广东工业大学

华立学院

# 本科毕业设计（论文） 基于即时通讯技术协商会管理的APP设计

论文题目

学 部 信息与工程学部

专 业 计算机科学与技术

班 级 13本计算机1班

学 号 511321010110

学生姓名 张松周

指导教师 程东胜

年 月 日

#### **摘 要**

如今，借助互联网技术的广泛应用，人们之间的联系通讯不再受到距离的限制。正因如此，人们在各个行业上对沟通交流的需求和依赖也越来越强烈，对需求业务一往如既地趋向人性化。通过各种的即时通讯软件，人们可以足不出门的情况下，与世界各地的人们进行实时聊天交流，并且通过翻译软件进行语言无障碍沟通。同时即时通讯技术使得人与人之间的交流带来了极大的便捷和大量的经济节省，促进了世界的快速发展。因此，即时通信在人们的生活工作和学习中得到了更广泛的应用。

著名的脸书、推特、qq、微信等知名的即时通讯软件当今的互联网上已得到了广泛的认可，无论在生活上，还是在工作上都是极大方便人们的交流和管理。同时适应各种场合的各式即时通信软件也开始出现在软件市场上，这就包括了一些与企业管理相关的即时通信软件，具有企业特色的信息管理和即时通讯，，，，反射式光纤位移传感器由于具有原理简单、实现容易、工作可靠等诸多优点而受到越来越广泛的重视。本系统由于要同时兼顾高精度和大量程的要求，因此在反射式光纤位移传感器的一般原理上进行了新的设计，使它较好的达到了实际的设计要求。鉴于本项目中光纤传感头的设计与实现工作已经基本完成，本文主要侧重于对电路部分的设计与调试工作进行描述。

**绪 论**

**开发背景**

伴随着互联网技术的快速发展，移动通讯技术已经进入了飞跃性的时代。软件和硬件的技术不断突破，通讯的传输速率和质量有大幅的提高，极大地改善人们的日常生活水平。以智能应用，比如手机、电脑、平板等为代表的移动互联终端，成为了在日常工作生活中人们体验移动服务的重要手段。 特别是即时通讯技术，已经融入了日常生活中的方方面面。在国内，随着腾讯为代表的QQ和微信软件的流行，即时通讯应用已经成为人们交流的一种重要工具。

随着生活质量的提升，人们渐渐地对文字、语音、图片这种实时通讯不感到满足，视频的实时通讯越来越受到人们的青睐。经过几年的迅速的发展，随着通讯的功能日益丰富，它不再是个单纯的聊天工具，它已经发展成集交流、资讯、娱乐、音乐、电视、游戏、电子商务等为一体的综合化信息平台。

即时通讯技术不同于E-mail在于它的交谈时即时的，大部分的即时通讯服务提供了状态信息的特性——显示联络人名单，联络人是否在线，能否与联络人交谈。现在，即时通讯软件，是以企业内部办公，建立员工交流平台为基础，通过整合、边缘功能，为企业提供一整套的囧事沟通和实时协作的解决方案。由于企业对信息类软件的需求还在探索与尝试阶段，而且现在这些软件也存在许多的安全隐患，但是优越性高始终被选用的重要条件。

**现状**

即时通讯软件，几乎是每个企业或者说是每个人上网的必备的联络工具。即时通讯可以分为个人即时通讯和企业即时通讯。个人即时通讯在中国最有影响的首推腾讯。旗下产品QQ和微信应用，不只聊天功能强大，注重保护个人隐私，其次的娱乐性也很强。但是它的竞争对手还是比较多，不同的用途也是一门应用，比如阿里巴巴，是专注商人和客户之间的聊天，这与现在风靡全球的网购有关；飞信是中国移动推出的服务；YY是专门对网络玩家创办的聊天系统；人人是专注设计给在校生的即时通讯系统，国内还有很多其它android软件供于用户使用。综上所述可以看出，人们早已经被即时通讯包围，并适应熟悉使用。未来的安卓即时通讯系统会越来越人性化。同时，协商业的即时通讯功能会更加庞大。

**目的意义：**

对于现在科学技术的快速发展，掌握了移动互联网就是掌握了未来的发展。通过做这份基于安卓的即时通讯技术是希望本人能更好地掌握计算机互联网知识，同时做商业管理软件附加功能不断完善，有助于对本人Android技术水平的提升，更加熟练解决安卓初级常见问题，以及更好了解商业项目的需求，不断完善项目功能，提高用户体验度。

经过这个毕业设计的磨炼，对互联万安卓行业有了更深的了解，也有更多的知识来充实自己的大脑。集成多媒体的商业管理应用有着更强的吸引力，为用户提供更多个性化的服务，将成为未来移动即时通讯发展的一个趋势。

