

# 高等教育自学考试

**毕业论文（设计）**

**基于Spring Cloud的社交聊天系统的**

**设计与实现**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **办学单位：** | **华南理工大学** | | | | | |
| **届** ●**专业：** | **2023届计算机科学与技术** | | | | | |
| **学 生：** | **孙鑫 （010621102593）** | | | | | |
| **指导教师：** | **王亮明 中级** | | | | | |
| **提交日期：** | **2023** | **年** | **3** | **月** | **1** | **日** |

# 摘 要

随着信息技术的不断发展，互联网得以广泛应用，人们的社交关系逐渐由线下转向线上，如何构建一套即时通讯的陌生人社交聊天系统成为当下的热点。在这种背景之下，开发一套简易的，标准的社交聊天系统具有重大意义。其功能至少需要囊括用户注册登录、聊天、建立关系、发布动态等。基于以上要点，结合当下流行、完善、社区活跃的Spring Cloud框架，设计并实现一套使用Java语言开发的社交聊天系统。

依据软件开发的流程，首先结合实际的社交聊天的应用范围，对社交聊天系统进行功能分析和非功能需求分析。然后对社交聊天系统进行设计，包括移动终端、服务端、数据库的三层C/S总体架构设计、前端设计、后端设计、前后端接口设计、数据库设计。再根据系统设计，利用开发工具对系统各项功能进行开发，从而实现功能。最后演示系统核心功能，进行系统结构测试、功能测试、兼容性测试和性能测试。

社交聊天系统的测试结果表示，该系统不仅满足基本的功能需求，也满足功能模块的可用性和稳定性。该系统可以为用户提供一个安全的，稳定的社交环境，对互联网产业起到促进作用。

**关键词：**陌生人社交；即时通讯；Spring Cloud

# Abstract

With the continuous development of information technology, the internet has been widely used, and people’s social relations have shifted from offline to online gradually. How to build a social chat system for strangers with instant messaging has become a hot topic at present. In this context, it is of great significance to develop a simple and standard social chat system. Its functions need to include at least user registration and login, chat, relationship building, post blog, etc. Based on the above points, combined with the current popular, perfect and active community Spring Cloud framework, design and implement a series of social chat system developed in Java language.

According to the process of software development, first of all, combined with the actual scope of application of social chatting, the functional analysis and non-functional requirements analysis of social chatting system are carried out. Then design the social chat system, including the three-tier C/S overall architecture design of the mobile terminal, server, and database, front-end design, back-end design, front-end and back-end interface design, and database design. Then according to the system design, use the development tools to develop the various functions of the system, so as to realize the functions. Finally, the core functions of the system are demonstrated, and the system structure test, function test, compatibility test and performance test are carried out.

The test results of the social chat system show that the system not only meets the basic functional requirements, but also meets the usability and stability of the functional modules. The system can provide users with a safe and stable social environment and promote the Internet industry.

**Key Words:** social for strangers；instant messaging；Spring Cloud

**目 录**

[摘 要 I](#_Toc5326)

[Abstract II](#_Toc15665)

[第一章 绪论 5](#_Toc20266)

[1.1研究的背景 5](#_Toc30727)

[1.2研究的意义 5](#_Toc8326)

[1.3国内外研究现状 5](#_Toc16250)

[1.4论文研究内容 6](#_Toc21558)

[1.5研究方法 6](#_Toc20580)

[第二章 相关理论与关键技术 7](#_Toc16166)

[2.1 Spring-Cloud微服务 7](#_Toc20913)

[2.1.1 注册中心Spring-Cloud-Eureka 7](#_Toc9100)

[2.1.2 服务网关Spring-Cloud-Gateway 7](#_Toc24499)

[2.2 Netty通讯模型 8](#_Toc27529)

[2.3 Redis缓存 8](#_Toc18131)

[2.4 Neo4j图数据库 8](#_Toc14723)

[2.5 Rabbit MQ消息队列 9](#_Toc4346)

[2.6 本章小结 9](#_Toc1135)

[第三章 聊天系统需求分析与概要设计 10](#_Toc25394)

[3.1业务需求分析 10](#_Toc26482)

[3.1.1用例关系 10](#_Toc21830)

[3.1.1登录鉴权的需求分析 10](#_Toc18302)

[3.1.2用户及好友关系的需求分析 11](#_Toc31597)

[3.1.3用户动态的需求分析 12](#_Toc18128)

[3.1.4聊天模块的需求分析 12](#_Toc23106)

[3.1.5内容审核的需求分析 13](#_Toc32134)

[3.2非功能性需求分析 13](#_Toc32670)

[3.3本章小结 13](#_Toc26522)

[第四章 社交聊天系统的设计 14](#_Toc3848)

[4.1微服务架构设计 14](#_Toc15369)

[4.2前端设计 14](#_Toc18133)

[4.3后端设计 15](#_Toc15861)

[4.3.1用户注册流程 16](#_Toc27447)

[4.3.2用户登录流程 16](#_Toc24571)

[4.3.3发布动态流程 18](#_Toc11974)

[4.3.4关注好友流程 19](#_Toc4885)

[4.3.5接入即时通讯流程 20](#_Toc11334)

[4.3.6内容审核流程 21](#_Toc9670)

[4.4接口设计 22](#_Toc25534)

[4.5数据库设计 28](#_Toc32093)

[4.5.1数据库E-R模型 28](#_Toc32024)

[4.5.2数据库表设计 29](#_Toc657)

[4.6本章小结 32](#_Toc25820)

[第五章 社交聊天系统的实现 33](#_Toc9160)

[5.1前端实现 33](#_Toc15312)

[5.1.1目录与文件结构 33](#_Toc8898)

[5.1.2核心功能实现 33](#_Toc6269)

[5.2后端实现 43](#_Toc20615)

[5.2.1概述 43](#_Toc20966)

[5.2.2用户注册的实现 44](#_Toc12737)

[5.2.3用户登录的实现 44](#_Toc1272)

[5.2.4发布动态的实现 45](#_Toc5868)

[5.2.5关注好友的实现 45](#_Toc21115)

[5.2.6即时通讯的实现 45](#_Toc20223)

[5.2.7内容审核的实现 46](#_Toc396)

[5.3系统部署运行 47](#_Toc4230)

[5.4本章小结 47](#_Toc21225)

[第六章 社交聊天系统的测试 48](#_Toc21560)

[6.1测试环境 48](#_Toc29590)

[6.2接口测试 48](#_Toc30094)

[6.3功能测试 54](#_Toc12703)

[6.4兼容性测试 57](#_Toc20725)

[6.5性能测试 57](#_Toc27655)

[结 论 61](#_Toc5249)

[参考文献 62](#_Toc16172)

[致 谢 63](#_Toc24404)

# 

# 第一章 绪论

### 1.1研究的背景

随着技术的不断发展，移动终端的普及，移动终端已经成为人们生活必备的工具。社交离不开沟通，人们的社交习惯逐渐从线下交流逐步转为线上交流。无论在生活还是在工作中，社交软件已成为人们重要的沟通工具。而陌生人社交软件是人们扩宽社交面，满足日常社交需求的重要工具。

而一方面随着互联网产业不断扩大，熟人社交软件市场变得萎缩，常用的熟人社交软件处于寡头公司垄断状态，另一方面人们对于隐私的重视程度日益增长，寡头公司对个人信息的控制收到抵制，使得重视隐私数据的小公司的社交聊天软件具有一定的市场，同时市场也需要更多元化的社交软件满足不同属性的社交需求。

由此看来，一系列新的陌生人社交软件成为众望所归。此系统的研究与实现具有广泛的现实意义与应用价值。用户通过移动设备发布自己的生活动态，获得与自己契合的人的关注，通过聊天交流加强社交连接，从而与陌生人产生互动，建立社交关系，满足日常社交需求。

## 1.2研究的意义

为了满足人们对陌生人社交的需求，提供高效的安全的聊天环境，本课题开发的社交聊天系统可以有效地满足这类需求。通过本系统，用户可以与陌生人建立社交连接，打破陌生人社交壁垒，提高了用户的使用体验，降低了人与人的社交成本，促进了互联网产业的发展。

### 1.3国内外研究现状

目前，国内外都有很多关于社交聊天应用的研究。

国内方面，一些研究关注于社交聊天应用的使用行为。例如，一些研究探讨了社交聊天应用在大学生中的使用情况和影响因素。其他研究则关注于社交聊天应用在企业中的使用情况和影响。国外方面，研究聚焦于社交聊天应用的多样性和使用情况。一些研究调查了不同国家和地区的人们使用不同社交聊天应用的趋势和偏好。还有一些研究关注于社交聊天应用的设计和用户体验，旨在提高用户满意度和使用效率。

此外，一些研究还探讨了社交聊天应用在个人和社会方面的影响。例如，有研究表明，社交聊天应用可能对人们的社交互动和沟通方式产生影响，同时也可能对人们的心理健康产生影响。

总的来说，社交聊天应用是一个非常活跃的研究领域，国内外都有很多研究。未来，研究人员可以进一步探讨社交聊天应用在不同领域的应用和影响，以及如何提高社交聊天应用的用户体验和效率。

### 1.4论文研究内容

通过各种学术期刊研究文献的收集与阅读，以及社会实践相结合，分析社交聊天系统的背景与意义分析，因此针对社交关系复杂，即时通讯难以构建的问题，本文研究了基于 Spring Cloud的社交聊天系统如何设计与实现，具体研究如下:

首先确里在系统中的所使用的各种技术框架的使用，其中以Spring Cloud作为后端 Web框架，使用Redis, Neo4j, Rabbit MQ为辅助的技术路线，利用软件工程设计中的方法，首先对社交聊天系统进行需求分析，确定功能模块的需求，其次进行详细的系统架构设计，前后端交互设计、API接口设计、数据库设计等，完成整体设计后，进行各功能模块的代码实现，最后对系统进行测试，实现基于Spring Cloud的社交聊天系统。

### 1.5研究方法

通过文献归纳与社会实践相结合的方法，通过对国内各种学术期刊等研究文献的收集与研读，以当前前沿的技术为前提，构建一套实用的架构。

# 第二章 相关理论与关键技术

## 2.1 Spring-Cloud微服务

传统的Web服务架构，由多位开发工程师共同开发同一个项目，当业务系统愈发庞大，代码量也变得巨大，开发过程变得沉重。微服务是一种新兴的软件架构模式，它把一个大型的单体应用或服务拆分为多个支持微服务[2]，每个微服务按照各自的功能划分开来，形成由多个责任单一的服务共同组成的一套服务，从而使开发变得灵活。

Spring Cloud是Java生态里微服务架构的实现之一，它利用Spring Boot开发便利性巧妙地简化了分布式系统基础设施的开发，它的组成部分有服务发现、配置中心、消息总线、负载均衡、断路器、数据监控等。

## 2.1.1 注册中心Spring-Cloud-Eureka

Spring Cloud Eureka是Spring Cloud中服务发现的一种实现，它包含了Server 端和Client端。Eureka Server 提供服务注册功能，其余服务节点启动完成后，会向Eureka Server发送注册请求，并获取其他服务的信息，利用心跳机制，每隔一段时间向Eureka Server发送心跳包更新健康状态，同时获取最新的其他服务的注册信息。过程中若有其他服务接入，则要等到下一次心跳便可获取到新的服务信息。

## 2.1.2 服务网关Spring-Cloud-Gateway

在微服务架构中，所有的请求均由客户端向网关发起，由网关服务利用特定规则，将请求数据转发至业务微服务节点。常见的有 Zuul、Spring-Cloud-Gateway、Kong。其中，Spring-Cloud-Gateway因为使用异步非阻塞IO模型，并且支持异步，故而性能要高于其他网关实现。

## 2.2 Netty通讯模型

Netty是一个异步的、基于事件驱动的网络应用框架，用以快速开发高性能、高可靠性的网络IO程序，简化和流程化了[NIO](https://so.csdn.net/so/search?q=NIO&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_36389060/article/details/_blank)的开发过程。Java NIO模型是同步非阻塞的，服务器实现模式为一个线程处理多个请求(连接)，即客户端发送的连接请求都会注册到多路复用器上，多路复用器轮询到连接有I/O请求就进行处理。



图 2-1 NIO模型

## 2.3 Redis缓存

Redis是一款开源的、网络调用的、基于内存的、原子性的、可持久化的Key-Value存储系统。它依据Key将数据结构存储在内存中，能够存储的数据结构有String, List(ziplist), Hash(Key-Value), Set(Hash的变种，没有value), ZSet(Value-Score, 根据Score排序的Set)等[7]。

Redis的key使用跳表构建索引，快表是一种接近于二叉树的数据结构，构建与查询的时间复杂度是O(log2N)。

## 2.4 Neo4j图数据库

[Neo4j](https://baike.baidu.com/item/Neo4j" \t "/Users/robertsun/Documents\\x/_blank)是一个高性能的NOSQL图形数据库[9]，其主要数据结构为图，图数据由节点，关系组成。Neo4j中，节点和关系可以有类型，属性，因此可以进行面向对象的。

由于图数据结构比较接近于社交网络的抽象意义，因此使用Neo4j来存储好友关系。

## 2.5 Rabbit MQ消息队列

消息队列是一种存储中间件，它将消息生产端发送的消息存起来，再发送给消息消费端进行消费处理。队列具有FIFO（先进先出）的特点。消息队列可以将消息的生产者和消息的消费者隔离开来，从而降低耦合性。

Rabbit MQ是一个在AMQP（Advanced Message Queuing Protocol）基础上实现的[8]，可复用的企业消息系统。

## 2.6 本章小结

本章简要阐述了该系统所选择的技术组成，后端使用 Spring Cloud作为 Web架构，主要用到了Spring-Cloud-Eureka作为注册中心，Spring-Cloud-Gateway作为服务网关，使用Netty作为即时通讯服务。使用Redis作为缓存数据库，Neo4j用于处理好友关系，Rabbit MQ作为消息队列，便于接下来的系统设计与实现。

