
软件工程作业管理系统 概要设计

	人员	日期
拟制	刑宇 郑非 钱劲翔	yyyy-mm-dd
评审人	刑宇	yyyy-mm-dd
批准	郑非	yyyy-mm-dd
签发	钱劲翔	yyyy-mm-dd

第 1 章 引言

1.1 编写目的

在本项目的前一阶段，也就是需求分析阶段，已经将系统用户对本系统的需求做了详细的阐述，这些用户需求已经在上一阶段中对不同用户所提出的不同功能，实现的各种效果做了调研工作，并在需求规格说明书中得到详尽得叙述及阐明。

本阶段已在系统的需求分析的基础上，对即时聊天工具做概要设计。主要解决了实现该系统需求的程序模块设计问题。包括如何把该系统划分成若干个模块、决定各个模块之间的接口、模块之间传递的信息，以及数据结构、模块结构的设计等。在以下的概要设计报告中将对在本阶段中对系统所做的所有概要设计进行详细的说明，在设计过程中起到了提纲挈领的作用。

在下一阶段的详细设计中，程序设计员可参考此概要设计报告，在概要设计即时聊天工具所做的模块结构设计的基础上，对系统进行详细设计。在以后的软件测试以及软件维护阶段也可参考此说明书，以便于了解在概要设计过程中所完成的各模块设计结构，或在修改时找出在本阶段设计的不足或错误。

本文档主要由开发人员撰写，意在指明即时通讯软件系统（以下简称通讯系统）的具体开发明细，记录下通信软件。本文档还可作为软件开发过程中的备案，为后续需求的提出给出参考。此外，非软件开发者可以将这个文件视为我们的即时通讯系统的开发纲要

1.2 项目背景

随着网络通信和信息技术的发展，现代社会人与人之间的交流越来越重要，同时也越来越便捷，并且出现了如 Facebook、QQ 等著名社交软件。于是我们组决定立足于当下标准，设计一款同样有着及时，便捷，潮流的通讯软件。

1.3 术语

[列出本文档中所用到的专门术语的定义和外文缩写的原词组]

表 1.1 术语表

缩写、术语	解释
c	d

第 2 章 任务概述

本系统的目标是实现一个即时通信系统，包括客户端、服务器端两个部分。

客户端面向 Android 或 windows 用户，为用户提供即时的文字、语言聊天，群聊等基本功能，考虑实现各种高级功能。

2.1 目标

2.1.1 总体目标

实现实时通信系统的基本功能，实现需求规格说明书中所描述的功能需求、性能需求，以及实现实时聊天功能和群聊功能，并且保证系统的健壮性和数据安全。

2.1.2 高级目标

提供稳定的视频聊天功能。

同时还能支持动态分享，皮肤自定义等个性化功能。

2.2 开发与运行环境

2.2.1 开发环境的配置

2.2.2 测试环境的配置

2.2.3 运行环境的配置

2.3 需求概述

功能需求包括：通讯模式

- 用户一对一通讯

- 群聊

- 随机匹配聊天

- 通讯方式

- 文字即时通讯

- 图片、视频、音频、链接、文件等多种媒体的发送

表 2.1 开发环境的配置

类别	标准配置	最低配置
计算机硬件	基于 x86 结构的 CPU 主频 >=2.4GHz 内存 >=8G 硬盘 >=200G	基于 x86 结构的 CPU 主频 >=1.6GHz 内存 >=512M 硬盘 >=2G
计算机软件	windows .net 版本 >4.0)	windows .net 版本 >4.0)
网络通信	至少要有一块可用网卡 能运行 IP 协议栈即可	至少要有一块可用网卡 能运行 IP 协议栈即可
其他	采用 MySQL 数据库	采用 MySQL 数据库

表 2.2 测试环境的配置

类别	标准配置	最低配置
计算机硬件	基于 x86 结构的 CPU 主频 >=2.4GHz 内存 >=8G 硬盘 >=200G	基于 x86 结构的 CPU 主频 >=1.6GHz 内存 >=512M 硬盘 >=2G
计算机软件	Linux (kernel version>=4.10) GNU gcc (version>=6.3.1)	Linux (kernel version>=3.10) GNU gcc (version>=5.4)
网络通信	至少要有一块可用网卡 能运行 IP 协议栈即可	至少要有一块可用网卡 能运行 IP 协议栈即可
其他	采用 MySQL 数据库	采用 MySQL 数据库

