SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

Seminar

**Objektno oblikovanje**

Flight Control

*Dokumentacija*

Mislav Žabčić, Natko Bišćan, Domagoj Grizelj

Zagreb, siječanj, 2018.

**Sadržaj**

1. Uvod

1.1 Design story

2. Use case

3. Opis objektnog modela domene

3.1 Class dijagram

3.2 Opis implementacije perzistencije

4. Opis izgrađenih REST servisa

4.1 Opis implementacije i testova

4.2 Objavljivanje web servisa

4.3 API dokumentacija

5. Opis izgrađene desktop aplikacije

6. Opis izgrađene mobilne aplikacije

7. Podjela posla

1. **Uvod**

Cilj projekta je bio kreirati aplikaciju u kojoj će se prikazati primjena tehnika i principa objektnog oblikovanja i razvoja na temelju testova. (test driven development). Izrađena je mobilna i desktop aplikacija za vođenje evidencije o letovima jedne aviokompanije.

1.1 Design story

FlightControl interna je aplikacija za airline kompanije za lakši pregled poslovanja i vođenje evidencije o internim procesima letova.

Služi za pohranu raznih podataka bitnih za vođenje takve vrste kompanije. Glavna dva fokusa aplikacije su podaci o letovima i prodaja karata. Kompaniji je bitna evidencija prodaja karata između ostalog i kako bi saznala koje trgovine bolje posluju, koji zaposlenici su prodali najviše karata, je li korisnije prodavati karte preko fizičkih ili web trgovina i slično, kako bi se mogla uvesti poboljšanja i optimizacije u pojedine aspekte poslovanja.

Bitno je bilježiti putnike kako bi ih se, primjerice, moglo kontaktirati u slučaju izgubljene ili neisporučene prtljage ili neke slične pogreške. Evidencija se vodi i o potražnji business klase sjedala te općenita potražnja za određenim putovanjima. S druge strane, kompanija mora bilježiti informacije o letovima, voditi računa o prometu i stanjima svojih aviona. Aplikacija omogućuje i prijavu pilota, te pregled ruta letova i aerodroma.

Korisniku je omogućen i statistički pregled prodaje i rezervacija karata koji ukazuje na uspješnost poslovanja pojedinih zaposlenika, poslovnica i internet prodaje. Postoji i statistika leta korisna kontrolorima, pilotima i sl.

**2. Use case**

U ovom poglavlju ćemo nabrojati sve scenarije korištenja te ih detaljnije opisati na principu Use case scenarija.

Kreirali smo sljedeće scenarije korištenja:

* Pregled karata određenog leta
* Dodavanje nove karte
* Uređivanje podataka karte
* Brisanje karte
* Pregled poslovnica kompanije
* Dodavanje nove poslovnice
* Uređivanje podataka poslovnice
* Brisanje poslovnice
* Pregled putnika određenog leta
* Dodavanje novog putnika
* Uređivanje putnika
* Brisanje putnika
* Pregled pilota
* Dodavanje novog pilota
* Uređivanje pilota
* Brisanje pilota
* Pregled aviona
* Dodavanje novog aviona
* Uređivanje aviona
* Brisanje aviona
* Postavljanje aviona u status “Pokvaren”
* Postavljanje aviona u status “Ispravan”
* Pregled aerodroma
* Dodavanje novog aerodroma
* Uređivanje aerodroma
* Brisanje aerodroma
* Postavljanje aerodroma u status “Neaktivan”
* Pregled ruta
* Dodavanje ruta
* Uređivanje ruta
* Brisanje ruta
* Pregled letova
* Dodavanje leta
* Uređivanje leta
* Brisanje leta
* Otkazivanje leta
* Statistika o prodanim kartama
* Statistika o prodavačima

1. Pregled karata određenog leta:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled karata iz padajućeg izbornika
2. Sustav prikazuje listu karata te omogućuje korisniku daljnje akcije

Alternativni scenarij:

1. Korisnik odabire pregled karata iz padajućeg izbornika
2. Ne postoji ni jedna karta
3. Korisnik ima mogućnost kreiranja nove karte

2. Dodavanje nove karte:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire dodavanje nove karte iz padajućeg izbornika
2. Sustav prikazuje formu za dodavanje nove karte
3. Korisnik unosi podatke potrebne za kreiranje nove karte
4. Korisnik pritiskom na tipku “Add” potvrđuje podatke
5. Sustav kreira novu kartu

Alternativni scenarij:

1. Korisnik nije unio sve potrebne podatke
2. Sustav ga upozorava koja polja mora ispuniti
3. Korisnik nije pravilno unio podatke
4. Sustav ga upozorava koja polja treba ispraviti

3. Uređivanje podataka karte:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled karata iz padajućeg izbornika
2. Korisnik odabire određenu kartu
3. Korisnik pritišće tipku “Edit”
4. Sustav prikazuje formu za uređivanje karte
5. Korisnik uređuje podatke i odabire “Save”

Alternativni scenarij:

1. Korisnik odabire pregled karata iz padajućeg izbornika
2. Korisnik odabire određenu kartu
3. Korisnik pritišće tipku “Edit”
4. Korisnik pogrešno unosi podatke
5. Sustav ga upozorava na greške pri izmjeni podataka
6. Korisnik ispravlja pogrešne unose

4. Brisanje karte:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled karata iz padajućeg izbornika
2. Korisnik odabire određenu kartu
3. Korisnik pritišće tipku “Remove”

5. Pregled poslovnica kompanije:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled poslovnica iz padajućeg izbornika
2. Sustav prikazuje listu poslovnica te omogućuje korisniku daljnje akcije

Alternativni scenarij:

1. Korisnik odabire pregled poslovnica iz padajućeg izbornika

2. Ne postoji ni jedna poslovnica

3. Korisnik ima mogućnost kreiranja nove poslovnice

6. Dodavanje nove poslovnice:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire dodavanje nove poslovnice iz padajućeg izbornika
2. Sustav prikazuje formu za dodavanje nove poslovnice
3. Korisnik unosi podatke potrebne za kreiranje nove poslovnice
4. Korisnik pritiskom na tipku “Add” potvrđuje podatke
5. Sustav kreira novu poslovnice

Alternativni scenarij:

1. Korisnik nije unio sve potrebne podatke
2. Sustav ga upozorava koja polja mora ispuniti
3. Korisnik nije pravilno unio podatke
4. Sustav ga upozorava koja polja treba ispraviti

7. Uređivanje podataka poslovnice:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled poslovnica iz padajućeg izbornika
2. Korisnik odabire određenu poslovnicu
3. Korisnik pritišće tipku “Edit”
4. Sustav prikazuje formu za uređivanje poslovnice
5. Korisnik uređuje podatke i odabire “Save”

Alternativni scenarij:

1. Korisnik odabire pregled poslovnica iz padajućeg izbornika
2. Korisnik odabire određenu poslovnicu
3. Korisnik pritišće tipku “Edit”
4. Korisnik pogrešno unosi podatke
5. Sustav ga upozorava na greške pri izmjeni podataka
6. Korisnik ispravlja pogrešne unose

8. Brisanje poslovnice:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled poslovnica iz padajućeg izbornika
2. Korisnik odabire određenu poslovnicu
3. Korisnik pritišće tipku “Remove”

Alternativni scenarij:

1. Korisnik odabire pregled poslovnica iz padajućeg izbornika
2. Greška: nije moguće obrisati poslovnicu sa zaposlenicima

9. Pregled putnika određenog leta:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled putnika iz padajućeg izbornika
2. Sustav prikazuje listu putnika te omogućuje korisniku daljnje akcije

Alternativni scenarij:

1. Korisnik odabire pregled putnika iz padajućeg izbornika
2. Ne postoji ni jedan putnik
3. Korisnik ima mogućnost kreiranja novog putnika

