**PROJEKAT iz predmeta “BAZE PODATAKA 3”**

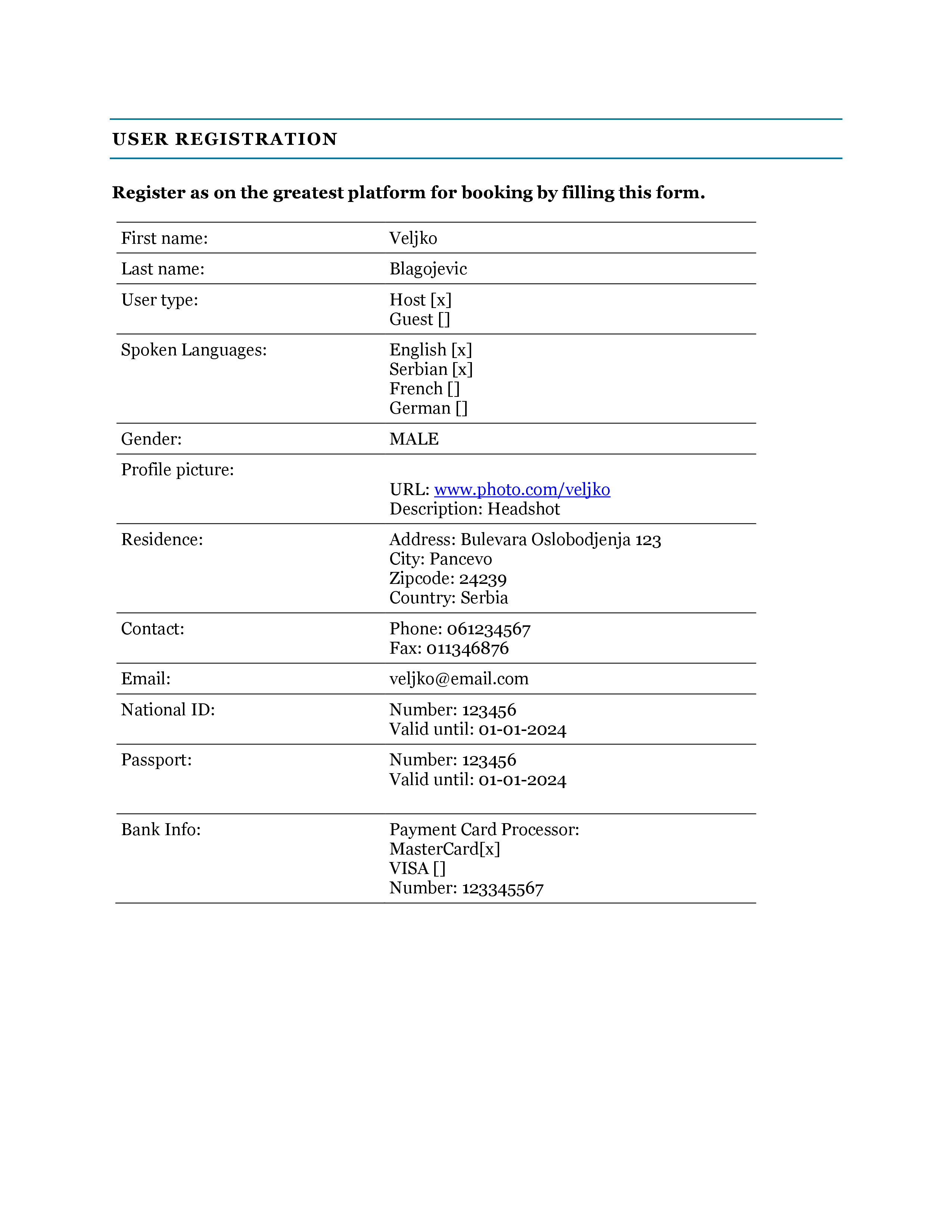
|  |  |  |  |
| --- | --- | --- | --- |
| **IME I PREZIME** | | **Veljko Blagojević** | |
| Broj indeksa | | **2023/3031** | |
| Studijski modul | | Informacioni sistemi | |
| Stečena diploma osnovnih akademskih studija | | Diplomirani inženjer organizacionih nauka | |
| **NAZIV PROJEKTA** | | Sistem za rezervaciju smeštaja | |
| Napomena | |  | |
| *Verzija* | *Datum promene* | | *Opis promene* | |
| 1.0 | 09-Nov-2023 | | **PMOV i relacioni model** | |
| 2.0 | 16-Nov-2023 | | **Denormalizacija relacija** | |
| 3.0 | 23-Nov-2023 | | **Korisnički definisani tipovi i trigeri** | |
| 4.0 | 07-Dec-2023 | | **Logička i fizička optimizacija** | |
| 5.0 | 21-Dec-2023 | | **Implementacija veb aplikacije** | |
|  |  | |  | |

