Санкт-Петербургский Государственный Политехнический Университет
Кафедра Компьютерных Систем и Программных Технологий

ОТЧЕТ

по лабораторной работе

«Разработка SMTP-клиента»

Дисциплина: Сети ЭВМ и телекоммуникации

Работу выполнил студент гр. 43501/3: Муравьёв Ф.Э.

Преподаватель: Вылегжанина К.Д.

Санкт-Петербург
2016

1. **Индивидуальное задание**

Разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции клиента протокола SMTP.

Основные возможности. Приложение должно реализовывать следующие функции:

1) Создание нового письма, включающего такие поля, как **From** (отправитель), **To** (получатель), **Subject** (тема), **Body** (текст).
2) Формирование всех необходимых заголовков письма, с тем, чтобы приёмная сторона не рассматривала данное письмо как спам.
3) Подключение к указанному SMTP-серверу и отсылка созданного письма.
4) Подробное протоколирование соединения клиента с сервером.

Поддерживаемые команды. Разработанное приложение должно реализовывать следующие команды протокола SMTP:

- HELO – передача серверу информации о домене пользователя;
- MAIL FROM – передача серверу адреса отправителя письма;
- RCPT TO – передача серверу адреса получателя письма;
- DATA – передача серверу тела письма;
- QUIT – завершение сеанса связи.

Настройки приложения. Разработанное приложение должно обеспечивать настройку следующих параметров:

1) Собственное доменное имя для передачи в команде HELO
2) Адрес отправителя
3) Доменное имя сервера SMTP

Методика тестирования. Для тестирования приложения следует использовать почтовые серверы, имеющиеся в лаборатории, а также бесплатные почтовые серверы, имеющиеся в сети Internet (http://www.mail.ru, http://www.yandex.ru, http://www.rambler.ru и т.п.). Средствами разработанного приложения осуществляется передача письма на указанные ящики электронной почты. Штатными клиентами электронной почты проверяется корректность доставки почты и правильность параметров письма.

Разработанное приложение должно осуществлять простейшую авторизацию на сервере (команда AUTH LOGIN), следовательно, к серверам использующим более сложную авторизацию подключиться будет невозможно.

2. Описание прикладного протокола для реализации

Сервер SMTP — это конечный автомат с внутренним состоянием. Клиент передает на сервер команду с параметрами. Сервер отвечает на каждую команду строкой, содержащей код ответа и текстовое сообщение, отделенное пробелом. Код ответа — число от 100 до 999, представленное в виде строки, трактующийся следующим образом:

- 2XX — команда успешно выполнена
- 3XX — ожидаются дополнительные данные от клиента
- 4XX — временная ошибка, клиент должен произвести следующую попытку через некоторое время
- 5XX — неустранимая ошибка

Текстовая часть ответа носит справочный характер. Установление и закрытие соединения непосредственно с сервером производится по TCP.

Команда	Описание	Положительный ответ сервера
HELO <домен>	Передает серверу домен отправителя.	250
AUTH LOGIN	Авторизация на сервере. Сервер проверяет, зарегистрирован ли пользователь.	334 – ответ на команду. Далее клиент должен послать закодированный логин. 334 – ответ на логин. Далее клиент должен послать закодированный пароль. 235 – авторизация пройдена успешно.
MAIL FROM <адрес>	Передает серверу адрес отправителя.	250
RCPT TO <адрес>	Передает серверу адрес получателя.	250
DATA <текс письма>	Передается письмо, со всеми заголовками и полями. Конец письма – точка в пустой строке.	354 – после отправки DATA. 250 – после отправки письма.
QUIT	Разрыв соединения.	221

3. Архитектура приложения на Java

Приложение реализовано в виде консольного приложения. В начале работы приложение требует ввести: доменное имя SMTP-сервера, собственное доменное имя, логин и пароль – данные необходимые для соединения и данные для формирования письма - поля, для формирования заголовков письма (от кого, кому, копия, скрытая копия, тема письма) и самого письма. В процессе работы отображаются посланные клиентом команды и ответы сервера.

Данные введенные пользователем используются как входные параметры для функций, отвечающие за отправку команд протокола SMTP:

- доменное имя SMTP-сервера для соединения с сервером;
- собственное доменное имя для команды HELO;
- ***login*** и ***password*** для команды AUTH LOGIN;

остальные для команды DATA (***From*** и ***To*** используются так же для команд MAIL FROM и RCPT TO соответственно).

