# AppRepositoryS

Madalina Varga

Universitatea Alexandru Ioan Cuza Iași

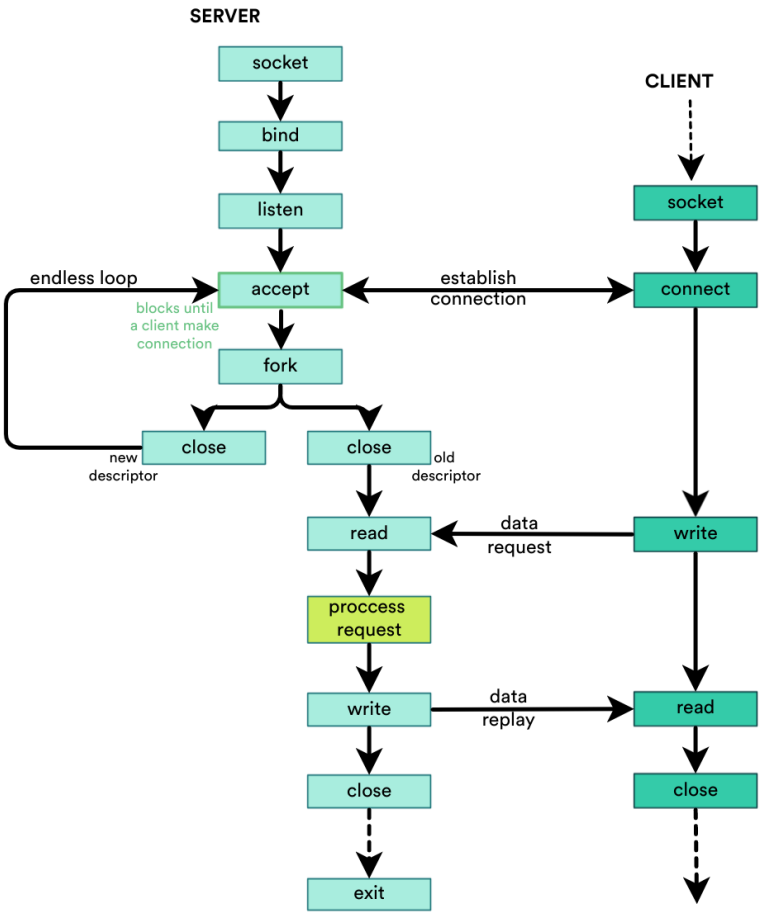
## 1 Introducere

**Acest document prezintă detalii de implementare a aplicației AppRepositoryS în limbajul C++. O aplicație cu potențial de dezvoltare și funcționalități practice și actuale.**
**Scopul** este de a dezvolta o aplicație server/client care să permită adăugarea de aplicații noi de către orice client autentificat care poate oferi toate specificațiile necesare rulării lor. De asemenea, clienții pot căuta aplicații după anumite filtre puse la dispoziție de server cu scopul de a reduce timpul de căutare.

