廣東工業大學

软件工程课程设计

基于 Android 的即时通讯软件 MeChat

学　　院	计算机学院
专　　业	软件工程
年级班级	19（4）班
学　　号	CH3-Orange
学生姓名	Orangepi
指导教师	

2021 年 06 月

目录

1 需求分析

1.1 功能需求

1.1.1 用户模块

在本即时通讯软件 MeChat 中（以下简称 MeChat），一个用户应具有以下功能。

表 1 用户模块功能需求

名称	详细描述
用户注册	用户输入相应信息后能够生成一个用户账户供用户登录
用户登录	用户根据正确的账号密码登入聊天系统中
修改用户信息	用户根据喜好来修改自己的个人信息
搜索用户	能够通过输入的信息在用户数据库中搜索匹配的用户
用户登出	在用户结束活动后可以自行选择登出聊天系统，即下线

1.1.2 社交模块

在 MeChat 中，社交模块应包含以下的功能。

表 2 社交模块功能需求

名称	详细描述
添加好友	用户之间能够相互添加好友
查看用户信息	用户可以查看其它用户的基本信息
新建用户群聊	用户可以将好友拉入新建的群聊中
好友聊天	用户和选定好友进行两两聊天
群聊聊天	用户在选定的群聊里聊天

1.2 性能需求

1.2.1 Android 客户端性能要求

MeChat 是基于 Android 的一款聊天软件，所以应尽可能适配不同 Android 版本的机型，以及不同屏幕类型的手机。在运行时应做到无闪退，获取到运行时异常能够以可视化的方式告知用户，并通知用户联系开发人员。

1.2.2 服务器端性能要求

服务器端应能够支持客户端的访问，且能够支持多用户的并发访问，在多用户进行聊天，服务器应有足够的运行内存来存储临时数据。在用户发送离线消息不能及时收到时，服务器应有足够的磁盘空间存储离线消息。

1.3 其他需求

1.3.1 数据安全性需求

用户密码等高度隐私性数据应通过“加盐”和离散化的方式做好足够的加密，防止因数据库被侵入导致的用户隐私泄露。

1.3.2 界面需求

界面应美观，符合 Material Design 设计理念。降低用户进行相应操作所点击的次数，界面风格简洁明朗，没有过多冗余按钮，当用户做出错误相应时应以可视化的方式提示用户。

1.3.3 可持续化需求

程序中所用到的类应功能明确，且可以进行后续开发，预留出可扩展的接口和字段。

1.4 功能模型

1.4.1 用例图

MeChat 主要分为用户注册、用户登录、搜索、创建群聊、查看好友、聊天、用户登出七个用例，其中搜索用例包含搜索用户和搜索群聊两个用例；聊天用例包含进行群聊和单聊的两个用例，而单聊和群聊又包含机器人自动回复的用例，具体用例图见下图。

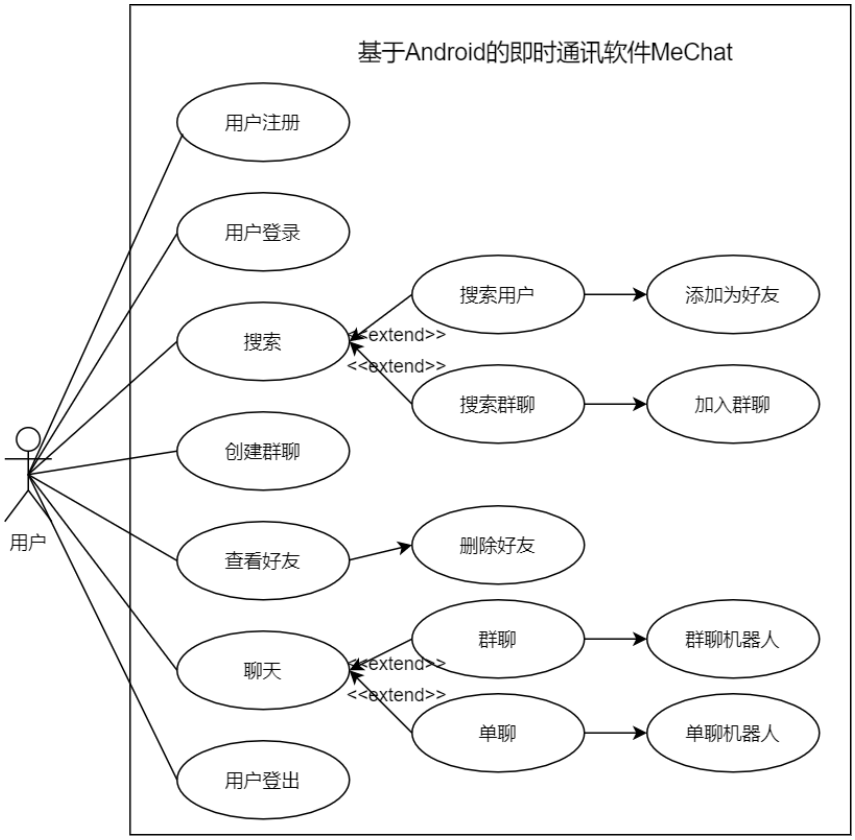

图 1 用例图

1.4.2 用例描述

以下各表为各用例的具体描述，包括了用例的编号、名称、描述、参与者、前置条件、后继条件、主要事件流和备选事件流。

表 3 用户注册用例描述表

名称	内容

用例编号	uc-01
用例名	用户注册
用例描述	用户输入用户名、昵称、性别、密码以及确认密码后生成该用户信息
参与者	用户
前置条件	用户未取得账号
后置条件	用户登录
主事件流	1. 输入用户名 2. 输入昵称 3. 选择性别 4. 输入密码 5. 确认密码 6. 注册成功
备选事件流	1.a 用户名已存在 1.提示用户修改用户名 1.b 用户名不合法 1.展示用户名要求 2.提示用户重新输入用户名 3.a 性别为空 1.提示用户选择性别 4.a 密码不合法 1.展示密码要求 2.提示用户重新输入密码 5.a 两次密码不匹配 1.提示用户密码不匹配

表 4 用户登录用例描述表

名称	内容
用例编号	uc-02
用例名	用户登录
用例描述	用户输入用户名和密码后经过验证进入聊天列表
参与者	用户
前置条件	用户名存在且密码正确
后置条件	进入聊天列表页面
主事件流	1. 输入用户账号 2. 输入用户密码 3. 进行身份验证 4. 获得登入成功通知 5. 绑定后台广播，并传入当前登入的用户信息 6. 进入聊天列表页面
备选事件流	4.a 获得用户名异常通知 1. 提示无此用户 2.结束用例 4.b 获得密码异常通知 1.提示密码错误 2.结束用例

表 5 搜索用例描述表

名称	内容
用例编号	uc-03
用例名	搜索
用例描述	用户点击搜索后可以选择搜索好友或群聊
参与者	用户

前置条件	用户点击搜索按钮
后置条件	搜索好友或群聊
主事件流	1. 用户选择搜索好友或群聊 2. 跳转到搜索页面进行搜索 3. 展示搜索结果
备选事件流	无

表 6 搜索用户用例描述表

名称	内容
用例编号	uc-04
用例名	搜索用户
用例描述	通过用户输入的用户名或用户昵称来搜索匹配用户
参与者	用户
前置条件	用户选择搜索用户
后置条件	无
主事件流	1. 获取用户输入的文本 2. 获取用户查询结果 3. 返回结果给主界面
备选事件流	无

表 7 搜索群聊用例描述表

名称	内容
用例编号	uc-05
用例名	搜索群聊
用例描述	通过用户输入的群聊名搜索匹配群聊
参与者	用户

前置条件	用户选择搜索群聊
后置条件	无
主事件流	1. 获取用户输入文本 2. 获取群聊查询结果 3. 返回结果给主页面
备选事件流	无

表 8 添加好友用例描述表

名称	内容
用例编号	uc-06
用例名	添加好友
用例描述	用户选择一位用户后添加其为好友
参与者	用户
前置条件	用户选择了一位陌生人，即非好友的用户
后置条件	无
主事件流	1. 用户选择一位用户添加好友 2. 将好友信息写入数据库 3. 将好友信息写入通讯录 4. 允许用户和好友进行聊天
备选事件流	无

表 9 加入群聊用例描述表

名称	内容
用例编号	uc-07
用例名	加入群聊
用例描述	用户选择一个群聊加入

参与者	用户
前置条件	用户选择了一个群聊成员不包含自己的群聊
后置条件	无
主事件流	1. 用户选择一个群聊加入 2. 将用户和群聊信息写入数据库 3. 允许用户进入群聊发言
备选事件流	无

表 10 创建群聊用例描述表

名称	内容
用例编号	uc-08
用例名	创建群聊
用例描述	用户输入群聊名称和加入的好友来创建群聊
参与者	用户
前置条件	选择创建群聊
后置条件	无
主事件流	1. 用户输入群聊名称 2. 用户选取好友 3. 向数据库注册群聊信息 4. 展示群聊对话框到主界面
备选事件流	2.a 用户未选择好友 1.提示至少选择一位好友

表 11 查看好友用例描述表

名称	内容
用例编号	uc-09
用例名	查看好友

用例描述	用户进入通讯录查看自己的全部好友
参与者	用户
前置条件	用户进入通讯录页面
后置条件	无
主事件流	1. 用户进入通讯录页面 2. 页面从后台服务拉取用户当前全部好友 3. 将数据反馈到界面
备选事件流	无

表 12 删除好友用例描述表

名称	内容
用例编号	uc-10
用例名	删除好友
用例描述	用户点击删除好友后删除该好友
参与者	用户
前置条件	用户点击好友列表对应好友的删除键
后置条件	无
主事件流	1. 用户点击好友列表的删除键 2. 向数据库发出删除好友的命令 3. 获取数据库反馈成功 4. 将该好友所在行删除 5. 重新渲染界面 6. 提示用户已经删除
备选事件流	3a.数据库反馈失败 1.提示用户删除失败

表 13 聊天用例描述表

名称	内容
用例编号	uc-11
用例名	聊天
用例描述	用户开始聊天
参与者	用户
前置条件	无
后置条件	进入群聊或单聊
主事件流	1. 用户选择一个聊天 2. 根据聊天类型进入相应的聊天
备选事件流	无

表 14 群聊用例描述表

名称	内容
用例编号	uc-12
用例名	群聊
用例描述	用户进入群聊
参与者	用户
前置条件	用户进入的聊天类型为群聊
后置条件	无
主事件流	1. 获取用户进入的聊天名称 2. 接收消息广播 3. 筛选广播中的消息，将用户当前群聊的消息显示到界面上
备选事件流	无

表 15 单聊用例描述表

名称	内容
用例编号	uc-13
用例名	单聊
用例描述	用户进入单聊
参与者	用户
前置条件	用户进入的聊天类型为单聊
后置条件	无
主事件流	4. 获取用户进入的聊天名称 5. 接收消息广播 6. 筛选广播中的消息，将用户当前群聊的消息显示到界面上
备选事件流	无

表 16 用户登出用例描述表

名称	内容
用例编号	uc-14
用例名	用户登出
用例描述	用户退出登录
参与者	用户
前置条件	用户选择退出登录
后置条件	无
主事件流	1. 回到登录界面 2. 解绑后台服务和广播
备选事件流	无

名称	内容
用例编号	uc-15

用例名	单聊机器人
用例描述	用户与单聊机器人聊天
参与者	用户
前置条件	用户选择单聊机器人的聊天
后置条件	无
主事件流	1. 机器人解析用户的消息 2. 生成自动回复返回给用户
备选事件流	无

名称	内容
用例编号	uc-16
用例名	群聊机器人
用例描述	用户进入群聊与机器人聊天
参与者	用户
前置条件	用户选择退出登录
后置条件	无
主事件流	1. 机器人解析用户消息 2. 判断消息的发送方进行鉴权 3. 将自动回复返回给群聊
备选事件流	无

1.5 静态模型

本系统的主要领域类图由 UserMessage、Mail、MyService、TextMessage、User 和 ServerUser 六个类构成，该图主要描述了在网络通信时几个类之间的关系。

