# 第一章 客户端设计及实施方案

1.1客户端的功能设计

报修客户端的操作流程图。

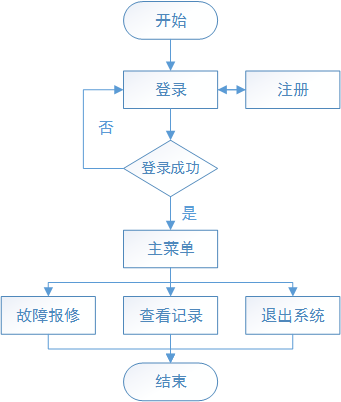


图2.1 报修客户端的操作流程图

2.1.1用户注册

第一次使用该报修系统的新用户，可以通过手机报修客户端上的注册界面进行新账号的注册，新注册的账号信息将通过Android Http通信协议传送到服务器端，服务端把新注册的用户信息存储到数据库中，并同时在手机报修客户端提示用户注册成功。

2.1.2用户登录

已经注册过账号的用户可以通过手机客户端上的登录界面进行账号登录，服务器端会根据用户填写的账号和密码信息选择数据库的数据表进行匹配验证，账号和密码都正确则登录成功，登录成功后则会进入到主菜单界面，可以进行设备故障报修，以及查看报修记录等操作。

2.1.3菜单界面

用户成功登录报修客户端后就会跳转到菜单界面，在菜单界面设计有故障报修、报修记录和退出系统三个功能按钮，点击相关按钮则会进入相应的功能界面。除此之外，在主菜单界面还会加载显示高德地图，并对用户当前的位置进行定位以及在地图上显示。

2.1.4位置定位和显示

在用户成功登录报修客户端后，客户端通过GPS定位或者网络定位获取用户当前位置的经纬度信息，并通过高德地图API在主菜单界面显示出用户的当前位置，以及获取用户当前位置的地点信息。获取到的经纬度信息和地点信息在故障报修时将会和故障报修信息一起提交到服务器端，并由服务器端存储到数据库中。

2.1.5故障报修

故障报修模块是整个报修客户端最主要的一部分，用户成功登录客户端后会跳转到菜单界面，在菜单界面选择故障报修功能则会进入到故障报修界面，在故障报修界面可以进行故障设备的相关报修。报修信息包括：报修事项、故障描述、报修地点、报修人员和联系电话，在报修信息填写完毕后，点击“提交”按钮，客户端就会通过Android Http通信协议向服务器端提交报修信息和用户当前位置的经纬度信息，服务端则会把这些信息存储到数据库中，以便后勤相关工作人员进行查看以及安排维修人员前去维修。

2.1.6报修记录

报修记录模块可以查询到当前登录用户的所有故障报修记录信息，以及每条报修记录的详细信息。在主菜单界面选择查看报修记录后，报修客户端将会向服务器端发送相应的请求，服务器端接收到请求后从数据库中读取相应的报修记录信息，返回给报修客户端并在报修客户端界面上显示，以供用户查看。

2.2客户端的实现方案

1）安装Ecplise程序开发工具，并在Ecplise搭配Android SDK开发环境，用Java程序设计语言进行程序开发，完成手机报修客户端的界面设计，以及实现报修客户端的逻辑功能。

2）利用基于Android的高德地图API应用开发的相关技术，实现在报修客户端上加载显示地图，以及对用户当前位置的定位，并获取用户当前位置的经纬度信息和在地图上显示用户的当前位置。

3）采用Android Http通信协议，实现手机报修客户端和web服务器端的数据交互，通过3G/4G/WIFI等无线通信方式将用户信息、故障报修信息和用户当前位置的经纬度信息提交到服务器端，再通过服务器端把数据信息存储到数据库，以便后勤中心工作人员查看，安排维修人员前去维修。

4）整合Android手机报修客户端和web服务器端，测试手机报修客户端的功能和性能。

2.3客户端的实现技术路线图

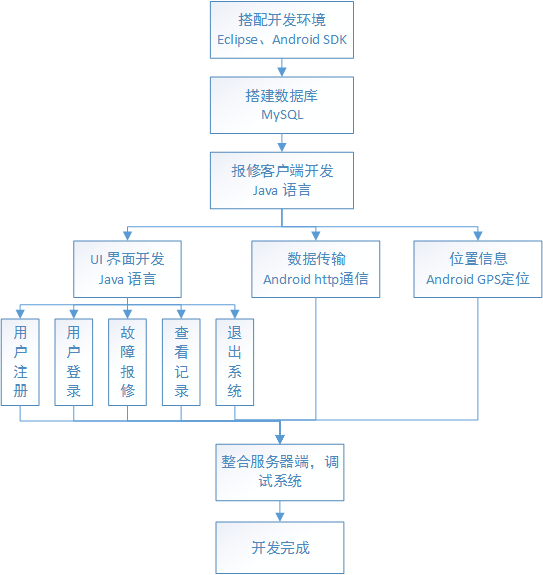


图2.2 客户端的实现技术路线图

# 第二章 系统开发环境配置

3.1Java开发工具包的安装和配置

Java 开发工具包（Java Development Kit,JDK）主要包括了Java运行环境（Java Runtime Environment，JRE）、一些Java命令工具和Java基础类库文件。JDK包含的主要组件如下所示。

