# 摘要

即时通信（Instant Messaging，IM）是随着互联网的出现而兴起的新型通信手段，发展非常迅速。为了方便用户利用手机或移动设备实现移动即时通信，在本课题中进行基于iPhoneOS平台相应的设计开发。

在本课题实现的即时通信系统中，新用户使用前需进行注册，注册信息通过服务器端写入数据库。登录之后，用户可添加好友，与好友进行聊天，实现消息的发送与接收。同时，用户还可以添加、删除好友。用户可以通过用户配置，修改个人信息及密码，也可以进行注销操作，注销之后，服务器将从数据库中删除该用户信息。

该项目的客户端可分为四个模块，用户注册登录模块、好友管理模块、消息收发模块、用户配置模块。客户端与服务器之间通过HTTP协议进行通信。服务器端采用MVC三层架构，通过Servlet响应客户端请求，对数据库中数据进行写入、修改、查找、删除等相关操作。

本系统功能现已基本实现，每部分模块也进行了局部的分析及整体的测试，基本满足了网络连接、好友管理、消息收发、用户配置等系统需求。

关键词:即时通信；iOS；HTTP协议

# **Abstract**

With the spread of the Internet, Instant Messaging (IM),a new means of communication,is developing rapidly. In order to meet users’ need to communicate with Instant Messaging on phones or mobile devices, this thesis ,which is based on iPhoneOS platform,attempts to make relevant designs and development.

In the Instant Messaging of this research, new users need to register at first, and their personal information will be recorded in the server side. After logging in the Instant Messaging, new users are able to add new friends, chat with them, send and receive messages. Meanwhile, they are also able to add and delete friends. By using user configuration, users can change their personal information and passwords as well as logout their accounts. The server will delete users’ information in the database after they cancel their accounts.

The client side of the project can be divided into four modules, user registration login module, friends management module, messaging module, and user configuration module. The client side and server communicate through HTTP protocol. The server side adopts the MVC pattern, and responds the requests of the client side through Servlet, writing, revising, searching and deleting the data in the database.

The functions of this system can be basically achieved and every part of it has been partly analyzed and completely tested, which will meet the needs of network connections, friends’ management, messaging, and user configuration of the system.

**Keywords:**Instant Messaging，iPhoneOS platform，HTTP protocol

# 目录

[摘要 I](#_Toc357414564)

[Abstract II](#_Toc357414565)

[目录 III](#_Toc357414566)

[第一章 绪论 1](#_Toc357414567)

[第一节 相关技术的发展现状 1](#_Toc357414568)

[一、移动设备系统平台的发展现况 1](#_Toc357414569)

[二、即时通信及相关软件的发展及介绍 2](#_Toc357414570)

[三、通信协议的发展现状 3](#_Toc357414571)

[第二节 课题选定内容与开发环境 3](#_Toc357414572)

[一、关于课题 3](#_Toc357414573)

[二、本文的内容和组织结构 4](#_Toc357414574)

[三、开发环境简介 4](#_Toc357414575)

第二章 [基于iPhoneOS的即时通信系统的需求分析 7](#_Toc357414576)

[第一节 任务概述 7](#_Toc357414577)

[一、目标 7](#_Toc357414578)

[二、用户特点 7](#_Toc357414579)

[第二节 需求规定 7](#_Toc357414580)

[一、性能规定 7](#_Toc357414581)

[二、功能用例分析 8](#_Toc357414582)

[第三章 基于iPhoneOS的即时通信系统的整体结构 10](#_Toc357414583)

第一节 [系统整体结构分析 10](#_Toc357414584)

[一、系统整体结构示意图 10](#_Toc357414585)

[二、MVC三层架构设计 11](#_Toc357414586)

[第二节 各主要功能处理流程 12](#_Toc357414587)

[一、处理流程示意图 12](#_Toc357414588)

[二、处理流程说明表 13](#_Toc357414589)

[第三节 相关技术及难点分析 15](#_Toc357414590)

[一、名词解释 15](#_Toc357414591)

[二、技术难点 16](#_Toc357414592)

[第四章 基于iPhoneOS的即时通信系统的概要设计 27](#_Toc357414593)

[第一节 系统数据结构设计 27](#_Toc357414594)

[一、数据库表关系设计 27](#_Toc357414595)

[二、数据表设计 28](#_Toc357414596)

[第二节 服务器端设计 29](#_Toc357414597)

[一、实体类与会话Bean 29](#_Toc357414598)

[二、Servlet对应响应操作 30](#_Toc357414599)

[第三节 客户端设计 32](#_Toc357414600)

[一、外部接口 32](#_Toc357414601)

[二、内部接口—界面跳转关系 33](#_Toc357414602)

[第四节 功能模块的传输设计 34](#_Toc357414603)

[一、用户注册 34](#_Toc357414604)

[二、用户登录 35](#_Toc357414605)

[三、添加好友 35](#_Toc357414606)

[四、删除好友 35](#_Toc357414607)

[五、消息收发 35](#_Toc357414608)

[六、其他功能 36](#_Toc357414609)

[第五章 基于iPhoneOS的即时通信系统的详细设计与实现 37](#_Toc357414610)

[第一节 服务器端的详细设计与实现 37](#_Toc357414611)

[一、数据库的实现 37](#_Toc357414612)

[二、Servlet调用数据库方式 39](#_Toc357414613)

[三、流程逻辑 39](#_Toc357414614)

[第二节 客户端的设计与实现 47](#_Toc357414615)

[一、UI界面设计实现 47](#_Toc357414616)

[二、各界面功能实现 51](#_Toc357414617)

[第六章 基于iPhoneOS的即时通信系统的部署及测试 54](#_Toc357414618)

[第一节 系统部署 54](#_Toc357414619)

[第二节 系统测试 54](#_Toc357414620)

[一、测试概要 54](#_Toc357414621)

[二、测试结果 55](#_Toc357414622)

[结束语 70](#_Toc357414623)

[不足 70](#_Toc357414624)

[展望 70](#_Toc357414625)

[致谢 71](#_Toc357414626)

[参考文献 72](#_Toc357414627)

# 绪论

本章主要对移动设备系统平台和移动即时通信等相关技术的发展现状进行了比较与分析，并最终选择采用Objective-C语言编写基于iPhoneOS客户端代码，用Java语言在Netbeans平台上架构服务器端，通过在服务器端编写Servlet应用程序，从而完成客户端与服务器端的HTTP通信，最终实现基于iPhoneOS的即时通信系统。

## 第一节 相关技术的发展现状

本课题主要实现基于移动设备系统平台的即时通信系统，下面针对课题相关技术发展状况作简单介绍。

### 一、移动设备系统平台的发展现况

当前移动设备的市场上主要有Android、iPhone、WindowsMobile和Symbian等开放手机平台，用户将获得更多的移动应用，而移动终端业务开发也会蓬勃兴起。关于各种平台的比较如下：

#### （一）Symbian系统平台

Symbian是一个实时性、多任务的纯32位操作系统，具有功耗低、内存占用少等特点，在有限的内存情况下，非常适合手机等移动设备使用，经过不断完善，可以支持GPRS、蓝牙、SyncML、以及3G技术。最重要的是它是一个标准化的开放式平台，任何人都可以为支持Symbian的设备开发软件。

由于对新兴的社交网络和web 2.0内容支持欠佳，塞班占智能手机的市场份额日益萎缩。2010年末，其市场占有量已被新系统超过。2013年1月24日晚间，诺基亚宣布，今后将不再发布塞班系统的手机，意味着塞班这个智能手机操作系统，在长达14年的历史之后，终于迎来了谢幕，但其仍拥有一定数量的用户。

#### （二）Windows Mobile系统平台

Windows Mobile系列操作系统是从微软计算机的Windows操作系统上变化而来的。Windows Mobile系列操作系统功能强大，而且支持该操作系统的智能手机大多数都采用了英特尔嵌入式处理器，主频比较高，在其它硬件配置（如内存、储存卡容量等）上也比采用其它操作系统的智能手机要高出许多，因此性能比较强劲，速度比较快。Windows Mobile系列操作系统包括SmartPhone以及Pocket PC Phone两种平台。Pocket PC Phone主要用于掌上电脑型的智能手机，而SmartPhone则主要为移动智能手机提供操作系统。

#### （三）Android系统平台

Android是一种基于Linux的自由及开放源代码的操作系统，主要使用于移动设备，如智能手机和平板电脑，由Google公司和开放手机联盟领导及开发。Android操作系统最初由Andy Rubin开发，主要支持手机。2011年第一季度，Android在全球的市场份额首次超过塞班系统，跃居全球第一。 2012年11月数据显示，Android占据全球智能手机操作系统市场76%的份额，中国市场占有率为90%。具有开放、不受运营商约束、硬件丰富、方便开发等特点。

#### （四）iPhone系统平台

iPhone由苹果公司首席执行官史蒂夫·乔布斯在2007年1月9日举行的Macworld宣布推出，2007年6月29日在美国上市，将创新的移动电话、可触摸宽屏iPod以及具有桌面级电子邮件、网页浏览、搜索和地图功能的突破性因特网通信设备这三种产品完美地融为一体。iPhone引入了基于大型多触点显示屏和领先性新软件的全新用户界面，让用户用手指即可控制iPhone。iPhone还开创了移动设备软件尖端功能的新纪元，重新定义了移动电话的功能。

因为iPhoneOS平台相对Symbian系统更加智能、先进，相对Windows Mobile系统具有更多的客户群体，相对于Android系统平台更具有封闭性、安全性，所以本系统客户端将基于iPhoneOS进行开发。

### 二、即时通信及相关软件的发展及介绍

即时通信（Instant messaging，简称IM）是一个终端服务，允许两人或多人使用网络即时的传递文字讯息、档案、语音与视频交流。即时通信不同于Email在于其交谈是即时的。大部分的即时通信服务提供了Presence Awareness的特性──显示联络人名单，联络人是否在线上与能否与联络人交谈。

在早期的即时通信程式中，使用者输入的每一个字元都会即时显示在双方的屏幕上，且每一个字元的删除与修改也会即时更新。这种模式比起使用Email更像是电话交谈。在现在的即时通信程式中，交谈中的一方通常只会在另一方本地端按下送出键（Enter或是Ctrl+Enter）后才会看到对方的讯息。

近年来，许多即时通信服务开始提供视讯会议的功能，网络电话(VoIP）与网路会议服务开始整合为兼有影像会议与即时讯息的功能。于是，这些媒体的区别变得越来越模糊。

据腾讯QQ官方消息，2005年，QQ在国内已拥有2.4亿多的注册用户，7000多万活跃用户，同时在线人数700万以上。目前，腾讯QQ已成为国内用户最多的个人即时通信工具，占据着国内约65％以上的市场份额，而且通过整合移动通信手段，可将客户端信息发送给手机用户。

邦邦通是一款Java软件，也是手机上网聊天中起步最早的软件之一。支持移动GPRS的cmwap以及cmnet两种接入点。它拥有自己独立的即时通信体系，每位用户都需要注册一个邦邦号才能使用聊天功能。在邦邦通本身的即时通信体系中，可以像QQ一样，通过邦邦号查找添加好友，也可以通过在线用户列表结识新的朋友。不过邦邦通的查找功能没有按照地域进行查找的方式，对于想结识同城好友的用户来说就有些困难。当然邦邦通最大的魅力还是在于可以捆绑 QQ以及MSN的聊天服务。通过绑定帐号，在登录邦邦通后，用户还可以同时登录QQ以及MSN两大即时通信服务，任何时间任何地点都可以与PC上的好友聊天。不过由于腾讯近日对第三方QQ软件的封锁，目前邦邦通登录QQ的功能已经无法使用。此外邦邦通还具有图铃点播以及聊天室等附加功能。

随e聊是手机即时通信软件中起步相对较晚的一个，虽然发展至今的时间并不长，但它总结了邦邦通、蜜友以及PICA等前辈们的优点以及不足，所以在功能方面，“随e聊”起步的台阶就比较高，它除了支持常规的QQ、MSN聊天以外还能够登录Yahoo Messenger。早在MSN Messenger流行之前，中国也有为数不少的Yahoo Messenger的用户，而“随e聊”是首个能够通过普通Java手机登录Yahoo Messenger聊天服务的软件，雅虎通的用户终于也可以享受随心所欲的移动聊天乐趣了。此外“随e聊”也是目前主流手机聊天软件中唯一一个能够支持 QQ以及MSN好友自定义分组功能的软件。

本文将通过对市场上各种流行的手机即时通信软件进行比较与分析，以iPhoneQQ为主要模型，采用HTTP协议，进行原型设计，完成基于iPhoneOS的即时通信系统。本课题中实现的基于iPhoneOS的即时通信系统在服务器端采用Java语言编写，可以实现跨平台通信。

### 三、通信协议的发展现状

XMPP协议属于当前主流IM使用协议的一种，下面将针对XMPP协议与HTTP协议进行分析与比较。

XMPP(ExtensibleMessageing and Presence Protocol：可扩展消息与存在协议)。XMPP是一种基于XML的协议，继承了XML环境中灵活的扩展性。经过扩展以后的XMPP允许通过发送扩展的信息来处理用户的需求，以及在XMPP的顶端建立如内容发布系统和基于地址的服务等应用程序，而且XMPP还包含了针对服务器的软件协议，实现了在不同服务器之间的通话。

HTTP超文本传输协议 (HTTP-Hypertext transfer protocol) 是分布式、协作式、超媒体系统应用之间的通信协议，是万维网（world wide web）交换信息的基础。HTTP工作在TCP/IP协议体系中的TCP协议上。HTTP协议具有支持客户/服务器模式，简单快速，灵活，无连接，无状态的主要特点。请求方法常用的有GET、HEAD、POST。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。

HTTP协议相对于XMPP协议而言，避免了使用XML文件进行传输，使用Post请求方式进行通信，使得信息传递具有安全性，采用数据库进行存储，可永久存储，并具有组织性、条理性、可共享性。另外，客户端与服务器间采用HTTP协议进行通信，对于客户端平台无强制限定，可实现跨平台通信。

## 第二节 课题选定内容与开发环境

### 一、关于课题

基于iPhoneOS的即时通信系统，是用于iPhone手机平台上的一款可供用户间进行收发消息的移动通信软件。

本课题主要目标是基于iPhoneOS平台，利用Xcode工具软件，使用Objective-C编程语言，通过分析了解即时通信系统的基本框架和流程，实现用户注册、登录，用户及系统配置，好友管理，消息的收发等功能。服务器端，运用Netbeans搭建服务器，与客户端通过HTTP协议进行通信，并利用Servlet与数据库实体会话Bean进行连接操作，实现服务器端对数据库的访问及修改。

