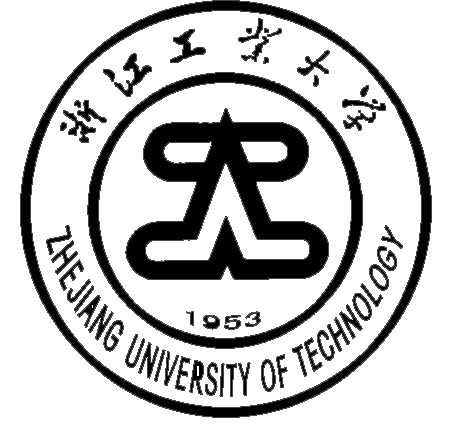


本科毕业设计说明书（论文）

**（2017届）**



论文题目 基于VOIP协议的优惠电话APP开发与设计

作者姓名 雷帮文

指导教师 徐利锋

学科(专业) 数字媒体技术

所在学院 计算机科学与技术学院

提交日期 2017年6月

[摘要 I](#_Toc13344)

[Abstract II](#_Toc11299)

**[第一章 绪论](#_Toc1226)** [1](#_Toc1226)

[1.1 课题背景及意义 1](#_Toc21555)

**[1.2 国内外现状](#_Toc18313)** [2](#_Toc18313)

**[1](#_Toc16323)**  [2](#_Toc16323)

**[2.1](#_Toc15247)**  [2](#_Toc15247)

**[1.3 系统开发目标](#_Toc18737)** [3](#_Toc18737)

**[1.4 开发基本内容](#_Toc20971)** [3](#_Toc20971)

**[1.5 功能实现的基础](#_Toc17615)** [4](#_Toc17615)

**[1.6 本文的主要工作](#_Toc7968)** [4](#_Toc7968)

**[1.7 本文的组织结构](#_Toc10403)** [4](#_Toc10403)

**[1.8 本章小结](#_Toc28261)** [5](#_Toc28261)

**[第二章 系统概要设计](#_Toc13103)** [6](#_Toc13103)

**[2.1 系统架构](#_Toc11216)** [6](#_Toc11216)

**[2.2 系统数据库设计](#_Toc4448)** [7](#_Toc4448)

**[2.3 本章小结](#_Toc17666)** [7](#_Toc17666)

**[第三章 通讯录模块开发与设计](#_Toc17690)** [9](#_Toc17690)

**[3.1 通讯录模块介绍](#_Toc23563)** [9](#_Toc23563)

**[3.2 通讯录模块设计](#_Toc22777)** [9](#_Toc22777)

**[3.3 通讯录模块开发](#_Toc18082)** [11](#_Toc18082)

**[3.4 本章小结](#_Toc4922)** [16](#_Toc4922)

**[第四章 钱包模块设计与开发](#_Toc25574)** [17](#_Toc25574)

**[4.1 钱包模块介绍](#_Toc10487)** [17](#_Toc10487)

**[4.2 钱包模块功能设计](#_Toc25912)** [17](#_Toc25912)

**[4.3 钱包模块UI设计](#_Toc15371)** [17](#_Toc15371)

**[4.4 钱包模块数据库设计](#_Toc19194)** [19](#_Toc19194)

**[4.5 钱包模块开发](#_Toc355)** [22](#_Toc355)

**[4.6 本章小结](#_Toc7222)** [25](#_Toc7222)

**[第五章 电话模块设计与开发](#_Toc6789)** [26](#_Toc6789)

**[5.1 电话模块介绍](#_Toc22643)** [26](#_Toc22643)

**[5.2 电话模块设计](#_Toc17548)** [26](#_Toc17548)

**[5.3 电话模块UI设计](#_Toc23878)** [26](#_Toc23878)

**[5.4 电话模块数据库设计](#_Toc20474)** [28](#_Toc20474)

**[5.5 电话模块开发](#_Toc1380)** [30](#_Toc1380)

**[5.6 本章小结](#_Toc12292)** [36](#_Toc12292)

**[第六章 总结](#_Toc13627)** [37](#_Toc13627)

[6.1 完成的工作 37](#_Toc9537)

[6.2 存在的问题及下一步工作 37](#_Toc26974)

摘要

近几年来，伴随着计算机技术和互联网的高速发展，传统语音通讯方式已经不能满足于 我们的日常生活通信需求，传统的语音通讯app功能比较单一，往往只有拨号功 能，联系人管理，并且界面比较简单，同时传统的电话通讯网络很难与现在的Intenet网络进行交互，并且在中国的传统通话业务被联系、移动、电信三家巨头所垄断，通话费用居高不下，而随着英特网的普及，IP技术已经被运用到了各个行业，VOIP协议就是这样的一个产物，可以在IP 网络上使用 IP 协议以数据包的方式传输语音，这样相对于传统的语音通话，通话资费则便宜了很多。并且使用IP网络来进行语音数据传输，可以进行无限扩展，达到传统语音通话实现不了的效果。

本文则以符合VOIP协议的第三方SDK为基础，介绍了开发和设计一个完整的网络电话APP的流程。基于第三方的支付平台，开发设计了一个完整的支付订单管理系统，用户可以充值话费，可以拨打电话消费话费和余额等功能，并且可以自由设置用户拨打电话的通话费率，使语音通话和我们的业务进行整合，开发一个可玩性和实用性兼备的APP。

**关键词：**VOIP，网络电话，通讯录，订单管理

Abstract

In recent years, with the rapid development of computer technology and the Internet, the traditional way of voice communication has been unable to meet our daily life communication needs, the traditional voice communication app function is relatively simple, often only dial-up function, contact management, and interface Relatively simple, while the traditional telephone communication network is difficult to interact with the current Intenet network, and in China's traditional call business is linked, mobile, telecommunications monopoly of the three monopoly, call costs remain high, and with the Internet Popularization, IP technology has been applied to various industries, VOIP protocol is such a product, you can use the IP network IP protocol to send voice packets, so compared to the traditional voice calls, call charges are much cheaper. And the use of IP network for voice data transmission, can be infinite expansion, to achieve the traditional voice calls can not achieve the effect.

Based on the third party SDK which conforms to the VOIP protocol, this paper introduces the process of developing and designing a complete network telephone APP, and designs and designs a complete payment order management system based on the third party payment platform. The user can recharge the bill , You can call the phone calls and balances and other functions, and can freely set the user to call the call rate, voice calls and our business integration, the development of a playability and practical both APP.

**Keywords：**voip, internet phone, address book, order management

# **第一章 绪论**

## **1.1 课题背景及意义**

VOIP可以为用户提供更加便捷和便宜的增值业务，内容更加的丰富，服务更加的全面，主要表现在，可以通过IP网络对语音视频以及数据和传真进行传送，具体的业务有虚拟电话、统一消息、显号业务以及因特网的呼叫中心和呼叫管理等等。对于用户来说，低廉的资费是其受到吸引而使用VOIP的最主要原因。其产生低资费原因是在网络语音通信平台中，由于放弃了TCP传输控制协议，信令和语音数据的传输全部通过无连接的UDP协议实现，因此突破了在TCP 协议中最大连接数的限制，打破了网络连接的瓶颈，极大地降低了运营成本。所以使用网络电话拨打话费可以比普通固话便宜将近80%的费用。但是资费的价格只是片面的一个因素，不能成为促进VOIP进行持续发展的动因。并且在未来VoIP的发展趋势是向着家庭的综合信息服务方向进行发展的，因此这种服务会包括语音通讯以及家庭化的娱乐和安全保障部分。

VOIP在价格上的吸引力不仅仅是对于个人用户，对于企业同样是一种使用的影响因素。这些企业大多在长途的使用以及具有多个分支机构或者是具有驻外机构。下面就这些个人和企业的需求特点进行详细的论述。以便表明该APP的开发目的。

（1）降低通话资费

使用基于Voip协议开发的电话系统具有通话资费低廉的特点。其实电话系统的开发商如果不向用户收取费用的话，那么对于用户来说是完全免费的，虽然直拨或者回拨之前需要收取少量的流量向电话系统发起呼叫请求，但是这些是非常低廉的，可以完全忽略不计，但是实际上由于电信运营商需要收取电话系统的开发商的费用，所以开发商需要向用户收取部分的资费，但是这些资费相对于直接用普通电话拨打电话要优惠许多，通常为0.03元/分钟~0.1元/分钟，相对于普通电话的0.2元/分钟要优惠一半以上。

（2）丰富人人机交互界面以及通话业务与其它业务整合

目前大部分的网络电话app的界面都是非常简陋的，要么只有单一的拨号功能，要么存在大量的广告，丑陋的界面和繁琐单一的功能让用户不能获得丰富舒适的体验，该系统为了改善这一现状，会开发一个丰富并且人性化的用户界面供用户使用，并且相对于传统通话业务来说，传统通话功能受三大运营商垄断，如果要与APP本身的业务进行整合是非常困难的，如果使用了基于VOIP协议的通话业务，那么理论上是可以与APP本身的业务无缝整合，方便APP的功能扩展等。

