Documentatie MySVN

Autor: Badita Alexandru-George

1 Introducere

MySVN este o aplicatie de tipul repository vision. Am ales sa fac aceasta aplicatie deoarece mereu cand am facut un proiect personal, mi-a placut la final sa am ceva despre care sa pot spune ca este util, ceva care sa fie ca o aplicatie pe care poate o folosesc chiar eu, dar la o scara mai mica, lucrul asta imi aduce o anumita satisfactie deoarece simt ca am cunostintele necesare pentru a face ceva palpabil, ceva care sa ajunga sa fie utilizat intr-o zi chiar de un utilizator obisnuit. MySVN m-a dus cu gandul la GitHub sau BitBucket, si atunci m-am gandit la cat ar fi de interesant sa dezvolt o astfel de aplicatie, ceva care sa semene cu una dintre cele doua platforme amintite mai sus, pe care le folosesc si eu si care sunt atat de utile atat de multor programatori sau creatori de continut, cat de multe as putea invata lucrand la un astfel de proiect, si cat de bine ar arta un proiect de genul MySVN in CV-ul meu personal.

2 Tehnologii Utilizate

2.1 TCP (Transmission Control Protocol)

Voi utiliza caracteristici precum bind sau listen pentru realizarea comunicarii intre mai multe procese.

Am ales sa utilizez aceasta tehnologie deoarece doresc sa existe o comunicare bidirectionala intre client si server, doresc ca datele sa ajunga la server si sa am o rata mica de esec la diferite operatii care ar putea veni din partea clientilor. Imi este cunoscuta structura unui server TCP, si consider ca pot dezvolta un sistem eficient de comunicare in cadrul aplicatiei mele bazandu-ma pe aceasta tehnologie.

Aceasta tehnologie imi ofera o cale de a gestiona clientii, utilizand numere de port, pot identifica fiecare client in parte pentru a ma asigura ca accesul la datele sale nu este garantat altcuiva in afara de el, si pentru a putea asigura in orice moment comunicarea intre server si client(trimiterea de mesaje din partea serverului in legatura cu consumarea spatiului de stocare disponibil, redirectionarea de mesaje private venite din partea altor client).

2.2 SQL(Structured Query Language)

SQL este un limbaj de manipulare a datelor care se foloseste in cadrul sistemelor de gestionare a bazelor de date relationale. Folosesc SQL deoarece imi este

cunoscuta sintaxa limbajului si sunt destul de familiarizat cu el, si consider ca il pot folosi pentru dezvoltarea unei logici bune de stocare a datelor utilizatorilor in cadrul aplicatiei mele.
Operatiile in cadrul bazelor de date SQL sunt rapide, si nu vor ingreuna timpii de asteptare pentru utilizatori. De asemenea, o astfel de baza de date este usor de configurat pentru a fi utilizata in cadrul aplicatiei mele.

