



桂林航天工业学院  
GUILIN UNIVERSITY OF AEROSPACE TECHNOLOGY

# 本科毕业设计（论文）

（ 2020 届）

论文题目：基于任务驱动与多用户权限的

C/S 框架通信系统

学 院：计算机科学与工程学院

专业班级：2016 级软件工程 3 班

学 号：2016070030301

学生姓名：梁 荣 钦

指导教师：邓 维

完成日期：2020 年 5 月 15 日

桂林航天工业学院教务处制

# 桂林航天工业学院本科毕业设计（论文）

## 基于任务驱动与多用户权限的 C/S 框架通信系统

学    院： 计算机科学与工程学院

专业班级： 2016 级软件工程 3 班

姓    名： 梁荣钦

学    号： 2016070030301

指导教师： 邓    维

完成日期： 2020 年 05 月

# 学生毕业设计（论文）诚信承诺书

本人郑重声明：在毕业设计（论文）工作中严格遵守学校有关规定，恪守学术规范；所提交的毕业设计（论文）是在导师的指导下独立设计（研究）的成果，设计（论文）中引用他人的文献、数据、图件、资料及研究成果均已在设计（论文）中加以说明；在毕业设计（论文）中未剽窃、抄袭他人的学术观点、思想和成果，未篡改实验数据。与我一同工作的同组人员对本设计（论文）所做出的贡献均已在文中说明并表示了谢意。

本设计(论文)和资料若有不实之处,本人愿承担一切相关责任。

毕业设计（论文）作者（签名）： 年 月 日

指导教师（签名）：                      年    月    日

作者 联系电话:

作者 电子邮箱:

## 摘 要

随着物联网的发展，硬件数量在不断增加，由于硬件只在本地局域网中整合，而软件系统并未很好兼容硬件并发挥硬件最大性能，本论文针对此问题，提出一种基于任务驱动与多用户权限的 C/S 框架通信系统。

本文借鉴了现有的，较为成熟的技术经验，重新定义了用户与硬件之间的关系，用户基于 B/S 架构的系统，设备方面基于 C/S 架构的系统；协议采用 Websocket，前后端分离，前端主要是采用了 LayUI 来实现用户系统页面的设计,后端则采用了 SpringBoot 框架来实现后台数据的处理。数据库方面采用了 NoSQL 较为经典的 MongoDB，大文件存储采用现在比较流行的开源的 MinIO。为求得最大并发量，服务器与设备端都采用了基于 NIO 的 Netty 框架。

该系统主要实现了用户注册登录、用户验证权限和设备端信息传输等功能。该系统有效的解决了硬件与硬件，硬件与用户和用户与用户之间的通信的问题，有助于简化硬件工程师的开发与用户的使用，促进智能化的物联网的发展。

**关 键 词：**通信系统；多用户权限；任务驱动；高并发

## ABSTRACT

With the development of the Internet of Things, the number of hardware is increasing. Because the hardware is only integrated in the local area network, and the software system is not well compatible with the hardware and maximizes the performance of the hardware. C / S framework communication system with user authority.

This article draws on the existing and more mature technical experience to redefine the relationship between the user and the hardware. The user is based on the B / S architecture system, and the equipment is based on the C / S architecture system; the protocol uses Websocket, and the front and back ends are separated. The front end mainly uses LayUI to realize the design of the user system page, and the back end uses the SpringBoot framework to realize the background data processing. The database uses the classic Nogo MongoDB, and the large file storage uses the popular open source MinIO. In order to obtain the maximum concurrency, both the server and the device end use the Netty framework based on NIO.

The system mainly implements functions such as user registration and login, user authentication authority and device-side information transmission. The system effectively solves the problems of communication between hardware and hardware, hardware and users, and users and users, which helps to simplify the development of hardware engineers and the use of users, and promote the development of intelligent Internet of Things.

**Keywords:** Communication system; Multi-user authority; Task-driven; High concurrency

## 目 录

1 绪论.....	1
1.1 发展背景 .....	1
1.2 研究趋势 .....	1
1.3 研究内容 .....	2
1.4 本章小结 .....	2
2 需求分析.....	3
2.1 可行性分析 .....	3
2.1.1 技术可行性分析.....	3
2.1.2 经济上可行性.....	3
2.2 技术简介 .....	3
2.2.1 后端技术简介.....	3
2.2.2 数据库简介.....	4
2.2.3 底层技术简介.....	4
2.2.4 前端 LayUI 框架简介 .....	5
2.3 系统需求分析 .....	5
2.3.1 功能需求.....	5
2.3.2 性能需求.....	5
2.4 本章小结 .....	6
3 系统设计.....	7
3.1 系统概要设计 .....	7
3.1.1 系统功能模块图.....	7
3.2 数据库设计 .....	7
3.2.1 数据字典.....	8
3.2.2 数据库表逻辑结构的设计.....	8
3.2.3 物理结构设计.....	8

3.2.4 数据库概念结构设计 .....	10
3.3 任务驱动与多用户权限信息实时传输模型设计 .....	13
3.4 功能模块设计 .....	14
3.4.1 用户模块设计 .....	14
3.4.2 管理模块设计 .....	18
3.4.3 任务管理模块设计 .....	19
3.4.4 设备模块设计 .....	23
3.4.5 消息管理模块设计 .....	25
3.5 本章小结 .....	28
4 系统实现 .....	29
4.1 系统开发环境 .....	29
4.2 功能模块实现 .....	29
4.2.1 用户模块 .....	29
4.2.2 管理模块 .....	32
4.2.3 任务管理模块 .....	34
4.2.4 设备模块 .....	35
4.2.5 消息管理模块 .....	37
4.3 本章小结 .....	42
5 系统测试 .....	43
5.1 系统测试方法 .....	43
5.2 单元模块测试 .....	43
5.2.1 用户模块测试 .....	43
5.2.2 管理模块测试 .....	54
5.2.3 任务管理模块测试 .....	57
5.2.4 设备模块测试 .....	60
5.2.5 消息管理模块测试 .....	65
5.3 本章小结 .....	75
6 结论 .....	76

参考文献.....	77
附录.....	79
附录 A Netty 底层服务器端实现代码 .....	79
附录 B Netty 底层客户器端实现代码 .....	82
致  谢.....	88



# 1 绪论

本章主要从发展背景开始分析，用发展的眼光来分析该课题的趋势并提出本论文的研究内容。

## 1.1 发展背景

物联网的市场预测显示，物联网对全球经济产生积极的影响。未来的十年之内，大多数人认为物联网技术对经济的影响是十分大的，不同组织对物联网在全球经济影响的评估略有不同，但是与大多数人的结论却是一样的<sup>[1]</sup>。

物联网分为硬件部分与软件部分，物联网的发展与硬件的发展息息相关，硬件作为物联网软件的媒介，其重要性可想而知。硬件在发展多年之后，技术越来越成熟。因此物联网的软件发展成为了当前物联网发展的重中之重。

现在物联网发展的核心，在于物联网平台的发展。物联网平台系统包括了用户，物联网设备，以及他们之间的通信。现如今比较流行的都是以服务器为中心的系统，在软件领域，在绝大多数联网的系统中，都是使用这种形式的。因此实时数据从物联网设备产生，通过服务器的解析，指定的流向用户，然后再从用户发送控制数据，经过服务器来控制物联网设备。此外，通信协议也十分之重要，通信协议对数据的大小，解析速度以及可扩展程度有着决定性的影响。

## 1.2 研究趋势

国内外不断涌现出许多应用于单个领域的物联网控制系统（面向电力行业等）<sup>[2]</sup>，只有极少数在开发通用的物联网控制系统（基于热插拔的物联网平台）<sup>[3]</sup>，然而这些平台都十分的复杂，牵扯到了许多软件功能模块，硬件工程师需要学习软件方面的知识，学习成本十分之高等。如若使用现有的中国移动、阿里云等平台，那么需要收取一定的传输成本，并且需要重新开发面向用户的前端界面，用户后端数据处理以及数据库存取等等。

### 1.3 研究内容

本课题所研发的系统为基于任务驱动与多用户权限的 C/S 框架通信系统,使用 Netty、Springboot 与 NoSQL 做为基础,使用 Json 作为通信的载体,基于 docker 的 Java 环境,git 管理代码,拥有前台任务,后台任务以及基于前台的后台方式。使得系统可以在软件层面上远程对设备进行控制,多用户可以实时控制同一个设备并接收设备的返回信息。通过使用通用的实时的全双工通信协议,达到用户与设备之间实时数据交流。

### 1.4 本章小结

绪论主要通过研究整个物联网的发展背景开始的,从而提出本论文的主要内容。首先了解了物联网的发展背景和物联网的划分方向得知软件层面为重点,其次对比国内外物联网控制系统的现状,最后系统的开发的目的是开发具有任务驱动与多用户实时访问权限的 C/S 框架通信系统。

## 2 需求分析

本章主要从通过本论文的主要研究内容来分析出本论文是否具有一定的可行性，本章使用了多个角度分析可行性，通过技术和经济来分析是否可行。

### 2.1 可行性分析

#### 2.1.1 技术可行性分析

以 MACOS 上 Eclipse 为开发环境，对 SpringMVC 层开发，对 Netty, NoSQL, docker 与 git 加之使用，运用面向切面开发技术，设计和开发此基于任务驱动与多用户权限的 C/S 框架通信系统，主要包括：Springboot, Netty, NoSQL, docker 和 git 等，这些都是较为常用的和成熟的技术，开发起来十分的方便和稳定。

#### 2.1.2 经济上可行性

一方面，系统中选择三层结构的 MVC 系统设计，使代码的效率进一步提高，节省人力、资源；另一方面，系统的开发维护成本低，开发所使用的都是成熟的技术和常见的软件，在后期维护中也极大的节约了维护的软件和人力成本。

### 2.2 技术简介

#### 2.2.1 后端技术简介

##### 1) Java 技术简介

Java 语言是当今最流行的编程语言。不同于 C++，它完全面向对象。而且他使用了接口来实现了类似 C++ 的多重继承。Java 通过 JVM 来实现了平台无关性，可以移植到各个不同类型的系统中，并且通过 JVM 虚拟机来实现了垃圾自动收集。不同于 Python 这类解释型语言，Java 通过 JVM 来保证了代码的安全。Java 还具有强大的扩展能力并且提供一定的安全性<sup>[4]</sup>。

##### 2) SpringBoot 框架简介

SpringBoot 是近年来新开发的开发框架，在产生该项目时有明确目标，可进一步简化 Spring 应用，并且配置经过优化的配置方式，能够简化出来的应用中的多项配置，除

此外，该项目采用了大量框架，能够对过去项目存在的稳定性问题以及版本过度依赖问题提供解决方案，同时利用该项目能够使众多组建项目更好利用，逐渐改变过去 Web 应用开发模式。从技术上来看，主要包括以下几个特点：首先在利用该项目进行应用时如同点菜，可进行选配组成，生成初始项目，并能够在内部进行设置服务器和将项目打包为 Jar 压缩包，进而为 Docker 专门设计，可以将其作为 StarterPOMs 配置方式使 Maven 配置能够显著简化，SpringBoot 同时还能够为一些大型项目提供非业务功能特点，包括安全检测，健康检测，无须在项目中配置 Xml 方式<sup>[5]</sup>。系统采用兼有开放性和稳定性特点的 SpringBoot 框架的开发模式<sup>[6]</sup>。SpringBoot 是一些库的集合，只要包含相应依赖，就可以方便地被项目使用。SpringBoot 可以轻松创建独立的、可生产的、基于 Spring 且可以直接运行的应用系统。由于其可以实现微服务架构且没有复杂配置，所以在应用开发中，可以将更多精力投入到项目的业务逻辑研发中<sup>[7]</sup>。

### 2.2.2 数据库简介

#### 1) MongoDB 简介

MongoDB 是一个基于分布式，文件存储的非关系型数据库。其特点是高性能、易部署、易使用，存储数据方便。不同于传统关系型数据库将数据以表的形式进行存储，在 MongoDB 中数据是以文档的形式进行存储。文档为 BSON 格式，其内部可以包含多种类型的文件、数据也可以内嵌别的文档，模式十分自由。MongoDB 也被称作非关系型数据库当中最像关系数据库的，是一个介于关系数据库和非关系数据库之间的产品，其功能也是众多非关系数据库当中最丰富的<sup>[8]</sup>。支持复杂数据类型、自动分片、自动故障转移等功能的 MongoDB 数据库作为该系统中数据存储和管理的平台<sup>[9]</sup>。MongoDB 数据库中文档存储和修正操作使用十分简单<sup>[10]</sup>。MongoDB 在现今已经支持跨文本事务控制<sup>[11]</sup>。

### 2.2.3 底层技术简介

#### 1) Docker 简介

Docker 是虚拟化技术的其中一种，它将底层系统、依赖环境和应用程序等进行统一打包，生成一个静态镜像，通过运行，建立一个独立的容器，部署在其他的平台或宿主机上。与其他虚拟化技术不同的是，Docker 是直接运行在宿主机的操作系统之上的容器，类似独立的沙箱，容器之间完全隔离，每个容器可以独立管理、配置<sup>[12]</sup>。采用 Docker

容器化技术进行分布式和平台的部署，更加易于维护<sup>[13]</sup>。docker 实现了一种名为 LXC（LinuxContainers）的轻量级虚拟化技术，借助 Linux 特性，如控件组（control group）和命名空间（namespace），实现了更彻底的容器隔离<sup>[14]</sup>。

## 2) Netty 简介

Netty 网络框架底层封装了 Java NIO<sup>[15]</sup>，具有使用简便、安全性高、性能优越等特点，并且 Netty 采用异步传输的方式<sup>[16]</sup>。Netty 可用于敏捷开发高性能、高可靠性的网络应用程序<sup>[17]</sup>。其采用 IO 多路复用技术，处理 IO 操作，把多个 IO 阻塞复用到一个 select 之上，满足单线程处理多个 IO 的连接需求<sup>[18]</sup>。这个框架支持多种通信协议，包括 UDP、TCP 以及文件传输协议等<sup>[19]</sup>。

## 3) MinIO 简介

MinIO 是一个基于 Apache License v2.0 开源协议的对象存储服务。它兼容亚马逊 S3 云存储服务接口，非常适合于存储大容量非结构化的数据，例如图片、视频、日志文件、备份数据和容器/虚拟机镜像等，而一个对象文件可以是任意大小，从几 kb 到最大 5T 不等<sup>[20]</sup>。

### 2.2.4 前端 LayUI 框架简介

Layui 是一款采用自身模块规范编写的国产前端 UI 框架，遵循原生 HTML / CSS / JS 的书写与组织形式。采用 Layui 可以缩减项目开发的周期<sup>[21]</sup>。

## 2.3 系统需求分析

### 2.3.1 功能需求

有下面的系统基本流程和功能。主要表现基于任务驱动与多用户权限的 C/S 框架通信系统的功能性流程。本系统采用 Java、SpringBoot 框架、Netty、MinIO 和 MongoDB 结合开发。从系统使用人员来说，可以分为设备、使用人员和平台管理者。使用人员可以进行个人信息管理、设备管理、信息通信、信息通知和任务管理等，设备可以信息管理，信息通信，任务下载等，管理员则可以管理对用户进行管理和发布系统公告等。如图 2-1 所示：

