Document de Proiectare Arhitecturala

Tehnologii

Python

Am ales limbajul Python deoarece este un limbaj de nivel inalt ce ne va permite sa ne concentram pe functionalitatile aplicatiei si mai putin pe detaliile de implementare. De asemenea limbajul este suficient de matur incat sa ofere o colectie generoasa de frameworkuri, documentatie si support ce ne vor fi de folos in dezvoltarea proiectului. Aplicatia va fi compatibila cu python versiunea 2.x

Flask http://flask.pocoo.org/

Un microframework pentru Python. In general folosit pentru servere web, Flask permite o configurare usoara de rute HTTP ideea fiind ca fiecare client este in acelasi timp si un server. Folosind simple requesturi HTTP putem interoga clientii si putem folosi acelasi protocol pentru upload si download. Am ales acest framework in favoarea altora deoarece are un footprint mic si unii din membrii echipei l-au mai folosit.

PyGTK http://www.pygtk.org/

Un framework ce permite crearea de interfete desktop folosind GTK. PyGTK merge pe principiul “code once, deploy everywhere” insemnand ca este compatibil cu Windows, Linux si MacOS nefiind astfel nevoie de a porta codul intre diferitele platforme. Am ales acest framework in favoarea altora deoarece este foarte bine documentat si prezinta foarte multe exemple de cod.

Client

GUI

Clasa GUI va încorpora toate funcționalitățile care țin de interfața cu utilizatorul și va relaționa doar cu clasa Controller, căreia îi va transmite toate request-urile și de la care va prelua toate informațiile necesare. Clasa GUI va conține submodule pentru pagini și tab-uri, care vor fi realizate folosind template-uri. De asemenea, va conține un modul EventListener, care va ”asculta” la evenimentele generate de butoane sau alte elemente din interfață. Aceste evenimente sunt generate local, ca urmare a unor acțiuni ale utilizatorului curent. Pe lângă EventListener, va exista și un modul ViewHandler, care va avea rolul de a afișa informațiile necesare în interfață și de a actualiza aceste informații pe baza datelor care se află în clasa Model a aplicației.

1. Pages and Tabs
Paginile și Tab-urile vor fi generate folosind template-uri. Acestea au rolul de a transmite evenimentele produse în interfață controller-ului pentru a fi procesate și de a afișa date utile utilizatorului(listă utlizatori, listă fișiere partajate de utilizator, listă fișiere partajate de alți utilizatori, informații log-in/sign-in, profil utilizator, progres Upload/Download). Metode include:

- **createPage()**: Creează o pagină cu diferite specificații, în funcție de use-case.
- **createTab():** Creează un tab cu diferite specificații, în funcție de use-case.

2. EventListener
Acest modul va conține sistemul folosit pentru a ”asculta” la evenimentele produse în interfață, care se vor transforma în request-uri trimise clasei Controller.

3. ViewHandler
Modulul ViewHandler se va ocupa cu menținerea coerenței între Model și interfață. În cazul în care s-au modificat date în Model(doar clasa Controller poate face acest lucru), Controller-ul va notifica GUI-ul, care va modifica datele afișate pentru a fi consistente cu cele din model. Metode incluse:

- **updateMyFiles():** Actualizează lista fișierelor partajate de utilizatorul curent.
- **updateSearchResults():** Actualizează informațiile afișate în pagina de search.
- **updateProgress():** Actualizează informațiile legate de progresul de Upload/Download.
- **updateUserList():** Actualizează lista de utilizatori activi afișată în interfață.
- **showUserProfile():** Afișează informațiile de profil ale utilizatorului curent

CONTROLLER
Va coordona celelate module din structura Clientului.

- **updateUserList()**: solicita de la NI lista utilizatorilor autentificati
- **processUserList(userList)**: solicita GUI sa afiseze lista de utilizatori
- **searchFiles(filePattern)**: solicita de la NI fisierele care indeplinesc un anumit sablon

- **searchFilesFromUser(filePattern, user)**: solicita de la NI fisierele care indeplinesc un anumit sablon si apartin utilizatorului specificat
- **fetchFileListFromUser(user)**: solicita de la NI lista de fisiere a utilizatorului specificat
- **downloadFile(filePath, user)**: solicita de la NI descarcarea fisierului avand calea si utilizatorul specificati
- **signIn(username, password)**: solicita NI sa execute autentificarea
- **signInResponse(response)**: actualizeaza GUI conform raspunsului la cererea de autentificare
- **signOut()**: solicita NI sa execute deautentificarea
- **changeUsername(newUsername)**: solicita NI sa trimita o cerere de modificare a username-ului
- **uploadFile(request)**: ia in considerare cererea de upload primita de la NI
- **downloadFile(request)**: ia in considerare cererea de download primita de la GUI
- **setUploadSlots(newUploadSlotCount)**: seteaza o noua valoare pentru numarul de sloturi de upload
- **shareFile(filePath)**: se adauga fisierul specificat la lista fisierelor partajate
- **unshareFile(filePath)**: se scoate fisierul specificat din lista fisierelor partajate
- **searchForUsername(pattern)**: se cauta utilizatori in lista utilizatorilor autentificati

