湖南大學
HUNAN UNIVERSITY

课程设计报告

课 程 名 称：计算机网络

研 究 选 题：基于 POP3 和 SMTP 的邮件服务器端和移动客户端设计

专 业 班 级：软件 2203

姓　　名：孙保庆、李景越、刘欢、丁海桐、白旭

学　　号：202226010306

指 导 老 师：刘琴

完 成 时 间：2024 年 11 月 19 日

信息科学与工程学院

目录

一、引言 3
二、项目概述 3
1. 项目简介 3
2. 项目任务 4
2.1 项目名称 4
2.2 项目内容 4
3. 技术方案 4
3.1 后台管理模块 4
3.2 服务器模块 5
3.3 客户端模块 6
三、概要设计 6
1. 协议简介 6
1.1 SMTP 协议 6
1.2 POP3 协议 6
2. 系统架构 7
3.部分功能流程图 7
3.1 POP3 用户认证流程 7
3.2 POP3 传送流程 8
3.3 POP3 演示 8
3.4 SMTP 演示 9
四、后台管理模块 9
1. 登录功能 9
2. 个人中心 10
3. 用户管理 11
4. 邮件管理 15
5. 日志管理 16
6. 系统管理 17
五、服务器模块 18
1. SMTP 服务模块 18
1.1 SMTP 状态变量 18
1.2 实现算法 18
1.3 SMTP 指令的具体处理函数 19
2.POP3 服务模块 20
2.1 POP3 状态变量 20
2.2 算法实现 21
2.3 POP3 指令的具体处理函数 21
3. IP 过滤 22
3.1 获取 IP 地址 22
3.2 拦截 IP 22
4. 接口文档 23
六、客户端模块 24
1. 登录功能 24

2. 注册功能.....25
3. 发送邮件功能.....27
4. 收件箱功能.....28
5. 发件箱功能.....29
七、 数据库模型.....30
1. 数据库表.....30
1.1 user.....30
1.2 email.....30
1.3 contact.....31
1.4 log.....31
1.5 filter.....31
1.6 server-msg.....32
八、项目总结.....32
1. 项目感想.....32

一、引言

电子邮件作为人们沟通交流的主要工具，在网络中有着广泛的应用。邮件系统的架构可分为邮件传输代理 MTA、邮件投递代理 MDA 和邮件用户代理MUA 。邮件用户代理是一个发信和收信的程序，负责将电子邮件发送到 SMTP 服务器或者从邮件服务器取回收到的邮件。常用的邮件用户代理有微软的OUTLOOK、腾讯的 FOXMAIL 等，其可以从遵循 POP3 协议的邮件服务器中收取邮件。

本实验报告基于 SMTP 和 POP3 的邮件服务端和移动客户端的设计需求，设计方法和环境介绍。

二、项目概述

1. 项目简介

项目是基于 POP3 和 SMTP 的邮件服务端和移动客户端（app）的设计，设计一个邮件服务器和一个移动端（app）的邮件客户端，服务器端除了提供最基本的收发邮件功能之外，还应具有注册新用户、管理用户、群发邮件以及修改服务器相关参数、修改管理员密码、邮件和 IP 地址过滤等功能。客户端分为普通用户端和管理员端。普通用户端可实现基本的注册、收发邮件，修改个人资料等功能；管理员端主要实现群发邮件功能，除此之外，它还可以实现浏览用户信息以及删除用户等操作。

2．项目任务

2.1 项目名称

基于 SMTP 和 POP3 协议的邮件服务端和移动客户端设计。

2.2 项目内容

(1)基于 SMTP 的邮件发送服务器设计与实现；
(2)基于 POP3 的邮件接收服务器设计与实现；
(3)移动Android端app设计与实现。

3．技术方案

3.1 后台管理模块

(1)Vue

Vue 是一套用于构建用户界面的渐进式框架。与其它大型框架不同的是， Vue被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，不仅易于上手， 还便于与第三方库或既有项目整合。另一方面，当与现代化的工具链以及各种支持类库结合使用时，Vue 也完全能够为复杂的单页应用提供驱动。

Vue.js 是一套构建用户界面的渐进式框架。与其他重量级框架不同的是， Vue 采用自底向上增量开发的设计。Vue 的核心库只关注视图层，并且非常容易学习， 非常容易与其它库或已有项目整合。另一方面，Vue 完全有能力驱动采用单文件组件和 Vue 生态系统支持的库开发的复杂单页应用。

Vue.js 的目标是，通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。Vue.js 自身不是一个全能框架——它只聚焦于视图层。因此它非常容易学习，非常容易与其它库或已有项目整合。另一方面，在与相关工具和支持库一起使用 时，Vue.js 也能地驱动复杂的单页应用。组件系统是Vue 的另一个重要概念，因为它是一种抽象，允许我们使用小型、独立和通常可复用的组件构建大型应用。

仔细想想，几乎任意类型的应用界面都可以抽象为一个组件树：

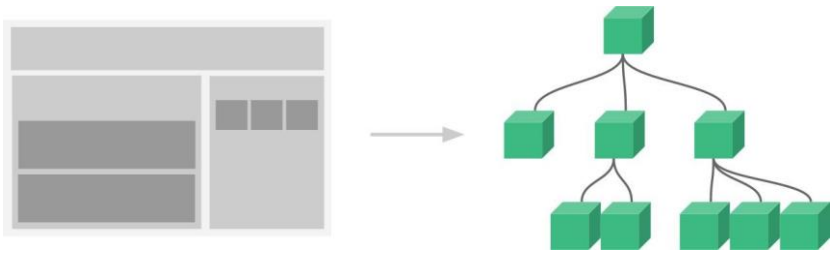


图 3-1-1

(2)View-UI

View UI，即原先的 iView，是一套基于 Vue.js 的开源 UI 组件库，主要服务于 PC 界面的中后台产品。

View UI 官网提供了详细的文档，上手非常容易，如果有 Element UI 类似的开发经验，几乎不需要适应，开箱即用。

View UI 是国内 Vue 生态圈内的佼佼者，由公司团队进行维护，周边有 Admin Pro、Admin UI Pro 等付费授权的产品都是基于 View UI ，这在很大程度上保证了持续高质量的维护。

特性：

丰富的组件和功能，满足绝大部分网站场景

提供开箱即用的 Admin 系统 和 高阶组件库，极大程度节省开发成本

提供专业、优质的一对一技术支持

友好的 API ，自由灵活地使用空间

细致、漂亮的 UI

事无巨细的文档

可自定义主题

3.2 服务器模块

(1)Java 编程语言

Java 是一门面向对象编程语言，不仅吸收了 C++语言的各种优点，还摒弃了C++里难以理解的多继承、指针等概念，因此 Java 语言具有功能强大和简单易用两个特征。Java 语言作为静态面向对象编程语言的代表，极好地实现了面向对象理论，允许程序员以优雅的思维方式进行复杂的编程。

Java 具有简单性、面向对象、分布式、健壮性、安全性、平台独立与可移植性、多线程、动态性等特点。Java 可以编写桌面应用程序、Web 应用程序、分布式系统和嵌入式系统应用程序等。

(2)SpringBoot

Spring Boot 是由 Pivotal 团队提供的全新框架，其设计目的是用来简化新 Spring 应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。

