**Ministerul Educaţiei al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Raport**

Lucrare de laborator **№3**

Disciplina: Programarea in Retea

A îndeplinit: st.gr.FI-161 Panfil Nicolae

A controlat: Gavrilita Mihai

Chișinău 2019

**Lucrare de Laborator Nr.3**

**Scop:**

**1. Studierea protocolului Http si utilizarea in Web communication;**

**2. Informarea despre Java components responsabile pentru implementarea Http methods;**

**3. Crearea unui client app care suporta principalele Http methods folosind**[**httpbin.org**](https://github.com/lfocsa/PR_Lab3/blob/master/httpbin.org)**;**

**Sarcina:**

Implementați un client HTTP care ar trebui să utilizeze metodele HTTP pentru a efectua requests.

Pentru a implementa aplicația am folosit Apache Maven - un instrument de automatizare al construirii, care mi-a dat de asemenea templateul proiectului. Fișierul care conduce structura proiectului, dependențele și alte bunuri este numit ***pom.xml***. În acest fișier la dependențe am inclus următorul bloc:

<dependency>

<groupId>org.apache.httpcomponents</groupId>

<artifactId>httpclient</artifactId>

<version>4.5.6</version>

</dependency>

În Java există mai multe moduri de a efectua cererile HTTP,am ales să folosesc ***HttpUrlConnection*** care face legătura dintre Browser și Server. Clasa UrlConnection are o mulțime de metode și funcții pentru realizarea funcționalității conexiunilor și a requestsurilor. Eu am clasa principală unde am USER\_AGENT și apelul pentru fiecare metodă.

public class HttpClientRequest {

private static final String USER\_AGENT = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.121 Safari/537.36";

private static final String GET\_URL = "https://httpbin.org/get";

private static final String POST\_URL = "https://httpbin.org/post";

private static final String DELETE\_URL = "https://httpbin.org/delete";

private static final String IMAGE\_URL = "https://httpbin.org/image";

public static void main(String[] args) throws Exception {

sendGET();

sendPost();

sendDelete();

getImage();

}

Deci, primele metode pe care le-am efectuat au fost GET, am stipulat acest lucru în con.setRequestMethod ("GET"). De asemenea, am setat proprietatea pentru User-Agent, pe care am menționat-o cu toate browserele posibile la și tipul de codificare gzip.

URL obj = new URL(GET\_URL);

HttpURLConnection con = (HttpURLConnection) obj.openConnection();

con.setRequestMethod("GET");

con.setRequestProperty("User-Agent", USER\_AGENT);

con.setRequestProperty("Accept-Encoding", "gzip");

Apoi, am capturat status code, care ajuta pentru a vedea dacă cererea a trecut sau nu, în conformitate cu codul:

* 200: succes.
* 405: metoda nu există.
* 404: eroare de server. Deci, dacă codul de răspuns este de 200, atunci am folosit Reader, Input / OutputStream pentru citirea și scrierea corpului de răspuns. Aici vreau să menționez diferența dintre Reader și InputStream, principala diferență fiind că cititorul este bazat pe caractere, dar InputStream este bazat pe octeți. Pentru citirea și afișarea răspunsului am folosit While(true), care se pare a fi o buclă infinită, dar în acest caz verificată dacă există caractere și dacă există, le afișeaza în aceeași formă (i.e parsing the response).

while (true) {

int ch = reader.read();

if (ch == -1) {

break;

}

System.out.print((char) ch);

}

Următoarea metodă pe care am implementat-o este POST, care a utilizat aprox. același cod, dar ceva ușor diferit. Aici apare noțiunea de setDoOutput(), care ar trebui să fie true pentru metoda POST, aceasta înseamnă că ar trebui să trimită response body serverului pentru efectuarea unor modificări. Dacă aceasta are opțiunea falsă, aceasta va face metoda GET. Am codificat un comentariu, acesta este pentru adăugarea unor câmpuri în body.

//String urlParameters = "mail=example@gmail.com&student=FAF"; here we can add some fields

//Send post request

con.setDoOutput(true);

DataOutputStream wr = new DataOutputStream(con.getOutputStream());

// wr.writeBytes(urlParameters); writing addtitional information

// wr.flush();

// wr.close();

În metoda DELETE acțiunile sunt aceleași ca în metoda POST.

Ultima metodă pe care am efectuat-o a fost pentru a obține o imagine de pe server. Aici am folosit un alt instrument din Java, care este Image și JFrame. Deci, pentru obținerea imaginii am folosit metoda GET. Cu biblioteca de imagini am citit imaginea și cu JFRame am setat parametrii pentru fereastra în care va fi afișată imaginea.

