Podaci sa kojima se radi su:

* Korisnici
  + Klijenti
  + Agenti
* Države
* Mesta
* Adrese
* Aranžmani
* Smeštaji
  + Hoteli
  + Apartmani
* Prevozna sredstva
  + Avioni
  + Vozovi
* Paketi

Na ovaj način se može videti i upravljati u koje države se može otputavati, u koja mesta u tim državama, na koje adrese, u koje hotele ili apartmane, kojim prevoznim sredstvom, aranžmane dostupne za dodavanje u pakete, kao i zakupljene i slobodne pakete.

Omogućena je i registracija novih korisnika.

Svi podaci se čuvaju u vidu objekata upotrebom serijalizacije u .xml fajlovima. Prilikom startovanja aplikacije program učitava sve podatke iz fajlova. Ukoliko ih ne pronđe, kreira nove.

# Uvod

Aplikacija predstavlja uprošćeni sistem turističke agencije i opisana je sledećim podacima:

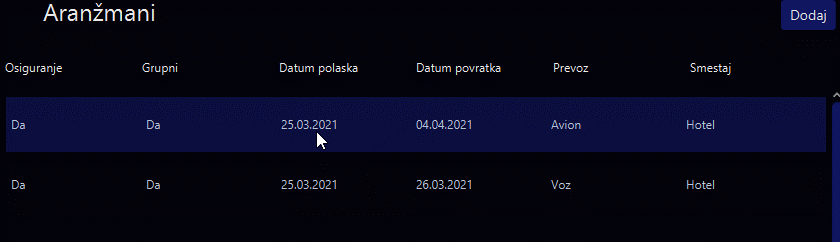
1. **Turistička agencija** je opisana nazivom i kolekcijom klijenata i kolekcijom agenata.
2. **Korisnik** je apstrakcija za agenta i klijenta i opisuju je ime, prezime, korisničko ime, lozinka, tip, prijavljen korisnik i svi korisnici.
3. **Agent** je korisnik koji izvršava specijalne fukcionalnosti ali nema dodatnih atributa u odnosu na korisnika kojeg nasleđuje.
4. **Klijent** je korisnik koji je još opisan i raspoloživim (novčanim) stanjem.
5. **Smeštaj** je apstrakcija za hotel i apartman, a opisan je površinom, brojem kreveta, tipom pansiona i adresom.
6. **Tip pansiona** je enumeracija vrednosti: noćenje, noćenje sa doručkom, polupansion, pun pansion, all inclusive (sveobuhvatni pansion).
7. **Prevozno sredstvo** je apstrakcija za avion i voz, a opisan je podatkom da li uključuje obrok ili ne, polaznom i odredišnom adresom.
8. **Hotel** je smeštaj koji je opisan i brojem zvezdica.
9. **Apartman** je smeštaj koji ne dodaje dodatna polja.
10. **Avion** je prevozno sredstvo koje ima klasu avionskog mesta.
11. **Voz** je prevozno sredstvo koje ima klasu voznog mesta.
12. **Klasa avionskog mesta** je enumeracija sa vrednostima: biznis i ekonomska.
13. **Klasa voznog mesta** je enumeracija sa vrednostima: prvi razred, drugi razred, spavaća kola.
14. **Adresa** je opisana ulicom, brojem i mestom.
15. **Mesto** je opisano državom.
16. **Država** je opisana imenom i kolekcijom mesta koja se u njoj nalaze.
17. **Aranžman** je opisan podatkom da li je sa osiguranjem ili bez, da li je grupni, datumom polaska, datumom povratka, prevoznim sredstvom i smeštajem.
18. **Paket** je opisan procentualnom vrednošću povrata novca u slučaju okazivanja, aranžmanom, korisnikom koji je kreirao paket i klijentom koji je zakupio paket (opciono).
19. **Zakupljivo** je interfejs koji implementiraju klase paket, smeštaj i prevozno sredstvo. Zakupljivo u klasama paket, smeštaj i prevozno sredstvo dodaje metodu zakupi. Metoda zakupi treba da izračuna vrednost zakupa tog elementa, te tako za zakup smeštaja se vrednost računa kao: broj\_dana ∗ tip\_pansiona ∗ VREDNOST, gde je vrednost: 1200 za noćenje, 1700 za noćenje sa doručkom, 2100 za polupansion, 2800 za pun pansion i 3600 za all inclusive (sveobuhvatni pansion). Ako je u pitanju zakupljivanje hotela na prethodnu vrednost se dodaje još i 123 ∗ broj\_zvezdica. Metoda zakupi za prevozno sredstvo se računa kao 540 ∗ ukljucuje\_obrok + 640. U slučaju aviona na cenu zakupa se dodaje i vrednost spram tipa (za biznis još 1000, a za ekonomsku 250). U slučaju voza na cenu zakupa se dodaje i 210 u slučaju prvog razreda ili 1000 u slučaju spavaćih kola. Metoda zakupi u klasi paket se računa kao suma svih aranžmana (njihovih smeštaja i prevoza). Rezultati ovih metoda su izračunate vrednosti.
20. **IO** je interfejs koji implementiraju klase Adresa, apartman, aranyman, avion, dryava, hotel, korisnik, mesto, paket, voz, agent, klijent, mesto. IO u ovim klasama. Dodaje metode sacuvajIzmene i upisi. Metoda upisi pri kreiranju novog objekta klase, objekat upisuje u listu svih objekata te klase, a zatim u odgovarajući fajl. Metoda sačuvajIzmene pri izmeni nekog od objekata, ažurira odgovarajući fajl.
21. **Controller** je klasa zadužena za upravljanje glavnim prozorom, gde se izvršava većina funkcionalnosti programa.
22. **LoginController** je klasa zadužena za upravljanje sistemima za registraciju i logovanje korisnika. Postojeći korisnik se može ulogovati, nakon čega će ga dočekati svi njegovi podaci, koji se učitavaju pri pokretanju programa. Ukoliko korisnik nema nalog, može ga kreirati. Dodavanje novog agenta vrši se unosom administratorskog koda pri registracija. Administratorski kod je "agent".
23. **ItemController** je klasa zadužena za upravljanje stavkama.
24. **DrzaveController** je klasa zadužena za upravljanje mestima, državama i adresama.

