PersonalBudgetingTool

Petraru Marcu-Darie[1][0009−0008−3940−4228]

[1] Facultatea de Informatică - Iași
[2] secretariat@info.uaic.ro
http://www.info.uaic.ro

1 Viziunea generală și obiectivele

Proiectul **PersonalBudgetingTool** propune o abordare inovatoare în controlul și analiza conturilor bancare și tranzacțiilor. Scopul primordial al acestei aplicații este de a asista utilizatorii în îmbunătățirea gestionării financiare personale și de a oferi o analiză detaliată a tipurilor de tranzacții care au cel mai mare impact asupra resurselor financiare, contribuind astfel la dezvoltarea lor financiară durabilă.

Principalele obiective ale aplicației includ:

1. O analiză financiară avansată asupra tuturor tranzacțiilor de pe fiecare card înregistrat. Se va monitoriza evoluția financiară prin obținerea unor rapoarte și statistici.

2. O sugestie personalizată bugetară pentru clienți astfel încât aceștia să poată să economisească cât mai mulți bani și să aibă o mai bună gestionare a tranzacțiilor și o înțelegere mai bună asupra propriilor cheltuieli.

3. Dorința de a oferi clientului o viziune pentru a-și crea un obicei financiar dezvoltator astfel încât să poată să aibă o mai bună gestionare personală a veniturilor și cheltuielilor.

2 Tehnologii aplicate

Tehnologia TCP implementată în "**PersonalBudgetingTool**" furnizează o comunicare sigură și fiabilă între client și server. TCP asigură o livrare ordonată și securizată a datelor, ceea ce este de o importanță crucială în manipularea informațiilor financiare sensibile. Fiabilitatea livrării datelor este garantată prin intermediul unui sistem de confirmare a primirii pachetelor, și în situația în care un pachet nu este receptat corect, se inițiază retransmiterea acestuia. TCP contribuie semnificativ la asigurarea integrității și corectitudinii datelor transmise de la client către server, fundamentale pentru buna funcționare a aplicației.

Tehnologia UDP nu poate fi folosită, deoarece e posibilă pierderea datelor bancare, ceea ce ar face ca aplicația să nu funcționeze în parametri normali și să nu poată transmite utilizatorului un raport corect cu realitatea.

SQLite3 este o opțiune potrivită pentru proiect datorită ușurinței de implementare, ajută la gestionarea parametrilor de login, cardurilor bancare și tranzacțiilor specifice fiecărui card. Este integrată în majoritatea limbajelor de programare, asigurând o stocare eficientă și structurată a datelor.

3 Structura aplicației

3.1 Concepte folosite în modelare

Entități

• Utilizator (Username, Password, Balanța Totală):

-Username: Identificator unic al utilizatorului.

-Password: Informație secretă pentru autentificare.

-Balanța Totală: Soldul total al utilizatorului, reflectând suma totală a banilor disponibili.

-Logged: Dacă utilizatorul este logat în momentul de față sau nu, pentru a nu crea conflicte cu altcineva care dorește să se conecteze cu acelasi cont.

• Card Bancar (Username, IBAN, Balanța):

-Username: Atribute pentru a stabili legătura între card și utilizator.

-IBAN: Identificator unic al cardului bancar.

-Balanța: Soldul curent al cardului bancar.

• Tranzacție (IBAN, Tip Tranzacție, Cost):

-IBAN: Legătura către cardul bancar asociat tranzacției.

-Tip Tranzacție: Indică dacă este o tranzacție de depunere sau retragere.

-Cost: Valoarea tranzacției.

• Categorii de tranzacții:

-Mâncare

-Apă

-Garderobă

-Jocuri video

-Gambling

-Activități

-Alcool

-Țigări

-Transport

Relații

• Utilizator ⇒ Card Bancar (Unul cu mai Mulți):

-Un utilizator poate avea mai multe carduri bancare.