# 第三章 聊天系统需求分析与概要设计

## 3.1业务需求分析

社交聊天系统应具有用户登录功能，社交关系功能，动态发布功能，好友聊天功能，内容审核功能，以及其他非功能性需求。

## 3.1.1用例关系

图3-1是本系统的用例图，内容是讲解不同的用户，与系统功能之间不同的交互方式，其主要讲解的是不同用户可以使用的功能模块。

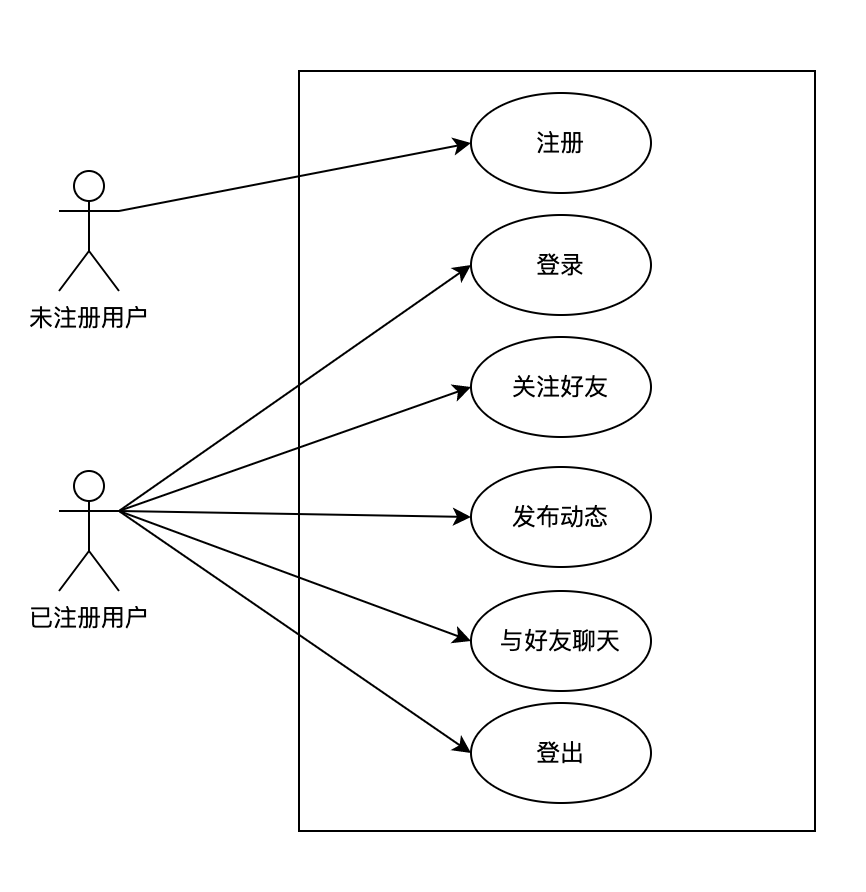


图 3-1 用例图

### 3.1.1登录鉴权的需求分析

用户登录模块要能够支持短信登录，密码登录。用户的权限包括普通用户和会员用户。为了记录用户具有的权限，需要对用户进行一些标记，并且在维持登录状态的同时体现出来。表3-1是用户短信注册用例文档：

表 3-1 用户短信注册

|  |  |
| --- | --- |
| 用例名称 | 用户短信注册 |
| 执行者 | 未注册用户 |
| 前置条件 | 所用手机号未在系统中注册 |
| 后置条件 | 填写用户信息 |
| 基本路径 | 1. 点击登录页面的注册按钮 2. 输入手机号，点击获取验证码 3. 输入验证码 4. 成功后进入用户信息界面 |

表 3-2 用户短信登录

|  |  |
| --- | --- |
| 用例名称 | 用户短信注册 |
| 执行者 | 已注册用户 |
| 前置条件 | 所用手机号已在系统中注册 |
| 后置条件 | 进入主页 |
| 基本路径 | 1. 点击登录页面的注册按钮 2. 输入手机号，点击获取验证码 3. 输入验证码 |

### 3.1.2用户及好友关系的需求分析

在陌生人社交中，应允许单边好友关系，也有双边好友关系，所以存在以下情况:

1. 无关系
2. 单向关注
3. 双向关注

表 3-3 用户关注/取消关注

|  |  |
| --- | --- |
| 用例名称 | 用户关注/取消关注 |
| 执行者 | 注册用户 |
| 前置条件 | 用户未关注/已关注对方，对方未拉黑用户 |
| 后置条件 | 开始聊天 |
| 基本路径 | 1. 点开目标用户主页； 2. 点击“关注”/“已关注”按钮； |

### 3.1.3用户动态的需求分析

用户动态不仅作为信息分享，也是用户的社交名片，应当被展示给好友及陌生人查看。动态的内容通常包括文字，图片，视频。当用户发布一条动态，动态将依据一定规则会发布到公共区域，陌生人可以通过公共区域查看该动态。当用户的粉丝进入关注人主页，返回所有关注人的动态给他。

表 3-5 用户发布动态

|  |  |
| --- | --- |
| 用例名称 | 用户发布动态 |
| 执行者 | 注册用户 |
| 前置条件 | 用户已注册且正常使用 |
| 后置条件 | 开放动态给粉丝或陌生人查看 |
| 基本路径 | 1. 点开动态页面右上角“发布动态”； 2. 输入文字内容，选择最多8张图片或一段视频 3. 点击“发布”按钮； |

### 3.1.4聊天模块的需求分析

在社交聊天系统中，聊天是最重要的一块。聊天是一种互动，具有很强的交互性，实时性要求比较高，当用户A与用户B发送消息，消息因在很短的时间内从A传输到B，所以需要接入实时通讯。过程如下：

1. 客户A向服务器建立连接
2. 客户A利用连接信道向服务器发送消息
3. 服务器找到与客户B的连接，向客户B主动推送客户A发送的消息
4. 如果找不到客户B的连接，则需要将数据存储到数据库中
5. 如果客户B在消息产生时不在连接状态，等到下一次连接到服务器时，服务器主动推送所有未读消息

### 3.1.5内容审核的需求分析

作为一款用户年龄跨度较大，人群受教育水平参差不齐的聊天系统，用户发布的内容动态、头像、昵称、个性签名具有较大的曝光量，其内容可能会存在不合时宜的情况，所以需要加以内容审查。随着人力成本上升，自动化必将取代人工。在系统中，有来自不同功能的内容需要审核，如何设计一套通用的审核模块是一个重点。

在实际的业务系统中，需要审核的内容有文字，图片，视频及其组合，因而需要将文字、图片、视频作为参数传给审核模块，由审核模块给出最终的结果。

## 3.2非功能性需求分析

本系统面向终端用户，需要具备较高的性能还有稳定性，可以给终端用户提供较为舒适的体验。

性能对于计算机系统来说是一项重要的评价指标，从终端用户的角度看，一款APP必须具备以下指标:

1. 接口响应时间。对于终端使用客户来说，操作等待时间不宜太长。一般来说，高频调用的接口响应时长最长不能超过1秒，0.5秒以下无感，0.3秒和0.5秒相差不大；低频接口响应时长最长不能超过3秒。较长的等待时间会降低用户体验，造成用户流失。
2. 稳定性。稳定性是一个系统在运行过程中是否能够正常地、持续地为用户提供服务，不会出现APP闪退、服务端持续不响应等异常情况。这类情况也会降低用户体验，造成用户流失。因此本系统要求网络正常的情况，在服务器的承载能力以内不出现错误与无响应。

## 3.3本章小结

本章使用用例文档详细阐述功能性需求分析，并且为确保系统的高效与稳定，还对非功能性需求作分析，为接下来的系统设计作骨架支撑。

# 第四章 社交聊天系统的设计

## 4.1微服务架构设计

本论文中的社交聊天系统使用了Spring Cloud中的注册中心和服务网关，组织架构上，由网关调用其他各个服务，为了减少服务之间的调用引起的I/O损耗，服务之间避免使用服务调用，并且连接到同一台数据库进行数据共享。该模式最大的优势是极致的高效运行。

本论文设计的移动医疗系统采用 C/S 结构，C代表移动用户终端，S代表服务器。移动端的原声应用程序具有较高的性能，服务器具备强大的计算能力。结合两者，充分使整个系统得以高效运行。图4-1 是系统架构图：

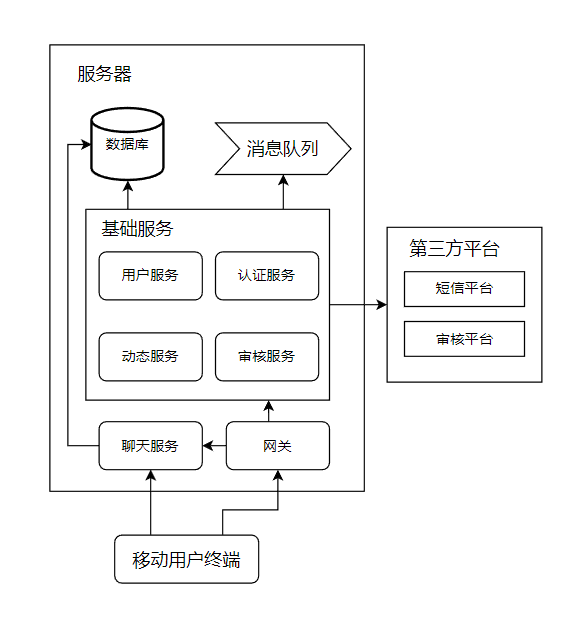


图 4-1 系统架构图

## 4.2前端设计

本论文的社交聊天系统前端设计着重于引导用户发布动态和聊天功能，使用精美、简明的视觉设计，贯彻KISS(Keep It Simple and Stupid)原则，重视用户的体验以及符合社交软件的使用流程。目标在于使用户可以无需花费学习时间即可使用。

用户体验：(1)用户提供手机号获取短信验证码，一旦匹配成功，则自动进入登录，如果客户没有在平台注册，则自动完成注册动作，并通知前端进入信息填写界面，用户无需完成注册。(2)查看和发布动态相关信息，用户通过点击中间的万能按钮，弹出所有需要的功能，就像呼唤助手一样，平时不用的时候可以收纳起来，不会占用过多的屏幕控件。(3)用户聊天，用户点进目标用户主页，看到对方的动态，在有一定了解的前提下，点击关注才能给对方发送消息，可以增进聊天的内容。

本系统前端主要包含以下内容:

1. 注册/登录页面：通过手机号和短信验证码或者手机号和密码方式登录，如果使用手机号和短信验证码登录的用户尚未注册，则自动进行注册后自动登录。
2. 动态广场/关注人动态：查看动态发布人，发布人标签，动态内容。
3. 聊天页面：对方头像，历史聊天记录，表情包，上传图片按钮。

图4-2是本论文系统的界面关系图，主要介绍各系统界面、界面实现的功能以及界面之间的关系。

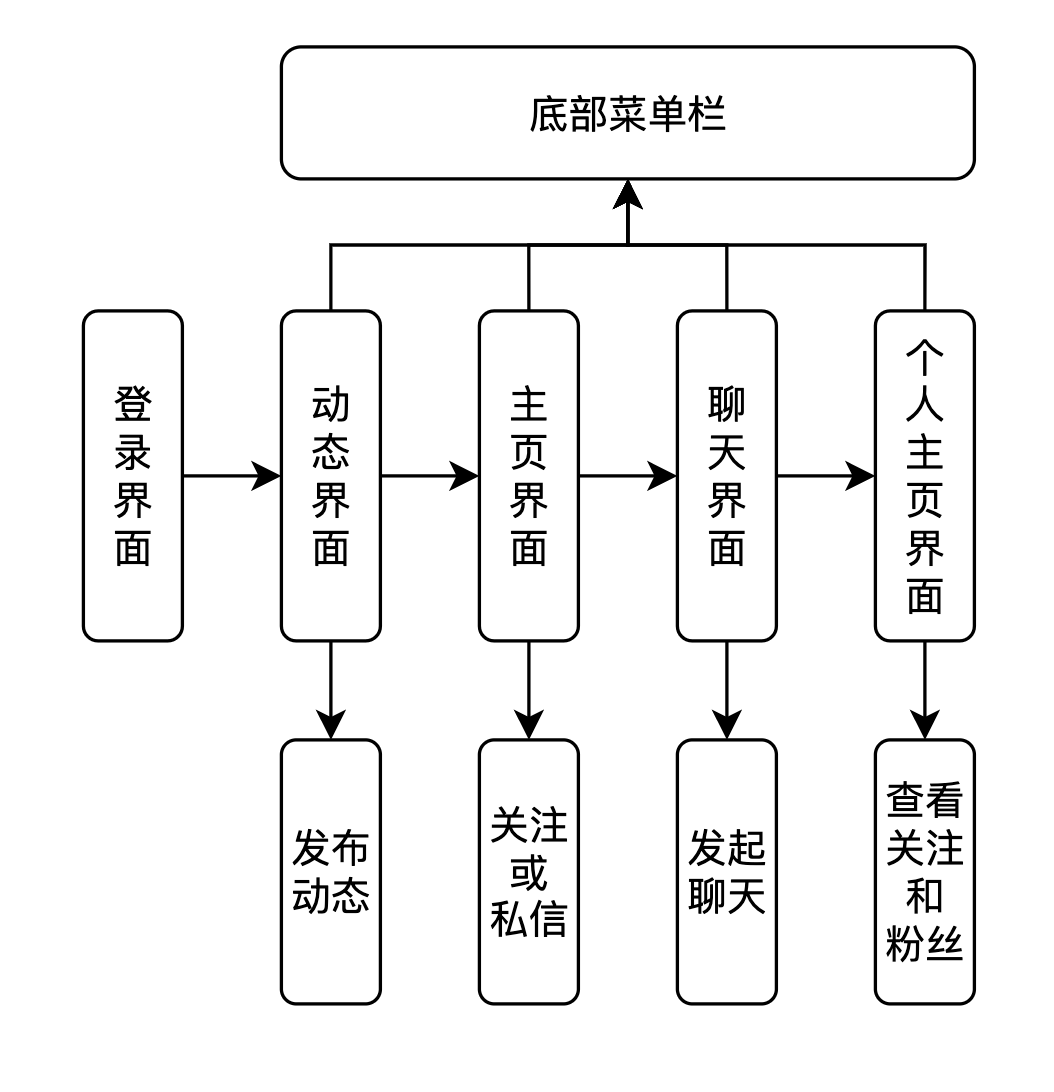


图 4-2 界面关系图

## 4.3后端设计

本小节通过文字阐述、时序图和流程图描述社交聊天系统的业务逻辑。

## 4.3.1用户注册流程

用户打开未登录的APP后，点击右上角的注册按钮，进入注册页面，提供手机号码，登录密码，确认密码，通过POST请求获取短信验证码，后端返回一个MD5加密后的短信验证码，用于前端首次校验，用户输入短信中发送的验证码后，调用POST请求给后端进行校验，成功则进入信息初始化界面，不成功前端提示错误。

