

UNIVERSITATEA “TITU MAIORESCU” DIN BUCUREȘTI
FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

COORDONATOR ȘTIINȚIFIC:

Conf.univ.dr. Cristina Dăscălescu

ABSOLVENT:

Birta Alexandru

SESIUNEA IUNIE - IULIE

2020

UNIVERSITATEA “TITU MAIORESCU” DIN BUCUREȘTI
FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Sistem de gestiune a biletelor de avion proiectat cu
tehnologia Java

COORDONATOR ȘTIINȚIFIC:

Conf.univ.dr. Cristina Dăscălescu

ABSOLVENT:

Birta Alexandru

SESIUNEA IUNIE - IULIE

2020

Cuprins

Introducere	1
Capitolul 1.....	2
Prezentarea tehnologiilor utilizate	2
1.1 Limbajul de programare Java.....	2
1.2 Sistemul de gestiune a bazelor de date MySQL și limbajul SQL.....	3
1.3 AWT	4
1.4 Java Swing	5
1.5 JavaMail.....	6
1.6 JDBC.....	6
1.7 API-uri de tip Java Security	6
1.8 JUnit.....	7
1.9 ZXing	7
Capitolul 2.....	9
Arhitectura aplicației.....	9
2.1 Model-view-controller	9
2.2 Reprezentarea aplicației în limbajul UML.....	12
2.3 Baza de date	14
2.4 Sistem de criptare a parolelor	17
Capitolul 3.....	22
Funcționalitatea aplicației	22
3.1 Prima fereastră – welcome screen.....	22

3.2 Secțiunea de logare	24
3.3 Secțiunea de înregistrare	25
3.3 Sistemul de resetare a parolelor	29
3.4 Sistemul de căutare a biletelor de avion	32
3.5 Panoul de administrator	37
Capitolul 4.....	40
Testarea aplicației	40
4.1 Testul modului de criptare	40
4.2 Testul formularului de înregistrare	41
4.3 Testul panoului de administrator	43
Concluzii	45
Bibliografie	46

Introducere

Am fost motivat să aleg tema cu titlul „Sistem de gestiune a biletelor de avion proiectat cu tehnologia Java” din motivul de a-mi cultiva cunoștințele din domeniul programării, în special folosind limbajul de programare Java și a modelului arhitectural Model-view-controller (denumit mai des MVC) pentru a dezvolta o aplicație ce oferă o experiență completă a unui sistem de gestiune a biletelor de avion.

Aplicația este constituită din mai multe module, acestea fiind separate în două părți: partea de utilizator și partea de administrator, ambele fiind diferențiate de sistemul de logare al aplicației.

Pe partea de utilizator se poate efectua o căutare specifică în baza de date a biletelor și eventual după întocmirea unei tranzacții se poate vedea biletul primit cu toate datele acestuia, iar pe partea de administrator se pot gestiona biletele (inserare și vizualizare a tuturor biletelor) și se pot observa tranzacțiile efectuate în secțiunea de rapoarte alături de utilizatori, bilete și numărul de bilete cumpărate de fiecare utilizator împreună cu totalul cheltuit în format tabelar.

- **Capitolul 1** reprezintă prezentarea tehnologiilor folosite în scopul dezvoltării acestei aplicații. Fiecare subcapitol descrie succint fiecare componentă funcțională a aplicației și oferă exemple simple ce descriu elementele de bază a acelor tehnologii și modul acestora de funcționare.
- **Capitolul 2** subliniază detaliile de arhitectură ale aplicației, cum ar fi structurarea acesteia prin intermediul principiului de inginerie software MVC, explicarea scopului aplicației folosind o diagramă în limbajul UML și integrarea unei baze de date MySQL în sistemul aplicației, împreună cu tehnicile de criptografie utilizate în criptarea parolilor utilizatorilor.
- **Capitolul 3** oferă un ghid pas cu pas al aplicației și explicarea funcționalității fiecărui modul în parte prin intermediul unor capturi de imagine și secvențe de cod reprezentative.
- **Capitolul 4** urmărește integrarea framework-ului de testare funcțională JUnit în ciclul de viață al aplicației, împreună cu detalierea testelor implementate pentru asigurarea robusteții și a funcționării corecte a aplicației urmărind cerințele acesteia.

Capitolul 1

Prezentarea tehnologiilor utilizate

Această temă de licență este construită pe baza limbajului de programare Java, iar modulele acesteia au fost proiectate cu ajutorul principiilor programării orientate pe obiect și a pattern-ului de inginerie software Model-View-Controller (denumit adesea MVC), însoțit și de un sistem de gestiune a bazelor de date, în acest caz prin platforma MySQL, o platformă care folosește limbajul de programare SQL (cunoscut de obicei ca Structured Query Language).

Am ales să folosesc limbajul Java deoarece acesta permite dezvoltarea ușoară a back-end-ului aplicației, și în același timp din motive de interes academic și profesional, alături și de restul tehnologiilor folosite în creația aplicației.

API-ul Java Swing, a fost o decizie perfectă pentru acest tip de aplicație ce are ca bază interfețe grafice pentru interacțiunea cu utilizatori, iar utilizarea unui sistem de gestiune a bazelor de date precum MySQL a adus o performanță mare în aspectul circulației de date prin aplicație, referitor la input și la output.

Prin alte tehnologii folosite pentru a-mi finaliza aplicația se mai numără și Java Swing, Java AWT, API-ul JavaMail, API-ul JDBC (Java Database Connectivity), API-ul Java Security, framework-ul de testare funcțională JUnit și librăria de procesare a codurilor de bare în Java cu numele de ZXing (sau zebra crossing) creată de Google.

Alte librării mai mici folosite în cazuri utilitare pentru aplicația mea fac parte din pachetul „java.util”, printre care se numără librăriile: Date, Locale, Properties, ArrayList, HashMap, Base64, List și librăriile de tip IO pentru procesarea fișierelor.

1.1 Limbajul de programare Java

Java este un limbaj de programare de tip general, bazat pe clasă, orientat pe obiecte, conceput să aibă cât mai puține dependențe de implementare. Dezvoltatorii de aplicații ar

trebui să poată scrie o dată și să ruleze oriunde. Aceasta înseamnă că generarea codului Java compilat poate fi rulat pe toate platformele care acceptă Java fără a fi nevoie de recompilare. Aplicațiile Java sunt de obicei compilate în bytecode și pot fi rulate pe orice mașină virtuală Java (JVM), indiferent de arhitectura computerului de bază. Sintaxa Java este similară cu C și C ++, dar are mai puține funcții de nivel inferior față de aceste limbaje. Potrivit GitHub, Java a fost unul dintre cele mai populare limbaje de programare din 2019, în special pentru aplicațiile web pentru client-server. Se presupune că există 9 milioane de dezvoltatori Java în prezent.

1.2 Sistemul de gestiune a bazelor de date MySQL și limbajul SQL

MySQL este un sistem de gestiune a bazelor de date relaționale open source. Numele său este o combinație între „My” (numele fiicei co-fondatorului Michael Widenius) și „SQL” (abrevierea pentru Structured Query Language) [8].

MySQL este un software gratuit și open source furnizat sub Licența Publică Generală GNU și oferă, de asemenea, diverse licențe de proprietate. MySQL este deținută și sponsorizată de MySQL AB, Suedia, care a fost achiziționată de Sun Microsystems (acum Oracle Corporation) [8].

Structured Query Language (SQL) este un limbaj specific de domeniu pentru programare, utilizat pentru gestionarea datelor într-un sistem relațional de gestiune a bazelor de date (SGBD) sau fluxuri de proces într-un sistem relațional de gestiune a fluxului de date. Este deosebit de util atunci când avem de a face cu date structurate (adică date care conțin relația dintre entități și variabile).

Comparativ cu vechiul API de citire / scriere (cum ar fi ISAM sau VSAM), SQL are două avantaje principale. În primul rând, conceptul a fost introdus pentru a accesa multe înregistrări cu o singură comandă. În al doilea rând, nu mai este necesară specificarea modului de înregistrare, de exemplu: cu sau fără indice.

SQL a fost inițial bazat pe algebra relațională și calculul relațional al tuplurilor. Este format din mai multe tipuri de instrucțiuni, care pot fi clasificate în mod informativ în sub-limbaj: Data Query Language (DQL), Data Definition Language (DDL), Data Control Language (DCL) și Data

Manipulation Language (DML) [9]. Domeniul de aplicare al SQL include interogarea de date, operațiunile de date (inserarea, actualizarea și ștergerea), definirea datelor (crearea și schimbarea modului) și controlul accesului la date. Deși partea DQL a SQL este un limbaj declarativ (4GL), acesta conține și elemente procedurale.

SQL a devenit standardul American National Standards Institute (ANSI) în 1986 și a devenit standardul International Organization for Standardization (ISO) în 1987. De atunci, standardul a fost revizuit pentru a include un set de funcții mai mare. În ciuda standardelor, majoritatea codului SQL necesită cel puțin unele modificări înainte de a fi migrate către diferite sisteme de baze de date.

1.3 AWT

Abstract Window Toolkit (AWT) este setul de instrumente original Java care se bazează pe widget-uri de tip fereastră, grafici și toolkit-uri de interfață de utilizator care precedă Swing. AWT face parte din clasele Java Foundation Classes (JFC), API-uri standard utilizate pentru a furniza o interfață grafică de utilizator (GUI) pentru programele Java. AWT este, de asemenea, un set de instrumente GUI pentru multe profiluri Java ME. De exemplu, fișierele de configurare pentru dispozitivele conectate necesită runtime-ul Java pe telefon pentru a accepta Abstract Window Toolkit. Elemente de tip AWT se pot observa în figura următoare (https://en.wikipedia.org/wiki/Abstract_Window_Toolkit):



Fig. 1-1 Fereastră cu elemente de tip AWT.

1.4 Java Swing

Swing este un widget toolkit de interfețe grafice pentru utilizatori în Java. Face parte din Oracle Java Foundation Class (JFC), un API care oferă o interfață grafică de utilizator (GUI) pentru programele Java [6].

În comparație cu Abstract Window Toolkit (AWT), predecesorul Swing, acesta a fost dezvoltat pentru a oferi un set mai complex de componente GUI. Swing oferă un aspect care imita mai multe platforme și suporturi de ferestre pentru a oferi aplicații cu un aspect de bază independent de platformă. Acesta are mai multe componente puternice și mai flexibile decât AWT. Pe lângă componentele cunoscute (cum ar fi butoane, casete de selectare și etichete), Swing oferă și diverse componente avansate, cum ar fi arbori, liste, panouri cu tab-uri sau tabele.

Spre deosebire de componentele AWT, componentele Swing nu sunt implementate folosind codul specific platformei. În schimb, sunt scrise în întregime în Java și, prin urmare, sunt independente de platformă. Un exemplu de componente Swing se pot observa în figura 1-2 ([https://en.wikipedia.org/wiki/Swing_\(Java\)](https://en.wikipedia.org/wiki/Swing_(Java))).

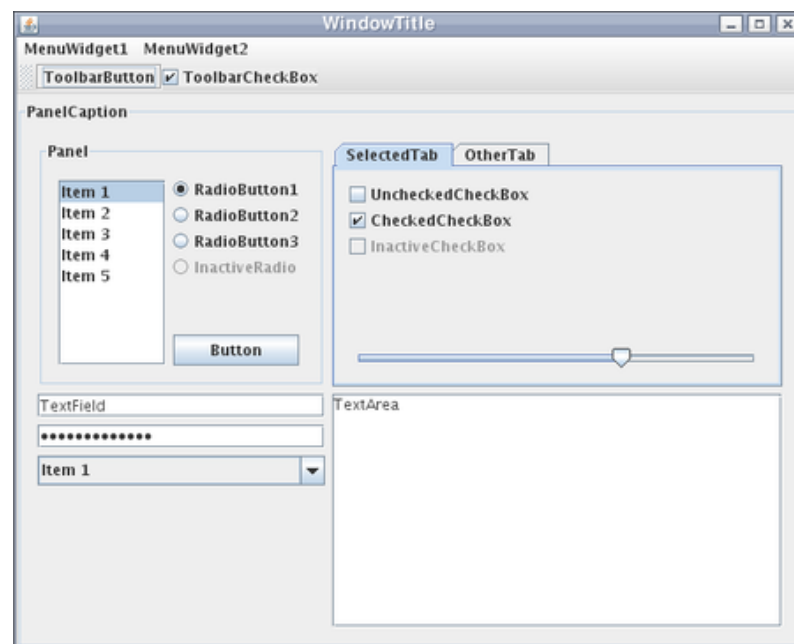


Fig. 1-2 Fereastră cu elemente de tip Java Swing.

1.5 JavaMail

JavaMail este un API Java pentru trimiterea și primirea de e-mailuri prin SMTP, POP3 și IMAP. JavaMail a fost integrat în platforma Java EE, dar oferă și un pachet software opțional este de asemenea disponibil pentru Java SE.

Versiunea actuală este 1.6.5 lansată în martie 2020. Există o altă implementare open source JavaMail - GNU JavaMail. Deși acceptă doar versiunea 1.3 a specificației JavaMail, aceasta oferă singurul backend NNTP gratuit care poate fi utilizat pentru a citi și trimite articole de tip news group.

Începând cu 2019, software-ul se numește Jakarta Mail și aparține brandului Jakarta EE (cunoscut anterior ca Java EE).

1.6 JDBC

Java Database Connectivity (JDBC) este un API al limbajului de programare Java care este utilizat pentru a defini modul în care clienții accesează baza de date. Este o tehnologie de acces de date bazată pe Java pentru conexiunea la baze de date de la diverși producători. Face parte din platforma Oracle Corporation Java Standard Edition. Oferă metode de interogare și actualizare a datelor din baza de date și este orientat către baze de date relaționale. Podul JDBC către ODBC (Open Database Connectivity) permite conectarea la orice sursă de date ODBC accesibilă în mediul gazdă Java Virtual Machine (JVM).

1.7 API-uri de tip Java Security

Platforma Java oferă multe funcții pentru a îmbunătăți securitatea aplicațiilor Java. Aceasta include utilizarea unei mașini virtuale Java (JVM) pentru a aplica restricțiile de operare, un manager de securitate care elimină codul de încredere din sistemele de operare rămase și multe API-uri de securitate pe care dezvoltatorii Java le pot utiliza. Cu toate acestea, deoarece din ce în ce mai multe programe malware au descoperit lipsurile de securitate în JVM, care ulterior nu au fost corectate de Oracle, limbajul de programare și Oracle au fost criticate.

Biblioteca clasei Java oferă multe API-uri legate de securitate, cum ar fi algoritmi criptografici standard, protocoale de autentificare și comunicații securizate.

Pentru aplicația mea m-am folosit de algoritmul de criptare cu cheie asimetrică RSA implementat cu ajutorul acestei librării.

