Team Batman

# ShareIt

## Arhitectural Design Document

Team Batman for
Team Superman’s SRS

# 1. Tehnologii folosite - Precizare si Descriere

In vederea implementarii proiectului solicitat, s-a decis asupra alegerii tehnologiilor, arhitecturilor si limbajelor de programare prezentate in continuare.

## 1.1 Limbajul de programare: Java

- este portabil, Java fiind un limbaj independent de platforma de lucru;
  - este neutru din punct de vedere arhitectural.
- dispune de o multitudine de clase, cu un nivel ridicat de optimizare;
  - faciliteaza programarea in retea prin punerea la dispozitie a numeroase clase si framework-uri in acest scop.
- este un limbaj de programare cu un nivel ridicat de securitate, asigurând mecanisme stricte de securitate a programelor concretizate prin:
  - verificarea dinamica a codului pentru detectarea secventelor periculoase,
  - impunerea unor reguli stricte pentru rularea programelor lansate pe calculatoare aflate la distanta, etc.

Detalii suplimentare:

- https://www.java.com
- http://www.oracle.com/us/technologies/java/overview/index.html

## 1.2 Mediul de dezvoltare: Eclipse

- unul din cele mai bune IDE-uri disponibile la ora actuala pentru Java
- Code Completion
  - Eclipse ofera cea mai buna metoda de code completion pentru Java. Acesta analizeaza contextul si sugereaza cea mai buna optiune.
- Syntax Checking;
- Multi-language Refactorings;
- Marketplace: ofera o multime de plug-in-uri care pot usura munca programatorului.

Detalii suplimentare:

- http://www.eclipse.org/ide/

## 1.3 Arhitecturi

Datorita portabilitatii limbajului Java, aplicatia va putea rula pe orice arhitectura acceptata de Java Virtual Machine, fara a fi necesare modificari asupra codului sursa. Acestea pot fi atat sisteme Windows cat si Unix/Linux, respectiv arhitecturi pe 32 biti sau 64 biti.

## 1.4 Biblioteci si Utilitare

Pentru obtinerea si prelucrarea obiectelor de tip user filelist, se va folosi clasa File, din pachetul java.io. Aceasta ofera o descriere abstracta a fisierelor si directoarelor organizate sub forma unei ierarhii independente de sistem.

Clasa File joaca un rol foarte important in dezvoltarea aplicatiei si in implementarea operatiilor principale pe care aceasta trebuie sa le execute. Astfel, se vor colecta informatii legate de fisiere precum tipul acestora, dimensiunea, informatiiile detinute de acestea, extragerea caii catre fisier etc.

Detalii suplimentare:

- http://docs.oracle.com/javase/7/docs/api/java/io/File.html .

Pentru a putea efectua eficient operatiile de download/upload, rularea managerilor de upload/download va avea loc simultan pe thread-uri diferite. Asadar, pentru a implementa cat mai eficient cererile inter-client, se va folosi clasa Thread. Aceasta ofera operatii de baza ce pot fi efectuate asupra thread-urilor precum: creerea acestora, terminarea lor, cat si interogarea status-ului fiecarui thread, id-ului sau, etc.

Detalii suplimentare:

- http://docs.oracle.com/javase/6/docs/api/java/lang/Thread.html .

In vederea stabilirii conexiunilor client-server sau client-client in scopul transferurilor de date si cereri, s-a optat pentru folosirea clasei Socket din pachetul java.net.Socket. Aceasta pune la dispozitie metode de creere socketi, stabilire si incheiere conexiuni, interogare stare conexiune, etc.

Detalii suplimentare:

- http://docs.oracle.com/javase/7/docs/api/java/net/Socket.html .

## 1.4 Frameworks

Pentru simplificarea implementarii operatiilor P2P, vom opta ulterior pentru unul din urmatoarele API-uri:

- Apache Mina:
  - Framework pentru dezvoltare de network applications de inalta performanta si scalabilitate ridicata
  - https://mina.apache.org/
- Netty:
  - Framework client-server pentru dezvoltarea rapida de aplicatii; simplifica foarte mult programarea cu socketi
  - http://netty.io/
- RMI
  - API object-oriented echivalent cu Remote Procedure Call (RPC); suporta transfer direct de obiecte si clase Java serializate
  - http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html

## 1.5 Interfata

Dorim sa urmarim liniile de design folosite in conceptele MetroUI, noua interfata Windows8. Astfel, vom folosi urmatoarele:

- JavaFX
  - platforma grafica customizabila prin CSS;
  - este noua biblioteca standard de Graphic User Interface pentru Java SE;
  - pentru utilizare, se foloseste cod nativ Java;
  - Scene Builder: o aplicatie de tip drag and drop pentru construirea interfetei aplicatiei.