**应用相关的技术与理论**

**语言：**java，C++，SQL，JavaScript+html，

**平台：**Android Studio ,javaee，eclipse

**开发系统：**苹果mini，windows

**调试测试机型：**小米2s（Android5+），华为（Android4.3），三星（Android 4.4），小米4（Android6.0），以及编译器自带模拟器

**1 Android的介绍：**

Android是Google开发基于Linux平台的开源手机操作系统。它包括操作系统、用户界面和应用程序——移动电话工作所需的全部软件，而且不存在任何以往阻碍移动产业创新的专有权障碍。Android采用WebKit浏览器引擎，具备触摸屏、高级图形显示和上 网功能，用户能够在手机上查看电子邮件、搜索网址和观看视频节目等，比iPhone等其他手机更强调搜索功能，界面更强大，可以说是一种全部Web应用的单一平台。

**2 Android系统架构：**

Android的系统架 构和其他操作系统一样，采用了分层的架构。Android分为四层，从高到低层分别是应用程序层、应用程序架构层、系统运行库层和Linux核心层。

1应用程序层：

Android会同一系列核心应用程序包一起发布，该应用程序包包括E-mail客户端、SMS短信程序、日历、地图、浏览器、联系人管理程序等。所有的应用程序都是使用java语言编写。

2：应用程序架构层：

该应用的架 构设计简化了组件的重用；任何一个应用程序都可以发布它的功能块并且任何其它的应用程序都可以使用其发布的功能块，不过这得遵循框架的安全性限制。同样，该应用程序重用机制也可以方便替换程序组件。

隐藏在每个应用后面的是一系列的服务和系统, 其中包括：

\* 丰富而又可扩展的视图(Views)，可以用来构建应用程序， 它包括列表(lists)，网格(grids)，文本框(text boxes)，按钮(buttons)， 甚至可嵌入的web浏览器。

\* 内容提供 器(Content Providers)使得应用程序可以访问另一个应用程序的数据(如联系人数据库)， 或者共享它们自己的数据

\* 资源管理器(Resource Manager)提供 非代码资源的访问，如本地字符串，图形，和布局文件( layout files )。

\* 通知管理器 (Notification Manager) 使得应用程序可以在状态栏中显示自定义的提示信息。

\* 活动管理器( Activity Manager) 用来管理应用程序生命周期并提供常用的导航回退功能。

3系统运行库：

1程序库：

　　 Android 包含一些C/C++库，这些库能被Android系统中不同的组件 使用。它们通过 Android 应用程序框架为开发者提供服务。

以下是一些核心库：

　　 \* 系统 C 库 - 一个从 BSD 继承来的标准 C 系统函数库( libc )， 它是专门为基于 embedded linux 的设备定制的。

　 \* 媒体库 - 基于 PacketVideo OpenCORE;该库支持多种常用的音 频、视频格式回放和录制，同时支持静态图像文件。编码格式包 括MPEG4, H.264, MP3, AAC, AMR, JPG, PNG 。

　　 \* Surface Manager - 对显示子系统的管理，并且为多个应用程序 提供了2D和3D图层的无缝融合。

　　 \* LibWebCore - 一个最新的web浏览器引擎用，支持Android浏览 器和一个可嵌入的web视图。

　 　 \* SGL - 底层的2D图形引擎

　　 \* 3D libraries - 基于OpenGL ES 1.0 APIs实现;该库可以使用硬 件3D加速(如果可用)或者使用高度优化的3D软加速。

　　 \* FreeType -位图(bitmap)和矢量(vector)字体显示。

　 　 \* SQLite - 一个对于所有应用程序可用，功能强劲的轻型关系型数 据库引擎。

2Android 运行库：

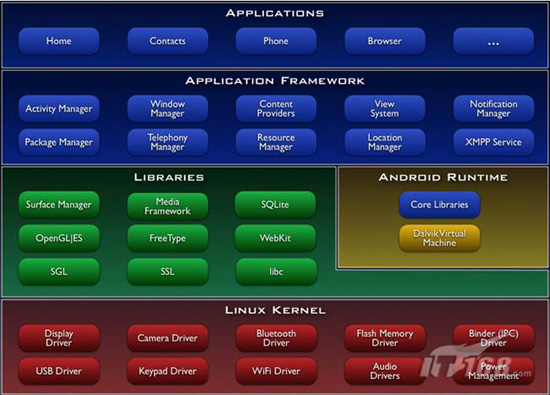
　　 Android 包括了一个核心库，该核心库提供了JAVA编程语言核心库 的大多数功能。

　　 每一个Android应用程序都在它自己的进程中运行，都拥有一个独 立的Dalvik虚拟机实例。Dalvik被设计成一个设备可以同时高效地运 行多个虚拟系统。 Dalvik虚拟机执行(.dex)的Dalvik可执行文件，该 格式文件针对小内存使用做了优化。同时虚拟机是基于寄存器的，所有 的类都经由JAVA编译器编译，然后通过SDK中 的 "dx" 工具转化成.dex 格式由虚拟机执行。