本课题主要研究内容包括：首先，学习了解Objective-C基础语法，了解其内部的消息机制和代理机制等，掌握iPhoneOS平台下控件编程和使用方法；其次，了解并分析移动即时通信系统基本框架及流程；再次，构建服务器，创建数据库，实现服务器对数据库的访问操作；最后，实现手机客户端与服务器端的HTTP通信，实现消息的发送及接收，完善软件各个功能及手机客户端的界面设计。

### 二、本文的内容和组织结构

本文主要设计和实现了基于iPhoneOS的即时通信系统，对设计过程中涉及的各个关键技术进行了深入的分析和讨论，主要包括iPhoneOS平台下控件编程和使用方法，客户端与服务器之间网络连接，Servlet搭建等等，并重点讨论了服务器端的设计与实现，客户端各界面间交互及客户端与服务器之间的网络连接。最终实现完整的移动即时通信系统。

文共分为六章，各章节的组织结构和内容如下：

第一章，首先分析了系统的研究背景，介绍了当前手机操作系统、iOS系统、即时通信及相关软件软件的发展现状，以及本文的主要内容，并对系统的开发环境进行了简要的介绍。

第二章，主要对基于iPhoneOS的即时通信系统的任务、目标进行了分析，并讨论了系统的功能与需求，采用软件工程的管理思想，对开发模型进行了选择。

第三章，提出了在iPhone平台上开发基于HTTP协议的即时通信系统，讨论了系统的总体结构设计，概述系统整体功能和处理流程，并对相关应用技术及难点进行了总结。

第四章，主要对系统的数据结构、服务器端、客户端和传输模块的解决方案进行概要的设计与分析。

第五章，对系统服务器端的运行流程进行了详细的分析，描述了系统的即时通信流程，最后对iPhone客户端从用户界面及相关功能进行了详细的设计与实现。

第六章，对系统进行部署，从系统注册登录、获取好友列表、添加和删除好友、消息的发送和接收、用户配置等几个方面进行了测试。

最后，总结了本文的主要研究内容和研究成果，分析了存在的问题，并对今后的发展方向进行了展望。

### 三、开发环境简介

本课题客户端开发环境为Xcode4.3，Xcode由苹果公司专门向开发人员提供的集成开发环境，专门用于开发Mac OS X和iOS的应用程序，使用Objective-C编程语言。

服务器及数据库采用Netbeans7.1.2版本搭建，使用Java语言。

#### 客户端

由于本系统客户端在iOS系统中进行开发，下面将针对iOS系统环境、开发工具Xcode和开发语言Objective-C进行简单介绍。

##### 1. iOS系统的历史和发展

iOS（原名iPhoneOS)是由苹果公司为iPhone开发的操作系统。它是给iPhone,iPod touch和iPad使用。就像其基于的Mac OS X操作系统一样，它也是以Darwin为基础的。

iOS的系统架构分为四个层次：核心操作系统层（the Core OS layer），核心服务层（the Core Services layer），媒体层（the Media layer），可轻触层（the Cocoa Touch layer）。操作系统占用大概240MB的内存空间。iOS的用户界面的概念基础上是能够使用多点触控直接操作。控制方法包括滑动,轻触开关及按键。与系统互动包括滑动 (swiping),轻按（tapping）,挤压(pinching)及旋转(reverse pinching)。

2012年6月12日，iOS 6发布，并于2012年9月19日开放下载。

##### 2. 关于Xcode

The Xcode suite 包含有GNU Compiler Collection自由软件，并支援 C语言、C++、Fortran、Objective-C、Objective-C++、Java、AppleScript、Python以及Ruby，还提供Cocoa、Carbon以及Java等编程模式。Xcode套件使用 GDB作为其后台调试工具。

从Xcode 3.1开始，Xcode也可被用为iPhone OS的开发环境。Xcode最新主要版本是Xcode 4.6.2。本课题中使用目前最新版本的Xcode进行系统的客户端开发，开发环境如图1。

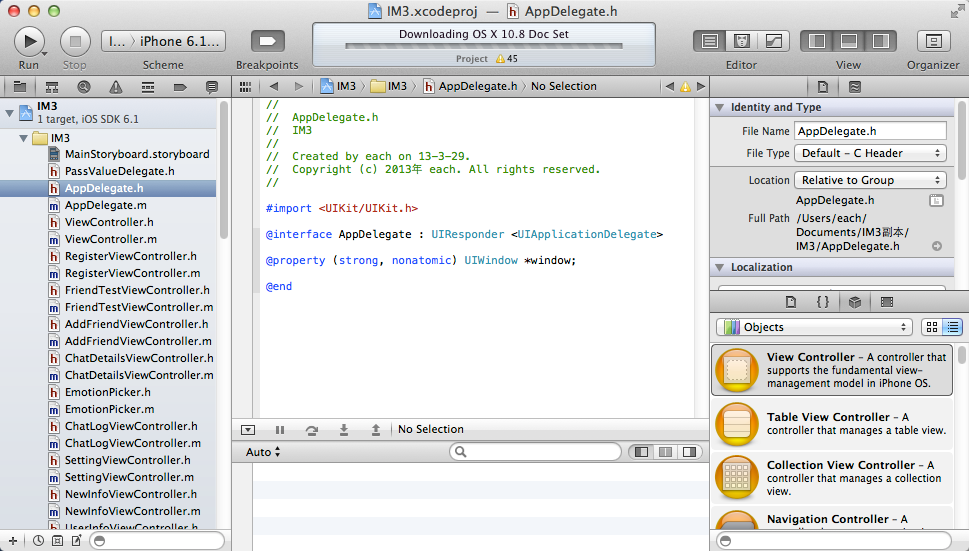


图 1 Xcode开发环境

##### 3. 关于Objective-C

Objective-C，是扩充C的面向对象编程语言。它主要使用于Mac OS X和GNUstep这两个使用OpenStep标准的系统。Objective-C可以在gcc运作的系统写和编译，因为gcc含Objective-C的编译器[1]。

Objective-C可用于编写iOS操作系统(如：iphone、ipod touch、ipad 等苹果移动终端设备)。它是用一个用C编写的运行库，只会令应用程序的大小增加很少，其生成的程序通常不会比其原始编码大很多[2]。

和C++不同，Objective-C不支持运算子重载(它不支持ad-hoc多型)。且Objective-C只容许对象继承一个类别(不设多重继承)，这点与C++不同，但和Java相同。

#### （二）服务器端

服务器端将采用Java语言，在Netbeans中进行Servlet应用程序编写，最终实现服务器端。下面对Netbeans平台和Java Servlet进行简要介绍。

##### 1.关于Netbeans

NetBeans IDE是一个可以方便的在Windows，Mac，Linux和Solaris中运行的集成开发环境。NetBeans包括开源的开发环境和应用平台，可提供丰富的产品文档和培训资源以及大量的第三方插件。NetBeans是开源软件开发集成环境，可以用于Java、C/C++，PHP等语言的开发，本身是一个开发平台，可以通过扩展插件来扩展功能。NetBeans IDE的7.0引入了对Java SE 7开发语言的支持，该版本还提供了OracleWebLogic服务器中Oracle数据库和GlassFish3.1的进一步整合与支持。

本课题中实现的基于iPhoneOS的即时通信系统，选用Netbeans 7.1.1在Mac OS X平台中对服务器进行开发，可以方便简单的利用GlassFish3.1架构服务器，也可以利用连接数据库，从而快速的对数据库进行访问和操作。

##### 2. 关于JavaServlet

在本课题中，服务器利用Servlet完成服务器端数据库的访问以及与客户端的沟通。一个Servlet就是Java编程语言中的一个类，它被用来扩展服务器的性能，服务器上驻留着可以通过“请求-响应”编程模型来访问的应用程序。虽然Servlet可以对任何类型的请求产生响应，但通常只用来扩展Web服务器的应用程序。Java Servlet技术为这些应用程序定义了一个特定于HTTP的Servlet类。

javax.servlet和javax.servlet.http包为编写servlet提供了接口和类。所有的Servlet都必须实现Servlet接口，该接口定义了生命周期方法。

当实现一个通用服务时，可以使用或扩展由Java Servlet API提供的GenericServlet类。HttpServlet类提供了一些方法，诸如doGet和doPost，以用于处理特定于HTTP的服务。

除上述技术外，在基于iPhoneOS的即时通信系统的开发中运用到了如下Java相关技术：

JDBC（Java Database Connectivity）提供连接各种关系数据库的统一接口，作为数据源，可以为多种关系数据库提供统一访问，它由一组用Java语言编写的类和接口组成。JDBC为工具/数据库开发人员提供了一个标准的API，据此可以构建更高级的工具和接口，使数据库开发人员能够用纯Java API 编写数据库应用程序[3]。

EJB(Enterprise JavaBeans）使得开发者方便地创建、部署和管理跨平台的基于组件的企业应用。在本课题实现的基于iPhoneOS的即时通信系统的Servlet中，通过添加代码（实体会话bean），可完成对数据库的相应操作。

# 基于iPhoneOS的即时通信系统的需求分析

本章首先对基于iPhoneOS的即时通信系统的任务目标及使用者特点进行了分析；然后，从性能和功能两方面，分析了系统的需求，并选择瀑布模型作为软件开发模型。

## 第一节 任务概述

### 一、目标

基于iPhoneOS给出的基本功能，实现核心部分——建立手机客户端和网络端的数据连接，服务器端通过应用程序连接后台数据库的连接；建立相关数据库；实现文本聊天、添加表情、保存历史记录等功能；进行测试，完成基于iPhoneOS的即时通信系统。

### 二、用户特点

众所周知，自从iPhone问世以来，手机市场就掀起一场新兴的革命。众多诺基亚、摩托罗拉、三星的忠实用户像发现了新大陆一样，纷纷转战iPhone战场。一时间，大尺寸屏幕、全屏触摸手写、重力感应成为人们选择手机的几个硬性指标。国人iPhone的拥有量也在不断增长。

调研显示：iPhone用户中本科及以上学历的占70%以上，月收入5000元以上的占60%，大部分为企业中/高层管理人员、专业技术人员和办公室职员。在2011年以前，iPhone手机用户年龄段主要以25〜34年龄段为主，而随着时代的发展，该手机使用群体不断向更低年龄层扩散。对于25〜34岁的人群，他们中的大多数是有稳定收入来源的工作者，更有一部分是企业的中坚力量，这类人群的消费能力也会较强。iPhone手机作为比较成熟的智能手机，不仅具备通信、娱乐等功能，还有诸如资讯、社交和商务的功能和应用，能够很好地满足25〜34岁这类带着明显“商务”标签人群的需求。对于20-25岁的人群，多为大学生一族和刚入职场的年轻白领一族，他们经常使用手机进行上网，与他人进行沟通交流。

## 第二节 需求规定

下面将简单介绍系统当中的性能要求，并对基本功能进行分析。

### 一、性能规定

时间特性要求：响应时间：正常网络条件下，文本显示（相关提示不得超过1秒）

数据转换和传送时间：0.5s以内

容量需求： 磁盘空间：40GB以上

内存： 1.00GB及以上

灵活性规定： 运行环境：客户端可接受iPhoneOS平台、安卓平台、Windows Phone平台

### 二、功能用例分析

下面将结合用户用例图进行基于iPhoneOS的即时通信系统的需求分析。

#### （一）用户用例图

本系统中用户用例图如图2所示。

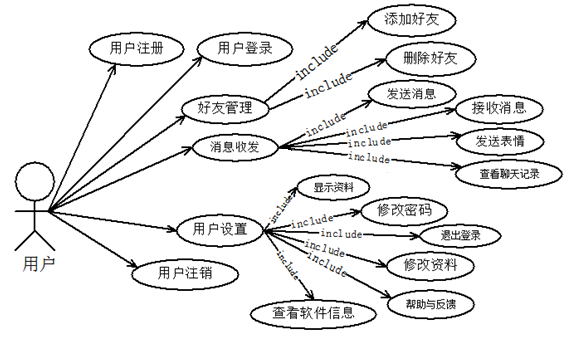


图 2用户用例图

#### （二）需求分析

用户注册：如果用户没有可用的账号ID，可以填写相关的个人信息（用户名、密码、性别、年龄、邮箱、所在城市）进行注册，向服务器提交后服务器返回相关信息。

用户登录：用户通过提交用户名和密码进行登录即时通信系统；

好友管理：用户可以通过好友ID号，进行添加、删除好友的操作；

用户向好友发送信息：用户登录成功后可选定某位好友并向其发送信息，若该好友不在线，该信息会暂存于服务器中，待其上线后收到信息；

用户接收好友信息：用户登录成功后会监听好友发送过来的信息，并显示到相应的界面上；

查看聊天记录：用户可查看与某位好友聊天记录；

发送表情：在聊天过程中，可插入表情进行发送；

用户配置：可以在客户端修改密码及个人资料，并将意见反馈给开发者，还能查看关于软件的信息；

用户退出：用户退出该系统后，可更换账号再次登录；

用户注销：用户在客户端进行注销，删除数据库中该用户信息，退出本系统。

#### （三）系统非功能性需求分析

可靠性：凡是合法用户可以在任意安装有该系统的手机客户端正确无误的访问系统中的信息；

安全性需求：在网络传输过程中，保证数据内容私密性，不公开。系统内部包含的所有资料只有注册用户方能浏览。系统应组织一切通过非登录手段获取系统信息的不安全行为。

可维护性：系统故障可以在24小时内得到解决；

可扩展性：系统需要具有良好的可扩展性，便于系统的维护和升级。整个系统采用弹性架构进行设计，降低需求变更后程序修改所付出的代价。

#### （四）软件生命周期过程选型

采用软件工程的管理思想，在项目开发前选用一个合适的软件生命周期模型，是项目成功的关键因素。由于本文中系统需求较为明确，选用软件工程中的瀑布模型作为开发模型，瀑布模型如图3所示。

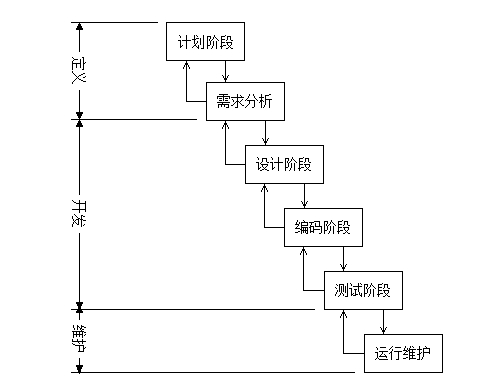


图 瀑布模型图

瀑布模型是当今应用最广泛的一个软件开发模型，其开发过程是通过一系列阶段顺序展开的，从系统需求分析开始直到产品发布，每个阶段都会产生循环反馈， 因此，如果有信息未被覆盖或者发现了问题，那么最好返回上一个阶段并进行适当的修改，开发进程从一个阶段流动到下一个阶段。瀑布开发模型的开发阶段主要分为需求分析、系统设计、实现和测试几个阶段。