用户注册流程时序图如图4-3所示

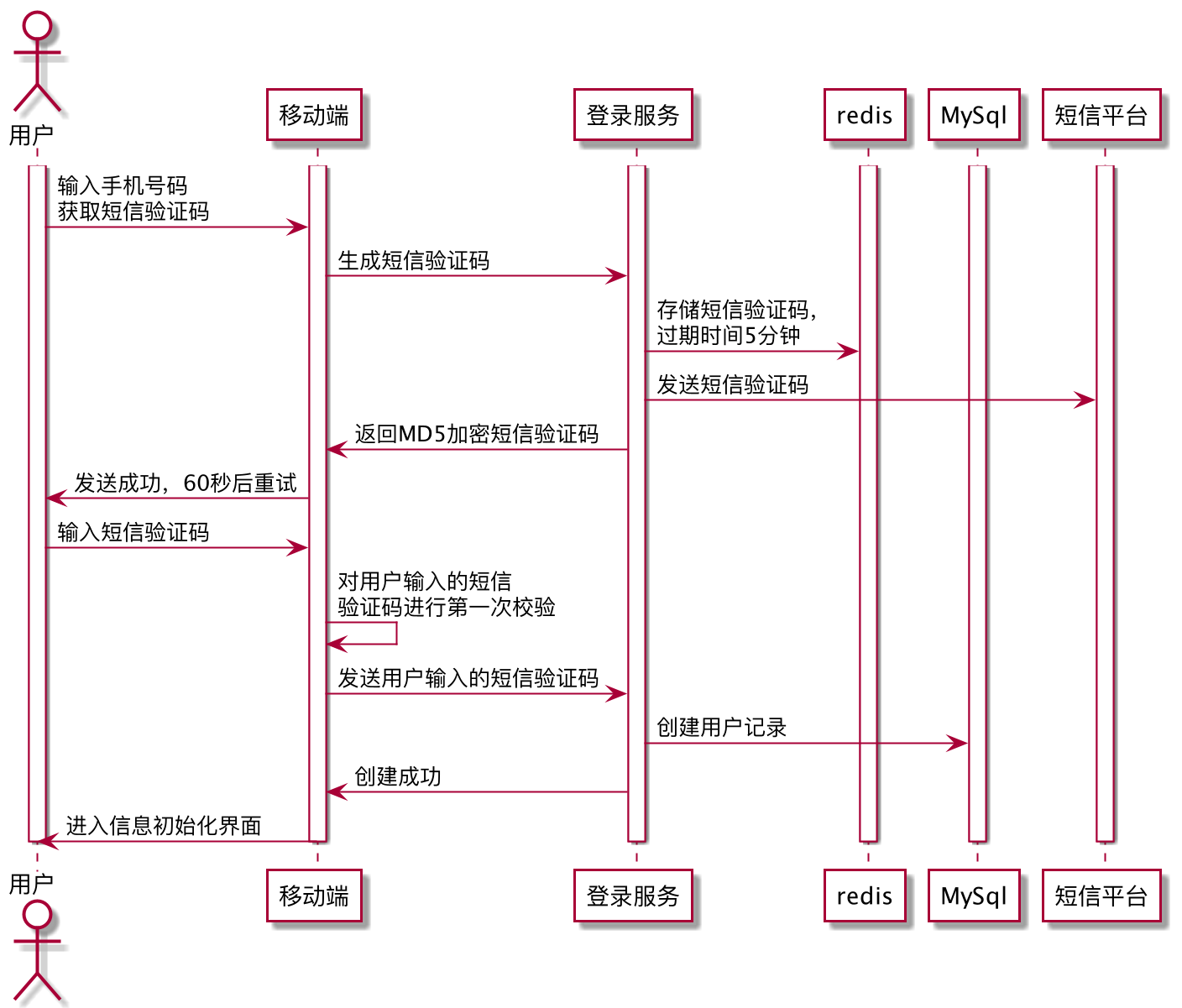


图 4-3 用户注册流程时序图

## 4.3.2用户登录流程

用户打开未登录的APP后，就进入了登录界面，默认使用密码登录，也可以点击密码框做下架的“短信登录”文本切换成短信登录。

1. 密码登录。用户在手机如输入框中填写手机号码，在密码框中填写注册时提供的登录密码，应用程序APP调用POST请求给后端进行校验，成功则进入主页界面，不成功前端提示错误。图4-4为用户密码登录流程时序图。

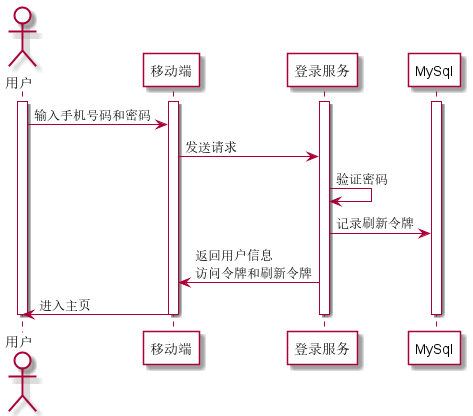


图 4-4 用户密码登录流程时序图

1. 短信登录。在切换成短信登录后，用户在手机如输入框中填写手机号码，点击验证码输入框右侧的获取短信验证码，后端返回一个BCrypt-5（强度为5）加密后的短信验证码，用于前端首次校验，用户输入短信中发送的验证码后，调用POST请求给后端进行校验，成功则进入信息初始化界面，不成功前端提示错误。图4-5为用户短信登录流程时序图。

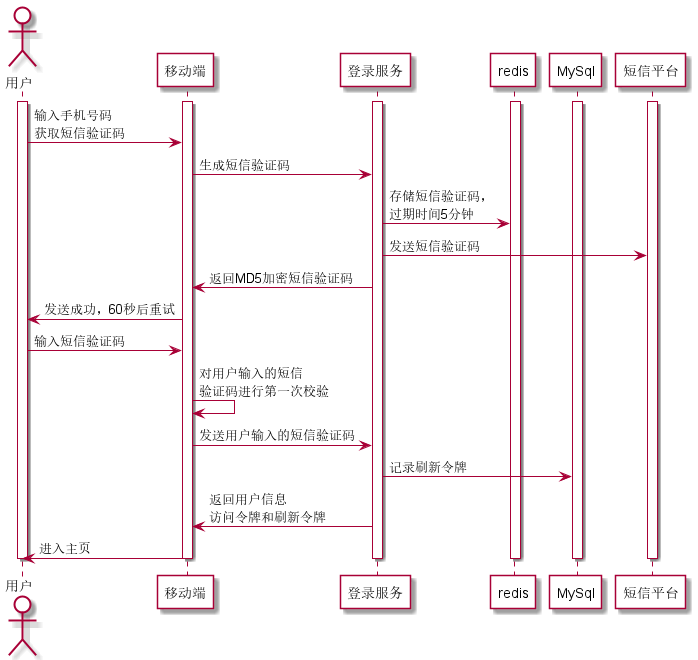


图 4-5 用户短信登录流程时序图

为了记录用户是否具有某些特权，可以记录客户维持登录状态的令牌（Token）。Token可以分为访问令牌（Access Token）和刷新令牌（Refresh Token）。

访问令牌除了可以用户接口的访问调用时，确定身份的依据，还可以用于存储一部分用户信息，用户权限等，访问令牌具有较短有效时间，当访问令牌过期后则失效，为了减少I/O的调用，本系统中的过期时间记录在访问令牌中，而不用从数据库或者Redis中校验访问令牌，所以令牌一旦发放，就不可撤回，直到令牌中存储的过期时间到期。

刷新令牌具有相对长的多的过期时间，用于在访问令牌已经过期或者即将过期的时候，获取新的访问令牌，本系统中，刷新令牌具有全局唯一性，如果在其他设备登录，原先的设备上存储的刷新令牌则会失效，原设备上将会在访问令牌过期后，因无法获取最新的访问令牌而退出登录。

## 4.3.3发布动态流程

用户在登录完成后就会进入动态广场页面，在页面底部中间的功能键，在弹出的功能列表中点击“发动态”按钮，应用跳转到动态发布界面，输入文本，选择图片，选择动态可见范围，可见范围包括：广场可见、关注人可见、仅自己可见，随后点击发布按钮，即可完成动态发布操作。

动态属于需要审核的功能，当动态发布完成后，动态处于仅自己可见状态，当审核通过审核之后转为原先设定的可见范围，若审核不通过，则删除处理，并予以警告。图4-6为用户发布动态流程时序图。

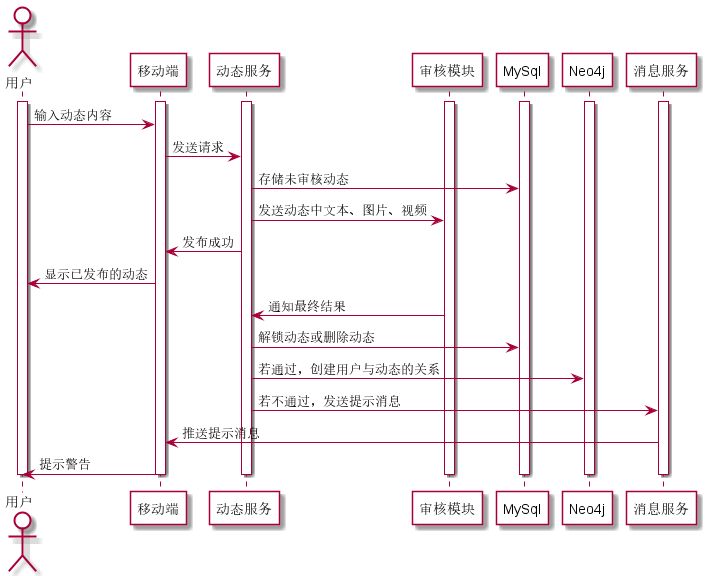


图 4-6 用户发布动态流程时序图

## 4.3.4关注好友流程

用户在广场上见到其他用户后，点击头像进入对方主页，如果未关注对方，则页面下方会出现两个按钮：“关注”和“私聊”，此时点击左下角的关注按钮即可完成关注。如果对方，则下方按钮只有私聊，点击右上角“更多操作”操作按钮，会弹出取消关注按钮，按下即可取消关注。

当点击关注后，当前用户的关注数和目标用户的粉丝数会同时加一，如果对方是已经是自己的粉丝时，双方的好友数会同时加一。相反的，取消关注时，当前用户的关注数和目标用户的粉丝数会同时减一，如果对方是已经是自己的粉丝时，双方的好友数会同时减一。图4-7为用户关注好友流程时序图，图4-5为用户取消关注流程时序图。

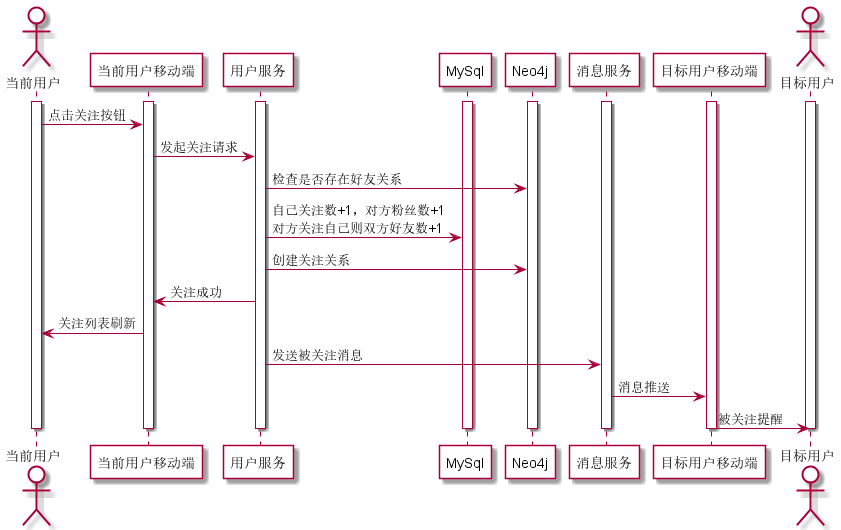


图 4-7 用户关注好友时序图

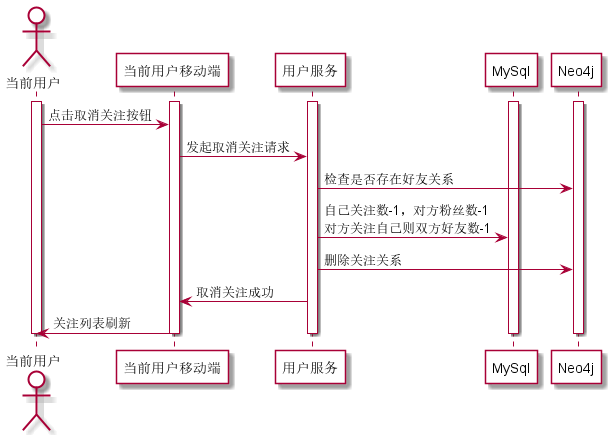


图 4-8 用户取消关注流程时序图

## 4.3.5接入即时通讯流程

当用户成功登录，随后与聊天服务建立TCP连接，使用访问令牌作为身份认证的依据，通过身份认证后，检查是否存在未读的历史消息，若存在，则以TCP下行的方式推送所有未读消息，随后进入监听消息发送消息阶段，期间每隔一段时间发送心跳包[5]，并有不定期的推送消息提醒，一直到应用关闭时通讯终止。图4-9为用户即时通讯接入流程时序图。

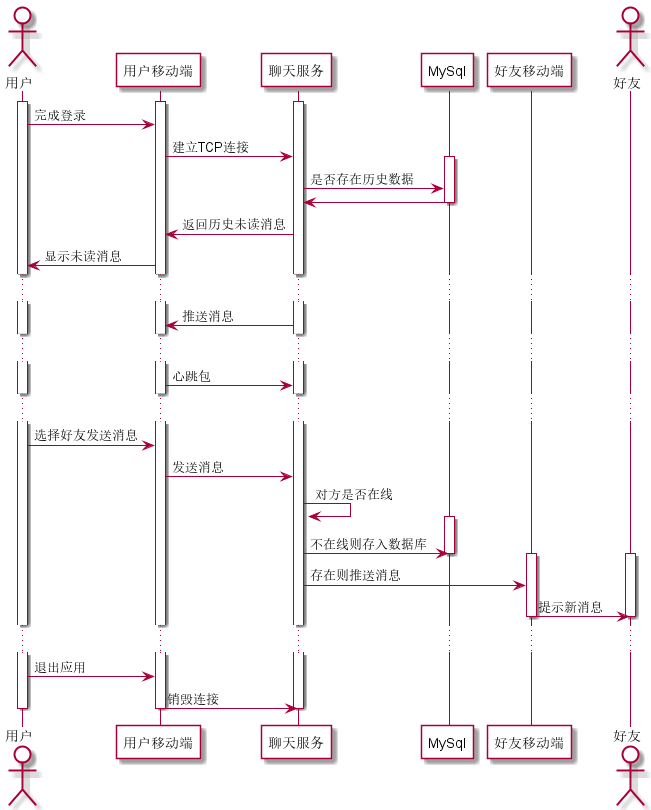


图 4-9 用户即时通讯接入流程时序图

用户所有的聊天信息均存储于客户端内部，用户一旦卸载应用并重装，就会丢失聊天记录。一方面，保护了用户隐私，另一方面，聊天数据是海量的，可以减少服务器数据存储压力。

## 4.3.6内容审核流程

内容审核模块为了与业务解耦，做到责任单一，那么就不能含有任何业务流程，本系统中，使用Rabbit MQ作为业务解耦中间件。

当业务模块需要进行模块审核时，需要提供业务id，回调所需的Rabbit MQ的queue关键字，文本内容，图片地址，视频地址，将如上信息发送给Rabbit MQ。审核模块在接收到消息后，先将所有任务暂时存储到Redis中，并依次对待审核内容发送给第三方审核。其中，文本可以同步完成审核，而图片和视频只能异步。过程中，如果出现任何一次审核不通过的情况，直接通过回调队列返回不通过信息，后续内容将自动忽略。如果全部内容均通过，则通过回调队列返回通过信息。图4-10为内容审核流程时序图。