1.8 JUnit

JUnit este un framework de testare pentru limbajul de programare Java. JUnit este foarte important pentru principiul dezvoltării bazate pe teste și aparține unei serii de cadre de testare a unităților, care se numesc colectiv xUnit, provenind de la SUnit.

În cadrul aplicației mele m-am folosit de JUnit pentru a crea teste funcționale pentru clasele de tip Java din pachetele proiectului.

Testarea funcțională este un tip de testare software în care sistemul este testat conform cerințelor/specificațiilor funcționale.

Funcțiile sunt testate prin datele de intrare și examinarea datelor de ieșire. Testarea funcțională asigură că aplicația îndeplinește corect cerințele. Acest tip de test nu implică metoda de prelucrare, ci rezultatul procesării. Acesta simulează utilizarea efectivă a sistemului, dar nu face nicio presupunere cu privire la structura sistemului.

Tehnica de testare a cutiei negre este utilizată în timpul testării funcționale, unde testerii nu sunt familiarizați cu logica internă a sistemului testat.

1.9 ZXing

ZXing („Zebra Crossing”) este o bibliotecă de procesare a imaginilor cu coduri de bare, implementată în Java, cu porturi în alte limbi. Oferă suport pentru coduri de bare de tip produs 1D, 1D industrial și 2D.

Căutarea pe Web folosește ZXing pentru a crea milioane de coduri de bare indexabile. De asemenea, este baza aplicației de scanare a codurilor de bare Android și este integrată cu produsele Google și căutarea de cărți.

Am folosit această librărie pentru a genera codurile de bare afișate pe biletul de avion generat de aplicație.



Fig. 1-3 Exemplu de cod QR generat de aplicație.

Prin urmarea citirii cu o aplicație specializată de citire a codurilor de bare, codul din figura 1-3 va afișa textul „376906810”, ce semnifică valoarea hashcode-ului obiectului de tip bilet la selectarea acestuia prin intermediul aplicației. Acesta este folosit ca identificator unic al biletului.

Capitolul 2

Arhitectura aplicației

Aplicația mea are la bază o arhitectură construită pe principiul de programare a aplicațiilor cu interfețe de utilizatori numit Model-view-controller, iar cu ajutorul acestui model am structurat logica aplicației și a pachetelor de clase create.

2.1 Model-view-controller

Model-View-Controller (denumit de obicei MVC) este un model de proiectare software care este utilizat în mod obișnuit pentru a dezvolta interfețe de utilizator și a împărți logica programelor asociate în trei elemente interconectate. Aceasta se face pentru a separa reprezentarea internă a informațiilor de modul în care utilizatorii prezintă și primesc informațiile. Acest tip de șablon este utilizat pentru a proiecta aspectul interfețelor grafice [10].

Succesul acestui model se datorează izolării între logica de afaceri și considerentele interfeței cu utilizatorul, ceea ce duce la apariția vizuală a aplicației și/sau la regulile de afaceri de nivel inferior fiind mai ușor de modificat fără a afecta alte niveluri.

Logica modelului implementată în aplicația realizată de mine este prezentă în structura pachetelor și interacțiunea dintre acestea.

Model:

Este partea centrală a principiului MVC. Indiferent de interfața cu utilizatorul, aceasta este structura de date dinamică a aplicației, gestionarea directă a datelor, logicii și regulilor aplicației.

În această parte am definit obiectele de tip utilizator și bilet, am realizat conexiunea cu sistemul de gestiune a bazelor de date folosind API-ul JDBC și am creat o clasă care se ocupă cu criptarea parolilor utilizatorilor folosind API-uri de tip Java Security.

View:

Este partea de reprezentare a informațiilor, cum ar fi diagrame, chart-uri sau tabele. Sunt posibile mai multe vizualizări ale acelorași informații, de exemplu: grafice de bare pentru management și vizualizări de tabele pentru contabili.

În cazul aplicației mele, în acest modul am introdus tot ce ține de interfețele grafice pentru utilizatori, precum și clase utilitare ce facilitează o funcționalitate mai aparte a ferestrelor și de a facilita o modularitate mai concisă a aplicației prin componente de tip panou.

Controller:

Acceptă datele de intrare și le transformă în comenzi pentru partea de model sau partea de view.

În acest modul am ales să creez anumite clase ce au rolul de a insera obiecte de tip utilizator, bilet sau tranzacție în baza de date MySQL.

Tot aici am realizat o conexiune la serviciul de poștă electronică folosindu-mă de API-ul JavaMail printr-un cont provizoriu de Gmail pentru a realiza funcția aplicației de a reseta parola unui utilizator, dacă acesta are nevoie, cu ajutorul unui cod de verificare de 6 caractere numerice trimis prin confirmare pe e-mail.

Funcționalitatea aplicației de a genera coduri de bare de tip QR code este realizată în clasa „QRCodeGenerator.java”. Această clasă poate crea un fișier de tip imagine după caracteristicile precizate de către utilizator prin cod, cum ar fi dimensiunile imaginii și calea fișierului prelucrat.

Codul QR este generat cu ajutorul unui șir de caractere, în cazul acesta este vorba despre codul „hash” al obiectului de tip bilet selectat de utilizator.

În controller se mai află și clasa „GenerateCountries.java”. Cu această clasă se creează o listă de tip ArrayList compusă din numele tuturor țărilor din lume pentru a crea mai apoi în interfața grafică o listă de tip JComboBox cu aceste nume de țări.

Pentru această facilitate a fost nevoie de clasa utilitară Java „java.util.Locale”.

Model-view-controller nu numai că împarte aplicația în aceste componente, ci definește interacțiunea dintre ele.

Modelul este responsabil de gestionarea datelor aplicației. Acesta primește datele de intrare ale utilizatorului de la controler.

View-ul înseamnă reprezentarea modelului într-un anumit format.

Controlerul reacționează la intrarea utilizatorului și interacționează cu obiectele modelului de date. Controlerul primește intrarea (opțional) ca intrare valabilă și apoi o transmite modelului.

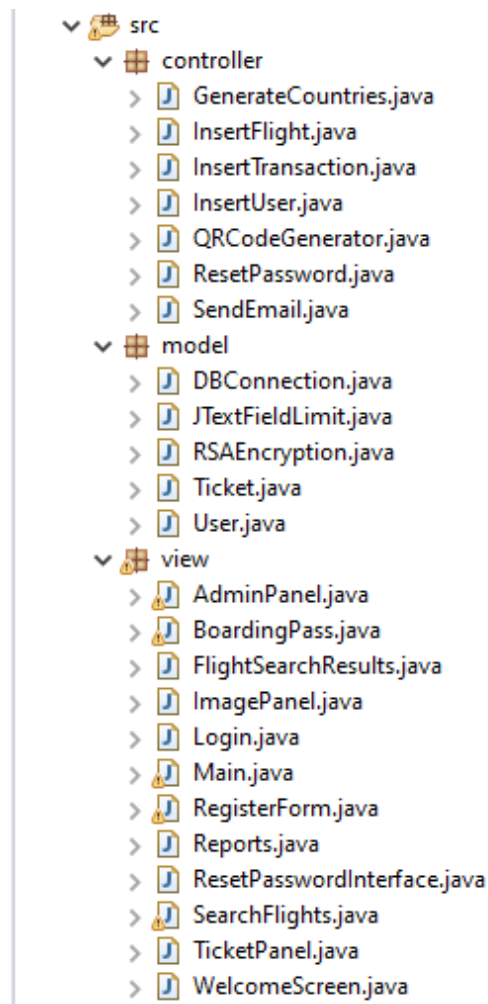


Fig. 2-1 Structura pachetelor de clase a aplicației.

2.2 Reprezentarea aplicației în limbajul UML

Unified Modeling Language (de obicei denumit ca UML) este un limbaj universal de modelare a dezvoltării în domeniul ingineriei software, care intenționează să ofere o metodă standard pentru a vizualiza proiectarea unui sistem [11].

UML a fost creat inițial din dorința de a standardiza diferite sisteme de notare și metode de proiectare software. A fost dezvoltat de Grady Booch, Ivar Jacobson și James Rumbaugh la Rational Software în 1994-1995 și a fost dezvoltat de aceștia până în 1996 [11].

În 1997, UML a fost adoptat ca standard de către Object Management Group (OMG) și a fost gestionat de această organizație încă de atunci. În 2005, UML a fost emis ca un standard ISO aprobat de Organizația Internațională pentru Standardizare (ISO). De atunci, standardul a fost revizuit în mod regulat pentru a acoperi ultimele revizii ale UML [12].

UML a început să se dezvolte în a doua jumătate a anilor '90 și a provenit din metodele de programare orientate pe obiect dezvoltate la sfârșitul anilor '80 și începutul anilor '90.

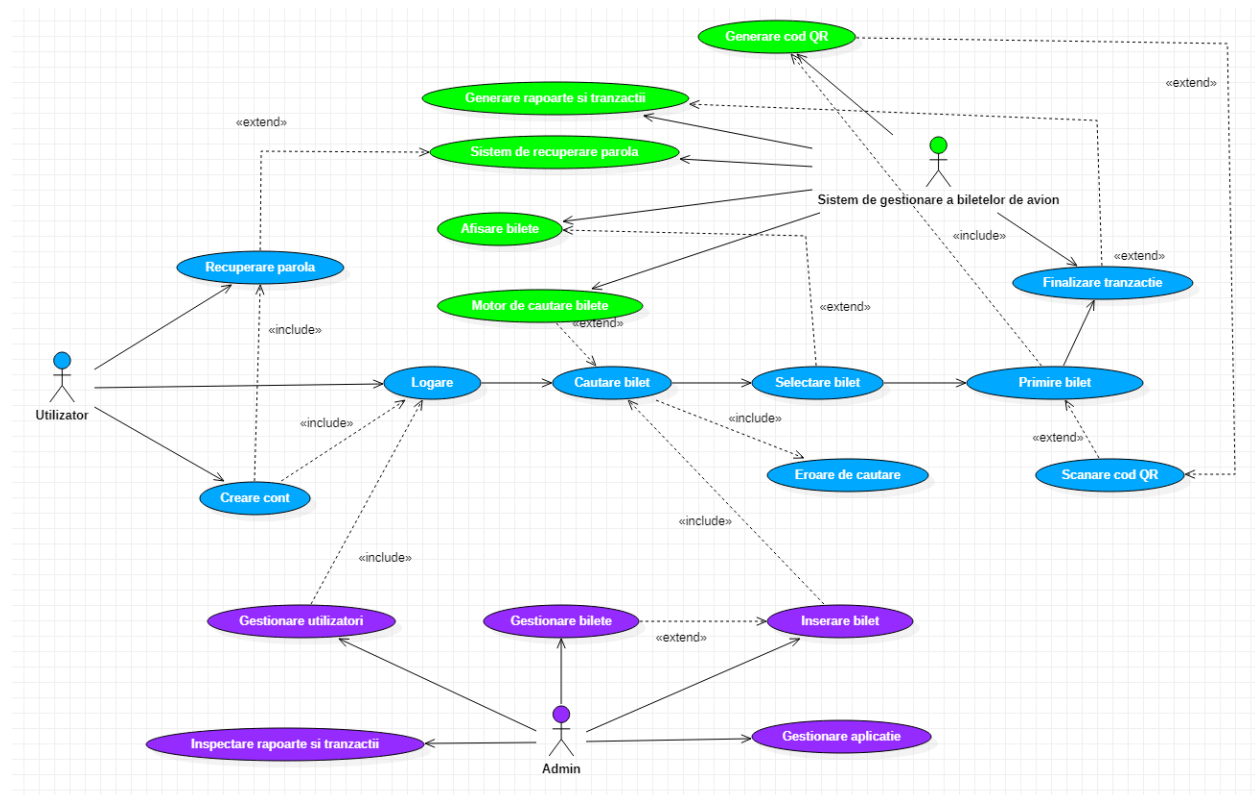
A fost inițial bazat pe metoda Booch, tehnologia de modelare a obiectelor (OMT) și notarea de dezvoltare a software-ului orientată pe obiecte (OOSE) ce s-a integrat într-un singur limbaj [13].

Cu ajutorul programului de dezvoltare a diagramelor UML numit StarUML am creat diagrama cazurilor de utilizare pentru aplicația mea.

Pe partea de utilizator am exemplificat lanțul de acțiuni pe care acesta le poate face când interacționează cu aplicația precum acțiunile de: creare de cont, recuperare de parolă, logare, căutare de bilet, întâmpinarea cu eroare la căutare de bilet, selectarea de bilet, primirea de bilet, abilitatea de a scana codul QR afișat pe bilet și în cele din urmă finalizarea tranzacției.

Pe partea de administrator am precizat rolul acestuia de a gestiona utilizatorii, de a gestiona biletele și de a le insera în baza de date a aplicației, de a inspecta rapoartele și tranzacțiile și de a gestiona aplicația.

În cele din urmă am specificat și rolul sistemului de gestiune a biletelor de avion, adică aplicația în sine, acela fiind de a oferi un motor de căutare a biletelor, a afișa biletele după căutare, de a oferi un sistem de recuperare a parolelor, de a genera rapoarte și tranzacții și de a genera codul QR.



Sistemul aplicației este foarte strâns interconectat cu utilizatorii si administratorii acesteia, lucru care se poate vedea din diagramă.

2.3 Baza de date

Pentru a reprezenta mai ușor legătura dintre tabelele bazei de date folosite pentru aplicație am dorit să folosesc sistemul de diagrame de tip Diagramă entitate-relație (sau DER).

Aceasta a putut fi creată cu ajutorul programului numit MySQL Workbench, luând datele din tabele și afișându-le sub forma de diagramă pentru a observa în ansamblu legăturile dintre tabele.

În acest caz tabelele „tickets” și „users” fac parte dintr-o relație de tipul one-to-many cu tabela „transactions”.

