Sistem de Gestiune și Rezervare Evenimente cu Autentificare

Student: Țăruș Adelina-Elena

Profesori coordonatori:

Drd. Ing. Olteanu Gabriela

Ș.I.Univ. Dr. Ing. Flangea Oana

Universtitatea Tehnică de Conctrucții București

2025

Cuprins

1. Introducere

1.1. Motivarea alegerii temei

Motivarea principală a acestui proiect constă în digitalizarea parțială a activității unui manager de restaurant specializat în evenimente. În mod tradițional, gestionarea rezervărilor și a preferințelor clienților implică procese manuale consumatoare de timp. Prin implementarea acestei soluții software, managerul beneficiază de automatizarea fluxului de informații, reducerea erorilor umane și eficiența operațională.

Automatizarea fluxului de date încadrează preluarea informațiilor de la client și organizarea lor imediată într-un format standardizat.

Reducerea erorii umane este eliminarea riscului de a pierde detalii esențiale, precum ora de închidere a barului sau tipul de flori ales.

Eficiență operațională încadrează permiterea managerului să se concentreze pe calitatea serviciilor, în timp ce sistemul preia sarcina de arhivare și securizare a contractelor.

Obiectivul principal al acestei aplicații constă în dezvoltarea unei soluții informatice integrate care să faciliteze managementul eficient al fluxului de date în cadrul unui restaurant de evenimente, prin digitalizarea parțială a proceselor de înregistrare și configurare. Proiectul urmărește implementarea unui sistem securizat de autentificare a utilizatorilor, capabil să valideze credențialele prin interfațarea cu o bază de date locală persistentă, asigurând astfel integritatea și confidențialitatea informațiilor. Totodată, aplicația vizează automatizarea generării fișelor de eveniment prin structurarea opțiunilor de business în pachete ierarhice (Bronz, Argint, Aur), permițând managerului să colecteze într-un mod organizat preferințele clienților, de la detalii logistice de bază până la cerințe

premium precum servicii foto sau meniuri personalizate. Rezultatul final urmărit este consolidarea acestor date în documente text individuale, oferind o trasabilitate clară și o evidență digitală ușor de consultat pentru fiecare contract în parte.

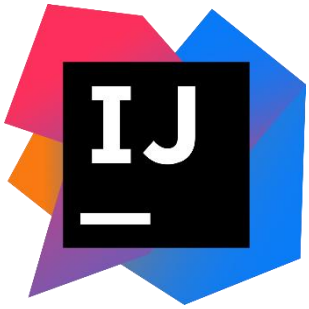
1.2. Tehnologii Utilizate

Proiectul a fost dezvoltat utilizând un set de tehnologii moderne care asigură atât eficiența scrierii codului, cât și stabitatea: Limbajul de programare Java, mediul de dezvoltare IntelliJ IDEA(JetBrains), kit-ul de dezvoltare JDK, pachetul Input/Output, pachetul Scanner, Algoritmi de căutare și validare și structuri de control decizionale

Java este un limbaj de programare de nivel înalt, cu scop general, sigur din punct de vedere al memoriei și orientat pe obiecte. Este conceput pentru a permite programatorilor să scrie o dată și să ruleze oriunde (WORA), ceea ce înseamnă că codul Java compilat poate rula pe toate platformele care suportă Java fără a fi nevoie de re-compilare. Aplicațiile Java sunt de obicei compilate în bytecode care poate rula pe orice mașină virtuală Java (JVM), indiferent de arhitectura computerului utilizat. Sintaxa limbajului Java este similară cu C și C++, dar are mai puține facilități de nivel scăzut decât acestea. Mediul de rulare Java oferă capabilități dinamice (cum ar fi reflecția și modificarea codului la rulare) care, de obicei, nu sunt disponibile în limbajele compilate tradiționale.

Java a câștigat popularitate la scurt timp după lansare și a rămas un limbaj de programare popular de atunci. Java a fost al treilea cel mai popular limbaj de programare în 2022, conform GitHub.

IntelliJ IDEA (JetBrains): Acesta a servit ca Mediu de Dezvoltare Integrat (IDE), facilitând scrierea codului prin funcții de autocompletion, refactoring și debugging avansat pentru identificarea rapidă a erorilor de sintaxă.