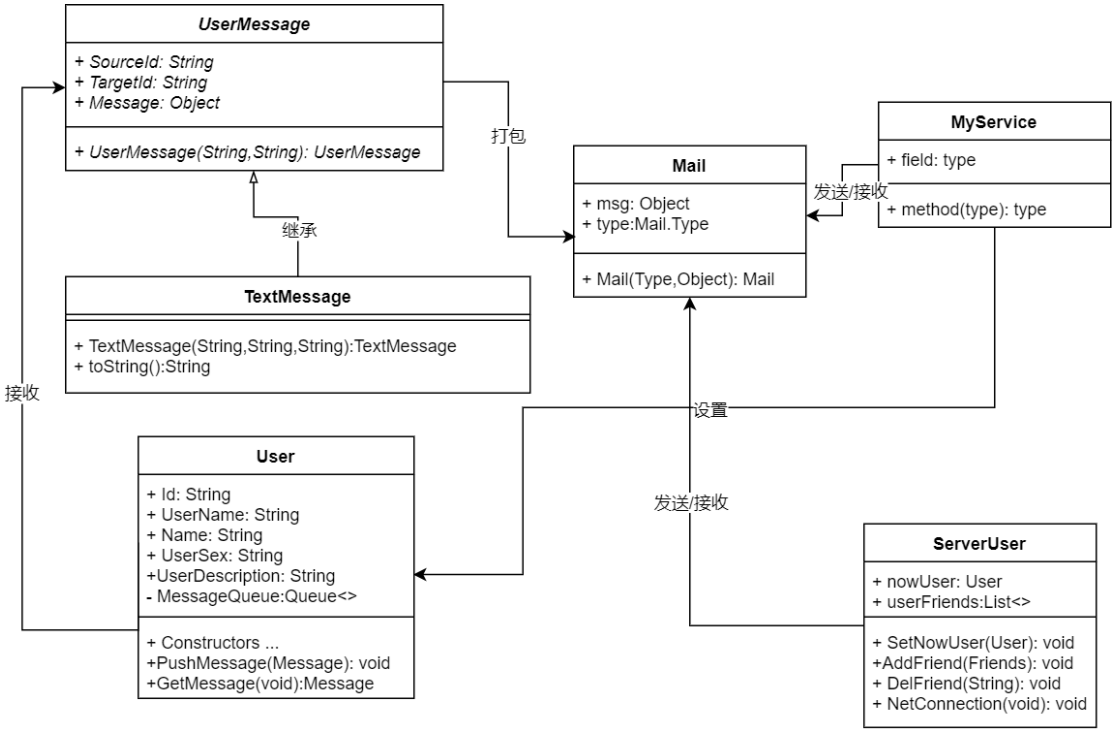

图 2 网络通信时的领域类图

1.6 实体联系图

经分析后提取出三个实体，分别是用户、好友和群聊，具体联系见下表。

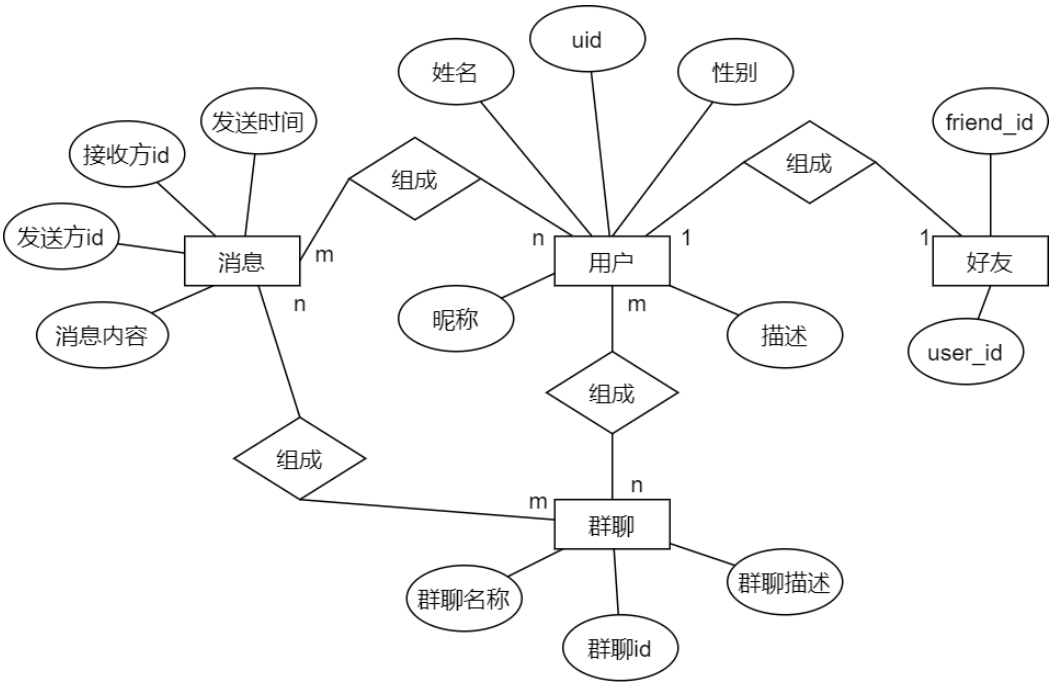

图 3 实体联系图

2 概要设计

2.1 系统整体架构设计

2.1.1 系统构件图

MeChat 在设计上由 MySQL 数据库、服务端和客户端三部分组成，其中数据全部保存在 MySQL 数据库上，服务端和客户端对于数据的更新与获取直接通过 MySQL，对于用户消息的分发则交由服务器进行处理，也就是用户的所有聊天信息都上传至服务器，服务器解包之后再进行分发。

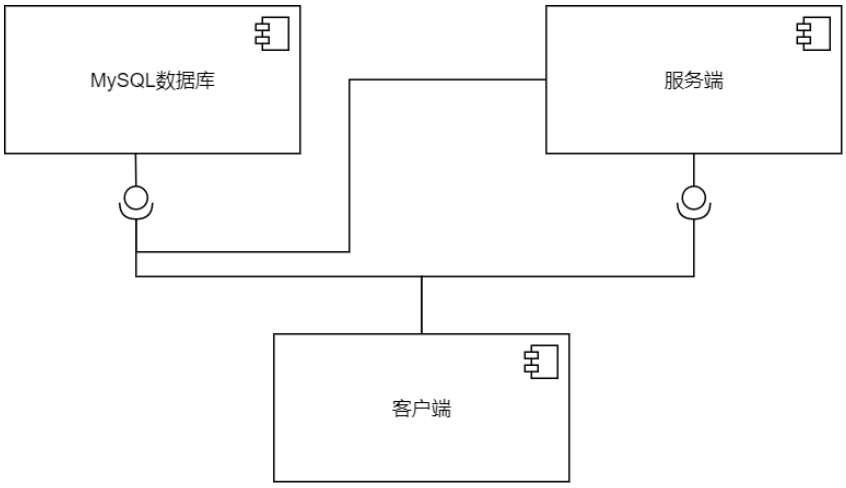

图 4 系统构件图

2.2 模块设计

2.2.1 客户端功能模块设计图

MeChat 主要有三大模块，分别是用户模块、好友模块和聊天模块。用户模块中包含了用户登录和注册的两个功能；好友模块中包括了添加和删除好友的两个功能；聊天模块中包含了发起、进入和删除聊天的三个功能。

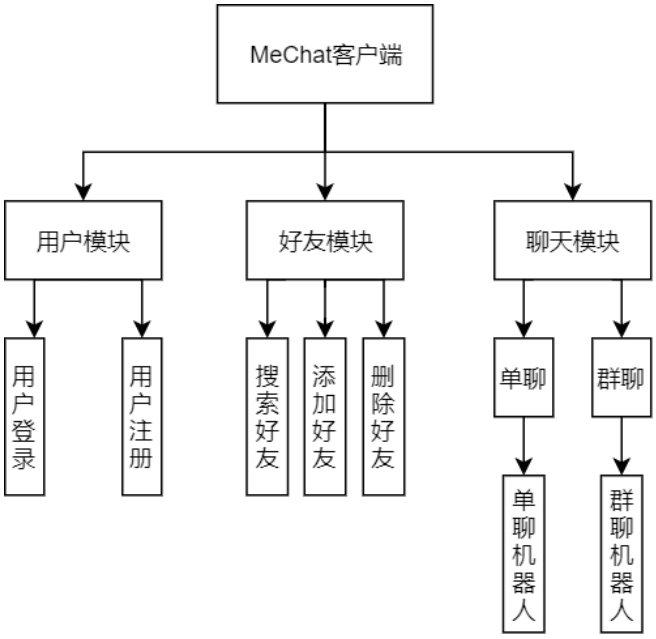

图 5 Android 客户端模块设计图

2.2.2 用户登录模块

用户填写完用户名和密码后点击登录按键触发用户登录的用例，从界面类获取到用户名和密码后调用数据库交互类的静态方法对用户密码进行验证（具体验证方法见*图 28 验证用户身份流程图*），验证通过后将用户信息发给后台服务，后台服务在设置完用户信息后开始连接远程服务端，连接服务端成功后获取 Socket 对象并开启两个监听线程，一个用于监听本地输出，将本地输出通过此 Socket 传给服务端；另一个用于监听来自服务端的 Socket 的输入，获取输入进行初步的解包，检测到服务器返回了登录成功的信息后发送广播给登录界面，登录界面接收到广播之后反馈给用户登录成功的信息，并进入到下一个界面。

下图为用户登录在 Android 客户端的顺序图。

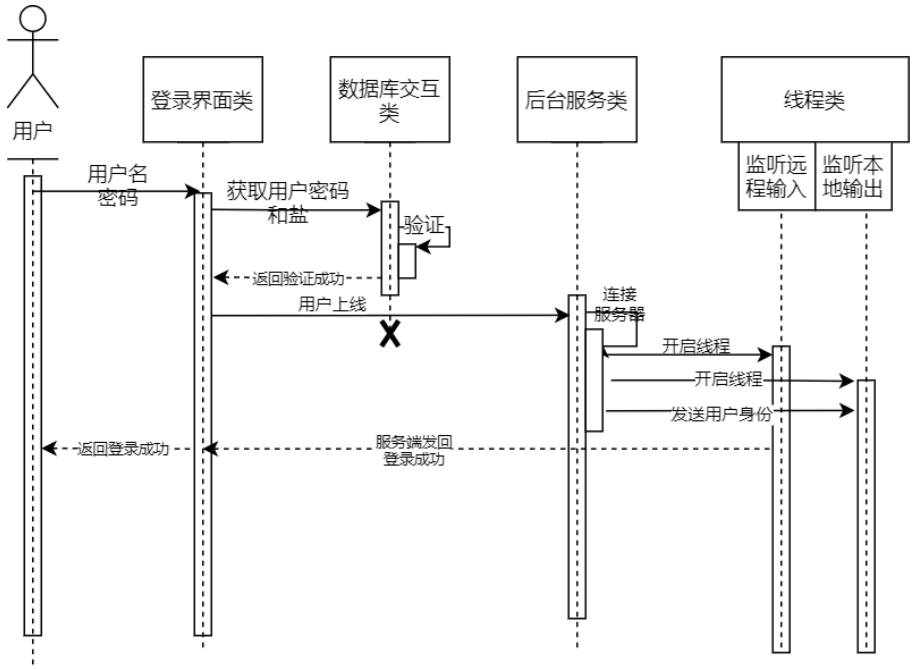

图 6 用户登录在 Android 客户端的顺序图

2.2.3 用户注册模块

用户输入各项信息后注册按键首先会在界面检查用户输入的合法性，比如用户名是否为纯数字和字母、性别项是否已选、密码是否包含一个数字和字母、两次密码是否一致。若通过了合法性检验就会触发用户注册的用例，在用户注册的用例中会先开一个线程与数据库通信，检测是否已有该用户名的用户，若没有则允许用户继续注册，界面类在等到线程返回允许继续注册后会将结果再次开启一个线程与数据库通信，将新用户的信息发给数据库交互类，数据库交互类在执行完数据的插入后会返回信息，界面类再将信息以可视化的形式展现给用户。

以下是用户注册的顺序图。

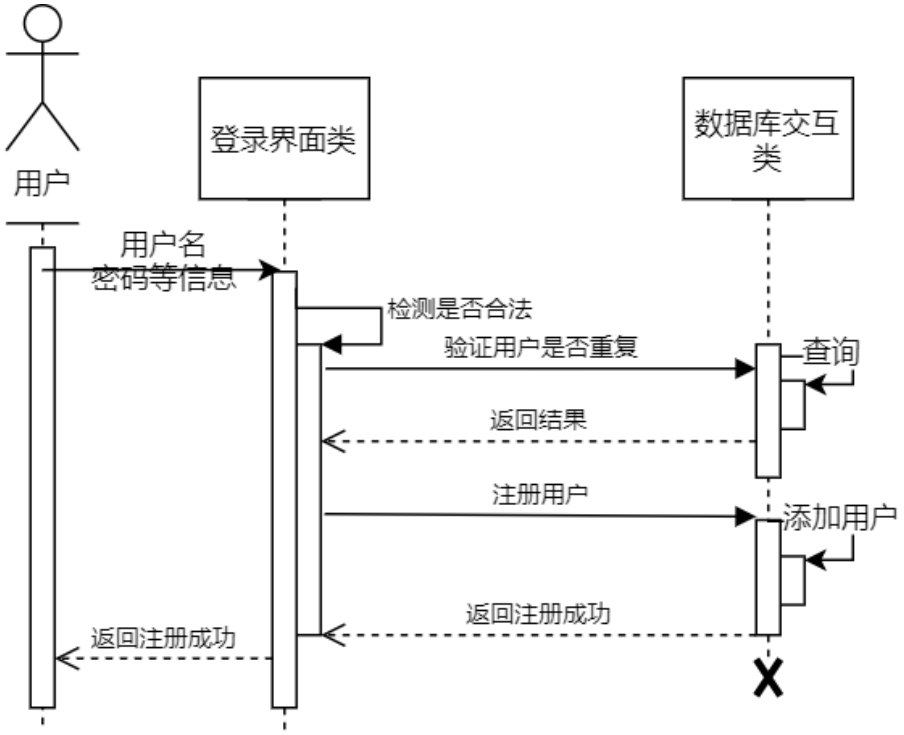

图 7 用户注册的顺序图

2.2.4 搜索用户模块

用户进入搜索界面后输入要搜索的用户名，然后点击搜索后进入搜索用例，在搜索的用例中会先向数据库拉取搜索的结果，然后再向好友表拉取当前用户的好友。将两个结果进行过滤处理：若搜索的用户已经是当前用户的好友，则在界面显示“聊天”和“删除”的按钮；若搜索的用户和当前用户不是好友，则只显示“加好友”的按钮。