（3）完善和丰富的通讯录管理和使用的生活小功能。

目前大部分的网络电话APP都是只有单一的通话记录管理以及全部调用系统自带的通讯录系统，导致通讯录界面看起来非常的简陋以及单调，开发出一个用户赏心悦目的APP界面，是该APP的初衷之一，APP中即要实现基本的优惠拨打电话的功能，也要实现一些用户可玩性的功能，以提高APP的用户粘性。

## **1.2 国内外现状**



我国IP电话业务市场已进入成熟期，资费优势是长途电话IP业务增长的直接原因，但随着传统长途电话资费的下降和移动长途电话漫游费的改革，IP电话业务将受到一定程度的冲击,但仍会保持超过传统长途电话的增长率。商务企业用户依旧主要被IP专用交换分机占领。对IP的管制策放松之后，VOBB宽带电话将获得快速的发展。即时通讯市场将会迎来更加激烈的竞争，新的移动IM的诞生会对传统的即时通讯工具产生一定的市场压力，但即时通讯的免费形式使它一直不能有一个好的盈利模式，但随着融合技术的发展，即时通讯将会逐渐走向标准化、移动化和多媒体化。

从市场角度来看，VOIP的准入门槛将越来越低，将从暴利行业向微利行业转变。对设备厂商来说，不得不面临从单纯设备生产商向综合解决方案提供商转化，只有围绕着应用和服务为核心的设备生产商才能生存下去。

家庭用户VoIP业务正快速融入数字家庭业务，企业客户VoIP业务正快速融入企业信息化业务，VoIP市场发展处于关键的变革期。对于VoIP业界运营类企业,如基础电信运营商、虚拟运营商，宽带接入商、IM运营商应该敏锐梳理客户需求、深入分析与借鉴国内外的成功案例、认清竞争态势，有效进行商业模式规划、价值链整合、针对市场需求与竞争环境开发产品与营销策略，使VoIP成为融合业务的基石，前瞻性地为企业铺垫与积累市场竞争优势。

**1.3 系统开发目标**

由于目前主要的通话方式被联通、电信和联通三大运营商所垄断，导致电话通话资费居高不下，为了迎合价格低廉，界面友好的通话软件需求而研究开发该项目。

**1.4 开发基本内容**

整个系统主要包括客户端和服务的，客户端主要负责与用户进行人机交互，服务端主要负责对数据进行存储，处理等等。APP的具体功能容包括以下几点。

（1）通话记录

一个完整的通话软件app最基本的功能之一，必须完整详细的记录下每一次通话，包括未接电话、已拨电话、已接电话等等的详细通话记录，让每一次通话有迹可寻，并且要对通话记录做好展示，分类和储存等等。

（2）联系人管理

为了方便用户对用户自己的通讯录进行管理，必须为app整合进一个完整和方便的通讯录管理系统，包括对联系人进行分类，联系人快速索引，查找等等功能，也要为用户提供高效直接的联系人信息展示和详细的联系人编辑。在联系人信息界面，还要为用户提供各种入口，方便用户拨打和联系该联系人。

（3）拨号系统

极具个性化和丰富的拨号系统，包括优美的界面切换特效，以及丰富的控制按钮和功能，拨号界面要包括拨号数字展示和已输入数字的快速联系人匹配等等。

（4）用户系统

用户需要消费，那么必须要拥有一个用户系统，提供用户注册、登录、消费等。

（5）钱包系统

用户每次拨打电话需要消耗一定金额的电话费，如果电话费用完了，那么用户可以通过支付宝充值后增加自己的话费余额以便继续拨打电话。那么就需要一个支付系统以便用户充值消费等。

## **1.5 功能实现的基础**

如果个人需要接入直接拨打电话的功能，几乎是不可能的，因为这需要与电信运营商沟通以接入，但是目前有一些第三方基于VOIP协议封装好的SDK可直接调用，当然这些SDK也只是提供了调用的借口，很多业务上的具体实现还是需要靠接入方自己开发，例如消费系统，支付充值系统，通话记录管理等等功能。该APP则以第三方SDK轻码云SDK为中心进行展开。

轻码云，深圳市轻码云科技有限公司（简称轻码云），是融合通讯能力PaaS（Platform as a Service,平台即服务)平台服务商。轻码云将基于互联网的语音、IM、视频、等融合通讯能力，通过简单、友好的API及客户端“SDK”的方式提供给开发者。简化移动通讯应用开发技术，最大的缩短通讯社交应用的开发周期。

## **1.6 本文的主要工作**

本文的主要工作是介绍该课题的研究背景和意义，然后说明了在设计和开发整个APP的过程中运用到的关键技术，包括功能设计，功能实现等。

## **1.7 本文的组织结构**

本文共分为六章，........，各章内容如下：

第一章，总结了课题研究的背景，国内外相关领域的研究及应用，课题研究的主要任务和本文的主要工作。

第二章，........。

第三章，........。

第四章，........。

第五章，........。

第六章，对本文进行总结并提出下一步工作。

## **1.8 本章小结**

本章简要分析项目的研究背景、在国内外相关领域的开发和应用现状以及项目的研究的任务和意义。最后，给出了本文的主要工作及本文的组织结构。

# **系统概要设计**

**2.1 系统架构**

钱包模块涉及到后台数据，所以整个实现的框架以C/S架构进行开发，客户端负责数据展示，部分逻辑需要写在服务端中，例如支付订单管理系统，消费系统等等。C/S架构，即Client/Server(客户机/服务器)结构，是大家熟知的软件系统体系结构，通过将任务合理分配到Client端和Server端，降低了系统的通讯开销，需要安装客户端才可进行管理操作。

客户端和服务器端的程序不同，用户的程序主要在客户端，服务器端主要提供数据管理、数据共享、数据及系统维护和并发控制等，客户端程序主要完成用户的具体的业务。开发比较容易，操作简便，但应用程序的升级和客户端程序的维护较为困难。

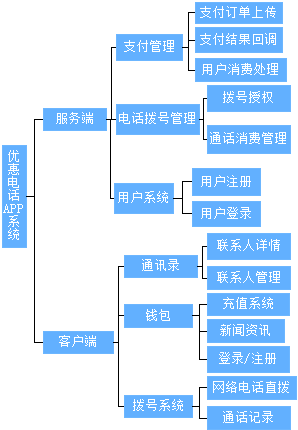
该APP系统总体系统架构如图2.1。其中服务端主要负责对充值时订单结果

图2.1 系统架构图

进行处理，并且记录和处理用户在使用APP的过程中的消费情况，并且在拨打电

话的时候，记录拨打电话的通话时间等来进行扣除相应的余额，拨号系统负责管理客户端发起的呼叫进行授权，用户系统则负责用户的注册，登录等等。

客户端则是进行联系人管理，钱包功能中集成了充值系统，用户登录注册和新闻资讯等功能，客户端的拨号系统负责发起拨号请求。

**2.2 系统数据库设计**

在小型的Android项目当中，常用的数据库是MySQL和SQLite数据库，而SQLite是APP本地存储数据的首选，相反的，MySQL是服务端存储数据的最近选择之一，这些数据库都具有轻量型的特定，开发和维护都非常简单，该系统的数据库也是使用这两者其中之一，MySQL，是一款轻型的数据库，是遵守ACID的关系型数据库管理系统，它包含在一个相对小的C库中。它是D.RichardHipp建立的公有领域项目。它的设计目标是嵌入式的，而且目前已经在很多嵌入式产品中使用了它，它占用资源非常的低，在嵌入式设备中，可能只需要几百K的内存就够了。它能够支持Windows/Linux/Unix等等主流的操作系统，同时能够跟很多程序语言相结合，比如Tcl、C#、PHP、Java等，还有ODBC接口，同样比起Mysql、PostgreSQL这两款开源的世界著名数据库管理系统来讲，它的处理速度比他们都快。该APP服务端使用MySQL作为存储数据数据库，整个数据库表单如表2-1所示。数据库的详细表字段将在相关章节中说明。

**2.3 本章小结**

本章主要描述了该APP的系统架构，然后说明了系统的数据库表单。以便后续章节对这些内容进行详细描述。

2‑1数据库表清单

| **序号** | 名称 | 说明 |
| --- | --- | --- |
|  | aswerfyempowerlog | 记录每次呼叫授权记录表 |
|  | bmorder | 支付订单回调记录表 |
|  | bmpayorderinfo | 支付订单支付结果表 |
|  | calllog | 通话结果记录表 |
|  | fyaskempowerlog | 呼叫请求授权记录表 |
|  | grantmoneylog | 赠送红包记录表 |
|  | mypayorder | 支付订单记录表 |
|  | phoneinfo | 用户电话数据表 |
|  | user | 用户基本数据表 |
|  | userinfo | 用户详细数据表 |

