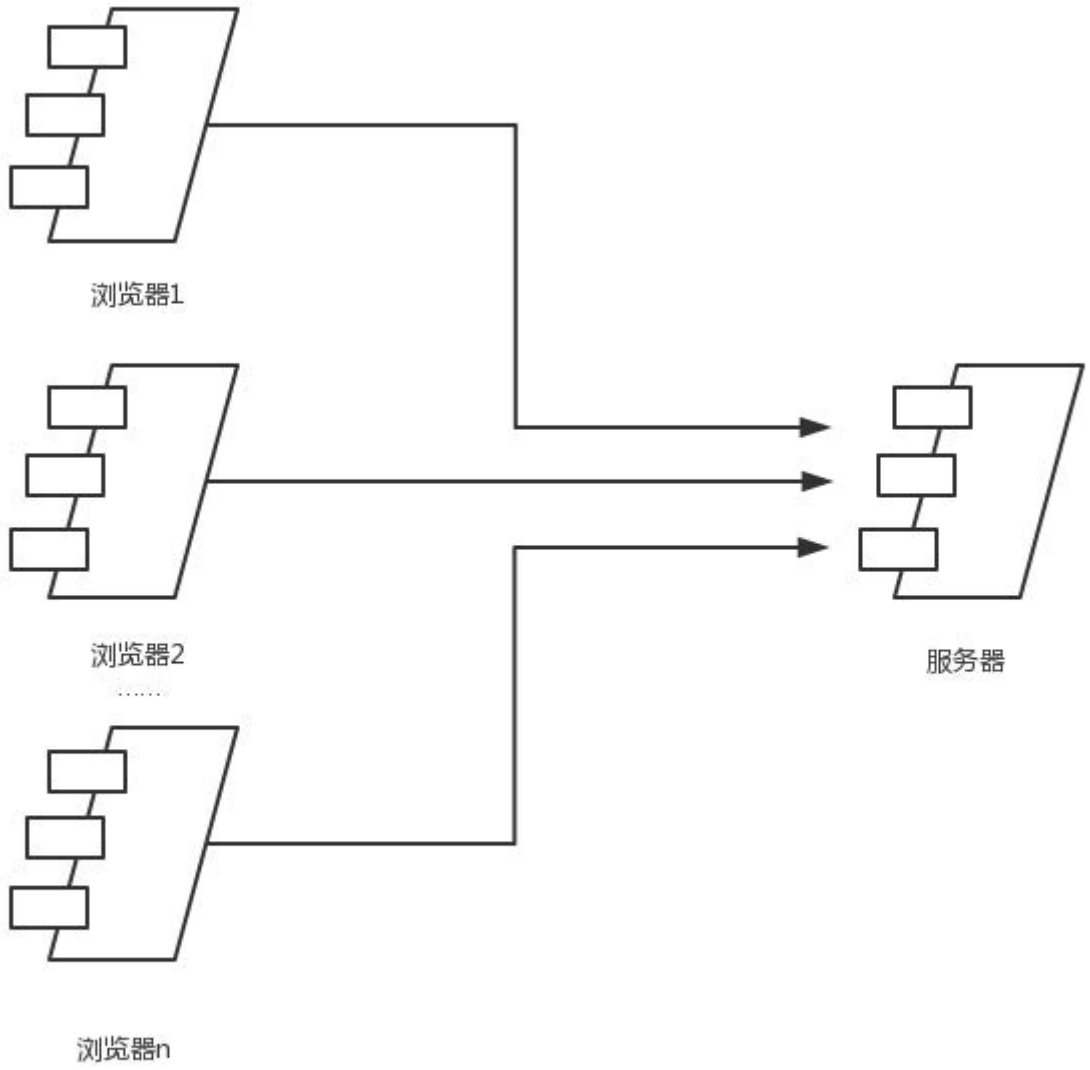

互联网购票乘车系统客户端开发包图




互联网购票乘车系统服务器端开发包图

## 4.2 运行时进程

在互联网购票乘车系统中，会有多个浏览器端进程和一个服务器端进程，其进程图如图5所示。结合部署图，浏览器端进程是在用户浏览器上运行，服务器端进程是在服务器机器上运行。

