Password Manager (C) - Raport Tehnic

Pâncă Aida-Gabriela, grupa A5

Facultatea de Informatică, Iași

Ianuarie 2025

1 Introducere

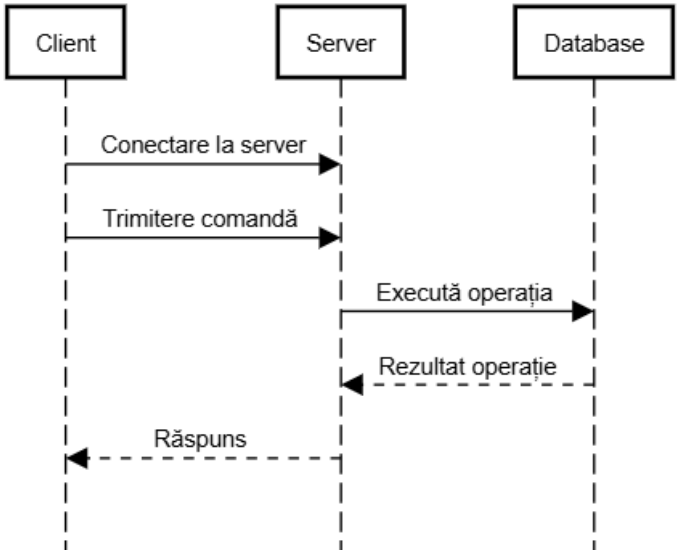
Proiectul Password Manager este o aplicație client/server care permite mai multor utilizatori să își gestioneze parolele într-un mod sigur și organizat. Utilizatorii pot crea conturi, se pot autentifica folosind o parolă master și pot salva parole organizate pe categorii. Aplicația oferă posibilitatea de a adăuga informații suplimentare pentru fiecare parolă, cum ar fi titlul, username-ul, URL-ul și notițe. Ei pot introduce/șterge categorii, introduce informații despre parole și să le modifice ulterior. De asemenea, include o funcționalitate pentru recuperarea parolei master, prin care utilizatorii pot seta și utiliza o întrebare de securitate pentru accesul în cazul în care au uitat parola master. Astfel, proiectul oferă utilizatorilor o modalitate de a-și organiza cu ușurință parolele.

2 Tehnologii Aplicate

SQLite3:
Folosită pentru a stoca toate datele proiectului, inclusiv conturile utilizatorilor, întrebările de securitate, categoriile și entitățile asociate. În `server.c`, operațiile de bază precum crearea tabelelor, salvarea și actualizarea datelor sunt realizate prin funcții specifice SQLite3 ceeaa ce asigură o integrare simplă și eficientă și permite acces rapid la date și stocare securizată.
Pthread:
Serverul gestionează fiecare client conectat printr-un thread dedicat. În `server.c`, biblioteca `pthread` permite crearea firelor de execuție pentru fiecare conexiune. Această implementare permite mai multor utilizatori să interacționeze simultan cu serverul.
Biblioteci de rețea:
Bibliotecile `sys/socket.h`, `netinet/in.h` și `arpa/inet.h` au fost utilizate pentru a configura și gestiona comunicarea între client și server. În `server.c`, funcțiile `socket`, `bind`, `listen` și `accept` inițializează conexiunile serverului, iar în `client.c`, funcția `connect` permite stabilirea unei conexiuni. Așadar, aceste biblioteci asigură transferul eficient al datelor între componente.
Protocol TCP/IP:
Aplicația utilizează protocolul TCP pentru o conexiune fiabilă între client și server. În `server.c`, serverul ascultă conexiunile pe un port dedicat (implicit 2500) și primește comenzi prin `recv`, iar clientul folosește `send` pentru a trimite cereri. TCP garantează livrarea completă și corectă a datelor, eliminând

riscurile de pierdere a informațiilor. Această fiabilitate este esențială pentru funcționalitățile de autentificare și gestionare a parolelor, deoarece asigură coerența și integritatea datelor transmise între client și server.

3 Structura Aplicației

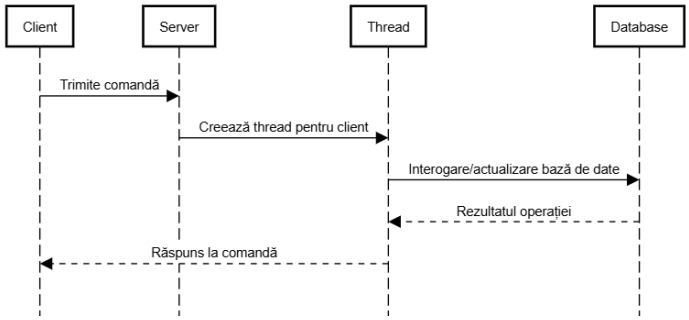


Structura aplicației este divizată în două componente principale ilustrate în diagrama de mai sus:

Server: gestionează operațiile principale ale aplicației, anume înregistrarea, autentificarea, organizarea datelor și gestionarea parolelor.

