

# 《移动互联网》项目报告-SSR

## 概述

本项目实现了一个共享淋浴房平台系统的原型（SSR, Shared Shower Room）。其中前端包括基于Android开发的移动端和基于Vue.js开发的PC端；后端采用了Django进行开发，引入了单元测试保证其稳定性。

## 功能介绍

### 商家

作为淋浴服务的提供者（下文称为商家），商家可以在该平台的PC端创建账号，进行账号相关的操作（注册登录，修改密码，重置密码）。

作为拥有设备的商家，商家可以创建并管理自己设置的地点信息和设备信息，进行增删查改。在查看设备信息时，PC端支持商家根据设备地点，设备状态等字段进行筛选，提高了系统的可用度。

为了和顾客进行交互，商家还可以查看顾客提交的反馈信息，根据反馈信息的类型对实际设备进行相关操作。

作为一个平台系统，PC端支持导出设备数据和用户数据，方便商家进行进一步统计。

### 顾客

作为淋浴服务的使用者（下文称为顾客），顾客可以在该平台的移动端创建账号，进行账号相关的操作（注册登录，修改密码，查看账号详情）。

作为使用设备的顾客，顾客可以利用Android设备提供的GPS定位服务查找附近的设备。附近设备的信息在应用列表中显示，顾客也可以在地图上查看其具体位置。

顾客在找到设备后，可以利用移动端提供的二维码扫描功能，扫描设备上的二维码，进入设备操作界面。当顾客结束使用设备后，系统会按照使用时长在其账户余额中扣除相应时长。

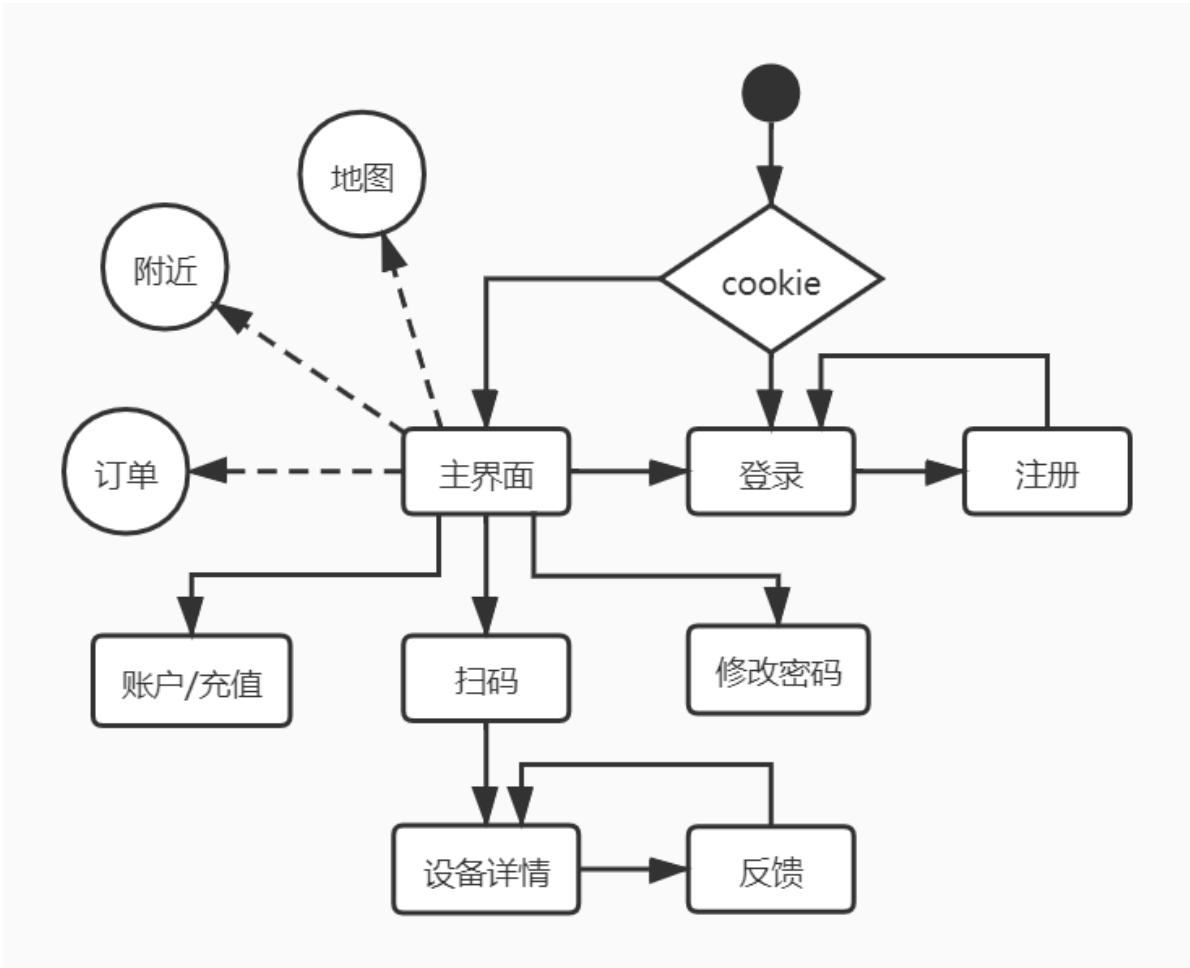
顾客如果发现设备发生故障，可以在移动端向服务器发送反馈信息。当商家登录PC端平台后，会从服务器拉取相应的反馈信息，并作出进一步反应。