Spring 框架是 Java 平台上的一种开源应用框架，提供具有控制反转特性的容器。尽管 Spring 框架自身对编程模型没有限制，但其在 Java 应用中的频繁使用让它备受青睐，以至于后来让它作为 EJB 模型的补充，甚至是替补。Spring框架为开发提供了一系列的解决方案，比如利用控制反转的核心特性，并通过依赖注入实现控制反转来实现管理对象生命周期容器化，利用面向切面编程进行声明式的事务管理，整合多种持久化技术管理数据访问，提供大量优秀的Web框架方便开发等等。Spring 框架具有控制反转（IOC）特性，IOC 旨在方便项目维护和测试，它提供了一种通过 Java 的反射机制对 Java 对象进行统一的配置和管理的方法。Spring 框架利用容器管理对象的生命周期,容器可以通过扫描 XML 文件或类上特定 Java 注解来配置对象，开发者可以通过依赖查找或依赖注入来获得对象。Spring 框架具有面向切面编程（ AOP ） 框架， SpringAOP 框架基于代理模式，同时运行时可配置；AOP框架主要针对模块之间的交叉关注点进行模块化。Spring 框架的 AOP 框架仅提供基本的 AOP 特性,虽无法与 AspectJ 框架相比，但通过与 AspectJ 的集成，也可以满足基本需求。

Spring MVC 框架提供了模型-视图-控制的体系结构和可以用来开发灵活、松散耦合的 web 应用程序的组件。MVC 模式导致了应用程序的不同方面(输入逻辑、业务逻辑和 UI 逻辑)的分离，同时提供了在这些元素之间的松散耦合。

(3)MySQL

MySQL 是一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，目前属于 Oracle 公

司。Mysql 是最流行的关系型数据库管理系统，在 WEB 应用方面 MySQL 是最好的 RDBMS 应用软件之一。MySQL 是一种关联数据库管理系统，关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。MySQL 所使用的 SQL 语言是用于访问数据库的最常用标准化语言。搭配 PHP 和 Apache 可组成良好的开发环境。

3.3 客户端模块

(1)app应用

app是安装在智能手机中应用软件，一般用于苹果和安卓等其他手机应用，目前的app是指多功能手机第三方应用程序。app优点如下：

用户群庞大：传统的互联网主要依靠电脑，因为使用不方便，但是移动互联网依靠智能手机，用户群数量庞大。

及时性：对比电脑，使用手机上网处理事情非常方便，随时随地都可以进行。真是这种方便性，让移动互联网取得快速发展。

功能强大：网页以为自身的限制，功能比较少，对网络的要求比较到，加载速度慢，用户使用不方便。但是手机App功能非常强大、齐全，对网络的依赖度低，加载速度快，用户体验流畅。

入口浅：网页需要大家输入网址或者依靠搜索，使用起来非常不方便，特别是在移动互联网时代。但是手机App抢占的是用户的手机界面，大家打开手机就可以点开直接使用。

三、概要设计

1．协议简介

1.1 SMTP 协议

简单邮件传输协议（英语：Simple Mail Transfer Protocol，缩写：SMTP）是一个在互联网上传输电子邮件的标准。

SMTP 是一个相对简单的基于文本的协议。在其之上指定了一条消息的一个或多个接收者（在大多数情况下被确认是存在的），然后消息文本会被传输。可以很简单地通过telnet 程序来测试一个SMTP 服务器。SMTP 使用TCP 端口 25。要为一个给定的域名决定一个 SMTP 服务器，需要使用 DNS 的 MX 记录。

由于这个协议开始是基于纯 ASCII 文本的，它在二进制文件上处理得并不好。例如 MIME 的标准被开发来编码二进制文件以使其透过 SMTP 来传输。今天，大多数 SMTP 服务器都支持 8 位 MIME 扩展，它使二进制文件的传输变得几乎和纯文本一样简单。

1.2 POP3 协议

邮局协议（英语：Post Office Protocol，缩写：POP）是 TCP/IP 协议族中的一员，由 RFC 1939 定义。此协议主要用于支持使用客户端远程管理在服务器上的电子邮件。最新版本为 POP3，全名“Post Office Protocol - Version 3”，而提供了SSL 加密的 POP3 协议被称为 POP3S。

POP 支持离线邮件处理。其具体过程是：邮件发送到服务器上，电子邮件客户端调用邮件客户机程序以连接服务器，并下载所有未阅读的电子邮件。这种离线访问模式是一种存储转发服务，将邮件从邮件服务器端送到个人终端机器上，一般是 PC 机或 Mac。一旦邮件下载到 PC 机或 Mac 上，邮件服务器上的邮件将会被删除。但目前的 POP3 邮件服务器大都可以“只下载邮件，服务器端并不删除”，也就是改进的 POP3 协议。

