**桂林理工大学**

**GUILIN UNIVERSITY OF TECHNOLOGY**

**生产实习**

**《详细设计说明书》**

**学 院： 信息科学与工程学院 #**

**专 业： 软件工程 #**

**班 级： 2016-1班 #**

**项目名称： 鸿鹄导航（Swan）**

**组 员1：曾鑫3162052051539**

**组 员2：刘笑枫3162052051534**

**组 员3：徐仕成3162052051433##**

**指导教师：**

**评 分： #**

**评 语：**

**2019年10月15日**

目录

[《详细设计说明书》 3](#_Toc23547125)

[说明 3](#_Toc23547126)

[完成时间 3](#_Toc23547127)

[撰写人 3](#_Toc23547128)

[1.具体模块设计 3](#_Toc23547129)

[1.1地图显示模块 3](#_Toc23547130)

[1.2注册模块 5](#_Toc23547131)

[1.3登录模块 7](#_Toc23547132)

[1.4忘记密码模块 9](#_Toc23547133)

[1.5地点搜索模块 11](#_Toc23547134)

[1.6路线规划模块 13](#_Toc23547135)

[1.7导航模块 15](#_Toc23547136)

《详细设计说明书》

说明： 本详细设计说明书用于系统设计阶段，明确在设计过程中的各模块的开发与模块之间的关系，便于后续的维护。

完成时间：2019年10月11日-2019年10月15日

撰写人： 徐仕成、刘笑枫、曾鑫

（备注：本说明书在系统设计阶段根据实际设计情况不断完善和补充）

# 1.具体模块设计

## 1.1地图显示模块

第一步：配置AndroidManifest.xml

首先，声明权限

//地图SDK（包含其搜索功能）需要的基础权限

<!--允许程序打开网络套接字-->

<uses-permission android:name="android.permission.INTERNET" />

<!--允许程序设置内置sd卡的写权限-->

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

<!--允许程序获取网络状态-->

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

<!--允许程序访问WiFi网络信息-->

<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />

<!--允许程序读写手机状态和身份-->

<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />

<!--允许程序访问CellID或WiFi热点来获取粗略的位置-->

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

然后，设置高德Key

在application标签中加入如下内容：

<meta-data  
 android:name="com.amap.api.v2.apikey"  
 android:value="开发者申请的key" />

第二步：向工程中添加地图开发包

通过Gradle集成SDK：

在Project的build.gradle文件中配置repositories，添加maven或jcenter仓库地址,

配置如下：

allprojects {

repositories {

jcenter() // 或者 mavenCentral()

}

}

在主工程的build.gradle文件配置dependencies

配置如下：

android {

defaultConfig {

ndk {

//设置支持的SO库架构（开发者可以根据需要，选择一个或多个平台的so）

abiFilters "armeabi", "armeabi-v7a", "arm64-v8a", "x86","x86\_64"

}

}

}

dependencies {

compile fileTree(dir: 'libs', include: ['\*.jar'])

//3D地图so及jar

compile 'com.amap.api:3dmap:latest.integration'

//定位功能

compile 'com.amap.api:location:latest.integration'

//搜索功能

compile 'com.amap.api:search:latest.integration'

}

第三步，初始化地图容器

首先在布局xml文件中添加地图控件:

<com.amap.api.maps.MapView

android:id="@+id/map"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"/>

第四步，显示地图

//定义了一个地图view

mapView = (MapView) findViewById(R.id.map);

mapView.onCreate(savedInstanceState);// 此方法须覆写，虚拟机需要在很多情况下保存地图绘制的当前状态。

//初始化地图控制器对象

AMap aMap;

if (aMap == null) {

aMap = mapView.getMap();

