**Site pentru agentie de turism**

**Bilete de avion**

Universitatea Tehnica din Cluj-Napoca

Studenti:

Mihnea Teodor Circiu

Mihai Cociubei

Raul Gorgan

Grupa: 30237

Data: 3 ianuarie 2022

Cuprins:

1. Introducere si descriere proiect
2. Detalii implementare
3. Diagrame
4. **Introducere si descriere proiect**

Echipa noastra este formata din: Mihnea Teodor Circiu, Mihai Cociubei si Raul Gorgan. Obiectivul proiectului nostru a fost realizarea unei agentii de turism online ce ofera posibilitatea clientilor sa se inregistreze, autentifice, sa caute trasee aeriene personale oferite de companiile aeriere cunoscute ca Blue Air, Ryanair, United Airlines, Wizz Air, etc, in tari ca Spania, Franta, Romania, Dubai, Olanda, Anglia, Statele Unite ale Americii si Germania (pana acum).

Clientul poate accesa site-ul unde se va inregistra si isi va pastra datele in baza noastra de date. Poate completa spatiile de text iar algoritmul nostru de cautare a zborurilor va gasi cel mai eficient traseu intre cele doua alegeri ale clientului, asta doar daca intre cele doua locatii exista posibilitatea de a fi legate prin escale. Ulterior, clientul poate accesa biletul in format pdf.

Adminul poate modifica permisiunile unui alt user (implica admin sau client) poate modifica zboruri sau aeropoarte.

1. **Detalii implementare**

Aplicatia noastra este realizata folosind limbajul Java, alaturi de Spring si MySQL, paginile web fiind construite folosind HTML si CSS.

1. Baza de date, cautare (Mihnea Teodor Circiu). Diagrama de secventa si Composite Design Pattern.

Baza de date a proiectului nostru este construita in MySQL. Aceasta contine 3 tabele: Zbor, Aeroport si Persoana.

Tabelul Persoana sugereaza persoanele care se inregistreaza si ulterior se pot autentifica cu datele sale, de asemenea contine si administratorii site-ului.

Tabelul Zbor prezinta zborurile pe care site-ul le pune la dispozitia clientului.

Tabelul Aeroport prezinta aeroporturile pe care site-ul le pune la dispozitia clientului.

Aplicatia noastra se conecteaza la baza de date utilizand clasa „ConnectionSql” din pachetul „connection”.

Clasele model ale aplicatiei prezinta cele 3 tabele ale bazei de date: Persoana, Zbor, Aeroport, dar si Bilet si de asemenea, datorita faptului ca desgin pattern-ul ales de mine a fost „Composite Design Pattern”, sunt prezente si clase ca „ZborComposite” si „ZborItem”.

Clasa ZborComposite reprezinta o clasa alcatuita din mai multe potentiale obiecte de tipul Zbor, acest lucru usureaza crearea zborurilor formate din escale, avand ca variabila instanta o lista de obiecte Zbor.

Clasa ZborItem reprezinta generalizarea obiectelor de tip Zbor si ZborComposite, fiecare din cele doua extinde clasa ZborItem, pentru a fi folosit si in cazul in care traseul cautat de client este simplu (exista zbor direct) sau este format din escale (nu exista zbor direct).

Cautarea traseelor este realizata in clasa „CautareZborCreareBilet” din pachetul „aministrate”. Pentru realizarea acestui obiectiv am implementat cautarea unui zbor in functie de ID-ul aeroportului de plecare si a unui aeroport in functie de ID-ul sau pentru a gasi un zbor din care traseul care porneste din prima locatie introdusa de client. Cautarea propriu-zisa se efectueaza folosindu-se algoritmul de cautare A-Star, utilizand ca euristica pretul minim dintre curse. De asemenea structurile folosite sunt PriorityQueue, HashSet si Triplet (utilizand JavaTuples) impreuna cu Aeroport, ZborComposite si Integer (pretul traseului).

Diagramele utilizate in aceasta parte a proiectului sunt diagrama SQL si diagramele de secventa, prezentate capitolul „Diagrame”.

1. Administrare, printare bilet (Gorgan Raul). Diagrama de activitate si Command Pattern.

Partea de administrare a site-ului se gaseste in pachetul „administrate”. Administrarea site-ului se poate face doar de catre o persoana care are acest drept, adica detine un cont de admin. Actiunile pe care le poate face un admin este de: adaugare de noi zboruri sau aeroporturi, editarea informatiilor unui zbor sau a unui aeroport, stergerea lor sau vizualizarea lor intr-un tabel. Administrarea se face intr-un mod simplu, intuitiv. Administratorul introduce datele cerute si ii va aparea o confirmare in caz de succes sau un mesaj de eroare.