2．系统架构

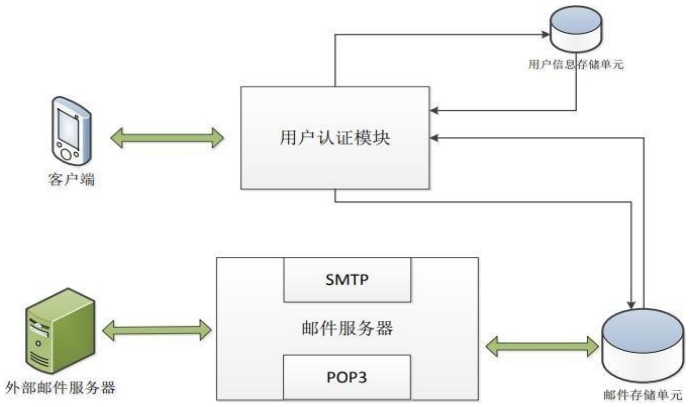


图 3-1-2

3.部分功能流程图

3.1 POP3 用户认证流程

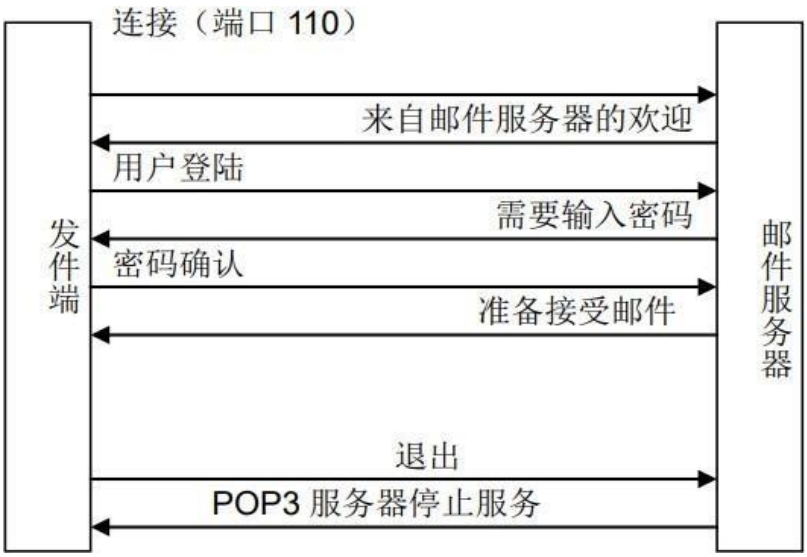


图 3-3-2

3.2 POP3 传送流程

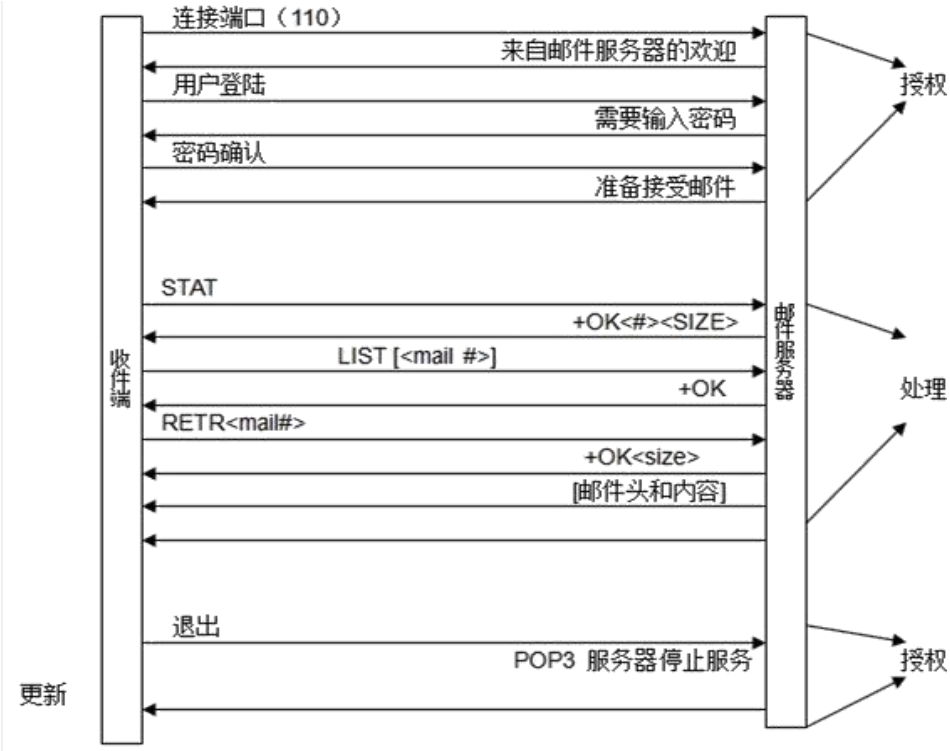


图 3-3-3

3.3 POP3 演示

图 3-3-4

3.4 SMTP 演示

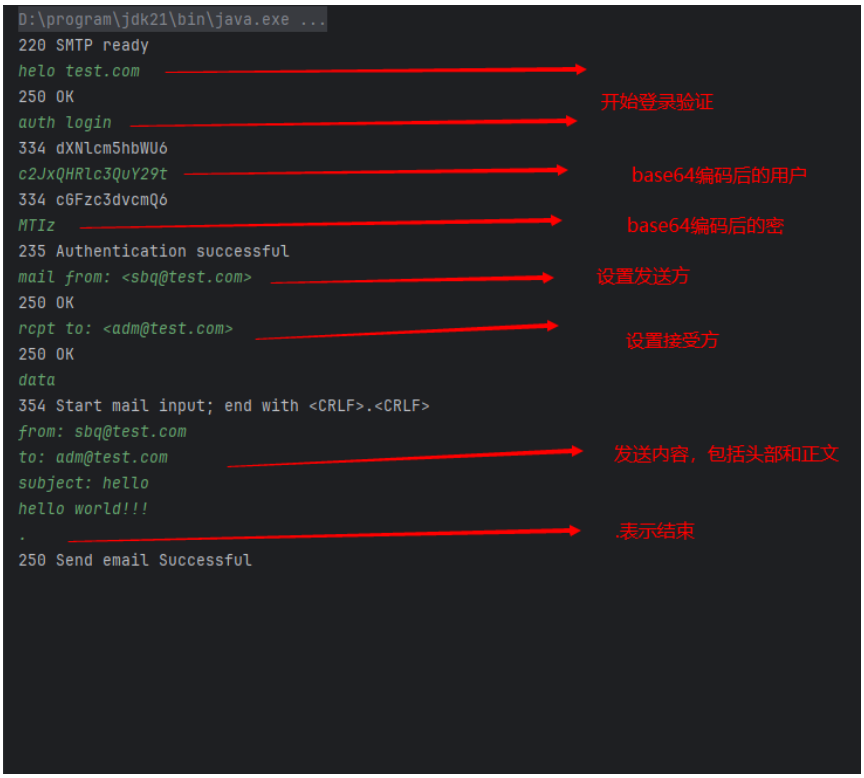


图 3-3-5

四、后台管理模块

1. 登录功能

当输入网址116.62.139.92时，就会自动跳转到登录界面。

图 4-1-1

具体功能如下：

1.1输入权限为“管理员”的账号和密码，点击“Signin”后台逻辑会根据用户所输入的信息判断是否可以成功登入系统。

1.2可以点击密码框后面的小图标选择是否显示明文密码。

1.3可以打开“记住密码”选项，下次用同一浏览器登录时就不需要再填写账号和密码。

1.4点击 Reset 可以重置输入框。

2．个人中心

登录成功后来到管理界面，在侧边导航栏上，点击下图所示的位置即可进入个人中心。

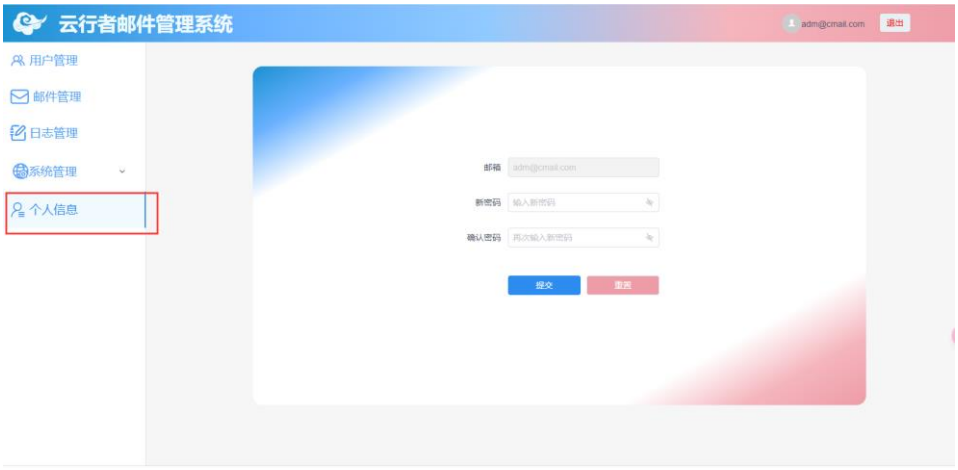