- JMetro
  - customizari CSS scrise peste JavaFX pentru a urmari liniile de design date de noua interfata Windows, Metro UI

## 2. Module

In aceasta sectiune va fi realizata o descriere succinta a modulelor de nivel inalt ce vor fi folosite in implementarea aplicatiei.

In principiu vom avea 3 module importante :

- ➢ Client
- ➢ Server
- ➢ Interfata

### A. Modulul Client

Aici sunt incluse toate clasele necesare implementarii aplicatiei pentru client.

Acesta va interactiona atat cu modulul Server cat si cu modulul interfata.

Acesta va contine urmatoarele clase:

**a) Client** :

- in principiu aceasta clasa se va ocupa de comunicatia cu serverul
- va putea cere lista de fisiere a unui utilizator prin conexiune directa
- campuri :
    - UserList (LinkedList<User>) : contine o lista cu utilizatorii online
    - FileList (LinkedList<FileInfo>) : contine lista fisierelor distribuite de utilizator
    - Username (String) : numele utilizatorului
    - IP (InetAddress) : adresa IP necesara conectarii la acest client
    - ServerIP (InetAddress) : adresa IP a serverului
    - ServerPort (int) : portul necesar conectarii la server
    - ClientPort (int): portul necesar conectarii la clientul actual
    - UploadMGRPort (int) : portul pe care trimite date
    - DownloadMGRPort (int) : portul pe care primeste date

- metode :
    - init () : va incarca pe disc lista de fisiere partajate
    - getFilelist() : intoarce lista de fisiere distribuite de acest utilizator
    - getUserFilelist(String user) : intoarce lista cu fisierele distribuite de utilizatorul al carui nume il primeste ca parametru
    - putFileDownloadJob(String username,String filename) :
        - primeste ca parametri numele utilizatorului de la care va descarca fisierul si numele fisierului
        - folosind acesti parametri va crea un obiect de tip DownloadJob pe care il va insera in DownloadWorkpool
        - va seta statusul jobului la “pending”
    - search (String searchPhrase) : stabileste o conexiune cu serverul prin care va trimite un pachet de date ce contine un String cu tipul operatiei(„search”) si

searhPhrase-ul dupa care realizeaza cautarea iar apoi va astepta sa primeasca de la acesta o lista de obiecte FileInfo cu informatii despre fisierele care au facut match cu searchPhrase(daca primeste o lista vida inseamna ca nu s-a gasit nici o potrivire).Dupa ce primeste raspuns , va inchide conexiunea si va transmite aceasta lista modulului de interfata grafica pentru a o afisa utilizatorului.

- getFilteredUsersList(String filter): va parcurge lista de utilizatori online si va intoarce o lista(un obiect LinkedList<String>) cu utilizatorii ai caror username incepe cu sirul dat ca parametru
- filterUserFilelist(String filter) : va parcurge lista de utilizatori online si va intoarce o lista(un obiect LinkedList<String>) cu utilizatorii ai caror username incepe cu sirul dat ca parametru
- addFile(String file) : primeste ca parametru un sir ce reprezinta calea catre fisierul pe care il adauga in lista de fisiere partajate
- removeFile(String file) : primeste ca parametru un sir ce reprezinta calea catre fisierul pe care il sterge din lista de fisiere partajate
- setDownloadSlots(int DownloadSlots) : seteaza numarul maxim de sloturi pentru download
- setUploadSlots(int UploadSlots) : seteaza numarul maxim de sloturi pentru upload
- logIn(String username,String IP,String port) : se va ocupa de procesul de autentificare/inregistrare
- logOut(): apelarea acestei metode va declansa procesul de delogare

**b) DownloadManager** :

- aceasta clasa mosteneste clasa Thread
- metodele acestei clase se vor ocupa de procesul de download
- va “comunica” cu clasa UploadManager a utilizatorului de la care dorim sa descarcam fisiere
- campuri :
  - WorkPool (DownloadWorkPool) :
- metode :
  - run() - aceasta metoda va stabili conexiunea cu UploadManagerul clientului de la care se descarca fisierul si se va ocupa efectiv receptionarea pachetelor de date
  - SetDownloadSlots(int DownloadSlods)
  - CancelJob(String filename,String username): metoda va cauta jobul in coada Jobs si il va elimina
  - getDownloadStatus(): va intoarce un intreg ce reprezinta ce procent din fisier a fost descarcat(ce procent din actiunea de download a fost finalizat)

