目录

[**一、** **项目选题** 2](#_Toc25597286)

[**程序功能：** 2](#_Toc25597287)

[**二、** **开发环境和工具** 2](#_Toc25597288)

[**三、** **项目使用说明手册** 2](#_Toc25597289)

[**四、** **系统结构** 2](#_Toc25597290)

[**1.** **项目目录结构** 2](#_Toc25597291)

[**2.** **系统执行过程** 3](#_Toc25597292)

[**五、** **系统内核心实现** 4](#_Toc25597293)

[**1.** **模块动态加载** 4](#_Toc25597294)

[**2.** **对webpage下的服务进行扫描** 4](#_Toc25597295)

[**3.** **通过retr命令获取邮件，并且通过json格式返回到web前端** 5](#_Toc25597296)

[**4.** **配置文件的读取** 6](#_Toc25597297)

[**六、** **项目运行截图** 7](#_Toc25597298)

[**1.** **首页** 7](#_Toc25597299)

[**2.** **邮箱邮件查询** 7](#_Toc25597300)

[**3.** **邮件发送** 8](#_Toc25597301)

[**4.** **邮件详情** 9](#_Toc25597302)

[**5.** **配置文件设置** 9](#_Toc25597303)

[**6.** **Web服务器错误页** 10](#_Toc25597304)

[**7.** **Web服务器主页** 10](#_Toc25597305)

[**8.** **两个不同的邮箱进行测试** 11](#_Toc25597306)

1. **项目选题**

基于实现的一个Web服务器的网页版邮箱客户端系统

项目中的服务器为自己实现的Web服务器，对http访问进行响应

项目中的邮箱客户端是基于socket进行smtp和pop3邮箱连接的客户端实现

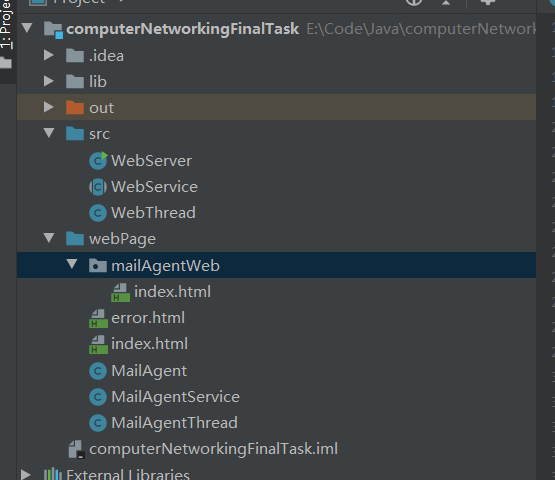
该项目将Web服务器和邮件客户端两个项目选题相结合

**程序功能：**

1. Web服务器功能对响应的HTTP资源的请求和对于邮件api的访问给予响应
2. 邮件能够实现查询邮件，查看邮件详细信息，删除邮件，发送邮件，邮件返回分页等功能
3. 此项目在163邮箱和qq邮箱测试成功
4. **开发环境和工具**

|  |  |
| --- | --- |
| **系统环境** | Windows10 1903 |
| **开发工具** | Intellij Idea Ultimate 2019、WebStorm、Chrome |
| **jdk版本** | 12.0.2（如果在jdk1.8的情况下报错，请按照错误修改） |

1. **项目使用说明手册**
2. 使用idea加载并打开工程
3. 在serverConfig.properties文件中进行相关的配置文件的配置
4. 浏览器输入<http://127.0.0.1:8080/index.html>进入主页面(如果未修改配置文件信息)，输入不存在的资源目录则进入错误页面
5. 浏览器输入<http://127.0.0.1:8080/mailAgentWeb/index.html进入邮箱客户端web>主页面
6. 在该页面可以实现邮件查询、邮件删除、邮件详细信息查看等功能，切换tab可以使用邮件发送功能
7. **系统结构**
8. **项目目录结构**