JDK în Java înseamnă ”Kit de dezvoltare Java”. Este o combinație de instrumente de dezvoltare software și biblioteci de suport combinate cu Java Runtime Environment (JRE) și Java Virtual Machine (JVM). Oferă mai multe instrumente și biblioteci esențiale pentru dezvoltarea aplicațiilor Java. JDK-ul conține Java Runtime Environment (JRE), un interpret (java), un compilator (javac) și câteva alte instrumente de dezvoltare.

Pachetul Input/Output oferă intrare și ieșire de sistem prin fluxuri de date, serializare și sistemul de fișiere. Dacă nu se specifică altfel, transmiterea unui argument nul către un constructor sau o metodă din orice clasă sau interfață din acest pachet va cauza generarea unei excepții NullPointerException.

Pachetul Scanner conține cadrul de lucru pentru colecții, clasele de colecții moștenite, modelul de evenimente, facilitățile de dată și oră, internaționalizarea și diverse clase utilitare (un tokenizator de șiruri de caractere, un generator de numere aleatorii și o matrice de biți).

2. Studiu de caz-Sistem de Gestiune și Rezervare Evenimente cu Autentificare

2.1. Obiectivul proiectului

Scopul aplicației este de a automatiza procesul de înregistrare a clienților și de a le permite acestora să își rezerve o dată pentru un eveniment, alegând dintr-o ofertă de pachete

Funcționalități Principale

Gestiune utilizatorilor se face prin implementarea unui meniu de tipul LogIn/SignIn reprezentat în figura 1. Aceștia trebuie să își stabilească un nume de utilizator și o parolă pe care o v-a ține minte urmat de un mesaj final de înregistrare, reprezentat de figura 2. De asemenea, programul nu lasă să existe mai mulți utilizatori cu același nume(figura 3)

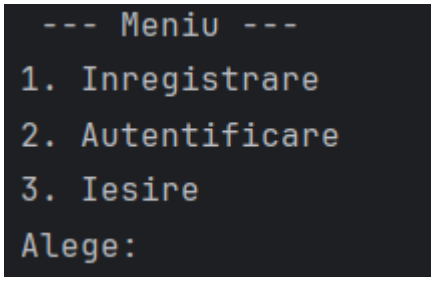


Figura 1.

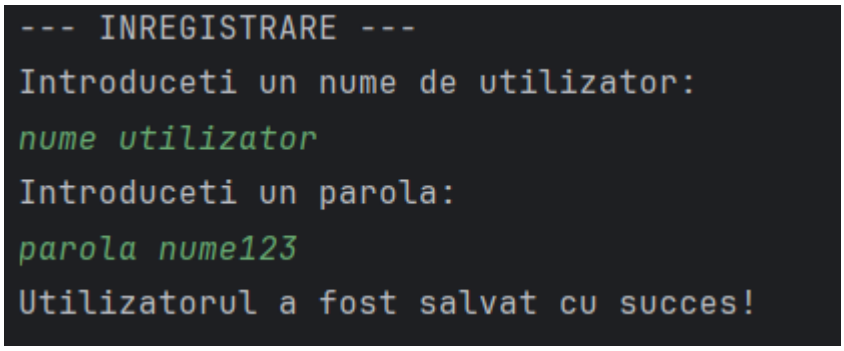


Figura 2.

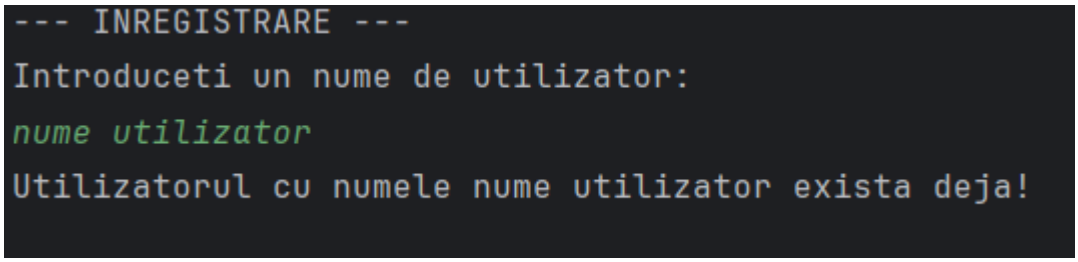


Figura 3.

După înregistrare, intră iar în meniul de alegere pentru a te autentifica și a intra în cont. După ce alegem autentificarea ne pune să scriem numele de utilizator și parola. În cazul în care avem o greșeală în numele de utilizator sau parolă programul ne va afișa o eroare(figura 4) și ne trimite în meniu.