Pentru aceasta parte s-a folosit design pattern-ul Command. Clasele existente sunt: interfata Command unde sunt metodele neimplementate, clasele InsertCommand, UpdateCommand si DeleteCommand care implementeaza interfata, clasa Invoker si clasa Operation.

Interfata Command are doua metode, una pentru operatii pe tabelul Aeroport, una pentru tabelul Zbor. Clasa Invoker este apelata din controller-ul aplicatiei cu comanda si parametri corespunzatori. In functie de comanda data, se va apela metoda potrivita din clasa Operation. Clasa Operation contine metodele care actioneaza direct asupra bazei de date. Relatia dintre aceste clase este ilustrata in sectiunea 3, prin diagrama de clase.

Pentru generarea biletului de avion in format pdf s-a folosit o librarie numita ITextPDF. Implementarea se gaseste in pachetul „tichet”, clasa „ComputeTicket”. S-a ales un design placut care contine cod QR, date despre persoana care a achitat biletul, date despre zbor, despre companie.

Diagramele utilizate in aceasta parte a proiectului sunt diagrama de clase Command Pattern si diagramele de activitate, prezentate capitolul „Diagrame”.

1. Log in Cociubei Mihai

Pentru implementarea acestei functionalitati am folosit design pattern „Chain of Responsability” acest tip de design pattern da posibilitatea de a transmite informatia de-a lungul unui lant, ceea ce este esential in cadrul logIn. In primul rand utilizatorul isi introduce datele acestea sunt transmise la structurile care se ocupa de procesare. Se cauta userul dupa email decriptam parola iar in cazul in care informatiile au fost introduse se trece la pasul urmator. Acesta consta in verificarea permisiunilor utilizatorului si cerintele pagini pe care doreste sa acceseze. In cazul in care totul merge bine si utilizatorul are permisiuni va fi redirectionat catre pagina urmatoare, in caz contrar va fi blocat.

Sablonul lantului de responsabilitati (chain of responsibility) este un sablon comportamental care permite evitarea cuplarii directe a expeditorului unei cereri cu un anumit destinatar, folosinduse in acest sens clase intermediare.

Penru ca un utilizator sa poata pune o comanda trebuie sa se logheze introducand informatiile cerute Fig10 (1:logIn request) dupa care i sunt verificate datele urmand sa fie redirectionat spre pagina ceruta. Pune comanda poate iesi.

Un admin parcurge exact aceiasi pasi dar el va fi redirectionat spre o pahina unde are mai multe optiuni (editare useri, editare zboruri/aeroporturi) .

Pentru o implementare mai usoara am decis ca un utilizator sa aiba mai multe permisiuni si nu spre exemplu daca e admin sa includa si ceea ce poate face un user ci un admin poate fi si user in acelasi timp astfel putand sa puna si comenzi.

Un alt detaliu de implementare util este ca am facut un tabel rol pentru ca in viitoarele dezvoltari ale aplicatiei dezvoltatorii sa aiba posibilitatea de a introduce mai multe functionalitati cu usurinta si nu de a modifica toata implementarea ci doar un insert in tabel cu noul rol.

Pentru a ma asigura de o buna functionare a repository am implementat o serie de teste. Printre care cele mai importante sunt testAddRoleToNewUser () care testeaza atat insertul de noi useri cat si setarea rolului. testAddRolesToExistingUser() testeaza modificarea rolului unui utilizator deja existent ceea ce poate fi foarte util cand ne creem baza de date si avem nevoie sa ne setam pe noi ca admin ca ulterior sa putem edita si restul utilzatorilor.

Un admin poate schimba parola si permisiunile unui user.

O persoana care isi face cont automat o sa fie inregistrat ca si user iar pentru permisiuni suplimentare acesta trebuie sa fie modificat de catre un admin in cazul nostru.