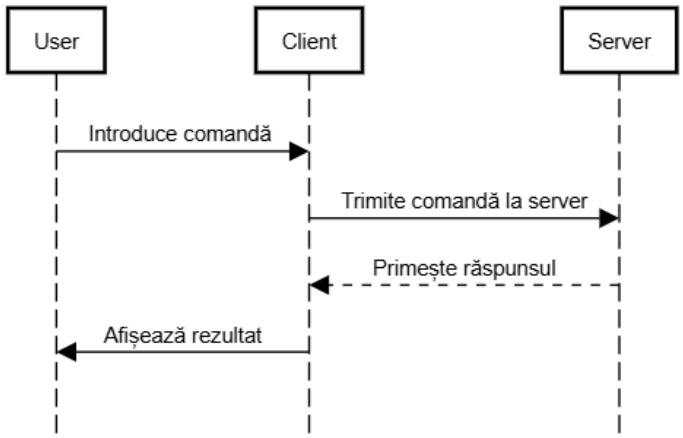
Clienți: oferă utilizatorilor o interfață text pentru introducerea comenzilor, care sunt procesate ulterior de server.
Pentru a detalia, avem următoarele diagrame detaliate:

3.1 Structura Serverului



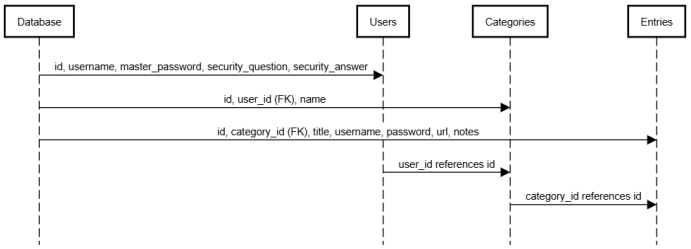
Serverul este responsabil pentru procesarea comenzilor primite de la clienți și gestionarea datelor în baza de date. Fiecare client conectat este gestionat printr-un **thread** dedicat, ceea ce asigură o scalabilitate ridicată. Diagrama evidențiază funcțiile principale implementate în `server.c`, precum gestionarea conexiunilor, operațiile cu baza de date și trimiterea răspunsurilor către clienți.

3.2 Structura Clientului



Clientul este interfața prin care utilizatorul interacționează cu aplicația. Comenzile introduse de utilizator sunt trimise către server prin intermediul protocolului TCP, iar răspunsurile sunt afișate utilizatorului. Structura clientului evidențiază funcțiile de conectare, trimitere a comenzilor și afișare a rezultatelor, toate implementate în `client.c`.

3.3 Structura Bazei de Date



Baza de date este componenta centrală pentru stocarea și gestionarea datelor aplicației. Structura acesteia include tabele pentru **utilizatori**, **categorii** și **entitati**. Fiecare tabel este conectat prin relații care facilitează accesul rapid și coerent la informații. De exemplu, tabelele `categories` și `entries` sunt legate prin chei străine de tabela `users`, ceea ce asigură că fiecare categorie și parolă aparțin unui utilizator valid.

4 Aspecte de Implementare

Arhitectura urmată este cea de client/server, fiecare componentă având un rol bine definit.

4.1 Funcționalități Implementate

Autentificare:
- Comenzi: `REGISTER`, `LOGIN`, `CHANGE_PASS`.
- Parola master este verificată la fiecare autentificare pentru a asigura securitatea utilizatorilor (adică să fie suficient de puternică).

Gestionare Parole:
- Adăugare categorii: `NEW_CAT`, `LIST_CAT`, `DEL_CAT`.
- Gestionare intrări: `NEW_ENTRY`, `MOD_ENTRY`, `DEL_ENTRY`.

Recuperare Parolă:
- Comenzi: `REGISTER_SEC`, `SEC_QUESTION`, `RECOVER_PAS`.
- Recuperarea implică răspunsul la întrebarea de securitate configurată de utilizator.

Interfața Client:
Clientul trimite comenzi specifice la server și afișează rezultatul utilizatorului. În Figura de mai jos sunt prezentate comenzile acceptate de aplicație, așa cum apar în terminal.

Fig. 1. Comenzile disponibile pentru client

4.2 Crearea Bazei de Date

Inițializarea bazei de date este realizată în `server.c`, folosind SQLite3. Figura de mai jos prezintă codul care creează tabelele necesare pentru utilizatori, categorii și intrări.

