软件工程课程设计

《4D GIS标记系统项目》

软件设计说明书

设计组学生:

姓名 姚翛潇（组长） 学号1120161821

姓名 朱婧婧 学号1120161826

姓名 谢蜜雪 学号1120161761

姓名 杨冰琪 学号1120161762

姓名 杨 俊 学号1120161820

姓名 霍萱甫 学号1120162099

姓名 刘楠彬 学号1120162132

指导教师: 赵刚

北京理工大学计算机学院

2019年10月

**4D GIS标记系统**

4D GISMS (4D GIS Marking System)

**《软件设计说明书》**

目录

[1、引言 4](#_Toc23348173)

[1.1编写目的 4](#_Toc23348174)

[1.2项目背景 4](#_Toc23348175)

[1.3设计目标 4](#_Toc23348176)

[1.4运行环境 5](#_Toc23348177)

[2、技术路线 6](#_Toc23348178)

[3、数据结构设计 6](#_Toc23348179)

[3.1数据关系建模 6](#_Toc23348180)

[3.2 数据结构建模 7](#_Toc23348181)

[3.3 数据库建模 11](#_Toc23348182)

[4、功能设计 13](#_Toc23348183)

[4.1 数据流图 13](#_Toc23348184)

[4.1.1 第0层 13](#_Toc23348185)

[4.1.2 第1层 14](#_Toc23348186)

[4.1.2 第2层 15](#_Toc23348187)

[4.1.2.1 即时查询模块 15](#_Toc23348188)

[4.1.2.2 信息标记模块 16](#_Toc23348189)

[4.1.2.3 图层保存与查看模块 17](#_Toc23348190)

[4.1.2.4数据库管理模块 18](#_Toc23348191)

[4.2 功能实现 19](#_Toc23348192)

[4.2.1登录注册 19](#_Toc23348193)

[4.2.2 视图切换 20](#_Toc23348194)

[4.2.3 即时查询 22](#_Toc23348195)

[4.2.4 离线数据同步 24](#_Toc23348196)

[4.2.5 标记信息分享 26](#_Toc23348197)

[4.2.6 信息标记 28](#_Toc23348198)

[4.2.7 线路标记 28](#_Toc23348199)

[4.2.8 图层保存与查看 30](#_Toc23348200)

[4.2.9 信息发布 33](#_Toc23348201)

[4.2.10 好友管理 34](#_Toc23348202)

[5、界面设计 36](#_Toc23348203)

[5.1 登录/注册和密码找回界面 36](#_Toc23348204)

[5.2主界面默认展示 38](#_Toc23348205)

[5.3 即时查询界面 38](#_Toc23348206)

[5.4 数据上传界面 40](#_Toc23348207)

[5.5 分享界面 41](#_Toc23348208)

[5.6 添加标记/路线界面 42](#_Toc23348209)

[5.7 数据库管理界面 44](#_Toc23348210)

[5.8 个人信息修改界面 47](#_Toc23348211)

[6、重要技术难点及解决方案 48](#_Toc23348212)

[6.1 在线聊天 48](#_Toc23348213)

[6.2 即时查询 50](#_Toc23348214)

[6.3 离线数据同步 50](#_Toc23348215)

[6.4 标记信息分享 51](#_Toc23348216)

[6.5 信息标记 53](#_Toc23348217)

[6.6 线路标记 54](#_Toc23348218)

[6.7 图层保存与查看 55](#_Toc23348219)

[6.8 信息发布 58](#_Toc23348220)

# 1、引言

## 1.1编写目的

该文档负责说明软件的详细设计，为编码和测试工作提供依据。根据软件需求分析的结 果，对项目的各个子系统和功能进行模块划分，并对其界面、模块、系统结构间的关系进行了设计与描述。这个阶段的任务解决了软件系统总体结构设计的问题，包括整个软件系统的结构、模块划分、模块功能和模块间的联系等。还要解决 如何实现各个模块的内部功能，即模块设计。模块的设计会为各子系统设计算法，是整个软件开发过程中所有后续工作的指导。

本文档面向多种读者对象：

（1）软件编写人员：了解数据结构，数据形式以及需实现的功能，据此进行软件编写。

（2）软件测试人员：根据此文档编写测试用例，对软件产品进行功能性测试。

## 1.2项目背景

项目名称：4D GIS标记系统

近来，行业普遍认为GIS将朝着普适化、智能化方向发展。在未来，任何人都可以在任何地方，拿着任意终端访问GIS服务，而不局限于特定用户和专业的终端，希望使普通用户也能够通过多媒介进行访问。随着云计算技术、移动终端、物联网等技术的快速发展，这一想法是极有可能实现的。因此，我们希望创造一个GIS环境，把大家的知识和经验用地图的方式来表达，让用户非常方便的获得地图数据，满足他们对于出行、旅游、测绘、科研等方面的需求。

## 1.3设计目标

本系统的设计说明书主要分为四个部分来完成：功能设计、数据结构设计、界面设计及程序伪代码的实现。功能设计部分将完成本系统的功能建模，包括软件的类图、数据流图等；数据结构设计部分讲完成软件的数据建模，其中包括数据的ER图、数据结构定义和数据库表的设计；界面设计部分将完成简要的界面设计和接口模块的设计。后续的软件开发编码人员和测试人员可根据此设计说明书进行编写。

## 1.4运行环境

* **客户端程序：**

客户端程序可以运行在装有 Android 操作系统或IOS的智能手机、平板电脑上，web网页以及个人笔记本电脑上，需要满足的具体要求如下：

Android系统：

|  |  |
| --- | --- |
| 操作系统 | Android 4.1  以上 |
| 处理器 | 高通骁龙  200 以上 |
| 运行RAM | 1G 以上 |
| 存储空间 | 50MB 以上 |
| 其他要求 | 支持 GPS 导航 |

IOS系统：

|  |  |
| --- | --- |
| 操作系统 | IOS 7.0  以上 |
| 处理器 | Apple A7 以上 |
| 运行RAM | 1G 以上 |
| 存储空间 | **50MB** 以上 |
| 其他要求 | 支持 GPS 导航 |

Web及PC端：

|  |  |
| --- | --- |
| 操作系统 | Windows |
| 运行RAM | 4G 以上 |
| 存储空间 | 100MB 以上 |

* **服务器程序：**

服务器程序运行在安装有Windows 操作系统的服务器上，需要满足的具体要求如下：

|  |  |
| --- | --- |
| 操作系统 | Windows 7以上 |
| 数据库 | MySQL |
| 处理器 | Intel |
| RAM | 8GB以上 |
| 数据存储空间 | 512GB以上 |
| 网络 | 支持TCP/IP |

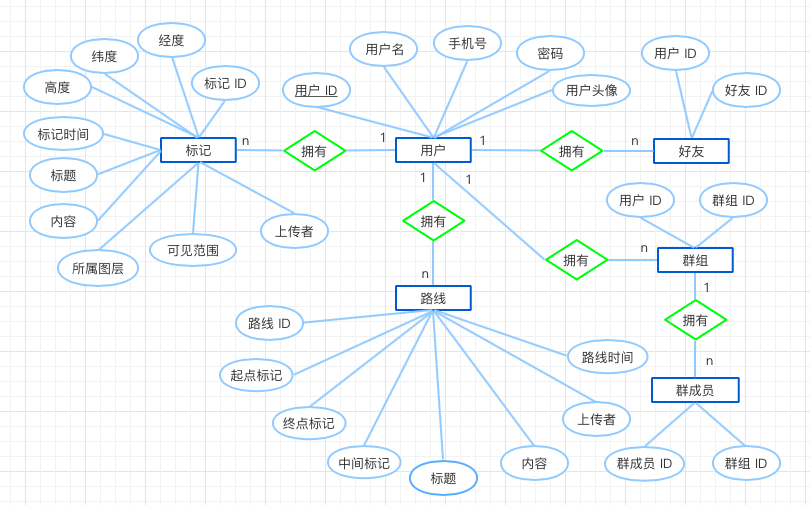
## 2、技术路线

本系统在设计时采用C/S的架构进行设计。产品设计分层按服务器端和客户端之间的不同来进行。其中服务器端使用apache服务器采用微服务器的架构进行设计，数据库使用SQL Server 2012数据库，编程语言使用Java。客户端使用多平台独立开发App的策略，在ios端和mac os端使用swift进行开发，在android端使用kotlin进行开发，windows端使用.Net开发框架进行开发。外部的数据接口我们采用两种策略，当显示的部分不要求相当精细时，使用百度地图的外部接口进行；当内容要求精细时，尤其是要显示地图的3D信息、地理信息等时，使用Arc GIS。

# 3、数据结构设计

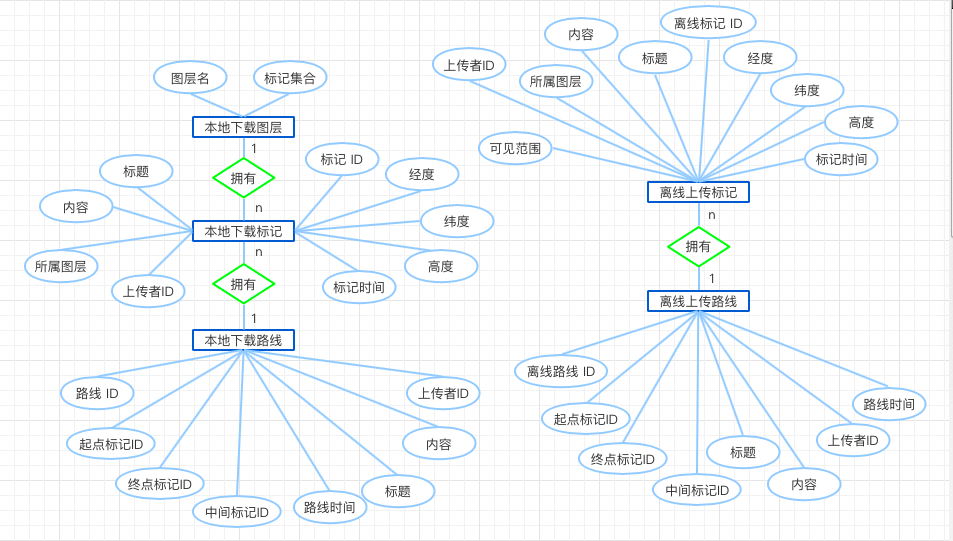
## 3.1数据关系建模

本系统服务器部分的数据ER 图如下图所示：



从图中可以看出本软件的数据部分由六个实体构成：用户、好友、群组、群成员、标记、路线，每个实体有不同的内容构成，每个实体之间的关系如图中所描述。

本地数据库的 ER 图如下：



## 3.2 数据结构建模

* 服务器端端：

用户表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **字段说明** | **类型** | **长度** | **属性** |
| 用户ID | 系统生成 | int | / | NOT NULL, UNIQUE, PRIMARY KEY |
| 用户名 | 用户注册填写 | varchar | 20 | NOT NULL, UNIQUE |
| 手机号 | 用户注册提供 | char | 11 | NOT NULL, UNIQUE |
| 密码 | 用户注册填写 | varchar | 16 | Number or letter, NOT NULL |
| 用户头像 | 用户填写 | image | / | / |

好友列表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **字段说明** | **类型** | **长度** | **属性** |
| 用户ID | 用户添加好友时系统获取 | INT | / | NOT NULL, FOREIGN KEY |
| 好友ID | 用户添加好友时用户指定 | INT | / | NOT NULL, PRIMARY KEY, FOREIGN KEY |

群组列表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **字段说明** | **类型** | **长度** | **属性** |
| 用户ID | 用户创建或加入群组时系统获取 | INT | / | NOT NULL, FOREIGN KEY |
| 群组ID | 用户创建或加入群组时用户指定 | INT | / | NOT NULL, PRIMARY KEY, FOREIGN KEY |

群成员列表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **字段说明** | **类型** | **长度** | **属性** |
| 群组ID | 系统获取 | INT | / | NOT NULL, PRIMARY KEY |
| 群成员ID | 系统获取 | INT | / | NOT NULL, PRIMARY KEY, FOREIGN KEY |

标记表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **字段说明** | **类型** | **长度** | **属性** |
| 标记ID | 系统生成 | INT | / | NOT NULL, UNIQUE, PRIMARY KEY |
| 经度 | 系统自动获取 | FLOAT | / | NOT NULL |
| 维度 | 系统自动获取 | FLOAT | / | NOT NULL |
| 高度 | 系统自动获取 | FLOAT | / | / |
| 标记时间 | 系统自动获取 | DATE | / | / |
| 标题 | 用户填写 | VARCHAR | 20 | 默认为“标记+标记ID“ |
| 内容 | 用户填写 | VARCHAR | 200 | / |
| 所属图层 | 用户指定 | ENUM | / | / |
| 可见范围 | 用户指定 | ENUM | / | 默认为所有人可见 |
| 上传者ID | 系统获取 | INT | / | NOT NULL, FOREIGN KEY |