1. **Diagrame**

Diagrama SQL:

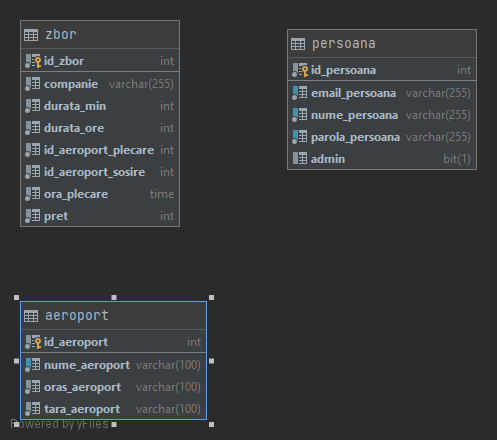


Fig. 1: Diagrama bazei de date

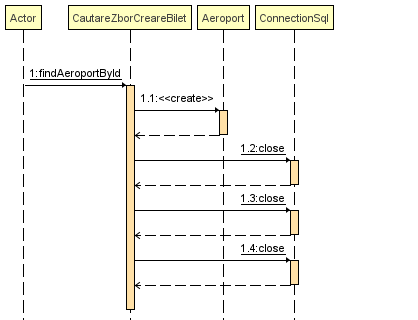


Fig. 2: Diagrama de secventa pentru cautarea unui aeroport in functie de ID-ul sau

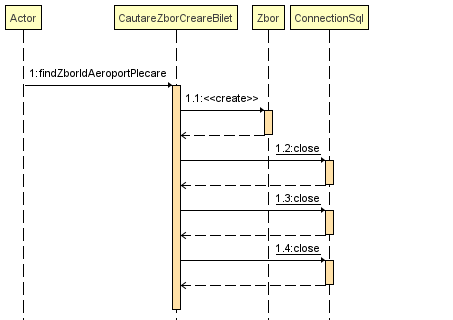


Fig. 3: Diagrama de secventa pentru cautarea unui zbor in functie de ID-ul aeroportului de plecare

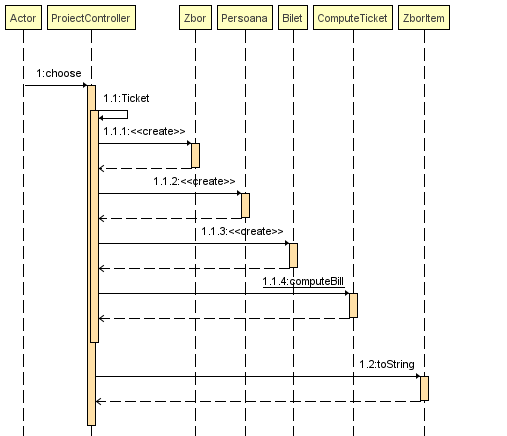


Fig. 4: Diagrama de secventa pentru crearea biletului atunci cand clientul acceseaza butonul „Buy”