# **第三章 通讯录模块开发与设计**

## **3.1 通讯录模块介绍**

作为一款语音通讯类型的APP，通讯录是必不可少的基本功能。通讯录模块需要记录丰富的联系人数据。包括如：姓名、电话号码、单位电话、移动电话、电子邮件、QQ、个人主页、公司、街道、邮编、生日、爱好等等。这些详细的数据是为了用户能把相关的联系人信息记录合理地保存起来，方便之后用户查看联系人，并且还需要有通话记录功能。在该APP中，联系人模块数据是从手机读取的。这样是为了保证手机联系人和该APP通讯录数据同步，以便用户辨识和维护。

**3.2 通讯录模块设计**

**3.2.1 通讯录模块功能设计**

通讯录模块主要功能是展示联系人的数据列表，以便用户方便和快速索引联系人。展示联系人的数据列表，为了直观的显示联系人和快速索引，展示联系人的ListView列表的Item需要显示联系人头像和联系人的名字，并且还需要展示出收藏的联系人列表，为了快速索引到联系人，联系人列表还需要支持字母快速滑动索引的功能，还要支持添加联系人、删除联系人、编辑联系人和查看联系人详细数据，在联系人详情页面，需要支持头像显示、通话记录查看、快速拨号功能以及显示联系人的各个详细的数据，例如姓名、QQ号、地址、电话号码等等。

联系人数据与Android原生的联系人数据要同步，所以联系人数据的数据库使用系统数据库Contacts.db进行存取。

**3.2.2 通讯录模块UI设计**

应用的UI是给用户呈现应用特色的最直观的途径，如果应用的UI效果能吸引住用户，那么功能即使不是非常完善，也会给用户留下非常深刻的印象，所以UI设计是一个APP设计的非常重要的环节。

如图1是联系人展示列表的UI效果图，在顶部横向展示收藏列表，这样是为了对收藏的联系人快速查看和选取，而不需要做太多的滑动操作就可以选择联系人。而联系人列表的右边则是联系人快速索引的控件，通过滑动右边的字母控件，即可快速滑动到指定联系人，而联系人头像则是如果联系人存在头像，则显示联系人头像，如果不存在头像，则显示随机的颜色，以便APP色彩丰富性展示列表底部则是一个添加联系人的按钮，这是为了方便用户快速添加联系人。每个未收藏的联系人长按后都会弹出一个随机颜色的工具条，包括拨打电话、发消息、删除、编辑、分享号码这五个功能，如图2所示。点击任何的联系人，会进入联系人详情界面，如图3所示，顶部是联系人名称，包括返回上一页、收藏、编辑按钮。而下面则包括通话记录和联系人详细的数据。联系人详细数据首先是电话号码列表、邀请好友列表、详细的数据，每个电话的左右两边分别有两个按钮，对应于拨打电话和发送消息，邀请按钮主要是为了通过短信发送邀请短信邀请好友。每个模块都通过卡片式布局展示联系人详情。在联系人详情界面，收藏按钮点击后会变成实心的五角星，代表该联系人已经被收藏。联系人详情界面的色调会动态变化，根据联系人的数据特点显示不一样的颜色，如图4所示。整个联系人模块围绕蓝色调来展开，色彩丰富。

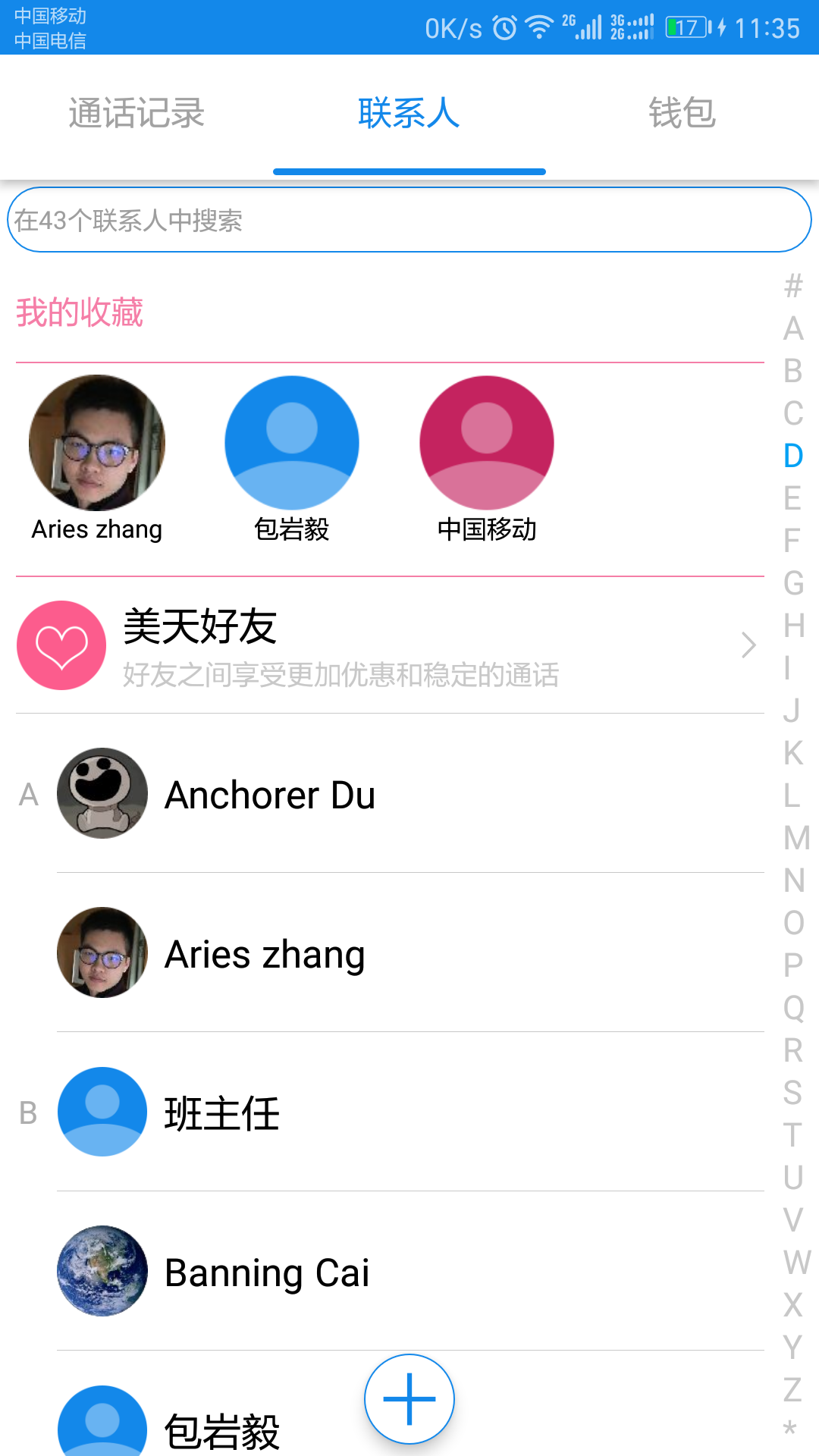
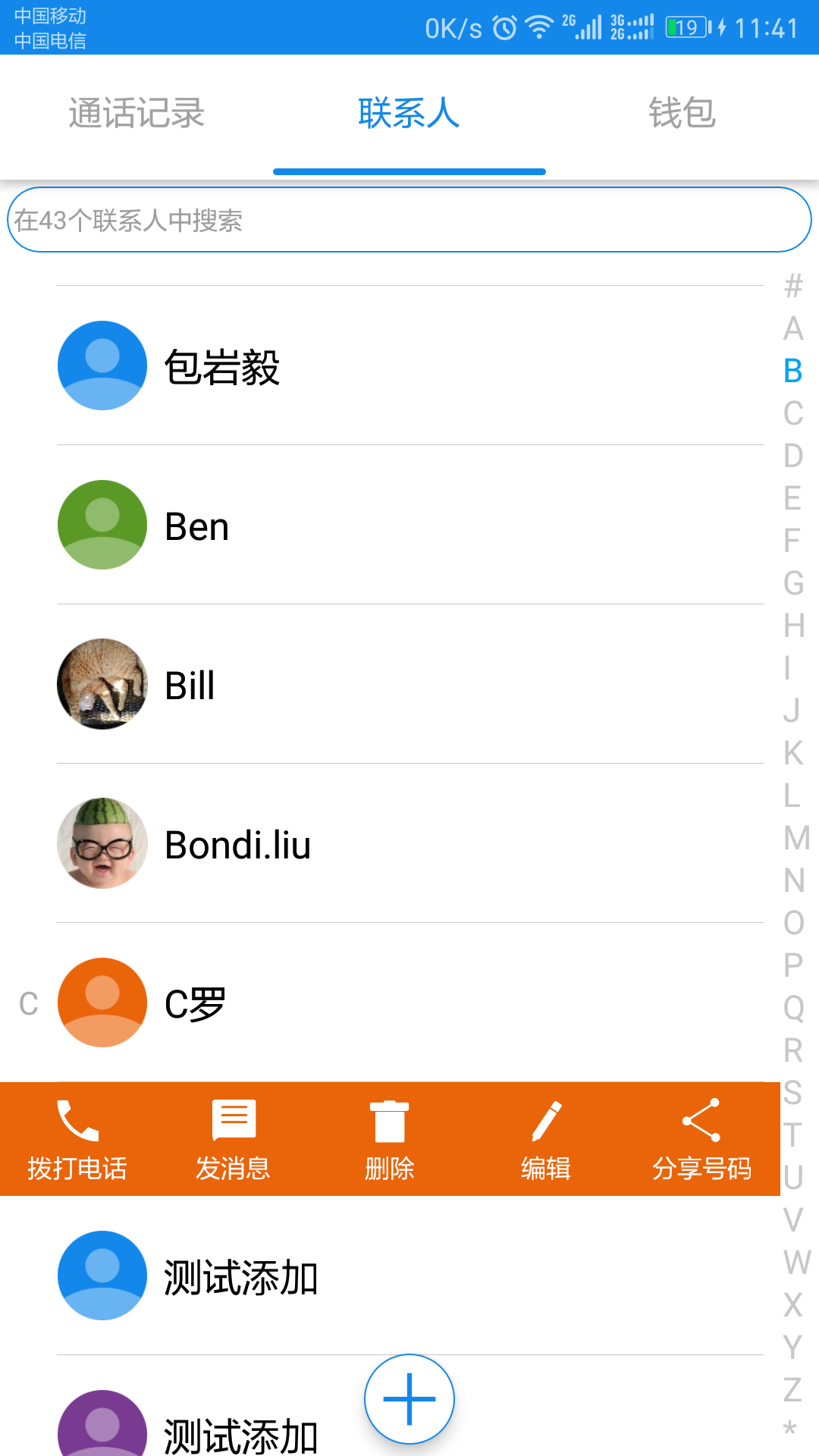
 