路线表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **字段说明** | **类型** | **长度** | **属性** |
| 路线ID | 系统生成 | INT | / | NOT NULL, UNIQUE, PRIMARY KEY |
| 起点标记ID | 系统查询获得 | INT | / | NOT NULL, FOREIGN KEY |
| 终点标记ID | 系统查询获得 | INT | / | NOT NULL, FOREIGN KEY |
| 中间标记ID | 系统查询生成 | SET | / | FOREIGN KEY |
| 标题 | 用户填写 | VARCHAR | 20 | 默认为“路线+路线ID” |
| 内容 | 用户填写 | VARCHAR | 200 | / |
| 上传者ID | 系统自动获取 | INT | / | NOT NULL, FOREIGN KEY |
| 路线时间 | 系统自动获取 | DATE | / | / |

* 用户本地数据库：

本地下载标记表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **字段说明** | **类型** | **长度** | **属性** |
| 标记ID | 系统获取 | INT | / | NOT NULL, UNIQUE, PRIMARY KEY |
| 经度 | 系统获取 | FLOAT | / | NOT NULL |
| 维度 | 系统获取 | FLOAT | / | NOT NULL |
| 高度 | 系统获取 | FLOAT | / | / |
| 标记时间 | 系统获取 | DATE | / | / |
| 标题 | 系统获取 | VARCHAR | 20 | / |
| 内容 | 系统获取 | VARCHAR | 200 | / |
| 所属图层 | 系统获取 | ENUM | / | / |
| 上传者ID | 系统获取 | INT | / | NOT NULL |

本地下载路线表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **字段说明** | **类型** | **长度** | **属性** |
| 路线ID | 系统获取 | INT | / | NOT NULL, UNIQUE, PRIMARY KEY |
| 起点标记ID | 系统获取 | INT | / | NOT NULL, FOREIGN KEY |
| 终点标记ID | 系统获取 | INT | / | NOT NULL, FOREIGN KEY |
| 中间标记ID | 系统获取 | SET | / | FOREIGN KEY |
| 标题 | 系统获取 | VARCHAR | 20 | / |
| 内容 | 系统获取 | VARCHAR | 200 | / |
| 上传者ID | 系统获取 | INT | / | NOT NULL |
| 路线时间 | 系统获取 | DATE | / | / |

本地下载图层表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **字段说明** | **类型** | **长度** | **属性** |
| 图层名 | 用户指定 | ENUM | / | NOT NULL, FOREIGN KEY |
| 标记集合 | 系统获取 | SET | / | FOREIGN KEY |

离线上传标记表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **字段说明** | **类型** | **长度** | **属性** |
| 离线标记ID | 系统生成 | INT | / | NOT NULL, AUTO\_INCREMENT, UNIQUE, PRIMARY KEY |
| 经度 | 系统获取 | FLOAT | / | NOT NULL |
| 维度 | 系统获取 | FLOAT | / | NOT NULL |
| 高度 | 系统获取 | FLOAT | / | / |
| 标记时间 | 系统获取 | DATE | / | / |
| 标题 | 用户指定 | VARCHAR | 20 | / |
| 内容 | 用户指定 | VARCHAR | 200 | / |
| 所属图层 | 用户指定 | ENUM | / | / |
| 可见范围 | 用户指定 | ENUM | / | 默认为所有人可见 |
| 上传者ID | 系统获取 | INT | / | NOT NULL |

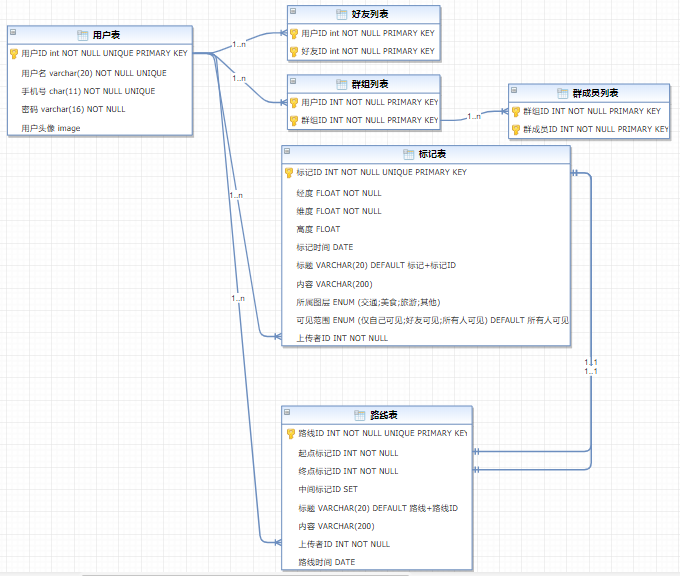
离线上传路线表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **字段说明** | **类型** | **长度** | **属性** |
| 离线路线ID | 系统生成 | INT | / | NOT NULL, UNIQUE, AUTO\_INCREMENT, PRIMARY KEY |
| 起点标记ID | 系统生成 | INT | / | NOT NULL, FOREIGN KEY |
| 终点标记ID | 系统生成 | INT | / | NOT NULL, FOREIGN KEY |
| 中间标记ID | 系统生成 | SET | / | FOREIGN KEY |
| 标题 | 用户指定 | VARCHAR | 20 | / |
| 内容 | 用户指定 | VARCHAR | 200 | / |
| 上传者ID | 系统获取 | INT | / | NOT NULL |
| 路线时间 | 系统获取 | DATE | / | / |

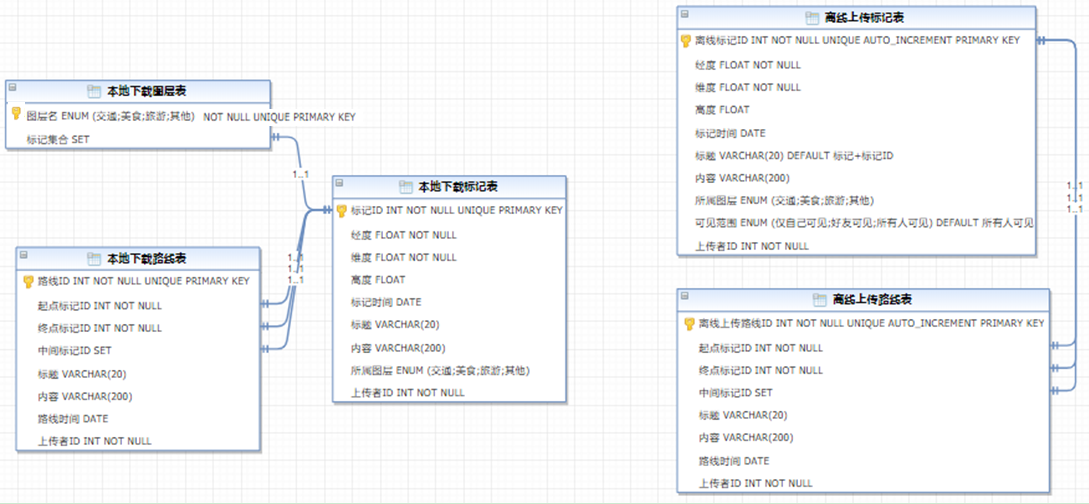
## 3.3 数据库建模

主要根据ER图和数据结构来进行数据库建模，数据库设计结果如下：

* 服务器端数据库：

****

* 用户本地数据库：

****

# 4、功能设计

## 4.1 数据流图

### 4.1.1 第0层

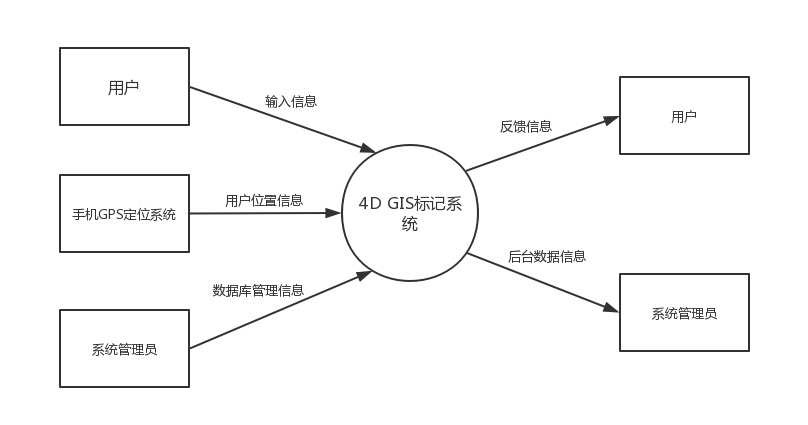


图1 第0层DFD图

顶层数据流图描绘了整体的用户（包括管理员）和软件之间的交互过程。本层图说明了这个系统面向的对象主要是用户。系统的输入由用户、手机GPS定位系统和系统管理员产生。用户产生的输入数据流是“输入信息”； 手机GPS定位系统产生的输入数据流是“用户位置信息”； 系统管理员产生的输入数据流是“数据库管理信息”。输出由用户和系统管理员接受。分别接受系统输出的“反馈信息”和“后台数据信息”。