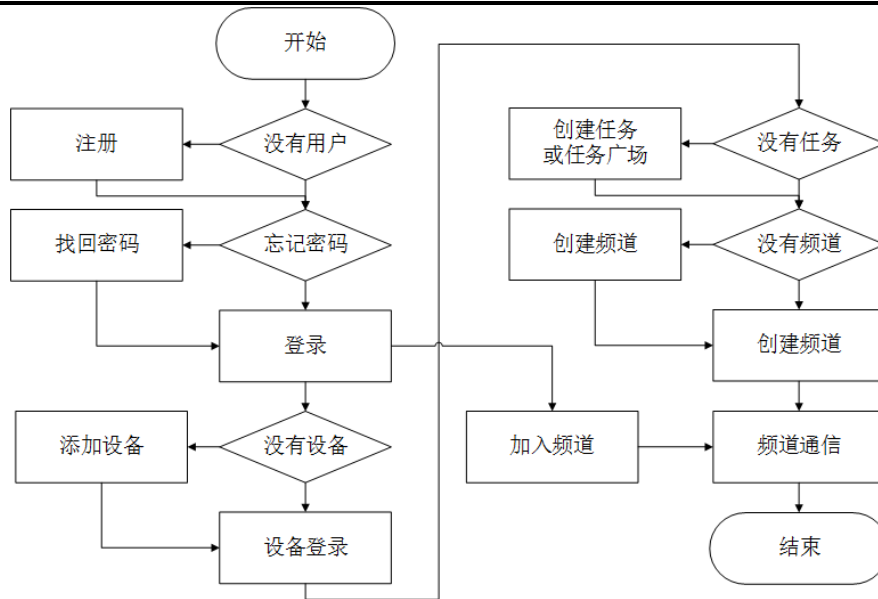


图 2-1 系统基本流程图

### 2.3.2 性能需求

基于任务驱动与多用户权限的 C/S 框架通信系统，通过本系统，多个用户可以实时获取设备信息或者对设备进行控制，使得用户快速高效的获得资讯。有效的提高了用户的使用效率。

本系统的目的主要是多用户通过任务驱动实时控制设备，本系统应包含以下内容：

- (1) 系统的使用角色分为设备、用户和管理员。
- (2) 用户功能：登录、修改密码、找回密码、注册、基础用户信息。
- (3) 消息管理、设备管理、任务管理等。
- (4) 管理员：管理用户，系统公告等。
- (5) 设备：信息传输、信息解析处理、登录验证等。

## 2.4 本章小结

本章从可行性分析、技术简介到系统需求分析，来从不同角度的对系统进行需求分析。在技术使用上，Java、MongoDB、Docker 和 Netty 都有较好实现系统的优势。在功能、性能需求中，更好的突出基于任务驱动与多用户权限的 C/S 框架通信的需求性。

## 3 系统设计

本章为系统设计的分析，通过本章的讨论，可以得出系统的整体架构，数据库设计和每个功能的详细设计。

### 3.1 系统概要设计

#### 3.1.1 系统功能模块图

基于任务驱动与多用户权限的 C/S 框架通信系统功能模块图如下：

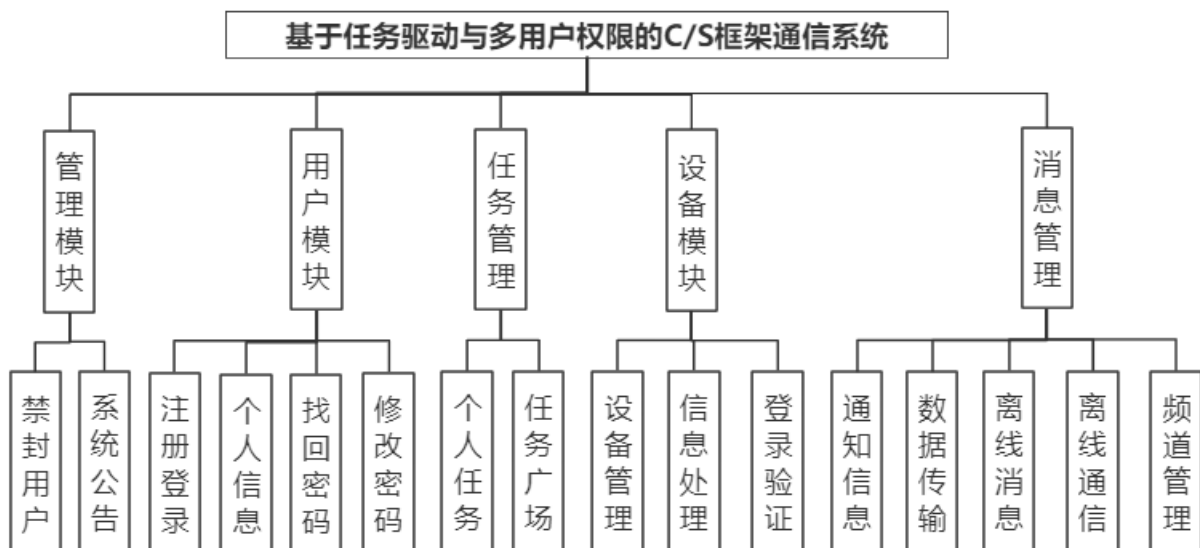


图 3-1 系统功能模块图

由上系统功能模块图可以得到，管理模块拥有封禁用户和系统公告的功能；用户模块拥有注册登录，个人信息，找回密码和修改密码的功能；任务模块拥有个人任务和任务广场的功能；设备端模块拥有登录验证，设备管理和信息处理的功能；消息管理拥有通知信息、数据传输、离线消息、离线通信和频道管理的功能。

### 3.2 数据库设计

通过上述分析出该系统的结构，通过分析每一个模块所需要的数据，来得出数据库的基本元素，通过数据库理论分解，来得出数据库的内容。

### 3.2.1 数据字典

本系统的数据字典如表 3-1 所示：

表 3-1 数据字典

表名	描述	包含的信息
user（用户登录）	记录用户登录信息	账号、密码、权限、上次离线时间、所属人
userinfo（个人信息）	记录用户个人信息	账号、用户名、性别、邮箱、手机号码、个性签名、个人网址、照片
message（消息）	记录实时通信信息	类型、数据、时间、发送者、接收者
task（任务）	记录用户任务信息	介绍、使用帮助、软件位置、脚本位置、是否公开、所属人
usergroup（用户频道）	记录用户之间通信群组，并且控制权限	创立者、任务名称、频道名称、频道号、成员频道规则

### 3.2.2 数据库表逻辑结构的设计

本系统的数据库表逻辑结构如表 3-2 所示：

表 3-2 数据库表逻辑结构

表名	表字段
(1) 用户登录信息表	id、账号、密码、登录权限、所属人、上次离线时间
(2) 用户信息表	id、账号、名字、性别、邮箱、手机号码、个性签名、个人网址、照片
(3) 任务信息表	id、任务介绍、功能介绍、软件位置、是否公开、所属人 id、任务名称
(4) 消息信息表	id、类型、数据、时间、发送者、接收者
(5) 频道信息表	id、创立者、频道名称、任务名称、频道 id、成员频道规则

### 3.2.3 物理结构设计

本系统的数据表物理结构如下表所示：



表 3-3 用户登录信息表

列名	字段描述	数据类型	允许为空
Id	Id	Integer	否
Username	账号	String	否
Password	密码	String	否
Role	权限	String	否
Owneduser	所属人	String	否
Time	上次离线时间	Integer	是

表 3-4 用户信息表

列名	字段描述	数据类型	允许为空
Id	id	Integer	否
Username	账号	String	否
Name	名字	String	是
Sex	性别	String	是
Email	邮箱	String	是
Phone	手机号码	String	是
Evaluate	个性签名	String	是
Site	网址	String	是
Picture	照片	String	是

表 3-5 任务信息表

列名	字段描述	数据类型	允许为空
Id	任务 id	Integer	否
Name	任务名称	String	否
Intro	任务介绍	String	否
Help	功能介绍	String	否
Path	软件位置	String	否
Show	是否公开	String	否

表 3-5（续）

列名	字段描述	数据类型	允许为空
User	所属人	String	否
JSPath	软件脚本位置	String	否

表 3-6 消息信息表

列名	字段描述	数据类型	允许为空
Id	id	String	否
Type	类型	String	否
Data	信息	String	是
Sender	发送者	String	是
Addressee	接收者	String	否
Time	时间	Integer	否

表 3-7 频道信息表

列名	字段描述	数据类型	允许为空
Id	id	String	否
Builder	创立者	String	否
Name	频道名称	String	否
Task	任务名称	String	否
Channel	频道 id	String	否
Roles	成员频道规则	Map	否