**c) UploadManager** :

- mosteneste clasa Thread
- metodele acestei clase se vor ocupa logica de upload
- va “comunica” cu clasa DownloadManager a utilizatorului care doreste sa descarce fisiere de la utilizatorul curent
- campuri :
    - pendingJobs (Queue <UploadJob>) : coada in care vor fi tinute UploadJob-urile(de fiecare data cand un slot va fi eliberat, un UploadJob va fi preluat din coada)
    - uploadSlots (int) : este un intreg ce reprezinta numarul maxim de sloturi de upload ce pot fi active simultan
- metode :
    - run() : in aceasta metoda se va stabili conexiunea cu DownloadManagerul clientului caruia vrem sa-i trimitem fisierul si va fi realizata efectiv trimiterea pachetelor de date
    - setUploadSlots(int uploadSlots) : seteaza numarul maxim de sloturi de upload
    - setJob(String filename,User user) : creeaza un obiect de tip UploadJob pe care il adauga in coada pendingJobs
    - getUploadStatus() : va intoarce un intreg ce reprezinta ce procent din fisier a fost uploadat (ce procent din procesul de upload a fost finalizat)

**d) DownloadWorkpool**

- campuri :
    - Jobs (Queue <DownloadJob>) : coada in care vor fi tinute toate joburile ce vor fi preluate apoi de DownloadManager
- metode:
    - GetWork(): va prelua un obiect DownloadJob din coada
    - PutWork(DownloadJob Job) : va adauga un obiect DownloadJob din coada
    - HasWork(): va intoarce True daca coada nu este goala si False in caz contrar
    - CancelJob(String filename,String username): metoda va cauta jobul in coada Jobs si il va elimina

**e) DownloadJob**

- campuri :
    - filename(String) : numele fisierului ce va fi downloadat
    - userFrom(User) : numele utilizatorului care detine fisierul ce va fi descarcat
- metode :
    - getFilename() : intoarce numele utilizatorului de la care se descarca fisierul
    - setFilename(String filename) : seteaza numele utilizatorului de la care se descarca fisierul
    - getUserInfo() : intoarce un obiect de tip User , cu datele utilizatorului de la care se descarca fisierul
    - setUserInfo(User UserInfo) : setaza campul useFrom cu datele din userInfo

**f) FileInfo**

- campuri:
    - path(String) : un sir ce reprezinta calea catre fisier
    - isDir (Bool) : va fi True daca fisierul este de tip director si False daca nu
    - content (LinkedList<FileInfo>) :daca fisierul nostru este director va contine o lista de obiecte FileInfo cu
    - type (String): reprezinta tipul fisierului
        - daca este director va fi „directory”
        - altfel va fi egal cu extensia fisierului
    - size (BigInteger) : dimensiunea fisierului in Bytes
- metode :
    - getPath(): intoarce un String ce contine calea catre fisier
    - setPath (String path): va seta campul path cu calea catre fisier
    - setDir(): seteaza campul isDir la True
    - clearDir(): seteaza campul isDir la False
    - getContent() : intoarce o lista cu obiecte de tip FileInfo cu informatii despre lista de fisiere din director
    - setContent (LinkedList<FileInfo> content) : adauga in lista “content” obiectele FileInfo din lista primita ca parametru
    - getType(): intoarce un String cu tipul fisierului
    - setType(String type) : seteaza tipul fisierului
    - getSize() : intoarce un BigInteger cu dimensiunea fisierului
    - setSize(BigInteger size) : seteaza dimensiunea fisierului

**g) UploadJob**

- campuri :
  - filename (String) : numele fisierului ce va fi uploadat
  - useFor (User) : un obiect de tip User pentru utilizatorul caruia ii vom trimite fisierul
- metode :
  - setFilename(String filename) : seteaza numele utilizatorului catre care se face upload
  - getFilename : intoarce numele utilizatorului catre care se face upload
  - setUserInfo(User userInfo) : setaza campul useFor cu datele din userInfo
  - getUserInfo() : intoarce un obiect de tip User , cu datele utilizatorului catre care vom trimite fisierul

**h) User**

- campuri:
  - username(String ) : reprezinta numele utilizatorului
  - IP(InetAddress) : contine adresa IP a utilizatorului
  - port (int): specifica portul folosit pentru a ne conecta la aplicatia client a acestui utilizator
- metode:
  - GetUsername() : intoarce un String cu numele utilizatorului
  - SetUsername(String username) : seteaza campul username cu parametrul primit
  - GetIP() : inoarce un obiect de tip InetAddress care contine adresa IP a utilizatorului
  - SetIP(InetAddress IP) : seteaza adresa IP a utilizatorului
  - GetPort() : intoarce portul utilizatorului
  - SetPort(Integer): seteaza portul utilizatorului

## B. Modulul Server

Aici sunt incluse toate clasele necesare implementarii aplicatiei pentru server.

Acest va interactiona cu modulul Client.

