易助 技术设计文档

2017（1.0）-2017-5-26

易助 技术设计文档

2017（1.0）-2017-5-26

状态 ：	文件标签：	技术设计文档、易助
[ ] 草稿	版本：	2017(1.0)
[√] 修改中	作者：	
[ ] 定稿	日期：	2017-5-26

编辑历史

文件名称：				
易助 技术设计文档				
文件说明：				
编辑历史：				
编辑时间	版本	作者	编辑内容	标记
2017.5.26	Build1		建立文档	正常
2017.6.28	Build2		完善技术设计文档	正常
2017.6.29	Build3		添加服务端设计文档	正常
22017.7.7	Build4		添加客户端设计文档	正常

目录

1 前言

1.1 项目的背景和意义

由于社会老龄化趋势逐渐加重，大量残障人士生活极其不便。社会对老年人、残障人士的生活将会越来越关注。易助推出的功能在社区居民遇到紧急情况时能及时地对他们进行救助，有利于营造良好的社会互助氛围，公益理念和切实可行的救助方式一定能够使易助在社区里迅速推广。

共享经济成为社会发展的浪潮，在社区间推出易助，可以更好地整合线下的闲散物品，劳动力，教育医疗资源。借助易助平台，交换闲置物品，分享自己的知识、经验。易助提供了一个第三方平台，以社区的形式整合身边的资源，帮助解决身边的问题，为帮助建立智慧社区、智慧城市降低成本。

1.2 项目的目标和范围

本项目将实现如下的功能：

①注册：用户用手机号注册；

②登录：用户名、密码登录；

③一键求救：向周围人推送求救信息。

④救助他人：查看周围求救信息，并提供救助；

⑤我要求助：向周围人发送求助信息；

⑥帮助他人：查看周围求助信息，并提供帮助；

⑦我要提问：提问并推送；

⑧回答问题：查看提问列表并选择回答问题；

⑨个人信息：修改个人信息；

⑩定位：接入地图，确定当前位置。

1.3 本文结构简介

本文的结构将以软件工程的结构来组织。

第二章：技术与原理部分，本文将分析为完成上述的项目目的和需求所需要的设计模式、编码原理和技术，将检索和参阅的技术资料和文献中的相关部分进行简要说明，并分析其在实现本工程设计上的意义。

第三章：架构设计部分，本文会在技术原理的基础上，分析整个工程系统架构的选择、设计的思路和相关技术的说明，同时对主要业务用例从技术角度进行详细分析说明。

第四章：总结部分，本文将总结本次项目所有工作成果，对项目的优势和不足、可拓展性、可维护性进行评估，对项目的发展进行展望。

2 技术与原理

本章将对易助项目中实施的设计模式和相关技术手段进行阐述，对在项目中采用的具体设计和实现方式进行说明，并分析其在实现本工程设计上的存在意义。

2.1 前端-Android 客户端

2.1.1 框架

1.Android Studio 作为开发 IDE。

2.底部分栏，类似微信 QQ 效果，使用 Fragment。

2.1.2 功能实现

1.推送功能：使用极光推送 SDK。

2.定位地图功能：使用百度地图 SDK。

3.使用 HTTP 协议与服务端交互。

2.2 服务端-Java web

2.2.1 框架

1. 主体框架使用 Spring。
2. MVC 框架使用 Springmvc+Thymeleaf;
3. ORM 持久层框架使用 Hibernate。

2.2.2 功能实现

1. 使用 JSON 进行数据交互。
2. 使用 MySQL 作为后端数据库。

3 架构设计

本章将对易助项目进行架构设计，以面向对象编程的方法，分析整个工程系统架构的选择、设计的思路和相关技术的说明。

3.1 服务端设计

3.1.1 目录结构

```
yizhu
        - src
                - main
                        - java
                                - com.sysu.yizhu
                                        - business              # 软件业务包
                                                - entities          # 实体
                                                        - repositories   # 实体持久层仓库
                                                - services          # 服务层
                                        - util                  # 工具
                                        - web                   # web层
                                                - controller        # 接口控制器路由
                                                - interceptor       # 拦截器
                        - resources                     # 配置资源文件
                        - webapp
                                - WEB-INF                   # 配置文件
                - test
                        - java
                                - test
        - pom.xml                               # 项目管理
```