### 3.2.4 数据库概念结构设计

在该系统中，根据该系统所需的主要功能，有以下为系统实体图：

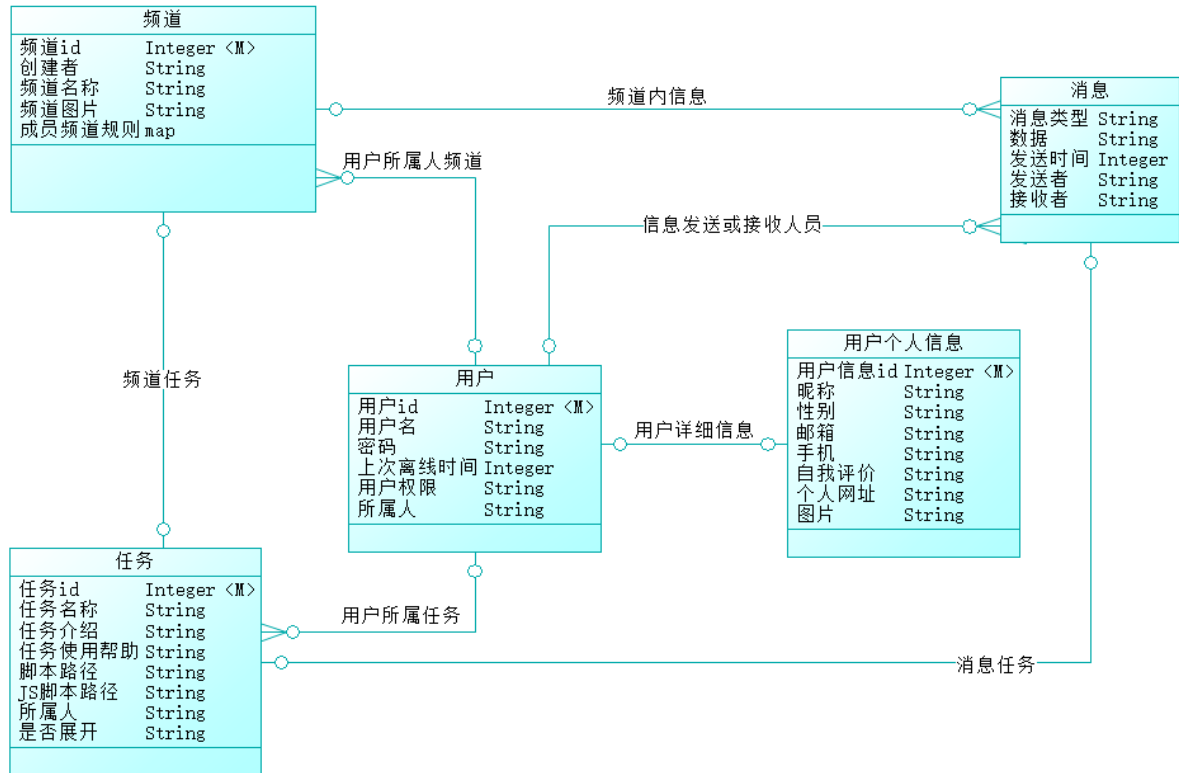


图 3-2 系统实体图

(1) 用户登录实体有以下属性：id、账号、密码、登录权限、所属人、时间。如图 3-3 所示：

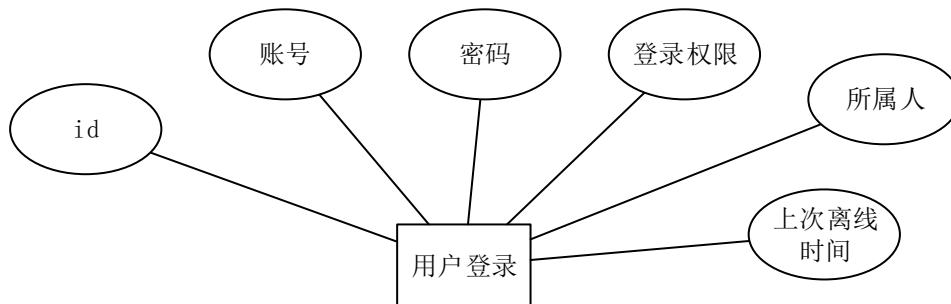


图 3-3 用户登录实体图

(2) 用户实体有以下属性：id、账号、名字、性别、邮箱、手机号码、个性签名、个人网址、照片。如图 3-4 所示：

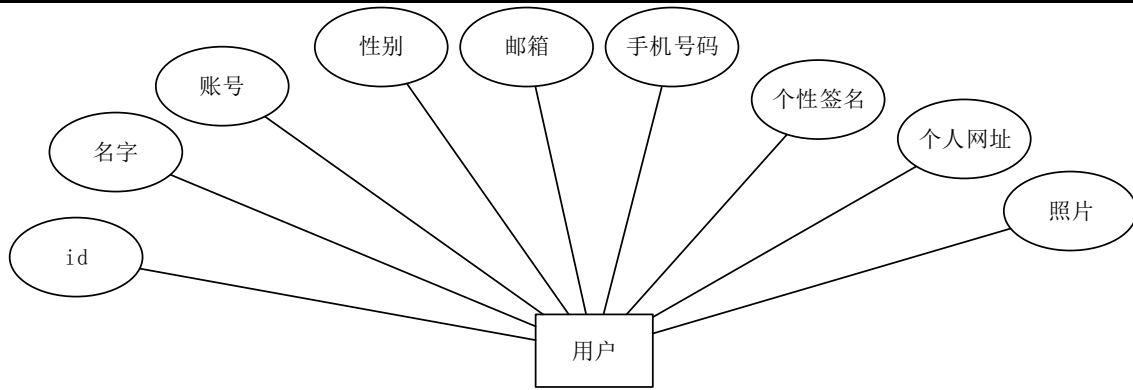


图 3-4 用户实体图

(3) 任务实体有以下属性：id、任务介绍、功能介绍、软件位置、脚本位置、是否公开、所属人 id、任务名称。如图 3-5 所示：

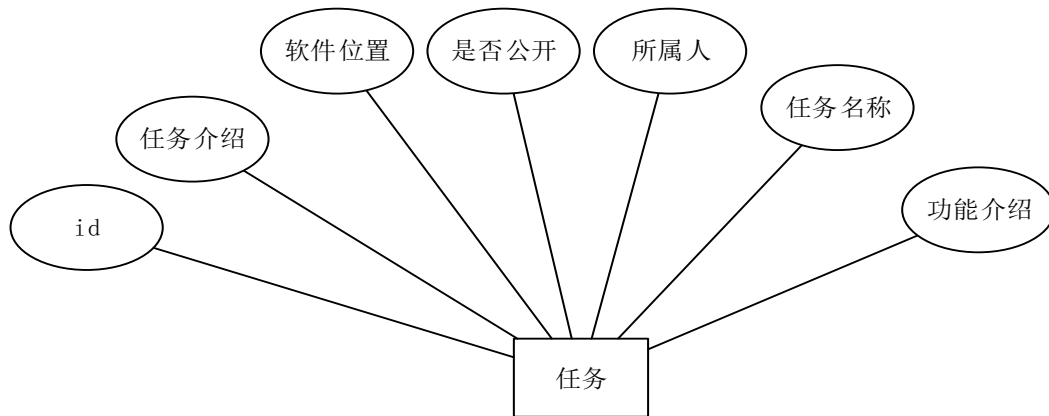


图 3-5 任务实体图

(4) 消息实体有以下属性：id、类型、数据、时间、发送者、接收者。如图 3-6 所示：

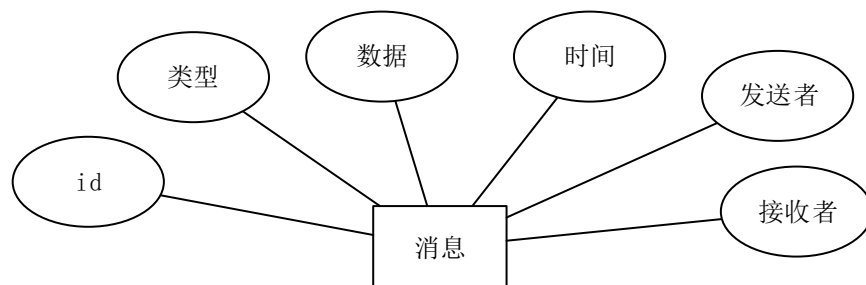


图 3-6 消息实体图

(5) 频道实体有以下属性：id、创立者、频道名称、任务名称、频道 id、可读写成员组、只读成员组。如图 3-7 所示：

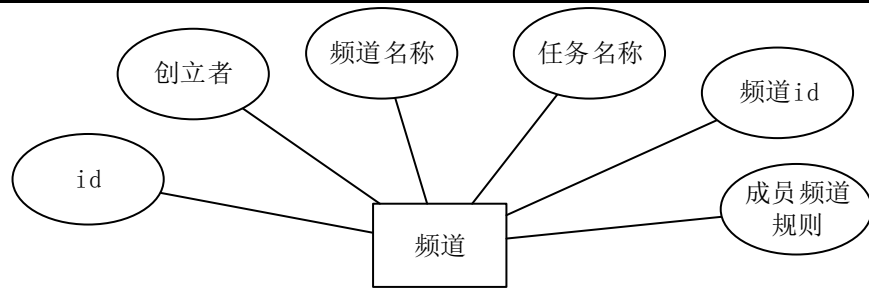


图 3-7 频道实体图

### 3.3 任务驱动与多用户权限信息实时传输模型设计

该模型采用 netty 实现 NIO 底层，使用 websocket 作为上层协议，可以在网页运行。websocket 负责实时通信；在线任务驱动与多用户权限管理均由频道负责；任务负责保存，上传，下载任务；离线任务驱动交由 Springboot 中的定时任务 + Email 负责。以下为该模型流程图：

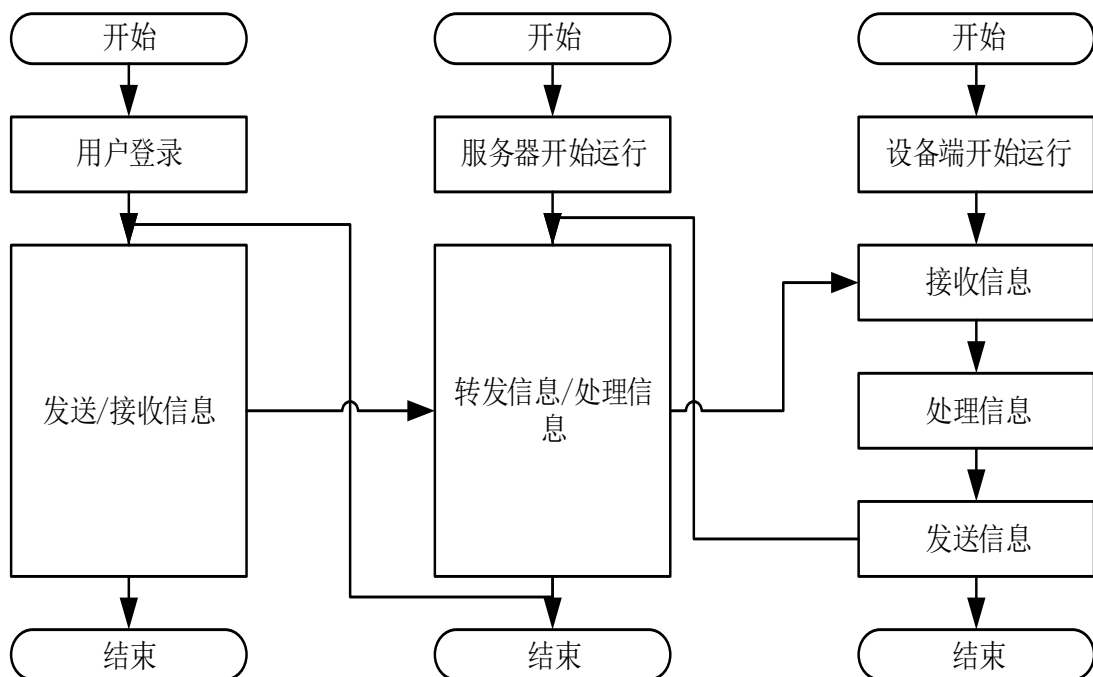


图 3-8 通信模型设计流程图

### 3.4 功能模块设计

#### 3.4.1 用户模块设计

##### 1) 登录模块设计

在该模块的要求就是实现管理员和用户两种类型角色的登录。首先是对用户名与密码验证是否为空，然后将密码一致在验证用户名与密码是否与数据库中一致，都可以通过验证即可以进入相应主页面，验证失败则需要重新输入账号与密码。

登录模块设计流程图如下：

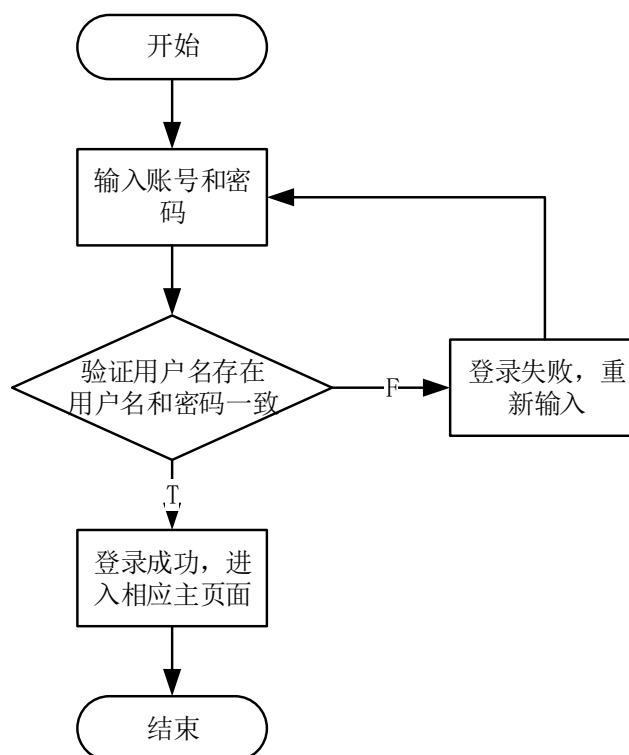


图 3-9 登录模块设计流程图

##### 2) 注册模块设计

在该模块的要求就是实现用户的注册身份信息。首先需要用户名与密码两个信息，验证两次密码是否一致，在验证是否用户名已存在数据库中，存在即失败，如果全部都成功就注册成功。

注册模块设计流程图：

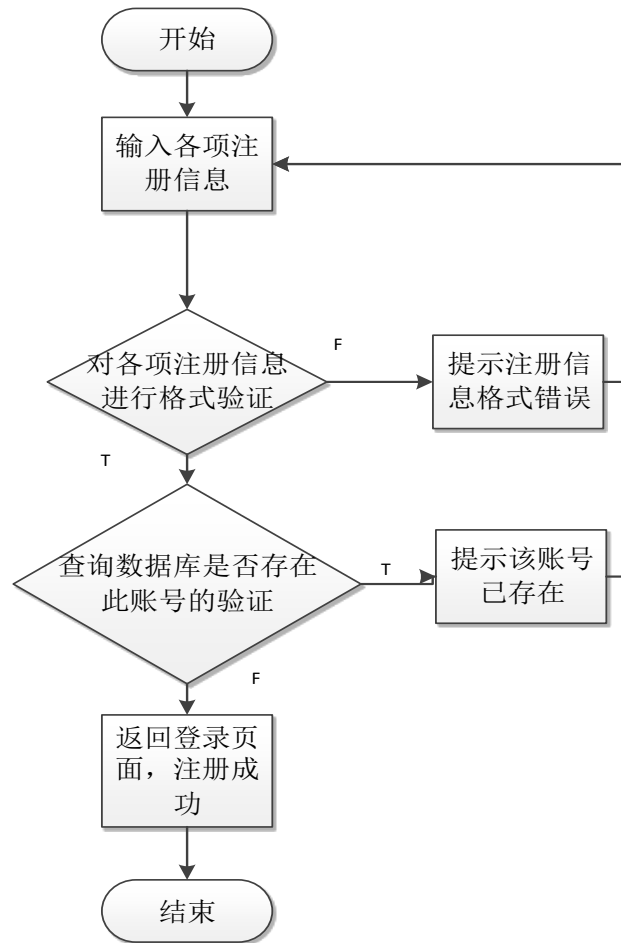


图 3-10 注册模块设计流程图

### 3) 找回密码模块设计

找回密码模块的设计最主要就是通过用户绑定的邮箱来验证用户是否是真正的此账号的主人，通过邮件模块发送邮件到该用户的邮箱中，获取邮箱中验证码并验证，验证成功后修改密码。

