天津大学 Tianjin University

密级：公开

《Web 编程实践》
软件设计说明书

题　　目：北洋拼车系统

作　　者：倪宇志、白晨、陈思恩

学　　号：2019229051、2019559052、2019229053

学　　院：国际工程师学院

专　　业：计算机技术

指导教师：李罡

2020 年 10 月

目录

1 需求分析

需求分析将从编写目的、背景、系统目标系统功能分析、系统总体功能图、 数据字典、数据需求及业务规则分析、实体和联系等方面进行分析。

1.1 编写目的

让该文档的使用成员成为拼车系统的开发人员及拼车系统具体组织管理人员。

1.2 背景

拼车是指具有相同路线的人乘坐同一车辆进行通勤及节假日出行， 车费由乘客协议分摊的行为。市场经济，理性经济人假设条件下，任何一项事物的产生都有着一定的必然性，就“拼车”而言，其产生的原因大致可概括为六个方面：

（1）随着社会经济的发展，人们生活水平的提高，小汽车开始走进千家万户， 这使得“拼车”成为了可能，即为“拼车”的出现提供了必要的物质基础；

（2）机动车保有（出行）成本的大幅攀升（如燃油、停车、维修保养等），迫使车主开始寻找成本分担方法；

（3）现有 615 快线运输的供给无法满足学生的出行需求（节假日期间尤为明显），那部分无法被满足的“过剩需求”必将转寻其他方式，“拼车”无疑是一种不错的选择；

（4）学生个性化出行需求的增多（特别是周末），目前天津大学的公交车无法快速满足学生们的出行需求；

（5） 一些交通管理措施的实行，推动了“拼车”的产生，目前滴滴出行，以及出租车无法真正保证学生这个弱势群体的安全。

（ 6）信息技术的飞速发展，特别是互联网的广泛普及，为“拼车”信息的发布、检索以及“拼车”条件的协商提供了中介平台，在百度上进行搜索可以发现， “拼”作为一种崭新的生活 方式已逐渐被人们所接受。

将上述六个方面的原因分为三类：第一，供给条件；第二，需求条件；第三，中介平台条件。从以上三种需求可以看出天津大学拼车的产生已经成为一种趋势，通过“北洋拼车”可以在解决同学们出行的问题的同时为天大师生提供一个安全系数高的平台。

1.3 系统目标

有车一族在网上发布出行信息（每天的出行、远途出行都可以），没车（或者有车，但不想开车）又顺路的人可以拼车，当然搭车的人需要付一定的金额。北洋拼车系统主要有一下功能需求：

1、车主可以发布可用拼车信息，包括：

发车时间，例如下午 16:00

座位数：例如 3 个座位

起止点：例如从新校区去老校区

行驶路线（车主选填）：在地图上标出行驶的路线，以便有些想在中途下车的人选择。

2、需要拼车的人，选择一个合适的车辆，下一个拼车订单，在备注中可以加上：

发车时间能不能微调，例如能不能晚 10 分钟

中途在 XX 地方下车可不可以。

其他需要商量的信息

然后车主选择同意或者不同意，拼车订单完成。订单成功时，应该减掉相应的可用座位数。

以上每一步都要有提示信息，避免错过。

2 系统功能分析

北洋拼车系统功能:用户基本信息管理、行程信息发布、拼车广场模块、历史发布信息查看。

2.1 用户信息管理功能

提供卖家基本信息注册、注销、修改、查询以及统计功能。包括：

（1）用户基本信息录入：注册时要求填写基本信息，包括性别、出生年月、微信号码、QQ 号码、所属区域、工作地等信息。系统检测所有信息填写正确后提示客户注册成功。

（2）用户基本信息修改： 用户基本信息注册错误或有所改变时可以进行信息的修改。

2.2 拼车信息功能

提供拼车信息的添加、拼车信息的查询、拼车信息的修改、拼车信息的删除、拼车加入、拼车结束。包括：

（1）拼车信息添加：车主或乘客对确定路线产生一个拼车广场信息（车找人或人找车）供其他用户的加入，增加拼车订单需要用户 ID、起始终止地址、联系电话、座位数、订单备注等信息；