* javac：Java编译器，将源程序转换成字节码。
* jar：Java打包工具，将相关的类文件打包成一个文件。
* javadoc：Java文档生成器，从源码注释中提取文档。
* jdb：Java查错工具。
* java：运行编译后的Java程序。

JDK是开发任何类型Java应用程序的基础。开发Android应用程序时，使用

的开发语言是Java，而且安装Eclipse集成开发环境也需要JDK的支持，所以在报修客户端开发之前必须首先在系统中正确安装和配置JDK。

3.1.1安装Java开发工具包

JDK程序的安装包可以Oracle公司的官方网站免费下载，目前最新的JDK版本是JDK 8.0，本毕设开发使用的JDK版本是JDK 7.0。下载后双击图标，即可进行安装。

3.1.2配置Java开发环境

JDK安装成功后，必须对其进行配置，才能够正确使用。JDK配置主要是JDK环境变量的设置，具体步骤如下：

1）打开“我的电脑”--->“属性”，选择“高级系统设置”，在弹出的对话框里选择“高级” --->“环境变量”，打开“环境变量”对话框，如图3.1所示。

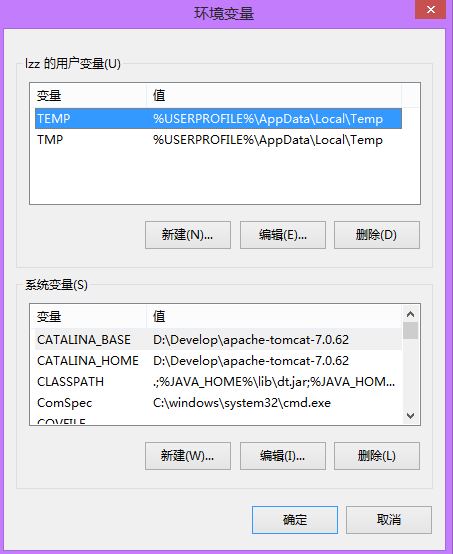


图3.1 环境变量对话框

2）在“系统变量”区域，选择“新建”，新建系统变量，变量名：JAVA\_HOME，变量值：JDK的安装目录，如图3.2所示。



图3.2 新建系统变量对话框

3）在“系统变量”里面选择“Path”环境变量，选择编辑，弹出 “编辑系统变量”对话框，在变量值中添加：%JAVA\_HOME%\bin; 如图3.3所示。

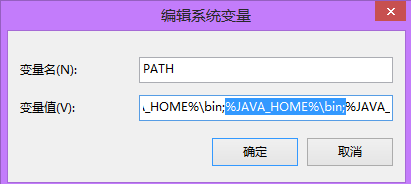


图3.3 向Path环境变量添加JDK路径

4）检验环境变量是否配置成功。在DOC窗口的命令提示符下输入“java -version”命令，如果出现图3.4所示运行界面，则环境变量配置成功。

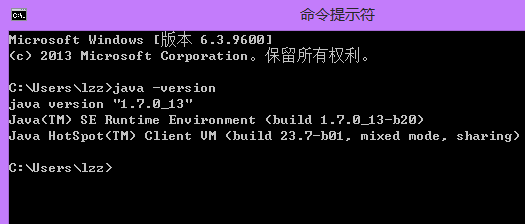


图3.4 JDK环境配置成功

3.2 Android SDK的下载和安装

Android SDK是进行Android应用程序开发的基础，所以在报修客户端进行开发之前，首先必须在系统中安装Android SDK。

3.2.1下载Android SDK

进入Android SDK官方开发网站：http://developer.android.com，选择对应的操作系统所使用的Android SDK下载，下载完毕后解压到磁盘中。

3.2.2安装Android SDK

1）双击解压文件中的“SDK Manager.exe”，进行SDK安装，如图3.5所示，选择要安装的 Android 版本，本文选择了Android 4.0.3版本，然后点击“Install packages”进行安装。