找回密码模块设计流程图：

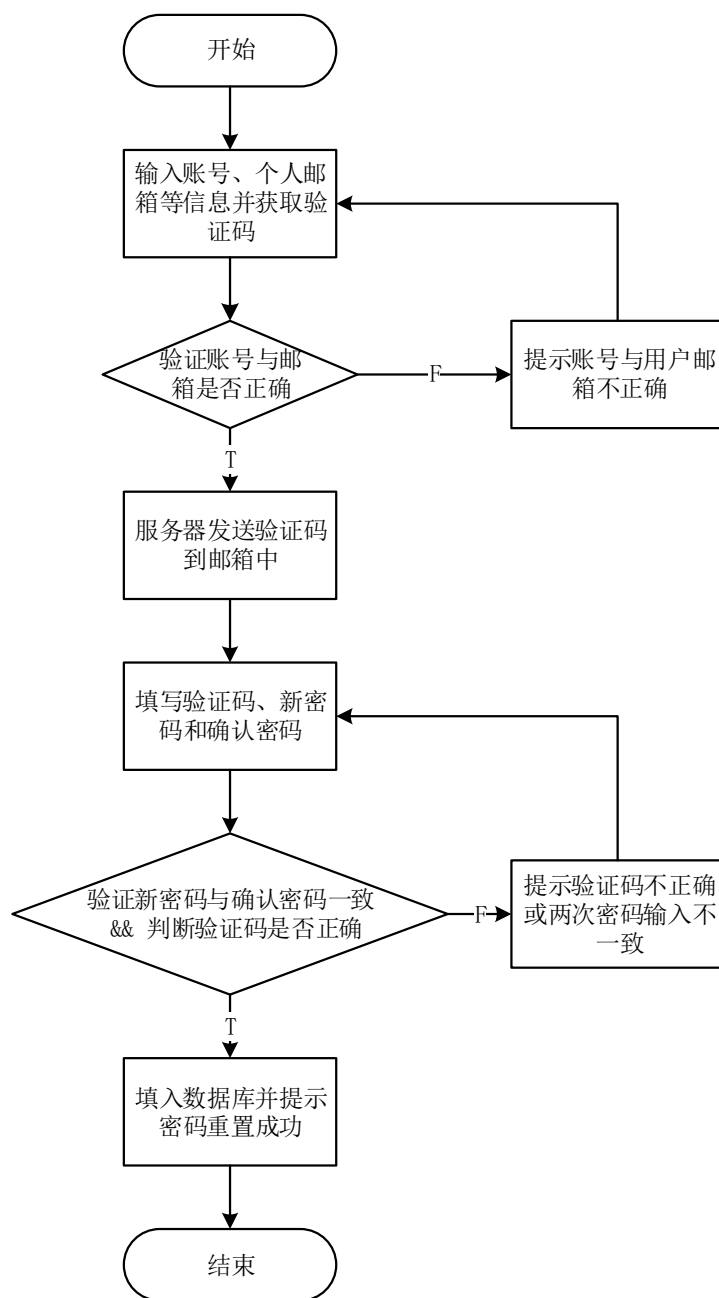


图 3-11 找回密码模块设计流程图

#### 4) 修改密码模块设计

修改密码模块的设计最主要就是验证原密码和两次新密码是否一致。修改密码模块设计流程图如下：



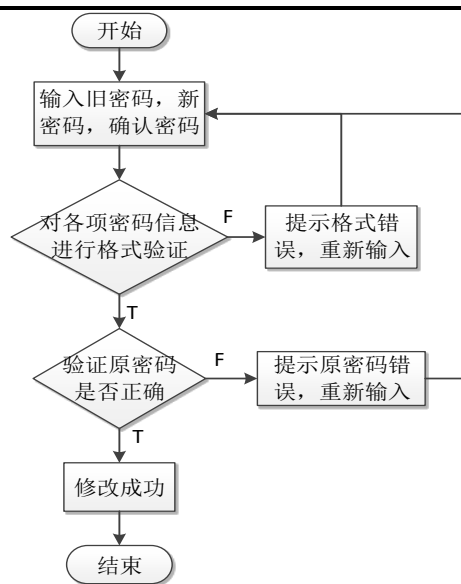


图 3-12 修改密码模块设计流程图

### 5) 个人信息模块设计

个人信息模块的设计最主要就是验证格式是否一致。如果一致直接保存到数据库当中。个人信息模块设计流程图如下：

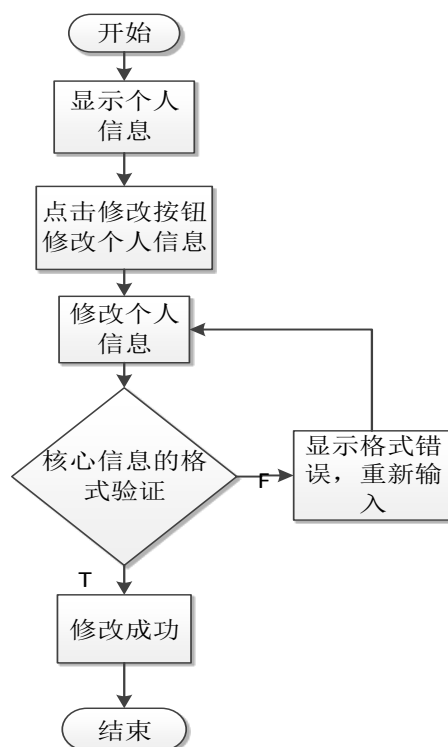


图 3-13 个人信息模块设计流程图

### 3.4.2 管理模块设计

#### 1) 封禁用户模块设计

在该模块的要求就是实现管理员通过权限管理用户不能进行登录。用户、管理员与设备每次登录时必须获得相应的实体权限才能登录成功，一旦改变该权限即可登录失败。

此模块设计流程图如下：

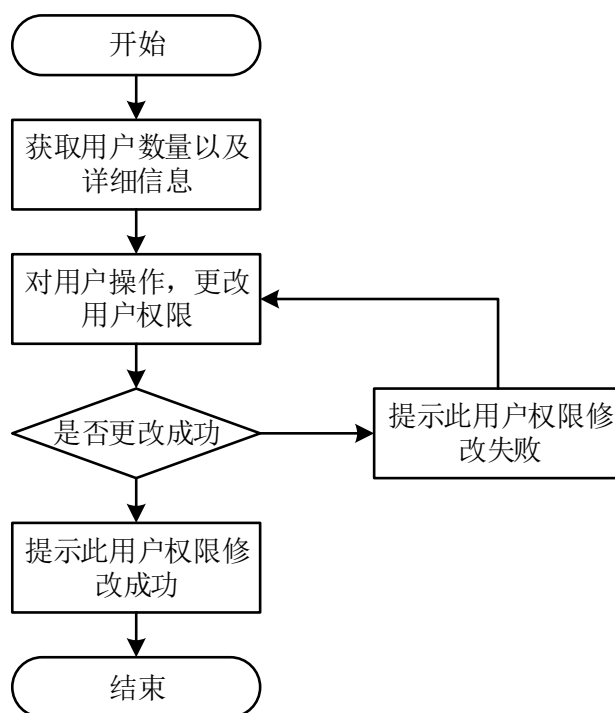


图 3-14 封禁用户模块设计流程图

#### 2) 系统公告模块设计

在该模块的要求就是实现管理员通过静态页面上传公告并反映到每一个在线的频道中。通过静态页面来上传公告内容，通过在线频道发送到每一个系统在线的位置。

此模块设计流程图如下：

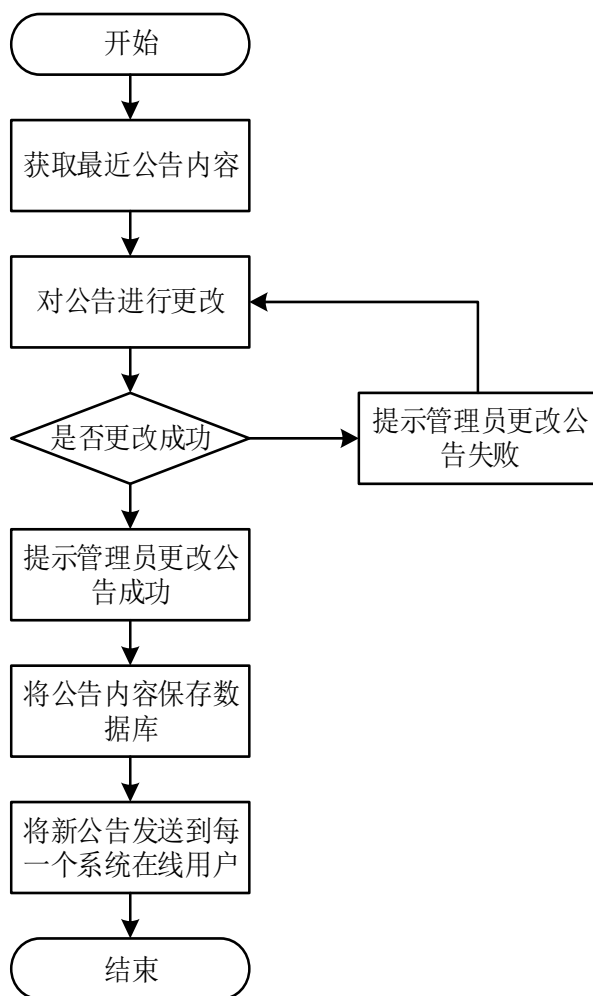


图 3-15 系统公告模块设计流程图

### 3.4.3 任务管理模块设计

#### 1) 添加任务模块设计

在该模块的要求就是实现用户通过界面添加任务的功能，添加内容包括上传任务文件、任务介绍、功能介绍、任务名称等。

此模块设计流程图如下：

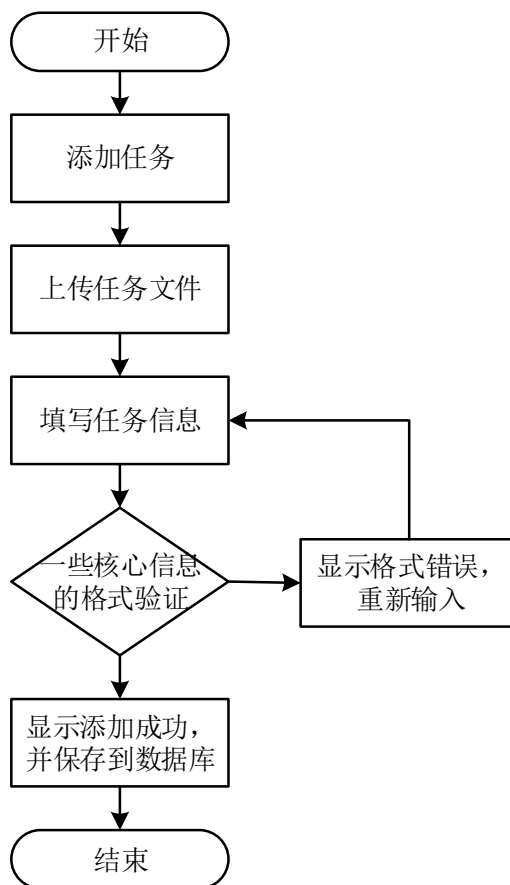


图 3-16 添加任务模块设计流程图

## 2) 修改任务模块设计

在该模块的要求就是实现用户通过界面修改任务，修改内容包括重新上传任务文件、任务介绍、功能介绍、任务名称等。

此模块设计流程图如下：

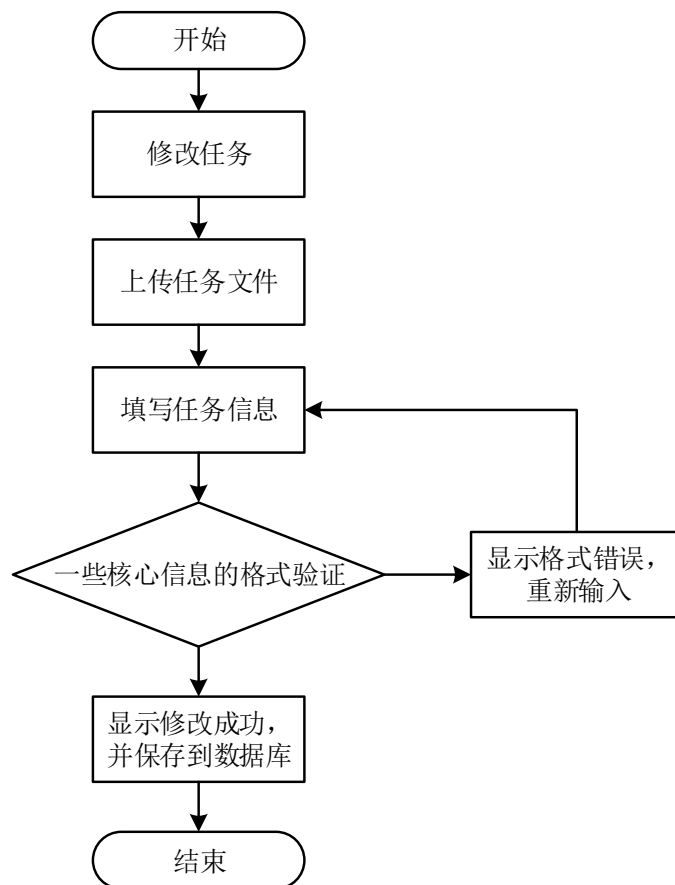


图 3-17 修改任务模块设计流程图

### 3) 删除任务模块设计

在该模块的要求就是实现用户删除不需要的任务。通过任务 id 进行删除，删除时同步删除任务文件与数据库中任务内容。此模块设计流程图如下：

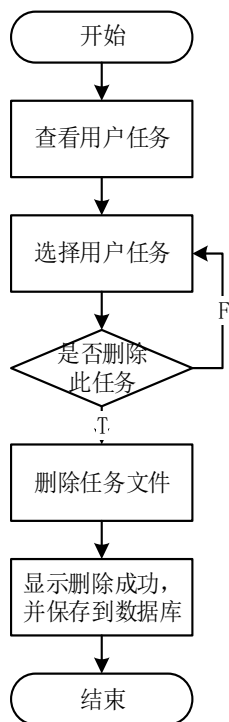


图 3-18 删除任务模块设计流程图

#### 4) 任务广场模块设计

在该模块的要求就是实现查看其它用户分享的任务，可以下载文件。通过数据传输使用任务 id 可以在自己设备上安装使用此任务。此模块设计流程图如下：

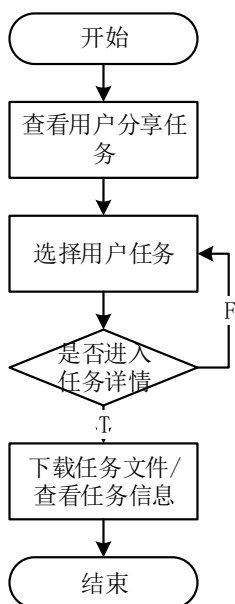


图 3-19 任务广场模块设计流程图

### 3.4.4 设备模块设计

#### 1) 设备管理模块设计

在该模块的要求就是用户可以新建设备，修改设备信息以及删除设备。

此模块设计流程图如下：

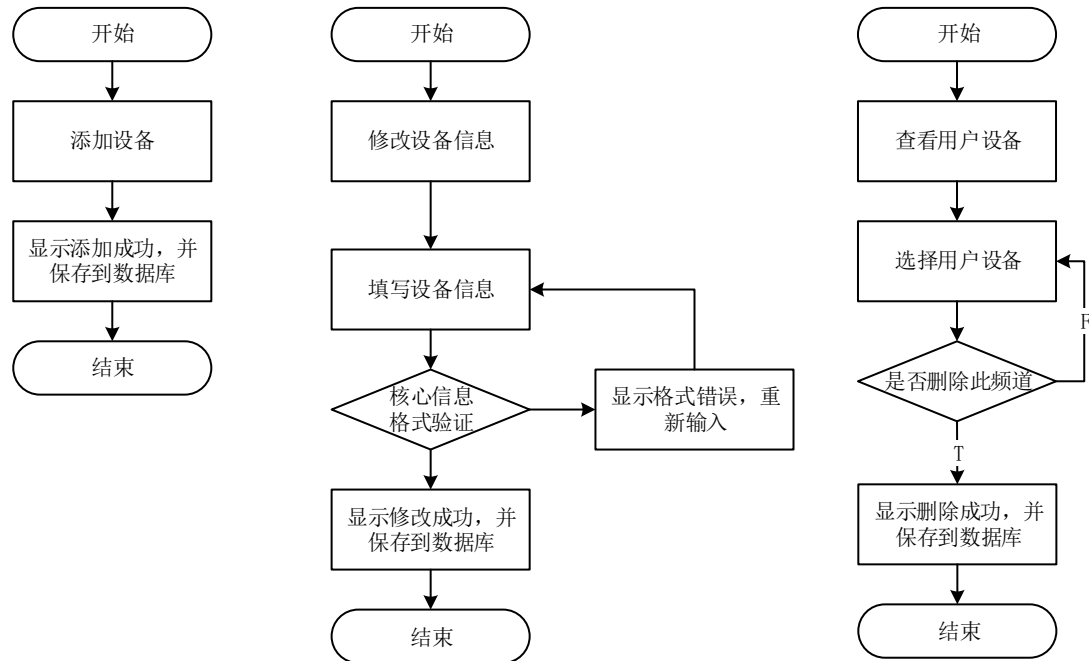


图 3-20 设备模块设计流程图

#### 2) 信息处理模块设计

在该模块的要求就是实现用户通过数据传输到设备时，设备接收到信息并进行解析处理。设备接收到信息后，首先进行类型解析，查看该任务是否为系统任务，如果为系统任务，转发到系统任务中，如果不是再进行频道分析，确定该数据传输到哪个任务模块，设备将数据传输到任务模块中。

此模块设计流程图如下：

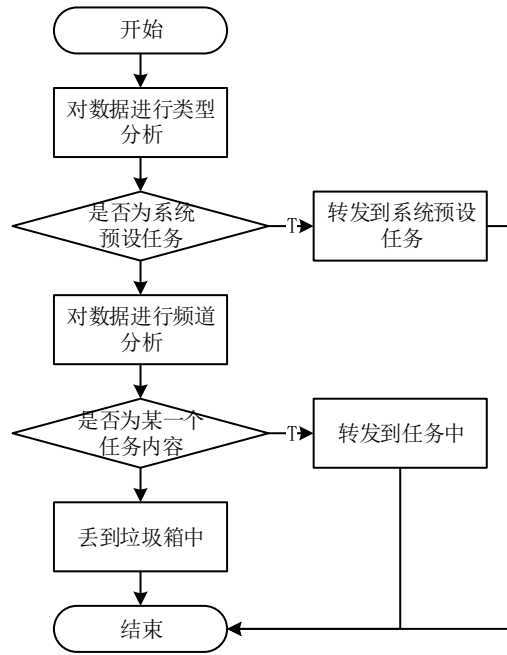


图 3-21 信息处理模块设计流程图

### 3) 登录验证模块设计

在该模块的要求就是实现设备角色的登录。首先是对用户名与密码进行是否为空的验证，其次需要判断的是用户名和密码是否与数据库是否一致，都可以通过验证即可以进入相应主页面，验证失败则需要重新输入用户名与密码。