图1 联系人展示列表UI效果图 图2 联系人长按效果

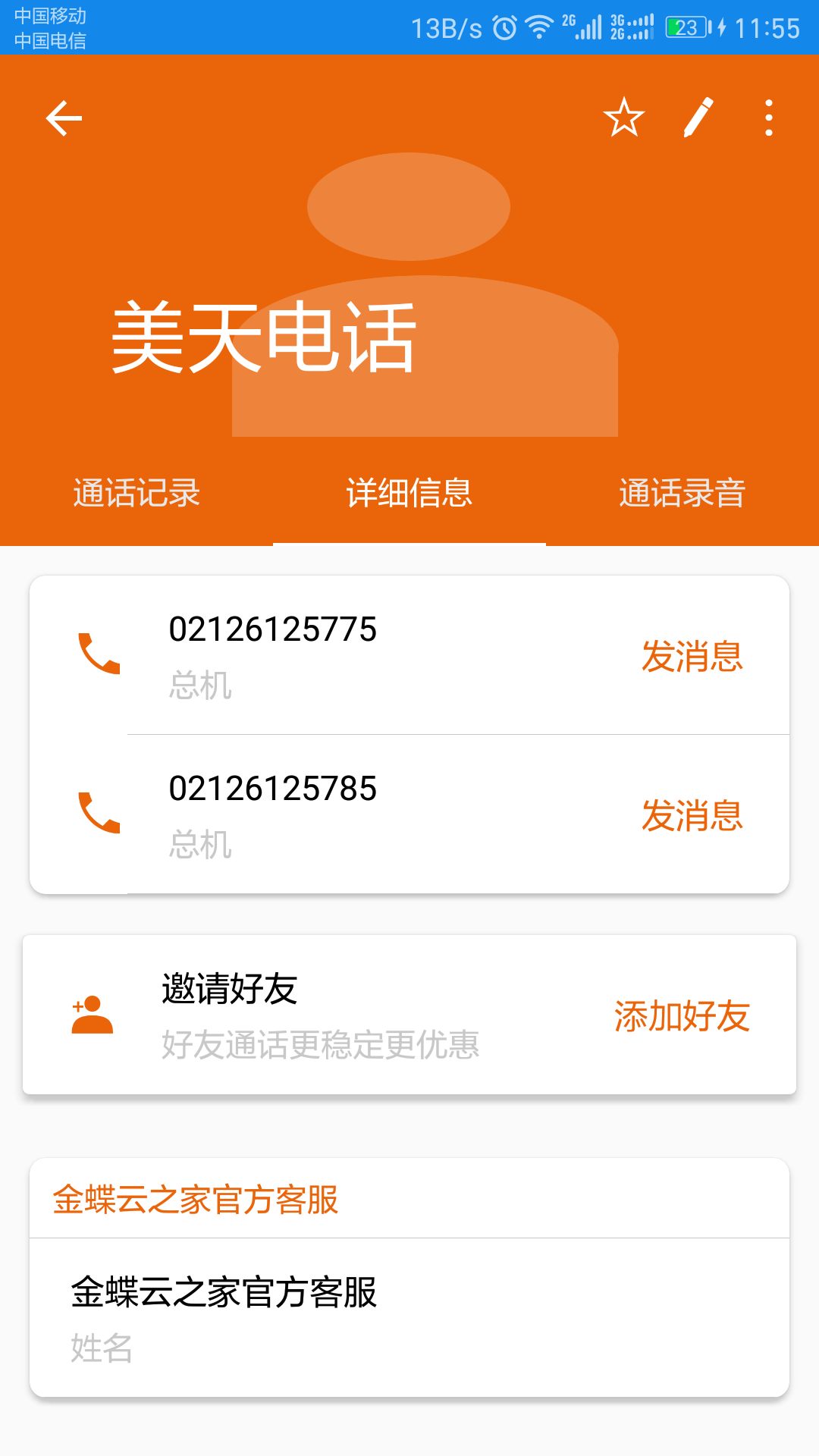
 

图3 联系人详情页面 图4 不同联系人详情界面色彩

**3.3 通讯录模块开发**

**3.3.1 通讯录数据文件contacts2.db**

Android手机自带的一个应用之一是通讯录，它是一个ContentProvider应用，其它应用可以对通讯录进行访问，进行对联系人的CRUD操作。为了APP联系人数据和手机的联系人数据库联系人同步，那么APP的数据需要和手机自带的联系人通讯录使用同一份数据，即需要公用contacts2.db文件，这样才能保证数据的统一性。

**3.3.2 通讯录数据库结构的介绍**

每个手机都会在手机的data/data/com.android.providers.contacts目录下存有一份contacts2.db文件。这是手机联系人数据库的文件，里面详细记录了联系人数据的所有信息，但是该文件在android手机未Root的情况下是不会被用户所看见的，这是一个系统文件，可以通过prividers进行存取，但是不能通过常用的文件管理器直接进入data/data目录下获取。APP要实现联系人的功能，那么首先得了解contacts2.db文件的存取以及联系人的数据库文件结构。要获取contacts2.db文件，我们可以在Eclipse的File Explorer视图下找到contacts2.db文件，这是通讯录的文件，当然，这个手机或者模拟器需要先root才能看到，不然这个文件对于用户来说是不可见的。如图5，然后，用SQLite打开，分析下它的数据库结构。

raw\_contacts表：该表保存了所有创建过的手机测联系人，每个联系人占一行，表里有一列标识该联系人是否被删除，该表保存了两个ID：RawContactID和ContactID,从而将contacts表和raw\_contacts表联系起来。该表保存了联系人的RawContactID、ContactID、联系次数、最后一次联系的时间、是否被添加到收藏夹、显示的名字、用于排序的汉语拼音等信息。

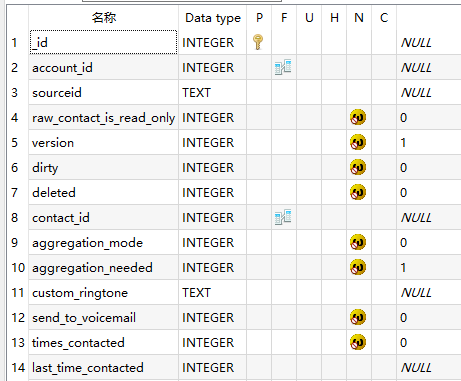


图6 raw\_contacts表

contacts表：该表保存了所有的手机测联系人，每个联系人占一行，该表保存了联系人的ContactID、联系次数、最后一次联系的时间、是否含有号码、是否被添加到收藏夹等信息，如图7所示。