10. Dodavanje novog putnika:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire dodavanje novog putnika iz padajućeg izbornika
2. Sustav prikazuje formu za dodavanje novog putnika
3. Korisnik unosi podatke potrebne za kreiranje novog putnika
4. Korisnik pritiskom na tipku “Add” potvrđuje podatke
5. Sustav kreira novog putnika

Alternativni scenarij:

1. Korisnik nije unio sve potrebne podatke
2. Sustav ga upozorava koja polja mora ispuniti
3. Korisnik nije pravilno unio podatke
4. Sustav ga upozorava koja polja treba ispraviti

11. Uređivanje podataka putnika:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled putnika iz padajućeg izbornika
2. Korisnik odabire određenog putnika
3. Korisnik pritišće tipku “Edit”
4. Sustav prikazuje formu za uređivanje podatak putnika
5. Korisnik uređuje podatke i odabire “Save”

Alternativni scenarij:

1. Korisnik odabire pregled putnika iz padajućeg izbornika
2. Korisnik odabire određenog putnika
3. Korisnik pritišće tipku “Edit”
4. Korisnik pogrešno unosi podatke
5. Sustav ga upozorava na greške pri izmjeni podataka
6. Korisnik ispravlja pogrešne unose

12. Brisanje putnika:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled putnika iz padajućeg izbornika
2. Korisnik odabire određenog putnika
3. Korisnik pritišće tipku “Remove”

13. Pregled pilota:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled pilota iz padajućeg izbornika
2. Sustav prikazuje listu pilota te omogućuje korisniku daljnje akcije

Alternativni scenarij:

1. Korisnik odabire pregled pilotaiz padajućeg izbornika

Ne postoji niti jedan pilot

Korisnik ima mogućnost dodavanja novog pilota

14. Dodavanje novog pilota:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire dodavanje novog pilota iz padajućeg izbornika
2. Sustav prikazuje formu za dodavanje novog pilota
3. Korisnik unosi podatke potrebne za kreiranje novog pilota
4. Korisnik pritiskom na tipku “Add” potvrđuje podatke
5. Sustav kreira novog pilota
6. Otvara se stranica pregleda pilota

Alternativni scenarij:

1. Korisnik nije unio sve potrebne podatke
2. Sustav ga upozorava koja polja mora ispuniti
3. Korisnik nije pravilno unio podatke
4. Sustav ga upozorava koja polja treba ispraviti

15. Uređivanje pilota:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled pilota iz padajućeg izbornika
2. Korisnik odabire određenog pilota
3. Korisnik pritišće tipku “Edit”
4. Sustav prikazuje formu za uređivanje pilota
5. Korisnik uređuje podatke i odabire “Save”

Alternativni scenarij:

1. Korisnik odabire pregled pilota iz padajućeg izbornika
2. Korisnik odabire određenog pilota
3. Korisnik pritišće tipku “Edit”
4. Korisnik pogrešno unosi podatke
5. Sustav ga upozorava na greške pri izmjeni podataka
6. Korisnik ispravlja pogrešne unose

16. Brisanje pilota:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled pilota iz padajućeg izbornika
2. Korisnik odabire određenog pilota
3. Korisnik pritišće tipku “Remove”

Alternativni scenarij:

1. Postoje letovi kojima pilot upravlja
2. Sustav javlja grešku. “Niije moguće obrisati pilota koji ima letove”

17. Pregled aviona:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled aviona iz padajućeg izbornika
2. Sustav prikazuje listu aviona te omogućuje korisniku daljnje akcije

Alternativni scenarij:

1. Korisnik odabire pregled aviona padajućeg izbornika
2. Ne postoji niti jedan avion
3. Korisnik ima mogućnost dodavanja novog aviona

18. Dodavanje novog aviona:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire dodavanje novog aviona iz padajućeg izbornika
2. Sustav prikazuje formu za dodavanje novog aviona
3. Korisnik unosi podatke potrebne za kreiranje novog aviona
4. Korisnik pritiskom na tipku “Add” potvrđuje podatke
5. Sustav kreira novi avion
6. Otvara se stranica pregleda aviona

Alternativni scenarij:

1. Korisnik nije unio sve potrebne podatke
2. Sustav ga upozorava koja polja mora ispuniti
3. Korisnik nije pravilno unio podatke
4. Sustav ga upozorava koja polja treba ispraviti

19. Uređivanje aviona:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled aviona iz padajućeg izbornika
2. Korisnik odabire određeni avion
3. Korisnik pritišće tipku “Edit”
4. Sustav prikazuje formu za uređivanje aviona
5. Korisnik uređuje podatke i odabire “Save”

Alternativni scenarij:

1. Korisnik odabire pregled aviona iz padajućeg izbornika
2. Korisnik odabire određeni avion
3. Korisnik pritišće tipku “Edit”
4. Korisnik pogrešno unosi podatke
5. Sustav ga upozorava na greške pri izmjeni podataka
6. Korisnik ispravlja pogrešne unose

20. Brisanje aviona:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled aviona iz padajućeg izbornika
2. Korisnik odabire određeni avion
3. Korisnik pritišće tipku “Remove”

Alternativni scenarij:

1. Postoje letovi koji su vezani na avion
2. Sustav javlja grešku. “Niije moguće obrisati avion koji ima letove”

21. Postavljanje aviona u status “Pokvaren”:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled aviona iz padajućeg izbornika
2. Korisnik pored određenog aviona pritišće gumb “Broken”
3. Avion prelazi u stanje pokvaren

Alternativni scenarij:

1. Postoje letovi koji su vezani na avion
2. Sustav javlja grešku. “Prvo promijenite avion na letovima: {broj leta} ”

22. Postavljanje aviona u status “Ispravan”:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled aviona iz padajućeg izbornika
2. Korisnik pored određenog aviona pritišće gumb “Working”
3. Avion prelazi u stanje ispravan

23. Pregled aerodroma:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled aerodroma iz padajućeg izbornika
2. Sustav prikazuje listu aviona te omogućuje korisniku daljnje akcije

Alternativni scenarij:

1. Korisnik odabire pregled aerodroma padajućeg izbornika
2. Ne postoji niti jedan aerodrom
3. Korisnik ima mogućnost dodavanja novog aerodroma

24. Dodavanje novog aerodroma:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire dodavanje novog aerodroma iz padajućeg izbornika
2. Sustav prikazuje formu za dodavanje novog aerodroma
3. Korisnik unosi podatke potrebne za kreiranje novog aerodroma
4. Korisnik pritiskom na tipku “Add” potvrđuje podatke
5. Sustav kreira novi aerodrom
6. Otvara se stranica pregleda aerodroma

Alternativni scenarij:

1. Korisnik nije unio sve potrebne podatke
2. Sustav ga upozorava koja polja mora ispuniti
3. Korisnik nije pravilno unio podatke
4. Sustav ga upozorava koja polja treba ispraviti

25. Uređivanje aerodroma:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled aerodroma iz padajućeg izbornika
2. Korisnik odabire određeni aerodrom
3. Korisnik pritišće tipku “Edit”
4. Sustav prikazuje formu za uređivanje aerodroma
5. Korisnik uređuje podatke i odabire “Save”

Alternativni scenarij:

1. Korisnik odabire pregled aerodroma iz padajućeg izbornika
2. Korisnik odabire određeni aerodrom
3. Korisnik pritišće tipku “Edit”
4. Korisnik pogrešno unosi podatke
5. Sustav ga upozorava na greške pri izmjeni podataka
6. Korisnik ispravlja pogrešne unose

26. Brisanje aerodroma:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled aerodroma iz padajućeg izbornika
2. Korisnik odabire određeni aerodrom
3. Korisnik pritišće tipku “Remove”

Alternativni scenarij:

1. Postoje rute letova koje su vezane na aerodrom
2. Sustav javlja grešku. “Niije moguće obrisati aerodrom koji je na nekim rutama letenja”

