Universitatea Tehnica a Moldovei
Facultatea Calculatoare Informatică și Microelectronică

Departamentul ISA

Raport

Lucrarea de laborator Nr. 4
la Programarea în rețea

Tema: Protocoalele poștei electronice. Proiectare și programare aplicație client

A elaborat st. gr. TI-144
D. Gorduz
A verificat lect. sup.
S. Ostapenco

Chișinău 2017

Obiective

Înțelegerea sistemului de poștă electronica și studiul acestuia în baza a două protocoale populare SMTP și POP3. Obiectivul specific constând în elaborarea unei aplicații de poștă electronica pentru trimiterea de mesaje electronice și citirea acestora din cutia poștală a utilizatorului.

Scopul lucrării

Lucrarea de laborator prevede crearea unui client simplu de poștă electronică care să trimită și să citească mesaje prin intermediul unui cont de poștă electronică.

Noțiuni teoretice

Simple Mail Transfer Protocol (prescurtat, SMTP; în traducere aproximativă Protocolul simplu de transfer al corespondenței) este un protocol simplu, folosit pentru transmiterea mesajelor în format electronic pe Internet. SMTP folosește portul de aplicație 25 TCP și determină adresa unui server SMTP pe baza înregistrării MX (Mail eXchange, „schimb de corespodență") din configurația serverului DNS.[1]

POP, sau *Post Office Protocol*, este o modalitate de a obține emailul din vremurile de început ale Internetului. Calculatoarele aveau acces de bandă mică (mult mai mică decât avem azi pe smartphone-uri) la Internet. De aceea a fost creat POP, ca un demers care să permită descărcarea locală a mesajelor pentru citirea lor în mod offline (deconectat de la Internet adică) și ștergerea lor de pe server la descărcare. Prima versiune POP a fost creată în 1984 și a avut o revizie în (POP2) în 1985. POP3 este versiunea curentă a acestui protocol de email și acest protocol este încă utilizat destul de mult.[2]

Mersul lucrării

În cadrul acestei lucrări de laborator a fost utilizată conexiunea prin socket, utilizând limbajul Java. Pentru a realiza conexiunea SMTP, a fost mimată conexiunea telnet cu hostul dorit. În cazul de față a fost utilizat serviciul existent: www.mail.com. Pentru a ne putea folosi de protocolul SMTP, avem nevoie să cunoaștem hostul responsabil de această conexiunea, dar și portul prin care ne putem adresa la el. Pentru a-l găsi, după un simplu search pe google, găsim informația necesară, link-ul fiind anexat în bibliografie [3].

În figura 1, este realizată conexiunea prin crearea unui obiect de tip Socket, în care specificăm hostul, dar și portul care va fi conectat. În cazul dat avem hostul: smtp.mail.com, iar portul 587 (Acesta este portul implicit de trimitere a mesajelor. Atunci când un client sau server de poștă electronică trimite un e-mail care urmează să fie redirecționat de către un server de mail adecvat, acesta ar trebui să utilizeze întotdeauna acest port. Acest port, împreună cu criptarea TLS, va asigura că e-mailurile sunt trimise în siguranță și că respectă liniile directoare stabilite de IETF). Pentru a putea citi și înscrie în conexiune de tip TCP, va fi nevoie să folosim un inputStream, care primește setul de bytes din conexiunea creată, iar pentru a citi un outputStream. Ele vor fi incluse într-un *BufferedReader* și într-un *PrintWriter*, care ne vor facilita citirea eficientă a caracterelor și respectiv scrierea (transmiterea) caracterelor.