在本系统开发过程中，在需求阶段通过系统功能用例图进行建模；设计阶段中首先形成系统总体架构，其次确定系统技术路线，最后完成各模块详细设计；系统实现部分中根据系统模块设计进行具体编码实现，最终对系统功能进行测试工作。

# 基于iPhoneOS的即时通信系统的整体结构

本章介绍系统的总体框架及相关解决方案。系统采用C/S体系结 构，采用HTTP协议作为即时通信协议，通过在Netbeans中架设GlassFish服务器作为即时通信平台，实现客户端与客户端之间的即时通信。

## 第一节 系统整体结构分析

下面将从系统整体结构和MVC架构思想两个方面进行介绍。

### 一、系统整体结构示意图

系统整体结构示意图如图4所示。

用户配置配置配置

修改个人信息

账户注销

消息收发

查看个人资料

修改密码

退出登录

初始界面

登录

注册

提示进入

服务器

数据库

好友管理

查询添加

删除

图 4整体结构示意图

基于iPhoneOS的即时通信系统，采用客户端(C)／服务器端(S)架构的体系结构，主要包括客户端、服务器、数据库三大部分。

客户端：以iPhone手机为平台，在客户端，用户需进行注册、登录，并可对好友进行管理，查询、修改账户信息、密码，通过服务器端实现与iPhone客户端之间的即时通信。客户端负责初始化通信过程，进行即时通信时，由客户端负责向服务器发起创建连接请求。

服务器端：采用Netbeans中的GlassFish架设服务器，允许多个客户端同时登录并且并发的连接到一个服务器上。通过服务器访问数据库，进行数据的写入、查找、更新等操作。

数据库：对用户ID、用户名、用户密码及其他相关资料，好友情况，聊天时间、内容、聊天对象等信息进行记录存储。

### 二、MVC三层架构设计

系统的MVC三层架构设计如图5所示。



图 5 MVC三层架构图

1．在本架构中，顶端为iPhone客户端，表现层与客户端相连，即面对用户；控制层由Servlet控制，其功能为：

* 接收表现层传递的信息；
* 比对数据库并获取结果；
* 指示表现层跳转页面；

2．Servlet将表现层传递的信息（如密码）与数据库中对应的信息比对，根据不同的结果（不同返回值），Servlet指示表现层跳转到相应的不同界面（ViewController）

3．数据库中的多个表分别对应四个实体类：用户信息实体类、好友管理实体类、好友请求实体类、消息收发实体类。Servlet比对数据库时，将通过这些实体类查找数据库，然后进行比对，获得结果后指示表现层页面的跳转。

## 第二节 各主要功能处理流程

下面将从系统的处理流程进行分析与说明。

一、处理流程示意图

基于iPhoneOS的即时通信系统的结构主要可以分为四个模块，分别为用户注册登录、好友管理、消息收发、用户配置，处理流程示意图如图6所示。

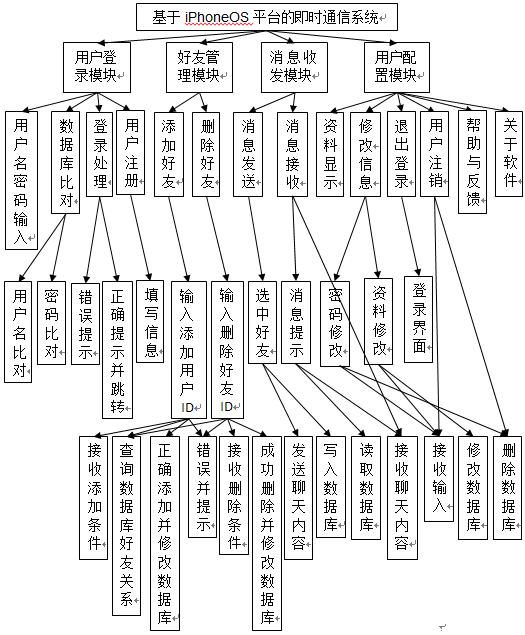


图 处理流程示意图

### 二、处理流程说明表

在此将模块按级别划分，分别从功能、输入输出、调用关系这三个方面对系统流程处理过程列出详细说明，如表1。

表 处理流程说明表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 模块清单 | | | | | | |
| 一级模块 | 二级模块 | 三级模块 | 四级模块 | 功能 | I/O | 调用 |
| 基于iPhoneOS的即时通信系统 | 用户注册登录 | 用户名密码输入 |  | 用户输入账号密码信息经验证后登录到系统中 | 输入：用户名密码输入 | 由“基于iPhoneOS的即时通信系统”调用 |
| 数据库比对 |  | 将用户输入的账号与密码分别与数据库中现存的账号密码进行比对 | 输入：用户名/密码  输出：比对结果 | 由“用户注册登录”模块调用 |
| 登录处理 |  | 根据数据库比对的结果，将登录信息反映到客户 | 输入：数据库比对结果  输出：错误提示/正确提示 | 由“用户注册登录”模块调用 |
| 用户注册 |  | 用户输入用户名、密码、性别、年龄、邮箱、城市进行注册，将信息写入数据库。 | 输入：用户信息（用户名、密码、性别、年龄、邮箱、城市）  输出：注册成功/失败提示 | 由“用户注册登录”模块调用 |
| 好友管理 | 添加好友 |  | 由用户输入要添加的用户ID，通过查找数据库找到匹配用户，返回添加提示 | 输入：用户ID 输出：数据库、添加提示（结果） | 由“添加好友”模块调用从而被“好友管理”模块调用 |
| 删除好友 |  | 由用户输入要删除的好友ID，通过查找数据库找到匹配用户，返回删除提示 | 输入：好友ID 输出：数据库、删除提示（结果） | 由“删除好友”模块调用从而被“好友管理”模块调用 |

续表 1处理流程说明表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 模块清单 | | | | | | |
| 一级模块 | 二级模块 | 三级模块 | 四级模块 | 功能 | I/O | 调用 |
|  |  | 消息发送 |  | 选中好友后，发送聊天内容，将聊天相应信息写入数据库 | 输入：好友用户名、聊天内容 | 由“消息发送”模块调用从而被“消息收发”模块调用 |
| 基于iPhoneOS的即时通信系统 | 消息收发 | 消息接收 | 弹窗查看 | 提示消息，点击查看，可通过查找数据库查看好友发送的聊天内容 | 输入：查看操作  输出：数据库（聊天内容） 提示信息 | 由“消息接收”模块调用从而被“消息收发”模块调用 |
| 对话框直接查看 | 与好友聊天对话框中直接显示好友发送聊天内容 | 输出：数据库（聊天内容） | 由“消息接收”模块调用从而被“消息收发”模块调用 |
| 用户配置 | 资料显示 |  | 读取数据库中该用户信息 | 输入：显示资料操作  输出：数据库 | 由“用户配置”模块调用 |
| 修改信息 | 修改密码 | 用户修改密码，修改数据库原有信息 | 输入：新的密码 输出：数据库 | 由“修改信息”模块调用从而被“用户配置”模块调用 |
| 修改个人信息 | 用户修改个人信息，修改数据库原有信息 | 输出：数据库 | 由“修改信息”模块调用从而被“用户配置”模块调用 |
| 退出登录 |  | 当前用户退出登录 | 输入：退出操作  输出：相关提示 | 由“退出登录”模块调用从而被“用户配置”模块调用 |
| 用户注销 |  | 用户输入用户名、密码并确认注销，从数据库删除该用户信息 | 输入：用户名、密码 输出：数据库 | 由“用户注销”模块调用从而被“用户配置”模块调用 |
| 帮助与反馈 |  | 用户提交软件使用意见或建议 | 输入：意见或建议  输出：数据库 | 由“帮助与反馈”模块调用从而被“用户配置”模块调用 |
| 关于软件 |  | 查看软件版本、开发者信息 | 输入：“关于软件”查看操作  输出：客户端 | 由“关于软件”模块调用从而被“用户配置”模块调用 |

## 第三节 相关技术及难点分析

### 一、名词解释

MVC三层架构： MVC是一种设计模式，它强制性的使应用程序的输入、处理和输出分开。使用MVC应用程序被分成三个核心部件：模型、视图、控制器。它们各自处理自己的任务。使用MVC架构可以使多个视图能共享一个模型，由于已经将数据和业务规则从表示层分开，所以可以最大化的重用代码了。其次，由于模型返回的数据没有进行格式化，所以同样的构件能被不同界面使用。再次，因为模型是自包含的，并且与控制器和视图相分离，所以很容易改变你的应用程序的数据层和业务规则。

本系统中采用的主要方法如下，在表现层中主要使用Storyboard技术、iPhone客户端开发技术，在控制层中使用用Servlet技术、会话Bean技术，在数据层中使用数据库技术。

服务器通信： 服务器端采用了Netbeans中自带的Java DB数据库，并通过Servlet对其进行创建、修改、访问等操作，用户在手机终端发出相应的HTTP请求，服务器端的Servlet捕捉到请求，拆分请求内容，对数据库进行操作，响应用户的请求。

Storyboard： Xcode4.2新加入的编辑场景。在传统的iOS程序中，通常你可以把一个nib/xib文件想象成是一个要设计应用程式的主界面，每一个nib/xib文件背后都会有一个控制元件，也就是MVC中的Controller，因此可以把一个View看做一个场景，而这个场景中会有各种角色（UIView）在里面活动。而新加入的Storyboard就是将每一个场景通过事件切换的方式贯穿起来，，图7为Storyboard搭建登录界面和注册界面的截图。



图 Storyboard搭建项目工程

### 二、技术难点

在系统实现时，需要攻克一些技术难点，下面将从服务器端和客户端两方面进行分析说明。

#### （一）服务器端

##### 1．读取字符流

由于服务器需要处理从客户端发送过来的信息，信息以数据流的形式发送。数据按流向可分为输入流和输出流，按照形式可分为字符流和字节流。在服务器Servlet中，response实现HttpServletResponse接口，request实现HttpServletRequest接口[3]。

（1）两种输出流：

字节形式输出：response.getOutputStream()

字符形式输出：response.getWriter（）

（2）两种输入流：

字节形式输入：request.getInputStream()

字符形式输入：request.getReader（）

数据流传送方式如图8所示。



图 8 数据流传送方式

BufferReader是Reader的直接子类。通过BufferReader.readLine()方法可以实现读取文本行、返回字符串，该方法能直接返回字符串，使用得比较广泛。

BufferReader具有以下两种构造方法。

BufferReader(Reader in):根据in代表的Reader对象创建BufferReader实例，缓冲区大小采用默认值。

BufferReader(Reader in,int sz):根据in代表的Reader对象创建BufferReader实例，缓冲区大小采用指定sz值。

BufferReader.readLine()方法遇到以下字符或者字符串认为当前行结束：‘\n’(换行符),'\r'(回车符),'\r\n'(回车换行)。返回值为该行内容的字符串，不包含任何行终止符，如果已到达流末尾，则返回null。

在服务器端，本系统采用BufferReader.readLine()的方法来读取客户端发送过来的数据流，如图9。

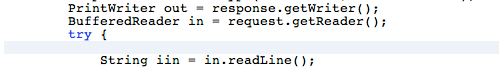


图 9读取数据流

处理输出流时，response.getWriter() 调用了 response的方法 getWriter()，这个getWriter()在response里面创建了一个对象，然后把内存地址传递给out，如图10。



图 10处理输出流

处理输入流是通过request.getReader()方法建立输入流。用readLine方法获得post方式所传过来的数据，如图11。



图 11处理输入流

##### 2．EJB技术中会话Bean的使用

会话Bean是Java组件，其既可运行于独立的EJB容器中，也可运行于作为标准Java平台、Java EE应用程序服务器一部分EJB容器中。这些Java组件通常用于建模某一特定用户任务，如输入顾客信息或实现某一维持客户端会话状态的过程。

会话Bean既能用于传统两层或三层架构的“胖”客户端程序，也能用于三层基于Web的应用程序，这些应用程序能被部署在不同的逻辑或物理层。在本课题实现的基于iPhoneOS的即时通信系统中，将会话Bean用于三层架构中，如图12所示。手机客户端设备，提供用户界面，在客户端提供用户动作，如登录、添加好友、发送消息等，通过Web技术（Servlet），调用运行某个EJB容器中的会话Bean。一旦会话Bean被调用，它会处理请求，并发回响应到Web程序（Servlet）中，而Web程序进行相应处理后，再发回到客户设备上（手机客户端）。

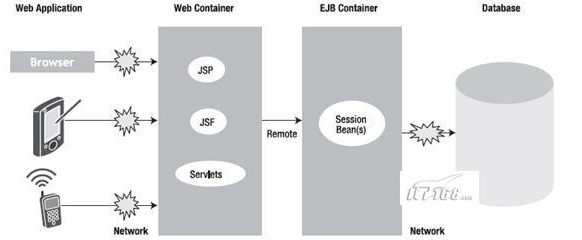


图 12三层架构应用程序中的会话Bean

##### 3．Servlet对数据库进行操作

由于在服务器端的Servlet中无法对数据库表进行直接操作，因此需要将数据库中的表转化为相应的实体类，以实现Servlet对数据库的操作。

实体类是用于对必须存储的信息和相关行为建模的类。实体对象（实体类的实例）用于保存和更新一些现象的有关信息。一个实体类对象实例对应数据库表中的一条记录，主键对应表的主键，字段对应表的列。在Netbeans环境中使用JavaDB创建数据库，指定数据库名，用户名（app），密码（app），再在创建的数据库中建表，如图13。之后通过数据库生成实体类，如图14，创建实体类的会话Bean，使用会话Bean进行数据库查询。



