Documentul de proiectare arhitecturala
Arhitectural Design Document (ADD)

Cuprins

3.5 Controlul accesului utilizatorilor la sistem

3.6 Fluxul global al controlului

3.7 Conditiile limita(cazurile de utilizare limită)

Glosar de termeni – daca este cazul

1. Introducere

1.1 Scopul sistemului

Sistemul permite utilizatorilor conectati intr-o retea locala sa puna in comun (partajare) documente si fisiere in functie de atributiile fiecaruia, sa caute fisierele dorite printre fisierele partajate de ceilalti utilizatori conectati la reteaua locala si sa aiba posibilitatea transferului sigur de informatii.

1.2 Definitii, acronime

BitBox - numele sistemului in curs de dezvoltare
Share - partajarea unui fisier sau orice alt tip de informatie
Lag - comunicatie intarziata
History - istoricul descarcarilor facute prin utilizarea sistemului BitBox
Uptime - timpul de disponibilitate a unui serviciu
Throughput - productivitate
Deployment - instalarea sistemului pregatit pentru productie
Chunk - bucata de bytes dintr-un fisier
Peer-to-peer - conexiune intre 2 clienti, fara a implica un server sau un al treilea mediator in comunicarea lor
Metadate - date despre date (ex: dimensiunea unui fisier este o metadata)
Environment - mediu de lucru

1.3 Documente aditionale si de referinta

Documentul de specificare a cerintelor (BitBox), ce este atasat la finalul portofoliului, contine date referitoare la descrierea mai in amanunt a scopului sistemului, cerintele functionale si nefunctionale, cat si diagrame ce reprezinta functionarea sistemului si interfata grafica minimala (interfata de referinta, nu prezinta toate detaliile grafice finale).

2. Obiectivele proiectarii

- Criterii de performanta
  - Timpul de raspuns: aplicatia trebuie sa nu dea senzatia ca raspunde incet (lagging)
  - Throughput: mare - aplicatia va putea descarca/incarca cat mai multe fisiere cat mai repede.
  - Consum de memorie: optimizat - sharing-ul de multe fisiere sau fisiere foarte mari nu trebuie sa incetineasca sistemul incarcand memoria

- Criterii de incredere
  - Fiabilitate: va fi oferita de dezvoltarea proprie a unui protocol de comunicare server-client
  - Disponibilitatea: serverul terbuie sa aiba uptime cat mai mare
  - Toleranta la defecte: in cazul deconectarii unui utilizator in timpul unui transfer, sistemul trebuie sa poata recupera
  - Securitate: sistemul trebuie sa ofere siguranta ca doar fisierele partajate pot fi descarcate de alti utilizatori; in plus cine nu are cont nu poate folosi aplicatia

- Criterii de cost
  - Costul dezvoltarii: patru saptamani

    - Cost de deployment: este nevoie de un calculator conectat la reteaua locala pentru a porni serverul si apoi trebuie instalata aplicatia pe calculatoarele (conectate in reteaua locala) utilizatorilor
    - Cost de upgrade: aplicatia se va folosi de resursele hardware existente ⇒ upgrade-ul necesita doar imbunatatirea hardware-ului (in principal RAM); sistemul nou va fi compatibil cu cel vechi in acest caz
    - Costul administrarii: o persoana autorizata (administrator) va fi responsabila de crearea conturilor

- Criterii de mentenanta
    - Claritatea codului: codul va fi comentat si scris intuitiv
    - Trasabilitatea cerintelor: cerintele trebuie sa se identifice usor in codul sursa sub forma de module