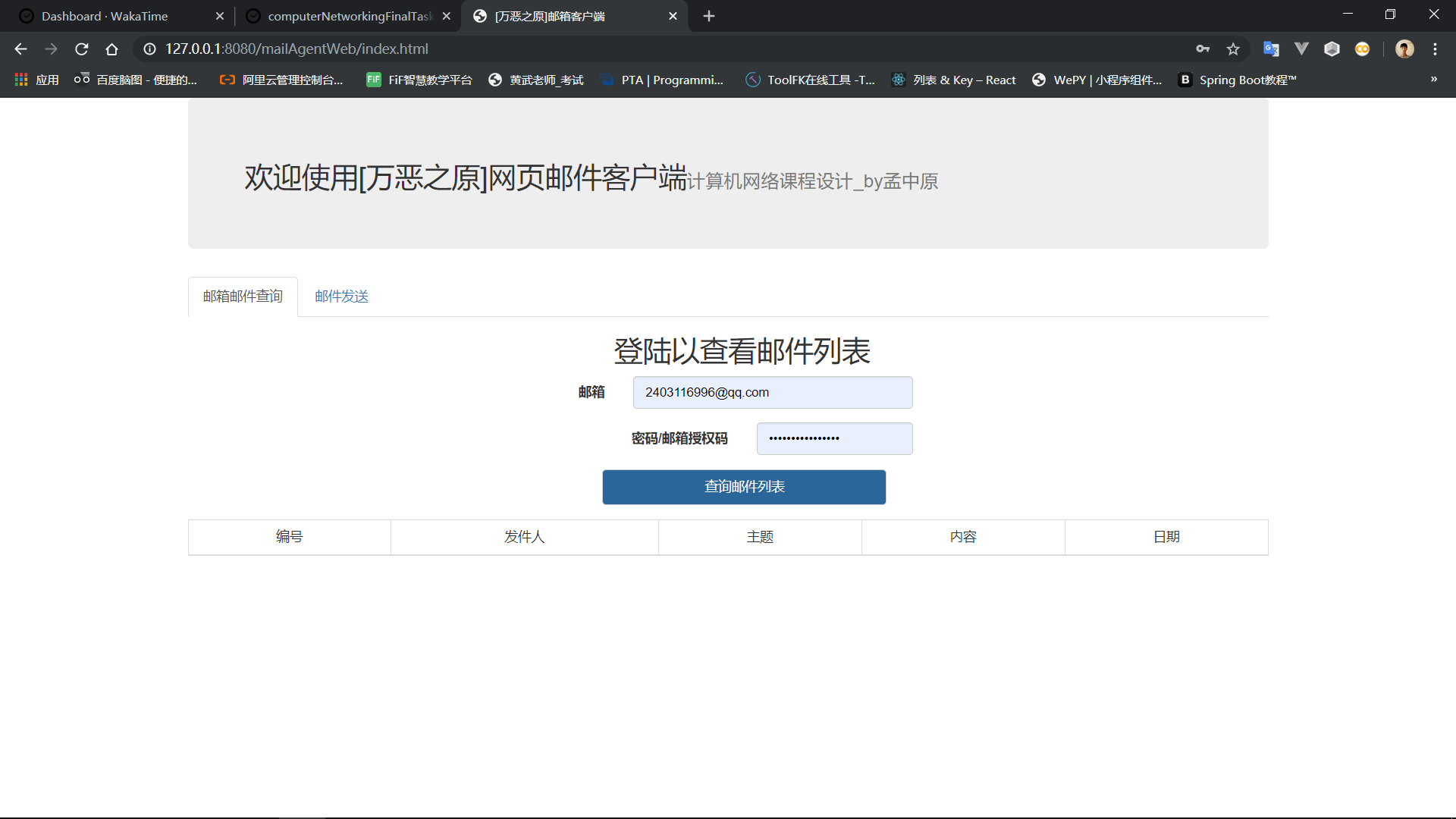
Src目录下为webserver的实现，webServer为web服务器主程序，监听8080端口对http请求进行响应，webThread为连接线程，当监听到socket连接时，创建一个webThread进行响应；WebServce为一个抽象类，是所有的可以部署在这个web服务器上的服务的主程序的父类，通过动态加载，将服务加载到服务器上进行运行；

Webpage目录下的index.html是服务器的默认首页，服务器启动之后，访问<http://127.0.0.1:8080/index.html>即可访问，mailAgentWeb文件夹下是基于web的邮件客户端的前端页面，主要使用bootstrap和jquery进行编写，error.html是访问到其他页面的错误页面的提示页面；以Mail开头的三个类是与邮件客户端的后端服务相关的类，负责对请求的响应作出回应，主要使用了stmp协议和pop3协议。

1. **系统执行过程**

当web服务器模块启动时，首先会寻找项目根目录文件夹下面的serverConfig.properties文件，将文件中对于服务器的相关的的配置项加载到一个map中，，当我们在进行相关的配置值的使用时，首先判断是否在配置文件已经覆盖，然后在进行默认值的加载配置。然后会遍历webpage目录中结尾为service的类，并通过Class.forName()进行加载之后放到服务的列表中，然后通过循环调用所有的类的start()函数启动服务线程，然后整个服务器启动完成。不仅可以实现对于8080端口的访问请求，而且可以根据webPage中的Service类中自定义的内容添加对应的服务模块进行服务的部署工作。

以mailAgentService为例，在服务器运行的时候该服务会被分配一个线程而启动，用于响应关于邮件方面的http请求，用户通过web服务器的<http://127.0.0.1:8080/mailAgentService/index.html>链接进入到邮箱服务的前端界面中，该前端界面是由自己实现的web服务器进行驱动的，然后用户可以在前端界面上进行邮件的发送和查看操作，并将通过前端进行响应。



1. **系统内核心实现**
2. **模块动态加载**

try {

Class tempClass = Class.forName(item);

service = (WebService) tempClass.newInstance();

System.out.println("[万恶之原]-服务----" + service.getClass().getName() + "----启动完毕！");

service.start();

} catch (Exception e) {

System.out.println("[万恶之原]-加载----" + item + "----启动器时出现问题!");