图 13数据库中建表

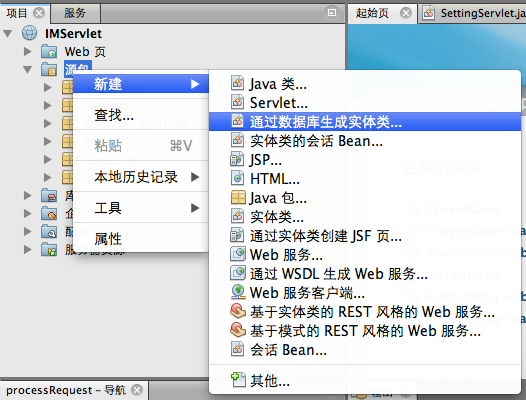


图 14数据库生成实体类

#### （二）客户端

在实现客户端时，将涉及到网络连接、页面间传值、全局变量声明、定时器的使用等相关技术难点，下面将进行详细分析。

##### 1．iOS中的网络连接

本系统的主要功能即是实现即时通信，所以，建立iOS与客户端的网络连接是开发软件过程中最重要的部分。

在iPhoneOS开发中，URL加载系统是一组类和协议，提供了访问URL指定的数据的基础能力。URL加载系统实际包含5类：URL加载、缓存管理、认证和凭据、Cookie存储和协议支持。在iOS中，这些类都是继承于根类NSObject的，包含在Foundation framework基础框架中[4]。

NSURLRequest对象代表一个请求URL内容的句柄。NSURLRequest类在一个独立于协议的方式下封装了一个网址和任何协议特定的属性。NSMutableURLRequest类是NSURLRequest可变的子类，允许客户端应用程序改变现有的请求。NSURLRequest和NSMutableURLRequest类均提供一套存储方法来管理网络协议的一些属性，如HTTP协议。NSURLRequest对象提供了返回HTTP请求正文及标题的方法，并且可以设置传输数据方法；NSMutableURLRequest在此基础上还提供了修改这些内容的方法。由于即时通信的信息具有可变性，本系统中采用NSMutableURLRequest实现HTTP通信。

在客户端开发中，网络连接流程如图15。



图 15网络连接流程

即时通信的具体实现如下，首先将网址名称存入NSURL，如图16所示；



图 16 定义网址名称

然后将要传递的内容的参数转换为NSData，并计算Post提交数据的长度，如图17；



图 17 参数格式转换和数据长度计算

最后设置请求URL，设置提交方法、响应内容长度、相应内容类型，设置通信方式为同步或异步，如图18所示，responseString即为发送请求后从服务器端收到的响应内容。



图 18 配置HTTP请求参数并发送

##### 2．页面间传值

在客户端中，在好友列表加载前需要获得好友名单，这就需要在登录界面时向服务器请求获得该名单，然后通过页面间传值，将当前用户名和其好友名单传到好友列表界面。在系统中，可采用delegate来实现页面间传值，其实质是先在当前页面A中得到B页面的指针，然后将需要传的值赋页面B，之后跳转。

具体实现如下，首先定义一个委托类UIViewPassValueDelegate用来传递值，如图19所示；



图 19新建delegate

然后在A页面的.h文件里，声明delegate，如图20所示；

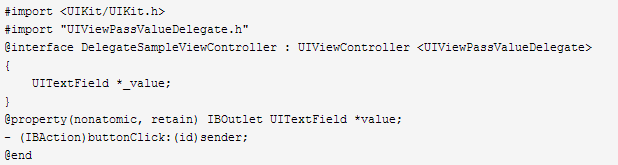


图 声明delegate

在A页面的.m文件里实现这个委托方法，如图21；

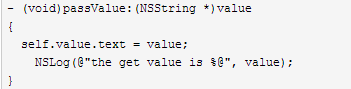


图 21实现委托方法

然后在需要跳转的时候，将页面B的delegate实现方法指向页面A，如图22；

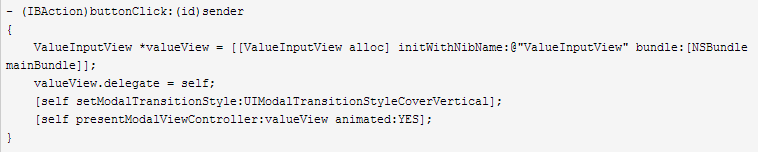


图 B中delegate实现方法

在B页面的.h文件里给相关对象加入协议类，如图23；

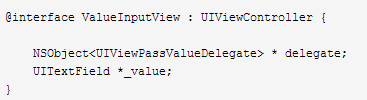


图 23 B中添加协议类

在B页面的.m文件里面，如下所示，调用协议类实现数值读取：

- (void) setValue:(NSString \*) value{  
     \_value = value;

}

##### 3．全局变量的声明

在客户端中，需要将身份表示定为全局变量，以方便在同一页面中的其他方法中调用身份信息向服务器发送请求，因此需要掌握全局变量声明的方法。此外，当在某一方法中向服务器端请求并获得返回值后，需在别的方法中调用该返回值，因此也需要在程序中使用到全局变量。在系统中，全局变量的声明主要采用了以下两种方法。

第一种方法是单例[5]的全局访问方法，首先在.h文件中设置全局变量，如图24；

图 24设置全局变量

@interface AppDelegate

{

NSString \*myName;

}

@property (nonatomic, retain) NSString \*myName;

@end

然后在对应的.m文件中实现全局变量：

@synthesize myName;

值得一提的是，@synthesize 应写在@implementation 下方，否则程序将报错。

第二种方法是使用extern关键字[6]，首先在.h文件中写入需要的全局变量名，如图25。

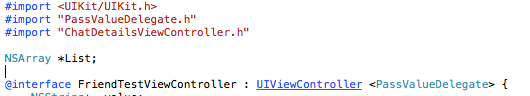


图 25 .h文件写入全局变量名

注意该全局变量名应置于#import之下，@interface之上。

然后，在相应的 .m文件中，给全局变量初始化或者赋值，如图26；

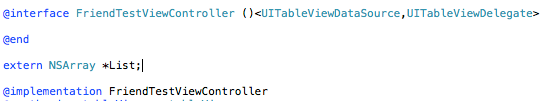


图 26 .m文件初始化全局变量

注意初始化内容应置于@interface@end之下，@implementation之上。

##### 4．动态列表

在系统中，用户登录后，进入到好友列表界面，该界面通过从数据库中获取到当前用户的好友名单，然后将其一列一列的显示在动态的tableView中[7]，实现方法如下：

首先，将好友名单存入List数组中；然后在tableView的cellForRowAtIndexPath方法中，设置cell中的text显示为List数组中的每一行内容，具体实现如图27；



图 27 tableView的cellForRowAtIndexPath方法

接下来，在tableView的numberOfRowsInSection方法中返回List数组的元素个数，实现如图28；

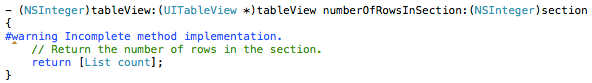


图 28 tableView的numberOfRowsInSection方法

最后，在viewDidLoad中加入List，如图29；

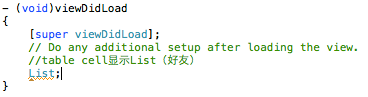


图 29 viewDidLoad方法

##### 5．实时监听、接收好友添加信息和聊天信息

由于请求只能由客户端主动发出，而服务器端无法主动将相应消息发送到相应客户端，所以实现获取好友添加请求、聊天消息等实时信息，只能通过在客户端设定一个定时器的方法。用户一旦登录成功，便在客户端定时的不断的向服务器端发送请求，服务器判定是否有对应发给该用户的消息，若有则响应，把相应消息发送至客户端。

iOS中定时器的使用流程如图30。



图 30定时器使用流程图

具体使用方法如下：

（1）初始化[8]

+(NSTimer\*)timerWithTimeInterval:(NSTimeInterval)titarget:(id)aTarget selector:(SEL)aSelector userInfo:(id)userInfo repeats:(BOOL)yesOrNo;

+(NSTimer\*)scheduledTimerWithTimeInterval:(NSTimeInterval)titarget:(id)aTarget selector:(SEL)aSelector userInfo:(id)userInfo repeats:(BOOL)yesOrNo;

（2）触发（启动）

当创建定时器后，该定时器将在初始化时指定的timeInterval秒后自动触发，也可以使用-(void)fire;方法立即触发该定时器。

（3）停止

- (void)invalidate;这个是唯一一个可以将计时器从runloop中移出的方法（在本课题实现的系统中需一直打开定时器，向服务器发送请求，实现即时接收消息，因此未使用停止的方法）。

##### 6．表情图标显示与存储

在聊天时，通常会加入一些形象的表情图标，一方面可以增添聊天趣味，另一方面也可以更准确的传达聊天者情绪。要实现表情图标的显示与存储，首先，需在系统的支持文件夹中加入一个表情文件夹AdiumEmoticon，用来存储需要用到的所有表情图标。

系统中的每个表情图标对应一个特殊符号组成的字符串，在表情页面中，每行tablecell对应一个图标，在每一行中同时显示图标与特殊字符。当用户从表情页面选定表情后，会在聊天消息中，插入该表情所对应的字符串，在点击发送之后，textView会监测该消息字符串中是否含有表示表情的特殊字符串，若有，则在该字符串所在位置，新建一个imageView,imageView中设置image与特殊字符串一一对应，将字符串替换为该image，具体代码如图31所示；



图 31特殊字符转换为表情图标

而发送至服务器的消息仍以字符串的形式进行发送和存储，在客户端读取聊天消息并进行显示时，在采用上述的方法在客户端进行转换。

##### 7．键盘隐藏

在系统的页面中，当输入框处于屏幕下方时，若不进行相关操作处理，当编辑UITextField的时候，弹出键盘就会挡住输入框，通过让输入框随键盘一起向上移动，当关闭键盘时让输入框随之一起向下移回原来位置的思想方法，就可以解决键盘挡住输入框的问题。实现方法如下：

首先，在viewDidLoad函数中加入如图32中代码，获得iOS系统版本号；

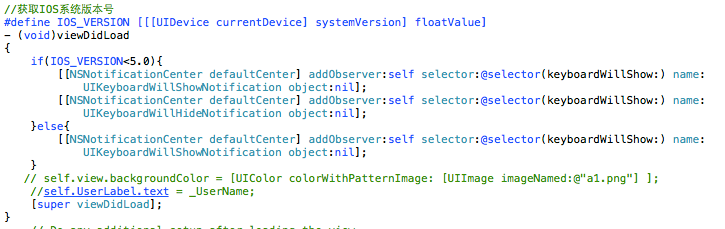


图 32 viewDidLoad中添加代码

然后，重写keyboardWillShow方法，如图33，即可实现视图随键盘的出现而上移的操作。



图 33 keyboardWillShow方法

接下来在.h文件中声明- (IBAction)backgroundTap:(id)sender;方法，在Storyboard中设置该页面Custom Class为UIControl，在Sent Events中，选中Touch Down右边的小圈圈，同时按住Ctrl键，将鼠标拖至.h中声明的backgroundTap方法。

最后在 .m文件中写入如图34代码，该backgroundTap方法即可实现鼠标单击空白处，键盘隐藏，视图回到原高度的操作。



图 34 backgroundTap方法

# 基于iPhoneOS的即时通信系统的概要设计

本章主要从系统数据结构、服务器端、客户端、传输模块四个方面进行概要设计。

由于本课题需要存储用户信息、用户好友关系以及聊天消息等内容，所以首先需要设计搭建相关的数据库表来实现数据的结构化、永久性存储。之后需要在服务器端构建Servlet来处理前端

用户对数据库的操作。客户端可获取外部客户输入信息，并按照用户的操作进行相应的页面跳转。最后，通过传输模块，完成服务器与客户端的连接，进行消息的收发与传送。

## 系统数据结构设计

### 一、数据库表关系设计

数据库关系表设计对应描述如下：

#### （一）实体表设计

##### 1．Usertable 用户信息表

用于存储用户ID、账号、用户密码、用户性别、用户年龄、用户邮箱和用户所在城市。

##### 2．Friendtable 好友管理表

用于存储用户与用户之间的好友关系。好友之间联系相对于某一用户实体而言属于一对多的关系，所以建表时，每两用户间的好友关系采用两条记录进行双向存储。

##### 3．FriendRequesttable 好友请求表

在两用户正式成为好友之前，需要进行“请求添加”、“通过/不通过添加请求”、“添加信息反馈”三个步骤，在此过程中，两用户的好友关系不能正式确立，需要建一个表来记录请求过程，当双方正式成为好友之后，好友关系才被记录进入FriendTable好友管理表。

##### 4．Messagetable 消息收发表

该表用于记录两用户间聊天的信息，包括发送者和接收者的ID、发送时间、发送内容、和是否成功发送的标记。若发送消息时，对方在线，成功发送后，发送标记置为1，表示发送成功；若当时对方不在线，标记置为0，表示还未发送，待对方上线后发送，再把标记置为1。

#### （二）表关系设计

四个表之间UML关联图，如图35所示。

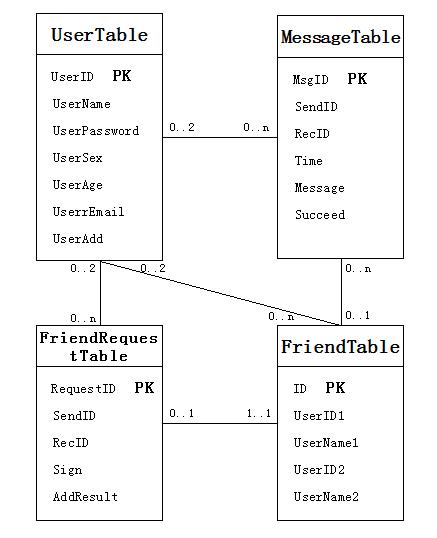


图 35 UML关联图

### 二、数据表设计

系统数据库中各类实体表详细设计如表2至表5所示。

表 Usertable 用户信息表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 版块表：Usertable | | | | | | | |
| 编号 | 属性名 | 属性描述 | 主键 | 外键 | 允许空值 | 数据类型 | 精度范围 |
| 1 | UserID | 编号 | 是 | 否 | 否 | INTEGER | 10 |
| 2 | UserName | 用户名 | 否 | 否 | 否 | VARCHAR | 20 |
| 3 | UserPassword | 密码 | 否 | 否 | 否 | VARCHAR | 40 |
| 4 | UserSex | 性别 | 否 | 否 | 是 | VARCHAR | 10 |
| 5 | UserAge | 年龄 | 否 | 否 | 是 | INTEGER | 10 |
| 6 | UserEmail | 邮箱 | 否 | 否 | 是 | VARCHAR | 50 |
| 7 | UserAdd | 地址（城市） | 否 | 否 | 是 | VARCHAR | 50 |

表 Friendtable 好友管理表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 版块表：Friendtable | | | | | | | |
| 编号 | 属性名 | 属性描述 | 主键 | 外键 | 允许空值 | 数据类型 | 精度范围 |
| 1 | ID | 编号 | 是 | 否 | 否 | INTEGER | 10 |
| 2 | UserID1 | 用户1编号 | 否 | 否 | 否 | INTEGER | 10 |
| 3 | UserName1 | 用户1用户名 | 否 | 否 | 否 | VARCHAR | 30 |
| 4 | UserID2 | 用户2编号 | 否 | 否 | 是 | INTEGER | 10 |
| 5 | UserName2 | 用户2用户名 | 否 | 否 | 是 | INTEGER | 30 |

