Ministerul Educației, Culturii și Cercetării Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică Departamentul Ingineria Software și Automatică

RAPORT

la Lucrarea de Laborator Nr. 3

Disciplina: Programarea în Rețea

TEMA: HTTP - Client

A elaborat: st. gr. TI-172, Movileanu Vladislav

A verificat: asistent universitar, Buldumac Oleg

Chișinău 2020

Sarcina lucrării:

Să se creeze o aplicație client HTTP. Să se utilizeze un ***proxy*** și ***expresiile regulate***.

Pentru nota 9 si 10:

- Să se utilizeze firele de execuții și tehnici de sincronizare.
- Clientul trebuie să se poată autentifica pe resursă utilizînd cookies.

Atenție:

- Pentru acest laborator utilizați librării HTTP deja existente, nu este necesar de a utiliza Sockets API. Cine dorește poate să facă prin socket ca și la primul laborator.
- Clientul trebuie să facă cereri ***GET***, ***POST***, ***HEAD*** și ***OPTIONS***.
- Aplicația poate fi GUI sau consolă.
- Nu sunteți limitați la funcțional, resura(**pagina web**) la care clientul o să facă cereri HTTP este la alegere.
- Vă recomand să folosiți proxy private și nu free: Link
- Aplicația elaborată trebuie să aibă o logică bine definită.

Raspunsuri la întrebări de control:

- Cum este formatat corpul unei cereri HTTP pentru o cerere HTTP de tip POST?
  Tipul de conținut utilizat pentru o solicitare de tip ***POST*** este "***application/x-www-form-urlencoded***".
- De unde știe un client HTTP ce tip de conținut trimite serverul HTTP?
  Serverul va trimite un răspuns, care va include un antet de tip ***Content*** care spune clientului care este tipul de conținut al conținutului returnat.
- Cum decide un client dacă ar trebui să aibă încredere în certificatul unui server?
  Browser-ul, sistemul de operare sau dispozitivul mobil va verifica certificatul utilizând programe CA "membership".

- Care este problema principală cu certificatele autosemnate?
  Problemă unui astfel de certificat este un atac de om în mijloc. Chiar dacă sunteți sigur 100% că sunteți pe site-ul web corect și aveți încredere completă în site (de exemplu, serverul dvs. de e-mail), cineva poate să intercepteze conexiunea și să vă prezinte propriul certificat auto-semnat. Astfel acesta poate primi acces la informația dvs.

- Conexiunea persistentă HTTP – care sunt principalele beneficii?
  - Latență redusă în cererile ulterioare;
  - Mai puține conexiuni ***TCP***;
  - Erorile pot fi raportate fără a închide conexiunea ***TCP***;
  - Putem utiliza ***HTTP Pipelining-ul***;

- Ce este negocierea conținutului în HTTP și cum are loc?
  Negocierea conținutului HTTP este mecanismul utilizat pentru afișarea diferitelor reprezentări ale unei resurse la același URI, astfel încât ***user agent-ul*** să poată specifica care este cel mai potrivit pentru client.

- Care sunt tipurile de negociere a conținutului HTTP?
  Cunoaștem 2 tipuri de negociere a contentului HTTP: ***Negociere de conținut bazată pe server*** și ***Negociere de conținut bazată pe client***.

- Ce este un ETag în HTTP și cum funcționează?
  O etichetă de entitate (***ETag***) este un antet HTTP utilizat pentru validarea cache-ului Web și cererile condiționate de la browsere pentru resurse. Etag-urile folosesc elemente de identificare persistente (PIE) care au fost etichetate în browserul utilizatorului.

- Diferența dintre protocoalele fără stare și cele cu stare. Cărui tip îi aparține HTTP?
  Aceste două tipuri de protocoale se disting după cerința ca software-ul server sau serverul să stocheze informații despre stare sau sesiune. ***HTTP*** este un protocol cu stare.

- Avantajele cheie ale HTTP/2 în comparație cu HTTP/1.1?
  - HTTP/2 utilizează comenzi binare;
  - HTTP/2 este complet multiplexat;
  - HTTP/2 poate folosi o conexiune pentru paralelism;
  - HTTP/2 Utilizează compresia antetului;

- Ce este un tip MIME, din ce constă și pentru ce se folosește?
  Un tip ***MIME*** este o etichetă folosită pentru identificarea unui tip de date. Este utilizat pentru ca software-ul să știe să gestioneze datele.

- Care este diferența dintre GET și POST?
  GET poartă parametrul de solicitare anexat în șirul URL, ***POST*** poartă parametrul de solicitare în corpul mesajului, ceea ce îl face mai sigur pentru a transfera date de la client la server.

- Care este diferența dintre PUT și POST?
  PUT este utilizat pentru actualizarea unui obiect. (***PUT ~/albums/1***)
  POST este utilizat pentru a crea/adăuga un obiect ***nou***. (***POST ~/albums***)

- Care sunt metodele idempotente în HTTP și care este scopul lor?
  DELETE, GET, HEAD, OPTIONS, PUT, TRACE sunt considerate metode "***idempotente***" în HTTP și au ca scop ca clientul să primească același răspuns indiferent de numărul de cereri trimise.

- Cum sunt identificate resursele în protocolul HTTP?
  Fiecare resursă este identificată prin ***URI***.

- Care sunt metodele sigure și nesigure în HTTP?
  GET, HEAD, OPTIONS, TRACE sunt considerate metode "***sigure***" în HTTP și acestea nu permit clientului să modifice conținutul resursei.

- Pentru ce este nevoie de cURL?
  cURL este utilizat în linia de comandă sau scripturi pentru a transfera date.

- Pentru ce este nevoie de HTTP Proxy?
  Un ***proxy*** este utilizat pentru prevenirea atacurilor potențiale de reîncărcare a tamponului.

- Diferența dintre autentificare și autorizare.
  Autentificarea înseamnă confirmarea propriei identități.
  Autorizarea înseamnă acordarea accesului la sistem.

- Care sunt metodele de autentificare HTTP?
  - Basic;
  - Bearer;
  - Digest;
  - HOBA;
  - Mutual;
  - Negociate;
  - OAuth;
  - SCRAM-SHA-1;
  - SCRAM-SHA-256;
  - vapid;

- Modalități de identificare a utilizatorilor în HTTP.
  Există mai multe moduri prin care un server Web ne poate identifica:
  - HTTP request headers;
  - IP address;
  - URL;
  - Cookies;
  - Autentificare;

- HTTP cookies – pentru ce se folosește?
  Cookie-urile sunt utilizate în principal cu 3 scopuri:
  - ***Managementul Sesiunii*** (Autentificări, coșuri de cumpărături, scoruri de jocuri sau orice altceva serverul ar trebui să-și ***amintească***);
  - ***Personalizare*** (Preferințe, teme și alte setări ale ***utilizatorului***);
  - ***Tracking*** (Înregistrarea și analizarea comportamentului ***utilizatorului***);

Descrierea Aplicației:

Utilizând limbajul ***Swift,*** am realizat o aplicație de tip ***Instagram.***
Am utilizat ***fire de execuție***, ***proxy*** propriu(***localhost***), ***regex*** și ***librării HTTP*** deja existente.
Pentru această aplicație am utilizat 2 API-uri GOREST și IMGUR.
Aplicația permite utilizatorului să facă cereri ***GET, POST, HEAD*** și ***OPTIONS***. Clientul se autentifică automat și poate crea posturi noi cu imagini utilizând un ***Access-Token***.