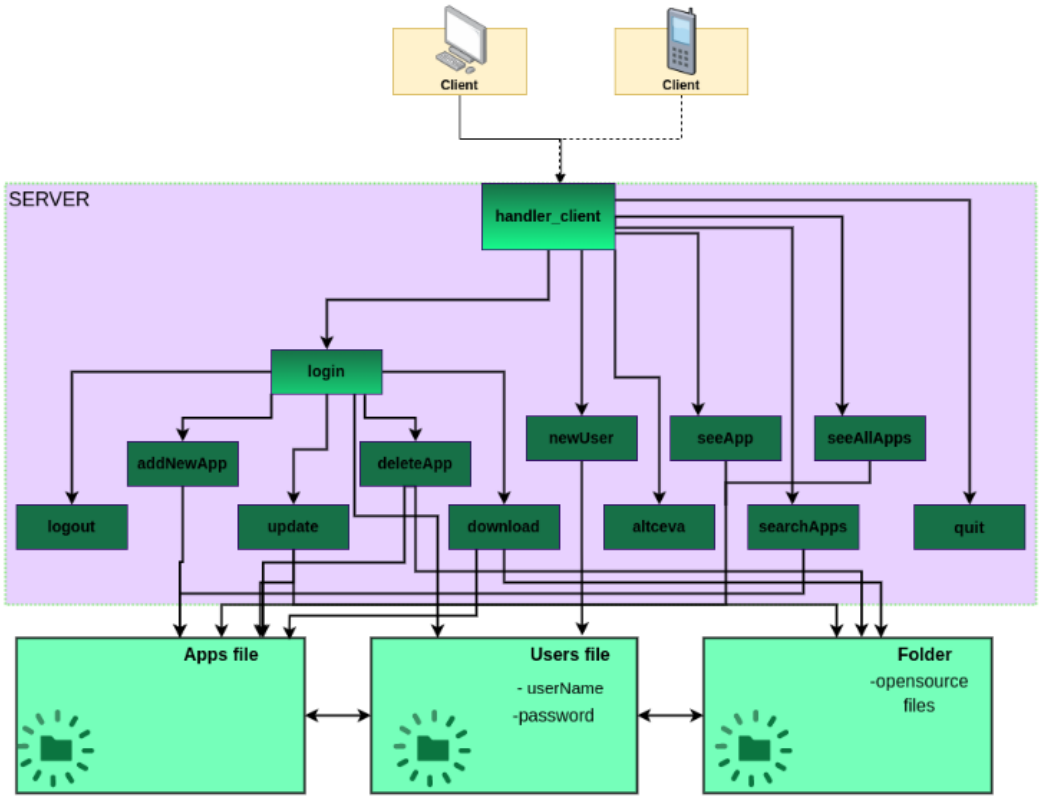
## 2 Tehnologii utilizate

Pentru realizarea proiectului am ales să folosesc Transmission Control Protocol (TCP) deoarece este un protocol de transport orientat, cu conexiune, fără pierdere de informații, ce controlează fluxul de date[1].
Am optat pentru varianta TCP concurent în detrimentul celui iterativ pentru că vreau să asigur dinamica aplicației și să ofer posibilitatea mai multor clienți să interacționeze cu aplicația în același timp,protocolul oferind siguranța transmiterii informațiilor în ordinea în care acestea au fost transmise.

Urmatoarea diagramă de secvență modelează comportamentul aplicației



# 3 Arhitectura aplicației



## 4 Detalii de implementare

Pentru toate funcții ce primesc parametrii se verifică dacă numărul parametrilor este cel corespunzător.

### 4.1 comanda "login:"

```
else if (strstr(msg, "login:")) // login:nume parola
{
    check_param = validParameters(msg, 1);
    if (check_param == 1)
    {
        if (isLogged == FALSE)
        {
            strcpy(userName, " ");
            char *userAccountDetails = getInputCommand(msg); // return username password
            int found = checkExistingUser(userAccountDetails);
            if (found != 0) //yes
            {

                isLogged = TRUE;
                // save the username for next requests
                char *user_ptr = getFirstParameter(userAccountDetails);
                strcpy(userName, user_ptr);
                strcpy(msg, "Logged in\n");
                writeInSocket(msg, client_fd); // return succes msg
            }
            else // user doesn't exist
            {
                strcpy(msg, "User does not exist\n");
                writeInSocket(msg, client_fd);
            }
        }
        else
        {
            strcpy(msg, "already logged in\n");
            writeInSocket(msg, client_fd);
        }
    }
    else
    {
        strcpy(msg, "incorrect number of parameters\n");
        writeInSocket(msg, client_fd);
    }
```

La apelul acestei funcții, serverul verifică dacă clientul nu **este deja autentificat**. În cazul unui raspuns negativ serverul verifică dacă numele de utilizator și parola se găsesc în fișierul ce conține utilizatorii înregistrați. Dacă există, serverul autentifică clientul și returnează un mesaj de succes. În caz contrat, serverul trimite un mesaj de eșec.

### 4.2 comanda ”logout”

```
else if (strstr(msg, "logout")) //logout
{
    if (isLogged == TRUE)
    {
        isLogged = FALSE;
        strcpy(msg, "Logged out\n");
        writeInSocket(msg, client_fd);
    }
    else
    {
        strcpy(msg, "you are not logged in\n");
        writeInSocket(msg, client_fd);
    }
}
```

Dacă un utilizator este logat, prin apelul acestei funcții se setează variabila isLogged la FALSE și se trimite un mesaj de succes către client.Altfel, se returneaza un mesaj de eroare.