Acesta va include urmatoarele clase:

**a) Server :**

- campuri :
  - UsersList (LinkedList) <User> ) : lista utilizatorilor online
  - FileLists (HashMap<String , <LinkedList<FileInfo>> >) : cheia reprezinta numele utilizatorului iar valoarea este o lista de obiecte FileInfo ce descriu fisierele distribuite de utilizatorul respectiv
- metode :
  - registerClient(String username,InetAddress IP,int port) :
    - - metoda verifica daca username exista deja in sistem
    - - daca da , atunci va intoarce valoarea False
    - - daca nu , folosind parametri primiti va crea un nou obiect de tip User pe care il va adauga in UsersList , iar apoi va intoarce True
  - search (String filter) :
    - - cauta prin filelist-urile tuturor utilizatorilor online o potrivire pentru sirul primit ca parametru (filter)
    - - intoarce o lista de fisiere care fac match cu sirul de cautare si care au distanta Livenstein <= 5 fata de acesta
  - updateFilelist(String username,LinkedList<FileInfo> filelist): primeste ca parametri numele utilizatorului a carui lista de fisiere partajate din FileLists va fi reactualizata cu cea primita ca parametru(filelist)
  - removeUser(String username): primeste ca parametru numele utilizatorului pe care il va sterge din UsersList

**b) FileInfo :**

- campuri:
  - path(String) : un sir ce reprezinta calea catre fisier
  - isDir (Bool) : ba fi True daca fisierul este de tip director si False daca nu
  - content (LinkedList<FileInfo>) :daca fisierul nostru este director va contine o lista de obiecte FileInfo cu informatii despre lista de fisiere distribuite(daca nu este director , atunci lista va fi goala)
  - type (String): reprezinta tipul fisierului
    - daca este director va fi „directory”
    - altfel va fi egal cu extensia fisierului
  - size (BigInteger) : dimensiunea fisierului in Bytes
- metode :
  - getPath(): intoarce un String ce contine calea catre fisier

- setPath (String path): va seta campul path cu calea catre fisier
- setDir(): seteaza campul isDir la valoarea True
- clearDir(): seteaza campul isDir la valoarea False
- getContent() : intoarce o lista cu obiecte de tip FileInfo cu informatii despre lista de fisiere din director
- setContent (LinkedList<FileInfo> content) : adauga in lista “content” obiectele FileInfo din lista primita ca parametru
- getType(): intoarce un String cu tipul fisierului
- setType(String type) : seteaza tipul fisierului
- setSize(BigInteger size) : seteaza dimensiunea fisierului
- getSize() : intoarce un BigInteger cu dimensiunea fisierului

**c) User :**

- campuri:
  - username(String ) : reprezinta numele utilizatorului
  - IP(InetAddress) : contine adresa IP a utilizatorului
  - port (int): specifica portul folosit pentru a ne conecta la aplicatia client a acestui utilizator
- metode:
  - GetUsername() : intoarce un String cu numele utilizatorului
  - SetUsername(String username) : seteaza campul username cu parametrul primit
  - GetIP() : inoarce un obiect de tip InetAddress care contine adresa IP a utilizatorului
  - SetIP(InetAddress IP) : seteaza adresa IP a utilizatorului
  - GetPort() : intoarce portul utilizatorului
  - SetPort(Integer): seteaza portul utilizatorului