表 FriendRequesttable 好友请求表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 版块表：FriendRequesttable | | | | | | | |
| 编号 | 属性名 | 属性描述 | 主键 | 外键 | 允许空值 | 数据类型 | 精度范围 |
| 1 | RequestID | 请求编号 | 是 | 否 | 否 | INTEGER | 10 |
| 2 | SendID | 发送者编号 | 否 | 否 | 否 | INTEGER | 10 |
| 3 | RecID | 接收者编号 | 否 | 否 | 否 | INTEGER | 10 |
| 4 | Sign | 标记 | 否 | 否 | 是 | VARCHAR | 10 |
| 5 | AddResult | 添加结果 | 否 | 否 | 是 | VARCHAR | 20 |

表 Messagetable 消息收发表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 版块表：Messagetable | | | | | | | |
| 编号 | 属性名 | 属性描述 | 主键 | 外键 | 允许空值 | 数据类型 | 精度范围 |
| 1 | MsgID | 消息编号 | 是 | 否 | 否 | INTEGER | 10 |
| 2 | SendID | 发送者编号 | 否 | 否 | 否 | INTEGER | 10 |
| 3 | RecID | 接收者编号 | 否 | 否 | 否 | INTEGER | 10 |
| 4 | Time | 发送时间 | 否 | 否 | 是 | VARCHAR | 50 |
| 5 | Message | 消息内容 | 否 | 否 | 是 | VARCHAR | 1000 |
| 6 | Succeed | 成功标记 | 否 | 否 | 是 | VARCHAR | 10 |

## 服务器端设计

服务器端可以由若干个相关Servlet组成，用来分别实现系统所要求的一些基本功能，这些基本功能主要包括会话的管理，用户与服务器端的通信，服务器之间的通信，存储用户的个人信息和联系人信息名单，同时也要在服务器端保留用户在离线时所收到的信息，用户注册，用户的身份和权限验证等等，这些也同样涉及到与数据库服务器之间连接并对数据进行操作，而Servlet对数据库的操作需要通过实体类和会话Bean实现。

### 一、实体类与会话Bean

Netbeans配置数据库时，可以使用内嵌的数据库，当然为了提高性能，也可以使用其它的大型数据库进行连接，如Oracle、Access等，使用大型数据库时需要提供JDBC的支持。在本服务器中，本系统中采用Netbeans内嵌的数据库，数据库中负责存储用户信息数据，好友关系列表，收到的消息等历史记录持久化的信息。

在Netbeans中创建好数据库和表之后，从数据库构建实体类，设置包名为“entity”，并将实体类存放于该包中，如图36。在系统生成相应的实体类之后，构建一个实体类的回话Bean来调用改项目访问数据库。将生成的相应会话Bean，存于名为“session”的包中，如图37。

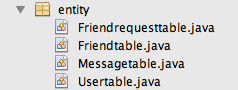


图 实体类 entity包



图 会话Bean session包

数据表—实体类—会话Bean对应关系，如表6所示。

表 数据表—实体类—会话Bean对应关系

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 数据表 | 实体类 | 会话Bean |
| 1 | Usertable | Usertable.java | UsertableFacade.java  UsertableFacadeLocal.java |
| 2 | Friendtable | Friendtable.java | FriendtableFacade.java  FriendtableFacadeLocal.java |
| 3 | Friendrequesttable | Friendrequesttable.java | FriendrequesttableFacade.java  FriendrequesttableFacadeLocal.java |
| 4 | Messagetable | Messagetable.java | MessagetableFacade.java  MessagetableFacadeLocal.java |

### 二、Servlet对应响应操作

在Netbeans项目中，通过构建多个不同Servlet来响应客户端发来的不同请求，以实现相应的功能操作。Servlet名称与实现功能及与会话Bean的调用关系如表7所示。在Servlet中通过操作实体类的会话Bean来实现对数据库的访问。

表 Servlet功能操作表

|  |  |  |  |
| --- | --- | --- | --- |
| Servlet | | | |
| 编号 | 名称 | 实现功能 | 需要调用的会话Bean |
| 1 | LoginServlet | 用户登录 | UsertableFacadeLocal usertableFacade; |
| 2 | RegisterServlet | 用户注册 | UsertableFacadeLocal usertableFacade; |
| 3 | AddFriendServlet | 添加好友（发送添加请求） | UsertableFacadeLocal usertableFacade;  FriendrequesttableFacadeLocalfriendrequesttableFacade;  FriendtableFacadeLocalfriendtableFacade; |
| 4 | AddedServlet | 处理是否有发送给当前用户的好友添加消息 | UsertableFacadeLocal usertableFacade;  FriendrequesttableFacadeLocalfriendrequesttableFacade;  FriendtableFacadeLocal friendtableFacade; |
| 5 | AgreeFriendServlet | 被添加用户响应添加请求 | UsertableFacadeLocal usertableFacade;  FriendrequesttableFacadeLocal  friendrequesttableFacade;  FriendtableFacadeLocal friendtableFacade; |
| 6 | AgreeBackServlet | 处理好友添加请求被响应后结果 | UsertableFacadeLocal usertableFacade; |
| 7 | DeleteFriendServlet | 删除好友 | FriendtableFacadeLocal friendtableFacade; |
| 8 | ChatGoServlet | 选定好友发起聊天，发送聊天消息 | UsertableFacadeLocal usertableFacade;  MessagetableFacadeLocal messagetableFacade; |
| 9 | ChatAlertServlet | 聊天消息提醒（好友列表页） | UsertableFacadeLocal usertableFacade; MessagetableFacadeLocalmessagetableFacade; |
| 10 | ChatRecAlertServlet | 处理接收消息弹窗操作 | UsertableFacadeLocal usertableFacade; MessagetableFacadeLocal messagetableFacade; |
| 11 | ChatRecServlet | 接收对方回复的聊天信息 | UsertableFacadeLocal usertableFacade;  MessagetableFacadeLocal messagetableFacade; |
| 12 | ChatLogServlet | 查看聊天记录 | UsertableFacadeLocal usertableFacade; MessagetableFacadeLocal messagetableFacade; |
| 13 | SettingServlet | 用户配置 | UsertableFacadeLocal usertableFacade;  FriendrequesttableFacadeLocal  friendrequesttableFacade;  FriendtableFacadeLocal friendtableFacade; |

## 第三节 客户端设计

### 一、外部接口

用户在客户端，通过外部接口进行人机交互，输入数据，客户端中涉及数据接口的相关界面如图38至图43所示。在登录界面中，用户可在客户端从外部输入用户名、密码；在注册界面中，可从外部输入用户名、密码、性别、年龄、邮箱、城市；好友管理界面中可输入用户ID；密码修改界面中，可从外部输入原密码和新密码；资料修改界面中，用户可从外部输入新的个人资料；在帮助与反馈界面中，用户可以从外部向文本框中输入意见与建议。

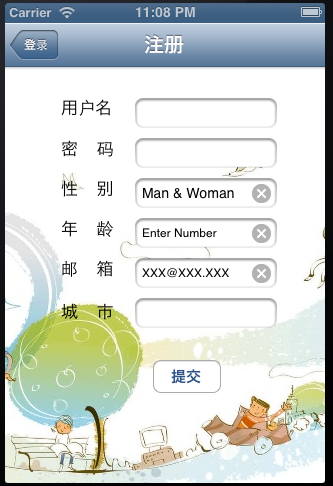
 

图 登录界面 ViewController 图 注册界面RegisterViewController

图 好友管理AddFriendViewController 图 密码修改NewPswViewController

图 资料修改NewInfoViewController 图 43 帮助与反馈helpViewController

### 二、内部接口—界面跳转关系

本系统中各应用界面之间衔接关系如图44所示。

注册界面

RegisterViewController

登录界面ViewController

主界面

Tab Bar Controller

关于软件界面

个人资料显示界面

UserInfoViewController

退出登录界面

好友管理界面

AddFriendViewController

资料修改界面

NewInfoViewController

聊天界面

ChatDetailsViewController

注销界面

LogOffViewController

帮助与反馈界面

账户安全（修改密码）

NewPswViewController

查看聊天记录界面ChatLogViewController

注册

登录成功

注册成功

好友界面

FriendViewController

用户配置界面

SettingViewController

图 44界面衔接关系图

## 第四节 功能模块的传输设计

功能模块包括用户注册、用户登录、添加好友、删除好友、消息收发、查看聊天记录、用户配置、用户退出、用户注销等内容，下面将针对部分模块的传输过程进行重点分析。

### 一、用户注册

在注册过程中，用户输入用户名、密码、性别、年龄、邮箱、城市，这些信息通过HTTP协议发送到服务器端。服务器端把接收到的信息与数据库进行比对，若无重复的用户名则将信息写入数据库，成功后返回该用户ID号，若不成功则在客户端提示注册失败，其传输流程如图45所示。

客户端

服务器

数据库

1．客户端发送用户名、密码、年龄等信息

2．服务器将用户信息发送到数据库中进行查找

4．服务器响应注册是否成功信息及用户ID

3．数据库检测是否有重复用户名，若没有，则返回用户名的ID

图 用户注册

### 二、用户登录

在登录过程中，用户应当输入自己的用户名和对应的密码，登录过程中需要与服务器进行通信验证。验证过程如图46所示，用户只需将自己的用户名和密码，通过HTTP协议发送到服务器端。而服务器端需要根据用户的用户名从数据库中取出相应的密码与之相比对，经过验证后返回是否登录成功的响应。

2．服务器将用户名发送到数据库中进行查找

1．客户端发送用户名和密码

客户端

服务器

数据库

3．数据库将对应用户名的密码返回到服务器端

4．服务器端验证，并响应是否验证成功

图 用户登录

### 三、添加好友

添加好友的过程如图47所示。

数据库

1．客户端1发送需添加的用户ID

2、服务器将发送者和接收者ID发送到数据库进行查找

2、服务器将客户端2与数据库中接收者ID比对

7．返回待客户端2同意/拒绝添加信息

服务器

服务器

客户端2

1、客户端2将本人ID发送到服务器

4、若客户端2ID与数据库中接收者ID匹配，则发送发送者ID

5、服务器发送者ID发送客户端2，提示好友添加消息，等待应答

6．返回客户端2应答结果并修改数据库

客户端1

图 47添加好友

### 四、删除好友

与添加好友步骤类似，但无需经过对方验证，可单方面直接删除好友，解除好友关系。客户端发送ID及需删除好友ID至服务器，服务器从数据库中查找两用户间好友关系，进行删除，之后返回删除成功/不成功提示。

### 五、消息收发

过程与添加好友类似，客户端1将发送聊天信息，通过服务器写入数据库，与此同时客户端2不断向服务器发送自身ID，服务器检测是否有接收者为客户端2ID的消息，若有，则响应客户端2的请求，返回发送至该用户的消息。

### 六、其他功能

除上述功能之外，本课题研究的基于iPhoneOS的即时通信系统还包括查看聊天记录、用户配置、退出登录、账户注销等功能服务器间消息收发传递的过程与上述所提到功能的实现过程基本类似，在此不再赘述。

# 基于iPhoneOS的即时通信系统的详细设计与实现

本章针对基于iPhoneOS的即时通信系统，将从服务器端和客户端两个方面阐述系统详细设计与实现的过程。

## 第一节 服务器端的详细设计与实现

服务器端的设计与实现包括数据库、Servlet以及通过Servlet调用数据库三个方面，下面将进行详细的分析与说明。

### 一、数据库的实现

在数据库中，总共插入了四个表，分别命名为Usertable、Friendtable、FriendRequesttable、Messagetable，如图48所示。

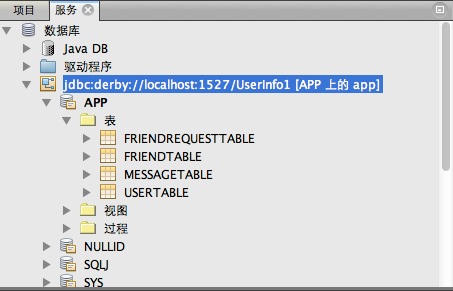
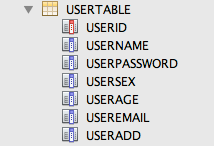
 

图 48建表 图 Usertable的字段

#### （一）用户信息表Usertable

用户信息表Usertable，包括字段UserID、UserName、UserPassword、UserSex、UserAge、UserEmail、UserAdd，如图49。