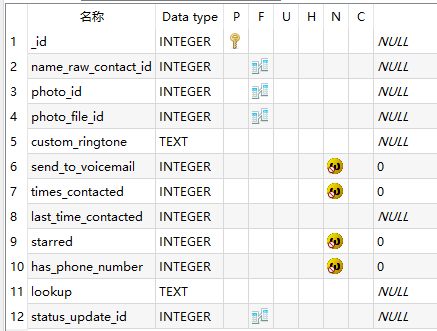


图7 contacts表结构

mimetypes表:该表定义了所有的MimeTypeID，即联系人的各个字段的唯一标志。

data表：raw\_contact\_id：通过raw\_contact\_id可以找到raw\_contact表中相对的数据。data1到data15这里保存着联系人的信息联系人名称联系人电话号码电子邮件备注等等。该表保存了所有创建过的手机测联系人的所有信息，每个字段占一行，该表保存了两个ID：MimeTypeID和RawContactID,从而将data表和raw\_contacts表联系起来。联系人的所有信息保存在列data1至data15中，各列中保存的内容根据MimeTypeID的不同而不同。如保存号码(MimeTypeID=5)的那行数据中，data1列保存号码，data2列保存号码类型(手机号码、家庭号码、工作号码等)。mimetypes表：该表定义了所有的MimeTypeID，即联系人的各个字段的唯一标志。

以上表示联系人模块主要的表，只要读取和操作这些表就可以实现联系人通讯录的功能。

**3.3.3 操作联系人数据库**

如果要想对手机联系人通讯录进行数据操作，那么首先要是加入这两个权限“android.permission.READ\_CONTACTS”，这是读取联系人数据的限权，然后是“android.permission.WRITE\_CONTACTS”，这是写入联系人数据的限权，只有这连个限权被用户授予之后APP才能正常的读取联系人数据。

（1）读取联系人  
 如果要读取联系人的数据，有如下的步骤。先读取contacts表，获取ContactsID，再在raw\_contacts表中根据ContactsID获取RawContactsID，然后就可以在data表中根据RawContactsID获取该联系人的各数据了，代码如下，该端代码运行过后，即可获取联系人的基本数据了。

**ContentResolver cr = getContentResolver();**

**Cursor cur = cr.query(ContactsContract.Contacts.CONTENT\_URI, null, null,**

**null, null);**

**if (cur.getCount() > 0)**

**{**

**while (cur.moveToNext())**

**{**

**String rawContactsId = "";**

**String id =**

**cur.getString(cur.getColumnIndex(ContactsContract.Contacts.\_ID));**

**str += "ID:" + id + "\n";**

**Cursor rawContactsIdCur = cr.query(RawContacts.CONTENT\_URI,**

**null,RawContacts.CONTACT\_ID +" = ?", new String[]{id}, null);**

**if (rawContactsIdCur.moveToFirst())**

**{**

**rawContactsId =**

**rawContactsIdCur.getString(rawContactsIdCur.getColumnIndex(**

**RawContacts.\_ID));**

**}**

**rawContactsIdCur.close();**

**}**

**}**

（2）新建联系人

新建联系人时，根据contacts、raw\_contacts两张表中ID的使用情况，自动生成ContactID和RawContactID。Android源码新建重复姓名的联系人的ContactID是不重复的，所以会重复显示。用下面的代码新建联系人，如果多次新建的联系人的姓名是一样的，生成的ContactID也会重复，RawContactID不会重复，我们在读取联系人的时候可以获取所有同姓名联系人的号码等信息，在显示联系人的时候，重复姓名的联系人的所有字段信息都会合并起来显示为一个联系人，代码如下。

**ContentValues values = new ContentValues();**

**Uri rawContactUri =**

**getContentResolver().insert(RawContacts.CONTENT\_URI, values);**

**long rawContactId = ContentUris.parseId(rawContactUri);**

**if (name != "")**

**{**

**values.clear();**

**values.put(Data.RAW\_CONTACT\_ID, rawContactId);**

**values.put(Data.MIMETYPE,**

**StructuredName.CONTENT\_ITEM\_TYPE);**

**values.put(StructuredName.GIVEN\_NAME, name);**

**getContentResolver().insert(ContactsContract.Data.CONTENT\_URI,**

**values);**

**}**

**if (phoneNum != "")**

**{**

**values.clear();**

**values.put(Data.RAW\_CONTACT\_ID, rawContactId);**

**values.put(Data.MIMETYPE, Phone.CONTENT\_ITEM\_TYPE);**

**values.put(Phone.NUMBER, phoneNum);**

**values.put(Phone.TYPE, Phone.TYPE\_MOBILE);**

**getContentResolver().insert(ContactsContract.Data.CONTENT\_URI,**

**values);**

**}**

**}**

（3）删除联系人

Android帮助文档中有说明，如果 raw\_contacts的RawContactID被删除，其他表的数据也会一一对应的删除，所以，要删除联系人，我们只需要将raw\_contacts表中指定RawContactID的行删除，其他表中与之关联的数据都会自动删除。

**public void delete(long rawContactId) {**

**getContentResolver().delete(ContentUris.withAppendedId(RawContac**

**ts.CONTENT\_URI, rawContactId), null, null);**

**}**

（4）更新联系人

联系人的所有信息都是保存在data表中，所以要更新联系人，我们只需要根据RawContactID和MIMETYPE修改data表中的内容。

**ContentValues values = new ContentValues();**

**values.put(Phone.NUMBER, “123");**

**values.put(Phone.TYPE, Phone.TYPE\_MOBILE);**

**String Where = ContactsContract.Data.RAW\_CONTACT\_ID + " = ? AND**

**" + ContactsContract.Data.MIMETYPE + " = ?";**

**String[] WhereParams = new String[]{"5", Phone.CONTENT\_ITEM\_TYPE};**

**getContentResolver().update(ContactsContract.Data.CONTENT\_URI,**

**values, Where, WhereParams);**

通过以上的代码后，APP中就可以实现联系人的增删改查了。满足了APP的功能需求。

**3.4 本章小结**

本章主要是对联系人模块展开。首先是介绍了联系人模块，然后是进行功能设计，然后是UI设计，最后进行功能实现。

# **第四章 钱包模块设计与开发**

**4.1 钱包模块介绍**

该APP中的钱包功能，主要是用来做用户注册、用户登录、生活服务功能和充值缴费的功能整合入口。在钱包模块，拥有丰富的最新新闻资讯，有海量的搞笑短视频资源，用户也可以打赏视频。

**4.2 钱包模块功能设计**

APP中的钱包功能，主要是集成各种生活功能接口，并且提供APP的用户登录，用户注册入口，在该模块还要显示用户的不同信息，例如用户是否登录、用户余额、充值入口等，点击用户头像则进入登录界面，并且在该模块中，集成了最新新闻资讯功能，新闻以大类进行分类，主要分为头条、科技、国内、国际、娱乐、体育、军事、社会、财经、时尚这十大类，用户可以选择相应的种类进行阅读新闻，为了使APP更加具有粘性，该模块也加入了搞笑短视频功能。因为搞笑视频更加具有用户粘性，所以把视频列表设置成默认显示。那么用户一进来就会看到视频，直接可以点击播放。如果用户喜欢该视频，那么可以点击该视频底部的打赏红包功能，也可以吧视频分享出去。

而用户登录注册入口也是在钱包功能中，点击顶部的头像按钮即可进入登录注册界面。

**4.3 钱包模块UI设计**

该模块是用户停留实际非常长的界面，所以该模块的UI设计也显得非常重要，整体界面以白色为主色调，蓝色为辅助色调，显示简洁干净，如图8所示，其它颜色作为点缀色，使APP看起来更加的多彩。布局则是顶部显示联系人头像加上联系人的名称。然后紧接着的是用户是否是VIP、当前余额、剩余时长、充值入口等，在列表中，只要随便选择一个新闻资讯类型，就可以刷新最新的新闻资讯，每个新闻的Item是呈单张图片和三张图片并列的两张方式来展示，这样是为了使新闻看起来丰富多样，不至于非常单调，如图9所示效果，这样看起来会感觉新闻内容非常丰富，用户会继续浏览下去。点击任何的新闻，就可以进入阅读界面进行阅读相关的新闻，如图10所示。点击充值按钮，即可进入充值余额界面，如图11所示，该界面可以显示用户名称、用户余额，并且可以选择相应的金额进行充值。如果用户未登录则会发出警告并且调整到登录界面，底部还可以选择支付宝登录和微信支付，点击充值按钮后即可打开支付宝或者微信进行相应的支付流程。

点击“未登录”按钮，即可进入登录界面。在登录界面，可以输入用户手机号码和密码进行登录，底部有登录按钮如图4.4.1，，左右滑动可以切换登录/注册界面，注册界面主要由验证码、手机号和密码组成，如图4.4.2。

图8 钱包模块整体UI 图9 新闻展示界面

图10 新闻阅读界面 图11 充值界面

图4.4.1 登录界面 图4.4.2注册界面

**4.4 钱包模块数据库设计**

钱包功能模块的数据库表主要用来存储用户充值的订单数据，需要记录用户充值的金额，商品名，充值时间等数据，存储钱包模块的数据，主要有三张表，分别为bmorder、bmpayorderinfo和mypayorder，其中bmorder表是用来存储每一笔订单的基本数据的，而bmpayorderinfo则用来记录每次支付结束后支付结果，记录订单支付记录以及是否支付成功等，mypayorder则是用来记录从客户端发起的支付订单请求，这是为了在服务端在回调失败的情况下载进行订单记录，以及订单核对，三张数据库表分别对应4-1、4-2、4-3。