27. Stavljanje aerodroma u status “Neaktivan”:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled aerodroma iz padajućeg izbornika
2. Korisnik odabire određeni aerodrom
3. Korisnik pritišće tipku “Inactive”

Alternativni scenarij:

1. Postoje rute letova koje su vezane na aerodrom
2. Sustav javlja grešku. “Niije moguće staviti aerodrom u status neaktivan ako postoje neki letovi koji dolaze u budućnosti na taj aerodrom ili poljeću s njega”

28. Pregled ruta:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled ruta iz padajućeg izbornika
2. Sustav prikazuje listu ruta te omogućuje korisniku daljnje akcije

Alternativni scenarij:

1. Korisnik odabire pregled ruta padajućeg izbornika
2. Ne postoji niti jedan ruta
3. Korisnik ima mogućnost dodavanja nove rute

29. Dodavanje nove rute:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire dodavanje nove rute iz padajućeg izbornika
2. Sustav prikazuje formu za dodavanje nove rute
3. Korisnik unosi podatke potrebne za kreiranje nove rute
4. Korisnik pritiskom na tipku “Add” potvrđuje podatke
5. Sustav kreira novu rutu
6. Otvara se stranica pregleda ruta

Alternativni scenarij:

1. Korisnik nije unio sve potrebne podatke
2. Sustav ga upozorava koja polja mora ispuniti
3. Korisnik nije pravilno unio podatke
4. Sustav ga upozorava koja polja treba ispraviti

30. Uređivanje rute:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled rute iz padajućeg izbornika
2. Korisnik odabire određenu rutu
3. Korisnik pritišće tipku “Edit”
4. Sustav prikazuje formu za uređivanje rute
5. Korisnik uređuje podatke i odabire “Save”

Alternativni scenarij:

1. Korisnik odabire pregled ruta iz padajućeg izbornika
2. Korisnik odabire određenu rutu
3. Korisnik pritišće tipku “Edit”
4. Korisnik pogrešno unosi podatke
5. Sustav ga upozorava na greške pri izmjeni podataka
6. Korisnik ispravlja pogrešne unose

31. Brisanje rute:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled ruta iz padajućeg izbornika
2. Korisnik odabire određenu rutu
3. Korisnik pritišće tipku “Remove”

Alternativni scenarij:

1. Postoje letovi koji su vezani uz tu rutu
2. Sustav javlja grešku. “Niije moguće obrisati rutu koji je na nekom letu”

32. Pregled letova:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled letova iz padajućeg izbornika
2. Sustav prikazuje listu letova te omogućuje korisniku daljnje akcije

Alternativni scenarij:

1. Korisnik odabire pregled letova iz padajućeg izbornika
2. Ne postoji niti jedan let
3. Korisnik ima mogućnost dodavanja novog leta

33. Dodavanje novog leta:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire dodavanje novog leta iz padajućeg izbornika
2. Sustav prikazuje formu za dodavanje novog leta
3. Korisnik unosi podatke potrebne za kreiranje novog leta
4. Korisnik pritiskom na tipku “Add” potvrđuje podatke
5. Sustav kreira novi let
6. Otvara se stranica pregleda letova

Alternativni scenarij:

1. Korisnik nije unio sve potrebne podatke
2. Sustav ga upozorava koja polja mora ispuniti
3. Korisnik nije pravilno unio podatke
4. Sustav ga upozorava koja polja treba ispraviti

34. Uređivanje leta:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled letova iz padajućeg izbornika
2. Korisnik odabire određeni let
3. Korisnik pritišće tipku “Edit”
4. Sustav prikazuje formu za uređivanje letova
5. Korisnik uređuje podatke i odabire “Save”

Alternativni scenarij:

1. Korisnik odabire pregled letova iz padajućeg izbornika
2. Korisnik odabire određeni let
3. Korisnik pritišće tipku “Edit”
4. Korisnik pogrešno unosi podatke
5. Sustav ga upozorava na greške pri izmjeni podataka
6. Korisnik ispravlja pogrešne unose

35. Brisanje leta:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled letova iz padajućeg izbornika
2. Korisnik odabire određeni let
3. Korisnik pritišće tipku “Remove”

Alternativni scenarij:

1. Postoje karte koje su vezane uz taj let
2. Sustav javlja grešku. “Niije moguće obrisati let za koji je netko kupio kartu”

36. Otkazivanje leta:

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled letova iz padajućeg izbornika
2. Korisnik odabire određeni let
3. Korisnik pritišće tipku “Cancel”

37. Statistika o prodanim kartama

Razina: korisnički cilj

Aktori: korisnik

Glavni scenarij:

1. Korisnik odabire pregled statistike prodanih karata u izborniku
2. Korisnik odabire let
3. Sustav prikazuje broj prodanih karata, te udio prodanih karata tog leta po prodavaču

Alternativni scenarij:

1. Korisnik odabire pregled statistike prodanih karata u izborniku
2. Korisnik odabire let
3. Ne postoji taj let
4. Sustav obavještava korisnika, da taj let ne postoji

38. Statistika o prodavačima

Razina: korisnički cilj

Aktori: korisnik

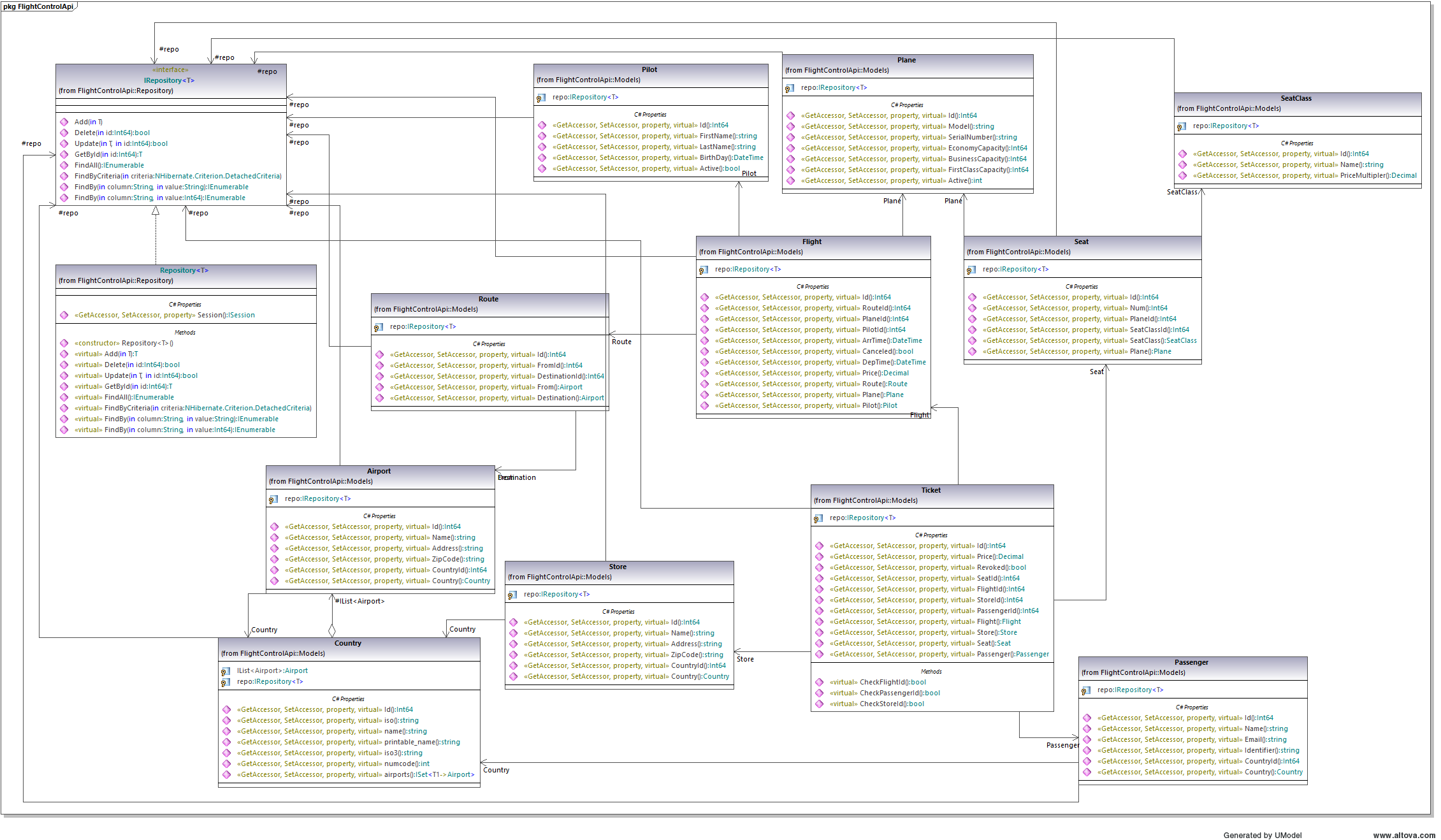
Glavni scenarij:

1. Korisnik odabire pregled statistike o prodavačima u izborniku
2. Sustav prikazuje popis prodavača prema broju udjelu prodanih karata ove komapnije

**3. Opis objektnog modela domene**

U ovom poglavlju ćemo opisati objektni model domene. To ćemo ostvariti pomoću UML dijagrama klasa i kratkog opisa klasa u njemu. Opisati ćemo i opis implementacije perzistencije u našoj aplikaciji.

3.1 Class dijagram



**Slika 3.1 Dijagram razreda modela domene**

Razredi modela domene su ustvari preslikane tablice iz baze podataka. Svaka tablica ima svoju klasu. U svakoj klasi se nalazi dodatak u obliku objekata iz klase koja je povezana vezom one-to-many ili many-to-one s nekom drugom klasom. Npr. klasa *Ticket* sadrži objekte tipa *Seat, Flight, Passenger i Store* zato što tablica je tablica Ticket vezana sa tim tablicama preko stranih ključeva. U nekim slučajevima klase sadrže i listu objekata klasa kojima su vezani vezom many-to-one, npr. postoji klasa *FlightWithTickets* koja uz svoje atribute te agregate s vezom one-to-many sadrži i listu objekata klase *Ticket*, na taj način možemo vidjeti koje su sve karte prodane za taj let.

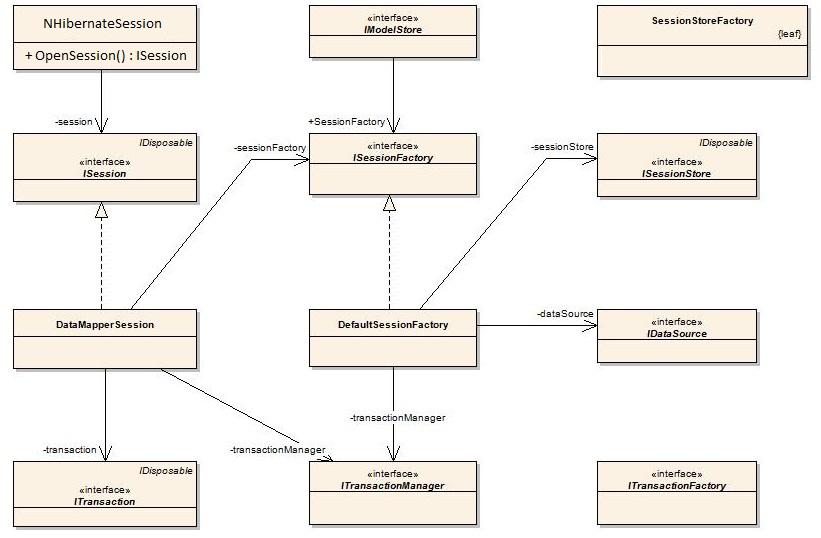
Osim razreda koji definiraju tablice u bazi podataka također imamo klasu *Repository* koja implementira sučelje *IRepository.* Klasa je generička te služi kao klasa koja služi za implementiranje obrasca repozitorij koji se koristi za operacije koje komuniciraju s bazom podataka. Više o njima u sljedećem potpoglavlju.

3.2 Opis implementacije perzistencije

Perzistencija je implementirana pomoću obrazaca repozitorij i unit of work, uz pomoć NHibernate razvojnog okvira koji služi za objektno-relacijsko preslikavanje. Unit of work obrazac je ustvari implementiran unutar NHibernate razvojnog okvira, taj obrazac služi za održavanje update-a u memoriji te da se te promijene iz memorije izvrše prema bazi podataka u obliku jedne transakcije.

Obrazac repozitorij je implementiran pomoću sučelja *IRepository* i njegove implementacije *Repository.* Klasa i sučelje su generički te je za svaki klasu korišten objekt iz klase Repository, ali sa generičkom vrijednosti te klase. Izdvojio sam sučelje u slučaju da se nekada odlučimo kreirati posebno sučelje za neku klasu, ali u ovoj domeni smo uvijek koristili Repository za tipom klase koju želimo dohvaćati iz baze podatka. Svaki kontroler ima repozitorij vezan uz klasu za čije objekte je on zadužen. Repository ima metode koje koristimo u svim slučajevima. Metoda *add* služi za dodavanje objekta u bazu podataka, *delete* služi za brisanje elemenata, *update* služi za uređivanje postojećih objekata u bazi podataka. Ostatak metoda služi za dohvaćanje vrijednosti iz baze podataka. *GetById* dohvaća objekt sa Id-om koji je predan kao parametar, *FindAll* vraća sve objekte tipa kojeg je generik u klasi, *FindByCriterion*  se služi objektom iz klase DetachedCriteria koji se šalje kao parametar i pomoću kojega se mogu složiti složeni upiti. Metoda *FindBy* prima dva parametrima od kojih je prvi ime stupca u tablici, a drugi je vrijednost koju tražimo, može biti string ili broj.

NHibernate mapiranje smo definirali pomoću XML datoteka te je to detaljnije opisano u dokumentu za laboratorij profila. NHibernate kontrolira perzistenciju te održava unit of work obrazac. Za pokretanje NHibernate sesije se koristi klasa koja implementira sučelje *ISession*, a ta klasa se generira pomoću obrasca tvornice iz klase *ISessionFactory*.



**Slika 3.2 Dijagram razreda NHibernate sesije**

**4. Opis izgrađenih REST servisa**

Servisi su implementirani za potrebne mobilne aplikacije. Implementirani su uz pomoć ASP.Net Web Api 2 razvojnog okvira koji je pisan u programskom jeziku C#. Kao tip dokumenta koji servisi vraćaju korišten je JSON (JavaScript Object Notation), a za arhitekturu je odabrana REST-ful arhitektura koja se danas najčešće koristi.

4.1 Opis implementacije

Projekt se sastoji od modela koji su klase koje definiraju tablice u bazi podataka, ali postoje i klase koje nam služe samo kao pomoć za kreiranje nekih objekata koji se sastoje od više različitih objekata. Npr. postoji klasa FlightCreator koja umjesto RouteId vrijednosti sadrži dvije vrijednosti koje definiraju odlazni i dolazni aerodrom na letu. S obzirom da Flight klasa predstavlja jedan let ona sadrži rutu koja je sastavljena od odlaznog i dolaznog aerodroma, ali kada kreiramo let u POST zahtjevu nam je lakše poslati oznake tih aerodroma pa zbog toga postoji klasa FlightCreator. U slučaju da već postoji ta ruta s tim aerodromima u bazi podataka koristi se ta ruta, a ako ne postoji automatski se kreira nova ruta i kao RouteId na tom letu se postavlja novo kreirana ruta.