用户信息表Usertable的具体实现如图50。

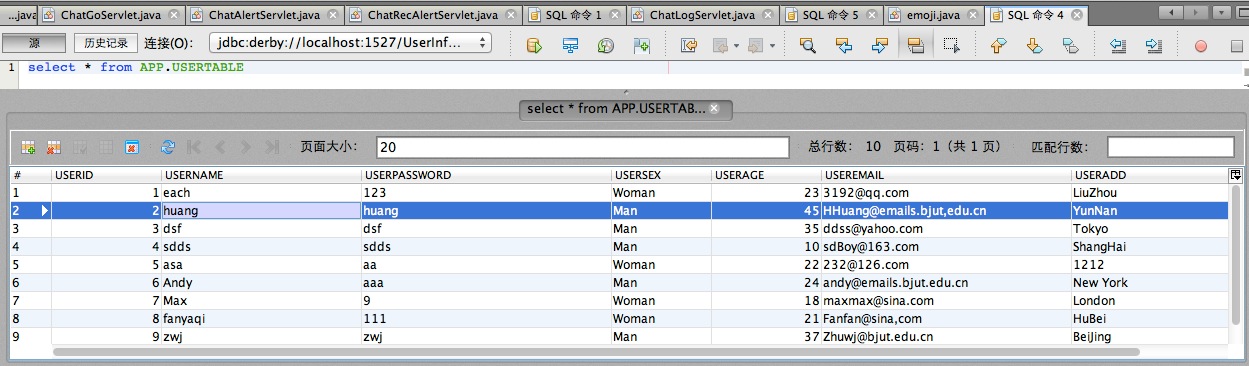


图 50 Usertable的实现

#### （二）好友管理表Friendtable

好友管理表Friendtable，包括字段ID、UserID1、UserName1、UserID2、UserName2，如图51。



图 51 Friendtable字段

好友管理表Friendtable的具体实现如图52。

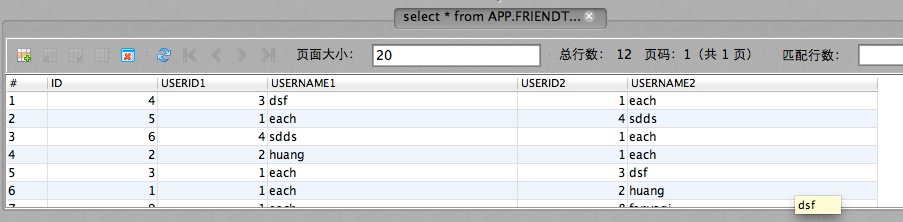


图 52 Friendtable的实现

#### （三）好友请求表FriendRequesttable

好友请求表FriendRequesttable，包括字段RequestID、SendID、RecID、Sign、AddResult，如图53。



图 53 FriendRequesttable字段

好友请求表FriendRequesttable的具体实现如图54。

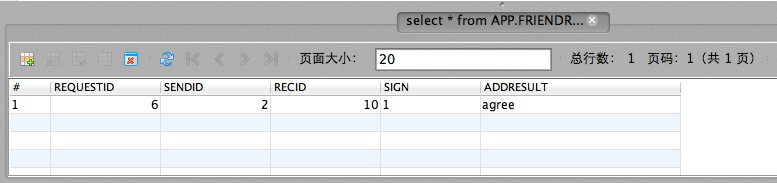


图 54 FriendRequesttable的实现

#### （四）消息收发表Messagetable

消息收发表Messagetable，包括字段MsgID、SendID、RecID、Time、Message、Succeed，如图55。

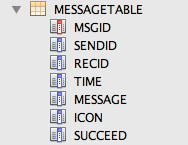


图 55 Messagetable的字段

消息收发表Messagetable的具体实现如图56。

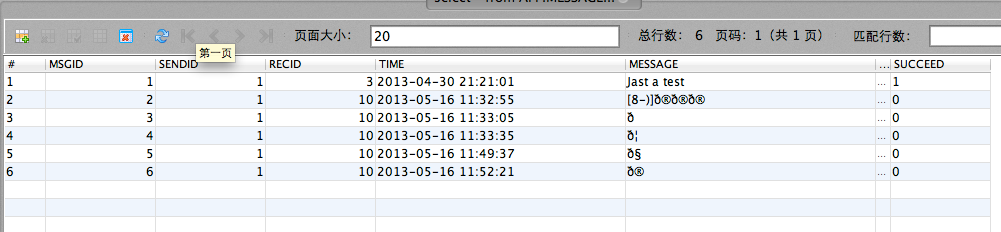


图 56 Messagetable的实现

### 二、Servlet调用数据库方式

1．若在服务器端Servlet中使用数据库中的数据，需要将数据库中的每个表均映射到IMServlet项目中来，这种映射的方法，就是为数据库的每个表构建实体类EntityClass；

2．EJB会话Bean调用实体类EntityClass访问数据库；

3．Servlet操纵实体类EntityClass的会话Bean访问数据库。

### 三、流程逻辑

在Netbeans项目中，通过构建多个不同Servlet来响应客户端发来的不同请求，在本课题实现的基于iPhoneOS的即时通信系统的服务器端中，本系统中构建了12个Servlet完成相应的功能，具体实现的功能及流程如下。

1．LoginServlet

用于实现用户登录的功能，流程图如图57：



图 57 LoginServlet流程图

2．RegisterServlet

用于实现用户注册功能，新用户在客户端注册时所填写的个人资料通过RegisterServlet的处理，写入数据库的用户信息表中，并将处理结果返回至客户端，该Servlet的流程图如图58所示。

图 RegisterServlet流程图 图 AddFriendServlet流程图



3．AddFriendServlet

用于完成添加好友模块中，一用户请求添加另一用户时发送请求的功能，流程图如图59所示。

4．AddedServlet

改Servlet用于处理客户端中定时器，定时的不断的发来的请求，用于判断是否有别的用户添加该用户为好友的请求，如果有，则将结果返回，在客户端提示，其流程图如图60所示。





图 AddedServlet流程图 图 AgreeFriendServlet流程图

5．AgreeFriendServlet

当被添加用户通过AddedServlet获得添加者发来的好友添加选择后，在客户端将询问该用户是否同意与对方成为好友，该结果将发送至AgreeFriendServlet进行处理，图61为AgreeFriendServlet的处理流程图。

6．AgreeBackServlet

该Servlet用于完成添加好友消息发出者获取对方是否同意添加结果的功能。在客户端通过定时器定时的不断的向AgreeBackServlet发送请求，查看是否有对方响应自己所发出的好友添加请求的反馈，若有，则返回对方同意/不同意添加的消息。AgreeBackServlet的流程图，如图62所示。





图 AgreeBackServlet流程图 图 DeleteFriendServlet流程图

7．DeleteFriendServlet

该Servlet用于完成删除好友操作，当之行完该功能之后，将对方从自己的好友列表中删除，不能再给对方发送消息，其流程图如图63。

8．ChatGoServlet

用于聊天时，用户选定好友发起聊天后，处理所发送聊天消息。流程图如图64。

9．ChatAlertServlet

当用户处于好友列表界面或与其他好友聊天的界面时，客户端会利用定时器不断的向该Servlet发送请求，查看是否有发送给自己的聊天消息，若有，则返回提示，在客户端显示弹窗，该Servlet的流程图如图65所示。





图 ChatGoServlet流程图 图 ChatAlertServlet流程图

10．ChatRecAlertServlet

用户在客户端好友界面收到有来自于某好友的聊天消息的弹窗，若用户点击“查看”则跳转进入与该好友的聊天界面。该Servlet则是用于处理，用户在点击“查看”并跳转入聊天界面后获取聊天消息的事务，流程图如图66所示。

11．ChatRecServlet

当用户正处于与好友的聊天界面中时，不断向该Servlet发送请求，该Servlet用于将好友回复的消息返回给客户端，流程处理如图67所示。





图 ChatRecAlertServlet流程图 图 ChatRecServlet流程图

12．ChatLogServlet

该Servlet用于完成用户查看与当前好友的聊天记录的功能，通过将客户端传来的用户ID查找数据库中消息收发表，并将结果反馈回客户端，ChatLogServlet的流程图如图68所示。



图 ChatLogServlet流程图

13．SettingServlet

该Servlet用于完成用户配置，包括查看资料、修改密码、修改用户资料、用户注销的功能操作，流程如图69。



图 SettingServlet流程图

## 第二节 客户端的设计与实现

下面将分别从界面设计与功能实现两个方面进行说明。

### 一、UI界面设计实现

本课题实现的基于iPhoneOS的即时通信系统的客户端共包含15个界面，分别用于实现客户端用户进行注册、登录、添加好友、聊天、查看聊天记录、用户配置等功能。在Storyboard中，各个界面之间交互如图70。

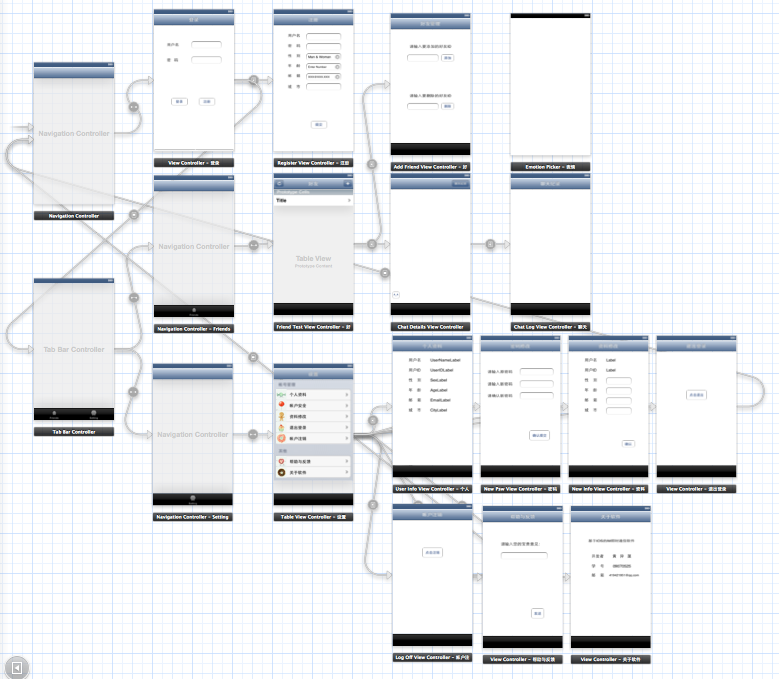


图 Storyboard界面交互图

下面，将本系统客户端UI界面按照“登录、注册界面”，“聊天、好友添加界面”，“用户配置界面”，划分为三块进行设计与实现。

#### （一）登录、注册界面的设计与实现

在登录界面前插入一个导航控制器（NavigationController）[9]，该导航控制器贯穿于用户登录、用户注册两个界面，在用户注册界面中，用户可通过使用导航控制器，返回到上一级——用户登录界面。

登录界面首先加入ViewController视图控制器，并在该ViewController中加入户名和密码标签、用户名和密码的文本输入框，还有“登录”和“注册”的按钮，从登录界面点击“注册”按钮，进入注册界面，在Storyboard中选中“注册”按钮，按住Ctrl键，按住鼠标左键一直拖至注册界面的视图控制器，Action Segue选择push。注册界面由用户名、密码、性别、年龄、邮箱、城市标签和响应的文本输入框，还有“注册”按钮组成。在该界面中，点击左上角的“登录”按钮，可返回登录界面。

#### （二）聊天、好友添加界面的设计与实现

当用户登录之后，将进入基于iPhoneOS的即时通信系统的主页面，在主页面中加入一个Tab Bar Controller[9]，底部设置“Friends”和“Setting”两个图标，分别点击两个图标可显示相应的“好友列表”界面和“用户配置”界面，如图71。



图 主页面Tab Bar控制器

在好友列表界面中，首先加入一个导航控制器，该控制器贯穿于好友列表、好友添加界面、聊天界面、聊天记录界面，页面间Action Segue均选择push，使得在当前页面时都可以返回上级菜单。上述几个页面间交互关系，如图72所示。

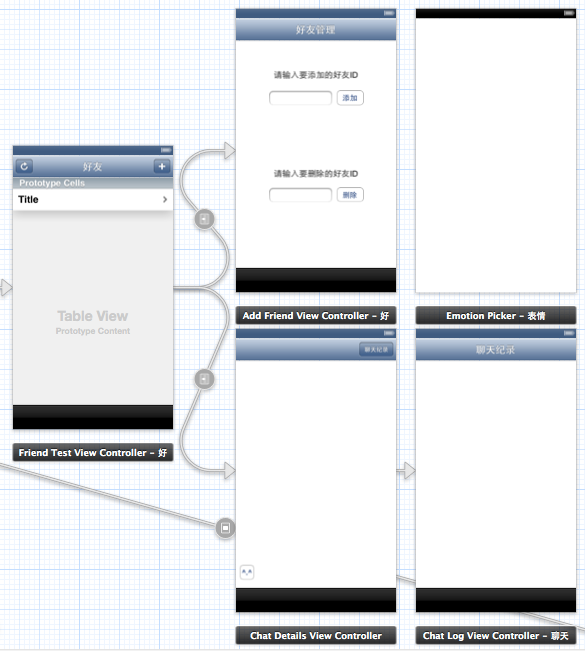


图 好友与消息页面交互图

好友列表界面在ViewController上加入了UITableView，用于分条显示好友名单，其中，每条Cell对应一位好友，点击Cell进入相应的聊天界面，右上角的“添加”按钮，点击后进入好友添加、删除界面。

聊天界面由textView——用于显示聊天内容、textField输入文本框——用于输入聊天消息、UIButton——点击发送消息、和表情按钮——点击进入表情页面组成，右上角的“聊天记录”按钮，点击后可进入查看聊天记录页面查看与当前好友的聊天记录。

聊天记录界面在视图控制器ViewController上加入textView用于显示聊天记录；好友管理（添加、删除好友）采用ViewController控制器[5]，并加入文本输入框、按钮、标签控件实现；表情界面的实现方式与好友列表界面类似，每个表情对应TableView中一个Cell；对于上述所提三个界面在此不再赘述。

#### （三）用户配置界面的设计与实现

用户界面由静态的Table View实现，分为“帐号管理”和“其他”两个section。在“帐号管理”中设置Rows为5，分别为“个人资料”、“帐户安全”、“资料修改”、“退出登录”、“帐户注销”，点击相应的Cell进入相应的操作页面。在“其他”Section中，设置Rows为2，分别为“帮助与反馈”、“关于软件”。

### 二、各界面功能实现

下面将各界面所实现的功能、内部控件说明和请求的Servlet进行汇总，列出表格如表8所示。

表 界面功能表

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 1 | 界面名称 | ViewController |
| 实现功能 | 用户登录，点击“登录”按钮后弹出消息提示框（输入错误/登录成功/登录失败），成功后点击提示框中“确定按钮”进入系统主界面。 | | |
| 控件说明 | 1．用户名、密码标签。  2．用户名、密码文本输入框。  3．“登录”按钮，点击后获取用户名、密码文本输入框内容，向服务器端LoginServlet发送请求。  4．“注册”按钮，点击后进入注册界面。 | | |
| Servlet | http://localhost:8080/IMServlet/LoginServlet | | |
|  | | | |
| 编号 | 2 | 界面名称 | RegisterViewController |
| 实现功能 | 用户注册，用户填写相应信息，点击“注册”后，弹窗提示（输入错误/注册成功/注册失败），注册成功后该用户可返回登录界面进行登录。 | | |
| 控件说明 | 1．用户名、密码、性别、年龄、邮箱、城市标签。  2．用户名、密码、性别、年龄、邮箱、城市文本输入框。  3．“注册”按钮，点击后获取用户名、密码、性别、年龄、邮箱、城市文本输入框内容，向服务器端RegisterServlet发送请求。 | | |
| Servlet | http://localhost:8080/IMServlet/RegisterServlet | | |
|  | | | |
| 编号 | 3 | 界面名称 | FriendViewController |
| 实现功能 | 显示用户好友，用户点击好友名字所在行，可进入相应的好友聊天界面。该界面中设置定时器不断向服务器发送请求，判断是否有好友添加请求、聊天消息，若有则弹出相应弹窗提示。 | | |
| 控件说明 | 1．TableCell，显示好友名单，点击后将该好友用户名作为参数传到与该好友的聊天界面中。  2．“添加”按钮，进入好友管理页面。 | | |
| Servlet | 1．http://localhost:8080/IMServlet/ AgreeBackServlet  2．<http://localhost:8080/IMServlet/ChatAlertServlet>  3．http://localhost:8080/IMServlet/AddDoneServlet | | |