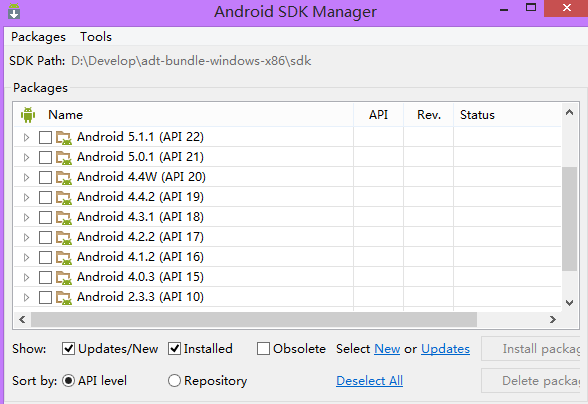


图3.5 Android SDK安装

2）安装完Android SDK后，打开AVD Manager.exe安装对应的安卓模拟器，如图3.6所示。

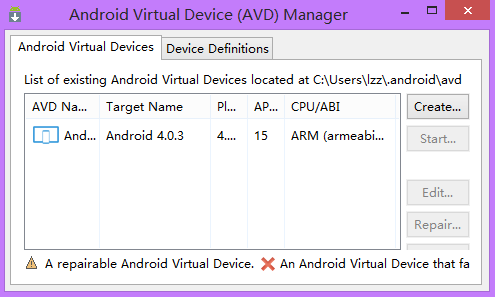


图3.6 安装Android 模拟器

3）单击“Start”按钮，启动Android模拟器，如图3.7所示。

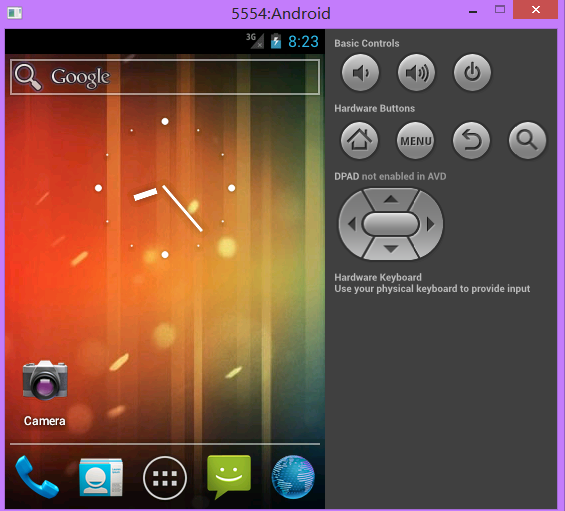


图3.7 Android 模拟器运行界面

3.3 Eclipse的下载和安装

正确安装和配置完Android SDK之后，就可以进行Android应用程序的开发了。但是Android SDK只是提供了Android应用程序的编译和执行工具，并没有提供程序代码的编写环境，通常可以通过Android Studio开发工具，或者在Eclipse中集成Android的Eclipse插件ADT（Android Development Tools），搭配开发环境进行开发，本论文使用的是Eclipse开发工具。

3.3.1下载Eclipse

Eclipse是款可以免费使用的软件，可以在Eclipse的官方网站上直接下载，本论文所用的Eclipse版本为Eclipse 4.2。

3.3.2安装Eclipse

Eclipse下载完成后，解压到相应的文件夹下，双击解压文件中的eclipse.exe文件，就可以启动Eclipse。启动Eclipse后，跳出Eclipse工作台的选择窗口，如图3.8所示，设置好工作台的路径后，点击“OK”按钮，则进入Eclipse的主界面。

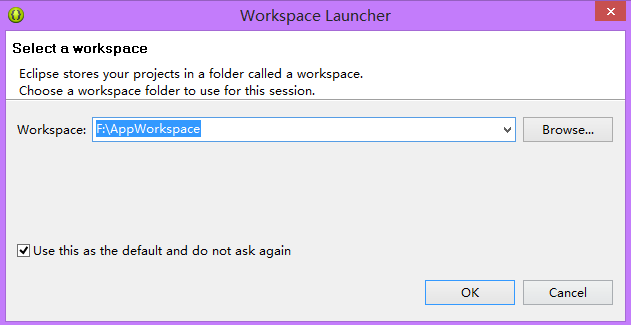


图3.8 选择Eclipse工作台的路径

3.3.3安装和配置Android插件（ADT）

仅仅安装了Eclipse还不能进行Android应用程序的开发，必须在Eclipse安装Android插件ADT。具体安装步骤如下：

1）启动Eclipse，选择菜单栏中“Help”菜单下的“Install New Software”选项，将弹出软件安装窗口，如图3.9所示。

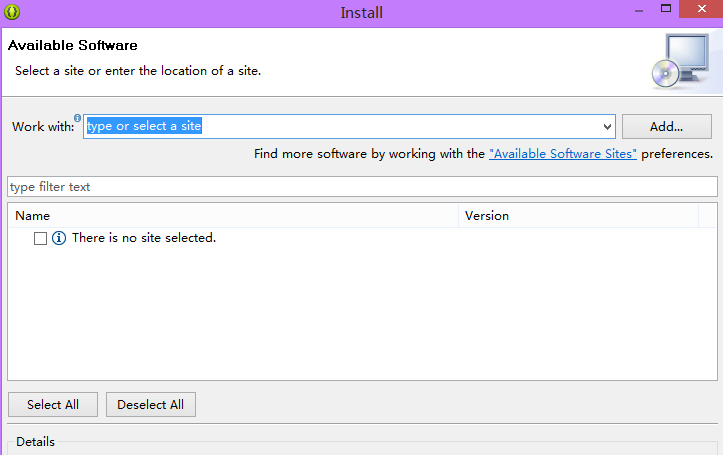


图3.9 软件安装窗口

2）点击“Add”按钮，弹出“Add Repository”对话框，如图3.10所示。点击Archive按钮，弹出要打开的文件，选择下载好的ADT，然后点击“OK”按钮，进行ADT插件的安装。

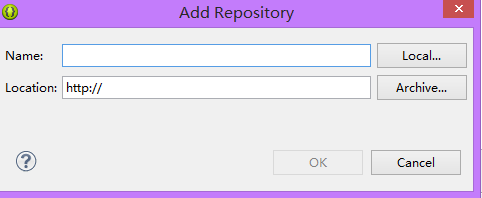


图3.10 “Add Repository”对话框

3）ADT插件安装完成之后，需要在Eclipse中进行配置。选择Eclipse菜单栏中“Window”菜单下的“Preferences”选项，如图3.11所示。选择左侧中的“Android”选项，在SDK Location中选择Android SDK的安装目录，则下面列表显示已安装可用的Android平台版本，至此开发环境已搭配完成，可以进行Android应用程序的开发了。