}

1. **对webpage下的服务进行扫描**

private static List<String> getFiles(String path) {

List<String> list = new ArrayList<>();

File file = new File(path);

String name = null;

File[] files = file.listFiles();

for (int i = 0; i < Objects.requireNonNull(files).length; i++) {

name = files[i].getPath().split("\\\\")[files[i].getPath().split("\\\\").length - 1];

if (name.split("[.]")[name.split("[.]").length - 1].equals("java")){

list.add(name.split("[.]")[0]);

}

}

return list;

}

1. **通过retr命令获取邮件，并且通过json格式返回到web前端**

jsonObject = new JSONObject();

List<Map<String,String>> mailList = new ArrayList<>();

int count = 0;

try {

dos.writeBytes("stat" + CRLF);

count = count();

for (int i = 1; i <= count; i++) {

Map<String,String> mailInfoList = new HashMap<>();

dos.writeBytes("retr " + i + CRLF);

while (true) {

String response = dis.readLine();

response = MimeUtility.decodeText(response);

if(response.contains(":") && response.split(":").length > 1){

mailInfoList.put(response.split(":")[0],response.split(":")[1]);

}else if (!response.contains(":") && !response.toLowerCase().equals(".") && !response.equals("")){

mailInfoList.put("Content",response);

}

if (response.toLowerCase().equals(".")) {

break;

}

}

mailList.add(mailInfoList);

}

jsonObject.put("data",mailList);

return jsonObject.toString();

} catch (IOException e) {

e.printStackTrace();

}

1. **配置文件的读取**

class ReadConfigUtils {

static Map<String,String> configList = new HashMap<>();

private static String ROOT = System.getProperty("user.dir");

static Map<String,String> getConfigList(String fileName){

if (fileName == null){

fileName = "serverConfig.properties";

}

String configFilePath = ROOT + "/" + fileName;

try {

InputStream inputStream = new FileInputStream(new File(configFilePath));

BufferedReader br = new BufferedReader(new InputStreamReader(inputStream));

String line = null;

while((line = br.readLine()) != null){

if(line.contains("="))

configList.put(line.split("=")[0],line.split("=")[1]);

}

return configList;

} catch (IOException e) {

e.printStackTrace();

}

return configList;