续表 8 界面功能表

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 4 | 界面名称 | AddFriendViewController |
| 实现功能 | 好友管理，输入好友ID可进行添加或删除好友操作。 | | |
| 控件说明 | 1．请输入要添加/要删除的好友ID标签。  2．添加、删除文本输入框。  2．“添加”/“删除”按钮，点击后分别向AddServlet/ DeleteFriendServlet发送请求，对数据库进行操作。 | | |
| Servlet | 1．http://localhost:8080/IMServlet/ AddServlet  2．[http://localhost:8080/IMServlet/DeleteFriendServlet](http://localhost:8080/IMServlet/ChatAlertServlet) | | |
|  | | | |
| 编号 | 5 | 界面名称 | ChatDetailsViewController |
| 实现功能 | 与好友聊天，textView中显示自己和好友间发送的消息。定时器不断向服务器端发送请求，若有来自其他好友的消息，则跳出弹窗提示。 | | |
| 控件说明 | 1．textView按条显示聊天消息。  2．textField文本输入框，用于输入聊天内容。  3．“发送”按钮，点击后获取文本输入框内容，向服务器端ChatGoServlet发送请求。  4．“表情”按钮，点击后进入表情界面。  5．“聊天记录”按钮，点击后进入聊天记录界面。 | | |
| Servlet | 1．<http://localhost:8080/IMServlet/ChatRecAlertServlet>  2．<http://localhost:8080/IMServlet/ChatRecServlet>  3．http://localhost:8080/IMServlet/ChatGoServlet | | |
|  | | | |
| 编号 | 6 | 界面名称 | EmotionPicker |
| 实现功能 | 添加表情 | | |
| 控件说明 | 1．点击相应表情，则在发送文本中添加表情，点击后自动跳回聊天界面。  2．点击“Cancel”按钮，返回聊天界面。 | | |
| Servlet | 无 | | |
|  | | | |
| 编号 | 7 | 界面名称 | ChatLogController |
| 实现功能 | 显示与当前用户的聊天记录，载入该界面View前向服务器端ChatLogServlet发送请求，获得聊天记录。 | | |
| 控件说明 | textView显示聊天记录 | | |
| Servlet | http://localhost:8080/IMServlet/ ChatLogViewServlet | | |
|  | | | |
| 编号 | 8 | 界面名称 | UserInfoViewController |
| 实现功能 | 显示当前用户资料，向SettingServlet发送请求，从数据库获取用户信息。 | | |
| 控件说明 | 1．用户名、ID、性别、年龄、邮箱、城市标签。  2．用于显示所获得的信息的相应标签。 | | |
| Servlet | 1．http://localhost:8080/IMServlet/ SettingServlet | | |

续表 8 界面功能表

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 9 | 界面名称 | NewInfoViewController |
| 实现功能 | 修改资料，用户填写需要修改的个人信息，将新填写的信息发送至SettingServlet，成功后跳出弹窗提示。 | | |
| 控件说明 | 1．用户名、密码、性别、年龄、邮箱、城市标签。  2．用户名、密码、性别、年龄、邮箱、城市文本输入框。  3．“确认”按钮，点击后获取用户名、密码、性别、年龄、邮箱、城市文本输入框内容，向服务器端SettingServlet发送请求。 | | |
| Servlet | http://localhost:8080/IMServlet/SettingServlet | | |
|  | | | |
| 编号 | 10 | 界面名称 | NewPswViewController |
| 实现功能 | 修改密码。 | | |
| 控件说明 | 1．请输入原始密码、请输入新密码、请确认新密码标签。  2．请输入原始密码、请输入新密码、请确认新密码所对应文本输入框。  3．“确认提交”按钮，点击后获取文本输入框内容，向服务器端SettingServlet发送请求。 | | |
| Servlet | http://localhost:8080/IMServlet/SettingServlet | | |
|  | | | |
| 编号 | 11 | 界面名称 | LogOffViewController |
| 实现功能 | 用户注销 | | |
| 控件说明 | “确认注销”按钮，点击后向SettingServlet发送请求，从数据库中清楚所有该用户相关资料。 | | |
| Servlet | http://localhost:8080/IMServlet/ SettingServlet | | |

对于表8中所没有提到的其他界面，通过在Storyboard直接实现，在此不再赘述。

# 基于iPhoneOS的即时通信系统的部署及测试

本章主要介绍基于iPhoneOS的即时通信系统在运行测试前的系统部署以及相关功能的测试要求与结果。

## 第一节 系统部署

由于苹果公司的规定，将系统放于真机上调试须注册开发者帐号并缴纳一定费用，所以本课题所实现的系统客户端的测试只能通过在PC机上的iPhone模拟器来完成。首先，运行Netbeans中的服务器端工程，然后打开Xcode，点击左上角的“Run”键，即可以在模拟器中打开所编写的系统程序。

## 第二节 系统测试

### 一、测试概要

测试将主要针对功能进行相应的检验测试，测试功能概要如表9。

表 测试概要

|  |  |  |
| --- | --- | --- |
| 测试功能 | 输入 | 输出 |
| 用户注册 | 用户名、密码、年龄、性别、邮箱、城市 | 注册成功/失败消息，成功后，可返回登录界面进行登录 |
| 用户登录 | 用户名、密码 | 进入系统主界面（好友列表/用户配置） |
| 选择好友进入聊天 | 选择好友 | 进入与当前好友聊天界面 |
| 发送聊天消息 | 聊天消息 | 对方客户端接收该聊天消息 |
| 好友列表界面接收聊天消息 | 点击查看消息 | 进入与该好友聊天界面，显示好友发出消息 |
| 添加表情 | 表情界面中选择表情 | 成功发送该表情 |
| 即时接收聊天消息 | 对方发送消息 | 显示该条消息 |
| 添加好友 | 要添加的好友ID | 添加成功/失败提示 |
| 删除好友 | 好友ID | 删除成功/失败提示 |
| 查看聊天记录 | 点击“聊天记录”按钮 | 进入聊天记录界面，并显示聊天记录 |
| 查看用户资料 | 用户配置界面中选择“个人资料” | 进入个人资料界面，并显示当前用户资料 |
| 修改密码 | 原密码、新密码、确认新密码 | 修改成功/失败提示 |
| 修改资料 | 新信息（性别、年龄、邮箱、城市） | 修改成功/失败提示 |
| 退出登录 | 用户配置界面中选择“退出登录” | 进入登录界面 |
| 用户注销 | 点击“确认注销” | 注销成功提示，返回登录界面 |

### 二、测试结果

1．用户注册

测试要求：在注册界面输入用户名，密码以“＊”显示，注册成功后，能在数据库用户信息表中加入该用户数据。如用户名已被占用，弹出提示框。

测试结果：该功能经测试可完全实现。原数据库表中总行数为10，如图73，用户Joy注册后，显示注册成功消息，如图74，总行书变为11，如图75。之后再次注册，提示用户名已被使用，如图76。

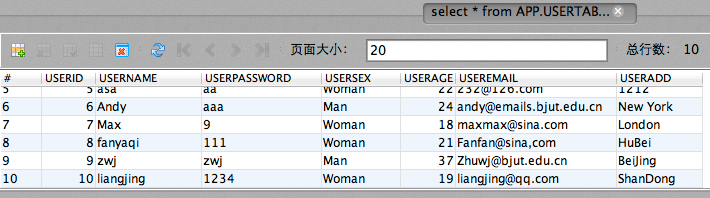


图 注册前数据库



图 注册成功后提示 图 注册失败提示

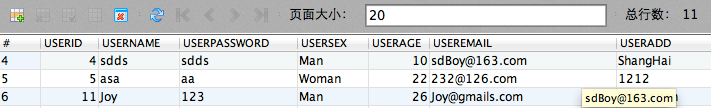


图 注册成功后数据库信息

2．用户登录

测试要求：在登录界面输入用户名，密码以“＊”显示，登录成功后，弹窗提示，点击确定，跳转进入主界面；若信息没填全，则弹窗提示；输入错误，弹窗提示。

测试结果：该功能经测试可完全实现。当用户填写用户名“each”和密码“123”后，点击登录，显示登录成功信息，如图77，之后显示好友列表界面，如图78。若密码输入错误则提示登录失败信息，如图79。若用户名或密码未填写，则提示输入错误，如图80。

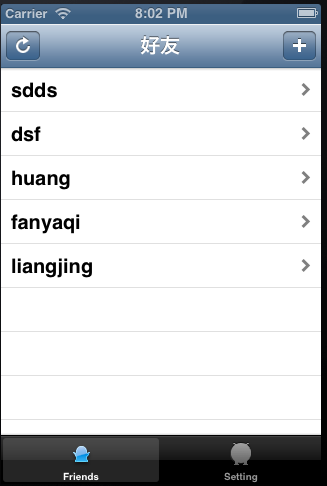


图 登录成功 图 点击OK 进入主界面

图 用户名或密码错误 图 未填写用户名/密码

3．选择好友进入聊天

测试要求：从好友列表中任选一位好友，点击进入聊天界面

测试结果：该功能经测试可完全实现。在列表中选择好友“huang”，如图81，之后进入与好友huang的聊天界面，如图82。

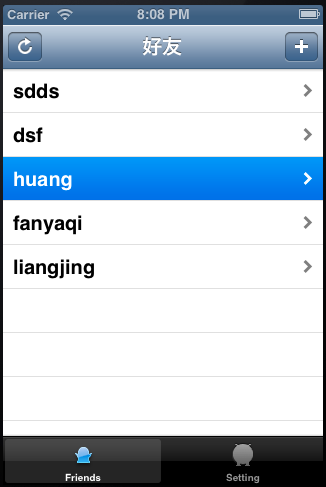
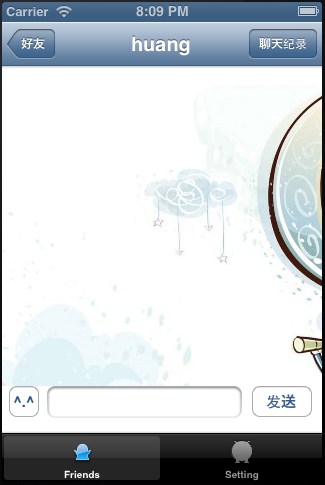
 

图 选择好友 图 进入聊天界面

4．发送聊天消息

测试要求：在文本框中输入聊天内容，点击发送，在界面中显示时间、发送者、发送内容，数据库中存储改消息、对方可接收到改消息。

测试结果：该功能经测试可完全实现。在文本框中输入“Hello”，点击发送，消息显示在界面上，如图84。发送后，消息发送者、消息接收者、消息发送时间、聊天消息写入数据库中，如图83。

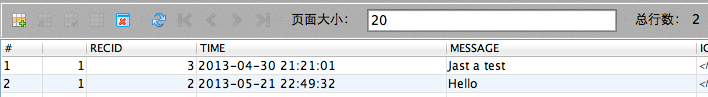


图 发送消息后数据库信息

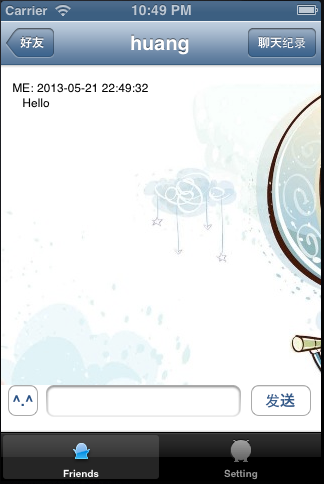


图 发送聊天消息

5．好友列表界面接收聊天消息

测试要求：在好友列表界面中，出现消息提示，点击查看，进入聊天界面并显示该好友发来信息。

测试结果：该功能经测试可完全实现。用“huang”账号登录系统，登录后在好友列表界面收到消息提醒，如图85。点击查看后，进入与“each”的聊天界面，如图86。

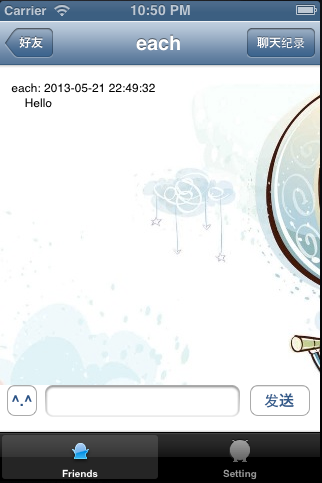
 

图 弹出消息提示 图 查看消息

6．添加表情

测试要求：点击表情按钮进入表情界面，选择后在文本输入框中显示表情转换符，发送后，在textView中显示该表情图表。

测试结果：该功能经测试可完全实现。在聊天界面中，输入聊天内容，插入表情时，点击表情按钮，进入表情列表，如图87。任意点击一个表情（第七列），之后跳转回聊天界面，点击发送后，表情在textView中显示，如图88。发送后，该消息存入数据库，表情以特殊字符存取，数据库信息如图89所示。

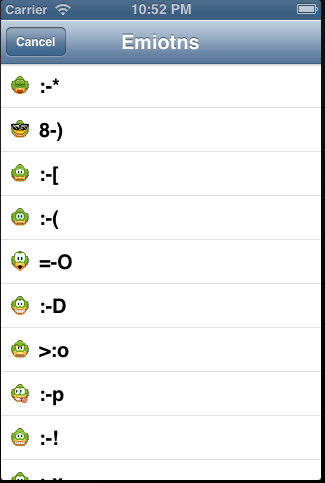
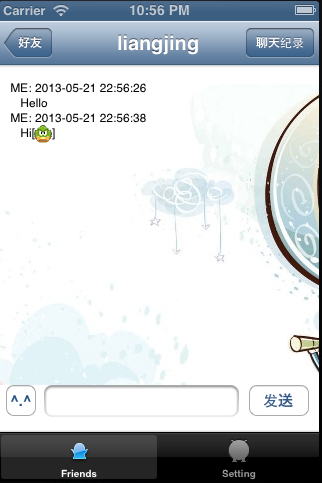
 

图 表情列表 图 发送表情



图 数据库信息

7．即时接收聊天消息

测试要求：在处于聊天界面时，对方发送的消息，可直接在textView中显示。由于硬件限制，无法实现两个用户同时在线，故添加用户本身为好友，进行测试。

测试结果：该功能经测试可完全实现。用户“each”登录后，选择好友“each”（自身）进行聊天，输入聊天消息“Have a good time”、“Thanks”，此时消息即时传送并显示在对方聊天界面上，如图90。添加表情发送后，对方也可以即时接收，如图91，用户huang，收到来自好友each的表情消息。消息发送后，相关信息被存于数据库中，如图92。