表 2.3 运行环境的配置

类别	标准配置	最低配置
计算机硬件	基于 x86 结构的 CPU 主频 $\geq 2.4\text{GHz}$ 内存 $\geq 8\text{G}$ 硬盘 $\geq 200\text{G}$	基于 x86 结构的 CPU 主频 $\geq 1.6\text{GHz}$ 内存 $\geq 512\text{M}$ 硬盘 $\geq 2\text{G}$
计算机软件	Linux (kernel version ≥ 4.10) GNU gcc (version $\geq 6.3.1$)	Linux (kernel version ≥ 3.10) GNU gcc (version ≥ 5.4)
网络通信	至少要有一块可用网卡 能运行 IP 协议栈即可	至少要有一块可用网卡 能运行 IP 协议栈即可
其他	采用 MySQL 数据库	采用 MySQL 数据库

- 语音、视频及时通讯
- 加密文字及时通讯
- 用户控制
- 注册/注销账号
- 添加/删除联系人
- 加入/退出/创建群聊

2.4 条件与限制

2.4.1 硬件约束

1. Android 平台：软件本身大小 $< 100\text{MB}$ ；本地聊天记录文件不大于 1G ，并且有着自动清除过时的聊天记录功能；文本通信延迟 $< 100\text{ms}$ ；内存占用 $< 100\text{MB}$ ；流量消耗不超过通信本身数据量的 1.1 倍；电量消耗不超过机体待机时的两倍
2. Windoes 平台：约束较少，主要考虑的磁盘和内存约束与 Android 一致

2.4.2 技术限制

1. 考虑到服务器网络带宽问题，视频聊天的清晰度最高维持在 360P ；
2. 由于技术上的限制，除去文本类的通信记录都不会被保存；
3. 用户无法在不同的客户端上共享聊天记录；
4. 用户无法在周日凌晨 2:00 至 4:00 使用通讯软件的聊天功能因为这个时

候应该是维护服务器的时候；

5. 由于服务器磁盘大小的限制，用户传输文件的大小有着 <8MB 的限制

第 3 章 总体设计

3.1 软件描述

系统包括前台和后台两个部分。

前台主要功能是：给用户提供交互的接口，用于各种功能。后台主要功能是：处理用户发出的请求，后台记录用户产生的数据等等。

3.2 处理流程

3.2.1 总体流程

3.2.2 系统基本流程

此处应当有一个图和对应的描述。

3.2.3 客户端基本流程

3.2.4 服务器端基本流程

3.2.5 功能 1 具体流程

用户注册或登录

在打开应用的第一个页面，会有用户名和密码的输入框，在正确输入用户名和密码之后点击 sign in 便可以登入。或者可以点击另外一个 sign up 的按钮，创建一个用户名没有重复的账户之后输入验证码便会通过，服务器数据更新之后便可以使用这个新的账户登录。

