FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Seminar iz kolegija

**Objektno oblikovanje**

**Sustav za iznajmljivanje**

**Case study**

Vinko Kodžoman

0036470625

Uvod

Cilj seminara je prikazati tehnike i principe DDD-a (domein driven design), TDD-a (test driven design) i objektnog oblikovanja na praktićan način izradom case studya. Iz design storya je probreno specificirati i realizirati sve zahtjeve. Aplikacije se realizirao sa OR mapperom (Nhibernate), windows forms i MVC (model view controller) arhitekturom.

Design storry

Agencija za vođenje apartmana omogućuje reklamiranje apartmana za svoje klijente i rezerviranje istih za kupce. Agencija uzima proviziju za svaku uspješnu transakciju – najam apartmana. Jedine vrste nekretnina sa kojima agencija radi su apartmani, ali sustav mora dopuštati jednostavno proširenje sa novim stvarima (auti, bicikli, skuteri, hosteli, ...) u budućnosti. Svi apartmani u sustavu agencije su aktivni – mogu se iznamljivati, i u trenutku kada klijent agencije odlući prestati iznamljivati svoj apartman, oni se brišu iz sustava zajedno sa klijentom. Svaki klijent se uparuje sa jednim agentom agencije, koji je zadužen za klijenta kao i za njegove apartmane.

Za svaki apartman klijent mora dati sve potrebne informacije (broj soba, broj kupaonica, iznos po noći, lokacija, klima uređaj, ...), ukoliko su informacije nepotpune apartman se ne može pohraniti unutar sustava. Uz same karakteristke apartmana bitno je i da su navedene sve ponuđene usluge koje klijent nudi korisniku (kabelska televizija, internet, kućni telefon, ...). Svaki apartman mora imati informacije o prošlim korisnicima (ljudi koju su iznamljivali apartmani u prošlosti), kao što su podaci o osobi, iznos plaćanja i vrijeme boravka u apartmanu.

Za svaki apartman je naveden vlasnik kao i njegovi vlastiti podaci (ime, prezime). U slućaju da vlasnik više ne posluje sa agencijom treba izbrisati sve njegove apartmane unutar sustava.

Da bi dijelatnik agencije mogao pristupiti sustavu mora se prijaviti u njega sa svojim korisničkim imenom i lozinkom. Nakon ulaska u sustav prikazuju mu se sve ponude, vlasnici i povijest svih transakcija – rezervacija (prošlih, ali i trenutnih). Djelatnik agencije može saznati više o vlasniku kao i o apartmanu, odabirom njih unutar sustava. Djelatniku sustava (agentu) je omogućeno pretraživanje apartmana po njihovim imenima.

Sustav mora voditi evidenciju koji agenti agencije su zadužen za koje vlasnike. Sustav u niti jednom trenu ne smije imati informacije od ljudi koji iznamljuju apartmane i trenutno se nalaze u njima. Nakon što korisnik plati rezervaciju; bilježi se cijena rezervacije po danu, od kada do kada je rezerviran i koji apartman je rezerviran.

Use case

**Zahtjevi:**

1. Pregled ponuđenih apartmana
2. Pretraživanje apartmana
3. Dodavanje novog apartmana u sustav
4. Dodavanje novog agenta u sustav
5. Dodavanje novog klijenta u sustav
6. Evidentiranje zauzeća apartmana
7. Izmjena informacija postojećeg klijenta
8. Izmjena informacija agenta sustava
9. Uklanjanje apartmana iz sustava nakon odlaska vlasnika
10. Uklanjanje klijenta iz sustava
11. Uklanjanje agenta iz sustava

Dateljna razrada nekoliko najvažnijih use caseova:

**Use case 1**: Pregled ponuđenih apartmana

**Primarni aktor**: Zaposlenik agencije (agent)

**Stakeholderi**: Agent – cilj je dobiti preglednu ponudu apartmana

**Preconditions**: Agent se uspješno prijavio u sustav

**Postconditions**: Agentu su prikazani svi apartmani, koji odgovaraju kriteriju koje je zadao

**Glavni scenariji:**

1. Korisnik (agent) unosi ime apartmana po kojem želi pretražiti apartmane
2. Sustav prikazuje sve apartmane koji zadovoljavaju korisnikov kriterij pretrage
3. Korisnik pregledava sve informacije o apartmanu
4. Korisnik završava pregledavanja nekretnina u sustavu