### 4.1.2 第1层

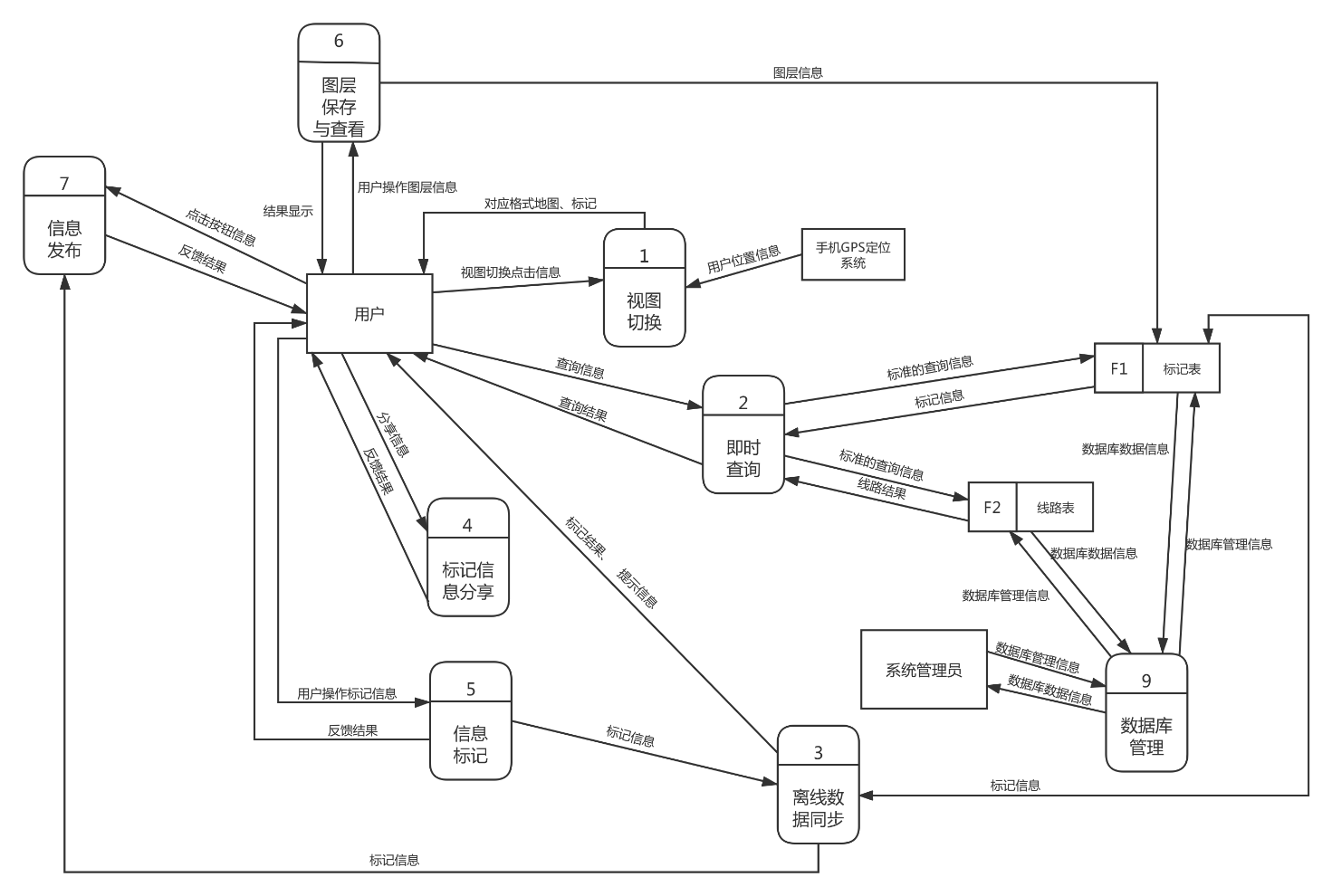


图2 第1层DFD图

第一层根据用户输入信息的不同将处理过程划分为九个部分：视图切换、即时查询、离线数据同步、标记信息分享、信息标记、信息发布、图层保存与查看、数据库管理。

### 4.1.2 第2层

### 4.1.2.1 即时查询模块

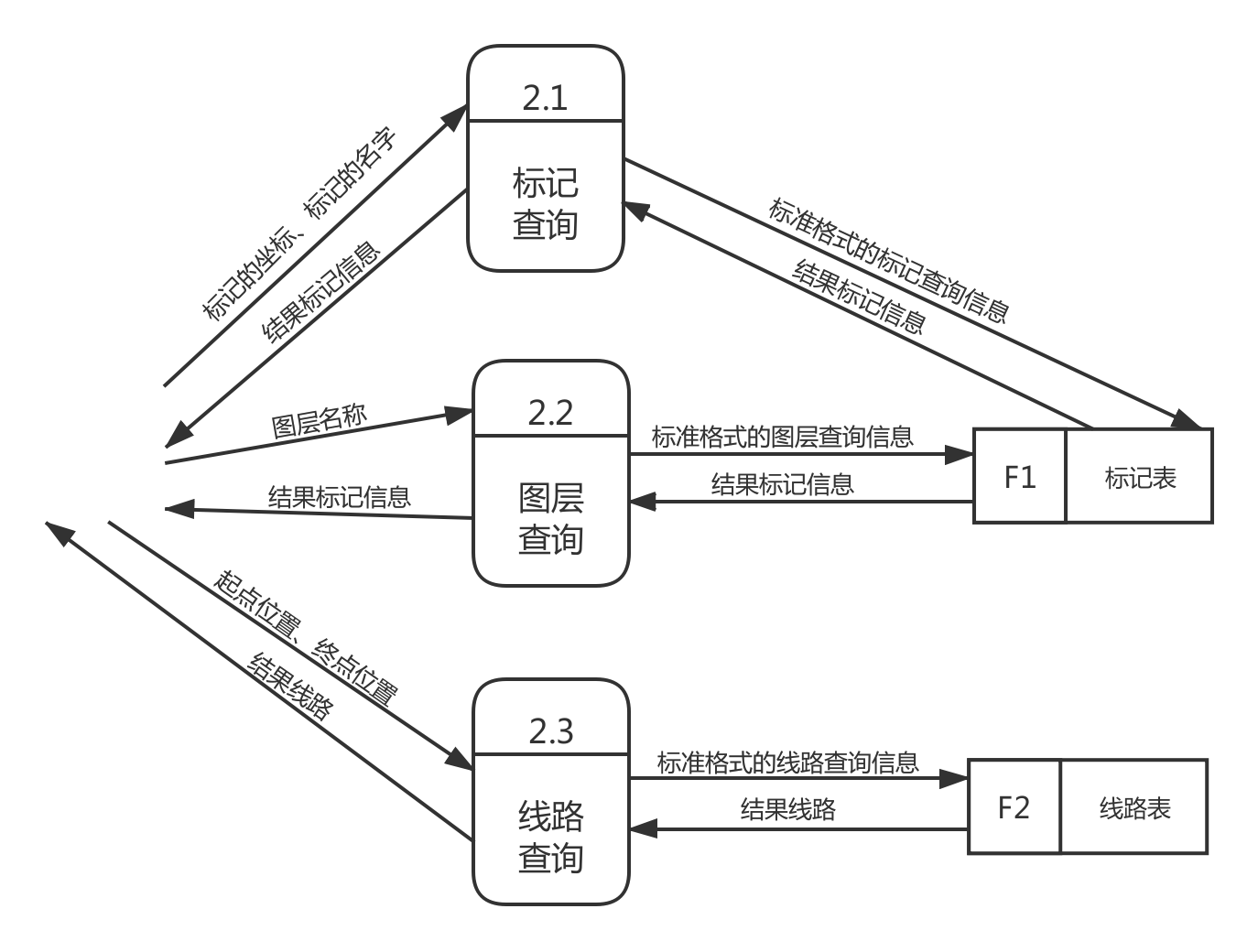


图3 即时查询模块的数据流图

即时查询模块是4D GIS系统较为重要的一部分。它分为标记查询、图层查询和线路查询。标记查询即根据用户输入的标记坐标或者标记名称在标记表中查询该标记的所有信息，并将该标记显示在地图上；图层查询即根据用户输入的图层名称在标记表中查询该图层的所有标记并显示在地图上；线路查询输入起点和终点位置信息，在线路表中查询一条符合的线路并显示在地图上。

### 4.1.2.2 信息标记模块

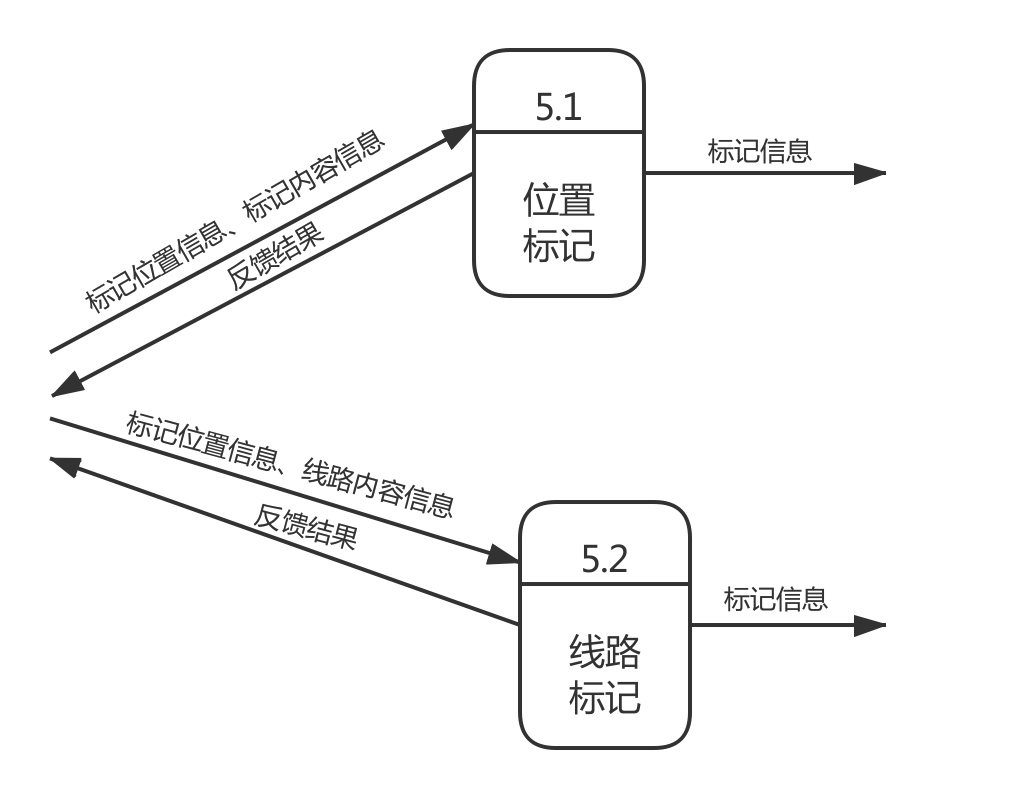


图4 信息标记模块的数据流图

信息标记模块分为位置标记和线路标记。位置标记是将用户输入的标记坐标和标记相关信息存储到标记表中。线路标记是将用户点击的若干个标记点以及用户输入的线路名称将其存放到线路表中。

### 4.1.2.3 图层保存与查看模块

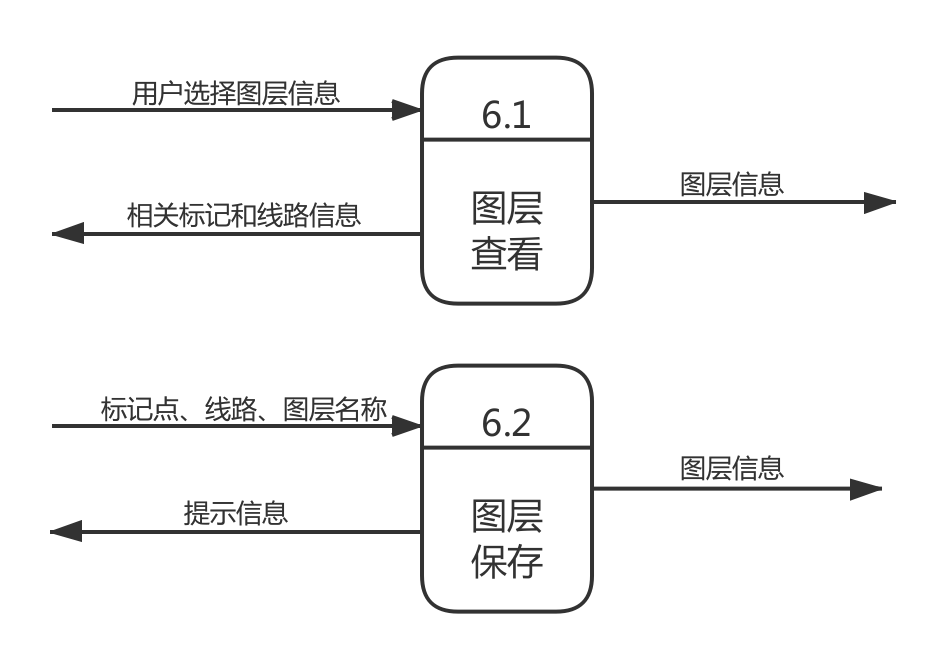


图5 图层保存与查看模块的数据流图

信息标记模块分为图层查看和图层保存。图层查看是用户选择系统已经设置好的某个图层，然后地图上显示该图层的标记和线路。图层保存是将用户选择的标记点或者线路的图层属性设置为某个图层。

### 4.1.2.4数据库管理模块

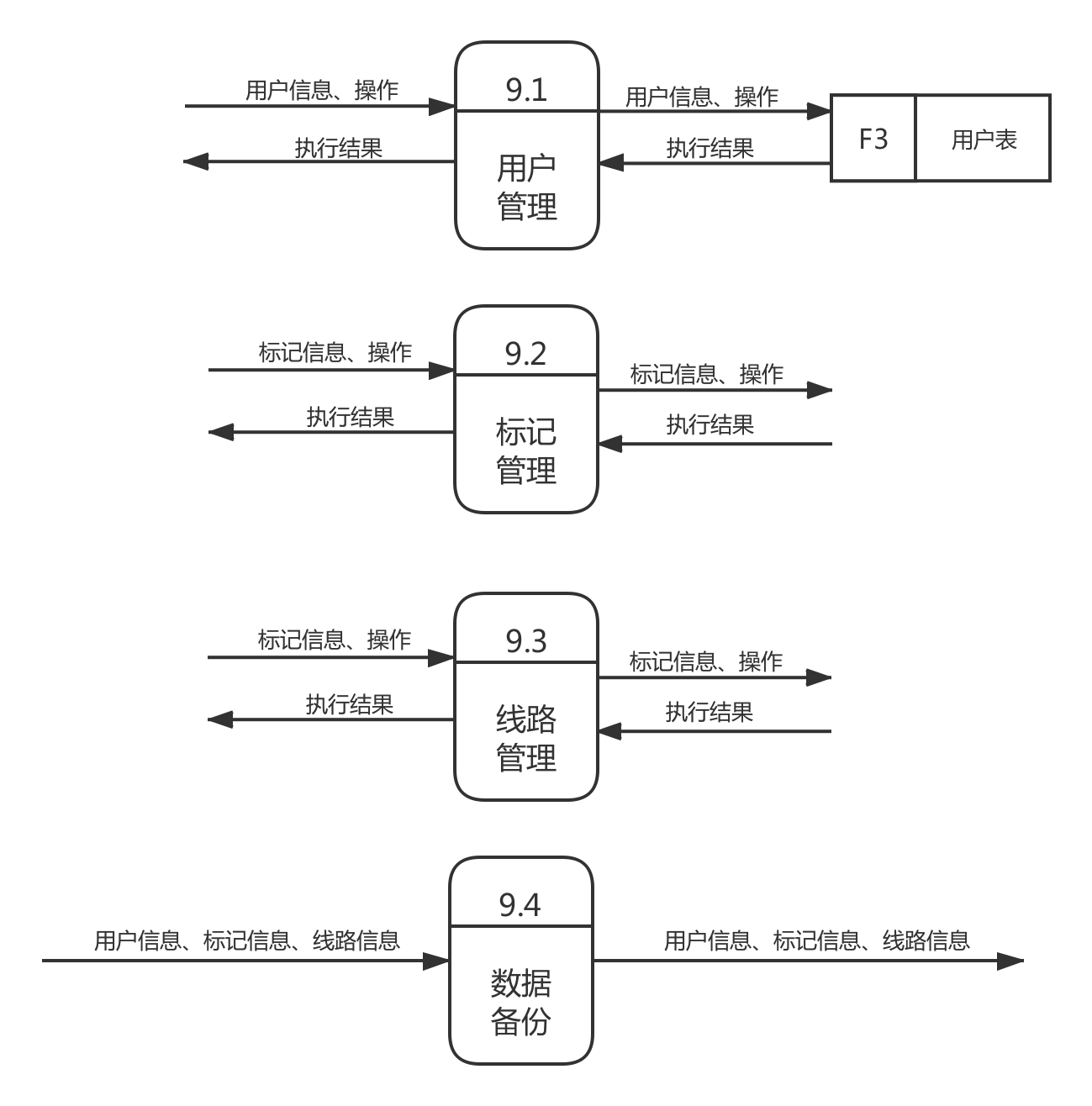


图6 数据库管理模块的数据流图

数据库管理模块分为用户管理、标记管理、线路管理和数据备份。用户管理是对用户表进行增删改查，数据库返回操作的结果。标记管理是对标记表进行增删改查，数据库返回操作的结果。线路管理是对线路表进行增删改查，数据库返回操作的结果。数据备份是每隔固定的时间，对数据库中所有数据进行一次备份。

