Ministerul Educației, Culturii și Cercetării al Republicii Moldova

Facultatea Calculatoare, Informatică şi Microelectronică

Departamentul Ingineria Software şi Automatică

RAPORT

Lucrarea de laborator NR.2

Disciplina: Programarea în rețea

A efectuat: Lașco Daniil, TI - 171

A verificat: Buldumac Oleg, asistent universitar

Chişinău 2020

**Scopul lucrării :**

Să se creeze o aplicație client HTTP. Cererile HTTP să fie făcute prin proxy. Să se utilizeze expresiile regulate. Să se utilizeze firele de execuții și tehnici de sincronizare. Clientul trebuie să se poată autentifica pe resursă utilizînd cookies.

**Întrebări la apărarea laboratorului:**

1. **Cum este formatat corpul unei cereri HTTP pentru o cerere HTTP de tip POST ?**

Formatul unei cereri HTTP pentru o cerere de tip POST este formata într-un bloc de cîmpuri caracteristice HTTP ca Content-Type,Accept, Content-Length,Accept-Language, Host, și se trimite body, conform Content-Type format.

1. **De unde știe un client HTTP ce tip de conținut trimite serverul HTTP ?**

Antetele Content-Encoding,Content-Type permit serverului să descrie proprietățile pagini sau conținutului pe care o transmit.

1. **Cum decide un client dacă ar trebui să aibă încredere în certificatul unui server ?**

Decide atunci cînd cheia publică a clientului și datele atribuite sunt diferite. Deaorece cheia publică poate să fie atribuită oricărei, dar deaorece există cheia privată, la criptara în SHA-1, algoritmul va genera o altă cheie, și clientul va deosebi frauda, sau că vorbește cu altcineva dar cu cheia publică la fel.

1. **Care este problema principală cu certificatele autosemnate ?**

Problema principală a certificatelor autosemnate este că cheia privată poate să fie obținută, plus o altă problemă pricipală este că gestionarea diferitor tipuri de formate de cerificate

1. **Conexiunea persistentă HTTP – care sunt principalele beneficii ?**

Conexiunea persistentă este mai benefică deoarece permite stabilirea unui conexiuni TCP, și trimiterea de cereri adiționale, respectiv răspunsuri adiționale. HTTP 1.0 solicita ca la fiecare cerere să fie creata noi conexiune. Conexiunea persistentă permite crearea cererilor pipeline, adică trimiterea a 2 cereri concomitente înainte ca la 1 să fi sosit.

1. **Ce este negocierea conținutului în HTTP și cum are loc ?**

HTTP este procesul de selectare a celei mai bune reprezentări pentru un răspuns dat atunci când există mai multe reprezentări disponibile.

Cînd un client dorește să obțină o resursă, el o face printr-un URL. Clientul trimite o cerere specific și modul în care se allege este legat de negocierea conținutului.

Un mecanism este prin anteturi HTTP specifice clientului (negociere bazată pe server).  
*Content-Type, Content-Lentght,Accept*

1. **Care sunt tipurile de negociere a conținutului HTTP ?**

### **Negociere bazată pe server** – clientul trimite request HTTP cu antete specifice, fiind folosite de server pentru a trimite ce-a mai apropiată resursă, de exemplu Accept-Encoding, Accept-Language, Accept-Charset

### **Negocierea bazată pe agenți** – Serverul trimite pagini care conțin legături către resursele alternative disponibile. Utilizator decide ce vrea să folosească. Se poate trimite din nou negocieri bazate pe server în baza celor gestionate de agenți.

1. **Ce este un ETag în HTTP și cum funcționează ?**

Etag este un identificator de pagină unic, care se referă la memoria ascunsă. El de obicei este indică către o pagina specifică într-o versiune. Funcționează la nivel de cerere HTTP, care presupune comparare ETag trimisă de un browser și server. Dacă se solicită acceași pagină se trimite empty body și se încarcă pagina din cache, dacă nu se generează alta cu alt ETag

1. **Diferența dintre protocoalele fără stare și cele cu stare. Cărui tip îi aparține HTTP ?**

Diferența dintre ele este că protocoale fară stare nu necesită reținerea informației sau sesiunii. O altă diferență este ca protocoale fără stare simplifică arhitectura serverului. O altă diferență este că protocoalele cu stare necesită mai puține resurse.

Protocoalele fără stare sunt protocoale simple folosite în comunicare Client-Server, și datele sau rezultate nu sunt salvate pe server, permițînd multe requeturi.

HTTP este un protocol fără stare, deoarece este utilizat la scară largă și menținrea de informații pe servere este practic imposibilă pentru rețele mari de comunicare.

1. **Avantajele cheie ale HTTP/2 în comparație cu HTTP/1.1**

* **Transfer de date mai rapid – HTTP/1.1 face mai multe cereri pentru a încărca simultan mai multe resurse. HTTP/2 transmite conexiuni concurente într-o singură conexiune TCP. Are loc principiul de programarea concurentă, se pot trimite mai multe și vin în ordine diferită**
* **Mai multa securitate -** HTTP/2 utilizeaza o conexiune criptată. Este necesar un certificat SSL pentru site cu scopul de a profita de HTTP/2. Diferența între ele este transferal de date criptat.
* **Comprimarea antetelor pentru a transfera un volum mai mic de date**

**Antele sunt trimise într-o cerere pentru a specifica serverului ce date, în ce format să fie returnate. HTTP/2 comprimă antetele pentru fiecare cerere, față de HTTP/1.1. Paginile Web sunt încarcate mai rapid.**

1. **Ce este un tip MIME, din ce constă și pentru ce se folosește ?**

MIME este este un standard de de mesaje care suportă reguile de codificare pentru mesajele non-ASCII. Constă din 5 antete :   
MIME-Version: este presupus ca fiind un mesaj în text pur  
Content-Description: este un şir de caractere ASCII specificând ce este în mesaj.  
Content-Id: identifică conţinutul  
Content-Transfer-Encoding: arată cum este împachetat pentru transmisie corpul mesajului  
Content-Type: specific tipul mesajului

1. **Care este diferența dintre GET și POST ?**

GET - Solicită date dintr-o resursă specificată (operație Citire).