**Alternativni scenariji:**

1a. Korisnik pretražuje bez specificiranje kriterija pretrage

1a1. Sustav prikazuje sve apartmane

1a1. Povratak na korak 3

**Use case 2**: Dodavanje novog apartmana u sustav

**Primarni aktor**: Zaposlenik agencije (agent)

**Stakeholderi**: Agent – cilj mu je imati više vlasnika za kojih je zadužen (provizija) i koji su zadovoljno kako bi generiraio veći promet agenciji i bio nagrađen

Klijent (vlasnik apartmana) – cilj mu je što brže i jednostavnije unjeti apartman u sustav

Agencija – cilj je imati više apartmana, kako bi imali veću zaradu od iznamljivanja

**Preconditions**: Agent se uspješno prijavio u sustav.

Vlasnik je poslao sve potrebne informacije za unos novog apartmana i svoje podatke.

**Postconditions**: Apartman je uspješno dodan u sustav i agent mu može pregledati informacije

**Glavni scenariji:**

1. Korisnik (agent) pokreće postupak za unos novog apartmana
2. Korisnik unosi sve potrebne podatke o apartmanu
3. Sustav provjerava jesu li svi podaci zadovoljeni
4. Agent odabire korisnika (vlasnika) kao vlasnika apartmana
5. Sustav pohranjuje podatke o apartmanu
6. Apartman je dodan u ponudu agencije

**Alternativni scenariji:**

4a. Vlasnik apartmana se ne nalazi u sustavu

4a1. Agent unosi sve podatke o vlasniku u sustav

4a2. Vlasnik je pohranjen u sustavu

4a3. Povratak na korak 4

**Use case 5**: Evidentiranje zauzeća apartmana

**Primarni aktor**: Zaposlenik agencije (agent)

**Stakeholderi**: Agent – cilj mu je uspješno evidentirati kako ne bi došlo do problema i iznamljivanja stanova za više osoba odjednom (odgovornost)

Korisnik (osoba koja će prebivati u apartmanu) – cilj mu je imati osiguran apartman

Agencija – cilj je zadovoljstvo korisnika, tj. izbjeći bilo kakve neugodnosti korisnika u slućaju da više osoba dobe isti stan u isto vrijeme

**Preconditions**: Agent se uspješno prijavio u sustav.

Korisnik (osoba koja iznajmljuje apartman na određeni period) daje sve potrebne informacije i plaća smještaj.

**Postconditions**: Apartman je uspješno iznamljen i nitko drugi ga ne može iznajmiti u tom periodu

**Glavni scenariji:**

1. Korisnik (agent) traži zadani apartman
2. Odabire apartman
3. Korisnik traži informacije od platitelja
4. Agent dodaje period rezervacije za odabrani apartman
5. Sustav pohranjuje sve izmjene

**Alternativni scenariji:**

3a. Platitelj je dao krive informacije

3a1. Korisnik traži od platitelja nove informacije

3a2. Povratak na korak 4

4a. Za zadani termin rezervacija već postoji

4a1 Korisnik traži novi datum rezervacije od korisnika

4b1 Korisnik ne želi novi datum već novi apartman

4b2 Povratak na korak 2

4c1 Korisnik daje za isti apartman novi datum

4c2 Povratak na korak 4

**Use case 6**: Izmjena informacija postojećeg apartmana

**Primarni aktor**: Zaposlenik agencije (agent)

**Stakeholderi**: Agent – cilj mu je uspješno evidentirati promjenu kako ne bi došlo do problema sa vlasnikom apartmana, zadovoljstvo sa vlasnicima mu daje veću proviziju