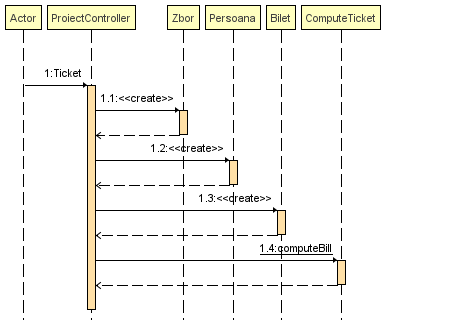


Fig. 5: Diagrama de secventa pentru crearea biletului

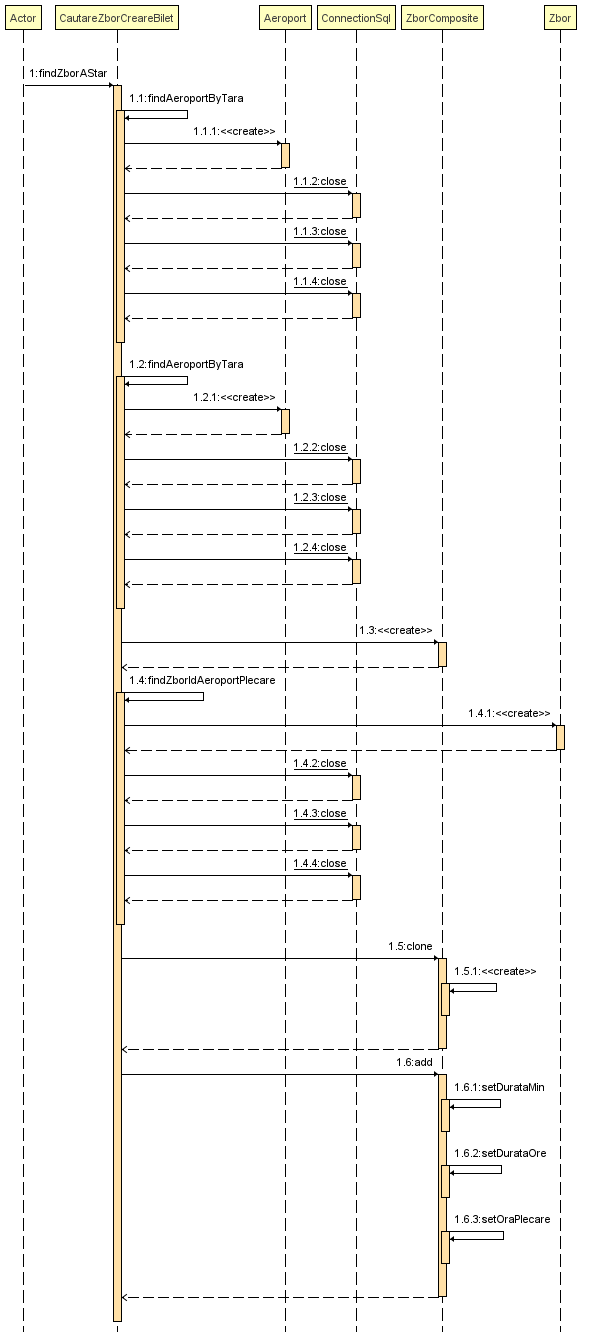


Fig. 6: Diagrama de secventa pentru algoritmul de cautare a celui mai eficient traseu pentru client, metoda „FindZborAStar”

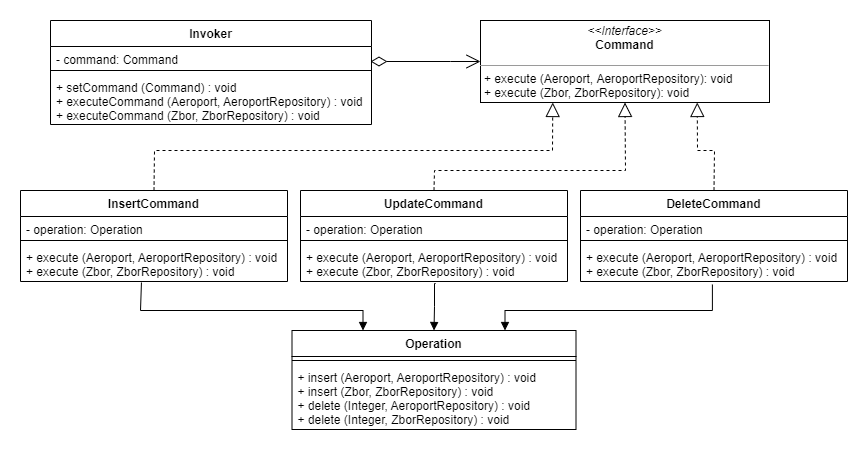


Fig. 7: Diagrama de clase al pachetului „administrate”.

Command Design Pattern

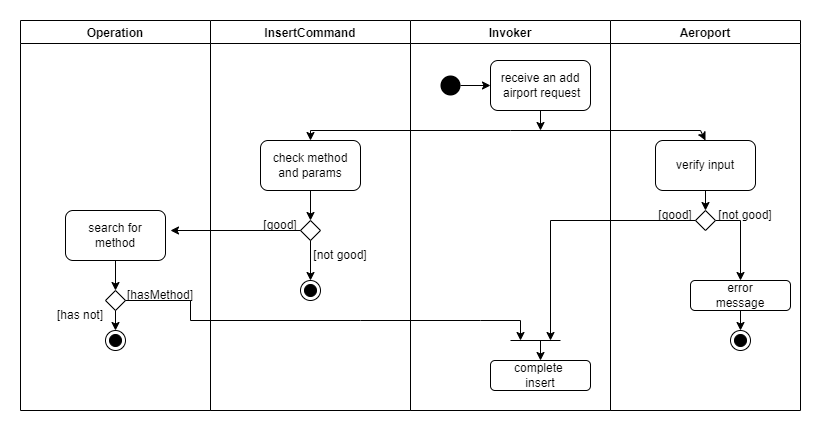


Fig. 8: Diagrama de activitate la operatia de

adaugare aeroport

(similar si la adaugare zbor)