## 4.2 功能实现

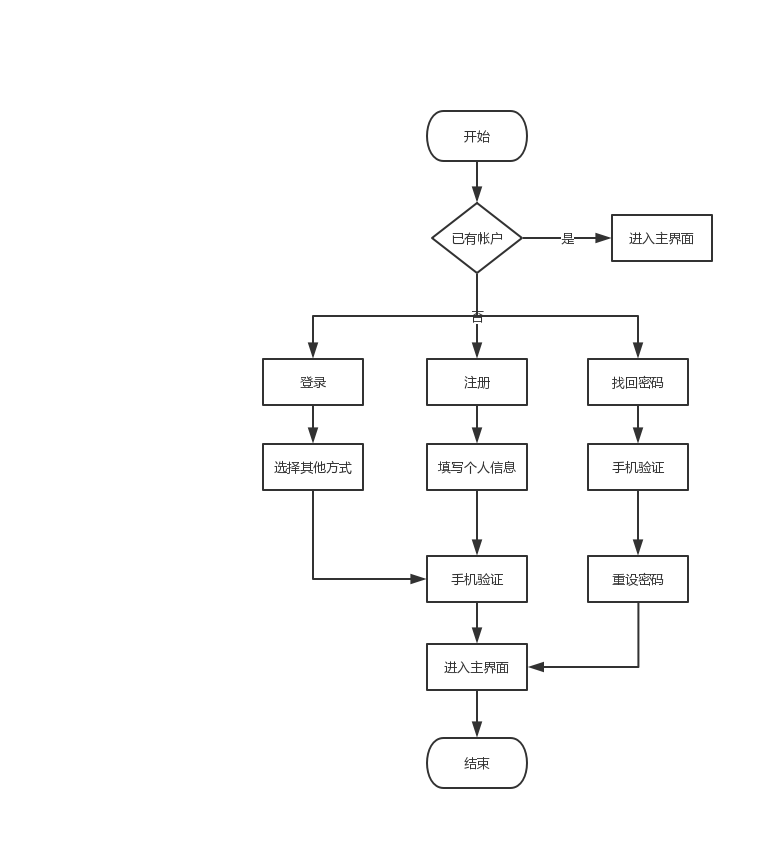
### 4.2.1登录注册

登录注册模块主要包括登录、注册和找回密码三个功能。用户第一次打开软件时，默认界面为登录界面，此时用户可以选择使用其他端口账号登录，如微信、QQ、微博等。当选择以这些账号登录时，会获取权限，并将该账号的头像，昵称等基本资料复制到本软件的账号上，此时用户只需要输入手机号、设置密码、填入发送过去的验证码即可，手机号即为用户账号。若用户进入注册界面，此时用户需要填写部分个人信息，再以手机号和密码进行注册。

当用户再次打开软件时，会直接以上一次登录的账号直接登录，可以退出账号重新以其他账号进行登录。

当用户忘记密码时可以选择找回密码，通过手机号的短信验证后即可重新设置密码。

登录注册的流程图如下图所示：



登陆注册模块共有一个用户信息类：

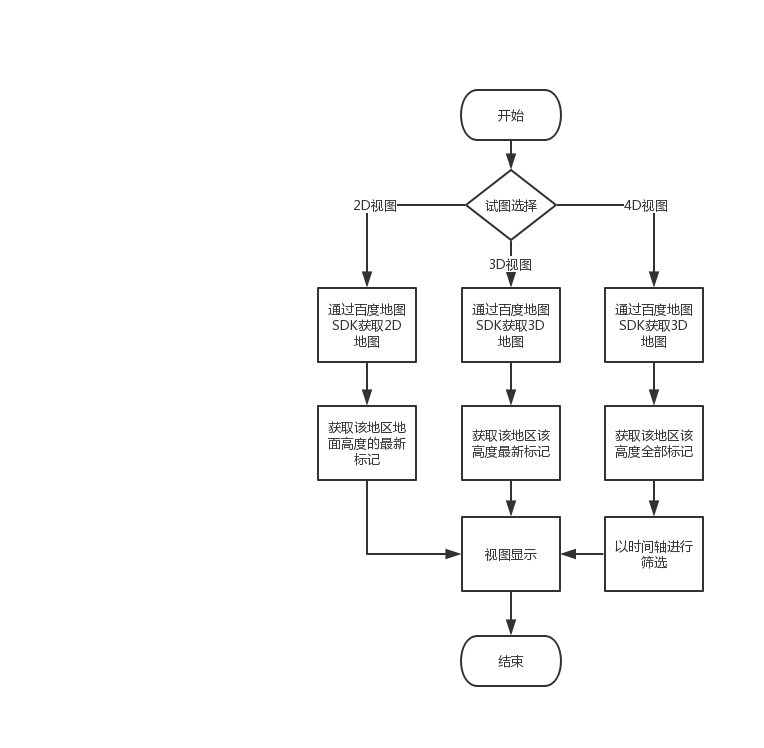


### 4.2.2 视图切换

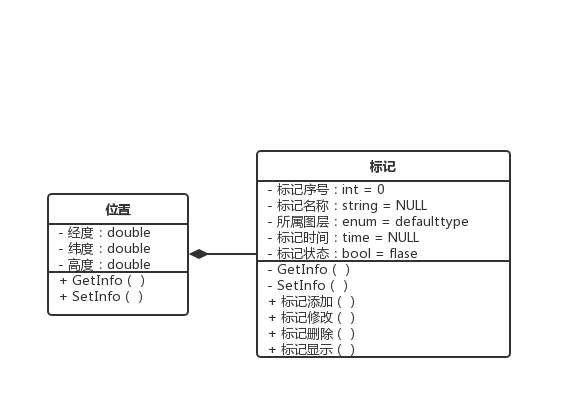
视图切换共包括3个功能，分别是切换2D视图、切换3D视图、切换4D视图。其显示时只有2D和3D地图可显示，4D需要加一个坐标轴来实现。

用户可以通过选择“2D”、“3D”和“4D”标识来切换视图。

视图切换的流程图如下图所示：



视图切换模块共有两个类，分别是标记类、位置类。各类的实现及它们之间的关系如下：



视图切换需要通过百度地图SDK获取地图位置信息，并筛选标记进行显示。该部分对应的接口如下表所示：

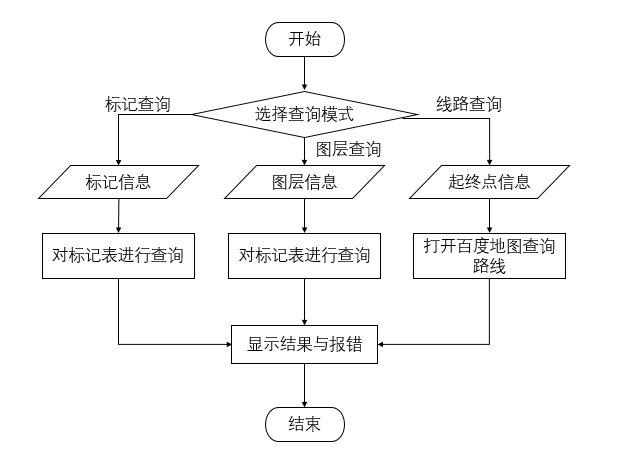
|  |  |  |  |
| --- | --- | --- | --- |
| **接口名称** | **参数** | **返回值** | **说明** |
| get\_baidu\_map\_info() | Null | MapView | 根据当前位置进行选择，返回该部分地图 |
| select\_tag() | enum pos | enum tag | 根据当前位置的坐标进行选择，返回tag列表 |

### 4.2.3 即时查询

即时查询可根据三种输入参数进行查询，输入标记信息及图层信息将返回从数据库中查询的标记，输入起终点为位置信息将在客户端以高德地图查询地铁路线的格式，显示线路。

当用户想对所需信息进行查询时，首先进入查询页面，再选择三种查询模式中的一种，输入查询信息进行搜索及结果显示。

图片 1 即时查询流程图



表格 1 即时查询接口

|  |  |  |  |
| --- | --- | --- | --- |
| **接口名称** | **参数** | **返回值** | **说明** |
| query\_tag\_send() | string query\_tag\_info | bool | 输入标记的信息（经度，维度，高度，时间）或标记的名字。  对标记表执行查询操作，将查询到的标记信息进行格式转换以及维度扩充 ,在客户端地图上展示从数据库中查询到的标记。返回是否查询成功。 |
| query\_layer\_send() | string query\_layer\_info | bool | 输入图层名称。对标记表执行查询操作，将查询到的标记信息进行格式转换以及维度扩充，在地图上显示。返回是否查询成功。 |
| Query\_liner\_send() | string query\_begin\_place,query\_end\_place | bool | 输入起点及终点信息。对线路表执行查询操作，并按照路程进行升序排序，将查询到的线路进行格式转换，在客户端显示。返回是否查询成功。 |

### 4.2.4 离线数据同步

离线数据同步共包括4个功能，分别是离线标记、标记逐条上传、在线集中上传和本地数据同步。用户在添加标记后，可以选择“提交”或“提交并上传”功能，若选择“提交”，则仅将标记保存在本地数据库，该 标记为离线标记，显示“未上传”按钮标识。