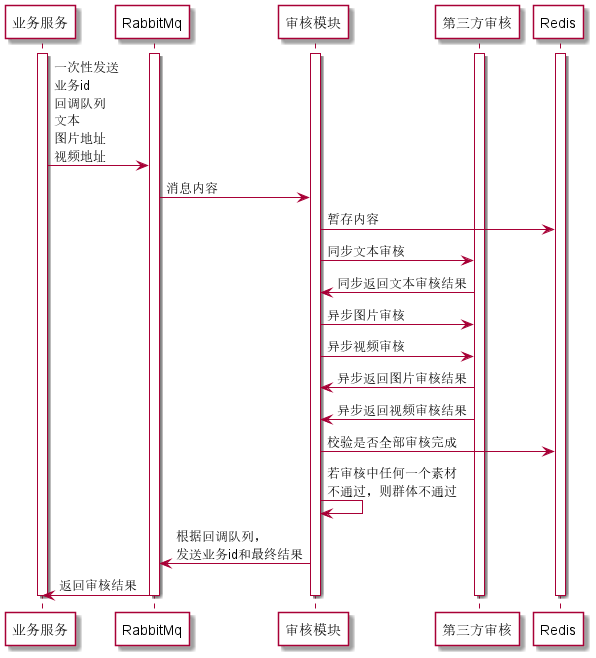


图 4-10 内容审核流程时序图

## 4.4接口设计

本节内容主要讲解系统中所用到的前端调用后端的Restful接口，以及几个关键接口的调用参数和返回值。

本系统中，不需要登录认证的接口有获取短信、注册用户、登录。其余接口需要将访问令牌以Bearer Token的形式传送给后端。Bearer Token指在接口调用时，以“Authorization”为请求头名，在访问令牌前拼上字符串“Bearer ”作为请求头值，后端读取到该头之后，便可获取到用户身份。

表 4-1 API接口总览

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 模块 | 名称 | 请求类型 | 请求URL |
| 1 | 认证模块 | 发送验证码 | POST | /auth/register/sendSms/{sense} |
| 2 | 手机短信登录 | POST | /auth/login/smsCode |
| 3 | 手机号绑定注册 | POST | /auth/register/smsRegister |
| 4 | 密码登录 | POST | /auth/login/password |
| 5 | 用户模块 | 关注/取关用户 | POST | /users/user/follow |
| 6 | 获取关注列表 | POST | /users/user/myFollows |
| 7 | 获取粉丝列表 | POST | /users/user/myFans |
| 8 | 设置好友别名 | POST | /users/user/friendAlias |
| 9 | 动态模块 | 动态广场 | POST | /blog/blog/recommend |
| 10 | 关注人动态 | POST | /blog/blog/follow |
| 11 | 发布动态 | POST | /blog/blog/publish |
| 12 | 删除动态 | POST | /blog/blog/delete |
| 13 | 动态点赞 | POST | /blog/blog/praise |
| 14 | 查看动态评论 | POST | /blog/comment/view |
| 15 | 添加动态评论 | POST | /blog/comment/add |
| 16 | 删除动态评论 | POST | /blog/comment/delete |

以下为本系统中核心接口的详细信息：

1. 短信登录接口

表 4-2 短信登录接口

|  |  |  |  |
| --- | --- | --- | --- |
| 请求URL | /auth/login/smsCode | | |
| 请求类型 | POST | | |
| 参数名 | 必填 | 类型 | 说明 |
| 请求参数 |  | | |
| account | 是 | string | 手机号 |
| smsCode | 是 | string | 短信验证码 |
| 返回参数 |  | | |
| code | 是 | int | 返回码 |
| success | 是 | bool | 是否成功 |
| message | 是 | string | 返回消息 |
| data | 否 | object | 返回数据，失败时为空 |
| data.token | 是 | object | 认证信息 |
| data.token.accessToken | 是 | string | 访问令牌 |
| data.token.expiresIn | 是 | int32 | 多久后过期，单位：秒 |
| data.token.refreshToken | 是 | string | 刷新令牌 |
| data.user | 是 | object | 用户基本信息 |
| data.user.id | 是 | int64 | 用户id |
| data.user.nickname | 是 | string | 用户昵称 |
| data.user.headUrl | 是 | string | 用户头像 |
| data.user.roles | 是 | string:int64 | 用户角色，key为角色名，value为过期时间戳，-1为永久 |

1. 关注/取关用户接口

表 4-3 关注/取消关注用户接口

|  |  |  |  |
| --- | --- | --- | --- |
| 请求URL | /users/user/follow | | |
| 请求类型 | POST | | |
| 参数名 | 必填 | 类型 | 说明 |
| 请求参数 |  | | |
| userId | 是 | int64 | 对方用户id |
| ops | 是 | int | 1关注 -1取关 |
| 返回参数 |  | | |
| code | 是 | int | 返回码 |
| success | 是 | bool | 是否成功 |
| message | 是 | string | 返回消息 |
| data | 否 | int | 成功时为1 |

1. 获取关注列表接口

表 4-4 获取关注列表接口

|  |  |  |  |
| --- | --- | --- | --- |
| 请求URL | /users/user/myFollows | | |
| 请求类型 | POST | | |
| 参数名 | 必填 | 类型 | 说明 |
| 请求参数 |  | | |
| pageNum | 是 | int | 页数 |
| pageSize | 是 | int | 每页长度 |
| 返回参数 |  | | |
| code | 是 | int | 返回码 |
| success | 是 | bool | 是否成功 |
| message | 是 | string | 返回消息 |
| data | 否 | array | 返回数据 |
| data[].id | 是 | int64 | 对方id |
| data[].nickname | 是 | string | 对方昵称 |
| data[].rel | 是 | int | 对对方的关系，1为有关系，0为无关系 |
| data[].revRel | 是 | int | 对方对我的关系，1为有关系，0为无关系 |
| data[].headUrl | 是 | string | 对方头像地址 |
| data[].alias | 否 | string | 给对方的备注 |

1. 动态广场接口

表 4-5 动态广场接口

|  |  |  |  |
| --- | --- | --- | --- |
| 请求URL | /blog/blog/recommend | | |
| 请求类型 | POST | | |
| 参数名 | 必填 | 类型 | 说明 |
| 请求参数 |  | | |
| pageNum | 是 | int | 页数 |
| pageSize | 是 | int | 每页长度 |
| 返回参数 |  | | |
| code | 是 | int | 返回码 |
| success | 是 | bool | 是否成功 |
| message | 是 | string | 返回消息 |
| data | 否 | array | 返回数据 |
| data[].id | 是 | int64 | 发布人id |
| data[].nickname | 是 | string | 发布人昵称 |
| data[].rel | 是 | int | 对发布人的关系，1为有关系，0为无关系 |
| data[].revRel | 是 | int | 发布人对我的关系，1为有关系，0为无关系 |
| data[].headUrl | 是 | string | 发布人头像地址 |
| data[].alias | 否 | string | 给发布人的备注 |
| data[].blog | 是 | object | 发布的动态 |
| data[].blog.id | 是 | int64 | 动态id |
| data[].blog.content | 是 | string | 动态文本部分 |
| data[].blog.resource | 否 | string | 动态图片视频资源部分 |
| data[].blog.genre | 是 | string | 动态类型，TEXT为纯文字，  IMAGE[空格][数字n]为n张图片，  VIDEO为视频 |
| data[].blog.praises | 是 | int | 动态点赞数 |
| data[].blog.comments | 是 | int | 动态评论数 |
| data[].blog.addTime | 是 | int64 | 动态创建时间，时间戳 |

1. 发布动态接口

表 4-6 发布动态接口

|  |  |  |  |
| --- | --- | --- | --- |
| 请求URL | /blog/blog/publish | | |
| 请求类型 | POST | | |
| 参数名 | 必填 | 类型 | 说明 |
| 请求参数 |  | | |
| content | 是 | string | 动态文本部分 |
| resource | 否 | string | 动态图片视频资源部分 |
| genre | 是 | string | 动态类型，TEXT为纯文字，  IMAGE[空格][数字n]为n张图片，  VIDEO为视频 |
| access | 是 | int | 0 自己可见 1关注可见 2广场可见 |
| 返回参数 |  | | |
| code | 是 | int | 返回码 |
| success | 是 | bool | 是否成功 |
| message | 是 | string | 返回消息 |
| data | 否 | int64 | 新建动态的id |

1. 查看动态评论接口

表 4-7 查看动态评论接口

|  |  |  |  |
| --- | --- | --- | --- |
| 请求URL | /blog/comment/view | | |
| 请求类型 | POST | | |
| 参数名 | 必填 | 类型 | 说明 |
| 请求参数 |  | | |
| blogId | 是 | int64 | 动态id |
| pageNum | 是 | int | 页数 |
| pageSize | 是 | int | 每页长度 |
| 返回参数 |  | | |
| code | 是 | int | 返回码 |
| success | 是 | bool | 是否成功 |
| message | 是 | string | 返回消息 |
| data | 否 | array | 返回数据 |
| data[].id | 是 | int64 | 对方id |
| data[].nickname | 是 | string | 对方昵称 |
| data[].headUrl | 是 | string | 对方头像地址 |
| data[].praised | 是 | bool | 是否点赞 |
| data[].comment | 是 | object | 评论 |
| data[].comment.id | 是 | int64 | 评论id |
| data[].comment.userId | 是 | int64 | 评论者id |
| data[].comment.content | 是 | string | 评论内容 |
| data[].comment.praises | 是 | int | 评论点赞数 |
| data[].comment.addTime | 是 | int64 | 评论时间，时间戳 |

## 4.5数据库设计

数据库设计包括数据实体关系(Entity Relationship，简称E-R)模型，和数据库表结构的设计。

## 4.5.1数据库E-R模型

本系统中数据库有Mysql和Neo4j。Mysql用于主要的数据存储，Neo4j用于存储好友多对多的关系，同时存储用户动态，以便查找用户关注人的动态。图4-11为Mysql的实体关系图，图4-12为Neo4j的实体关系图

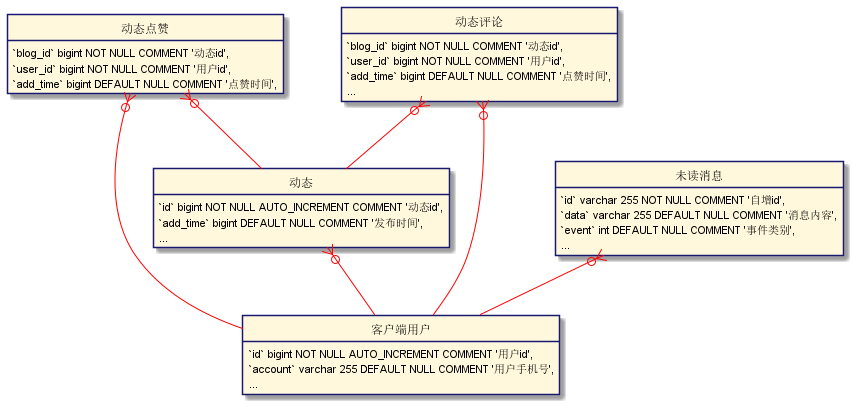


图 4-11 Mysql E-R图

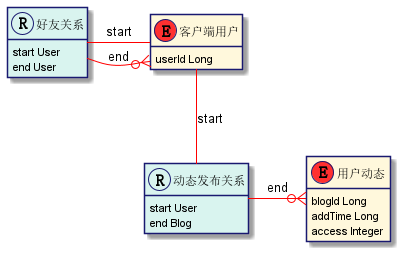


图 4-12 Neo4j E-R图

## 4.5.2数据库表设计

本系统中主要有五张数据表。主要用于存放用户信息、动态信息、动态点赞信息、动态评论信息、未读消息。

表client\_user用于存放用户信息；表blog用于存放动态信息，表中通过user\_id关联到client\_user的id字段；表blog\_praise用于存放点赞信息，表中通过blog\_id关联到blog表的id字段或关联到comment表的id字段；表comment用于存放评论信息，表中通过blog\_id关联到blog的id字段，user\_id关联到client\_user的id字段；表unread\_message用于存放用户因为不在线而错过的未读消息，表中通过user\_id关联到client\_user的id字段。

表 4-8 client\_user表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 允许空 | 说明 |
| id | bigint(0) | 否 | 自增id |
| account | varchar(255) | 是 | 手机号 |
| add\_time | bigint(0) | 是 | 创建时间戳 |
| fans | int | 是 | 粉丝数 |
| follows | int | 是 | 关注数 |
| friends | int | 是 | 好友数 |
| head\_url | varchar(255) | 是 | 头像地址 |
| Login\_time | bigint(0) | 是 | 登录时间戳 |
| nickname | varchar(255) | 是 | 昵称 |
| password | varchar(255) | 是 | 密码，BCrypt-5加密 |
| self\_state | varchar(255) | 是 | 个性签名 |
| status | int | 是 | 状态0无效用户1有效 |
| update\_time | bigint(0) | 是 | 更新时间戳 |
| uid | bigint(0) | 是 | 用户易记的号码 |
| active\_time | bigint(0) | 是 | 活跃时间戳 |
| roles | json | 是 | key是角色名，value是过期时间，-1永不过期 |

表 4-9 blog表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 允许空 | 说明 |
| id | bigint(0) | 否 | 自增id |
| add\_time | bigint(0) | 是 | 创建时间戳 |
| comments | int | 是 | 评论数 |
| content | varchar(255) | 是 | 内容 |
| genre | varchar(255) | 是 | 动态类型，TEXT为纯文字，  IMAGE[空格][数字n]为n张图片，  VIDEO为视频 |
| praises | int | 是 | 点赞数 |
| resource | varchar(255) | 是 | 资源地址 |
| user\_id | bigint(0) | 是 | 发布者id |
| access | int | 是 | 0 自己可见 1关注可见 2广场可见 |
| snapshot | json | 是 | 评论快照 |
| status | int | 是 | 状态码0删除1可用 |

表 4-10 blog\_praise表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 允许空 | 说明 |
| blog\_id | bigint(0) | 否 | 联合主键，动态或评论id |
| user\_id | bigint(0) | 否 | 联合主键，用户id |
| add\_time | bigint(0) | 否 | 创建时间戳 |

表 4-11 comment表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 允许空 | 说明 |
| id | bigint(0) | 否 | 自增id |
| blog\_id | bigint(0) | 是 | 动态id |
| add\_time | bigint(0) | 是 | 创建时间戳 |
| content | varchar(255) | 是 | 内容 |
| praises | int | 是 | 点赞数 |
| user\_id | bigint(0) | 是 | 发布人 |
| reply | bigint(0) | 是 | 回复人 |
| status | int | 是 | 状态0无效1有效 |

表 4-12 unread\_message表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 允许空 | 说明 |
| id | varchar(255) | 否 | 自定义id |
| data | varchar(255) | 是 | 消息内容 |
| event | bigint(0) | 是 | 事件类型 |
| from\_id | bigint(0) | 是 | 创建用户 |
| time | bigint(0) | 是 | 创建时间 |
| user\_id | bigint(0) | 是 | 目标用户 |