顾客在结束使用设备后，可以在订单区域查看自己之前的使用记录，包括设备ID，开始使用时间和结束使用时间。

## 界面设计

### 移动端

#### 界面逻辑

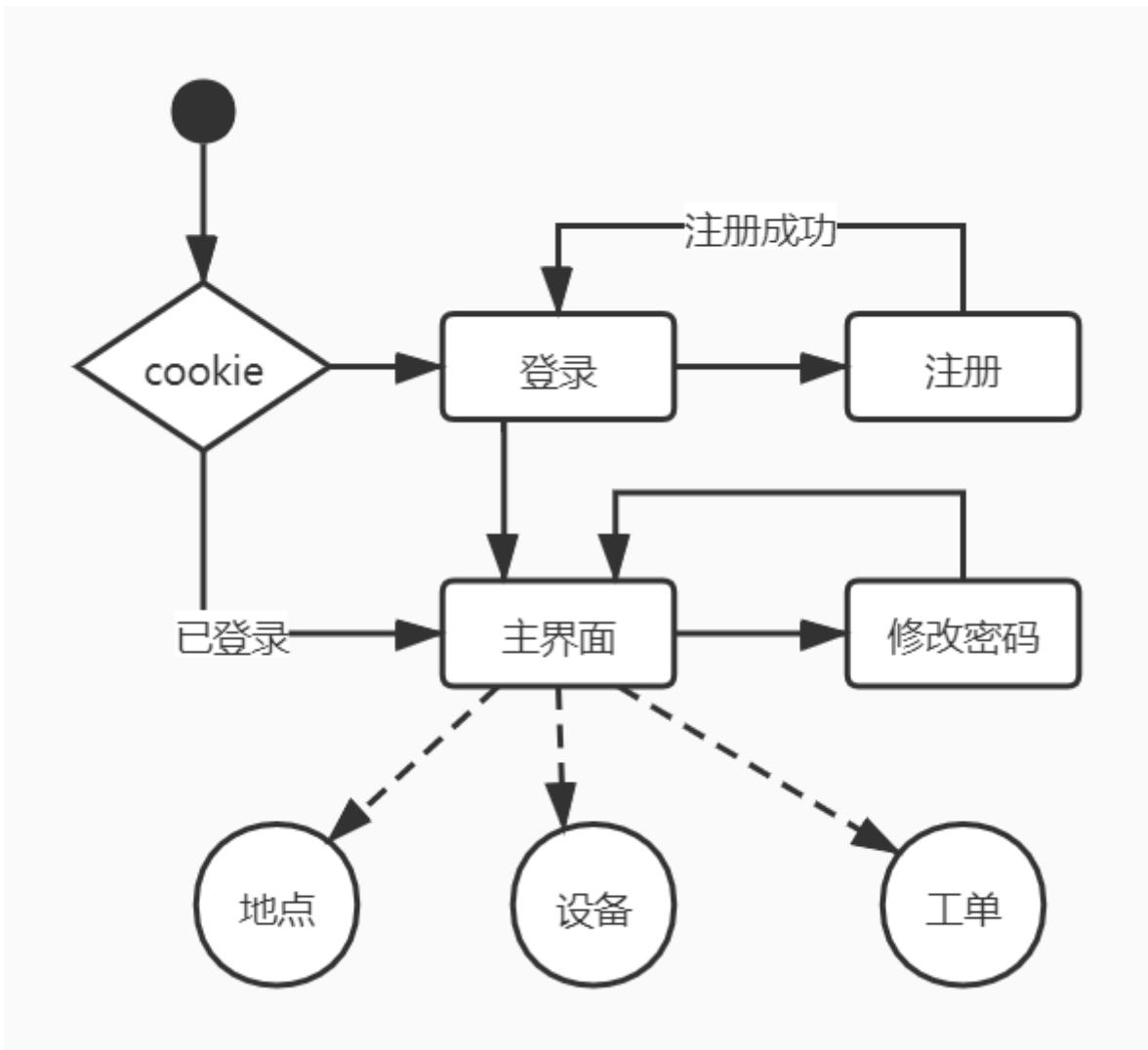


## 界面UI

移动端使用的大部分UI组件均为Android提供的原生组件。在附近设备和订单视图中，使用了GitHub开源的[CardStackView](#)。

## PC端

## 界面逻辑

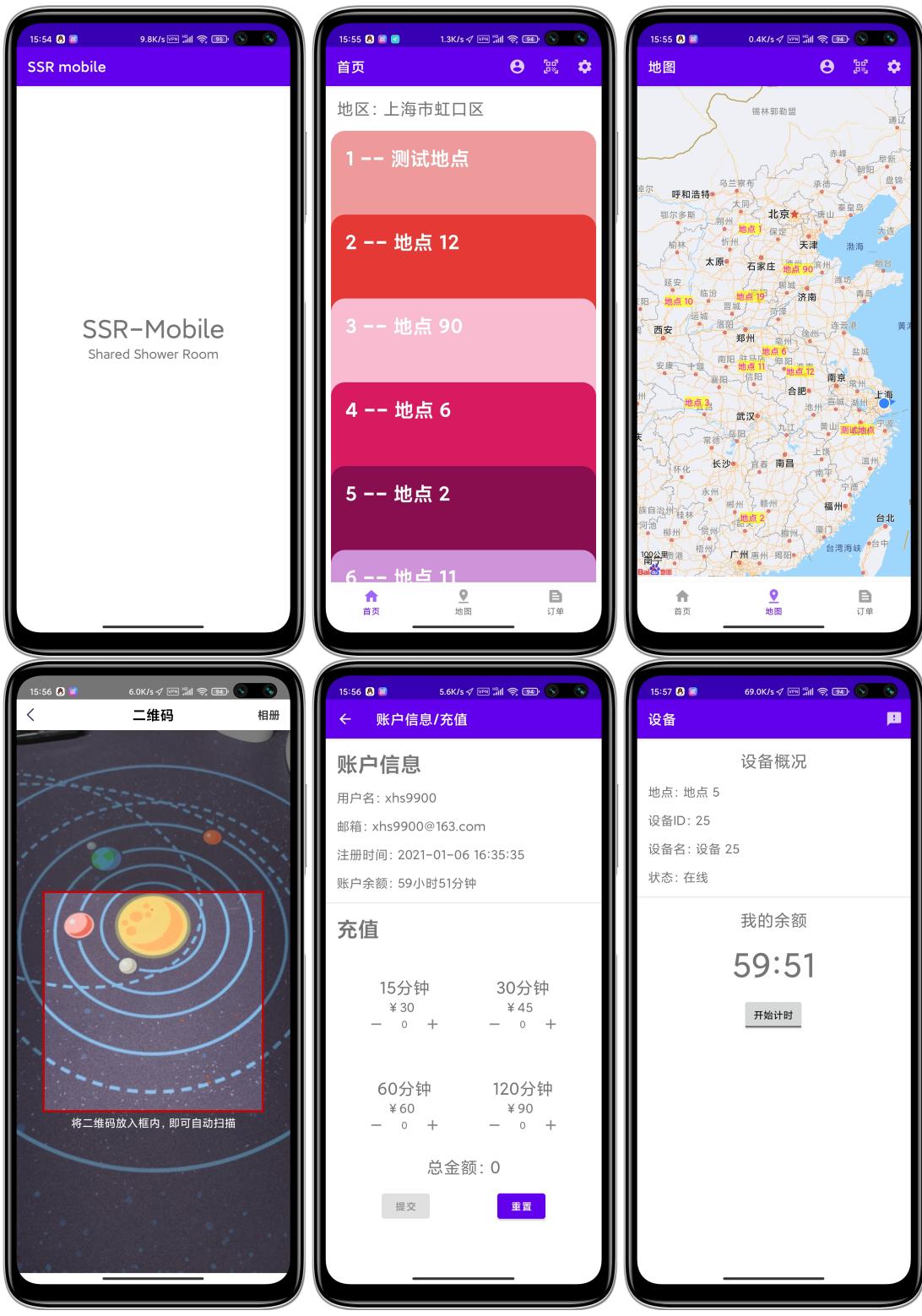


## 界面UI

PC端主要使用了[Element-UI](#)提供的UI组件。

## 项目效果

### 移动端



PC端

ssr-frontend-web × + localhost:8080/region

用户名: xhs  
修改密码

地点 1 113.88423638566184 39.40294891567807 编辑 删除

地点 10 109.833344539691 36.13917550555817 编辑 删除

地点 11 113.96411843670178 33.08268206366456 编辑 删除

地点 12 116.7424992809228 32.638275564514557 编辑 删除

地点 13 109.39955289875199 28.86258716562501 编辑 删除

地点 14 101.99601621522667 27.375884259811755 编辑 删除