　　 Dalvik虚拟机依赖于linux内核的一些功能，比如线程机制和底层 内存管理机制。

4Linux 内核

Android 的核心系统服务依赖于 Linux 2.6 内核，如安全性，内存管理，进程管理， 网络协议栈和驱动模型。 Linux 内核也同时作为硬件和软件栈之间的抽象层。



Google提供的架构图

**3 javaEE后台服务器：**

javaee在JavaSE基础之上建立起来的一种标准开发架构，提供了一套设计、开发、汇编和部署企业应用程序的规范，来实现企业级应用程序。技术平台的核心思想是：容器+组件

**JavaEE体系结构：**

Applet—JSP+Servlet—EJB（重量级框架）。

组件是一个包含一定功能的软件单元，它有相关的类和文件一起组成，并与 其他组件进行通信。

EJB组件：分为会话EJB、实体EJB和消息驱动EJB；

EJB设计的缺陷：EJB采用的过程设计，不是面向对象设计。

EJB开发的问题：EJB开发和测试非常麻烦和冗长。

* 编辑、编译、调试周期长；
* 编码冗余、繁琐；
* 必须编写数据传输对象（DTO）。

    面向对象的设计更容易理解、维护、扩展和测试。

**基于MVC的轻量级框架：**

主流JavaEE框架：

表现层框架：SpringMVC框架，Struts2框架、JSF框架、Tapestry框架、WebWork框架

业务逻辑层框架：Spring框架

持久层框架：Hibernate框架、MyBatis

企业级应用需求：

通过将框架集成应用，可以发挥各框架的最大优势，良好的解决企业级应用的需求。

Struts+Spring+Hibernate框架（SSH框架）

SpringMVC+Spring+Hibernate框架

SpringMVC+Spring+MyBatis框架（SSM框架）

企业级应用新解决方案：框架集成应用

**4 应用第三方库**

即时通讯和直播：环信sdk

图片加载：Picasso

支持放大查看图片的控件：photoView

带左滑动菜单listview：swipemenulistview

网络请求库：okhttp

百度地图sdk

**系统的设计与实现**

**App结构介绍**

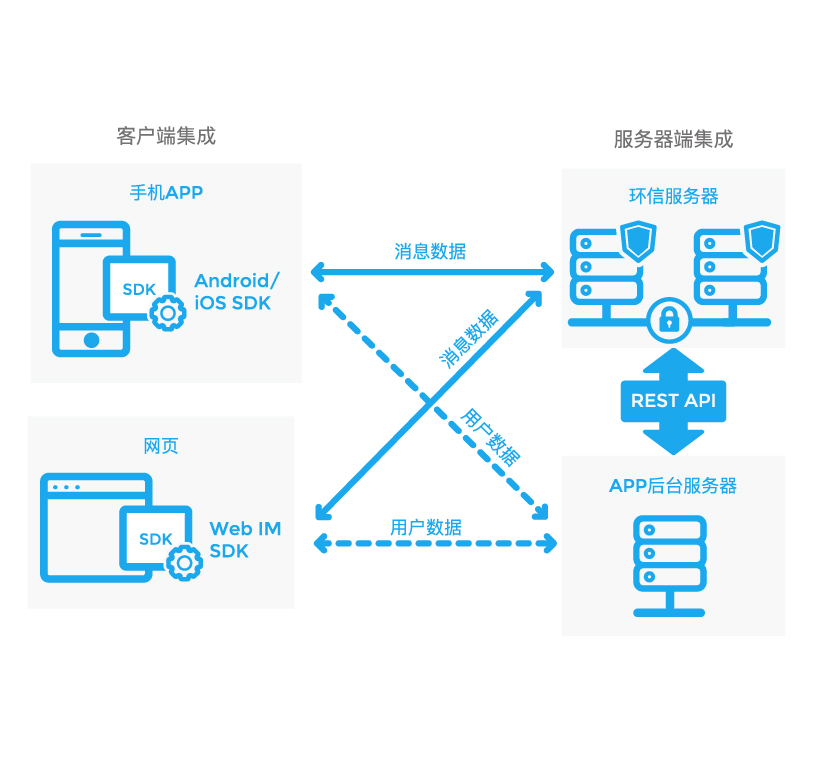
参考了微信app的交互流程，实现一个Android平台及时通讯系统，实现各类型客户端之间的互操性功能，提供好友的实时状态，随时随地与他人进行即时通讯交流。

主要基于java语言在Android Studio上调试编译，gradle项目自动化建构打包，build脚本编写。

项目整合了Android常用的sdk为开发基础，参考了商业项目开发流程以及框架技术，根据自身的技术水平，不断完善app功能，涉及到Android初级到中级的技术水平，界面表现形式主要是Android常用控件（TextView、EditeText、ImageView、Button、ListView、RadioButtom等），期间制作了列表的下拉刷新和上拉加载等的操作；对接javaee后台服务器，异步缓存请求网络json数据，实时更新界面UI；三级图片框架缓存；对于缓存数据操作，涉及到Android的sqlite和sharepreference库。