（2）拼车信息修改：对拼车信息具体内容修改；

（3）拼车信息查询：用户可以根据不同的条件筛选查询拼车信息，包括时间、车找人或人找车等条件；

（4）拼车信息删除：拼车信息的发布者对该信息进行删除；

（5）拼车加入：车主或乘客加入另一方发布的某个拼车信息，形成拼车订单；

（6）拼车结束：拼车订单在行程结束后完成并评分，或未结束时加入方退出该拼车行程。

2.3 后台统计功能

提供后台管理员对拼车订单的查询、拼车排行榜生成以及拼车信息导出功能。包括：

（1）拼车订单查询：管理员查询一段时期内的全部拼车记录或者某位车

主的拼车记录；

（2）拼车排行榜生成：管理员查询一段时期内按成功的拼车记录数量排序的车主列表，以便进行表彰；

（3）拼车信息导出：管理员将拼车记录、拼车排行榜等信息导出为电子表格形式，便于进行进一步统计。

2.4 系统总体功能图

该系统功能图如图 2-1 所示。

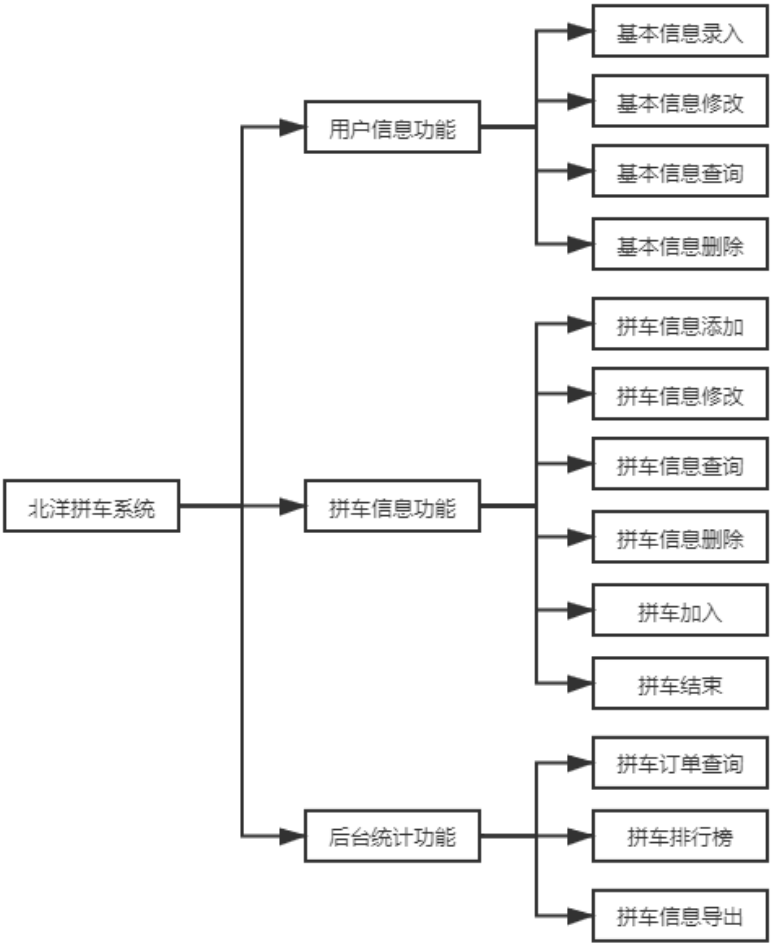


图 2-1:系统功能图

3 数据库设计

MySQL 是一一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，属于 Oracle 旗下公司。MySQL 最流行的关系型数据库管理系统，在 WEB 应用方面 MySQL 是最好的 RDBMS (Relational Database Management System 关系数据库管理系统)应用软件之一。MySQL 是一种关联数据库管理系统，关联数据库将数据保存在不同的表中，而不是将所有数据放在一一个大仓库内，这样就增加了速度并提高了灵活性。

MySQL 所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 软件采用了双授权政策，它分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。

