Raport tehnic

Disciplina: Reţele de calculatoare
Proiect: WebS (C)

Druţă Georgiana, grupa 2A2

Universitatea Alexandru Ioan Cuza din Iaşi
Facultatea de Informatică

1 Introducere

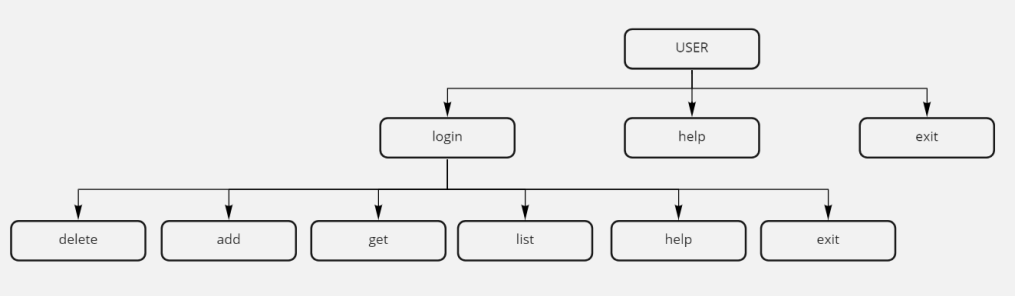
Această fișă de raport are rolul de a prezenta detalii ce au legătură cu realizarea proiectului viitor pe care doresc să îl fac în limbajul C, și anume WebS. Proiectul este o aplicaţie care conține un client și un server web de bază care vor comunica folosind protocolul TCP. Serverul va conține câteva pagini web (fișiere .html simple) și va pune la dispoziția clienților următoarele comenzi: login (autentificarea la server), list (furnizarea listei cu fișierele conținute), get (copierea unei pagini web de pe server), add (uploadarea pe server a unei pagini web), delete (ștergerea unei pagini web de pe server). Clientul va interacționa cu serverul folosind comenzile oferite.

2 Tehnologii utilizate

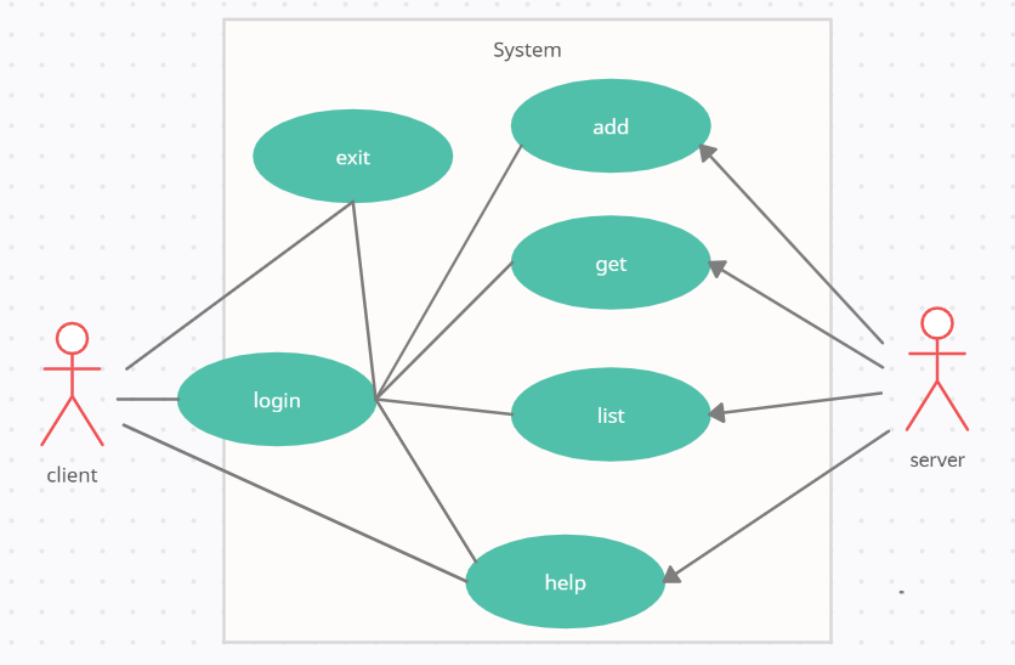
TCP este un protocol orientat-conexiune, de unde reiese că o conexiune este realizată și menținută până când programele de aplicații implicate în procesul de comunicare termină schimbul de mesaje.
Avantajele utilizării unui protocolul TCP sunt: păstrarea ordinii pachetelor ajunse la destinație și asigurarea retransmiterii pachetelor în cazul în care acestea se "pierd". Astfel efectuează o conectare virtuală full duplex între două puncte terminale, fiecare punct fiind definit de către o adresă IP și de către un port TCP, cu alte cuvinte comunicarea dintre client (utilizator) și server se realizează pe un schimb de mesaje de tipul cerere-răspuns.
Servirea iterativă constă în tratarea fiecărei cereri pe rând, secvențial.

3 Arhitectura aplicaţiei

Cu ajutorul următoarelor diagrame se va arăta cum va funcţiona codul în C odată ce programul va fi lansat în linia de comandă.



