

Personal Budgeting Tool

Bejan Andreea 2E1

University "Alexandru Ioan Cuza", Faculty of Computer Science

Abstract. In this technical report is presented the project Personal Budgeting Tool and the logics behind it: the implemented technologies, the general architecture and implementation details.

1 Introduction

1.1 The overall vision of this project:

Personal Budgeting Tool is a project created based on the client/server structure, where the server can process different commands based on financial related issues for multiple users. The server works with the information provided through an sql table, in order to answer to the possible commands of the clients and process their transactions and bank accounts. Moreover, it can provide a personalised financial analysis based on the information from the database.

1.2 The objectives of this project:

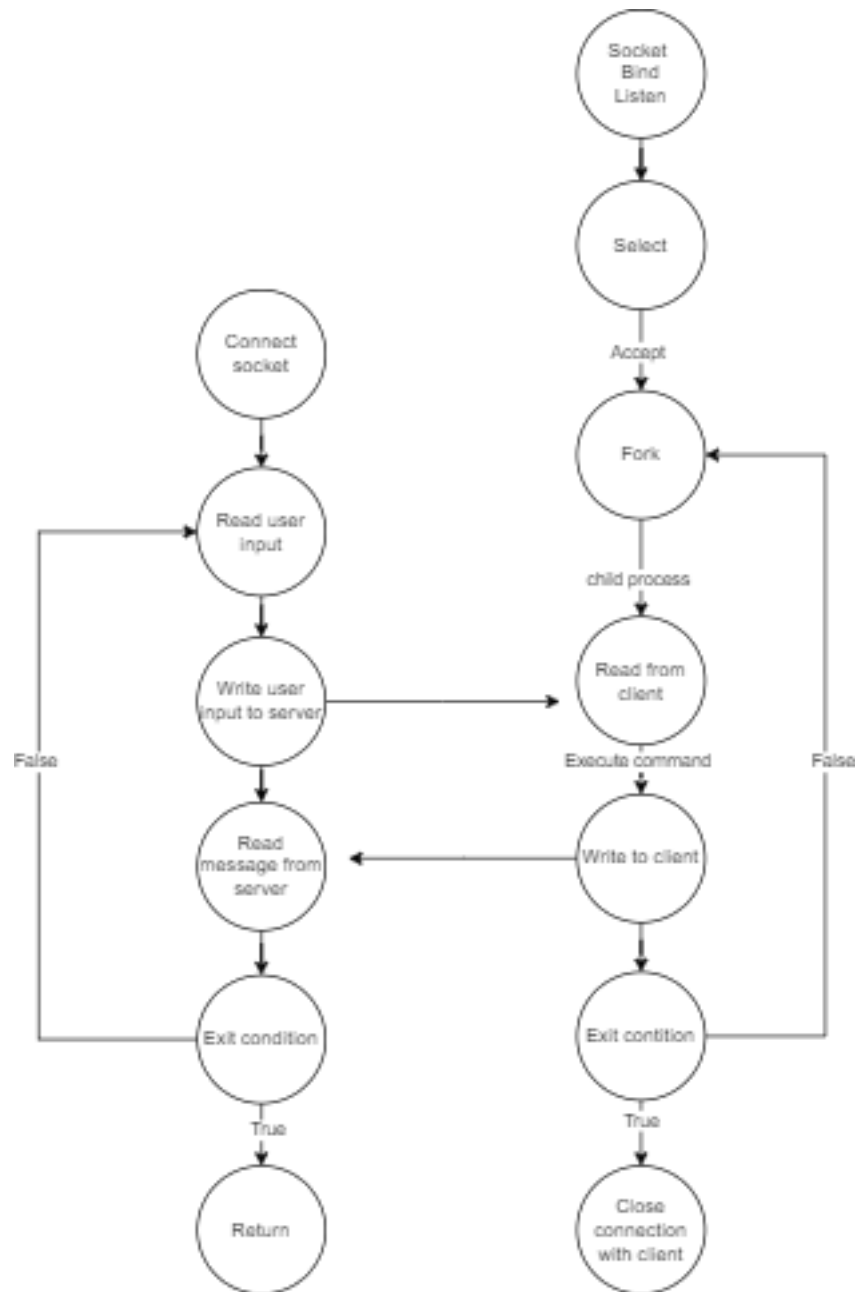
This project involves being capable of handling multiple users at once, and their specific financial details. I chose this topic because it can implement multiple functionalities and can be improved in many different ways.

2 Applied Technologies

I used the implementation of TCP (Transmission Control Protocol) with multiplexing via select and fork. First of all, I used TCP because such an application requires that the communication is connection-oriented and reliable. Secondly, I used fork to be able to serve multiple clients concurrently. Last but not least, I used select, which helps the program to monitor multiple descriptors (in our case, them being the representation of users), to help me monitor login possibilities.