3.2.6 功能 2 具体流程

单人聊天

已登录用户会进入一个新的功能页面（称为主页面）。在主页面首先在消息列表中选择已有对话，或者在联系人列表中选择联系人新建对话。

3.2.7 功能 3 具体流程

大厅聊天

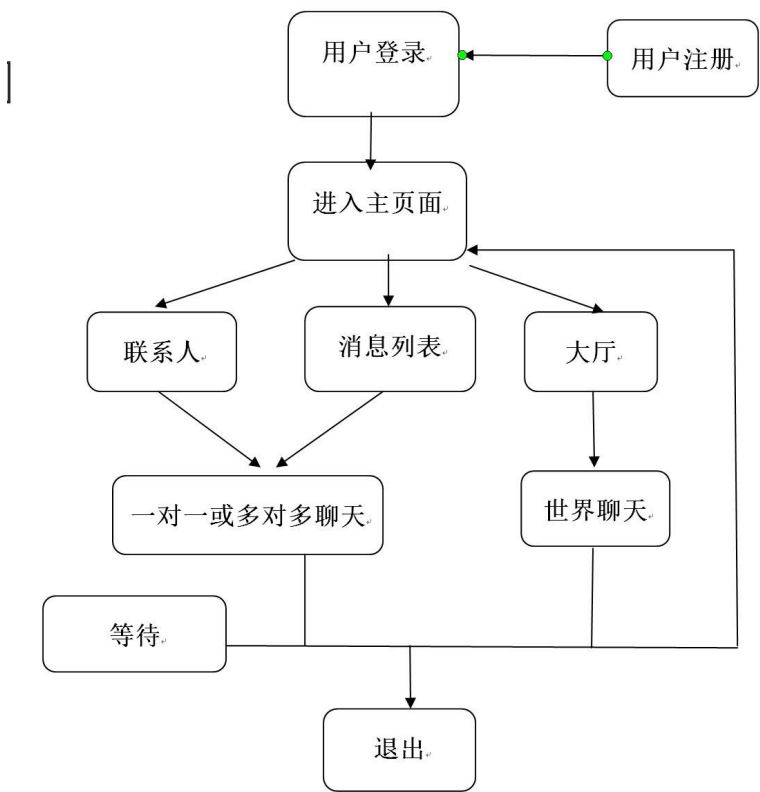


图 3.1 客户端流程

已登录用户在主页面选择大厅聊天，输入任何自己想说的并发送即可。要注意的是，在大厅说的所有话都会被所有在线用户看到。

3.3 功能结构设计

3.3.1 整体结构

此处应当有一个图和对应的描述。系统如果像微内核那样，划分成核心模块和若干个子系统，此处应当有图示及说明，然后后续几个节应当描述这几个子系统。如果系统像宏内核，那应当说明有哪些紧密联系的模块，并在后续几个节内描述这些模块。

3.3.2 用户端结构

此处应当有一个图和对应的描述。这只是举个例子。可能的内容包括用户端的具体模块、耦合情况等。

3.3.3 服务器端结构

此处应当有一个图和对应的描述。这只是举个例子。

3.3.4 后台数据库维护模块结构

此处应当有一个图和对应的描述。这只是举个例子。

3.4 功能需求与程序代码的关系

[此处指的是不同的需求分配到哪些模块去实现。可按不同的端拆分此表]

表 3.1 功能需求与程序代码的关系表

•	模块 1	模块 2	模块 3
需求 1	•	Y	•
需求 2	•	Y	•
需求 3	•	Y	•
需求 4	Y	•	•
需求 5	•	•	Y

注：各项功能需求的实现与各个程序模块的分配关系

第 4 章 接口设计

4.1 外部接口

比如说需要用到支付宝等外部支付系统，接口应当如何封装。

4.1.1 支付宝接口

详细讲述不同的接口（查询状态、支付交易、获取回执等）

4.2 内部接口

客户端

- void App.SignIn(String username, String password)
客户端登录
- void App.SignUp(String username, String password)
客户端注册
- void App.CreateLink(String username)
客户端添加联系人
- void App.DeleteLink(String username)
客户端删除联系人
- void App.Send(String username, String message)
客户端发送信息给指定联系人
- void App.SendToRoom(String message)
客户端发送信息给聊天室
- void App.Receive(String username, String message)
客户端获取某联系人发来的信息
- void App.ReceiveRoomMessage(String message)
客户端获取聊天室发来的信息

- void Packet.ToBytes()
把数据包转换成字节流
- void Packet.FromBytes()
把字节流转换成数据包
- void ConnectionManager.Send(Packet packet)
发送数据包
- Packet ConnectionManager.ReceiveNext()
接收下一个数据包

服务器端

- ChatServer.Receive() return packet
服务器获取数据包
- ChatServer.ProcessMessage()
服务器处理数据包
- ChatServer.Broadcast()
服务器广播数据包
- ChatServer.Disconnect()
服务器断开连接
- DBManager.SignUp(user, salt, MD5) return status
数据库模块把注册信息发送给数据库
- DBManager.SignIn(user, salt, MD5) return status, contacters
数据库模块把登录信息发送给数据库
- DBManager.GetContacterList(user) return contacters
数据库模块获取联系人列表
- DBManager.AddLink(user1,user2) return info
数据库模块某用户添加联系人
- DBManager.DeleteLink(user1,user2) return info
数据库模块某用户删除联系人

- DBManager.FetchUnreceivedMsg(user) return messages
数据库模块获取离线消息

第 5 章 数据结构设计

5.1 逻辑结构设计

5.1.1 用户管理系统数据结构设计

用户信息

- 用户名 varchar(48)
- 密码盐 char(10)
- MD5 值 char(32)

用户关系信息

- 用户名 1 varchar(48)
- 用户名 2 varchar(48)

离线消息信息

- 发送用户 varchar(48)
- 接收用户 varchar(48)
- 信息 varchar(1024)
- 发送时间 datetime

5.1.2 客户端数据结构

数据包

- 消息类型 MessageType
- 字符串列表 List<String>

数据包序列化结构

- 消息类型 byte[1]