### 4.3 comanda ”quit” sau Ctrl+C

```
if (strstr(msg, "quit")) // quit
{

    writeInSocket(msg, client_fd);
    close(client_fd);
    exit(0);
}
```

Utilizatorul se deconectează sau părăsește aplicația.

### 4.4 comanda ”newUser:”

```
else if (strstr(msg, "newUser:")) //newUser:nume parola
{
    check_param = validParameters(msg, 1);
    if (check_param == 1)
    {
        char *new_user_details = getInputCommand(msg); // return username password
        char *wanted_name = getFirstParameter(new_user_details);
        int found = checkExistingUserNameOnly(wanted_name);

        if (found == 0)
        {
            // the wanted username is avaible => continue
            char *password = getSecondParameter(new_user_details);
            int check_password = validPassword(password);
            if (check_password == 1)
            {
                //good pass => create account
                writeInFile(new_user_details, config_file);
                strcpy(msg, "user created\n");
                writeInSocket(msg, client_fd);
            }
            else
            {
                strcpy(msg, "incorrect password format\n");
                writeInSocket(msg, client_fd);
            }
        }
        else
        {
            strcpy(msg, "username already exists\n");
            writeInSocket(msg, client_fd);
        }
    }
    else
    {
        strcpy(msg, "incorrect number of parameters\n");
        writeInSocket(msg, client_fd);
```

Funcția primește ca parametrii detaliile despre noul client ce dorește să se înregistreze,adică user-name și parola. Se verifică dacă username-ul nu aparține unui alt utilizator.În cazul unui răspuns negativ se verifică dacă parola respectă condițiile minime.Contul este creat prin adăugarea acestuia în fișierul cu utilizatori.

### 4.5 comanda ”addNewApp:”

```
else if (strstr(msg, "addNewApp:")) //addNewApp:file
{
    check_param = validParameters(msg, 0);
    if (check_param == 1)
    {

        if (isLogged == TRUE) // must be logged in
        {
            listOfApps = getListOfApps();          //refresh
            id_g = listOfApps.end()->id;           //set the last id nr
            char *fileName = getInputCommand(msg); // take the parameter: a file with the new app details
            AppDetails app(userName);              //create an object
            //set the object attributes
            app.setFromtxtFile(fileName);
            int dir_error = 0, open_fd = 0;
            if (strcmp(app.src_file, "default") != 0) // check is open source…
            if (dir_error == -1 || open_fd == -1)
                strcpy(msg, "Error to copy the files");
            else
                snprintf(msg, SIZE, "App loaded with id: %d", app.id);
            char *output_string = app.toString();
            writeInFile(output_string, apps_file); //save the new app
            writeInSocket(msg, client_fd);
        }
        else
        {
            strcpy(msg, "You must be logged in to add a new app\n");
            writeInSocket(msg, client_fd);
        }
    }
    else
    {
        strcpy(msg, "incorrect number of parameters\n");
        writeInSocket(msg, client_fd);
    }
```

Doar utilizatorii logați pot adăuga noi aplicații prin trimiterea unui fișier cu detalii despre aplicație. Funcția crează un obiect nou și setează atributele cu detaliile găsite în fișier. Atributele obiectului obținut sunt convertite prin funcția ’.toString()’, într-un singur string ce este salvat în fișierul cu aplicații pe o linie.Dacă aplicația este open source atunci fisierul este salvat intr-un director cu id-ul aplicației.

### 4.6 comanda ”deleteApp:”

```
else if (strstr(msg, "deleteApp:")) //id
{
    check_param = validParameters(msg, 0);
    if (check_param == 1)
    {
        int found = 0, unathorised = 0;

        if (isLogged == TRUE) //must be logged in
        {
            char *parameter = getInputCommand(msg);
            int id = atoi(parameter);
            int line_id = 0;
            FILE *file_fd_w, *file_fd_r;
            char path_file[256];

            for (auto app = listOfApps.begin(); app != listOfApps.end(); app++) //search for the id
            {
                if (app->id == id)
                {
                    if (strcmp(app->owner, userName) == 0) //check the username = owner
                    {
                        strcpy(path_file, app->src_file);
                        //open the apps file in read mode and a temporary app in write
                        if ((file_fd_r = fopen(apps_file, "r")) == NULL || (file_fd_w = fopen("delete.tmp", "w+")) == NULL)
                            printf("eroare deschidere fisier\n");

                        long file_size = sizeOfFile(apps_file);
                        char *fileContent = (char *)malloc(file_size + 1);
                        while (1) //while i still have lines to read
                        fclose(file_fd_w);
                        fclose(file_fd_r);
                        remove(apps_file);
                        rename("delete.tmp", apps_file);

                        free(fileContent);
                        if (strcmp(path_file, "default") != 0)
                        found = 1;
                        strcpy(msg, "app deleted\n");
                    }
```