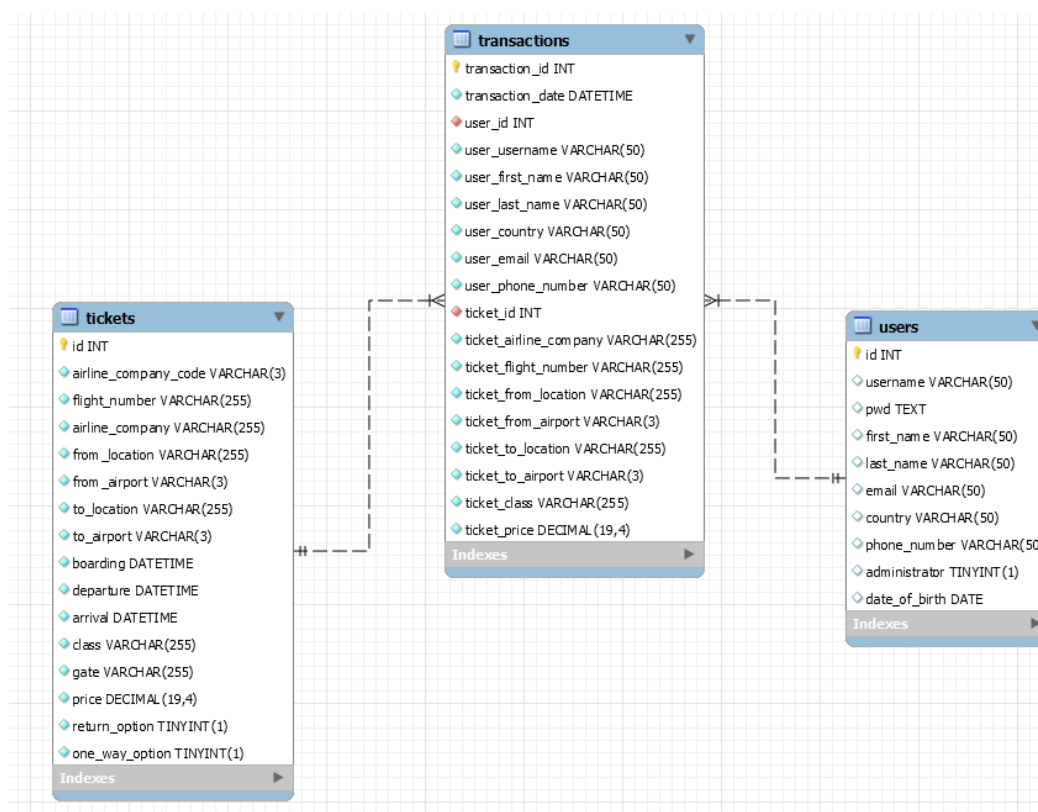


Fig. 2-3 Diagrama entitate-relație a bazei de date.

Baza de date în sistemul de gestiune a bazelor de date MySQL este scrisă sub numele de „flights”, iar tabelele acestora sunt numite „tickets”, „transactions” și „users”.

Tabela „tickets”:

Această tabelă se ocupă cu stocarea informațiilor legate de datele biletelor cu câmpurile:

- „id” - cheia primară a tabelului și identificator unic al fiecărui bilet în parte
- „airline_company_code” – codul companiei aeriene
- „flight_number” – identificator unic al zborului
- „airline_company” – numele companiei aeriene
- „from_location” – locația de unde va pleca avionul
- „from_airport” - aeroportul de unde va pleca avionul
- „to_location” - locația unde va ajunge avionul
- „to_airport” - aeroportul unde va ajunge avionul
- „boarding” – data de îmbarcare
- „departure” – data de plecare
- „arrival” - data de sosire
- „class” – clasa aleasă
- „gate” – poarta din aeroport
- „price” – prețul biletului
- „return_option” – opțiunea de reîntoarcere
- „one_way_option” – opțiunea de sens unic

Tabela „users”:

Această tabelă se ocupă cu stocarea informațiilor legate de datele utilizatorilor cu câmpurile:

- „id” – cheia primară a tabelului și identificator unic al fiecărui utilizator în parte
- „username” – aliasul utilizatorului
- „pwd” – parola utilizatorului în formă criptată
- „first_name” – prenumele utilizatorului
- „last_name” – numele utilizatorului
- „email” - adresa de email a utilizatorului
- „country” - țara utilizatorului
- „phone_number” – numărul de telefon al utilizatorului
- „administrator” – câmp ce denotă starea contului de tip utilizator sau administrator
- „date_of_birth” – data de naștere a utilizatorului

Tabela „transactions”:

Este o tabelă compusă din celelalte două tabele, și are rolul de a păstra date în legătură cu tranzacțiile efectuate de către utilizatori, oferind și date despre biletele cumpărate.

- „transaction_id” - cheia primară a tabelului și identificator unic al fiecărei tranzacții în parte
- „transaction_date” – data la care s-a efectuat tranzacția
- „user_id” – cheie străină a tabelului și identificator unic al fiecărui utilizator în parte
- „ticket_id” – cheia străină a tabelului și identificator unic al fiecărui bilet în parte

Restul câmpurilor se repetă din celelalte două tabele, dar în această tabelă au fost introduse doar unele câmpuri specifice, nu doar ambele tabele în întregime.

Prin intermediul API-ului Java Database Connectivity (JDBC) am conectat această bază de date cu aplicația, iar prin intermediul unor clase create de mine am realizat instrucțiuni de inserare și căutare în baza de date pentru a afișa cu ajutorul interfeței grafice biletele căutate de utilizatori.

2.4 Sistem de criptare a parolelor

Câmpul denumit „pwd” din tabela „users” este câmpul unde se stochează parolele utilizatorilor atunci când aceștia își creează un cont nou pe platforma aplicației. În acesta nu se pot vedea efectiv valorile adevărate ale parolelor ci doar un text criptat. Acest lucru a fost făcut posibil prin utilizarea unui sistem de criptare a parolelor folosind algoritmul de criptare RSA (Rivest–Shamir–Adleman), un algoritm cu cheie asimetrică.

Criptografia este dedicată dezvoltării și analizei algoritmilor care împiedică accesul unor persoane nedorite la informații secrete. Confidențialitatea, integritatea datelor și verificarea identității sunt diverse aspecte ale securității informațiilor esențiale pentru criptografia modernă [14].

Criptografia computerizată folosește algoritmi similari cu cei utilizați în criptografia tradițională, adică transpunerea și înlocuirea, în schimb în acest domeniu cu ajutorul tehnologiei mai avansate algoritmii folosiți au fost dezvoltați la un nivel mai înalt și complex. Prin urmare, atunci când sunt interceptate un număr mare de cifruri, criptanalistul nu poate extrage informații relevante din textul clar sau cifru fără a folosi cheia secretă [14].

Decriptarea este aceeași tehnică, dar inversată, adică conversia de la text criptat în text clar. Cifrul este o pereche de algoritmi de criptare sau decriptare, iar funcționarea complexă a cifrurilor este controlată de parametrii numiți chei, ele fiind private sau publice [14].

O altă ramură a criptografiei mai este și analiza criptografică, sau denumită mai des criptanaliză. Aceasta este o metodă de a obține semnificația tehnicilor de criptare fără a obține informații secrete. De obicei, aceasta presupune găsirea unei chei secrete. Termenul de analiză criptografică este, de asemenea, utilizat în mod obișnuit pentru a face referire la orice încercare de a ocoli mecanismele de securitate ale diferitelor tipuri de algoritmi criptografici, nu doar pentru a cripta informații [14].

În dezvoltarea tehnologiei moderne de criptare, au fost dezvoltate trei tipuri de algoritmi de criptare printre care se numără: algoritmul de criptare simetric, unde cheia este utilizată pentru criptare și decriptare; algoritmul criptografic asimetric, care constă în cheia publică folosită pentru criptare în timp ce o altă cheie secretă este folosită pentru decriptare (criptografia cu chei publice).

Al treilea tip de algoritm criptografic este de tip funcție hash, care reprezintă transformarea matematică ireversibilă a informațiilor criptate prin intermediul semnăturilor digitale. Este utilizat în special pentru a asigura integritatea mesajelor [14].

RSA (Rivest-Shamir-Adleman) este un algoritm folosit de computerele moderne pentru criptarea și decriptarea mesajelor. Algoritmul a fost dezvoltat în 1977 și a fost lansat de către Ron Rivest, Adi Shamir și Leonard Adleman al MIT în 1978 și a fost numit după inițialele celor trei autori [7]. Este un algoritm criptografic asimetric. Asimetric înseamnă că există două chei distribuite diferite. Aceasta este cunoscută și sub denumirea de criptare a cheilor publice, deoarece cheia poate fi distribuită oricui. Cealaltă cheie trebuie ținută secretă. Algoritmul se bazează pe faptul că este dificil să găsești factori cu numere complexe mari: dacă factorii sunt numere prime, problema se numește factorizare primă. Este, de asemenea, un generator de perechi de chei (chei publice și private).

Într-o schemă de criptare a cheilor asimetrice, oricine poate utiliza o cheie publică pentru a cripta un mesaj, dar numai proprietarul cheii private împerecheate îl poate decripta. Securitatea depinde dacă cheia privată este ținută secretă sau nu.

RSA include cheia publică și cheia privată. Toată lumea știe cheia publică - este folosită pentru criptarea mesajelor. Mesajele criptate cu cheia publică pot fi decriptate numai cu cheia privată. Generarea de chei a algoritmului RSA este următoarea [5]:

1. Se generează două numere prime, de preferat mari, p și q
2. Se calculează $n = pq$ și $\phi = (p-1)(q-1)$
3. Se alege un întreg aleator e , $1 < e < \phi$, astfel încât $\text{cmmdc}(e, \phi) = 1$. Perechea (n, e) este cheia publică.
4. Folosind algoritmul lui Euclid extins, se calculează întregul d , unicul cu proprietatea că de este congruent cu $1 \bmod \phi$. (n, d) constituie cheia secretă.

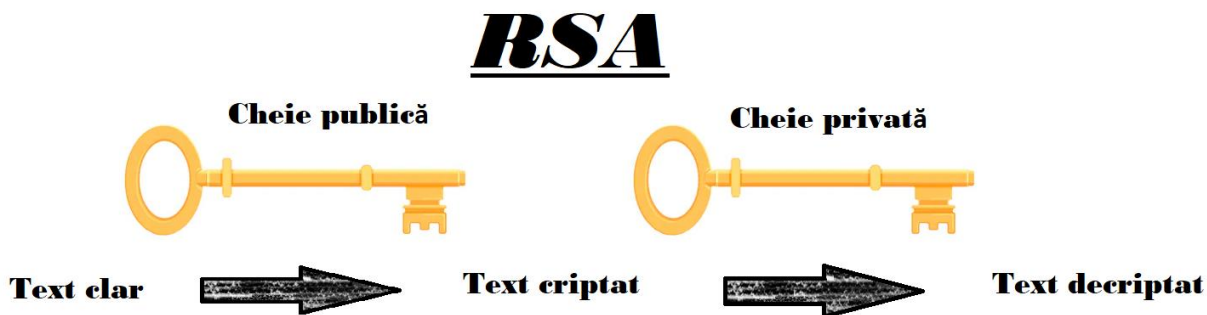


Fig. 2-7 Modul de criptare prin RSA simplificat.

După cum se poate observa din figura 2-7, se utilizează o cheie privată pe care numai persoana care o comunică o poate cunoaște și cheia publică oferită terților care folosesc algoritmul de criptare, acestea fiind un parametru al funcției de criptare și decriptare. Se utilizează funcția de criptare a cheii publice pentru a converti mesajul clar într-un mesaj criptat și se obține mesajul clar folosind cheia privată.

În arhitectura aplicației, acest sistem de criptare își are locul în modulul „model”, în clasa „RSAEncryption.java”.

Clasa specifică criptării folosește un sistem de stocare a cheilor private și publice prin conceptul de programare orientată pe obiect numit serializare și deserializare.

Serializarea este un mecanism care transformă starea unui obiect într-un flux de octeți. Deserializarea este procesul invers, în care fluxul de octeți este utilizat pentru a recrea obiectul Java real în memorie. Acest mecanism este utilizat pentru stocarea obiectelor.

Pentru a crea obiecte Java serializabile, trebuie implementă interfața „java.io.Serializable”. Aceasta este doar o interfață de tip marker care spune platformei Java că obiectele pot fi serializate.

Serializare

Deserializare

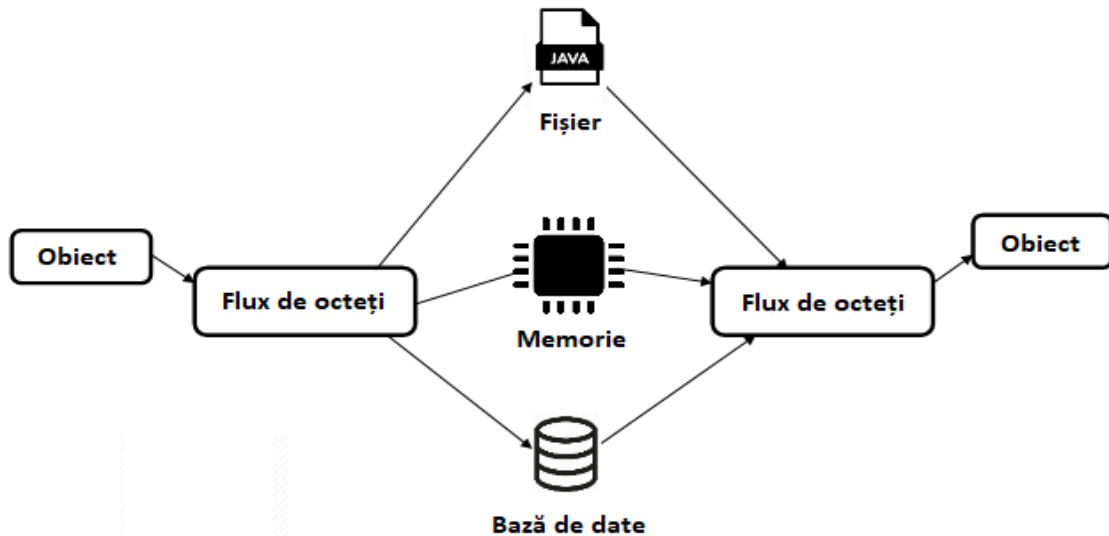


Fig. 2-8 Procesul de serializare simplificat.

Deserializarea este exact opusul. Pe scurt, serializarea convertește obiectul într-un flux de octeți, iar deserializarea va reconstrui obiectul în fluxul de octeți. Serializarea cu API-ul Java oferă funcții de serializare și deserializare.

Fluxul de octeți creat este independent de platformă. Prin urmare, obiectele serializate de pe o platformă pot fi deserializate pe o altă platformă.

Anumite clase la nivel de sistem (cum ar fi Thread, OutputStream și subclasele sale și Socket) nu pot fi serializate. Dacă clasa serializabilă conține astfel de obiecte, acestea trebuie marcate ca "transient".

Doar obiectele din aceste clase care implementează interfața java.io.Serializable pot fi serializate.

Serializable este o interfață de marcare (nu are elemente de date și nici o metodă). Este utilizată pentru „etichetarea” claselor Java, astfel încât obiectele din aceste clase să primească anumite funcții. Alte exemple de interfețe de etichetare includ: Cloneable sau Remote.

Serializarea la runtime atribuie un număr de serie fiecărei clase serializabile, care se numește `SerialVersionUID`. Aceasta se folosește în timpul deserializării pentru a verifica dacă expeditorul și receptorul obiectului serializat au încărcat clasa obiectului serializat. Dacă destinatarul încarcă clasa obiectului și UID-ul obiectului este diferit de UID-ul clasei expeditorului corespunzător, deserializarea va provoca o excepție de tipul `InvalidClassException`. Clasele serializabile pot declara în mod explicit propriul UID prin declararea numelor de câmp.