## 4.6本章小结

本章按照第三章的需求分析的前提下，对功能模块做了详细设计，利用文字结合图片的形式，讲解了功能执行的过程，包括整体服务架构、前端设计、后端设计、前后端接口设计、数据库设计，为第五章提供指导骨架。

# 第五章 社交聊天系统的实现

## 5.1前端实现

## 5.1.1目录与文件结构

为了降低前端的性能损耗，前端不使用第三方框架，以纯原生编写。系统中界面包括登录界面，动态界面，发布动态界面，关注列表界面，聊天界面。图5-1为前端代码目录结构。

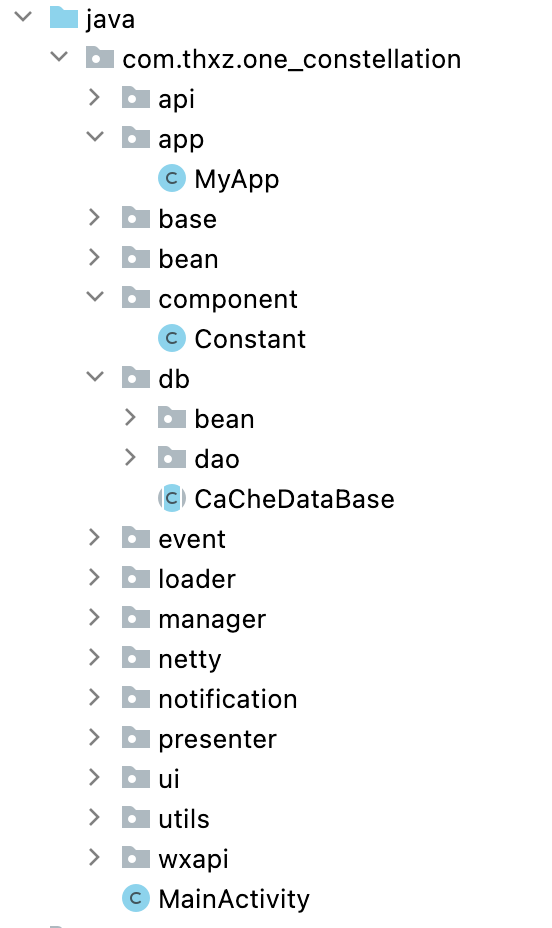


图 5-1 前端代码目录结构

## 5.1.2核心功能实现

本节主要讲述前端的核心功能其实现，以文字结合图片的形式展示核心功能。

1. 登录界面

图5-2为用户登录界面，用户登录有两种方式登录，一种是密码登录，一种是短信登录。该功能通过“短信登录/密码登录”进行切换隐藏验证码框或密码框。通过点击登录按钮触发登录事件，发送POST请求到后端。

短信验证码登录时，先要通过后端返回来经过BCrypt-5加密的验证码做第一步的验证，才能向后端发起请求，以减少不必要的失败的场景。



a) 用户密码登录界面 b) 短信验证码登录界面

图 5-2 用户登录界面

前端调用的核心代码如下：

1. if(isLogin){//密码登录
2. String pass = pass1.getText().toString();
3. if ("".equals(pass)) {
4. ToastUitl.showLong("密码不能为空");
5. return;
6. }
7. if (pass.length() < 8) {
8. ToastUitl.showLong("密码长度不够");
9. return;
10. }
11. mPresenter.loadGetLoginPassInfo(phone,pass);
12. }else { //验证码登录
13. if (TextUtils.isEmpty(tcVerifi)) {
14. ToastUitl.showLong("请先获取验证码");
15. return;
16. }
17. String codeStr = code.getText().toString();
18. if (TextUtils.isEmpty(codeStr)) {
19. ToastUitl.showLong("验证码不能为空");
20. return;
21. }
22. if (bcrypt.matches(codeStr,tcVerifi)) {
23. mPresenter.loadGetLoginAccountInfo(phone, codeStr);
24. } else {
25. ToastUitl.showShort("验证码错误");
26. }
27. }
28. 动态界面

完成登录后，页面跳转到动态界面。动态界面如图5-3所示，前端从后端获取到动态数据后，将数据展示到页面。



a)关注人动态 b)动态广场

图 5-3 动态界面

前端调用的核心代码如下：

1. if (bean.isSuccess()) {
2. List<ElementsBean> data = bean.getData();
3. if (data != null && data.size() > 0) {
4. if (page == Constant.DEFOUT\_LOAD\_PAGE) {
5. mList = data;
6. mAdapter.setNewData(mList);
7. recyclerView.scrollToPosition(0);
8. setDefaultReleasePostData();
9. bean.setDate(TimeUtils.getNowLong() + "");
10. SharedPreferencesUtils.setParam(getActivity(),
11. Constant.HOME\_TITLE\_RECOMMEND
12. + MyApp.getUSER().getId(), new Gson().toJson(bean));
13. } else {
14. mAdapter.addData(data);
15. mAdapter.loadMoreComplete();
16. }
17. } else {
18. mAdapter.isUseEmpty(true);
19. mAdapter.notifyDataSetChanged();
20. mAdapter.loadMoreEnd();
21. }
22. } else {
23. mAdapter.isUseEmpty(true);
24. mAdapter.notifyDataSetChanged();
25. mAdapter.loadMoreEnd();
26. ToastUitl.showShort(bean.getMessage());
27. }
28. 发布动态界面



图 5-4 动态发布界面

图5-4为动态发布界面。当用户想要发布动态时，点击底部功能栏中间的功能键，可以弹出隐藏菜单，点击“发动态”按钮，即可跳转到动态发布界面，设置文本，图片，访问权限，点击发布即可完成发布动作。

前端调用的核心代码如下：

1. switch (event.getType()) {
2. case Constant.EVENT\_TYPE\_ADD:
3. mVp.setCurrentItem(1, false);//显示对应布局
4. if (blog.getGenre().contains(Constant.POST\_TYPE\_TEXT)) {
5. mPresenter.loadReleaseInfo(
6. blog.getContent(),
7. blog.getResource(),
8. blog.getGenre(),
9. blog.getAccess());
10. }else if (blog.getGenre().contains(Constant.POST\_TYPE\_IMAGE)){
11. time = TimeUtils.getNowLong() +"";
12. mPresenter.loadGetQiNiuTokenInfo();
13. }
14. break;
15. }
16. 关注人列表

图5-5为关注列表界面。所展示的内容为用户所关注的用户。



图 5-5 关注列表

前端调用的核心代码如下：

1. switch (title) {
2. case "星友":
3. mPresenter.loadUserFriendsListInfo(page, size);
4. break;
5. case "关注":
6. mPresenter.loadUserFollowerListInfo(page, size);
7. break;
8. case "粉丝":
9. mPresenter.loadUserFansListInfo(page, size);
10. break;
11. }
12. 聊天界面

图5-6为聊天界面，用户可以发送文字，图片，表情包，也会接收对方发送的消息。历史的聊天记录存储在前端的SqlLite中。

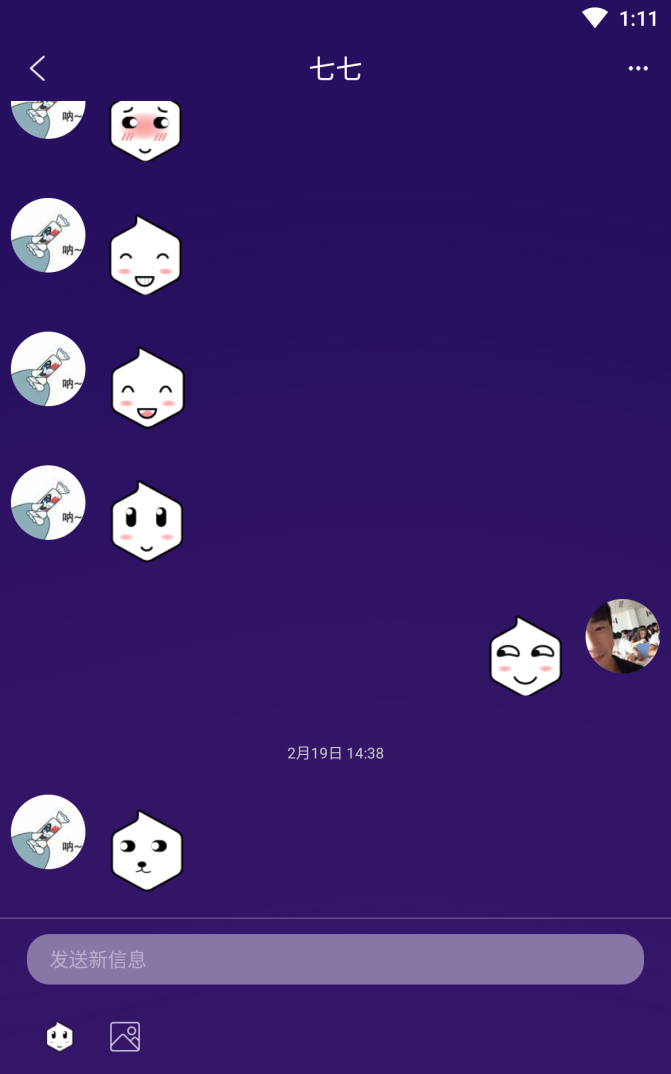


图 5-6 聊天界面

前端获取本地聊天历史的核心代码如下：

1. Disposable subscribe = Observable.create(
2. new ObservableOnSubscribe<List<ChatDataBean>>() {
3. @Override
4. public void subscribe(ObservableEmitter<List<ChatDataBean>> emitter)
5. throws Exception {
6. List<ChatDataBean> chatDataBeans = MyApp.getmCacheDatabase()
7. .chatDataDao()
8. .loadChatDataUnreadBeans(MyApp.getUSER().getId(), false);
9. emitter.onNext(chatDataBeans);
10. }
11. }).subscribeOn(Schedulers.io())
12. .observeOn(AndroidSchedulers.mainThread())
13. .subscribe(new Consumer<List<ChatDataBean>>() {
14. @Override
15. public void accept(List<ChatDataBean> chatDataBeans) throws Exception {
16. chatDataBeansList = chatDataBeans;
17. showUnread();//显示未读数
18. }
19. });
20. 即时通信

前端接入即时通讯使用IO框架的是Netty-client，使用的通讯方式是TCP。

其构建代码如下：

1. @Override
2. public void initChannel(SocketChannel ch) {
3. ch.pipeline().addLast(
4. new ReconnectHandler(tcpClient),
5. new LengthFieldBasedFrameDecoder(2048, 0, 2, 0, 2),
6. stringDecoder,
7. new LengthFieldBasedFrameEncoder(),
8. new IdleStateHandler(20, 20, 0),
9. new HeartbeatHandler(),
10. new ServiceHandler());
11. }

其业务处理代码如下：

1. LOG.e("SSSSSS", "收到消息:" + msg.toString() +
2. ",chanel="+ctx.channel().toString());
3. ClientBean clientBean = new Gson().fromJson(msg.toString(), ClientBean.class);
4. if (clientBean.getData() == null) return;
5. LOG.e("SSSSS", "getEvent=" + clientBean.getEvent());
6. switch (clientBean.getEvent()) {
7. case Constant.NETTY\_EVENT\_INFORM:
8. //通知
9. InformDataBean informDataBean = new InformDataBean();
10. informDataBean.id = clientBean.getId();
11. informDataBean.oneselfId = MyApp.getUSER().getId();
12. informDataBean.nickname = clientBean.getData().getNickname();
13. informDataBean.headUrl = clientBean.getData().getHeadUrl();
14. informDataBean.sign = clientBean.getData().getSign();
15. informDataBean.userId = clientBean.getFromId();
16. informDataBean.operate = clientBean.getData().getOperate();
17. informDataBean.toId = clientBean.getData().getToId();
18. informDataBean.subject = clientBean.getData().getSubject();
19. informDataBean.genre = clientBean.getData().getGenre();
20. informDataBean.preview = clientBean.getData().getPreview();
21. informDataBean.time = clientBean.getTime();
22. informDataBean.isRead = false;
23. EventBus.getDefault().postSticky(new ClientInformWorkEvent(
24. Constant.EVENT\_TYPE\_ADD,
25. Constant.NETTY\_EVENT\_INFORM, informDataBean));
26. ChatPushBean infrom = new ChatPushBean();
27. infrom.setEvent(-clientBean.getEvent());
28. infrom.setId(clientBean.getId());
29. ctx.channel().writeAndFlush(new Gson().toJson(infrom));
30. break;
31. case Constant.NETTY\_EVENT\_CHAT:
32. //聊天
33. ChatDataBean chatDataBean = new ChatDataBean();
34. chatDataBean.id = clientBean.getId();
35. chatDataBean.userId = clientBean.getUserId();
36. chatDataBean.toId = clientBean.getFromId();
37. chatDataBean.genre = clientBean.getData().getGenre();
38. chatDataBean.content = clientBean.getData().getContent();
39. chatDataBean.time = clientBean.getTime();
40. chatDataBean.oneselfId = MyApp.getUSER().getId();
41. String itemId = chatDataBean.oneselfId +
42. Constant.EVENT\_TYPE\_CHAT + chatDataBean.toId;
43. chatDataBean.itemId = itemId;
44. chatDataBean.isRead = false;
45. EventBus.getDefault().postSticky(new ClientChatWorkEvent(
46. Constant.EVENT\_TYPE\_ADD,
47. Constant.NETTY\_EVENT\_CHAT, chatDataBean));
48. ChatPushBean chat = new ChatPushBean();
49. chat.setEvent(-clientBean.getEvent());
50. chat.setId(clientBean.getId());
51. ctx.channel().writeAndFlush(new Gson().toJson(chat));
52. break;
53. case Constant.NETTY\_EVENT\_UNREAD:
54. if (clientBean.getData() != null) {
55. LOG.e("SSSSS", "未读消息=" + clientBean.getData().getInform());
56. //未读消息
57. if (clientBean.getData().getInform() > 0) {
58. EventBus.getDefault().postSticky(new ClientInformUIEvent(
59. Constant.EVENT\_TYPE\_INFORM,
60. Constant.NETTY\_EVENT\_UNREAD,
61. clientBean.getData().getInform()));
62. } else if (clientBean.getData().getChat() > 0) {
63. EventBus.getDefault().postSticky(new ClientChatUIEvent(
64. Constant.EVENT\_TYPE\_CHAT,
65. Constant.NETTY\_EVENT\_UNREAD,
66. clientBean.getData().getChat()));
67. }
68. }
69. break;
70. case Constant.NETTY\_EVENT\_RECEIPT:
71. //回执消息
72. if (clientBean.getData() != null) {
73. LOG.e("SSSSS","回执消息");
74. EventBus.getDefault().postSticky(
75. new MsgRectiptEvent(clientBean));
76. }
77. break;
78. }

## 5.2后端实现

## 5.2.1概述

图5-7为所有后端项目的目录结构图。其中，auth目录为认证服务，职责是维护用户登录和权限；blog目录为动态服务，职责是维护用户动态及相关功能；examine目录为审核模块，职责是内容审核；gateway目录为网关服务；register目录为注册中心；users目录为用户服务，职责是处理用户数据以及用户关注信息。