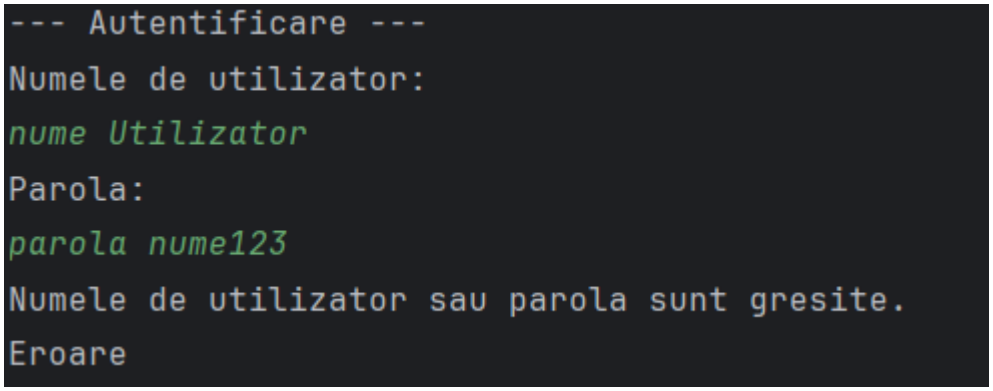


Figura 4.

Dacă utilizatorul a reușit autentificarea cu succes ne va arăta mesajul „Autentificarea a fost reușită! Bine ați venit” și ne va intra în formularul propriu zis.

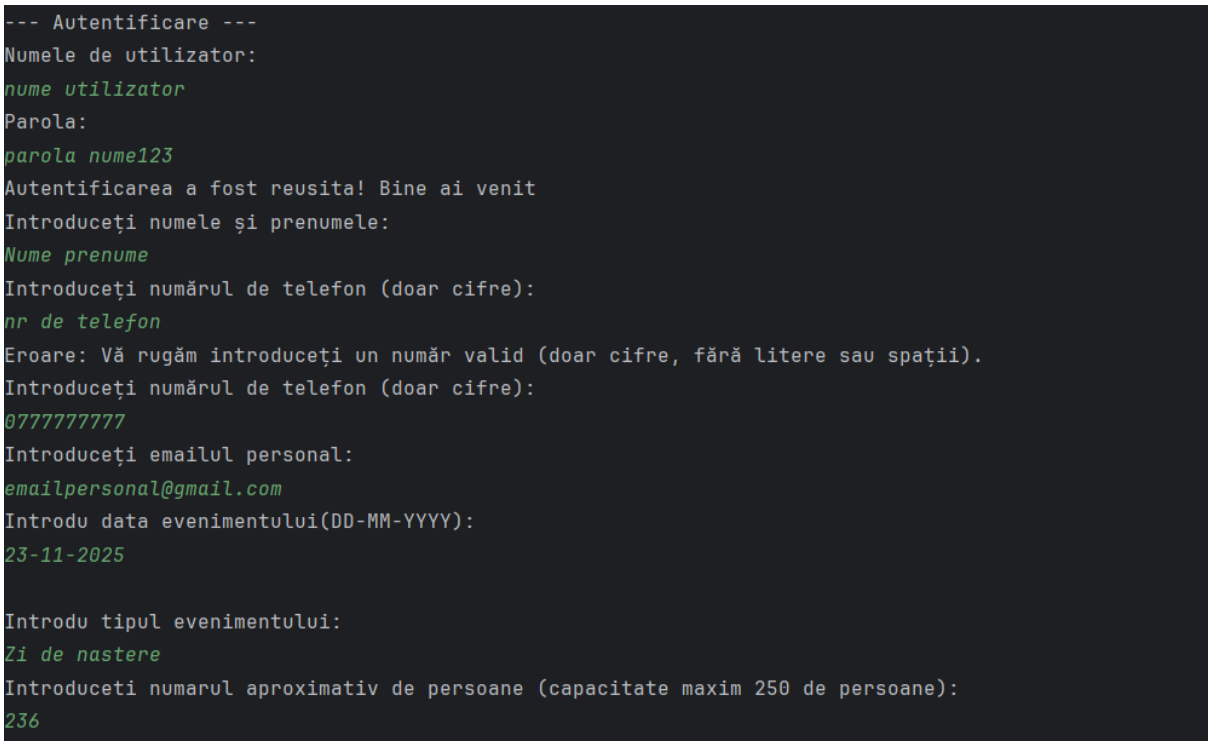


Figura 5.

