

Password Manager

Frateanu Tudor (X/3E2)
UAIC - Facultatea de Informatica

18th of January 2024

1 Introduction

"Password Manager" is a C/C++ client-server application that allows users to save their passwords for various websites and also allows to update and categorize said passwords. The server listens for incoming connections and processes client requests to register/login/logout/quit, also add/update/delete passwords in a database and last, to see those passwords. Those actions are sent as commands from the client to the server and the server processes them.

2 Used technologies

2.1 TCP

The Transmission Control Protocol (TCP) is a fundamental communication protocol in computer networks, providing reliable, connection-oriented communication between two devices. It operates at the transport layer of the Internet Protocol (IP) suite. TCP ensures the accurate and ordered delivery of data by establishing a connection through a three-way handshake (SYN, SYN-ACK, ACK) before data transfer begins. TCP is widely used for applications that demand guaranteed and sequential delivery of data, such as web browsing, email, and file transfers. Despite its slightly higher overhead compared to User Datagram Protocol (UDP), TCP is crucial for applications where data integrity and reliability are paramount.

TCP Components

Socket A socket serves as a bidirectional communication mechanism, facilitating interaction between processes on a computer, with a primary focus on enabling network communication. Each connection is uniquely identified by an IP address and port number pair. The IP address comprises four numbers between 0 and 255 separated by periods, while the port number ranges from 1 to 65535 (excluding 0, which is reserved for system use). Together, this IP address-port number pair constitutes a socket.

IP address It is a sequence of four numbers separated by dots, ranging from 0 to 255, which serves as an identifier for an entity within a computer network.

Port Is a 16-bit number that uniquely identifies services or applications within a computer system. Any application involved in network communication associates a port with its connection. Ports ranging from 0 to 1023 are specifically reserved for system processes.

2.2 C/C++ Libraries

The application uses several libraries such as: *sys/types.h*, *sys/socket.h*, *netinet/in.h*, *errno.h*, *unistd.h*, *stdio.h*, *string.h*, *stdlib.h*, *signal.h*, *pthread.h*, *stdint.h*, *sqlite3.h*, *netdb.h*. These libraries ensure functionality for socket programming, IO, multithreading and SQL database management.

2.3 SQL - SQLITE

The application uses *sqlite3.h* so that it provides a connection with a local database where the information about users is stored.

3 Architecture

The application follows a client-server architecture. The server listens for incoming connections on a specified port (port 2908) and creates a new thread for each client that connects to the server. The client connects to the server's IP address and port and interacts with the server by sending commands followed by data to the server. The server processes the commands, sends data (if it's the case) to the data base and sends a response to the client, where it can be visualised by the user in the terminal.