Agencija – cilj je zadovoljstvo korisnika

Korisnik – cilj je da informacije prikazane uz njegov apartman su točne

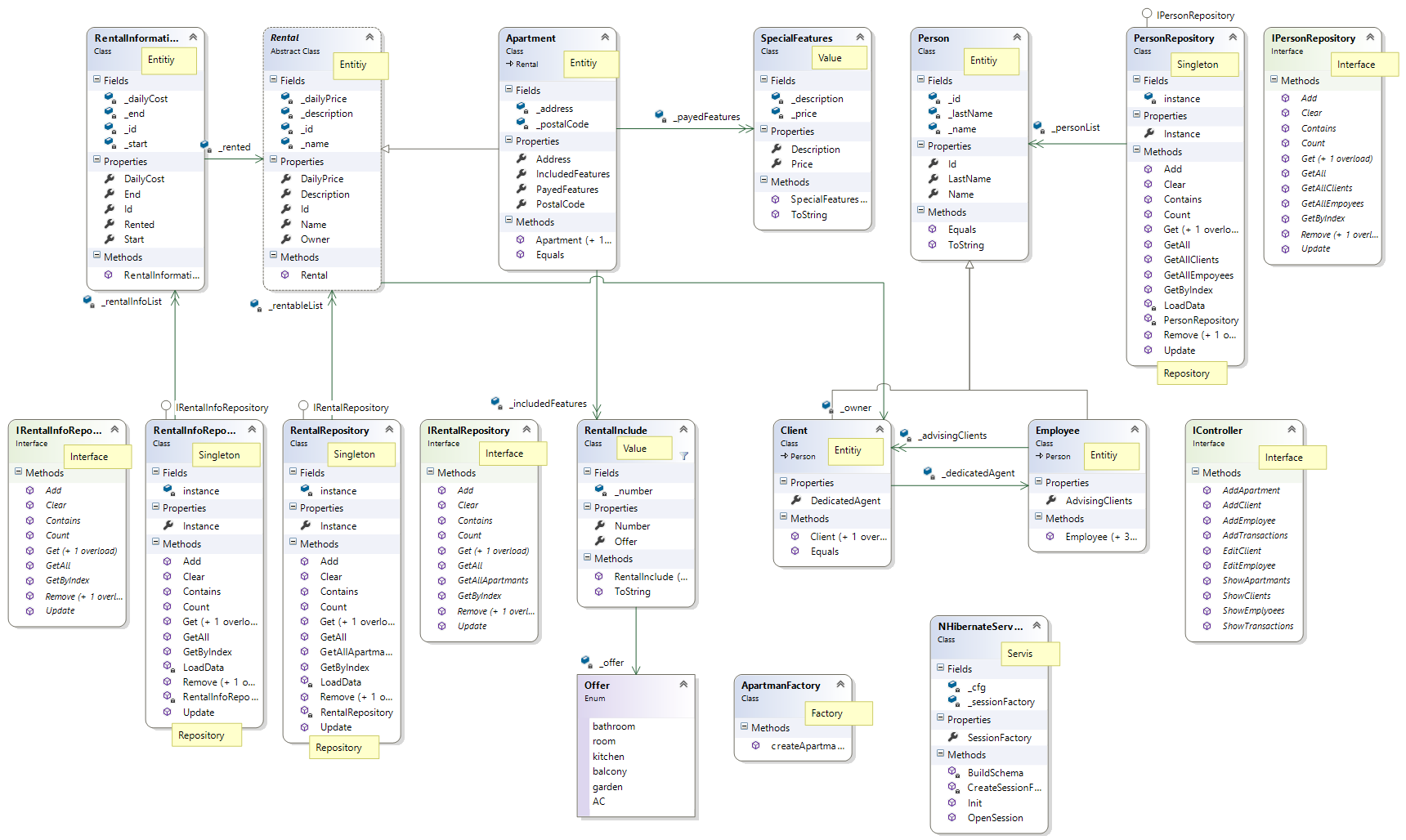
**Preconditions**: Agent se uspješno prijavio u sustav.

Klijenjt (vlasnik apartmana) daje sve potrebne informacije potrebne za promjenu.

**Postconditions**: Apartmanu su uspješno ažurirane informacije

**Glavni scenariji:**

1. Korisnik (agent) preuzime nove informacije o apartmanu od vlasnika
2. Traži zadani apartman
3. Korisnik unosi nove informacije.
4. Sustav ažurira informacije o apartmanu

Class diagram

Agregate

Slika 1 Dijagram razreda sustava

Opis objektnog modela

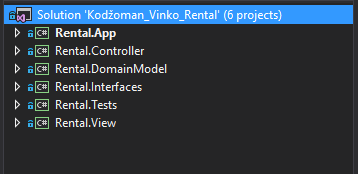
Objektno model domene nalazi se na Slici 1. Osoba u sustavu opisana je razredom Person (entity), koji sadrži podatke važne za svaku osobu u našem problemu (ime, prezime i jedinstveni identifikator, te metode za rad sa navedenim podacima). Iz razreda Person izvedeni su specijalni razredi Client i Employee (također entity) kojima se modeliraju specifične osobe unutar našeg problema, razred Person omogućava dodavanje novih vrsta osoba u sustav za eventualne buduće zahtjeve. PersonRepository razred implementira repository pattern i omogućava CRUD (create, read, update i delete) plus dodatne pomoćne opracije za rad s podacima unutar baze.