图 4-2-1

具体功能如下：

2.1 在“新密码”输入框输入新密码，并且要二次确认新密码，如果两次输入的密码不一致会有提示。

图 4-2-2-1

2.2 通过二次确认的验证后，点击“提交”，弹出“修改成功”即表示已修改成新密码。

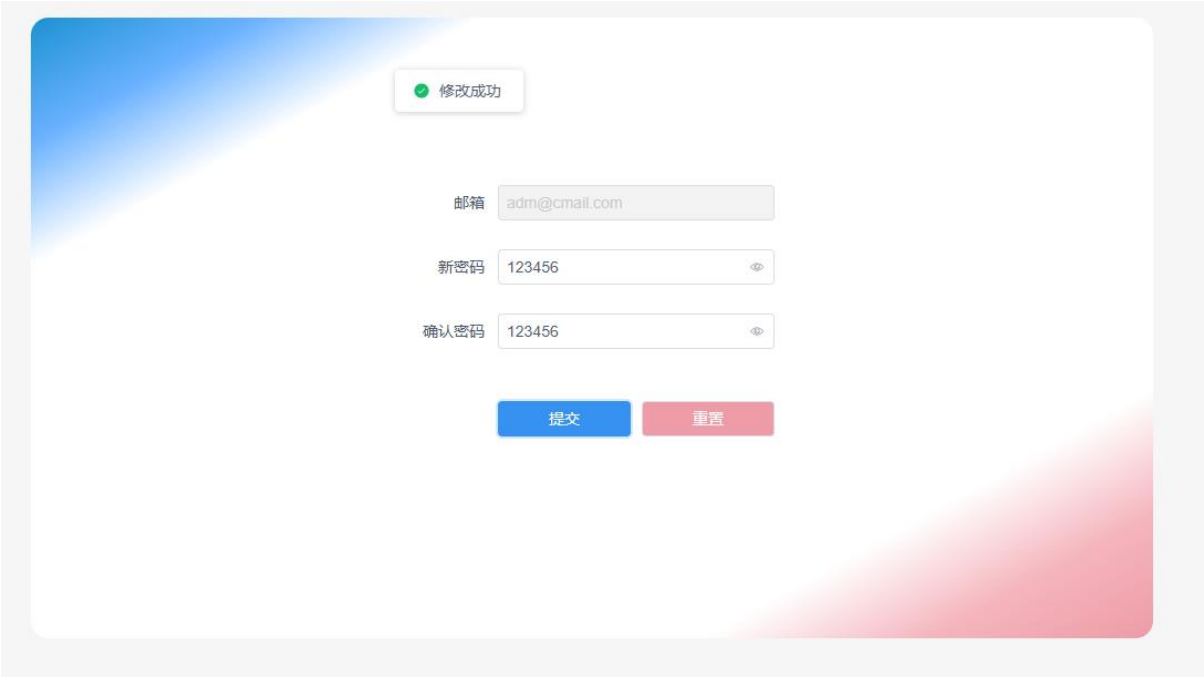


图 4-2-2-2

2.3点击“重置”按钮可以重置输入框，但是邮箱是不能被修改的。

2.4点击“退出登录”按钮可以登出系统，回到登录界面。

3．用户管理

登录成功后，进入的第一个界面就是用户管理界面。对应顶部导航栏上的第一个选项。

图 4-4-1

具体功能如下：

3.1群发邮件：点击“群发邮件”可以选择对哪些用户同时发送同一封邮件，可以设置邮件主题和邮件内容。邮件内容最多输入 200 个字符。而且必须选择收件人邮箱和指定邮件主题，否则会提示发送失败等提示。

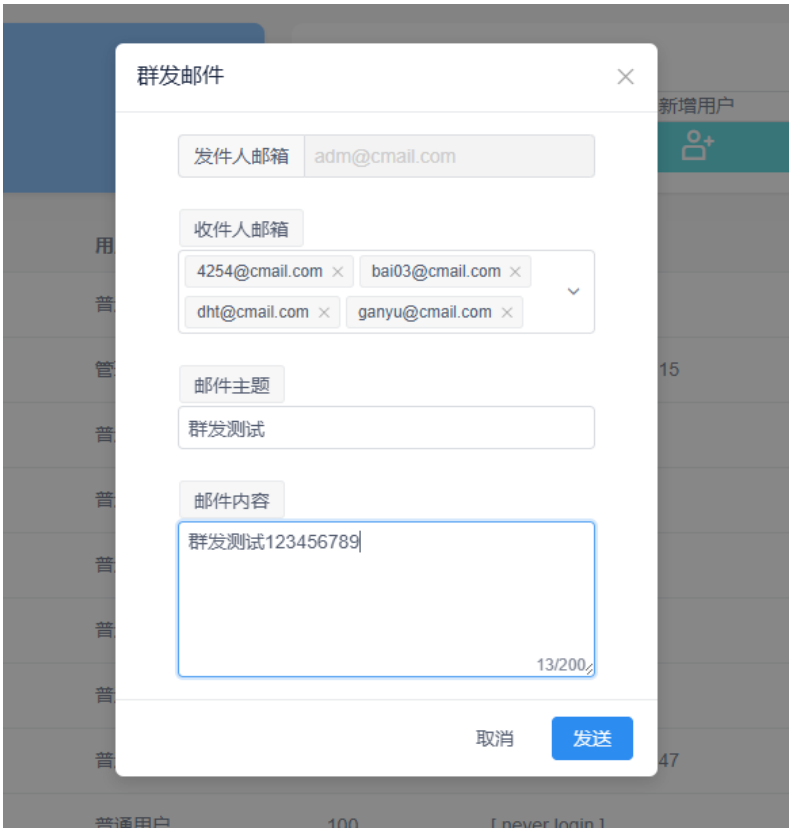


图 4-4-1-2

3.2新增用户：点击“新增用户”按钮可以创建新用户。可以设置用户邮箱、密码、用户权限、邮箱大小。其中用户邮箱前缀不超过 20 个字符，密码不超过 16 个由数字、字母组成的字符，输入框已经对二者做出了限制。用户权限分为“普通用户”和“管理员”，邮箱大小可以设置为 0~65536 之间的整数。