**第三方库介绍：**

环信：是一家全通讯能力云服务提供商；该项目 用到的版本是V3.0，新型的通信协议（基于消息同步的私有协议，在不稳定网络环境下更稳定更省流量，确保消息投递可靠、顺序以及 实时性，并具有更高的安全性。同时提供了更好的拓展性，将支持更多的对接和设备同步场景）；全新sdk（全面重构，将核心通信 模块做了更好的封装；简化了接口，结构更清晰，集成更容易，提升了登录速度和弱网络环境下的可靠性）。

平台架构

OKHttp：引用官方的自我介绍来说：HTTP is the way modern applications network. It&rsquo;s how we exchange data & media. Doing HTTP efficiently makes your stuff load faster and saves bandwidth。Okhttp是Android一款优秀的Http框架，支持和post请求，支持基于Http的 文件上传和下载，支持加载图片，支持下载文件透明的GZIP压缩，支持响应缓 存避免重复的网络请求，支持使用连接池来降低响应延迟问题。

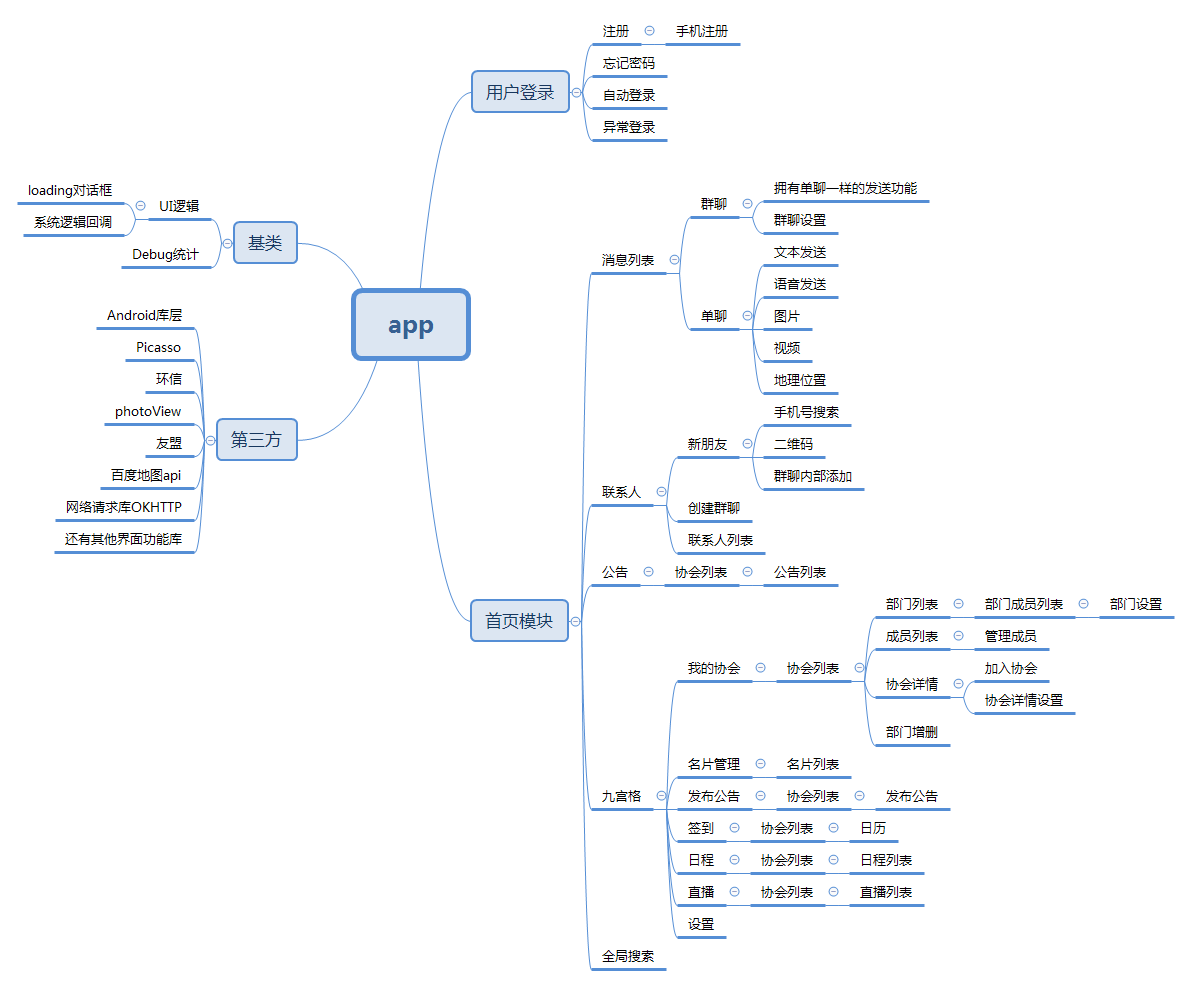
Picasso：Square公司出品的一款非常 优秀的开源图片加载库，是目前Android开发中比较 流行的图片加载库之一。解决使用复杂的图片压缩转换来尽可能的减少内存消耗，而且还自带内存和硬盘二级缓存功能。