- 消息长度 byte[2]
- 字符串 1 byte[n1]
- 分隔符 byte[1] 0x00
- 字符串 2 byte[n2]
- 分隔符 byte[1] 0x00
- 字符串 3 byte[n2]
- ...
- 字符串 m byte[nm]

联系人列表

- 列表 List<String>

未收消息列表

- 列表 Map<String,List<Packet>> Packet: 数据包结构

表示一个值是数据包列表的字典，键是对应的联系人。

5.1.3 服务器端数据结构

同客户端，并且加上以下：

连接列表 List<Socket>

连接字典 Map<username,Socket>

IP 字典 Map<IP,Socket>

连接字典反向 Map<Socket,username>

5.2 物理结构设计

数据包都转换成字节数组。

5.3 数据结构与程序模块的关系

[此处指的是不同的数据结构分配到哪些模块去实现。可按不同的端拆分此表]

表 5.1 数据结构与程序代码的关系表

•	TCP 接收程序	数据包解码程序	客户端主程序	服务器端主程序
数据包字节序列	Y	.	.	.
数据包	•	Y	Y	Y
联系人列表	•	.	Y	Y
离线信息列表	•	.	Y	Y

注：各项数据结构的实现与各个程序模块的分配关系

第 6 章 数据库设计

6.1 数据库环境说明

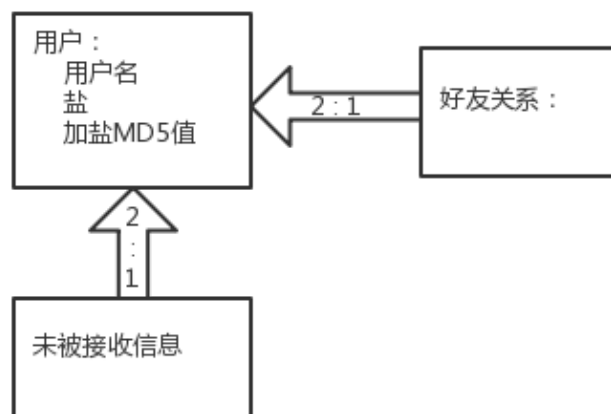
本聊天软件的服务器在云服务器上运行，操作系统是 Ubuntu16.04，所以本系统的数据系统采用 MySQL 数据库系统。

6.2 数据库的命名规则

不允许单词缩写，表名是复数形式。

6.3 逻辑设计

本数据库的 users 和 links 表能满足 BCNF 范式，但是 unreceived_message 表并不设主键，仅仅满足第一范式。



6.4 物理设计

6.4.1 数据库产品

使用 MySQL 数据库。根据目前本软件的使用情况，一台云服务器已经可以很好地满足所有需求，所以没有采用分布式数据库。

6.4.2 实体属性、类型、精度

6.4.2.1 客户数据表设计

表 6.1 用户数据表 Users 设计

字段名	类型	大小	说明	备注
uname	varchar	48	用户名	主键
md5	char	10	随机产生的加盐值	•
pw	char	32	用户的登录密码加盐的 MD5 值	•

注：用户数据表 Users 设计

6.4.2.2 联系人数据表设计

表 6.2 联系人表 Links 设计

字段名	类型	大小	说明	备注
uname_1	varchar	48	用户 1	主键，外键，引用 Users
uname_2	varchar	48	用户 2	主键，外键，引用 Users