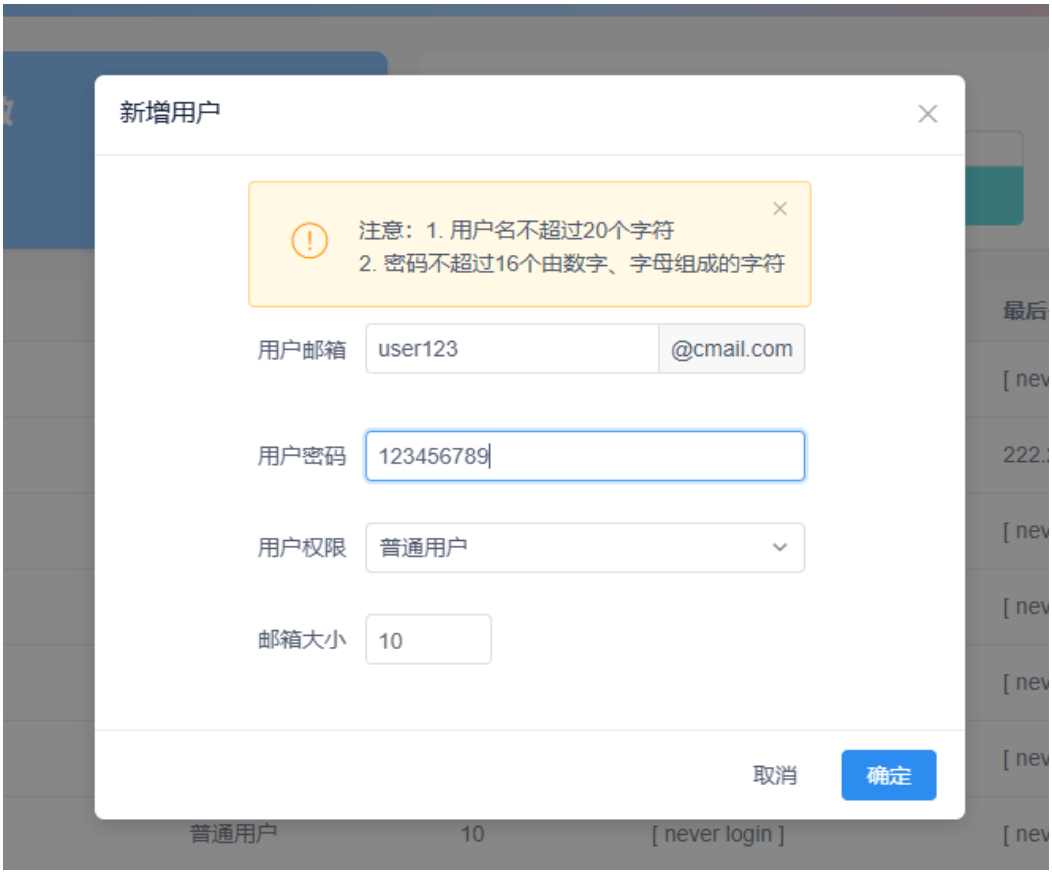


图 4-4-2-1

3.3 清空列表：点击“清空列表”按钮可以删除所有用户信息，建议非必要时千万别乱点。

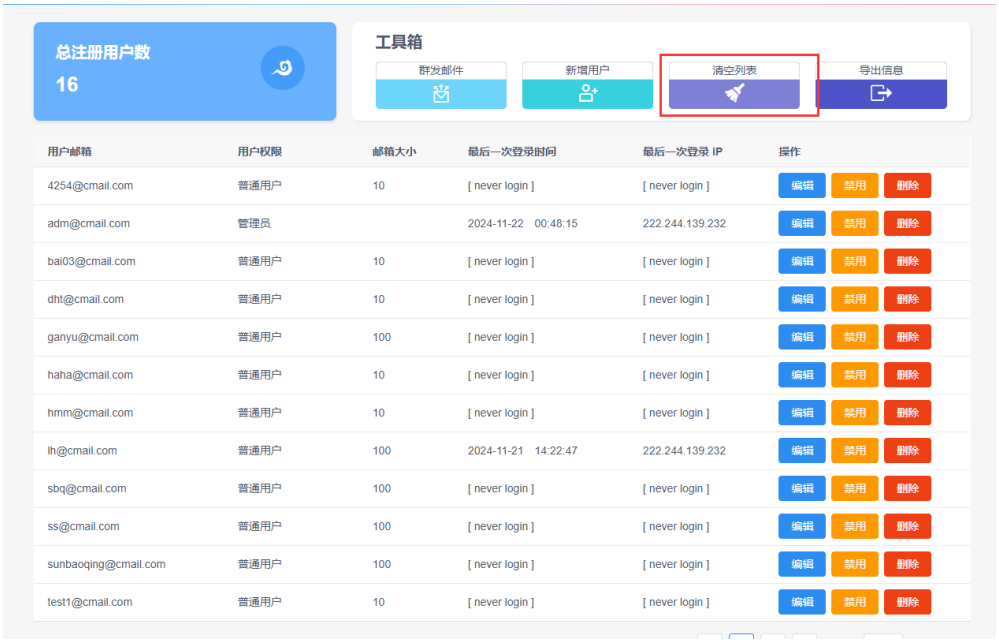


图 4-4-3-1

3.4 导出信息：点击“导出信息”按钮可以导出所有用户的信息为 excel 表格，格式为.csv。

图 4-4-4-1

```
用户邮箱,用户权限,邮箱大小,最后一次登录时间,最后一次登录 IP,操作
4254@cmail.com,0,10,,,
adm@cmail.com,1,,2024-11-21T16:48:15.000+00:00,222.244.139.232,
bai03@cmail.com,0,10,,,
dht@cmail.com,0,10,,,
ganyu@cmail.com,0,100,,,
haha@cmail.com,0,10,,,
hmm@cmail.com,0,10,,,
lh@cmail.com,0,100,2024-11-21T06:22:47.000+00:00,222.244.139.232,
sbq@cmail.com,0,100,,,
ss@cmail.com,0,100,,,
sunbaoqing@cmail.com,0,100,,,
test1@cmail.com,0,10,,,
```

图 4-4-4-2

3.5 编辑用户：点击行的【编辑】按钮可以修改用户权限和邮箱大小。用户权限分为“普通用户”和“管理员”，邮箱大小可以设置为 0~65536 之间的整数。无法修改用户邮箱。

图 4-4-5

3.6 禁用/恢复：点击【禁用】/【恢复】按钮可以注销用户和恢复用户。注销用户是指被注销的用户无法连接到服务器，即无法使用所有功能。恢复用户就是取消对被注销用户的限制，使其恢复正常使用。

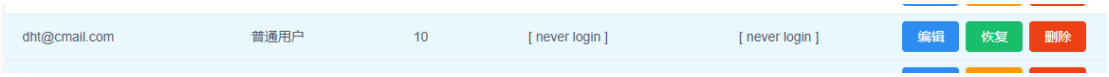
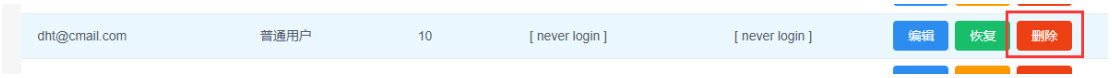