Se distinge următorul use case:



4 Detalii de implementare

Funcţiile implementate vor respecta diagramele de mai sus, pentru o funcţionarea dorită.

```
// This checks if the file is a html one by only checking it's extension, it doesn't validate the contents of the file
int isHTMLFile(const char* filename) {
    const char* fileExtension = strrchr(filename, '.');
    if(fileExtension == NULL) {
        return 0;
    }
    if(strncmp(fileExtension, ".html", 5) == 0 || strncmp(fileExtension, ".htm", 4) == 0) {
        return 1;
    }
    return 0;
}
```

```c
// the command validation is done by comparing the typed in command(const char* command) with each valid command(const char* validCommands[])
int validate_command(const char* command, const char* validCommands[], long int validCommandsLength) {
    int isCleanCommand = 1;
    // this hack is to empty the stdin. Because scanf won't flush the stdin, so when enter is pressed the string after the format given is still in stdin
    // and it will attempt to assign the following strings in the following scanf calls.
    // For example: if i have scanf("%s", command); other instrunctions; scanf("%s", command2);
    // on the first scanf if I type: `add file1` than command="add" and command2="file1" without prompting you to write anything in
    while(1) {
        int c = getchar();
        if (c == '\n') {
            break;
        }
        else {
            isCleanCommand = 0;
        }
    }
    if(isCleanCommand == 1) {
        for(int i = 0; i < validCommandsLength; i++) {
            if(strncmp(command, validCommands[i], strlen(validCommands[i])) == 0) {
                return 1;
            }
        }
    }
    return 0;
}
```

```c
// After both client and server establish that it is "safe" to transfer the file between the two parties, it'll do the file transfer.
// By "safe" it is meant that the file exists, it's an html file and so on. For all the checks see commands.c isValidFile()
void handleAddCommand(int sockfd) {
    char filename[FILENAME_LENGTH];
    printf("File to be added to the server: ");
    scanf("%s", filename);

    char* sourcefilepath = getFilePath(filename, CLIENT_DIRECTORY);
    int sourceFileDescriptor = getSourceFileDescriptor(sourcefilepath);
    send(sockfd, &sourceFileDescriptor, sizeof(int), 0);
    if(sourceFileDescriptor == -1) {
        return;
    }
    send(sockfd, filename, FILENAME_LENGTH, 0);
    int destinationFileReady;
    recv(sockfd, &destinationFileReady, sizeof(int), 0);
    if(destinationFileReady == -1) {
        fprintf(stderr, "Server Error on preparing the destination file\n");
        return;
    }
    sendFile(sockfd, sourceFileDescriptor);
    free(sourcefilepath);
}
```

```c
// This is exactly as the add command, only that the transfer is the other way around.
void handleGetCommand(int sockfd) {
    char filename[FILENAME_LENGTH];
    printf("File to get from the server: ");
    scanf("%s", filename);
    send(sockfd, filename, FILENAME_LENGTH, 0);

    int sourceFileReady;
    recv(sockfd, &sourceFileReady, sizeof(int), 0);
    if(sourceFileReady == -1) {
        fprintf(stderr, "Server Error on preparing the source file\n");
        return;
    }

    char* destfilepath = getFilePath(filename, CLIENT_DIRECTORY);
    int destinationFileDescriptor = getDestinationFileDescriptor(destfilepath);
    send(sockfd, &destinationFileDescriptor, sizeof(int), 0);
    if(destinationFileDescriptor == -1) {
        return;
    }

    receiveFile(sockfd, destinationFileDescriptor);
    free(destfilepath);
}
```

```c
// As the previous commands, first it needs to be clear that the file is safe to be deleted
// these checks are done, so the client and server are always in sync
void handleDeleteCommand(int sockfd) {
    char filename[FILENAME_LENGTH];
    printf("File to delete from the server: ");
    scanf("%s", filename);
    send(sockfd, filename, FILENAME_LENGTH, 0);
    int isFileValid;
    recv(sockfd, &isFileValid, sizeof(int), 0);
    if(isFileValid == -1) {
        fprintf(stderr, "Server Error on preparing file for deletion\n");
        return;
    }
    int removeStatus;
    recv(sockfd, &removeStatus, sizeof(int), 0);
    if(removeStatus == -1) {
        fprintf(stderr, "Server error on deleting file\n");
    }
}
```

5 Concluzii

În privința tehnologiilor utilizate, un server UDP ar fi mai bun din punct de vedere al vitezei. Acesta ar fi mai optim, dar nu garantează că pachetele vor ajunge în ordine și întregi.
O altă îmbunătățire adusă proiectului poate fi căutarea recursivă în fișierul Server, pentru a căuta fișiere html și în subfoldere. Deoarece nu s-a specificat căutarea recursivă, am căutat fișiere html doar în fișierul Server.

References

1. https://profs.info.uaic.ro/~computernetworks/
2. Use case, https://app.creately.com/diagram/
3. https://www.geeksforgeeks.org/socket-programming-cc/
4. https://linux.die.net/man/3/htonl
5. https://pubs.opengroup.org/
6. https://ro.wikipedia.org/wiki/Transmission_Control_Protocol/
7. https://pubs.opengroup.org/onlinepubs/7908799/xsh/dirent.h.html/
8. https://miro.com/app/dashboard/