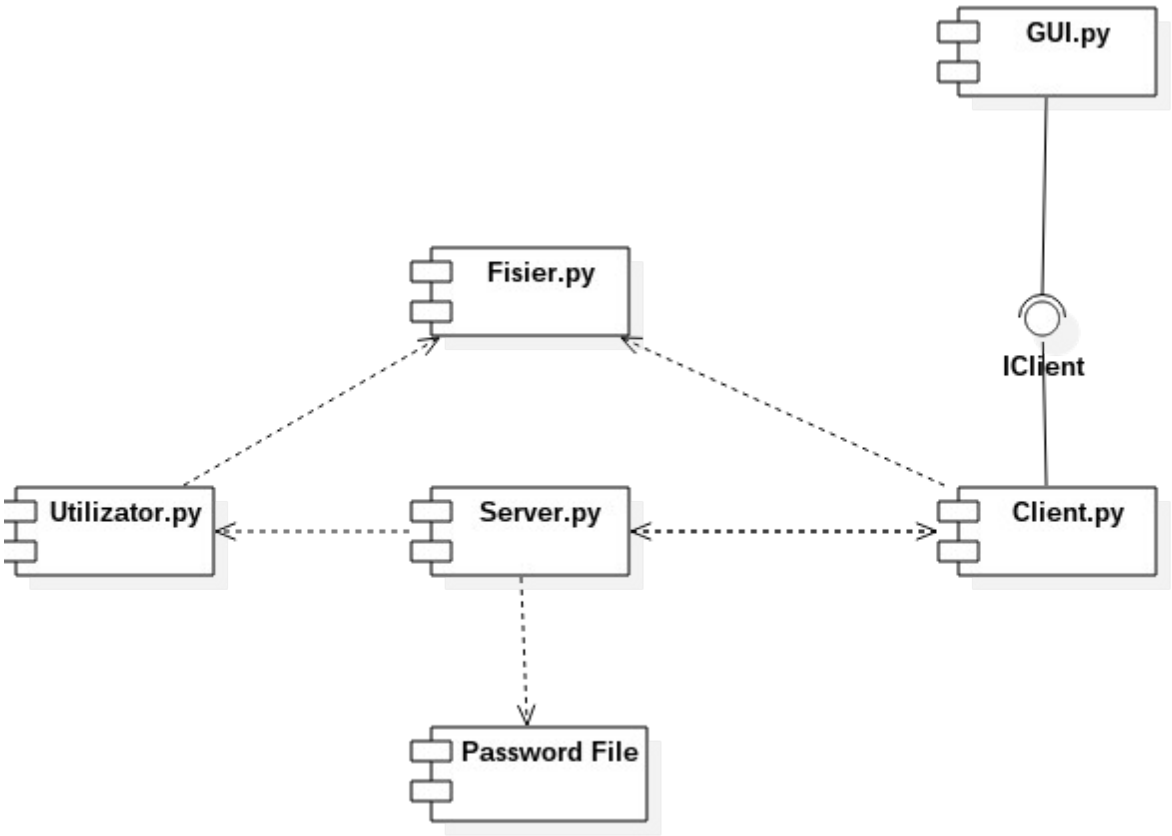
- Criterii ale utilizatorului final
    - Utilitatea: aplicatia va face partajarea fisierelor foarte usuoara
    - Usurinta de utilizare: in documentul de specificatii am explicat ca principalul scop al proiectului este simplitatea utilizarii; vrem sa oferim un user-friendly enviroment

3. Arhitectura propusa

3.1 Prezentare generala a arhitecturii sistemului

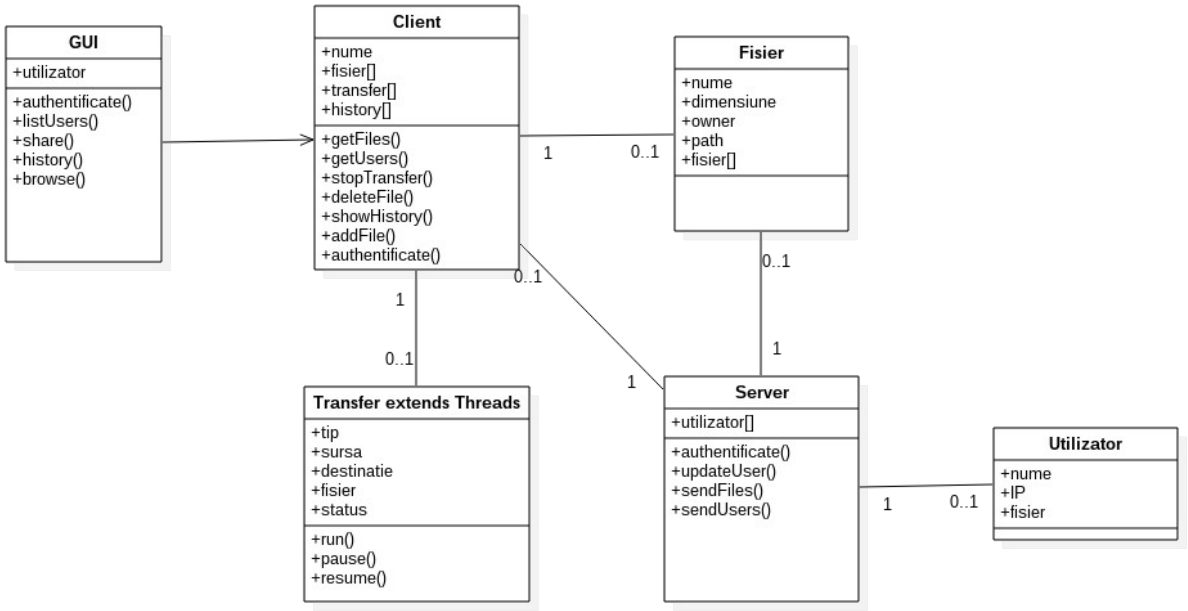
Arhitectura aleasa prezinta modelul clasic client-server pentru schimbul de metadate (liste de fisiere, ip-urile clientilor, numele clientilor, etc), iar pentru transferul efectiv de fisiere se va efectua o conexiune peer-to-peer astfel asigurand cea mai buna viteza de transfer.