百度地图sdk：基于Android2.3及 以上版本设备的应用程序接口，适用于Android系统移动设备的地图应用， 通过调用地图sdk可以轻松访问百度地图服务和数据，构建功能丰富、交互性强的地图类应用程序。

友盟：第三方全域大 数据服务提供商，通过全面覆盖PC、手机、传感器、无线路由器等多种设备数据，打造全 域数据平台。提供全业务链数应用解决方案，包括基础统计、运营分析、数据决 策和数据业务等，帮助企业实现数据化运行和管理。

其他第三方库：都是Github优秀开源项目。

**app结构图：**

****

**App适用Android版本:**

适用Android最低版本为16（ Android 4 . 1 ），最高版本为22 （ Android 5 . 1 ）

< uses - sdk

Android : minSdkVersion = " 14 "

Android : targetSdkVersion = " 2 2 " / >

**APP程序入口：**

程序启动的入口第一个界面：

< activity

Android : name = "app.logic.activity.launch.LaunchActivity"

Android : screenOrientation = " portrait "

Android : theme = " @style / windowallwhite " >

<intent-filter>

<action android : name = " android . Intent . Action . MAIN " / >

< category android : name = " android.intent . category . LAUNCHER " / >

</intent-filter>

</activity>

**App交互流程浏览：**

**启动页：**

使用Viewpager装载多张图片来向用户介绍该app的大体功能和用户

主要代码：

((TextView)view01.findViewById(R.id.launch\_tv01)).setText("协作从格局开始"); ((TextView)view01.findViewById(R.id.launch\_tv02)).setText("发布需求、创建组织、管理部门成员");

((TextView)view01.findViewById(R.id.launch\_tv03)).setText(" 一步到位");

(TextView) view02.findViewById(R.id.launch\_tv01)).setText("创建自己的工作圈");

((TextView)view02.findViewById(R.id.launch\_tv02)).setText("组织成员、客户伙伴各种人脉快速找到");

((TextView)view02.findViewById(R.id.launch\_tv03)).setText("协作关系更加轻松");

((TextView)view03.findViewById(R.id.launch\_tv01)).setText("圈内公告");

((TextView)view03.findViewById(R.id.launch\_tv02)).setText("随时随地查看最新动态，收纳繁琐消息");

((TextView)view03.findViewById(R.id.launch\_tv03)).setText("不易遗漏错过");

((TextView)view04.findViewById(R.id.launch\_tv01)).setText("组织管家");

((TextView)view04.findViewById(R.id.launch\_tv02)).setText("一目了然，随我掌控");

((TextView)view04.findViewById(R.id.launch\_tv03)).setText("开启智能化管理");

****

**登录页：**

这里是主要的登录方法，多重检查，人性化判断懒惰的操作。

这也是用户信息的缓存操作，单一用户信息全局化

Android轻量级缓存share preference，存储用户id和token信息

/\*\*

\* 用户登录方法

\*

\* @param user name

\* @param pass word

\*/

Private void login ( final String username, final String password ) {

showWai tDialog ( ) ;

if ( ! Username.equals ( " " ) && ! Password .equals (" " ) ) {

User ManagerController.Login(LoginActivity.this, username, password, new Listener < Integer, UserInfo > ( ) {

@Override

public void on CallBack(Integer status, UserInfo reply ) {

if ( status == 1 ) {

// 登录成功

QLConstant.client\_id = reply.getWp\_member\_info\_id ( ) ;

QLConstant.user\_picture\_url = reply.get Picture\_url ( ) ;

JPushI terface.setAlias ( LoginActivity.this, reply.getWp\_member\_info\_id ( ) , null ) ;

utils.setUserName ( username ) ; //本地保存用户名

utils.setPassword(password); //本地保存密码

// im聊天登录

String chartpsw = "wudi#" + username;

new EMOptions().setAutoLogin ( true ) ;

EMClient.getInstance ( ) . Login ( username, chartpsw, new EMCallBack ( ) {// 回调

@Override

public void onSuccess ( ) {

Run On UiThread(new Runnable ( ) {

public void run ( ) {

Log.v("hhhh", "im登陆成功");

//从本地加载群组列表

EMClient . getInstance ( ).groupManager ( ).getAllGroups ( ) ;

EMClient.getInstance ( ) .chatManager ( ) . loadAllConversations ( ) ;

// EMGroupManager . getInstance().loadAllGroups ( ) ;

/ / EMChatManager.getInstance ( ) . loadAllConversations ( ) ;

Mhanler . sendEmptyMessage ( 0 0 0 ) ;

}

});