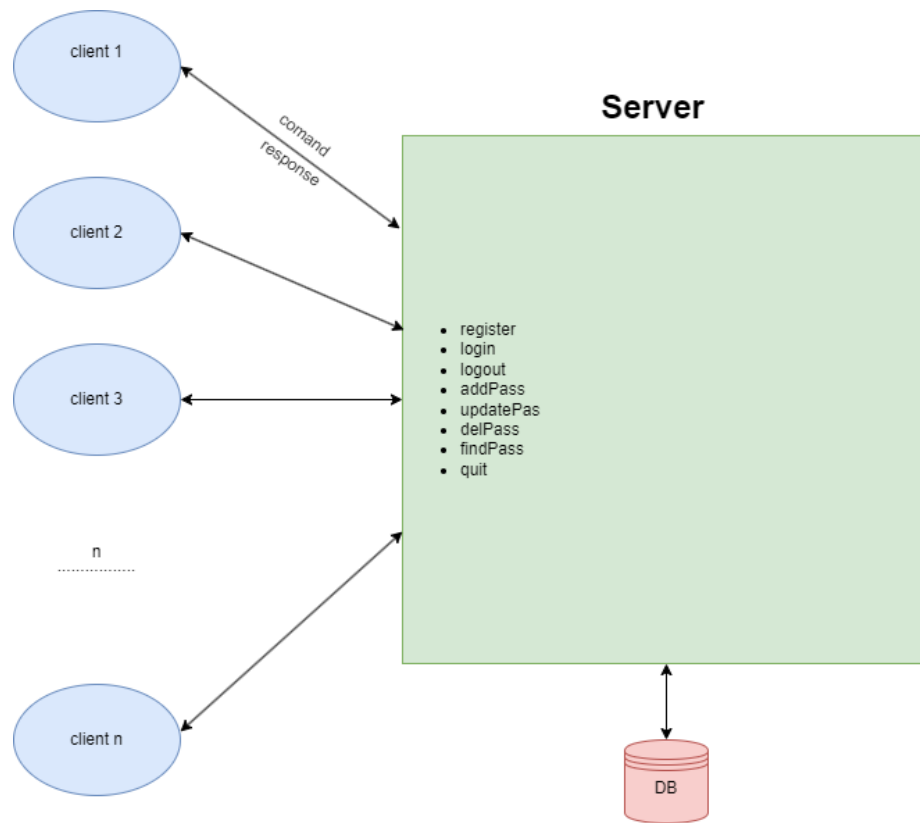


Figure 1: Architectural diagram.

Diagram of the architecture of the app

4 Implementation details

4.1 Server (servPassMan.c)

The servers lets multiple clients connect to it and handles them simultaneously using multithreading.

```
/* servim in mod concurrent clientii...folosind thread-uri */
while (1)
{
    int client;
    thData *td; // parametru functia executata de thread
    int length = sizeof(from);

    printf("[server]Asteptam la portul %d...\n", PORT);
    fflush(stdout);

    // client= malloc(sizeof(int));
    /* acceptam un client (stare blocanta pina la realizarea conexiunii) */
    if ((client = accept(sd, (struct sockaddr *)&from, &length)) < 0)
    {
        perror("[server]Eroare la accept().\n");
        continue;
    }

    /* s-a realizat conexiunea, se astepta mesajul */

    // int idThread; //id-ul threadului
    // int cl; //descriptorul intors de accept

    td = (struct thData *)malloc(sizeof(struct thData));
    td->idThread = i++;
    td->cl = client;

    pthread_create(&th[i], NULL, &treat, td);
} // while
```

Figure 2: This is how the server creates a thread for each client connected.

4.1.1 Functions

treat()

Description: Thread function that handles communication with a client.

raspunde()

Description: Function that processes client commands and generates responses.

`logic(comanda, raspuns, ok, toKeep)`

Description: Function that serves as a command router based on client input.

`regis(comanda, raspuns)`

Description: Function that handles client registration.

`login(comanda, raspuns, ok, toKeep)`

Description: Function that handles client login.

`addPass(comanda, raspuns, ok, toKeep)`

Description: Function that handles addPass command.

`findPass(comanda, raspuns, ok, toKeep)`

Description: Function that handles findPass command.

`updatePass(comanda, raspuns, ok, toKeep)`

Description: Function that handles updatePass command.

`delPass(comanda, raspuns, ok, toKeep)`

Description: Function that handles delPass command.

4.1.2 Database Operations

`openDB()`

Description: Opens the SQLite database.

`closeDB()`

Description: Closes the SQLite database.

`addUserDB(username, password, response)`

Description: Adds a new user to the database.

`checkCredentials(username, password, toKeep, response)`

Description: Checks user credentials for login.

`addPassDB(toKeep, category, website, username, password, response)`

Description: Adds a new password entry to the database.

`updatePassDB(toKeep, category, website, username, password, response)`

Description: Updates an existing password entry in the database.

`delPassBD(toKeep, website, username, response)`

Description: Deletes a password entry from the database.

`findByCatBD(category, response, toKeep)`

Description: Finds password entries by category in the database.

`findByWebBD(website, response, toKeep)`

Description: Finds password entries by website in the database.