在北洋拼车的项目中，我们定义了三张表，分别是 tb_user、tb_release、tb_admin。分别用与存储用户信息、存储发布信息、存储后台管理数据。

3.1 用户信息相关

我们定义 tb_user 表用于存储用户相关数据，具体内容如表 3-1 所示。

表 3-1 用户相关数据表结构

属性名称	数据类型	解释
id	int(11)	用户在数据库中的 ID
username	varchar(255)	用户名称
avatar_url	varchar(1000)	用户头像图片地址
open_id	varchar(50)	用户在微信认证平台上的 openid
login_date	datetime	上次登录时间
sex	varchar(255)	性别
born	date	出生日期
wx_number	varchar(25)	微信号
qq_number	varchar(255)	QQ 号
local_address	varchar(255)	住址
work_address	varchar(255)	工作地址

constellation	varchar(255)	星座

3.2 订单相关

我们定义 tb_release 表用于存储订单相关数据，具体内容如表 3-2 所示。

表 3-2 订单相关数据表结构

属性名称	数据类型	解释
id	int(11)	订单 ID
user_id	varchar(50)	用户 ID
start_address	varchar(255)	出发地址
end_address	varchar(255)	目的地址
go_date	datetime	出发时间
tel	varchar(255)	联系电话
p_count	varchar(11)	座位数
mark	varchar(255)	该订单评分
release_date	datetime	到达时间
is_show	tinyint(1)	该订单显示/隐藏
ftype	varchar(255)	订单备注

3.3 管理后台相关

我们定义 tb_admin 表用于存储管理后台相关数据，具体内容如表 3-3 所示。

表 3-3 管理后台相关数据表结构

属性名称	数据类型	解释
id	int(11)	后台管理员在数据库中的 ID
login	varchar(50)	后台管理员登录名
password	varchar(255)	后台管理员经过散列处理后的密码
login_date	datetime	上次登录时间

work_address	varchar(255)	后台管理员工作地址
tel	varchar(255)	后台管理员联系电话

4 后端接口设计

接下来，从项目的开发、部署（测试）和上线三个阶段介绍北洋拼车项目。

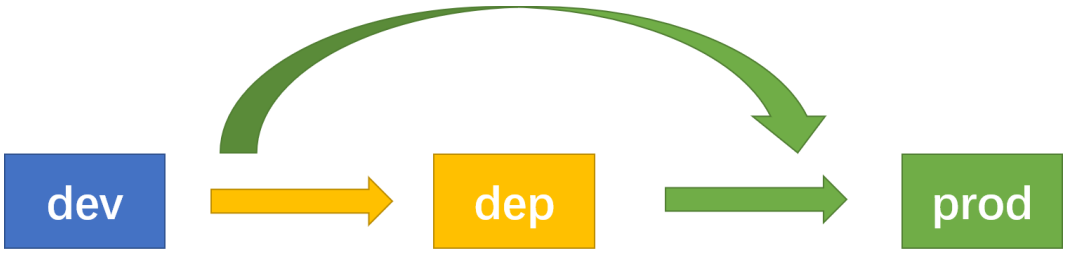


图 4-1 北洋拼车项目阶段图

首先需要明确的是三个不同阶段：

（1）dev

即 develop 或者 development, 这里指开发阶段，通常代码是直接在本地编译、运行和测试。 此外，这里服务访问地址通常是 localhost。这里的“用户”主要是指开发者本身。

（2）dep

即 deploy 或者 deployment，这里指测试阶段，通常代码已经编译打包运行在远程服务器中，可以对外服务。此外，这里服务访问地址通常是 IP 地址。如果 IP 是公网 IP，那么 部署以后就可以对外服务；如果是内网地址，那么只能内网访问。这里的“用户”主要是 指开发者本身、测试者；当然，如果是局域网或者不介意 IP 访问的，那么这里的“用户” 也可能是最终使用者用户。

（3）prod

即 product 或者 production，这里指上线阶段，通常也是代码编译打包运行在远处服务器中可以对外服务。此外，这里服务访问地址通常是域名地址，同时端口是 80 web 端口。上线以后直接面向的是最终用户。虽然服务的代码本身和 dep 是完全一样的，但是考虑到场景的不同，上线阶段可能在运行环境方面需要做调整，例如采用反向代理屏蔽内部实际项目结构。此外，最大的不同应该是上线环境下要使用域名和 80 端最后，其实 dep 和 prod 不存在先后关系。例如，如果开发者已经存在域名和生产环境，可以直接 跳过 dep 阶段，而直接部署在线上环境中。因此有些时候，这里部署和上线是一个阶段。