图 4-4-6

3.7删除用户：点击【删除】按钮可以删除对应的用户信息。



4. 邮件管理

图 4-4-7

登录成功后，点击顶部导航栏上的第二个选项“邮件管理”。具体功能如下：

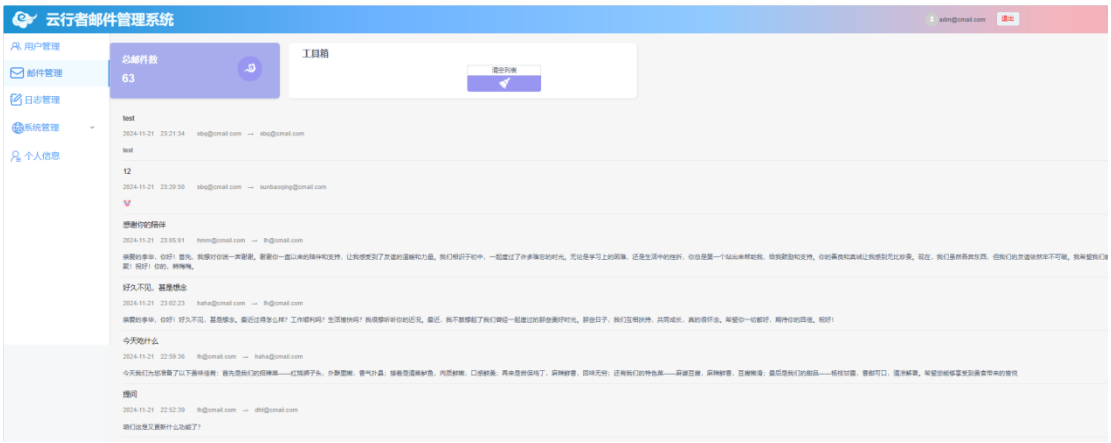


图 4-5-0

4.1清空列表：点击“清空列表”按钮可以清空所有日志信息，建议别乱点。

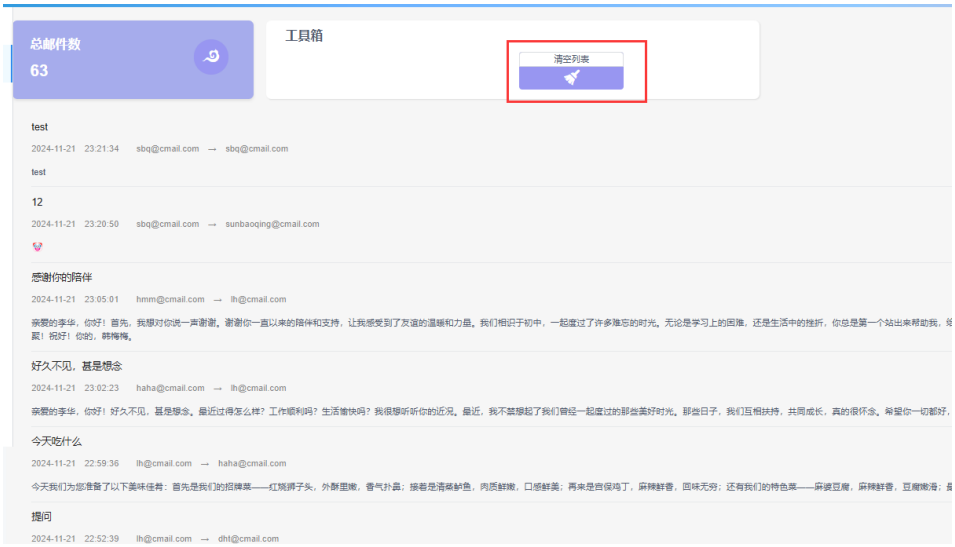


图 4-5-1-1

5. 日志管理

登录成功后，点击顶部导航栏上的第三个选项“日志管理”。具体功能如下：

5.1清空列表：点击“清空列表”按钮可以清空所有日志信息，建议别乱点。

图 4-5-1-1

5.2 导出信息：点击“导出信息”按钮可以导出所有日志信息为 excel 表格，格式为.csv。

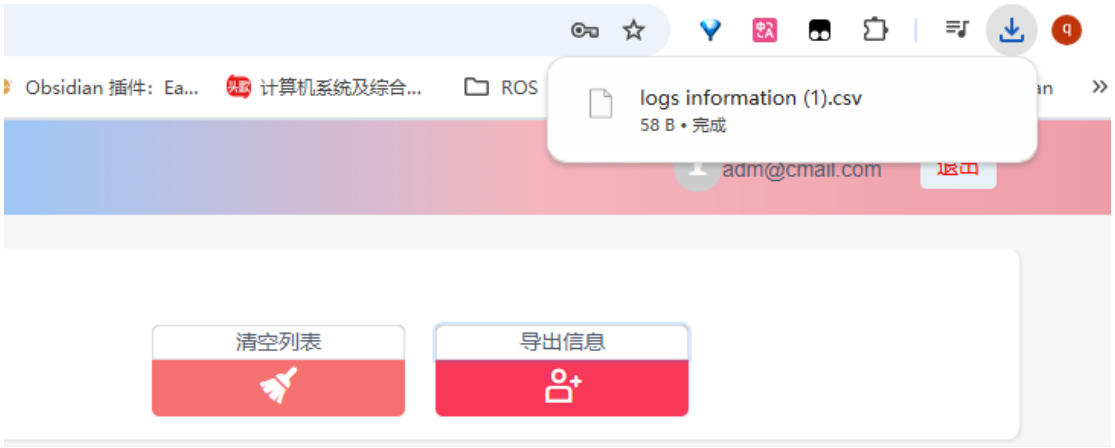


图 4-5-2-2

5.3删除日志：点击【删除】按钮可以删除对应的日志信息

6. 系统管理

登录成功后，点击顶部导航栏上的第四个选项“系统管理”。

图 4-6-1

具体功能如下：

6.1 服务设置：进入“系统管理”功能页面首先显示的是“服务设置”子功能，可以关闭和开启 SMTP 和 POP3 服务，以及修改他们的端口号，端口号范围严格限制为 0~65535 之间的整数，且 SMTP 和 POP3 服务的端口号不能相同。

图 4-6-1-1

6.2IP 过滤：在左侧导航栏点击“IP 过滤”，被拦截的 ip 无法访问服务器.

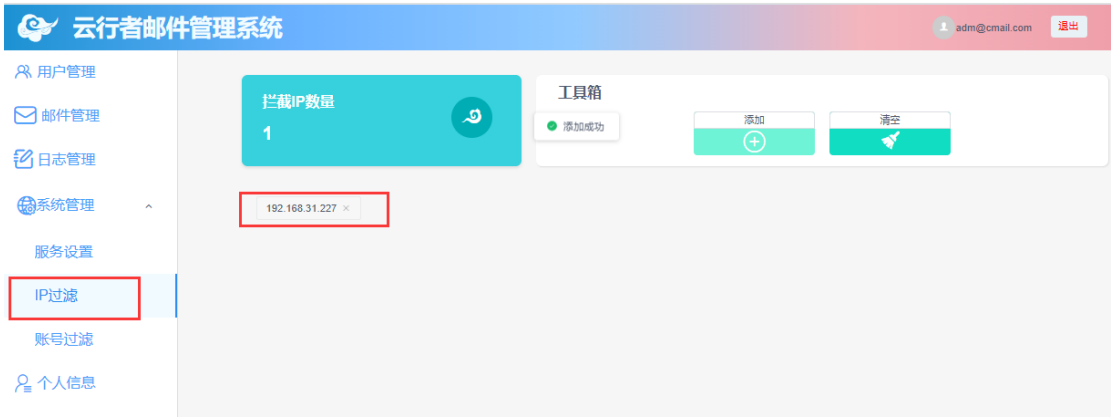


图 4-6-2-1

6.3账号过滤：其他用户将无法收到被过滤账号发来的邮件。

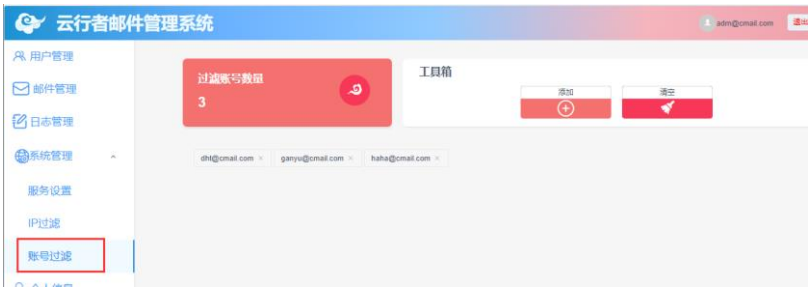


图 4-6-3-1

五、服务器模块

1. SMTP 服务模块

1.1 SMTP 状态变量

SMTP 服务在后端的实现为开启一个线程进行相关处理,这个服务线程为全局变量，并且为单例实例;维护全局静态的服务端 ServerSocket 对象、SMTP服务端口、SMTP 启停状态、线程池(用于处理前端提交的 smtp 任务)以及所有与服务端连接的套接字列表。

```
public class SmtpServer extends Thread {

    private static ServerSocket serverSocket;
    private static int port;
    private static boolean shutDown;
    private static ThreadPoolExecutor executor;
    private static List<Socket> clients;
```

图 5-1-1-1

1.2 实现算法

使用面向连接的多线程循环服务器算法；smtp 服务线程为主线程，在一个 while 循环里面不断监听代理服务器的连接请求,并使用 accept 函数建立 tcp 连接,每个连接封装为 SmtpServerRunnable 对象，交由线程池进行执行。

```
public void run() {
    try {
        while (true) {
            if(isShutDown()){
                return;
            }
            Socket socket = serverSocket.accept();
            clients.add(socket);
            PrintWriter printWriter = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()), autoFlush: true);
            printWriter.println("220 SMTP ready");
            SmtpServerRunnable t = new SmtpServerRunnable(socket);
            executor.execute(t);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

图 5-1-1-2

SmtpServerRunnable 里面的具体处理过程,匹配指令,并调用对应的处理函数进行处理。