Se foloseste de modulul GUI pentru a prezenta informatii utilizatorului si trateaza evenimentele pe care le genereaza acesta (spre exemplu, apasarea unui buton).
Utilizeaza modului NI pentru a trimite date la alti Clienti si la Server si trateaza solicitarile de la acestia.
Salveaza si solicita date in/din modulul Model.

MODEL

Structura de baza a diferitelor tipuri de date folosite in aplicatie.

Modelul **User** va contine date specifice fiecarui utilizator

- istoricul cautarilor
    - **searchResults -** rezultate cautare

    - **searchTimestamp -** data cautarii
    - **searchKeywords -** cuvinte cheie folosite
- preferintele utilizatorului
    - **serverIP, serverPORT -** server + port de conectare
    - **username -** username
    - **sharedFiles -** fisiere si directoare partajate
    - **noOfSlots -** numar de sloturi
    - **downloadDir -** director de descarcare
    - **logHistory -** perioada de log (cat timp va fi pastrat un log file)
- sesiune curenta (timestamp cu data ultimei autentificari)
    - **lastLogin**

Metode:

- logUserSearch(results, keyword);
- setConnectionDetails(serverIP, serverPORT);
  setUsername(username);
  addSharePath(path);
  setUploadSlots(slotsNumber);
  setDownloadPath(path);
  setLogTime(duration);
- setLoginTimestamp(timestamp);

Modelul **Transfers** contine informatii despre fisere in curs de descarcare

- **filename -** numele fisierului
- **downloadPath -** locatia in care se realizeaza descarcarea
- **owner** - numele utilizatorului de la se realizeaza descarcarea
- **progress** - progres
- **client -** numele utilizatorului care primeste fisierul

Metode:

- Transfer(fileName, userIP, userPORT, userName); *//contructor*
  locatia pentru descarcare va fi luata din preferintele utilizatorului care a realizat operatiunea
- updateProgress(value);
- getProgress();

Modelul **SearchResults** contine informatii despre diferitele rezultate ce se obtin in urma cautarilor atat prin functionalitatea de search cat si prin descarcarea listei de fisiere a unui utilizator

- **searchResultType -** tip rezultat (cautare / lista fisiere)
- rezultate
    - **filename -** nume fisier

    - **filetype -** tip(format)
    - **filesize -** dimensiune fisier
    - **owner -** utilizator proprietar

Metode:

- Results(type, fileList); *//constructor*
  va realiza parsarea listei de fisiere si popula campurile mentionate

NETWORK INTERFACE:

Parametrii:
Status = indica daca cererea s-a indeplinit cu succes
Mesaj = reprezinta mesajul care este afisat daca este necesar
Observatii:
Comunicarea intre metodele de tip listener are loc doar intre NI-ulire a doi clienti sau client - server.

0. Deschidere conexiune client server:
    0.1. **sendConnectToServerRequest(portClinet, ipClient, portServer, ipServer):** NI Clientului trimite NI Serverului o cerere de deschidere de conexiune NIc -> NIs. Serverul poate fi si un alt client in cazul realizarii conexiunii peer-to-peer -> transferul de fisiere.
    0.2. **receiveConnectToServerRequest(status, message)**: indica daca conexiunea s-a realizat cu succes.
    0.3. **sendConnectToServerResponse(status, message)**: se trimite mesajul de eroare Controllerului, in cazul in care conexiunea nu s-a putut realiza.

1. SignUp:
    1.1. **signUpResponseListener(status, message, request):** Astepata raspunsul la o cerere de tip inregistrare.
    1.2. **sendSignUpRequest(serverIP, ServerPort, username, password, clientIP, clientPort):** Controllerul clientului curent trimite informatiile necesare pentru a se crea cererea de autentificare. Aceasta va fi trimisa catre interfata serverului.