}

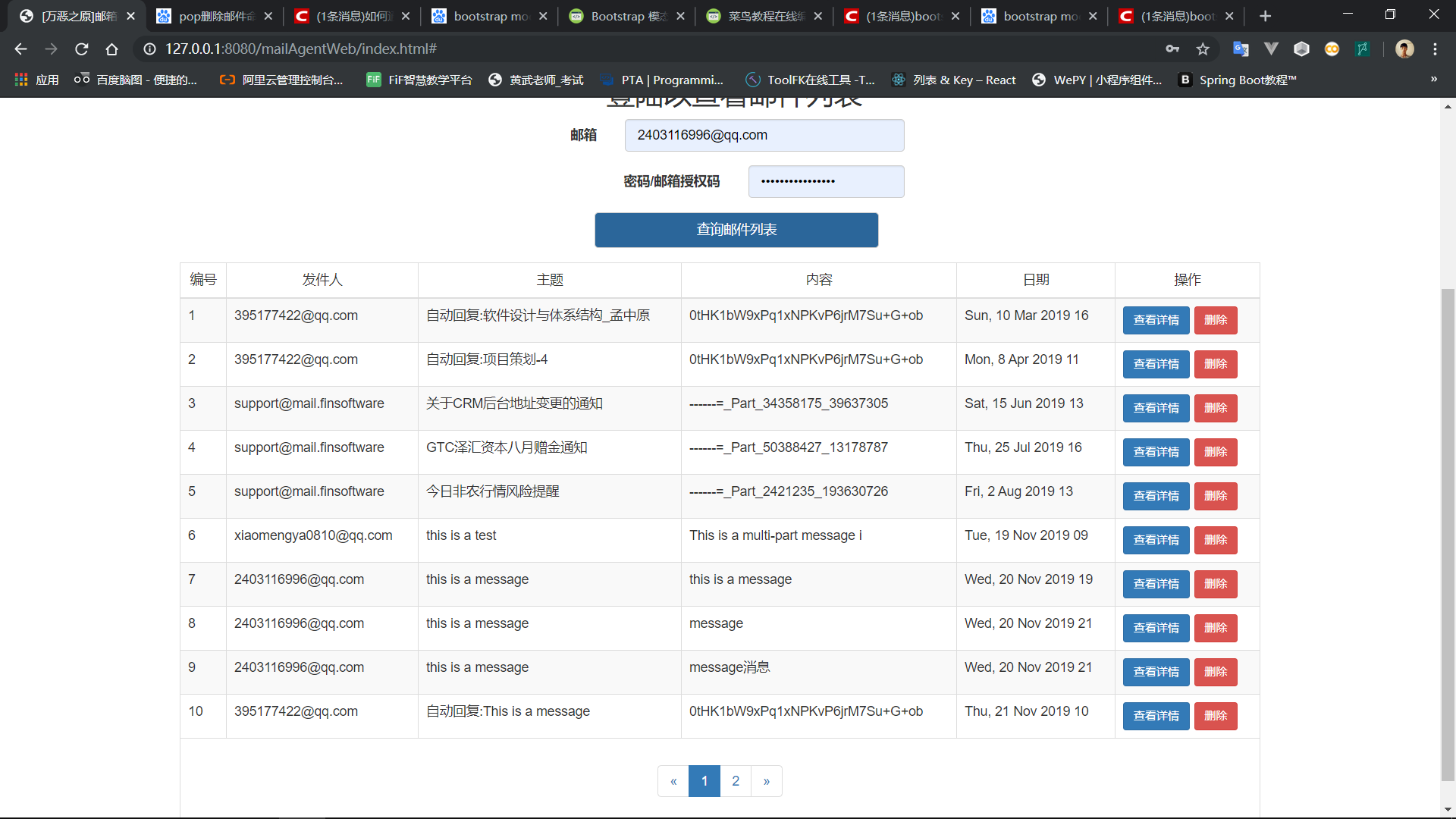
}

1. **项目运行截图**
2. **首页**



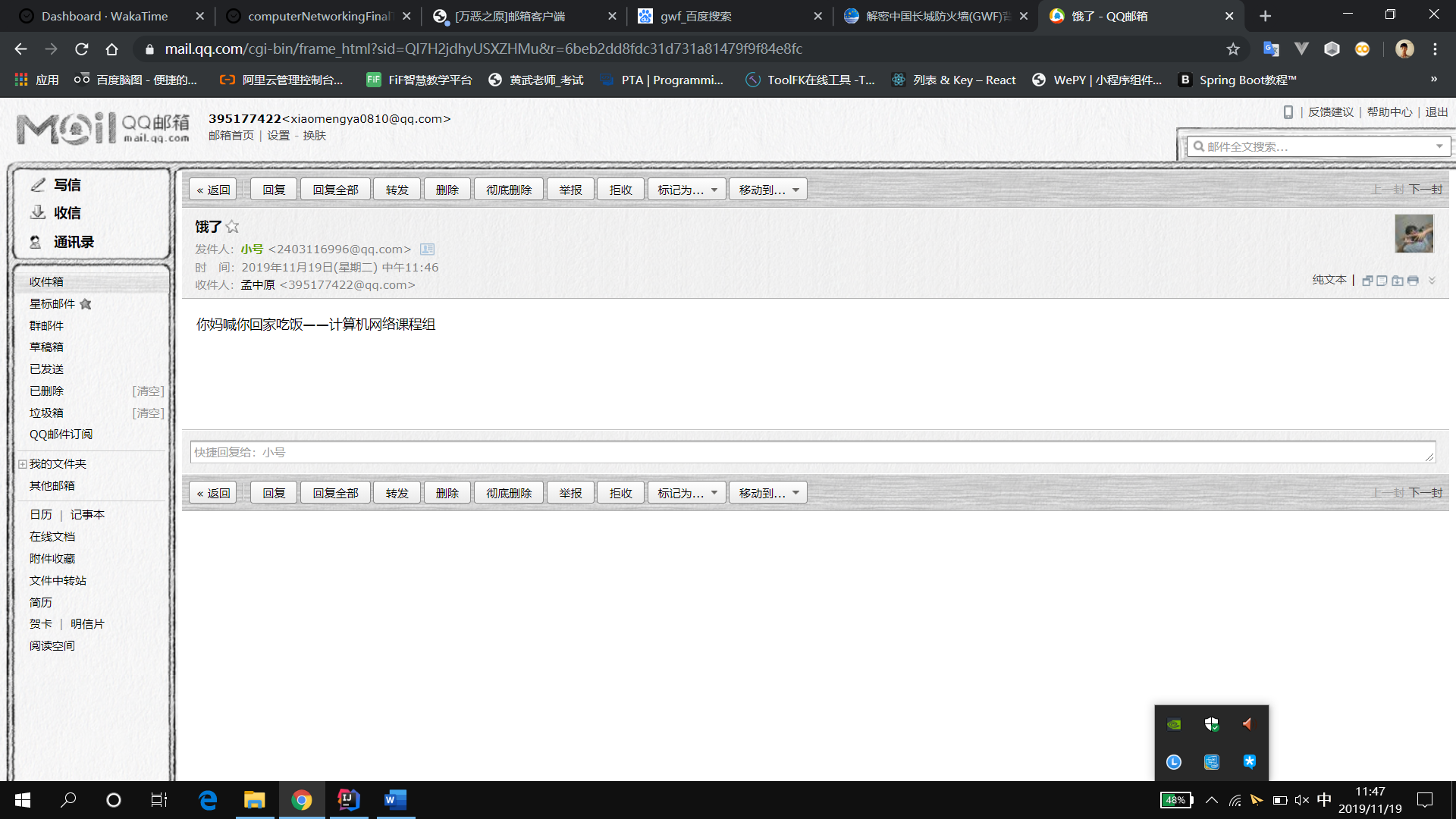