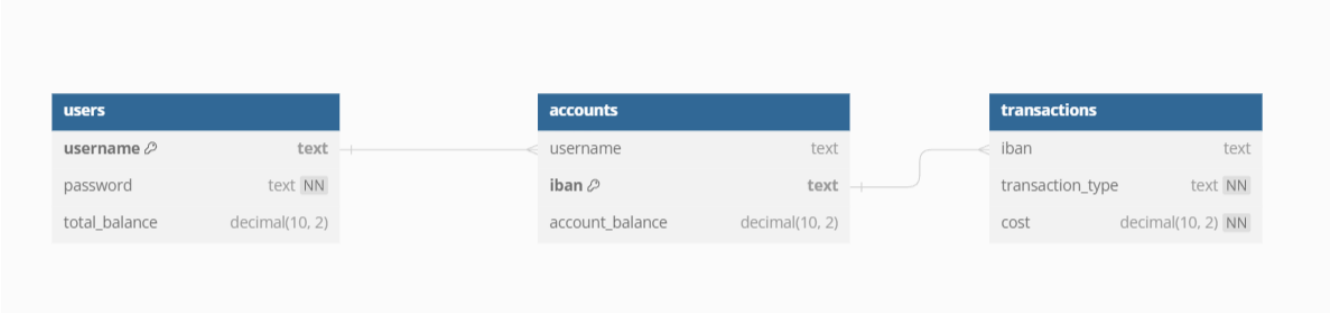
-Un card bancar este asociat unui singur utilizator.

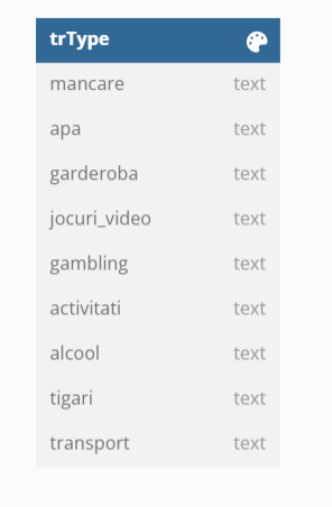
• Card Bancar ⇒ Tranzacție (Unul cu mai Mulți):

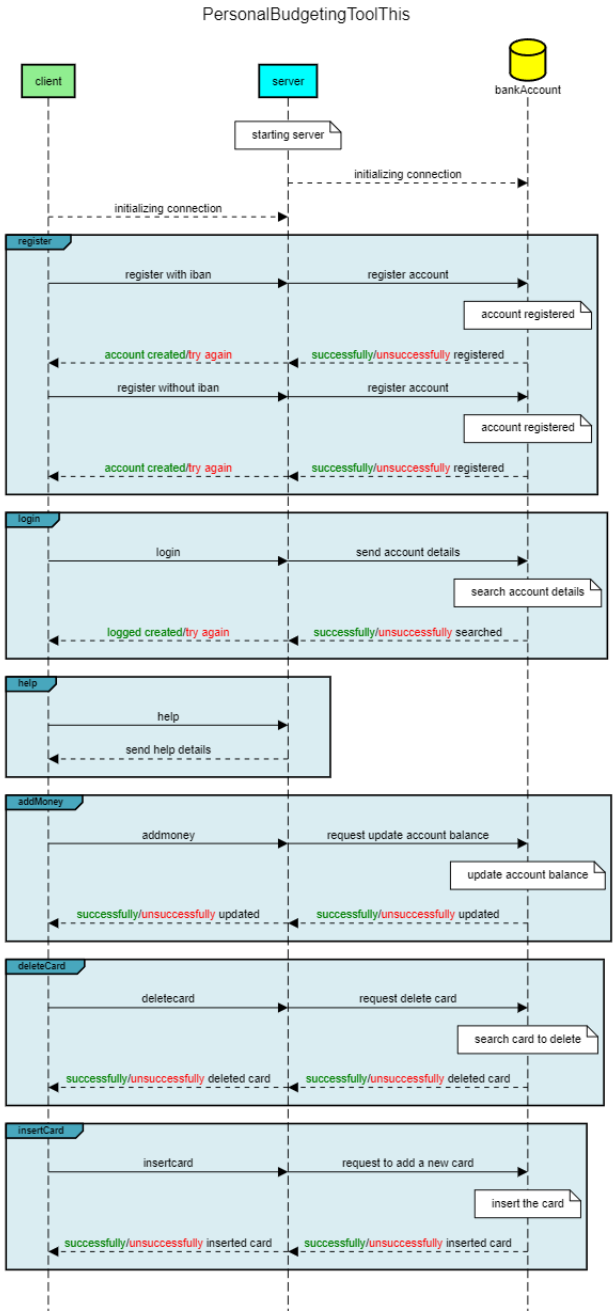
-Un card bancar poate fi asociat cu mai multe tranzacții.