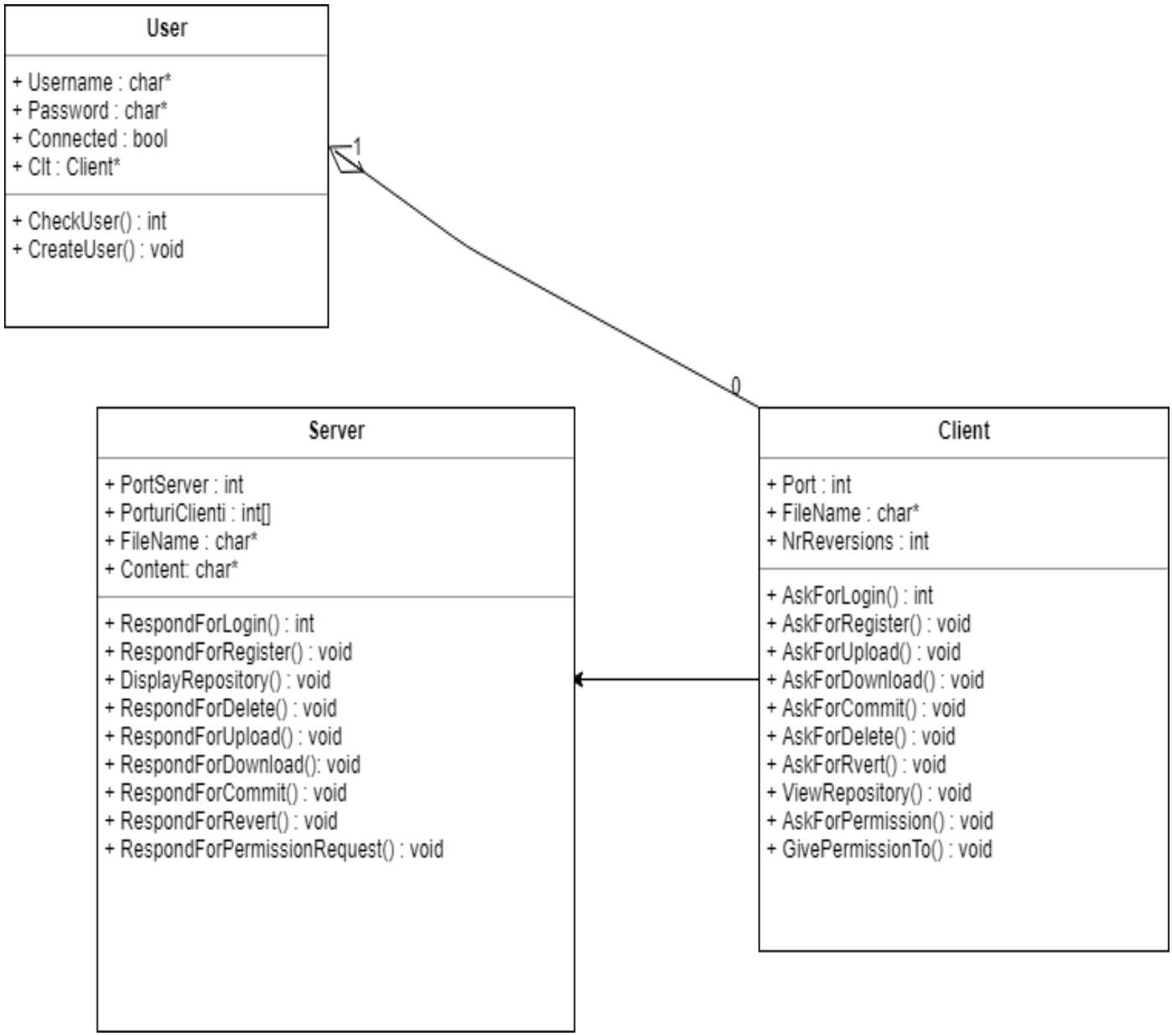
2.2 Threading Technology

Utilizez thread-urile din cadrul C/C++ pentru a asigura partea de concurenta din cadrul aplicatiei mele, a putea gestiona in acelasi timp mai multe cereri venite din partea clientilor si a putea executa simultan numeroase operatii, de asemenea, utilizez aceasta tehnologie pentru a gestiona cat mai bine scrierile simultane asupra aceleeasi zone de memorie, care pot avea loc.

3 Arhitectura Aplicatie

3.1 Model View Controller

Am folosit aceasta paradigma in cadrul proiectului meu pentru a izola in intregime logica din spatele aplicatiei mele de interactiunea dintre utilizator si aplicatie. Astfel, am partea de model care contine majoritatea functiilor ce se traduc in operatii pe care utilizatorul le poate face in cadrul aplicatiei, si logica din spatele gestionarii fisierelor, am partea de controlor care consta in paradigma client/server, aceasta parte asigura conectarea si deconectarea de la aplicatia efectiva, iar partea de vizualizare consta in interfata grafica facuta in QT cu care interactioneaza utilizatorul, utilizatorul poate interactiona de asemenea si cu terminalul.


User
+ Username : char*
+ Password : char*
+ Connected : bool
+ Clt : Client*
+ CheckUser() : int
+ CreateUser() : void
1
0
Server
+ PortServer : int
+ PorturiClienti : int[]
+ FileName : char*
+ Content: char*
+ RespondForLogin() : int
+ RespondForRegister() : void
+ DisplayRepository() : void
+ RespondForDelete() : void
+ RespondForUpload() : void
+ RespondForDownload(): void
+ RespondForCommit() : void
+ RespondForRevert() : void
+ RespondForPermissionRequest() : void
Client
+ Port : int
+ FileName : char*
+ NrReversions : int
+ AskForLogin() : int
+ AskForRegister() : void
+ AskForUpload() : void
+ AskForDownload() : void
+ AskForCommit() : void
+ AskForDelete() : void
+ AskForRvert() : void
+ ViewRepository() : void
+ AskForPermission() : void
+ GivePermissionTo() : void

4 Detalii de implementare

In cadrul proiectului MySVN avem clientul si serverul ca doua entitati separate iar pentru a elimina cozile de asteptare si blocajele, vom utiliza thread-uri, astfel vom avea concurenta garantata, serverul va putea satisfice cereri de la mai multi clienti in acelasi timp, si vom putea rezolva si cazul in care doi clienti modifica aceeasi zona de memorie simultan. Serverul va putea efectua urmatoarele comenzi:

- Upload FileName – incarca un fisier in repository-ul asociat contului pe care este logat utilizatorul. Fisierul este salvat in baza de date si poate fi accesat apoi oricand. Daca denumirea fisierului nu este gasita, atunci utilizatorul primeste o eroare care il anunta ca fisierul nu a putut fi gasit.
- Delete FileName – sterge un fisier din repository, fisierul este sters permanent din baza de date, si nu va mai putea fi vizualizat ulterior. Daca fisierul nu exista in repository, utilizatorul primeste un mesaj de eroare care il anunta ca nu exista un fisier cu aceasta denumire.
- Download FileName – descarca pe dispozitiv fisierul din repository, in cazul in care denumirea fisierului este gasita in repository. In cazul in care denumirea nu este gasita, utilizatorul primeste un mesaj de eroare.
- Commit FileName – se marcheaza faptul ca urmeaza sa se faca o modificare definitive asupra fisierului, iar aceasta sa devina publica pentru orice utilizator care il poate vizualiza. In cazul in care denumirea nu este gasita, utilizatorul primeste un mesaj de eroare.
- Revert FileName Number(int) – se revine la o alta versiune a fisierului, sunt preluate modificarile care au fost facute intre cele doua versiuni din cadrul fisierelor log in care retinem diferentele care survin de la o versiune la alta. Editam fisierul astfel, trecem iterativ prin toate diferentele care apar de la o versiune la alta, pana ajungem la versiunea la care dorim sa revenim. Pentru orice versiune i, vom avea in cadrul fisierelor log, elementele eliminate din versiunea anterioara si cele adaugate fata de versiunea anterioara, vom avea o structura de date in cadrul careia vom adauga elementele pe care le-am eliminate si vom elimina elementele adaugate, exact inversul operatiilor din fisierele log, astfel, la final vom avea in cadrul structurii exact modificarile pe care trebuie sa le facem pentru a reveni la o versiunea anterioara k. In cazul in care numele fisierului nu apare in repository sau nu exista o versiune aflata la atatea iteratii in spate, utilizatorul va primi un mesaj de eroare.
- ViewRepo – vizualizarea repository-ului, sunt afisate toate fisierele ce exista in repository, iar in dreptul fiecarui nume de fisier, este un contor care ne indica de cate ori a fost modificat fisierul de cand a fost adaugat in repository. De asemenea, putem vedea data ultimei modificari, si dimensiunea fisierului. In cazul in care repository-ul este gol, va fi afisat un mesaj in locul listei de fisiere.
- Open FileName – deschide un fisier in cadrul unui mod de vizualizare protejata, implementat in aplicatie, utilizatorul poate viziona continutul unui fisier fara a il descarca. In cazul in care denumirea fisierului nu este gasita, se va afisa un mesaj de eroare.

- Login username@password – utilizatorul se logheaza in aplicatie pentru a avea acces la repository-ul asociat unui anumit cont. Dupa logare poate face operatii asupra continutului repository-ului si poate vizualiza continutul acestuia, in cazul in care nu exista un cont care sa aiba numele sau parola introdusa de utilizator, acesta va primi un mesaj de eroare.

4.1 Cod relevant

4.1.1 Pseudocod Revert

```
//Parcurgem lista in care retinem operatiile pe care le-am facut de la o versiune la alta si codul asupra caruia am facut operatiile
for(i=curr;i>=r;i--) // curr este versiunea curenta, r este versiunea la care vem sa revenim
{
        if( operatie["elimina"]==1 )
        {
                revertFl+=instructiuni[i]; //daca am eliminat pentru a obtine noua versiune i, in drumul spre r trebuie sa facem operatia opusa
        }

        if( operatie["adaug"]==1 ) //in cazul opus, vom vrea sa eliminam orice set de instructiuni adaugat.
        {
                revertFl-=instructiuni[i];
        }
}
Fl=revertFl; //revertFl este versiunea la care dorim sa ajungem
```

4.1.2 Pseudocod Upload

```
Read from Socket;
PortSursa=Port(From) //salvam portul clientului de la care am primit fisierul
Insert FileName into DataBase Where TableName=ToString(Port) //inseram in tabela care are aceeasi denumire ca si portul clientului
if(!Error) Write("Reusit!") on Socket; //anuntam clientul ca a reusit upload-ul
else Write("Nereusit") on Socket; //anuntam clientul ca nu a reusit upload-ul daca este cazul
```

4.1.3 Download Pseudocod

```
#Partea de Server
prt=Port(from);
Select From ToString(prt) Where name=FileName //selectam din baza de date, fisierul cu denumirea FileName
```

```
if(errno) return "Nu Exista"+FileName; //tratam cazurile speciale ce pot aparea
Write FileName on Socket; //trimitem fisierul clientului care l-a cerut
#Partea de Client
Read FileName From Socket;      //obtine fisierul trimis de server
Create FileName On Device; //il salveaza in memoria dispozitivului de pe care este logat utilizatorul
```