3.2 Decompozitia in subsisteme si responsabilitatile fiecarui subsistem



Programul va fi scris in python si astfel vom avea un fisier GUI.py ce va contine codul pentru interfata grafica a aplicatiei.

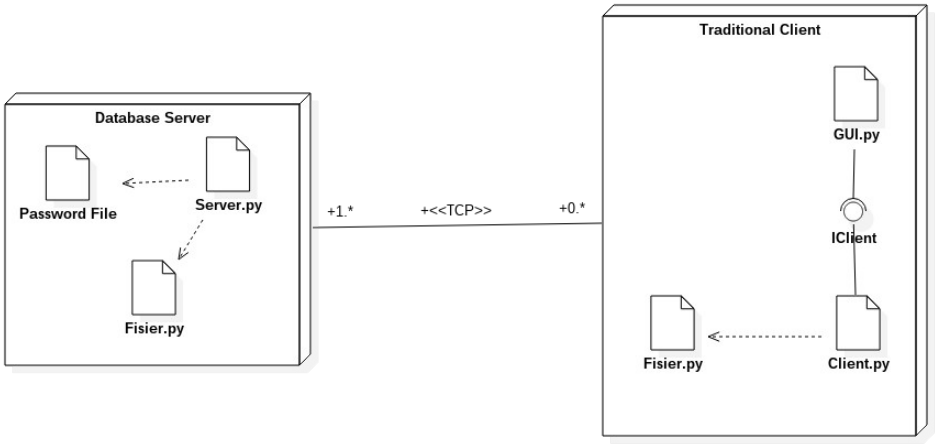
Numai fisierul Client va beneficia de interfata GUI, Serverul neavand o interfata grafica. Atat clientul cat si serverul sunt intr-o legatura de dependenta comunicand intre ele informatii legate de fisiere si utilizatori. Serverul are o dependenta atat de catre fisieul Password File ce detine datele utilizatorilor(nume utilizator si parola) ce sunt introduse de catre administrator pentru a se autentifica, dar si de catre fisierul Utilizator.py ce detine toate informatiile necesare ce le are un client conectat la server(nume, IP si lista de fisiere). Atat Utilizator.py cat si Client.py au o dependenta fata de fisierul Fisier.py ce e o clasa python ce va detine toate informatiile necesare pentru a partaja un fisier(numele, dimensiunea, cine il detine si tot asa).



Clasa GUI are un obiect utilizator, iar functiile GUI-ului sunt date de toate butoanele din interfata grafica precum butonul de connect pentru autentificate, butonul de List Users pentru a lista utilizatorii si fisierele acestora, si tot asa. Clasa Client are atributele: numele clientului, o lista cu fisierele lui partajate, o

lista transfer ce se va ocupa de trafnsferul fisierului de la un client la altul, o lista de history ce va detine lista cu toate fisierele descarcate de catre utilizator, dar si fisierele detinute de utilizator si descarcate de alti utilizatori. Clasa Transfer va extinde clasa Thread si se va ocupa de transferul fisierului, acesta are implementate functia de run ce va initia descarcarea si pause/resume pentru a opri temporar descarcarea/ a reporni descarcarea . Lista de transfer si history din clasa Client vor fi de tipul transfer, history fiind un transfer incheiat. Clasa Server va comunica cu clasa Client si informatiile primite le va introduce intr-un vector de utilizatori, vector de tip Utilizator ce va detine numele acestuia, IP-ul(necesar pentru transfer) si fisierele partajate de utilizatorul respectiv. Serverul se va ocupa de autentificarea utilizatorilor, reinoirea informatiilor acestuia, trimiterea listei de utilizatori si fiesierele partajate de un anumit utilizator, la cerere. Clasa Fisier va fi folosita atat pentru fisiere normale dar si foldere(ce sunt si ele fisiere). Din acest motiv clasa fisier are un vector de alte fisiere, in cazul in care este folder sa se poata acesa si acele fisiere. Mai are deasemena si numele fisierului, dimensiunea acestuia si calea unde se afla in calculatorul utilizatorului.