}

显示地图：见图1-1 显示地图



图1-1 显示地图

## 1.2注册模块

第一步：在app/src/main/res/layout/activity\_register.xml中实现注册页面的布局

布局架构：

<?xml version="1.0" encoding="utf-8"?>

<android.support.constraint.ConstraintLayout>

<ScrollView>

<LinearLayout>

<com.qmuiteam.qmui.widget.QMUITopBar>

<ImageView/>

<TextView/>

</com.qmuiteam.qmui.widget.QMUITopBar>

<!--头部的logo和标题-->

<LinearLayout>

<ImageView/>

<TextView/>

</LinearLayout>

<!--注册信息输入框-->

<LinearLayout>

<!--账号文字-->

<TextView/>

<!--账号输入框-->

<EditText/>

<!--密码文字-->

<TextView/>

<!--密码输入框-->

<EditText/>

<!--确认密码文字-->

<TextView/>

<!--确认密码输入框-->

<EditText/>

<!--验证码文字-->

<TextView/>

<LinearLayout>

<!--验证码输入框-->

<EditText/>

<!--图形验证码-->

<ImageView/>

</LinearLayout>

</LinearLayout>

<!--注册按钮-->

<Button/>

</LinearLayout>

</ScrollView>

</android.support.constraint.ConstraintLayout>

第二步：在app/src/main/java/com/example/swan/activity/RegisterActivity.java中添加引用布局文件的代码：

protected void onCreate(@Nullable Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_register);

getSupportActionBar().hide();

}

第三步：在app/src/main/AndroidManifest.xml中添加注册文件：

<activity

android:name=".activity.RegisterActivity"

android:parentActivityName=".activity.LoginActivity" />

注册页面显示：见图1-2注册页面显示



图1-2 注册页面显示

第四步：在app/src/main/java/com/example/swan/activity/RegisterActivity.java中实现注册验证的处理，代码较长，见系统源码。

## 1.3登录模块

第一步：在app/src/main/res/layout/activity\_login.xml中实现登录页面的布局

布局架构:

<?xml version="1.0" encoding="utf-8"?>

<android.support.constraint.ConstraintLayout>

<ScrollView>

<LinearLayout>

<com.qmuiteam.qmui.widget.QMUITopBar>

<ImageView/>

<TextView/>

</com.qmuiteam.qmui.widget.QMUITopBar>

<!--头部的logo和标题-->

<LinearLayout>

<ImageView/>

<TextView/>

</LinearLayout>

<!--登录信息输入框-->

<LinearLayout>

<TextView/>

<EditText/>

<LinearLayout>

<TextView/>

<TextView/>

</LinearLayout>

<EditText/>

<Button/>

</LinearLayout>

<!--新用户注册-->

<LinearLayout>

<TextView/>

<TextView/>

</LinearLayout>

</LinearLayout>

</ScrollView>

</android.support.constraint.ConstraintLayout>

第二步：在app/src/main/java/com/example/swan/activity/LoginActivity.java中添加引用布局文件的代码：

引用代码：

protected void onCreate(@Nullable Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_login);

getSupportActionBar().hide();

}

第三步：在app/src/main/AndroidManifest.xml中注册文件：

<activity

android:name=".activity.LoginActivity"

