



UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET
ODSJEK ZA AUTOMATIKU I ELEKTRONIKU

Izvještaj projekta

APLIKACIJA AERODROM
PREDMET: RAZVOJ PROGRAMSKIH RJESENJA

Student:
Amila Šikalo (17973)

Profesor:
Doc. dr Vedran Ljubović

Sarajevo,
30.01.2020.

Sadržaj

Popis slika	ii
1 Opis aplikacije	1
2 Osnovne ideje pri implementaciji	2
3 Implementacija	3
3.1 Osnovni koncepti	3
3.2 Baza podataka	3
3.3 Koncepti objektno orijentiranog programiranja	3
3.4 Grafički interfejs	4
3.5 Datoteke	4
3.6 Enumi	4
3.7 Interface	4
3.8 Generika	4
3.9 Mrežno programiranje i tredovi	5
3.10 Lokalizacija / Internacionalizacija	5
3.11 Funkcionalno programiranje	5
3.12 Izvještaji	5
3.13 Maven	5
3.14 Testiranje	5
3.15 Vlastiti izuzeci	6
3.16 Pokretanje	6
4 Buduće napredovanje	7

Popis slika

3.1	ERD baze podataka	3
-----	-----------------------------	---

Poglavlje 1

Opis aplikacije

Aplikacija opisana u ovom izvještaju predstavlja aplikaciju koja se može koristiti za upravljanje podacima vezanim za jedan aerodrom. Obzirom da sam još prošle godine uzela ovu temu, te da je prethodna verzija bila mnogo skromnije implementirana, odlučila sam da je nadogradim s potrebnim funkcionalnostima. Aplikacija je zamišljena na sljedeći način: osoba (User) koja upravlja aktivnostima na Aerodromu može imati različite uloge (Role) - najčešće su Administrator ili Operator. Osoba može unositi različite Avione (Airplanes), za koje je potrebno unijeti određene specifikacije (između ostalog i kojoj aviokompaniji pripadaju (Airline), te da jedan avion ne smije imati više od 300 sjedala [pretpostavlja se da aerodrom nije toliko veliki da može primiti avione veličine Boeing 777X]). Zatim, mogu se unositi vrste letova (Flight Types), Letovi (Flights), za koje je potrebno unijeti avion koji vrši taj let, na koji izlaz (Gate) putnik treba doći, koja je vrsta leta, vrijeme zauzimanja i oslobađanja piste. Naravno, potrebno je unijeti i putnika, pri čemu se putnik veže za let, te se generiše QR kod kod registriranja Check In-a. Uz putnika je vezana i prtljaga - koja može biti standardna, ručna, te dodatna. Pod standardnom prtljagom, smatra se prtljaga uključena u cijenu, koja ne ide u kabinu s putnikom. Pod ručnom prtljagom, smatra se prtljaga do 10 kg, te dodatnom cijenom. Pod dodatnom prtljagom, smatra se nestandardna prtljaga koju korisnik može imati uz sebe (recimo veća količina novca, metal i slično).

User ima mogućnost da izgeneriše različite vrste izvještaja: može dobiti izvještaj o svim putnicima, letovima, te svim osobama koje imaju pristup aplikaciji.

Također, aplikacija je prilagođena različitim podnebljima - postoji mogućnost promjene na osam svjetskih jezika.

Poglavlje 2

Osnovne ideje pri implementaciji

Uz rad je prikazan class diagram, kao i ERD dijagram projekta. Pri implementaciji korištena je SQLite baza podataka, obzirom da projekat nije preveliki da bi se koristile baze poput PostgreSQL. Osnovna ideja bila je da se prije svega razviju neke osnovne funkcionalnosti, te na kreativan način iskoriste oblasti koje su rađene u okviru ovoga predmeta. Izgled aplikacije je jednostavan obzirom da je prioritet demonstracija funkcionalnosti. U dosta situacija sam pokušala izbjeći duplikaciju koda, međutim, u nekim situacijama to nisam uspjela u potpunosti, obzirom da su neke metode bile dosta slične, a opet različite u odnosu na druge. I klase i kontroleri su napravljeni tako da prate JavaBeans specifikaciju, te Camel case notaciju.