}

@Override

public void onProgress(int progress, String status ) {

}

@Override

public void onError ( int code, String message ) {

/ / Message msg = new Message();

/ / msg.what = 111;

/ / msg.obj = message;

/ / mhanler.sendMessage(msg);

/ / Log.v ( "h h h h " , "登陆聊天服务器失败！" ) ;

if ( code == 200 ) {

runOnUiThread ( n ew Runnable ( ) {

public void run() {

Log. V ( " h h h h " , " i m 登 陆 成 功 " ) ;

//从本地加载群组列表

EMClient . getInstance ( ) . groupManager() . getAllGroups ( ) ;

EMClient . getInstance ( ) .chatManager ( ) . l oadAllConversations ( ) ;

/ / EMGroupManager.getInstance().loadAllGroups ( ) ;

/ / EMChatManager . getInstance().loadAllConversations ( ) ;

Mhanler. sendEmptyMessage ( 00 0 ) ;

}

} ) ;

} else {

Message msg = new Message ( ) ;

Msg . what = 111;

Msg .obj = message ;

Mhanler .sendMessage ( m s g ) ;

Log .v("hhhh", "登陆聊天服务器失败！" ) ;

}

}

} ) ;

} else if ( status == - 1 ) {

dismissWaitDialog ( ) ;

Toast . makeText ( LoginActivity.this, " 请检查账号或者密码是否输入正确！", Toast.LENGTH\_SHORT ) .show () ;

}

}

});

} else {

dismissWaitDialog ( ) ;

Toast . makeText ( LoginActivity .this, " 用户名或者密码不能为空", Toast . LENGTH\_SHORT) . Show ( ) ;

}

}

**消息列表：**

获取环信数据列表，通过按照时间戳的排序，最新消息依次置顶，同时回调和统计未读消息的数量，点击对话聊天后，自动将消息置零或不显示。

当消息列表上没有对方消息时，对方发来消息时调用服务器接口，将对方注册在对话列表上。

/\*\*

\*获取列表消息

\*\*/

private void loadMessageList ( ) {

sessionList.clear ( ) ;

UserManagerController . getChatList ( getActivity ( ).getApplicationContext ( ), new Listener < Integer, List < YYChatSessionInfo > > ( ) {

@Override

public void onCallBack(Integer status, List <Y YChatSessionInfo > reply ) {

ArrayList < YYChatSessionInfo > \_ t m p L i s t = new ArrayList < YYChatSessionInfo > ( );

int unReadCountTemp = 0;

listView.stopLoadMore ( ) ;

listView.stopRefresh ( ) ;

if ( r eply != null && reply.size ( ) > 0 ) {

for ( YYChatSessionInfo yyChatSessionInfo : reply ) {

if ( yyChatSessionInfo.getChatroom () == null ) { //单聊

// 过滤掉和自己的聊天记录

if ( yyChatSessionInfo. getWp\_member\_info\_id ( ) . Equals (y yChatSessionInfo.getWp\_other\_info\_id ( ) ) ) {

continue;

}

// sessionList.add(yyChatSessionInfo ) ;

// 计算未读

EMConversation conversation = EMClient . getInstance( ).chatManager( ) . getConversation(yyChatSessionInfo . getPhoneNumber ( ) ) ;

if (conversation != null ) {

unReadCountTemp += conversation.getUnreadMsgCount ( ) ;

}

} else { //群聊

EMConversation conversation = EMClient . getInstance() . chatManager().getConversation(yyChatSessionInfo.getChatroom ( ) .getRoom\_id ( ) );

if (conversation ! = null ) {

unReadCountTemp += conversation. getUnreadMsgCount ( );

}

}

\_ tmpList . add( yyChatSessionInfo );

}

/ / 设置未读数量

List< YYChatSessionInfo > sortList = sortYYChatSessionInfos( setUnreadYYChatSessionInfos( \_tmpList ) );

sessionList.addAll(sortList);

if (sortList == null || sortList.size() < 1) {

nullDataView. setVisibility ( View.VISIBLE );

} else {

nullDataView. setVisibility( View.GONE );

}

adapter.setDatas ( sortList );

// 读完的时候回调下,刷新Tab

ZSZSingleton.getZSZSingleton( ).getStatusMessageListener( ).callbackStatusUpdata( unReadCountTemp );

} else {

nullDataView.setVisibility( View.VISIBLE );

}

}

});

}

// 判断对话列表是否有对方信息

private void haveMessageList( final String phone, final EMMessage message ) {

UserManagerController.getChatList( getActivity(), new Listener< Integer, List < YYChatSessionInfo > >() {

@Override

public void onCallBack ( Integer status, List < YYChatSessionInfo > reply ) {

if (reply == null || reply.size( ) < 1) {

boolean temp = true;

for ( YYChatSessionInfo info : reply ) {

if (info.getPhoneNumber( ).equals(phone)) {

temp = false;

break;