```
public void run() {
    try {
        writer = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()), autoFlush: true);
        reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));

        while (true) {
            String command = reader.readLine();
            System.out.println("客户端: " + command);
            //解析客户端指令
            String[] args = CommandParse.parseCommand(command);
            if (args == null) {
                writer.println(SmtpStateCode.COMMAND_ERROR_DESC);
            } else {
                if (HELO.equals(args[0])) {
                    System.out.println("执行" + HELO);
                    smtpService.handleHelloCommand(args);
                } else if (CommandConstant.AUTH_LOGIN_PREFIX.equals(args[0])) {
                    System.out.println("执行" + AUTH_LOGIN);
                    smtpService.handleAuthCommand(args);
                } else if (CommandConstant.MAIL_FROM_PREFIX.equals(args[0])) {
                    System.out.println("执行" + MAIL_FROM);
                    smtpService.handleMailCommand(args);
                } else if (CommandConstant.RCPT_TO_PREFIX.equals(args[0])) {
                    smtpService.handleRcptCommand(args);
                    System.out.println("执行" + RCPT_TO);
                } else if (CommandConstant.DATA.equals(args[0])) {
                    smtpService.handleDataCommand(args);
                    System.out.println("执行完成");
                } else if (CommandConstant.REST.equals(args[0])) {
                    smtpService.handleResetCommand(args);
                } else if (CommandConstant.QUIT.equals(args[0])) {
                    smtpService.handleQuitCommand(args);
                    reader.close();
                    writer.close();
                    socket.close();
                    break;
                } else {
                    System.out.println("SMTP 没有命令" + args[0]);
                    writer.println(SmtpStateCode.COMMAND_ERROR_DESC);
                }
            }
        }
```

图 5-1-1-3

1.3 SMTP 指令的具体处理函数

根据 SMTP 指令的含义，实现其处理逻辑，例如 HELO、AUTH LOGIN、 MAIL 等指令，这部分是具体的处理逻辑，是实现各个指令功能的关键，涉及到指令解析，参数提取，访问数据库等操作。采用依赖倒置的设计原则，将 SMTP 处理函数的约束和具体实现解耦合，使得代码维护起来更加方便。

以 AUTH LOGIN 的处理函数为例：

```java
public void handleAuthCommand(String[] args) throws Exception {
    if (!this.session.isHelloSent()) {
        this.writer.println(SmtpStateCode.SEQUENCE_ERROR + " send HELO first");
        return;
    }
    if (args.length != 2) {
        this.writer.println(SmtpStateCode.COMMAND_ERROR_DESC);
    } else {
        String command = args[0] + args[1].toUpperCase();
        if (!CommandConstant.AUTH_LOGIN.replaceAll( regex: " ",  replacement: "").equals(commar
            this.writer.println(SmtpStateCode.COMMAND_ERROR_DESC);
        } else {
            this.writer.println(SmtpStateCode.USERNAME_SENT_DESC);
            // base64 user
            String encodedUsername = this.reader.readLine();
            this.writer.println(SmtpStateCode.PASSWORD_SENT_DESC);
            // base64 pass
            String encodedPassword = this.reader.readLine();

            String username = Base64Util.decodeByBase64(encodedUsername.getBytes());
            String password = Base64Util.decodeByBase64(encodedPassword.getBytes());
            if (!serverService.isRightServer(username)) {
                this.writer.println(SmtpStateCode.ADDRESS_NOT_AVAILABLE_DESC + "<\"" + use
                return;
            }
            username = username.split( regex: "@")[0];
            // 登录验证
            String result = authService.handleLogin(username, password);
            if ("SUCCESS".equals(result)) {
                result = SmtpStateCode.AUTH_SUCCESS_DESC;
                this.session.setAuthSent(true);
                //这才是真正的发信人地址
```

图 5-1-1-4

2.POP3 服务模块

2.1 POP3 状态变量

POP3 服务在后端的实现为开启一个线程进行相关处理，这个服务线程为全局变量，并且为单例实例；维护全局静态的服务端 ServerSocket 对象、POP3服务端口、POP3 启停状态、线程池(用于处理前端提交的 smtp 任务)以及所有与服务端连接的套接字列表。

```java
public class Pop3Server extends Thread {

    private static ServerSocket serverSocket;
    private static int port;
    private static boolean shutDown;
    private static ThreadPoolExecutor executor;
    private static List<Socket> clients;

    private int port1;
```

图 5-2-1-1

2.2 算法实现

使用面向连接的多线程循环服务器算法：smtp 服务线程为主线程，在一个while 循环里面不断监听代理服务器的连接请求，并使用 accept 函数建立 tcp 连接，每个连接封装为 POP3ServerRunnable 对象，交由线程池进行执行。

```
public void run() {
    try {
        while (true) {
            if (isShutDown()) {
                break;
            }
            Socket socket = serverSocket.accept();
            clients.add(socket);
            PrintWriter printWriter = new PrintWriter(new OutputStreamWriter(socket.getOutpu
            printWriter.println(Pop3StateCode.READY);
            Pop3ServerRunnable t = new Pop3ServerRunnable(socket);
            executor.execute(t);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

图 5-2-2-1

2.3 POP3 指令的具体处理函数

根据 POP3 协议，实现其处理逻辑,例如 USER,PASS,RETR 等指令的处理逻辑，是实现各个指令功能的关键，涉及到指令解析，参数提取，访问数据库等操作。采用依赖倒置的设计原则，将 POP3 处理函数的约束和具体实现解耦合，使得代码维护起来更加方便。以 AUTH LOGIN 的处理函数为例：

```
public void run() {
    try {
        writer = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()), autoFlush
        reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));

        while (true) {
            String command = reader.readLine();
            System.out.println("客户端: " + command);
            String[] args = CommandParse.parseCommand(command);
            // 输入空格或回车
            if (args == null) {
                //writer.println(Pop3StateCode.ERR + Pop3StateCode.STNTAX);
                int a = 1;
            } else {
                if (USER.equals(args[0])) {
                    System.out.println("执行" + USER);
                    pop3Service.handleUserCommand(args);
                } else if (PASS.equals(args[0])) {
                    System.out.println("执行" + PASS);
                    pop3Service.handlePassCommand(args);
                } else if (STAT.equals(args[0])) {
                    System.out.println("执行" + STAT);
                    pop3Service.handleStatCommand(args);
                } else if (LIST.equals(args[0])) {
                    System.out.println("执行" + LIST);
                    pop3Service.handleListCommand(args);
                } else if (RETR.equals(args[0])) {
                    System.out.println("执行" + RETR);
                    pop3Service.handleRetrCommand(args);
                } else if (DELE.equals(args[0])) {
                    System.out.println("执行" + DELE);
```

图 5-2-3-1

3. IP 过滤

3.1 获取 IP 地址

源IP 地址位于 HTTP 请求头部的 x-forwarded-for 字段中，这里我们考虑到客户端的请求可能经过很多代理服务器才到服务端，因此这里要代理服务器的 ip 的存储位置,比如 Proxy-Client-IP 等字段。

```java
public class IpUtil {

    public static String getIp(HttpServletRequest request) {

        String ip = request.getHeader( s: "x-forwarded-for");
        if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {
            ip = request.getHeader( s: "Proxy-Client-IP");
        }
        if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {
            ip = request.getHeader( s: "WL-Proxy-Client-IP");
        }

        if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {
            ip = request.getRemoteAddr();
        }
        //真实的IP地址是32位
        return "0:0:0:0:0:0:0:1".equals(ip) ? "127.0.0.1" : ip;

    }
}
```