下图是搜索用户的数据流图和顺序图。

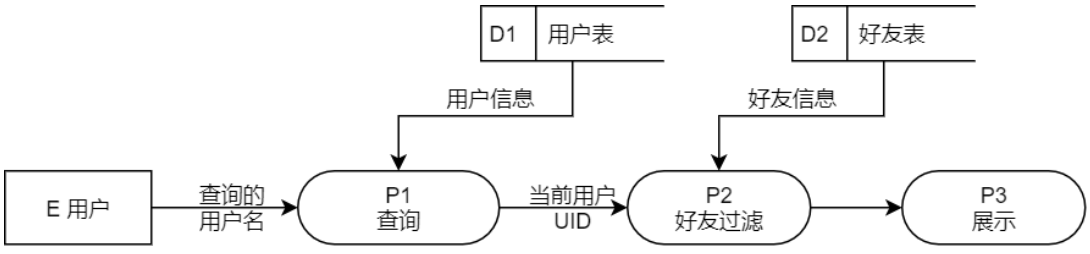

图 8 搜索用户的数据流图

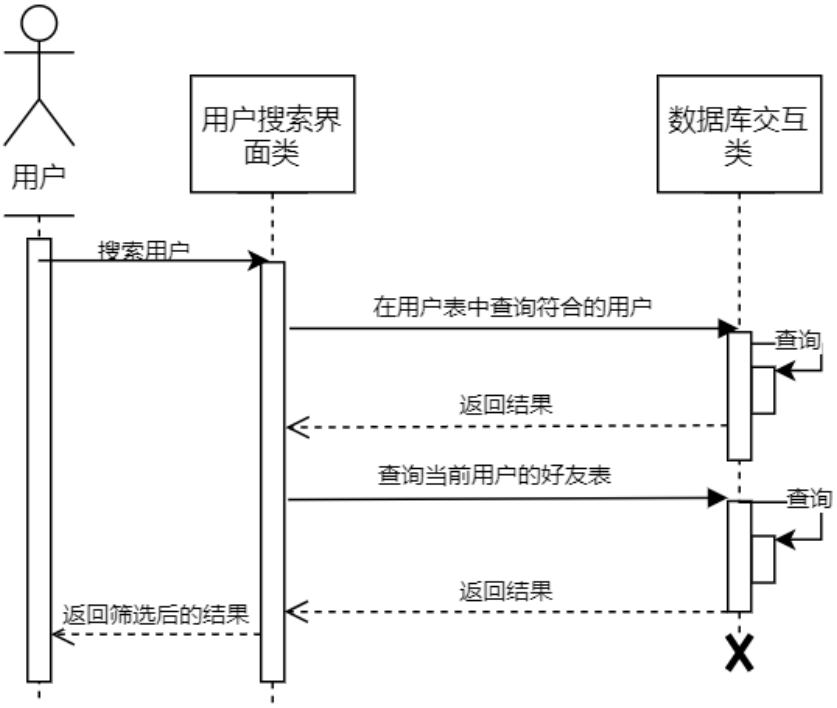

图 9 搜索用户的顺序图

2.2.5 添加好友模块

用户点击添加好友后进入添加好友的用例，在添加好友的用例中首先新建一个线程用于和数据库进行交互，然后将自己和对方添加一条单向好友关系，待添加成功后，再将对方和自己添加一条单向好友关系，再将最后的结果反馈给搜索界面。这里添加好友关系可以待以后添加好友确认功能后，只保留第一次的添加单向好友关系，等待对方确认后才能再次添加一条好友关系。

下图为添加好友用例的顺序图。

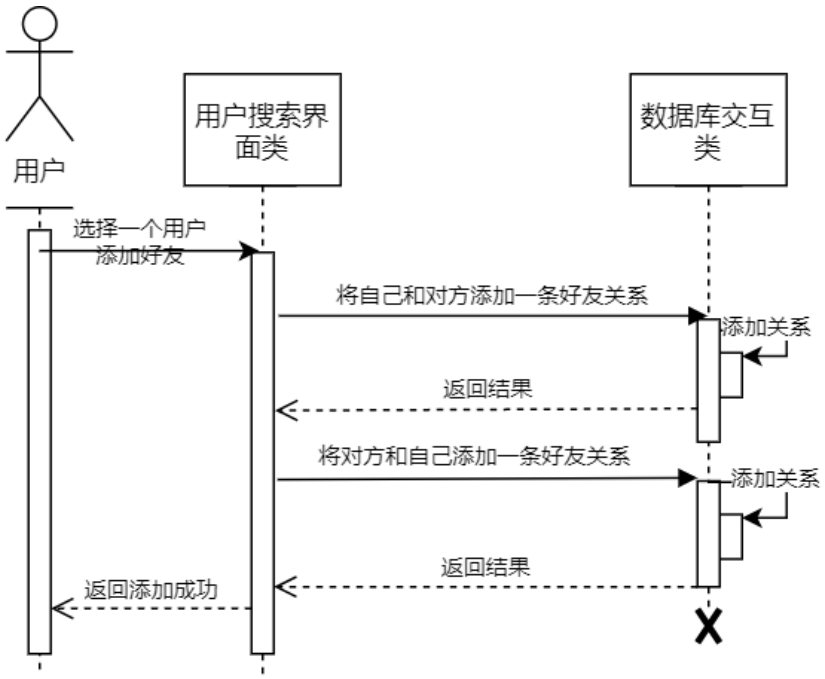

图 10 添加好友的顺序图

2.2.6 删除好友模块

删除好友是添加好友的逆向操作，只需在对数据库操作时将添加关系改为删除关系即可，这里不再赘述。

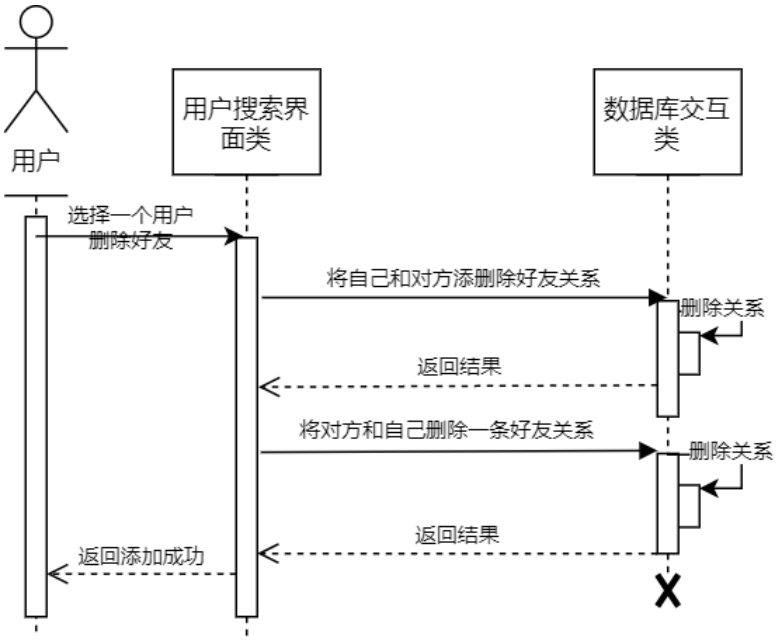

图 11 删除好友顺序图

2.2.7 单聊模块

聊天是整个系统最核心的功能，聊天中一条信息的传输要首先通过自己 MeChat 客户端的 Socket 提交到服务端，服务端接收到该条消息后会对消息进行初步解包，获取消息的发送对象，将该条消息 push 到对方消息的消息队列中。此时，若对方在线，则服务器上会有一条对方服务的线程一直监听他的消息队列，当获取的新的信息后会将当前的消息交由 Socket 发送给对方的客户端，再将消息从消息队列中 pop 出。对方客户端监测到新消息后会通过广播的形式发送给各个绑定广播的 Activity 类。聊天列表类（ChatListActivity）接收到消息后会先检测当前聊天列表中是否存在消息发出人的聊天，如果有则将消息放入该“表项”的消息队列中，当用户点击该“表项”后会由消息队列重建出未读的聊天信息展示到界面中；若当前列表没有该消息发出人的聊天，则拉取该用户的所有好友进行匹配，匹配到后新建一个该好友的“表项”，并将消息放到该表项的消息队列中，再将该表项加入到页面的列表中进行展示。

以下是用户 1 和用户 2 分别发了一条聊天消息的顺序图

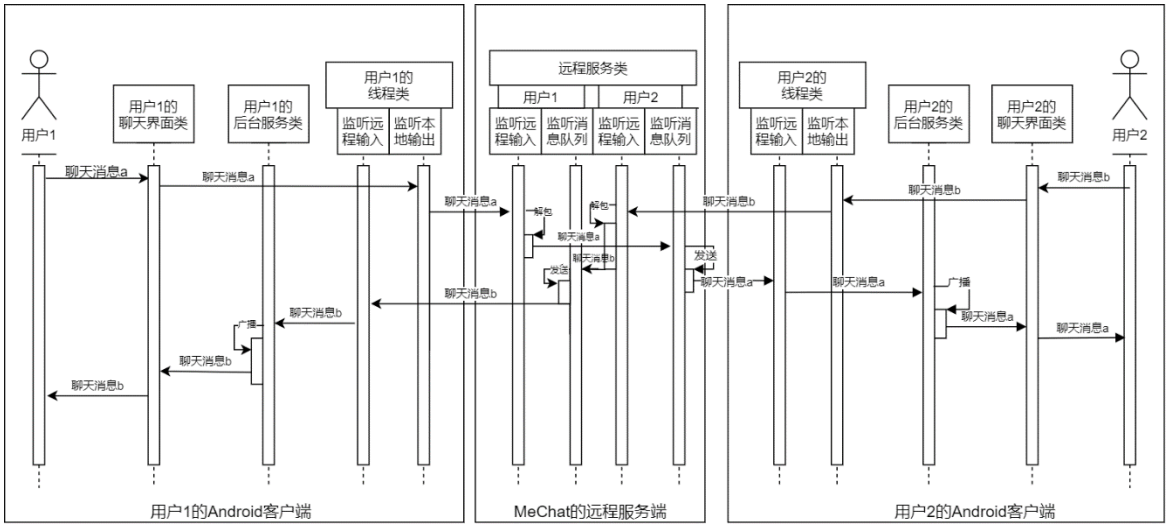

图 12 单聊的顺序图

2.2.8 群聊模块

群聊与单聊的逻辑类似，只是在单聊中，用户发送方是另外一个用户。而在

群聊中，用户的发送方是群组，同时消息类型也为群消息类型。对于单聊消息和群聊消息的分发处理交由服务端进行。服务端在获取一个数据时会进行初步解包，判断是是否是聊天消息，如果是聊天消息则继续判断是单聊还是群聊消息，若是群聊消息则读取目标群聊 id，通过群聊 id 找到加入群聊的用户 id，再将消息原封不动发送给群聊中包含的各个用户，时序图如下（简化了客户端的时序关系）。

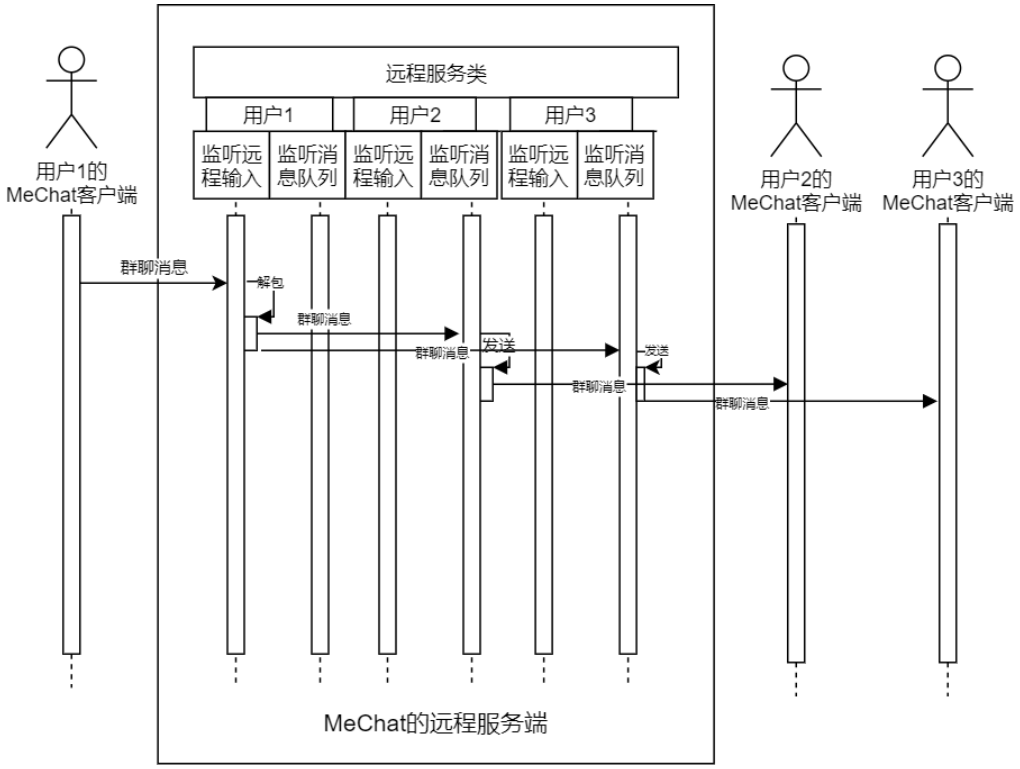

图 13 群聊的时序图

2.2.9 单聊机器人

首先，机器人与用户类似，都是一个 MeChat 的用户，只不过机器人的不在客户端登录，而在服务端登录。机器人与服务端一同运行在服务器上，用户与机器人的聊天与单聊无异，但是机器人会对用户的数据进行分析处理，并将分析的结果发送回用户。