此模块设计流程图如下：

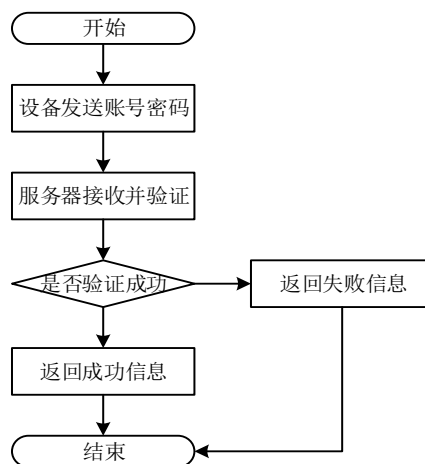


图 3-22 登录验证模块设计流程图



### 3.4.5 消息管理模块设计

#### 1) 数据传输模块设计

在该模块的要求就是将用户/设备需要内容发送到该用户/设备中。

此模块设计流程图如下：

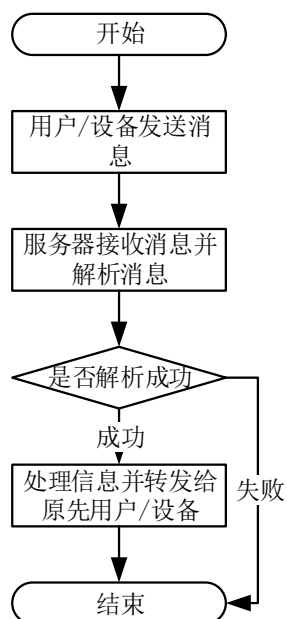


图 3-23 数据传输模块设计流程图

#### 2) 离线消息模块设计

在该模块的要求就是实现消息的加载，每一个频道都有只有一条消息链表，通过时间先后将消息连接在一起。在数据传输模块当中已经实现在线消息保存到数据库中。因此该模块就是在使用实时通信时可以加载之前的信息。

此模块设计流程图如下：

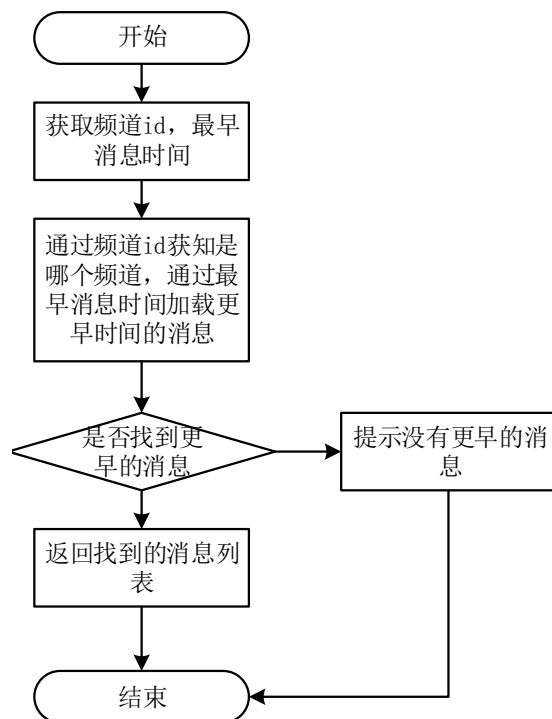


图 3-24 离线消息模块设计流程图

### 3) 离线通信模块设计

在该模块的要求就是通过 Email 实现离线消息获取，此模块用于用户一直未上线而有重要消息需要处理。该模块使用 Spring 中定时任务，每天一次将离线消息推送到用户邮箱中。如果消息非常重要，当频道有信息而用户不在线时，也可启动 Email 推送信息。此模块设计流程图如下：

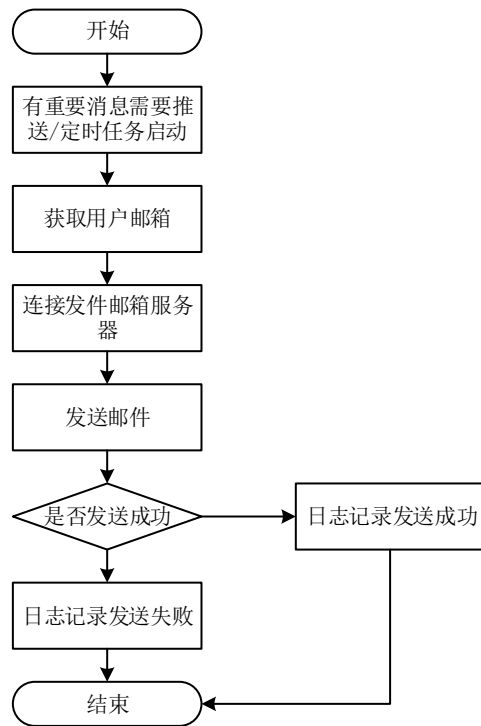


图 3-25 离线通信模块设计流程图

#### 4) 频道管理模块设计

在该模块的要求就是实现用户通过频道管理来管理设备运行任务中实时通信与多个用户的使用任务权限与通信权限。

此模块设计流程图如下：

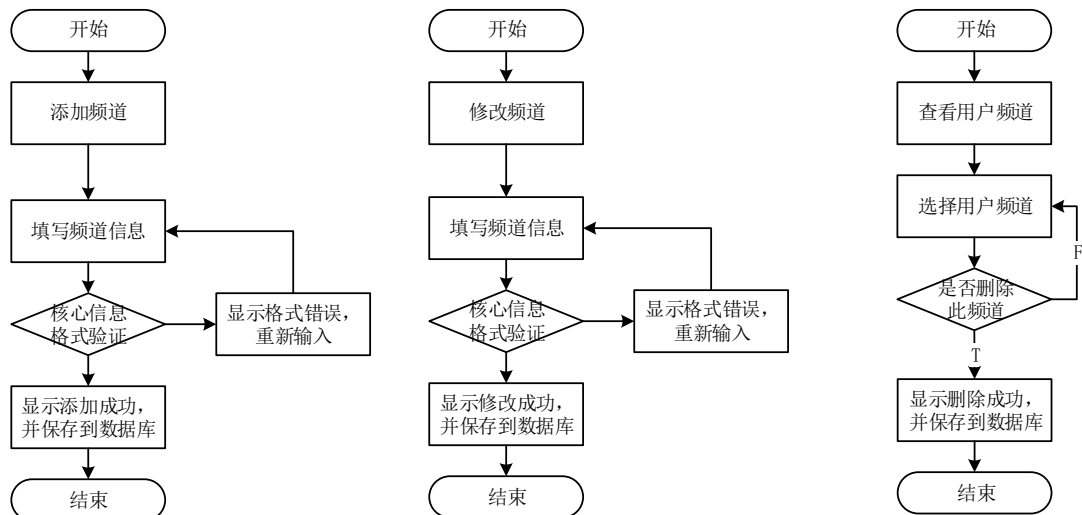


图 3-26 频道管理模块设计流程图

### 5) 通知信息模块设计

在该模块的要求就是实现用户在进行一些特殊的操作时对被操作方发送通知信息并记录在通知信息中。

此模块设计流程图如下：

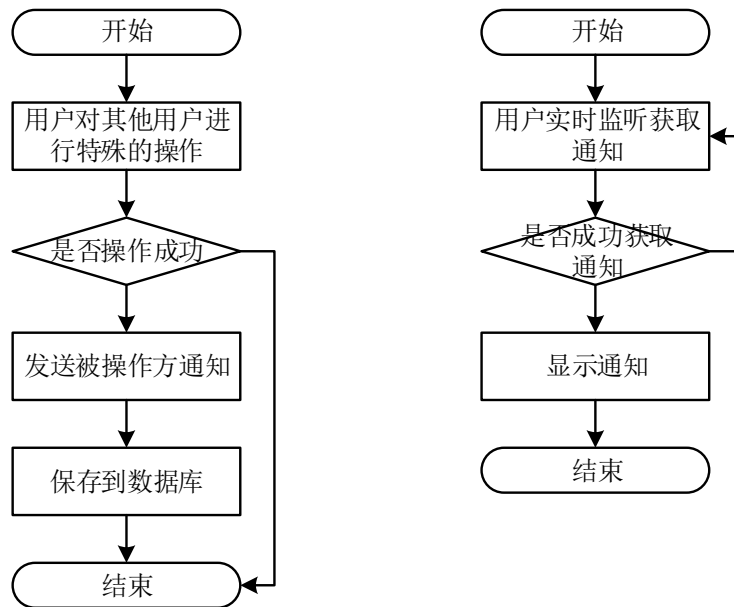


图 3-27 通知信息模块设计流程图

## 3.5 本章小结

本章从系统的概要设计开始进行分析整个系统的结构，然后通过整个系统结构来决定数据库的表结构，然后提取整个系统最核心部分进行讨论，最后将整个系统按功能划分并把全部功能逐一分析。

## 4 系统实现

该章具体讨论了系统的实现内容，以及实现内容的简单介绍等。

### 4.1 系统开发环境

开发环境：系统使用 MACOS 开发，开发软件为 Eclipse，数据库使用为非关系型数据库 MongoDB；对象存储采用 MinIO 开源，用户短链接采用 SpringBoot，用户长连接底层采用 Netty，协议采用比较广泛的 HTML5 下一种新的 Websocket 协议<sup>[22]</sup>。

操作系统：MacOS 10.14.6

技术框架：用户后端采用 SpringBoot 框架,前端使用的是 LayUI 框架，使用前后端分离，实时通信中服务器采用 Netty 框架实现，设备端与服务器采用的是 C/S 架构。

### 4.2 功能模块实现

#### 4.2.1 用户模块

##### 1) 登录模块

该模块可以作为用户登录、管理员登录入口，使用了 SpringBoot 中的 security 做登录管理；红色字体为使用说明：

使用说明：用户名与密码不得低于6位数！

用户名	<input type="text" value="qweasd"/>
	<a href="#">没账号？注册一发</a>
密码	<input type="password" value="....."/>
	<a href="#">忘记密码？找回一波</a>
	<input type="button" value="立即登陆"/> <input type="button" value="重置"/>

图 4-1 登录界面图

## 2) 注册模块

此模块为用户注册模块，首先检查两次密码是否一致，在检查用户名是否存在，不存在就将注册内容写入表中并返回成功。

使用说明：用户名与密码不得低于6位数！

用户名	<input type="text" value="qweasd"/>
新密码	<input type="password" value="....."/>
确认密码	<input type="text" value="请确认密码"/>
<input type="button" value="立即注册"/> <input type="button" value="重置"/>	

图 4-2 注册界面图

## 3) 找回密码模块

此模块为找回密码模块，首先是检查用户名和邮箱是否存在，不存在就无法找回密码，之后获取邮箱中验证码，填写信息后比较两次密码是否一致后，验证验证码是否一致，如一致就将密码写入表中并返回成功。

使用说明：用户名与密码不得低于6位数！

用户名	<input type="text" value="qweasd"/>
电子邮箱	<input type="text" value="请输入你的电子邮箱"/>
验证码	<input type="text" value="请输入邮箱中的验证码"/>
<input type="button" value="验证邮箱并发送验证码"/>	
新密码	<input type="password" value="....."/>
确认密码	<input type="text" value="请确认密码"/>
<input type="button" value="立即修改"/> <input type="button" value="重置"/>	

图 4-3 找回密码界面图

#### 4) 修改密码模块

此模块为修改密码模块，首先验证两次密码是否一致，再检查旧密码与原来密码是否一致，全一致后修改密码。

新密码必须两次输入一致才能提交

用户名	qweasd
旧密码	请输入旧密码
新密码	请输入新密码
确认密码	请确认密码

立即修改重置

图 4-4 修改密码界面图

#### 5) 个人信息模块

此模块是个人信息模块，加载页面时异步加载个人信息，修改信息后，检查全部可填写项是否填写完整，且格式一致，一致后写入数据库中。

用户名

qweasd

个人昵称

梁荣钦测试号

性别

☒ 男 ☐ 女 ☐ 保密

手机号码

13580640725

个人网址

https://MasterMind.ink/

邮箱

root@mastermind.ink

个性签名

庚子年闰四月初一

立即提交

重置

+ 点击即可换头像

注意：图片控制在1M以下!




图 4-5 个人信息界面图

#### 4.2.2 管理模块

##### 1) 封禁用户模块

此模块为管理员专用模块，为封禁用户模块，加载页面时自动加载全部用户数据，在操作中可封禁用户和解封用户。













用户id	用户 名	用户 权限	所属 用户	最后退出时 间	操作
5e8d78cbda460f0bc7e521cf	zxczx c	ROLE _USE R	zxczx c	2020-05-24 08:42	 解禁  禁用
5e8d78dada460f0bc7e521d0	qweq we	ROLE _USE R	qweq we	2020-05-23 20:51	 解禁  禁用
5e8de4e95c79ad12994414eb	admin s	ROLE _ADM IN	admin s	2020-05-25 16:34	 解禁  禁用
5e8ee9e08f779005426f10a9	19970 302	ROLE _USE R	19970 302	2020-04-16 00:53	 解禁  禁用
5e8f52eded2dae0a23e3ec3a	asdas d	ROLE _USE R	asdas d	2020-05-23 15:08	 解禁  禁用

图 4-6 封禁用户界面图

2) 系统公告模块

此模块为管理员专用模块，为系统公告模块，该模块提交后可向全部在线用户发送信息，并将信息保存到数据库中，供离线用户访问。

公告标题

请输入公告标题

文章内容

立即提交

重置

图 4-7 系统公告界面图

### 4.2.3 任务管理模块

#### 1) 添加/修改任务模块

此模块流程为上传两个文件后获取文件 id 后写入输入框中，也可以复制其他任务的 ID 使用。提交信息后检查是否符合格式，符合格式写入数据库中。

添加任务

任务名称

是否公开 ☐ 公开 ☐ 不公开

软件位置

软件显示脚本位置

任务介绍

功能介绍

图 4-8 添加/修改任务界面图

#### 2) 删除任务模块

此模块功能为删除任务，通过匹配 id 删除任务，删除任务时检查任务是否属于删除用户者，不属于就删除失败。属于就删除成功。



图 4-9 删除任务界面图

3) 任务广场模块

此模块功能为查看所有用户共享的任务，目的是为了帮助萌新用户使用更加快捷方便。



图 4-10 任务广场界面图

4.2.4 设备模块

1) 添加设备模块

此模块功能为添加设备，该模块不需要填写任何内容就自动生成。



图 4-11 添加设备界面图

2) 修改设备模块

此模块功能为修改设备信息，加载页面时自动获取设备信息，设备密码无法获取。设备密码为明文密码。提交后检查是否属于该用户，属于就写入数据库中。



图 4-12 修改设备界面图

3) 删除设备模块

此模块功能为删除设备，删除设备时需验证设备属于该用户，属于即删除成功。

<input type="checkbox"/>	用户ID	所属人	当前操作权限	操作
<input type="checkbox"/>	5e9425bfed2dae076e169e95	qweqwe	ROLE_DEVICE	<a href="#">编辑</a> <a href="#">删除</a>

图 4-13 删除设备界面图

#### 4) 信息处理模块

此模块为设备端专用模块，为数据处理模块，无界面。获取信息后检查是否为 WebSocketFrame，是就开始解析信息，解析信息后如为使用任务信息，即创建新进程处理任务信息，如为下载任务信息即获取该 BinaryWebSocketFrame，并写入文件中。如为其他信息，打印到控制台并丢弃。

#### 5) 登录验证模块

此模块为设备端专用模块，为登录验证模块，设备开始运行时加载该模块，为设备端必备模块，该模块加载失败即无法使用设备端全部内容。在频道信息读取成功后加载模块，加载后由信息处理模块处理服务器信息。

### 4.2.5 消息管理模块

#### 1) 数据传输模块

此模块为服务器专属模块，为处理用户网页端或设备端信息，用户/设备发送信息后，服务器接收信息，解析信息，处理信息，分发信息。

#### 2) 离线消息模块

此模块功能为将信息自动存入数据库中，为数据传输模块子模块。



图 4-14 离线消息界面图

3) 离线通信模块

此模块为服务器专用模块，功能是在用户离线时接收的信息定时通过邮箱发送给用  
户，用户如不设置邮箱，将无法发送到用户中。

4) 添加频道模块

此模块为频道管理中子模块，提交时检查格式是否正确，正确就存入数据库中。

频道名称 请输入频道名称

任务ID 请输入任务ID

点击即可换头像

注意：图片控制在1M以下!