注：联系人表 Links 设计

6.5 安全性设计

服务器调用数据库全部采用存储过程，不允许直接执行数据库语句，一般不会产生安全性问题。

6.6 数据库管理与维护说明

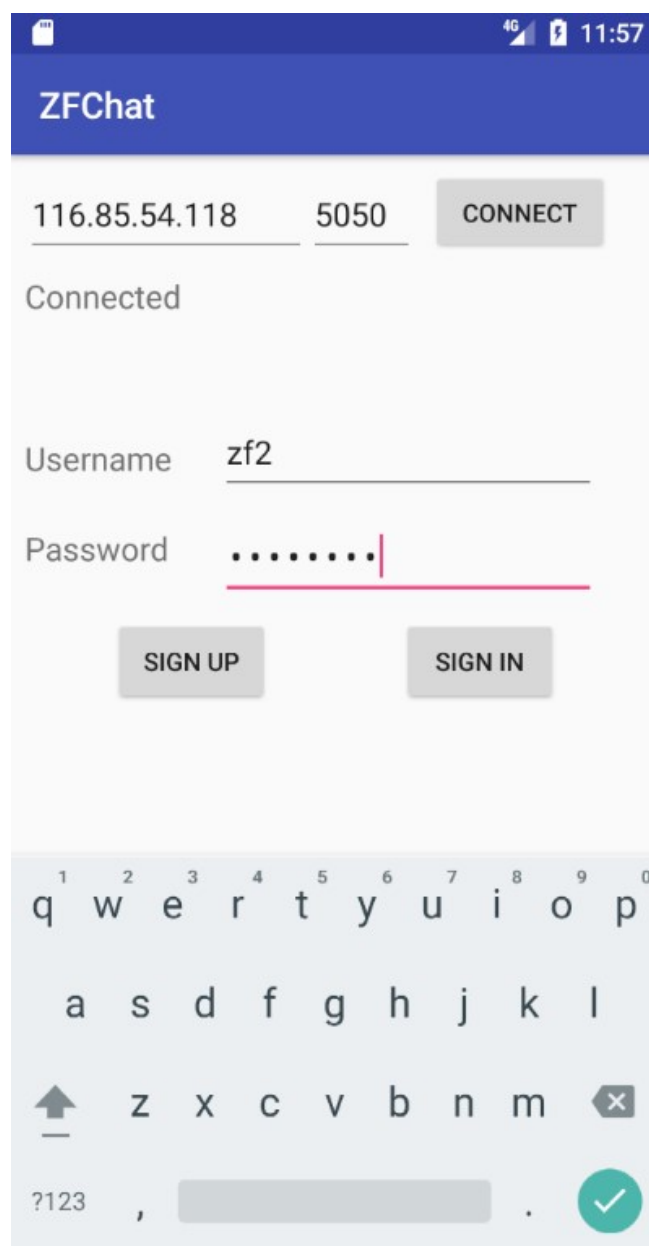
对于数据库的维护，随时对数据库中的信息加以调试和保存备份。同样需要个工作人员进行系统的分析和用户的反馈，对系统进行升级以及功能的完善。同时保证系统安全有序的运行。