3.1.2 外部系统

```
database
        mysql
                user: root
                pass: 123456
```

3.1.3 控制器路由

1. 实现

```
# 位于com.sysu.yizhu.web.controller包中
        - UserController.java          # 实现用户类接口
        - QuestionController.java      # 实现提问、回答类接口
        - SOSController.java           # 实现一键求救接口
        - HelpController.java          # 实现求助功能接口
```

2. 返回值

```
# 位于com.sysu.yizhu.util包中
# 继承自LinkedHashMap<String, Object>，通过Spring返回值格式化为JSON格式
        - ReturnMsg.java
```

3. 拦截器

```
# 位于com.sysu.yizhu.web.interceptor包中
        - LoginInterceptor.java
```

通过拦截用户请求，区分请求方法，POST 方法需要登录，将用户记录于 session 中。

3.1.4 业务层

1. 向上接口实现

```
# 位于com.sysu.yizhu.business.services包中
        - UserService.java
        - QuestionService.java
        - SOSService.java
        - HelpService.java
```

介于数据库持久层与控制器层中间，集成数据库操作并向控制器提供业务接口。

2. 实体

```
# 位于com.sysu.yizhu.business.entities包中
# 实体设计
-  User
-  Comment
-  Question
-  Answer
-  AgreeAnswer
-  SOS
-  SOSResponse
-  Help
-  HelpResponse


# 持久层设计
- repositories
```

3. 框架

   使用 Hibernate 持久层框架，将实体映射为数据库的表结构。

   通过 JPA 注解设计数据库，如 SOSResponse 实体。

```
// 定义实体
@Entity
// 定义表名
@Table(name="sos_response")
public class SOSResponse {
    // 定义各列，面向对象设计
    private Integer sosResponseId;
    private SOS sos;
    private User sosResponseUser;

    public SOSResponse() {
    }

    // Id列，可设置自动增长，并设置列名
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "sos_response_id")
```

```
    public Integer getSosResponseId() {
        return sosResponseId;
    }
    public void setSosResponseId(Integer sosResponseId) {
        this.sosResponseId = sosResponseId;
    }

    // 多对一，建立外键，以及设置懒加载
    @ManyToOne(cascade = {CascadeType.MERGE}, fetch = FetchType.LAZY)
    @JoinColumn(name = "sos_id")
    public SOS getSos() {
        return sos;
    }
    public void setSos(SOS sos) {
        this.sos = sos;
    }

    @ManyToOne(cascade = {CascadeType.MERGE}, fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id")
    public User getSosResponseUser() {
        return sosResponseUser;
    }
    public void setSosResponseUser(User sosResponseUser) {
        this.sosResponseUser = sosResponseUser;
    }
}
```

4. 数据库方法接口

   使用 Hibernate 提供的 JPA CURD 类模板，HelpResponseRepository。

```
// 注解仓库
@Repository
public interface HelpResponseRepository extends CrudRepository<HelpResponse, Integer> {
    // 使用查询语句注解，模板将自动将参数传入执行SQL并将结果写入对象返回
    @Query("select r.helpResponseUser.userId from HelpResponse r where r.help.helpId = ?1")
    List<String> findAllUserIdByHelpId(Integer sosId);

    // 使用模板方法名，By关键字后为属性名，定义传入属性，自动生成并执行SQl，获得结果返回
    List<HelpResponse> findByHelpAndHelpResponseUser(Help help, User user);
}
```

3.1.5 工具类

位于 com.sysu.yizhu.util 包中。

1. 推送、短信服务

```
# 使用leancloud提供的服务
- LCConfig    # 读取leancloud配置数据，如app-key
- LCUtil      # 自定义封装leancloud服务，便于调用
```