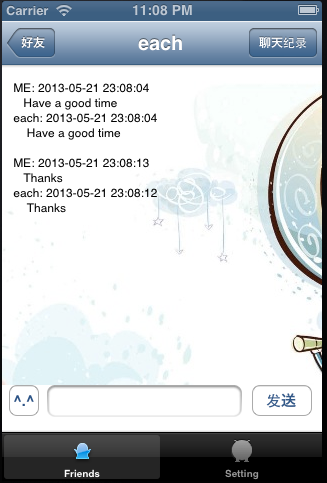
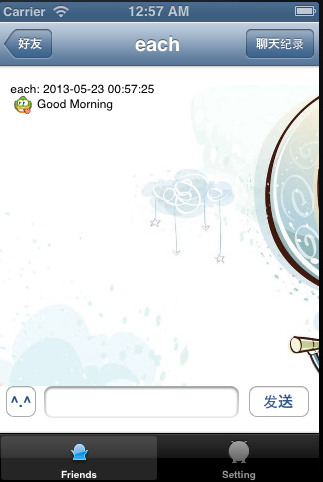
 

图 即时接收聊天消息 图 接收表情消息



图 数据库信息

8．添加好友

测试要求：输入对方ID，点击添加后，对方收到提示，同意后即确立好友关系，在列表中新增对方为好友；若对方不同意，则好友列表不变。需输入正确ID，否则提示。若对方已成为该用户好友，则提示不可再次添加。

测试结果：该功能经测试可完全实现。在添加好友文本框中输入ID“11”，提示发送成功，如图93。在用户Joy（ID：11）端提示好友添加消息，如图94。点击同意后，在

用户each的好友列表中，新添了好友Joy，如图95。若输入ID为12（无此用户），则提示发送失败，如图96。ID为2的用户之前与each已经成为好友，此时再次添加2号用户，则提示添加失败，如图97。添加时，好友请求表中，写入each（ID：1）与Joy（ID：11）数据信息，如图98。添加成功后，好友管理表，写入两条好友数据信息（each-Joy，Joy-each），如图99。

图 发送添加请求 图 收到好友添加请求

图 对方同意后好友列表 图 ID输入错误



图 好友无法再次添加



图 添加后数据库信息（好友请求表）



图 添加后数据库信息（好友管理表）

9．删除好友

测试要求：输入要删除的好友ID，若对方不是该用户好友，提示输入错误，若输入正确，从数据库中删除该好友关系，再次登录时，好友列表中不出现该好友。

测试结果：该功能经测试可完全实现。输入ID为13（无此用户），提示删除失败，如图100。输入ID为11，提示删除成功，如图101。删除前，数据库好友管理表中，含两条好友关系数据（#14、#15），如图102。删除后，#15被删除，如图103。

图 ID输入错误 图 删除成功

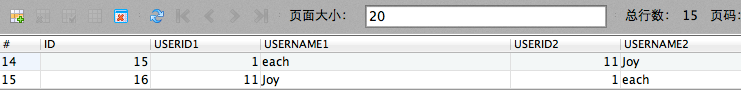


图 删除前，数据库信息

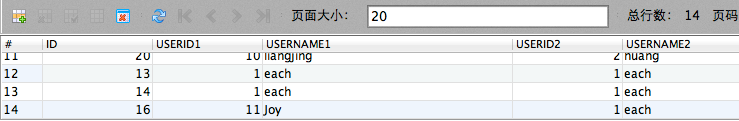


图 删除后（单向删除）数据库信息

10．查看聊天记录

测试要求：点击“聊天纪录”，显示与该好友聊天纪录。

测试结果：该功能经测试可完全实现，如图104，从用户each与huang的聊天界面中点击进入聊天记录界面后，能正常显示之前所有的聊天消息。

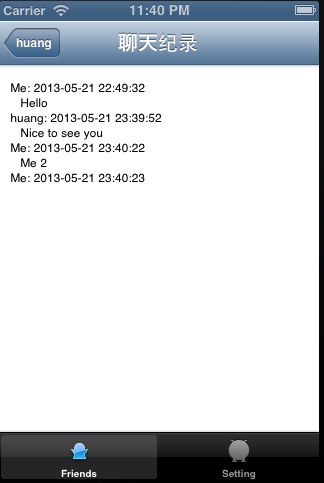


图 聊天记录

11．查看用户资料

测试要求：从用户配置界面，点击“个人资料”，可查看用户资料。

测试结果：该功能经测试可完全实现。点击屏幕右下方Setting，进入设置界面，如图105。点击个人资料后，进入“个人资料”界面，并显示当前用户资料，如图106。

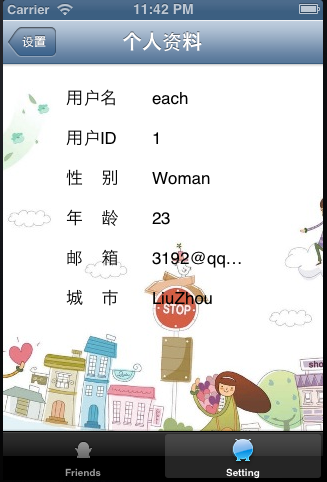
 

图 用户配置界面 图 个人资料显示

12．修改密码

测试要求：用户输入正确的原密码，和新密码，并再次确认新密码。

测试结果：该功能经测试可完全实现。若原密码输入错误（原密码为“123”，测试时输入“1”），提示修改失败，如图107；若新密码两次输入不一致（请输入密码处填“1”，确认处填“111”），提示输入错误，如图108；若均无误（原密码填写“123”，新密码填写“111”），提示修改成功信息，如图109。数据库信息由修改前，图110所示，密码为“123”，改为如图111所示的“111”。

图 原密码输入错误 图 两次输入不一致



图 密码修改成功



图 修改前数据库信息



图 修改后数据库信息

13．修改资料

测试要求：用户填写新的资料，若没修改或未填写的项仍保留原信息，修改成功后弹窗提示成功信息。

测试结果：该功能经测试可完全实现。修改前数据库信息如图112。将年龄改为“20”，邮箱改为“each@126.com”，城市文本框不填写，如图113，提交后，提示修改成功信息，如图114。修改后，数据库表中UserAge变为20，UserEmail变为each@126.com，UserAdd仍为LiuZhou，如图115。



图 修改前数据库信息

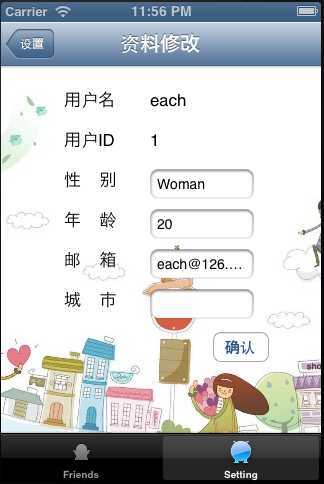
 

图 资料修改并提交 图 资料修改成功提示



图 修改后数据库信息

14．退出登录

测试要求：点击按钮“点击退出”后，跳转至主页面。

测试结果：该功能经测试可完全实现，如图116，点击按钮“点击退出”后，页面跳转如图117。

**** ****

图 用户退出 图 跳转至登录页面

15．用户注销

测试要求：点击注销后，弹出注销提醒，确认后从数据库好友管理表中删除有关信息，从用户信息表中删除该用户资料（以用户Joy测试）。注销成功后，弹窗提示并跳转入登录界面。

测试结果：该功能经测试可完全实现。注册前，数据库用户信息表中，总行数位11，#6为Joy信息，如图118；好友请求表总行数为5，#4中RecID为11，如图119；好友管理表总行数为15，#14、#15为Joy与each的好友关系信息，如图120。点击注销后弹出确认提示，如图121，确认后页面跳转如图122。注销后数据库用户信息表中，总行数位10， Joy信息被删除，如图123；好友请求表总行数为4，原#4被删除，如图124；好友管理表总行数为13，#14、#15被删除，如图125。

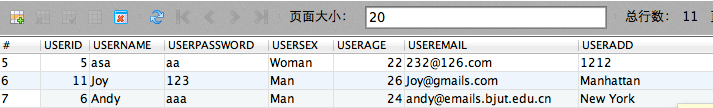


图 注销前用户信息表

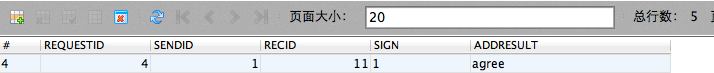


图 注销前好友请求表

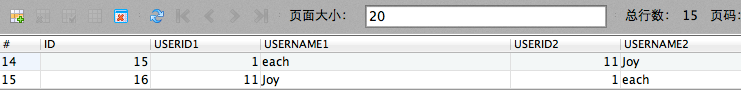


图 注销前好友管理表

图 注销提醒 图 注销成功

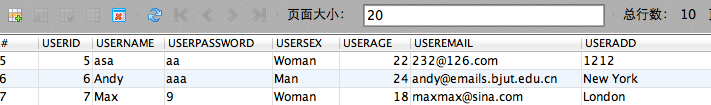


图 注销后用户信息表



图 注销后好友请求表

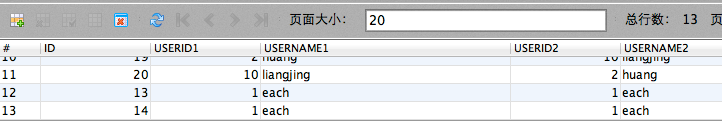


图 注销后好友管理表

# 结束语

随着移动通信设备的快速发展，人与人之间的通信方式不再局限于邮件、电话和短信，由于移动即时通信所能传递的信息更加多元化，使用更为方便、省时，通过移动设备中的即时通信系统进行即时消息的传递将变得越来越普遍。为了能够更好的满足现代人的需求，适应社会的发展，方便人们进行沟通交流，在此对基于iPhoneOS的即时通信系统进行了设计与实现。

本系统在开发过程中以软件工程理论为指导，采用软件工程的开发模式，使用UML进行建模，使分析文档和设计文档都具有良好的可读性和直观性，并对后期扩展进行二次开发具有极大的指导作用。

本文对基于iPhoneOS的即时通信系统进行了相关背景分析、需求分析、整体结构分析、设计与实现。用户可在iPhone客户端进行注册并登录本系统，在系统中添加其他用户为好友，实时的向好友发送聊天消息和接收聊天消息，实现即时通信，另外用户还能在客户端用户配置界面中设置、修改个人资料与密码。所有从客户端输入的信息资料，均可通过服务器端Servlet写入数据库中，实现永久化存储。系统的全部功能现已基本实现，并分别针对每个模块及功能进行了详细的测试和分析，测试结果可基本满足用户需求。

## 不足

由于开发时间和个人能力所限，本课题所研究的基于iPhonOS平台的即时通信系统仍存在许多需要改进的地方。在界面设计方面，客户端界面的设计风格和友好程度有待提高；在功能实现方面，系统仍具有一定限制，如不能实现图片、文件的传输等。

## 展望

即时通信系统实效、方便的特点决定了其在未来移动商务、在线协作及Interact应用中将扮演更为重要的角色。而随着移动通信技术的迅猛发展，中国3G网络投入商业试运营，为移动即时通信提供了更强有力的支撑平台，实时多媒体技术也会移植到移动即时通信应用中来。集成多媒体的应用，在现有文本消息、表情图标传送的基础上加入语音、文件、照片传送，将能为用户提供更多个性化的服务，具有更强的吸引力。

希望今后能修改完善系统中的不足，在现有基础上添加更多实用性、创新性功能，为扩展开发人员思路和移动即时通信的研究做出一点微小的贡献。

# 致谢

本次毕业论文设计，从前期的准备到系统的研发再到最后的论文撰写，历时4个多月，这期间凝聚了很多人的帮助与关心。值此论文完成之际，谨向本科毕业论文工作进行期间所有给过我无私帮助的老师、同学、家人和朋友表示衷心的感谢。

首先，最应该感谢的是我的指导老师朱文军老师，毕业设计的系统开发和论文撰写工作是在朱老师的悉心指导下完成的。在系统的开发阶段，朱老师曾给予我多次的帮助，对于难以攻克的技术难点，朱老师所提出宝贵意见，使我深受启发。当我在开发过程屡次失败，遭受打击时，是朱老师的鼓励与支持让我从挫败中走出，重拾信心与希望。在论文撰写阶段，朱老师耐心地对我的论文进行批改、审阅并提出修改意见，让我的论文质量有了更进一步的提高。此外，朱老师渊博的学识，敏捷的思维，创新执着的科研作风以及踏实严谨的治学态度，为我树立了求学以及做人的楷模。在此，谨向朱老师在这一学期以来在我学习生活上的关怀和指导表示最诚挚的敬意！

其次，我要感谢所有在我的课题进行过程中陪伴我一起奋斗的同学们，他们是敖菁、李亚博、李炫等，他们不仅在软件开发的技术上给予我意见与帮助，毕业论文设计期间，他们在生活中给予我的帮助与支持也格外令我感动！从他们身上，我学到了很多宝贵的东西。

我还要感谢所有无私帮助过我的不知名的陌生人，比如图书馆里帮助查找文献的老师，iOS开发论坛里对于我所遇到的技术难点提出解决方案的程序开发者，相关文献资料的研究撰写者等等，在此，我向他们的无私精神表示崇高的敬意！

最后，我想感谢我的家人，感谢他们在生活方面予以的关心与支持，虽然远离家乡求学，但是他们的爱却使我每一步都走的更加坚定并充满希望与力量！

# 参考文献

1. （美） Carlo Chung 著,刘威 译. Objective-C编程之道：iOS设计模式解析.北京：人民邮电出版社,2011.11
2. DevDiv移动开发社区. iOS开发实战体验. 北京：海洋出版社,2012.8
3. 李刚. 疯狂 Java 讲义[M]. 北京：电子工业出版社,2008
4. 张照. iPhone开发入门很简单. 北京：清华大学出版社,2013.2
5. Mark D, Nutting J, LaMarche J. Beginning iOS 5 Development. Apress, 2011.705-710.
6. (美)Jocelyn Harrington,Brandon Trbitowski,Christopher Allen,Shannon Appelcline 著,袁国忠 译. iOS 4实战：iPhone与iPad应用开发. 北京：人民邮电出版社,2012.4
7. 王乃奇. GeekHead Xcode.http://www.geekhead.org，2013.4
8. Alex Rozanski. How do I use NSTimer.http://stackoverflow.com/questions/1449035/how-do-i-use-nstimer. 2012.9
9. Zdziarski J A. Iphone open application development. O'Reilly, 2008