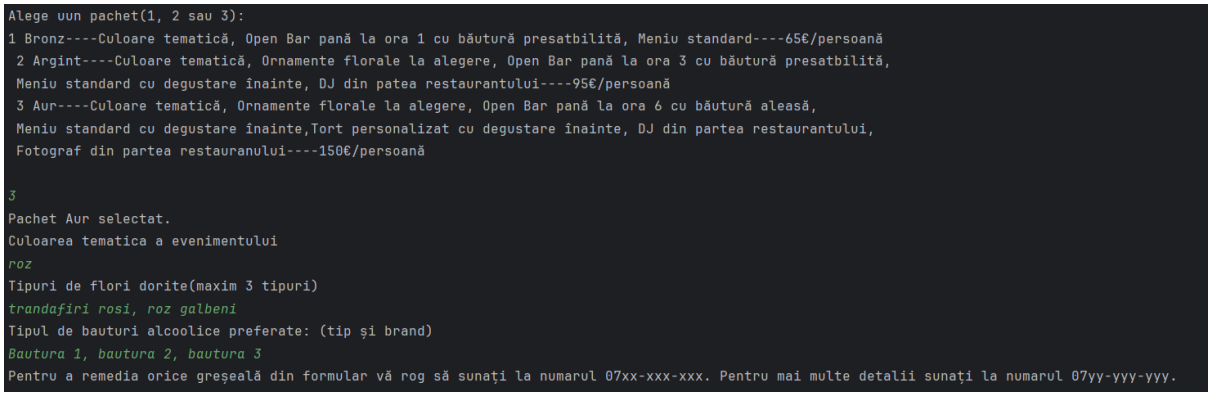


Figura 6.

Putem observa în figura 5, eroarea la introducerea unui număr de telefon invalid. De asemenea dacă utilizatorul introduce un număr care depășește capacitatea maximă de persoane ne va afișa o eroare punându-l pe utilizator să pună alt număr de persoane (figura 7).

O altă eroare mai dă atunci când pui o dată care este deja rezervată și pune utilizatorul să aleagă altă dată pentru eveniment(figura 8).