}

}

if ( temp ) {

addChart( phone, message );

} else {

msgHandler.sendEmptyMessage ( kRefrashUI );

}

}

}

});

}

// 获取用户信息，注册回话

private void addChart ( String phone, final EMMessage msg ) {

UserManagerController.getPhoneMemerInfo ( mContext, phone, new Listener < Integer, List < UserInfo > > ( ) {

@Override

public void onCallBack( Integer status, List < UserInfo > reply ) {

if ( reply != null && reply.size( ) > 0) {

String latestChart = null;

if ( msg != null && msg.getBody( ) != null ) {

if (msg.getBody() instanceof EMTextMessageBody ) {

latestChart = ( ( EMTextMessageBody ) msg.getBody( ) ).getMessage( );

} else if ( msg.getBody( ) instanceof EMLocationMessageBody ) {

latestChart = ( ( EMLocationMessageBody ) msg.getBody( ) ).getAddress ( );

} else if (msg.getBody( ) instanceof EMImageMessageBody ) {

latestChart = "图片[ " + ( ( EMImageMessageBody ) msg.getBody( ) ).getFileName( ) + " ] ";

} else if ( msg.getBody( ) instanceof EMVideoMessageBody ) {

latestChart = " [ 视频 ] ";

} else if ( msg.getBody( ) instanceof EMVoiceMessageBody ) { latestChart = " [ 语音 ] "; } }

UserManagerController.addChatWith( getContext( ), reply.get( 0 ).getWp\_member\_info\_id( ), latestChart, new Listener < Integer, String >( ) {

@ Override

public void onCallBack (Integer status, String reply ) {

loadMessageList( );

}

} );

}

}

});

}



**联系人：**

包括新建群聊操作、添加好友操作（可以通过手机号查找或者个人信息生成二维码，对方通过扫描解析里面的二维码信息，读取到json字段对应的手机号码）、联系人列表展示

对于联系人列表，主要先是对数据列表的按照拼音排序，然后对采取首个字节的拼音首字母进行分组，相同字母进行分块处理，最终在listView上多布局显示。

//获取联系人列表

private void loadContacts( ) {

UserManagerController.getFriendsList( getActivity( ), new Listener < List < FriendInfo >, List < FriendInfo > > ( ) {

@Override

public void onCallBack ( List < FriendInfo > request, List < FriendInfo > reply ) {

ArrayList < FriendInfo > tmpInfos = new ArrayList < FriendInfo > ( );

if ( reply != null && reply.size( ) > 0 ) {

String myPhone = UserManagerController.getCurrUserInfo ( ) .getPhone ( ) ;

for ( FriendInfo friendInfo : reply ) {

if ( friendInfo.getPhone( ) != null && !friendInfo.getPhone( ).equals ( myPhone ) ) {

if ( friendInfo.isResponse( ) && friendInfo.isRequest\_accept ( ) ) {

friendInfo.setOtherRequest ( false );

tmpInfos.add( friendInfo );

}

}

}

}

String [ ] data = new String [ tmpInfos.size() ] ;

for ( int i = 0; i < tmpInfos.size( ); i++) {

Data [ i ] = tmpInfos.get ( i ) .getNickName( );

}

SourceDateList.clear ( ) ;

SourceDateList.addAll ( filledDataList ( tmpInfos ) ) ;

//自定义排序

Collections.sort ( SourceDateList, pinyinComparator ) ;

adapter.notifyDataSetChanged ( ) ;

if (SourceDateList.size() > 0 ) {

empty\_view.setVisibility( View.GONE );

} else {

empty\_view.setVisibility(View.VISIBL E);

}

}

});

}

/\*

填充数据

\*/

private List< SortModel > filledDataList ( List < FriendInfo > models ) {

List < SortModel > list = new ArrayList < SortModel > ( ) ;

for (int i = 0; i < models.size(); i++) {

SortModel sortModel = new SortModel ( models.get ( i ) ) ;

String tempString = models.get ( i) .getFriend\_name ( ) == null || TextUtils.isEmpty(models.get ( i ). getFriend\_name ( ) ) ? Models.get ( i ). getNickName ( ) : models.get ( i ).getFriend\_name ( ) ;

sortModel.setName ( tempString ) ;

String pinyinString = characterParser.getSelling ( tempString ) ;

String sortString = pinyinString.substring ( 0 , 1 ) . toUpperCase ( ) ;

if ( s ortString.matches ( "[ A - Z ] " ) ) {

sortModel.setSortLetters ( sortString.toUpperCase ( ) );

} else {

sortModel.setSortLetters ( " # " ) ;

}

List.add ( sortModel ) ;

}

return list;

}

**公告：**

协会列表到公告列表，动态排序，即时回调其他发布公告消息，快速刷新未读数，以及点击后刷新状态。也能通过点击通知栏的消息推送，调到指定的公告详情。

/\*\*

\* 获取发了公告的组织

\*/

private void getMyOrg() {

// ( ( T Y Ba s e A c t iv i t y ) context ) . showWaitDialog ( );