4.1.3 Main Execution

`main()`

Description: Main function that sets up the server, listens for clients, and handles client requests.

4.1.4 Data Base

Uses sqlite for database management. The database is called PassManDB in wich we have 2 tables: 'users' and 'listaparole'.

users Table

In *users* table are stored 2 values for each entry: MASTusername(master username) and MASTparola(master password). MASTusername has type TEXT, set as NOTNULL and PRIMARY KEY so that no two users are the same. MASTparola has type TEXT and is set as NOTNULL.



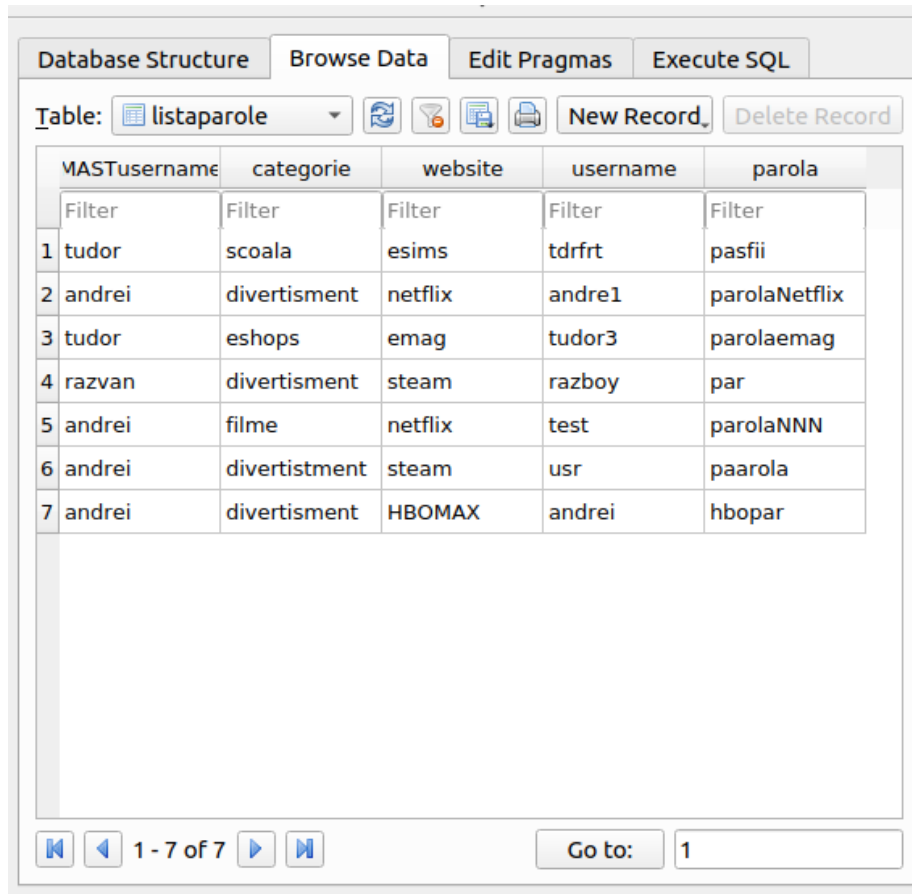
The screenshot shows a database management interface with tabs for 'Database Structure', 'Browse Data', 'Edit Pragmas', and 'Execute SQL'. The 'Browse Data' tab is active, showing the 'users' table. The table has two columns: 'MASTusername' and 'MASTparola'. There are four records listed, each with a number in the first column. The records are: 1 tudor parola, 2 andrei parola, 3 razvan parola, and 4 bigu parola. At the bottom, there are navigation buttons and a 'Go to:' field set to 1.

	MASTusername	MASTparola
1	tudor	parola
2	andrei	parola
3	razvan	parola
4	bigu	parola

Figure 3: User values stored in users.

listaparole Table

In *listaparole* table are stored five values for each entry: MASTusername(master username), categorie, website, username, parola. They store the masterusername, category, website, username and password. All of them are set as NOT-NULL.