在其中任意一个目录下，src/main/java/com/topsign/+模块名为代码部分，config目录下代码是项目配置，controller目录下代码是请求接口定义，exception下代码是异常处理，message下代码用于消息队列发送和处理，model下代码是数据库实体类，node下代码是neo4j实体类，nodeRepository下代码是neo4j访问类，repository下代码是数据库访问类，security下代码是用户权限鉴定模块，service下代码是业务处理类，util下代码是工具类。

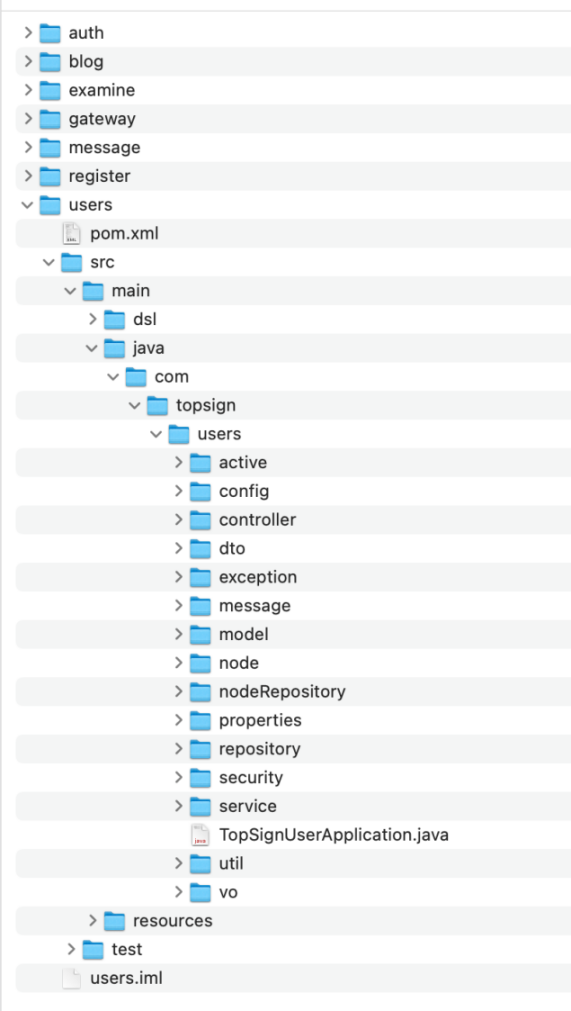


图 5-7 后端项目目录结构图

## 5.2.2用户注册的实现

代码位置：auth/src/main/java/com/topsign/auth/service/impl/RegisterServiceImpl.java的smsRegister方法。

文字描述：用户在登录页面点击右上角注册按钮，进入注册页面，通过手机号获取验证码，进行注册。后端接收到前端的请求后，先确认验证码是否正确，再确认账号是否存在，如存在则返回失败，如不存在则继续，创建为当前手机号创建新用户，向Mysql和Neo4j中创建一个用户节点，并返回访问令牌和刷新令牌。

图5-8是本系统中的一个访问令牌的案例，访问令牌由三个部分组成：head、playload、tail。head为头部，是加密算法名经过base64编码后的字符串；playload是信息存储单元，由图可以看到它的组成部分：exp为过期时间，单位是秒；user\_name是对称加密的手机号+分隔符+昵称，authorities是权限内容。借由以上信息，可以确立一个token的过期时间，持有人，权限；tail为尾部，是通过head中记录的加密方式，加上特定的密钥，对playload进行加密后的密串，以验证整个令牌的有效性[4]。

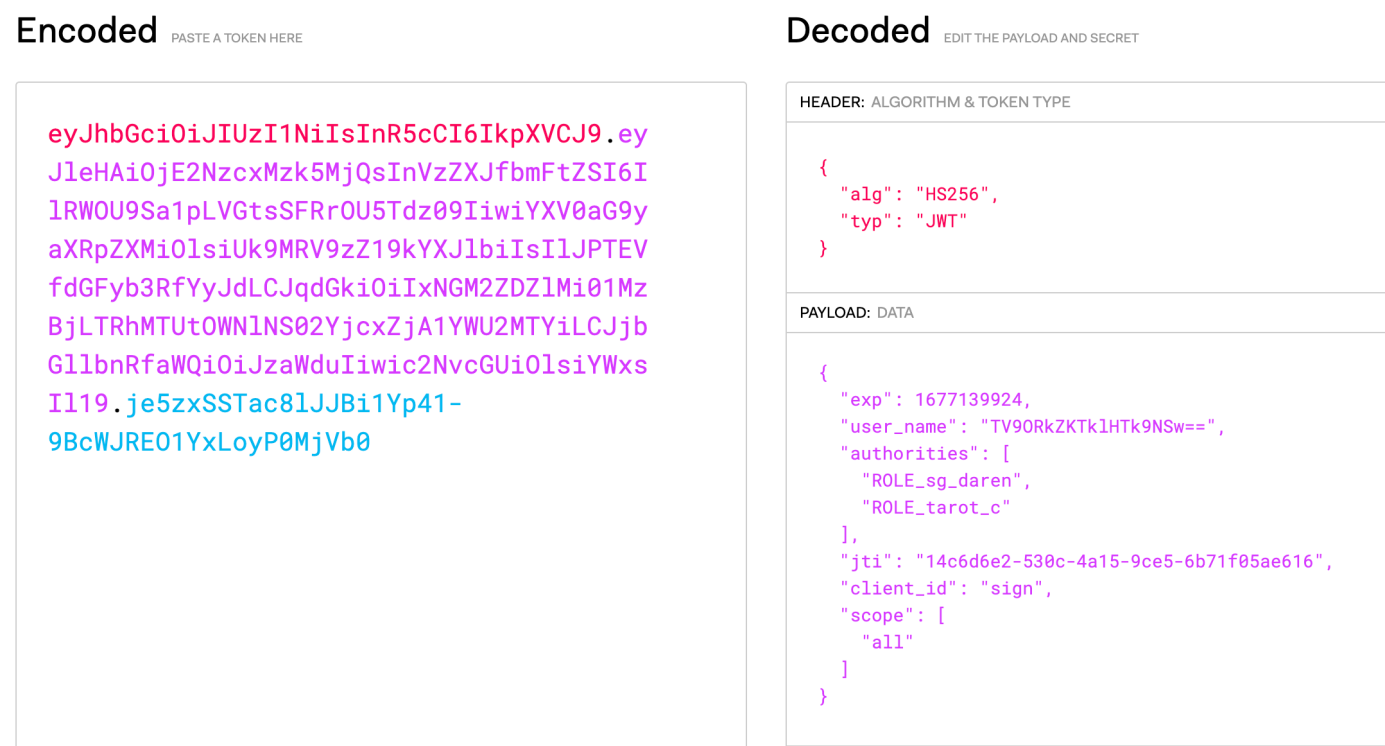


图 5-8 访问令牌案例

## 5.2.3用户登录的实现

代码位置：auth/src/main/java/com/topsign/auth/security/endpoint/LoginEndpoint.java

文字描述：用户在登录页通过密码或手机号登录。当使用密码登录时，首先获取查找用户是是否存在，如果不存在则返回失败，如果存在，则通过数据库存储的密码，进行Bcrypt-5校验，如不匹配则返回密码错误，如果通过则返回成功。当用户使用短信登录是，首先校验短信验证码是否正确，如若不正确，则返回失败，如正确，获取查找用户是是否存在，如果不存在则创建用户，如果存在，则直接返回成功。

## 5.2.4发布动态的实现

代码位置：blog/src/main/java/com/topsign/blog/service/impl/BlogServiceImpl.java

文字描述：用户在动态发布页面提交了动态信息后，后端接收到这些信息，首先将数据的有效字段置为0，存储入数据库中，再发送到审核模块，经由审核模块完成审核后，返回最终结果，当未通过审核，给予用户通知，并删除动态，当通过审核，则将数据的有效字段置为1，更新到数据库中，并在neo4j中创建一个动态节点以及动态节点与发布用户节点之间的发布关系。

## 5.2.5关注好友的实现

代码位置：users/com/topsign/users/service/impl/ClientUserServiceImpl.java follow方法

文字描述：用户点进他人主页后，点击左下角的关注按钮，调用关注接口。后端接收到请求后，首先校验是否被关注人是自己，是则返回失败，再建立一个存在时长3秒的Redis key，如果Redis key已存在于Redis中，则返回失败，如果Redis key不存在则向Redis中写入该key，以达到限制重复操作的效果。随后进入关注操作。关注操作中，校验是否存在关系，当自己对对方陌生人时，对方对自己也是陌生人，则变为单方面关注，当对方关注自己时，变为好友关系；当自己对对方关注，对方对自己时陌生人时，取消关注，当对方也关注自己时，取消朋友关系，转为对方单方面关注自己。

## 5.2.6即时通讯的实现

代码文件：业务代码类为message/src/main/java/com/topsign/ message/ service/impl/MessageServiceImpl.java，构建Netty类为message /src/main/java/com/topsign/message/socket/ChannelInit.java

文字描述：用户在登录成功后就已经接入即时通讯，即时通讯消息的结构如图5-9，消息事件的类型有：通知，聊天，未读消息。当前端在登录成功后，通过三次握手连接上后端服务，第一次需要发送用访问令牌到后端，后端通过访问令牌确认用户身份，并以用户id为key，连接通道为value存到服务内部的ConcurrentHashMap中，以便查找到用户的连接通道并主动推送消息。在完成身份确认后，从数据库中查找用户的未读消息，并清空，将未读消息主动推送到前端。当其他业务服务需要向用户推送消息是，使用消息队列向消息服务发送信息。

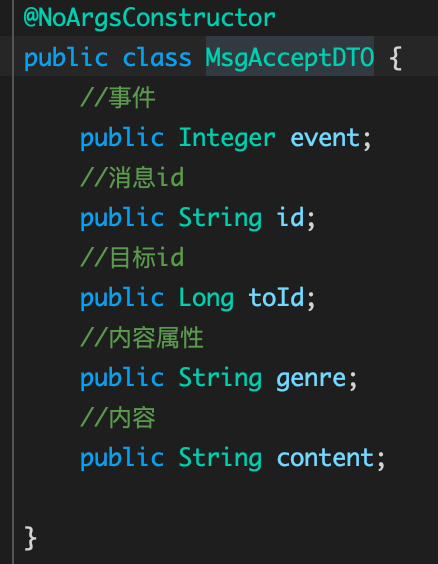


图 5-9 即时通讯消息实体

## 5.2.7内容审核的实现

代码位置：examine/src/main/java/com/topsign/examine/service/impl/ ExamineServiceImpl.java

文字描述：图5-10为审核请求消息结构体，当动态发布后，动态服务需要提供业务id，回调队列名，业务标识，资源列表，资源列表是以资源类型为key，资源数组为值的键值对。审核模块在接收到该消息后，将业务id和业务标识组成Redis key，以set的数据结构，将parts存储进数据库。如果审核不通过则会清理掉该Redis key，并根据module发送失败结果给动态服务；如果审核通过，则检查是否存在Redis key，当Redis key不存在，则说明审核整体性失败，则不处理，若存在，则消除该Redis key下的该项，每次审核通过一个就清理调set中的当前值，当当前值是set中最后一个，就根据module发送成功结果给动态服务。

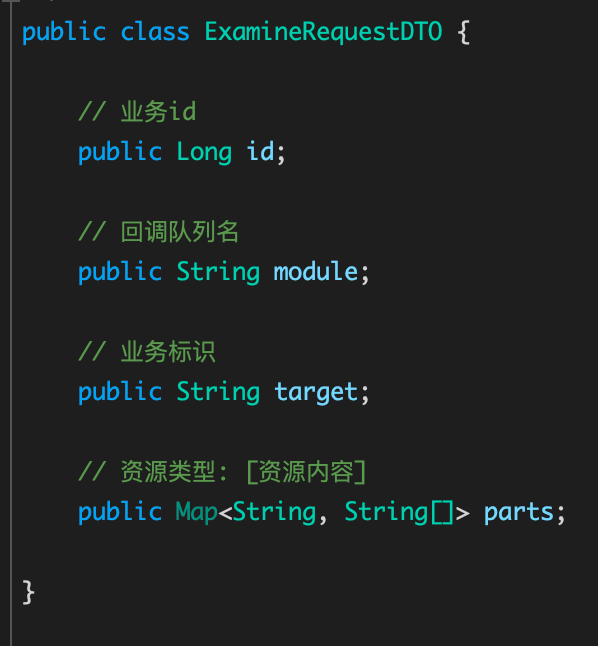


图 5-10 审核请求消息结构体

## 5.3系统部署运行

系统部署使用的安卓/服务器模式。在完成所有组件的安装工作后，打包服务端，后端是Spring Boot，需要将项目打包成jar包后，通过Shell命令 “nohup java -jar 包名 -spring.profiles.active=prod &”启动执行，最先启动注册中心，其次是网关，最后是各个业务模块，即可完成后端的部署。客户移动端需要安装前端打包完成的APK。

## 5.4本章小结

本章以图片加文字加代码的方式，讲述了前后端代码的实现过程，本章在第四章的举出上，将所有功能设计落地。

# 第六章 社交聊天系统的测试

## 6.1测试环境

硬件环境：

1. CPU：Intel(R) Core(TM) i7-7660U CPU @ 2.50GHz
2. 内存：8G
3. 硬盘：20G

软件环境

1. JDK版本：1.8
2. Mysql版本：8.0
3. Redis版本：6.0.8
4. Neo4j版本：3.5.25
5. Rabbitmq版本：3.11.9
6. 系统版本：Ubuntu 22
7. 接口测试工具：Postman
8. 功能测试工具：安卓模拟器
9. 性能测试工具：Jmeter

## 6.2接口测试

将对社交聊天系统核心API进行黑盒测试，用表格形式展示请求地址、请求参数、 响应结果。表 6-1 为接口测试总表:

表 6-1 接口测试总表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 服务器地址：http://127.0.0.1:9071 | | | | |
| 序号 | 模块 | 名称 | 请求URL | 结果 |
| 1 | 认证模块 | 发送验证码 | /auth/register/sendSms/{sense} | 通过 |
| 2 | 手机短信登录 | /auth/login/smsCode | 通过 |
| 3 | 手机号绑定注册 | /auth/register/smsRegister | 通过 |
| 4 | 密码登录 | /auth/login/password | 通过 |
| 5 | 用户模块 | 关注/取关用户 | /users/user/follow | 通过 |
| 6 | 获取关注列表 | /users/user/myFollows | 通过 |
| 7 | 获取粉丝列表 | /users/user/myFans | 通过 |
| 8 | 设置好友别名 | /users/user/friendAlias | 通过 |
| 9 | 动态模块 | 动态广场 | /blog/blog/recommend | 通过 |
| 10 | 关注人动态 | /blog/blog/follow | 通过 |
| 11 | 发布动态 | /blog/blog/publish | 通过 |
| 12 | 删除动态 | /blog/blog/delete | 通过 |
| 13 | 动态点赞 | /blog/blog/praise | 通过 |
| 14 | 查看动态评论 | /blog/comment/view | 通过 |
| 15 | 添加动态评论 | /blog/comment/add | 通过 |
| 16 | 删除动态评论 | /blog/comment/delete | 通过 |