Deci ca și în conexiunea realizată prin *telnetm* primul mesaj transmis către host va fi *HELO* sau *EHLO*. Mail.com nu permite transmiterea mesajului *HELO*, din această cauză vom utiliza comanda *EHLO*, care este varianta mai nouă a lui *HELO*, dar care returnează și comenzile pe care le suportă clientul. De notat că după comanda EHLO, ar trebui să urmeze un identificator. În cazul de față acesta a fost numele de pe localhost.

```
if (smtp == null) {
    label.setText("Connection not established!");
    label.setVisible(true);
    return;
}
InputStream inputStream = smtp.getInputStream();
OutputStream outputStream = smtp.getOutputStream();
BufferedReader in = new BufferedReader(new InputStreamReader(inputStream));
PrintWriter out = new PrintWriter(new OutputStreamWriter(outputStream),  autoFlush: true);

if (inputStream == null || outputStream == null) {
    label.setText("Failed to open streams!");
    label.setVisible(true);
    return;
}
// Get the response
String response = in.readLine();
System.out.println(response);

// Send EHLO and get the response
System.out.println("EHLO " + local.getHostName());
out.println("EHLO " + local.getHostName());
for (int i = 0; i < 4; i++) {
    response = in.readLine();
    System.out.println(response);
}
```

Figura 1 – Conexiunea prin Socket la host

În figura 2, este prezentată continuarea comenzilor transmise și răspunsurilor primite. Prima comandă transmisă ar fi AUTH LOGIN, care cere serverului autentificarea. Răspunsul primit va fi encodat în Base64, și cu ajutorului mecanismului inclus în Java, cu ușurință se poate decoda răspunsul, care defapt ar fi numele utilizatorului. De notat este că și transmiterea numelui utilizatorului trebuie să fie encodadă în Base64. Acest lucru este realizat la fel cu ajutorul mecanismului inclus în Java de encodare. Introducerea parolei are loc, exact la fel.

Pentru transmiterea unui mesaj se folosește comanda MAIL FROM: *nume*, unde se specifică de la cine este mesajul. Pentru alegerea recipientului se utilizează comanda RCPT TO: *nume*, unde nume este emailul la persoana dorită de a transmite mesaj. Prin comanda DATA specificăm mesajul dorit. De notat este faptul că mesajul trebuie să se termine cu „ . ” (punct) din rând nou. După răspunsul primit putem observa dacă mesajul a fost transmis, sau nu. Pentru a închide conexiunea se transmite comanda QUIT.

```
// Send auth login and get the response
System.out.println("AUTH LOGIN");
out.println("AUTH LOGIN");
response = in.readLine();
System.out.println(new String(Base64.getDecoder().decode(response.substring(4))) + " " + username.getText());
out.println(new String(Base64.getEncoder().encode(username.getText().getBytes())));
response = in.readLine();
System.out.println(new String(Base64.getDecoder().decode(response.substring(4))) + " ***");
out.println(new String(Base64.getEncoder().encode(password.getText().getBytes())));
response = in.readLine();
System.out.println(response);

// Send email
System.out.println("MAIL FROM: " + username.getText());
out.println("MAIL FROM:" + username.getText());
response = in.readLine();
System.out.println(response);
System.out.println("RCPT TO: " + to.getText());
out.println("RCPT TO:<" + to.getText() + ">");
response = in.readLine();
System.out.println(response);
System.out.println("DATA");
out.println("DATA");
out.println("From: " + username.getText());
out.println("Subject: " + subject.getText());
out.println(message.getText());
System.out.println(".");
out.println(".");
```

Figura 2 – Transmiterea unui mesaj

Logica pentru citirea mesajelor rămâne aceiași, doar că se schimbă hostul, care pentru utilizarea protocolului POP3, ar fi pop.mail.com, conexiunea realizându-se prin portul 110 (Acesta

este portul implicit POP3 non-criptat). Autentificarea are loc prin conexiune plain text, unde prima comanda USER *nume*, specifică numele utilizatorului, și PASS *parola* specifică parola utilizatorului.