Poglavlje 3

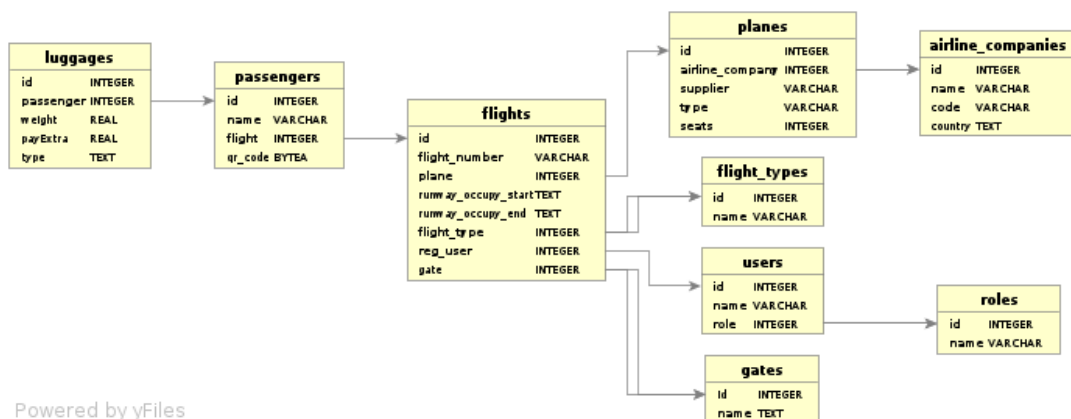
Implementacija

3.1 Osnovni koncepti

U projektu se koriste model klase koje prate JavaBeans specifikaciju. Zastupljena je i upotreba osnovnih Java kolekcija, klasičnih nizova i kolekcije ArrayList(). Također, korištene su mape kako bi se primijenilo stečeno znanje (između ostalog da se izlistaju sve Aviokompanije, koje imaju Avion prijavljen avion u bazi).

3.2 Baza podataka

Za bazu podataka je korištena SQLite baza podataka. Organizirana je kao standardna relacionala baza podataka čiji je Entity-Relationship-Diagram (ERD) dat na sljedećoj slici:



Slika 3.1: ERD baze podataka

Treba napomenuti da je baza dizajnirana u DB Browser for SQLite alatu, dok je ERD generisan u DB Visualizer alatu.

3.3 Koncepti objektno orijentiranog programiranja

Koncepti OOP-a (nasljeđivanje, polimorfizam i enkapsulacija) iskorišteni su pri kreiranju ove aplikacije. Najviše su korišteni za implementaciju različitih vrsta prtljaga. Osnovna, apstraktna klasa jeste AbstractLuggage, iz koje su izvedeni prvo Luggage, a zatim i Hand Luggage, te

Additional Luggage. Kompletan dijagram klasa nije dio ovog izvještaja, zbog veličine, ali se nalazi u prilogu. Dijagram klasa je generisan korištenjem alata Umbrello.

3.4 Grafički interfejs

Prilikom implementiranja grafičkog interfejsa korišteni su razni GUI elementi. Funkcije koje se nalaze unutar tih elemenata su validirane unutar kontrolera unutar kojih se nalaze listeneri, koji osiguravaju validan unos podataka. Korišteno bidirekcionalno povezivanje polja. Implementirana je i opcija search unutar Airlines, koji vrši pretražu aviokompanija po nazivu. Za tu svrhu su korišteni streamovi, te je ta opcija implemetirana unutar glavnog kontrolera.

Uzeti su u obzir koncepti dobrog dizajna korisničkog interfejsa (Gestalt principi) gdje su svi logički vezani elementi grupisani i na ekranu. Za prikaz validnih/nevalidnih polja pozivamo se na .css file, te se postavlja određena boja u zavisnosti od toga da li je polje validno ili ne. Također, podešeno je da prozori ne mogu biti Resizable.

3.5 Datoteke

Datoteke su korištene pri generisanju QR koda, korištenjem biblioteke ZXing. Korišteno je zapisivanje u datoteku qrcode.png (klasa Utills), koja se poslije koristi za generisanje Boarding Pass izvještaja.

3.6 Enumi

Enumi se koriste u izvedenoj klasi Additional Luggage, za odabir između različitih tipova dodatne prtljage. Definirana su tri različita tipa: Metal, Clothes, Money.

3.7 Interface

U projektu se koristi jedan interfejs FlightInterface, u kojem su definirane dvije metoda, te kasnije implementirane u Flights, te pozvane u Mainu. Prva metoda obavještava da je preostalo još sat vremena do boardinga, dok druga obavještava da je "posljednji voz" za ukrcavanje, te se javlja 30 minuta prije leta (obje su vezane za metodu startOfUsingTheRunway). Obavješene se šalje kao ispis u konzolu. Ovo se može poslije reimplementirati da šalje i na drugačiji način (recimo na eksterni displej ili kiosk). Za sada je samo kao *proof of concept*.

3.8 Generika

Iako nisu korištene generičke klase i metode u striktnom smislu, korištena je evaluacija tipova objekata koji predstavljaju prtljagu, na osnovu unešenih parametara. Obzirom da u startu nisam razmišljala o korištenju ovih koncepata, naknadni dizajn bi zahtijevao jako mnogo pomjena unutar kompletnog projekta.