2. SignIn:
    2.1. **signInResponseListener(status, message, request):** Asteapta ca interfata careia i s-a trimis cererea sa raspunda.
    2.2. **sendSignInRequest(serverIP, serverPort, username, password, clientIP, clientPort):** Controllerul clientului curent trimite informatiile necesare pentru a se crea cererea de log in. Aceasta va fi trimisa catre interfata serverului.

3. Search:
    3.1. **searchResponseListener(status, message, request):** Asteapta ca interfata careia i s-a trimis cererea sa raspunda.
    3.2. **sendSearchResponse(status, message, request):** Trimite raspunsul la cerere NI-lui care a facut requestul.
    3.3. **sendSearchRequest(<criterii filtrare,campuri de filtrat>):** creeaza o cerere specifica tipului de filtrare si a campului de filtrat. Cererea va fi trimisa catre server.

4. DownloadFile:
    4.1. **downloadFileRequestListener(request):** Asteapta cereri care constau in cererea de descarcare a unui fisier.
    4.2. **downloadFileResponseListener(status, message, request):** asteapta raspunsul la o cerere
    4.3. **sendDownloadFileResponse(status, message, request):** daca fisieul nu exista se trimite eroare. Daca fisierul exista, se va actualiza numarul de octeti care mai trebuie trimisi. Se vor trimite cereri pana cand fisieul este descarcat complet.
    4.4. **sendDownloadFileRequest(fileName):** creeaza o cerere de download pentru fisierul specificat
    4.5. **searchDownloadFileList(fileList):** pentru fiecare fisier din lista se apeleaza sendDownloadFileRequest(fileName).

5. Filelist:

5.1. **filelistRequestListener(username)**: cand vine o cerere de tip descarcare lista fisiere, pentru fiecare fisier se va crea o cerere de download fisier (punctul 4)

5.2. **filelistResponseListener(status, message, request)**: este primit dupa ce un fisier a fost primit in intregime. Procesul de descarcare a unei liste de fisiere se incheie cand s-au trimis complet toate fisierele.

5.3. **updateFilelist(filename)**

6. Activity:

6.1. **activityInfoResponseListener(status, message, request)**

6.2. **sendSctivityInfoRequest(username)**: se creeaza cererea pentru descarcarea istoricul de activitati al utilizatorului specificat. Requestul este trimis serverului.

6.3 **filterActivity(filter, username)**

7. LogOut:

7.1. **logOutResponseListener(status, message, request)**

7.2. **sendLogOutRequest(serverIP, serverPort, username, password, clientIP, clientPort)**

8. UsersList:

8.1. **sendUsersListRequest(filter)**

8.2. **usersListResponseListener(status, message, request)**

9. ChangeUsername:

9.1. **sendChangeUsernameRequest(newUsername, oldUsername)**

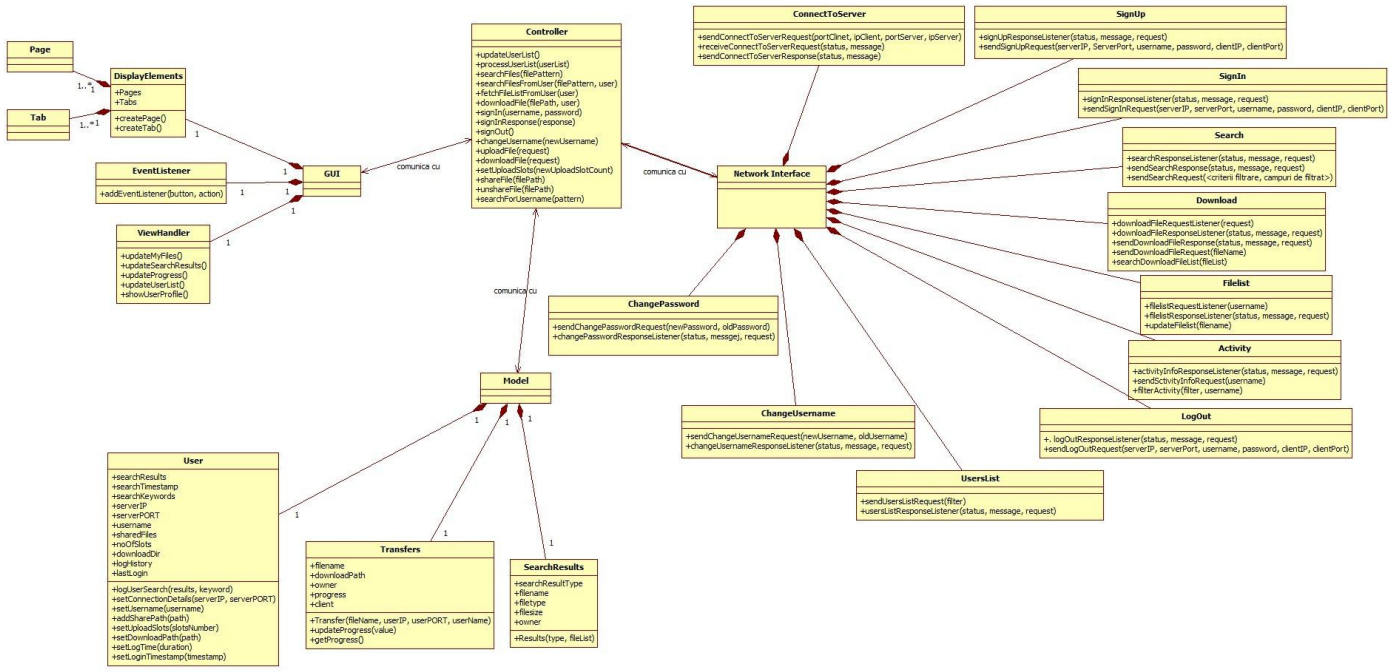
9.2. **changeUsernameResponseListener(status, message, request)**