Din răspuns putem vedea dacă serverul a acceptat credențialele sau nu. Această succesiune de comenzi este specificată în figura 3.

```
InetAddress host = InetAddress.getByName("pop.mail.com");
Socket smtp = new Socket(host,  port: 110);

if (smtp == null) {
    label.setText("Connection not established!");
    label.setVisible(true);
    return;
}
InputStream inputStream = smtp.getInputStream();
OutputStream outputStream = smtp.getOutputStream();
BufferedReader in = new BufferedReader(new InputStreamReader(inputStream));
PrintWriter out = new PrintWriter(new OutputStreamWriter(outputStream),  autoFlush: true);

if (inputStream == null || outputStream == null) {
    label.setText("Failed to open streams!");
    label.setVisible(true);
    return;
}
String response = in.readLine();
System.out.println(response);
```

Figura 3 – Autentificarea în POP3

Pentru a vedea numărul de mesaje în Inbox se transmite comanda STAT. Astfel toate mesajele vor fi înscrise într-un StringBuilder, rând după rând. Astfel când linia are doar „ . ”, va specifica ca aceste este sfârșitul mesajului, și că ciclul de înscriere se va opri. Mesajele vor fi înscrise respectiv într-un array de obiecte de tip StringBuilder.

De notat este faptul că pentru returnarea mesajului se utilizează comanda RETR *index,* unde index este numărul mesajului din inbox. Pentru a închide conexiunea, la fel se utilizează comanda QUIT.

```
out.println("STAT");
response = in.readLine();
System.out.println(response);
String number = response.substring(4, 6);
int n = Integer.parseInt(number);
System.out.println(n);
StringBuilder[] sb = new StringBuilder[n + 1];
String line;
outer:
for (int i = 1; i <= n; i++) {
    out.println("RETR " + i);
    System.out.println("RETR " + i);
    sb[i] = new StringBuilder();
    while ((line = in.readLine()) != null) {
        sb[i].append(line).append("\n");
        if (line.equals(".")) {
            continue outer;
        }
    }
}

out.println("QUIT");
System.out.println("QUIT");
```

Figura 4 – Citirea mesajelor POP3 și înscrierea

Pentru exemplificarea interacțiunii cu utilizatorul a fost creată o interfață grafică, care va fi prezentată în figura 5 și 6.