В каждой функции осуществляется посылка команды серверу и проверка ответа от него. Если полученный ответ не соответствует положительному, то происходит вывод сообщения, которое уведомляет пользователя об ошибке.

При запуске приложения и ввода данных происходит соединение с сервером и последовательная посылка команд HELO и AUTH LOGIN.

При отправки письма происходит последовательная посылка команд MAIL TO, RCPT TO и DATA.

При передачи письма формируются следующие поля:

- **From** –отправитель;
- **To** – получатель;
- **Subject** – тема письма;
- **Content-Type: text/plain; charset=ISO-8859-1** – текстовое сообщение;
- **Content-Transfer-Encoding: base64** – кодировка письма;
- **Body** – текст письма.

По окончанию работы происходит разрыв соединения, посредством посылки команды QUIT.

4. Тестирование приложения

4.1.Отправка письма на почтовый ящик mail.ru

Smtp.mail.ru

Результат работы:

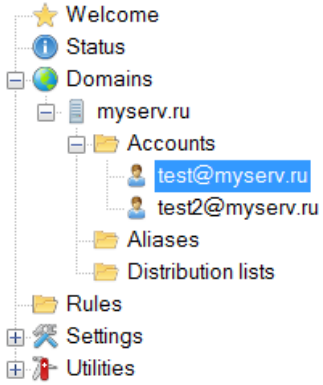
```
SMTP-клиент

>Введите домен SMTP-сервера:
smtp.mail.ru
220 smtp29.i.mail.ru ESMTP ready (Looking for Mail for your domain? Visit https://biz.mail.ru)
Connect

220 smtp43.i.mail.ru ESMTP ready (Looking for Mail for your domain? Visit https://biz.mail.ru)
Connect
```

>Введите ваш домен:
250 smtp43.i.mail.ru
HELO

>Введите ваш login:
muravevfedor
>Введите ваш password:
****
500 AUTH available only with SSL or TLS

Как видно из результатов работы smtp.mail.ru требует защищенного подключения SSL или TLS. В данном приложении была попытка реализовать данное защищенное подключение, но к успеху она не привела.

В связи с полученными результатами тестирование будет проводится на локальном сервере: hMailServer.



Был создан локальный домен myserv.ru. На сервере два аккаунта test@myserv.ru и test2@myserv.ru.

Попробуем отправить письмо с аккаунта test@myserv.ru на аккаунт test2@myserv.ru:

SMTP-клиент

>Введите домен SMTP-сервера:
0.0.0.0
220 myserv.ru ESMTP
Connect

>Введите ваш домен:
0.0.0.0
250 Hello.
HELO

>Введите ваш login:
test@myserv.ru
>Введите ваш password:
a044611z
334 VXNlcm5hbWU6
334 UGFzc3dvcmQ6
235 authenticated.
>Введите адрес отправителя:
test@myserv.ru