Cererile GET pot fi memorate în cache

Cererile GET rămân în istoricul browserului

Cererile GET pot fi marcate

Cererile GET nu trebuie utilizate niciodată atunci când aveți de-a face cu date sensibile

Cererile GET au restricții de lungime

Cererile GET trebuie utilizate numai pentru a prelua date

Datele GET sunt afișate pe URL și sunt vizibile tuturor

POST - Trimite datele care urmează să fie procesate într-o resursă specificată (operație de scriere).

Cererile POST nu sunt niciodată în cache

Solicitările POST nu rămân în istoricul browserului

Cererile POST nu pot fi marcate

Solicitările POST nu au restricții privind lungimea datelor

Datele POST nu sunt afișate pe URL

1. **Care este diferența dintre PUT și POST ?**

Metoda PUT scrie o pagină. Această metodă permite crearea unei colecţii de pagini de Web pe un server la distanţă. Corpul cererii conţine pagina. Pagina poate să fie codificată utilizând MIME, caz în care liniile care urmează după PUT pot include Content-Type şi antete de autentificare, pentru a demonstra că într-adevăr cel care face cererea are dreptul de a realiza operaţia cerută.

POST conţine un URL, dar în loc să înlocuiască date existente, noile date se vor adăuga într-un mod generalizat.

1. **Care sunt metodele idempotente în HTTP și care sunt scopul lor.**

O metodă HTTP idempotentă este o metodă HTTP care poate fi numită de apelată de mai multe ori și rezultatul să fie același. Nu ar conta dacă metoda este numită o singură dată sau de zece ori peste. Rezultatul ar trebui să fie același. Scopul lor este de a tolera erorile. Dacă POST în unele cazuri nu poate să complecteze cerearea, atunci aceasta va ramîne nevalidata. Metodele idempotente permit ca acestea să fie realizate, trimițînd cererea pînă vom primi răspuns înapoi de la server.

METODELE IDEMPOTENTE SUNT : DELETE, GET, HEAD, OPTONS, PUT, TRACE.

1. **Cum sunt identificate resursele în protocolul HTTP ?**

Resursele în HTTP sunt indificate cu ajutorul URI trimise de-a lungul unei cereri HTTP. Avem URL(formatul URI). Constă din protocol (http://) , Authority (adresa site-ului ), Port(80 sau 443), Path-ul către resursă.

1. **Care sunt metodele sigure și nesigure în HTTP ?**

**Metodele sigure sunt :** GET, HEAD, OPTIONS , deaorece permit doar citirea, fără modficarea resursei. **Metodele nesigure sunt :** POST, DELETE, duc la modifcarea resursei.

1. **Pentru ce este nevoie de cURL ?**

cURL permite testarea și transferul de date folosind o varietate de protocoale. Permite folosirea în protocoale FTP, SMTP,POP3. Este un mod caracteristic a unui client de a folosi o modalitate, înafară de browser.

1. **Pentru ce este nevoie de HTTP Proxy?**

Pentru a îmbunătăţi performanţa este necesar de a salva paginile care au fost cerute pentru cazul în care ele vor fi utilizate din nou. Paginile pot fi păstrate pentru utilizări ulterioare în cache. Proxy va întreţine această memorie și o va accesa(Proxy pattern) .Dacă proxy-ul are pagina, o returnează imediat. Dacă nu, aduce pagina de la server, o adaugă în memoria ascunsă pentru utilizarea ulterioară şi o întoarce clientului care a cerut-o.

1. **Diferența dintre autentificare și autorizare**

Difereneța dintre autentificare și autorizare este că autentificarea identifică un client/user care are permisiunea de a accesa resursa. Autorizarea atribuie careva drepturi de acces a reusursei

1. **Care sunt metodele de autentificare HTTP ?**

Proxy Authentification  
Authentification Schema  
Cookie Authentification  
Client Credentials  
Token Authentification(Oauth Implicit flow)

1. **Modalități de identificare a utilizatorilor în HTTP**

**Cookie –** dacă cookie rămîne activ atunci se permite autentificarea pe o resursă. **Client Credentials –** Cereri POST cu credențialele unui user **HTTP Request Header –** Token de obicei, cîmpul Authorization

1. **HTTP cookies – pentru ce se folosește ?**

HTTP Cookie este un cîmp al protocolului HTTP. Cookie se populează cu date la autentificarea pe o resursă. Permite salvarea preferințelor utilizatorului și a credențialelor acestuia. Cookie este o modalitate de a și securizare unele date în Internet, salvîndule local pe un PC. Cookie presupune și urmărirea utilizatorului, preferințe, etc.

**Concluzie:**

În urma efectuării lucrări date de laborator, am făcut cunoștiință cu protocolul HTTP, cîmpurile acestuia, metodele caracteristice, și mecanismele de sincronizare. Am studiat de asemenea modalități de autentificare, autorizare folosind un HttpClient. Tot odată, am accentuat rolul expresilor regulate în folosirea acestora pentru filitrarea de conținut.