Aplikacija omogućuje sledeće funkcionalnosti:

1. **Prijava korisnika** se vrši na početku rada aplikacije unosom korisničkog imena i lozinke, na osnovu tih podataka, se vrši provera u postojećim korisnicima iz datoteke i ukoliko se poklapaju podaci korisnik je uspešno prijavljen. U zavisnosti od tipa korisnika, nude mu se odgovarajuće funkcionalnosti u meniju.
2. **Odjava korisnika** vrši je prethodno prijavljen korisnik. Odjavom se napušta aplikacija.
3. **Uplatu novca** vrši klijent tako što unosi pozitivan broj kojim uvećava svoje raspoloživo stanje.
4. **Kreiranje aranžmana** vrši agent unošenjem podataka koji opisuju aranžman. Prilikom biranja smeštaja i prevoznog sredstva omogućen je izbor spram konkretnih tipova (npr. da li hoće hotel ili apartman i slično za prevozno sredstvo), realizovan kroz podmenije. Nakon odabera tipa prevoza ili smeštaja, klikom na dugme za dodavanje novog prevoza/smeštaja, prikazuju se sami oni tipovi koje je korisnik odabrao.
5. **Kreiranje paketa** vrši agent tako što bira prethodno kreiran aranžman. Kao kreator paketa se postavlja agent koji ga je kreirao. Agent može uneti povrat novca u slučaju otkazivanja paketa.
6. **Kreiranje paketa** vrši i klijent, tako što mu se nudi da on sam izabere aranžman za svoj paket, dakle slično kao što radi i agent. Kreirani paketi po želji klijenta, imaju fiksnu vrednost povrata novca u slučaju otkazivanja, koju klijent ne može da izmeni. Paket koji klijent kreira pokušava automatski da mu se dodeli, ukoliko ima dovoljno novca.
7. **Kreiranje države** vrši agent unosom imena države.
8. **Kreiranje mesta** vrši agent unosom imena mesta i odabirom države u kojoj se mesto nalazi iz liste postojećih država. Pri kreiranju mesta ono se dodaje u kolikciju svih mesta u državi koja je odabrana.
9. **Kreiranje adrese** vrši agent unosom ulice, broja, i odabirom mesta iz liste postojećih mesta.
10. **Kreiranje smeštaja** vrši agent unosom površine smeštaja, broja kreveta u smeštaju, tipa pansiona(iz liste postojećih tipova pansiona), tipa smeštaja, broja zvezdica ukoliko se radi o hotelu(iz liste postojećih). Na kraju iz menija sa svim dostupnim mestima, bira mesto u kome se smeštaj nalazi.
11. **Kreiranje prevoza** vrši agent unosom podatka da li prevoz uključuje obrok ili ne. Nakon toga, potrebno je odabrati tip prevoza, na osnovu kojeg će se prikazati odgovarajuće klase prevoza za odabir. Unose se i polazna i odredišna adresa iz menija za odabir, gde su prikazane do sada dodate adrese.
12. **Zakupljivanje paketa** vrši klijent tako što mu se nude prethodno kreirani paketi, koji nisu vezani ni za jednog klijenta (slobodni paketi). Klijent bira neki od ponuđenih paketa tako što klikće na njega u meniju za odabir (o paketu su prikazani podatci o adresama, vrednosti, datumu polaska, povratka, aranžmana). Prilikom zakupljivanja se računa ukupna vrednost paketa na osnovu sume aranžman, koja uključuje sumu vrednosti smeštaja i vrednosti prevoznog sredstva. Klijentu se prikazuje izračunata vrednost, a on potvrđuje ili odbija zakupljivanje paketa i to samo ako ima dovoljno raspoloživog novca za taj paket.
13. **Otkazivanje paketa** vrši klijent tako što mu se prikazuju svi paketi u kojima je on klijent. Klijent unosom rednog broja ispred izlistanog paketa bira koji želi da briše, pri tome mu se u raspoloživo stanje vraća onaj iznos koji se računa kao ukupan\_iznos\_aranzmana ∗ povrat\_u\_slucaju\_otkazivanja. Ttaj paket se briše iz klijentovih paketa, a paketu se briše klijent koji ga je rezervisao.

# Aplikacija

Da bi se kreirala aplikacija, potrebno je prvo kreirati projekat u programu *IntelliJ IDEA*. Potrebno je kreirati *JavaFX* projekat, čime se kreira i prozor *Main*. Kreiramo novi FXML fajl u SceneBuilderu, kako bi razdvojili deo za dizajn od logičkog dela aplikacije. Zatim je potrebno kreirati klasu zaduženu za upravljanje glavnim prozorom, koju ćemo nazvati Controller. U fajlu Home.fxml (glavni prozor) potrebno je vezati klasu koja će njime upravljati upotrebom svojstva fx:controller. Zatim sledi kreiranje svih ostalih prozora. Većina klasa(adresa, država...) imaju svoje "prozore" u .fxml fajlovima. Ti prozori se učitavaju u liste i menije svih dostupnih objekata. Na slici ispod, polje osenčeno plavom bojom predstavlja prozor klase aranžman, učitanog iz aranzmani.fxml.



Slika 1 - Prozori klase aranžman

## Čuvanje podataka

Svaka od klasa u sebi sadrži listu svih svojih instanci. Uzećemo za primer klasu Agent.

