SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Seminar iz kolegija

**Objektno oblikovanje**

**„ScooterRent“**

*Case study*

Gabriela Dorić

Zagreb, siječanj 2021.

Sadržaj

[1. Uvod 3](#_Toc62718762)

[2. Zahtjevi 4](#_Toc62718763)

[3. Opis modela 7](#_Toc62718764)

[4. Arhitektura 8](#_Toc62718765)

[5. Objektno-relacijsko mapiranje 14](#_Toc62718766)

# Uvod

Pružatelj usluge iznajmljivanja električnih romobila želi proširiti svoje poslovanje na više poslovnica. Kako bi što lakše vodio posao, želi na jednom mjestu imati evidenciju o svim električnim romobilima kao i podatak o tome koji od njih su iznajmljeni i te koji korisnici su ih iznajmili.

Iz te ideje nastala je potreba za aplikacijom „*ScooterRent*“ koja vodi evidenciju o poslovnicama, električnim romobilima i iznajmljivačima.

Aplikacija je namijenjena vlasniku za evidenciju poslovnica i njihova rada u različitim gradovima, te zaposlenicima koji u poslovnicama mogu dodavati nove romobile, iznajmljivati romobile te dodavati nove korisnike.

Aplikacija omogućuje dodavanje novih romobila u sustav. Svaki romobil ima naziv, ime proizvođača, maksimalnu brzinu izraženu u km/h, vrstu romobila kojoj pripada, cijenu iznajmljivanja izraženu u kn/dan te poslovnicu u kojoj se nalazi. Omogućeno je dodavanje novih romobila i brisanje postojećih romobila u slučaju povlačenja istih iz ponude.

Za svaki iznajmljeni romobil vodi se evidencija pri čemu se evidentira datum i vrijeme iznajmljivanja i datum i vrijeme vraćanja romobila te podatci o osobi koja ga je iznajmila. Jedna osoba može iznajmiti više električnih romobila.

Klijenti koji posuđuju romobile evidentirane su u sustavu. Svaka osoba ima ID, ime, prezime, e-mail adresu, OIB, dugovanje s obzirom na iznajmljene romobile te matični ured u kojem je osoba registrirana. Iako osoba ima matičnu poslovnicu u kojem je registrirana, može iznajmljivati romobile u bilo kojoj poslovnici. Podatke o korisnicima moguće je i brisati.

Poslovnice su opisane s ID vrijednošću, nazivom, gradom u kojem se nalaze, datumom osnivanja, adresom i radnim vremenom. Poslovnice je moguće dodavati, brisati i pregledavati statistike pojedine poslovnice, koje sadržavaju informacije o svim romobilima u odabranoj poslovnici, o iznajmljenim romobilima te o korisnicima koji su registrirani u toj poslovnici.

Također, korisniku aplikacije prikazuje se dugovanje svakog iznajmljivača koje se računa na temelju broja dana koliko je romobil bio iznajmljen( ili romobili, s obzirom da jedna osoba može posuditi više romobila).

# Zahtjevi

Popis bitnijih *use case-ova* ( obrazaca uporabe)

1. Dodavanje novog romobila
2. Brisanje postojećeg romobila
3. Pregled svih romobila
4. Pregled posuđenih romobila
5. Dodavanje novog klijenta
6. Brisanje klijenta iz sustava
7. Prikaz svih klijenata
8. Dodavanje poslovnice
9. Brisanje poslovnice
10. Pregled stanja u odabranoj poslovnici

Analiza odabranih obrazaca uporabe

|  |
| --- |
| **Use case 4: Pregled posuđenih romobila** |
| **Primarni aktor:** Zaposlenik (vlasnik)  **Stakeholders:**   * Zaposlenik: osoba koja pregledava sve romobile u sustavu   **Preduvjeti:** Aplikacija pokrenuta  **Glavni scenarij:**   1. Aplikacija dohvaća zahtijevani popis romobila.   **Alternativni scenarij:**  1.a Ne postoji niti jedan posuđeni romobil u poslovnicama. |