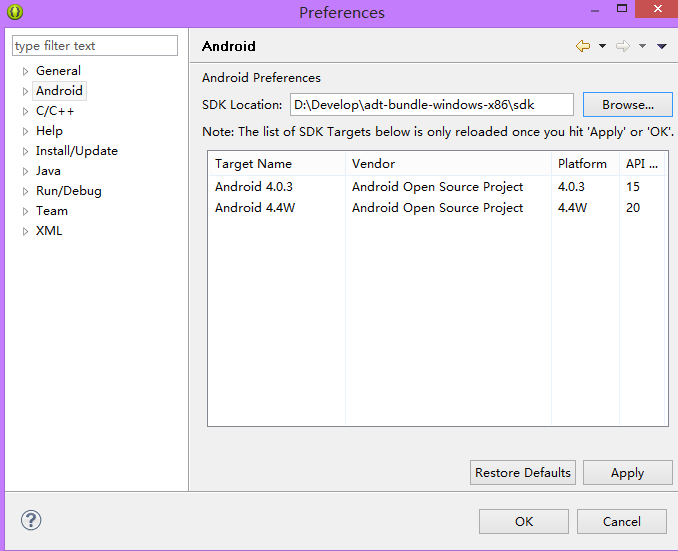


图3.11 配置ADT插件

3.4使用Eclipse开发Android应用

3.4.1 使用Eclipse创建Android工程

1）选择Eclipse菜单栏中的“File”菜单下的“New”选项中的“Android Application Project”选项，如图3.12所示。

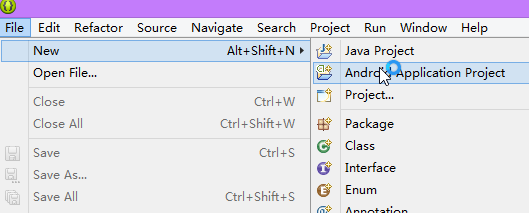


图3.12 新建Android工程

2）在弹出的新建Android工程对话框中，在“Application Name”中设置应用程序的名称，在“Project Name”中设置工程名称，在“Package Name”中设置包名，并选择相应的SDK版本，如图3.13所示。最后点击“Next”,直到“Finish”，完成Android工程的创建。

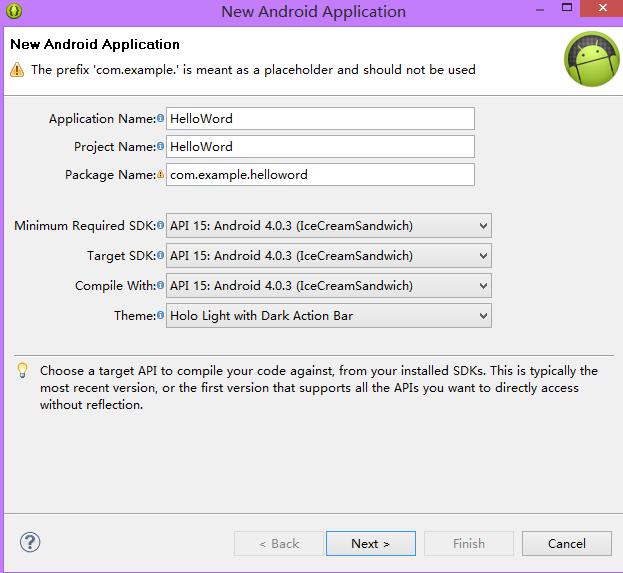


图3.13 新建Android工程对话框

3.4.2 Eclipse中的Android工程架构

在Eclipse的“Package Explorer”视图中，可以看到新建的Android工程的目录架构，如图3.14所示。

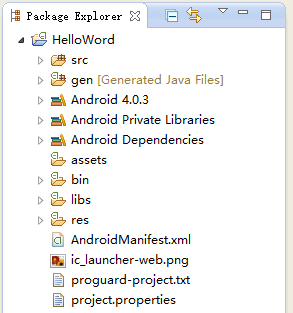


图3.14 Android工程架构

具体情况如下：

1）src文件夹：存放整个项目的JAVA源代码文件。

2）gen文件夹：存放编译器自动生成的一些JAVA代码文件，比如R.java文件，R.java是在建立项目时自动生成的，这个文件是只读模式的，不能更改。

3）Android 4.031文件夹：该文件夹下包含android.jar文件，其中包含构建应用程序所需的所有的Android SDK 库和API。

4）assets文件夹：包含应用程序需要使用到的视频和音频文件。

5）res文件夹：资源文件夹，包含你项目中的资源文件并将编译进应用程序。向此目录添加资源时，会被R.java自动记录。

6）AndroidManifest.xml：项目工程的总配置文件，用来配置应用程序中所使用的各种组件。

3.4.3 Eclipse中的Android工程的调试和运行

Android工程的代码经过编译无误后，就可以进行程序调试和运行了。鼠标右键单击项目工程名称，选择“Run As”选项中的“Android Application”选项，即可在默认的Android模拟器中运行相应的程序，HelloWord应用程序运行结果如图3.15所示。