4.1 spring boot 环境配置

Spring Boot 开发环境

（1）安装 JDK8（可以是 Oracle JDK 或者 OpenJDK）

（2）安装 Maven

（3）安装 Git（可选）

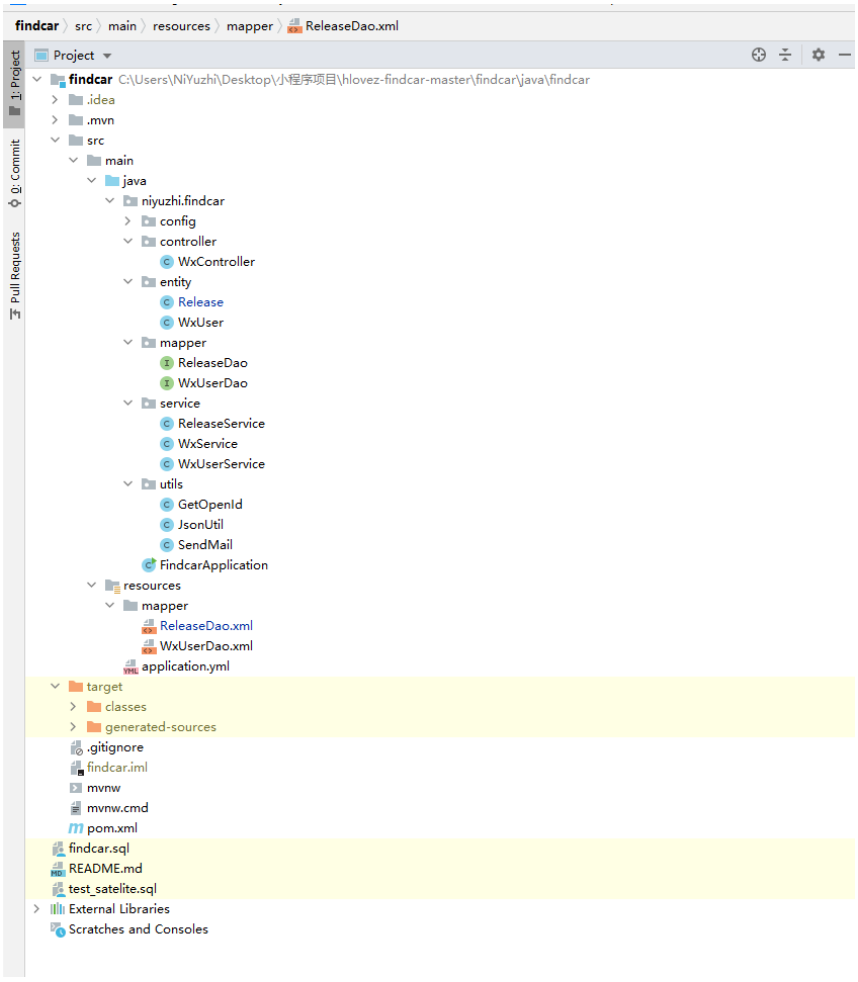
（4）安装 IDEA Community，建议安装 Maven 插件和 Git 插件。 这里 IDEA 社区版即可，不要求 IDEA 商业版。 Eclipse 没有试过，但应该也是可行的。

（5）IDEA 导入本项目

（6）采用 Maven 命令安装依赖库

4.2 系统业务设计

SpringBoot 分为四层：controller 层、service 层、dao 层、entity 层。整个后端的项目结构如下所示：



4.2.1 entity 层

和 model 层一样，存放的是实体类，属性值与数据库值保持一致，在本项目中，我们创建了两个实体：Release 与 WxUser，如下图所示。

```
package niyuzhi.findcar.entity;

import ...

@{...}
//发布者的数据库
public class Release {
    private Integer id;//系统自增
    private String userId;//用户id
    private String startAddress;
    private String endAddress;
    private String goDate;
    private String tel;
    private String peopleCount;//人数
    private String mark;
    private String releaseDate;
    private Boolean isShow;
    private String ftype;//人找车or车找人
    private Integer pageNum;
    private Integer pageSize;
    private List<WxUser> list;//响应的人

}
```

```
package niyuzhi.findcar.entity;

import ...

@Component
@Data
public class WxUser {
    private Integer id;
    private String userName;
    private String openId;//用户识别码
    private String avatarUrl;
    private String loginDate;
    private String sex;
    private String bornDate;
    private String wxNumber;
    private String qqNumber;
    private String localAddress;
    private String workAddress;
    private String constellation;//星座
}
```