3 Application Structure

Client (left) and server (right) diagram:





False cases look the same for transfer as for login*

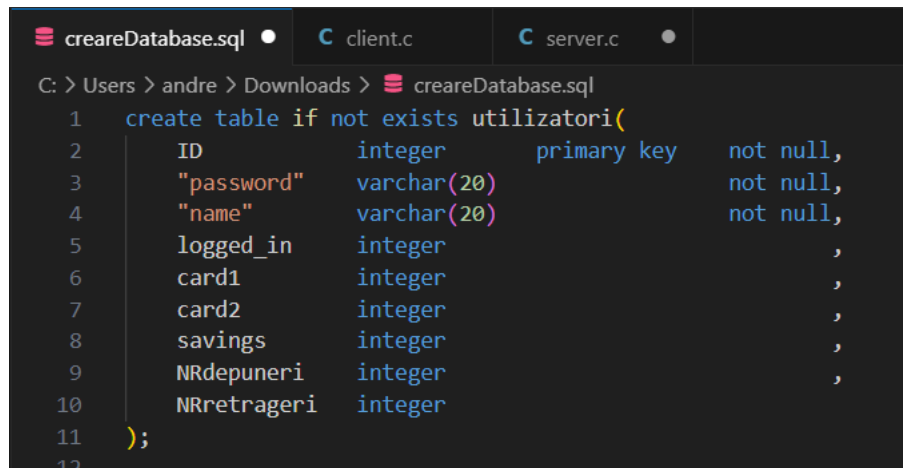
4 Implementation aspects

The project that I chose requires that multiple users need to be able to access their data consequently, in a secure way, so that is why I implemented the TCP protocol via select and fork. However, select is used for a more specific scope in my program, regarding the possibilities of login.

First of all, for handling the information about the users, and being able to process every command efficiently, I used an sql structure.

	ID	password	name	logged_in	card1	card2	savings	NRdepuner	NRretrager
1	111	pisica1	Andrei	0	300	100	200	1	9
2	222	pisica2	Antonia	0	1500	50	20	4	2
3	333	pisica3	Maria	0	150	0	100	2	4
4	444	pisica4	Darius	0	170	100	100	4	4

In order to make the program similar to one used in a real context, I chose to create the information table via a .sql file, and open it for use in my C file. I wanted to make it possible for the changes done in the table through my program to persist.



```

createDatabase.sql • client.c server.c
C: > Users > andre > Downloads > createDatabase.sql
1  create table if not exists utilizatori(
2      ID            integer      primary key  not null,
3      "password"    varchar(20)  not null,
4      "name"        varchar(20)  not null,
5      logged_in     integer
6      card1         integer
7      card2         integer
8      savings       integer
9      NRdepuneri    integer
10     NRretrageri   integer
11 );
12

```

The CARD1, CARD2, SAVINGS, NRdepuneri and NRretrageri columns help the program to create a personalised, updated financial analysis for each one of the users. This is the one of the most interesting usages of the Personal Budgeting Tool.

```

void BazaDeDate()
{
    /*Open a database file*/
    int rc = sqlite3_open("database.db", &db);
    if(rc != SQLITE_OK){
        printf("[server] Eroare la deschiderea bazei de date");
        fflush (stdout);
        exit(0);
    }

    sqlite3_prepare_v2(db,"SELECT ID, password, name, logged_in, card1, card2,
    savings, NRdepuneri, NRretrageri FROM utilizatori", -1, &stmt, 0);

    while (sqlite3_step(stmt) != SQLITE_DONE){
        printf("%d|", sqlite3_column_int(stmt,0));
        printf("%s|", sqlite3_column_text(stmt,1));
        printf("%s|", sqlite3_column_text(stmt,2));
        printf("%d|", sqlite3_column_int(stmt,3));
        printf("%d|", sqlite3_column_int(stmt,4));
        printf("%d|", sqlite3_column_int(stmt,5));
        printf("%d|", sqlite3_column_int(stmt,6));
        printf("%d|", sqlite3_column_int(stmt,7));
        printf("%d|", sqlite3_column_int(stmt,8));
        printf("\n");
        fflush (stdout);
    }

    sqlite3_finalize(stmt);
}

```

Another very interesting and helpful concept that I used in my program is in the login process: once someone logged in, the "logged_in" column from the database will remember its descriptor number. Doing so will clarify for the program which client called what command. Once the user is logging out, the column will get back to the 0 value.

5 Conclusions

The project Personal Budgeting Tool can benefit from multiple types of improvements. Besides the most obvious one, which is a graphical interface, the project could allow the users table to mention through a special column the title of the person (for example, a simple client or an administrator). Therefore, someone identified as an administrator could add or remove users through the actual app.

References

1. Concurrent TCP
https://profs.info.uaic.ro/~georgiana.calancea/Laboratorul_7.pdf
2. Communication via SELECT
https://profs.info.uaic.ro/~georgiana.calancea/Laboratorul_9.pdf
3. SQL
https://www.w3schools.com/sql/sql_create_table.asp
<https://www.sqlite.org/index.html>
<https://www.linuxjournal.com/content/accessing-sqlite-c>