### C. Modulul Interfata

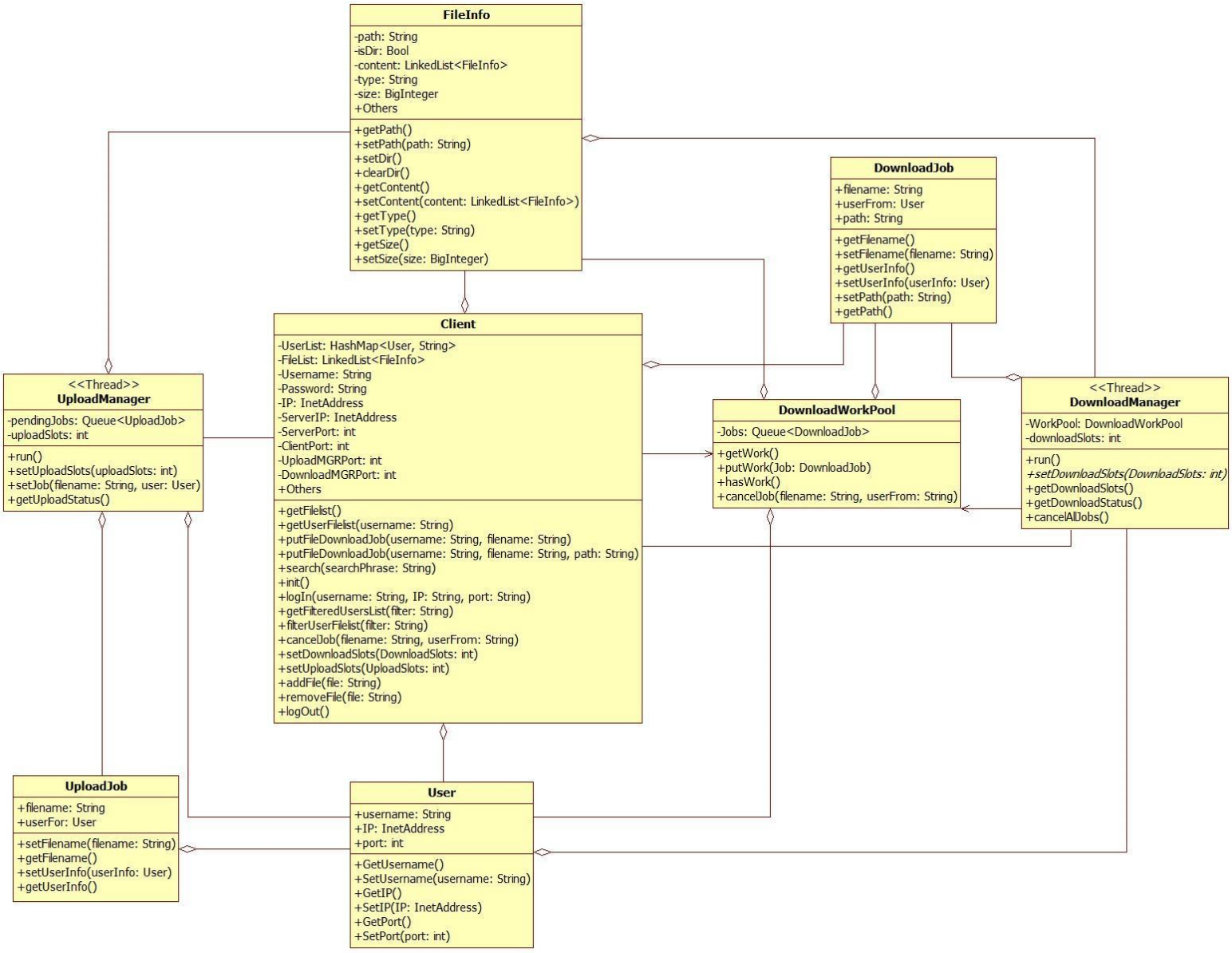
In acest modul vor fi include toate clasele necesare implementarii interfetei grafice(GUI) pentru utilizator.

Acest modul va interactiona in mod direct cu modulul Client.Pentru fiecare caz de utilizare a aplicatiei de catre client , interfata va apela metode din clasele modulului Client.Schimbul de informatii dintre cele doua module va fi bidirectional , interfata poate sa trimita si in acelasi timp , sa primeasca date de la Cllient.

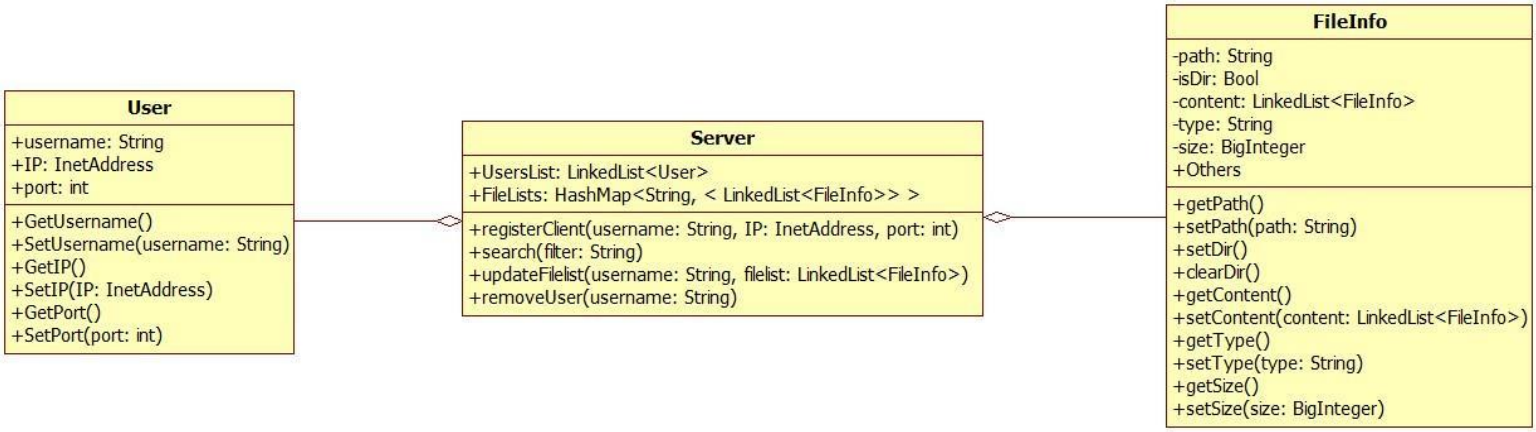
Ca exemplu , putem sa presupunem ca ne aflam in fereastra de start a aplicatiei si ca utilizatorul doreste sa se inregistreze.El va completa campurile necesare autentificarii si va apasa butonul de <Login>.In acest moment , in spate , interfata va apela metoda logIn din cadru clasei Client , care se va ocupa de restul procesului.