第 7 章 界面设计

7.1 服务器端界面

服务器是命令行的，没有可见界面。

7.2 登录界面

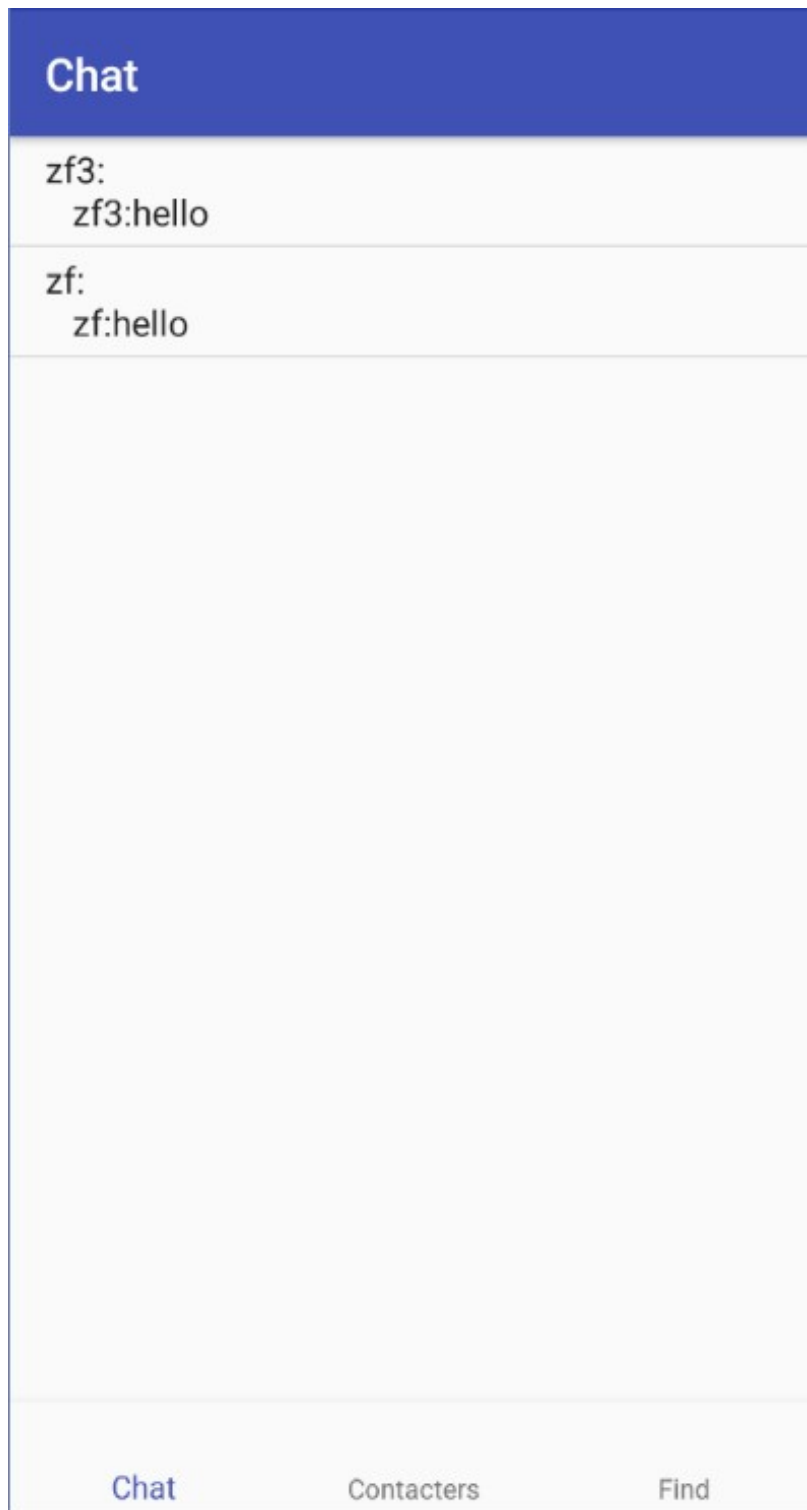


The screenshot displays the ZFChat mobile application interface. At the top, a blue header bar contains the app name "ZFChat". Below this, there are input fields for an IP address (116.85.54.118) and a port number (5050), followed by a "CONNECT" button. The status "Connected" is shown below these fields. Further down, there are input fields for "Username" (zf2) and "Password" (masked with dots). Below the password field, there are two buttons: "SIGN UP" and "SIGN IN". At the bottom of the screen, a standard QWERTY keyboard is visible, indicating the app is running on an Android device. The status bar at the very top shows a 4G signal, battery level, and the time 11:57.