封装了：短信验证服务，消息推送服务。

通过包装 LeanCloud 接口服务，给易助前端提供更简单的调用方式。

如消息推送服务：

```
// LCUtil类内方法
// 字符串模板，包装了一个leancloud请求的语句
private static final String REQ_PUSH_SOS_TEMPLATE =
        "{\n" +
        "    \"where\": {\n" +
        "          \"location\": {\n" +
        "            \"$nearSphere\": {\n" +
        "              \"__type\": \"GeoPoint\",\n" +
        "              \"latitude\": {latitude},\n" +
        "              \"longitude\": {longitude}\n" +
        "            },\n" +
        "            \"$maxDistanceInMiles\": {maxDistance}\n" +
        "          }\n" +
        "    },\n" +
        "    \"data\": {\n" +
        "        \"alert\": \"{alertContent}\"\n" +
        "    }\n" +
        "}";

// 自定义方法，通过替代占位符达到传参的目的
private String getPushSOSJSON(Double latitude, Double longitude, Double maxDistance, String alertContent) {
    return REQ_PUSH_SOS_TEMPLATE.replace("{latitude}", latitude.toString())
            .replace("{longitude}", longitude.toString())
            .replace("{maxDistance}", maxDistance.toString())
            .replace("{alertContent}", alertContent);
}

// 对外接口方法，在Http头中封装了信息，直接发送即可
public boolean pushHelp(Double latitude, Double longitude) {
    try {
        HttpEntity<String> reqEntity = new HttpEntity<String>(
                getPushHelpJSON(latitude, longitude, 100.0, "一条新的求助消息！"),
                headers);
        String res = restTemplate.postForObject(URL_PUSH, reqEntity, String.class);
        LOG.info("pushSOS() res: " + res);
        return true;
    } catch (HttpClientErrorException e) {
```

```
            LOG.error("pushSOS() res: " + e.getMessage());
            LOG.error("pushSOS() res: " + e.getResponseBodyAsString());
            return false;
        }
    }
```

2. MD5 计算

```
- MD5Parser
        - getMD5()     # 将传入字符串计算MD5值并返回
```

3. 数据库 MySQL 自定义连接配置类

```
- MySQL5Dialect    # 配置了innoDB和建表编码utf-8
```

4. 手机号码检验

```
- PhoneNumUtil
        - isPhone()    # 检验传入手机号是否为有效格式的手机号
```

   检验使用了正则表达式

5. 返回消息

   见控制层叙述。

3.1.6 配置资源文件

```
# 位于yizhu/src/main/resources中
```

1. LeanCloud 配置文件

```
- leancloud.properties     # 需要X-LC-Id，X-LC-Key字段，由leancloud官网提供
```

2. log4j 日志系统配置

```
- log4j.properties          # 配置log4j插件的具体项
```

   使用 AOP 的编程模式

3. 各运行环境配置

```
- settings-development.properties    # 开发环境配置
- settings-jenkins.properties        # 集成测试环境配置
- settings-production.properties     # 正式环境配置

# 均有字段
db.user
db.password
db.jdbcUrl
```

3.1.7 系统运行部署

1. 项目 github 地址

```
https://github.com/SYSU-yizhu/yizhu
```

2. Mysql 数据库设置

   见配置文件，配置数据库端口

3. 运行

```
# 如使用production配置
# 进入yizhu项目文件夹，运行命令行
mvn clean package tomcat7:run -Dspring.profiles.active=production
```

4. 集成环境配置

   http://119.29.166.163/index.php/2017/06/28/jenkins-%E7%AE%80%E5%8D%95ci-%E5%A4%9Adocker%E8%BF%9E%E6%8E%A5/

3.1.8 对外接口

3.1.8.1 用户类

1. 注册前发送短信验证码

Code	Content	Description
200	OK	请求成功
400	FAILED	手机号已存在
403	FORBIDDEN	手机号（用户名）无效

URI：

```
GET /user/sendSms/{userId}
```

GET 参数：

字段	描述	类型
userId	用户名（手机号）	string

成功例子：

```
{
    "userId":"11111111111"
}
```

2. 注册

Code	Content	Description
200	OK	请求成功
400	FAILED	用户名已存在
403	FORBIDDEN	用户名不是手机号或参数格式错误
450	MISS	验证码错误

URI:

```
POST /user/register
```

POST 参数：

字段	描述	类型
userId	用户名（手机号）	string
password	密码（未处理）	string
code	短信验证码	string
name	姓名	string
gender	性别	string - "male"/"female"
birthDate	出生日期	string - YYYY-MM-DD
location	常住地	string