android:parentActivityName=".activity.MenuActivity" />

登录页面显示：见图1-3 登录页面显示

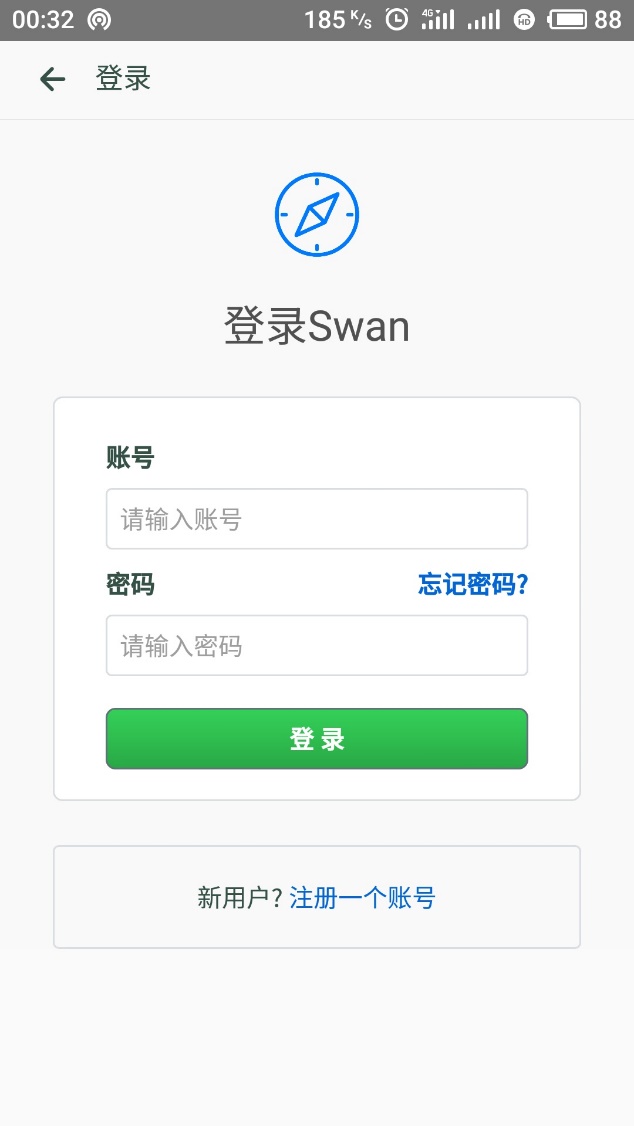


图1-3 登录显示

第四步：在app/src/main/java/com/example/swan/activity/LoginActivity.java中进行登录验证处理，代码较长，可参见系统源码。

## 1.4忘记密码模块

第一步：app/src/main/res/layout/activity\_forget\_pwd.xml中实现忘记密码页面布局

布局架构：

<?xml version="1.0" encoding="utf-8"?>

<android.support.constraint.ConstraintLayout>

<ScrollView>

<LinearLayout>

<com.qmuiteam.qmui.widget.QMUITopBar>

<ImageView/>

<TextView/>

</com.qmuiteam.qmui.widget.QMUITopBar>

<!--头部的logo和标题-->

<LinearLayout>

<ImageView/>

<TextView/>

</LinearLayout>

<!--修改密码信息输入框-->

<LinearLayout>

<!--账号文字-->

<TextView/>

<!--账号输入框-->

<EditText/>

<!--新密码文字-->

<TextView/>

<!--新密码输入框-->

<EditText/>

<!--确认新密码文字-->

<TextView/>

<!--确认新密码输入框-->

<EditText/>

</LinearLayout>

<!--修改密码按钮-->

<Button/>

</LinearLayout>

</ScrollView>

</android.support.constraint.ConstraintLayout>

第二步：在app/src/main/java/com/example/swan/activity/ForgetPwdActivity.java中添加引用布局文件的代码：

protected void onCreate(@Nullable Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_forget\_pwd);

getSupportActionBar().hide();

}

第三步：在app/src/main/AndroidManifest.xml中添加注册文件：

<activity

android:name=".activity.ForgetPwdActivity"

android:parentActivityName=".activity.LoginActivity" />

第四步：在app/src/main/java/com/example/swan/activity/ForgetPwdActivity.java中实现处理忘记密码的逻辑，具体代码参加系统源码。

忘记密码页面显示：见图1-4 忘记密码显示



图1-4 忘记密码页面显示

## 1.5地点搜索模块

第一步：在app/src/main/res/layout/activity\_search.xml中实现搜索框的显示，布局架构：

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout>

<com.qmuiteam.qmui.widget.QMUIWindowInsetLayout>

<com.qmuiteam.qmui.widget.QMUITopBar>

<SearchView>

</SearchView>

</com.qmuiteam.qmui.widget.QMUITopBar>

</com.qmuiteam.qmui.widget.QMUIWindowInsetLayout>

<!-- 清空历史记录 -->

<LinearLayout>

<ListView/>

<TextView/>

</LinearLayout>

</LinearLayout>

在app/src/main/java/com/example/swan/activity/SearchActivity.java中调整搜索框的样式：

TextView searchText = (TextView) searchView.findViewById(searchView

.getContext()

.getResources()

.getIdentifier("android:id/search\_src\_text", null, null));

searchText.setTextSize(TypedValue.COMPLEX\_UNIT\_SP, 14);

searchText.setTextColor(Color.parseColor("#365146"));

searchText.setHintTextColor(Color.parseColor("#9e9e9e"));

第二步：在app/src/main/java/com/example/swan/activity/SearchActivity.java中引用布局文件：

setContentView(R.layout.activity\_search);

getSupportActionBar().hide();

第三步：在app/src/main/AndroidManifest.xml中注册文件：

<activity android:name=".activity.SearchActivity" />

第四步：在app/src/main/java/com/example/swan/activity/SearchActivity.java实现搜索的逻辑处理，代码较长，请参见系统源码。