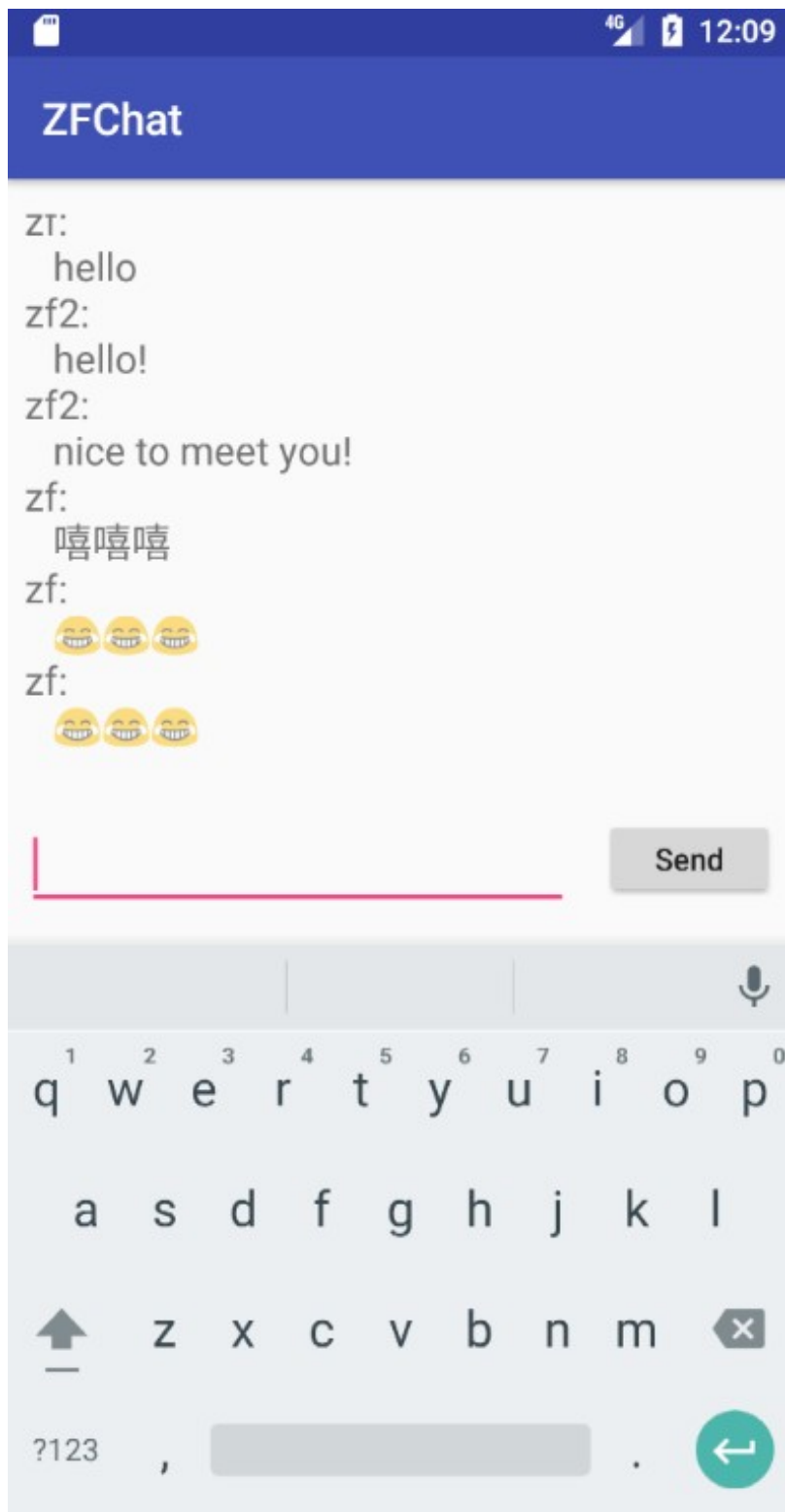
7.3 联系人界面



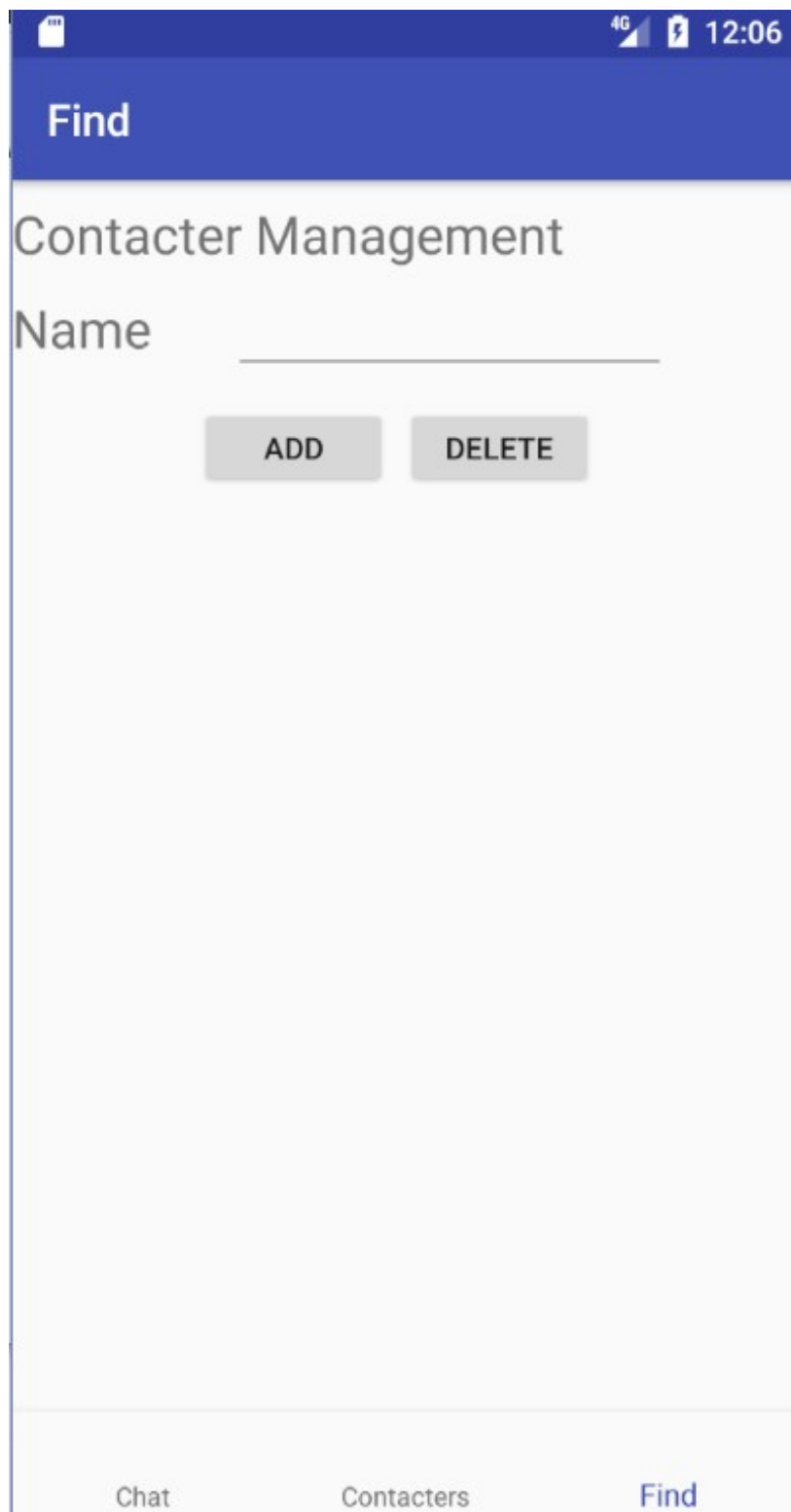
7.4 消息界面



7.5 聊天界面



7.6 好友管理界面



第 8 章 出错处理设计

8.1 数据库出错处理

««««< HEAD

8.1.1 事务故障

如果出现事务故障，则遵循数据库系统设计的一般原则，找出并将故障事务从数据库中撤销。

8.1.2 系统崩溃故障

对于类似于停电，死机等系统崩溃故障，由于我们的数据库设计每次操作都会生成日志以及每周都会人工维护，故会尽可能恢复到故障前的样子。

8.1.3 磁盘故障

由于开发者资金紧缺，人手不足，如果出现了服务器上的磁盘故障，我们会在保障还能使用基本的聊天功能的情况下尝试恢复用户数据，但是完全丢失的数据无法由我们找回，只能由使用者将其重新恢复（如重新注册和原来一样的账号，重新添加原来的联系人）。

8.1.4 不可抗力

如果发生诸如地震火灾等不可抗力，我们只能十分遗憾地选择删库跑路。

8.2 某模块失效处理

大多数情况下我们都会提供聊天的功能，除非我们处于维护状态或出现了不可抗力，否则不会停止服务。

如果某版本的客户端出现 bug，欢迎访问开发者论坛并提出，开发者会以最快的速度修复这个 bug。=====

8.3 某模块失效处理

如果服务器出现故障,重启一次。>>>>>>> 29d41c134fbb1596db211db949e2cd5634007bb