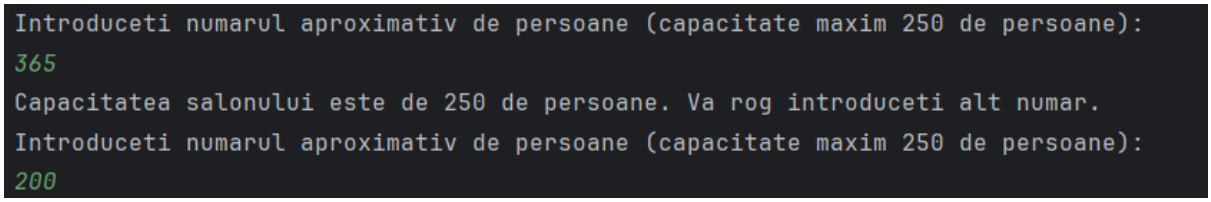


Figura7

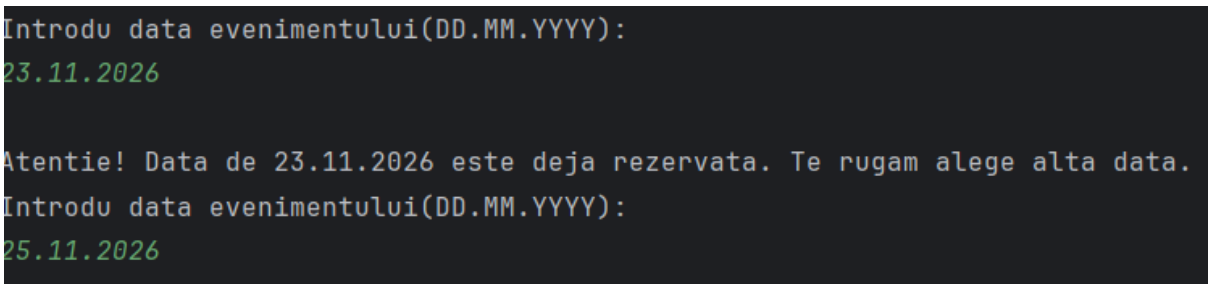


Figura 8

Fiecare pachet are opțiunile lui de la cel cu cele mai puține opțiuni de personalizat(pachetul bronz) până la cel mai complex(pachetul aur) afișate în figura 9.

```
Alege uun pachet(1, 2 sau 3):
1 Bronz----Culoare tematică, Open Bar până la ora 1 cu băutură presatbilită, Meniu standard----65€/persoană
 2 Argint----Culoare tematică, Ornamente florale la alegere, Open Bar până la ora 3 cu băutură presatbilită,
 Meniu standard cu degustare înainte, DJ din patea restaurantului----95€/persoană
 3 Aur----Culoare tematică, Ornamente florale la alegere, Open Bar până la ora 6 cu băutură aleasă,
 Meniu standard cu degustare înainte,Tort personalizat cu degustare înainte, DJ din partea restaurantului,
 Fotograf din partea restauranului----150€/persoană
```

Figura 9.

După ce trece personalizarea pachetului ești întâmpinat de mesajul „Pentru a remedia orice greșeală din formular vă rog să sunați la numărul 07xx-xxx-xxx. Pentru mai multe detalii sunați la numărul 07yy-yyy-yyy.”

În spatele acestui cod stau 3 clase. Clasa SignUpUser(figura 10) cu metodele insertData(figura 11), userExists(figura 12), SaveToTextFile(figura 13), verificareUser(figura 14 și figura 15).

```
class SignUpUser{  2 usages
    Scanner sc;  5 usages
    String numeutil;  7 usages
    String parola;  3 usages
    final String Delimiter = " , ";  3 usages
    final String numeFis = "User.txt";  3 usages

    public SignUpUser(Scanner sc){  1 usage
        this.sc = sc;
    }
```

Figura 10

Metoda insertData gestionează procesul de creare a unui cont nou în aplicație. În esență, aceasta colectează date de la tastatură, verifică dacă sunt valide și, dacă totul este în regulă, le salvează.

Verificarea se face prin metoda userExists din figura 12. La final salvează numele și parola în fișierul „User.txt” .

```java
public void insertData(){  1 usage
    System.out.println("--- INREGISTRARE ---");
    System.out.println("Introduceti un nume de utilizator: ");
    numeutil = sc.nextLine().trim();

    if(userExists(numeutil)){
        System.out.println("Utilizatorul cu numele " + numeutil + " exista deja!");
        return;
    }

    System.out.println("Introduceti un parola: ");
    parola = sc.nextLine().trim();


    if(numeutil.isEmpty() || parola.isEmpty()){
        System.out.println("Eroare: Numele de utilizator sau parola nu pot fi goale!");
        return;
    }
    SaveToTextFIle();
}
```

Figura 11

Metoda userExists are rolul de a căuta un nume de utilizator în fișierul text pentru a vedea dacă acesta este deja înregistrat. Este "motorul de căutare" care previne crearea de conturilor duplicate.

Metoda parcurge fișierul linie cu linie într-un buclă while. O împarte în bucăți folosind un separator Delimiter. Extrage prima coloană (date[0]), care este considerată a fi numele de utilizator. Compară numele din fișier cu cel primit ca parametru (userCheck). Dacă găsește o potrivire, returnează imediat true.

```java
private boolean userExists(String userCheck){  1 usage
    File f = new File(numeFis);
    if (!f.exists()){
        return false;
    }

    try(Scanner fileScanner = new Scanner(f)){
        while(fileScanner.hasNextLine()){
            String linie = fileScanner.nextLine();
            String[] date= linie.split(Delimiter);
            if(date.length >=1){
                String userDinFisier = date[0];
                if(userDinFisier.equals(userCheck)){
                    return true;
                }
            }
        }
    }catch(FileNotFoundException e){

    }
    return false;
}
```

Figura 12

Metoda SaveToTextFIle este responsabilă pentru scrierea efectivă a datelor pe hard disk. Aceasta transformă informațiile temporare din memoria RAM în date permanente în fișierul text „User.txt” .