Orice utilizator ce dorește să steargă o aplicație ce îi aparține poate folosi această comandă. Funcția verifică dacă există id-ul dat și dacă aceasta aparține utilizatorului ce a facut solicitarea

### 4.7 comanda ”update:”

```
else if (strstr(msg, "update:")) // update:id file with the new details
{
    check_param = validParameters(msg, 1);
    if (check_param == 1)
    {
        if (isLogged == TRUE) //must be logged in
        {
            char *parameters = getInputCommand(msg);
            char *id_app = getFirstParameter(parameters);      //take the id
            char *fileUpdate = getSecondParameter(parameters); //take the file name
            int id = atoi(id_app);
            int found = 0, unathorised = 0;
            FILE *file_fd = fopen(apps_file, "w+");
            fseek(file_fd, 0, SEEK_SET);
            for (auto app = listOfApps.begin(); app != listOfApps.end(); app++)
            {
                if (app->id == id) // check if the id is in the list
                {
                    if (strcmp(app->owner, userName) == 0) // check if the user is the owner
                    {
                        found++;

                        if (strcmp(app->src_file, "default") != 0)
                        else
                            app->setFromtxtFile(fileUpdate); //reset the attributes
                    }
                    else
                        unathorised = 1;
                }

                char *output_string = app->toString();
                fprintf(file_fd, "%s\n", output_string);
            }
            strcpy(msg, "app updated\n");
            fclose(file_fd);
```

Utilizatorii pot îmbunătăți și pot aduce schimbări aplicațiilor prin această comanda.Doar proprietarul poate face update.

### 4.8 comanda ”seeApp:”

```
else if (strstr(msg, "seeApp:")) //seeApp:id
{
    check_param = validParameters(msg, 0);
    if (check_param == 1)
    {
        char *id_app = getInputCommand(msg); //take the id and search it in the list
        int id = atoi(id_app);
        int found = 0;
        for (auto app = listOfApps.begin(); app != listOfApps.end(); app++) //check if the id is in the list
        {
            if (app->id == id)
            {
                strcpy(msg, app->toString()); //convert it to string and return the details to the client
                found = 1;
                break;
            }
        }
        if (found == 0)
            strcpy(msg, "ivalid id\n");

        writeInSocket(msg, client_fd);
    }
    else
    {
        strcpy(msg, "incorrect number of parameters\n");
        writeInSocket(msg, client_fd);
    }
}
```

Serverul primeste id-ul aplicației dorite de client și caută în lista cu aplicații dacă există un corespondent pentru acest id. În cazul unui raspuns pozitiv, serverul returnează toate datele aplicației clientului.

### 4.9 comanda ”seeAllApps:”

```
else if (strstr(msg, "seeAllApps:")) //page nr
{
    check_param = validParameters(msg, 0);
    if (check_param == 1)
    {
        char *firstParam = getInputCommand(msg);
        int page_number = atoi(firstParam);  // take the page number
        int first_app = page_number * 5 - 5; //5 apps per page
        int line_index = 0, count = 5;
        int size = sizeOfFile(apps_file);
        char returnedString[size] = ""; //a string with all apps

        for (auto app = listOfApps.begin(); app != listOfApps.end(); app++)
        {
            if (line_index == first_app) //find the first app wanted
            {
                if (count > 0) // take the 5 apps wanted in returned string
                {
                    char *output_string = app->toString();
                    strcat(returnedString, output_string);
                    strcat(returnedString, "\n");
                }

                first_app++;
                count--;
            }
            line_index++;
        }
        if (strlen(returnedString) < 1)
        {
            strcpy(msg, "No apps avaible\n");
            writeInSocket(msg, client_fd);
        }
        else
            writeInSocket(returnedString, client_fd);
    }
```

Clienții pot apela această comandă pentru a vedea toate aplicațiile disponibile pe site. Pentru că numărul acestora poate crește considerabil funcția returnează 5 aplicații pe o pagină.