成功例子：

```
{
    "userId":"11111111111"
}
```

3. 登录

Code	Content	Description
200	OK	请求成功
403	FORBIDDEN	用户名不是手机号或参数格式错误
400	NOT FOUND	用户名或密码错误

URI

```
POST /user/login
```

POST 参数

字段	描述	类型
userId	用户名（手机号）	string
password	密码（未处理）	String

成功例子：

```
{
    "userId":"11111111111"
}
```

4. 修改个人信息

Code	Content	Description
200	OK	请求成功
401	FAILED	未登录
403	FORBIDDEN	性别或日期参数格式不正确

URI

```
POST /user/modifyInfo
```

POST 参数

字段	描述	类型
name	姓名	string
gender	性别	string - "male"/"female"
birthDate	出生日期	string - YYYY-MM-DD
location	常住地	string

成功例子：

```
{
    "userId":"11111111111"
}
```

5. 获取个人信息

Code	Content	Description
200	OK	请求成功
401	FAILED	未登录

URI

```
POST /user/info
```

POST 参数：无

成功例子：

```
{
    "userId":"11111111111",
    "name":"张三",
    "gender":"male",
    "birthDate":"1995-12-02",
    "location":"中山大学"
}
```

6. 推送更新用户 objectid（leancloud SDK 获取）

   该接口需要登录

Code	Content	Description
200	OK	请求成功
401	FAILED	未登录
404	MISS	objectId不存在

   URI

```
POST /user/updateObjectId
```

   POST 参数

字段	描述	类型
objectId	安装Id	string

   成功例子：

```
{
    "userId":"11111111111"
}
```

7. （推送）更新用户定位位置

   该接口需要登录

Code	Content	Description
200	OK	请求成功
401	FAILED	未登录
403	FORBIDDEN	纬度不在-90~90或经度不在-180~180间
450	MISS	用户未记录安装Id
500	Error	服务器错误

URI

```
POST /user/updateLocation
```

POST 参数

字段	描述	类型
latitude	纬度	double
longitude	经度	double

成功例子：

```
{
    "userId":"11111111111"
}
```

3.1.8.2 提问类

1. 提问

Code	Content	Description
200	OK	请求成功
401	FAILED	未登录

URI

```
POST /question/ask
```

POST 参数：

字段	描述	类型
title	标题	string
content	提问内容	string

成功例子：

```
{
    "questionId":1
}
```

2. 回答

Code	Content	Description
200	OK	请求成功
401	FAILED	未登录
450	MISS	该提问id不存在

URI

```
POST /question/answer
```

POST 参数

字段	描述	类型
questionId	问题id	int
content	回答内容	string

成功例子：

```
{
    "answerId":1
}
```

3. 赞同

Code	Content	Description
200	OK	请求成功
401	FAILED	未登录
450	MISS	该回答id不存在

URI

```
POST /question/agreeAnswer
```

POST 参数

字段	描述	类型
answerId	回答id	int
agreeOrNot	是否赞同	bool - true赞同/false不赞同

成功例子

```
{
    "answerAgreeId":1
}
```

4. 获取所有问题 id

Code	Content	Description
200	OK	请求成功

URI

```
GET /question/getAllId
```

成功例子

```
{
    "count":2,
    "data":[1,2]
}
```

5. 根据问题 id 获取问题摘要

Code	Content	Description
200	OK	请求成功
404	NOT FOUND	questionId不存在

URI

```
GET /question/digest/{questionId}
```

成功例子

```
{
    "questionId":1,
    "userId":"133133123456",
```

```
    "userName":"张三",
    "title":"扶老奶奶过马路是一种怎样的体验？",
    "createDate":"2017-05-17"
}
```

6. 根据问题 id 获取完整问题内容

Code	Content	Description
200	OK	请求成功
404	NOT FOUND	questionId不存在

URI

```
GET /question/detail/{questionId}
```

成功例子

```
{
    "questionId":1,
    "userId":"133133123456",
    "userName":"张三",
    "title":"扶老奶奶过马路是一种怎样的体验？",
    "content":"bla bla blabla",
    "createDate":"2017-05-17"
}
```