在单聊的机器人中，用户不光可以与机器人进行聊天对话，还可以向机器人发出指令，机器人会先将用户的消息进行关键词检测，若检测到相应的相应语句后会对响应语句进行分析，若语句中为 mqtt 的操作语句则向服务器中的 mosquitto 服务发送相关指令，并等待服务返回结果。此时，物联网终端设备会监

听到服务器 mosquitto 服务的请求，并作出相应的动作，然后将动作结果再返回给服务器的 mosquitto 服务，机器人则会监听到 mosquitto 服务的相应返回结果，然后再将结果返回给用户，完成用户对物联网设备的操控。例如：“用户发送‘能帮忙开灯吗’这个消息”的事件，首先会先发送给服务端，服务端解包后将消息转发给机器人，机器人进行关键字检测，检测到“开灯”后读取相应的响应语句，然后判断响应语句中含有 mqtt 的“mqtt,ledon”操作，读取操作中向 mosquitto 发送的话题“652/led_send”和接收的话题“652/led_recv”，并向发送的话题发送指令语句“led on”，然后订阅接收的话题并阻塞等待。物联网终端设备在通电联网后就会向服务器订阅一个话题“652/led_send”，并持续监听是否有消息。当有消息传回时对消息进行分析并作出相应的点亮 led 操作，然后将结果“led on ok”发送给服务器的“652/led_recv”话题。机器人在接收到返回的订阅消息“led on ok”后再将该消息作为最后的消息发送回用户，用户最终受到反馈“led on ok”。

下面是该例子的时序图（简化了客户端的时序关系）。

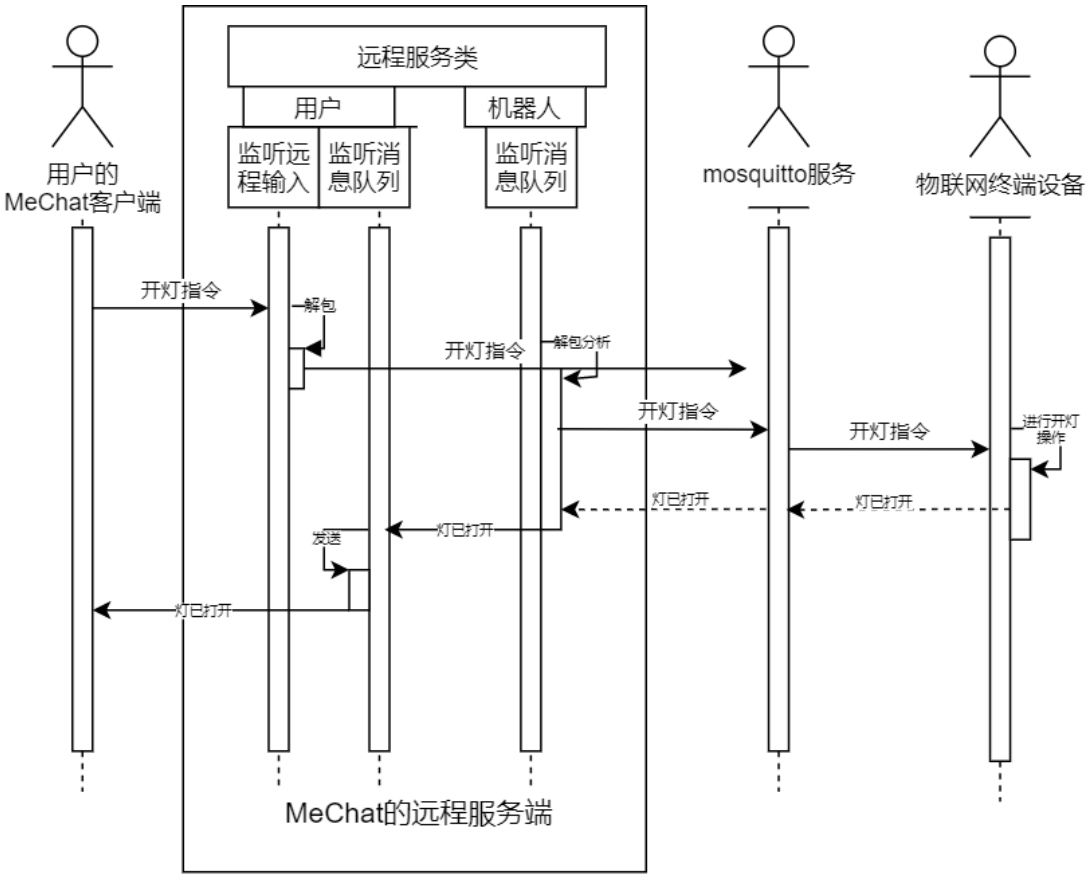

图 14 单聊机器人开灯例子的时序图

2.2.10 群聊机器人

在群聊中，机器人只进行基于关键词的自动回复，若没有检测到回复的关键词则不作处理，同时不响应对应物联网终端的指令操作，即检测到关键词的响应语句中有 mqtt 指令也不作操作。

下面给出在群聊中机器人的自动回复时序图（简化了客户端的时序关系）。

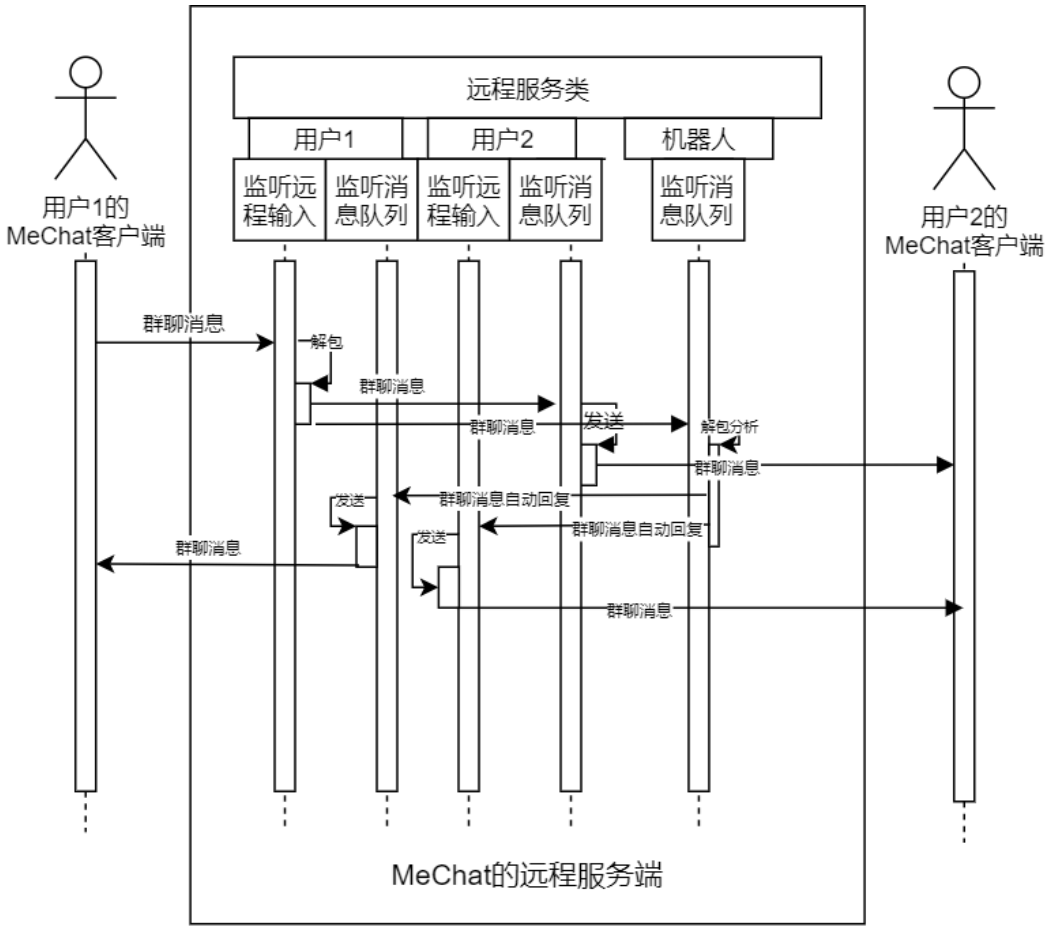

图 15 群聊机器人的自动回复

3 详细设计

3.1 Android 交互界面设计

3.1.1 页面总体设计

MeChat 中一共包含登录界面、注册界面、聊天列表界面、添加用户界面、好

友列表界面、聊天界面，各界面之间的跳转关系如下。

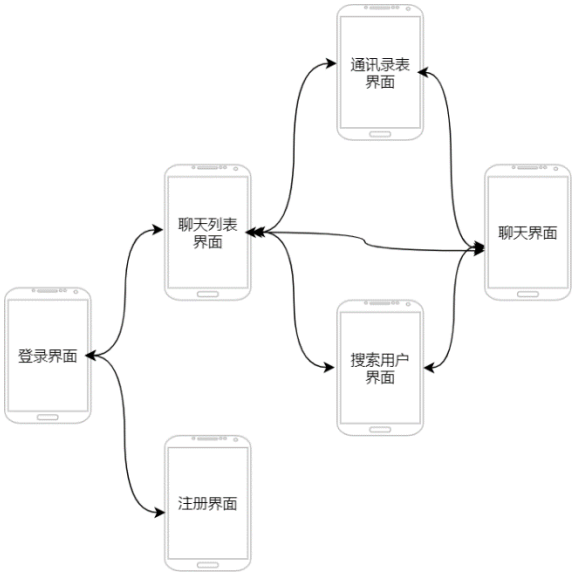

图 16 Android 页面跳转关系图

3.1.2 登录界面设计

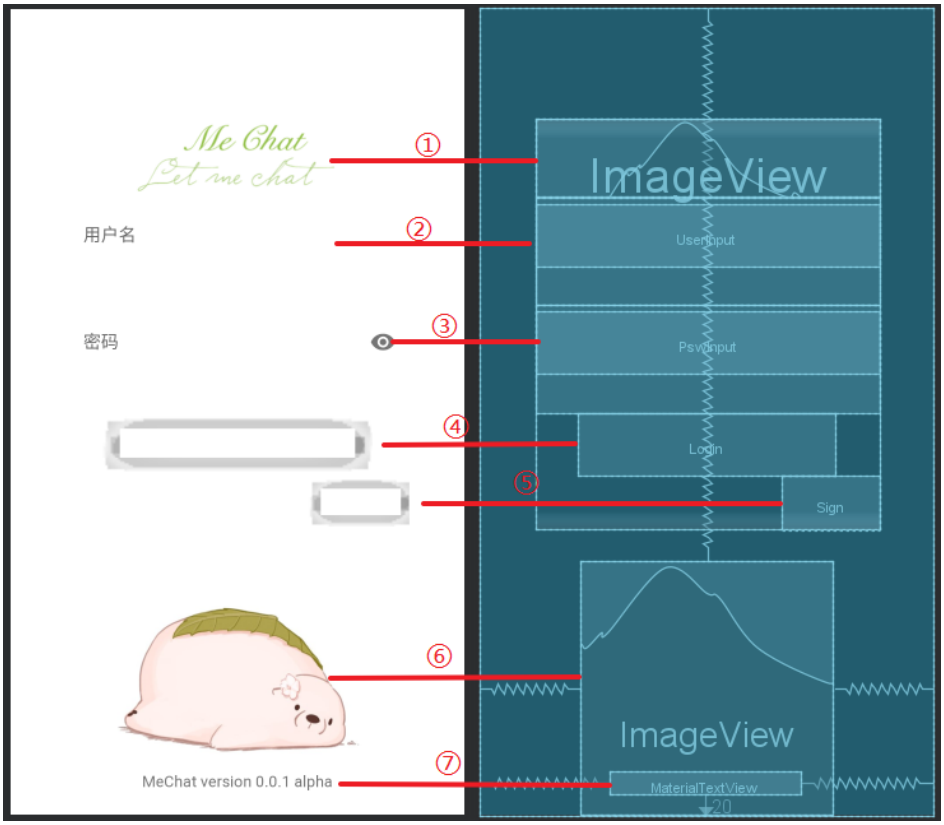

图 17 登录界面蓝图

表 17 登录界面控件表

编号	类型	说明
①	图片	MeChat 的 LOGO
②	输入框	输入用户名
③	输入框	输入密码
④	按钮	登录按钮【在未获取到登录结果时不可重复按】
⑤	按钮	注册按钮
⑥	图片	背景图
⑦	文本	用于放软件信息