|  |
| --- |
| **Use case 5: Dodavanje novog klijenta** |
| **Primarni aktor**: Zaposlenik  **Stakeholders:**   * Zaposlenik: osoba koja unosi podatke o klijentu * Klijent: osoba koja mora biti unesena u sustav kako bi mogla posuditi romobil   **Preduvjeti:**   * Aplikacija pokrenuta * Klijent nije u sustavu   **Glavni scenarij:**   1. Zaposlenik odabire opciju kreiranja klijenta 2. Popunjava se forma sa svim potrebnim podatcima. 3. Aplikacija provjerava ispravnost podataka. 4. Klijent se unosi u sustav.   **Alternativni scenarij:**  3.a Uneseni podatci nisu ispravni/potpuni  3.b Aplikacija obavještava zaposlenika o neispravnosti podataka te odbija zahtjev za unos novog klijenta. |
| **Use case 9: Brisanje poslovnice** |
| **Primarni aktor:** Zaposlenik (vlasnik)  **Stakeholders:**   * Zaposlenik - osoba koja briše poslovnicu   **Preduvjeti:**   * Aplikacija pokrenuta * Poslovnica postoji u sustavu   **Glavni scenarij:**   1. Zaposlenik odabire poslovnicu koju želi ukloniti. 2. Sustav zahtjeva trajno brisanje poslovnice. 3. Aplikacija vraća podatak o brisanju poslovnice.   **Alternativni scenarij:**  2.a Postoje posuđeni romobili i zaduženja unutar odabrane poslovnice  2.b Aplikacija obavještava korisnika o tome i odbija zahtjev za brisanjem poslovnice. |

|  |
| --- |
| **Use case 10: Pregled stanja u odabranoj poslovnici** |
| **Primarni aktor:** Zaposlenik (vlasnik)  **Stakeholders:**  Zaposlenik - želi provjeriti stanje iznajmljivanja romobila u poslovnici  **Preduvjeti:**   * Aplikacija pokrenuta * Zaposleniku prikazana forma za pregled stanja u poslovnici   **Glavni scenarij:**   1. Zaposlenik odabire opciju pregleda stanja u određenoj poslovnici 2. Aplikacija dohvaća podatke. 3. Aplikacija vraća podatke o stanju u poslovnici   **Alternativni scenarij:**  2.a Ne postoji niti jedan posuđeni romobil ili niti jedan klijent u odabranoj poslovnici. |

# 3. Opis modela

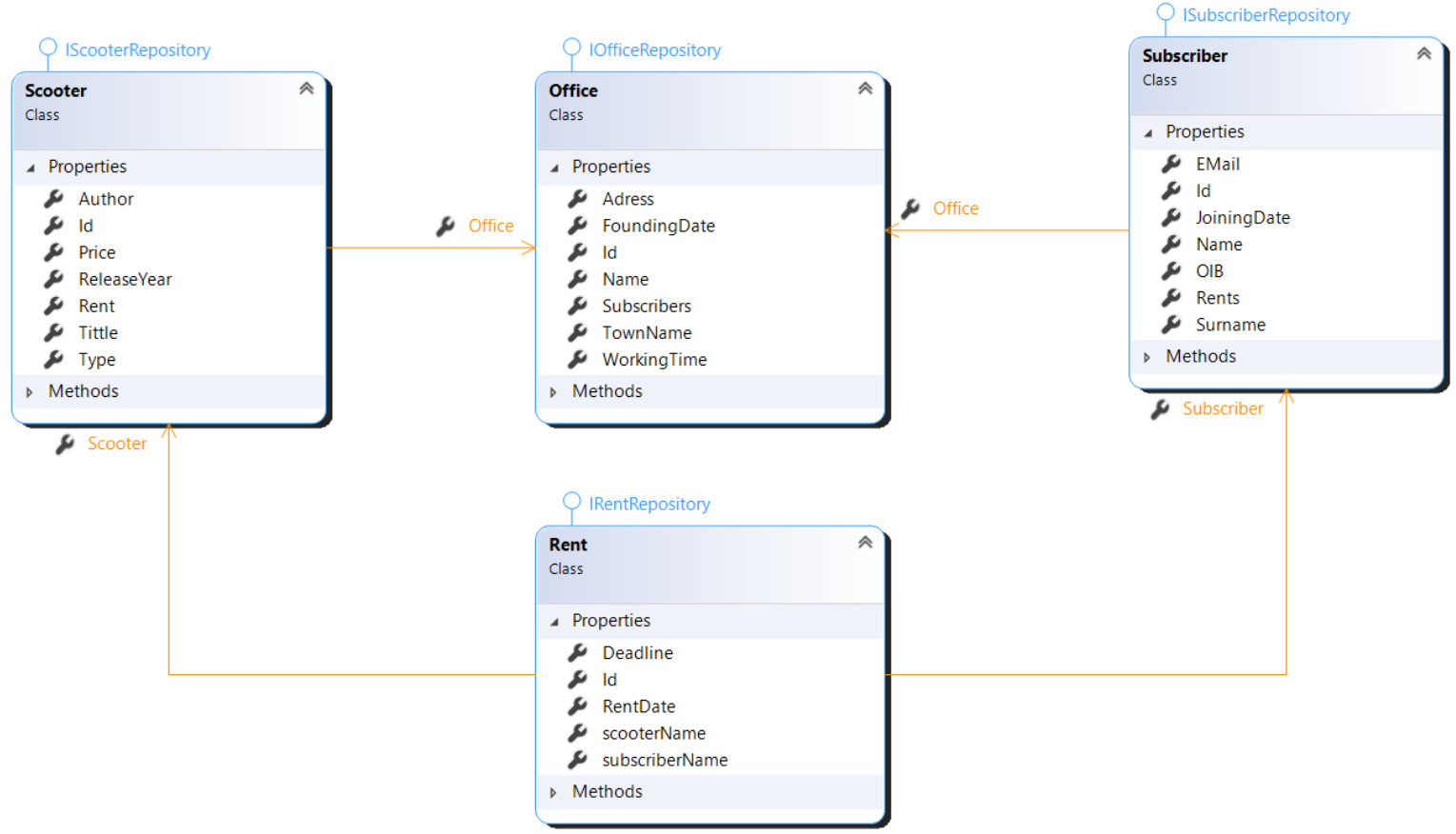
Objektni model domene čine klase Office, Scooter, Subscriber i Rent.