7. 根据问题 id 获取回答所有 id

Code	Content	Description
200	OK	请求成功
404	NOT FOUND	questionId不存在

URI

```
GET /question/getAnswerIds/{questionId}
```

成功例子

```
{
    "count":2,
    "data":[1,2]
}
```

8. 根据回答 id 获取回答内容

Code	Content	Description
200	OK	请求成功
404	NOT FOUND	answerId不存在

URI

```
GET /question/getAnswer/{answerId}
```

成功例子

```
{
    "answerId":1,
    "userId":"133133123456",
    "userName":"张三",
    "content":"无可奉告",
    "createDate":"2017-05-17",
    "good":5,
    "bad":30
}
```

3.1.8.3 一键求救（推送、导航、评价）

1. 发起求救

   该接口需要登录

Code	Content	Description
200	OK	请求成功
401	FAILED	未登录
403	FORBIDDEN	纬度不在-90~90或经度不在-180~180间
450	MISS	用户未记录安装Id
500	Error	服务器错误

   URI

```
POST /sos/push
```

   POST 参数

字段	描述	类型
latitude	纬度	double
longitude	经度	double

成功例子

```
{
    "sosId":1
}
```

2. 响应求救

该接口需要登录

Code	Content	Description
200	OK	请求成功
401	FAILED	未登录
403	FORBIDDEN	已响应，不能重复响应
404	NOT FOUND	SOS id 不存在或已完成
450	MISS	用户未记录安装Id

URI

```
POST /sos/response
```

POST 参数

字段	描述	类型
sosId	求救Id	string

成功例子

```
{
    "userId":"11111111111"
}
```

3. 查询所有有效求救 id

Code	Content	Description
200	OK	请求成功

URI

```
GET /sos/allValidId
```

成功例子

```
{
    "count":2,
    "data":[1,2]
}
```

4. 根据 sos id 获取 sos 内容

Code	Content	Description
200	OK	请求成功
404	NOT FOUND	SOS id 不存在

URI

```
GET /sos/get/{sosId}
```

成功例子

```
{
    "sosId": 2,
    "latitude": 0.0,
    "longitude": 5.0,
    "createTime":"2017-06-27 10:15",
    "finished":false,
    "pushUserId": "11111111111"
}
```

5. 根据 sos id 获取所有响应者 id

Code	Content	Description
200	OK	请求成功
404	NOT FOUND	SOS id 不存在或已完成

URI

```
GET /sos/response/{sosId}
```

成功例子

```
{
    "count":2,
    "data":["1234568911", "12345678922"]
}
```

6. 结束求救

   该接口需要登录

Code	Content	Description
200	OK	请求成功
401	FAILED	未登录
404	NOT FOUND	SOS id 不存在或已完成
450	MISS	非发起用户无法结束该求救

URI

```
POST /sos/finish
```

POST 参数

字段	描述	类型
sosId	sos id	integer

成功例子

```
{
    "sosId":2
}
```

3.1.8.4 求助（图片、文字、评价）

1. 发起求助

   该接口需要登录

Code	Content	Description
200	OK	请求成功

401	FAILED	未登录
402	WRONG	需求人数无效，应在1-10人之间
403	FORBIDDEN	纬度不在-90~90或经度不在-180~180间
450	MISS	用户未记录安装Id
500	Error	服务器错误

URI

```
POST /help/push
```

POST 参数

字段	描述	类型
latitude	纬度	double
longitude	经度	double
title	事件标题	string
detail	事件信息	string
needs	需要多少帮助者	integer

成功例子

```
{
    "helpId":1
}
```

2. 响应求助

   该接口需要登录

Code	Content	Description
200	OK	请求成功
401	FAILED	未登录
402	WRONG	该求助人数已满
403	FORBIDDEN	已响应，不能重复响应
404	NOT FOUND	Help id 不存在或已完成
450	MISS	用户未记录安装Id