### 4.10 comanda ”searchApps:”

```
else if (strstr(msg, "searchApps:")) //searchApps:filename
{
    check_param = validParameters(msg, 0);
    if (check_param == 1)
    {
        char returnedString[SIZE] = "";
        char *nameOfFile = getInputCommand(msg);
        char delim[] = "{},:\" \t\n";
        bool valid = true;
        bool foundOneApp = false;

        for (auto app = listOfApps.begin(); app != listOfApps.end(); app++)
        {
            char *filtersDetails = readFile(nameOfFile); // filters
            char *field = strtok(filtersDetails, delim);
            char *value = strtok(NULL, delim);

            while (field) // for each app check all filters
            {
                valid = isValidField(field, value, *app); // return if the current

                if (valid == false)
                {
                    break;
                }
                field = strtok(NULL, delim);
                value = strtok(NULL, delim);
            }
            if (valid == true) //match…
        }
        if (foundOneApp == false)
        {
            strcpy(returnedString, "Didn't find any app\n");
        }

        writeInSocket(returnedString, client_fd);
    }
```

Această comandă este pentru a facilita interacțiunea utilizatorului cu aplicația. Serverul primește un fișier cu filtrele dorite de utilizator.Se parcurg aplicațiile iar cele care îndeplinesc toate criteriile sunt returnate utilizatorului.

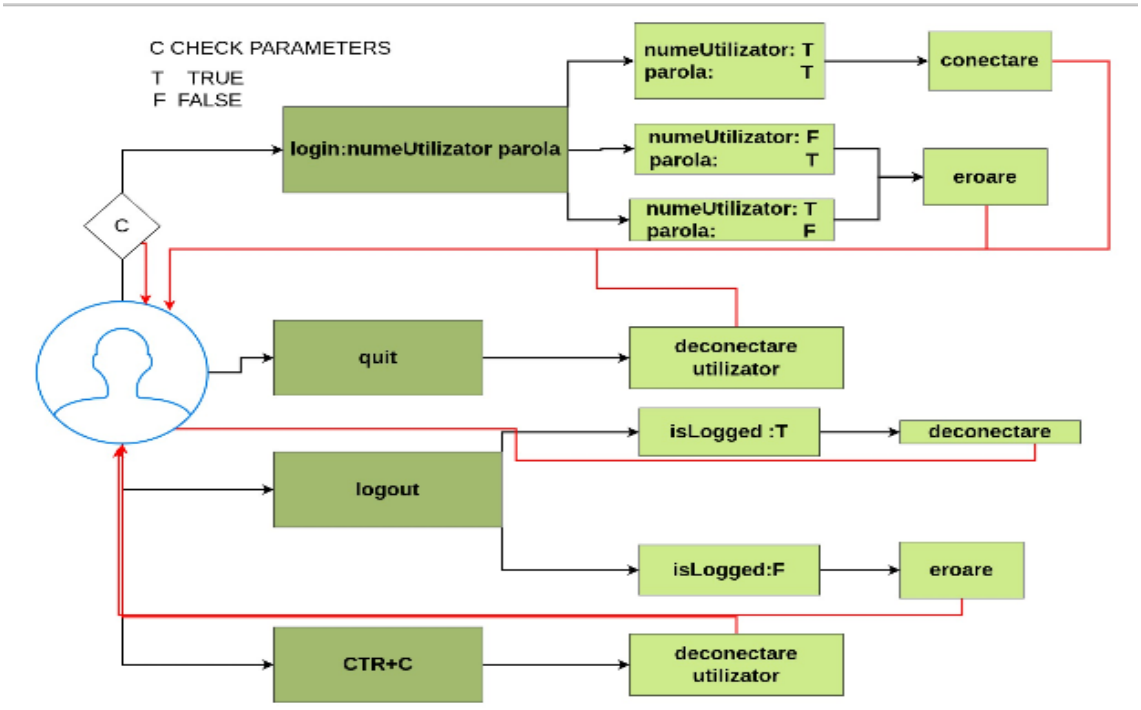
### 4.11 comanda ”downloadApp:”