当用户想上传某条标记时，点击“未上传”按钮，进行上传，上传时会根据网络状态判断上传是否成功并返回给用户。

用户可以长按标记进入选择模式，可选择多条或全选标记，进行集中上传。

用户下拉界面，更新信息，自动进行本地数据与服务端数据同步（参考QQ消息刷新功能）。

离线数据同步的流程图如下图所示：

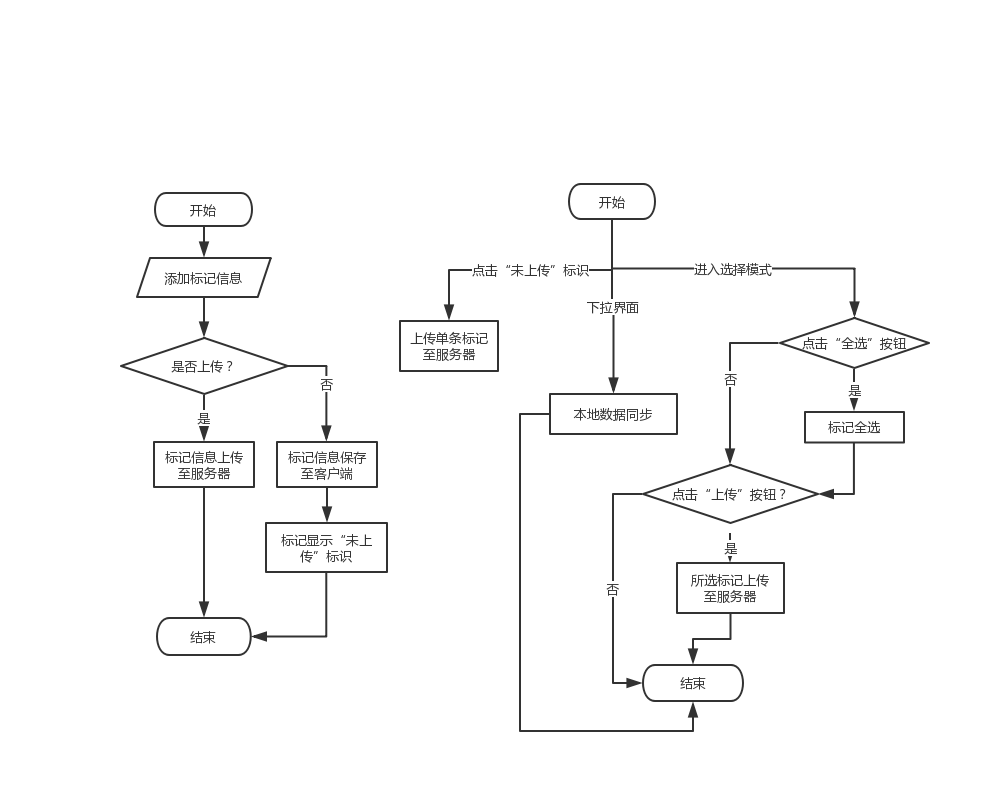


图 1 离线数据同步流程图

离线数据同步模块共有4个类，分别是离线标记类、标记类、位置类、上传信息类。各类的实现及它们之间的关系如下：

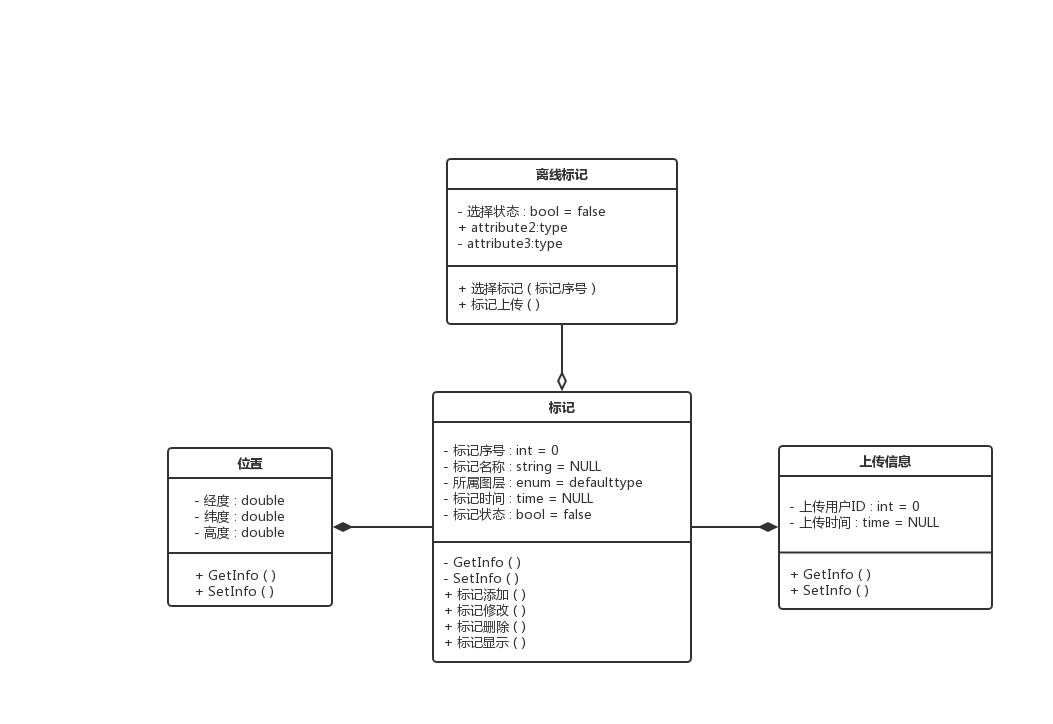


图 2 离线数据同步类图

离线数据同步需要显示离线标记信息，并对选择上传的标记进行上传操作。该部分对应的接口如下表所示：

表格 1 离线数据同步接口

|  |  |  |  |
| --- | --- | --- | --- |
| **接口名称** | **参数** | **返回值** | **说明** |
| show\_offline\_tag() | int offline\_tag\_id | bool | 根据离线标记的id显示离线比较（包括“未上传”标识按钮），返回是否显示成功 |
| upload\_offline\_tag() | int select\_tag\_id | bool | 根据选择标记的id将标记上传至服务器 |

### 4.2.5 标记信息分享

标记信息分析包括标记分享和线路分享两个部分。当用户想向第三方平台分享标记时，长按标记进入选择模式，选择一条或多条标记，点击“分享”按钮，选择想要分享的第三方平台，添加分享文字信息，分享标记；线路同理。

标记信息分享的流程图如下图所示：

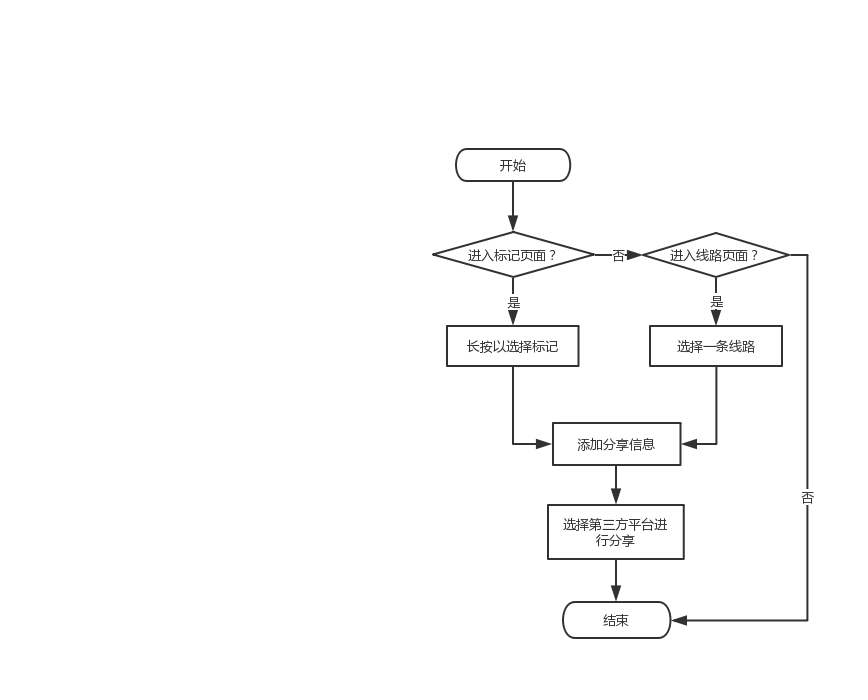


图 3 标记信息分享流程图

标记信息分享模块共有6个类，分别是分享类、分享标记类、分享线路类、链路类、位置类、上传信息类。各类的实现及它们之间的关系如下：

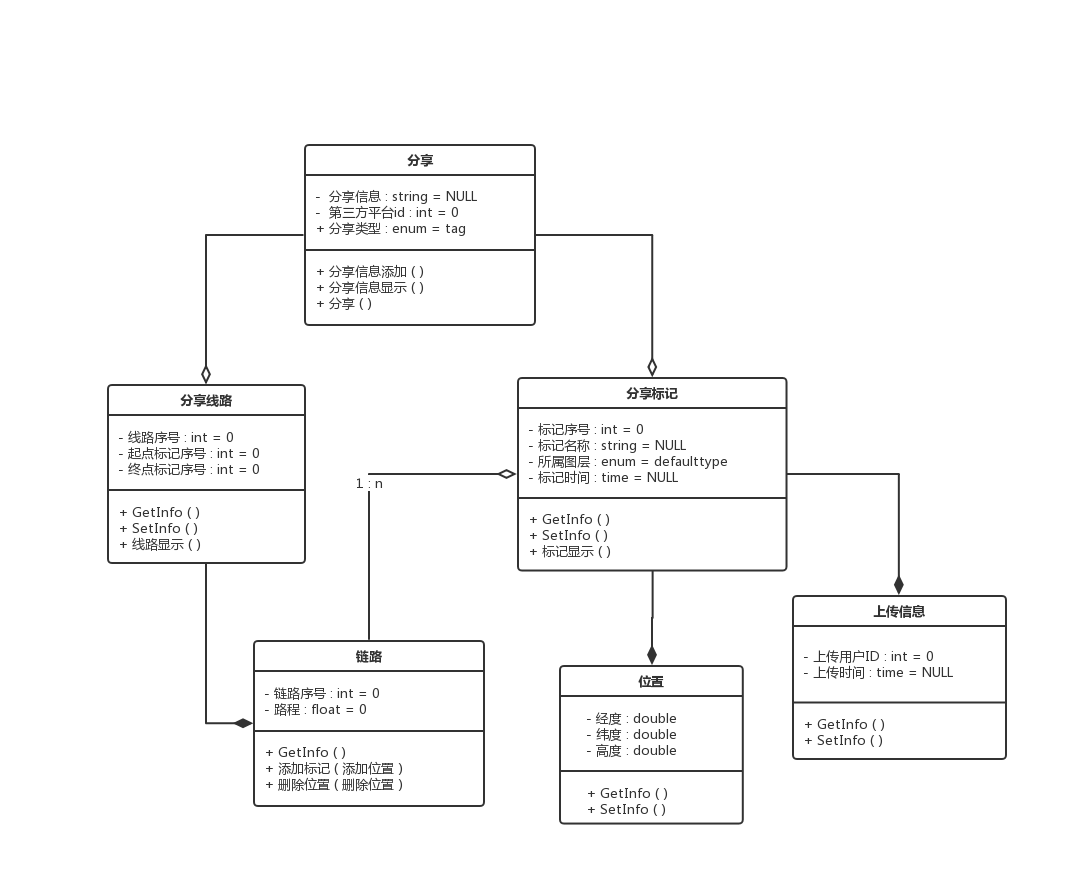


图 4 标记分享类图

标记信息分享需要选择要分享的标记或路线，添加分享信息，选择第三方平台并分享。该部分对应的接口如下表所示：

表格 2 标记分享接口

|  |  |  |  |
| --- | --- | --- | --- |
| **接口名称** | **参数** | **返回值** | **说明** |
| select\_share\_tag() | int select\_tag\_id | int | 根据选择标记的id进行选择，返回其id |
| select\_share\_line() | int select\_line\_id | int | 根据选择线路的id进行选择，返回其id |
| add\_share\_info() | string share\_info | bool | 添加分享内容，返回是否添加成功 |
| select\_platform() | int platform\_id | int | 根据第三方平台的id进行选择，返回其id |
| share() | int share\_platform\_id | bool | 将分享类中的实例化内容分享至所选平台，返回是否成功 |

### 4.2.6 信息标记

用户可以在地图的指定位置进行信息标记，点击“添加标记”按钮，便可以对指定位置进行标记，并可以在标记处添加具体标记内容。

位置标记的流程图如下图所示：

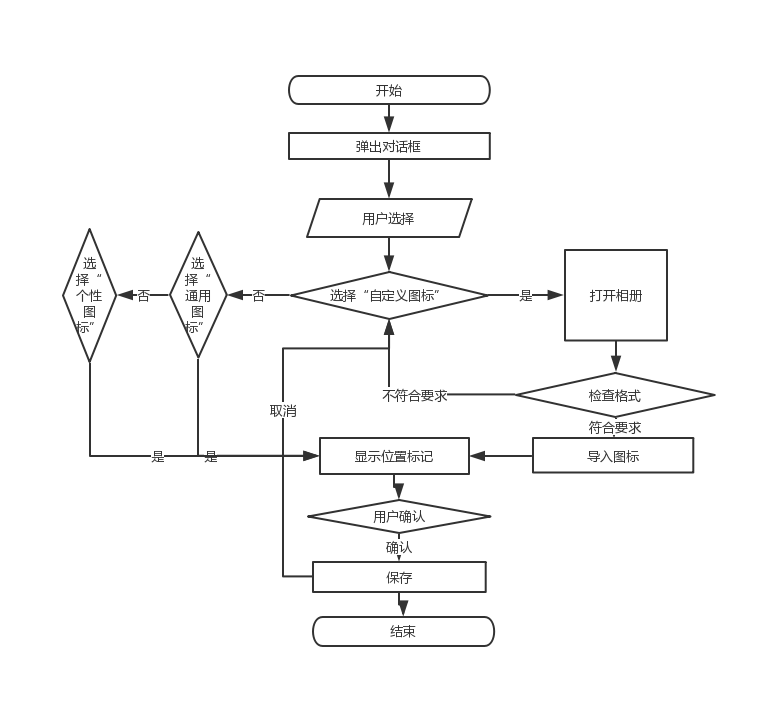


图1 位置标记流程图

### 4.2.7 线路标记

用户可以在地图上标记路线。点击“添加路线”按钮，依次点击线路上的标记点，然后系统将其依次连接，保存为路线。

线路标记的流程图如下图：

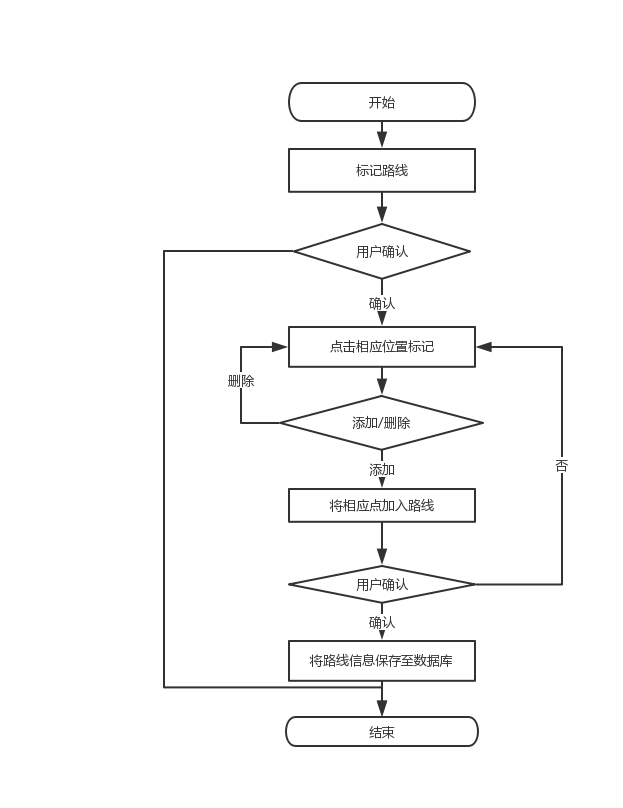


图2 路线标记流程图

信息标记需要选择指定位置，添加位置标记和内容标记，其接口信息如下表所示：

表格1 信息标记模块接口

|  |  |  |  |
| --- | --- | --- | --- |
| **接口名称** | **参数** | **返回值** | **说明** |
| select\_location() | double& longitude,  double& latitude,  double& height | void | 将该位置的位置信息保存在引用类型的数据中 |
| add\_location\_tag() | void | int | 在该位置新建一个location\_tag对象，并添加位置标记信息，返回该标记序号 |
| add\_content\_tag() | void | int | 在该位置新建一个content\_tag对象，并添加内容标记信息，返回该标记序号 |