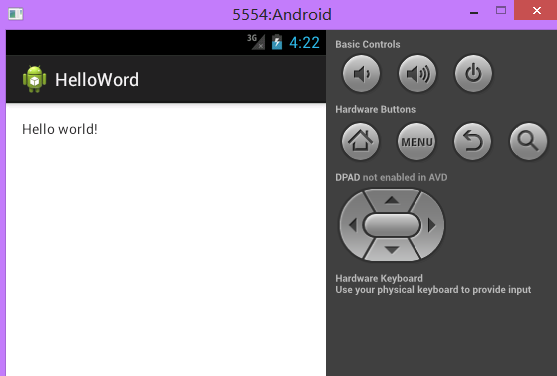


图3.15 Android应用程序运行结果

若要进行Android应用程序的调试，鼠标右键单击项目工程名称，选择“Debug As”选项中的“Android Application”选项，即可进入程序的调试过程。

# 第三章 客户端功能模块开发和实现

4.1客户端工程结构

分析并设计好报修客户端的功能需求后，就可以在搭配好Android开发环境的Eclipse程序开发工具里创建报修客户端Android工程了，本报修客户端的Android工程名称为：RepairApp。工程结构如图4.1所示。

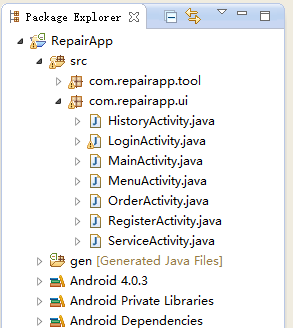


图4.1 报修客户端工程结构

其中com.repairapp.tool目录中包含的是各种工具类和封装的方法；而com.repairapp.ui目录中则是报修客户端各个功能的实现类，具体情况如下：

RegisterActivity.java：实现注册功能；

LoginActivity.java：实现登录功能；

MenuActivity.java：主菜单，用于选择故障报修或者查看报修记录；

ServiceActivity.java：实现故障报修功能；

HistoryActivity.java：实现查看报修记录功能；

在工程res目录下的layout文件里则是存放各个功能界面的布局文件，如图4.2所示。具体情况如下：

activity\_register：用户注册界面布局；

activity\_login：用户登录界面布局；

activity\_menu：主菜单界面布局；

activity\_service：故障报修界面布局；

activity\_history：查看报修记录界面布局；

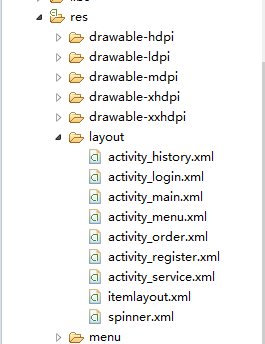


图4.2 报修客户端界面布局文件

4.2用户注册模块

用户注册模块是一个系统的开始，也是比较重要的一部分，是用户登录系统的前提。通过用户注册模块注册的用户信息，包括用户账号和用户密码，都将提交到服务器端的数据库中进行存储，以便用户登录系统时进行验证。注册界面的布局文件对应为layout文件夹下的activity\_register.xml文件，实现用户注册功能的程序为com.repairapp.ui目录下的RegisterActivity.java文件。用户注册界面如图4.3所示。在输入框输入账号和密码，点击“立即注册”便可以注册成功，注册信息将会被报修客户端提交到服务器端，通过服务器端写进数据库进行存储，以便用户进行系统登录时验证。其关键代码如下：

|  |
| --- |
| **public** **void** doRegister(String userName, String password) {  **try** {  // 创建一个HttpClient对象  HttpClient httpclient = **new** DefaultHttpClient();  //登录URL，向服务器提交用户注册信息：账号和密码  String url = HttpConfig.*register\_url* + "userName=" +  userName + "&password=" + password;  Log.*v*("url ", url);  // 创建HttpGet对象  HttpGet request = **new** HttpGet(url);  request.setHeader("Content-Type",  "application/x-www-form-urlencoded; charset=utf-8");  request.addHeader("Accept", "text/json");  // 获取响应的结果  HttpResponse response = httpclient.execute(request);  // 获取HttpEntity  HttpEntity entity = response.getEntity();  // 获取响应的结果信息  String json = EntityUtils.*toString*(entity, "UTF-8");  // JSON的解析过程  **if** (json != **null**) {  JSONObject jsonObject = **new** JSONObject(json);  result = jsonObject.get("message").toString();  }  **if** (result == **null**) {  json = "注册失败，请重新注册！";  } **else** {  json = "注册成功，请登录！";  }  } |



图4.3 用户注册界面

4.3用户登录模块

用户注册完成之后，必须先使用注册的账号进行登录才能使用报修系统，所以用户登录模块是用户使用系统的前提，是系统的门面。用户登录模块所要完成的功能是验证登录用户的账号是否正确，验证用户账号和密码若正确，则进入系统。用户登录界面的布局文件为layout文件夹下的activity\_login.xml文件，实现用户登录功能的程序为com.repairapp.ui目录下的LoginActivity.java文件。用户登录界面如图4.4所示。