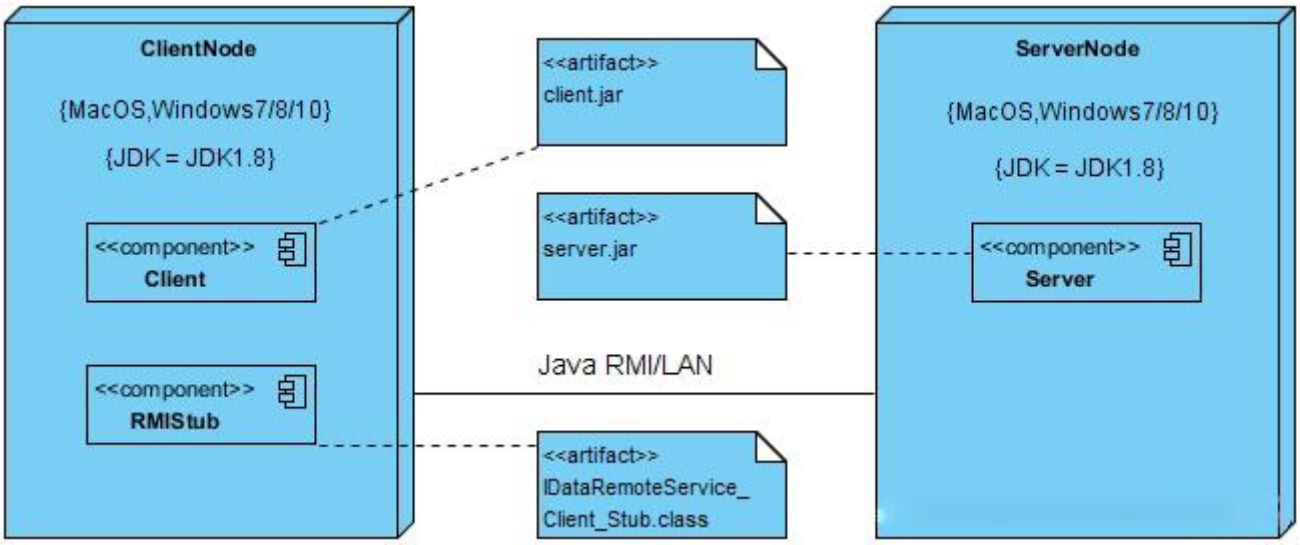



进程图

## 4.3 物理部署

互联网购票乘车系统中客户端构件是放在客户端机器上，服务器端构件是放在服务 器端机器上。在客户端节点上，还要部署 RMIStub 构件。由于 Java RMI 构件属于 JDK 1.8 的一部分。所以，在系统 JDK 环境已经设置好

的情况下，不需要再独立部署。部署图如图所示。




部署图

# 5. 接口视角

## 5.1 模块的职责

浏览器端模块和服务器端模块视图分别如图5.1.1和图5.1.2所示。浏览器各层和服务器端各层的职责分别如表5.1.1和表5.1.2所示。


**图5.1.1 浏览器端模块视图**

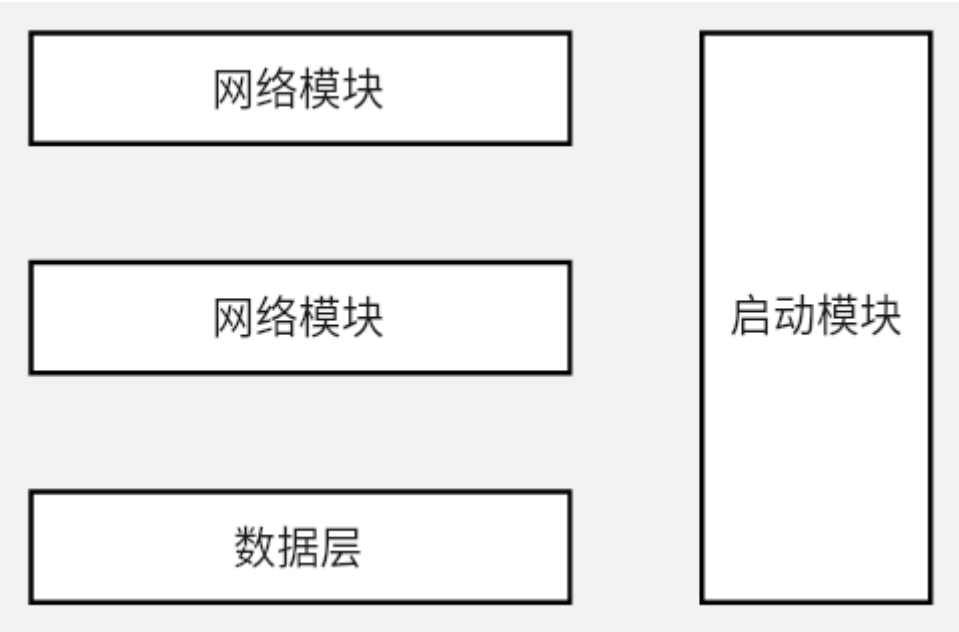



**图5.1.2 服务器端模块视图**

**表5.1.1 浏览器端各层的职责**

| 层 | 职责 |
| --- | --- |
| 展示层 | 基于网络的互联网购票乘车系统浏览器端用户界面 |
| 浏览器端网络模块 | 利用HTTP协议访问服务器端服务 |

**表5.1.2 服务器端各层的职责**

| 层 | 职责 |
| --- | --- |
| 启动模块 | 初始化数据库 |
| 服务器端网络模块 | 利用HTTP协议接受浏览器端数据以及返回数据 |
| 业务逻辑模块 | 对浏览器端的请求进行响应并执行业务处理逻辑 |
| 数据层 | 负责数据的持久化及数据访问接口 |

每一层只是使用下方直接接触的层。层与层之间仅仅是通过接口的调用来完成的。层之间调用的接口如表5.1.3所示。

**表5.1.3 层之间调用的接口**

| 接口 | 服务调用方 | 服务提供方 |
| --- | --- | --- |

| | | |
|---|---|---|
| UserBLService<br>StationBLService<br>RouteBLService<br>TrainBLService<br>OrderBLService | 浏览器端展示层 | 服务器端业务逻辑层 |
| UserDataService<br>StationDataService<br>RouteDataService<br>TrainDataService<br>OrderDataService<br>DatabaseFactory | 服务器端业务逻辑层 | 服务器端数据层 |

## 5.2 用户界面层的分解

根据需求，系统存在22个用户界面：登录/注册界面、客户主界面、客户个人中心界面、车票列表界面、车票详情界面、购票界面、车票订单界面、购买预定车票界面、线上退票界面、会员开通界面、铁路管理员主界面、铁路管理员个人中心界面、车票信息管理界面、铁路信息管理界面、余票管理员主界面、余票管理员个人中心界面、余票管理界面、票务员主界面、票务员个人中心界面、售票界面、换票界面、线下退票界面。界面跳转如图5.2.1所示。