OrganizationController.getOrgUnreadNumber ( g e t C on text(), new Listener < Integer, List < OrgUnreadNumberInfo > > ( ) {

@Override

public void onCallBack(Integer status, List<OrgUnreadNumberInfo> reply) {

yyListView.stopRefresh();

// ( ( T Y B a seA ctivity ) context).dismissWaitDialog();

Datas.clear ( ) ;

if ( reply != null || reply.size ( ) > 0) {

datas.addAll ( reply ) ;

}

if( datas.size ( ) > 0 ) {

empty\_view.setVisibility ( View.GONE ) ;

}else{

empty\_view.setVisibility ( View.VISIBLE ) ;

}

adapter.setDatas ( datas ) ;

}

});

}

@Override

@Nullable

public View onCreateView ( LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState ) {

view = inflater.inflate ( R.layout.fragment\_notice\_list, null ) ;

fragmentTabHost = ( Fra g m e n tT a b H o s t ) view. findViewById ( R . I d . N o t i c e \_f tg ) ;

fragmentTabHost.setup ( view.getContext ( ) , getChildFragmentManager ( ) , R.id.notic\_fragment ) ;

View unreadView = getIndicatorView ( " 未 读 " , getResources().getDrawable(R.drawable.selector\_tabnotice\_unread ) ) ;

fragmentTabHost.addTab ( fragmentTabHost.newTabSpec ( " unread " ) .setIndicator ( unreadView ) , FragmentUnread.class, null ) ;

View readView = getIndicatorView ( " 已 读 " , getResources ( ).getDrawable ( R.drawable.selector\_tabnotice\_unread ) ) ;

fragmentTabHost.addTab ( fragmentTabHost.newTabSpec ( " r ead " ) . setIndicator (r eadView ), FragmentRead.class, null);

// getActivity ( ).getResources ( ) .getDimension ( R.dimen.dabc\_action\_bar\_default\_height\_material ) ;

// getActivity().getActionBar().getHeight ( ) ;

// 解决tabhost会被压缩问题

fragmentTabHost.getTabWidget().getChildAt ( 0 ) .getLayoutParams ( ) .height = ( i n t ) getActivity ( ).getResources ( ) .getDimension (R . D i m e n . A c t i o n b a r \_h e i g h t) ;

fragmentTabHost.setCurrentTab ( 0 ) ;

fragmentTabHost.setOnTabChangedListener(new OnTabChangeListener ( ) {

@Override

public void onTabChanged(String tabId) {

// for ( i n t i = 0 ; i < fragmentTabHost.getChildCount ( ) ; i + + ) {

// System.out.println ( fragmentTabHost.getTabWidget ( ) .getChildAt ( i ) .getTag ( ) ) ;

// }

}

});

return view;

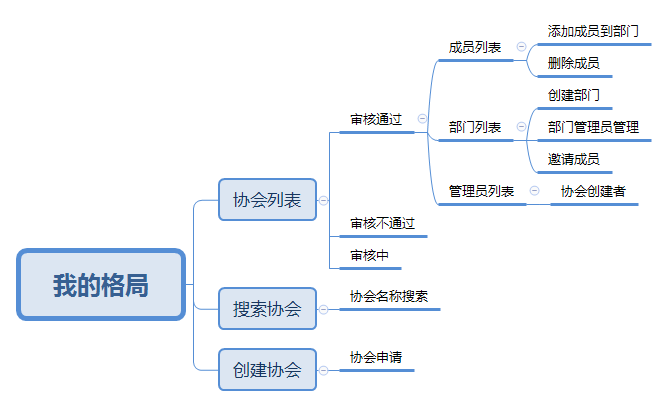
}



**九宫格：**

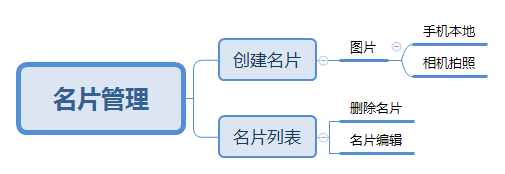
九宫格页面包括了个人设置，这里头像的点击是图片放大的操作，点击整个背景是直接进入个人设置。

我的格局：这里是管理所有协会的入口，包括创建协会，搜索协会信息，加入协会，管理协会等



主要格局流程

名片管理：包括创建名片，名片列表，图片主要有两种形式获取，一种是手机本地，另一种调用本地手机拍照；名片列表主要展现当前所有名片信息，可删除，可编辑。



发布公告：

经过选择协会列表发布公告，如果没有协会列表则可以点击创建协会，带通过后可发布公告；发布公告可选择图片发布，图片主要以手机本地和手机拍照两种方式获取，内增加了裁剪框，自动生成缩略图，带编辑文本后可发布公告。发布公告后，会推送一条消息，所属该协会的人便可收到该公告信息。