而用户注册模块则由user、userinfo表和phoneinfo表组成，两张表字段对应表4-4和4-5，4-6，user表用来存储基本的用户数据，例如用户账号，用户密码和手机号等基本数据。而userinfo表用来存储用户的基本数据如电话号码，昵称等，phoneinfo表用来存储用户的账号数据，例如通话余额和通话剩余时间等。

表4‑1 bmorder表字段说明

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 |  |
| BmorderId | int | 订单ID | 是 |
| StradeStatus | int | 支付状态 |  |
| OutTradeNo | varchar | 订单唯一标识符 |  |
| TradeNo | varchar | 订单号 |  |
| CreateData | text | 创建时间 |  |

表4‑2 bmpayorderinfo表字段说明

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 主键 |
| BmPayInfoId | int | 订单ID | 是 |
| PayAccount | varchar | 支付账号 |  |
| Name | varchar | 订单名称 |  |
| Body | varchar | 订单号 |  |
| CreateTime | varchar | 创建时间 |  |
| OutTradeNo | varchar | 订单唯一标识符 |  |
| TransactionId | varchar | 订单ID |  |
| PayType | varchar | 支付类型 |  |
| TotalFee | float | 支付金额 |  |
| TradeState | varchar | 支付结果 |  |
| CreateData | varchar | 创建时间 |  |

表4‑3 mypayorder表字段说明

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 主键 |
| MyPayOrderId | int | 自定义订单ID | 是 |
| BmOrderId | varchar | 自动生成订单ID |  |
| PayState | Int | 支付状态 |  |
| PayAccount | varchar | 支付账号 |  |
| PayMoney | float | 支付金额 |  |
| TruePayMoney | float | 实际支付金额 |  |
| IsDealWith | int | 是否处理了 |  |
| CreateData | text | 创建时间 |  |

表4‑3 user表字段说明

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 主键 |
| UserId | int | 用户ID | 是 |
| AppId | text | APPID |  |
| AppToken | text | APP Token |  |
| ChannelId | text | 渠道号 |  |
| UserAccount | varchar | 用户账号 | 是 |
| UserPassword | text | 用户密码 |  |
| FYAccount | text | 轻码云账号 |  |
| FYPassword | text | 轻码云密码 |  |
| UserState | int | 用户状态 |  |
| CreatDate | text | 创建时间 |  |

表4‑3 userinfo表字段说明

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 主键 |
| InfoId | int | 数据ID | 是 |
| UserAccount | varchar | 用户账号 |  |
| UserName | text | 用户名 |  |
| UserNikeName | text | 用户昵称 |  |
| UserNumber | text | 用户号码 |  |
| IsVip | Int | 是否VIP |  |
| CreateDate | text | 创建时间 |  |

表4‑3 phoneinfo表字段说明

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 主键 |
| PhoneInfoId | int | 数据ID | 是 |
| UserAccount | text | 用户账号 |  |
| AppId | text | APP ID |  |
| ChannelId | text | 渠道号 |  |
| BalanceMoney | float | 剩余金额 |  |
| MaxPhoneTime | Int | 最大通话分钟数 |  |
| CreateDate | text | 创建时间 |  |

**4.5 钱包模块开发**

**4.5.1 支付模块开发**

支付的流程图如图4.5.1所示，首先用户如果通过客户端发起支付操作，那么会进入支付金额选择界面，选择好相应的金额后选择相应的支付方式，目前支持支付宝支付和微信支付两种支付方式，选择好了支付方式后点击支付按钮后则会生成一个支付订单，该订单会被上传到服务端，这样服务端可以记录相应的符合业务逻辑的订单数据，如果用户选择了确认支付后会进入支付宝或者微信进行支付，支付成功后，第三方平台会吧相应的支付结果回调到APP的服务端，在服务端中就可以通过已经上传的支付订单和第三方支付平台回调回来的订单数据进行核对和给用户做相应的增加余额或者通话分钟数的操作。

在服务端中，支付订单上传的接口如表4-5所示，每次订单生成后都会把该订单数据长传到服务端以便第三方支付结果回调回来以后进行数据核对后给用户加钱。第三方支付结果回调接口如表4-6所示。

表4‑5 支付订单上传接口

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名 | /MyPayOrderServlet | | 请求方式 | POST |
| 功能描述 | 订单数据上传接口 | | | |
| 参数 | 参数名 | 类型 | 说明 | |
|  | userAccount | String | 用户账号 | |
|  | payOrderId | String | 订单ID | |
|  | payMoney | String | 支付金额 | |
|  | truePayMoney | String | 实际支付金额 | |
| 返回结果 | 上传结果 | | | |

表4‑6 支付结果回调接口

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名 | /BmobPayOrderServlet | | 请求方式 | POST |
| 功能描述 | 支付结果回调接口 | | | |
| 参数 | 参数名 | 类型 | 说明 | |
|  | trade\_status | String | 支付状态 | |
|  | out\_trade\_no | String | Bmob返回的订单号 | |
|  | truePayMoney | String | 支付宝或微信返回的订单号 | |
| 返回结果 | 回调结果 | | | |

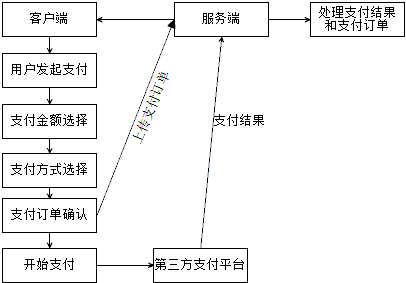


图4.5.1 支付流程

**4.5.2 用户模块开发**

用户模块主要是进行用户登录、注册的。在该APP中，用户的登录注册以手机号为注册名，这是为了在注册后拨打电话的功能正常，方便用户名和手机号进行统一，并且注册时需要输入验证码，验证码是通过第三方平台向我们发送的，该APP中接入了bmob平台。如果用户在首次进入APP时没有登录或者注册过，那么在进行关键操作时会进入注册和登录界面。以注册为例，首先用户输入手机号后，需要点击获取验证码按钮，那么APP会判断是否是合法的手机号，如果合法，那么会向服务端发起发送验证码的请求并且验证该手机号是否已经被注册，如果已注册则提示已存在用户，如果没有注册则会发送一条验证码到指定手机号，输入验证码后输入密码即可点击注册按钮，诸恶成功，伪代码如下。

**if(电话号码合法）{**

**向服务端请求号码是否可注册验证；**

**if(验证通过）{**

**通过APP向第三方平台发起发送验证码的请求；**

**接收验证码；**

**通过第三方平台的接口验证验证码是否正确；**

**if(正确）{**

**注册成功；**

**}**

**}**

**}**

**4.5.3 新闻资讯开发**

为了使APP更加具有用户粘性和实用性，在APP的钱包模块中集成了新闻资讯功能，这些功能对于大部分来说都非常便利地看到实时新闻，例如国内外新闻、军事、财经等等，并且加入了搞笑短视频，在实用性的同事也更加具有娱乐性。

新闻接口使用的是聚合数据的新闻接口。新闻数据并不经过该APP的服务端，而是直接由聚合平台返回给我们，APP中会对这些数据进行整理分类。在客户端，只要调用接口“http://v.juhe.cn/toutiao/index?type=top&key=APPKEY”即可获取json格式的新闻数据。

表4.7 新闻数据接口

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名 | http://v.juhe.cn/toutiao/index | | 请求方式 | GET |
| 功能描述 | 返回头条，社会，国内，娱乐，体育，军事，科技，财经，时尚等新闻信息 | | | |
| 参数 | 参数名 | 类型 | 说明 | |
|  | key | String | appkey | |
|  | type | String | 请求类型 | |
| 返回结果 | Json格式的新闻数据 | | | |

**4.6 本章小结**

本章主要围绕钱包功能展开，先是对钱包功能进行描述，说明了要达到什么效果以及开发什么功能，然后对钱包功能的UI进行设计，同时也设计了该模块对应的数据库，然后对系统的开发进行讲解。

# **电话模块设计与开发**

**5.1 电话模块介绍**

这是该APP的核心模块，在该模块中，主要的功能点是通过基于VOIP协议的通话功能的实现。VOIP的基本原理是通过语音的压缩算法对语音数据编码进行压缩处理，然后把这些语音数据按 TCP/IP 标准进行打包，经过 IP 网络把数据包送至接收地，再把这些语音数据包串起来，经过解压处理后，恢复成原来的语音信号，从而达到由互联网传送语音的目的。

如果网络电话要实现手机对手机之间的APP内通话功能并不困难，应为这完全就是IP技术可以实现的，但是，如果需要实现手机对手机端的电话号码直拨的功能，需要与运营商进行合作，虽然在VOIP协议中，大部分的数据包都是通过IP网络进行传输，但是任然需要传统运营商的专用网络对特定手机号码发起呼叫，所以该APP中的拨号功能是基于第三方SDK轻码云进行开发的。