立即提交 重置

图 4-15 添加频道界面图

### 5) 修改频道模块

此模块为频道管理中子模块，按钮前一部分为频道信息修改，修改后验证格式是否一致，是否属于该用户，全部一致即写入数据库中。后一部分为频道内用户权限修改，可将用户在频道内删除。



图 4-16 修改频道界面图

6) 删除频道模块

此模块功能为删除频道，通过匹配 id 删除频道，删除频道时检查频道是否属于删除用户者，不属于就删除失败。属于就删除成功，删除成功后删除与频道相关的全部内容。





图 4-17 删除频道界面图

7) 加入频道模块

此模块为用户加入某频道时使用，填写信息后发送信息给频道管理员申请处理该信息。



图 4-18 加入频道界面图

8) 邀请加入频道模块

此模块为管理员邀请用户加入模块，从频道管理进入，输入用户 id 即可邀请该用户进入频道。



图 4-19 邀请加入频道界面图

### 9) 通知消息模块

此模块功能为存放该用户全部的通知信息。



图 4-20 通知信息界面图

## 4.3 本章小结

本章从开发环境开始介绍，在对每一个功能进行实现后的结果进行简单的介绍。对于 Netty 底层框架的核心模块的代码，全部放在附录中。

## 5 系统测试

### 5.1 系统测试方法

本次测试方法使用黑盒测试来进行单元测试，分别对每一个功能进行单元测试。

### 5.2 单元模块测试

#### 5.2.1 用户模块测试

##### 1) 登录模块测试

此模块有四项测试，分别为：登录名或密码为空；登录名不存在；密码错误；全部正确。

表 5-1 登录模块测试用例表

测试 编号	测试内容	输入	预期输出	实际输出
1	登录名或密码为 空	qweqwe,	必填项不能 为空	必填项不能 为空
2	登录名不存在	qweqwe3,qweqwe	用户名不存 在	登录失败
3	密码错误	qweqwe,asdasd	密码错误	登录失败
4	全部正确	qweqwe, qweqwe	直接进入系 统	直接进入系 统

测试编号 1 实际结果图：



图 5-1 登录模块测试图 1

测试编号 2 实际结果图：



图 5-2 登录模块测试图 2

测试编号 3 实际结果图：



图 5-3 登录模块测试图 3

测试编号 4 实际结果图：



图 5-4 登录模块测试图 4

2) 注册模块测试

此模块有四项测试，分别为：登录名或密码为空；两次密码错误；全部正确；登录名已存在。

表 5-2 注册模块测试用例表

测试 编号	测试内容	输入	预期输出	实际输出
1	登录名或密码为空	qweqwe2,qweqwe,	必填项不能为空	必填项不能为空
2	两次密码错误	qweqwe2,asdasd , asdarda	两次密码输入不一致	两次密码输入不一致
3	全部正确	qweqwe2, qweqwe, qweqwe	进入登录界面	进入登录界面
4	登录名已存在	qweqwe, qweqwe, qweqwe	用户名已存在	

测试编号 1 实际结果图：

使用说明：用户名与密码不得低于6位数！

用户名

新密码

确认密码

图 5-5 注册模块测试图 1

测试编号 2 实际结果图：

使用说明：用户名与密码不得低于6位数！

用户名

新密码

确认密码

图 5-6 注册模块测试图 2

测试编号 3 实际结果图：

使用说明：用户名与密码不得低于6位数！

用户名

没账号？注册一发

密码

忘了密码？找回一波

图 5-7 注册模块测试图 3

测试编号 4 实际结果图：

使用说明：用户名与密码不得低于6位数！

用户名

qweqwe

新密码

..... 用户名已存在!

确认密码

.....

立即注册

重置

图 5-8 注册模块测试图 4

3) 找回密码模块测试

此模块有五项测试，分别为：用户名或邮箱不存在；验证码错误；两次密码不一致；全部正常；用户名与邮箱不匹配

表 5-3 找回密码模块测试用例表

测试编号	测试内容	输入	预期输出	实际输出
1	用户名或邮箱不存在	dfgdfg,994508794@qq.com,	用户信息不存在	用户信息不存在
2	验证码错误	qweqwe , root@mastermind.ink , 902974 , qweqwe, qweqwe	验证码错误	修改密码错误
3	两次密码不一致	qweqwe , root@mastermind.ink , 902975 , qweqweq, qweqwe	两次密码输入不一致	两次密码输入不一致
4	全部正常	qweqwe , root@mastermind.ink , 902975 , qweqwe, qweqwe	进入登录页面	进入登录页面
5	用户名与邮箱不匹配	qweqwe, 994508794@qq.com	用户名与邮箱不匹	用户名与邮箱不匹

测试编号 1 实际结果图：

使用说明：用户名与密码不得低于6位数！

用户名

dfgdfg

电子邮箱

请输入你的电子邮箱

验证码

请输入邮箱中的验证码

验证邮箱并发送验证码

用户信息不存在，你号没了。

新密码

.....

确认密码

请确认密码

立即修改

重置

图 5-9 找回密码模块测试图 1

测试编号 2 实际结果图：

使用说明：用户名与密码不得低于6位数！

用户名

qweqwe

电子邮箱

root@mastermind.ink

验证码

189448

验证邮箱并发送验证码

修改密码失败！

新密码

.....

确认密码

.....

立即修改

重置

图 5-10 找回密码模块测试图 2



测试编号 3 实际结果图：

使用说明：用户名与密码不得低于6位数！

用户名

qweqwe

电子邮箱

root@mastermind.ink

验证码

189448

 两次输入密码不一致，请重新输入！

新密码

.....

确认密码

.....|

立即修改

重置

图 5-11 找回密码模块测试图 3

测试编号 4 实际结果图：

使用说明：用户名与密码不得低于6位数！

用户名

qweqwe

没账号？注册一发

密码

.....

忘了密码？找回一波

立即登陆

重置

图 5-12 找回密码模块测试图 4

测试编号 5 实际结果图：

使用说明：用户名与密码不得低于6位数！

用户名

qweqwe

电子邮箱

994508794@qq.com

验证码

请输入邮箱中的验证码

新密码

.....

确认密码

请确认密码

立即修改

重置

你的账号邮箱与你当前邮箱错误，请检查输入账号验证邮箱并发送验证码和邮箱！

图 5-13 找回密码模块测试图 5

4) 修改密码模块测试

此模块有三项测试，分别为：原密码错误；两次密码不一致；全部正常

表 5-4 修改密码模块测试用例表

测 试 编号	测试内容	输入	预期输出	实际输出
1	原密码错误	qweqwes,qweqwe, qweqwe	原 密 码 错 误	修 改 密 码 失败

表 5-4（续）

测 试 编号	测试内容	输入	预期输出	实际输出
2	两次密码不一致	qweqwe , qweqwe, qweqweq	两次密码输 入不一致	两次密码输 入不一致
3	全部正常	qweqwe , qweqwe, qweqwe	修改成功	修改密码成 功

测试编号 1 实际结果图：

新密码必须两次输入一致才能提交



用户名 qweqwe

旧密码 请输入旧密码

新密码 请输入新密码

确认密码 请确认密码

立即修改 重置

图 5-14 修改密码模块测试图 1

测试编号 2 实际结果图：

新密码必须两次输入一致才能提交



用户名 qweqwe

旧密码 .....

新密码 😞 两次输入密码不一致，请重新输入!

确认密码 .....

立即修改 重置

图 5-15 修改密码模块测试图 2

测试编号 3 实际结果图：

新密码必须两次输入一致才能提交

用户名

qweqwe

旧密码

请输入旧密码

新密码

请输入新密码

确认密码

请确认密码

立即修改

重置

修改密码成功!

图 5-16 修改密码模块测试图 3

5) 个人信息模块测试

此模块有两项测试，分别为：部分信息未填写；全部正常

表 5-5 个人信息模块测试用例表

测试 编号	测试内容	预期输出	实际输出
1	部分信息未填写	必填项不能为 空	必填项不能为空
2	全部正常	修改成功	更新成功

测试编号 1 实际结果图：

用户名

qweqwe

个人昵称

Tim Joshua

性别

☐ 男 ☐ 女 ☒ 保密

手机号码

请输入手机号码

个人网址

https://MasterMind.ink/

邮箱

root@mastermind.ink

个性签名

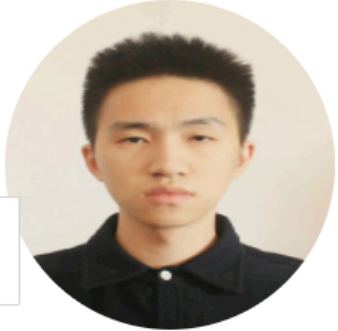
动个锤子

立即提交

重置

+ 点击即可换头像

注意：图片控制在1M以下!



😞 必填项不能为空

图 5-17 个人信息模块测试图 1

测试编号 2 实际结果图：

用户名

qweqwe

个人昵称

Tim Joshua

性别

☐ 男

☐ 女

☒ 保密

手机号码

13580640725

更新成功

个人网址

https://MasterMind.ink/

邮箱

root@mastermind.ink

个性签名

动个锤子

立即提交

重置

+ 点击即可换头像

注意：图片控制在1M以下!




图 5-18 个人信息模块测试图 2

5.2.2 管理模块测试

1) 封禁用户模块测试

此模块有两项测试，分别为：封禁用户；解封用户

表 5-6 封禁用户模块测试用例表

测试 编号	测试内容	预期输出	实际输出
1	封禁用户	必填项不能为 空	必填项不能为空
2	解封用户	修改成功	修改成功

测试编号 1 实际结果图：

用户id	用户名	用户权限	所属用户	最后退出时间	操作
5e8d78cbda460f0bc7e521cf	zxczxc	ROLE_BAN	zxczxc	2020-04-18 17:24	 解禁  禁用
5e8d78dada460f0bc7e521d0	qweqwe	ROLE_USER	qweqwe	2020-04-18 17:20	 解禁  禁用
5e8de4e95c79ad12994414eb	admins	ROLE_ADMIN	admins	2020-04-18 17:24	 解禁  禁用
5e8ee9e08f779005426f10a9	19970302	ROLE_USER	19970302	2020-04-16 00:53	 解禁  禁用
5e8f52eded2dae0a23e3ec3a	asdasd	ROLE_USER	asdasd	2020-04-16 00:53	 解禁  禁用
5e9425bfed2dae076e169e95	5e9425bfed2dae076e169e95	ROLE_DEVICE	qweqwe	2020-04-18 16:11	 解禁  禁用
5e9ab6158f77900490e17b87	qweqwe2	ROLE_USER	qweqwe2	2020-04-18 16:11	 解禁  禁用
5e9ab95a8f779005d018b6be	dfgdfg	ROLE_USER	dfgdfg		 解禁  禁用

图 5-19 封禁用户模块测试图 1

测试编号 2 实际结果图：

用户id	用户名	用户权限	所属用户	最后退出时间	操作
5e8d78cbda460f0bc7e521cf	zxczxc	ROLE_USER	zxczxc	2020-04-18 17:24	 解禁  禁用
5e8d78dada460f0bc7e521d0	qweqwe	ROLE_USER	qweqwe	2020-04-18 17:20	 解禁  禁用
5e8de4e95c79ad12994414eb	admins	ROLE_ADMIN	admins	2020-04-18 17:24	 解禁  禁用
5e8ee9e08f779005426f10a9	19970302	ROLE_USER	19970302	2020-04-16 00:53	 解禁  禁用
5e8f52eded2dae0a23e3ec3a	asdasd	ROLE_USER	asdasd	2020-04-16 00:53	 解禁  禁用
5e9425bfed2dae076e169e95	5e9425bfed2dae076e169e95	ROLE_DEVICE	qweqwe	2020-04-18 16:11	 解禁  禁用
5e9ab6158f77900490e17b87	qweqwe2	ROLE_USER	qweqwe2	2020-04-18 16:11	 解禁  禁用
5e9ab95a8f779005d018b6be	dfgdfg	ROLE_USER	dfgdfg		 解禁  禁用

图 5-20 封禁用户模块测试图 2

2) 系统公告模块测试

此模块有两项测试，分别为：部分信息未填写；全部正常

表 5-7 系统公告模块测试用例表

测试编号	测试内容	预期输出	实际输出
1	部分信息未填写	必填项不能为空	必填项不能为空
2	全部正常	发送成功	发送信息成功

测试编号 1 实际结果图：



聊天窗口

系统公告

公告标题

请输入公告标题

文章内容

这是一个测试公告

立即提交

重置

必填项不能为空

图 5-21 系统公告模块测试图 1

测试编号 2 实际结果图：

公告标题

这是一个测试公告标题

文章内容

这是一个测试公告

立即提交

重置

发送信息成功

图 5-22 系统公告模块测试图 2

5.2.3 任务管理模块测试

1) 添加/修改任务模块测试

此模块有两项测试，分别为：部分信息未填写；全部正常

表 5-8 添加/修改任务模块测试用例表

测 试 编号	测试内容	预期输出	实际输出
1	部分信息未填写	必填项不能为 空	必填项不能为空
2	全部正常	修改成功	修改成功

测试编号 1 实际结果图：

聊天窗口

任务管理

修改任务

任务名称

bash (mac/linux通用)

软件位置

5e94628eed2dae0cd1365aca

上传sh文件

软件显示脚本位置

5e9590d18f779007a97645bf

上传js文件

是否公开

true or false

任务介绍

这个是输入输

必填项不能为空

齐全。

功能介绍

没啥功能，就输入输出

立即提交

重置

图 5-23 添加/修改任务模块测试图 1

测试编号 2 实际结果图：

聊天窗口

任务管理

请输入关键字

查询

添加任务

批量删除

注意事项：任务路径先上传获取唯一号码，在填入路径中，JS脚本路径也是！

	任务ID	任务名称	是否公开	操作
<input type="checkbox"/>	5e9462f3ed2dae0cd1365ac	bash (mac/linux通用)	<input type="checkbox"/>	<div>编辑</div> <div>删除</div>