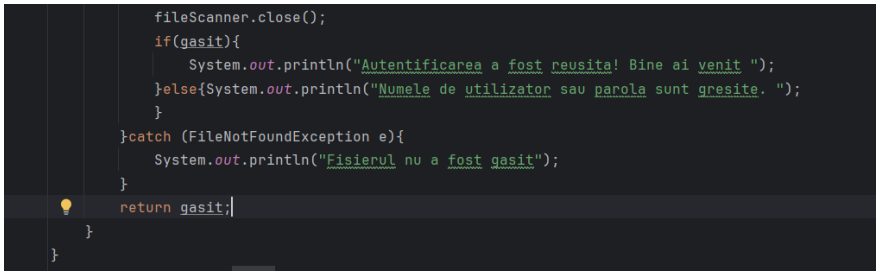
```
private void SaveToTextFIle(){  1 usage
    try{
        FileWriter fw = new FileWriter(numeFis,  append: true);
        fw.write( str: numeutil + Delimiter + parola + "\n");
        fw.close();
        System.out.println("Utilizatorul a fost salvat cu succes!");
    }catch(IOException e){
        System.out.println("Eroare la scrierea in fisier: " + e.getMessage());
    }
}
```

Figura 13

Metoda verificareUser() reprezintă sistemul de Login al aplicației tale. Aceasta verifică dacă datele introduse de cineva la tastatură se potrivesc cu "baza de date" (fișierul text) creată anterior.

```
public Boolean verificareUser(){  1 usage
    System.out.println("--- Autentificare ---");
    System.out.println("Numele de utilizator: " );
    String inputUser = sc.nextLine();
    System.out.println("Parola: " );
    String inputParola = sc.nextLine();

    boolean gasit = false;

    try{
        File f = new File(numeFis);
        if(!f.exists()){
            System.out.println("Nu exista utilizatori inregistrati inca. ");
            return false;
        }
        Scanner fileScanner = new Scanner(f);

        while(fileScanner.hasNextLine()){
            String linie = fileScanner.nextLine();
            String[] date = linie.split(Delimiter);

            if(date.length >=2){
                String userDinFisier = date[0];
                String parolaDinFisier = date[1];

                if(userDinFisier.equals(inputUser) && parolaDinFisier.equals(inputParola)){
                    numeutil = userDinFisier;
                    gasit = true;
                    break;
                }
            }
```

Figura 14



```
            fileScanner.close();
            if(gasit){
                System.out.println("Autentificarea a fost reusita! Bine ai venit ");
            }else{System.out.println("Numele de utilizator sau parola sunt gresite. ");
            }
        }catch (FileNotFoundException e){
            System.out.println("Fisierul nu a fost gasit");
        }
        return gasit;
    }
}
```

Figura 15

Mai avem clasa Fisaevenimente cu metodea „esteDataOcupata” și constructorul „Fisaevenimente”

Clasa Fisaevenimente reprezintă formularul digital de rezervare al unui eveniment (nuntă, botez, petrecere, etc.). Rolurile princilape sunt: Colectarea și validarea datelor personale și detaliile despre eveniment și gestionarea datelor, salveând toate aceste informații într-un fișier text unic, numit după numele de utilizator utilizatorul care face rezervarea (ex: andrei.txt).

Metoda esteDataOcupata cu variabila dataDeVerificat (figura 16) asigură că nu se fac două rezervări în aceeași zi.Scanează toate fișierele „ .txt” (cu excepția fișierului general de utilizatori). Deschide fiecare fișier de rezervare și verifică linia a 4-a, unde programul salvează data. Dacă găsește data introdusă de tine în fișierul altui client, returnează true (data e ocupată).

```
class Fisaevenimente {  2 usages
    Scanner sc, filescanner;  no usages
    String numeutil;  1 usage
    String nume, numarultelefon, emailpersonal, dataeveniment, numarpersoane, pacheteveniment, tipeveniment, culoareeveniment, florieveniment, bauturieveniment;  2 usages

    private boolean esteDataOcupata(String dataDeVerificat) {  1 usage
        File folder = new File( pathname: ".");
        File[] listaFisiere = folder.listFiles(( File dir,  String name) -> name.endsWith(".txt") && !name.equals("User.txt"));

        if (listaFisiere != null) {
            for (File fisier : listaFisiere) {
                try (Scanner cititor = new Scanner(fisier)) {
                    int contorLinii = 0;
                    while (cititor.hasNextLine()) {
                        String linie = cititor.nextLine();
                        contorLinii++;
                        // In structura ta, data este scrisa pe a 4-a linie a fisierului
                        if (contorLinii == 4 && linie.trim().equals(dataDeVerificat.trim())) {
                            return true;
                        }
                    }
                } catch (FileNotFoundException e) {
                }
            }
        }
        return false;
    }
```