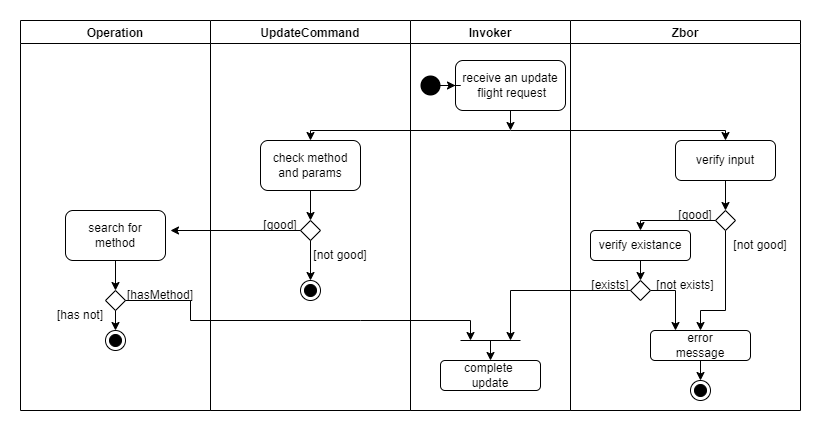


Fig. 9: Diagrama de activitate la operatia de

editare zbor

(similar si la editare aeroport)

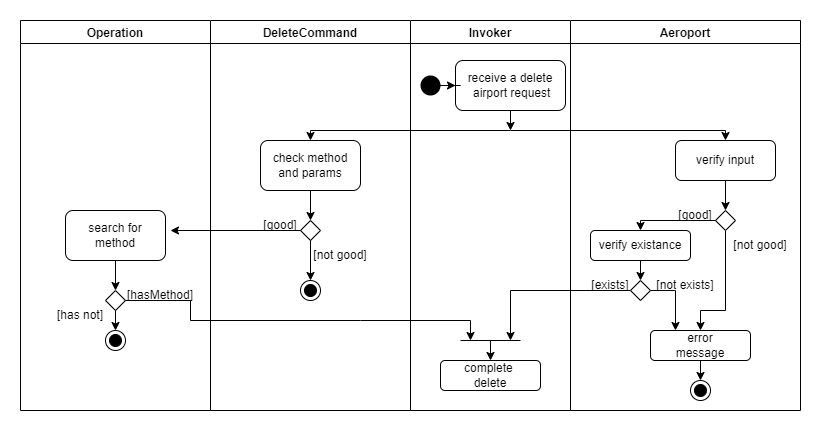
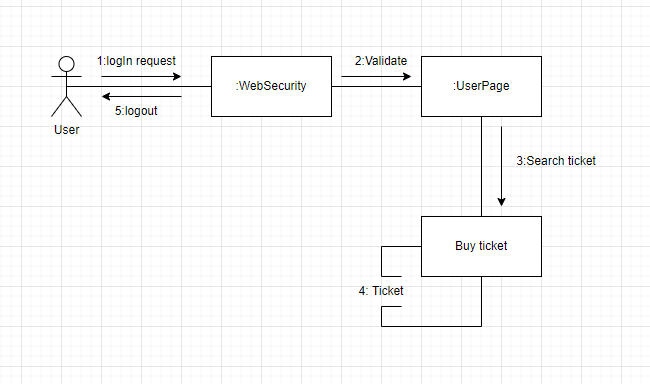


Fig. 10: Diagrama de activitate la operatia de

stergere aeroport

(similar si la stergere zbor)



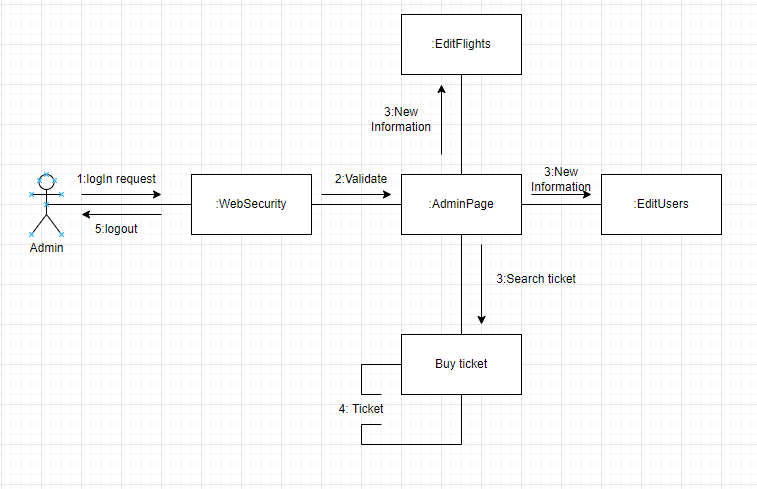
Fig. 11: Diagrama de comunicatie logIn Client

Fig. 11: Diagrama de comunicatie logIn Admin