Poslovnica je opisana *entity* razredom Office koji sadrži sljedeće atribute: ID, naziv, grad, datum osnivanja, adresa, radno vrijeme. IOfficeRepository omogućuje dohvaćanje i spremanje poslovnica, a ostvaren je *Singleton patternom* kako bi u bilo kojem trenutku izvođenja postojala samo jedna instanca repozitorija s ciljem izbjegavanja nekonzistentnosti podataka.

Električni romobil je realiziran *entity* klasom Scooter koja sadrži sljedeće atribute: ID, naziv, proizvođač, maksimalna brzina, vrsta romobila, cijena iznajmljivanja po danu te poslovnica kojoj pripada.

Pretplatnik, odnosno klijent, opisan je *entity* klasom Subscriber koja sadrži sljedeće atribute: ID, ime, prezime, email, OIB, datum registracije, matična poslovnica u kojoj je pretplatnik unesen u sustav.

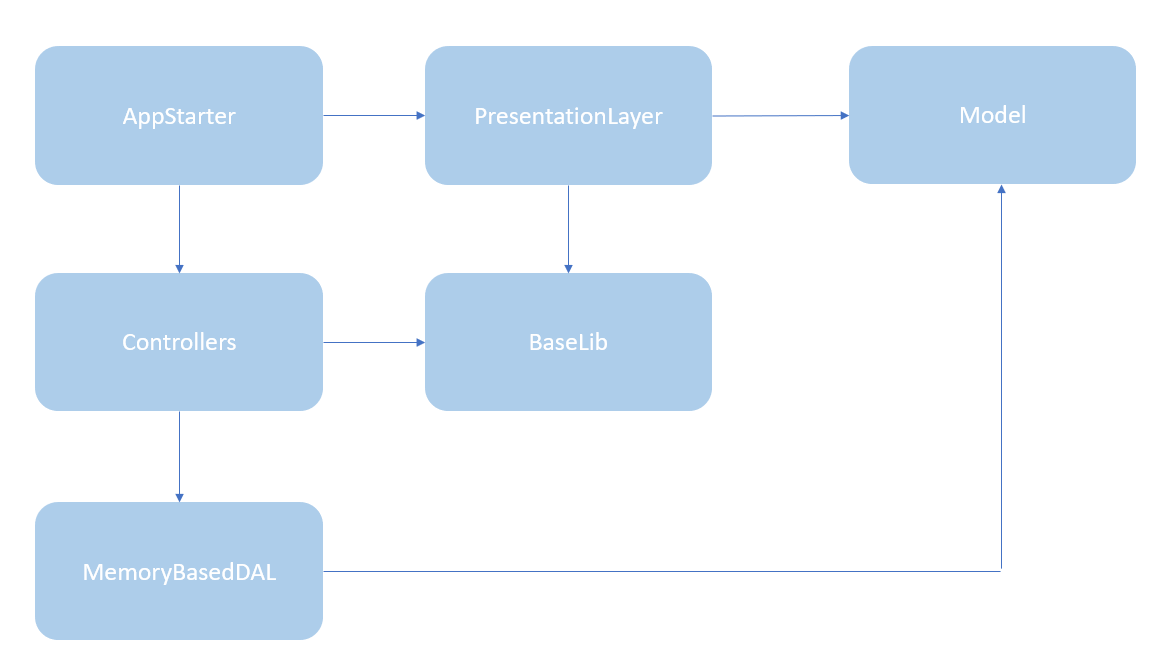
Iznajmljivanje je ostvareno *entity* klasom Rent koja sadrži sljedeće atribute: ID, iznajmljeni električni romobil, datum iznajmljivanja, datum isteka najma, klijent koji je iznajmio romobil. Uloga ove klase je pohranjivanje svake posudbe u poslovnicama. Na slici 1 prikazana je opisana arhitektura projekta.