3.1.3 注册界面设计

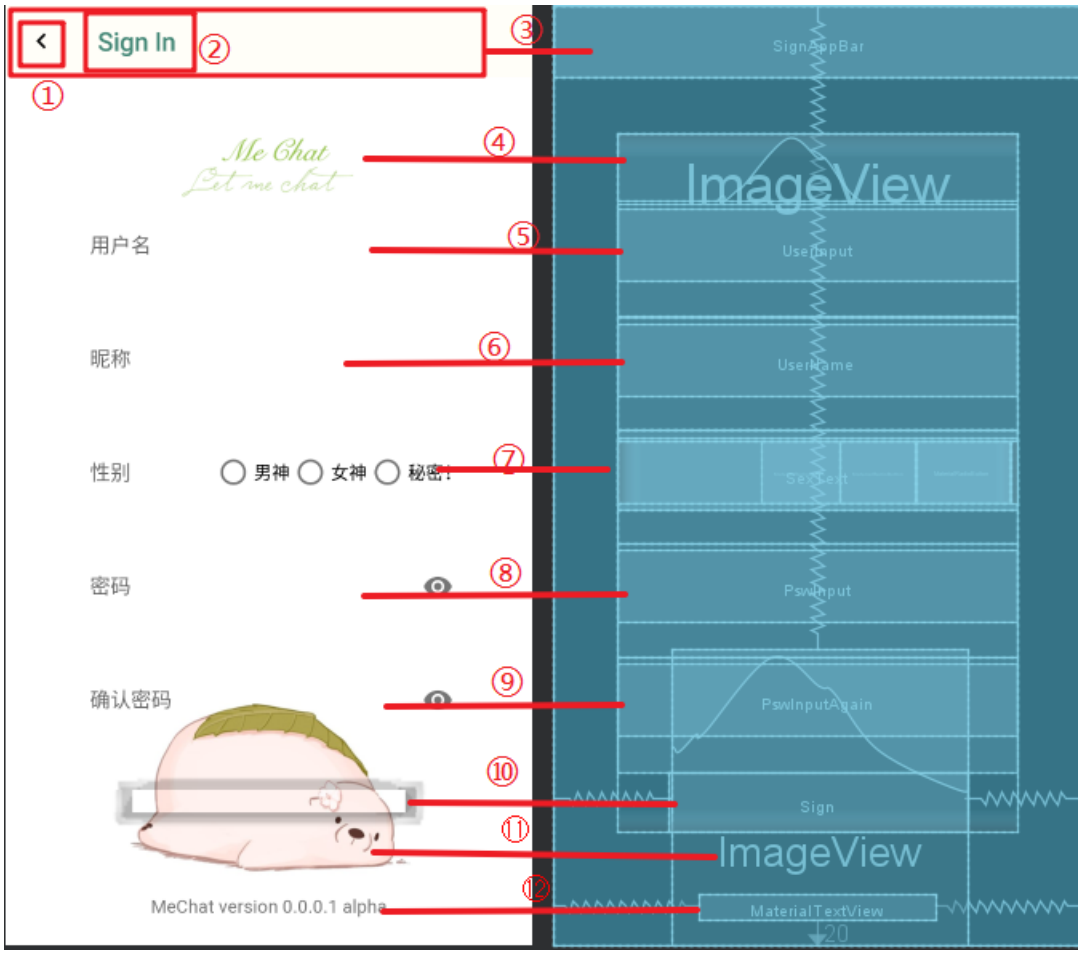

图 18 注册界面蓝图

表 18 注册界面控件表

编号	类型	说明
①	按钮	返回上一级界面
②	文本	展示当前界面名称
③	顶部工具栏	无
④	图片	MeChat 的 LOGO
⑤	输入框	输入用户名【检查数据库是否存在同名】【检查只允许数字字母】
⑥	输入框	输入昵称
⑦	单选框	选择性别【检查不得为空】
⑧	输入框	设定密码【检查至少包含一个数字一个字母】
⑨	输入框	再次输入密码【检查与⑧中是否一致】
⑩	按钮	注册按钮【未获取到结果时不可多次点击】
⑪	图片	背景图
⑫	文本	用于放软件信息

3.1.4 聊天列表界面设计

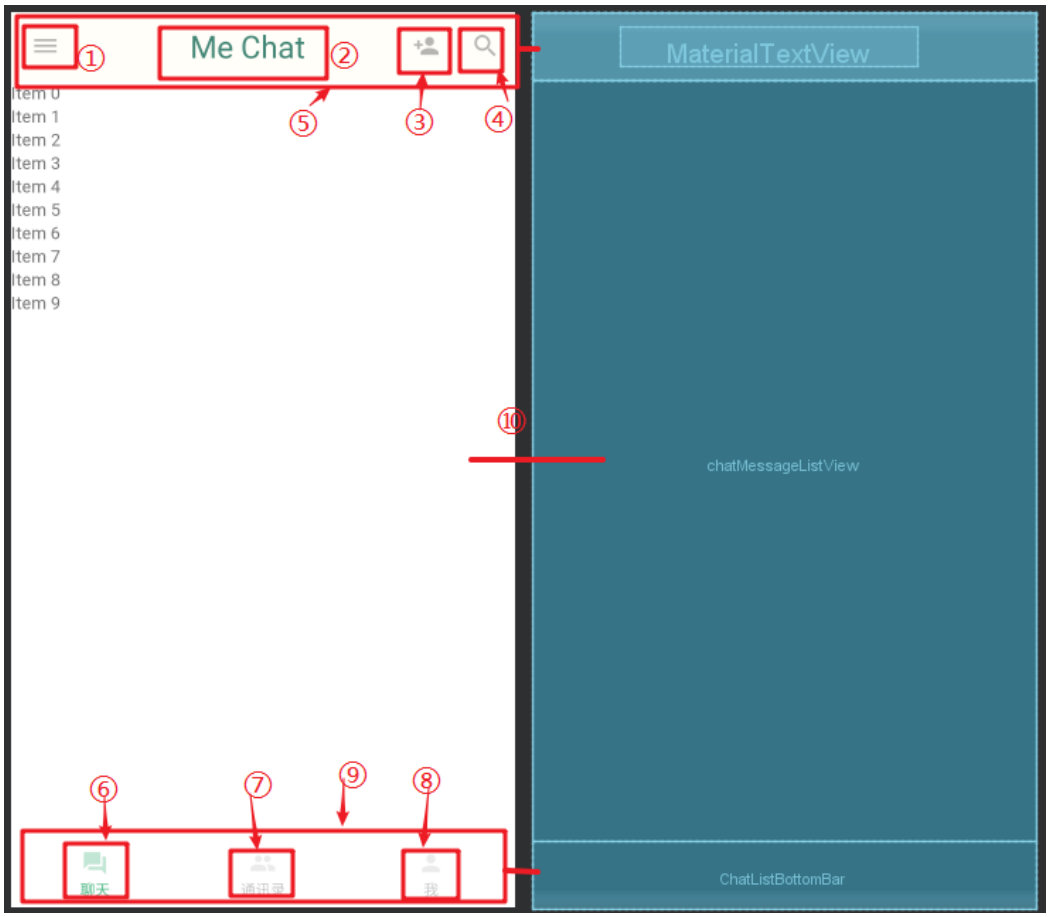

图 19 聊天列表界面蓝图

表 19 聊天界面控件表

编号	类型	说明
①	按钮	返回上一级界面
②	文本	展示当前界面名称
③	按钮	跳转至搜索用户界面
④	按钮	跳转至搜索界面
⑤	顶部工具栏	无
⑥	按钮	跳转至聊天界面
⑦	按钮	跳转至通讯录界面
⑧	按钮	跳转至个人信息界面
⑨	底部工具栏	无

⑩	列表	展示当前所有聊天信息列表

3.1.5 聊天界面设计

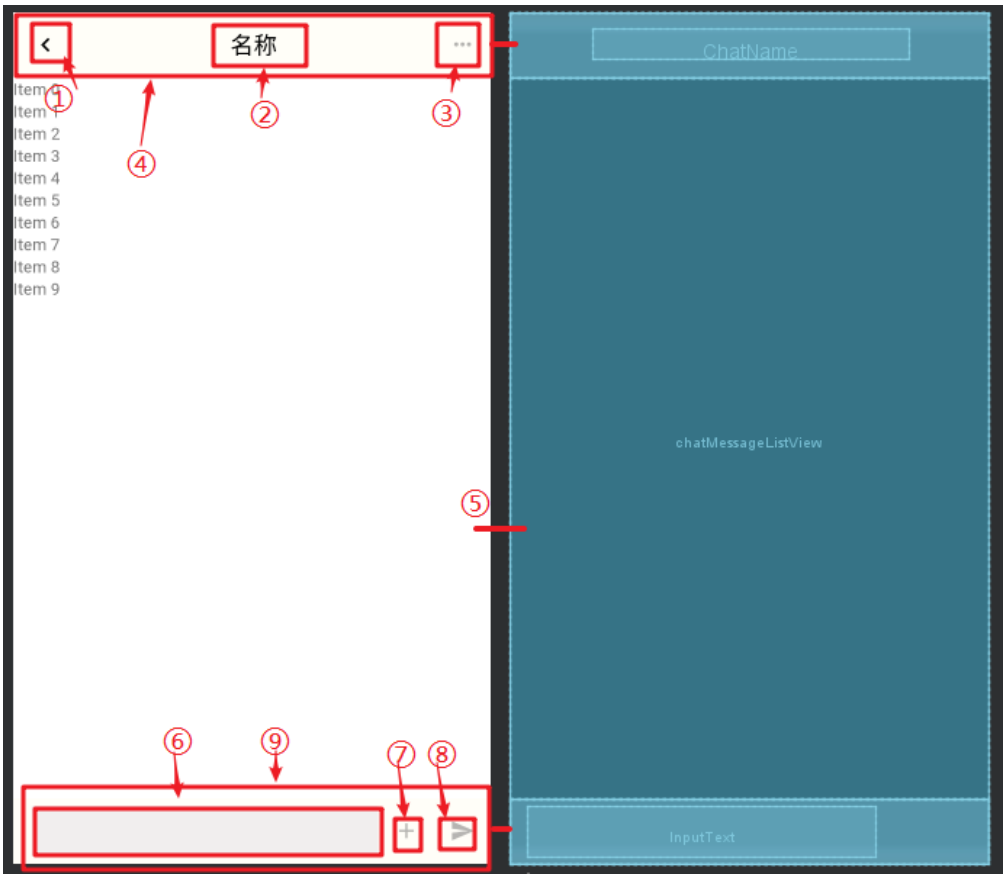

图 20 聊天界面蓝图

表 20 聊天界面控件表

编号	类型	说明
①	按钮	返回上一级界面
②	文本	展示当前界面名称
③	按钮	显示更多信息
④	顶部工具栏	无
⑤	列表	展示当前聊天内容列表
⑥	输入框	输入聊天信息
⑦	按钮	输入其他类型信息