//region Polja  
protected static final long serialVersionUID = -1980325513659886298L;  
public static ArrayList<Agent> sve = TuristickaAgencija.getAgenti();  
public static String path = "podaci/agenti.xml";  
//endregion

Promenljiva path predstavlja putanju do fajla gde će se čuvati agenti.

Promenljiva serialVersionUID neophodja je za klase koje implementiraju interfejs java.io.Serializable, kako bi se obezbedio integritet objekata. Sve klase koje će se čuvati moraju implementirati ovaj interfejs, ili ga naslediti iz nadklase. Bitno je napomenuti da i podklasa mora imati svoj serialVersionUID, za koji se preporučuje ručna dodela vrednosti, a ne prepuštanje automatskoj generaciji.

Prilikom kreiranja novog objekta klase agent, u njegovom konstruktoru poziva se metoda upisi, koja novokreirani objekat dodaju u tu listu. Zatim se lista cuva u fajl.

public Agent(String ime, String prezime, String korisnickoIme, String lozinka) {  
 super();  
 this.ime = ime;  
 this.prezime = prezime;  
 this.korisnickoIme = korisnickoIme;  
 this.lozinka = lozinka;  
 this.tip = "Agent";  
 upisi();  
}

public void upisi(){  
 try {  
 ObjectOutputStream oos=new ObjectOutputStream(new FileOutputStream(path));  
 sve.add(this);  
 oos.writeObject(sve);  
 oos.close();  
 }  
 catch (IOException e) {  
 System.out.println("Fajl nije pronadjen. Bice kreiran novi.");  
 }  
}

Sada u fajlu imamo sačuvane sve agente, zajedno sa svim svojim pripadajućim objektima, i njihovim objektima i tako dalje. Ukoliko bi Agent sadržao podatak o Državi na primer, Država bi takođe morala da implementira interfejs java.io.Serializable, i da se ona ne čuva pojedinačno.

Prilikom izmene nekog od objekata, nad njime se može pozvati metoda sacuvajIzmene(), koja će listu kojoj on pripada da sačuva u odgovarajući fajl. Metoda sacuvajIzmene() funkcioniše po istom principu kao i upisi().

Ucitavanje fajlova se vrši statičkom metodom ucitaj(), za svaku klasu. Sadržaj fajla se kopira u listu svih objekata te klase. Metoda ucitaj je statička jer pre učitavanja ne postoji objekat nad kojim bi je pozvali. Zbog toga ona i nije uključena u IO interfejs.

Ovakav način čuvanja nam omogućuje upotreba referenci, bez nepotrebnog kreiranja novih objekata.

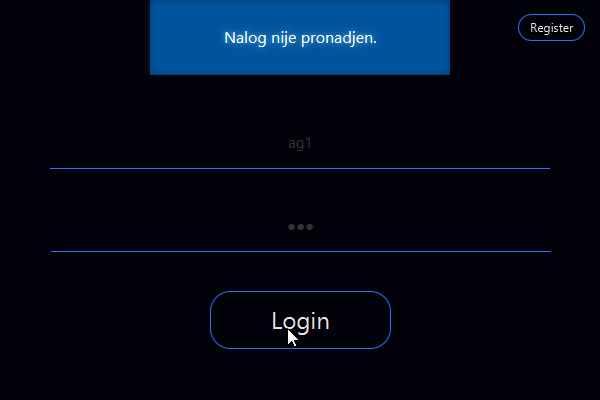
public static void ucitaj() {  
 try {  
 ObjectInputStream ois=new ObjectInputStream(new FileInputStream(path));  
 try {  
 sve =(ArrayList<Agent>) ois.readObject();  
 ois.close();  
 } catch (ClassNotFoundException e) {  
 System.out.println("Fajl nije pronadjen. Bice kreiran novi.");  
 }  
 }  
 catch (IOException e) {  
 System.out.println("Fajl nije pronadjen. Bice kreiran novi.");  
 }  
}

## Kreiranje klasa

Nakon što je obezbeđeno čuvanje podataka, potrebno je kreirati klase sa svim svojim atributima i metodoma. Svakoj klasi potrebno je dodati i gettere i settere, zbog pridržavanja principa o enkapsulaciji.

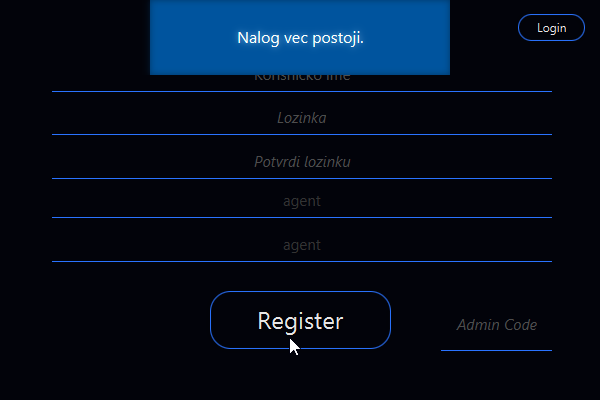
## Logovanje i registracija

Pri pokretanju programa pojavljuje se prozor za logovanje. ukoliko ne unesemo ispravno podatke, na ekran putem animacije uleće obaveštenje. Program poznaje i korišćenje dugmeta enter, kao i pritisak na dugme login.



Slika 2 - Logovanje, neuspeli pokušaj

U slučaju da nemamo nalog, pritiskom na dugme register prebacujemo se na prozor za registraciju u koji unosimo podatke. Ukoliko se lozinke ne poklapaju, ili unesemo već postojeće korisničko ime, prikazuje nam se poruka o grešci.