UID-ul trebuie să fie static, final și de tip long.

Dacă acea clasă serializabilă nu declară în mod explicit `serialVersionUID`, runtime-ul de serializare va calcula timpul de rulare standard pentru clasă în funcție de diverse aspecte ale clasei, așa cum este descris în Specificația de Serializare a Obiectelor Java. Cu toate acestea, se recomandă cu tărie ca toate clasele serializabile să declare în mod explicit valoarea `serialVersionUID`, deoarece calculele lor sunt foarte sensibile la detaliile clasei, în funcție de implementarea compilatorului. Orice modificare a clasei sau utilizarea altor ID-uri pot afecta datele serializate.

În modulul de criptare al aplicației am ales să serializez obiectele de tip `PublicKey` și `PrivateKey` pentru a da accesul administratorilor de a distribui eventual aceste chei prin intermediul unor fișiere binare.

Fișierele care stochează aceste chei sunt denumite „private.key”, și respectiv „public.key”. Aceste fișiere în sine nu se pot descifra cu ochiul uman, ci doar de către mașina virtuală Java prin procesul de deserializare.

Capitolul 3

Funcționalitatea aplicației

În acest capitol voi începe detalierea procesului de funcționalitate al aplicației și parcurgerea acesteia pas cu pas din punctul de vedere al unui utilizator sau al unui administrator.

Aplicația are rolul de a furniza un sistem de gestiune a biletelor de avion pentru orice tip de călătorie aeriană și cu destinații oferite pe tot globul, utilizatorii având alegerea de a căuta dintr-o bază de date după criterii specifice pentru a găsi un bilet de avion la un preț convenabil.

După ce utilizatorul și-a ales biletul dorit, acestuia i se oferă un bilet de tip boarding pass unde se pot vedea toate datele referitoare la călătorie și un cod QR.

Pe partea de administrator există opțiunea de a interoga baza de date pentru a introduce un nou bilet de avion în sistem, iar ca funcționalitate secundară mai există și partea de gestiune a rapoartelor unde administratorul poate vedea lista clienților, lista biletelor, lista tranzacțiilor efectuate și eventual o listă unde se pot vedea toate tranzacțiile efectuate de către fiecare utilizator și totalul acestora.

Țin să menționez că aplicația a fost dezvoltată în întregime cu ajutorul limbajului de programare Java și a API-urilor pentru interfețe grafice pentru utilizatori și anume Java Swing și AWT (Abstract Window Toolkit), împreună cu API-urile de Java Security pentru criptare, Google ZXing pentru generarea codului de tip QR și Java JDBC pentru a menține conexiunea la baza de date MySQL server.

3.1 Prima fereastră – welcome screen

De prima dată când deschide un utilizator aplicația, acel utilizator este întâmpinat de o fereastră (figura 3-1) în care sunt poziționate în partea de jos a ecranului trei poze a unor destinații populare de călătorie, iar mai sus o bară de meniu cu logo-ul aplicației în partea din stânga și trei butoane ce îndeplinesc diverse funcții pe partea dreaptă.

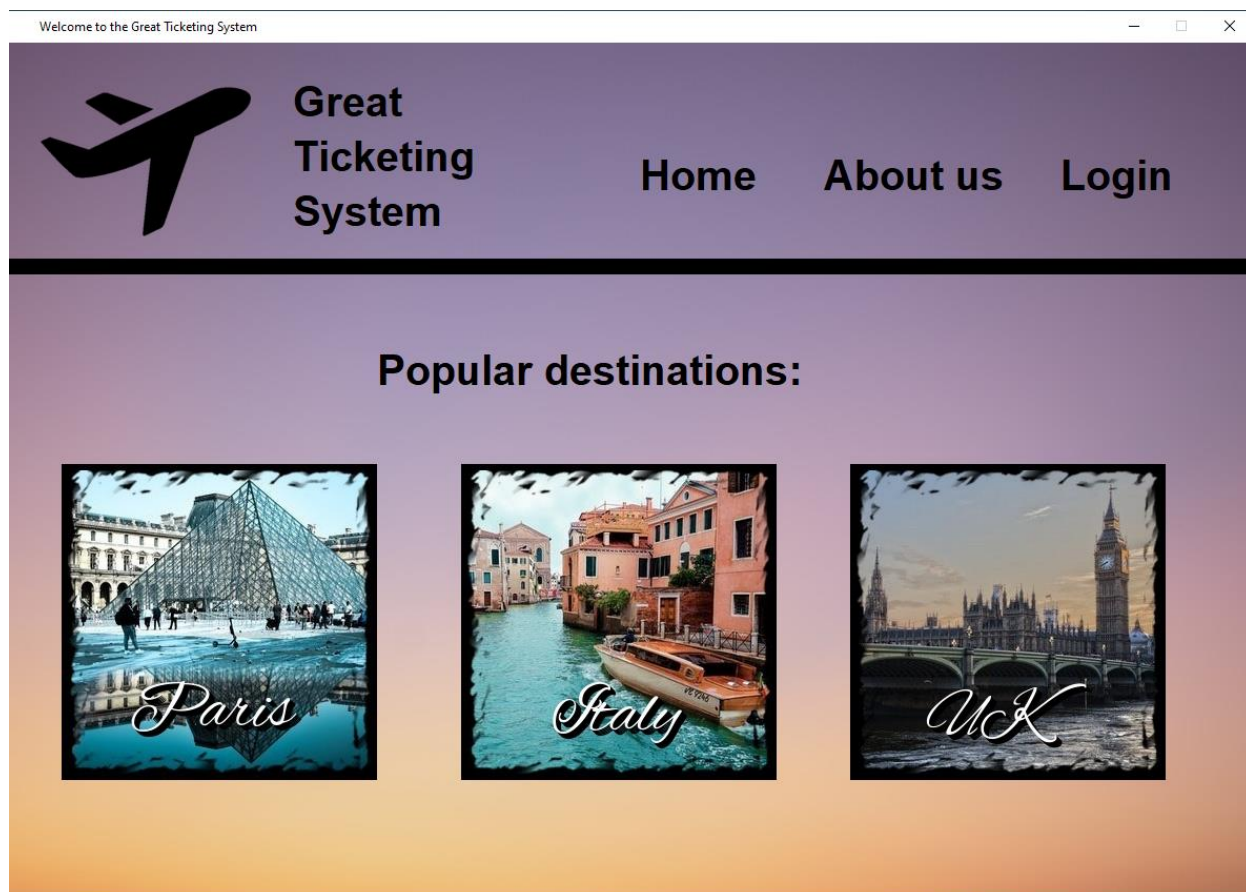


Fig. 3-1 Pagina inițială de start a aplicației.

În partea de sus este afișată o pictogramă a unui avion și numele aplicației: „Great Ticketing System”, un nume ales de mine pentru a transmite brand-ul aplicației mai departe. Tot în această parte este aflat și bara de meniu cu butoanele „Home”, „About us” și „Login”.

Butonul „Home” readuce utilizatorul la pagina inițială de când s-a lansat aplicația, iar butonul „About us” duce utilizatorul pe pagina următoare (figura 3-2):

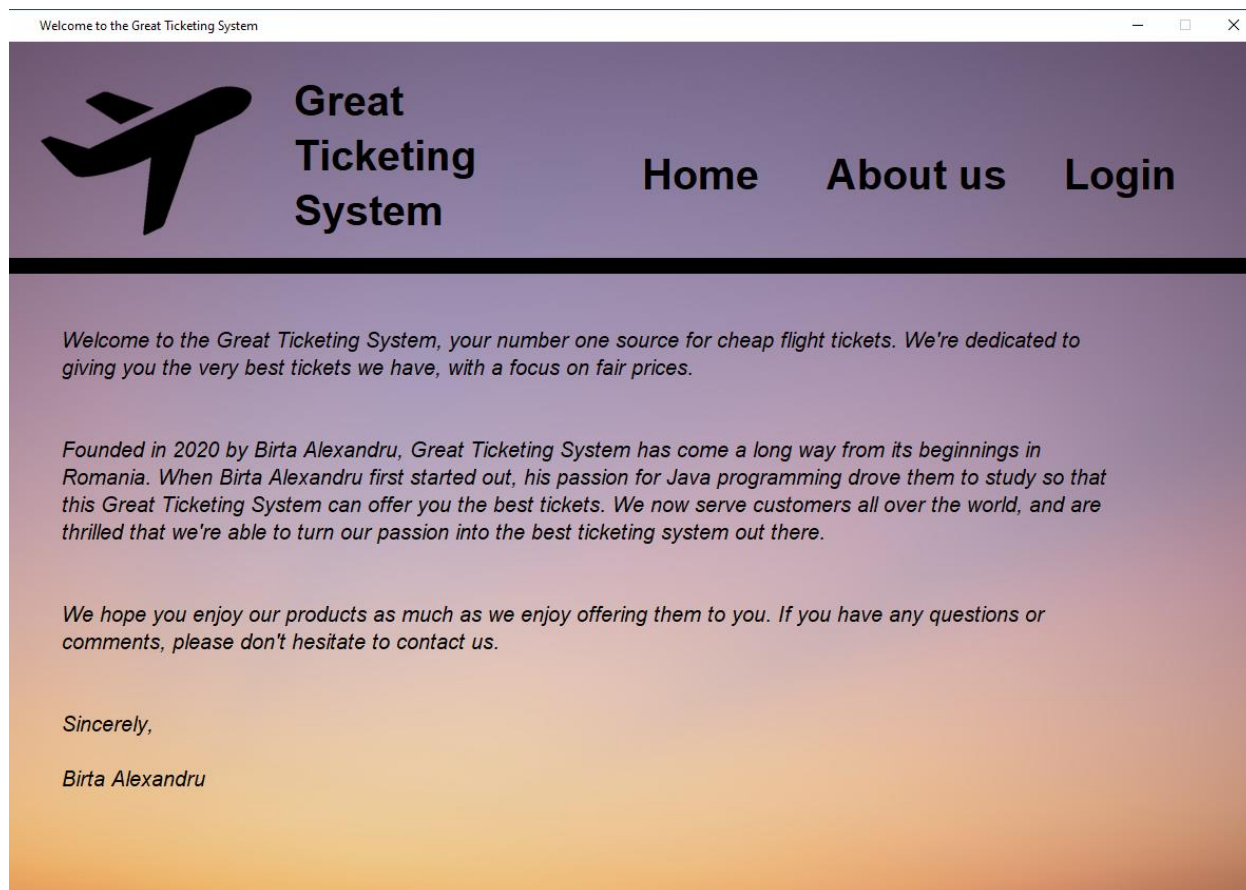


Fig. 3-2 Secțiunea „About us”.

Am ales să ofer un tip de introducere și o mică poveste a aplicației pentru fiecare utilizator ce folosește de prima dată această aplicație, ce se poate observa din figura 3-2. Este evident că acest text este unul generic și în mare parte este de o categorie fictivă dar în cazul lansării acestei aplicații pe un market adevărat, pe această pagină se poate schimba acest text cu unul ce cuprinde cu adevărat valorile aplicației și detalii reale despre aceasta.

3.2 Secțiunea de logare

Mai apoi dacă utilizatorul apasă pe butonul de „Login”, acesta este trimis pe secțiunea de logare a aplicației unde poate să își introducă datele pentru a se loga și poate să își recupereze parola printr-un sistem de poștă electronică unde i se trimite un cod de 6 caractere pentru a își crea

o nouă parolă pentru contul personal. Utilizatorul mai are acces și la pagina de înregistrare tot de pe secțiunea de logare.

Toate acestea se pot observa din figura următoare:

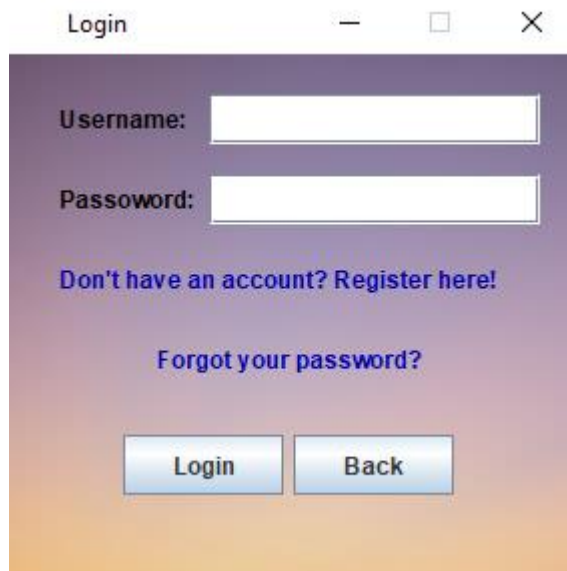
The image shows a login window with a purple-to-orange gradient background. At the top, the title bar says 'Login' with standard window controls. Below the title bar, there are two white input fields. The first is labeled 'Username:' and the second is labeled 'Password:'. Under the password field, there are two blue links: 'Don't have an account? Register here!' and 'Forgot your password?'. At the bottom, there are two light blue buttons with black text: 'Login' and 'Back'.

Fig. 3-3 Secțiunea de logare a aplicației.

3.3 Secțiunea de înregistrare

Prin apăsarea textului de culoare albastră cu conținutul „Don't have an account? Register here!”, utilizatorul este trimis pe pagina de înregistrare a aplicației unde acesta își poate crea un cont funcțional pentru căutarea biletelor de avion.

Pe această pagină este necesar ca utilizatorul să își introducă un „Username”, sau alias unic pentru folosirea aplicației, o parolă care trebuie să fie identică cu cea scrisă în câmpul „Confirm password”, prenumele, numele de familie, adresa de email care trebuie să fie unică la fel ca la procedeul de alegere a username-ului, țara de proveniență, numărul de telefon și data de naștere.

Formularul de înregistrare se poate observa în următoarea imagine (figura 3-3):

Register

Username:

Password:

Confirm password:

First name:

Last name:

Email:

Country

Andorra

Phone number:

Date of birth:

May 25, 2020

Register Back

Fig. 3-3 Secțiunea de înregistrare a aplicației.

Tot în secțiunea de înregistrare a aplicației este de menționat că fiecare câmp este de tip obligatoriu și că username-ul și adresa de email trebuie să fie unice față de datele celorlalți utilizatori.

Pentru a verifica validitatea adresei de email a fost necesar să folosesc o metodă ce funcționează pe bază de comparații de șiruri de caractere „boolean java.lang.String.matches(String regex)” folosind librăria de bază „java.lang.String” ce verifică utilizând o expresie regulată data introdusă din acel câmp și confirmă dacă adresa de email este validă sau nu.