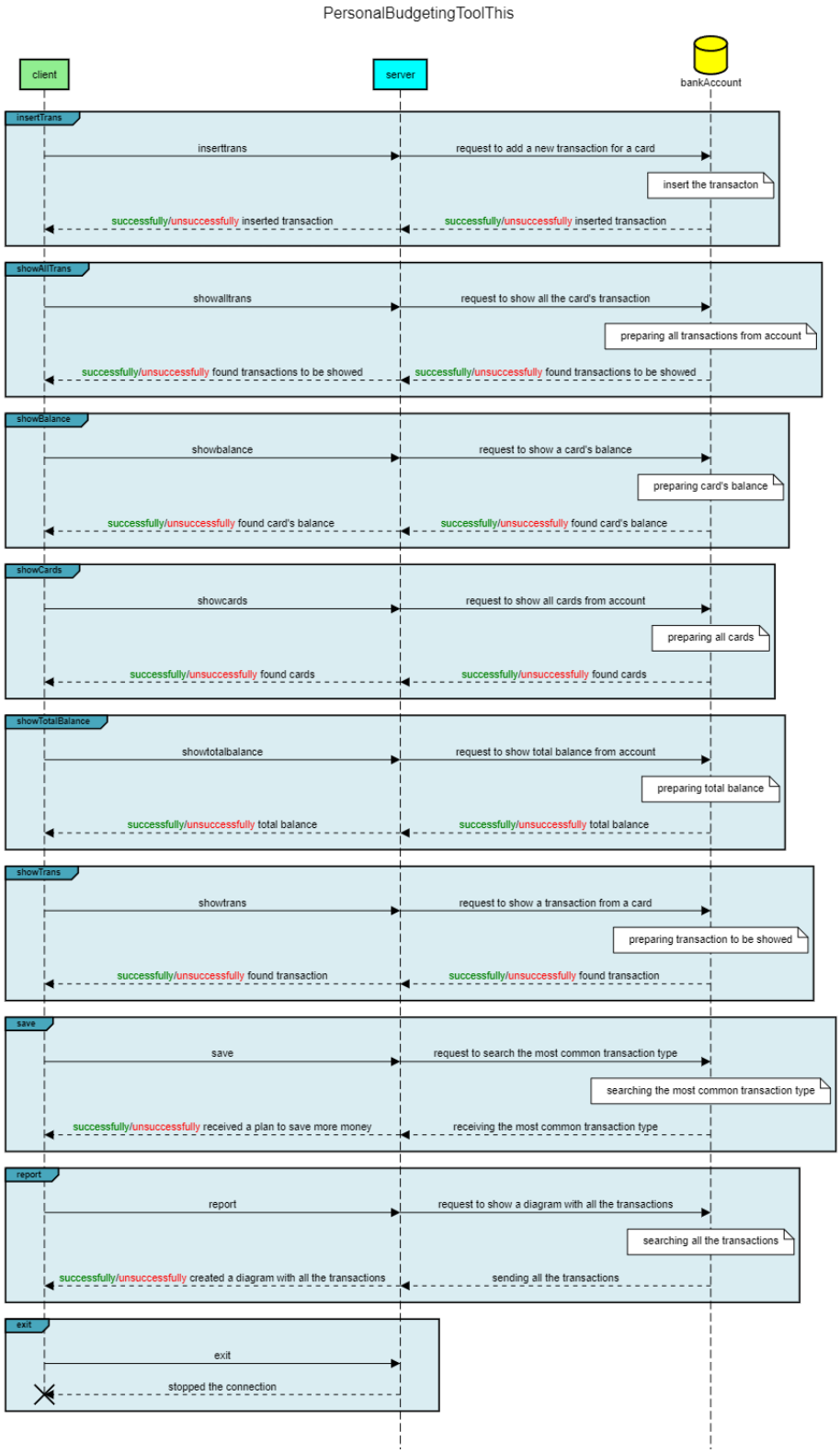
-O tranzacție este asociată cu un singur card bancar.

3.2 Diagrame


users
username text
password text NN
total_balance decimal(10, 2)
accounts
username text
iban text
account_balance decimal(10, 2)
transactions
iban text
transaction_type text NN
cost decimal(10, 2) NN


trType
mancare text
apa text
garderoba text
jocuri_video text
gambling text
activitati text
alcool text
tigari text
transport text


PersonalBudgetingToolThis
client
server
bankAccount
starting server
initializing connection
initializing connection
register
register with iban
register account
account registered
account created/try again
successfully/unsuccessfully registered
register without iban
register account
account registered
account created/try again
successfully/unsuccessfully registered
login
login
send account details
search account details
logged created/try again
successfully/unsuccessfully searched
help
help
send help details
addMoney
addmoney
request update account balance
update account balance
successfully/unsuccessfully updated
successfully/unsuccessfully updated
deleteCard
deletecard
request delete card
search card to delete
successfully/unsuccessfully deleted card
successfully/unsuccessfully deleted card
insertCard
insertcard
request to add a new card
insert the card
successfully/unsuccessfully inserted card
successfully/unsuccessfully inserted card


PersonalBudgetingToolThis
client
server
bankAccount
insertTrans
inserttrans
request to add a new transaction for a card
insert the transacton
successfully/unsuccessfully inserted transaction
successfully/unsuccessfully inserted transaction
showAllTrans
showalltrans
request to show all the card's transaction
preparing all transactions from account
successfully/unsuccessfully found transactions to be showed
successfully/unsuccessfully found transactions to be showed
showBalance
showbalance
request to show a card's balance
preparing card's balance
successfully/unsuccessfully found card's balance
successfully/unsuccessfully found card's balance
showCards
showcards
request to show all cards from account
preparing all cards
successfully/unsuccessfully found cards
successfully/unsuccessfully found cards
showTotalBalance
showtotalbalance
request to show total balance from account
preparing total balance
successfully/unsuccessfully total balance
successfully/unsuccessfully total balance
showTrans
showtrans
request to show a transaction from a card
preparing transaction to be showed
successfully/unsuccessfully found transaction
successfully/unsuccessfully found transaction
save
save
request to search the most common transaction type
searching the most common transaction type
successfully/unsuccessfully received a plan to save more money
receiving the most common transaction type
report
report
request to show a diagram with all the transactions
searching all the transactions
successfully/unsuccessfully created a diagram with all the transactions
sending all the transactions
exit
exit
stopped the connection

3.3 Prezentare diagrame

Prima diagramă, cea pentru baza de date, reprezintă relațiile prin care un utilizator își salvează contul (username, password, total_balance) în tabelul users. La rândul lui, fiecare cont va dispune de tabelul accounts, în care sunt stocate informații specifice lui (username, iban, account_balance). Un username poate avea mai multe IBAN-uri, dar un IBAN nu poate fi asociat mai multor username-uri. În tabela transactions, un IBAN poate avea mai multe tranzacții (transaction_type, cost).

A doua diagramă, cea pentru baza de date, unde se cunosc tipurile de tranzacții si categoriile lor, este folosita pentru a putea categorisi mai multe tranzacții. Principalul scop este pentru a ajuta funcțiile report și save să recunoasca mai usor tranzacțiile și să poată calcula exact costul și recomandările.

Următoarele două diagrame reprezintă o diagramă mai mare. Acolo sunt prezentate relațiile dintre client, server și baza de date. Când pornește serverul, acesta face o conexiune la baza de date pentru a putea lucra cu comenzile. Când pornește clientul, acesta se conectează la server. Clientul poate trimite diferite tipuri de comenzi serverului, iar în funcție de comandă, serverul decide dacă este nevoie să se folosească de baza de date sau nu. Dacă este nevoie de baza de date, serverul trimite o cerere bazei de date, baza de date procesează și trimite înapoi în server un răspuns, iar apoi serverul trimite clientului răspunsul de care are nevoie. TCP este util, deoarece asigură o transmitere sigură a pachetelor pentru a nu se ajunge la neînțelegeri de logică între client, server și baza de date.

4 Aspecte de implementare

4.1 Secțiuni de cod

Inițializare conexiune server-baza de date

```
sqlite3 *db1;
int rc1 = sqlite3_open("bankAccount.db", &db1);
if (rc1 != SQLITE_OK)
{
    perror("Nu s-a putut deschide baza de date bankAccount.db.\n");
    return rc1;
}
```

În această secțiune de cod se realizează crearea legăturii între server și baza de date. Descriptorul "db1" va fi cel care va ajuta la comunicarea între comenzile primite de la client pentru a comunica și cu baza de date.

Exemplu de citire a unei comenzi primite de la client

```
if (read(tdL.cl, &msg_sizen, sizeof(size_t)) < 0)
{
    perror("[server]Eroare la read() de la client.\n");
}

msg_size = ntohl(msg_sizen);
msg = (char *)malloc(msg_size);

if (read(tdL.cl, msg, msg_size) < 0)
{
    perror("[server]Eroare la read() de la client.\n");
}
```

Pentru a putea comunica cu clientul, serverul va folosi funcția standard "read". Serverul va dispune de "tdL.cl" (descriptorul prin care pentru un client va fi folosit un thread), "msg_sizen" (se va salva mărimea comenzii înainte de trimiterea acesteia). "msg_size" va fi transformat în endianul specific sistemului prin funcția "ntohl(msg_sizen)", iar apoi se va crea un buffer "msg" de lungime "msg_size" în care se va stoca comanda propriu-zisă de la client.

Exemplu de parsare a comenzii login primită de la client

```
char *p = strtok(msg, ":");
if (p != NULL)
{
    command = (char *)malloc(strlen(p) + 1);
    strcpy(command, p);
}

p = strtok(NULL, " ");
if (p != NULL)
{
    username = (char *)malloc(strlen(p) + 1);
    strcpy(username, p);
}

p = strtok(NULL, " ");
if (p != NULL)
{
    password = (char *)malloc(strlen(p) + 1);
    strcpy(password, p);
}
```

Comanda salvată în "msg" va fi parsată pentru a putea afla ce fel de comandă este și ce parametri o însoțesc. Pentru asta, s-a folosit un token "p" care mai întâi va reține numele comenzii, iar se va refolosi tot "p" pentru a putea afla și username-ul și password-ul, care vor fi folosite în baza de date.

Exemplu pentru a putea apela o funcție specifică comenzii

```
if (strcmp(command, "login") == 0)
{
    loginFunction(tdL.cl, db1, username, password, &login);
    free(msg);
    free(command);
}
```

După parsarea lui "msg", în "command" se va afla numele comenzii. Pentru fiecare comandă se va apela funcția specifică. În imaginea de mai sus, vom apela

funcția "loginFunction" cu parametrii, "tdL.cl" (descriptorul), "db1" (descriptorul bazei de date), "username", "password" (parametrii pentru a putea face verificarea contului în baza de date), "login" (o variabilă care va reprezenta dacă un utilizator s-a putut loga sau nu).

Exemplu de utilizare a bazei de date pentru comanda login

```c
#include "functions.h"

#define LOGIN "Te-ai logat cu succes!\n"
#define BADLOGIN "Username sau parola gresite! Incearca din nou!\n"
#define OTHERUSER "Acest cont este deja folosit! Incearca din nou!\n"

void loginFunction(int client, sqlite3 *db1, char *username, char *password, int *login)
{
    char *sql = sqlite3_mprintf("SELECT logged FROM users WHERE username=? AND password=?;");
    sqlite3_stmt *stmt;
    int rcsql = sqlite3_prepare_v2(db1, sql, -1, &stmt, NULL);

    if (rcsql != SQLITE_OK)
    {
        fprintf(stderr, "Eroare la pregătirea instrucțiunii SQL: %s\n", sqlite3_errmsg(db1));
        sqlite3_free(sql);
        return;
    }

    sqlite3_bind_text(stmt, 1, username, -1, SQLITE_STATIC);
    sqlite3_bind_text(stmt, 2, password, -1, SQLITE_STATIC);

    int step = sqlite3_step(stmt);

    if (step == SQLITE_ROW)
    {
        int logged = sqlite3_column_int(stmt, 0);
        if (logged == 1)
        {
            writeToClient(client, OTHERUSER);
        }
        else
        {
            writeToClient(client, LOGIN);
            *login = 1;

            char *sql_update = sqlite3_mprintf("UPDATE users SET logged=1 WHERE username=?;");
            sqlite3_stmt *stmt_update;
            if (sqlite3_prepare_v2(db1, sql_update, -1, &stmt_update, NULL) == SQLITE_OK)
            {
                sqlite3_bind_text(stmt_update, 1, username, -1, SQLITE_STATIC);
                sqlite3_step(stmt_update);
                sqlite3_finalize(stmt_update);
            }
            sqlite3_free(sql_update);
        }
    }
    else if (step == SQLITE_DONE)
    {
        writeToClient(client, BADLOGIN);
    }

    sqlite3_finalize(stmt);
    sqlite3_free(sql);
}
```

În funcția "loginFunction", vom utiliza baza de date pentru a putea identifica dacă există username-ul și password-ul trimis de client. Vom folosi un char pointer "sql" pentru a aloca dinamic în heap comanda specifică SQL, adică "SELECT * FROM users WHERE username=? AND password=?;".

"stmt" poate fi folosit pentru a putea executa interogarea din "sql", funcția "sqlite3_prepare_v2" va fi cea care va face legătura între "stmt" și interogarea SQL.

Funcția "sqlite3_bind_text" va înlocui "?" din interogarea SQL astfel încât să se poată verifica username-ul și password-ul.

În bucla "while", vom itera prin rezultatele interogării, iar dacă am găsit un rând care conține username-ul și password-ul, vom apela funcția "writeToClient", iar variabila "login" va deveni 1 pentru ca serverul să cunoască reușita comenzii primite.

La final, se va elibera memoria ocupată de către "stmt" și "sql".

Exemplu de timitere raspuns de la server la client

```
void writeToClient(int client, const char *msg_sv)
{
    size_t msg_size = strlen(msg_sv);
    size_t msg_sizen = htonl(msg_size);

    printf("[server]Trimitem mesajul inapoi...%s\n", msg_sv);

    if (write(client, &msg_sizen, sizeof(size_t)) <= 0)
    {
        perror("[server]Error writing to client.\n");
        return;
    }

    if (write(client, msg_sv, msg_size) <= 0)
    {
        perror("[server]Error writing to client.\n");
    }
}
```

Functia "writeToClient", va fi folosita pentru a putea trimite de la server la client mesajul specific comenzii dorite de client.

4.2 Protocolul la nivelul aplicatiei

Când se conectează un client, acesta va fi nevoit să se autentifice sau să creeze un cont. Serverul va primi informația și se va folosi de baza de date pentru a verifica. Când clientul va trimite o comandă, aceasta va fi parsată în server, iar mai apoi parametrii specifici vor fi parsați și ei pentru a putea fi trimiși împreună cu comanda la baza de date. Se va trimite de la client automat mai întâi mărimea comenzii și după comanda în sine, serverul va primi mărimea și o va transforma în endianul specific acestuia, și va aloca un buffer de acea mărime, iar mai apoi va salva în buffer comanda. Serverul va utiliza funcții specifice comenzii pentru

a putea comunica cu baza de date și a primi un răspuns. Când verificarea bazei de date e gata, aceasta va trimite la server mai întâi mărimea mesajului și mai apoi mesajul, asemenea cum se întâmplă și în comunicarea client-server. În final, serverul va trimite la fel răspunsul pentru client.

Aplicația **PersonalBudgetingTool** are o largă arie de utilizare în realitate deoarece poate reprezenta o unealtă utilizată de toată lumea. Orice persoană care are nevoie de o mai bună gestionare și organizare a cheltuielilor.

Poate fi integrată și de către bănci în aplicațiile lor pentru a putea interacționa direct cu baza de date a băncii fără a fi nevoie de introducerea manuală a tranzacțiilor sau IBAN-urilor. Poate fi utilizată și în aplicațiile de comerț online sau chiar în aplicații de tranzacționare cu criptomonede sau la bursa.

Aplicația poate ajuta la o mai bună dezvoltare financiară și la o scădere a cheltuielilor inutile.

5 Concluzii

"**PersonalBudgetingTool**" își propune să devină partenerul de încredere al utilizatorilor în călătoria lor către o mai bună gestionare a finanțelor personale și dezvoltare financiară.

5.1 Îmbunătățiri

Îmbunătățiri ale proiectului pot fi reprezentate de:

Crearea unui GUI astfel încât clientul să aibă o utilizare mai prietenoasă și ușor de înțeles pentru a folosi tot potențialul comenzilor serverului.

Utilizarea unui sistem de securitate și confidențialitate pentru baza de date cu detaliile tuturor conturilor.

Portarea aplicației pe AppStore și PlayStore pentru a putea avea o utilizare mai ușoară și mai compactă.

Crearea de grafice interactive care pot afișa un pattern cu referință în viitor.

References

1. Computer Networks Course, https://profs.info.uaic.ro/ computernetworks/index.php,
2. Computer Networks Matei Mădălin, https://profs.info.uaic.ro/ matei.madalin/rc/,
3. Computer Networks Laboratory, https://profs.info.uaic.ro/ eonica/rc/,
4. Shichao's Notes, https://notes.shichao.io/unp/,
5. man7.org, https://man7.org/index.html,
6. SQLite3 Cheat Sheet, https://vhernando.github.io/sqlite3-cheat-sheet,