URI

```
POST /help/response
```

POST 参数

字段	描述	类型

helpId	求助Id	string

成功例子

```
{
    "userId":"11111111111"
}
```

3. 查询所有有效求助 id

Code	Content	Description
200	OK	请求成功

URI

```
GET /help/allValidId
```

成功例子

```
{
    "count":2,
    "data":[1,2]
}
```

4. 根据 help id 获取求助内容

Code	Content	Description
200	OK	请求成功
404	NOT FOUND	Help id 不存在

URI

```
GET /help/get/{helpId}
```

成功例子

```
{
    "helpId": 2,
    "latitude": 0.0,
    "longitude": 5.0,
    "finished":false,
    "title":"帮忙抬米",
    "detail":"抬三袋米上五楼",
    "needs":3,
    "responseNum":2,
    "pushUserId": "11111111111"
}
```

5. 根据 help id 获取所有响应者 id

Code	Content	Description
200	OK	请求成功
404	NOT FOUND	Help id 不存在或已完成

URI

```
GET /help/response/{helpId}
```

成功例子

```
{
    "count":2,
    "data":["1234568911", "12345678922"]
}
```

6. 结束求助

   该接口需要登录

Code	Content	Description
200	OK	请求成功
401	FAILED	未登录
404	NOT FOUND	Help id 不存在或已完成
450	MISS	非发起用户无法结束该求救

URI

```
POST /help/finish
```

POST 参数

字段	描述	类型
helpId	help id	integer

成功例子

```
{
    "helpId":2
}
```

3.2 客户端设计

3.2.1 项目信息

开发环境：Android Studio

开发语言：Java

信息传递：HTTP 协议，JSON

运行环境：Android（最小 SDK 版本：19）

权限需求：存储，位置信息，网络连接

3.2.2 功能实现

登录注册

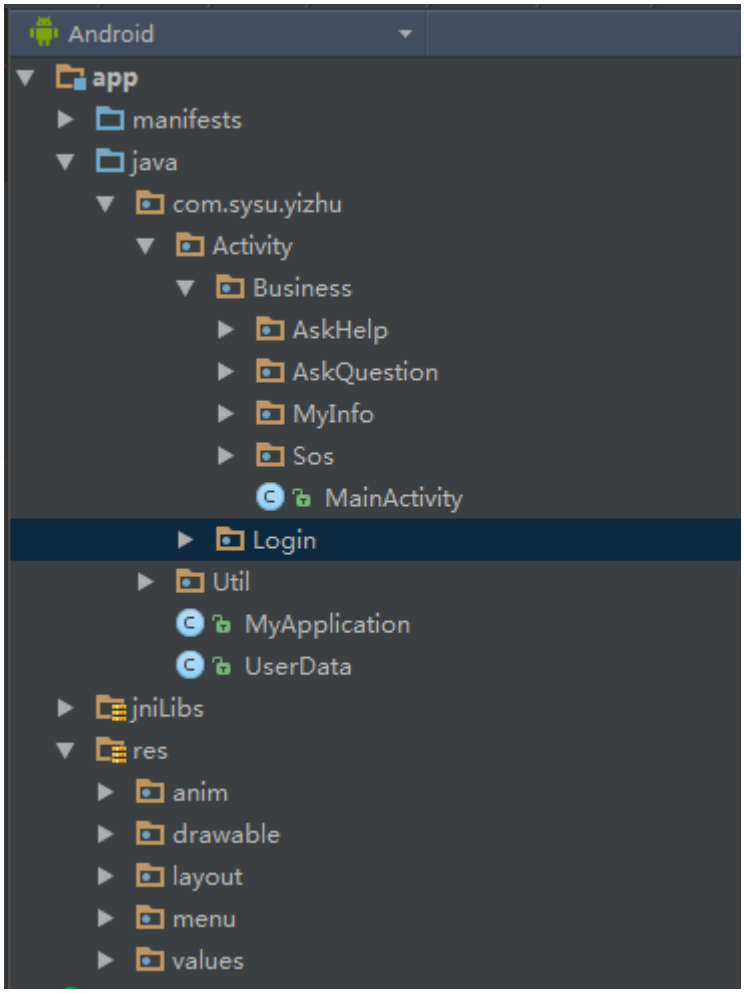
求救

求助

提问

个人信息

3.2.3 目录结构



3.2.4 外部控件与框架

1.百度地图 SDK（定位，地图）

2.LeanCloud（推送）

3.Ultra Pull To Refresh（下拉刷新框架）

Github 地址:

https://github.com/liaohuqiu/android-Ultra-Pull-To-Refresh

4.FloatingActionButton（悬浮按钮控件）

Github 地址:

https://github.com/futuresimple/android-floating-action-button

5. BottomNavigation（底部导航栏控件）

Github 地址:

https://github.com/Ashok-Varma/BottomNavigation

6. CircleImageView（圆形图片控件）

Github 地址:

https://github.com/hdodenhof/CircleImageView

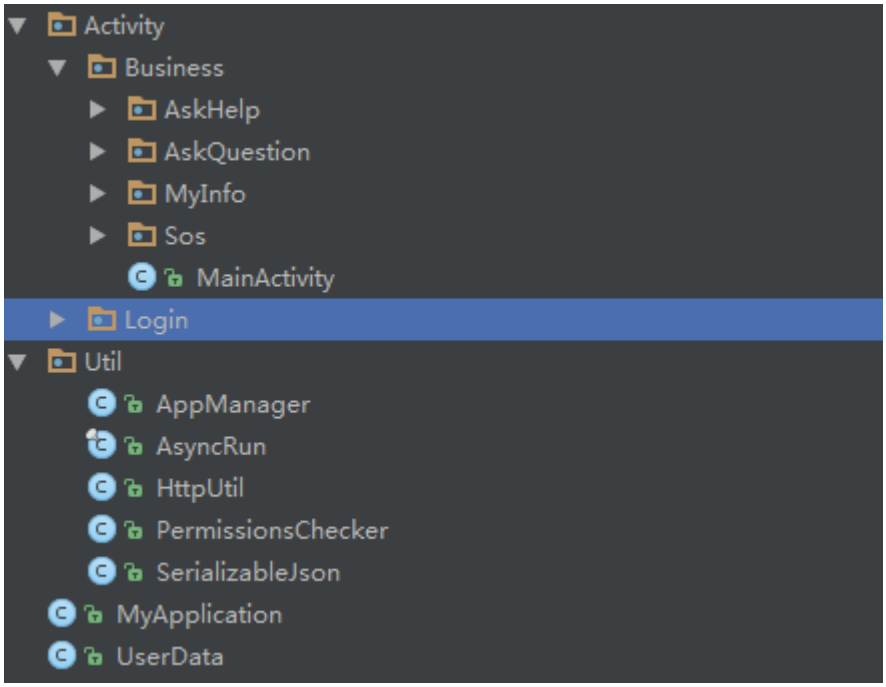
build.gradle 文件:

```
apply plugin: 'com.android.application'


android {
    compileSdkVersion 25
    buildToolsVersion "24.0.2"


    defaultConfig {
        applicationId "com.sysu.yizhu"
        minSdkVersion 19
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
        }
    }
    sourceSets {
        main() {
            jniLibs.srcDirs = ['libs']
        }
    }
}
```

```
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:25.3.1'
    compile 'de.hdodenhof:circleimageview:2.1.0'
    compile 'com.ashokvarma.android:bottom-navigation-bar:2.0.1'
    compile 'in.srain.cube:ultra-ptr:1.0.11'
    compile 'com.getbase:floatingactionbutton:1.10.1'
    compile files('libs/BaiduLBS_Android.jar')
    compile files('libs/httpmime-4.1.2.jar')
    compile files('libs/IndoorscapeAlbumPlugin.jar')
    compile files('libs/leancloud/avoscloud-push-v4.1.1.jar')
    compile files('libs/leancloud/avoscloud-sdk-v4.1.1.jar')
    compile files('libs/leancloud/fastjson-1.2.30.jar')
    compile files('libs/leancloud/Java-WebSocket-1.3.2-leancloud.jar')
    compile files('libs/leancloud/okhttp-3.8.0.jar')
    compile files('libs/leancloud/okio-1.13.0.jar')
    compile files('libs/leancloud/protobuf-java-2.6.1.jar')
}
```