4.1.4 ViewRepository Pseudocod

```
#Partea de Server
prt=Port(from);
content=Select * From ToString(prt); //selectam totul din baza de date asociata clientului
i=0;
while(i<content.size()){
        Write content[i].FileName on Socket; //trimitem denumirile fisierelor din baza de date, adica fisierele salvate in repository
}
#Partea de Client
 while( Exista informatie pe Socket ){
        Read FileName from Socket  //citim denumirea fiecarui fisier si o afisam apoi, la final vom avea lista denumirilor tuturor fisierelor din baza de date
        Display FileName+"\n";
 }
```

4.2 Use-Cases

- Conectare
- Inregistrare
- Mentinerea informatiilor utilizatorului
- Incarcare
- Commit
- Stergerea unui fisier
- Descarcare
- Vizualizarea continutului repository-ului
- Revenirea la o versiune mai veche a unui fisier

4.3 Descriere Use-cases

4.3.1 Inchiderea Inregistrarii

Inchiderea formularului de inregistrare va face ca utilizatorul sa fie adus inapoi la pagina principala, unde i se va cere sa se autentifice sau sa se inregistreze.

4.2.2 Inregistrarea

Completarea formularului de inregistrare va duce la salvarea datelor noului utilizator in baza de date a aplicatiei, apoi utilizatorul va fi directionat spre meniul de unde poate face operatii asupra continutului repository-ului sau.

4.2.3 Logarea

Orice utilizator trebuie sa se conecteze pentru a putea utiliza aplicatia. Logarea va directiona utilizatorul catre meniul de unde poate face operatii asupra repository-ului asociat contului sau,

4.2.4 Mentinerea informatiilor utilizatorului

Acest caz presupune gestionarea informatiilor pentru fiecare utilizator in functie de operatiile facute de acesta. Operatiile include inregistrarea, adaugarea unui fisier, descarcarea acestuia, stergerea, revenirea la o versiune anterioara. Aceasta parte este asigurata prin logica construita in server.

4.2.5 Incarcare

Acest caz presupune salvarea in cadrul repository-ului unui utilizator, un fisier ales de acesta. Pentru acest caz, actorul este clientul.

4.2.6 Commit

Acest caz presupune confirmarea faptului ca un fisier urmeaza sa fie modificat astfel incat modificarile facute asupra lui sa fie vizibile pentru oricine il poate vizualiza. Pentru acest caz, actorul este clientul.

4.2.7 Descarcare

Acest caz presupune salvarea in memoria computerului a unui fisier din cadrul repository-ului. Pentru acest caz, actorul este clientul.

4.2.8 Vizualizarea continutului repository-ului

Acest caz presupune vizualizarea continutului repository-ului asociat contului utilizatorul, mai exact toate fisierele care au fost incarcate, si datele la care acestea au fost incarcate sau modificate cel mai recent, de asemenea, fisierele modificate sunt marcate. Pentru acest caz, actorul este clientul.

4.2.9 Stergerea unui fisier

Acest caz presupune eliminarea din repository a unui fisier incarcat anterior de catre utilizator. Actorul pentru acest caz este utilizatorul.

4.2.10 Revenirea la o versiune anterioara

Acest caz presupune revenirea la o versiune mai veche a unui fisier, mai exact, vor disparea modificarile facute inainte de salvarea oricarei versiuni ce succeed versiunea la care dorim sa ne intoarcem. Pentru acest caz, actorul este clientul.

5 Concluzii

MySVN este o aplicatie cu un nivel ridicat de utilitate, motiv pentru care in prezent exista mai multe implementari la scara mai mare, elementele care fac MySVN o alegere viabila pentru persoanele care doresc sa depoziteze orice tip de materiale sunt faptul ca exista o sansa mica de esec in cadrul unei operatii de incarcare sau descarcare a fisierelor, cat si la alte operatii, este o aplicatie destul de rapida, cu timpi mici de asteptare, fisierele aflata intr-un repository pot fi vizualizate de mai multi utilizatori care au drepturi de acces la acel repository, si chiar editate, un aspect important pentru cazul in care este vorba de un proiect de echipa. MySVN are cateva limitari, mai ales in ceea ce priveste gestionarea si administrarea continutului unui repository. Proiectul de fata poate fi imbunatatit prin utilizarea unor algoritmi de revert cu o complexitate timp mai mica, prin utilizarea unei metode mai rapide pentru stocarea fisierelor decat SQL.

6 Referinte

- https://ro.wikipedia.org/wiki/Apache_Subversion
- http://subversion.apache.org/docs/
- http://tortoisesvn.tigris.org/
- https://en.wikipedia.org/wiki/Comparison_of_version_control_software
- https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php