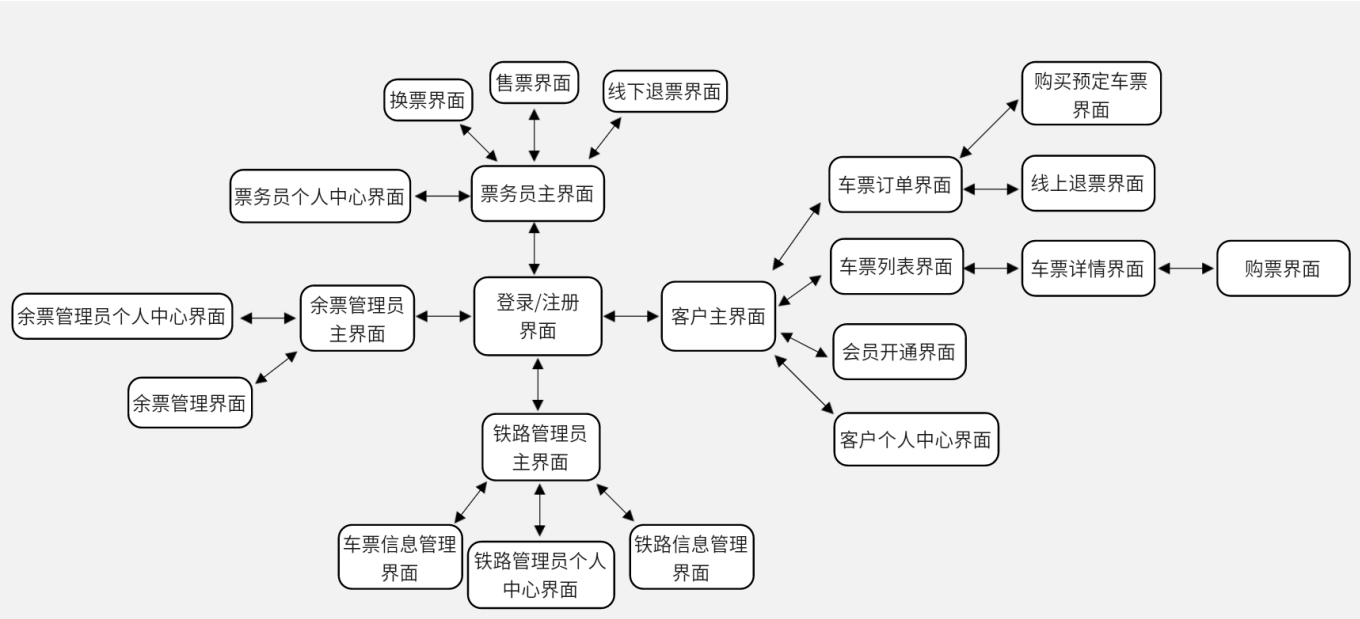



**表5.2.1 用户界面跳转**

### 5.2.1 用户界面层模块的职责

| 模块 | 职责 |
|---|---|
| mainui | 负责首页的显示和界面的跳转 |
| userui | 负责客户、管理员登录注册、个人信息、会员注册相关的显示 |
| stationui | 负责车站相关界面的显示 |

| | |
|---|---|
| routeui | 负责铁路线路相关界面的显示 |
| trainui | 负责车次相关界面的显示 |
| orderui | 负责订单相关界面的展示 |

### 5.2.2 用户界面层模块的接口规范

#### 5.2.2.1 mainui模块接口规范

| | | |
|---|---|---|
| MainUI | 语法 | init(args:String[]) |
| | 前置条件 | 无 |
| | 后置条件 | 显示首页以及登录、注册部分入口 |

#### 5.2.2.2 userui模块接口规范

| | | |
|---|---|---|
| UserUI.login | 语法 | public CommonResponse<?> login(String username, String password, String role) |
| | 前置条件 | 用户输入的登录信息合法 |
| | 后置条件 | 显示用户登录操作的结果如登录成功或错误提示 |
| UserUI.logout | 语法 | public CommonResponse<?> logout() |
| | 前置条件 | 用户已登录 |
| | 后置条件 | 跳转到首页 |
| UserUI.register | 语法 | public CommonResponse<?> register(String username, String password, String name, String idn, String phone, String type, String role) |
| | 前置条件 | 用户输入的个人信息和密码合法 |
| | 后置条件 | 显示用户注册操作的结果如注册成功或错误提示 |
| UserUI.userInfo | 语法 | public CommonResponse<UserVO> userInfo() |
| | 前置条件 | 用户已登录 |
| | 后置条件 | 显示用户基本信息 |
| UserUI.editInfo | 语法 | public CommonResponse<?> editInfo(String name, String idn, String phone, String type) |
| | 前置条件 | 用户输入的个人信息合法 |
| | 后置条件 | 提示修改信息成功 |
| UserUI.registerMember | 语法 | public CommonResponse<?> registerMember(String vippassword) |
| | 前置条件 | 用户输入的交易密码合法 |
| | 后置条件 | 提示注册会员成功 |

| 服务名 | 服务 |
|---|---|
| UserBLService.login | 登录界面的业务逻辑接口 |
| UserBLService.register | 注册界面的业务逻辑接口 |
| UserBLService.findByUsername | 展示用户基本信息界面的业务逻辑接口 |
| UserBLService.editInfo | 修改用户基本信息界面的业务逻辑接口 |
| UserBLService.getMembership | 注册会员界面的业务逻辑接口 |