⑧	按钮	发送当前输入框信息
⑨	底部工具栏	无

3.1.6 通讯录界面设计

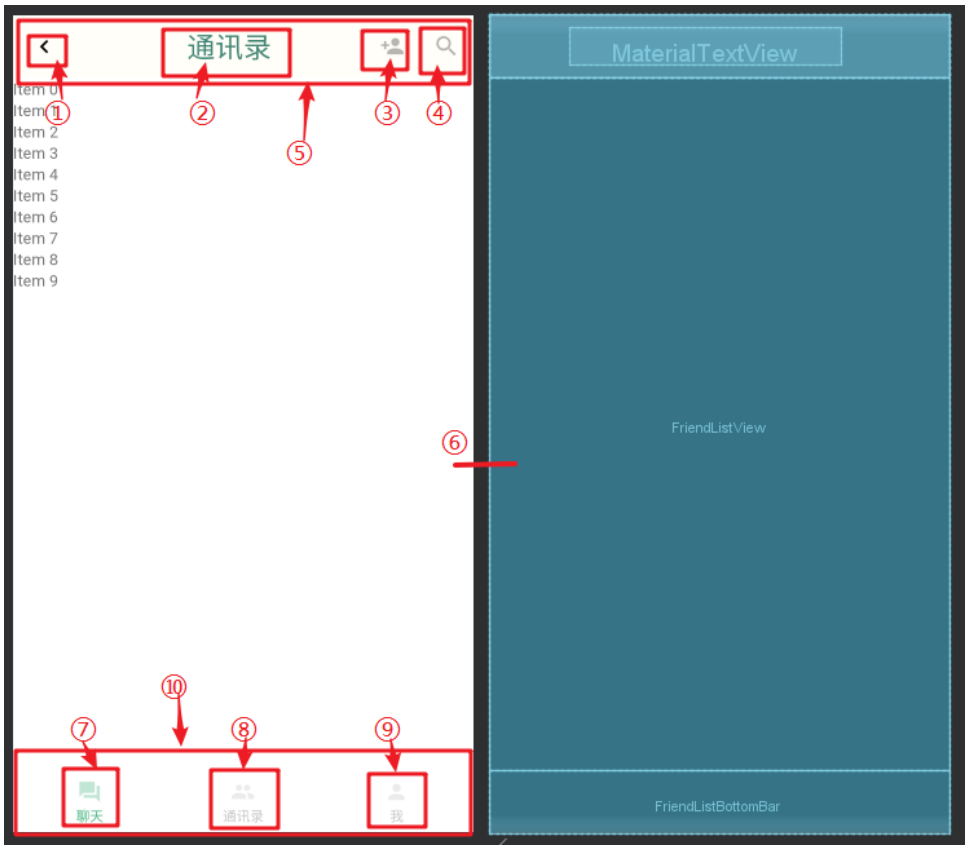

图 21 通讯录界面蓝图

表 21 通讯录界面控件表

编号	类型	说明
①	按钮	返回上一级界面
②	文本	展示当前界面名称
③	按钮	跳转至添加好友界面
④	按钮	跳转至搜索界面
⑤	顶部工具栏	无
⑥	列表	展示当前好友和群聊列表

⑦	按钮	跳转至聊天界面
⑧	按钮	跳转至通讯录界面
⑨	按钮	跳转至个人信息界面
⑩	底部工具栏	无

3.1.7 搜索用户界面设计

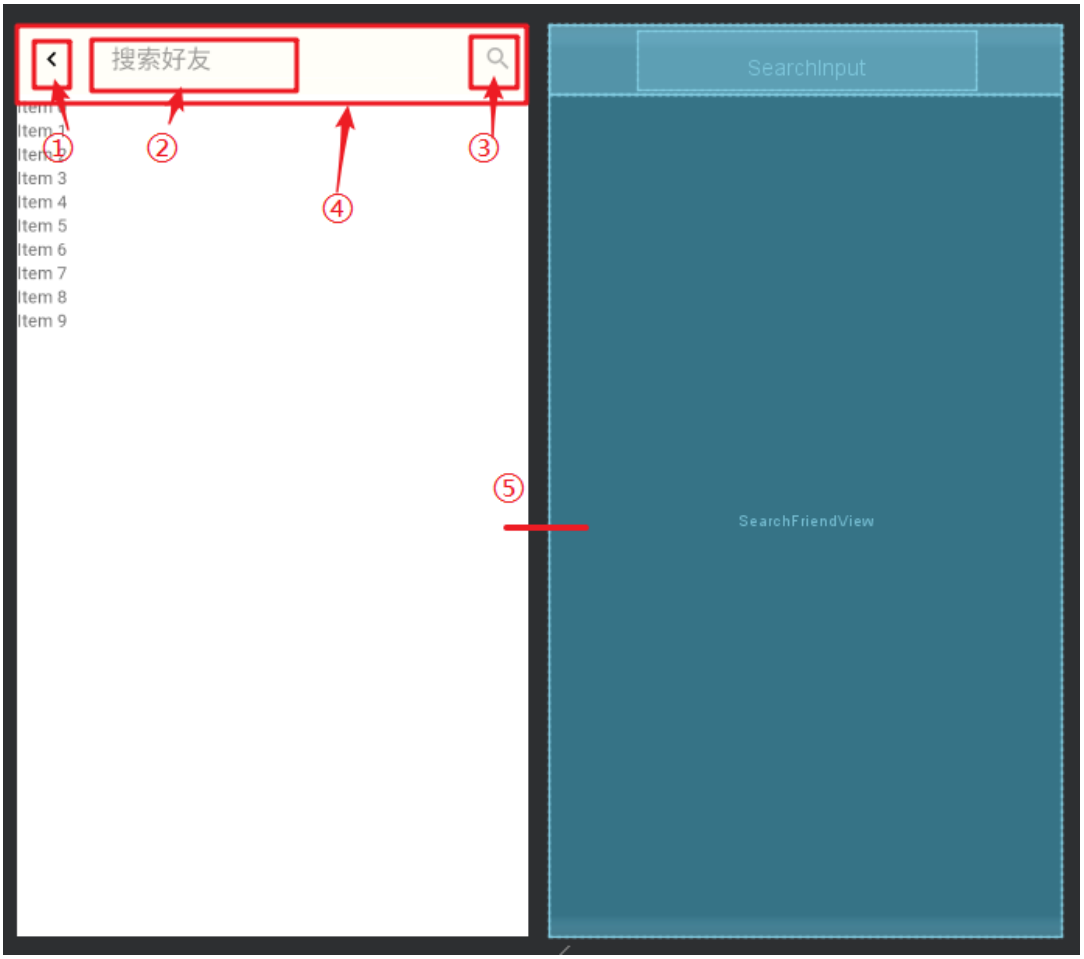

图 22 搜索用户界面蓝图

表 22 搜索用户界面控件表

编号	类型	说明
①	按钮	返回上一级界面
②	输入框	输入用户名或用户昵称

③	按钮	开始搜索【搜索未完成时不能多次点击】
④	顶部工具栏	无
⑤	列表	展示搜索的好友结果列表

3.1.8 聊天信息表项界面设计

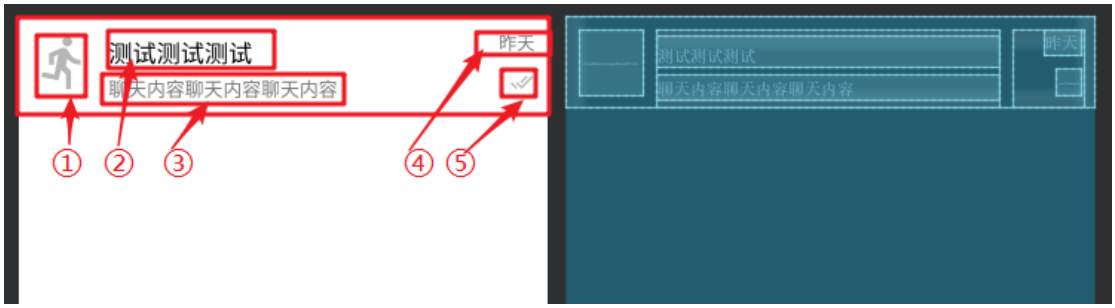

图 23 聊天信息表项界面蓝图

表 23 聊天信息表项界面控件表

编号	类型	说明
①	图片	对方用户或群聊头像
②	文本	对方用户名或群聊名
③	文本	最后收到的消息
④	文本	消息收到的时间
⑤	图片	消息信息图标

3.1.9 聊天对话信息表项界面设计

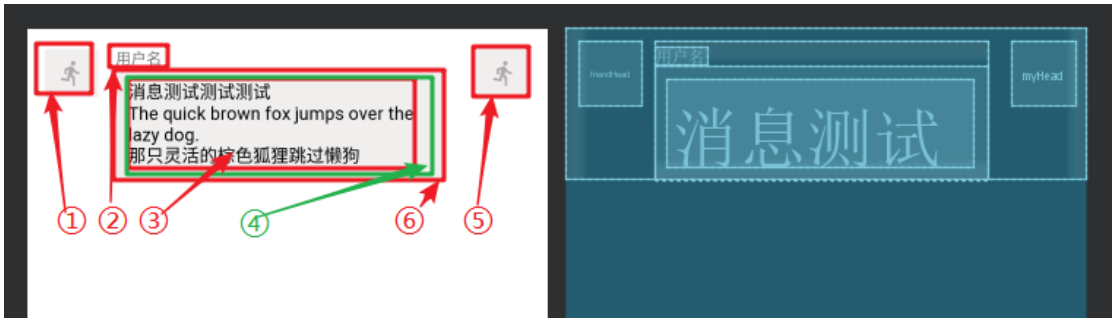

图 24 聊天对话信息表项界面蓝图

表 24 聊天对话信息表项界面控件表

编号	类型	说明
①	图片	对方用户头像【消息类型为“收到”则显示】
②	文本	对方用户名【消息类型为“收到”则显示】
③	文本	文字消息具体内容【消息类型为“文字”则显示】
④	图片	图片消息内容【消息类型为“图片”则显示】
⑤	图标	自己的头像【消息类型为“发送”则显示】
⑥	背景	消息背景框

3.1.10 好友表项界面设计

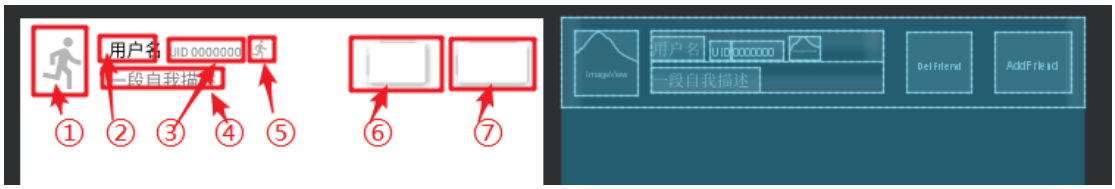

图 25 好友表项界面蓝图

表 25 好友表项界面控件表

编号	类型	说明
①	图片	用户头像
②	文本	用户名
③	文本	用户 UID
④	文本	用户描述
⑤	图片	用户性别
⑥	按钮	删除按钮 【如果该用户和自己是好友则显示】
⑦	按钮	聊天/加好友按钮 【如果该用户和自己是好友则显示“聊天”，否则显示“加好友”】

3.2 数据设计

在 Android 中，对于数据的处理较为复杂，将 Android 中的 Activity 类分为带对应 Adapter 类的界面类和没有对应 Adapter 类的界面类。其中 Adapter 类是对该界面类中的 RecycleView 控件中所显示的“表项”处理的类，通过 Adapter 类可以对 RecycleView 中显示的各“表项”的界面进行处理。所有的界面都有 MyServiceConn 和 ConnectReceiver 内部类，这两个内部类用于获取绑定服务和接收服务发出的广播。远程服务端发来的数据首先被 MyService 中的方法接收，接收后会对数据包进行预处理，将不同的数据打上标签后通过广播发给绑定了广播的各个类，各个类接到广播后会对数据包的标签进行筛选，留下需要自己处理的数据并进行处理。若是需要对“表项”更新的数据，则通过对应的 Adapter 类中的 MyHolder 进行“表项”界面的更新；若是需要对自己界面进行更新，则打包成一个消息，交由内部类 MyHandler 处理并更新界面。下图简要展示了不同的 Activity 类对应数据的处理。

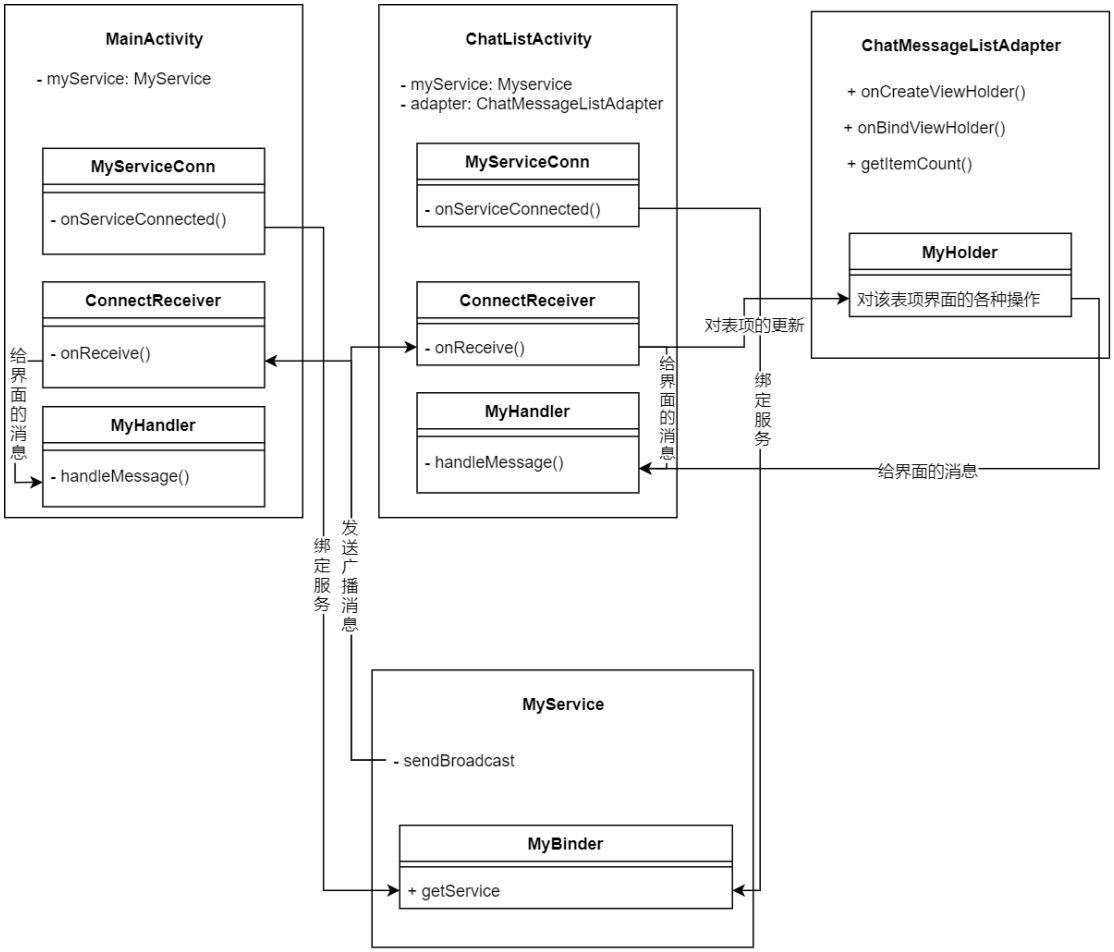

图 26 Android 中的领域类图

3.3 核心功能设计

3.3.1 用户密码的加密及验证功能

用户密码是用户最为隐私的数据，在用户注册时输入的密码，首先获取一个 UUID（ Universally Unique Identifier 通用唯一识别码），UUID 具有唯一性，重复率接近于零。UUID 在密码中相当于“盐”。获取到 UUID 之后将 UUID 拼接在用户输入的原始密码后，生成一个新的字符串，最后将该字符串使用 SHA256 算法生成一个散列值，该散列值几乎不可能被逆向解密出原始数据。假设用户的密码极为简单，但由于在散列化前在密码上拼接了一串 UUID，这也使得别人无法通过彩虹表（存储了大部分简单密码和对应散列值的表）逆向出散列化前的数