250 OK
MAIL FROM

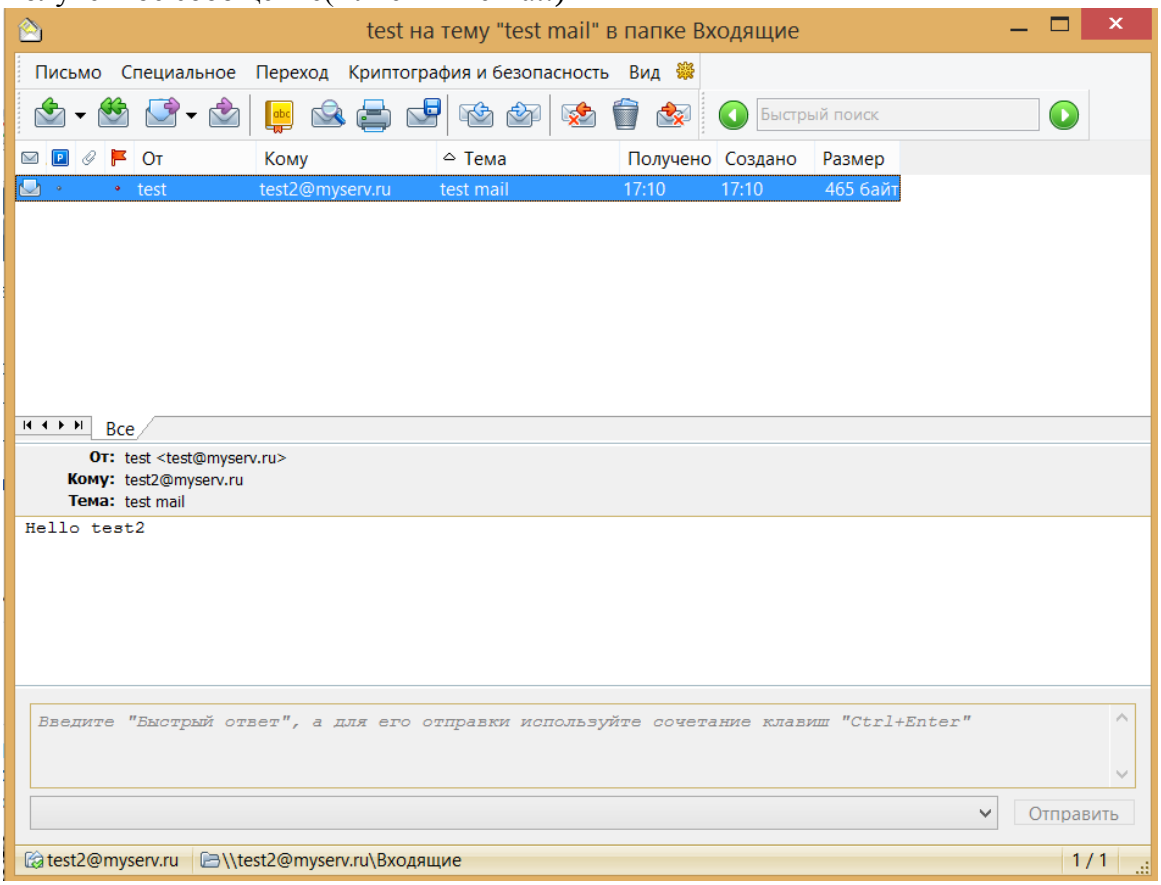
>Введите адрес получателя:
test2@myserv.ru
250 OK
RCPT TO

>Введите тему письма:
test mail
>Введите письмо:
Hello test2
354 OK, send.

LOG сервера:

Type	Session	Thread	Time	IP	Text
SMTPD	6	2848	2016-05-02...	127.0...	SENT: 220 myserv.ru ESMTP
SMTPD	6	2848	2016-05-02...	127.0...	RECEIVED: EHLO localhost
SMTPD	6	2848	2016-05-02...	127.0...	SENT: 250-myserv.ru[nl]250-SIZE 2...
SMTPD	6	2848	2016-05-02...	127.0...	RECEIVED: AUTH LOGIN
SMTPD	6	2848	2016-05-02...	127.0...	SENT: 334 VXNlcm5hbWU6
SMTPD	6	2848	2016-05-02...	127.0...	RECEIVED: dGVzdEBteXNlcnYucnU=
SMTPD	6	2848	2016-05-02...	127.0...	SENT: 334 UGFzc3dvcmQ6
SMTPD	6	2848	2016-05-02...	127.0...	RECEIVED: ***
SMTPD	6	2848	2016-05-02...	127.0...	SENT: 235 authenticated.
SMTPD	6	2848	2016-05-02...	127.0...	RECEIVED: MAIL FROM:<test@myse...
SMTPD	6	2848	2016-05-02...	127.0...	SENT: 250 OK
SMTPD	6	2844	2016-05-02...	127.0...	RECEIVED: RCPT TO:<test2@myser...
SMTPD	6	2844	2016-05-02...	127.0...	SENT: 250 OK
SMTPD	6	2848	2016-05-02...	127.0...	RECEIVED: DATA
SMTPD	6	2848	2016-05-02...	127.0...	SENT: 354 OK, send.
SMTPD	6	2700	2016-05-02...	127.0...	SENT: 250 Queued (0.015 seconds)
SMTPD	6	2844	2016-05-02...	127.0...	RECEIVED: RSET
SMTPD	6	2844	2016-05-02...	127.0...	SENT: 250 OK
SMTPD	6	2848	2016-05-02...	127.0...	RECEIVED: QUIT
SMTPD	6	2848	2016-05-02...	127.0...	SENT: 221 goodbye

Полученное сообщение(Клиент The Bat!)