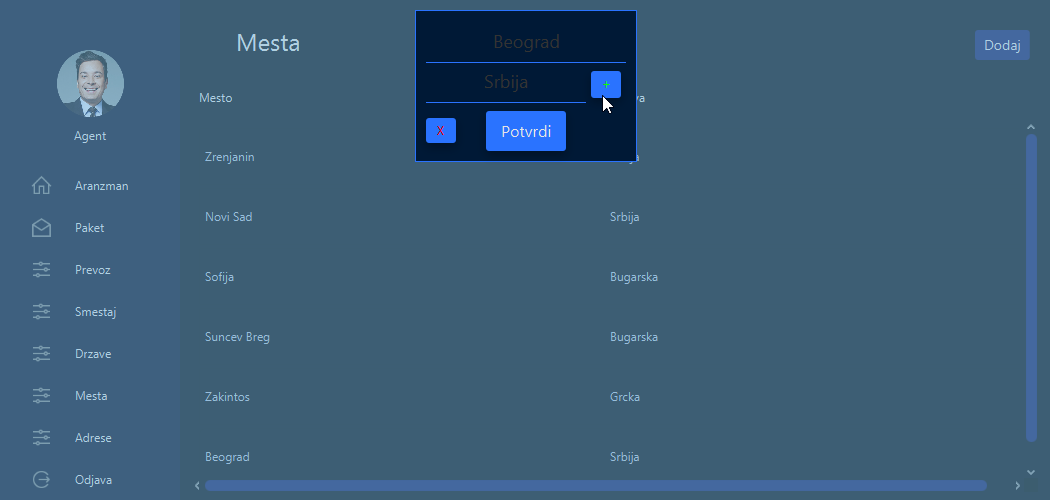
Slika 3 - Registracija, neuspeli pokušaj

Ukoliko kreiramo nalog bez upotrebe admin koda ili upotrebom neispravnog, on će automatski biti tipa klijent. Nakon uspešne registracije automatski ćemo biti prebačeni na prozor za logovanje. Postoji opcija i za povratak na ekran za logovanje ručno, pritiskom na malo login dugme u gornjem desnom uglu.Nakon logovanja, ispod slike biće prikazan tip korisnika koji je prijavljen, agent ili klijent.

## Agent

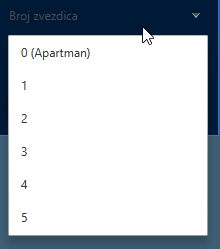
Mogućnosti agenta opisane su u uvodnom delu. Sledi prikaz izgleda njegovih menija.

U svakom od menija postoji prikaz Do sada dodatih stavki, kao i dugme za dodavanje novih. Prilikom pritiska na dugme za dodavanje, pojavljuje se novi prozor specifičan za tip objekta koji se dodaje. Prozor za dodavanje se može zatvoriti pritiskom na x u donjem levom uglu, ili ispravnim unosom podataka i pritiskom na dugme potvrdi. Dok god se to ne desi, pristup glavnom prozoru je onemogućen, što je korisniku signalizirano njegovim zamućivanjem i promenom kursora.

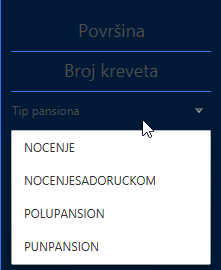


Slika 4 - Prozor za dodavanje novog mesta

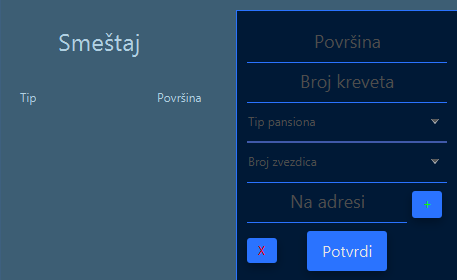
.Klikom na dugme za dodavanje novog objekta otvara se prozor za selektovanje, koji možemo zatvoriti pritiskom na taster escape. U prozor za selektovanje dinamički se učitavaju svi objekti koji mogu biti selektovani. Na primer sve države, svi paketi, svi slobodni paketi, svi paketi zakupljeni od strane ulogovanog klijenta... Klikom na jedan od ponuđenih izbora, on se pamti i upisuje u polje predviđeno za njega automatski. To polje moguće je promeniti samo ponovnim selektovanjem nekog drugog postojećeg izbora, nikako ručno i proizvoljno. U slucaju neispravnog unosa podataka, prikazuje se poruka.