**5.2 电话模块设计**

电话模块主要分为两个小模块，分别为通话记录管理模块和电话系统模块，通话记录模块主要是用来管理手机的通话记录，包括对通话记录进行删除、每次拨号后进行电话记录等，并且通话记录要记录下通话时间、通话时长。对每天记录做好分类标识。在通话记录模块整合了拨号盘功能，用户可以输入用户想呼叫的手机号码，点击呼叫按钮后可以选择普通通话和优惠电话，拨号过程中为了方便查看号码数据，还可以查看相应的手机号码归属地。用户点击任何一条记录即可查看详情。

**5.3 电话模块UI设计**

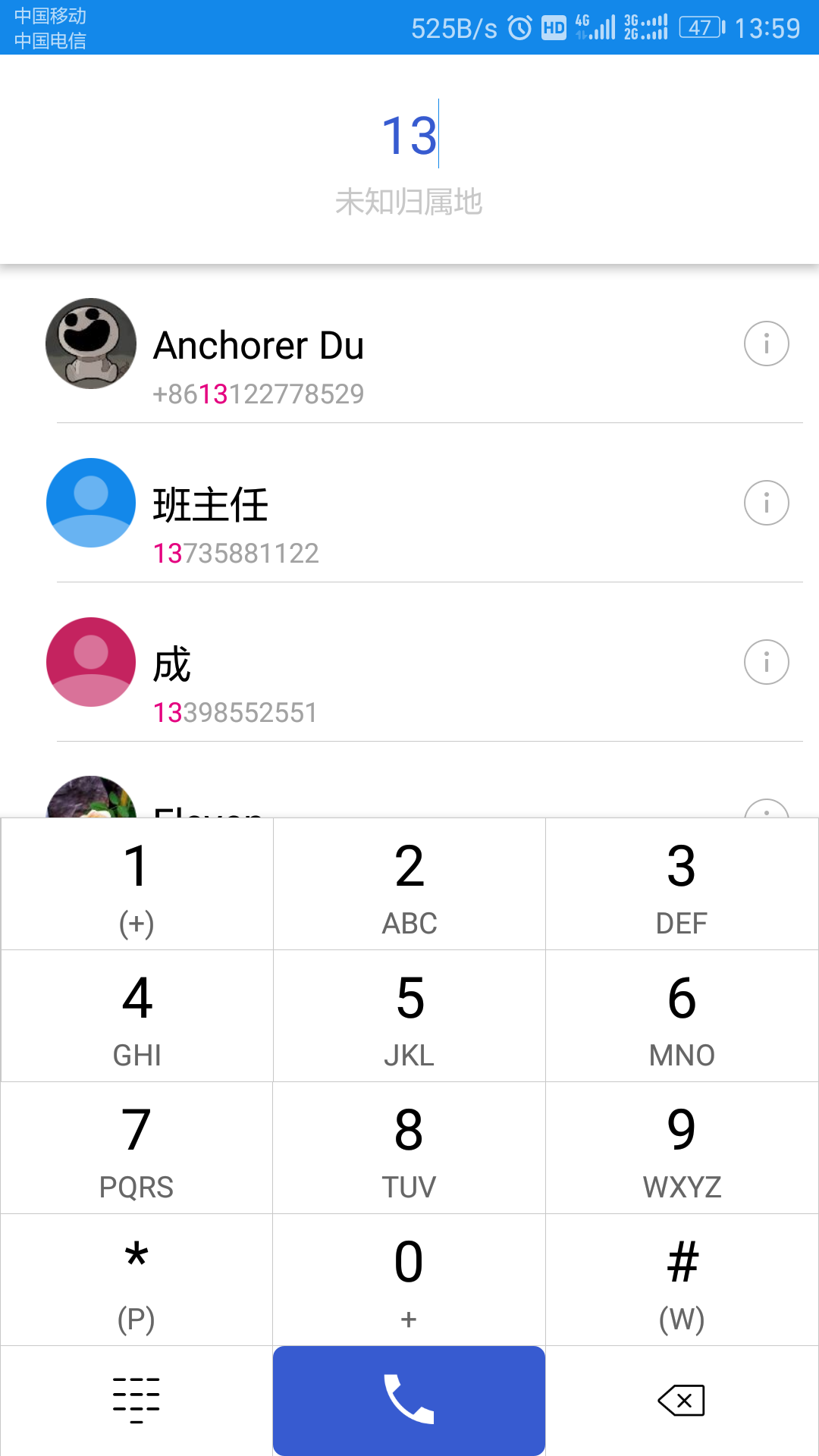
电话模块的UI界面，主要以百色为主色调。蓝色作为衬托，如如5.1所示，是通话记录 

图5.1 通话记录界面 图5.2 拨号盘拨号界面

界面，通话记录数据被分为三个子类，全部通话、未接电话、陌生电话，其中未接电话会以红色进行标记，让用户一眼就看出哪些电话没有接听到，在通话记录界面的底部，有个工具条，主要是对通话记录进行全选删除的功能。拨号盘如图5.2所示，输入号码后，在顶部显示出已输入的手机号码，如果手机号码字数多余7个，那么会显示出该号码的归属地。在拨号盘上方会根据已输入手机号码匹配手机已存在的联系人。

输入了11位数的手机号码以后可以点击拨号按钮，选择拨号方式，然后进入呼叫界面，如图5.3所示，顶部会显示对方的手机号码数据，使用了模糊效果的背景，底部拥有一些呼



图5.3 呼叫界面

叫时的常用功能。

**5.4 电话模块数据库设计**

电话模块的数据库设计工作主要是设计数据库表以便存储通话记录到服务端中，这些通话记录与手机通讯录上的通话记录有着本质的区别，手机通讯录上的通话记录主要是用来提供用户查看最近的通话记录数据，而在服务端存储的数据库中的通话记录，其主要的功能是用来记录每次通话的数据，以便在服务端对用户的通话分钟数和通话消费金额进行统计和扣除，电话模块的数据在服务端中主要存储在calllog、fyaskempowerlog和aswerfyempowerlog三张中，三张数据库表的字段分别如表5-4、5-5、5-6所示，calllog表主要是用来存储通话结束后详细的通话数据，包括通话开始时间，通话时长，通话结束时间，通话类型等详细数据，而fyaskepowerlog表是用来存储每次呼叫前的授权请求记录，aswerfyempowerlog表用来记录每次授权请求的授权结果。

表5‑4 calllog表字段说明

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 主键 |
| FYLogId | int | 通话记录ID | 是 |
| Action | text | 呼叫动作 |  |
| AppId | text | APP ID |  |
| AppCallId | text | 渠道号 |  |
| FyCallId | text | 通话ID |  |
| AppServerExtraData | text | 额外外部数据 |  |
| CallbackFirstStartTime | text | 回拨开始时间 |  |
| CallbackFirstEndTime | text | 回拨结束时间 |  |
| CallStartTime | text | 通话开始时间 |  |
| CallEndTime | text | 通话结束时间 |  |
| Duration |  | 通话时长 |  |
| StopReason | text | 结束通话原因 |  |
| TrueShowNumberType | int | 实际是否显示号码 |  |
| TrueIfRecord | text | 实际是否录音 |  |
| time | text | 通话时间 |  |
| Au | text | 记录唯一标识符 |  |
| CreateDate | text | 创建时间 |  |

表5‑4 fyaskempowerlog表字段说明

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 主键 |
| fyaskLogId | int | 通话记录ID | 是 |
| Action | text | 呼叫动作 |  |
| CallType | int | 呼叫类型 |  |
| ShowNumberType | int | 显号类型 |  |
| Caller | text | 呼叫方 |  |
| Callee | text | 被叫方 |  |
| AppExtraData | text | 外部数据 |  |
| FyCallId | text | 呼叫ID |  |
| FyAccountId | text | 飞语账号 |  |
| AppAccountId | text | App账号 |  |
| AppId | text | APP ID |  |
| ChannelId | text | 渠道号 |  |
| IfRecord | text | 是否录音 |  |
| Time | text | 时间 |  |
| Au | text | 记录唯一标识符 |  |
| CreateDate | text | 创建时间 |  |

表5‑4 aswerfyempowerlog表字段说明

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 主键 |
| AswerId | int | 授权ID | 是 |
| ResultCode | text | 授权结果 |  |
| ResultMsg | text | 授权描述 |  |
| AppCallId | text | 通话ID |  |
| MaxCallMinute | int | 最大通话分钟数 |  |
| AppServerExtraData | text | 额外外部数据 |  |
| ShowNumberType | int | 是否显号 |  |
| IfRecord | int | 是否录音 |  |
| CreatDate | text | 创建时间 |  |

**5.5 电话模块开发**

**5.5.1 通话记录模块开发**

获取手机通话记录流程主要包括以下步骤：

1. 获取ContentResolver

**ContentResolver resolver = getContentResolver()。**

1. resolver.query(\*)