1. **邮箱邮件查询**



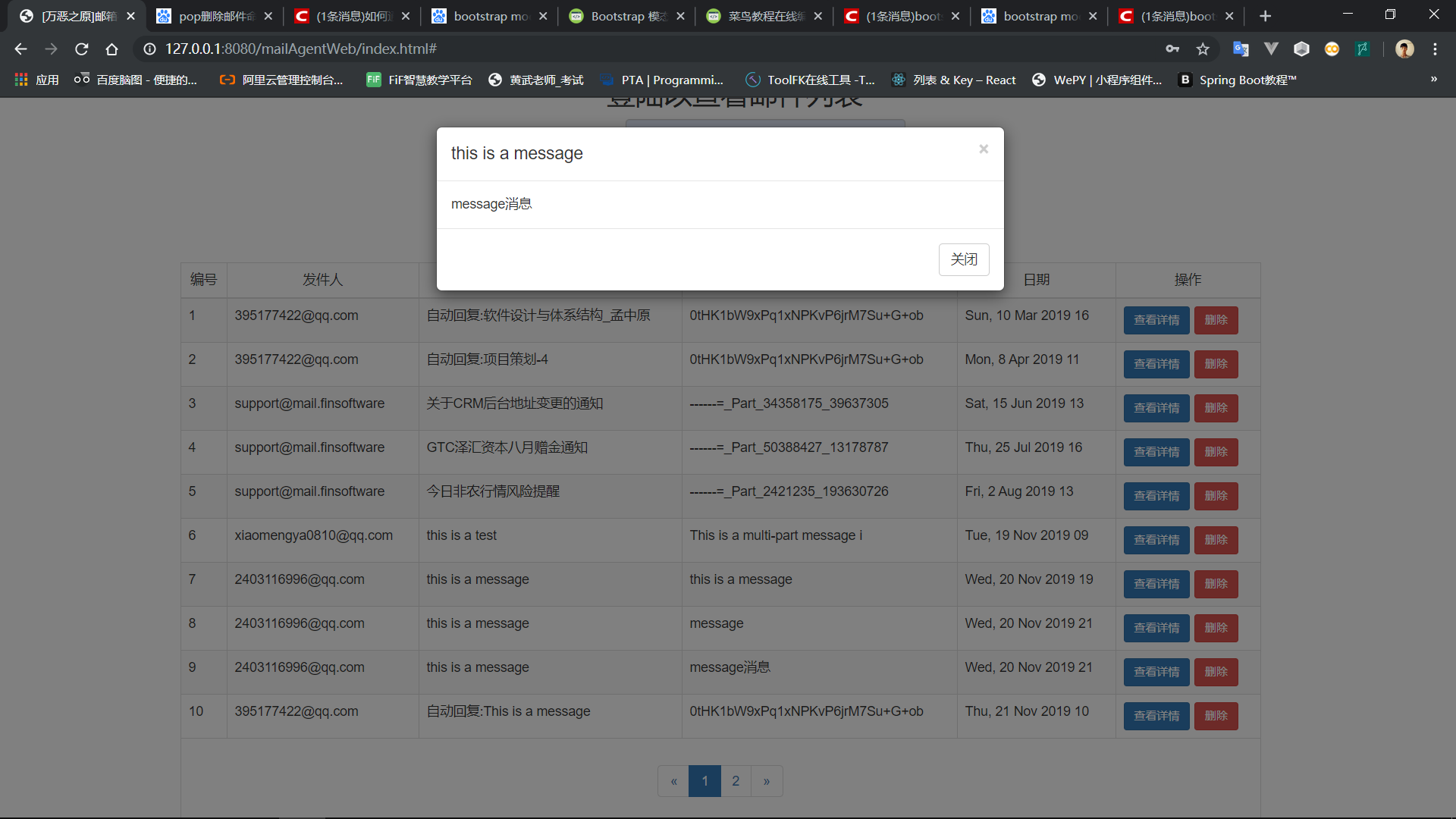


1. **邮件发送**