The screenshot shows a database management tool interface. At the top, there are tabs: 'Database Structure', 'Browse Data' (selected), 'Edit Pragmas', and 'Execute SQL'. Below the tabs, there's a 'Table:' dropdown menu set to 'listaparole'. To the right of the dropdown are icons for refresh, filter, export, and print, along with 'New Record' and 'Delete Record' buttons. The main area displays a table with 5 columns: 'MASTusername', 'categorie', 'website', 'username', and 'parola'. Each column has a 'Filter' button above it. The table contains 7 rows of data, numbered 1 to 7. At the bottom, there are navigation buttons (first, previous, next, last) and a 'Go to:' field with the value '1'.

	MASTusername	categorie	website	username	parola
	Filter	Filter	Filter	Filter	Filter
1	tudor	scoala	esims	tdrft	pasfii
2	andrei	divertisment	netflix	andre1	parolaNetflix
3	tudor	eshops	emag	tudor3	parolaemag
4	razvan	divertisment	steam	razboy	par
5	andrei	filme	netflix	test	parolaNNN
6	andrei	divertisment	steam	usr	paarola
7	andrei	divertisment	HBOMAX	andrei	hbopar

Figure 4:

4.2 Client (cliPassMan.c)

The client connects to the server using the server's IP address and port given as command line arguments. It sends the commands given by the user as input to the server and displays the server's response in the terminal.

4.2.1 Socket Creation and Connection:

The client creates a socket using `socket()` and connects to the server's IP address and port using `connect()`.

```
/* stabilim portul */
port = atoi(argv[2]);

/* cream socketul */
if ((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
    perror("Eroare la socket().\n");
    return errno;
}

/* umplem structura folosita pentru realizarea conexiunii cu serverul */
/* familia socket-ului */
server.sin_family = AF_INET;
/* adresa IP a serverului */
server.sin_addr.s_addr = inet_addr(argv[1]);
/* portul de conectare */
server.sin_port = htons(port);

/* ne conectam la server */
if (connect(sd, (struct sockaddr *)&server, sizeof(struct sockaddr)) == -1)
{
    perror("[client]Eroare la connect().\n");
    return errno;
}
printf(" P A S S W O R D   M A N A G E R \n");
printf(" ----- \n");
```

Figure 5:

4.2.2 User Interaction and Data Transmission

After the client successfully connects to the server it displays in the terminal a list of valid commands for the user. For commands like register, login and quit the user can use them without being authenticated by the server. If the user is authenticated it can use the other set of commands.

The user types a command in the terminal which is sent to the server using `write()` then the server awaits for the server's response using `read()`. The response is displayed in the terminal for the user to see.

4.2.3 logged Function

The `logged(int sd)` function is being called only when the server sends the 'logged' response. It behaves just like the while loop in `main()` but it let's the user know it is logged in the system and can access the other commands,

4.3 Flow

The user inputs a command which is sent to the server, the client awaits the server's response. The server reads that command and sends it to the router logic(`comanda`, `raspuns`, `ok`, `toKeep`), if the command is invalid, logic() will update the server's response, if not, the command is passed to it's corresponding function, there it is being parsed and validated. If it doesn't pass validation, response will be updated, else it is sent further to the corresponding function that handles database operations, there, the information will be validated once again with the data stored in the database and response will be updated. As an argument in all those functions, response is being passed back in the server loop and sent to the client. In the client, the response is being displayed to the user, which will be able to see if the command ran successfully or not.

5 Conclusions

'Password Manager' is a client-server application that provides an easy way for users to store their passwords and usernames, and also categorise their accounts through terminal interface. Using TCP and multithreading it can reliably handle multiple connections concurrently. This app can be easily scaled as an widespread use WEB application. One possible improvement is changing the database layout so that it allows more flexibility when updating rows.