#### 5.2.2.3 stationui模块接口规范

| | | |
|---|---|---|
| StationUI.listStations | 语法 | public CommonResponse<List<StationVO>> listStations() |
| | 前置条件 | 无 |
| | 后置条件 | 显示所有车站 |
| StationUI.getStation | 语法 | public CommonResponse<StationVO> getStation(Long stationId) |
| | 前置条件 | 无 |
| | 后置条件 | 显示所查询车站 |
| StationUI.addStation | 语法 | public CommonResponse<?> addStation(String name) |
| | 前置条件 | 用户输入车站名字合法 |
| | 后置条件 | 提示添加车站成功 |
| StationUI.editStation | 语法 | public CommonResponse<?> editStation(Long stationId, String name) |
| | 前置条件 | 用户输入车站名字合法 |
| | 后置条件 | 提示修改车站成功 |
| StationUI.deleteStation | 语法 | public CommonResponse<?> deleteStation(Long stationId) |
| | 前置条件 | 无 |
| | 后置条件 | 提示删除车站成功 |

| 服务名 | 服务 |
|---|---|
| StationBLService.listStations | 展示所有车站的业务逻辑接口 |
| StationBLService.getStation | 展示所查询车站的业务逻辑接口 |
| StationBLService.addStation | 添加车站的业务逻辑接口 |
| StationBLService.editStation | 修改车站的业务逻辑接口 |
| StationBLService.deleteStation | 删除车站的业务逻辑接口 |

#### 5.2.2.4 routeui模块接口规范

| | | |
|---|---|---|
| RouteUI.addRoute | 语法 | public CommonResponse<?> addRoute(String name, List<Long> stationIds) |
| | 前置条件 | 管理员输入信息合法 |
| | 后置条件 | 提示增加路线成功 |
| RouteUI.getRoutes | 语法 | public CommonResponse<List<RouteVO>> getRoutes() |
| | 前置条件 | 无 |
| | 后置条件 | 显示所有路线 |
| RouteUI.getRoute | 语法 | public CommonResponse<RouteVO> getRoute(Long routeId) |
| | 前置条件 | 无 |
| | 后置条件 | 显示所查询路线 |
| RouteUI.editRoute | 语法 | public CommonResponse<?> editRoute(Long routeId, String name, List<Long> stationIds) |
| | 前置条件 | 管理员输入信息合法 |
| | 后置条件 | 提示修改路线成功 |
| RouteUI.deleteRoute | 语法 | public CommonResponse<?> deleteRoute(Long routeId) |
| | 前置条件 | 无 |
| | 后置条件 | 提示删除路线成功 |

| 服务名 | 服务 |
|---|---|
| RouteBLService.addRoute | 管理员增加路线界面的业务逻辑接口 |
| RouteBLService.getRoutes | 展示所有路线界面的业务逻辑接口 |
| RouteBLService.getRoute | 管理员查询路线界面的业务逻辑接口 |
| RouteBLService.editRoute | 管理员修改路线界面的业务逻辑接口 |
| RouteBLService.deleteRoute | 管理员删除路线界面的业务逻辑接口 |

#### 5.2.2.5 trainui模块接口规范

| | | |
|---|---|---|
| TrainUI.listTrains | 语法 | public CommonResponse<List<TrainVO>> listTrains(Long startStationId, Long endStationId, String date) |
| | 前置条件 | 无 |
| | 后置条件 | 显示符合条件的所有车次 |
| TrainUI.getTrain | 语法 | public CommonResponse<TrainDetailVO> getTrain(Long trainId) |
| | 前置条件 | 无 |
| | 后置条件 | 显示该车次的详细信息 |
| TrainUI.addTrain | 语法 | public CommonResponse<?> addTrain(String name, Long routeId, TrainType trainType, String date, List<Date> arrivalTimes, List<Date> departureTimes) |
| | 前置条件 | 管理员输入信息合法 |
| | 后置条件 | 提示添加车次成功 |
| TrainUI.listTrainsAdmin | 语法 | public CommonResponse<List<AdminTrainVO>> listTrainsAdmin() |
| | 前置条件 | 无 |
| | 后置条件 | 显示所有车次 |
| TrainUI.changeTrain | 语法 | public CommonResponse<?> changeTrain(Long trainId, String name, Long routeId, TrainType trainType, String date, List<Date> arrivalTimes, List<Date> departureTimes) |
| | 前置条件 | 管理员输入信息合法 |
| | 后置条件 | 提示修改车次信息成功 |
| TrainUI.deleteTrain | 语法 | public CommonResponse<?> deleteTrain(Long trainId) |
| | 前置条件 | 无 |
| | 后置条件 | 提示删除车次成功 |

| 服务名 | 服务 |
|---|---|
| TrainBLService.listTrains | 用户查询车次界面的业务逻辑接口 |
| TrainBLService.getTrain | 展示车次详细信息界面的业务逻辑接口 |
| TrainBLService.addTrain | 管理员增加车次的业务逻辑接口 |
| TrainBLService.listTrainsAdmin | 管理员显示所有车次的业务逻辑接口 |
| TrainBLService.changeTrain | 管理员修改车次信息界面的业务逻辑接口 |
| TrainBLService.deleteTrain | 管理员删除车次界面的业务逻辑接口 |

#### 5.2.2.6 orderui模块接口规范

| | | |
|---|---|---|
| OrderUI.createOrder | 语法 | public CommonResponse<OrderIdVO> createOrder(String name, String idn, String phone, String type, Long trainId, Long startStationId, Long endStationId, String seatType, Integer price) |
| | 前置条件 | 用户已登录 |
| | 后置条件 | 显示创建的订单信息 |
| OrderUI.listOrders | 语法 | public CommonResponse<List<OrderVO>> listOrders() |
| | 前置条件 | 用户已登录 |
| | 后置条件 | 展示该用户的所有订单 |
| OrderUI.getOrder | 语法 | public CommonResponse<OrderVO> getOrder(Long orderId) |
| | 前置条件 | 无 |
| | 后置条件 | 显示详细订单信息 |
| OrderUI.patchOrder | 语法 | public CommonResponse<String> patchOrder(Long orderId, OrderStatus status, String type) |
| | 前置条件 | 订单状态合法 |
| | 后置条件 | 取消订单、退票或支付订单成功 |
| OrderUI.usePoints | 语法 | public CommonResponse<Integer> usePoints(Long orderId) |
| | 前置条件 | 用户选择使用积分 |
| | 后置条件 | 展示使用积分后的订单价格 |
| OrderUI.cancelUsePoints | 语法 | public CommonResponse<Integer> cancelUsePoints(Long orderId) |
| | 前置条件 | 用户取消使用积分 |
| | 后置条件 | 展示不使用积分后的订单价格 |
| OrderUI.checkIllegal | 语法 | public CommonResponse<Boolean> checkIllegal(Long orderId) |
| | 前置条件 | 用户取消或完成一个订单 |
| | 后置条件 | 展示用户是否存在恶意购票行为 |