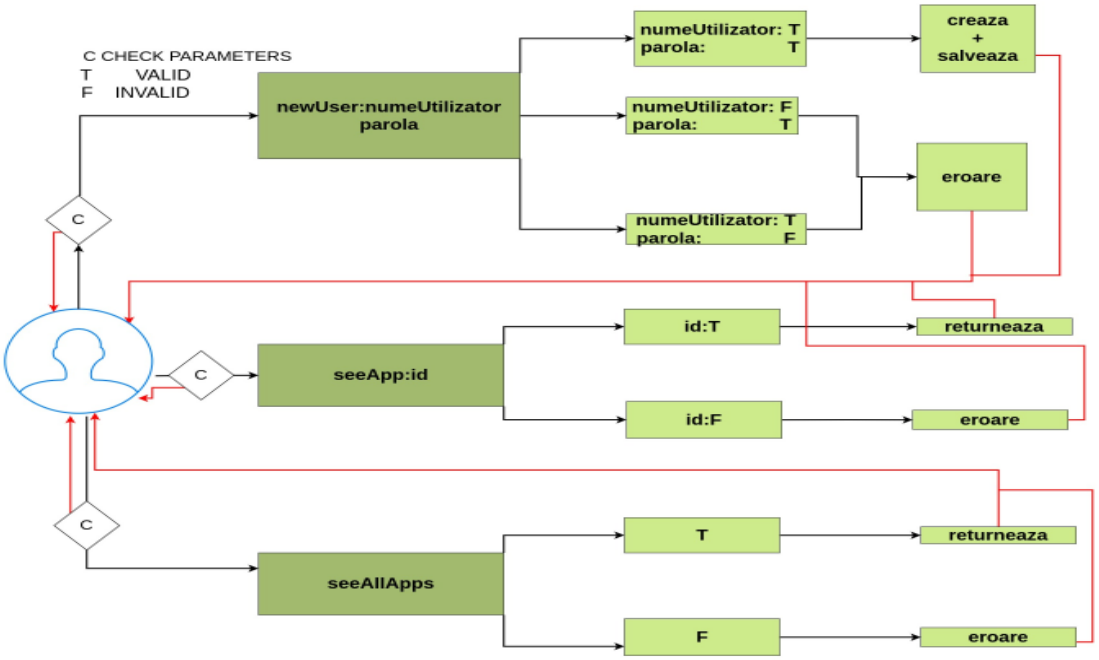
```
else if (strstr(msg, "downloadApp:")) //downloadApp:id
{
    check_param = validParameters(msg, 0);
    if (check_param == 1)
    {
        if (isLogged == TRUE)
        {
            char *id_app = getInputCommand(msg); //take the id and search it in the list
            int id = atoi(id_app);
            int found = 0;
            for (auto app = listOfApps.begin(); app != listOfApps.end(); app++)
            {
                if (app->id == id)
                {
                    found = 1;
                    if (strcmp(app->src_file, "default") == 0) // check if the app is open source
                        strcpy(msg, "this app is not open source\n");
                    else
                    {
                        strcpy(msg, app->src_file);
                    }
                }
            }
            if (found == 0)
                strcpy(msg, "app doesn't exist\n");

            writeInSocket(msg, client_fd);
        }
        else…
    }
    else
    {
        strcpy(msg, "incorrect number of parameters\n");
        writeInSocket(msg, client_fd);
    }
```