Stvari za iznamljivanje u sustavu su realizirane preko klase Rental (entitiy, agregat). Sustav trenutno radi sa apartmanima ali važan je naglasak na budućem proširenju i sustav mora omogućavati lako proširivanje, sa time na umu rental sadržave se potrebne informacije za iznamljivanje (ime, opis, dnevna cijena i jedinstveni identifikator) i iz njega je izveden specificiriani razred Apartmant. Apartman se dodatno proširuje sa kolekcijom klase RentalInclude koja omogućuje unos ukljućenih ponuda u apartman (broj spavaćih soba, klima, balkona, ...), svaka ukljućena ponuda sadrži broj i samu stvar u ponudi koja je realizirana ka enumeracija Offer. Uz ukljućene ponude, razred apartmant sadrži i specijalne ponude realezirane kao kolekciju. Svaka specijalna ponuda ima cijenu i opis same ponude. Za rad sa bazom koristi se razred RentalRepository koji implementira repository pattern i omogućava CRUD i dodatne operacije nad stvarima za iznamljivanje.

Sustav mora imati podršku za rezervacije, međutim bitno je da kako je opisano u design storyu, sustav ne sprema nikakve privatne informacije od osobe koja radi rezervaciju. Rezervacija je realizirana preko razreda RentalInformation i sadrzi referencu na Rental razred (tj. objekt koji se rezervira), vrijeme od kada do kada i cijenu po danu. Za rad sa bazom koristi se razred RentalInfoRepository koji implementira repository pattern i omogućava CRUD i dodatne operacije nad transakcijama. Sustav omogućava i pregled svih transakcija, bitno je provoditi validaciju kako ne bi više ljudi isti objekt rezervirali u isto vrijeme ili da se njihova vremena rezervacija preklapaju.

Arhitektura aplikacije – GUI

Aplikacija je realizirana po arhitekturnom patternu MVC (*Model-View-Controller*) pomoću window forms GUI librarya unutar Micrososft Visual Studia, struktura sustava je prikazana na Slici 2.



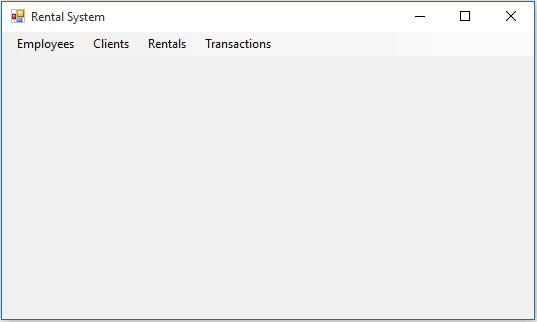
**Slika 2** Organizacija koda unutar sustava (po projektima)

*App* predstavlja ulaz u program i podiže glavnu formu koja se nalazi u *View* zajedno sa ostalim formama. *Controller* predstavlja kontroler za MVC (Slika 3) i delegira rad između *Viewa* i *DomainModela*  koji predstavlja domenu tj. model za naš problem. *Interfaces* sadrži bitna globalna sučelja radi povečanja decouplinga (Icontroller i IRepositoriy). Pošto je sustav rađen pomoću TDD, unutar *Tests* projekta nalaze se svi testovi koji su korišteni u izradi sustava.