4.2.2 dao 层

即 mapper 层，对数据库进行持久化操作，他的方法使针对数据库操作的，基本上用的就是增删改查，他就是个接口，只有方法名，具体实现在 mapper.xml 中实现，此处包括 `ReleaseDao` 和 `WxUserDao`。

```
package niyuzhi.findcar.mapper;

import ...

@Repository
public interface ReleaseDao {
    int insert(@Param("pojo") Release pojo);

    List<Release> select(@Param("pojo")Release pojo);

    List<Map<Object,Object>> findCount(@Param("userId")String userId);

    int delete(@Param("id")Integer id);
}
```

```
package niyuzhi.findcar.mapper;

import ...

@Repository
public interface WxUserDao {

    int insert(@Param("pojo") WxUser pojo);

    int insertList(@Param("pojos") List< WxUser> pojo);

    List<WxUser> select(@Param("pojo") WxUser pojo);

    int update(@Param("pojo") WxUser pojo);

    WxUser selectByOpenId(@Param("userId")String userId);

}
```

4.2.3 service 层

业务层，存放业务逻辑处理，不直接对数据库进行操作，有接口和接口实现类，提供 controller 层调用方法。

`ReleaseService` 类：

```
package niyuzhi.findcar.service;

import ...

@Service
public class ReleaseService {
    @Autowired
    ReleaseDao releaseDao;
    @Autowired
    JsonUtil util;
    public JSONObject addRelease(Release release){
        return util.updateJson(releaseDao.insert(release));
    }
//获取某人发布的信息
    public PageInfo<Release> selectRelease(Release release){
        PageHelper.startPage(release.getPageNum(),release.getPageSize());
        List<Release> list = releaseDao.select(release);
        return new PageInfo<>(list);
    }
    public  List<Map<Object,Object>> findCount(String userId){
        return releaseDao.findCount(userId);
    }
    public JSONObject delete(Integer id){
        return util.updateJson(releaseDao.delete(id));
    }
}
```

WxService 类：

```
package niyuzhi.findcar.service;

import ...

@Service
@Slf4j
public class WxService {
    @Autowired
    private GetOpenId getOpenId;
    @Autowired
    private JsonUtil jsonUtil;
//    @Autowired
//    private SendMail sendMail;

    public JSONObject getOpenId(String code){
//        log.info("code:"+code);
       return jsonUtil.toJson(getOpenId.getOpenId(code));
    }

    public void sendMessage(String message,String userName){
//        sendMail.sendSimpleMail(message,userName);
    }
}
```

WxUserService 类：

```
package niyuzhi.findcar.service;

import ...

@Service
public class WxUserService {

    @Autowired
    WxUserDao wxUserDao;
    @Autowired
    JsonUtil util;

    public JSONObject insert(WxUser pojo) { return util.updateJson(wxUserDao.insert(pojo)); }

    public int insertList(List< WxUser> pojos) { return wxUserDao.insertList(pojos); }

    public List<WxUser> select(WxUser pojo) { return wxUserDao.select(pojo); }

    public JSONObject update(WxUser pojo) { return util.updateJson(wxUserDao.update(pojo)); }

    public WxUser selectByOpenId(String userId){
        return wxUserDao.selectByOpenId(userId);
    }

}
```