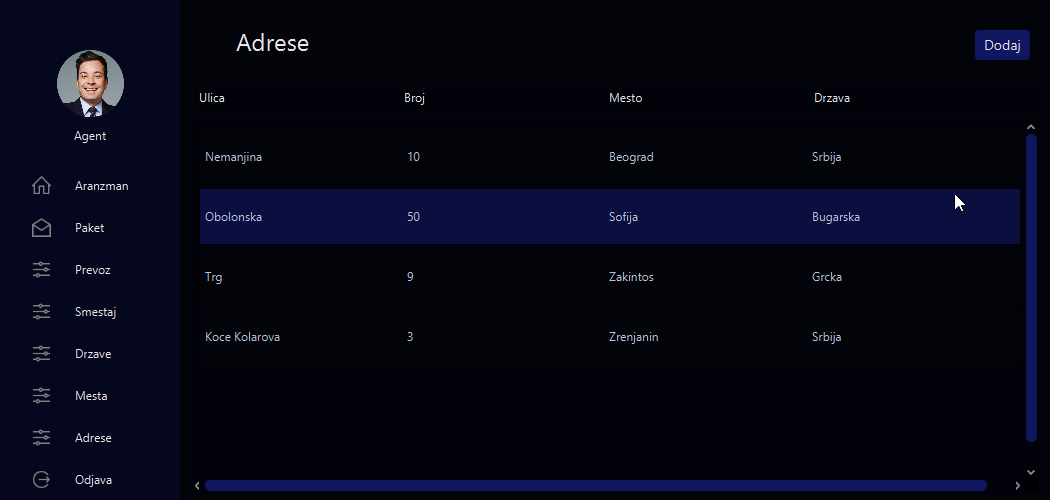
Slika 5 - Odabir broja zvezdica



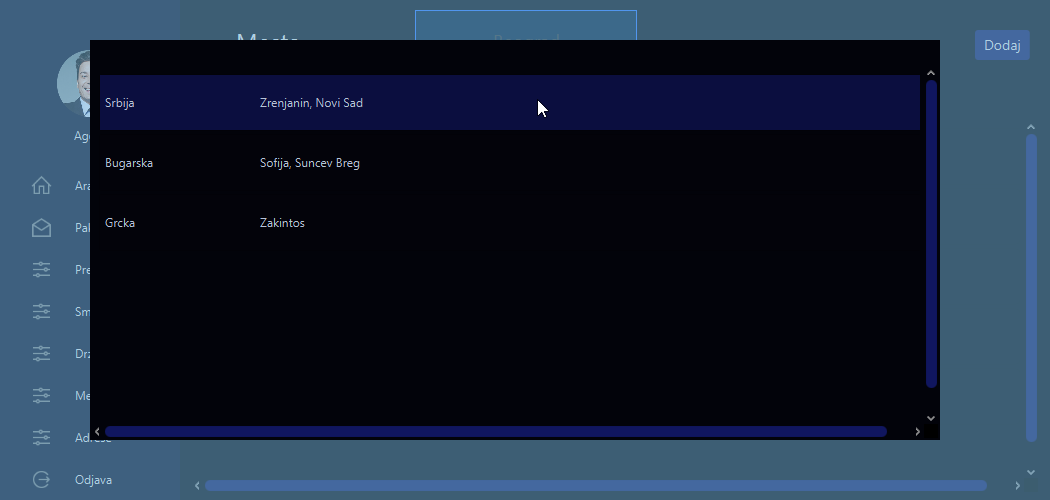
Slika 6 - Odabir pansiona



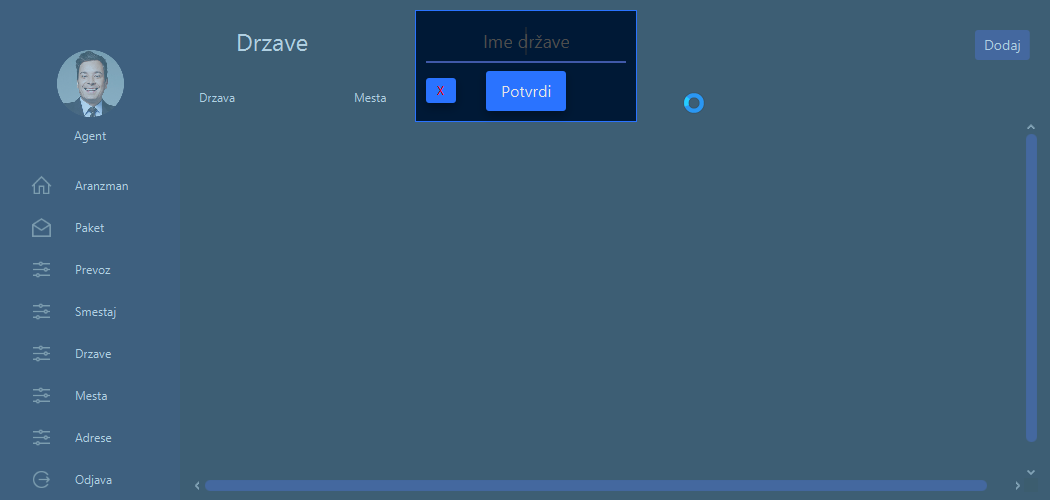
Slika 7 - Prozor za dodavanje smeštaja



Slika 8 - Prozor za prikaz adresa



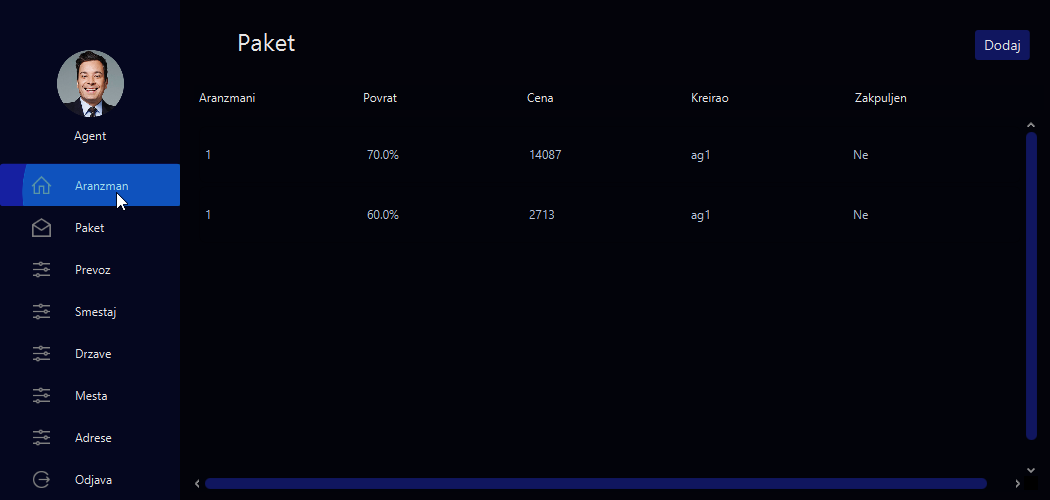
Slika 9 - Prozor za biranje postojece drzave, u kojoj se mesto nalazi