Slika 1. Dijagram razreda

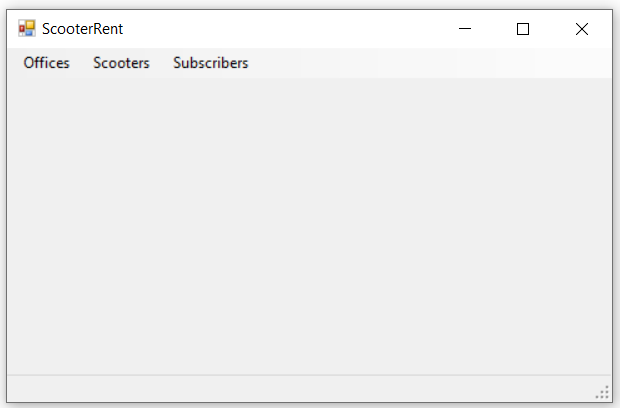
# 4. Arhitektura

U ovom projektu korištena je MVC arhitektura. U *PresentationLayer* projektu nalaze se svi *viewi* odnosno forme koje se korisniku mogu prikazati. Projekt *Controllers* ima referencu na projekt *PresentationLayer*, ali obrnuto ne vrijedi jer bi došlo do kružne ovisnosti između dva modula (engl*. circular dependency*). Iz tog razloga oba projekta referenciraju projekt *BaseLib* koji sadrži sučelja preko kojih se obavlja komunikacija. Kontroleri implementiraju ta sučelja te im klase iz *PresentationLayer* projekta pristupaju preko tih apstraktnih sučelja. Kontroler se ne može instancirati iz *PresentationLayer projekta* pa se aplikacija pokreće iz AppStarter projekta u kojem su referencirani projekti *Controllers* i PresentationLayer. U projektu AppStarter stvaraju se konkretni kontroleri koji se predaju u *PresentationLayer* gdje ih *viewi* koriste za komunikaciju. Na slici 2 prikazana je arhitektura opisanog projekta.

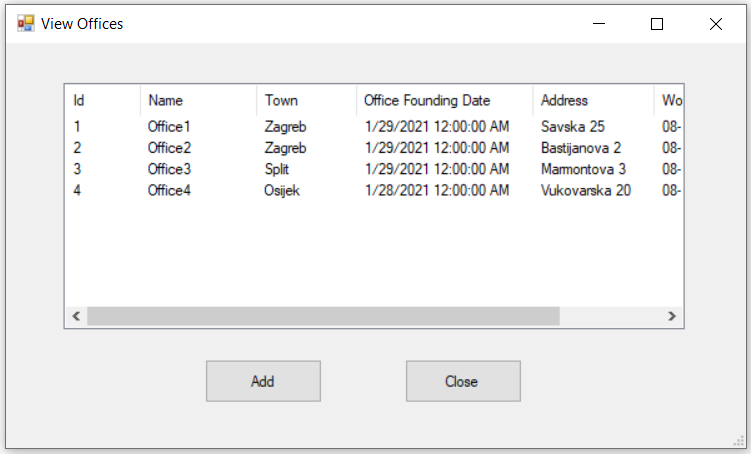


Slika 2. Arhitektura projekta

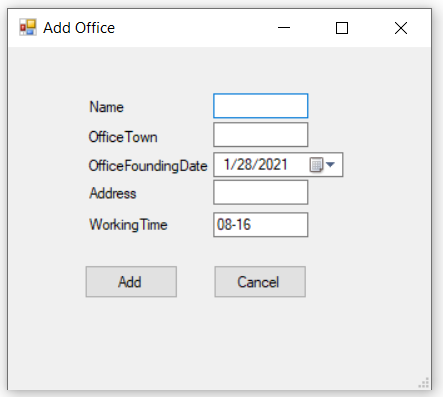
U nastavku su prikazane slike svih formi.