4.2.4 controller 层

控制层，导入 service 层，调用 service 方法，controller 通过接受前端传来的参数进行业务操作，在返回一个制定的路径或数据表。

```java
import ...

@Slf4j
@RestController
@RequestMapping("/wx")
public class WxController {
    @Autowired
    private WxService wxService;
    @Autowired
    private WxUserService wxUserService;
    @Autowired
    private ReleaseService releaseService;

    @GetMapping("/getOpenId")
    public JSONObject getOpenId(String code) { return wxService.getOpenId(code); }


    @PostMapping("/loginInfo")
    public JSONObject loginInfo(@RequestBody WxUser user){...}
    @PostMapping("/addRelease")
    public JSONObject addRelease(@RequestBody Release release){...}

    @PostMapping("/selectRelease")
    public PageInfo<Release> selectRelease(@RequestBody Release release){...}

    @GetMapping("/test")
    public String test() { return "Successful"; }

    @GetMapping("/findCount")
    public List<Map<Object,Object>> findCount(String userId) { return releaseService.findCount(userId); }

    @PostMapping("/deleteInfo")
    public JSONObject deleteInfo(@RequestBody Map<String,Integer> map){...}

    @PostMapping("/getUserInfo")
    public WxUser getUserInfo(@RequestBody Map<String,String>map){...}

    @PostMapping("/updateInfo")
    public JSONObject updateInfo(@RequestBody WxUser user) { return wxUserService.update(user); }

    @PostMapping("/sendMessage")
    public void sendMessage(@RequestBody Map<String,String>map){...}
}
```

5 前端小程序设计

5.1 小程序概述

微信小程序是一种不用下载就能使用的应用，也是一项创新，经过将近三年的发展，已经构造了新的微信小程序开发环境和开发者生态。微信小程序也是这么多年来中国 IT 行业里一个真正能够影响到普通程序员的创新成果，已经有超过 150 万的开发者加入到了微信小程序的开发，与我们一起共同发力推动微信小程序的发展，微信小程序应用数量超过了一百万，覆盖 200 多个细分的行业。

微信小程序有很多优点，在对比了几种前端后，最终选择了小程序作为我们的前端技术。首先，无需下载安装：小程序跟 APP 不一样，只要打开微信找到相应的小程序就可以使用，无需任何下载和安装，不占用手机一点内存 。即用即走：用户扫一扫或者搜一下即可打开应用，用完退出即可，简单快捷的方式更容易提高使用率。功能更丰富：与传统 APP 相比，它更轻便易用，与服务号相比，它的功能也更丰富。试想一下，在很多推广中，我们都要求用户安装 APP 才能享受福利，步骤繁琐，很容易引起用户反感，而小程序则可以改变这种情况，用户只要扫一扫就能找到活动入口，同时它还能实现很多服务号实现不了的功能，为用户提供更好的体验。综上所述，对于北洋拼车这种小型应用程序，如果采用浏览器网页作为我们的前端便失去了他的便捷性，如果生成 APP 又会造成占用内存空间的问题，所以我们选择小程序作为我们的前端。