地点 15 107.27705578180306 27.232590334672935 编辑 删除

地点 16 107.06750265670844 34.22053196687367 编辑 删除

地点 17 109.4137715955119 36.58607388216339 编辑 删除

地点 18 101.23570954002429 39.89663298512121 编辑 删除

地点 19 113.91290179494985 36.362897289182 编辑 删除

地点 2 114.0124562641381 25.5561272888177 编辑 删除

地点 20 104.9405779796736 38.742477801126256 编辑 删除

地点 3 110.8754902890049 31.342118892950587 编辑 删除

ssr-frontend-web × + localhost:8080/device

地点  
设备名  
地点  
状态  
操作

设备 8 地点 1 离线 编辑 删除

设备 16 地点 10 在线 编辑 删除

设备 53 地点 11 离线 编辑 删除

**设备 12 地点 12 故障** (selected)  
地点 13 离线 编辑 删除

设备 70 地点 14 故障 编辑 删除

设备 74 地点 15 在线 编辑 删除

设备 81 地点 16 离线 编辑 删除

设备 97 地点 17 在线 编辑 删除

设备 56 地点 18 离线 编辑 删除

设备 72 地点 1 离线 编辑 删除

设备 76 地点 10 故障 编辑 删除

设备 77 地点 10 在线 编辑 删除

设备 82 地点 10 离线 编辑 删除

设备 85 地点 10 离线 编辑 删除

设备 87 地点 10 离线 编辑 删除

ssr-frontend-web × + localhost:8080/order

地点  
设备  
**工单**  
设备名  
工单类型  
描述

设备 8 物理损坏 This is a work order description typed "physical d"

设备 16 物理损坏 This is a work order description typed "physical d"

设备 53 无法启动 This is a work order description typed "unable to"

设备 70 其他原因 This is a work order description typed "other": Lo

设备 74 物理损坏 This is a work order description typed "physical d"

设备 74 其他原因 This is a work order description typed "other": Lo

设备 97 无法启动 This is a work order description typed "unable to"