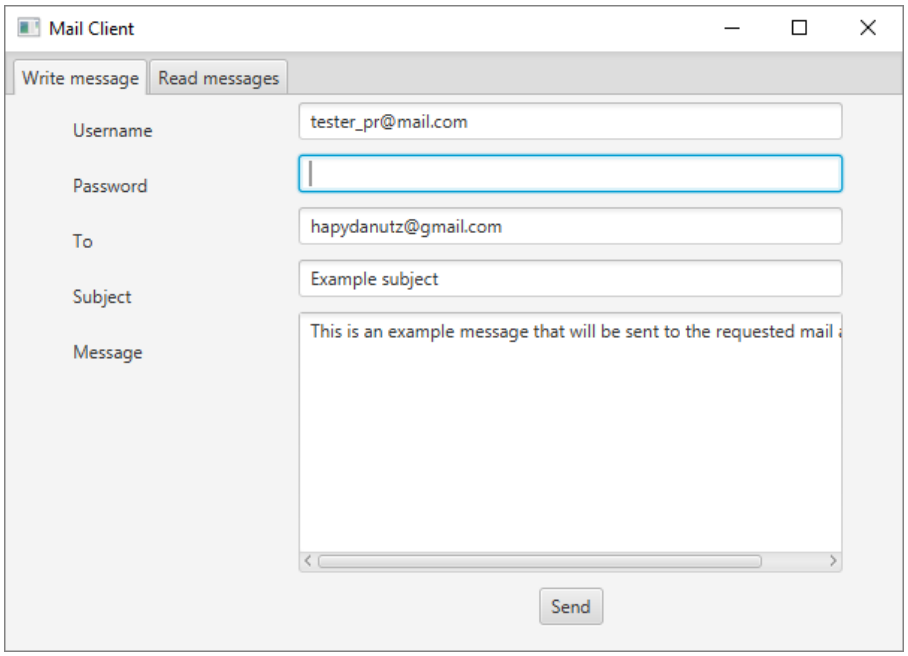

Figura 5 – Screierea unui mesaj

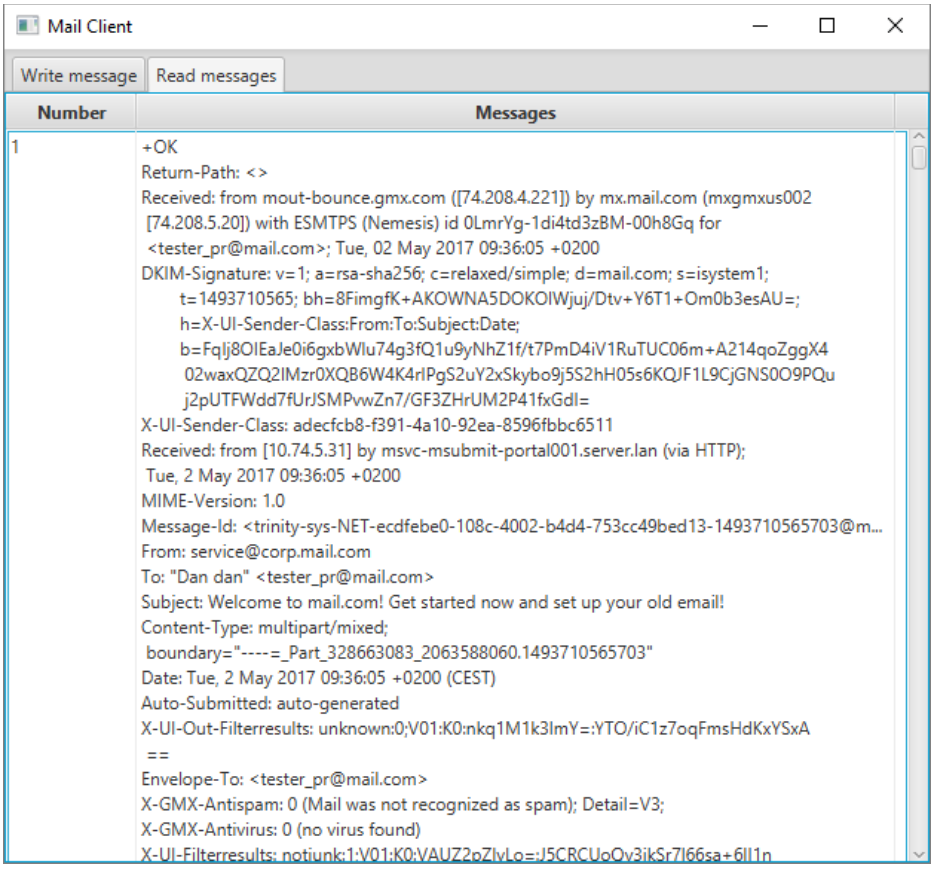

Figura 6 – Citirea mesajelor

Concluzie

În urma acestei lucrări de laborator a avut loc înțelegerea de bază a sistemului de poștă electronică. Au fost analizate și s-a lucrat cu două protocoale populare SMTP și POP3, la un nivel destul de jos, la nivelul socket-urilor. A fost obținută experinență în elaborarea unei aplicații de poștă electronică pentru trimiterea de mesaje electronice, cât și pentru citirea acestora din cutia poștală a utilizatorului.

Au fost studiate și utilizate instrumentele limbajului Java, care oferă o implimentare clară și citibilă. A fost utilizată și au fost puse bazele conectării client-server prin intermediul comenzilor telnet pentru SMTP și POP3, cât și interacțiunea lor în paralel cu o interfață grafică, cu ajutorul librăriei JavaFX.

Bibliografie

1. Răzvan Rughiniș, Răzvan Deaconescu, George Milescu, Mircea Bardac**.** Introducere în sisteme de operare, România, Editura Printech, Capitolul 9, pag. 265 [Text]
2. Windows Inserder RO [Resurs electronic]. Regim de acces: https://windowsinsider.ro/windows/ce-inseamna-pop-imap-si-exchange/
3. SMTP and POP3 E-mail Settings [Resurs electronic]. Regim de acces: http://www.smtp-pop3.com/mailcom-settings.html