3.9 Mrežno programiranje i tredovi

Mrežno programiranje se koristi da bi se sa Countries REST API-a (<https://restcountries.eu/rest/v2/all>) dobavila lista zemalja. Ovo je riješeno u okviru klase `AirlineController` koja popunjava `ComboBox` za odabir zemlje. Ujedno je iskorišten thread kojim se ovo dobavljanje i parsiranje JSON objekta izvršava u pozadini. Korisnik je u okviru UI-a obaviješten, korištenjem `ProgressIndicator` klase iz JavaFX-a da je u toku dobavljanje (*loading*).

3.10 Lokalizacija / Internacionalizacija

Lokalizacija / Internacionalizacija je zastupljena u dijelu View, gdje korisnik može odabrati između različitih jezika: bosanski, engleski, španjolski, njemački, francuski, ruski, arapski i kineski jezik.

3.11 Funkcionalno programiranje

Funkcionalno programiranje je iskorišteno na više mjesta, korištenjem lambda funkcija. Npr. u slučaju listenera u kontrolerima ili kao event handling prilikom dobavljanja odgovora iz REST servisa.

3.12 Izvještaji

Unutar projekta postoje tri različita izvještaja koji se mogu kreirati: Flights Report, Passengers Report, Users Report. Sva tri izvještaja se formiraju na principu formiranja Java kolekcije, koja se zatim prosljeđuje Jasper kompajleru. Na taj način se formira izgled izvještaja, koji se kasnije prikazuje na ekranu.

3.13 Maven

Korišten je Maven alat za automatsku izgradnju koda. Uvedene su sljedeće dependencies: `sqlite-jdbc`, `jasperreports`, `zxing` (biblioteka za rad sa QR kodovima), `tornado` (biblioteka za `Date-TimePicker` komponentu). Pored toga, za testiranje su korišteni `junit5` i `testfx` biblioteke. Svi dependencies se mogu pogledati detaljno u `pom.xml` datoteci. Svim ovim smo postigli i da se generiše izvršni `.jar` file u target folderu (`rpr-projekat-1.0-SNAPSHOT-jar-with-dependencies.jar`) koji se može pokrenuti i iz komandne linije.

3.14 Testiranje

Za testiranje ovoga projekta korišteni su JUnit testovi, kroz JUnit5 i TestFX biblioteke. Ukupno je generisano 250 testova, pri čemu je Code Coverage po klasama 88%, dok je pokrivenost metoda 80 %. U nastavku je dato objašnjenje zbog čega nije potpuna automatska pokrivenost koda.

Željela bih naglasiti da se znalo dešavati da `MainTest` klasa testova (u kojoj se nalaze TestFX testovi), ponekad padne i prije nego se pokrene, kao i da neki testovi znaju pasti kada se pokrenu svi TestFX testovi uz Code Coverage (pojedinačno prolaze svi, bez Code Coverage). Pokušala

sam dati problem riješiti tako da stavljam pauze pomoću funkcije `Thread.sleep()`, međutim, i dalje se zna desiti da neki padnu pri pokretanju svih. Ostatak testiranja koji nedostaje do 100%, za koji nisam našla pogodan način kako automatski testirati, jesu Reports (Users, Passengers, Flights), te Lokalizacija.

3.15 Vlastiti izuzeci

Vlastiti izuzeci su iskorišteni u dijelu koda gdje se unosi broj sjedišta unutar aviona.

3.16 Pokretanje

Najjednostavnije pokretanje je direktno iz IntelliJ IDE-a. Da bi se to omogućilo, potrebno je u postavkama Mavena u dijelu Runner označiti polje (CheckBox) *Delegate IDE build/run actions to Maven*. Nakon toga, potrebno je desnim klikom na klasnu `Main.java` odabrati Run i IDE će pozvati Maven build sekvencu, te pokrenuti aplikaciju.

Poglavlje 4

Buduće napredovanje

Svjesna sam da je ovaj projekat namijenjen za integraciju svih lekcija koje smo naučili na ovome predmetu, ali moram naglasiti da postoji dosta stvari koje bih implemetirala u daljnjem radu. Osim većeg broja funkcionalnosti (kao što su dodatne aktivnosti na aerodromu, radnje, restorani, lista nepoželjnih putnika,...), potrebno je obratiti pažnju i na ulazak u aplikaciju. Obzirom da nisam razmišljala o tome kada sam započela aplikaciju, bilo je zaista teško promijeniti sve ispočetka, te započeti s tim. Trebalo bi se omogućiti da samo jedna (ili nekoliko osoba) ima mogućnost da se upiše u bazu kao administrator, te da ona može uređivati sve podatke unutar aplikacije, dok osoba koje je operator može samo dodati letove, putnike i prtljagu. Zbog toga mislim da bi ova aplikacija bila bolja da je tako implementirana, odnosno korištenjem autentikacije (poželjno dvostruke autentikacije zbog veće sigurnosti). Također, bilo bi poželjno da se aplikacija (bar view mode) prilagodi mobilnim telefonima, zbog lakšeg pristupa podacima.