O expresie regulată definește șablonul de căutare a șirurilor. Prescurtarea comună a expresiei este „regex” sau „regexp”. Șablonul de căutare poate fi caractere simple, șiruri fixe sau expresii complexe care conțin caractere speciale care descriu șablonul. Pentru un șir dat, șablonul definit de expresia regulată se poate potrivi exact de una sau de mai multe ori.

Expresiile regulate pot fi utilizate pentru a găsi, edita și manipula text.

Procesul de utilizare a expresiilor regulate pentru a analiza sau modifica textul se numește aplicarea unor expresii regulate asupra textului/șirurilor. Șablonul definit de expresia regulată este aplicat textului de la stânga la dreapta. După ce caracterul sursă este utilizat în procesul de potrivire, acesta nu mai poate fi utilizat. De exemplu, expresia regulată „aba” este potrivită doar pentru „ababababa” de două ori (aba_aba_).

Java oferă pachetul „java.util.regex” pentru a se potrivi cu șabloanele cu expresii regulate. Expresiile Java obișnuite sunt foarte similare cu limbajul de programare Perl.

Expresia regulată folosită pentru a verifica adresa de email este vizibilă în figura 3-4.

```
70 private static final String EMAIL_VALIDATION_REGEX = "(?:[a-z0-9!#$%&'*/+=?^_`{|}~-]+"  
71 + "(?:\\. [a-z0-9!#$%&'*/+=?^_`{|}~-]+)*|\"(?:[\\x01-\\x"  
72 + "x08\\x0b\\x0c\\x0e-\\x1f\\x21\\x23-\\x5b\\x5d-\\x7f]|\\\""  
73 + "\\[\\x01-\\x09\\x0b\\x0c\\x0e-\\x7f])*\\")@"  
74 + "(?:(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\\\""  
75 + ".)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\\\"(?:25[0-5]|2[0-4][0-9]|"  
76 + "{3}(?:25[0-5]|2[0-4][0-9]|"  
77 + "[01]?[0-9][0-9]?|[a-z0-9-]*[a-z0-9]"  
78 + ":(?:[\\x01-\\x08\\x0b\\x0c\\x0e-\\x1f\\x21-\\x5a\\x53-\\x7f]|\\\""  
79 + "\\[\\x01-\\x09\\x0b\\x0c\\x0e-\\x7f])+\\\"")";
```

Fig. 3-4 Expresia regulată pentru verificare de adresă email.

Această expresie regulată (figura 3-4) tinde să verifice următoarele reguli ale unei adrese de email internațională corespunzând memorandumurilor de tip RFC ale IETF (Internet Engineering Task Force) din 2001 de poștă electronică [4]:

- *Toate adresele de e-mail sunt date în 7-bit US-ASCII. (RFC 2822, secțiunea 2.2)*
- *Adresele de e-mail constau dintr-o parte locală, simbolul @ și domeniul. (RFC 2822, secțiunea 3.4.1)*
- *Partea locală nu poate fi citată, completă sau citată pe o bază de etichetă. (RFC 2822, secțiunile 3.4.1 și 4.4)*
- *Părțile locale care nu sunt citate pot consta din TEXT, opțional separate prin puncte. Niciun punct nu poate începe sau încheia partea locală. Două puncte împreună sunt nevalide. (RFC 2822, secțiunea 3.4.1)*
- *TEXT-ul poate conține alfabetic, numerice și următoarele simboluri: ! # \$ % ' * + - / = ? ^ _ ` { | } ~ (RFC 2822, secțiunea 3.4.1)*
- *Partea locală citată începe cu ghilimele și se termină cu un ghilimele. (RFC 2822, secțiunea 3.4.1)*
- *Conținutul unei părți locale citate nu trebuie să conțină caractere 9 (TAB), 10 (LF), 13 (CR), 32 (spații), 34 (,), 91-94 ([, \,], ^). (RFC 2822, secțiunea 3.4.1)*
- *Dacă partea locală specificată are o inversare, următorul caracter este mascat și nu trebuie să fie 10 (LF), 13 (CR). Aceasta înlocuiește regula anterioară și permite spații și ghilimele pe adresa de e-mail atât timp cât sunt mascate. (RFC 2822, secțiunea 3.4.1)*
- *Dacă un e-mail folosește caracterul \, adresa de e-mail constă din bucăți care nu sunt citate, separate prin puncte. (RFC 2822, secțiunea 4.4)*
- *Domeniul poate fi între paranteze sau simplu. (RFC 2822, secțiunea 3.4.1)*
- *Un domeniu simplu este format din etichete separate prin puncte. Niciun punct nu poate începe sau încheia un nume de domeniu. Un nume de domeniu nu poate conține două puncte consecutive. (RFC 1035, secțiunea 2.3.4)*
- *Lungimea maximă a unei etichete este de 63 de caractere. (RFC 1035, secțiunea 2.3.4)*

- *etichetă poate conține cratime, dar nu două cratime la rând. O etichetă nu poate începe sau termina cu o cratime. (RFC 1035, secțiunea 2.3.4)*
- *Domeniile între paranteze trebuie să înceapă cu [, să termine cu] și să nu conțină caractere 9 (TAB), 10 (LF), 13 (CR), 32 (spații), 91-94 ([, \,], ^). (RFC 2822, secțiunea 3.4.1)*
- *Conținutul unui domeniu din paranteze poate avea un \ în fața unui caracter pentru a scăpa de el, iar următorul caracter nu poate fi 10 (LF) sau 13 (CR). Aceasta permite spații din domeniu atât timp cât sunt mascate. (RFC2822, secțiunea 3.4.1)*
- *Lungimea maximă a părții locale este de 64 de caractere. (RFC 2821, secțiunea 4.5.3.1)*
- *Domeniile trebuie rezolvate cu o înregistrare A sau MX. (RFC 2821, secțiunea 3.6)*
- *Lungimea maximă a unui domeniu trebuie să fie de 255 de caractere dacă este transmisă prin cablu (pentru căutări DNS). Prin urmare, lungimea maximă a domeniului (domeniul normal sau conținutul domeniului între paranteze) este de 253 de caractere. (RFC 1034, secțiunea 3.1)*
- *Fiecare etichetă de domeniu poate avea maximum 63 de caractere. (RFC 1034, secțiunea 2.3.1)*
- *Lungimea maximă a unei adrese de e-mail „utile” este de 255 de caractere. (RFC 2821, secțiunea 4.5.3.1)*
- *Lungimea maximă a unei adrese de e-mail este de 320 de caractere. (RFC 3696)*
- *Domeniul de nivel superior trebuie să fie complet alfabetic. (RFC 3696 Secțiunea 2)*

3.3 Sistemul de resetare a parolelor

În cazul în care utilizatorul aplicației și-a uitat parola sau și-a pierdut-o, acesta are la dispoziție un sistem de resetare a parolelor care se folosește de API-ul JavaMail pentru a trimite la adresa de email a utilizatorului un cod de șase caractere cu care acesta își poate reseta parola cu ușurință.

Procesul de resetare este asemănător cu procesul de înregistrare în sensul că la introducerea cu succes a codului de verificare, utilizatorul este pus să își introducă o nouă parolă pentru a accesa aplicația.

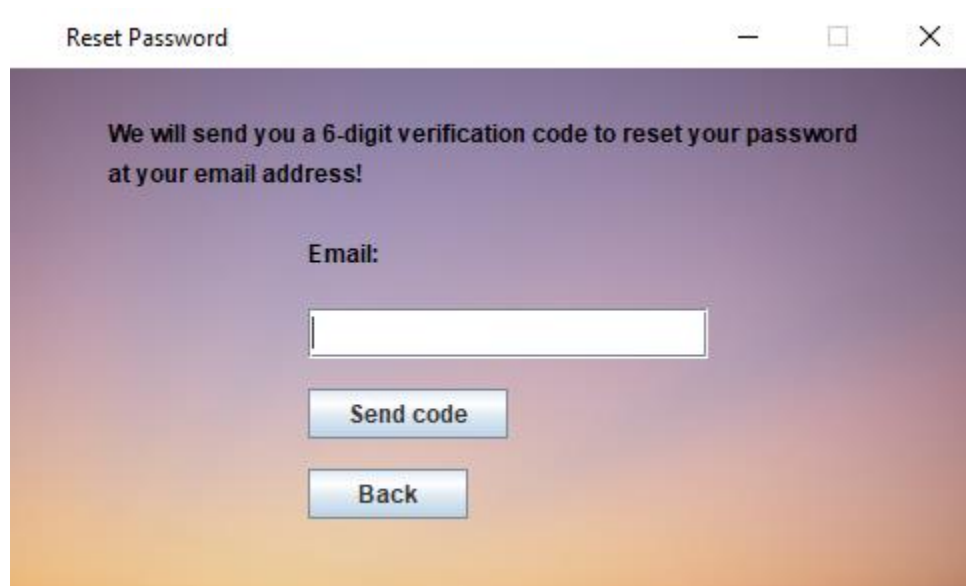


Fig. 3-5 Primul pas pentru a reseta parola.

În câmpul „Email” din figura 3-5 utilizatorul este solicitat să își scrie adresa de email pentru a primi codul de verificare prin mesaj.

După apăsarea butonului „Send code”, utilizatorul va trebui să aștepte câteva secunde pentru a primi codul pe email, și va continua la pasul doi de resetare a parolei.

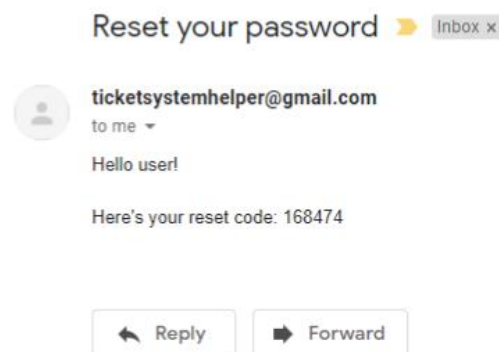


Fig. 3-6 Tipul de mesaj trimis pe adresa de email a utilizatorului.

În pasul următor se introduce codul primit de pe email în următoarea fereastră:

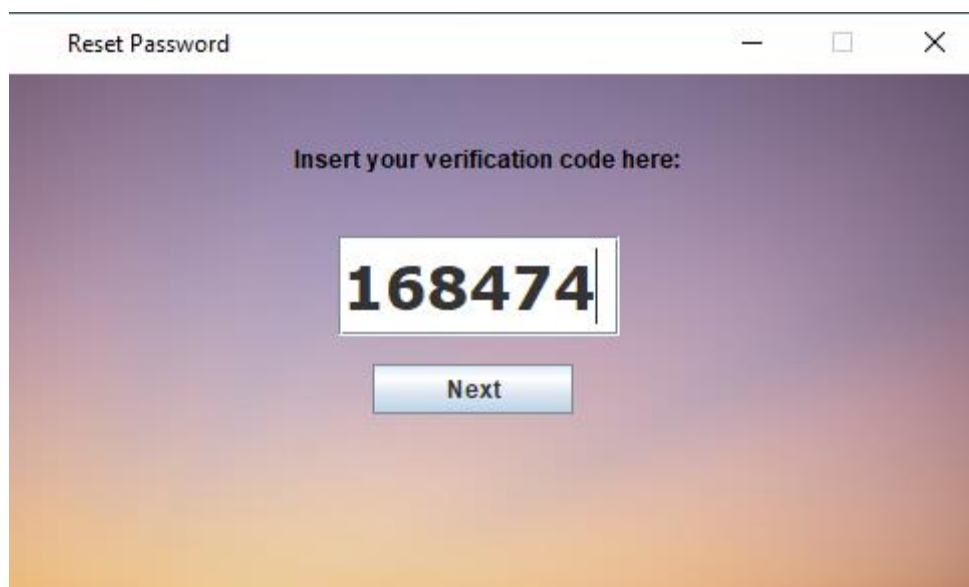
A screenshot of a web application window titled "Reset Password". The window has a purple-to-orange gradient background. At the top, there are standard window controls (minimize, maximize, close). Below the title bar, the text "Insert your verification code here:" is displayed. In the center, a white rectangular box contains the verification code "168474" in a large, bold, black font. Below this box is a blue button with the text "Next" in white.

Fig. 3-7 Pasul doi de resetare a parolei.

Pentru a finaliza procesul, utilizatorul are de ales o parolă nouă și de a o confirma în fereastra care urmează în figura 3-8.

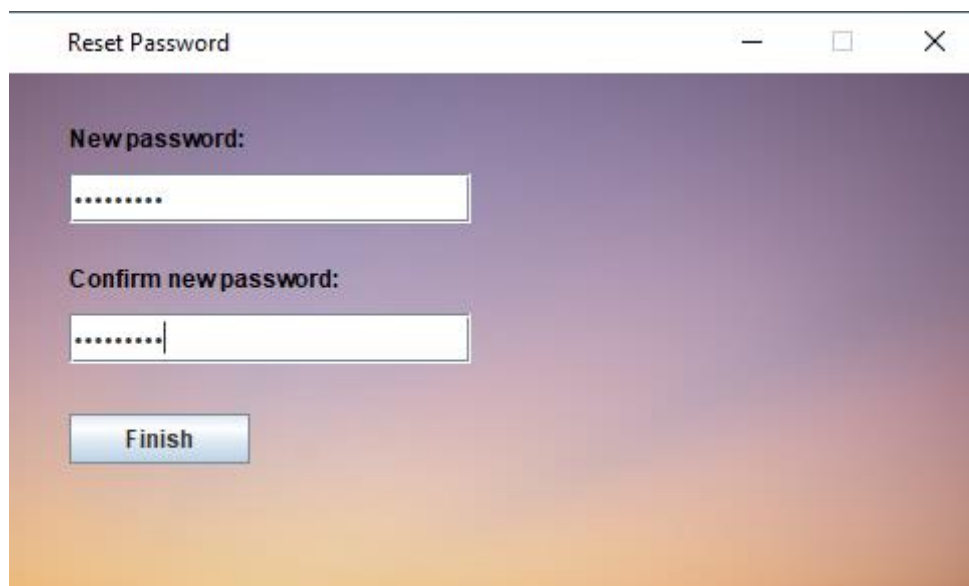
A screenshot of a web application window titled "Reset Password". The window has a purple-to-orange gradient background. At the top, there are standard window controls (minimize, maximize, close). Below the title bar, the text "Newpassword:" is displayed. Below this text is a white rectangular input field containing seven dots. Below the input field, the text "Confirm new password:" is displayed. Below this text is another white rectangular input field containing seven dots. At the bottom of the window is a blue button with the text "Finish" in white.

Fig. 3-8 Ultimul pas de resetare a parolei.