第 9 章 安全保密设计

用户密码采用加盐 MD5 方法存储

对于每个用户都随机生成盐，然后拼接到用户密码上，然后通过 MD5 加密存储到数据库。这一过程在服务器上运行。

登陆的时候客户端发送用户名和密码到服务器，然后服务器从数据库里取出盐，再次进行 MD5 计算确密码是否正确。

服务器在每一次启动的时候随机产生一个对称密钥，然后分发给客户端。客户端的信息通过密钥加密后发送给服务器。客户端收到的信息则在客户端上进行解码。

第 10 章 维护设计

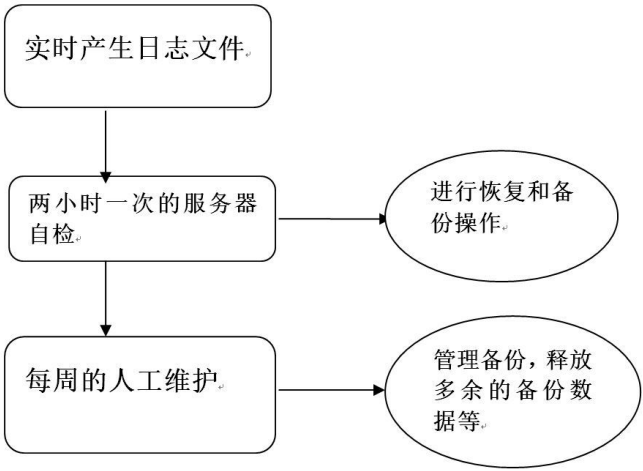


图 10.1 维护流程

参考文献

- Knuth D E. May 1984. Literate programming[J]. *The Computer Journal*. 27(2):97–111.
- Knuth D E. 1986. Computers and Typesetting: A The $\text{T}_{\text{E}}\text{X}$ book[M]. Reading, MA, USA: Addison-Wesley.
- Mittelbach F, Goossens M, Braams J, et al. 2004. The \LaTeX Companion[M]. 2nd ed. Reading, MA, USA: Addison-Wesley.