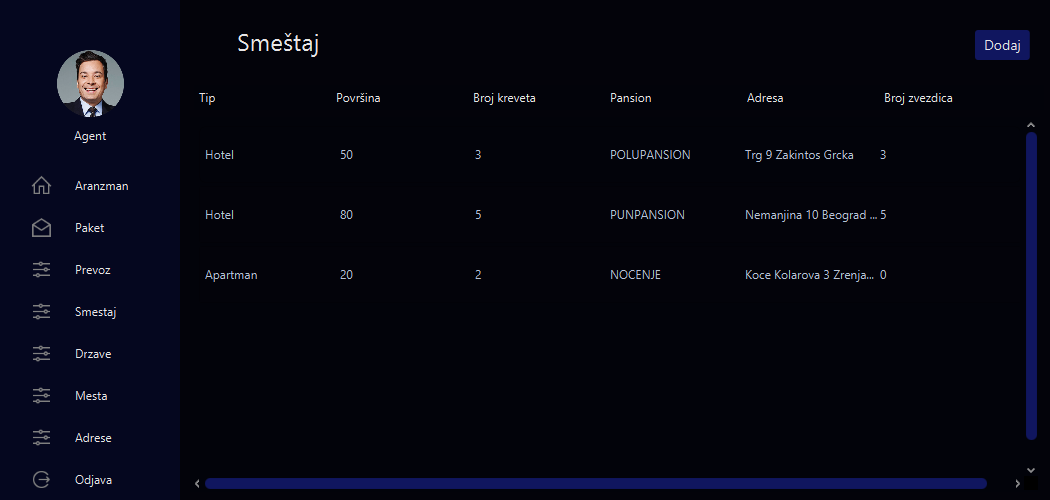
Slika 10 - Prozor za dodavanje države



Slika 11 - Prikaz aranžmana



Slika 12 - Prikaz paketa



Slika 13 - Prikaz smeštaja

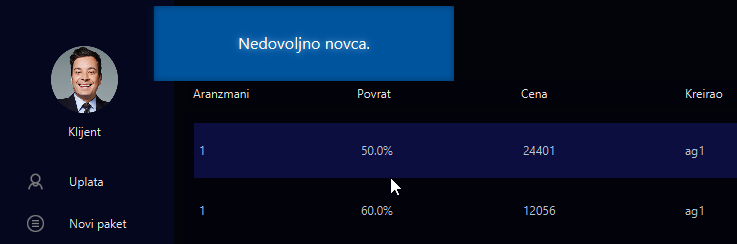
## Klijent

Mogućnosti klijenta opisani su u uvodnom delu. Klijent desnim klikom na neki od postojećih paketa, u zavisnosti od toga da li ga je zakupio ili ne, dobija odgovarajuću opciju - da taj paket zakupi ili otkaže.



Slika 14 - opcija za zakupljivanje paketa

Ukoliko klijent nema dovoljno novca na računu, neće mu biti dozvoljeno da zakupi paket.



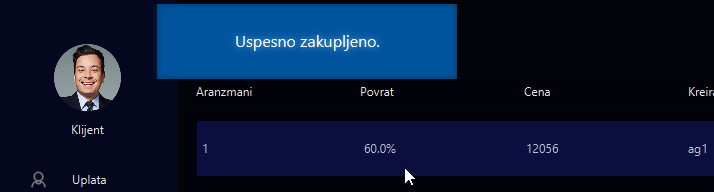
Slika 15 - Prikaz poruke o nedovoljnoj količini novca

Klijent ima mogućnost uplate novca na račun, koju ostvaruje putem prozora za uplatu.



Slika 16 - Prozor za uplatu

Ukoliko se sada vrati u prozor novi paketi, i ponovo pokuša da zakupi isti paket, dobija poruku o uspešnom zakupu. Paket se briše iz liste dostupnih paketa, jer je sada zakupljen od strane nekog klijenta i nije više dostupan.

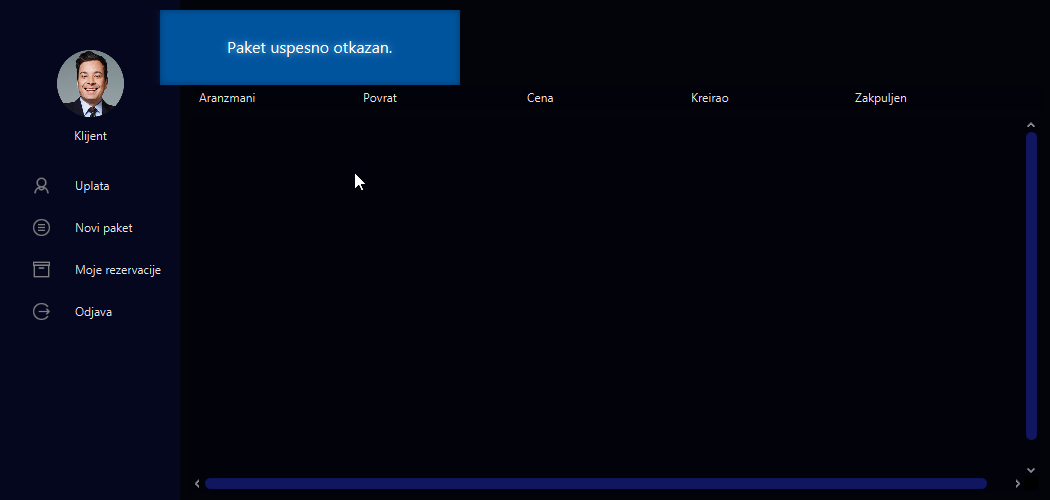


Slika 17 - Uspešno zakupljivanje paketa

Zakupljeni paket se sada nalazi u listi zakupljenih paketa za tog klijenta. On ga desnim klikom i odabirom opcije otkaži može otkazati, u kom slučaju dobija deo novca nazad.