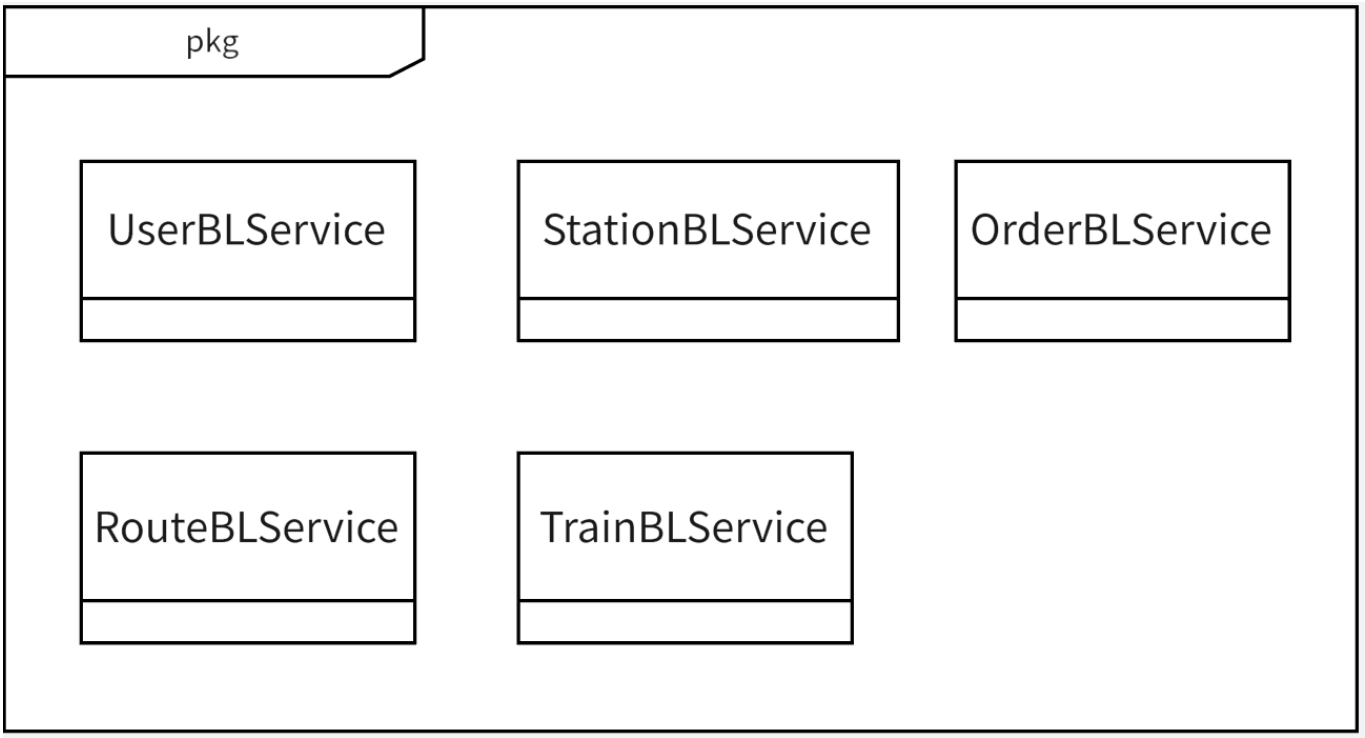
| 服务名 | 服务 |
|---|---|
| OrderBLService.createOrder | 用户创建订单的业务逻辑接口 |
| OrderBLService.listOrders | 展示用户所有订单界面的业务逻辑接口 |
| OrderBLService.getOrder | 展示订单详细信息界面的业务逻辑接口 |
| OrderBLService.cancelOrder | 取消订单界面的业务逻辑接口 |
| OrderBLService.payOrder | 支付订单界面的业务逻辑接口 |
| OrderBLService.usePoints | 用户使用积分的业务逻辑接口 |
| OrderBLService.cancelUsePoints | 用户取消使用积分的业务逻辑接口 |
| OrderBLService.checkIllegal | 检查用户是否存在恶意购票的业务逻辑接口 |

#### 5.2.3 用户界面模块设计原理

用户界面利用Vue.js框架实现

## 5.3 业务逻辑层的分解

业务逻辑层分解如图所示:




### 5.3.1 业务逻辑层模块的职责

| 模块 | 职责 |
| --- | --- |
| UserBL | 负责用户注册界面的业务<br>负责用户登录页界面的业务<br>负责用户查看个人信息的业务<br>负责用户注册会员的业务 |
| StationBL | 负责车站管理的业务 |
| RouteBL | 负责铁路线路信息管理的业务 |
| TrainBL | 负责车次管理的业务 |
| OrderBL | 负责订单信息的管理业务 |