图4.3 用户登录界面

其关键代码如下：

|  |
| --- |
| //登录按钮响应函数监听  btnLogin.setOnClickListener(**new** View.OnClickListener() {  @Override  **public** **void** onClick(View v) {  //获取用户输入的登录账号  String userName = txUserName.getText().toString();  //获取用户输入的账号密码  String password = txPassword.getText().toString();  //向服务器提交登录信息验证用户账号密码是否正确，正确则登录成功  loginRemoteService(userName, password);  }  }); |

4.4菜单界面模块

用户登录验证成功后即可以进入报修系统，用户登录系统后进入的是主菜单界面，主菜单界面主要是报修客户端功能按钮的设置以及地图容器的布局，主菜单界面如图4.4所示。菜单界面上设置有故障报修、报修记录和退出系统三个功能菜单，点击相应的菜单则会进入相应的功能界面。在三个功能菜单的下面则是用来加载地图的容器，当报修客户端对用户的当前位置进行GPS定位时，地图容器里则会加载地图，并显示出用户的当前位置。



图4.4 主菜单界面

4.5位置定位和显示模块

本报修系统的一个主要特点是可以对当前用户的位置进行GPS定位或者网络定位，采集其位置信息并在报修客户端上显示出用户的当前位置，这样可以方便用户了解当前所处位置，方便进行故障报修。所获取到的位置信息将提交到服务器端，通过服务器端存储到数据库中，以供管理中心的工作人员定位查看，方便安排维修人员前去维修。

本系统的地图显示和位置定位采用的基于Android的高德地图API技术开发实现的。高德定位服务包含GPS和网络定位（Wi-Fi和基站定位）两种能力，本系统采用的是高精度定位模式，即会同时使用网络定位和GPS定位，优先返回最高精度的GPS定位结果。这样不管用户是在室外还是在室内进行故障报修，报修客户端都可以获取当前用户的位置信息并进行位置定位。具体实现步骤如下：

1）到高德地图API官网注册成为开发者并申请API Key，然后下载高德地图Android SDK，并导入到报修客户端Android工程的“libs”文件夹下；

2）在报修客户端Android工程的“ AndroidManifest.xml ”文件中添加在高德地图API官网申请的用户Key,以及地图显示所需的权限代码，具体代码如下：

|  |
| --- |
| <meta-data  android:name=*"com.amap.api.v2.apikey"*  <!—添加申请到用户Key -->  android:value=*"4428ddde15c921294aa8e721534e218b"* />  <!-- 定位需要的服务 -->  <service  android:name=*"com.amap.api.location.APSService"* >  </service> |

3）在报修客户端主菜单界面加载高德地图并显示，其主要代码如下：

|  |
| --- |
| **private** **void** init() {  **if** (aMap == **null**) {  aMap = mapView.getMap();  //设置地图显示中心点和显示缩放比例  LatLng marker1 = **new** LatLng(32.11005,118.91667);  aMap.moveCamera(CameraUpdateFactory.*changeLatLng*(mar  ker1));  //aMap.moveCamera(CameraUpdateFactory.zoomTo(16));  setUpMap();  aMap.getUiSettings().setZoomControlsEnabled(**false**);  }  } |

4）对用户的当前位置进行定位，返回定位结果并在报修客户端的地图上显示出来，其主要代码如下：

|  |
| --- |
| **private** **void** setUpMap() {  // 自定义系统定位蓝点  MyLocationStyle myLocationStyle = **new** MyLocationStyle();  // 自定义定位蓝点图标  myLocationStyle.myLocationIcon(BitmapDescriptorFactory.  *fromResource*(R.drawable.*location\_marker*));  // 设置圆形的边框颜色  myLocationStyle.strokeColor(Color.*argb*(0, 0, 0, 0));  // 设置圆形的填充颜色 myLocationStyle.radiusFillColor(Color.*argb*(0, 0, 0, 0));  //自定义精度范围的圆形边框宽度  // 将自定义的 myLocationStyle 对象添加到地图上  aMap.setMyLocationStyle(myLocationStyle);  //设置定位资源。如果不设置此定位资源则定位按钮不可点击。  aMap.setLocationSource(**this**);  //设置默认定位按钮是否显示  aMap.getUiSettings().setMyLocationButtonEnabled(**true**);  // 设置为true表示显示定位层并可触发定位，false表示隐藏定位层并不可  触发定位，默认是false  aMap.setMyLocationEnabled(**true**);  } |

4.6故障报修模块

故障报修模块是整个报修客户端的核心模块，在该模块可以实现校园故障设备的报修。故障报修界面的布局文件为layout文件夹下的activity\_service.xml文件，实现用户设备故障报修的程序为com.repairapp.ui目录下的ServiceActivity.java文件。故障报修界面如图4.5所示。