3.2.1 主要类说明

1.类包结构



2.Activity-Login

APP 的登录注册页面的逻辑

3.**Activity-Business**

主体业务页面的逻辑，（求救、求助、提问、个人信息）

4.Util-AppManager

对 Activity 进行统一管理，方便页面切换，采用单例模式，程序调用如：

```
AppManager.getAppManager().finishAllActivity();

AppManager.getAppManager().addActivity(MainActivity.this);
```

5. Util-HttpUtil

提供 Http 协议的 Post 和 Get 方法接口，方便与 Web 服务器进行数据交互，使用 Session 保持，主要接口方法如下：

```
public static void get(final String url, final HttpResponseCallBack callBack) {

    }
```

```
public static void post(final String url, final Map<String, String> params, final

HttpResponseCallBack callBack) {

    }
```

6.程序使用示例：

Get

```
HttpUtil.get(getCodeUrl + phoneNumText.getText().toString(), new
HttpUtil.HttpResponseCallBack() {
    @Override
    public void onSuccess(int code, String result) {
        switch (code) {
            case 200:
                Toast.makeText(SignUpActivity.this, "验证码已发送到您手机上",
Toast.LENGTH_SHORT).show();
                break;
            case 400:
                Toast.makeText(SignUpActivity.this, "手机号码已注册",
Toast.LENGTH_SHORT).show();
                break;
            case 403:
                Toast.makeText(SignUpActivity.this, "手机号码无效",
Toast.LENGTH_SHORT).show();
                break;
            default:
                break;
        }
    }

    @Override
    public void onFailure(String result, Exception e) {

    }
});
```

Post

```
HashMap<String, String> params = new HashMap<String, String>();
```

```
params.put("userId", phoneNumText.getText().toString());
params.put("password", passwordText.getText().toString());
params.put("code", codeText.getText().toString());
params.put("name", nameText.getText().toString());
params.put("gender", gender);
params.put("birthDate", birthDateText.getText().toString());
params.put("location", locationText.getText().toString());
HttpUtil.post(signUpUrl, params, new HttpUtil.HttpResponseCallBack() {
    @Override
    public void onSuccess(int code, String result) {
        switch (code) {
            case 200:
                resultAnalysis(result);
                Toast.makeText(SignUpActivity.this, "注册成功", Toast.LENGTH_SHORT).show();
                Intent intent = new Intent();
                intent.setClass(SignUpActivity.this, MainActivity.class);
                SignUpActivity.this.startActivity(intent);
                break;
            case 400:
                Toast.makeText(SignUpActivity.this, "手机号已被注册",
Toast.LENGTH_SHORT).show();
                break;
            case 403:
                Toast.makeText(SignUpActivity.this, "手机号码无效",
Toast.LENGTH_SHORT).show();
                break;
            case 450:
                Toast.makeText(SignUpActivity.this, "验证码错误", Toast.LENGTH_SHORT).show();
                break;
            default:
                break;
        }
```

```
    }


    @Override
    public void onFailure(String result, Exception e) {


    }
});
```

7.UserData

存储用户数据，方便程序获取用户信息，单例模式，存储使用 SharedPreferences

程序中的使用如：

```
    UserData.getInstance().getUserId();
    UserData.getInstance().getPassword();
```

```
UserData.getInstance().setUserId(sign_in_username.getText().toString());
UserData.getInstance().setPassword(sign_in_password.getText().toString());
UserData.getInstance().setLoginState(true);
```

8.Http 接口（参见后台技术文档）

4 总结

易助作为一款便利城市社区中居民生活的 APP，优点是非常明显的：它提供一键求救、求助、提问三个功能。针对社区中的老人，残障人士和其他居民，帮助他们应对生活中的突发事件和其他日常困难。在城市社区中推广易助，搭建互

助社区，简化帮助形式以提高互助效率，增大用户量，使居民愿意求助，乐于帮助有需要的人。

但同时我们也看到，易助并非完美，它也存在着短板和不足。例如只是实现了具体功能而没有考虑到其中的信用信誉机制，无法保证整个帮助过程是值得信赖的。因此，易助可能更适合生活中不太紧急问题的救助，在未来还需要进一步完善信用监督体系。我们也相信在不久的将来公益互助 APP 在社会中会取得更广泛的应用。