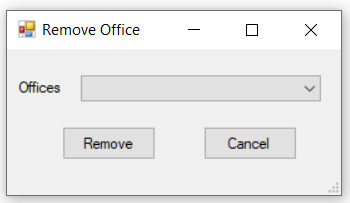
Slika 3. Inicijalna forma



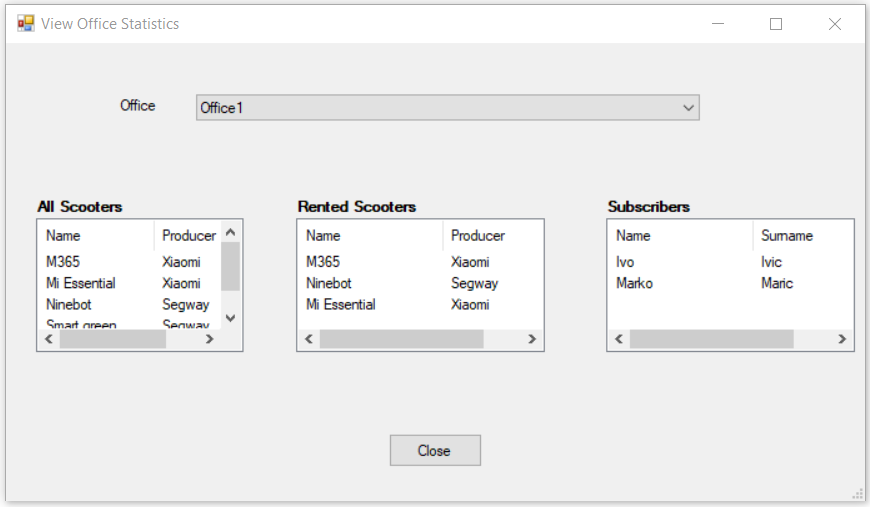
Slika 4. Forma za pregled poslovnica



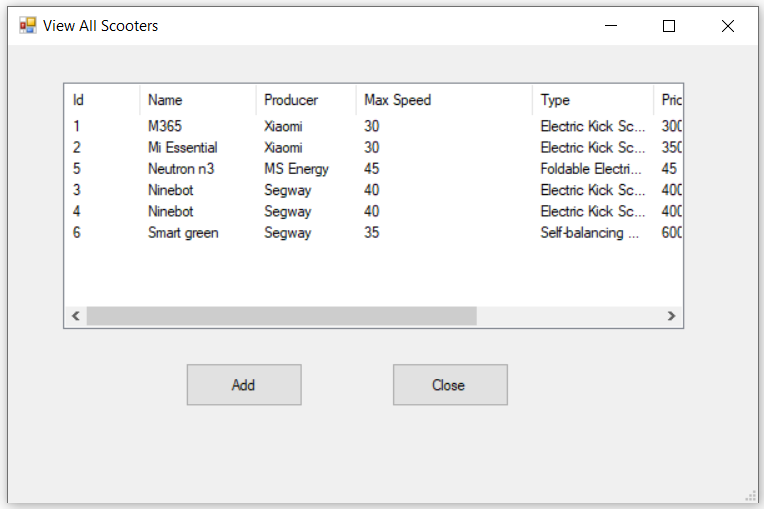
Slika 5. Forma za dodavanje poslovnice



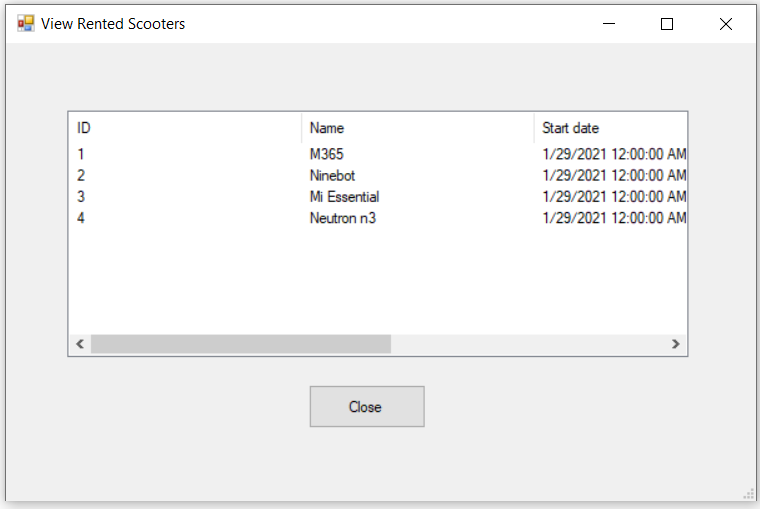