更新成功

图 5-24 添加/修改任务模块测试图 2

2) 删除任务模块测试

此模块有一项测试，分别为：全部正常

表 5-9 删除任务模块测试用例表

测 试 编号	测试内容	预期输出	实际输出
1	全部正常	删除成功	删除成功

测试编号 1 实际结果图：



图 5-25 删除任务模块测试图 1

3) 任务广场模块测试

此模块有一项测试，分别为：全部显示

表 5-10 任务广场模块测试用例表

测 试 编号	测试内容	预期输出	实际输出
1	全部正常	无	无

测试编号 1 实际结果图：



图 5-26 任务广场模块测试图 1

5.2.4 设备模块测试

1) 添加设备模块测试

此模块有一项测试，分别为：添加一个设备

表 5-11 添加设备模块测试用例表

测 试 编号	测试内容	预期输出	实际输出
1	添加一个设备	添加成功	添加成功

测试编号 1 实际结果图：



图 5-27 添加设备模块测试图 1

2) 修改设备模块测试

此模块有两项测试，分别为：部分信息未填写；全部正常

表 5-12 修改设备模块测试用例表

测试编号	测试内容	预期输出	实际输出
1	部分信息未填写	必填项不能为空	必填项不能为空
2	全部正常	修改成功	修改成功

测试编号 1 实际结果图：

修改设备

设备名称

设备密码

此密码为明文密码，只能修改，无法找回！

立即提交

重置

+ 点击即可换头像

注意：图片控制在1M以下！

必填项不能为空

图 5-28 修改设备模块测试图 1

测试编号 2 实际结果图：

添加设备

重刷页面

批量删除

注意事项

<input type="checkbox"/>	用户ID	所属人	当前操作权限	操作
<input type="checkbox"/>	5e9425bfed2dae076e169e95	qweqwe	ROLE_DEVICE	<div><div>编辑</div><div>删除</div></div>
<input type="checkbox"/>	5e9acfcc8f779006e79f11c5	qweqwe	ROLE_DEVICE	<div><div>编辑</div><div>删除</div></div>

更新成功

图 5-29 修改设备模块测试图 2

3) 删除设备模块测试

此模块有一项测试，分别为：全部正常

表 5-13 删除设备模块测试用例表

测试编号	测试内容	预期输出	实际输出
1	全部正常	删除成功	删除成功

测试编号 1 实际结果图：



图 5-30 删除设备模块测试图 1

4) 信息处理模块测试

此模块有一项测试，分别为：全部正常

表 5-14 信息处理模块测试用例表

测 试 编号	测试内容	预期输出	实际输出
1	全部正常	channelRead true	channelRead true

测试编号 1 实际结果图：



图 5-31 信息处理模块测试图 1

5) 登录验证模块测试

此模块有一项测试，分别为：全部正常；密码不正确；账号不存在；

表 5-15 登录验证模块测试用例表

测试 编号	测试内容	预期输出	实际输出
1	全部正常	登录成功	登录成功
2	密码不正确	登录失败	账号或密码不正确
3	账号不存在	登录失败	账号或密码不正确

测试编号 1 实际结果图：

```
Hello World!  
channelActive  
channelRead0 false  
WebSocket Client connected! response headers[sec-websocket-extensions]:{}DefaultHttpHe  
channelRead0 true  
TextWebSocketFrame :{"data":"a9e8d861","time":1587204802333,"type":"ChannelRegister"}  
channelRead0 true  
"登录成功! "  
TextWebSocketFrame :{"task":"","addressee":"","data":"\"登录成功! \",\"sender":"","time"
```

图 5-32 登录验证模块测试图 1

测试编号 2 实际结果图：

```
Hello World!  
channelActive  
channelRead0 false  
WebSocket Client connected! response headers[sec-websocket-extensions]:{}Di  
channelRead0 true  
TextWebSocketFrame :{"data":"7841cfce","time":1587206720138,"type":"Channe  
channelRead0 true  
"账号或密码不正确! "  
TextWebSocketFrame :{"task":"","addressee":"","data":"\"账号或密码不正确! \",
```

图 5-33 登录验证模块测试图 2

测试编号 3 实际结果图：



```
Hello World!
channelActive
channelRead0 false
WebSocket Client connected! response headers[sec-websocket-extensions]:{}DefaultHttpHea
channelRead0 true
TextWebSocketFrame :{"data":"a8f24aa7","time":1587206753497,"type":"ChannelRegister"}
channelRead0 true
"账号或密码不正确! "
TextWebSocketFrame :{"task":"","addressee":"","data":"\"账号或密码不正确! \",\"sender\":\"",
```

图 5-34 登录验证模块测试图 3

5.2.5 消息管理模块测试

1) 数据传输模块测试

此模块有一项测试，分别为：全部正常

表 5-16 数据传输模块测试用例表

测 试 编号	测试内容	预期输出	实际输出
1	全部正常	接收到的信息	接收到的信息

测试编号 1 实际结果图：

```
qweqwe:5e94469bed2dae0b144d88f0_writer
5e9425bfed2dae076e169e95:5e94469bed2dae0b144d88f0_writer
{"task":"text","data":"<p><span>大哥我是啊威呀</span></p>",</pre>
```

图 5-35 数据传输模块测试图 1

2) 离线消息模块测试

此模块有一项测试，分别为：全部正常

表 5-17 离线消息模块测试用例表

测 试 编号	测试内容	预期输出	实际输出
1	全部正常	离线信息	离线信息

测试编号 1 实际结果图：



图 5-36 离线消息模块测试图 1

### 3) 离线通信模块测试

此模块有一项测试，分别为：全部正常

表 5-18 离线通信模块测试用例表

测试 编号	测试内容	预期输出	实际输出
1	全部正常	无	无

测试编号 1 实际结果图：

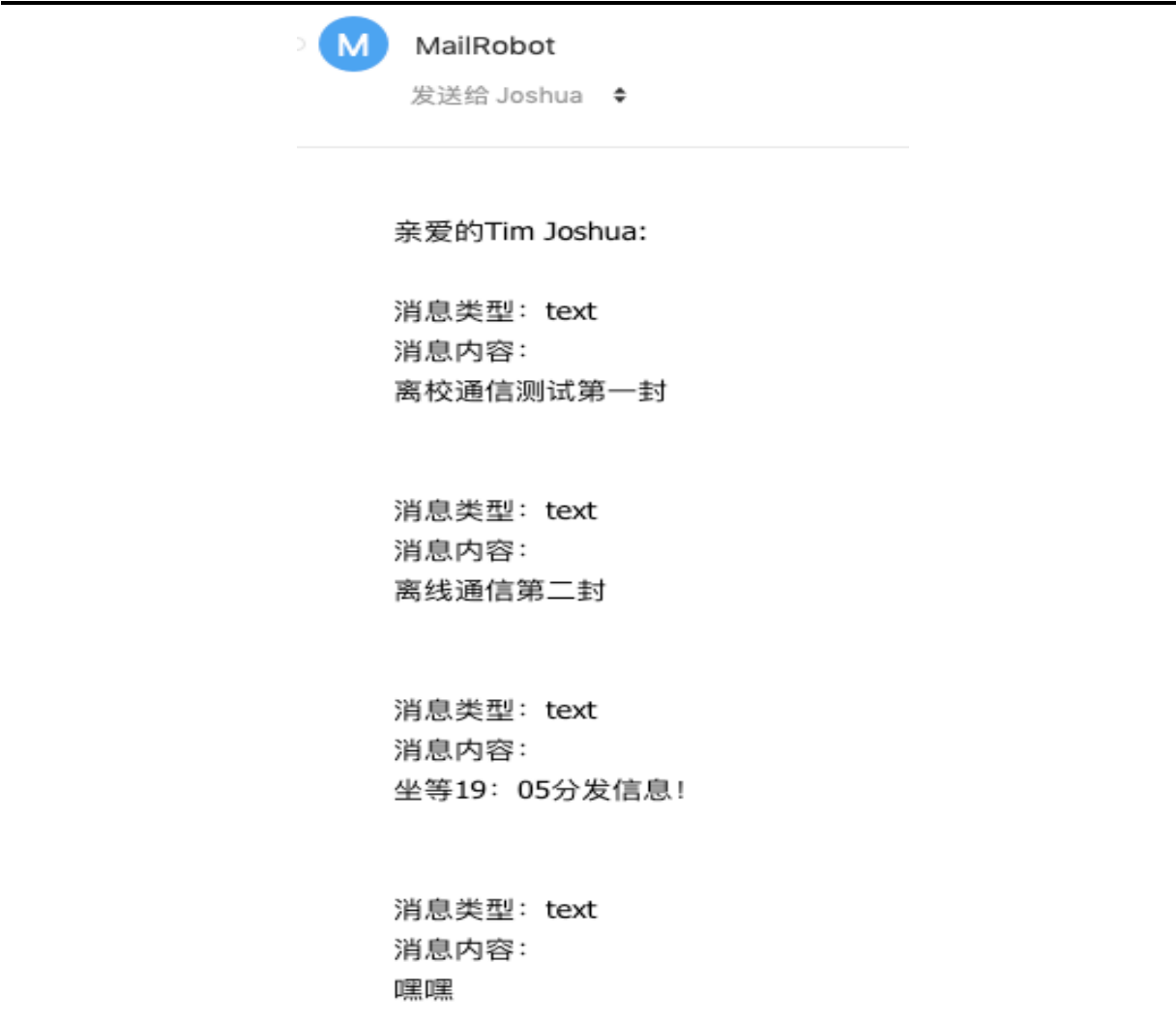


图 5-36 离线通信模块测试图 1

4) 添加频道模块测试

此模块有两项测试，分别为：部分信息未填写；全部正常

表 5-19 添加频道模块测试用例表

测 试 编号	测试内容	预期输出	实际输出
1	部分信息未填写	必填项不能为 空	必填项不能为空
2	全部正常	添加成功	添加成功

测试编号 1 实际结果图：

添加频道

频道名称

请输入频道名称

任务名称

请输入任务名称

+

 点击即可换头像

注意：图片控制在1M以下！

必填项不能为空

立即提交

重置

图 5-37 添加频道模块测试图 1

测试编号 2 实际结果图：

请输入关键字

查询

添加频道

加入频道

批量删除

重刷页面

注意事项：删除后无法恢复

	频道id	频道名称	任务名称	创立者	操作
<input type="checkbox"/>	5e9adf408f7790084715c78f	zhongqian	666	ZXCZXC	<div><div>邀请加入</div><div>编辑</div><div>删除</div></div>

添加成功

图 5-38 添加频道模块测试图 2

5) 修改频道模块测试

此模块有三项测试，分别为：修改为 writer；修改为 reader；踢出

表 5-20 修改频道模块测试用例表

测试 编号	测试内容	预期输出	实际输出
1	修改为 writer	修改成功	修改成功
2	修改为 reader	修改成功	修改成功
3	踢出	修改成功	修改成功

测试编号 1 实际结果图：



图 5-39 修改频道模块测试图 1

测试编号 2 实际结果图：



图 5-40 修改频道模块测试图 2

测试编号 3 实际结果图:



图 5-41 修改频道模块测试图 3

6) 删除频道模块测试

此模块有一项测试，分别为：全部正常

表 5-21 删除频道模块测试用例表

测 试 编号	测试内容	预期输出	实际输出
1	全部正常	删除成功	删除成功

测试编号 1 实际结果图：



图 5-42 删除频道模块测试图 1

7) 加入频道模块测试

此模块有四项测试，分别为：部分信息未填写；全部正常；频道名称或 id 错误；重复添加

表 5-22 加入频道模块测试用例表

测 试 编号	测试内容	预期输出	实际输出
1	部分信息未填写	必填项不能为空	必填项不能为空
2	全部正常	添加成功	添加成功

3	频道名称或 id 错误	频道名称或 id 错误	未找到该频道
4	重复添加	该用户已存在	该用户已存在

测试编号 1 实际结果图：

聊天窗口

频道管理

加入频道

频道id

5e94469bed2dae0b144d88f0

频道名称

请输入频道名称

立即提交

必填项不能为空

图 5-43 删除频道模块测试图 1

测试编号 2 实际结果图：

AliiNOne

asdasd

聊天窗口

频道管理

请输入关键字

查询

添加频道

加入频道

批量删除

重刷页面

注意事项：删除后无法恢复

加入成功

	频道id	频道名称	任务名称	创立者	操作
暂无数据					

图 5-44 加入频道模块测试图 2

测试编号 3 实际结果图：





图 5-45 加入频道模块测试图 3

测试编号 4 实际结果图：



图 5-46 加入频道模块测试图 4

8) 邀请加入频道模块测试

此模块有两项测试，分别为：部分信息未填写；全部正常

表 5-23 邀请加入频道模块测试用例表

测试编号	测试内容	预期输出	实际输出
1	部分信息未填写	必填项不能为空	必填项不能为空
2	全部正常	邀请成功	邀请成功

测试编号 1 实际结果图：

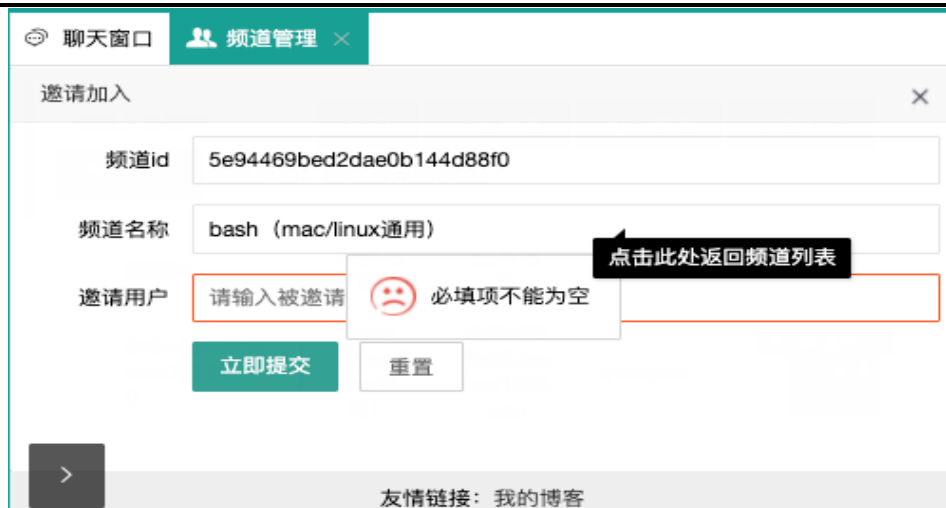


图 5-47 邀请加入频道模块测试图 1

测试编号 2 实际结果图:



图 5-48 邀请加入频道模块测试图 2

9) 通知消息模块测试

此模块有一项测试，分别为：全部显示

表 5-24 通知消息模块测试用例表

测试编号	测试内容	预期输出	实际输出
------	------	------	------

1	全部正常	无	无
---	------	---	---

测试编号 1 实际结果图：



图 5-49 通知消息模块测试图 1

5.3 本章小结

本章通过使用黑盒测试来进行单元测试，在对每一个功能测试过程中，发现了许多完全没有遇到过的 BUG，对单元测试后，各个模块都验证完全正常使用。

## 6 结论

通过系统的实现和测试过程中，我们可以得出结论，基于任务驱动与多用户权限的 C/S 框架通信系统需要有一些基本的用户、信息、任务、设备、频道管理等功能，达到基本无差错管理，其中第一个核心部分为使用 Netty 框架实现的 Websocket 服务器与客户端无差错运行，并且完全嵌入 SpringBoot 框架当中；第二个核心部分为与用户的交互部分，该部分使用网页前后端分离，采用 json 实现通信，实现了全部用户功能；第三个核心部分为设备端运行部分，该部分使用 java 最基础的部分实现了异步处理，使用线程池管理线程部分，下载任务文件使用可重用锁保证单线程下载。