3.4 Sistemul de căutare a biletelor de avion

După logare utilizatorul poate căuta în baza de date a biletelor o călătorie anume după diferite date de intrare cum ar fi țara de unde va pleca avionul, destinația unde va ateriza, data de plecare, data de sosire, opțiunea de dus-întors sau zbor direct, clasa și prețul (se poate seta cu valoarea 0 pentru a afișa toate biletele disponibile pentru datele de intrare).

Biletele vor fi afișate în ordinea crescătoare a prețului. Toate acestea se pot observa în figura următoare:

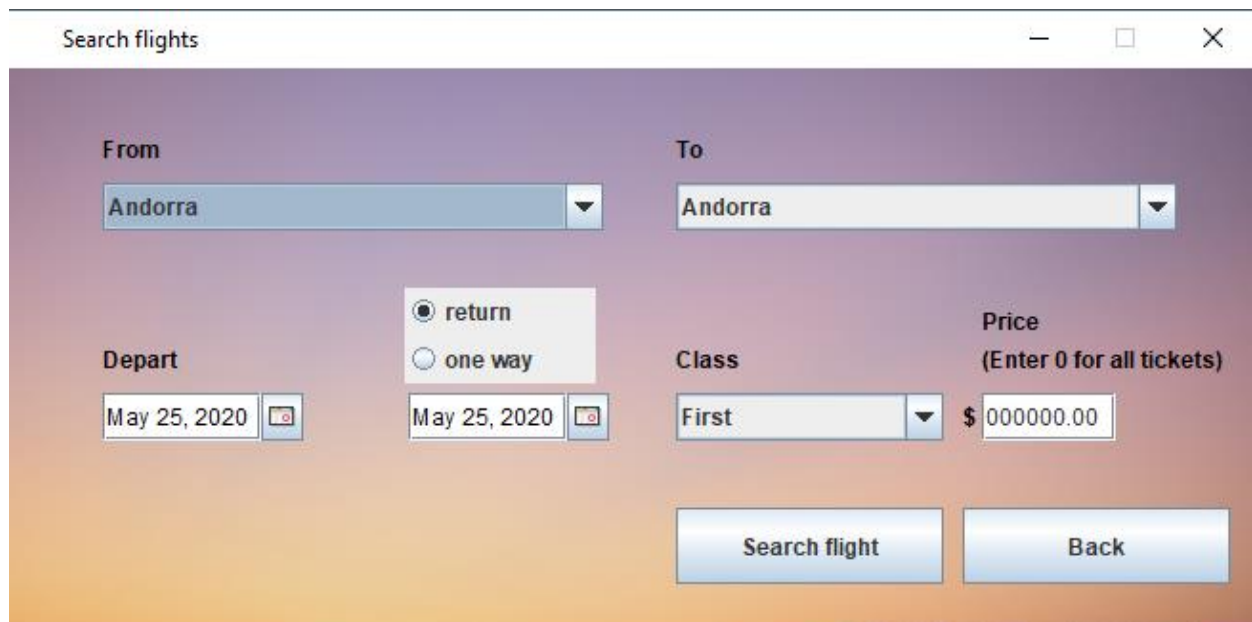


Fig. 3-9 Sistemul de căutare a biletelor de avion.

Cu ajutorul librăriei „java.util.Locale” am reușit să creez o listă cu toate țările actual existente pentru a le afișa în componenta Swing de tip JComboBox pentru o căutare mai ușoară. În secțiunile de „From” și „To” se pot observa aceste elemente (figura 3-10).

Pentru selectarea unei date specifice pentru căutare am folosit un obiect de tip JDateChooser, ce se poate observa în figura 3-11. Chiar dacă în interfață data aleasă este într-un format diferit față de cel al datelor inserate în baza de date, cu ajutorul unor formătări prin cod am rezolvat această problemă cu ușurință și am omogenizat formatul datelor pe întregul aplicației.

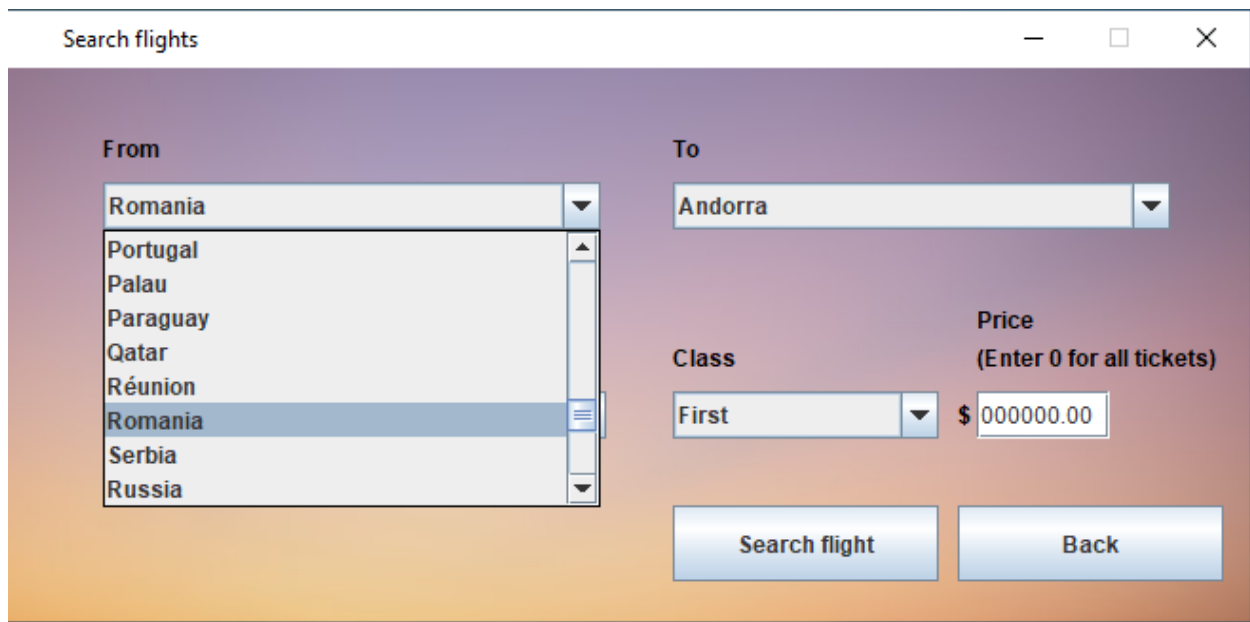


Fig. 3-10 Exemplu de JComboBox.

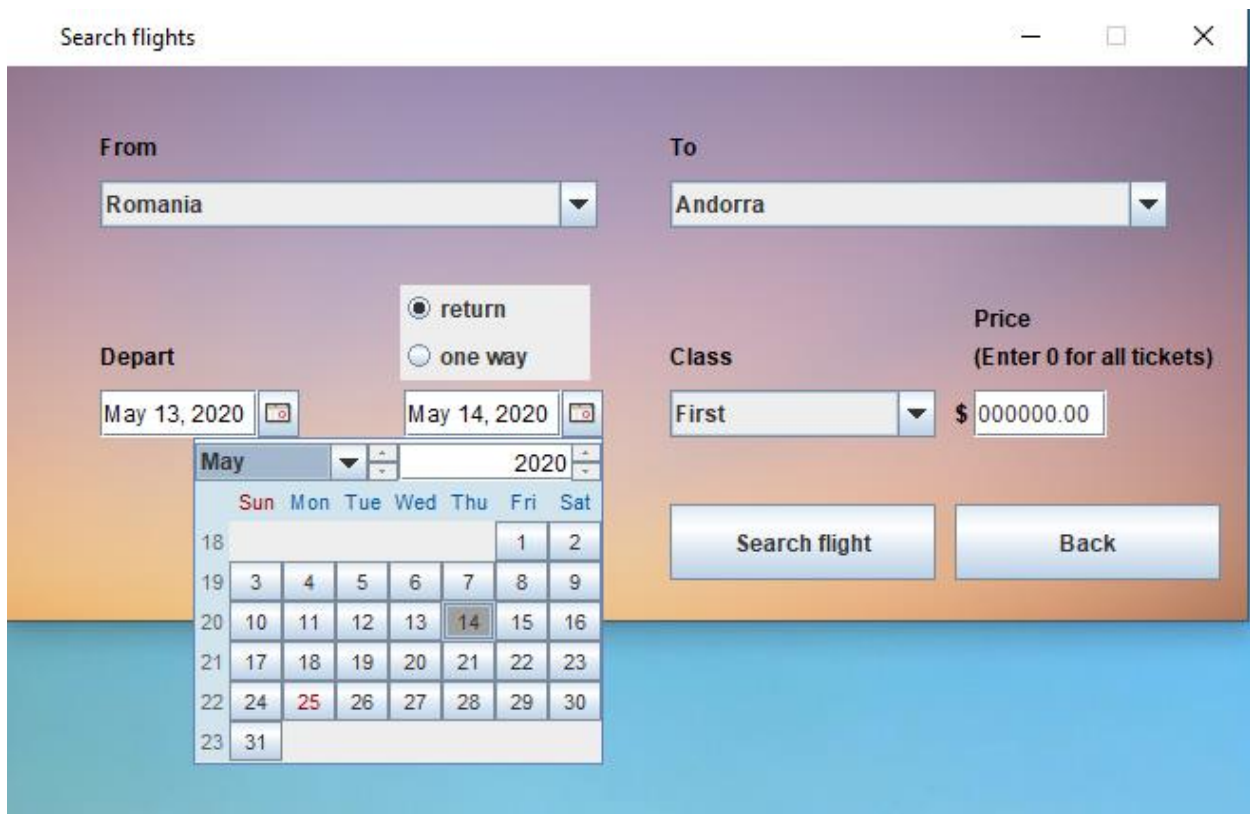


Fig. 3-11 Exemplu de JDateChooser.

Search flights

From: Romania

To: Andorra

Depart: May 13, 2020

Return: May 14, 2020

Class: First (dropdown menu open showing: First, Business, Economy, Premium Economy)

Price: \$ 000000.00

Buttons: return (selected), one way, Back

Fig. 3-12 Tipuri de clase de călătorii.

Se poate menționa că utilizatorul poate căuta un bilet de avion după standardul de clase de aviație comercială după cum se vede în figura 3-12.

Search results

There are no available flights based on the data you have entered!

Back

Fig. 3-13 Eroare la căutare.

Dacă utilizatorul nu poate găsi un bilet de avion după datele de intrare alese, fereastra din figura 3-13 va apărea pentru a-l informa că nu există bilete de avion în baza de date cu acele criterii.

În urma unei căutări de bilete după datele introduse în aplicație ca în figura 3-12 următoarea fereastră va apărea:

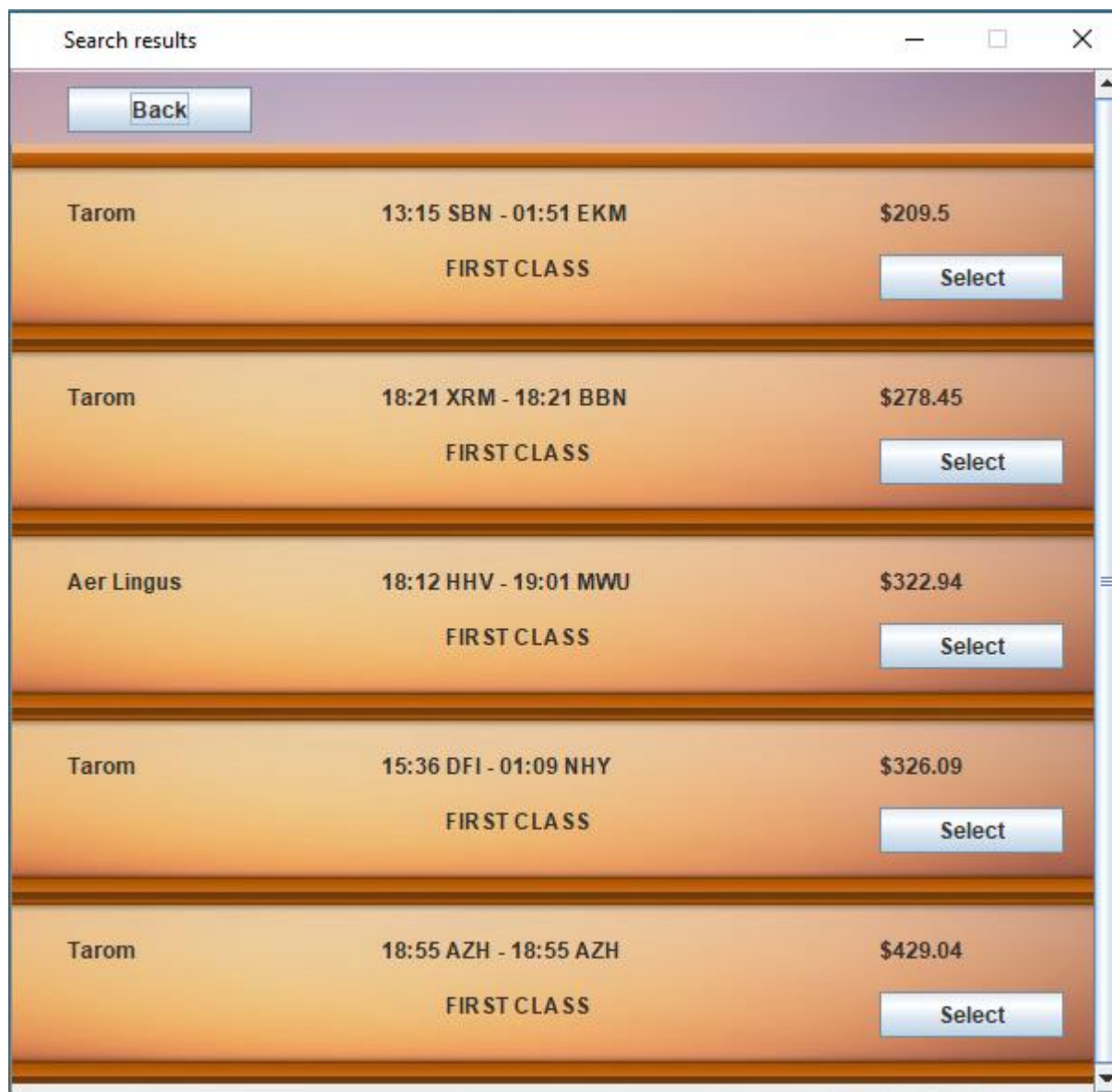


Fig. 3-14 Rezultatul unei căutări.

După căutare se vor afișa toate biletele ce îndeplinesc criteriile date de utilizator în interfața de căutare de bilete. În această fereastră se pot observa numele companiilor aeriene ce emit biletele

corespunzătoare, timpii și aeroporturile de plecare și de sosire, clasa și prețul biletelor, și un buton de selectare a biletului.

În cazul în care în caseta de preț a interfeței de căutare se introduce numărul 300 se vor afișa biletele cu prețul până în 300 de dolari.

