UNIVERSITATEA TEHNICĂ A MOLDOVEI

FACULTATEA CALCULATOARE, INFORMATICĂ ȘI MICROELECTRONICĂ

PROTOCOLUL HTTP

LUCRARE DE LABORATOR NR. 1

la disciplina ”Programarea în rețea”

Autor:

studentul gr. TI 141,

învățământ cu frecvența redusă

MIROVSCHI Eugen

Profesor:

Lector universitar,

ANTOHI Ionel

_____

(semnătura)

Chișinău, 2018

I. Scopul lucrării

Studierea protocolului HTTP și utilizarea acestuia pentru a accesa informația disponibilă în spațiul World Wide Web.

II. Sarcina lucrării

De dezvoltat o aplicație Java/C# care va utiliza metodele GET și POST la nivel de transfer pe protocolul HTTP.

Ca sursă de testare va fi utilizat resursa http://httpbin.org/:

1. GET – http://httpbin.org/get
2. POST – http://httpbin.org/post

În momentul adresării la pagina cu metoda GET, se va extrage conținutul din body și va fi afișat la ecran. Pentru metoda POST se va crea un mesaj care va fi expediat paginii respective, după care se va extrage conținutul din body și va fi afișat la ecran.

III. Efectuarea lucrării

Java oferă implicit o metodă de executare a cererilor HTTP folosind clasa HttpURLConnect. Această clasă se află în pachetul java.net și ne permite să comunicăm cu un server folosind protocolul HTTP fără a folosi careva biblioteci adiționale.

Dezavantajul folosirii acestei clase în comparație cu instrumentele oferite de alte biblioteci terțe este incomoditatea și necesitatea de a scrie mai mult cod pentru a obține același rezultat. Acest lucru poate fi văzut și ca un avantaj deoarece ne oferă posibilitatea de a efectua cereri la cel mai jos nivel.

Clasa HttpURLConnect este definită ca o clasă abstractă, ceea ce înseamnă ca nu poate fi instanțială în același mod ca alte obiecte. Pentru a crea o astfel de instanță trebuie să definim adresa resursei și să deschidem o conexiune către aceasta. Exemplu de cod pentru crearea unei conexiuni:

```
final URL url = new URL(request.getUrl());
final HttpURLConnection con = (HttpURLConnection) url.openConnection();
```

Următorul pas este definirea modul de cerere cum ar fi GET, POST, HEAD, OPTIONS, PUT, DELETE, TRACE etc. Acest lucru este specificat folosind metoda setRequestMethod, iar tipul cererii este trimis ca parametru:

```
con.setRequestMethod(request.getMethod().toString());
```

Acești pași pot fi suficienți pentru apelarea unei resurse, însă de obicei vom trimite și alte date cum ar fi antetul cererii sau un mesaj (body). Câmpurile din antet sunt adăugate folosind metoda setRequestProperty, iar mesajul este populat prin intermediul fluxului de ieșire a cărui referință poate fi obținută folosind metoda getOutputStream. Un moment important ce necesită specificat este faptul că implicit, clasa HttpURLConnect ignoră acest mesaj. Pentru a modifica acest comportament trebui să schimbăm un flag folosind metoda setDoOutput(true). Exemplu de definire a antetului și a mesajului:

```
request.getHeaders().forEach(con::setRequestProperty);

if (request.getBody() != null && !request.getBody().isEmpty())
{
    con.setDoOutput(true);
    IOUtils.write(request.getBody(), con.getOutputStream());
}
```

În momentul când cererea este completă, conținând toate datele necesare, putem să o efectuăm folosind metoda connect. Rezultatul execuției este păstrat în alte câmpuri ale clasei cum ar fi getHeaderFields pentru antet și getInputStream pentru mesajul din răspuns. Exemplu de citire a răspunsului:

```
private Response createResponse(final HttpURLConnection con)
{
    final Map<String, String> headers = con.getHeaderFields().entrySet().stream()
            .collect(toMap(Map.Entry::getKey, entry ->
entry.getValue().stream().collect(joining(DELIMITER))));

    return new Response(headers, readResponseBody(con));
}

private String readResponseBody(final HttpURLConnection con)
{
    try
    {
        return IOUtils.toString(con.getInputStream());
    }
    catch (Throwable throwable)
    {
        return "";
    }
}
```

Exemplu de execuție a algoritmului specificat mai sus poate fi văzut în figurile 1 și 2.