### 4.2.8 图层保存与查看

用户可以将指定标记点或者标记路线保存为特定图层。用户可以查看本地保存的“交通图层”、“旅游图层”和“美食图层”等，以及查看其他用户通过网络分享的图层。

图层保存的流程图如下图：

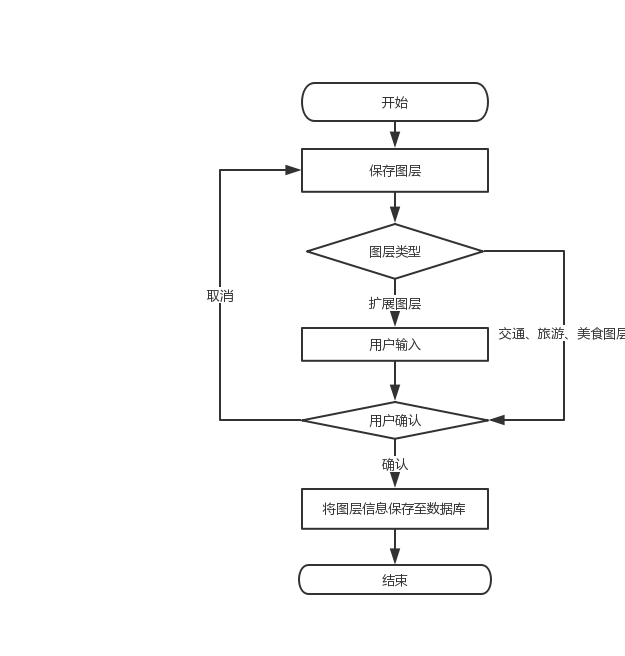


图3 图层保存流程图

图层查看的流程图如下图：

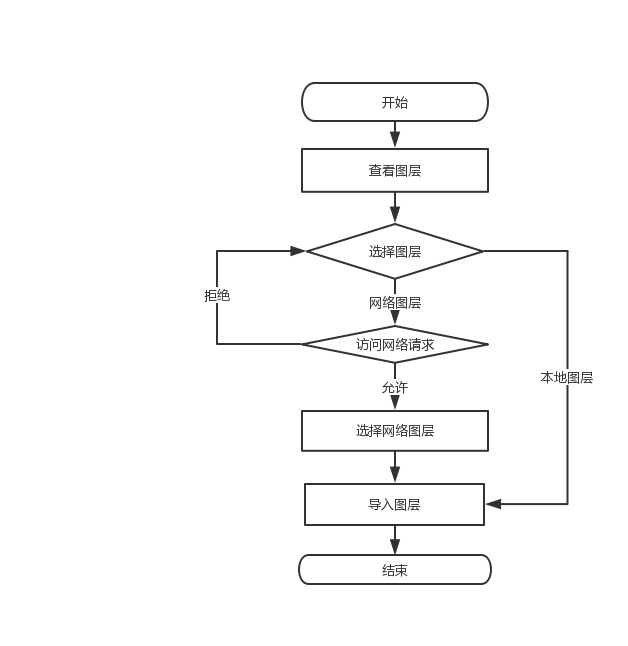


图4 图层查看流程图

信息标记、线路标记和图层保存与查看模块包括“标记”、“位置”、“位置标记”、“内容标记”、“图层”和“路线标记”六个类，其具体关系如下图所示：

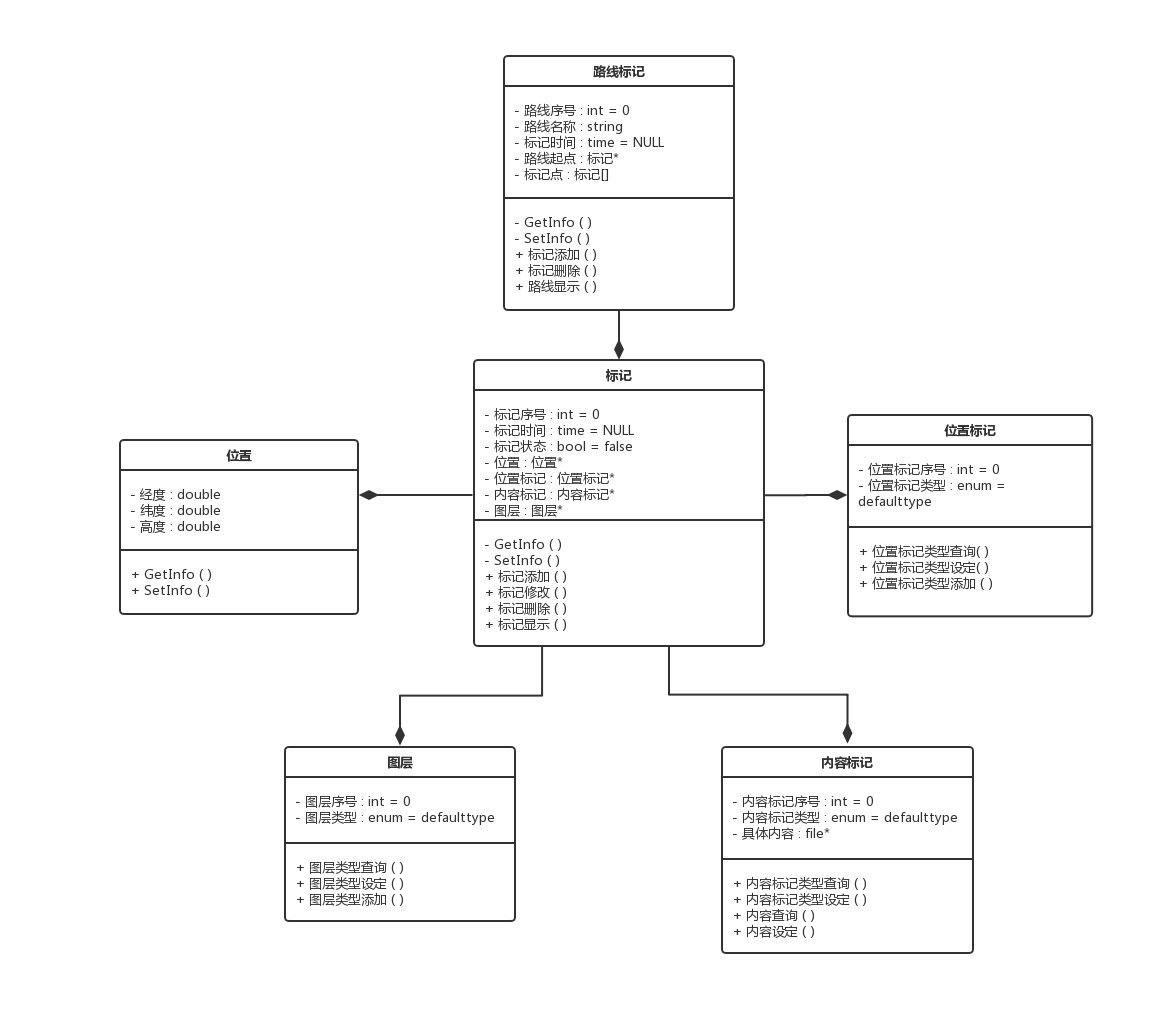


图4 信息标记和图层查看类图

图层保存包括将标记点和标记线路保存至图层；图层查看包括从本地导入图层和从网络导入图层，该部分的接口设计如下表所示：

表格2 图层保存与查看模块接口

|  |  |  |  |
| --- | --- | --- | --- |
| **接口名称** | **参数** | **返回值** | **说明** |
| import\_local() | int select\_local\_layer\_id | int | 从本地文件夹选择相应图层，并返回其id，失败则返回-1 |
| import\_net() | int select\_net\_layer\_id | int | 从服务器选择相应图层，并返回其id，失败则返回-1 |
| save\_layer() | int\* tags, int layer | int | 将标记内容保存至指定图层，并返回图层编号，失败则返回-1 |
| save\_routine() | int\* tags, int start, int end, int routine | int | 将标记点保存至指定路线，并记录起点位置和终点位置，返回路线编号，失败则返回-1 |

### 4.2.9 信息发布

在信息发布部分用户可以发布文字、图片或者链接类型的标记，共分两种情况。

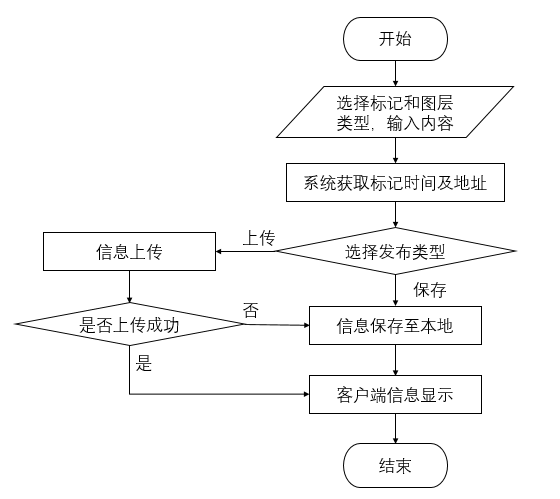
上传标记时，用户首先选择标记类型和图层类型，输入内容，选择保存或者上传按钮，系统自动获取点击按钮时的时间和用户所在地点。如点击上传并且上传成功，将获取的时间作为上传时间，将标记上传至服务器端，客户端显示信息“标记已成功上传”。

若用户选择上传且并未上传成功，则跳至第一种情况，自动保存。

以获取的时间作为标记时间，系统将用户编辑的内容、标记时间和地点保存至本地。客户端显示信息“标记未上传成功，已将其保存至本地”。

若用户选择保存按钮，以获取的时间作为标记时间，系统将用户编辑的内容、标记时间和地点保存至本地。客户端显示信息“标记已成功保存至本地”。

图一 信息发布流程图



### 4.2.10 好友管理

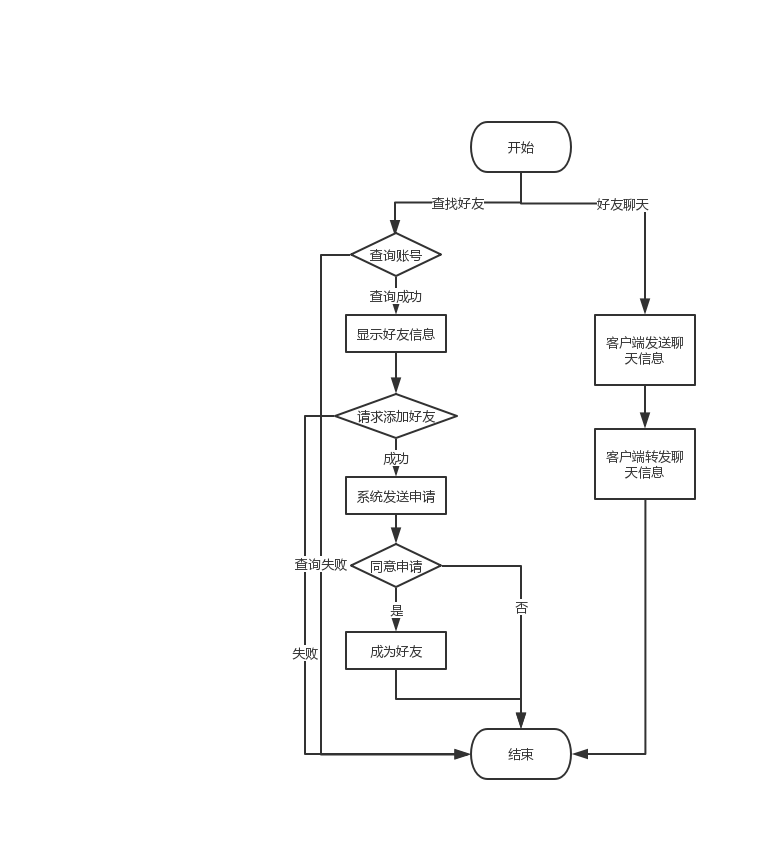
在好友管理部分可以查找好友、添加好友，在点击好友后可以打开聊天界面进行聊天。

查找好友是通过用户输入希望查询的帐号，在数据库中进行查询，成功查询后返回该账号的信息，失败则提示“未能查找到该用户”。

添加好友一般在查询好友之后，在用户信息界面可以选择添加好友，这时系统会像被添加用户发送申请，当被添加用户同意该申请时，可以在数据库建立好友联系。同时，将双方加入到对方的好友列表。