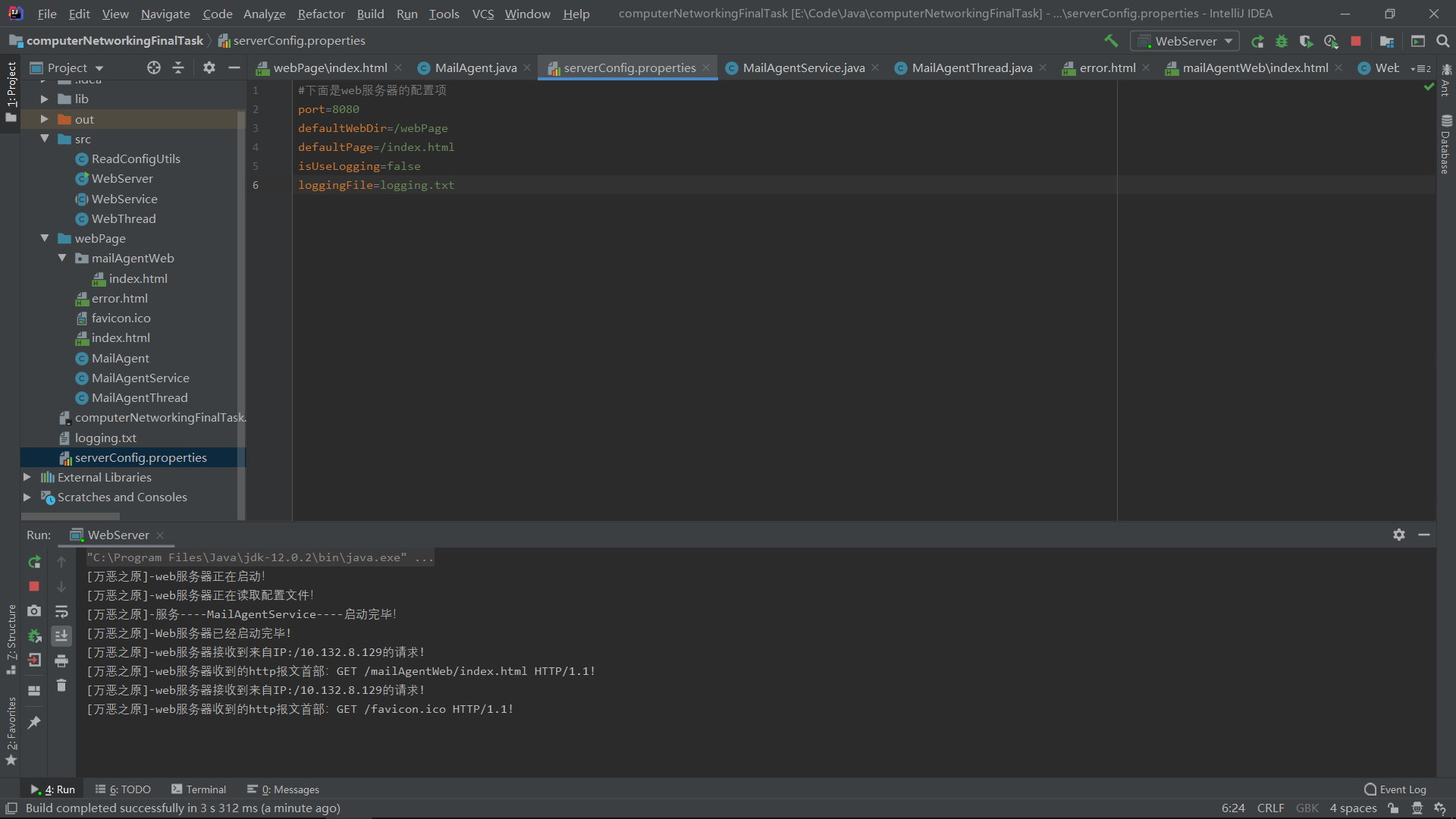




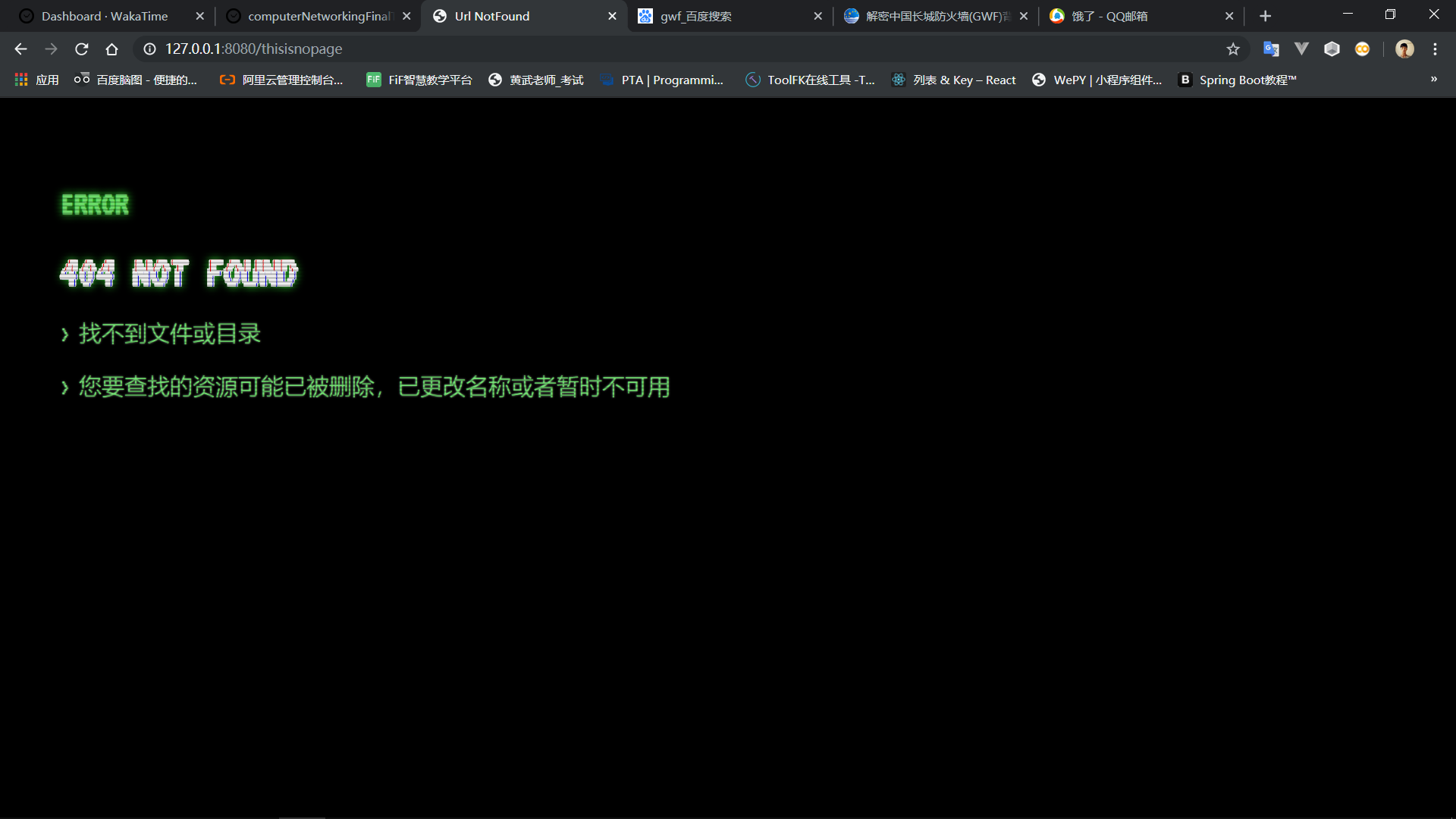
1. **邮件详情**