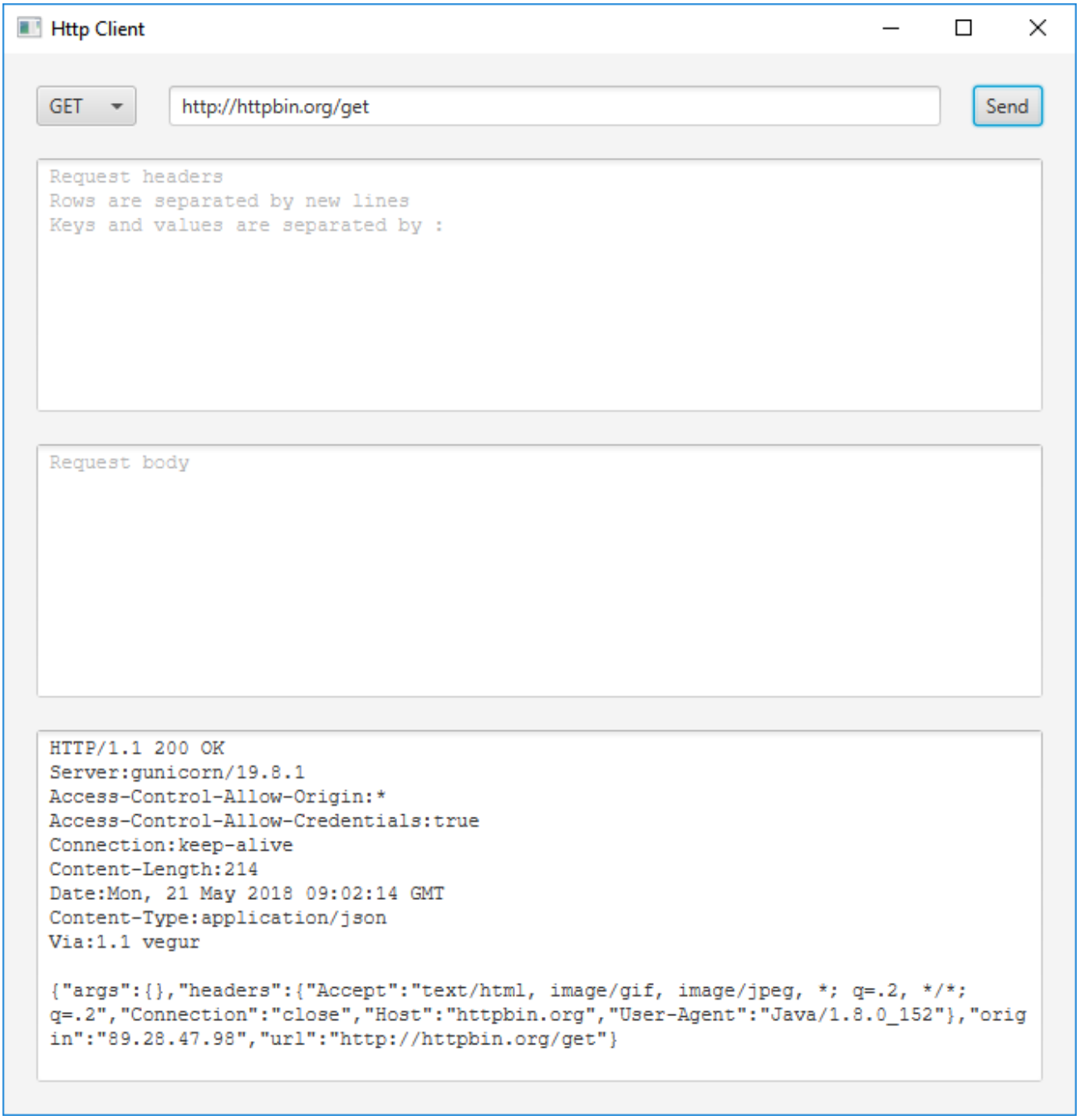

Figura 1 – Exemplu de cerere folosind metoda GET

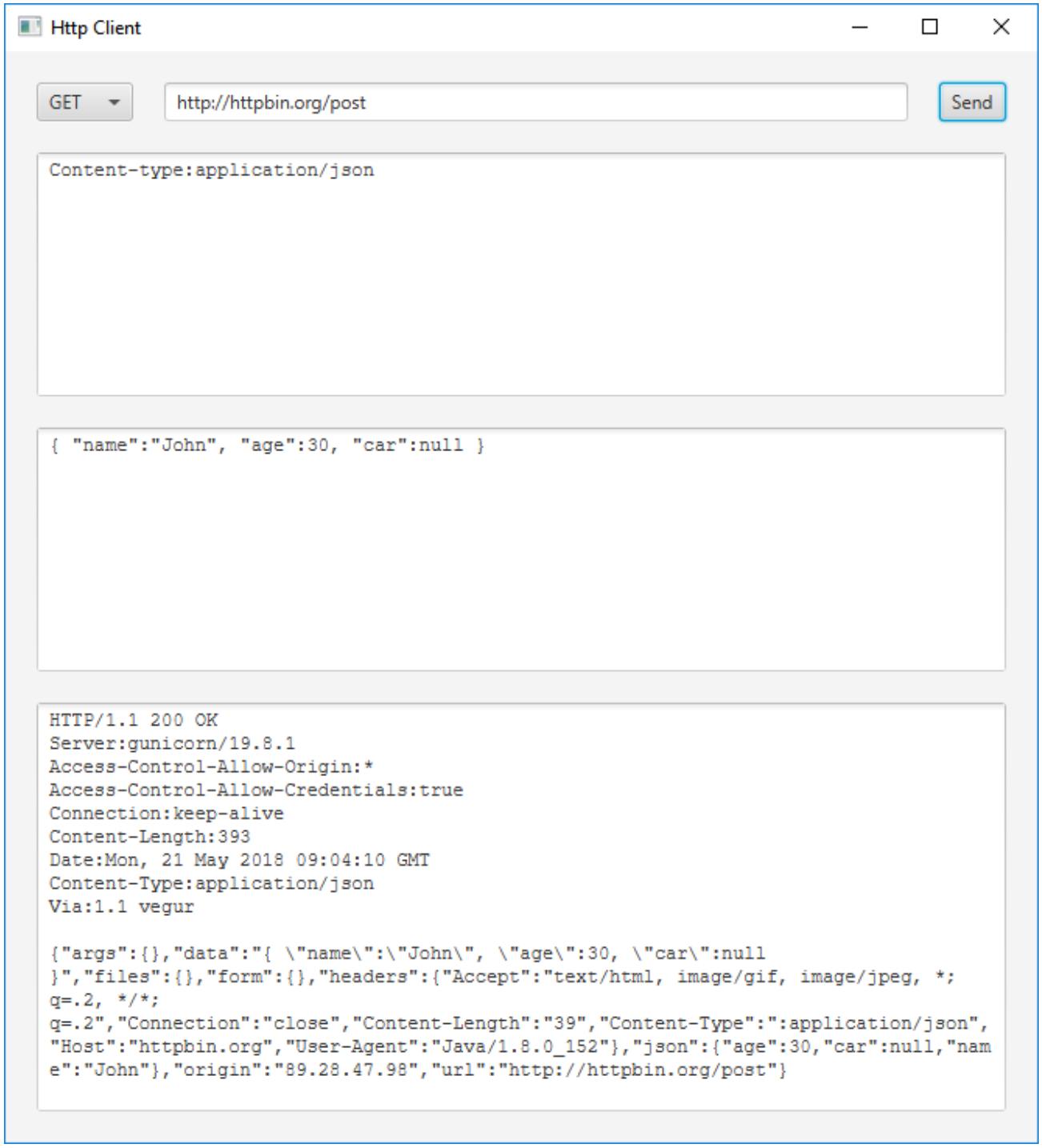

Figura 2 – Exemplu de cerere folosind metoda POST

IV. Concluzii

Efectuând această lucrare de laborator am studiat conceptele de bază a protocolului HTTP și modul de utilizare pentru a comunica cu o resursă de pe un server din spațiul World Wide Web.

Platforma Java oferă implicit instrumente necesare pentru crearea unei cereri HTTP. Deși acestea sunt un pic incomode comparativ cu alternativele oferite de alte biblioteci, aceasta ne oferă control deplin asupra datelor care sunt transmise și primite.

Pentru a crea o cerere avem nevoie de următoarele componente obligatorii: adresa resursei care încercăm să o accesăm și metoda de apelare cum ar fi GET sau POST. Adresa este reprezentată printr-un URL care este bine definit și standardizat. Metoda GET este de obicei folosită pentru a citi datele de pe un server, pe când metoda POST se folosește cu scopul de a trimite un mesaj către server. Există și alte metode definite în protocolul HTTP, iar fiecare dintre acestea au un scop specific și bine definit.

Pe lângă câmpurile de mai sus, o cerere poate conține și altă informație opțională cum ar fi antetul, mesajul, cookies etc. Antetul conține o serie de câmpuri care descrie cererea, cum ar fi dimensiunea și tipul mesajului, denumirea și versiunea clientului etc. Mesajul poate fi orice șir de biți, însă trebuie să corespundă tipului specificat în antet, pentru ca acesta sa fie prelucrat cu succes de către server.

După efectuarea cererii, rezultatul acesteia este salvat în alte câmpuri adiționale a clasei HttpURLConnect. Printre datele obținute în răspuns putem vedea antetul primit care descrie răspunsul, precum și un mesaj.