设备 56 其他原因 This is a work order description typed "other": Lo

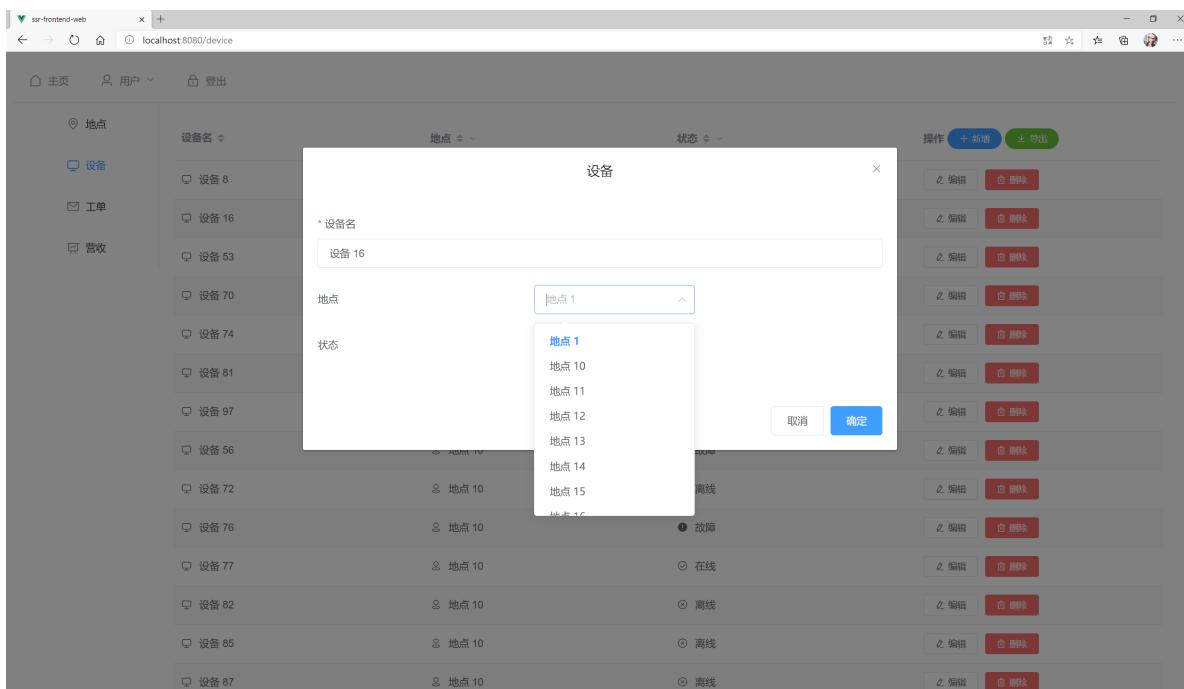
设备 76 无法启动 This is a work order description typed "unable to"

设备 76 ANTIHACKING 描述

完整描述: This is a work order description typed "unable to enter in/out": Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam porttitor ligula massa, vitae porta lectus imperdiet nec. Quisque quis accumsan erat. Nulla ultricies ac purus at vestibulum. Nunc eu est cursus, mattis diam sit amet, egestas massa. Suspendisse vel felis vitae velit fermentum fringilla. Quisque gravida mauris erat, sed mollis augue ultrices at. Proin mattis cursus laoreet. Suspendisse pellentesque orci tristique erat molestie tempus. Pellentesque iaculis libero id risus tincidunt, sit amet viverra leo dapibus. Curabitur id nulla lacus. Nunc at mauris sed metus tempus lacinia. Duis in lacus id ex molestie lacinia nec quis odio. Praesent varius nulla sed ipsum ullamcorper eleifend. Nulla maximus lorem quis leo euismod, nec blandit lectus auctor. Quisque gravida velit eget neque semper vehicula. Donec fringilla molestie luctus.

设备 87 无法进出 This is a work order description typed "unable to"

设备 58 无法启动 This is a work order description typed "unable to"



# 项目运行方法

## 后端

后端使用了Django框架和一些辅助模块。在运行之前，请运行下列命令，确保电脑上安装有以下模块：

```
1 $pip3 install django
2 $pip3 install django-cors-headers
3 $pip3 install djangorestframework
4 $pip3 install validators
```

要运行后端，请在项目目录 `SSR_backend` 下，运行以下命令：

```
1 python3 manage.py runserver address:port
```

其中 `address` 指后端运行所使用的IP地址，`port` 指后端运行使用的端口号，请按需配置。

## 移动端

建议使用Android Studio运行移动端。

在运行前，请修改项目目录 `SSRmobile` 中的 `build.gradle` 文件，相对路径为 `/SSRmobile/app/build.gradle`。