需要传入通话记录的URI：CallLog.Calls.CONTENT\_URI。

（3）对查询得到的Cursor进行数据获取

**Uri uri = Uri.parse("content://call\_log/calls");**

**ContentResolver reslover = context.getContentResolver();**

**Cursor cursor = reslover.query(查询条件);**

**List<Data> calllogList = new List<Data>();**

**while(cursor.moveToNext()){**

**从cursor读取相应的通话记录数据；**

**CalllogList.add(读取出来的数据);**

**}**

**return calllogList**

**5.5.2 电话拨号模块开发**

1. 服务端接口开发

该模块使用了基于VOIP协议的飞语SDK，当客户端发起呼叫请求的时候，飞语云服务器会向应用服务器请求授权，以确认是否可以拨打，如表5-2所示，是飞语云授权接口，该接口可以在飞语官网平台进行配置，如图5.2所示，在APP的服务端开发好相应的服务端接口以后，把该接口配置到飞语官网中，那么当用户使用了APP进行拨号的时候，飞语服务端会把相关数据通过该接口回调到我们的服务端。代码如图5.3所示，当用户拨打电话接收以后，5.3所示的代码块会运行。

表4.7 新闻数据接口

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名 | http://192.168.1.104:8080/BeautPhoneServer/FYCallEmpower | | 请求方式 | POST |
| 功能描述 | 呼叫请求授权接口 | | | |
| 参数 | 参数名 | 类型 | 说明 | |
|  | action | String | 请求类型 | |
|  | callType | int | 呼叫类型 | |
|  | showNumberType | int | 外呼显号标示 | |
|  | caller | String | 呼叫方 | |
|  | callee | String | 被叫方 | |
|  | appExtraData | String | 第三方私有数据 | |
|  | fyCallIdString | String | 飞语产生的呼叫的唯一标识 | |
|  | fyAccountId | String | 提交呼叫请求的飞语账户号码 | |
|  | appAccountId | String | 在用户服务器端用户的唯一名称 | |
|  | appId | String | 应用ID | |
|  | channelId | String | 渠道ID | |
|  | ifRecord | int | 是否需要录音 | |
|  | ti | long | 自1970年1月1日0时起的毫秒数 | |
|  | au | String | MD5（appId+ appToken+ti） | |
| 返回结果 | Json格式的新闻数据 | | | |

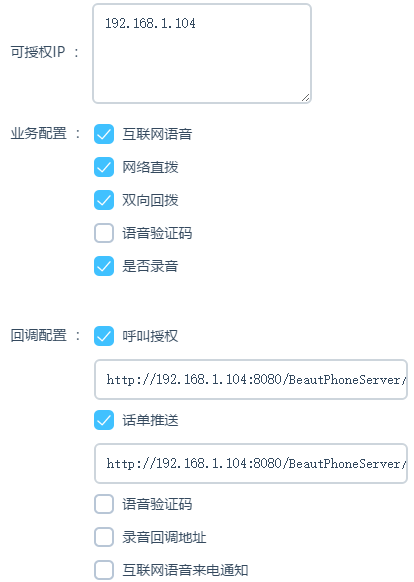


图5.2 飞语拨号回调接口配置界面

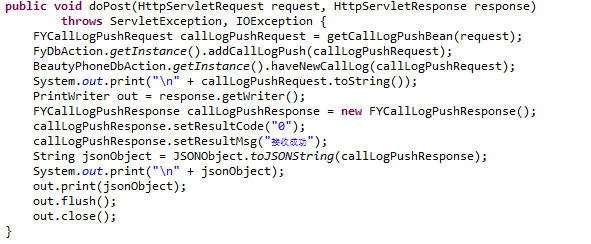


图5.3 通话记录接收接口。

1. 客户端开发

客户端主要是接入SDK后，在相应的地方加入呼叫请求功能代码，接入飞语SDK流程如下所述。

（a）导入SDK

将SDK中libs文件夹下的fycloud\_release.jar文件复制到工程目录libs下，将armeabi文件夹下的libfycloud.so文件复制到工程目录armeabi下，如图5.5所示。

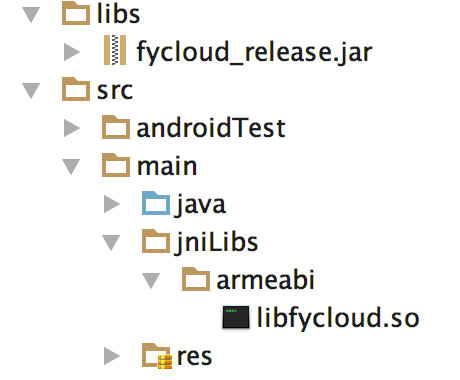


图5.5 飞语sdk的jar包项目结构

(b)配置AndroidManifest.xml

把以下权限添加到maiifest文件中，为APP做好权限请求。

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

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

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

**<uses-permission android:name="android.permission.RECORD\_AUDIO" />**

**<uses-permission android:name="android.permission.WAKE\_LOCK" />**

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

**<uses-permission android:name="android.permission.MODIFY\_AUDIO\_SETTINGS"/>**

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

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

（c）初始化SDK

在连接云平台之前需要初始化SDK，并且添加必要的接口，用以监听连接云平台状态和通话状态。

**FYClient.instance().init(Context context, String channelId, boolean toggleLog, boolean needEC);**

**FYClient.addListener(FYClientListener listener);**

**FYCall.addListener(FYCallListener listener);**

(d)通话

连初始化成功后，可以通过FYCall中的以下方法发起互联网语音，网络直拨，双向回拨。

**/\*\***

**\* 互联网语音**

**\*/**

**FYCall.networkCall(String fyAccountId, int showNumberType, boolean isRecord, String extraData);**

**/\*\***

**\* 网络直拨**

**\*/**

**FYCall.directCall(String number, int showNumberType, boolean isRecord, String extraData);**

**/\*\***

**\* 双向回拨**

**\*/**

**FYCall.callback(String number, int showNumberType, boolean isRecord, String extraData);**

（e）通话回调接口

**// 新来电**

**public void onIncomingCall(String calleeNumber);**

**// 发起呼叫**

**public void onOutgoingCall(String calleeNumber);**

**// 对方正在振铃**

**public void onCallAlerting(String calleeNumber);**

**// 正在通话**

**public void onCallRunning(String peerNumber);**

**// 通话结束**

**public void onCallEnd();**

**// 呼叫失败**

**public void onCallFailed(FYError error);**

**// 回拨成功**

**public void onCallbackSuccessful();**

**// 回拨失败**

**public void onCallbackFailed(FYError error);**

**// 收到dtmf**

**public void onDtmfReceived(char dtmf);**

当APP客户端通过以上步骤的代码加入拨号的功能之后，用户每次拨打电话结束后飞语服务器都会向APP服务端发送通话记录数据，服务端则可以在接收到这些数据后对用户的通话资金和余额等进行扣除操作。

**5.6 本章小结**

本章主要是对APP的电话模块的主要技术点和功能模块进行了详细的描述，首先介绍了电话模块，然后进行电话模块的功能设计，紧接着是UI设计，最后是主要功能的点的实现。

# **第六章 总结**

## **6.1** **完成的工作**

目前该APP已经实现了主体功能，首先是拨打电话，用户可以输入标准的11位国内手机号码，然后通过网络进行拨号，即使对方并没有注册和安装该APP，被叫也可以正常响铃并接通电话，并且通话资费可控，如果APP不向用户收取费用，那么对于用户来说，通话功能可以完全免费，当然，目前接入的是第三方SDK，第三方SDK并不是完全免费，对于APP开发商或者开发者来说，如果APP对于用户完全免费，除非开发商或者开发者找到另外的填补这部分支出的盈利方式。然后该APP也实现了联系人的管理整合，对联系人进行合理地整理，使用户可以高效地查找联系人，快速拨号等。最后该APP为了使功能更加丰富，还加入了娱乐性质的功能，例如新闻资讯，搞笑关视频等，增加APP的可玩性。

## **6.2 存在的问题及下一步工作**

**6.2.1 存在的问题**

目前虽然该APP实现了主体功能，但是在很多细节上并不是一时半会能解决好的，首先是UI界面，虽然经过了设计，但是界面并没有达到理想的效果。其次加如新功能将会越来越麻烦，因为很多功能点是随着APP功能点的增加，整个APP的架构的设计模式缺陷就显现出来，功能添之间存在交叉部分，不能很好的进行解耦，导致在许多地方牵一发而动全身。APP中的许多功能点没有达到商业化的标准，并且在编写代码时许多地方没有做好变量判空，导致许多地方仍然会出现BUG而Crash。

**6.2.2 下一步工作**

虽然毕设到此告一个段落，但是该APP的开发与设计并不会因此而停止，接下来将会重新构架整个APP的代码架构，使代码的各个模块耦合性更加低，然后UI界面重新调整设计，使UI界面更加美观，大部分的功能点重新添加，代码重新编写。在之后的工作中，将把更多的生活服务类的功能加入进来，例如查询快递、违章查询、身份证查询，GIF搞笑动图等等功能，让APP的功能更加全面更加人性化。