### 5.3.2 业务逻辑层模块的接口规范

| 提供的服务（供接口） | | |
|---|---|---|
| UserBLService.login | 语法 | public void login(String username, String password, String role) |
| | 前置条件 | 用户名密码合法 |
| | 后置条件 | 更新用户登录状态或抛出异常 |
| UserBLService.register | 语法 | public void register(String username, String password, String name, String idn, String phone, String type, String role) |
| | 前置条件 | 信息合法 |
| | 后置条件 | 增加一个用户对象或抛出异常 |
| UserBLService.findByUsername | 语法 | public UserVO findByUserName(String username) |
| | 前置条件 | 无 |
| | 后置条件 | 返回 UserVO 对象 |
| UserBLService.editInfo | 语法 | public void editInfo(String username, String name, String idn, String phone, String type) |
| | 前置条件 | 信息合法 |
| | 后置条件 | 持久化更新该用户的信息 |
| UserBLService.getMembership | 语法 | public void getMembership(String username,String vippassword) |
| | 前置条件 | 用户交易密码合法且两次相同 |
| | 后置条件 | 用户成为会员，持久化更新会员身份 |

| 需要的服务（需接口） | |
|---|---|
| 服务名 | 服务 |
| UserDataService.findByUsername | 根据 username 从数据库中得到用户对象 |
| UserDataService.save | 向数据库中插入用户对象或更新用户信息 |

| 提供的服务（供接口） | | |
|---|---|---|
| StationBLService.listStations | 语法 | public List<StationVO> listStations() |
| | 前置条件 | 无 |
| | 后置条件 | 按名字升序返回所有 StationVO 对象 |
| StationBLService.getStation | 语法 | public StationVO getStation(Long stationId) |
| | 前置条件 | 无 |
| | 后置条件 | 返回对应的 StationVO 对象 |
| StationBLService.addStation | 语法 | public void addStation(String name) |
| | 前置条件 | 无 |
| | 后置条件 | 增加一个车站对象或抛出异常 |
| StationBLService.editStation | 语法 | public void editStation(Long id, String name) |
| | 前置条件 | 无 |
| | 后置条件 | 持久化更新车站信息或抛出异常 |
| StationBLService.deleteStation | 语法 | public void deleteStation(Long stationId) |
| | 前置条件 | 无 |
| | 后置条件 | 在数据库中删除车站对象及所影响的路线和车次对象 |
| 需要的服务（需接口） | | |

| 服务名 | 服务 |
| --- | --- |
| StationDataService.findById | 根据 id 从数据库中得到车站对象 |
| StationDataService.findAll | 从数据库中得到所有车站对象 |
| StationDataService.findByName | 根据 name 从数据库中得到车站对象 |
| StationDataService.save | 向数据库中插入车站对象或更新车站信息 |
| StationDataService.delete | 在数据库中删除车站对象 |
| RouteDataService.findAll | 从数据库中得到所有路线对象 |
| RouteDataService.delete | 在数据库中删除路线对象 |
| TrainDataService.findByRouteId | 根据 routeId 从数据库中得到车次对象 |
| TrainDataService.deleteAll | 在数据库中删除多个车次对象 |

| 提供的服务（供接口） | | |
| --- | --- | --- |
| RouteBLService.addRoute | 语法 | public void addRoute(String name, List<Long> stationIds) |
| | 前置条件 | 信息合法 |
| | 后置条件 | 增加一个路线对象 |
| RouteBLService.getRoutes | 语法 | public List<RouteVO> listRoutes() |
| | 前置条件 | 无 |
| | 后置条件 | 按名字升序返回所有 RouteVO 对象 |
| RouteBLService.getRoute | 语法 | public RouteVO getRoute(Long id) |
| | 前置条件 | 无 |
| | 后置条件 | 返回对应的 RouteVO 对象 |
| RouteBLService.editRoute | 语法 | public void editRoute(Long id, String name, List<Long> stationIds) |
| | 前置条件 | 信息合法 |
| | 后置条件 | 持久化更新路线信息 |
| RouteBLService.deleteRoute | 语法 | public void deleteRoute(Long routeId) |
| | 前置条件 | 无 |
| | 后置条件 | 在数据库中删除路线对象及所影响的车次对象 |

**需要的服务（需接口）**

| 服务名 | 服务 |
| --- | --- |
| RouteDataService.save | 向数据库中插入路线对象或更新路线信息 |
| RouteDataService.findAll | 从数据库中得到所有路线对象 |
| RouteDataService.findById | 根据 id 从数据库中得到路线对象 |
| RouteDataService.delete | 在数据库中删除路线对象 |
| TrainDataService.findByRouteId | 根据 routeId 从数据库中得到车次对象 |
| TrainDataService.deleteAll | 在数据库中删除多个车次对象 |

| 提供的服务（供接口） | | |
|---|---|---|
| TrainBLService.listTrains | 语法 | public List<TrainVO> listTrains(Long startStationId, Long endStationId, String date) |
| | 前置条件 | 无 |
| | 后置条件 | 返回符合查询条件的所有 TrainVO 对象 |
| TrainBLService.getTrain | 语法 | public TrainDetailVO getTrain(Long trainId) |
| | 前置条件 | 无 |
| | 后置条件 | 返回 TrainDetailVO 对象 |
| TrainBLService.addTrain | 语法 | public void addTrain(String name, Long routeId, TrainType type, String date, List<Date> arrivalTimes, List<Date> departureTimes) |
| | 前置条件 | 信息合法 |
| | 后置条件 | 增加一个车次对象或抛出异常 |
| TrainBLService.listTrainsAdmin | 语法 | public List<AdminTrainVO> listTrainsAdmin() |
| | 前置条件 | 无 |
| | 后置条件 | 按名字升序返回所有 AdminTrainVO 对象 |
| TrainBLService.changeTrain | 语法 | public void changeTrain(Long id, String name, Long routeId, TrainType type, String date, List<Date> arrivalTimes, List<Date> departureTimes) |
| | 前置条件 | 信息合法 |
| | 后置条件 | 持久化更新车次对象信息或抛出异常 |
| TrainBLService.deleteTrain | 语法 | public void deleteTrain(Long id) |
| | 前置条件 | 无 |
| | 后置条件 | 在数据库中删除车次对象 |