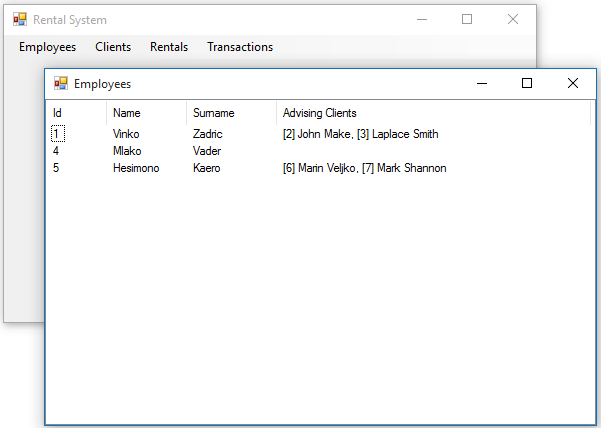


**Slika 3** MVC arhitekturni pattern (model view controller)

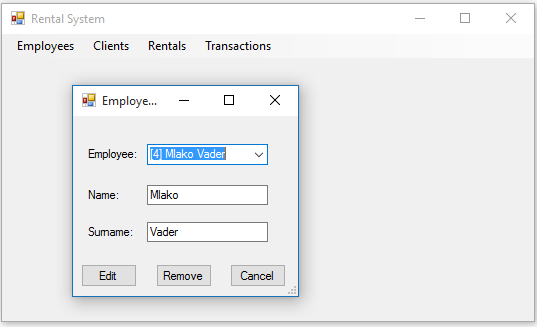
MVC je prikazan na Slici 3, i takvom arhitekturom je realiziran sustav. MVC ima brojne realizacije i varijante te sljedi opis varijante korištene u ovom seminaru. Sastoji se od tri komponente *View,* predstavlja presentation layer tj. pogled koji vidi korisnik – GUI. *Model*, domain model aplikacije u kojem se nalazi model (podaci) zajedno sa implementiranim ponašanjem (srce sustava). *Controller* delegira i šalje *model* u *view* te poziva i prikazuje *view* na zaslon, kontroler implementira sučelje *Icontroller* koje omogućuje veće decouplanje između *viewa* i *controllera*, *view* ima direktnu referencu samo na interface a sam *controller* mu je preda preko konstruktora. MVC je pogdan jer omogućava decoupling i povećava reusability samog programskog koda, pogotovo modela domena. Uz decoupling, olakšava i testiranje same aplikacije. *Viewovi* su realizirani kao windows forme i svaki *view* je zasebna windows forma.U nastavku sljedi pregled svih bitnih formi (*viewova*) u sustavu.



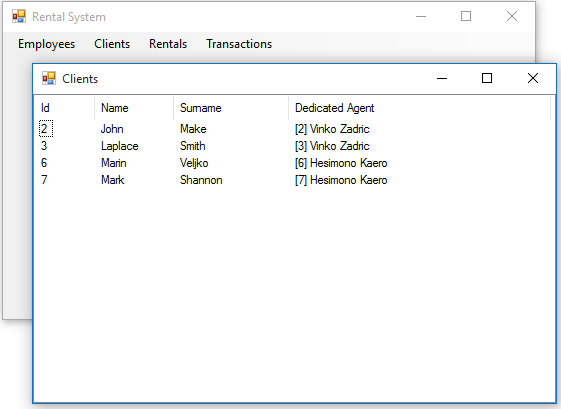
**Slika 4** Glavni izbornik (MainForm), omogućuje pokretanje ostalih forma