O astfel de căutare se poate observa în următoarea figură:



Fig. 3-15 Căutare specifică până în 300 de dolari.

În momentul în care utilizatorul s-a hotărât la biletul pe care dorește să-l achiziționeze acesta este nevoit să apese pe butonul de select. În urma apăsării butonului acesta este prezentat cu biletul final (figura 3-16) ce conține toate datele specifice biletului precum numele companiei aeriene, numele pasagerului, numărul de zbor, data eliberării biletului, aeroportul de plecare, țara și aeroportul de sosire, data de îmbarcare și plecare, țara și aeroportul de plecare și codul QR al biletului care la scanare va afișa hashcode-ul obiectului de tip bilet.

Din acest moment se poate preciza că biletul este în posesia clientului și a fost efectuată o tranzacție.



Fig. 3-16 Biletul de avion selectat.

3.5 Panoul de administrator

Până în acest punct am prezentat partea de utilizator a aplicației, dar din motive de securitate aplicația are nevoie și de o secțiune unde administratorii acesteia pot gestiona biletele și utilizatorii, pot insera bilete și pot observa rapoarte generate în urma tranzacțiilor dintre utilizatori și aplicație.

Panoul de administrator nu se poate observa doar dacă în sistem este efectuată o logare cu un cont special de administrator. Acest cont nu se poate crea din secțiunea de înregistrare ci doar din interogări a bazei de date, acest lucru fiind introdus din motive de securitate. Tot prin interogarea bazei de date un administrator poate face ca orice cont din sistem să fie trecut pe partea de administrator, modificând valoare câmpului „administrator” din 0 în valoarea 1 din tabela „users”.

În acest panou observabil din figura 3-17 un administrator poate să introducă o călătorie nouă în baza de date a biletelor după următoarele criterii: codul companiei aeriene, numărul călătoriei, numele companiei aeriene, locația de unde va pleca avionul, aeroportul de unde va pleca avionul, locația unde va ajunge avionul, aeroportul unde va ajunge avionul, data de îmbarcare, data de plecare, data de sosire, clasa, poarta din aeroport, prețul biletului, opțiunea de reîntoarcere și opțiunea de sens unic. Se poate observa mesajul de atenționare de culoare roșie care apare în cazul

în care sunt introduse date incorecte pentru inserare de bilet în partea din dreapta jos unde se menționează caracterul datelor necesar pentru a introduce o călătorie validă în baza de date.

The screenshot shows a web application window titled "Admin Panel". It contains a form for adding a new flight. The form is divided into two columns. The left column contains fields for "Airline code:", "Flight number:", "Airline company:", "From:" (a dropdown menu showing "Andorra"), "From airport:", "To:" (a dropdown menu showing "Andorra"), and "To airport:". The right column contains fields for "Boarding date:" (a date and time picker showing "May 25, 2020" and "18:47:32"), "Departure date:" (a date and time picker showing "May 25, 2020" and "18:47:32"), "Arrival date:" (a date and time picker showing "May 25, 2020" and "18:47:32"), "Class:" (a dropdown menu showing "First"), "Gate:", "Price:" (a text input showing "\$ 000000.00"), and radio buttons for "return" (selected) and "one way". At the bottom of the form are three buttons: "Show reports", "Insert new flight", and "Back". Below the "Insert new flight" button, there is a red error message that reads: "Incorrect details! Airline code must be numeric! Flight number must be numeric! Airports must be only in alphabetic letters! Price must not be 0! Boarding date must be before departure date and departure date before arrival date!".

Admin Panel

Airline code:

Flight number:

Airline company:

From:

From airport:

To:

To airport:

Boarding date:

Departure date:

Arrival date:

Class:

Gate:

Price:

return one way

Show reports

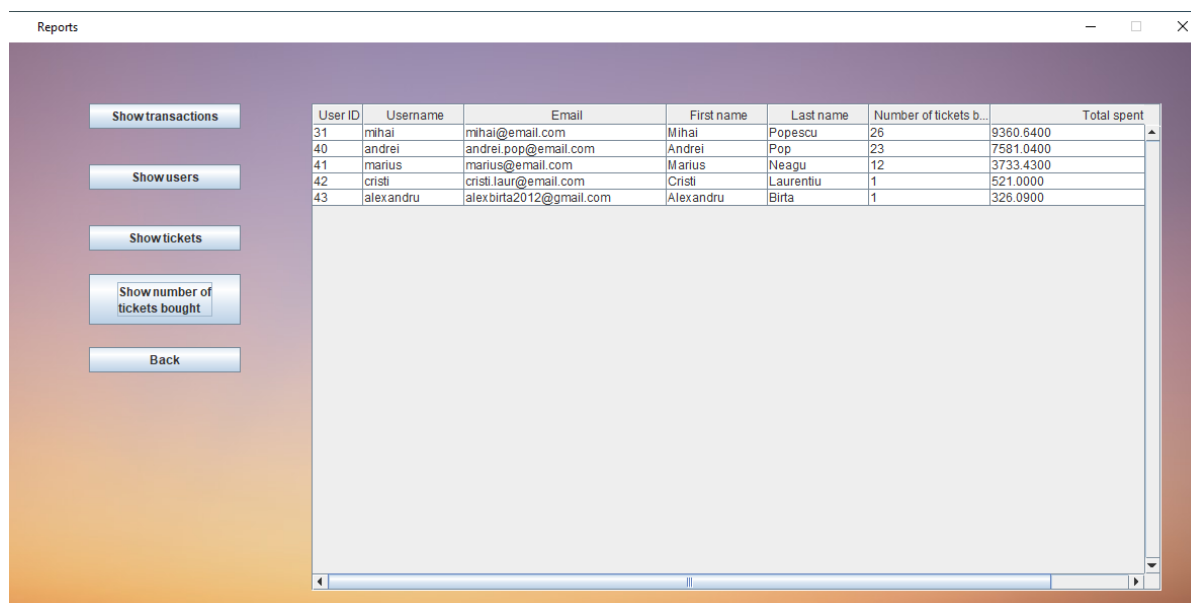
Insert new flight

Back

Incorrect details!
Airline code must be numeric!
Flight number must be numeric!
Airports must be only in alphabetic letters!
Price must not be 0!
Boarding date must be before departure date and departure date before arrival date!

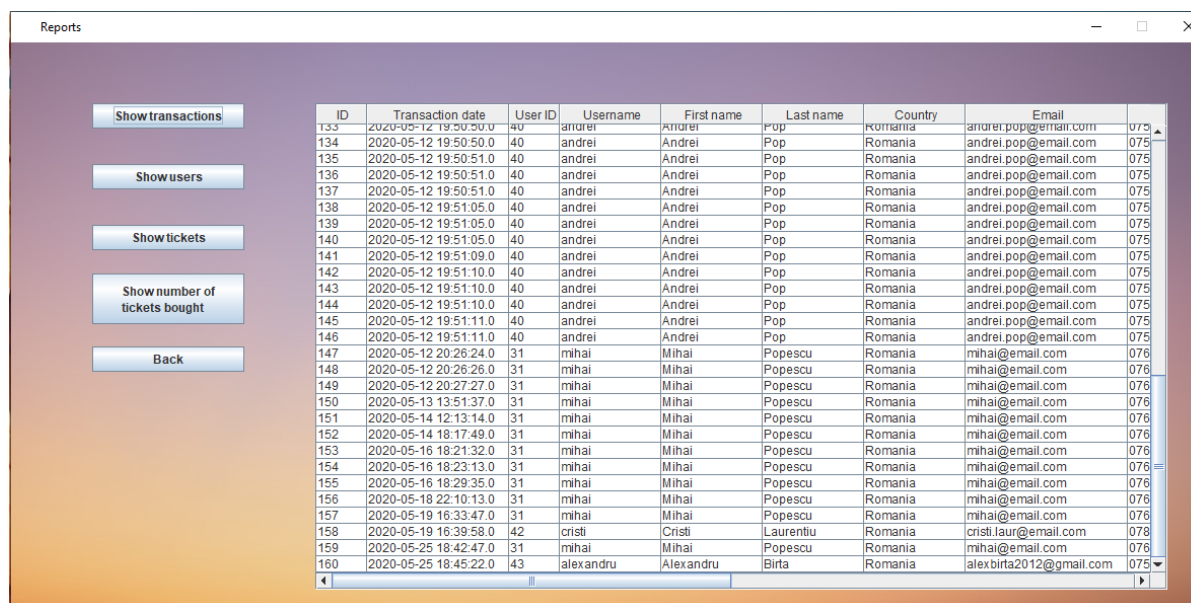
Fig. 3-17 Panoul de administrator.

În cazul în care administratorul apasă pe butonul „Show reports”, acesta este prezentat cu panoul în care se pot vedea diferite tipuri de rapoarte cum ar fi tranzacțiile (figura 3-19), numărul de bilete cumpărate de un singur utilizator împreună cu totalul plătit (figura 3-18), toți utilizatorii curenți ai aplicației și toate biletele din sistem în format tabelar.



User ID	Username	Email	First name	Last name	Number of tickets b...	Total spent
31	mihai	mihai@email.com	Mihai	Popescu	26	9360.6400
40	andrei	andrei.pop@email.com	Andrei	Pop	23	7581.0400
41	marius	marius@email.com	Marius	Neagu	12	3733.4300
42	cristi	cristi.laur@email.com	Cristi	Laurentiu	1	521.0000
43	alexandru	alex.birta2012@gmail.com	Alexandru	Birta	1	326.0900

Fig. 3-18 Secțiunea de rapoarte, partea „Show number of tickets bought”.



ID	Transaction date	User ID	Username	First name	Last name	Country	Email
133	2020-05-12 19:50:50.0	40	andrei	Andrei	Pop	Romania	andrei.pop@email.com
134	2020-05-12 19:50:50.0	40	andrei	Andrei	Pop	Romania	andrei.pop@email.com
135	2020-05-12 19:50:51.0	40	andrei	Andrei	Pop	Romania	andrei.pop@email.com
136	2020-05-12 19:50:51.0	40	andrei	Andrei	Pop	Romania	andrei.pop@email.com
137	2020-05-12 19:50:51.0	40	andrei	Andrei	Pop	Romania	andrei.pop@email.com
138	2020-05-12 19:51:05.0	40	andrei	Andrei	Pop	Romania	andrei.pop@email.com
139	2020-05-12 19:51:05.0	40	andrei	Andrei	Pop	Romania	andrei.pop@email.com
140	2020-05-12 19:51:05.0	40	andrei	Andrei	Pop	Romania	andrei.pop@email.com
141	2020-05-12 19:51:09.0	40	andrei	Andrei	Pop	Romania	andrei.pop@email.com
142	2020-05-12 19:51:10.0	40	andrei	Andrei	Pop	Romania	andrei.pop@email.com
143	2020-05-12 19:51:10.0	40	andrei	Andrei	Pop	Romania	andrei.pop@email.com
144	2020-05-12 19:51:10.0	40	andrei	Andrei	Pop	Romania	andrei.pop@email.com
145	2020-05-12 19:51:11.0	40	andrei	Andrei	Pop	Romania	andrei.pop@email.com
146	2020-05-12 19:51:11.0	40	andrei	Andrei	Pop	Romania	andrei.pop@email.com
147	2020-05-12 20:26:24.0	31	mihai	Mihai	Popescu	Romania	mihai@email.com
148	2020-05-12 20:26:26.0	31	mihai	Mihai	Popescu	Romania	mihai@email.com
149	2020-05-12 20:27:27.0	31	mihai	Mihai	Popescu	Romania	mihai@email.com
150	2020-05-13 13:51:37.0	31	mihai	Mihai	Popescu	Romania	mihai@email.com
151	2020-05-14 12:13:14.0	31	mihai	Mihai	Popescu	Romania	mihai@email.com
152	2020-05-14 18:17:49.0	31	mihai	Mihai	Popescu	Romania	mihai@email.com
153	2020-05-16 18:21:32.0	31	mihai	Mihai	Popescu	Romania	mihai@email.com
154	2020-05-16 18:23:13.0	31	mihai	Mihai	Popescu	Romania	mihai@email.com
155	2020-05-16 18:29:35.0	31	mihai	Mihai	Popescu	Romania	mihai@email.com
156	2020-05-18 22:10:13.0	31	mihai	Mihai	Popescu	Romania	mihai@email.com
157	2020-05-19 16:33:47.0	31	mihai	Mihai	Popescu	Romania	mihai@email.com
158	2020-05-19 16:39:58.0	42	cristi	Cristi	Laurentiu	Romania	cristi.laur@email.com
159	2020-05-25 18:42:47.0	31	mihai	Mihai	Popescu	Romania	mihai@email.com
160	2020-05-25 18:45:22.0	43	alexandru	Alexandru	Birta	Romania	alex.birta2012@gmail.com

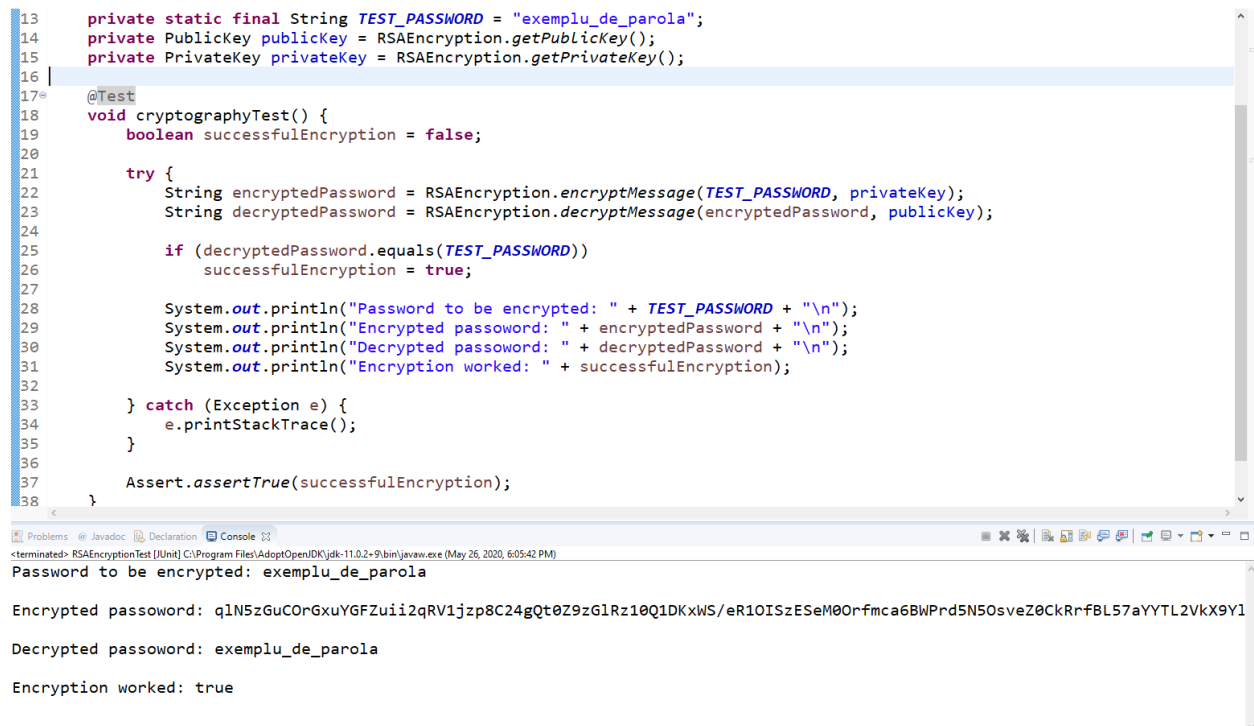
Fig. 3-19 Secțiunea de rapoarte, partea „Show transactions”.