该系统创新点有以下几个部分：

- 1、前端采用网页前后端分离，未来可以实现多个前端，比如微信小程序，电脑客户端，手机客户端等等。满足大多数人的需要。
- 2、后端使用了 SpringBoot 满足用户短连接通信，后期单机转集群可以使用 SpringCloud。
- 3、后端的 Websocket 协议使用了 Netty 的 NIO 实现，进一步提高单机的并发量。
- 4、后端文件存储部分使用的是 MinIO，后期可以从单机转集群非常方便。
- 5、后端数据库存储部分使用的是 MongoDB，后期可以从单机转为集群非常方便。
- 6、该系统可以由用户自定义 JS 文件，当作遥控器，遥控器减少操作设备难度。

## 参考文献

- [1] 展望2020年物联网 IoT 发展：打造智能且安全的物联网 IoT 平台[J].家电科技,2019(06):10-11.
- [2] 郭文静,刘迪,丁学英.面向电力行业的物联网平台设计及应用[J].供电,2019,36(06):46-54.
- [3] 卢永华.基于热插拔的物联网平台[J].数字技术与应用,2019,37(06):106+108.
- [4] 封琪,王贵鑫.Java 技术框架的发展及其应用[J].科学技术创新,2018(09):67-69.
- [5] 熊永平.基于 SpringBoot 框架应用开发技术的分析与研究[J].电脑知识与技术,2019,15(36):76-77.
- [6] 颜治平.基于 SpringBoot 和 Vue 框架的教代会提案系统的设计与实现[J].科技创新与应用,2020(03):91-93+95.
- [7] 吴志伟,钟立民,黄永亮,李国华.基于微服务架构的铁路货运基础字典统一运维应用系统的设计与实现[J].铁路计算机应用,2020,29(03):19-23.
- [8] 任明飞,李学军,崔蒙蒙,杨双龙,孙小奇.基于 MongoDB 的非关系型数据库的设计与开发[J].电脑知识与技术,2019,15(34):1-2.
- [9] 柴莹莹,安博文,周凡.基于 MongoDB 的海上移动执法文档管理与查询系统[J].现代电子技术,2020,43(06):86-89.
- [10] 卢奇荣.基于 Vue2+Koa2+MongoDB 平台的网站技术分析[J].广播电视信息,2020(02):103-105.
- [11] 肖宇.关系型与非关系型数据库融合的数据库课程建设[J].福建电脑,2019,35(12):111-112.
- [12] 叶连杰.Docker 虚拟化技术在嵌入式平台的应用实践研究[J].制造业自动化,2019,41(12):107-109+144.
- [13] 冯一凡,朱文龙,杨双双.基于 Docker 的分布式网络安全攻防平台设计与实现[J].计算机产品与流通,2020(03):47.
- [14] 张聪,户利利,徐明,秦斌.基于 docker 的高校云平台运维体系探讨[J].信息与电脑(理论版),2020,32(01):3-6.
- [15] 任汉秋.基于 Netty 多租户的企业即时通讯系统的设计与实现[D].哈尔滨工业大

学,2019.

- [16] 张娜,史佳炳,吴彪,包晓安,文艺霏.基于 Netty 和 Kafka 的 IOT 终端服务系统设计方案 [J/OL]. 浙江理工大学学报 (自然科学版):1-6[2020-03--18].<http://kns.cnki.net/kcms/detail/33.1338.TS.20200218.1439.027.html>.
- [17] 庄榕. 基于 Netty 的分布式数据传输系统的设计与实现 [J]. 中国新通信,2019,21(17):144-145.
- [18] 王杰,高永平.一种基于 netty 通信的增强现实方案的设计与实现[J].电脑知识与技术,2019,15(36):241-243.
- [19] 叶为正,林声肯,黄立轩,许志明,李晶.即时通讯系统的设计与实现[J].计算机技术与发展,2020,30(02):216-220.
- [20] 李旭光. 翻转课堂的设计与实现[D].山东大学,2019.
- [21] 曹灿,刘志刚.基于 SSH 和 Layui 的工程科学前沿与实践系统[J].工业控制计算机,2019,32(02):91-92+96
- [22] 李翔.ERP 系统中基于 websocket 协议的实时通讯机制的设计与实现[J].数字通信世界,2020(02):104-105.

## 附录

### 附录 A Netty 底层服务器端实现代码

ChatServer 类：服务器启动类

```
@Service("chatServer")
public class ChatServer {
    public ChannelFuture start(InetSocketAddress address) {
        //引导服务器
        ServerBootstrap bootstrap = new ServerBootstrap();
        bootstrap.group(group)
            .handler(new LoggingHandler(LogLevel.INFO))
            .channel(NioServerSocketChannel.class)
            .childHandler(createInitializer());
        ChannelFuture future = bootstrap.bind(address);
        future.syncUninterruptibly();
        channel = future.channel();
        return future;
    }

    //处理服务器关闭，并释放所有的资源
    public void destroy() {
        if (channel != null) {
            channel.close();
        }
        channelKeeper.close();
        group.shutdownGracefully();
    }
}
```

### ChatServerInitializer 类：服务器通道初始化

```
@Service("chatServerInitializer")
public class ChatServerInitializer extends ChannelInitializer<Channel> {
    //将所有需要的 ChannelHandler 添加到 ChannelPipeline 中
    protected void initChannel(Channel ch) throws Exception {
        ChannelPipeline pipeline = ch.pipeline();
        pipeline.addLast(new HttpServerCodec());
        pipeline.addLast(new ChunkedWriteHandler());
        pipeline.addLast(new HttpObjectAggregator(64 * 1024));
        pipeline.addLast(new WebSocketServerProtocolHandler("/ws"));
        pipeline.addLast(new TextWebSocketFrameHandler(chatServerInitializer));
    }
```

### TextWebSocketFrameHandler 类：Websocket 协议处理类

```
public class TextWebSocketFrameHandler extends
SimpleChannelInboundHandler<TextWebSocketFrame> {

    private ChatServerInitializer chatServerInitializer;

    /**
     * @param chatServerInitializer
     */
    public TextWebSocketFrameHandler(ChatServerInitializer chatServerInitializer) {
        // TODO Auto-generated constructor stub
        this.chatServerInitializer = chatServerInitializer;
    }

    /**
     * 添加到用户在线池中，并返回频道 ID 供注册使用。
     */
}
```



```

        */

        @Override
        public void userEventTriggered(ChannelHandlerContext ctx, Object evt) throws
Exception {
            if (evt ==
WebSocketServerProtocolHandler.ServerHandshakeStateEvent.HANDSHAKE_COMPLETE
) {

                chatServerInitializer.getChannelKeeper().firstChannel(ctx, evt);
            } else {
                super.userEventTriggered(ctx, evt);
            }
        }

        /**
         * 收到信息后干嘛
         */

        @Override
        public void channelRead0(ChannelHandlerContext ctx, TextWebSocketFrame msg)
throws Exception {

            //(3) 增加消息的引用计数，并将它写到 ChannelGroup 中所有已经连接的客
            户端

            chatServerInitializer.getChannelKeeper().handleInformation(ctx, msg);
        }

        /**
         * 断开连接
         */

        @Override
    
```

```

public void channelInactive(ChannelHandlerContext ctx) throws Exception {
    // TODO Auto-generated method stub
    super.channelInactive(ctx);
    chatServerInitializer.getChannelKeeper().removeChannel(ctx.channel());
}

public void channelReadComplete(ChannelHandlerContext ctx) throws Exception {
    // TODO Auto-generated method stub
    super.channelReadComplete(ctx);
    ctx.flush();
}

/**
 * 信息报错，反馈给管理员日志去
 */
@Override
public void exceptionCaught(ChannelHandlerContext ctx, Throwable cause) throws
Exception {
    // TODO Auto-generated method stub
    super.exceptionCaught(ctx, cause);
    chatServerInitializer.getChannelKeeper().channelLog(ctx, cause);
}
}

```

## 附录 B Netty 底层客户端实现代码

WebsocketClient 类：客户端启动类

```

public class WebsocketClient {
    private Channel channel = null;

```

```

private EventLoopGroup group = new NioEventLoopGroup();

public ChannelFuture start(String url, final DeviceLogin deviceLogin, final String path,
final ExecutorService executorService) {
    try {
        Bootstrap boot = new Bootstrap();
        boot.option(ChannelOption.SO_KEEPALIVE, true)
            .option(ChannelOption.TCP_NODELAY, true)
            .group(group)
            .handler(new LoggingHandler(LogLevel.INFO))
            .channel(NioSocketChannel.class)
            .handler(new ChannelInitializer<SocketChannel>() {
                protected void initChannel(SocketChannel socketChannel) throws
Exception {
                    ChannelPipeline p = socketChannel.pipeline();
                    p.addLast(new ChannelHandler[] { new HttpClientCodec(),
                        new HttpObjectAggregator(64 * 1024 *
1024) });

                    p.addLast("hookedHandler", new
WebSocketClientHandler(deviceLogin, path, executorService));
                }
            });
        URI websocketURI = new URI(url);
        HttpHeaders httpHeaders = new DefaultHttpHeaders();
        // 进行握手
        WebSocketClientHandshaker handshaker =
WebSocketClientHandshakerFactory.newHandshaker(websocketURI,
        WebSocketVersion.V13, (String) null, true, httpHeaders, 64 * 1024
* 1024);
    }
}

```

```
        ChannelFuture future = boot.connect(websocketURI.getHost(),
websocketURI.getPort()).sync();

        channel = future.channel();

        WebSocketClientHandler handler = (WebSocketClientHandler)
channel.pipeline().get("hookedHandler");

        handler.setHandshaker(handshaker);

        handshaker.handshake(channel);

        // 阻塞等待是否握手成功

        handler.handshakeFuture().sync();

        return future;
    } catch (Exception e) {

        // TODO: handle exception

    }

    return null;
}

public void destroy() {

    // TODO Auto-generated method stub

    if (channel != null) {

        channel.close();

    }

    group.shutdownGracefully();

}
}
```

WebSocketClientHandler 类：客户端 Websocket 处理类

```
public class WebSocketClientHandler extends SimpleChannelInboundHandler<Object> {

    private WebSocketClientHandshaker handshaker;
```

```

private ChannelPromise handshakeFuture;

private DeviceLogin deviceLogin;

private String path;

private ExecutorService executorService;


public WebSocketClientHandler(DeviceLogin deviceLogin, String path,
ExecutorService executorService) {

    super();

    this.deviceLogin = deviceLogin;

    this.path = path;

    this.executorService = executorService;

}


public void handlerAdded(ChannelHandlerContext ctx) {

    this.handshakeFuture = ctx.newPromise();

}


protected void channelRead0(ChannelHandlerContext ctx, Object msg) throws
Exception {

    System.out.println("channelRead0  " + this.handshaker.isHandshakeComplete());

    final Channel ch = ctx.channel();

    FullHttpResponse response;

    if (!this.handshaker.isHandshakeComplete()) {

        try {

            response = (FullHttpResponse) msg;

            // 握手协议返回， 设置结束握手

            this.handshaker.finishHandshake(ch, response);

            // 设置成功

```

```

        this.handshakeFuture.setSuccess();

        System.out.println("WebSocket Client connected! response headers[sec-
websocket-extensions]:{"}

            + response.headers());
    } catch (WebSocketHandshakeException var7) {
        FullHttpResponse res = (FullHttpResponse) msg;
        String errorMsg = String.format("WebSocket Client failed to
connect,status:%s,reason:%s", res.status(),
            res.content().toString(CharsetUtil.UTF_8));
        this.handshakeFuture.setFailure(new Exception(errorMsg));
    }
} else if (msg instanceof FullHttpResponse) {
    response = (FullHttpResponse) msg;
    throw new IllegalStateException("Unexpected FullHttpResponse (getStatus="
+ response.status() + ", content="
        + response.content().toString(CharsetUtil.UTF_8) + ')');
} else {
    WebSocketFrame frame = (WebSocketFrame) msg;
    if (frame instanceof TextWebSocketFrame) {
        TextWebSocketFrame textFrame = (TextWebSocketFrame) frame;
        final JSONObject jsonObject = JSON.parseObject(textFrame.text());
        if ("UseTask".equals(jsonObject.getString("type"))) {
            executorService.execute(new Thread(new Runnable() {
                public void run() {
                    MessageHandler.input(jsonObject, ch, path, deviceLogin);
                }
            }));
        } else if ("DeviceRegister".equals(jsonObject.getString("type"))) {

```

```
        System.out.println(jsonObject.getString("data"));
    }
    System.out.println("TextWebSocketFrame :" + textFrame.text());
} else if (frame instanceof BinaryWebSocketFrame) {
    BinaryWebSocketFrame binFrame = (BinaryWebSocketFrame) frame;
    System.out.println("BinaryWebSocketFrame");
    MessageHandler.download(frame.content());
} else if (frame instanceof CloseWebSocketFrame) {
    System.out.println("receive close frame");
    ch.close();
}
}
}
public void channelReadComplete(ChannelHandlerContext ctx) throws Exception {
    super.channelReadComplete(ctx);
    deviceLogin.register(ctx);
}
}
```

## 致 谢

毕业设计是自己结合四年学习的专业知识进行实现，在这几个月的时间里，自己的开发思维都得到进一步的提升。在充满青春奋斗的气息里，实现系统过程中遇到很多困难，但是自己还是在寻找方法去不断的去尝试克服。感谢学校的图书资源，让自己学习到很多之前没有学到的内容，更好的了解 Netty 与 NoSQL 框架与思想。强烈的感谢指导老师邓维老师的鼓励和耐心的指导，让自己在初始开发系统没有头绪时，给自己一些关键性的引导，让自己更有信心的实现自己的系统；感谢朱昌洪老师给予物联网的指导与建议；感谢 GitHub 上 Siwash 提供的 Netty 的 Websocket 实现方案，让我从代码中学习到了不一样的编程视角；感谢 GitHub 上 jhalterman 提供的 expiringmap，能让我不需要使用 Redis 时开发出限时的 HashMap；感谢《Netty 实战》和《Netty、Redis、Zookeeper 高并发实战》的作者们，让我从这本书中了解 Netty 理论与知识；感谢《String Boot 实战》的作者汪云飞，让我在这本书中获得了灵感将 Netty 与 SpringBoot 融合而不发生冲突；感谢 LayUI 作者贤心，他的前端框架能快速开发一个友好的前端页面，使我不再纠结在前端界面上；感谢 MinIO 作者们，他们的文档指导使我快速开发出属于这个毕设的驱动。

同时，我也要感谢本论文所引用的文献作者和 CSDN 论坛的博主，是他们分享的经验和方法，让自己不仅更容易的解决代码问题，也可以便利的学习一些开发系统框架，更明确的知道自己需要学习什么和如何学习，更好做一名合格的程序员。