据。

以下是加密用户密码过程的流程图。

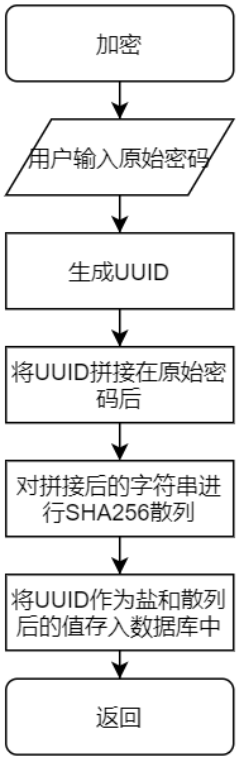

图 27 加密密码流程图

当用户登录验证密码时，首先从数据库获取到用户的密码（散列值）和盐，再将盐拼接到用户输入的密码后利用 SHA256 进行散列操作，对散列后的值和数据库中存储的散列值进行比较，若一致则说明密码正确，若不一致则密码不正确。

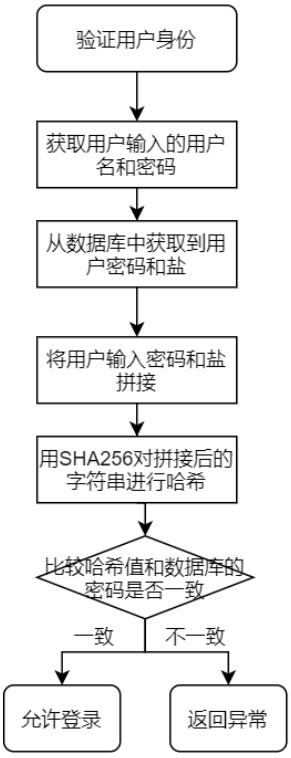

图 28 验证用户身份流程图

3.3.2 服务端接收用户消息功能

服务端在接收到 socket 请求后会将 socket 连接交给连接处理线程进行消息的收发操作。在接收消息的线程，主要是对接收的包进行初步的解包，所有的消息都是以 Mail 类进行传输，在接到消息后将其反序列化成 Mail 实例，然后判断 Mail 实例中消息的类型，对于不同的类型进行不同的操作，如果是用户信息就判定此次消息为用户上线的请求，若是普通的消息信息就继续解包，将消息转为 UserMessage 实例，读出其目标用户 ID，再将该包 push 到对应用户的消息队列中，等待该用户上线后被其发送消息线程处理。

以下是服务端消息接收线程流程图。

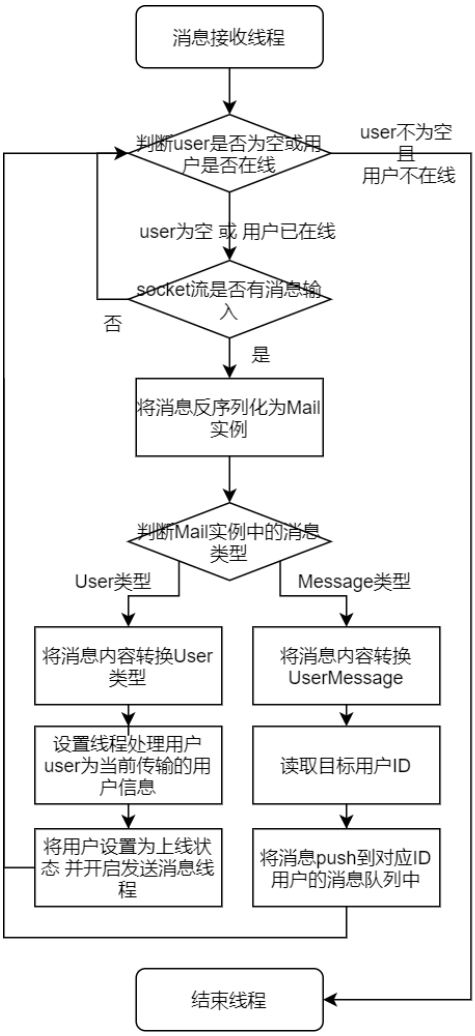

图 29 服务端消息接收线程流程图

3.3.3 服务端发送用户消息功能

当用户上线后就会开启发送用户消息的线程，该线程会循环判断用户是否在线，若在线则询问该用户的消息队列是否为空，若不为空则会获取消息队列中第一个消息，然后封装成 Mail 类交由 Socket 发送给客户端，最后再将该消息从用户的消息队列中删去。

具体流程图如下。

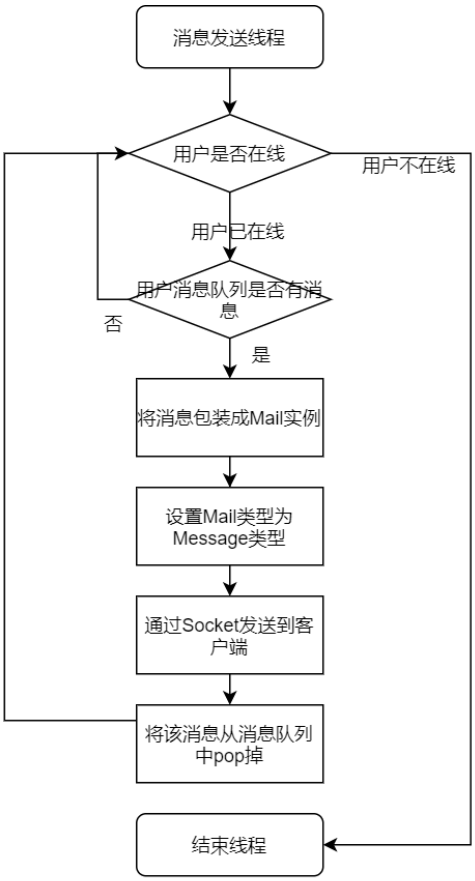

图 30 服务端发送消息线程流程图

4 系统实现

4.1 编码选择

4.1.1 界面绘制语言

MeChat 客户端是基于 Android 实现的，考虑到自身能力以及学习时间成本，采用 Android 原生的界面绘制，即用 xml 绘制 Android 界面。界面绘制、控件配色以及控件选择整体符合 Material Design 的设计理念。

4.1.2 客户端后台编写语言

Android 客户端的后台可以选择 Java 和 Kotlin 两种语言，但是考虑到自身学习能力和网络上关于 Android 的参考资料中所使用的语言，选择了较为主流的 Java 作为后台编写的主要语言。Java 语言具有良好的平台移植性，可以利用 JDBC 对数据库进行操作，利用其包含的 net 包进行 Socket 网络通信，可以实现所有的功能需求。

4.1.3 服务端编写语言

为了与 Android 客户端通信更加方便，服务端同样采用 Java 进行编码，服务端也采用 Java 编码的好处在于，与客户端通信时可以利用 Java 的序列化操作，将通信的内容封装成一个类，发送时将该类进行序列化，在接收时进行反序列化即可很方便的获取到通信的内容。而且 Java 中的多线程操作可以很好的解决多客户端并发访问问题，在服务端利用 Java 的线程池，可以将每一个 socket 通信放入线程池进行多线程通信。

4.2 软件测试

4.2.1 注册测试

用户在注册时的输入有：用户名、昵称、性别、密码和确认密码。其中用户名要求为数字或字母，性别不能为空，密码至少有一个数字和字母，确认密码和密码要一致，每一项输入均不能超过 20 字符。对此划分等价类如下表。

表 26 注册测试中的等价类划分表

输入	有效等价类	无效等价类
用户名	1 数字 2 字母 3 数字和字母	4 出现中文 5 出现表情符号 6 空白 7 超出 20 字符 8 SQL 语句
昵称	9 数字 10 字母 11 汉字 12 数字字母和汉字	13 出现表情符号 14 空白 15 超出 20 字符 16 SQL 语句
性别	17 male 18 female 19 保密	20 不选 21 选择多项
密码	22 数字和字母	23 只有数字 24 只有字母 25 为空 26 超出 20 字符
确认密码	27 和密码相同	28 和密码不同

进行黑盒测试，预期结果如下表

表 27 注册用户的黑盒测试预期结果表

方案	用户名	昵称	性别	密码	确认密码	输出	选取序号
1	11	22	male	c1	c1	有效	1 9 17 22 27
2	xcy	xcy	female	c1	c1	有效	2 10 18 22 27
3	xcy2001	徐诚远	保密	c1	c1	有效	3 11 19 22 27
4	11	徐诚远 2001	保密	c1	c1	有效	3 12 19 22 27
5	徐诚远	😅	[不选]	11	11	无效	4 13 20 23 27
6	😅	[留空]	male female	cz	11	无效	5 14 21 24 28
7	[留空]	123456789123456789123	[不选]	[留空]	11	无效	6 15 20 25 28
8	123456789123456789123	‘DROP TABLE Users’	male	12345678a123456789123	11	无效	7 16 17 26 28
9	‘DROP TABLE Users’	111	male	c1	z1	无效	8 11 17 22 28