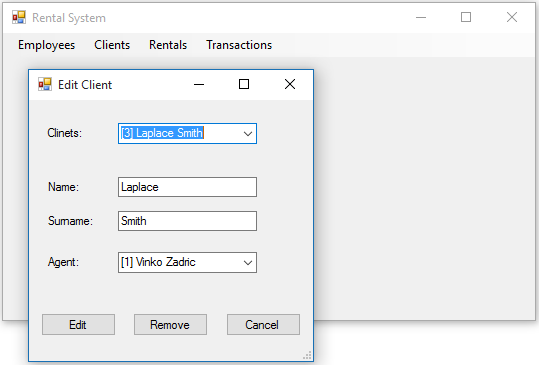
**Slika 5** Pregled zaposlenika firme



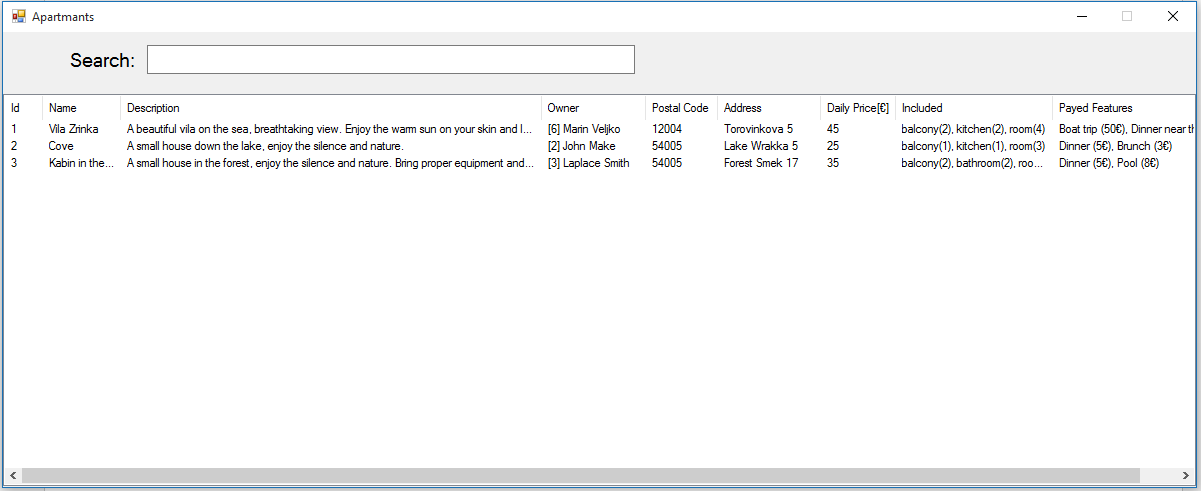
**Slika 6** Izmjena i brisanje zaposlenika firme

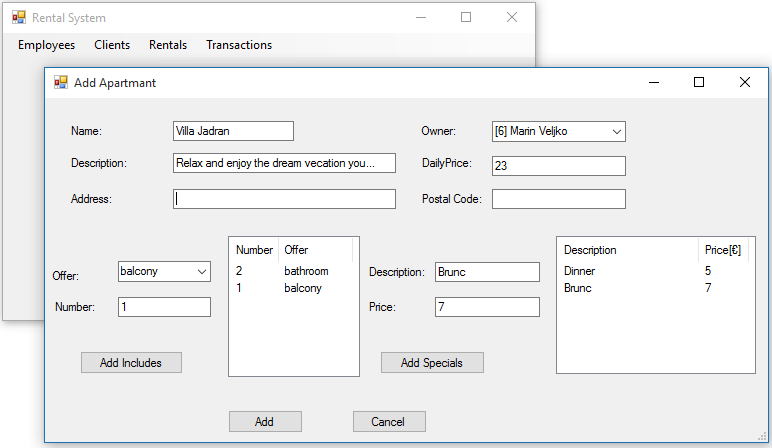


**Slika 7** Pregled korisnika (klijenta) firme

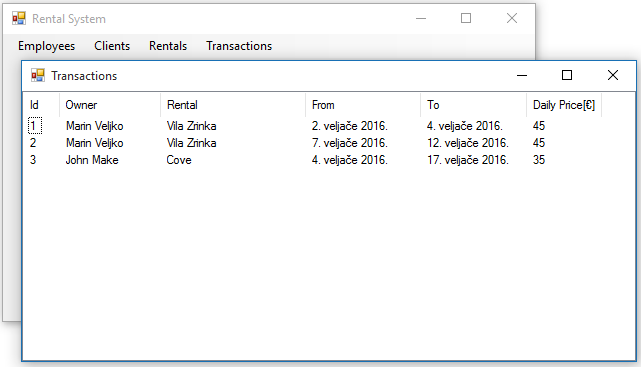


**Slika 8** Izmjena i brisanje korisnika (klijenta) firme

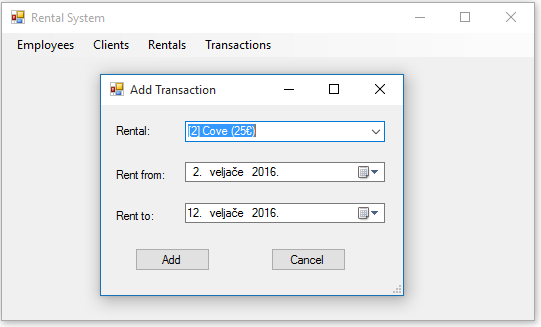
**Slika 9** Prikaz apartmana koji su unutar sustava (svi se mogu ujedno i iznajmiti) i trake za pretraživanje



**Slika 10** Dodavanje novog apartmana u sustav



**Slika 11** Prikaz svi prošlih i trenutnih transakcija – rezervacija



**Slika 12** Dodavanje novih transakcija – rezervacija u sustav