U projektu se također nalaze kontroleri koji služe za obradu zahtjeva te vraćanje rezultata, svaki kontroler ima asocijaciju na repozitorij za objekte koje vraća. Kontroler vraća odgovore sa kodovima koji su standardizirani. U slučaju da je zahtjev ispravan vraća se kod 200, a u slučaju grešaka vraćaju se kodovi u rasponu od 400-500, npr. 404 ako ruta ili traženi resurs ne postoji, 500 se ne bi smio vraćati, ali je moguće da neka grešku nismo predvidjeli pa će se u tom slučaju vratiti kod 500. Uz kodove se naravno vraćaju podaci koji su zatraženi, npr. U slučaju da je zahtjev ispravan te tražimo sve objekte neke klase tada se oni vraćaju u obliku objekta koji implementira sučelje IEnumerable<>, a ako dohvaćamo neki objekt prema primarnom ključu tada se vraća objekt klase koju pretražujemo.

U projektu se također nalaze konfiguracije za NHibernate koje su opisane u laboratoriju profila, te konfiguracije samog .Net razvojnog okvira.

Testirali smo jedan kontroler pomoću Mock repozitorija te smo testirali model domene na primjeru jedne klase. Za mocking smo koristili biblioteku *Moq*, a za kreiranje unit testova smo koristili biblioteku *Microsoft.VisualStudio.TestTools.UnitTesting.* Testirali smo kontroler koji služi za kontrolu putnika, preciznije *PassengerControler.* Testirali smo sve metode u kontroleru te sve validacije koje se vrše unutar kontrolera. Testove modela domene smo vršili nad klasom *Ticket* koja predstavlja sve prodane karte. U njoj se nalaze metode za validaciju pri kreiranju objekta, te smo također provjerili sve getter i setter funkcije.

[TestMethod]

public void PassengerControllerPostValidationsEmailUnique()

{

PassengerController pc = new PassengerController(repo, repoCountry, repoTicket);

//Email vec postoji

IHttpActionResult actionResult = pc.Post(new Passenger { Email = "mocking@mocking.com", CountryId = 1, Identifier = "01923", Name = "Đuro" });

Assert.IsTrue(repo.FindAll().Count() == 3);

Assert.IsInstanceOfType(actionResult, typeof(BadRequestErrorMessageResult));

var contentResult = actionResult as BadRequestErrorMessageResult;

Assert.IsTrue(contentResult.Message == "User with this email already exists");

}

**Isječak koda 4.1 - metoda koja testira PassengerControlle**r

U isječku koda 4.1 nalazi se kod pomoću kojeg smo testirali da li kontroler vraća ispravan odgovor ukoliko kreiramo putnika koji ima isti email kao neki drugi korisnik, jer smo postavili da email putnika mora biti jedinstven. U Assertu *IsIstanceOfType* provjeravamo da li je kod greške jednak onom koji se šalje u slučaju lošeg zahtjeva *BadRequest.* Također provjeravamo da li je poruka greške ispravna.

4.2 Objavljivanje web servisa

Web servise i bazu podataka smo objavili na Microsoft cloud servisu Azure kako bismo olakšali izradu mobilne aplikacije te kako bismo osigurali jednaku bazu podataka svim korisnicima. Url web servisa se nalazi na web lokaciji koja nam je dana od strane azura <http://flightcontrolapi20180119015256.azurewebsites.net>.

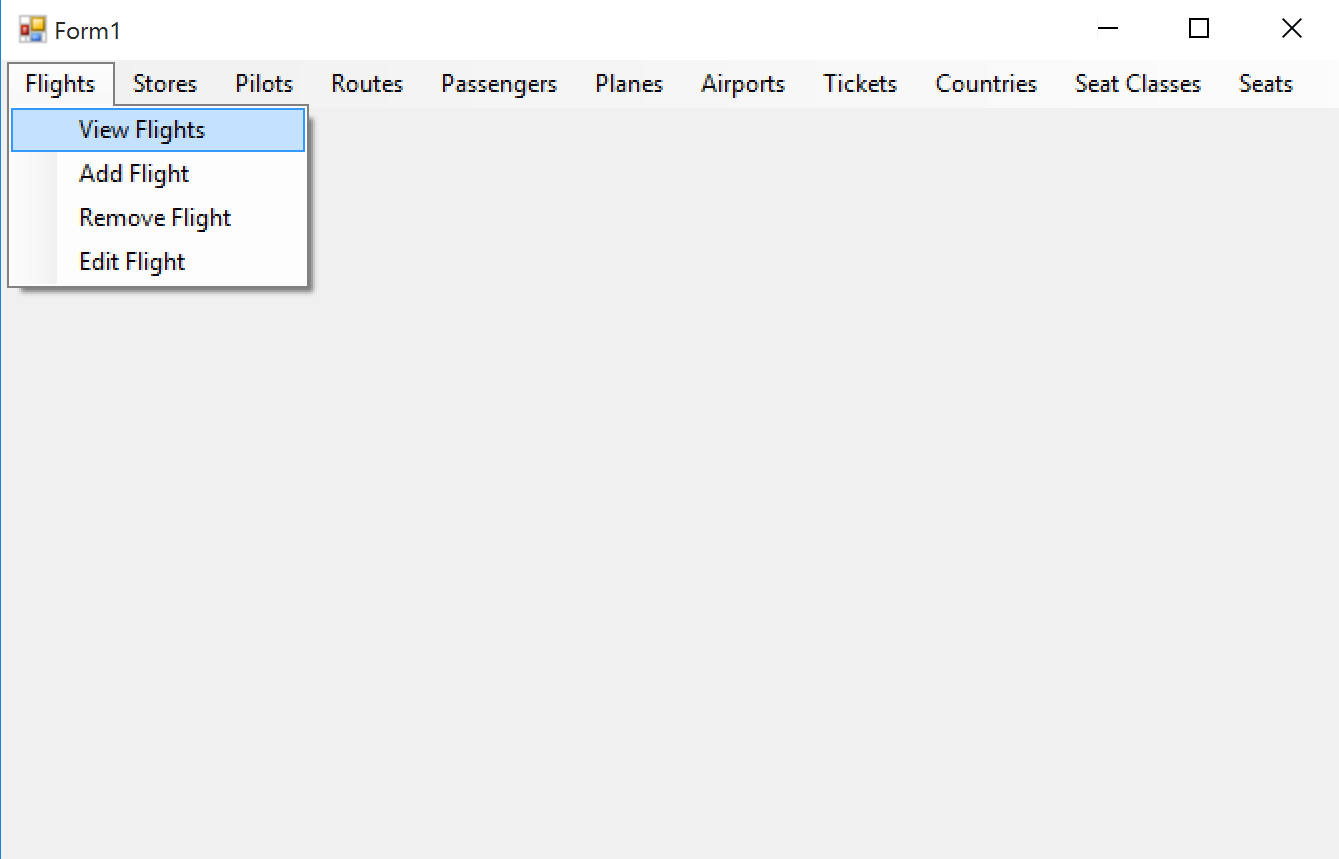
4.3 API dokumentacija

* Pregled karata određenog leta > GET /ticket
* Dodavanje nove karte > POST /ticket
* Uređivanje podataka karte > PUT /ticket/{id}
* Brisanje karte > DELETE /ticket/{id}
* Pregled poslovnica kompanije > GET /store
* Dodavanje nove poslovnice > POST /store
* Uređivanje podataka poslovnice > PUT /store/{id}
* Brisanje poslovnice > DELETE /store/{id}
* Pregled putnika određenog leta > GET /passenger/flight/{id}
* Dodavanje novog putnika > POST /passenger
* Uređivanje putnika > PUT /passenger{id}
* Brisanje putnika > DELETE /passenger/{id}
* Pregled pilota > GET /pilot
* Dodavanje novog pilota > POST /pilot
* Uređivanje pilota > PUT /pilot/{id}
* Brisanje pilota > DELETE /pilot/{id}
* Pregled aviona > GET /airplane
* Dodavanje novog aviona > POST /airplane
* Uređivanje aviona > PUT /airplane/{id}
* Brisanje aviona > DELETE /airplane/{id}
* Postavljanje aviona u status “Pokvaren” > PUT /airplane/broken/{id}
* Postavljanje aviona u status “Ispravan” > PUT /airplane/fixed/{id}
* Pregled aerodroma > GET /airport
* Dodavanje novog aerodroma > POST /airport
* Uređivanje aerodroma > PUT /airport/{id}
* Brisanje aerodroma > DELETE /airport/{id}
* Postavljanje aerodroma u status “Neaktivan” > PUT /airport/inactive/{id}
* Pregled ruta > GET /rute
* Dodavanje ruta > POST /rute
* Uređivanje ruta > PUT /rute
* Brisanje ruta > DELETE /rute
* Pregled letova > GET /flight
* Dodavanje leta > POST /flight
* Uređivanje leta > PUT /flight/{id}
* Brisanje leta > DELETE /flight/{id}
* Otkazivanje leta > PUT /flight/canceled/{id}