Slika 6. Forma za uklanjanje poslovnice



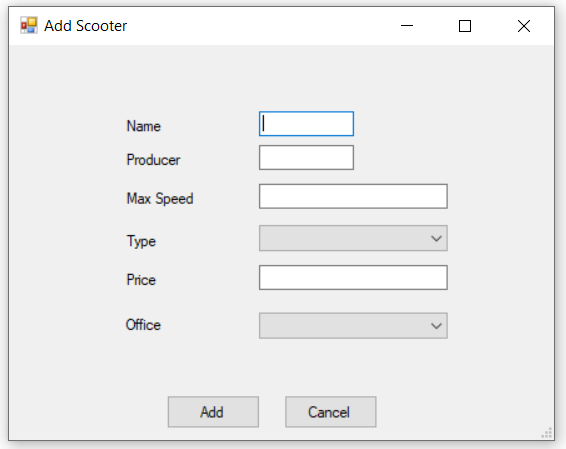
Slika 7. Forma za pregled pojedine poslovnice



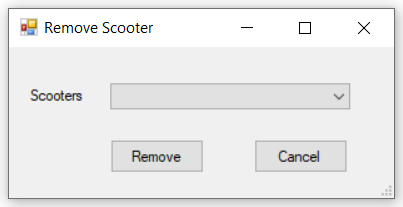
Slika 8. Forma za pregled svih romobila



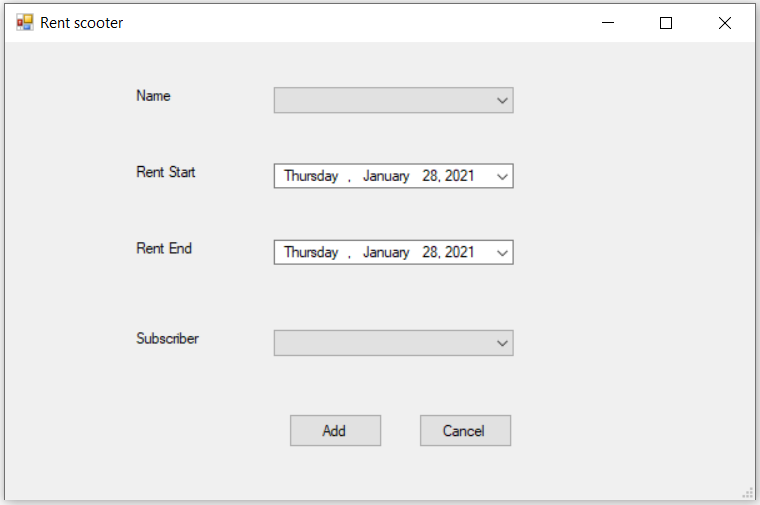
Slika 9. Forma za pregled iznajmljenih romobila