Capitolul 4

Testarea aplicației

Pentru a testa punctele slabe ale aplicației și a diferitelor metode create în pachetele de clase a proiectului am ales să folosesc un framework de testare funcțională în Java numit JUnit. Cu ajutorul acestui framework am reușit să creez o serie de teste funcționale ce au rolul de a oferi o claritate asupra datelor de ieșire a unor metode și de a stabili robustețea aplicației pentru o utilizare mai securizată și lipsită de defecte cu acordul cerinței aplicației.

4.1 Testul modulului de criptare



```
13 private static final String TEST_PASSWORD = "exemplu_de_parola";
14 private PublicKey publicKey = RSAEncryption.getPublicKey();
15 private PrivateKey privateKey = RSAEncryption.getPrivateKey();
16
17 @Test
18 void cryptographyTest() {
19     boolean successfulEncryption = false;
20
21     try {
22         String encryptedPassword = RSAEncryption.encryptMessage(TEST_PASSWORD, privateKey);
23         String decryptedPassword = RSAEncryption.decryptMessage(encryptedPassword, publicKey);
24
25         if (decryptedPassword.equals(TEST_PASSWORD))
26             successfulEncryption = true;
27
28         System.out.println("Password to be encrypted: " + TEST_PASSWORD + "\n");
29         System.out.println("Encrypted password: " + encryptedPassword + "\n");
30         System.out.println("Decrypted password: " + decryptedPassword + "\n");
31         System.out.println("Encryption worked: " + successfulEncryption);
32     } catch (Exception e) {
33         e.printStackTrace();
34     }
35
36     Assert.assertTrue(successfulEncryption);
37 }
38 }
```

Problems | Javadoc | Declaration | Console

<terminated> RSAEncryptionTest [JUnit] C:\Program Files\AdoptOpenJDK\jdk-11.0.2-9\bin\javaw.exe (May 26, 2020, 6:05:42 PM)

Password to be encrypted: exemplu_de_parola

Encrypted password: q1N5zGuC0rGxuYGFZui12qRV1jzp8C24gQt0Z9zG1Rz10Q1DKxWS/eR10ISzESEm0Orfmca6BWPrd5N50sveZ0CkRrFBL57aYYTL2VkX9Y1

Decrypted password: exemplu_de_parola

Encryption worked: true

Fig. 4-1 Testul modulului de criptare.

Un prim test al aplicației a fost acela de a verifica funcționarea corectă a modulului de criptare. Din figura 4-1 se poate observa că această testare a fost trecută cu succes. Obiectele de

tip `PublicKey` și `PrivateKey` au fost preluate prin deserializare din fișierele cu obiectele respective original serializate și au fost utilizate la criptarea și decriptarea parolei de test „exemplu_de_parola” cu ajutorul metodelor de criptare și decriptare folosite de algoritmul RSA.

În urma acestui test am conchis că modulul de criptare funcționează corect și poate fi implementat în sistemul de înregistrare a utilizatorilor pentru a cripta parolele acestora.

4.2 Testul formularului de înregistrare

Pentru a testa formularul de înregistrare am creat trei teste diferite ce pot fi rulate simultan. Aceste teste verifică dacă utilizatorul are adresa de email validă, dacă are username-ul unic în baza de date și dacă are adresa de email unică în baza de date a utilizatorilor.

```
57 @Test
58 public void validEmailTest() {
59
60     boolean emailCheck = RegisterForm.isValidEmail(testUser.getEmail());
61
62     Assert.assertTrue(emailCheck);
63 }
64
```

Fig. 4-2 Primul test al formularului de înregistrare.

```
65 @Test
66 public void uniqueUsernameTest() {
67
68     ResultSet rs = null;
69     String username = testUser.getUsername();
70     int counter = 0;
71
72     try (Connection conn = DBConnection.getConnection();
73         PreparedStatement stmt = conn.prepareStatement(SELECT_USERS_STATEMENT,
74             ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);) {
75
76         rs = stmt.executeQuery();
77
78         while (rs.next()) {
79             if (username.equals(rs.getString("username")))
80                 counter++;
81         }
82     } catch (SQLException e) {
83         e.printStackTrace();
84     }
85
86     Assert.assertEquals(counter, 1);
87 }
88
```

Fig. 4-3 Al doilea test al formularului de înregistrare.

```

90  @Test
91  public void uniqueEmailTest() {
92
93      ResultSet rs = null;
94      String email = testUser.getEmail();
95      int counter = 0;
96
97      try (Connection conn = DBConnection.getConnection();
98           PreparedStatement stmt = conn.prepareStatement(SELECT_USERS_STATEMENT,
99                                                         ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);) {
100
101          rs = stmt.executeQuery();
102
103          while (rs.next()) {
104              if (email.equals(rs.getString("email")))
105                  counter++;
106          }
107      } catch (SQLException e) {
108          e.printStackTrace();
109      }
110
111      Assert.assertEquals(counter, 1);
112  }
113

```

Fig. 4-4 Al treilea test al formularului de înregistrare.

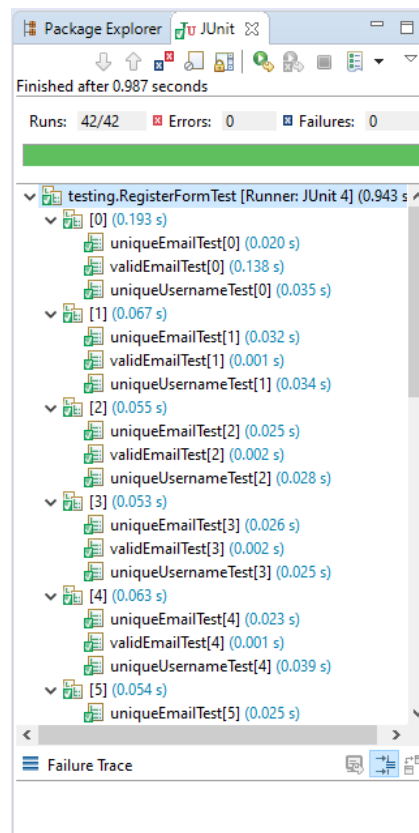


Fig. 4-5 Rularea testelor simultan.

Pentru a oferi o testare completă am ales să aplic toate cele trei teste simultan. Din figura 4-5 se poate observa că și acest test a fost completat cu succes. Datele de intrare pentru verificare au fost alese ca fiind chiar înregistrările din baza de date actuală cu utilizatori, concluzionând că prin acest test putem verifica și integritatea bazei de date și funcționalitatea formularului de înregistrare în același timp, și se poate rula oricând pentru a efectua o verificare generală.

4.3 Testul panoului de administrator

Testele panoului de administrator constau din testarea datelor introduse în baza de date în legătură cu inserarea biletelor. Se verifică dacă codul companiei aeriene este strict un șir de caractere numeric, dacă numărul zborului este strict numeric, dacă aeroporturile sunt scrise sub formă alfanumerică, dacă prețul biletelor este diferit de 0 și dacă data de îmbarcare este înainte de data de plecare și data de plecare este înainte de data de sosire. Toate aceste teste asigură integritatea datelor introduse și verifică dacă există anomalii printre datele deja introduse în baza de date.

```
64@ @Test
65 public void validTicketCompanyCodeTest() {
66     Assert.assertTrue(StringUtils.isStrictlyNumeric(testTicket.getAirline_company_code()));
67 }
68
69
70@ @Test
71 public void validTicketFlightNumberTest() {
72     Assert.assertTrue(StringUtils.isStrictlyNumeric(testTicket.getFlight_number()));
73 }
74
75@ @Test
76 public void validTicketAirportsTest() {
77     boolean airportCheck = false;
78
79     if (AdminPanel.isAlpha(testTicket.getFrom_airport()) && AdminPanel.isAlpha(testTicket.getTo_airport()))
80         airportCheck = true;
81
82     Assert.assertTrue(airportCheck);
83 }
84
85@ @Test
86 public void validTicketPriceTest() {
87     Assert.assertTrue(testTicket.getPrice() != 0.0);
88 }
89
```

Fig. 4-6 Primele patru teste ale panoului de administrator

```

90= @Test
91 public void validTicketDatesTest() {
92     boolean datesCheck = false;
93     Date boardingCheck;
94     Date departureCheck;
95     Date arrivalCheck;
96
97     try {
98         boardingCheck = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").parse(testTicket.getBoarding());
99         departureCheck = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").parse(testTicket.getDeparture());
100        arrivalCheck = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").parse(testTicket.getArrival());
101
102        if (boardingCheck.before(departureCheck) && departureCheck.before(arrivalCheck))
103            datesCheck = true;
104    } catch (ParseException e) {
105        e.printStackTrace();
106    }
107
108    Assert.assertTrue(datesCheck);
109 }

```

Fig. 4-7 Ultimul test al panoului de administrator.

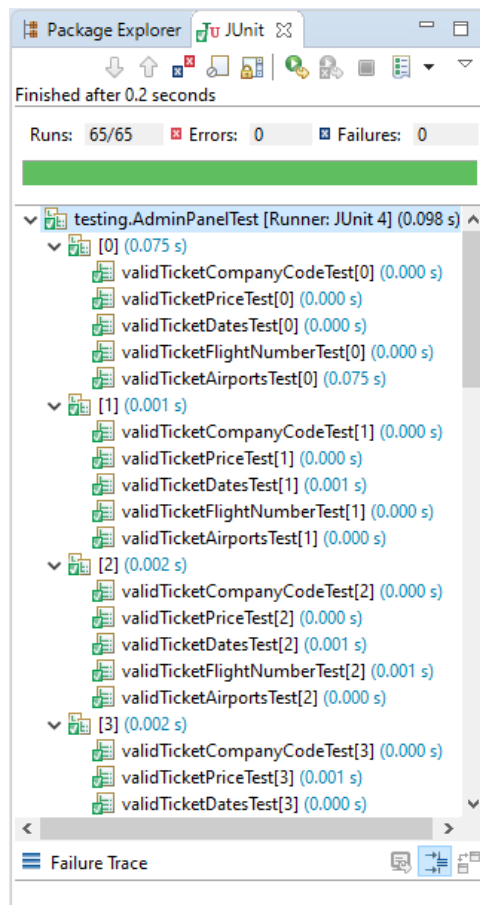


Fig. 4-8 Rularea celor cinci teste simultan.

Prin rularea tuturor testelor simultan se concluzionează că datele introduse în baza de date sunt valide și obiectele de tip bilet generate la inserarea din panoul de administrator pot urma diferite reguli specifice pentru a standardiza formatul de bilet introdus pentru căutare.

Concluzii

Am dezvoltat o aplicație de gestiune a biletelor de avion cu ajutorul limbajului de programare Java și a unor API-uri specifice acestui limbaj. Prin intermediul diferitelor librării folosite am reușit să creez un sistem de logare pentru utilizatorii aplicației, un sistem de înregistrare pentru conturile acestora, un mecanism de resetare a parolelor folosind protocolul SMTP și librăria JavaMail, motorul de căutare a biletelor de avion și modulul dedicat administratorilor aplicației pentru gestionarea biletelor. Modulul de administrator constă în inserarea biletelor în baza de date, abilitatea de a vizualiza rapoarte ce constau în tranzacții între utilizatori și aplicație, numărul de bilete cumpărate și totalul cheltuit și vizualizarea în format tabelar a utilizatorilor și a biletelor.

Printre unele direcții viitoare de dezvoltare pot să specific integrarea acestei aplicații într-un mediu on-line cu ajutorul servlet-urilor Java, pentru că în acest stadiu numărul utilizatorilor este minimal din cauza lipsei accesului la internet.

O altă direcție viitoare de dezvoltare mai poate fi și reconstruirea interfeței grafice folosind tehnologiile JavaFX, fiind o tehnologie mai avansată în sensul de design al interfețelor grafice pentru utilizatori și eventual refacerea design-ului de la început cu scopul de a aduce mai mulți utilizatori către aplicație cu ajutorul unui aspect mai atractiv.

Altă opțiune viitoare de dezvoltare ar putea fi și integrarea unui sistem de alegere a scaunului de avion pentru călătoria utilizatorului, deoarece această alegere este considerată foarte importantă în sistemele actuale de gestiune de bilete de avion de pe piață sau chiar luând ca exemplu și sistemele de bilete de cinema sau de teatru.

Bibliografie

- [1] **J. Bloch** – Effective Java, Addison Wesley, 2013
- [2] **P. Deitel, H. Deitel** - Java How to Program ,10th Ed. 2014
- [3] **B. Eckel** - Thinking in Java, 2012
- [4] **Peter W. Resnick** – Internet Message Format, 2001
- [5] **Alfred Menezes** - Handbook of Applied Cryptography. Paul van Oorschot, Scott Vanstone.
- [6] **Matthew Robinson, Pavel Vorobiev** - Swing, Second Edition, Manning, ISBN 1-930110-88-X
- [7] **Bruce Schneier** – Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C, Ed. Wiley, 1996, ISBN 0471128457
- [8] **Oracle Corporation** - MySQL 8.0 Reference Manual, April 2020
- [9] **Chatham, Mark** - Structured Query Language By Example - Volume I: Data Query Language. p. 8., 2012, ISBN 978-1-29119951-2.
- [10] **Reenskaug, Trygve; Coplien, James O.** - "The DCI Architecture: A New Vision of Object-Oriented Programming", 20 March 2009
- [11] **Addison-Wesley** - Unified Modeling Language User Guide, The (2 ed.). 2005. p. 496. ISBN 0321267974.
- [12] **ISO/IEC 19501:2005** – “Information technology - Open Distributed Processing - Unified Modeling Language (UML) Version 1.4.2". Iso.org. 1 April 2005.
- [13] **Object Management Group** - "OMG Unified Modeling Language (OMG UML), Superstructure. Version 2.4.1".
- [14] **C. Răuciu** – Criptografie și securitatea informației, Editura Renaissance, București, 2010, ISBN 978-606-8321-89-9