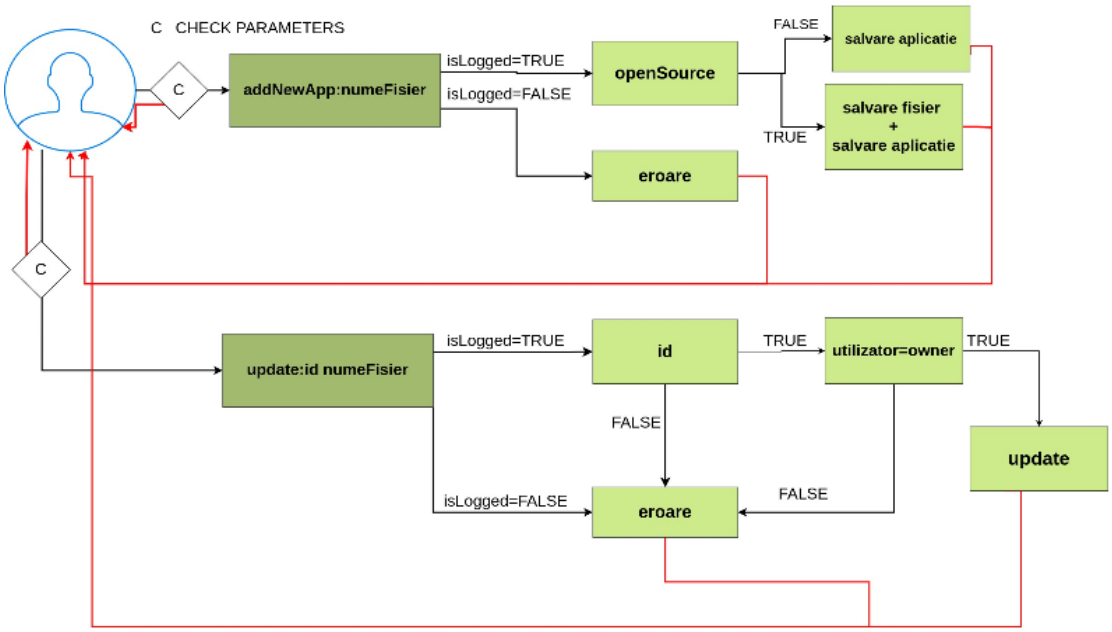
Orice utilizator ce are un cont poate descărca aplicațiile open-source.