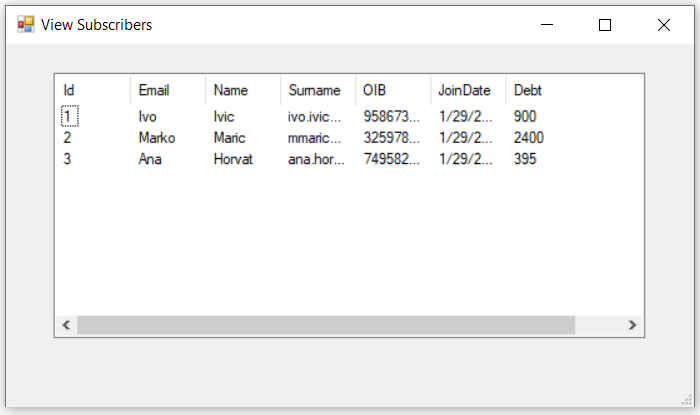
Slika 10. Forma za dodavanje romobila



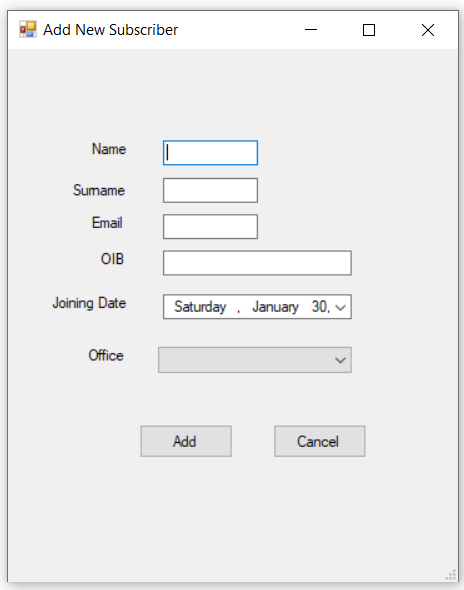
Slika 11. Forma za uklanjanje romobila



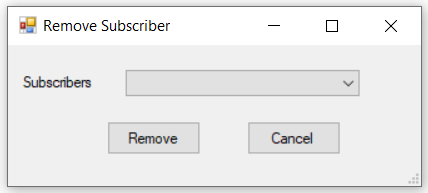
Slika 12. Forma za iznajmljivanje romobila



Slika 13. Forma za pregled korisnika



Slika 14. Forma za dodavanje novog korisnika



Slika 15. Forma za uklanjanje korisnika

# 5. Objektno-relacijsko mapiranje

Mapiranje je ostvareno korištenjem FluentNHibernate frameworka i SQLite baze podataka. Mapirana su četiri razreda iz objektnog modela: Office, Scooter, Subscriber i Rent.

Office mapiranje sastoji se od definiranja parametara poslovnice i definiranja HasMany odnosa prema korisnicima.

public class OfficeMapping:ClassMap<Office>{

public OfficeMapping() {

Id(x => x.Id).GeneratedBy.Native();

Map(x => x.Name).Length(30).Not.Nullable();

Map(x => x.TownName).Length(30).Not.Nullable();

Map(x => x.FoundingDate).Not.Nullable();

Map(x => x.Adress).Length(30).Not.Nullable();

Map(x => x.WorkingTime).Not.Nullable();

HasMany(x => x.Subscribers);

Table("Office");

}

}

Scooter mapiranje sastoji se od mapiranja parametara romobila i reference na Rent i Office entitete.

public class ScooterMapping : ClassMap<Scooter>{

public ScooterMapping() {

Id(x => x.Id).GeneratedBy.Identity();

Map(x => x.Tittle).Length(30).Not.Nullable();

Map(x => x.Producer).Length(30).Not.Nullable();

Map(x => x.ReleaseYear).Not.Nullable();

Map(x => x.Type).Length(30).Not.Nullable();

Map(x => x.Price).Not.Nullable();

References(x => x.Rent).Not.LazyLoad();

References(x => x.Office).Not.LazyLoad();

Table("Scooter");

}

}

Subscriber mapiranje sastoji se od mapiranja parametara koi pripadaju korisniku (iznajmljivaču) te referencu na poslovnicu te iznajmljivanja romobila.

public class SubscriberMap : ClassMap<Subscriber>{

public SubscriberMap(){

Id(x => x.Id).GeneratedBy.Identity();

Map(x => x.Name).Length(30);

Map(x => x.Surname).Length(30);

Map(x => x.Email).Length(30);

Map(y => y.OIB);

Map(y => y.JoiningDate);

References(y => y.Office);

HasMany(y => y.Rents).Cascade.All();

Table("Subscriber");

}

}

Rent mapiranje sastoji se od definiranja parametara vezanih za iznajmljivanje te reference na Subscriber i Scooter entitete.

class RentMapping:ClassMap<Rent>{

public RentMapping() {

Id(x => x.Id).GeneratedBy.Identity();

Map(x => x.RentDate).Not.Nullable();

Map(x => x.Deadline).Not.Nullable();

References(x => x.Subscriber).Not.LazyLoad();

References(x=> x.Scooter).Not.LazyLoad();

Table("Rent");

}

}