5.2 小程序架构设计

小程序主要页面由以下几部分构成：

1、about 页面：主要负责整个项目的介绍。

2、add 页面：负责信息的发布，包括人找车和车找人两个模块，对表单进行收集，包括：起点、终点、出发日期、出发时间、联系方式、出行人数、实行路线。

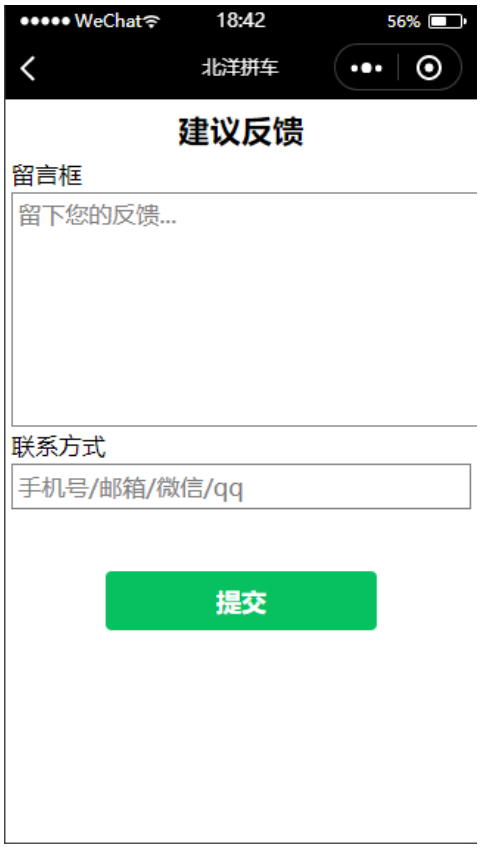
3、detail 界面用于详细展示拼车信息。

4、history 界面：对于用户的发布历史数据进行展示。

5、index 页面：首页，主要展示当前的一些拼车信息，还包括定位、检索模块。

6、info 页面：对于用户的个人信息进行维护更新。

7、message 模块：收集反馈意见。



8、publish 模块：形成信息发布选择身份模块。

9、user 模块：展示<我的>

6 项目部署与测试

6.1 小程序项目部署

6.1.1.开发环境要求

操作系统平台：Windows、MacOS

数据库平台：MySQL

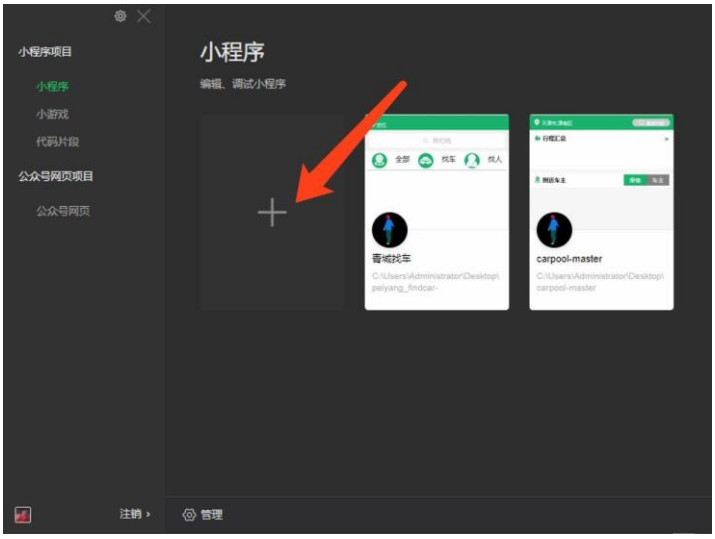
开发语言要求：Java、JavaScript、 PHP

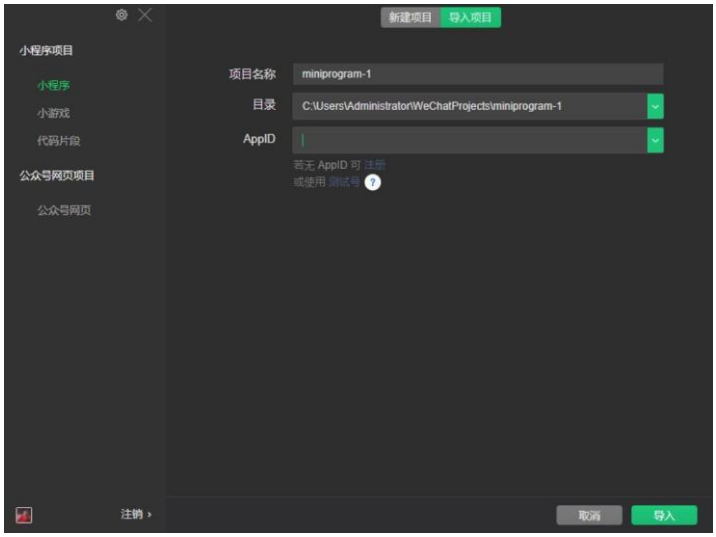
开发工具：微信小程序开发工具、IDEA

6.1.2.项目部署

- 微信小程序：

1.在微信小程序中添加-》导入项目，AppID 填写微信开发者账号的AppID即可。





2.在模拟器中即可对小程序的功能进行测试，在编辑器中进行开发调试。

- 小程序后台

1.在可视化数据库中执行.sql 导入数据库表的结构和测试信息

2.在applicationConfig.xml 中配置本地数据库的url、用户名和密码

6.2 项目测试

对“北洋拼车系统”的测试有助于实现以下目标：

- 通过软件测试的测试方法和测试样例暴露软件中的缺陷和 BUG。
- 记录软件运行中的数据，避免不合法的数据和边界值。
- 确保该项目现有的功能构件可以达到系统分析中的目标
- 测试软件在不同平台不同客户端下的兼容性，功能是否可以正常运行
- 测试软件的健壮性即在高并发、网络状态不好情况下的性能表现等。

具体的测试情况请查看测试文档，在此对项目的界面测试情况进行呈现。