Slika 18 - Prikaz opcije za otkazivanje zakupljenog paketa



Slika 19 - Prikaz poruke o ukljanjanju paketa, i njegovo uklanjanje iz klijentove liste

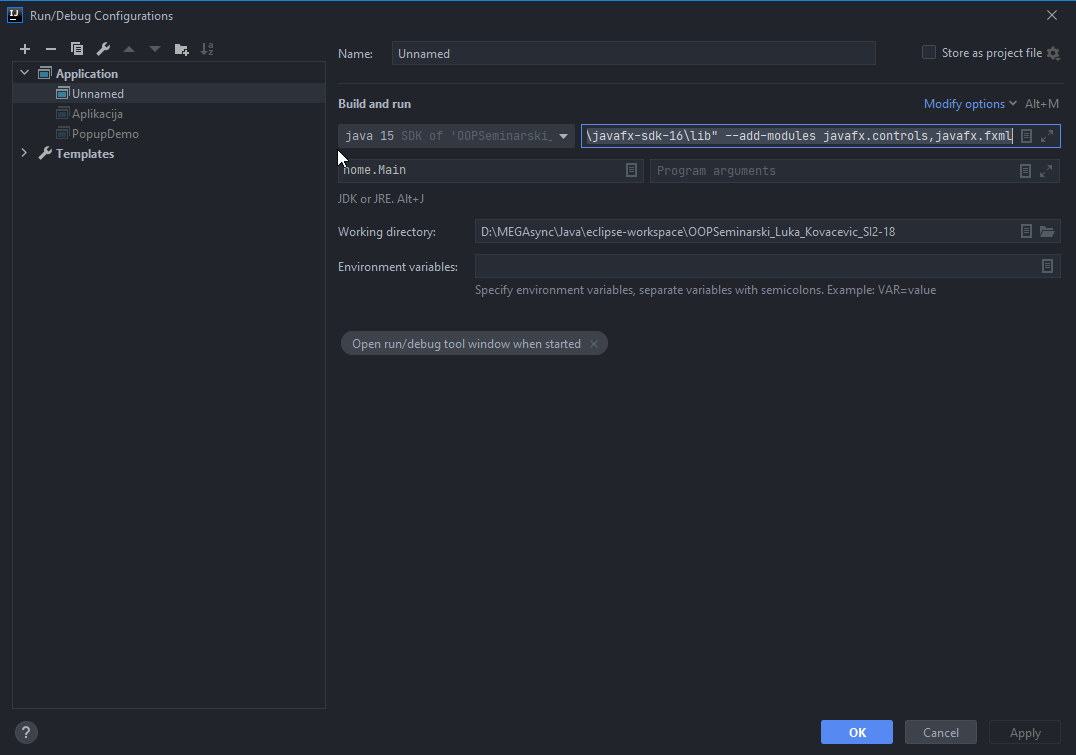
Klijent ja napočetku imao, nakon uplate, 25170 dinara. Nakon zakupa paketa koji košta 24401, i njegovim otkazivanjem za koje se dobija 50% cene (12200 dinara), dolazimo do trenutnog stanja na računu klijenta.



Slika 20 - Prikaz trenutnog stanja klijenta nakon zakupljivanja i otkazivanja

## Pokretanje

Projekat je rađen u JDK 15 verziji Jave, ali s obzirom da je u njegovu izradu uključena i JavaFX, potrebno je pre pokretanja dodati odredjene parametre za JVM u "run configuration" u razvojnom okruženju.

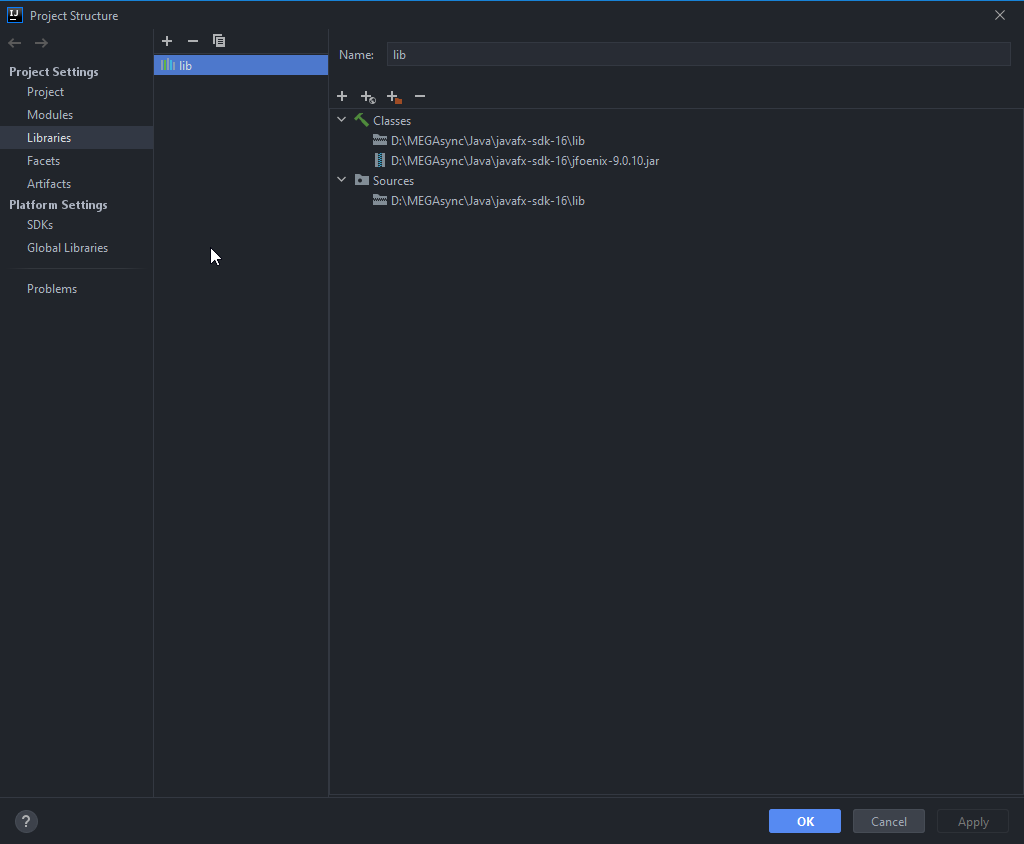


Slika 21 - run configuration

--module-path PUTANJA\_DO\_JAVAFX\_**LIB**\_FOLDERA --add-modules javafx.controls,javafx.fxml