//add reuqest header

con.setRequestMethod("GET");

con.setRequestProperty("User-Agent", USER\_AGENT);

image = ImageIO.read(imageURL);

} catch (IOException e) {

e.printStackTrace();

}

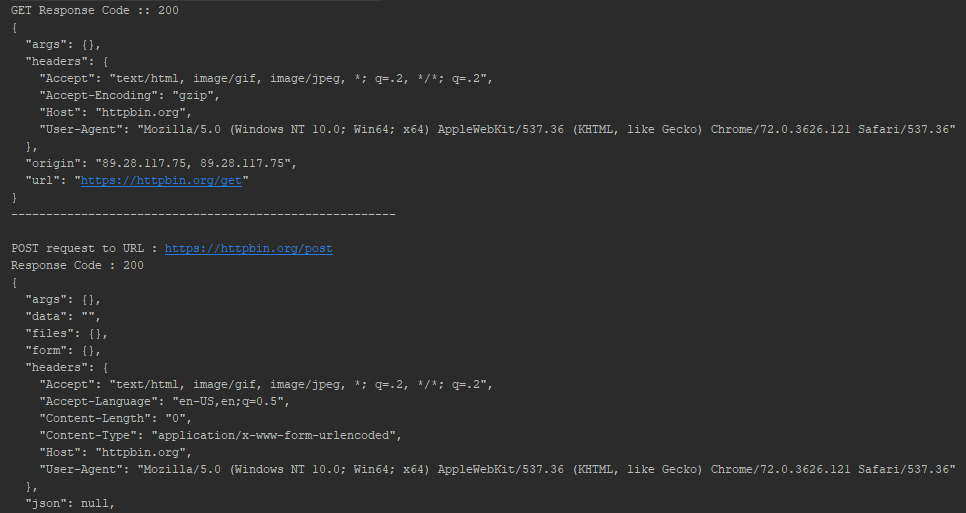
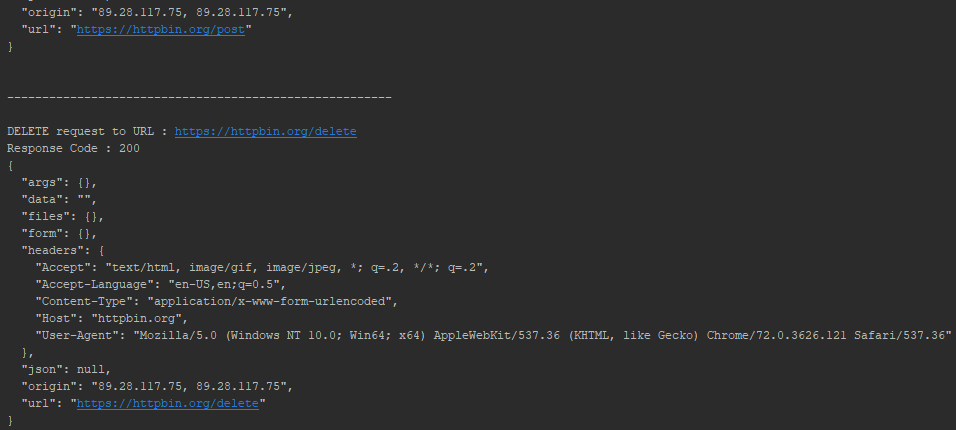
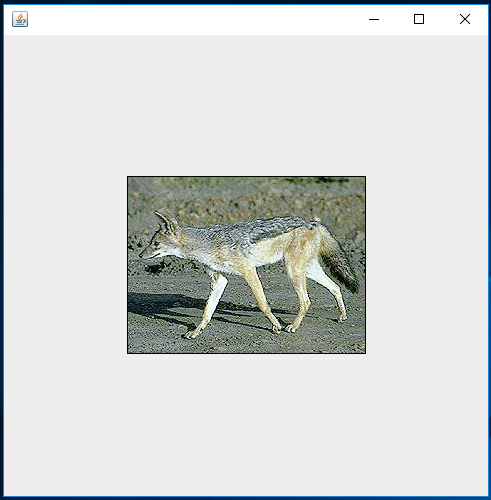
JFrame frame = new JFrame();

frame.setSize(500, 500);

JLabel label = new JLabel(new ImageIcon(image));

frame.add(label);

frame.setVisible(true);

**Console results :** [](https://user-images.githubusercontent.com/43058513/54092523-dbe64480-4395-11e9-9fd7-3bb9deb3ef55.PNG)[](https://user-images.githubusercontent.com/43058513/54092551-105a0080-4396-11e9-9bb7-9b0cabb97640.PNG)[](https://user-images.githubusercontent.com/43058513/54092562-3384b000-4396-11e9-84c9-e24560f042bb.PNG)