图 5-3-1-1

3.2 拦截 IP

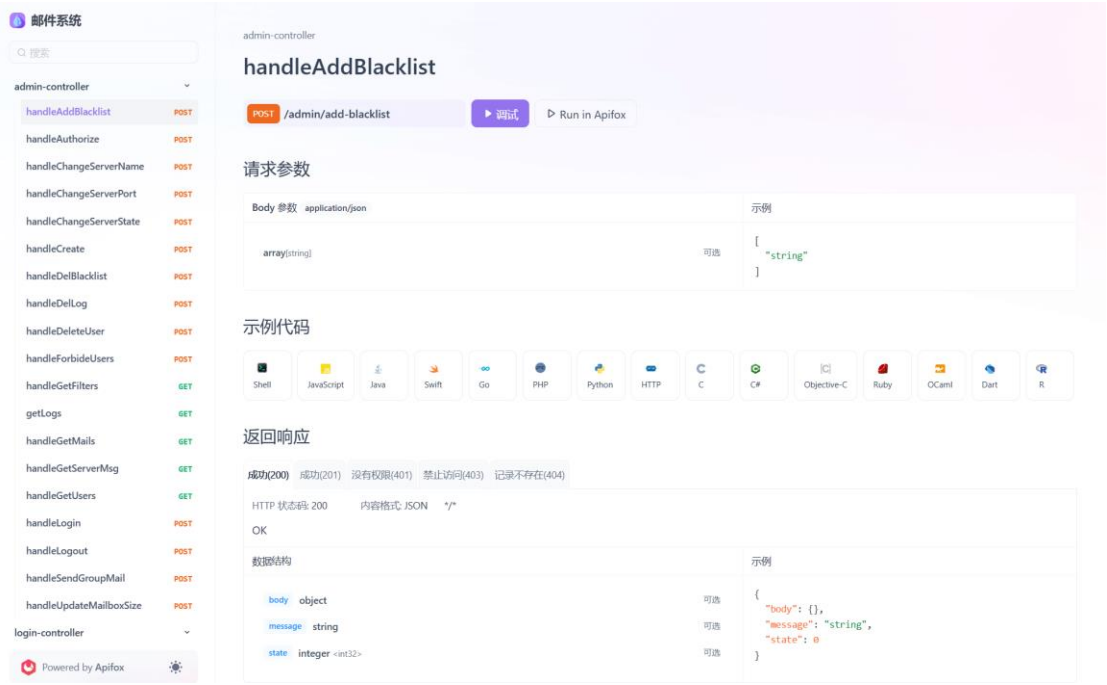
这里使用 SpringBoot 支持的 HandlerInterceptor 接口来实现 ip 地址过滤功能，通过捕获的客户端 IP 地址，再去数据库里面的 IP 黑名单进行查询，如果存在 该 IP 地址，则禁止其访问。统一的全局拦截器，实现和维护都很方便。

```java
@Override
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object
    String ip = IpUtil.getIp(request);
    JsonResult jsonResult = JsonResultFactory.buildJsonResult(
            JsonResultStateCode.INTERRUPTED,
            JsonResultStateCode.INTERRUPTED_DES,
             body: null
    );
    JSONObject jsonObject = (JSONObject) JSONObject.toJSON(jsonResult);
    response.setCharacterEncoding("UTF-8");
    response.setContentType("application/json; charset=utf-8");
    PrintWriter writer = null;
    System.out.println("客户端的IP地址: " + ip);
    if (ip == null || ip.isEmpty()) {
        writer = response.getWriter();
        writer.append(jsonObject.toJSONString());
        writer.flush();
        return false;
    }
    List<Filter> filters = adminMapper.selectFilter();
    for (Filter filter : filters
    ) {
        if (filter.getIpAddress().equals(ip)) {
            writer = response.getWriter();
            writer.append(jsonObject.toJSONString());
            writer.flush();
            System.out.println("拦截客户端: " + ip);
            return false;
        }
    }
    return true;
}
```

图 5-4-1-4

4. 接口文档

访问https://bsu4pwft87.apifox.cn
密码：KaF2OfaU



六、客户端模块

1．登录功能

用户点击登录，输入账户和密码即可登录，账户必须是已经注册的才可以登录。登录后的页面为

图 6-1-1

2．注册功能

图 6-1-2

用户点击注册输入邮箱名和密码即可注册用户，注册的用户名不能与已注册的用户名重复。

图 6-1-3

3．发送邮件功能

用户点击界面上的加号发送邮件，第一行输入接收邮箱，第二行输入邮件主题，第三行输入邮件内容点击发送即可发送邮件到目标用户的收件箱。

4．收件箱功能

图 6-1-4

用户点击收件箱，会显示出近期接收到的邮件，包括收件人，邮件标题，内容，按发送时间由近到远从上到下排列，并且用户可以通过点击右上角的删除按钮删除发件箱中的邮件。

图 6-1-5

5．发件箱功能

图 6-1-6

用户点击发件箱，会显示出近期发送的邮件，包括邮件标题，发件人，发送时间，内容，按发送时间由近到远从上到下排列，并且用户可以通过点击右上角的删除按钮删除收件箱中的邮件。

图 6-1-7

七、 数据库模型

1. 数据库表

1.1 user

```sql
-- auto-generated definition
create table user
(
    username          varchar(30)   not null
        primary key,
    password          char(44)      not null,
    phone             varchar(50)   null,
    account_type      int           not null,
    latest_login_time timestamp     null,
    latest_login_ip   varchar(45)   null,
    mail_box_size     int           null,
    avatar_url        varchar(2048) null,
    logout            tinyint(1)    null,
    forbidden         tinyint(1)    null
)
    charset = utf8mb4
    row_format = DYNAMIC;
```

1.2 email

```sql
-- auto-generated definition
create table email
(
    mid           int           not null
        primary key,
    sender_email   varchar(30)   null,
    receiver_email varchar(30)   null,
    send_time      datetime      null,
    subject        varchar(50)   null,
    body           text          null,
    `read`         tinyint(1)    null,
    deleted        tinyint(1)    null,
    tag            tinyint(1)    null,
    send           tinyint(1)    null,
    annex_url      varchar(2048) null,
    summary        varchar(50)   null,
    size           double        null
)
    charset = utf8mb4
    row_format = DYNAMIC;
```

1.3 contact

```sql
-- auto-generated definition
create table contact
(
    username     varchar(30) not null,
    contact_name varchar(30) not null,
    add_time     datetime    null,
    primary key (username, contact_name)
)
    charset = utf8mb4
    row_format = DYNAMIC;
```

1.4 log

```sql
-- auto-generated definition
create table log
(
    log_id    int         not null
        primary key,
    username  varchar(30) null,
    time      timestamp   null,
    operation text        null,
    state     tinyint(1)  null,
    reason    text        null
)
    charset = utf8mb4
    row_format = DYNAMIC;
```

1.5 filter

```sql
-- auto-generated definition
create table filter
(
    fid        int         not null
        primary key,
    ip_address varchar(15) null
)
    charset = utf8mb4
    row_format = DYNAMIC;
```

1.6 server-msg

```sql
-- auto-generated definition
create table `server-msg`
(
    sid         int         not null
        primary key,
    server_name varchar(10) null,
    server_ip   varchar(15) null,
    smtp_state  tinyint(1)  null,
    smtp_port   int         null,
    pop3_state  tinyint(1)  null,
    pop3_port   int         null
)
    charset = utf8mb4
    row_format = DYNAMIC;
```

八、项目总结

1. 项目感想

在本次课程设计中，我们体会到了虽然一开始接触一个项目会觉得陌生、令人望而却步，但是我们要不断挑战自己未知的领域，才能够不断进步。并且在学习的过程中我们要不断利用我们已经学过的知识来化解难题。

做项目之前，还是要弄清楚需求，再根据需求设计可行性方法，然后再反过来修改需求。软件开发流程中每上一个阶段都是下一个阶段的实施进行的基础。编程也是根据对软件设计，将软件设计的各部分需求通计算机程序代码来实现运行，编程有统一、规范的程序编写规则，保证软件程序的易懂性、易维护性。这次课程设计我们组在整理需求上的时间较少。不过最后还算比较好的完成了任务。

总之通过这一次实验，开发过程中组员之间积极交流、通力合作也让我们的项目能够出色的完成，我们的收获很大，希望以后在学习的道路上能够更进一步， 提高了专业素养。