好友聊天是通过客户端发送信息，服务端进行转发，目前只提供在本地保存聊天记录。

好友管理的流程图如下图所示：

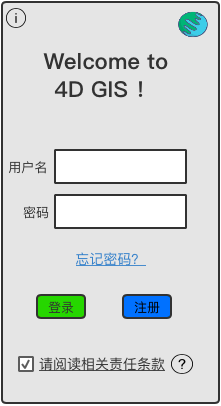


该部分的接口如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **接口名称** | **参数** | **返回值** | **说明** |
| get\_user\_info() | char[] | enum user | 根据账号在数据库中查找，返回帐户信息 |
| send\_req() | char[], char[] | bool | 发送给指定账号1对于账号2 的好友请求，返回是否成功 |
| add\_friend() | char[], char[] | bool | 建立好友联系，返回是否成功 |
| send\_msg() | string | bool | 客户端发送消息，服务端进行转发，返回是否成功 |

# 5、界面设计

## 5.1 登录/注册和密码找回界面

登录 注册 密码找回

登录/注册和密码找回界面接口：

① void GetUserLoginInfo(Shell Login, char\* username, char\* password);

功能：从登录界面获取登录用户的用户名和密码信息，并放入参数username和password所指向的内存区域中；

参数1：名称：LoginOrRegieter；类型：Shell；说明：登录窗口；

参数2：名称： username；类型：char\* ；说明：所指向区域用于存放用户名；

参数3：名称：password；类型：char\* ；说明：所指向区域用于存放密码；

返回值：无

② void GetMobileInfo(Shell RegisterOrFindPassword, char\* mobile);

功能：从注册界面或找回密码界面获取手机号信息，并放入参数mobile所指向的内存区域中，初始化与发送验证码函数相关的手机号参数；

参数1：名称：RegisterOrFindPassword；类型：Shell；说明：注册窗口或找回密码窗口；

参数2：名称：mobile；类型：char\* ；说明：所指向区域用于存放手机号；

返回值：无

③ void GetRegisterUserInfo(Shell Register, char\* username, char\* password, char\* confirm\_password, char\* verify\_code);

功能：从注册界面获取注册用户的用户名、密码和验证码信息，并放入参数username、password、confirm\_password 、mobile和verify\_code所指向的内存区域中；

参数1：名称：Register；类型：Shell；说明：注册窗口；

参数2：名称： username；类型：char\* ；说明：所指向区域用于存放用户名；

参数3：名称：password；类型：char\* ；说明：所指向区域用于存放密码；

参数4：名称：confirm\_password；类型：char\* ；说明：所指向区域用于存放确认密码；

参数5：名称：verify\_code；类型：char\* ；说明：所指向区域用于存放验证码；

返回值：无

④ void GetFindPasswordInfo(Shell FindPassword, char\* new\_password, char\* confirm\_new\_password, char\* verify\_code);

功能：从找回密码界面获取用户的新密码和验证码信息，并放入参数new\_username和verify\_code所指向的内存区域中；

参数1：名称：FindPassword；类型：Shell；说明：找回密码窗口；

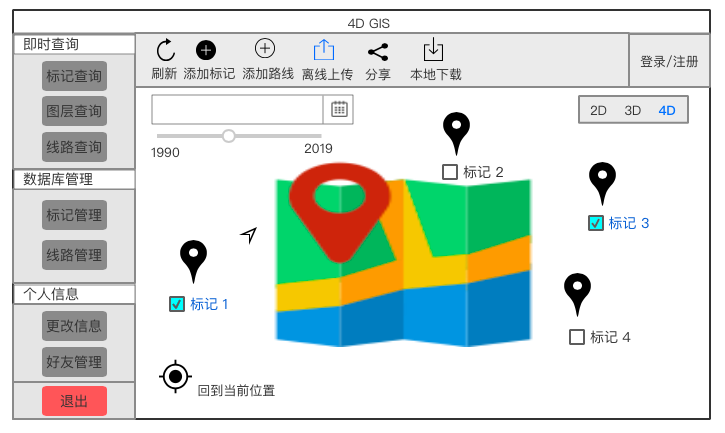
参数2：名称：new\_password；类型：char\* ；说明：所指向区域用于存放新密码；

参数3：名称：confirm\_new\_password；类型：char\* ；说明：所指向区域用于存放确认新密码；

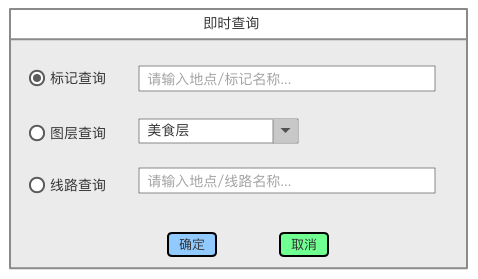
参数4：名称：verify\_code；类型：char\* ；说明：所指向区域用于存放验证码；

返回值：无

## 5.2主界面默认展示



## 5.3 即时查询界面



即时查询界面接口：

① 标记查询

void GeQueryTagInfo(Shell Query, char\* location\_or\_tagname);

功能：从即时查询界面获取待查询标记的信息，并放入参数location\_or\_tagname所指向的内存区域中；

参数1：名称：Query；类型：Shell；说明：即时查询窗口；

参数2：名称：location\_or\_tagname；类型：char\* ；说明：待查询标记的地点或名称；

返回值：无

② 图层查询

void GeQueryLayerInfo(Shell Query, int\* layer);

功能：从即时查询界面获取待查询图层的信息，并放入参数layer所指向的内存区域中；

参数1：名称：Query；类型：Shell；说明：即时查询窗口；

参数2：名称：layer；类型：int\* ；说明：待查询图层的ID；

返回值：无

③ 线路查询

void GeQueryLineInfo(Shell Query, char\* strat\_tag, char\* end\_tag, char\* line\_name);

功能：从即时查询界面获取待查询路线的信息，并放入参数strat\_tag、end\_tag和line\_name所指向的内存区域中；

参数1：名称：Query；类型：Shell；说明：即时查询窗口；

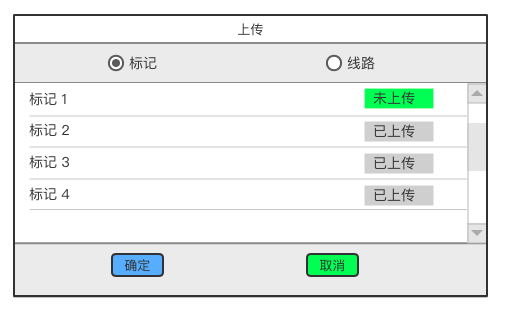
参数2：名称：strat\_tag；类型：char\* ；说明：待查询路线的起点标记名称；

参数3：名称：end\_tag；类型：char\* ；说明：待查询路线的终点标记名称；

参数4：名称：line\_name；类型：char\*；说明：待查询路线的名称；

返回值：无

## 5.4 数据上传界面



数据上传界面接口：

① 标记上传

void GeUploadTagInfo(Shell Upload, Set\* tag\_id);

功能：从数据上传界面获取待上传标记的信息，并放入参数tag\_id所指向的内存区域中；

参数1：名称：Upload；类型：Shell；说明：数据上传窗口；

参数2：名称：tag\_id；类型：Set \* ；说明：待上传标记ID的集合；

返回值：无

② 线路上传

void GeUploadLineInfo(Shell Upload, List\* tag\_id);

功能：从数据上传界面获取待上传路线包含的标记的信息，并放入参数tag\_id所指向的内存区域中；

参数1：名称：Upload；类型：Shell；说明：数据上传窗口；

参数2：名称：tag\_id；类型：List \* ；说明：待上传路线包含的标记ID的链表；

返回值：无

## 5.5 分享界面



分享界面接口：

void GeShareInfo(Shell Share, List\* tag\_id, int\* platform\_id, char\* content);

功能：从分享界面获取分享标记/线路及平台的信息，并放入参数tag\_id、platform\_id和content所指向的内存区域中；

参数1：名称：Share；类型：Shell；说明：分享窗口；

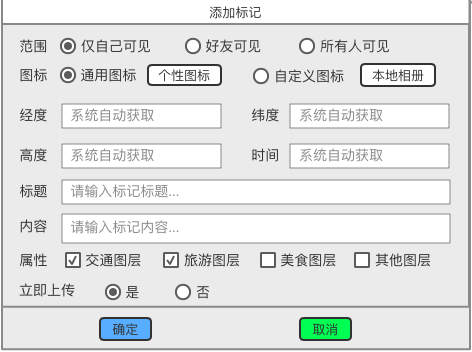
参数2：名称：tag\_id；类型：List \* ；说明：待分享标记ID的链表；

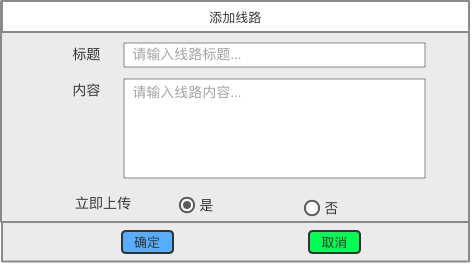
参数3：名称：platform\_id；类型：int \* ；说明：分享平台的ID；

参数4：名称：content；类型：char\*；说明：分享内容的描述；

返回值：无

## 5.6 添加标记/路线界面





添加标记/路线界面接口：

① 添加标记界面

void GetAddTagInfo(Shell AddTag, BufferedImage\* image, float\* longitude, float\* latitude, float\* height, Time\* time, char\* title, char\* content, set\* layer, boolean \* isUpload, int\* scope);

功能：从添加标记界面获取待添加标记的信息，并放入参数image 、longitude、latitude、height、time、title、content、layer、isUpload和scope所指向的内存区域中；

参数1：名称：AddTag；类型：Shell；说明：添加标记窗口；

参数2：名称：image；类型：BufferedImage\*；说明：标记的图标；

参数3：名称：longitude；类型：float \* ；说明：标记的经度；

参数4：名称：latitude；类型：float\* ；说明：标记的纬度；

参数5：名称：height；类型：float \*；说明：标记的高度；

参数6：名称：time；类型：Time\* ；说明：标记的时间；

参数7：名称：title；类型：char\* ；说明：标记的标题；

参数8：名称：content；类型：char\* ；说明：标记的描述信息；

参数9：名称：layer；类型：set\* ；说明：标记所属图层的ID集合；

参数10：名称：isUpload；类型：boolean\* ；说明：是否立即上传标记；

参数11：名称：scope；类型：int\* ；说明：可见范围；

返回值：无

② 添加线路界面

void GetAddLineInfo(Shell AddLine, char\* line\_title, char\* content);

功能：从添加线路界面获取待添加路线的信息，并放入参数tag\_num和content所指向的内存区域中；

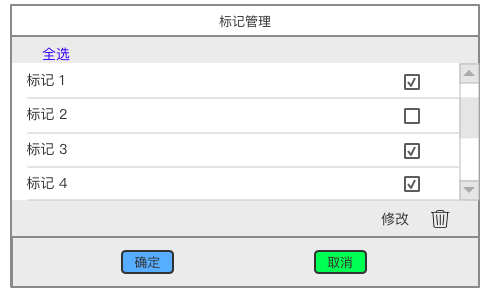
参数1：名称：AddLine；类型：Shell；说明：添加路线窗口；

参数2：名称：tag\_num；类型：int\* ；说明：待添加线路中含有的标记总个数；

参数6：名称：content；类型：char\* ；说明：上传路线的描述内容；

返回值：无

## 5.7 数据库管理界面









数据库管理接口：

① 标记删除

void GetDeleteTagInfo(Shell DeleteTag, set\* tag\_id);