4.2.2 运行测试

下面是注册和登录的运行实测图，注册成功和登录成功时会显示用户名称和uid，失败时会提示错误信息

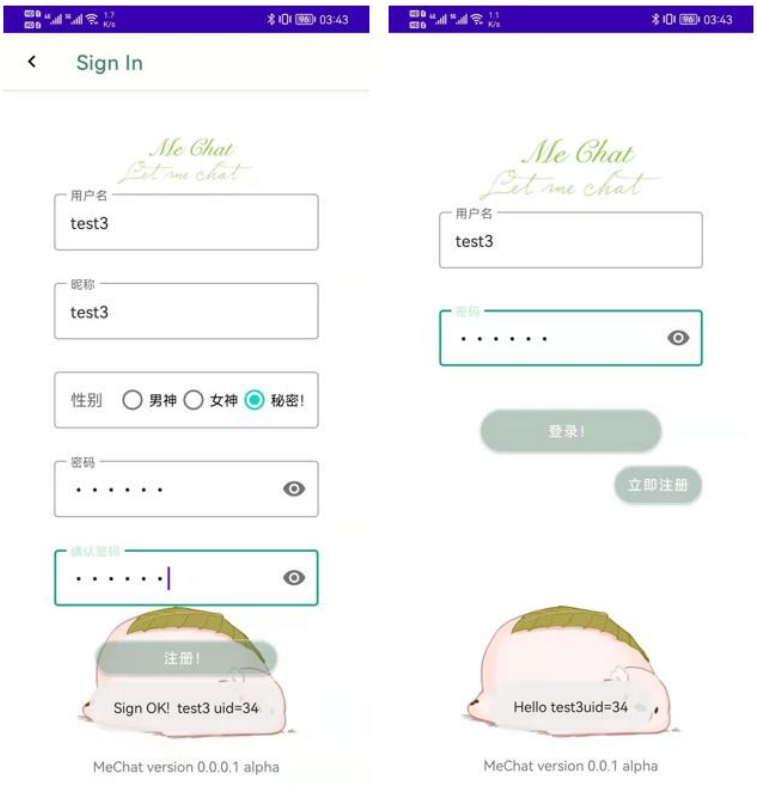

图 31 注册和登录成功的运行实测图

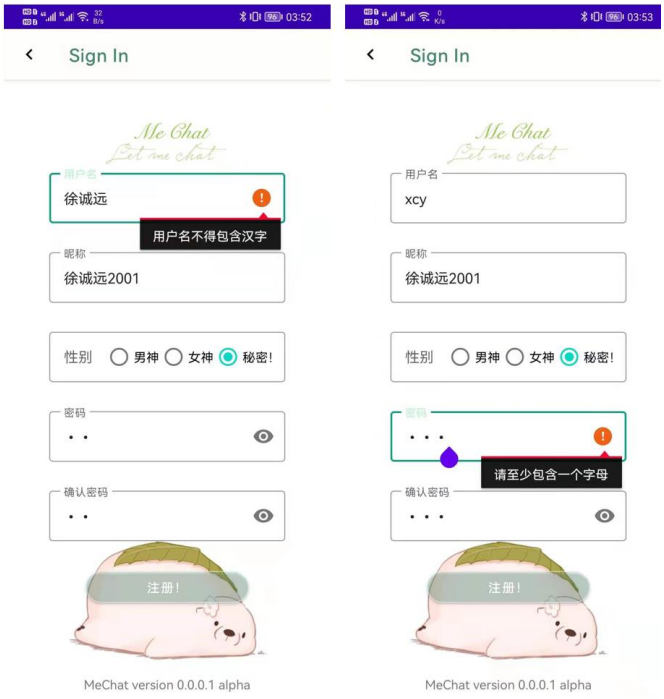

图 32 用户注册时的输入检测运行实测图

下面是搜索用户的运行实测图，若该用户已经是好友，则会显示删除和聊天的按钮，否则显示加好友的按钮。

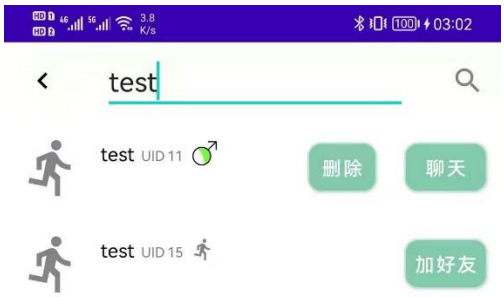

图 33 搜索用户的运行实测图

下面是用户通讯录的运行实测图。

图 34 打开通讯录的运行实测图

下面是聊天列表和进入聊天的运行实测图。

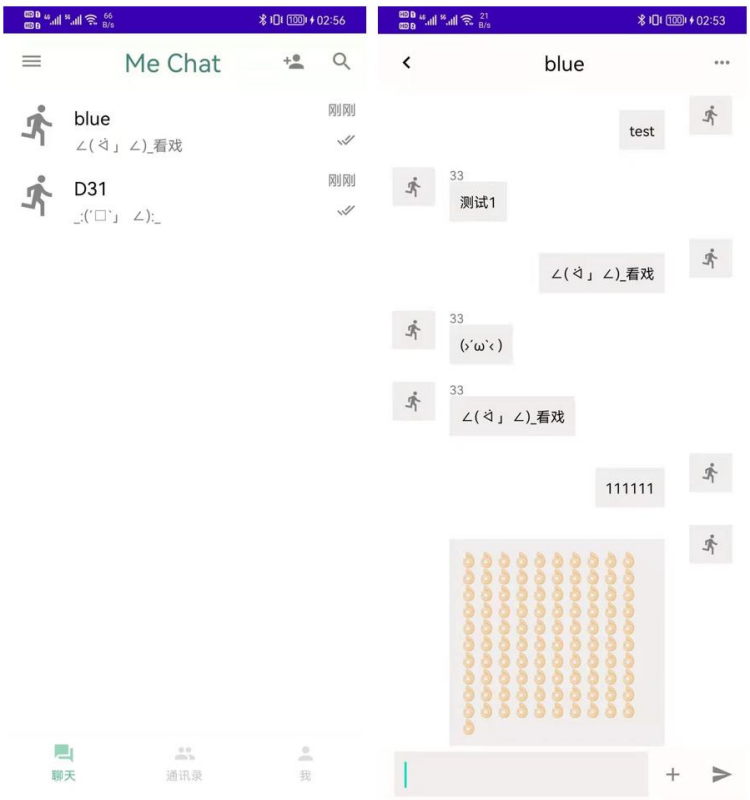

图 35 聊天列表和进入聊天的运行实测图

下面是群聊的运行实测图，最左为电脑端 Android 模拟器，中上为控制台运行的客户端，中下为 cmd 通过 ssh 连接的远程服务器上运行的服务端程序，最右为真机运行的截图。

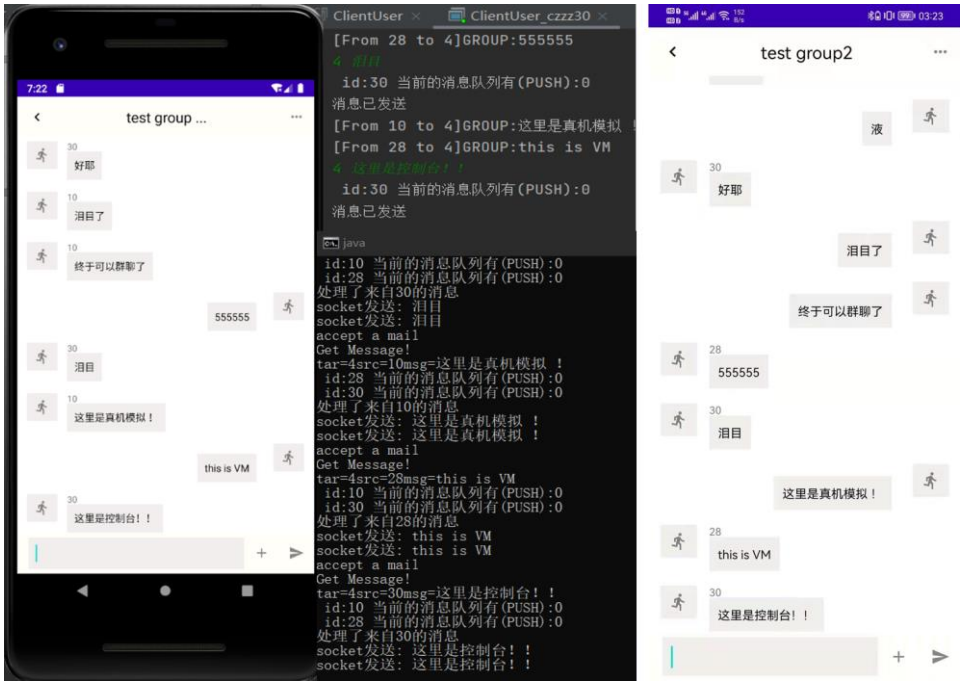

图 36 群聊运行实测图

下面是在单聊中与机器人聊天的开关灯（蓝灯为控制的灯，红灯为网络连接指示灯，连接成功后红灯常亮）指令测试图。

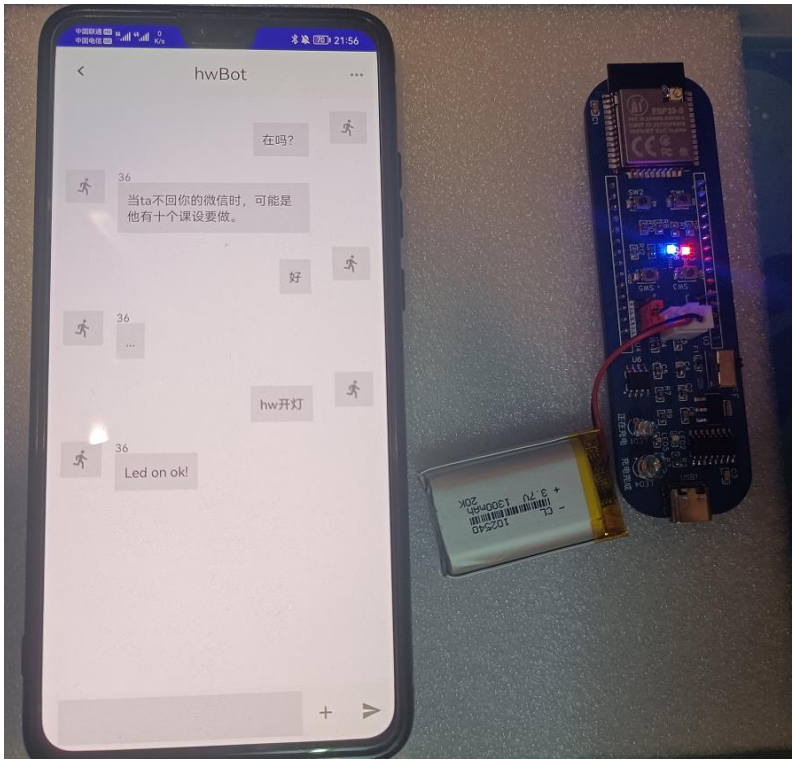

图 37 单聊机器人开灯实测图

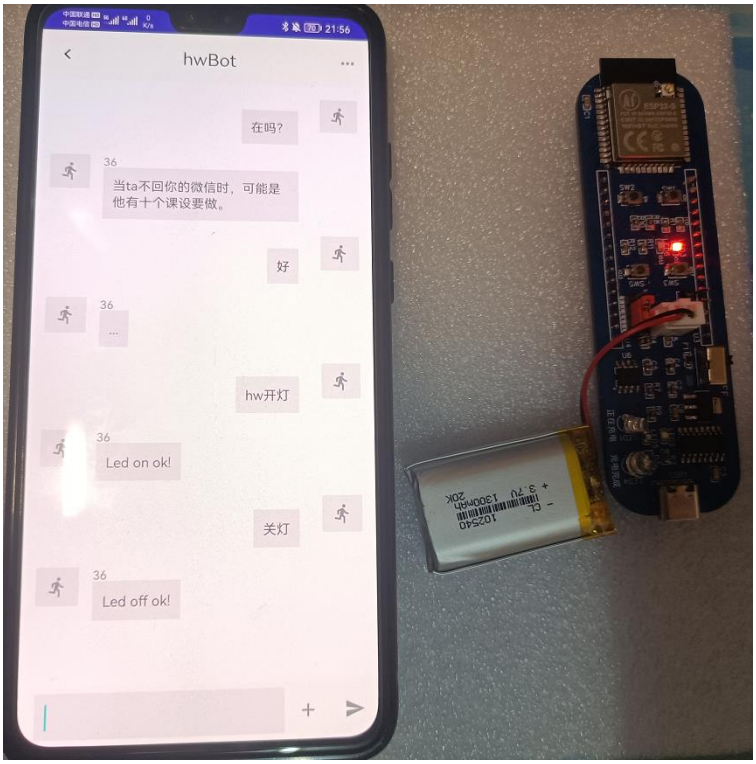

图 38 单聊机器人关灯实测图

下面是群聊中对机器人的测试。28 号和 10 号都是真人用户，36 号是机器人用户，会根据聊天消息进行关键词匹配的自动消息应答。

图 39 群聊中机器人的自动回复测试

5 总结展望

在本次课程设计中，MeChat 主要实现了注册登录、添加好友进行聊天的主要功能，进行了一次较为完整的从需求分析到设计到实现的软件开发步骤。但是由于是个人开发，对 Android 也不是足够了解，在设计实现的过程中也是边学边做，很多事情考虑的还是不够周到。比如在概要设计中的系统构件就不是很合理，按照本次的系统设计，需要将数据库的用户密码存入 Android 客户端，这对数据库安全性的保障造成了很大影响，以后在设计的时候应该将与数据库交互的部分放在服务端，每一次的交互都通过服务端进行。或者在后续维护开发的时候可以在与数据库交互前先连接服务端获取数据库的用户密码，每次动态的获取，且传输的时候按照一定规则进行加密，可以一定程度上保证数据库的安全。

由于时间关系，在一周多的时间内明确需求，进行分析，再学习 Android 的界面绘制以及后台服务广播的各项细节，工期安排的实在是很满，还有的一些细小的地方预留了按键 UI 和函数接口以供后续的再次开发。

MeChat 能继续扩展开发的功能还有很多，服务端的程序也需要进行优化和维护，后续希望能够解决数据库用户名和密码本地存储的问题，再完善更多的群聊单聊功能，比如支持发送图片文件等类型的消息。对于自动回复机器人也有很多改进的地方，现阶段的机器人还只是停留在对消息中关键词的提取，然后进行自动回复，自动回复的内容来源于舍友的“经典语录”，以后可以对消息进行脱敏采集，然后将所有使用者的消息作为样本，利用人工智能的相关算法实现一个真正能够模仿人的机器人做自动回复的聊天功能。总之可以扩展的方向还有很多，还需要继续努力。

6 参考文献

[1] 张海藩，牟永敏. 软件工程导论（第 6 版）[M] ．北京：清华大学出版社, 2013.

[2] 王珊，萨师煊．数据库系统概论[M] ．北京：高等教育出版社，2014：

[3] 赵锐，李卫华．Java 技术及应用（第 2 版）[M] ．北京：清华大学出版社, 2017.

[4] 张思民. Android Studio 应用程序设计（第二版）[M] ．北京：清华大学出版社，2017。

[5] MassimoBanzi, MichaelShiloh，程晨（译）．爱上 Arduino.第 3 版[M]. 人民邮电出版社, 2016.