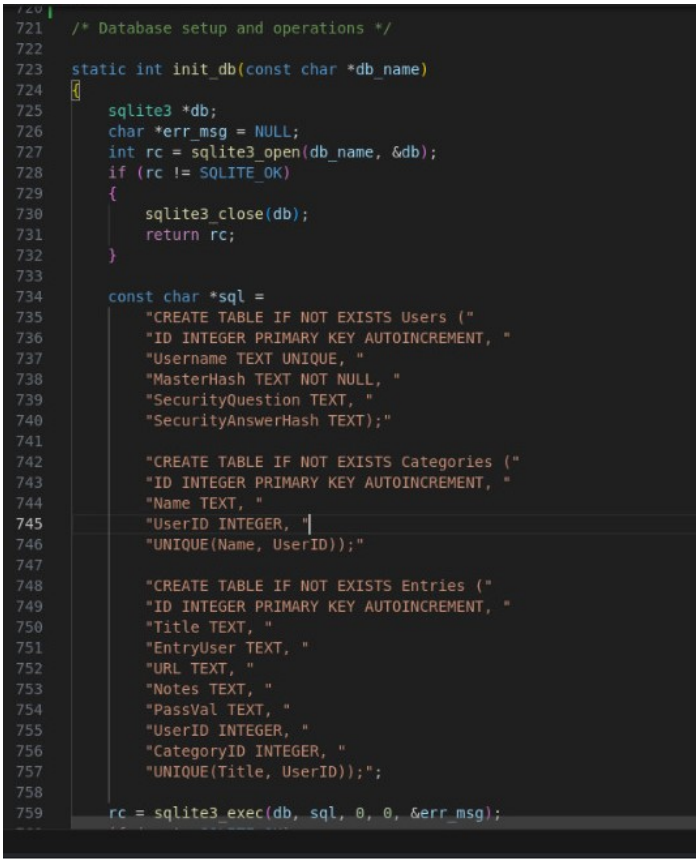


Fig. 2. Cod pentru inițializarea bazei de date

4.3 Secțiuni Relevante din Cod

Cod pentru comunicarea client-server:
Client:
Trimiterea comenzilor către server:

```
if (write(sd, buffer, strlen(buffer)) <= 0) {
    perror("Write to server failed.\n");
    break;
}
```

Primirea răspunsurilor de la server:

```
int recv_len = read(sd, buffer, sizeof(buffer) - 1);
if (recv_len < 0) {
    perror("Read from server failed.\n");
    break;
```

```
}
buffer[recv_len] = '\0';
printf("Server: %s\n", buffer);
```

Server:
Primirea comenzilor de la client:

```
int rbytes = read(ctx->client_fd, buffer, sizeof(buffer) - 1);
if (rbytes <= 0) {
    perror("Client read error.\n");
    break;
}
buffer[rbytes] = '\0';
```

Trimiterea răspunsurilor către client:

```
if (write(ctx->client_fd, response, strlen(response)) <= 0) {
    perror("Client write error.\n");
    break;
}
```

Aceste fragmente de cod evidențiază modul în care clientul și serverul comunică prin transmiterea comenzilor și răspunsurilor utilizând funcțiile `read` și `write`. Acest proces asigură un flux bidirecțional de date între cele două componente.

4.4 Thread-uri pe Server

Serverul creează un thread dedicat pentru fiecare client conectat, asigurând gestionarea eficientă a conexiunilor multiple. Aceasta permite utilizatorilor să acceseze și să utilizeze aplicația simultan, fără întreruperi.

5 Concluzii

Proiectul reprezintă astfel o aplicație client/server ce poate fi folosită pentru a organiza și gestiona parole în diferite categorii și cu diferinte informații suplimentare într-un mod sigur, folosind SQLite3 pentru stocarea datelor, pthread pentru conexiuni multiple și TCP/IP pentru transfer de date.

6 Referințe Bibliografice

[1] Majoritatea informației e de pe site-ul cursului de "Rețele de calculatoare" https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php

[2] M-am ghidat după exemplele și explicațiile de la laboratorul materiei (trimise pe Discord)

[3] Formatul după care am redactat Raportul Tehnic: https://www.springer.com/gp/computer-science/lncs/conference-proceedings-guidelines

[4] Site-ul utilizat ca să scriu în LaTeX:
https://www.overleaf.com/project
[5] Site-ul utilizat pentru realizarea diagramei aplicației:
https://sequencediagram.org/
[6] M-a ajutat cu unele neclarități de scriere în limbajul C:
https://www.geeksforgeeks.org/c-programming-language/
[7] Informații despre TCP:
https://www.geeksforgeeks.org/what-is-transmission-control-protocol-tcp/
[8] Informații despre socket-uri:
https://www.cs.rpi.edu/~moorthy/Courses/os98/Pgms/socket.html
[9] Informații despre baze de date SQLite:
https://www.geeksforgeeks.org/introduction-to-sqlite/