| 需要的服务（需接口） | |
|---|---|
| 服务名 | 服务 |
| TrainDataService.findById | 根据 Id 从数据库中得到车次对象 |
| TrainDataService.findByRouteIdAndDate | 根据 RouteId 和 Date 从数据库中得到车次对象 |
| TrainDataService.findAll | 从数据库中得到所有车次对象 |
| TrainDataService.save | 向数据库中插入车次对象或更新车次信息 |
| TrainDataService.deleteById | 在数据库中根据 id 删除车次对象 |
| RouteDataService.findAll | 从数据库中得到所有路线对象 |
| RouteDataService.findById | 根据 id 从数据库中得到路线对象 |

| 提供的服务（供接口） | | |
|---|---|---|
| OrderBLService.createOrder | 语法 | public Long createOrder(String username, Long trainId, Long fromStationId, Long toStationId, String seatType, Long seatNumber,Integer price) |
| | 前置条件 | 信息合法 |
| | 后置条件 | 返回订单 id |
| OrderBLService.listOrders | 语法 | public List<OrderVO> listOrders(String username) |
| | 前置条件 | 无 |
| | 后置条件 | 返回该用户的所有订单的 OrderVO 对象 |
| OrderBLService.getOrder | 语法 | public OrderVO getOrder(Long id) |
| | 前置条件 | 无 |
| | 后置条件 | 返回 OrderVO 对象 |
| OrderBLService.cancelOrder | 语法 | public void cancelOrder(Long id) |
| | 前置条件 | 无 |
| | 后置条件 | 取消订单，订单状态更新 |
| OrderBLService.payOrder | 语法 | public String payOrder(Long id,String type) |
| | 前置条件 | 无 |
| | 后置条件 | 支付订单，订单状态更新 |
| OrderBLService.usePoints | 语法 | public Integer usePoints(Long orderId) |
| | 前置条件 | 无 |
| | 后置条件 | 返回使用积分后的订单价格 |
| OrderBLService.cancelUsePoints | 语法 | public Integer cancelUsePoints(Long orderId) |
| | 前置条件 | 无 |
| | 后置条件 | 返回不使用积分后的订单价格 |
| OrderBLService.checkIllegal | 语法 | public boolean checkIllegal(Long id) |
| | 前置条件 | 无 |
| | 后置条件 | 返回用户是否存在恶意购票 |

| 需要的服务（需接口） | |
|---|---|
| 服务名 | 服务 |
| OrderDataService.save | 向数据库中插入订单对象或更新订单信息 |
| OrderDataService.findByUserId | 根据 UserId 从数据库中得到订单对象列表 |
| OrderDataService.findById | 根据 Id 从数据库中得到订单对象 |
| UserDataService.findByUsername | 根据 username 从数据库中得到用户对象 |
| UserDataService.findById | 根据 id 从数据库中得到用户对象 |
| UserDataService.save | 向数据库中插入用户对象或更新用户信息 |
| TrainDataService.findById | 根据 Id 从数据库中得到车次对象 |
| TrainDataService.save | 向数据库中插入车次对象或更新车次信息 |
| RouteDataService.findById | 根据 id 从数据库中得到路线对象 |

## 5.4 数据层的分解

### 5.4.1 数据层模块的职责

| 列1 | 列2 |
|---|---|
| UserDataService | 持久化数据库接口，提供客户账号和基本信息的增加、修改、查询等操作 |
| StationDataService | 持久化数据库接口，提供车站信息的增加、修改、查询、删除等操作 |
| RouteDataService | 持久化数据库接口，提供路线信息的增加、修改、查询、删除等操作 |
| TrainDataService | 持久化数据库接口，提供车次信息的增加、修改、查询、删除等操作 |
| OrderDataService | 持久化数据库接口，提供订单信息的增加、修改、查询、删除等操作 |

### 5.4.2 数据层模块的接口规范

| 提供的服务（供接口） | | |
|---|---|---|
| **UserDataService.findByUsername** | 语法 | UserEntity findByUsername(String username) |
| | 前置条件 | 无 |
| | 后置条件 | 根据 username 从数据库中得到用户对象 |
| **UserDataService.save** | 语法 | Void save(UserEntity user) |
| | 前置条件 | 无 |
| | 后置条件 | 向数据库中插入用户对象或更新用户信息 |
| **UserDataService.findById** | 语法 | UserEntity findById(Long id) |
| | 前置条件 | 无 |
| | 后置条件 | 根据 id 从数据库中得到用户对象 |

| 提供的服务（供接口） | | |
|---|---|---|
| **StationDataService.findById** | 语法 | StationEntity findById(Long id) |
| | 前置条件 | 无 |
| | 后置条件 | 根据 id 从数据库中得到车站对象 |
| **StationDataService.findAll** | 语法 | List<StationEntity> findAll() |
| | 前置条件 | 无 |
| | 后置条件 | 从数据库中得到所有车站对象 |
| **StationDataService.findByName** | 语法 | StationEntity findByName(String name) |
| | 前置条件 | 无 |
| | 后置条件 | 根据 name 从数据库中得到车站对象 |
| **StationDataService.save** | 语法 | void save(StationEntity station) |
| | 前置条件 | 无 |
| | 后置条件 | 向数据库中插入车站对象或更新车站信息 |
| **StationDataService.delete** | 语法 | void delete(StationEntity station) |
| | 前置条件 | 在数据库中存在该 StationEntity 对象 |
| | 后置条件 | 在数据库中删除车站对象 |