Na primer:  
--module-path "D:\MEGAsync\Java\javafx-sdk-16\lib" --add-modules javafx.controls,javafx.fxml

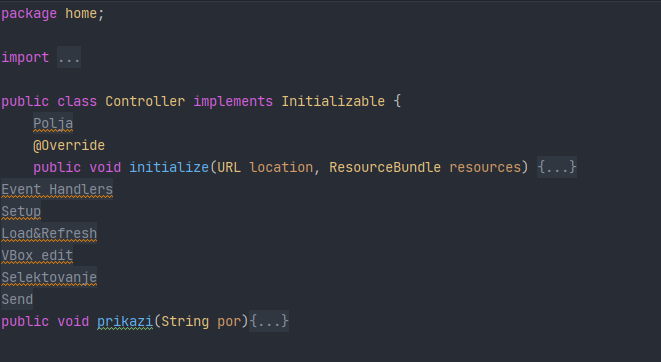
Takođe u projekat treba dodati sledeće biblioteke.



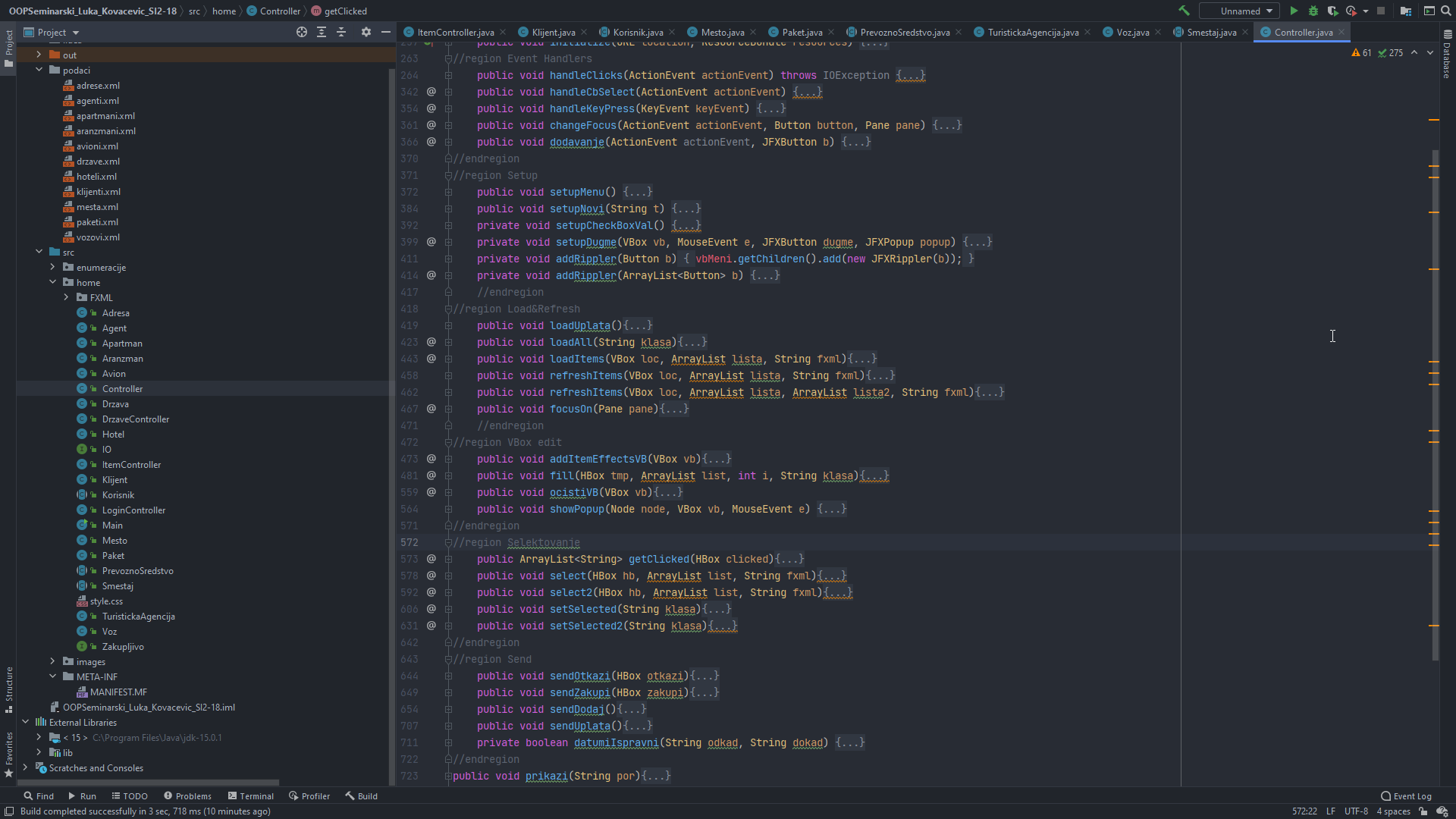
Slika 22 - Project structure, dodavanje biblioteka

Folder sa JavaFX bibliotekom će biti zakačen zajedno sa projektom. Sve gore navedeno je već podešeno u postojećem projektu i ne bi trebalo da mora opet da se podešava.

U projektu je urađeno formatiranje koda sa //region IME //endregion, prečica za skupljanje svih regiona radi bolje preglednosti je CTRL+ SHIFT + -.



Slika 23 - Prikaz skupljenog koda nakon korišćenja prečice



Slika 24 - Prikaz dela raširenih regiona