Figura 16

Constructorul „Fisaevenimente” este inima clasei. Fiind un constructor, el rulează imediat ce creezi un obiect nou și ghidează utilizatorul prin tot procesul de completare a fișei.

```java
public Fisaevenimente(Scanner sc, String numeutil){  1 usage
    this.numeutil = numeutil;
    File f = new File( pathname: numeutil+".txt" );
    try {
        FileWriter fw = new FileWriter(f.getName(),  append: true );

        System.out.println("Introduceti numele și prenumele: ");
        this.nume = sc.nextLine();
        fw.write( str: this.nume + "\n");

        boolean telefonValid = false;
        while (!telefonValid) {
            try {
                System.out.println("Introduceti numărul de telefon (doar cifre): ");
                String input = sc.nextLine();
                Long.parseLong(input);
                this.numarultelefon = input;
                fw.write( str: this.numarultelefon + "\n");

                telefonValid = true;

            } catch (NumberFormatException e) {
                System.out.println("Eroare: Vă rugăm introduceti un număr valid (doar cifre, fără litere sau spatii).");
            }
        }
```

Figura 17

```java
System.out.println("Introdu tipul evenimentului: ");
this.tipeveniment = sc.nextLine();
fw.write( str: this.tipeveniment + "\n");


int x = 0;
boolean inputValid = false;

while (!inputValid) {
    System.out.println("Introduceti numarul aproximativ de persoane (capacitate maxim 250 de persoane): ");
    String input = sc.nextLine();

    try {
        x = Integer.parseInt(input);

        if (x > 250) {
            System.out.println("Capacitatea salonului este de 250 de persoane. Va rog introduceti alt numar.");
        } else if (x <= 0) {
            System.out.println("Va rugam introduceti un numar pozitiv.");
        } else {
            this.numarpersoane = input;
            fw.write( str: this.numarpersoane + "\n");
            inputValid = true;
        }
    } catch (NumberFormatException e) {
        System.out.println("Eroare: Va rugam introduceti un numar valid (cifre).");
    }
}
```