以下是核心API详细测试信息：

1. 短信登录接口

表 6-2 短信登录接口

|  |  |
| --- | --- |
| 请求类型 | POST form |
| 请求URL | http://127.0.0.1:9071/auth/login/smsCode |
| 请求头 | Authorization: Basic c2lnbjoxMjM0NTY= |
| 请求参数 | account=199xxxxxxxx  smsCode=123456 |
| 响应参数 | {  "code": 0,  "success": true,  "message": "请求成功",  "data": {  "token": {  "accessToken": "....",  "tokenType": "bearer",  "expiresIn": 599,  "refreshToken": "..."  },  "user": {  "id": 2,  "nickname": "会飞的白羊",  "headUrl": "user\_head\_2\_1604365592",  "roles": {  "vip\_vip": 1603976824674,  "ROLE\_sg\_daren": -1  },  }  }  } |

1. 关注/取关用户接口

表 5-3 关注/取关用户接口

|  |  |
| --- | --- |
| 请求类型 | POST form |
| 请求URL | http://127.0.0.1:9071/users/user/follow |
| 请求头 | Authorization: Basic 访问令牌 |
| 请求参数 | userId=100  ops=1 |
| 响应参数 | {  "code": 0,  "success": true,  "message": "请求成功"  } |

1. 获取关注列表接口

表 5-4 获取关注列表接口

|  |  |
| --- | --- |
| 请求类型 | POST form |
| 请求URL | http://127.0.0.1:9071/users/user/myFollows |
| 请求头 | Authorization: Basic 访问令牌 |
| 请求参数 | pageNum=1  pageSize=20 |
| 响应参数 | {  "code": 0,  "success": true,  "message": "请求成功",  "data": [  {  "id": 310,  "nickname": "KIK",  "headUrl": "BLOG\_IM\_310\_1621702234495\_0",  "rel": 1,  "revRel": 0  },....  ]  } |

1. 动态广场接口

表 5-5 动态广场接口

|  |  |
| --- | --- |
| 请求类型 | POST form |
| 请求URL | http://127.0.0.1:9071/blog/blog/recommend |
| 请求头 | Authorization: Basic 访问令牌 |
| 请求参数 | pageNum=1  pageSize=20 |
| 响应参数 | {  "code": 0,  "success": true,  "message": "请求成功",  "data": [  {  "id": 5,  "nickname": "七七",  "headUrl": "user\_head\_5\_1606099951",  "rel": 1,  "revRel": 0,  "blog": {  "id": 1140,  "content": "失眠。",  "resource": "BL5\_1621961433IM0",  "genre": "IMAGE 1",  "praises": 0,  "comments": 0,  "addTime": 1621961435465  },  "praised": false  },...  ]  } |

1. 发布动态接口

表 5-6 发布动态接口

|  |  |
| --- | --- |
| 请求类型 | POST |
| 请求URL | http://127.0.0.1:9071/blog/blog/publish |
| 请求头 | Authorization: Basic 访问令牌 |
| 请求参数 | content=哈哈哈  resource=BLOG\_IM\_2\_1619962874818\_0  genre=TEXT  access=2 |
| 响应参数 | {  "code": 0,  "success": true,  "message": "请求成功",  "data": 1144  } |

1. 查看动态评论接口

表 5-7 查看动态评论接口

|  |  |
| --- | --- |
| 请求类型 | POST |
| 请求URL | http://127.0.0.1:9071/blog/comment/view |
| 请求头 | Authorization: Basic 访问令牌 |
| 请求参数 | — |
| 响应参数 | {  "code": 0,  "success": true,  "message": "请求成功",  "data": [  {  "id": 5,  "nickname": "七七",  "praised": false,  "comment": {  "id": 634,  "userId": 132,  "blogId": 757,  "content": "你想多了。。",  "praises": 0,  "status": 1,  "addTime": 1612179227829  }  },...  ]  } |

根据接口测试结果可得，系统接口符合预期的需求。

## 6.3功能测试

本小节的内容是测试社交聊天系统的主要功能，首先做好测试流程，其次通过黑盒测试，检测系统是否符合预期的功能需求。 功能测试无需考虑业务代码实现的内容，只关心实际的操作是否满足要求。表6-8 是流程测试总表。

表 6-8 流程测试总表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试编号 | 测试项 | 步骤 | 结果 |
| T001 | 登录 | 输入手机号码、登录密码进入应用 | 正常 |
| T002 | 发布动态 | 提供文字，图片，选择访问范围，提交 | 正常 |
| T003 | 关注好友 | 点进陌生人主页，点击关注 | 正常 |
| T004 | 发送聊天信息 | 在关注人的主页点击私聊，发送消息 | 正常 |

1. 登录测试

未登录的用户打开应用进入，就可以看到登录页面，输入手机号并输入密码，点击登录按钮，就可以进入应用主页，完成登录。图 6-1 为用户登录过程。

a) 登录界面 b)登录完成后进入主页

图 6-1 用户登录过程

1. 发布动态测试

已经登录的用户在底部菜单栏里，点击中间的功能按钮，点击“发动态”按钮，进入发布动态页面，输入文字，图片，选择访问范围，点击右上角的发布，即可在广场上看到自己的动态。图6-2为用户发布动态到广场的过程。

a) 点击发动态 b) 输入动态内容 c) 在广场可以看到自己的动态

图 6-2 用户发布动态到广场过程

1. 关注好友测试

在广场页面看到陌生人，通过点击对方头像对方主页，点击左下角的关注按钮，即可完成关注，在自己的关注列表中也能看到对方。图6-3为用户关注好友的过程。

a) 陌生人主页 b) 关注成功 c) 关注列表

图 6-3 用户关注好友过程

1. 发送聊天测试

在图6-3 a 中展示的页面中，点击下方私聊按钮，即可进入与对方的聊天界面，发送一条文本消息，发送成功。图6-4为消息发送成功后的界面。

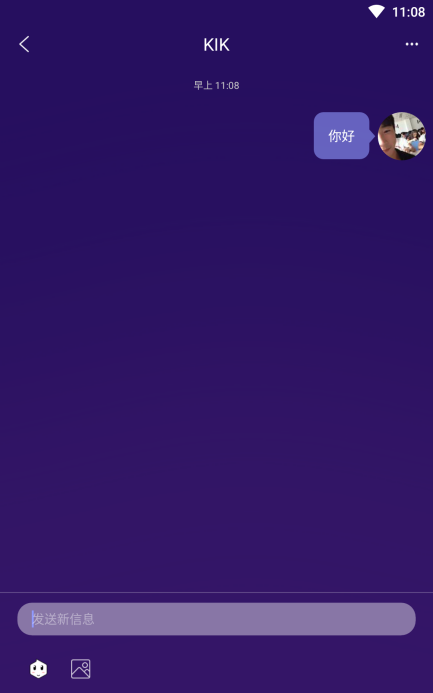


图 6-4 消息发送成功

## 6.4兼容性测试

本节使用安卓模拟器模拟具有代表性的移动设备显示尺寸，对社交聊天系统按照表 6-8的项目进行兼容性测试，检测界面是否正常显示，功能是否正常执行。结果如 6-9所示:

表 0-9 兼容性测试表

|  |  |  |
| --- | --- | --- |
| 设备型号 | 测试内容 | 结果 |
| HUAWEI P10 | T001, T002, T003, T004 | 正常 |
| Xiaomi MIX2 | 正常 |
| VIVO X20 | 正常 |
| OPPO R11 | 正常 |