该文件的部分内容如下所示：

```
1 android {
2     compileSdkVersion 30
3     buildToolsVersion "30.0.2"
4
5     defaultConfig {
6         applicationId "com.example.ssrmobile"
7         minSdkVersion 26
8         targetSdkVersion 30
9     }
10 }
```

```
9         versionCode 1
10        versionName "1.0"
11        buildConfigField "String", "BASE_URL",
12        "\"http://45.77.18.246:8000/users/"
13    //        buildConfigField "String", "BASE_URL",
14    //        "\"http://192.168.1.105:8000/users/"
15
16
17    buildTypes {
18        release {
19            minifyEnabled false
20            proguardFiles getDefaultProguardFile('proguard-android-
optimize.txt'), 'proguard-rules.pro'
21        }
22    }
23    compileOptions {
24        sourceCompatibility JavaVersion.VERSION_1_8
25        targetCompatibility JavaVersion.VERSION_1_8
26    }
27    sourceSets{
28        main{
29            jniLibs.srcDir 'libs'
30            jni.srcDirs = []      //disable automatic ndk-build
31        }
32    }
33 }
```

请将 `defaultConfig` 中 `buildConfigField` 字段中的 `BASE_URL` 的IP地址和端口号改为后端自行配置的IP地址和端口号。注意不要删去 `/users/` 部分。

修改结束后，请重新编译项目（顶部状态栏->Build->Rebuild Project）。

请在运行前确保安装了必需的Android SDK。

## PC端

请在运行前确保安装了 `node.js`（包含 `npm`）以及 `vue-cli`。

在运行前，请修改项目目录 `ssr-frontend-web` 中的配置文件，相对路径为 `/ssr-frontend-
web/.env`。

请将文件中 `VUE_APP_URL` 中IP地址和端口号改为后端自行配置的IP地址和端口号。

运行PC端，请在项目目录下运行以下命令：

```
1 npm run serve
```

# 项目工作介绍

## 工作量

### 后端

2000行代码，主要为单元测试部分（936行）和视图逻辑部分（762行）。

#### 单元测试

该项目完成了后端视图逻辑代码的单元测试，覆盖率91%。具体细节可前往项目目录 `SSR_backend` 的 `htmlcov` 目录下 `index.html` 查看。

### 移动端

2000行代码，主要为复杂Activity，例如 `BasicActivity`（376行）、`RegisterActivity`（163行）等等。

### PC端

1600行代码，主要为Vue组件的html模板部分以及Fetch API的回调函数部分。

## 项目技术

### Fragment和Activity交互

Android移动端的主界面使用了Google原生的Bottom Navigation Activity。这个Activity由3个互相切换的Fragment和一个底部导航栏组成。为了实现在Fragment和Activity之间传递数据和信号，我采用了不同的方法。

#### Fragment向Activity获取数据

在移动端，Fragment需要不定期向Activity获取相关数据，例如附近地点，交易记录等等。我采取的解决方式是将Activity中的相关数据权限设为全局（静态公开），Fragment需要的时候直接访问即可。

#### Activity向Fragment发送信号

在移动端，Activity可能需要向当前活跃的Fragment发送信号，例如命令Fragment刷新UI。我使用了接口+回调函数的方式解决了这个问题。

具体实现方式是在Activity中定义一个公开接口 `FragmentListener`，其中包含一个回调函数 `callback`；随后让三个Fragment均实现该接口，决定当该回调函数被触发后需要完成的操作。当Activity需要向当前活跃的Fragment发送信号时，通过 `ChildFragmentManager` 提供的 `getPrimaryNavigationFragment` 方法获取当前活跃的Fragment，在其实现的接口上执行回调函数即可。

### 父组件和router-view交互

PC端通过route动态决定父组件下绑定的子组件。如何在父组件和router-view之间交互成了一个问题，因为父组件和子组件之间不存在显式的绑定关系。我的解决方法是在html模板的 `router-view` 中添加 `v-bind` 参数，绑定需要传递的数据/信号，在子组件的Vue实例中定义 `props` 属性，再添加一个观察属性 `watch`，就实现了父组件向 `router-view` 传递数据/信号的功能。

## 外部调用

### 移动端

移动端使用了[百度地图SDK](#)提供的定位服务和地图界面。

移动端使用了GitHub开源的[ZXing](#)的一部分，用于实现二维码扫描功能。