## 3. Diagrame de Clase

## 3.1 Aplicatia client




## 3.2 Aplicatia Server




## 4. Realizarea functiilor din specificatie

Precizari:

- clasa Client se ocupa de comunicatia cu server-ul si are posibilitatea sa ceara lista de fisiere a altui utilizator prin conexiune directa;

- la Log In, clasei Server ii sunt trimise de catre client informatiile necesare pentru conectarea la clasa UploadManager;
- operatiile de descarcare fisier si descarcare lista fisiere a unui utilizator sunt vazute ca operatii de upload de catre clientul destinatie si sunt procesate de UploadManager; acesta este motivul pentru care la Log In clasei server ii sunt trimise informatiile despre UploadManager;
- clasa DownloadManager se ocupa exclusiv de descarcarea de fisiere si comunica cu clasa UploadManager a celuilalt client;
- clasele UploadManager si DownloadManager implementeaza logica de download respectiv upload tinand cont de numarul de sloturi alocate;

### 4.1 Log In

Secventa de operatii este urmatoarea:

1. Interfata asteapta ca user-ul sa completeze toate campiurile necesare:
    1.1. username
    1.2. ip server;
    1.3. port server;
2. Din interfata se apeleaza metoda logIn() a clasei Client;
3. In metoda logIn() a clase Client:
    3.1. se apeleaza metoda init() care incarca de pe disc lista de fiere partajate;
    3.2. se creeaza o conexiune la server;
    3.3. se transmite un pachet de date in care specifica tipul operatiei(login) username-ul ales, IP-ul si portul UploadManager-ului;
    3.4. se asteapta raspuns de la server;
4. Clasa Server accepta conexiunea si apeleaza metoda registerClient();
5. In metoda registerClient():
    5.1. se verifica daca username-ul exista deja in sistem:
    5.2. daca deja exista metoda intoarce false;
    5.3. deca nu exista:
        5.3.1. se construieste un obiect de tip User caruia ii sunt setate campurile username, IP si port la valorile furnizate de client;
        5.3.2. se adauga obiectul listei de utilizatori;
        5.3.3. se intoarce true;
6. Clasa Server trimite un pachet de date clasei Client care contine true sau false, ceea ce intorce metoda registerClient si inchide conexiunea;
7. Clasa client primeste raspuns:
    7.1. daca raspunsul este true atunci utilizatorul a fost autentificat cu succes si se poate trece mai departe;
    7.2. daca raspunsul este false atunci se informeaza interfata ca datele introduse nu au fost corecte;
    7.3. inchide conexiunea;

### 4.2 Filtrarea listei de utilizatori

Fluxul operatiilor este urmatorul:

1. Interfata preia sirul dupa care se face filtrarea si apeleaza metoda getFilteredUsersList() a clasei Client, specificand sirul de filtrare ca parametru;
2. In medota getFilteredUsersList() se parcurge lista de utilizatori online si se construieste o lista cu utilizatori ai caror unsername-uri incep cu sirul dat ca filtru;
3. Lista creata anterior este intoarsa de metoda si preluata de interfata care o afiseaza;

### 4.3 Cautarea prin fisierele partajate de toti utilizatorii online:

Flux de operatii:

1. Interfata preia string-ul introdus in casuta de cautare si apeleaza metoda search() a clasei Client, oferind sirul de cautare ca parametru;
2. In cadrul metodei search din clasa Client:
    2.1. se stabileste o conxiune cu server-ul;
    2.2. se trimite un pachet de date care contine tipul operatiei(search) si sirul dupa care se va realiza cautarea;
    2.3. asteapta sa primeasca lista de obiecte de tip FileInfo care descriu fisiere ce se potrivesc cu sirul de cautare; daca lista este vida atunci inseamna ca nu s-a gasit niciun fisier care sa se potriveasca cu sirul dat;
    2.4. inchide conexiunea;
    2.5. ofera aceasta lista interfetei pentru a o afisa utilizatorului;
3. La nivelui clasei Server:
    3.1. se accepta conexiunea de la Client;
    3.2. se receptioneaza pachetul de date si se apeleaza metoda search(), oferind sirul de cautare primit ca parametru;
    3.3. metoda search cauta prin filelist-urile tuturor utilizatorilor online si returneaza o lista de fisere care respecta distanta Levenshtein, mai mica sau egala cu 5, fata de sirul primit;
    3.4. aceasta lista este trimisa aplicatiei client;
    3.5. se inchide conexiunea;

### 4.4 Descarcarea listei de fisire a unui utilizator

Clasele interactioneaza astfel:

1. Intefata apeleaza metoda getUserFilelist() a clasei Client oferind ca parametru numele utilizatorului selectat;
2. Metoda getUserFilelist() a clasei Client:
    2.1. preia informatiile de conectare la UploadManager-ul clasei Client a carui lista de fisiere se doreste;
    2.2. deschide o conexiune cu UploadManager-ul;
    2.3. trimite un pachet de date care specifica tipul cererii - get filelist;
    2.4. primeste lista de fisiere de la UploadManager si o salveaza local;
    2.5. inchide conexiunea;
    2.6. trimite lista de fisiere primita intefetei pentru afisare;
    2.7. UploadManager-ul clasei Client destiantie:
    2.8. accepta conexiunea;
    2.9. verifica tipul operatiei si observa ca este o cerere pentru lista de fisiere;
    2.10. deoarece acest tip de cerere este procesat imediat, raspunde trimitand lista de fisiere proprie;
    2.11. inchide conexiunea;

### 4.5 Cautarea prin fisierele partajate a unui utilizator

Cand aceasta operatie este lansata utilizatorul a fost deja selectat si lista de fisiere a acestuia a fost descarcata, salvata local si este afisata in sectiunea “File List”. In aceste conditii cautarea se face astfel:

1. Interfata apeleaza metoda filterUserFilelist() a clasei Client oferind ca parametru filtrul dupa care se va face cautarea;
2. metoda filterUserFilelist() a clasei Client:
    2.1. parcurge lista de fisere a utilizatorului selectat in prealabil si contruieste o noua lista care contine fisierele aflate la distanta Levenshtein mai mica sau egala cu 5 fata de sirul de filtrare primit;
    2.2. intoarce noua lista formata;
3. Interfata preia lista intoarsa de metoda getUserFilelist() si o afiseaza in in sectiunea “File List”;

### 4.6 Descarcare fisier

In momentul lansarii acestei operatii a fost efectuata operatia 4.3 sau 4.4 si exista o lista de fisere din care utilizatorul poate sa aleaga. Se executa urmatoarele:

1. Interfata apeleaza metoda putFileDownloadJob() a clasei Client oferind ca parametri numele utilizatorului de la care se va descarca fisierul, numele fisierului si optional calea unde acest fisier va fi descarcat;
2. Metoda putFileDownloadJob:
    2.1. creaza un obiect de tip DownloadJob si il insereaza in DownloadWorkPool prin apelarea metodei putWork();
    2.2. seteaza statusul job-ului ca pending;
3. Job-ul este preluat din DownloadWorkPool de catre DownloadManager, prin apelarea metodei getWork(), dupa ce se proceseaza toate job-urile adaugate inaintea lui sau se elibereaza un slot;
4. DownloadManager-ul :
    4.1. se conecteaza la UploadManager-ul clientului de la care se va descarca fisierul si trimite un pachet de date in care specifica tipul cererii(descarcare fisier) si numele fisierului care se doreste a fi descarcat;
    4.2. statusul job-ului este schimbat din pendind in connecting;
    4.3. in momentul in care se primeste un pachet de date cu tipul start transfer de la UploadManager, statusul job-ului este schimbat din connecting in downloading;
    4.4. primeste secvential pachete de date de la UploadManager care contin fragmente din fisierul dorit;
    4.5. la primirea unui pachet de tipul end of transfer statusul job-ului este schimbat din downloading in finished, si se elibereaza un slot;
    4.6. inchide conexiunea cu UploadaManager-ul;

5. UploadManager-ul clientului de la care se descarca fisierul:
    5.1. accepta conexiunea de la DownloadManager;
    5.2. primeste pachetul de date cu cererea, observa ca tipul este descarcare fisier si apeleaza metoda setJob() care creeaza un UploadJob si il pune in coada pendingJobs;
    5.3. cand se elibereaza un slot si UploadJob-urile adaugate anterior au fost procesate, job-ul este preluat din coada;
    5.4. trimite un pachet de date de tipul start transfer;
    5.5. trimite pachete de date care contin fragmente din fisierul dorit pana la trimiterea completa a fisierului;
    5.6. trimtie un pachet de date de tipul end of transfer pentru a anunta ca fisierul a fost trimis complet;

inchide conexiunea;

### 4.7 Anularea descarcarii unui fisier;

Secventa de pasi este urmatoarea:

1. Interfata apeleaza metoda cancelJob() a clasei Client oferind ca parametrii numele fisierului si user-ul de la care s-a cerut descarcarea; e nevoie sa fie specificat si user-ul pentru a elimina abiguitatea in cazul se descarca doua fisiere cu nume indentice de la doi useri diferiti;
2. In functie de statusul job-ului metoda cancelJob() se comporta astfel:
    - 2.1. job-ul are statusul pending:
        - 2.1.1. apeleaza metoda cancelJob() din DownloadWorkPoo;;
        - 2.1.2. metoda cancelJob() din DownloadWorkPool cauta job-ul in coada si il elimina;
    - 2.2. job-ul are statusul connecting sau downloading:
        - 2.2.1. se creaza o conexiune cu UploadManager-ul clientului de la care se descarca fisierul;
        - 2.2.2. se trimite un pachet de date care specifica tipul(cancel upload), numele fisierului si username-ul propriu pentru a identifica UploadJob-ul;
        - 2.2.3. UploadManager-ul clientului de la care se descarca:
            - 2.2.3.1. accepta conexiunea;
            - 2.2.3.2. primeste pachetul de date care indica tipul cererii ca fiind cancel job;
            - 2.2.3.3. cauta UploadJob-ul printre cele active sau in coada pendingJobs;
            - 2.2.3.4. trimte DownloadManager-ului un mesaj de tipul cancel job;
            - 2.2.3.5. elimina UplodJob-ul si inchide conexiunea asociata;
            - 2.2.3.6. inchide conexiunea;
        - 2.2.4. DownloadManager-ul:
            - 2.2.4.1. primeste pachetul de tip cancel job;
            - 2.2.4.2. elimina DownloadJob-ul;
            - 2.2.4.3. sterge fisierul asociat;
            - 2.2.4.4. inchide conexiunea;
    - 2.3. job-ul are statusul finished:
        - 2.3.1. informeaza interfata de statusul job-ului;

### 4.8 Utilizatorul adauga un fisier la lista fisierelor partajate

Secventa de pasi:

1. Interfata apeleaza metoda addFile() a clasei Client oferind ca parametru calea catre fisier;
2. metoda addFile() a clasei Client:
    2.1. aduna informatile despre tipul, dimesiunea etc. a fisierului;
    2.2. construieste un nou obiect de tip FileInfo care contine informatiile adunate anterior;
    2.3. adauga obiectul FIle la lista de fisiere proprie;
    2.4. se conecteaza la Server;
    2.5. trimte un pachet de date de tipul update filelist si username-ul propriu;
    2.6. asteapta confirmare de la Server;
    2.7. inchide conexiunea;
3. Server-ul:
    3.1. accepta conexiunea;
    3.2. identifica tipul cererii ca fiind update filelist;
    3.3. receptioneaza noul filelist;
    3.4. apeleaza metoda updateFilelist() care Inlocuieste filelist-ul user-ului cu cel nou;
    3.5. timite un pachet de date de confirmare;
    3.6. inchide conexiunea;

### 4.9 Utilizatorul sterge un fisier din lista de fisiere partajate

Secventa de pasi:

1. Interfata apeleaza metoda removeFile() a clasei Client oferind ca parametru calea catre parametrul selectat;
2. metoda removeFile() a clasei Client:
    2.1. cauta obiectul FileInfo asociat fisierului indicat  in lista de fisiere;
    2.2. elimina obiectul FileInfo gasit din lista de fisiere;
    2.3. se conecteaza la Server;
    2.4. trimte un pachet de date de tipul update filelist si username-ul propriu;
    2.5. asteapta confirmare de la Server;
    2.6. inchide conexiunea;
3. Server-ul:
    3.1. accepta conexiunea;
    3.2. identifica tipul cererii ca fiind update filelist;
    3.3. receptioneaza noul filelist;
    3.4. apeleaza metoda updateFilelist() care Inlocuieste filelist-ul user-ului cu cel nou;
    3.5. trimite un pachet de date de confirmare;
    3.6. inchide conexiunea;

### 4.10 Utilizatorul doreste sa se delogheze

Secventa de pasi:

1. Interfata apeleaza metoda logOut() a clasei Client;
2. metoda logOut() a clasei Client:
    2.1. apeleaza metoda cancelAllJobs() a clasei DownloadManager care va anula toate descarcarile active;
    2.2. salveaza intr-un fisier binar lista de fisere partajate;
    2.3. se conecteaza la server;
    2.4. trimtie un pachet de date cu tipul log out continand unsername-ul propriu;
    2.5. asteapta confirmare de la server;
    2.6. inchide conexiunea la server;
3. Server-ul:
    3.1. accepta conexiunea;
    3.2. identifica tipul cererii ca fiind log out si apeleaza metoda removeUser() care sterge user-ul din UsersList precum si lista de fisere asociata;
    3.3. trimite un pachet de date de confirmare;
    3.4. inchide conexiunea;
4. Interfata afiseaza utilizatorului tab-ul de log in;

## Legenda

In cadrul sectiunii de module , am folosit urmatoarea notatie pentru descrierea campurilor si metodelor din clase:

- NumeCamp (tipul de date al campului) : descriere camp
- NumeMetoda (TipParametru NumeParametru) : descriere functionalitate metoda