**5. Opis izgrađene desktop aplikacije**

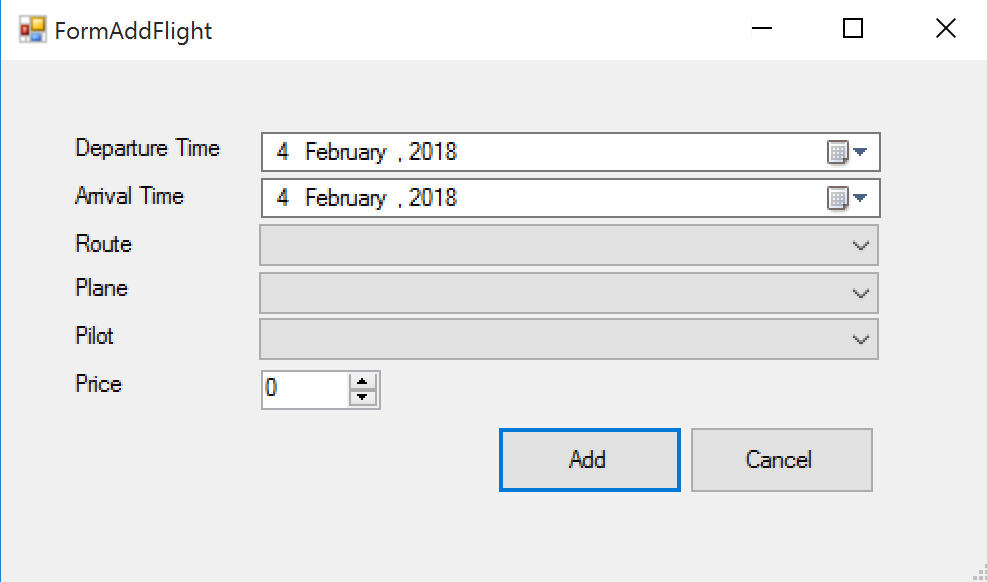
5.1. Upute za korisnika

Desktop aplikacija izgrađena je koristeći Windows Forms. Aplikacija omogućuje korisniku sve potrebne akcije nad podacima. Pri pokretanju aplikacije otvara se prozor s glavnim izbornikom gdje je korisniku ponuđen pregled svih entiteta, te u padajućem izborniku korisnik može odabrati koju akciju želi napraviti nad njime.

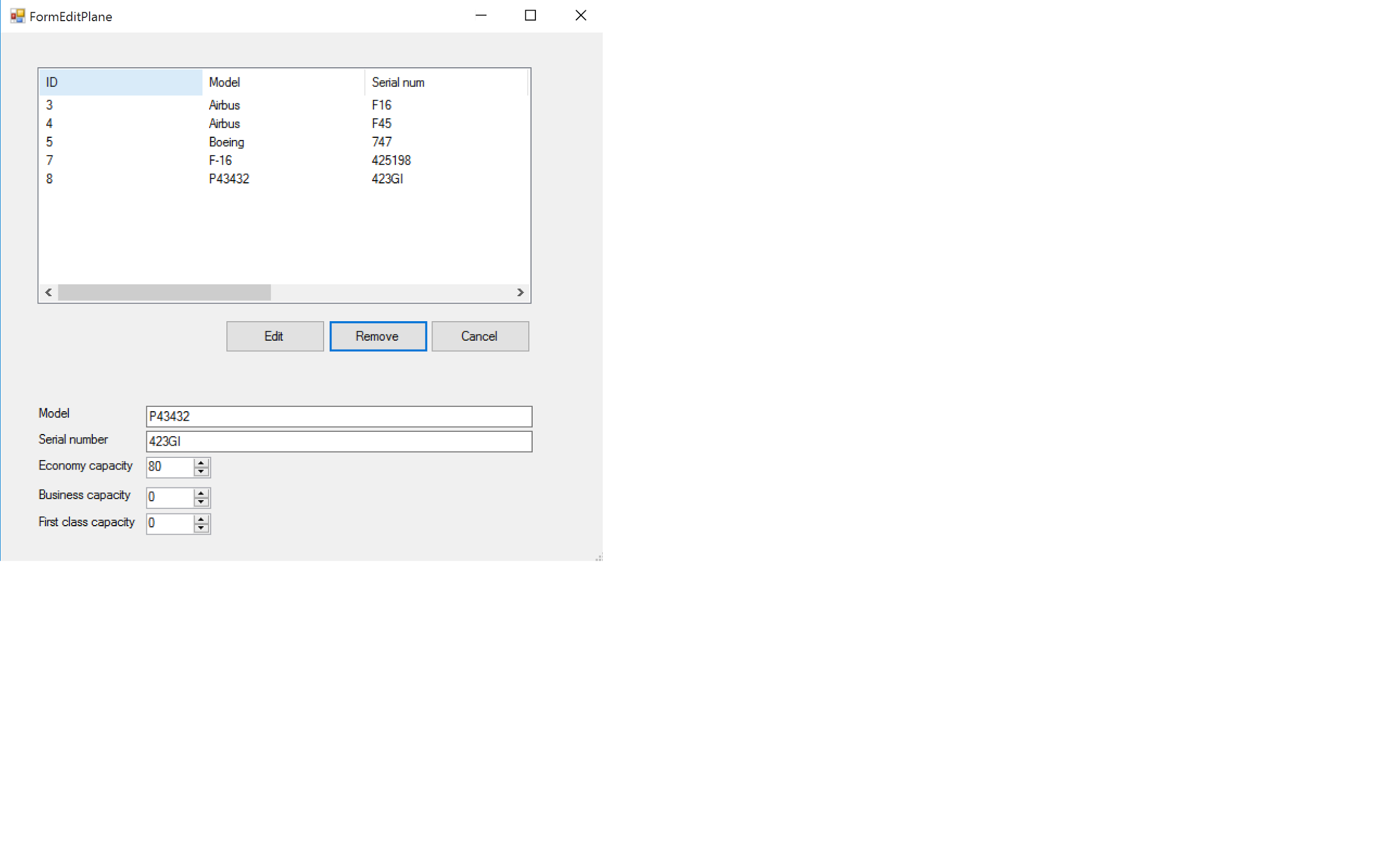


Za svaki entitet postoje dvije vrste pregleda: dodaj novi i uredi. Prozor za uređivanje se sastoji od pregleda podataka s njihovim detaljima, mogućnosti brisanja ili deaktiviranja pojedinog podatka, ovisno o prirodi entiteta, te forme za uređivanje, odnosno izmjenu podataka.

Odabirom dodavanja novog podatka otvara se prva vrsta prozora. Ovdje se korisniku prikazuje forma za dodavanje novog podatka, pri čemu se podaci mogu unositi tekstualno, numerički, odabirom u kalendaru, iz padajućeg izbornika i slično. Nakon unosa korisnik odabire ‘Add’ i sustav mu ispisuje poruku o uspješnosti unosa.