功能：从标记管理界面获取待删除的标记id，并放入参数tag\_id所指向的内存区域中；

参数1：名称：DeleteTag；类型：Shell；说明：标记管理窗口；

参数2：名称：tag\_id；类型：set\*；说明：待删除的标记ID的集合；

返回值：无

② 修改标记

void GetModifyTagInfo(Shell ModifyTag, BufferedImage\* image, char\* title, char\* content, set\* layer, boolean \* isUpload, int\* scope);

功能：从修改标记界面获取修改后的标记信息，并放入参数image 、title、content、layer、isUpload和scope所指向的内存区域中；

参数1：名称：ModifyTag；类型：Shell；说明：修改标记窗口；

参数2：名称：image；类型：BufferedImage\*；说明：标记的图标；

参数3：名称：title；类型：char\* ；说明：标记的标题；

参数4：名称：content；类型：char\* ；说明：标记的描述信息；

参数5：名称：layer；类型：set\* ；说明：标记所属图层的ID集合；

参数6：名称：isUpload；类型：boolean\* ；说明：是否立即上传标记；

参数7：名称：scope；类型：int\* ；说明：标记可见范围；

返回值：无

③ 修改线路

void GetModifyLineInfo(Shell ModifyLine, char\* line\_title, char\* content);

功能：从修改线路界面获取修改后的路线信息，并放入参数line\_title和content所指向的内存区域中；

参数1：名称：ModifyLine；类型：Shell；说明：修改路线窗口；

参数2：名称：line\_title；类型：char\* ；说明：线路的名称；

参数3：名称：content；类型：char\* ；说明：路线的描述；

返回值：无

④ 路线删除

void GetDeleteLineInfo(Shell DeleteLine, set\* line\_id);

功能：从线路管理界面获取待删除的线路id，并放入参数line\_id所指向的内存区域中；

参数1：名称：DeleteLine；类型：Shell；说明：线路管理窗口；

参数2：名称：line\_id；类型：set\*；说明：待删除的线路ID的集合；

返回值：无

## 5.8 个人信息修改界面





个人信息修改界面接口：

① 个人信息修改

void Get ModifyUserInfo(Shell ModifyUser, BufferedImage\* image,char\* username, char\* password);

功能：从个人信息修改界面获取用户个人修改的相关信息，并放入参数image 、username和password所指向的内存区域中；

参数1：名称：ModifyUser；类型：Shell；说明：个人信息修改窗口；

参数2：名称：image；类型：BufferedImage\*；说明：修改后的个人图像；

参数3：名称：username；类型：char\* ；说明：修改后的用户名；

参数4：名称：password；类型：char\* ；说明：修改后的密码；

返回值：无

② 好友管理

void GetFriendsInfo(Shell ManageFriends, List\* friends\* friends, List\* groups);

功能：从好友管理界面获取好友和群组的相关信息，并放入参数friends和groups所指向的内存区域中；

参数1：名称：ManageFriends；类型：Shell；说明：好友管理窗口；

参数2：名称：friends；类型：List\*；说明：好友链表；

参数3：名称：groups；类型：List\* ；说明：群组链表；

返回值：无

# 6、重要技术难点及解决方案

## 6.1 在线聊天

**技术难点：**

在线聊天的难点在于多个客户端同时连接后会一直循环监听客户端输入，造成阻塞以至于服务端无法二次监听另外的客户端。

**解决方法：**

为了解决这个问题，我们采用了多线程的方法。每当有一个新的客户端连接上来，服务端便需要新启动一个线程进行处理，从而解决之前的循环读取中造成阻塞的问题。

**客户端：**

|  |  |
| --- | --- |
| **算法：** | 在线聊天算法 |
| **输入：** | Text |
| **输出：** | NULL |
| 1. public void connect() { 2. 新建服务器连接socket 3. 获取客户端输出流 4. } 5. public void disConnect(){ 6. 关闭客户端资源 7. } 8. private void sendMessage(String text) { 9. 向服务端发送消息 10. } | |

**服务端：**

|  |  |
| --- | --- |
| **算法：** | 在线聊天算法 |
| **输入：** | NULL |
| **输出：** | Text |
| 1. public void start() { 2. 启动服务器 3. 启动监听 4. 启动客户端线程 5. } 6. class Client implements Runnable { 7. public Client(Socket socket) { 8. 获取客户端输入流 9. } 10. public void run() { 11. WHILE 连接 12. 读取客户端传递的数据 13. 关闭相关资源 14. } 15. } | |

## 6.2 即时查询

## 6.3 离线数据同步

**技术难点：**

离线数据同步模块的难点在于本地数据同步时，本地数据库和服务器数据库中内容可能发生混乱，造成数据一致性问题。

**解决方法：**

在我们的设计方案中，优先考虑本地离线标记，将其先更新到服务器数据库中。因服务器数据库由多用户共同上传数据进行修改，所以是不断更新的。我们默认每次查询时，都要删除之前的本地数据库，重新下载服务端数据到本地。本地数据同步算法如下：

|  |  |
| --- | --- |
| **算法：** | 数据同步算法 |
| **输入：** | NULL |
| **输出：** | 是否成功 |
| 1. **FUNCTION** Data\_Synchronize() 2. **IF** size(离线标记表) > 0 3. **FOR** 离线标记 **IN** 离线标记表 4. upload (离线标记) 5. **END** **IF** 6. delete(客户端信息) 7. download(服务器信息) 8. **IF** error 9. **RETURN** false 10. **ELSE** 11. **RETURN** true 12. **END** **IF** 13. **END** **FUNCTION** | |

## 6.4 标记信息分享

**技术难点：**

标记信息分享可以将选择的标记或线路发送给好友，或添加分享信息后分享至所选的第三方平台，其技术关键在于对服务器数据库的多次查询。此外，还要考虑网络状况等多项因素的影响。

**解决方法：**

在我们的设计中，用标记id和好友或平台id作为查询条件，进行查询优化，尽可能提高查询效率，带给用户良好的体验，核心代码如下：

|  |  |
| --- | --- |
| **算法：** | 标记信息分享算法 |
| **输入：** | 标记或线路ID，是否为标记，分享ID，是否为好友，分享信息 |
| **输出：** | 分享是否成功 |
| 1. **FUNCTION** Share\_TagorLine\_to\_FrientorPlatform(**int** id1, **bool** is\_tag, **int** id2, 2. **bool** is\_friend, **string** info) 3. **IF** 无网络连接 4. **RETURN** **false** 5. **END** IF 6. **IF** is\_tag == **true** /\* 分享标记 \*/ 7. **SELECT** share\_from = \* 8. **FROM** tag\_table 9. **WHERE** id1 == tag.id 10. **ELSE** /\* 分享线路 \*/ 11. **SELECT** share\_from = \* 12. **FROM** line\_table 13. **WHERE** id1 == line.id 14. **END** **IF** 15. **IF** is\_friend == true /\* 分享给好友 \*/ 16. **SELECT** share\_to = \* 17. **FROM** friend\_table 18. **WHERE** id2 == friend.id 19. **ELSE** /\* 分享至第三方平台 \*/ 20. **SELECT** share\_to = \* 21. **FROM** platform\_table 22. **WHERE** id2 == platform.id 23. **END** **IF** 24. share(share\_from, share\_to, share\_info) 25. **IF** error 26. **RETURN** **false** 27. **ELSE** 28. **RETURN** **true** 29. **END** **IF** 30. **END** **FUNCTION** | |

## 6.5 信息标记

**技术难点：**

信息标记模块的难点在于：如何根据用户点击位置，从而获取该位置的经纬度信息

**解决方法：**

通过上网搜索，发现可以通过调用百度地图的API从而获取该位置的经纬度信息。在设计过程中，通过新建一个标记，然后将获取的位置信息赋给该标记。同时，可以由用户修改该标记的内容信息，等待用户确认后将其保存至本地数据库。核心代码如下：

|  |  |
| --- | --- |
| **算法：** | 信息标记 |
| **输入：** | 标记位置 |
| **输出：** | 标记序号 |
| 1. **FUNCTION** add\_location\_tag () 2. new(标记) 3. get(该点经度、纬度、高度信息) 4. set(标记，经度，纬度，高度) 5. 用户选择图标类型 6. **IF** "自定义图标" 7. error = 打开相册 8. **IF** error 9. **exit(0)** //打开失败 10. **ELSE** 11. 导入图标 12. **END** **IF** 13. display(标记) 14. ok = 用户确认() 15. **IF** ok 16. save() //保存该位置标记 17. **ELSE** 18. **exit(0)** //保存失败 19. **END IF** 21. **IF** 添加内容 //添加内容标记 22. ok = 导入内容() 23. **IF !**ok 24. **exit(0)**  //添加内容标记失败 25. **END IF** 26. **END IF** 28. **RETURN ++**location\_tag //返回信息标记序号，并将全局变量location\_tag++ 29. **END FUNCTION** | |

## 6.6 线路标记

**技术难点：**

线路标记模块的难点在于：如何根据用户点击的若干位置点，从而将其连接起来形成一条路线。

**解决方法：**

根据之前所学的知识，由于每个位置点的数据结构类型是相同的，可以考虑使用数组或者链表将其保存起来。但是，考虑到一条线路每次添加时是从起点开始顺序添加或者删除，而且不需要对中间的某个节点进行修改、查询操作，并且每条线路中节点数量不确定，所以采取带头指针的双向链表来保存线路信息。核心代码如下：

|  |  |
| --- | --- |
| **算法：** | 离线数据同步 |
| **输入：** | 若干标记 |
| **输出：** | 线路序号 |
| 1. **FUNCTION** Routine\_Add() 2. new(标记\* routine) 3. **WHILE TRUE:** 4. **IF** end: 5. **BREAK** 6. **IF** 添加 7. Link\_add(routine, 标记) //通过链表操作将标记添加至循环链表中 8. **ELSE IF** 删除 9. Link\_delete(routine) //通过链表操作将表尾的标记从链表删除 10. **RETURN ++**routine\_tag //返回路线标记序号，并将全局变量routine\_tag++ 11. **END FUNCTION** | |

## 6.7 图层保存与查看

**技术难点：**

图层保存与查看模块的难点在于：如何根据将用户选择的多个图层同时显示在屏幕上

**解决方法：**

为了解决这一问题，首先得设计图层的保存过程。根据以往经验并由数据库的保存得到启发，对数据进行增量保存可以节省存储开销，而且多个副本文件也易于叠加。所以，在保存图层时，每次仅保存用户新添加的信息标记和路线标记，这样在图层查看时，可以将多个增量文件同时叠加在地图上，相当于形成一个新的文件，从而实现该功能需求。核心代码如下：

|  |  |
| --- | --- |
| **算法：** | 信息标记保存至图层 |
| **输入：** | 信息标记ID，图层类型 |
| **输出：** | 图层序号 |
| 1. **FUNCTION** Save\_Layer(**int** ID, Layer\_Type layer\_type) 2. **SELECT** 信息标记 //从数据库中找到该标记 3. **FROM** 信息标记表 4. **WHERE** 信息标记ID == ID 5. 信息标记->图层 = layer\_type //修改该标记的图层属性 7. **SELECT** 图层序号 8. **FROM** 信息标记表 9. **WHERE** 图层类型 == layer\_type 10. **IF** 查找失败 //图层类型扩展 11. layer\_tag++ //图层序号自加 12. **END IF** 13. **INSERT INTO** 图层表 **VALUES (**图层序号，layer\_type，信息标记，信息标记ID**)** 15. **RETURN** layer\_type 16. **END FUNCTION** | |

|  |  |
| --- | --- |
| **算法：** | 路线标记保存至图层 |
| **输入：** | 路线标记ID，图层类型 |
| **输出：** | 图层类型 |
| 1. **FUNCTION** 2. **SELECT** 路线标记 //从数据库中找到该路线 3. **FROM** 路线标记表 4. **WHERE** 路线标记ID == ID 5. 路线标记->图层 = layer\_type //修改该路线的图层属性   **7. SELECT** 图层序号  **8.**  **FROM** 路线标记表  **9.** **WHERE** 图层类型 == layer\_type  **10.** **IF** 查找失败 //图层类型扩展  **11.** layer\_tag++ //图层序号自加  **12.** **END IF**  **13.** **INSERT INTO** 图层表 **VALUES (**图层序号，layer\_type， 线路标记， 线路标记ID**)**  **14.**  **15. RETURN** layer\_type   1. **END FUNCTION** | |

|  |  |
| --- | --- |
| **算法：** | 图层查看 |
| **输入：** | 图层序号[] |
| **输出：** | 图层内容 |
| 1. **FUNCTION** 2. **FOR** layer **IN**图层序号[] 3. **SELECT** 信息标记 4. **FROM** 信息标记表 5. **WHERE** 图层序号 == layer 6. display(信息标记) //显示该图层下的所有信息标记 8. **SELECT** 路线标记 9. **FROM** 路线标记表 10. **WHERE** 图层序号 == layer 11. display(路线标记) //显示该图层下的所有路线标记 12. **END FOR** 13. **END FUNCTION** | |

## 6.8 信息发布