3.3 Distributia subsistemelor pe platforme hardware/software



Pentru aplicatia noastra, vom avea un server, la care mai multi clienti se vor conecta prin TCP. Serverul va fi reprezentat de catre un calculator central, pe cand clientii vor fi reprezentati de catre calculatoarele angajatilor.

3.4 Managementul datelor persistente

Sistemul BitBox nu necesita implementarea unei baze de date. Datele persistente rezultate in urma executiei aplicatiei vor fi reprezentate de:

- Fisierul de logare din server - fisierul contine username-ul si parola fiecarui utilizator. Fisierul poate fi modificat doar de un administrator, cu scopul crearii/stergerii conturilor.
- Fisierele si documentele descarcate de un utilizator de la ceilalti clienti din retea.

Pentru logare, clientul va trimite o cerere catre server si va primi ca raspuns de la acesta daca logarea a fost valida, sau invers, invalida.
Pentru transferul unui fisier, sunt 4 etape (cereri):

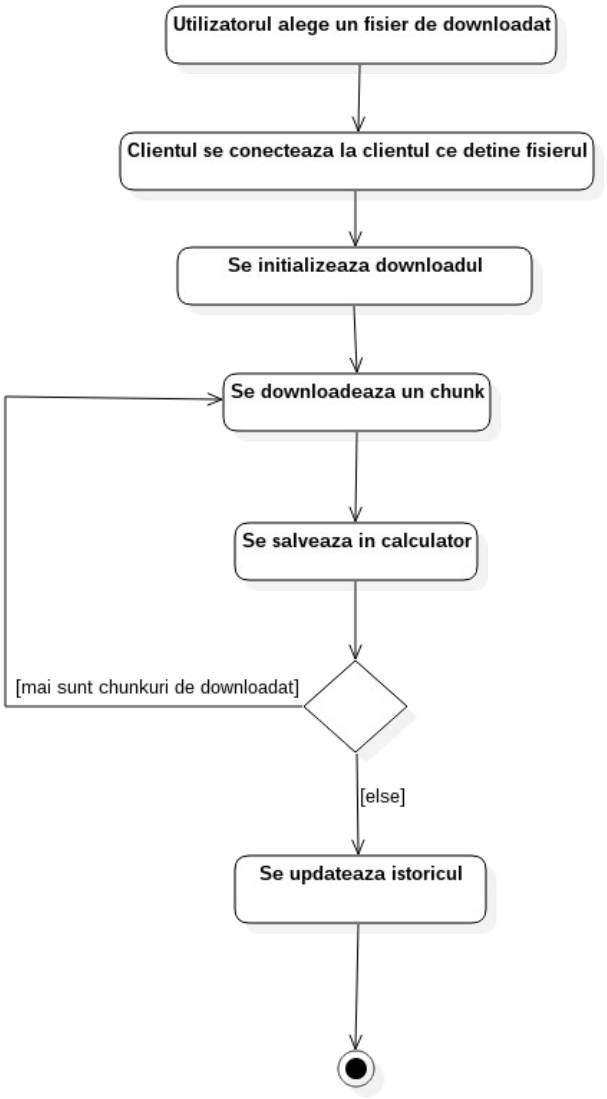
- Clientul1 trimite cerere catre server pentru aflarea IP-ului Clientului2 de la care vrea sa faca descarcarea

- Serverul ii trimite raspuns cu IP-ul cerut, daca exista
- Clientul1 trimite cerere de download catre Clientul2
- Clientul2 ii raspunde Clientului1 si apoi incepe sa trimita chunk-uri, folosind o conexiune peer-to-peer.

3.5 Controlul accesului utilizatorilor la sistem

Fiind un singur tip de utilizator (basic user), identificarea lui se face prin logarea cu username si parola personale. In absenta acestui criteriu de autentificare, un utilizator nu va putea folosi aplicatia.
Administratorul nu este vazut ca un utilizator. Aplicatia nu ofera interfata grafica pentru acesta. El are rolul de a introduce/sterge manual conturi de utilizatori din fisierul de logare din server.

3.6 Fluxul global al controlului



Utilizatorul isi alege fisierul pe care vrea sa il descarce dintr-o lista. Dupa ce il selecteaza, clientul se va conecta peer-to-peer la clientul ce detine fisierul si se va initializa descarcarea. Atata timp cat mai exista chunk-uri de downladat, se vor descarca si salva in calculator in locul selecat de la bun inceput. In final se updateaza istoricul si se incheie transferul.

3.7 Conditiile limita

Cel mai important caz limita este intampinat atunci cand sunt foarte multi clienti conectati in acelasi timp (limitat).
Numarul de clienti conectati la server depinde de arhitectura sistemului pe care opereaza serverul; in special numarul de socketi deschisi permisi de sistemul de operare.