<在线购票系统>

软件架构文档

版本 <2.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <23/7/2021> | <1.0> | 完成用例视图 | 全体人员 |
| <25/7/2021> | <2.0> | 修改了逻辑视图和部署视图 | 李天策、闫星羽 |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 参考资料 4

2. 用例视图 4

3. 逻辑视图 4

3.1 概述 4

3.2 在构架方面具有重要意义的设计包 4

4. 进程视图 4

5. 部署视图 4

6. 实现视图 5

7. 技术视图 5

8. 数据视图（可选） 5

9. 核心算法设计（可选） 5

10. 质量属性的设计 5

软件架构文档

# 简介

## 目的

本文档将从构架方面对在线购票系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

该文档主要面向本项目的开发人员以供架构参考，说明了软件的设计层次和逻辑，指导本项目的开发与部署过程，有助于项目后期的维护。

## 参考资料

软件工程原理.2013年2月版.高等教育出版社.沈备军，陈昊鹏，陈雨亭编著

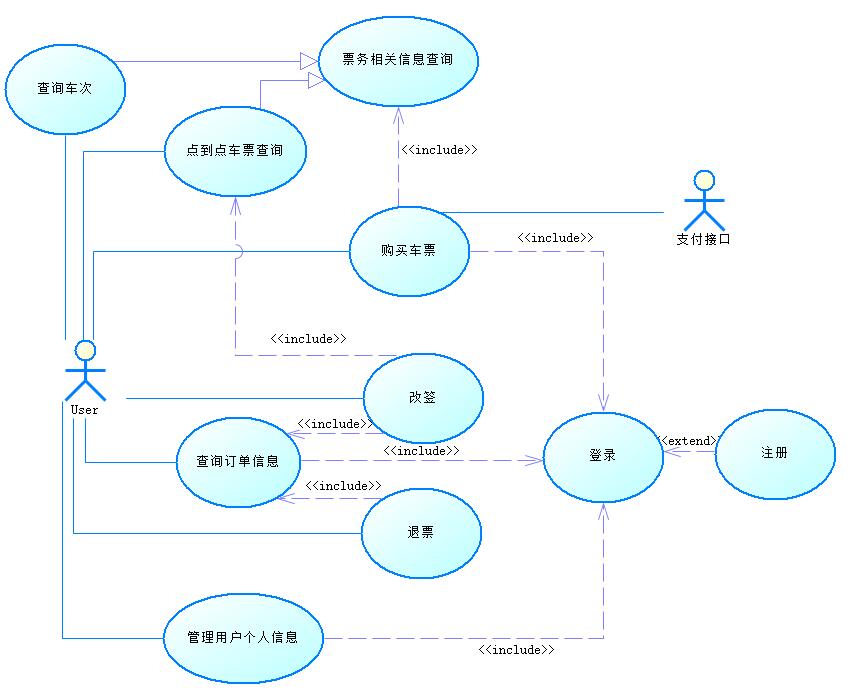
# 用例视图

该视图列举并描述了该项目的一组重要的场景和用例，这些场景和用例代表了该项目将要实现的中心功能，这些用例的实现是每次迭代的工作焦点和中心。

它主要包括以下用例：

* 点到点车票查询
* 车次查询
* 车票购买
* 改签
* 退票
* 订单信息查询
* 用户个人信息管理

## Use-Case展示与说明



### 点对点车票查询：

在点对点车票查询时，用户需要确定起点站与终点站以及日期，点击查询后可以得到满足上述条件并且有空余座位的所有列车信息，同时提供中转方案。

### 车次查询：

车次查询时，用户需要输入车次号，点击查询按钮后，可得到该车次的经停车站和到达离开车站的时刻表信息。

### 车票购买：

用户在查询车票并且完善乘车人信息后，可以通过提交订单按钮进行车票购买。

### 改签：

### 用户已经购买的车次发车前一段时间内，用户可以在订单界面点击改签按钮进行改签。

### 退票：

在用户已经购买的车次发车前一段时间内，用户可以在订单界面点击退票按钮进行退票。

### 订单信息查询：

用户在订单页与订单详情页可以进行订单信息查询。

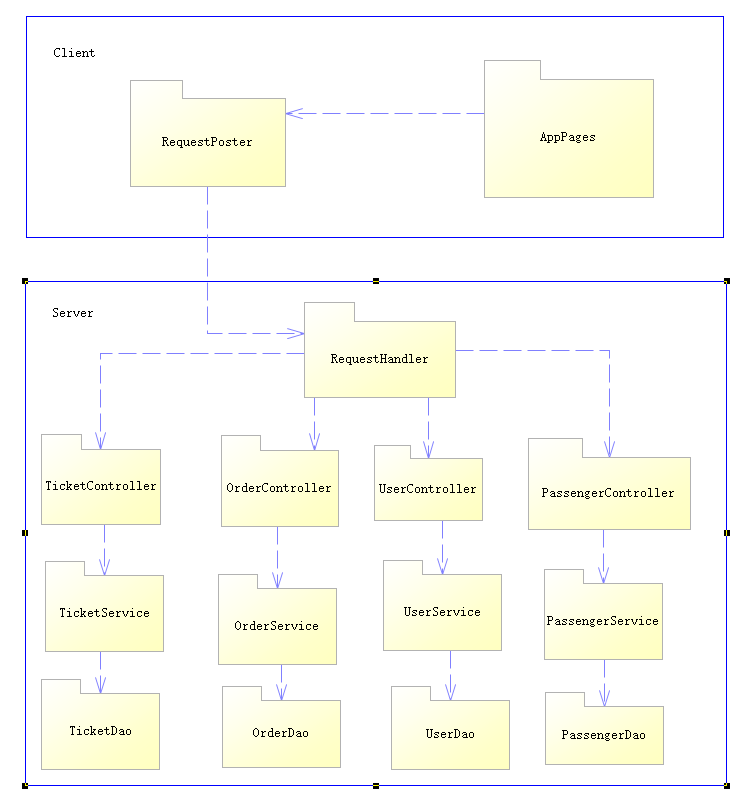
### 用户个人信息管理：

用户可以在个人中心页面可以进行个人信息的查询与修改。

# 逻辑视图

## 概述

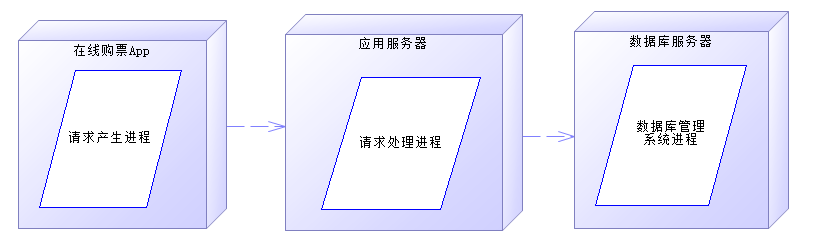
本项目采用系统整体采用C/S的架构风格。用户浏览安卓端App界面即AppPages包中的类，通过交互由RequestPoster中的各类发送请求至Server端，Server中TicketController处理与查询票务信息有关的请求，OrderController处理购票、退票、改签等业务，UserController处理用户注册、登录和信息修改请求，PassengerController用于管理乘客信息。



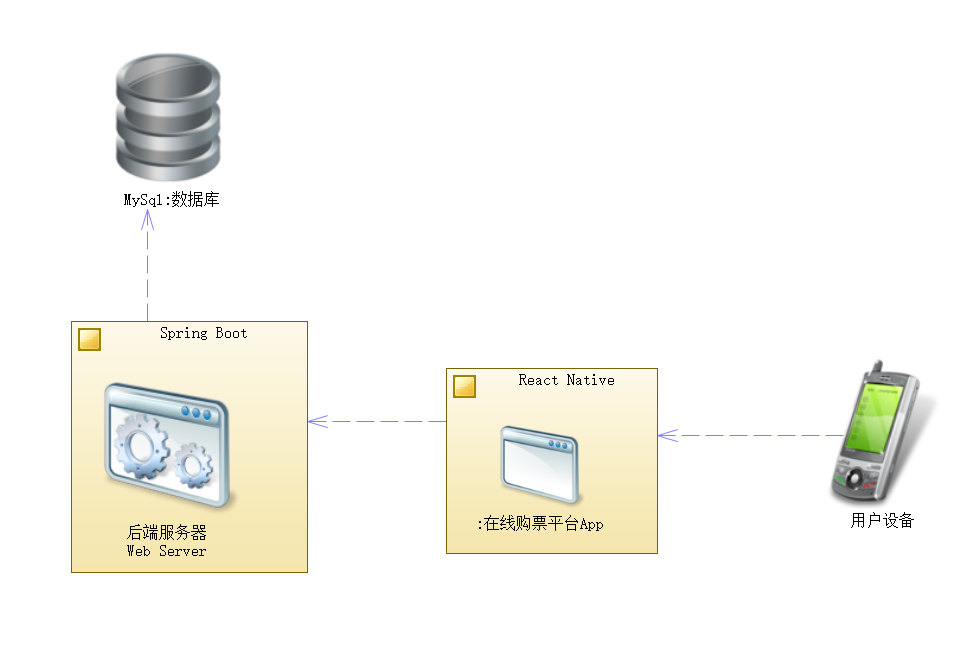
## 在构架方面具有重要意义的设计包

待定

# 部署视图



# 技术视图



软件语言： 前端开发使用React-Native框架，语言使用JavaScript，后端使用Spring框架，语言选择Java语言。

组件和依赖库： 前端开发使用React与 React-Native组件库以及ant-design的组件库；后端开发使用同一版本的Java库以及Spring JPA。

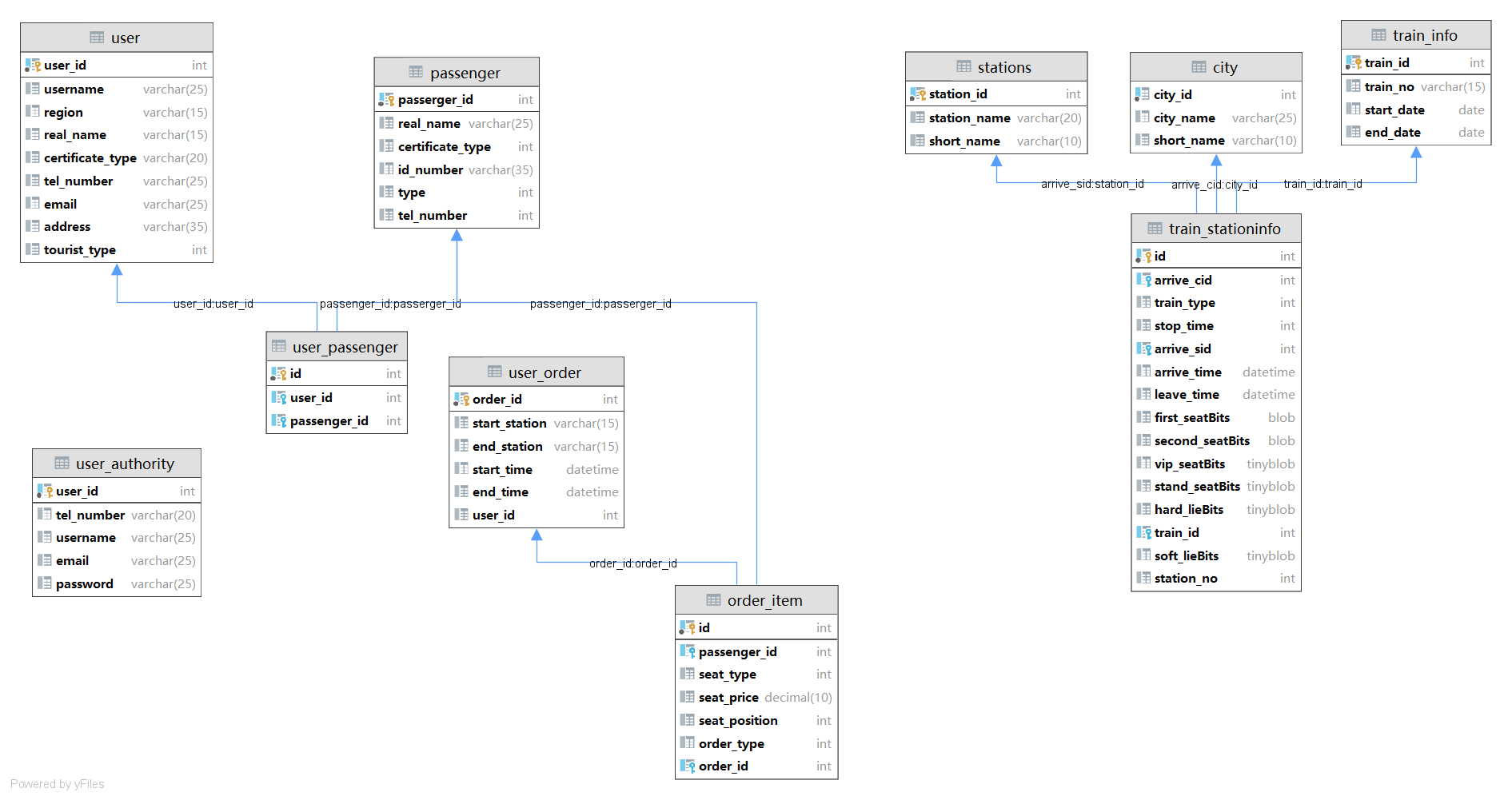
开发工具的指定用途： Windows使用Idea Intellij或者Android Studio进行软件开发，Mac使用Idea Intellij结合Xcode进行软件开发，利用PowerDesigner进行系统的建模，统一使用华为云以及Github进行项目管理。

购买的构件： 待定。

数据库：MySql

中间件：Tomcat

# 数据视图（可选）



# 质量属性的设计

## 易用性

### 用户快速上手使用

使得使用过主流票务软件的用户可以即刻上手，能够进行查询、购票、改签、退票操作的时间低于十分钟。

### 典型操作便利

用户点到点查询车次信息：选择起始和目标车站、选择时间、选择单程或往返、点击查询。

### 适应用户中转需求

用户可以像市面上主流软件一样，进行有中转的点对点查询服务，给用户提供更多信息以满足出现规划需求。

## 可靠性

### 可靠性需求

可用性： 用户能够正常进行查询、购票和改签等操作，在无软件升级或维护等特殊情况下保证7\*24小时使用

平均故障间隔时间：24h

平均修复时间：1h

精准度：无

小错误：车次查询信息显示不完全 （2/1000行代码）

大错误：车次信息显示错误（0/1000行代码）

严重错误：数据丢失无法进行购票、改签、退单操作 （0/1000行代码）

## 性能

### 性能需求

事务响应时间：

1. 用户普通的点击事件响应与页面跳转响应：平均：0.01s，最长：0.1s
2. 用户购票与抢票事务响应： 平均：0.1s, 最长：0.5s
3. 冷启动响应：平均：0.5s， 最长：1s

吞吐量：

每秒平均处理事务数：6000/s

容量：

1. 客户：可容纳客户数：10000，并发服务客户数： 2000
2. 可容纳事务数：100000件

降级模式：

维护软件多个可运行版本，当高版本运行故障，亟待修复时，可以先及时降级到较低版本，保证软件的正常运行，同时进行故障的修复工作

资源利用：待定（受经费制约）

## 可支持性

### 可支持性需求

编码标准： 采用Google编程规范

命名约定： 统一采用驼峰命名法标准，并且利用版本管理工具与项目管理工具保证成员间开发过程中不会出现重名情况。

类库： 前端开发使用React与 React-Native组件库以及ant-design的组件库等；后端开发使用同一版本的Java库以及Spring JPA。

维护访问权： 系统服务器可以识别不同用户的权限类型，并给予不同的访问或维护的权限。

维护实用程序： 待定