地点搜索显示：见图1-5 地点搜索显示



图1-5 地点搜索显示

## 1.6路线规划模块

第一步：在app/src/main/res/layout/main\_bottom.xml中实现显示路线的可点击按钮

布局架构：

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout>

<LinearLayout>

<!-- 搜附近 -->

<TextView/>

<View/>

<!-- 导航 -->

<TextView/>

<View/>

<!-- 路线 -->

<TextView/>

</LinearLayout>

</RelativeLayout>

第二步：在app/src/main/java/com/example/swan/route/RouteActivity.java中实现显示路线的布局：

private RelativeLayout mBottomLayout;

mBottomLayout = (RelativeLayout) findViewById(R.id.bottom\_layout);

第三步：在app/src/main/AndroidManifest.xml中注册文件：

<activity android:name=".route.RouteActivity" />

第四步：在app/src/main/java/com/example/swan/route/RouteActivity.java中设置监听事件，同时进行一系列的数据处理，主要包括：

初始化 RouteSearch 对象：

mRouteSearch = new RouteSearch(this);

设置数据回调监听器：

mRouteSearch.setRouteSearchListener(this);

设置搜索参数：

DriveRouteQuery query = new DriveRouteQuery(fromAndTo, mode, null,

null, "");// 第一个参数表示路径规划的起点和终点，第二个参数表示驾车模式，第三个参数表示途经点，第四个参数表示避让区域，第五个参数表示避让道路

发送请求：

mRouteSearch.calculateDriveRouteAsyn(query);// 异步路径规划驾车模式查询

接收数据：

public void onDriveRouteSearched(DriveRouteResult result, int errorCode) {

//具体代码参加系统源码

}

路线规划功能演示：见图1-6路线按钮显示、图1-7驾车路线显示、图1-8 路线详情显示、图1-9 公交路线显示、图1-10 步行路线显示

图1-6 路线按钮显示 图1-7 驾车路线显示



图1-8 路线详情显示 图1-9 公交路线显示



图1-10 步行路线显示

## 1.7导航模块

第一步：在app/src/main/res/layout/activity\_navi.xml中定义AmapNaviView：

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<com.amap.api.navi.AMapNaviView

android:id="@+id/navi\_view"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_weight="1"/>

</LinearLayout>

第二步：实现 AMapNaviView 的生命周期

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

//获取 AMapNaviView 实例

mAMapNaviView = (AMapNaviView) findViewById(R.id.navi\_view);

mAMapNaviView.setAMapNaviViewListener(this);

mAMapNaviView.onCreate(savedInstanceState);

}

@Override

protected void onResume() {

super.onResume();

mAMapNaviView.onResume();

}

@Override

protected void onPause() {

super.onPause();

mAMapNaviView.onPause();

}

@Override

protected void onDestroy() {

super.onDestroy();

mAMapNaviView.onDestroy();

}

第三步：进行路线规划

第四步：开启导航

public void onCalculateRouteSuccess() {

super.onCalculateRouteSuccess();

mAMapNavi.startNavi(NaviType.GPS);

}

导航页面显示：见图1-11 导航显示

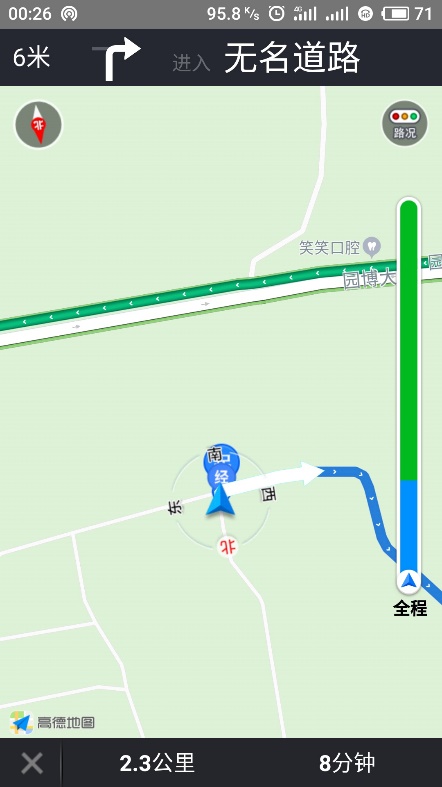


图1-11 导航显示