Код полученного письма:

```
Return-Path: test@myserv.ru
Received: from localhost (Acer [127.0.0.1])
      by myserv.ru with ESMTPA
      ; Mon, 2 May 2016 17:10:54 +0300
Date: Mon, 2 May 2016 17:10:54 +0300
From: test <test@myserv.ru>
X-Priority: 3 (Normal)
Message-ID: <694287881.20160502171054@myserv.ru>
To: test2@myserv.ru
Subject: test mail
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-EsetId: 37303A296EAC7A6F667664

Hello test2
```

Из листинга видно, что письмо сформировано верно и имеет все необходимые заголовки.

5. Выводы

Анализируя результаты работы можно констатировать, что написание SMTP-клиента не является большой проблемой. По сути, нужно лишь отправлять команды, получать ответы и правильно их интерпретировать.

Проблема заключается в том, что не на всех серверах можно аутентифицироваться с помощью команды AUTH LOGIN. Например на gmail.com используется множество механизмов аутентификации (CRAM-MD5 и т.д.), для которых необходимо реализовывать определенную хэш-функцию. Это значительно осложняет разработку приложения. Также все SMTP-сервера используют защищенное TLS SSL соединение, с его созданием связаны определенные проблемы. Поэтому тестирование клиента проводилось на локальном почтовом сервере. Все цели и необходимые навыки по выполнению данной лабораторной работы были получены.

К достоинствам SMTP протокола можно отнести довольно большое количество настраиваемых заголовков для формирования письма.

ПРИЛОЖЕНИЕ 1

Main.java

```
package smtpclient;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.security.NoSuchAlgorithmException;
import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;

public class Main {
   public static void main(String[] args) throws IOException,
NoSuchAlgorithmException{

      System.out.println("SMTP-клиент\n");
      BufferedReader inu;
      inu  = new BufferedReader(new InputStreamReader(System.in));
      String fuser=null, log=null, passwd=null, from=null, to=null, subject=null;
      SMTPClient smtp = new SMTPClient();

      while(true){
        System.out.println(">Введите домен SMTP-сервера:");
        if ((fuser = inu.readLine())!=null){
           smtp.Connect(fuser);
          // smtp.Connect("smtp.mail.ru");
        }
```

```
        System.out.println(">Введите ваш домен:");
        if ((fuser = inu.readLine())!=null){
           smtp.Helo(fuser);}
           //smtp.Helo("localhost");

        System.out.println(">Введите ваш login:");
        if ((fuser = inu.readLine())!=null){
           log=fuser;
           System.out.println(">Введите ваш password:");
           if ((fuser = inu.readLine())!=null){
              passwd=fuser;
              smtp.Auth(log, passwd);
           }
        }

        System.out.println(">Введите адрес отправителя:");
        if ((fuser = inu.readLine())!=null){
           from=fuser;
           smtp.Mailfrom(fuser);
        }

        System.out.println(">Введите адрес получателя:");
        if ((fuser = inu.readLine())!=null){
           to=fuser;
           smtp.Rcptto(fuser);
        }

        System.out.println(">Введите тему письма:");
        if ((fuser = inu.readLine())!=null){
           subject=fuser;
           System.out.println(">Введите письмо:");
           if ((fuser = inu.readLine())!=null){
           smtp.Data(from, to, subject, fuser);
           }
        }
        System.out.println("****");
        smtp.Quit();
     }
     }
}
```