| 提供的服务（供接口） | | |
|---|---|---|
| **RouteDataService.findAll** | 语法 | List<RouteEntity> findAll() |
| | 前置条件 | 无 |
| | 后置条件 | 从数据库中得到所有路线对象 |
| **RouteDataService.delete** | 语法 | void delete(RouteEntity route) |
| | 前置条件 | 在数据库中存在该 RouteEntity 对象 |
| | 后置条件 | 在数据库中删除路线对象 |
| **RouteDataService.save** | 语法 | Void save(RouteEntity route) |
| | 前置条件 | 无 |
| | 后置条件 | 向数据库中插入路线对象或更新路线信息 |
| **RouteDataService.findById** | 语法 | RouteEntity findById(Long id) |
| | 前置条件 | 无 |
| | 后置条件 | 根据 id 从数据库中得到路线对象 |

| 提供的服务（供接口） | | |
|---|---|---|
| **TrainDataService.findByRouteId** | 语法 | List<TrainEntity> findByRouteId(Long routeId) |
| | 前置条件 | 无 |
| | 后置条件 | 根据 routeId 从数据库中得到车次对象 |
| **TrainDataService.deleteAll** | 语法 | void deleteAll(List<TrainEntity> trains) |
| | 前置条件 | 在数据库中存在这些 TrainEntity 对象 |
| | 后置条件 | 在数据库中删除多个车次对象 |
| **TrainDataService.findById** | 语法 | TrainEntity findById(Long id) |
| | 前置条件 | 无 |
| | 后置条件 | 根据 Id 从数据库中得到车次对象 |
| **TrainDataService.findByRouteIdAndDate** | 语法 | List<TrainEntity> findByRouteIdAndDate (Long routeid, String date) |
| | 前置条件 | 无 |
| | 后置条件 | 根据 RouteId 和 Date 从数据库中得到车次对象 |
| **TrainDataService.findAll** | 语法 | void findAll() |
| | 前置条件 | 无 |
| | 后置条件 | 从数据库中得到所有车次对象 |
| **TrainDataService.save** | 语法 | Void save(TrainEntity train) |
| | 前置条件 | 无 |
| | 后置条件 | 向数据库中插入车次对象或更新车次信息 |
| **TrainDataService.deleteById** | 语法 | Void deleteById(Long id) |
| | 前置条件 | 在数据库中存在该 id 对应的 TrainEntity 对象 |
| | 后置条件 | 在数据库中根据 id 删除车次对象 |

| 提供的服务（供接口） | | |
|---|---|---|
| **OrderDataService.save** | 语法 | Void save(OrderEntity order) |
| | 前置条件 | 无 |
| | 后置条件 | 向数据库中插入订单对象或更新订单信息 |
| **OrderDataService.findByUserId** | 语法 | List<OrderEntity> findByUserId (Long userId) |
| | 前置条件 | 无 |
| | 后置条件 | 根据 UserId 从数据库中得到订单对象列表 |
| **OrderDataService.findById** | 语法 | OrderEntity findById(Long id) |
| | 前置条件 | 无 |
| | 后置条件 | 根据 Id 从数据库中得到订单对象 |

# 6. 信息视角

## 6.1 数据持久化对象

系统的PO类就是对应的相关的实体类，在此只做简单的介绍。

UserPO类包含用户编号、用户名、密码、身份证号、姓名、电话号码、证件类型、身份、是否是会员、里程积分、创建时间、更新时间等属性。

StationPO类包含车站编号、车站名称、创建时间、更新时间等属性。

RoutePO类包含路线编号、路线名称、途径车站编号、创建时间、更新时间等属性。

TrainPO类包含车次编号、车次名称、路线编号、座位、列车类型、出发时间、到点数组、开点数组、额外信息、创建时间、更新时间等属性。

OrderPO类订单编号、用户编号、车次编号、出发站编号、到达站编号、订单状态、座位号、价格、优惠、创建时间、更新时间等属性。

```java
public class UserEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotNull
    @Column(unique = true)
    private String username;
    @NotNull
    private String password;

    private String name;
    private String phone;
    private String type;//证件类型
    private String idn;//身份证号
    private String role;
    private Long mileagePoints;//里程积分

    @CreationTimestamp
    private Date createdAt;

    @UpdateTimestamp
    private Date updatedAt;

    //用户是否为会员
    private Boolean isMember;
    private String vippassword;

}
```

```java
public class StationEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```

```
    @NotNull
    @Column(unique = true)
    private String name;

    @CreationTimestamp
    private Date createdAt;

    @UpdateTimestamp
    private Date updatedAt;
}
```

```
public class RouteEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotNull
    @Column(unique = true)
    private String name;

    private List<Long> stationIds;

    @CreationTimestamp
    private Date createdAt;

    @UpdateTimestamp
    private Date updatedAt;
}
```

```java
public class TrainEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotNull
    private String name;

    @NotNull
    private Long routeId;

    @NotNull
    @Type(BooleanArrayType.class)
    @Column(name = "seats", columnDefinition = "boolean[][]")
    private boolean[][] seats;

    @NotNull
    private TrainType trainType;

    @NotNull
    private String date;

    @NotNull
    private List<Date> arrivalTimes;

    @NotNull
    private List<Date> departureTimes;

    @NotNull
    private List<String> extraInfos;

     @CreationTimestamp
     private Date createdAt;

     @UpdateTimestamp
     private Date updatedAt;
}
```

```java
public class OrderEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotNull
    private Long userId;

    @NotNull
    private Long trainId;

    @NotNull
    private Long departureStationId;

    @NotNull
    private Long arrivalStationId;

    @NotNull
    private OrderStatus status;

    @NotNull
    private String seat;
    @NotNull
    private Integer price;
    @NotNull
    private Integer discount;

    @CreationTimestamp
    private Date createdAt;
    @UpdateTimestamp
    private Date updatedAt;
}
```

### 6.2 数据库表

数据库中包含User表、Station表、Route表、Train表、Order表