Odabirom pregleda podataka otvara se drugi prozor gdje korisnik može pregledavati detalje, uređivati podatke te brisati jedinke. Odabirom na pojedini redak forma se automatski ispunjava postojećim detaljima te korisnik može izmjeniti bilo koji podatak po po potrebi i zaključati izmjenu.



5.2. Struktura desktop aplikacije

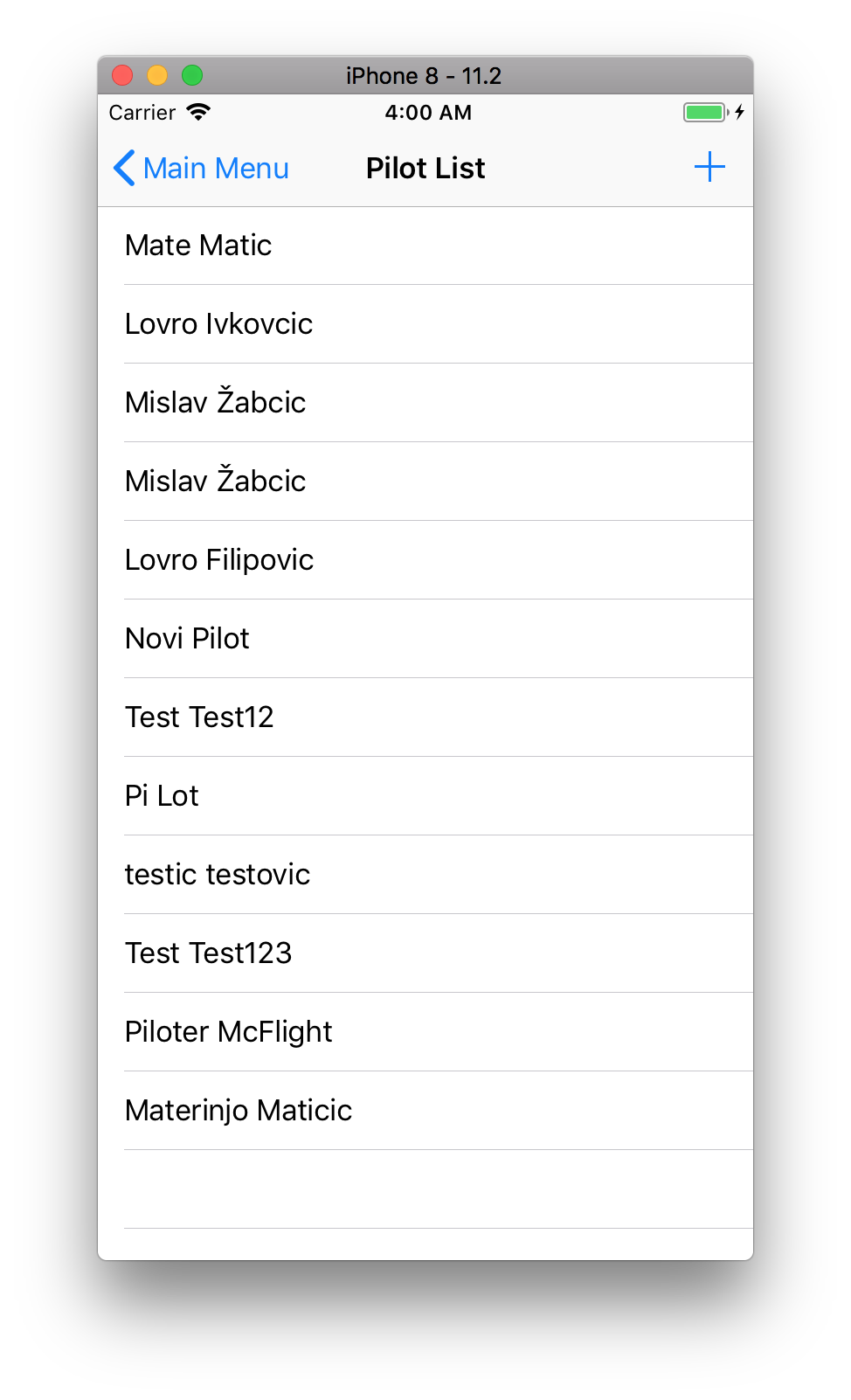
Desktop aplikacija izgrađena je na principu MVP obrasca. Aplikacija se strukturalno sastoji od AppStarter projekta, Controllera, prezentacijskog sloja s formama, Modela podataka i testova.