SMTPClient.java

```
package smtpclient;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;
import java.security.*;
import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;
import javax.net.SocketFactory;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.SSLSocket;
import java.net.*;

public class SMTPClient {

    private Socket clientsocket = null;
    private PrintWriter out = null;
    private BufferedReader in = null;

    // Соединение с SMTP-сервером
    public void Connect(String host) throws UnknownHostException, IOException{

        String reply = null;
        try{

            clientsocket = new Socket(host, 25);

            in  = new BufferedReader(new InputStreamReader(clientsocket.getInputStream()));
            out = new PrintWriter(clientsocket.getOutputStream(),true);
            reply = Reply();
            System.out.println(reply);

            if(reply==null){
                System.out.println("Error: Connect!");
                System.exit(-1);
            }
            if (reply.substring(0, 3).equals("220")){
                System.out.println("Connect\n");
            }
            else System.out.println("Error Connect");
```

```
        }catch(IOException ex) {
            System.out.println("Error Connect");
            System.exit(-1);
         }

}

// Передача серверу информации о домене пользователя
public void Helo(String domainUser){
    String reply;
    try{
        reply=SendCommand("HELO " + domainUser);
        System.out.println(reply);
        if(reply.substring(0, 3).equals("250")){
            System.out.println("HELO\n");
        }
        else System.exit(-1);
    }catch(IOException ex) {
        System.out.println("Error  HELO\n");
        System.exit(-1);
    }
}

// Аутентификация
public void Auth(String login, String password) {
    String reply, cod;
    try{
        reply=SendCommand("AUTH LOGIN");
        System.out.println(reply);
        if(reply.substring(0, 3).equals("334")){
            //cod="boytsev.andrey@rambler.ru";
            cod= new BASE64Encoder().encode(login.getBytes());
            reply=SendCommand(cod);
            System.out.println(reply);
            if(!reply.substring(0, 3).equals("334")){
                System.exit(-1);
            }
            //cod="i1r9i9n4a";
            cod= new BASE64Encoder().encode(password.getBytes());
            reply=SendCommand(cod);
            System.out.println(reply);
            if(!reply.substring(0, 3).equals("235")){
                System.exit(-1);
            }
        }
        else System.exit(-1);
```

```
    }catch(IOException ex) {
        System.out.println("Error\n");
        System.exit(-1);
    }
}

// Передача серверу адреса отправителя письма
public void Mailfrom(String mailclient){
    String reply;
    try{
        reply=SendCommand("MAIL FROM: " + mailclient);
        System.out.println(reply);
        if(reply.substring(0, 3).equals("250")){
            System.out.println("MAIL FROM\n");
        }
        else{
            System.out.println("Error  MAIL FROM\n");
        }
    }catch(IOException ex) {
        System.out.println("Error\n");
        System.exit(-1);
    }
}

// Передача серверу адреса получателя письма
public void Rcptto(String maildest){
    String reply;
    try{
        reply=SendCommand("RCPT TO: " + maildest);
        System.out.println(reply);
        if(reply.substring(0, 3).equals("250")){
            System.out.println("RCPT TO\n");
        }
        else{
            System.out.println("Error  RCPT TO\n");
        }
    }catch(IOException ex) {
        System.out.println("Error  RCPT TO\n");
        System.exit(-1);
    }
}

// Передача серверу тела письма
public void Data(String from, String to, String subject, String dat){
    String reply;
    String msg;
```

```
        try{
            msg="From:  "+from+"\nTo:  "+to+"\nSubject:  "+subject+"\nContent-Type:  text/plain;
charset=ISO-8859-1\nContent-Transfer-Encoding: base64\n"+dat+"\n\n.";
            //System.out.println(msg);
            reply=SendCommand("DATA ");
            System.out.println(reply);
            if(reply.substring(0, 3).equals("354")){
                reply=SendCommand(msg);
                if(reply.substring(0, 3).equals("250")){
                    System.out.println(reply);
                }
            }
            else{
                System.out.println("Error  DATA\n");
            }
        }catch(IOException ex) {
            System.out.println("Error\n");
            System.exit(-1);
        }
    }

    // Завершение сеанса связи
    public void Quit() {
        String reply;
        try{
            reply=SendCommand("QUIT");
            System.out.println(reply);
            if(reply.substring(0, 3).equals("221")){
                System.out.println("QUIT\n");
                in.close();
                out.close();
                clientsocket.close();
            }
            else{
                System.out.println("Error  QUIT");
            }
        }catch(IOException ex) {
            System.out.println("Error");
            System.exit(-1);
        }
    }

    //Чтение ответа от SMTP-сервера
    public String Reply() {
        String reply = null;
        try {
```

```
            reply = in.readLine();
        } catch (IOException e) {
            System.out.println("System Error: read!!");
            System.exit(-1);
        } finally {
            return reply;
        }
    }

    //Посылка команды ПОП3-серверу
    public String SendCommand(String command) throws IOException {
        out.println(command);
        return Reply(); // ответ сервера на команду
    }



}
```