## 6.5性能测试

最后做性能测试，使用Jmeter工具对某一接口，通过高频访问做压力测试，测试接口执行时长，正确率，QPS，以测试系统的稳定性和性能。测试过程如下：

1. 建立线程组。

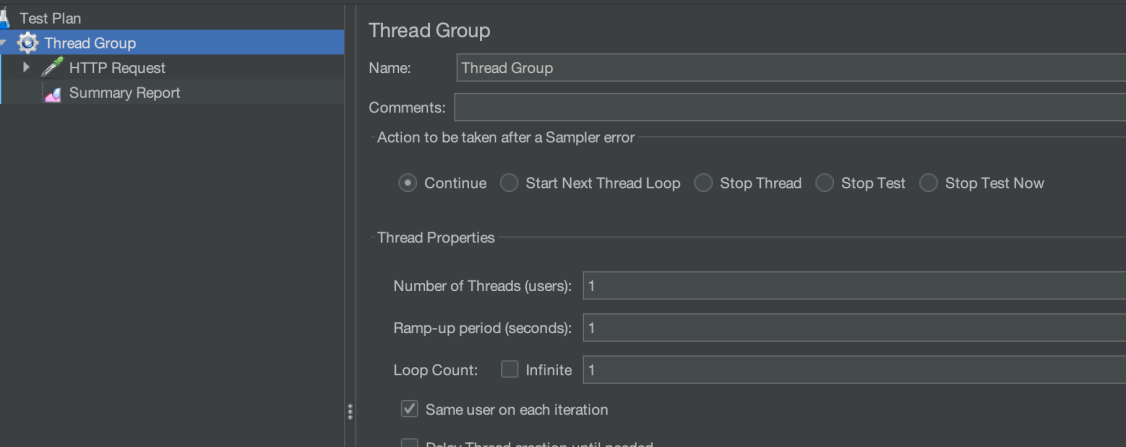


图 6-5 创建线程组

创建请求，接口选用返回数据较多的关注人动态。

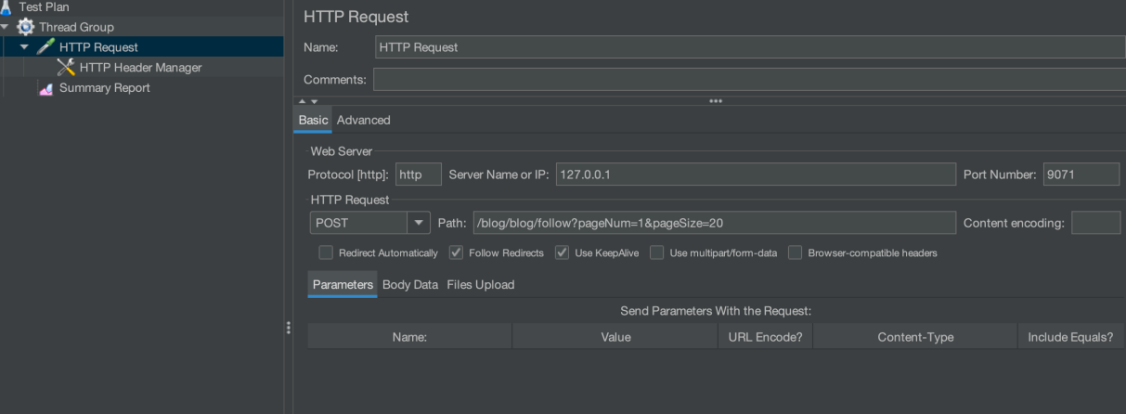


图 6-6 创建HTTP请求

创建请求头组，添加请求所需要的访问令牌。

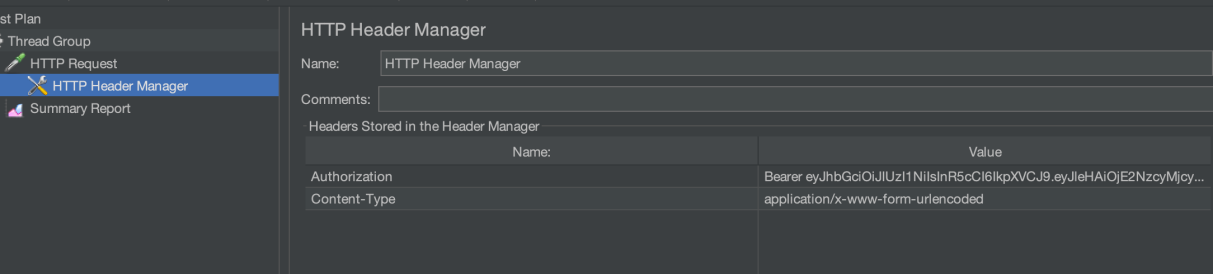


图 6-7 创建请求头组

1. 当线程数为200时：

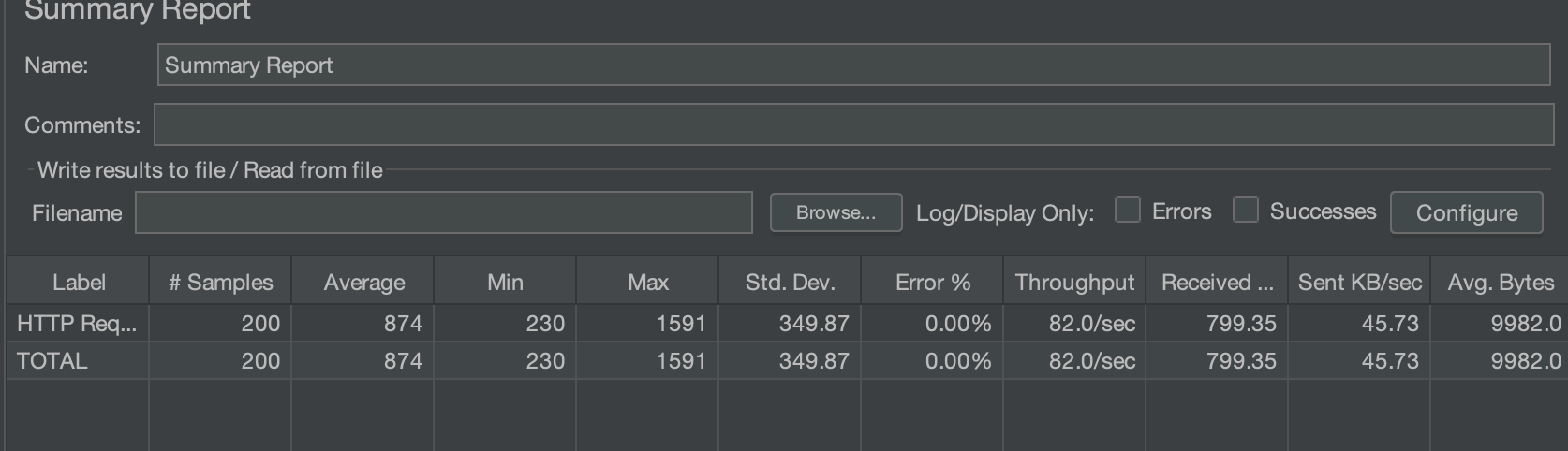


图 6-8 当线程数为200时的汇总结果图

如图6-8所示，当线程数为200是，这是一个较为低频的访问，错误率为0%，没有错误。

1. 当线程数为500时：

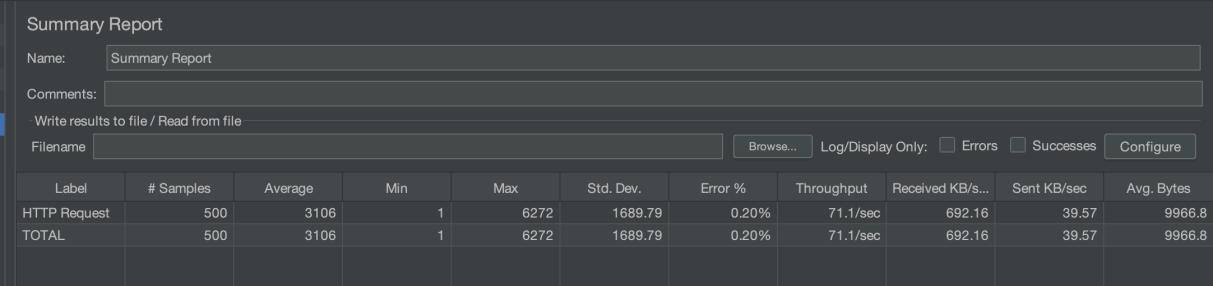


图 6-9 当线程数为500时的汇总结果图

如图6-9所示，当线程数为500时，中等频率，错误率为0.2%，有1次请求错误。

1. 当线程数为750时：

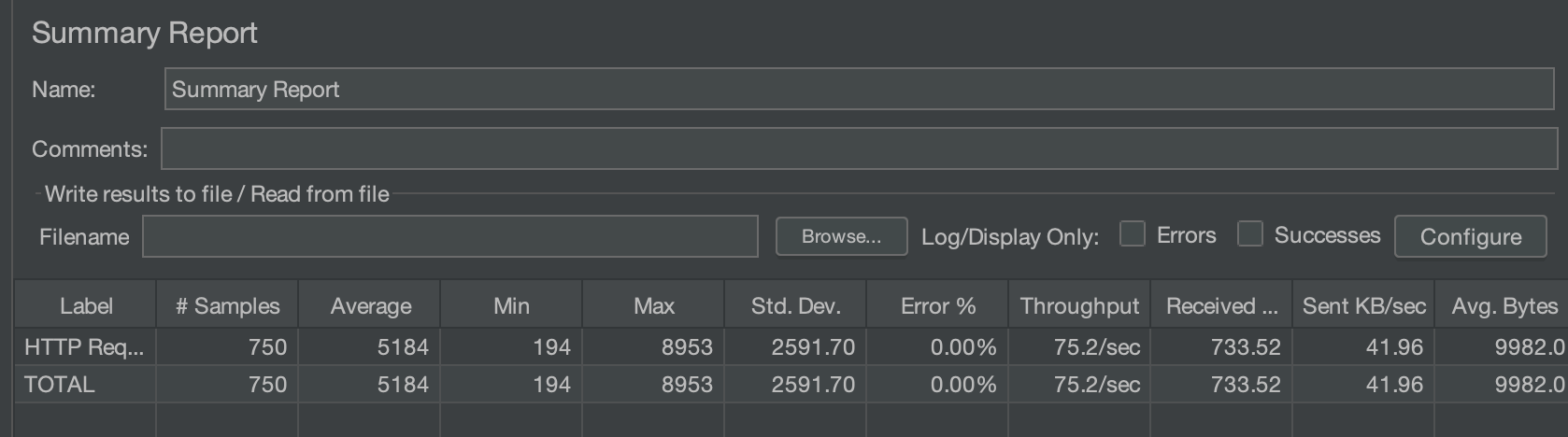


图 6-10 当线程数为750时的汇总结果图

如图6-10所示，当线程数为750时，稍微高频，错误率为0%，没有错误产生。

1. 当线程数为1000时：

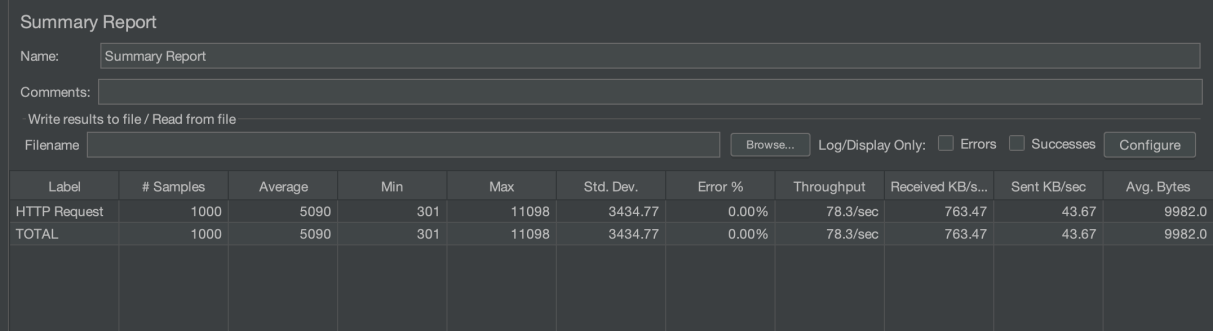


图 6-11 当线程数为1000时的汇总结果图

如图6-10所示，当线程数为1000时，较高频，错误率为0%，仍然没有错误。

1. 当线程数为2000时：

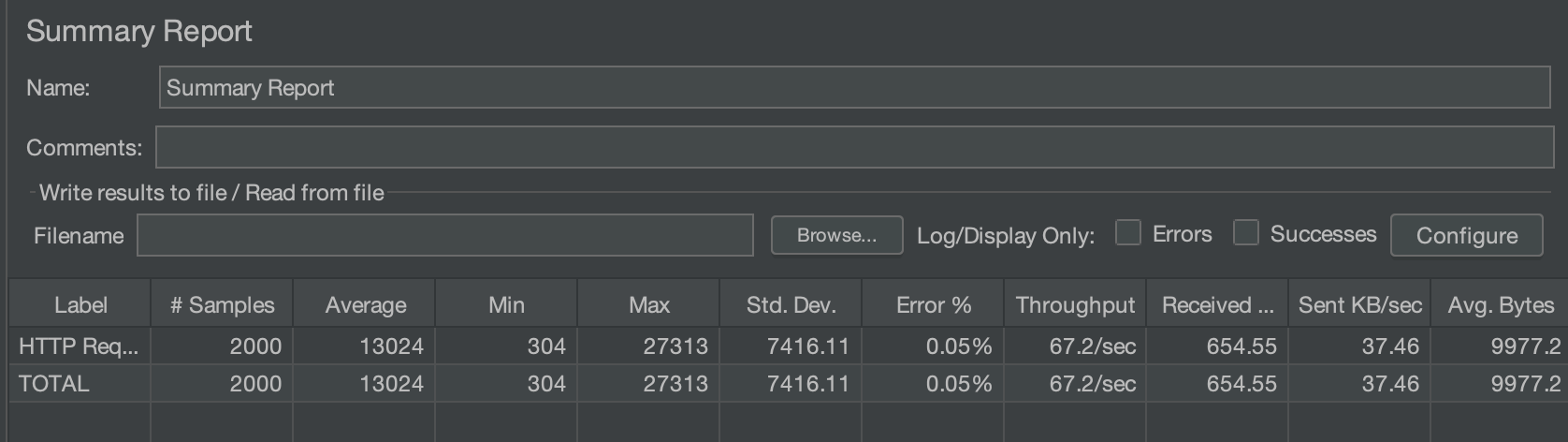


图 6-12 当线程数为2000时的汇总结果图

如图6-10所示，当线程数为2000时，请求频率颇高，错误率为0.05%，仅有1次请求错误。

1. 当线程数为1时：

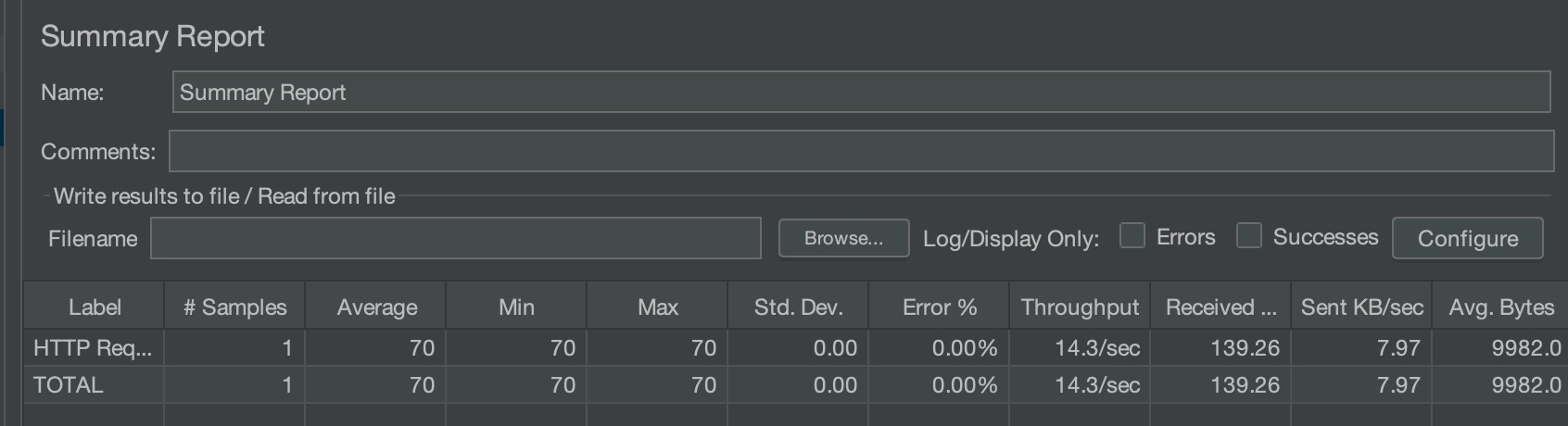


图 6-13 当线程数为1时的汇总结果图

接口单次请求的响应时间结果如图6-13所示，根据结果可得响应时间为 70ms，满足单个接口请求时间少于0.5s。

1. 汇总。

表 6-10 性能测试总结

|  |  |  |  |
| --- | --- | --- | --- |
| 测试编号 | 线程数量 | 吞吐量 | 异常率 |
| P001 | 200 | 82.0/sec | 0% |
| P002 | 500 | 71.1/sec | 0.2% |
| P003 | 750 | 75.2/sec | 0% |
| P004 | 1000 | 78.3/sec | 0% |
| P005 | 2000 | 67.2/sec | 0.05% |

表6-10为性能测试总结, 计算得出推测系统在当前的环境下QPS约为 1999/sec左右。 吞吐量 67.2/sec左右，满足系统运行需求。

# 结 论

本主题围绕“基于Spring Cloud的社交聊天系统设计与实现”展开论述。在绪论章节中通过分析陌生人社交的研究背景与意义来阐述社交聊天系统的必要性和重要性。其后，系统技术、需求分析、系统设计、系统实现、系统测试为本文的重点章节，是系统从构想到落地整个开发过程的重组成环节，主要包括以下内容:

1. 系统架构所需要用到的技术和组件。
2. 功能性与非功能性的需求分析。功能性分析，通过用例文档描述系统的主要功能。非功能性分析主要做用户使用体验相关的分析。
3. 针对系统前后端的功能设计，运用时序图配合文字讲解功能运转的过程，使用E-R图讲解了数据的设计
4. 根据上一章的设计，构建数据库表，利用开发工具实现功能设计。
5. 对整个系统进行测试，内容有接口测试、功能测试、性能测试，确保整个系统不仅可以满足功能需求，也可以满足性能需求，以保障用户体验。

本系统实现过程也存在不足之处:

1. 系统拓展性缺失，所有的模块均连接到同一台数据库，使得数据库的性能成为瓶颈。
2. 系统功能较为稀少，仅仅满足了基本的社交功能，用户需要更多元化的功能以提升用户间的互动性[11]。

# 参考文献

1. 辛园园,钮俊,谢志军,张开乐,毛昕怡.微服务体系结构实现框架综述[J].计算机工程与应用,2018,54(19):10-17.
2. 李娜.基于Spring Cloud微服务架构的应用[J].电子技术与软件工程,2019(12):142.
3. 王萍.基于Spring Security和JWT实现无状态登录[J].电子技术,2021,50(12):114-115.
4. 童敏,张黎娜,梁伍七.基于JWT的分布式系统认证授权机制设计和实现[J].合肥师范学院学报,2022,40(03):7-10.
5. 魏井辉,吕明.基于Netty通信的消息推送系统的设计与实现[J].工业控制计算机,2020,33(12):57-59.
6. 曾超宇,李金香.Redis在高速缓存系统中的应用[J].微型机与应用,2013,32(12):11-13.DOI:10.19358/j.issn.1674-7720.2013.12.004.
7. 马豫星.Redis数据库特性分析[J].物联网技术,2015,5(03):105-106.DOI:10.16667/j.issn.2095-1302.2015.03.032.
8. 唐权,周蓉,张勇.Rabbit MQ消息中间件在Spring Boot教学中的应用[J].现代信息科技,2020,4(18):125-127.DOI:10.19850/j.cnki.2096-4706.2020.18.035.
9. 王余蓝.图形数据库NEO4J与关系据库的比较研究[J].现代电子技术,2012,35(20):77-79.DOI:10.16652/j.issn.1004-373x.2012.20.045.
10. 王红,张青青,蔡伟伟,姜洋.基于Neo4j的领域本体存储方法研究[J].计算机应用研究,2017,34(08):2404-2407.
11. 陈丽珊.如何从社交软件市场分一杯羹[J].通信企业管理,2018(04):50-51.
12. 梁剑豪.网络废话的基本特征、成因及其对策探究[J].新媒体研究,2021,7(16):107-109+119.DOI:10.16604/j.cnki.issn2096-0360.2021.16.026.

# 

# 致 谢

在[论文](http://www.so.com/s?q=%E8%AE%BA%E6%96%87&ie=utf-8&src=internal_wenda_recommend_textn" \t "_blank)完成之际，我要特别感谢我的指导老师王亮明老师的热情关怀和悉心指导。在我撰写论文的过程中王亮明老师倾注了大量的心血和汗水，无论是在论文的[选题](http://www.so.com/s?q=%E9%80%89%E9%A2%98&ie=utf-8&src=internal_wenda_recommend_textn" \t "_blank)、构思和[资料](http://www.so.com/s?q=%E8%B5%84%E6%96%99&ie=utf-8&src=internal_wenda_recommend_textn" \t "_blank)的收集方面，还是在论文的[研究方法](http://www.so.com/s?q=%E7%A0%94%E7%A9%B6%E6%96%B9%E6%B3%95&ie=utf-8&src=internal_wenda_recommend_textn" \t "_blank)以及成文定稿方面，我都得到了王亮明老师的悉心细致的教诲和无私的帮助，特别是他广博的[学识](http://www.so.com/s?q=%E5%AD%A6%E8%AF%86&ie=utf-8&src=internal_wenda_recommend_textn" \t "_blank)、深厚的[学术](http://www.so.com/s?q=%E5%AD%A6%E6%9C%AF&ie=utf-8&src=internal_wenda_recommend_textn" \t "_blank)素养、严谨的治学精神和一丝不苟的工作作风使我终生受益，在此表示真诚地感谢。