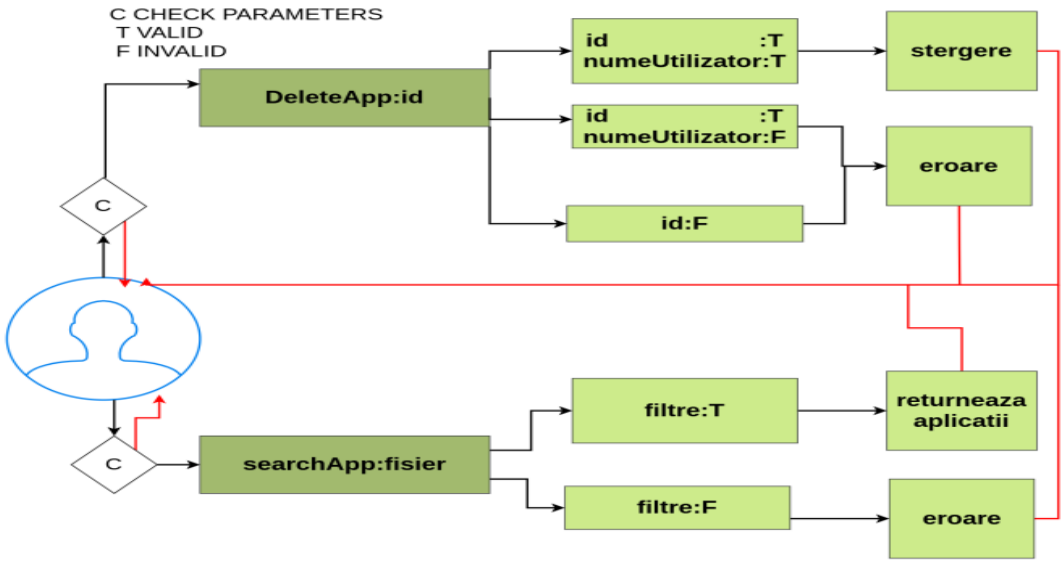
### Scenarii de utilizare

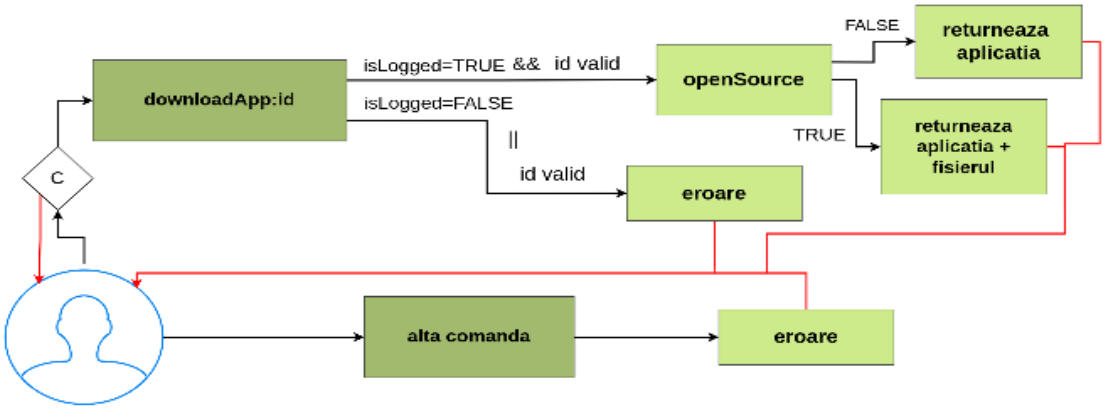
**Clientul caută aplicații:**
Clientul care a decis să folosească aplicația poate să vadă toate aplicațiile disponibile sau să caute aplicații după filtrele dorite și disponibile.Dacă a găsit o aplicație ce prezintă interes, poate vedea detalii despre aceasta.Dacă este open source și vrea să o descarce trebuie să se logheze. Dacă nu are un cont de utilizator poate să își facă unul.Dacă utilizatorul părăsește brusc aplicația pentru că este nemulțumit,serverul primește un mesaj și îl deconectează.
**Clientul vrea să încarce o aplicație:**
Clientul care nu este autentificat și dorește să încarce o aplicație primește un mesaj de eroare. În urma autentificării,acesta poate adăuga aplicația. În orice moment poate să o modifice sau dacă este nemulțumit să o steargă.După încarcarea aplicației clientul primește id-ul asignat acesteia.Clientul caută aplicația pe site după filtre. Se uită la detalii și mulțumit se deconectează și părăsește aplicația.




C CHECK PARAMETERS
T VALID
F INVALID
C
newUser:numeUtilizator
parola
numeUtilizator: T
parola: T
numeUtilizator: F
parola: T
numeUtilizator: T
parola: F
creaza
+
salveaza
eroare
returneaza
C
seeApp:id
id:T
id:F
eroare
C
seeAllApps
T
F
returneaza
eroare


C CHECK PARAMETERS
C
addNewApp:numeFisier
isLogged=TRUE
isLogged=FALSE
openSource
FALSE
TRUE
salvare aplicatie
salvare fisier
+
salvare aplicatie
eroare
C
update:id numeFisier
isLogged=TRUE
id
TRUE
utilizator=owner
TRUE
FALSE
update
isLogged=FALSE
eroare
FALSE


C CHECK PARAMETERS
T VALID
F INVALID
DeleteApp:id
id :T
numeUtilizator:T
id :T
numeUtilizator:F
id:F
stergere
eroare
C
C
searchApp:fisier
filtre:T
filtre:F
returneaza aplicatii
eroare


downloadApp:id
isLogged=TRUE && id valid
isLogged=FALSE
||
id valid
openSource
FALSE
returneaza aplicatia
TRUE
returneaza aplicatia + fisierul
eroare
C
alta comanda
eroare

## 5 Concluzii

Pentru implementarea aplicației am folosit protocolul TCP concurent unde am creat o functie handler_client ce face posibilă interacțiunea dintre client-server.
Aplicația poate fi accesată de oricine dar unele funcționalități sunt restricționate pentru utilizatorii neînregistrați.
Pentru salvarea informațiilor despre utilizatori și aplicații înregistrate, utilizez fișiere.
Toți utilizatorii pot accesa aplicații după anumite filtre, pentru a scurta timpul de căutare.
Fiecarei aplicații ii este oferit un id unic.
O aplicație open source poate fi descărcată de orice utilizator înregistrat
**Idei de dezvoltare:**
O idee de îmbunătățire este crearea unei baze de date cu utilizatorii și aplicațiile înregistrate.
De asemenea, se poate dezvolta o nouă componentă pentru aplicațiile open source unde utilizatorii pot aduce îmbunătățiri si pot încarca noua versiune pe pagina aplicației respective.
Dezvoltarea unui sistem de interacțiune între utilizatori pentru a putea oferii feedback și pentru a putea adauga comentarii.

## 6 Bibliografie

[1] https://profs.info.uaic.ro/ computernetworks