10. ChangePassword:

10.1. **sendChangePasswordRequest(newPassword, oldPassword)**

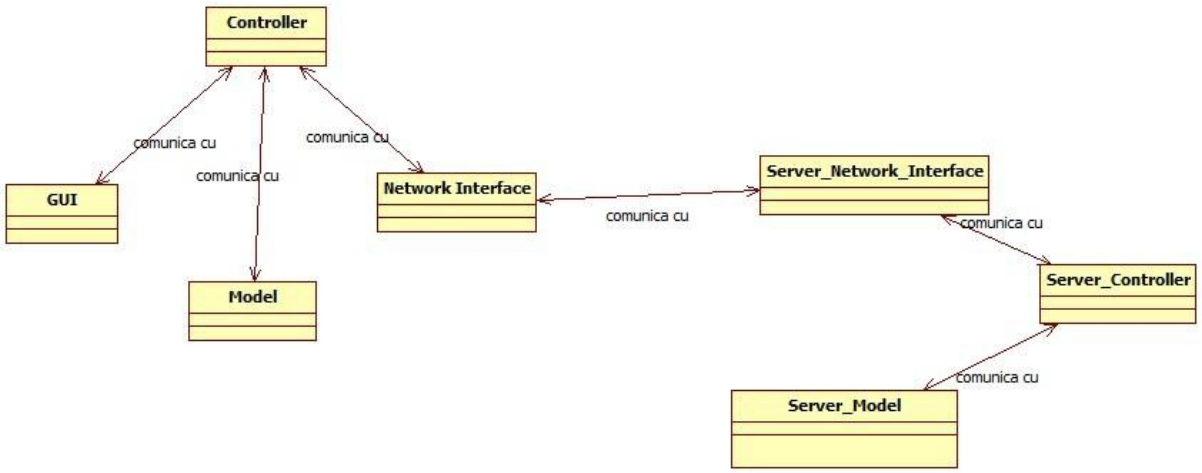
10.2. **changePasswordResponseListener(status, messgej, request)**

Diagrama de clase pentru Client:



Full size: http://i.imgur.com/mAu5JwI.jpg?1

Diagrama de comunicatie pentru Client:



Full size: http://imgur.com/apOlaz0.jpg

Server

1. ServerCore

- **initConnection(ipAddress, portNumber)**: initiaza conexiunea pe care asculta Server-ul
- **registerUser(username, password)**: inregistreaza un nou utilizator
- **updateUserInfo(oldUsername, newUsername, password)**: actualizeaza datele unui utilizator
- **processPingFromUser(username)**
- **processUserListRequest(username)**
- **signInUser(username, password)**

2. ServerModel

Va fi organizat sub forma unui dictionar avand drept cheie username-ul si drept valoare tripletul alcatuit din (password, ipAddress, portNumber)

3. Metode:

- **saveToFile(filePath)**: salveaza datele in fisier
- **loadFromFile(filePath)**: incarca datele dintr-un fisier

ServerNI

- **signInRequest(request)**
- **signOutRequest(request)**
- **sendSignInResponse(username, response)**
- **broadcastUserList(userList)**
- **sendPing(username)**: solicit ping de la un Client
- **pingReceived(username)**: am primit un ping (raspuns) de la un Client
- **userListRequest(username)**: am primit o cerere pentru lista de utilizatori
- **sendUserList(username)**

Diagrama Server: