

Universitatea Națională de Știință și Tehnologie POLITEHNICA București Facultatea  
de Electronică, Telecomunicații și Tehnologia Informației

**TEHNOLOGII DE PROGRAMARE ÎN INTERNET**  
**SISTEM DE GESTIUNE A FINANTELOR PERSONALE**  
*TEMA E.26*

STUDENȚI:

Nanu Ana-Maria

Nistor Elena-Daniela

GRUPA:

434D

PROFESOR COORDONATOR:

Șl. Dr. Ing. Boicescu Laurențiu

2023-2024

# 1. INTRODUCERE

## 1.1 Obiectivul lucrării

Ne propunem să dezvoltăm o aplicație Android care să ajute utilizatorii să gestioneze eficient finanțele personale. Acest proiect vine în întâmpinarea unei nevoie crescânde de educație financiară în România, având în vedere rata ridicată a sărăciei și excluziunii sociale înregistrată în anul 2022, de 34,4%.

Obiectivul principal al aplicației este să ofere utilizatorilor o platformă simplă și eficientă pentru gestionarea veniturilor, cheltuielilor și economiilor personale. Prin intermediul acestei aplicații, dorim să promovăm înțelegerea și controlul financiar, indiferent de nivelul de experiență al utilizatorului.

## 1.2 Caracteristicile cheie ale aplicației

Utilizatorii pot crea un cont introducând datele personale de baza, precum: nume, prenume, adresa de email, număr de telefon.

Configurarea contului se face prin setarea veniturilor lunare și cheltuielilor lunare, cum ar fi abonamente, credite, facturi, salariu, cadouri etc. De asemenea, utilizatorul își poate stabili o sumă pe care să o economisească regulat, fiind prezentă și opțiunea de alegere a monedei pentru administrarea contului.

După configurarea contului, utilizatorul poate vizualiza și gestiona veniturile și cheltuielile într-un mod intuitiv. Se vor genera rapoarte grafice, pe mai multe categorii, pentru o mai bună înțelegere și vizualizare a fluxurilor financiare.

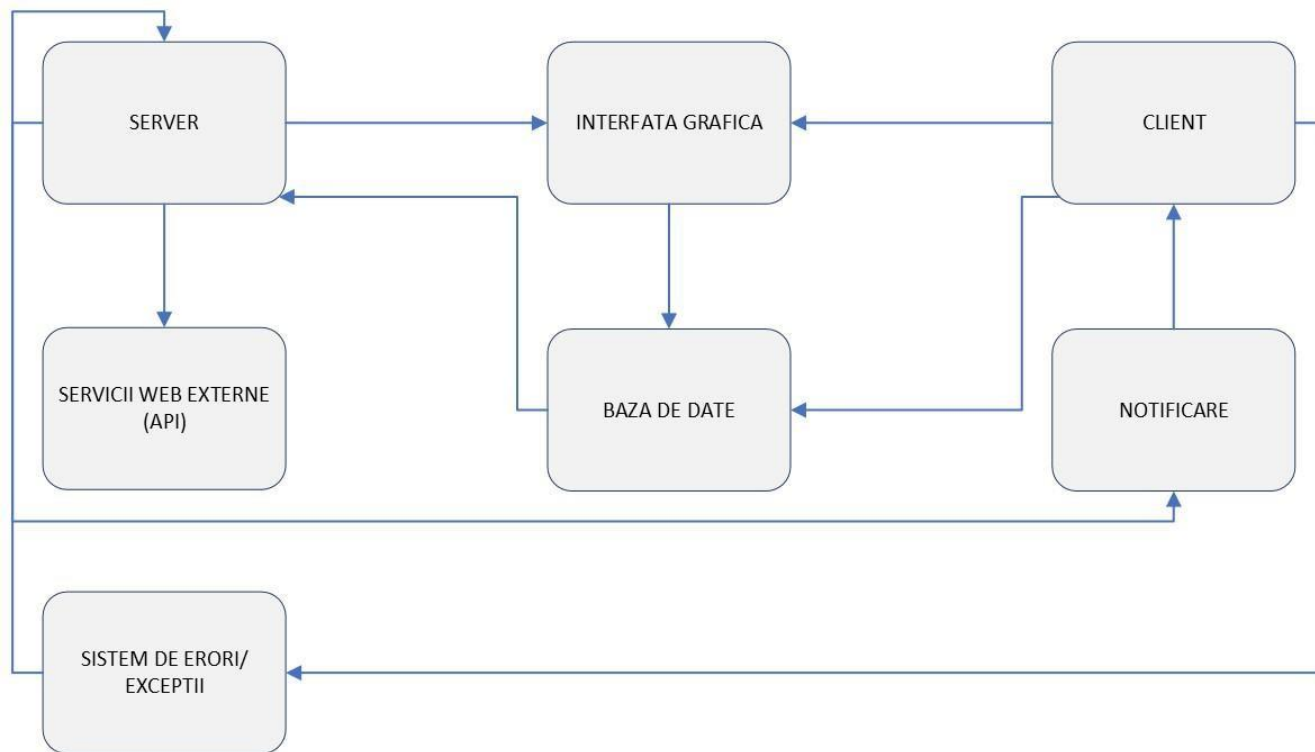
Utilizatorul poate alege opțiunea de notificări și avertismente. Această opțiune a aplicației va emite notificări personalizate pentru plăți, economii și alte evenimente financiare relevante.

## 1.3 Utilizatori țintă

Aplicația este destinată tuturor persoanelor interesate de gestionarea finanțelor personale, indiferent de nivelul de cunoștințe în domeniu. Este ideală pentru cei care doresc să își îmbunătățească abilitățile de administrare a banilor, să-și monitorizeze mai eficient situația financiară sau pentru cei care își doresc să înceapă să economisească.

## 2. PROIECTARE

### 2.1 Schema bloc



Figură 1 Schema bloc

Blocul de server reprezintă componenta centrală a aplicației care gestionează procesele de gestionare a datelor. Acesta primește și procesează cererile de la client (aplicația Android) și gestionează calculul sumelor veniturilor, cheltuielilor și economiilor. De asemenea, interconectează cu baza de date pentru a accesa și actualiza informațiile utilizatorilor și administrează autentificarea și autorizarea utilizatorilor.

Blocul client reprezintă aplicația Android instalată pe dispozitivul utilizatorului final. Interacționează cu utilizatorul prin intermediul interfeței grafice pentru a colecta și afișa informațiile financiare. De asemenea, clientul trimite cereri către server pentru a obține și actualiza datele din aplicație și gestionează aspectele legate de securitatea și confidențialitatea datelor utilizatorului.

Interfața grafică reprezintă componenta vizuală a aplicației, cu care utilizatorii interacționează direct. Aceasta oferă utilizatorilor o interfață intuitivă pentru introducerea datelor personale, setarea bugetului, vizualizarea rapoartelor financiare etc. și afișează informațiile într-un mod clar și ușor de înțeles.

Blocul pentru servicii web externe (API-uri) reprezintă serviciile externe la care aplicația se poate conecta pentru a accesa informații suplimentare sau pentru a efectua anumite operațiuni. Facilitează comunicarea cu alte sisteme sau servicii cum ar fi informațiile de schimb valutar sau actualizările de piețe financiare.

Blocul bazei de date se ocupă cu stocarea informațiilor financiare și detaliile utilizatorilor. Una dintre responsabilitățile bazei de date este de a asigura persistența și integritatea datelor.

Blocul pentru sistemul de erori/execuții gestionează erorile și excepțiile care pot apărea în timpul funcționării aplicației. Acesta monitorizează și înregistrează erorile pentru a asigura stabilitatea și performanța aplicației, dar și tratează și gestionează excepțiile pentru a asigura o experiență fidelă a utilizatorului.

Sistemul de notificare se ocupă de trimiterea de notificări către utilizatori pentru a informa despre evenimente importante sau pentru a oferi avertismente. Acesta trimite notificări personalizate pentru plăți, economii atinse, depășirea bugetului și interacționează cu alte componente pentru a asigura actualizarea și afișarea notificărilor în timp real.

## 2.2 Propunere interfață grafică

Pentru a vizualiza mai ușor o parte dintre funcționalitățile aplicației, s-a realizat o propunere a unei interfețe grafice prietenoase cu utilizatorul.



Figura 2a Pagina de start

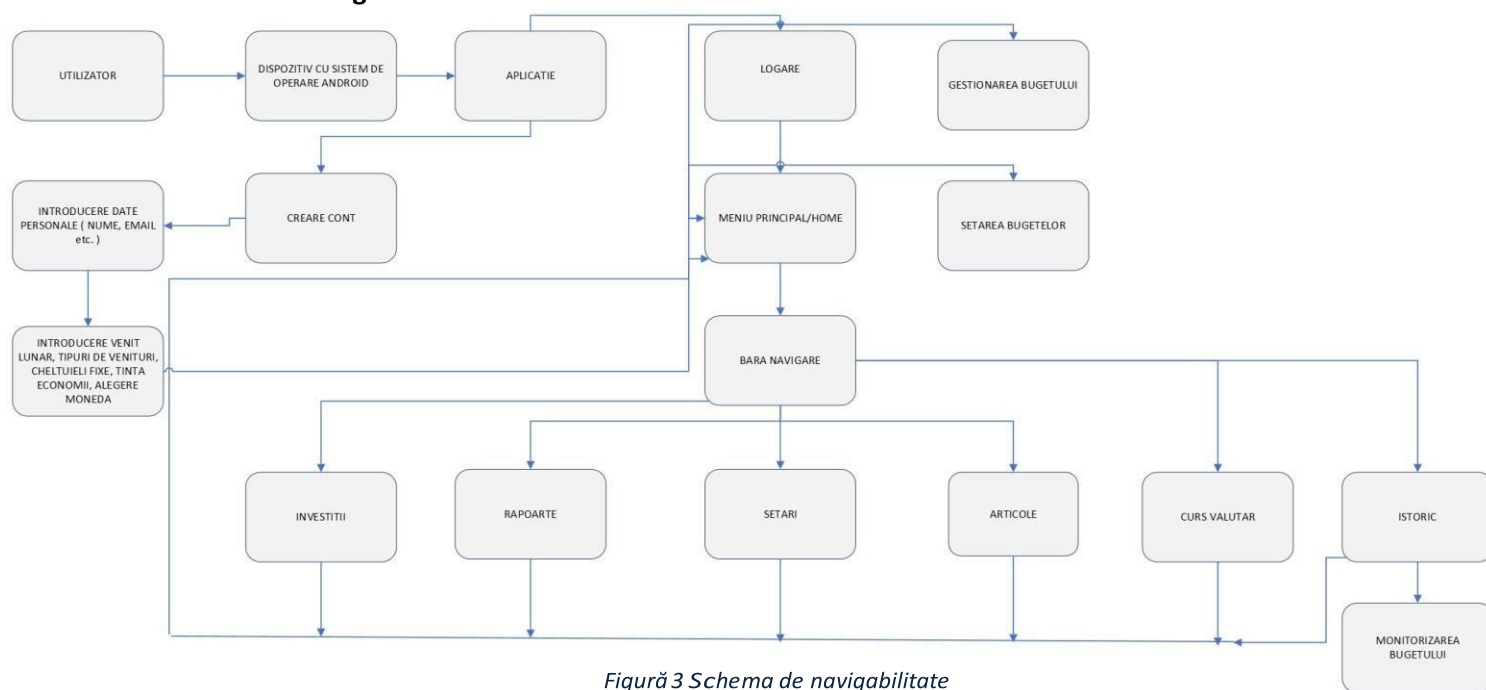


Figura 2b Meniul Home

The image shows the 'Adaugă tranzacție' (Add transaction) screen. It has a blue background. At the top, there is a 'Cancel' button and the title 'Adaugă tranzacție'. Below the title, there are two buttons: 'CHELTUIELI' (Expenses) and 'VENITURI' (Income). Under these, there is a 'Sumă' (Amount) field with 'RON' and '0'. Below that, there is a 'Selectează categoria' (Select category) dropdown menu. Further down, there is a 'Notă' (Note) field, a date field set to '15 Aprilie 2024', and a toggle switch for 'Adaugă la raport' (Add to report).

Figura 2c Adaugă tranzacție

## 2.3 Schema de navigabilitate



Figură 3 Schema de navigabilitate

Principalele caracteristici ale aplicației sunt:

- **Gestionarea bugetului** – această funcționalitate va fi posibilă din meniul „Acasă”. Utilizatorii își pot adăuga cheltuielile/veniturile și categoria din care acestea fac parte (exemple: transport, alimentație, distracție, donații etc.)
- **Setarea bugetelor** – pentru a veni în sprijinul utilizatorilor care doresc să economisească mai mult, aceștia vor putea să seteze anumite bugete lunare pentru diferite categorii de cheltuieli. Această funcționalitate va fi disponibilă tot din meniul „Acasă”.
- **Monitorizarea bugetului** – aceasta va putea fi făcută pe zile/luni/ani, din secțiunea „Istoric”, afișându-se în pagina nou deschisă toate tranzacțiile introduse de utilizator în perioada aleasă.
- **Rapoarte** – pentru o urmărire mai clară a cheltuielilor/veniturilor, utilizatorul poate accesa meniul „Rapoarte”. Aici se vor regăsi grafice intuitive cu ajutorul cărora poate fi vizualizată gestionarea bugetului într-o anumită perioadă (predefinită sau personalizată).
- **Curs valutar** – poate fi mereu accesat din meniul cu același nume.
- **\*Investiții** – în meniul astfel denumit, utilizatorul își va putea gestiona portofoliul (Acțiuni, ETF-uri etc).
- **Educație financiară** – aplicația oferă resurse și informații utile despre concepte financiare de bază, cum ar fi economisirea, investițiile, gestionarea datoriilor etc. Aceste resurse sunt prezentate în meniul „Articole” într-un mod simplu și accesibil, astfel încât oricine să poată înțelege și să aplice principiile financiare fundamentale.
- **Setări** – Utilizatorul poate modifica oricând setările făcute la crearea contului din acest meniul. În plus, poate modifica mail-ul și parola și gestiona frecvența cu care primește notificările din partea aplicației.

## 3. IMPLEMENTARE

### 3.1 Implementarea bazei de date

Baza de date pentru aplicația noastră este construită folosind MySQL, un sistem de gestionare a bazelor de date relaționale open-source, recunoscut pentru performanța, scalabilitatea și fiabilitatea sa. MySQL utilizează limbajul standard de interogare SQL (Structured Query Language) pentru a permite manipularea și interogarea datelor în cadrul bazei de date.

Am implementat chei străine pentru a stabili relațiile între tabele și pentru a asigura integritatea referențială a datelor. Modelarea relațiilor între tabele a fost realizată cu atenție, identificând și definind corect relațiile de tip unu-la-unu, unu-la-multi între entități. De asemenea, am aplicat principiile normalizării bazelor de date pentru a minimiza redundanța și a asigura coerența și integritatea datelor.

Pentru administrarea și gestionarea bazei de date MySQL am folosit MySQL Workbench, care ne permite să realizăm operațiuni precum crearea și modificarea tabelelor și interogarea datelor.

```
CREATE TABLE `wallet`.`rapoarte` (  
  `IDraport` INT UNSIGNED NOT NULL AUTO_INCREMENT,  
  `IDUser` INT UNSIGNED NULL,  
  `IDcategorie` INT UNSIGNED NULL,  
  `DataInceput` DATE NULL,  
  `DataSfarsit` DATE NULL,  
  `Suma` INT NULL,  
  PRIMARY KEY (`IDraport`),  
  INDEX `fk5_idx` (`IDUser` ASC) VISIBLE,  
  INDEX `fk6_idx` (`IDcategorie` ASC) VISIBLE,  
  CONSTRAINT `fk5`  
    FOREIGN KEY (`IDUser`)  
      REFERENCES `wallet`.`utilizatori` (`IDUser`)  
      ON DELETE CASCADE  
      ON UPDATE CASCADE,  
  CONSTRAINT `fk6`  
    FOREIGN KEY (`IDcategorie`)  
      REFERENCES `wallet`.`categorii` (`IDcategorie`)  
      ON DELETE CASCADE  
      ON UPDATE CASCADE);
```

Figura 4. Scriptul pentru crearea tabelii „rapoarte”

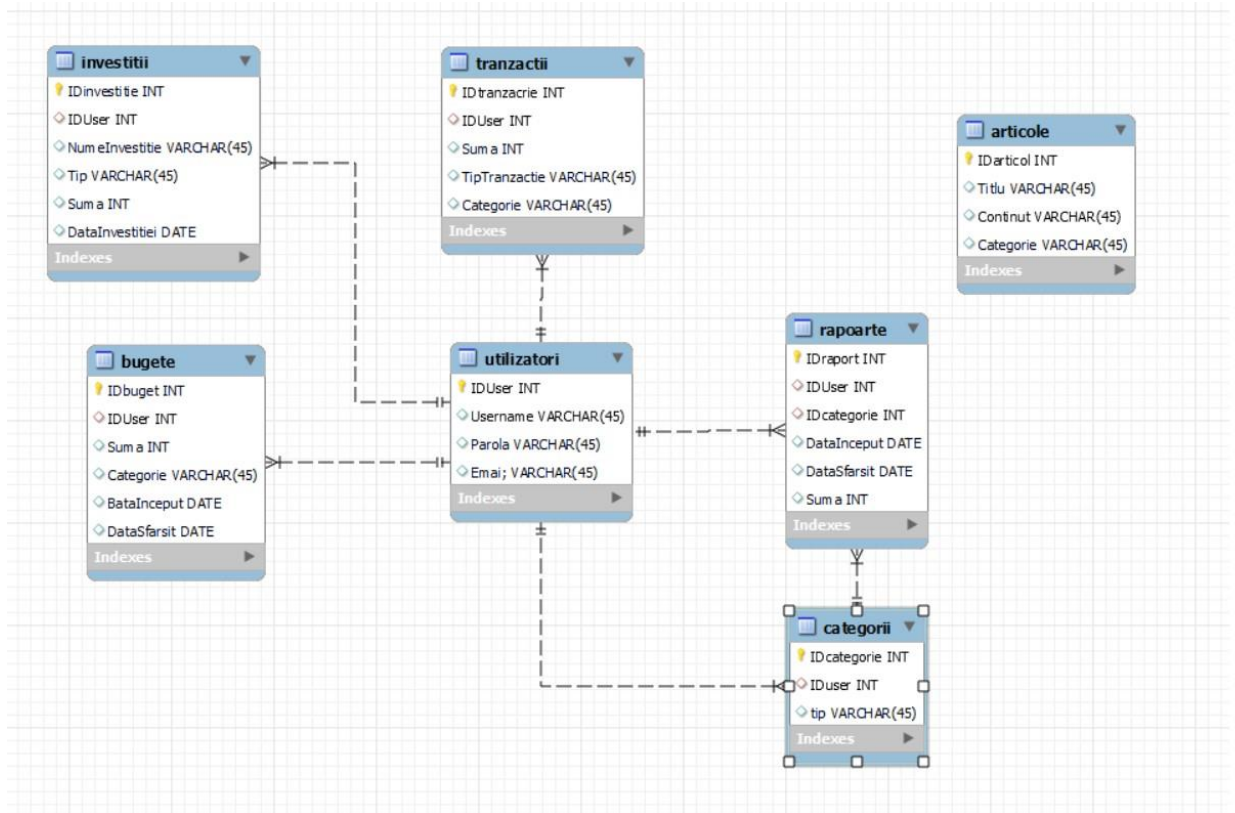


Figura 5. Diagrama EER (Extended Entity-Relationship)

### 3.2 Implementarea în Android Studio

Android Studio este un mediu de dezvoltare integrat (IDE) oficial pentru dezvoltarea aplicațiilor Android. Oferă un set de instrumente pentru scrierea, testarea și depanarea aplicațiilor Android. Include editor de cod, emulator de dispozitive, suport pentru gestionarea dependențelor, funcționalități de depanare și multe altele. Acesta oferă suport complet pentru dezvoltarea în limbajul Java și Kotlin.

Pentru partea de back-end am ales să folosim limbajul de programare Java deoarece este un limbaj orientat pe obiecte, robust, și versatil, utilizat pe scară largă pentru dezvoltarea aplicațiilor Android. Oferă o sintaxă clară și ușor de înțeles, precum și un sistem puternic de gestionare a memoriei și de tratare a excepțiilor. Limbajul Java este utilizat pentru a scrie codul logic al aplicației, inclusiv manipularea datelor, gestionarea evenimentelor și navigarea între ecrane.

Pentru partea de front-end folosim XML, un limbaj de marcare utilizat pentru a defini structuri de date și documente cu semantică proprie. Este folosit pe scară largă în dezvoltarea aplicațiilor Android pentru a defini aspectul interfeței utilizatorului (UI) în fișierele de aspect. Fișierele XML descriu aranjarea elementelor grafice, dimensiunile, stilurile și alte atribute. Android utilizează un set specific de etichete și atribute XML pentru a defini interfețe de utilizator, cum ar fi LinearLayout, RelativeLayout, TextView, EditText.

```

package com.example.tpi;

import ...

public class LoginActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.login);

        Button loginButton = findViewById(R.id.button1);
        Button registerButton = findViewById(R.id.button2);

        loginButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent( packageContext: LoginActivity.this, LoginScreen.class);
                startActivity(intent);
            }
        });

        registerButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent( packageContext: LoginActivity.this, RegisterScreen1.class);
                startActivity(intent);
            }
        });
    }
}

```

Fig.6 Codul sursă pentru ecranul de start

Acest cod Java este o clasă care reprezintă activitatea pentru pagina de start în cadrul aplicației. Acest cod controlează comportamentul activității de logare, permițând utilizatorului să navigheze către ecranele de logare și înregistrare atunci când sunt apăsate butoanele corespunzătoare.

Clasa „LoginActivity” este o subclasă a *AppCompatActivity*, care este folosită pentru a crea o activitate în cadrul aplicației. *AppCompatActivity* este o clasă din Android Jetpack care este utilizată în mod obișnuit pentru a crea activități în aplicații.

În metoda *onCreate()* se inițializează layout-ul activității folosind metoda *setContentView()* pentru a încărca layout-ul definit în fișierul XML.

Sunt inițializate doua obiecte de tip butoane pentru logare și înregistrare. Se adaugă un ascultător de butoane folosind metoda *setOnClickListener()*. Când un buton este apăsător, acest ascultător de evenimente este activat și se execută codul din interiorul lui. În interiorul ascultătorului celor două



butoane se creează un obiect de tip Intent care indică către activitatea LoginScreen, respectiv RegisterScreen.

Pentru a face posibilă comunicarea între diferite componente ale aplicației, cum ar fi activități, servicii, am importat clasa fundamentală din cadrul Android SDK, *android.content.Intent*. În cadrul codului, Intent este folosit pentru a porni o nouă activitate atunci când utilizatorul apasă un buton.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/confirm_button"
        android:layout_width="315dp"
        android:layout_height="71dp"
        android:text="Continuă"
        android:textSize="18sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/number_picker" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#E1CFFB"
        android:text="Introduceți venitul lunar"
        android:textColor="#000000"
        android:textSize="18sp"
        app:layout_constraintBottom_toTopOf="@+id/monthly_income"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.513"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <NumberPicker
        android:id="@+id/number_picker"
        android:layout_width="136dp"
        android:layout_height="52dp"
        android:background="#2F6200EE"
        app:layout_constraintBottom_toTopOf="@+id/confirm_button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView3" />

    <EditText
        android:id="@+id/monthly_income"
        style="@style/Widget.AppCompat.AutoCompleteTextView"
        android:layout_width="382dp"
        android:layout_height="58dp"
        android:layout_marginStart="12dp"
        android:layout_marginEnd="12dp"
        android:hint="Introduceți venitul lunar"
        android:hyphenationFrequency="normal"
        android:inputType="numberDecimal"
        android:shadowColor="#E1CFFB"
        android:textAppearance="@style/TextAppearance.AppCompat.Body2"
        android:textColorHighlight="@color/black"
        android:textColorHint="#9E000000"
        app:layout_constraintBottom_toTopOf="@+id/textView3"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.4"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView4" />
```

Fig.7. Codul sursă XML pentru layout-ul ecranului de introducere a venitului

Acest fișier XML definește aspectul paginii de introducere a venitului lunar în cadrul aplicației Android.

*<androidx.constraintlayout.widget.ConstraintLayout* este elementul rădăcină al fișierului XML și definește un layout de tip ConstraintLayout, care permite poziționarea și dimensionarea elementelor UI utilizând constrângeri.

Elementele Button, TextView, NumberPicker și EditText sunt etichete pentru un buton, pentru introducerea venitului lunar, un selector de numere pentru a alege ziua venitului și un camp pentru introducerea venitului lunar.

Elementul Button permite utilizatorului să confirme introducerea venitului lunar. Elementul TextView este utilizat pentru a afișa eticheta Introduceți venit lunar. Elementul NumberPicker permite utilizatorului să aleagă ziua venitului lunar, iar elementul EditText permite introducerea venitului lunar.

Elementele *android:text*, *android:textSize*, *android:layout* etc. sunt atribute XML care sunt utilizate pentru a specifica diverse aspecte ale elementelor UI în aplicații. Acestea sunt utilizate în fișierele de aspect pentru a defini aspectul și comportamentul elementelor UI, cum ar fi texte, dimensiuni, culori.



Fig.8a Ecranul de start



Fig.8b Ecranul de înregistrare

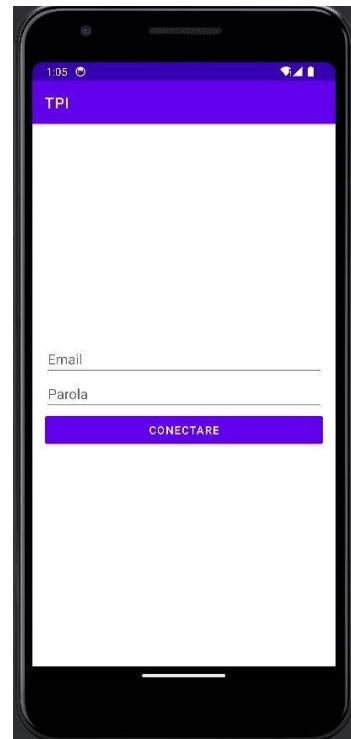


Fig.8c Ecranul de conectare



Fig.8d Ecranul pentru introducerea venitului

### 3.3 Implementarea serverului

Am implementat un server in limbajul de programare JAVA, folosind mediul de dezvoltare Eclipse, pentru gestiona procesele de gestionare a datelor.

```
public static void main(String[] args) {
    try {
        ServerSocket serverSocket = new ServerSocket(8080);
        System.out.println("Serverul asculta pe portul 8080...");
        while (true) {
            Socket clientSocket = serverSocket.accept();
            System.out.println("Clientul s-a conectat: " + clientSocket);
            BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
            PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
            String message = in.readLine();
            System.out.println("Mesaj primit de la client: " + message);

            String[] data = message.split("\\|");
            if (data.length == 4 && data[0].equals("CREATE_ACCOUNT")) {
                String email = data[1];
                String username = data[2];
                String password = data[3];

                adaugaUtilizator(username, password, email);

                out.println("Cont creat cu succes!");
            } else if (data.length == 3 && data[0].equals("LOGIN")) {
                String email = data[1];
                String password = data[2];

                if (autentificareUtilizator(email, password)) {
                    out.println("Autentificare reușită!");
                } else {
                    out.println("Eroare: Email sau parolă incorecte!");
                }
            } else {
                out.println("Eroare: Cerere invalidă!");
            }
            clientSocket.close();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Fig9. Funcția main a serverului

Metoda principală a serverului creează un ServerSocket care ascultă pe portul 8080. Intră într-o buclă infinită pentru a accepta conexiunea de la clienți. Pentru fiecare conexiune creează un Socket pentru comunicarea cu clientul. Creează un BufferedReader pentru a citi datele de la client și un PrintWriter pentru a trimite răspunsuri către client.

Pentru început am implementat două tipuri de cerere de la client, CREATE\_ACCOUNT și LOGIN.

```

public static void adaugaUtilizator(String username, String parola, String email) {
    String url = "jdbc:mysql://localhost:3306/wallet?useSSL=false";
    String username1 = "root";
    String password1 = "elenadaniela16";
    try {

        Connection connection = DriverManager.getConnection(url, username1, password1);

        PreparedStatement preparedStatement = connection.prepareStatement("INSERT INTO utilizatori (Username, Parola, Email) VALUES (?, ?, ?)");
        preparedStatement.setString(1, username);
        preparedStatement.setString(2, parola);
        preparedStatement.setString(3, email);
        preparedStatement.executeUpdate();

        System.out.println("Utilizatorul a fost adăugat cu succes în baza de date!");

        connection.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

*Fig10. Funcția adaugaUtilizator*

Metoda adaugaUtilizator() se conectează la baza de date MySQL folosind informațiile de conectare specificate, creează un PreparedStatement pentru a inserva un nou utilizator în tabelul utilizatori.

```

public static boolean autentificareUtilizator(String email, String parola) {

    String url = "jdbc:mysql://localhost:3306/wallet?useSSL=false";
    String username1 = "root";
    String password1 = "elenadaniela16";

    try {

        Connection connection = DriverManager.getConnection(url, username1, password1);

        PreparedStatement preparedStatement = connection.prepareStatement("SELECT * FROM utilizatori WHERE Email=? AND Parola=?");
        preparedStatement.setString(1, email);
        preparedStatement.setString(2, parola);
        ResultSet resultSet = preparedStatement.executeQuery();

        if (resultSet.next()) {
            connection.close();
            return true;
        }

        connection.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return false;
}

```

*Fig.11 Funcția autentificareUtilizator*

Metoda autentificareUtilizator(), se conectează la baza de date, creează un PreparedStatement pentru a selecta un utilizator din tabelă pe baza email-ului și a parolei. Execută instrucțiunea SQL și verifică dacă utilizatorul există. Se închide conexiunea la baza de date și se returnează true dacă autentificarea a avut succes, altfel se returnează false.

```

package server;

import java.net.*;

public class GetIPAddress {
    public static void main(String[] args) {
        try {
            Enumeration<NetworkInterface> interfaces = NetworkInterface.getNetworkInterfaces();
            while (interfaces.hasMoreElements()) {
                NetworkInterface iface = interfaces.nextElement();
                if (iface.isLoopback() || !iface.isUp()) {
                    continue;
                }

                Enumeration<InetAddress> addresses = iface.getInetAddresses();
                while (addresses.hasMoreElements()) {
                    InetAddress addr = addresses.nextElement();
                    System.out.println("Adresa IP: " + addr.getHostAddress());
                }
            }
        } catch (SocketException e) {
            e.printStackTrace();
        }
    }
}

```

Fig.12 Funcția pentru obținerea adresei IP

Pentru început serverul este implementat la nivel local, așa că am creat o clasă pentru obținerea adresei IP, necesară clientului pentru a se conecta.

### 3.4 Comunicarea Client – Server

Am creat clasa ServerCommunicator pentru a trimite cererile de autentificare și crearea de cont către serverul specificat, precum și pentru gestionarea răspunsurilor serverului.

Se creează o conexiune socket la severul specificat și se transmite un mesaj serverului. În funcție de tipul contextului, se construiește mesajul corespunzător pentru cererea de autentificare sau creare cont. Mesajul este transmis serverului, iar apoi se citește și se returnează răspunsul serverului.

```

public class ServerCommunicator extends AsyncTask<String, Void, String> {
    11 usages
    private Object context;
    3 usages
    public ServerCommunicator(Object context) { this.context = context; }
    @Override
    protected String doInBackground(String... params) {
        String serverAddress = "192.168.0.199";
        int serverPort = 8080;
        try {
            Socket socket = new Socket(serverAddress, serverPort);
            OutputStream outputStream = socket.getOutputStream();
            PrintWriter out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(outputStream)));
            String message = "";
            if (context instanceof LoginScreen) {
                message = "LOGIN|" + params[0] + "|" + params[1];
            } else if (context instanceof RegisterScreen1) {
                message = "CREATE_ACCOUNT|" + params[0] + "|" + params[1] + "|" + params[2];
            }
            out.println(message);
            out.flush();
            InputStream inputStream = socket.getInputStream();
            BufferedReader in = new BufferedReader(new InputStreamReader(inputStream));
            String response = in.readLine();
            socket.close();
            return response;
        } catch (IOException e) {
            e.printStackTrace();
            return "Eroare la conectare cu serverul";
        }
    }
}

```

Fig.13 Clasa ServerCommunicator

11:28 69%

TPI

utilizator\_nou@gmail.com

utilizator123

.....

CONTINUA

Fig.14 Meniul de înregistrare

11:32 69%

TPI

utilizator\_nou@gmail.com

.....

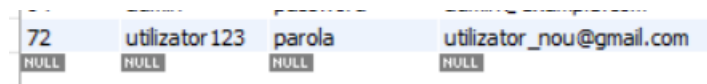
CONECTARE

Fig.15 Meniul de conectare

După apăsarea butoanelor CONTINUĂ/CONECTARE aplicația android trimite datele introduse de către utilizator către server. Serverul preia datele și în funcție de cererile clientului le procesează.

```
Server [Java Application] D:\PROGRAME\ECLIPSE\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v202
Serverul asculta pe portul 8080...
Clientul s-a conectat: Socket[addr=/192.168.0.241,port=59252,localport=8080]
Mesaj primit de la client: CREATE_ACCOUNT|utilizator_nou@gmail.com|utilizator123|parola
Utilizatorul a fost adăugat cu succes în baza de date!
Clientul s-a conectat: Socket[addr=/192.168.0.241,port=54436,localport=8080]
Mesaj primit de la client: LOGIN|utilizator_nou@gmail.com|parola
```

Fig.16 Confirmarea conectării aplicației la server



72	utilizator123	parola	utilizator_nou@gmail.com
NULL	NULL	NULL	NULL

Fig.17 Verificare că utilizatorul apare în baza de date.

Dacă conectarea sau înregistrarea s-au efectuat cu succes, utilizatorul este redirecționat către pagina “Acasă” a aplicației, unde are un meniu pentru setări, un meniu pentru adăugare tranzacție/venit și un meniu pentru investiții.

Prima pagină a aplicației este destinată introducerii venitului lunar al utilizatorului. Interfața acestei pagini este simplă și intuitivă, incluzând un câmp de text unde utilizatorul poate introduce valoarea venitului lunar. De asemenea, există un selector de zi, sub formă de picker de tip carousel, care permite utilizatorului să aleagă ziua lunii în care venitul este primit. După completarea acestor informații, utilizatorul poate apăsa butonul „Continuă” pentru a confirma datele introduse și a trece la următoarea etapă a aplicației.

Pagina principală oferă utilizatorului acces rapid la diferitele funcționalități disponibile. Aceasta include mai multe butoane pentru navigare. Butonul „Setări” permite utilizatorului să acceseze setările aplicației și să facă modificările necesare. Butonul „Adaugă Tranzacție” redirecționează utilizatorul către pagina de adăugare a unei noi tranzacții. De asemenea, există un buton „Investiții” care permite utilizatorului să gestioneze investițiile personale. În final, butonul „Adaugă Venit” permite navigarea către pagina de introducere a unui nou venit. Această structură facilitează accesul rapid și eficient la toate funcțiile esențiale ale aplicației.

Următoarea pagină este dedicată adăugării unei noi tranzacții, fie că este vorba de un venit sau de o cheltuială. Utilizatorul are la dispoziție un câmp de text pentru a introduce valoarea tranzacției și un alt câmp de text pentru a specifica categoria acesteia, cum ar fi alimente, transport sau alte cheltuieli. În centrul paginii se află un selector de dată, sub formă de calendar, care permite utilizatorului să aleagă data tranzacției. De asemenea, există opțiuni de tip butoane radio pentru a selecta dacă tranzacția este un venit sau o cheltuială. După completarea tuturor detaliilor, utilizatorul poate apăsa butonul „Adaugă Tranzacție” pentru a salva tranzacția în aplicație.

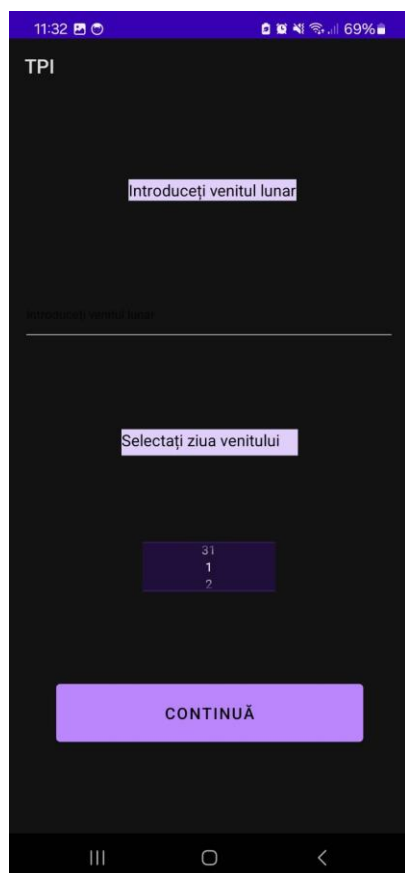


Fig. 18 Pagina de introducere a venitului lunar

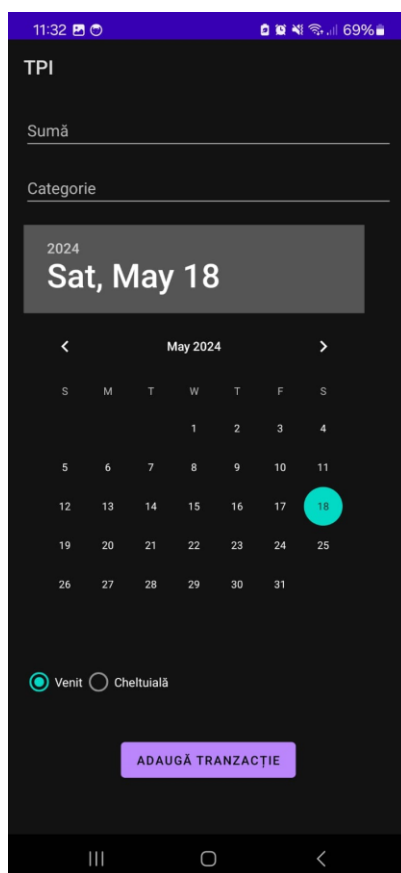


Fig. 19 Pagina „Adaugă Tranzacție”

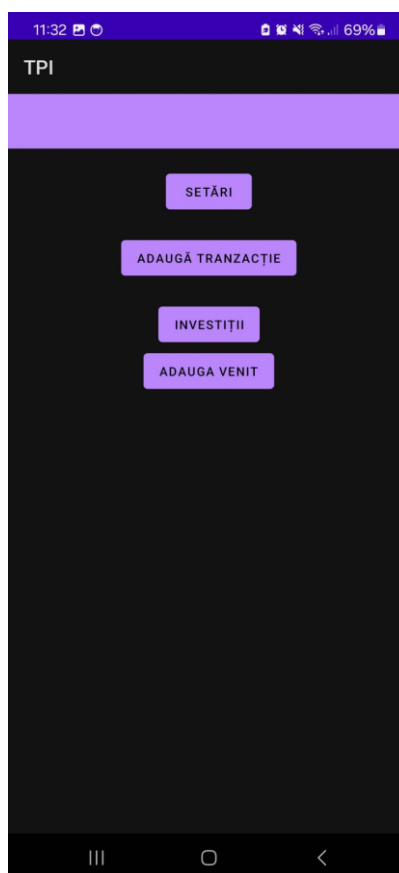


Fig. 20 Meniul principal



## 4. Concluzii

Proiectul de față de dezvoltare a unei aplicații Android pentru gestionarea finanțelor personale a avut ca scop principal facilitarea unui management financiar eficient și accesibil pentru utilizatori. Având în vedere contextul socio-economic al țării, aplicația se adresează unei nevoi reale de educație financiară și administrare corectă a resurselor personale. În acest demers, am realizat o soluție tehnologică ce integrează multiple funcționalități utile, cum ar fi gestionarea veniturilor și cheltuielilor, setarea de bugete, monitorizarea economiilor și emiterea de notificări personalizate.

Printre principalele contribuții ale proiectului se numără dezvoltarea unei interfețe grafice intuitive și prietenoase, implementarea unei baze de date relaționale MySQL pentru stocarea și gestionarea datelor utilizatorilor, și utilizarea limbajelor de programare Java și XML pentru dezvoltarea atât a părții de back-end, cât și a celei de front-end. De asemenea, am implementat un server în Java pentru a gestiona procesele de autentificare și creare a conturilor utilizatorilor, asigurând astfel o comunicare eficientă între client și server.

În ceea ce privește dezvoltarea ulterioară a aplicației, avem în vedere câteva direcții esențiale. În primul rând, dorim să extindem funcționalitățile existente prin adăugarea de noi opțiuni de personalizare și optimizare a bugetului personalizat, precum și funcționalitățile stabilite încă din primul capitol, precum crearea de rapoarte și grafice.

În al doilea rând, planificăm integrarea unor servicii financiare suplimentare, cum ar fi posibilitatea de a efectua plăți direct din aplicație și conectarea cu conturile bancare ale utilizatorilor pentru actualizarea automată a tranzacțiilor, acestea necesitând politici clare de securitate și o implementare riguroasă.

De asemenea, vom explora utilizarea tehnologiilor de inteligență artificială pentru a oferi sugestii personalizate de economisire și investiții, bazate pe comportamentul financiar al utilizatorului.

Aceste concluzii evidențiază impactul și utilitatea aplicației noastre în contextul nevoilor financiare ale utilizatorilor din România, subliniind în același timp potențialul considerabil de dezvoltare și îmbunătățire continuă a proiectului.

## 5. Bibliografie

1. [Developing Android Apps](#)
2. *Curs și laborator de Tehnologii de Programare în Internet*
3. [Android Studio. Official IDE for Android Development.](#)
4. [Java Documentation](#)
5. [MySQL Documentation](#)
6. [Cursuri ABCode](#)