Figura 18

Figura 19

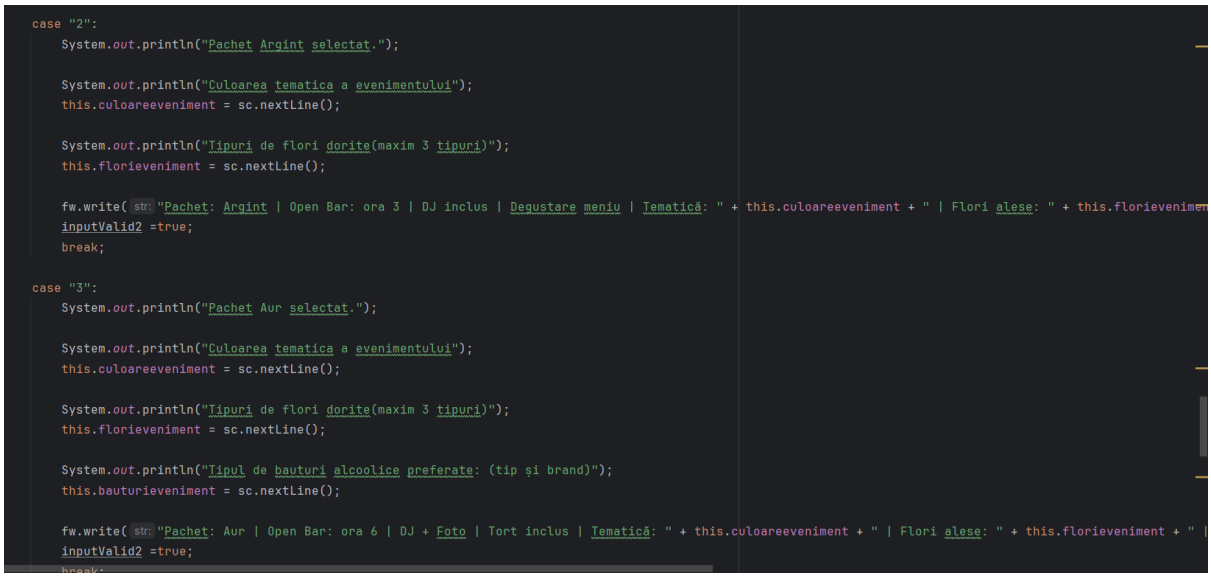


Figura 20

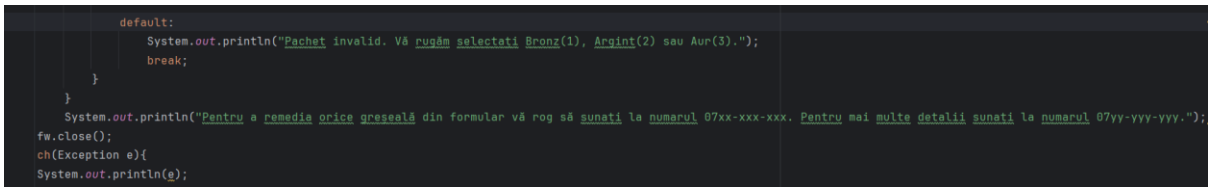


Figura 21

Clasa Main (figura 22) este punctul de pornire al întregii aplicații. Ea leagă toate componentele (SignUpUser și Fisaevenimente) într-un flux logic și interactiv.

Metoda public static void main este prima care se execută. Ea inițializează Scanner-ul pentru a citi de la tastatură pe tot parcursul programului și creează obiectul userManager (din clasa SignUpUser), care conține metodele de înregistrare și autentificare.

Bucla meniului structură menține programul deschis. Fără ea, programul s-ar închide imediat după prima acțiune. Rolul este de afișare a opțiunilor (1. Inregistrare, 2. Autentificare,

3. Iesire) și așteaptă decizia utilizatorului.Variabila meniuInitial devine false doar când utilizatorul alege "Iesire" sau când se autentifică cu succes (pentru a trece la pasul următor).

```
public class Main{
    public static void main(String[] args) throws IOException{
        Scanner sc = new Scanner(System.in);
        SignUpUser userManager = new SignUpUser(sc);

        boolean meniuInitial = true;

        while(meniuInitial){
            System.out.println("\n --- Meniu ---");
            System.out.println("1. Inregistrare");
            System.out.println("2. Autentificare");
            System.out.println("3. Iesire");
            System.out.println("Alege: ");
```

Figura 22

În secțiunea „switch(optiune)”, reprezentată în figura 23, se decide ce cod se execută în funcție de ce tasta a apăsat utilizatorul dupa cazuri.

Cazul "1" (Înregistrare): Apelează insertData() pentru a crea un cont nou. După finalizare, meniul reapare.

Cazul "2" (Autentificare) apelează „verificareUser()”, care returnează un rezultat de tip boolean (true sau false). Dacă login-ul reușește (IsValid e true) se creează un obiect nou de tip „Fisaevenimente”. Acesta pornește imediat procesul de completare a fișei de eveniment. Dacă login-ul eșuează afișează "Eroare" și te întoarce la meniu.

Cazul "3" (Ieșire): Închide programul prin setarea variabilei de control pe false.

Observă cum ai conectat datele: new Fisaevenimente(sc, userManager.numeutil). Prin acest apel, transmiți numele utilizatorului care tocmai s-a logat către clasa Fisaevenimente. Astfel, programul știe exact cum să numească fișierul text în care va salva rezervarea (ex:

nume_utilizator.txt).

```
            String optiune = sc.nextLine();

            switch(optiune){
                            case "1":
                                userManager.insertData();
                                    break;
                            case "2":
                                boolean IsValid = userManager.verificareUser();

                                if(IsValid) {
                                    Fisaevenimente fisa = new Fisaevenimente(sc, userManager.numeutil);
                                    meniuInitial = false;
                                    }else{
                                System.out.println("Eroare");
                                }
                                      break;
                             case "3":
                                 meniuInitial = false;
                                    break;
                            default:
                                System.out.println("Optiune gresita");
            }

        }
    }
}
```

Figura 23

3. Inbunătățiri

Programul încă ar avea nevoie de înbunătățiri spre exemplu crearea unei interfețe plăcute utilizatorului, criptarea fișierului „User.txt” pentru a nu fi accesat de oricine are acces la program ci a fi accesibil doar adminului și posibilitatea de a fi partajat cu mai multe persoane, completarea în acest moment făcându-se local de la un singur device.