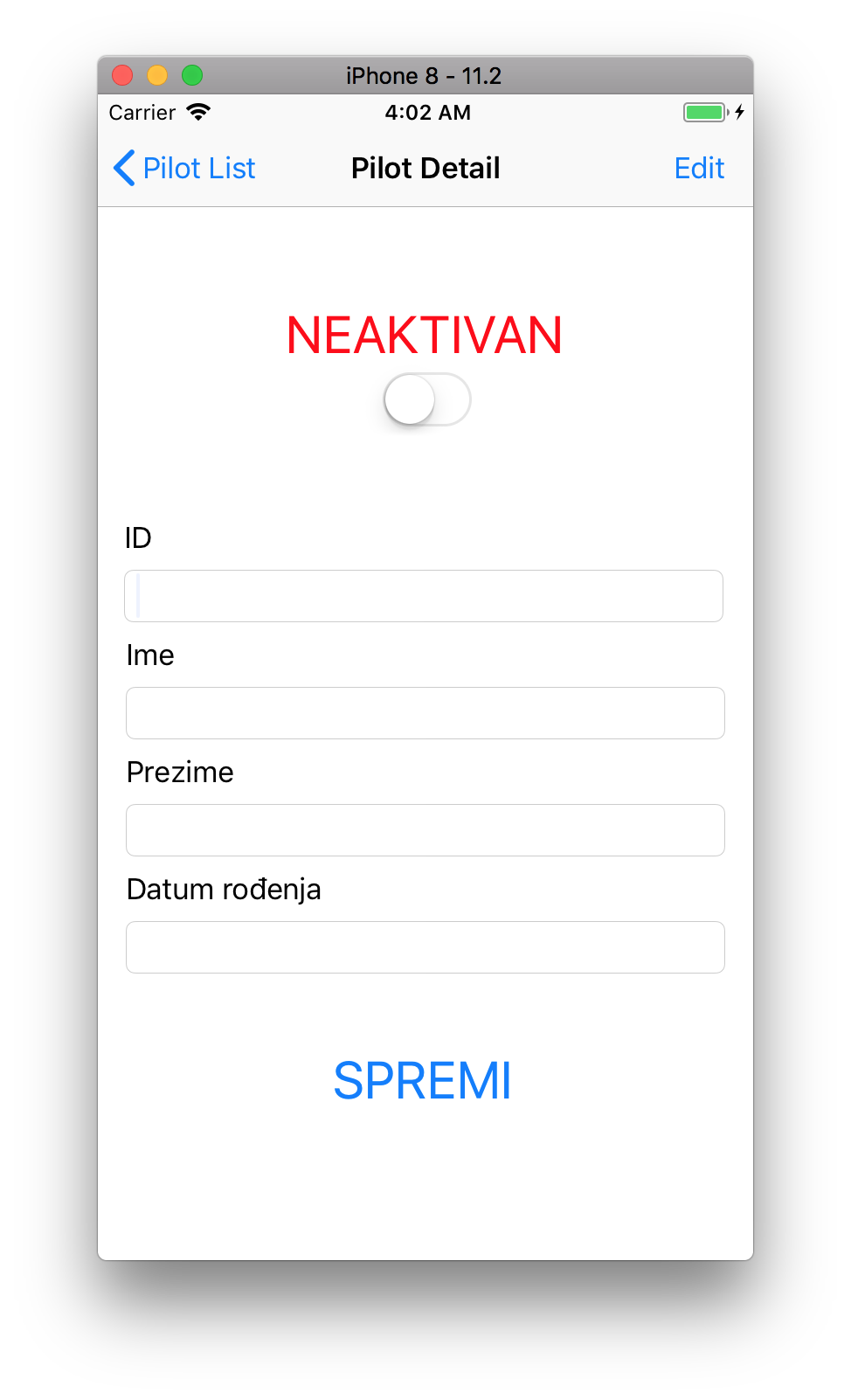
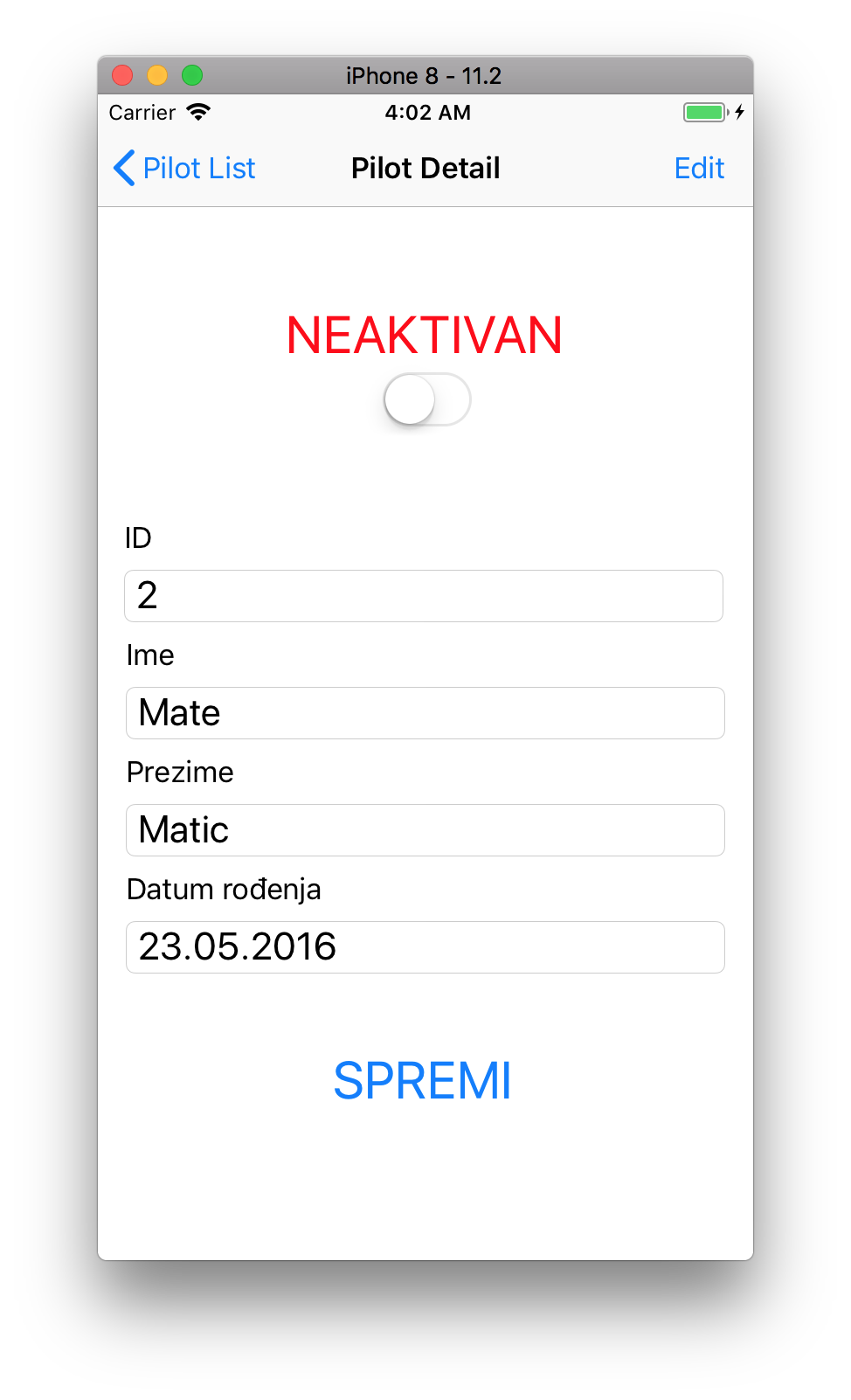
App starter spaja komponente i pokreće aplikaciju, odnosno generira main formu projekta koja se prikazuje korisniku. Presentation layer sastoji se glavne forme, forma za uređivanje i forma za dodavanje novih podataka. Kontroleri se brinu za generiranje određenih formi te prosljeđivanje podataka. Model se sastoji od repozitorija, entiteta i servisa. Ukratko, repozitoriji sadrže CRUD metode za operiranje s podacima preko baze podataka, dok servisi instanciraju repozitorije. Entiteti prikazuju preslikane objekte iz baze s pripadajućim atributima i međusobnim vezama. Repozitoriji implementiraju kreiranu apstraktnu metodu koja se spaja na bazu podataka koristeći connection-string. Svaki pojedini repozitorij koristeći IDbConnection i IDbCommand sučelja za manipulaciju podacima koristeći SQL naredbe.

**6. Opis izgrađene mobilne aplikacija**

5.1 Upute za korištenje

Aplikacija je izgrađena korištenjem XCode 9 razvojnog okruženja u jeziku objective-c.

Interface je izgrađen korištenjem storyboardova. Pri pokretanju aplikacije prikazuje se glavni izbornik u kojem se nalazi popis svih kategorija nad kojima se mogu obavljati operacije (piloti, karte, aerodromi, države, putnici…). Odabirom neke kategorije, otvara se novi ekran u kojem je popis svih entiteta koji se nalaze u toj kategoriji (npr. popis svih pilota). Korisnik tada ima mogućnost dodavanja novog entiteta odabirom pripadajućeg buttona, brisanje negog entiteta swipeanjem ili dodirom na naziv odlazak na novi ekran sa detaljima. Na tom ekranu je moguće urediti entited odabirom buttona “uredi”.



5.2 Struktura mobilne aplikacije

Aplikacije je izgrađena korištenjem verzije MVC obrasca - Model - View - ViewController. Komunikacija sa backendom je izvedena preko REST sučelja uz pomoć RestKit libraryja koji se koristi za parsiranje JSON-a i mapiranje na objekte, te AFNetwokring librarija za komunikaciju preko HTTP-a. Pri samom pokretanju aplikacije poziva se AppDelegate u kojem se provode početne inicijalizacije objekata koji će biti zajednički za ostatak aplikacije. U ovom slučaju to je inicijalizacija HTTP klijenta sa baznom adresom <http://flightcontrolapi20180119015256.azurewebsites.net/> na koju će se nadodavati path ovisno o pojedinoj akciji. Također se inicijalizira singleton objekt RKObjectManager koji kao parametar prihvaća gornji HTTP klijent, te uz pomoću kojeg će se obavljati sva parsiranja i mapiranja JSON objekata.

Modeli se sastoje od samih entiteta koji ujedno sadrže metode za mapiranje JSON-a, te service klasa za spajanje na backend, formatiranje podataka, mapiranje itd. Viewovi su u suštini storyboardovi, te samo sadrže forme za prikaz i uređivanje podataka. U slučaju iOS sprega između Viewa i ViewControllera je iznimno velika što uzrokuje masivne ViewControllere. U njima se obavlja inicijalizacija formi, njihovo pozicioniranje, te odgovaranje na korisničke akcije.

**7. Podjela posla**

|  |  |
| --- | --- |
| Mislav Žabčić | Baza podataka i API |
| Domagoj Grizelj | iOS aplikacija |
| Natko Bišćan | Desktop aplikacija |