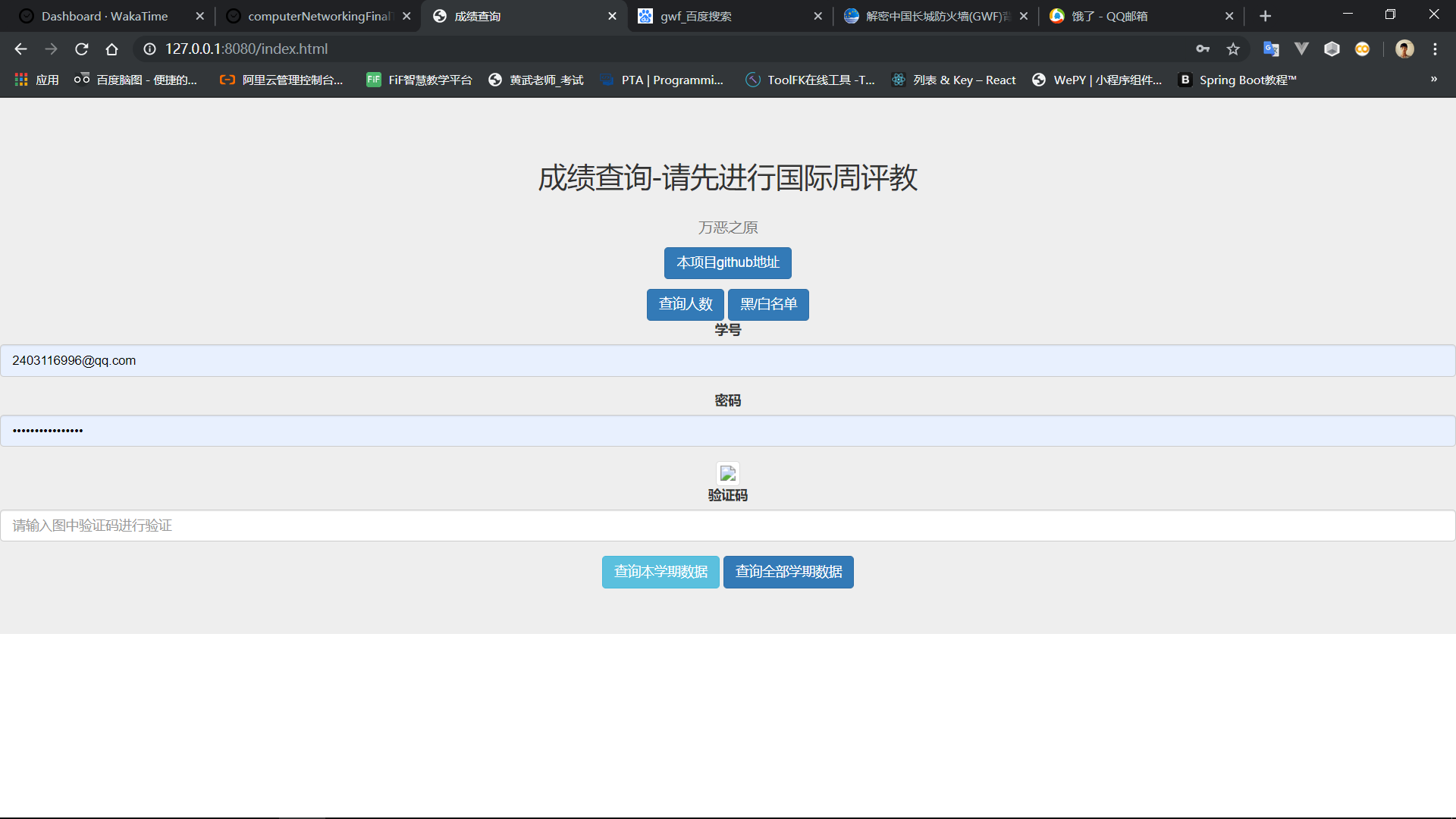
1. **配置文件设置**



1. **Web服务器错误页**