图4.5 故障报修界面

在故障报修界面，当用户进行故障报修时，需要填写故障设备的报修信息，包括：报修事项、故障描述、报修地点、报修人员和联系电话。填写完毕后，点击“提交”按钮，报修客户端会把用户填写的报修信息以及报修客户端获取到的用户当前位置的经纬度信息一起发送到服务器端，由服务端接收数据后存储到数据库中，以便后勤管理中心的工作人员查看，安排维修人员前去维修，至此故障设备的报修工作已完成。其关键代码如下：

|  |
| --- |
| **public** **void** doRepair(String thing, String describe, String  address,String name, String tel, String time, String  longitude,String latitude, String username, String order) {  //把故障报修信息封装成JSON对象  Map<String, String> params = **new** HashMap<String, String>();  params.put("thing", thing);  params.put("describe", describe);  params.put("address", address);  params.put("name", name);  params.put("tel", tel);  params.put("time", time);  params.put("longitude", longitude);  params.put("latitude", latitude);  params.put("username", username);  params.put("order", order);  **try** {  String url = HttpConfig.*repair\_url*;  //返回报修结果bl,如果bl为true则报修成功，若为false则失败  bl = HttpRequest.*sendRequestFromHttpClient*(url,  params, "UTF-8");  } **catch** (Exception ee) {  // **TODO** Auto-generated catch block  ee.printStackTrace();  }  **if** (bl) {  //报修结果bl为true，报修成功  String string = "报修提交成功！";  alertClick(string);  } **else** {  //报修结果为false，报修失败  String string = "报修失败，请重新提交！";  alertClick(string);  } |

4.7报修记录模块

报修记录模块，该模块可供用户查看当前报修系统所登录的账号的所有报修记录，点击每条报修记录可查看该报修记录的详细报修信息。报修记录界面的布局文件为layout文件夹下的activity\_history.xml文件，实现用户查看报修记录的程序为com.repairapp.ui目录下的HistoryActivity.java文件。报修记录界面如图4.6所示。

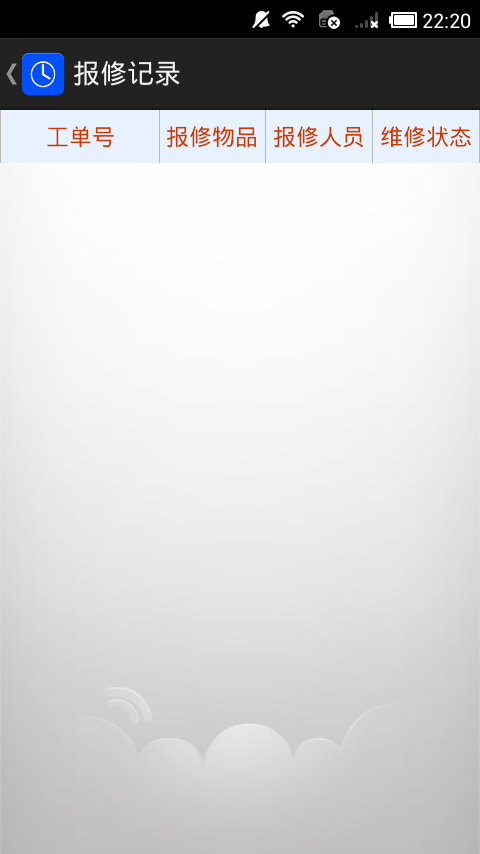


图4.6 报修记录界面

其关键代码如下：

|  |
| --- |
| **public** **void** orderJson(String url){  **try** {  // 得到Json解析成功之后数据  orders = JsonParse.*getListPerson*(url);  **for** (**int** i = 0; i < orders.size(); i++) {  HashMap<String, Object> map = **new** HashMap<String,  Object>();  map.put("orders", orders.get(i).getOrders());  map.put("thing", orders.get(i).getThing());  map.put("describes", orders.get(i).getDescribes());  map.put("address", orders.get(i).getAddress());  map.put("name", orders.get(i).getName());  map.put("tel", orders.get(i).getTel());  map.put("time", orders.get(i).getTime());  map.put("state", orders.get(i).getState());  data.add(map);  }  } **catch** (Exception e) {  Toast.*makeText*(HistoryActivity.**this**, "获取数据失败！",  2000).show();  Log.*i*(*TAG*, e.toString());  }  } |

4.8数据交互

报修客户端的界面和功能都已实现后，剩下的就是报修客户端与服务器端的通信和数据交互了。实现了报修客户端与服务器端的通信只有，报修客户端就可以向服务器端提交相应请求，以及向服务器发送用户账号信息和故障报修信息，实现整个系统的故障设备报修的工作。

Android客户端与服务器端的通信通常是采用HTTP通信方式和Socket通信方式，而在本系统中，报修客户端和服务器端之间是采用HTTP协议的通信方式进行通信，报修客户端与服务器端通信的内部原理如图4.7所示。

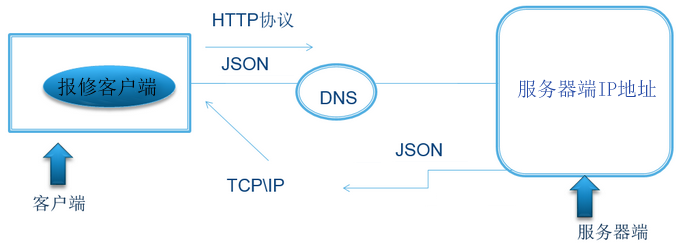


图4.7 报修客户端与服务器端通信

在报修客户端把要向服务器端提交的信息，封装成JSON对象数据格式，然后通过HTTP协议发送出去，网址经过DNS域名解析，解析成指定的服务器端IP地址，服务器端在相应的端口上监听报修客户端的请求。服务器监听到客户端的请求之后，接收并处理客户端提交的信息，并把请求结果以JSON对象形式返回给报修客户端，报修客户端在接收到服务器端返回的结果之后，对JSON对象进行解析，获取相应的信息，并在报修客户端上显示出来，完成报修客户端与服务器端的数据交互。其关键代码如下：

|  |
| --- |
| // HttpClient组件 SSL HTTPS Cookie  **public** **static** **boolean** sendRequestFromHttpClient(String path,  Map<String, String> params, String enc) **throws** Exception {  List<NameValuePair> paramPairs = **new** ArrayList<NameValuePair>();  **if** (params != **null** && !params.isEmpty()) {  **for** (Map.Entry<String, String> entry :  params.entrySet()) {  paramPairs.add(**new** BasicNameValuePair(entry.getKey(),  entry.getValue()));  }  }  // 得到经过编码过后的实体数据  UrlEncodedFormEntity entitydata = **new**  UrlEncodedFormEntity(paramPairs,enc);  HttpPost post = **new** HttpPost(path); // form  post.setEntity(entitydata);  // 创建http请求客户端  DefaultHttpClient client = **new** DefaultHttpClient();  // 执行请求  HttpResponse response = client.execute(post);  // 返回请求结果，如果返回值为200则请求提交成功  **if** (response.getStatusLine().getStatusCode() == 200) {  **return** **true**;  }  **return** **false**;  } |

# 第四章 客户端功能测试

5.1用户注册测试

用户在手机打开报修客户端应用程序后，首先进入到的是用户登录界面，若已注册账号，可直接在登录界面进行账号登录使用系统。若还没有账号，则可点击登录界面底部的“新用户注册”按钮跳转到用户注册界面进行注册。在注册界面需要用户输入账号和密码，账号为邮箱或者手机号，如果邮箱或者手机号输入有误，则客户端会提示“邮箱或者手机号格式不正确”；同时注册用户需要输入两次注册密码，如果两次密码输入不一致将不能进行注册，会提示重新填写，如果用户输入信息不完整或有遗漏的，则会提示填写完整。如果用户注册成功，客户端则会显示服务器端返回的信息“注册成功，请登录！”，点击“确定”按钮则跳转到登录界面进行登录。用户注册测试如图5.1所示。



图5.1 用户注册测试

5.2用户登录测试

用户注册了账号之后，可在登录界面进行系统登录，登录成功则可以使用报修系统进行相关的功能操作。在登录界面，用户填写了账号和密码之后，点击“登录”按钮，则报修客户端会把用户的账号和密码提交到服务器端进行账号和密码的校验，若校验不成功，则服务器返回“账号或密码错误，请重新登录！”信息给报修客户端并提示用户，若通过校验，则系统登录成功，并提示用户“登录成功”，点击“确定”按钮则会跳转到系统的主菜单界面，在主菜单界面则可以进行相应系统功能的选择。用户登录测试如图5.2所示。



图5.2 用户登录测试

5.3位置定位和显示测试

位置定位模块也是整个系统很重要的一个模块，该模块可以在用户使用报修客户端进行故障报修的时候获取用户当前的位置信息，并提交到服务器端。这样既可以使得用户可以确定自己当前所处的位置，方便进行故障报修；同时又可以在服务端对到达该故障报修点进行路线规划，方便维修人员前去维修，这样可以大大提高故障维修效率。用户位置的定位和显示是在主菜单界面进行的，位置定位和显示测试如图5.3.所示。



图5.3 位置定位和显示测试

5.4故障报修测试

在主菜单界面有着三个功能按钮，分别是故障报修、报修记录和退出系统。点击“退出系统”则会退出报修客户端，若想再次使用则需重新登录。点击“故障报修”按钮则会进入到故障报修界面，在故障报修界面，当用户进行故障设备报修时，需要填写相关的故障报修信息，包括：报修事项、故障描述、报修地点、报修人员和联系电话等。用户填写完毕之后，点击“提交”按钮，若用户填写的信息不全或者有遗漏的，报修客户端则会提示相应信息不能为空，请填写完整。若信息填写无误，则报修客户端会把故障报修信息提交到服务器端，若服务器端成功接收报修客户端提交的故障报修信息并处理后，则会返回“报修提交成功”给报修客户端并提示用户，若服务器端接收不到报修客户端提交的故障报修信息，则会返回“故障报修失败”给报修客户端。故障报修功能测试如图5.4所示。



图5.4 故障报修功能测试

5.5报修记录测试

在报修记录模块，用户可以查询当前登录系统的账号的所有报修记录以及对应报修记录的详细信息，这样在用户对故障设备进行报修后，可以方便用户跟踪查询设备的维修进度。在主菜单界面点击“报修记录”按钮就会跳转到报修记录界面，同时报修客户端会向服务器端提交查询报修记录的请求，服务器端在接收到报修客户端的请求后，会从数据库中读取相关信息并返回给报修客户端。在客户端的报修记录界面就会以列表的形式显示所有报修记录的简要信息，包括：工单号、报修物品、报修人员和维修状态，查询报修记录结果如图5.5所示。



图5.5 报修记录查询结果

在图5.5的报修记录查询结果列表中，点击其中一条报修记录就会显示该报修记录的详细信息，包括：报修物品、故障描述、报修地点、报修人员、联系电话、报修时间、报修工单和维修状态，报修记录详细信息查询结果如图5.6所示。



图5.6 报修记录详细信息