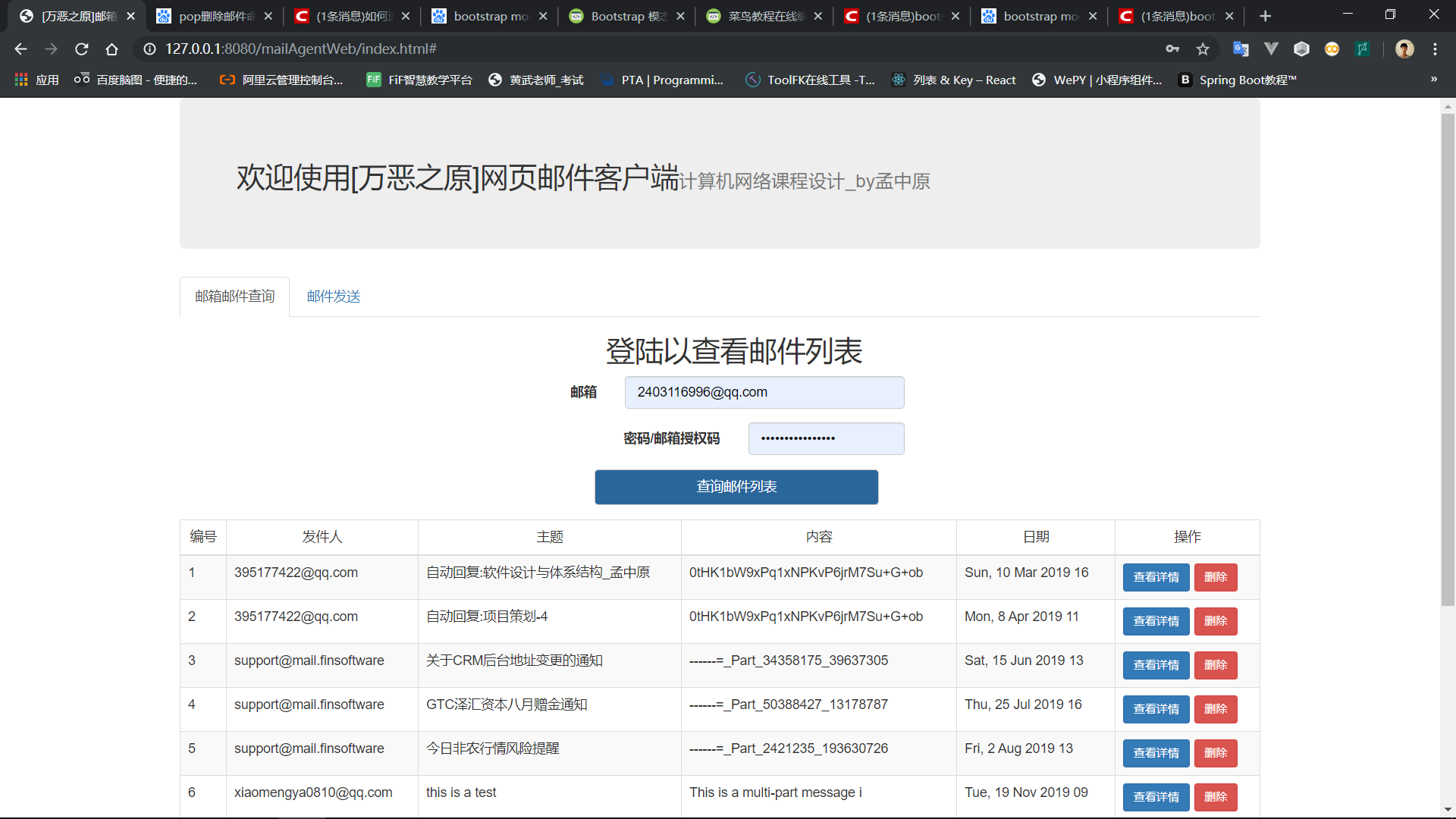


1. **Web服务器主页**



1. **两个不同的邮箱进行测试**

**qq邮箱测试通过：**



**163邮箱测试通过：**