# OPIS PROJEKTA

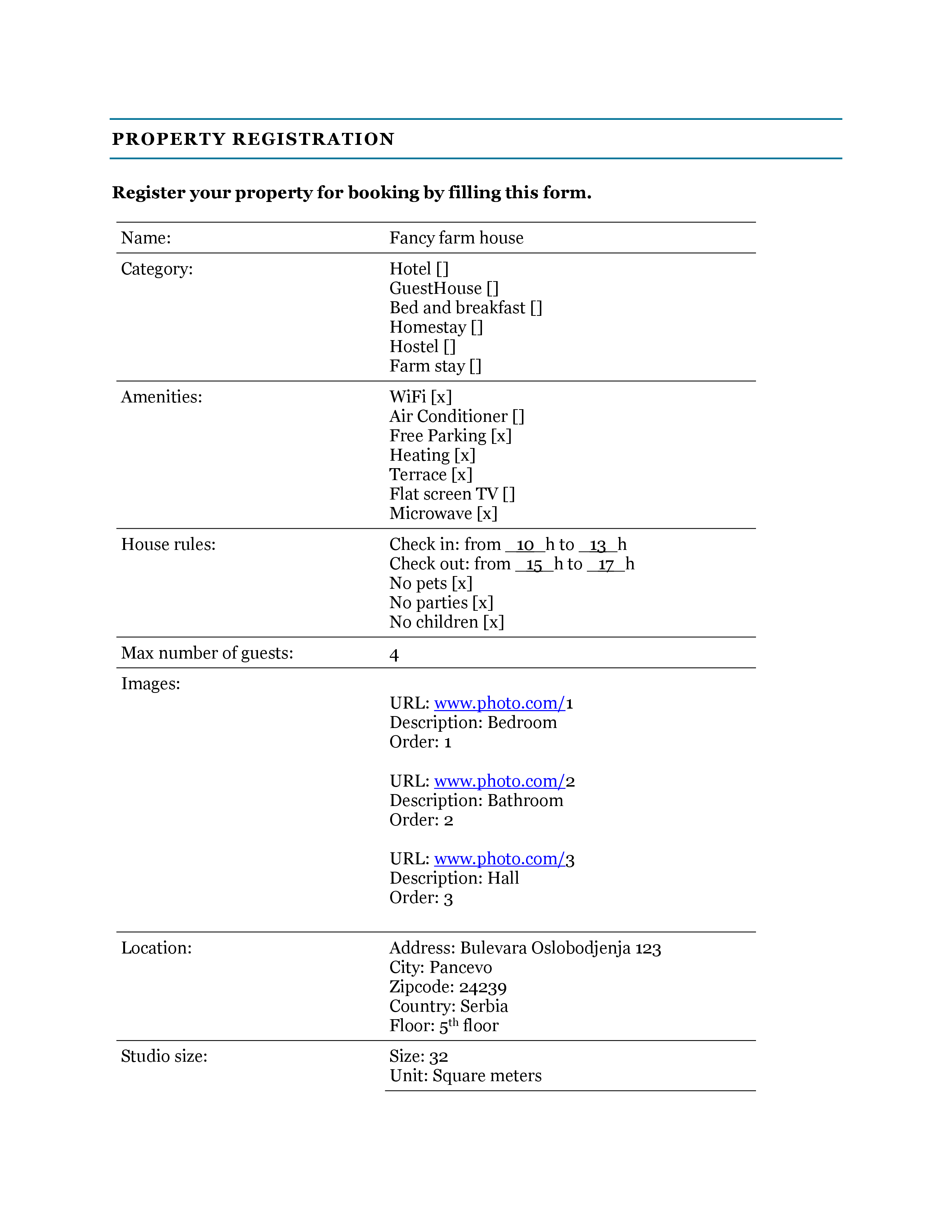
## Opis podsistema za koji se projektuje baza podataka

|  |
| --- |
| Uspešna putovanja zahtevaju kratkotrajno izdavanje smeštaja zarad odmora i prenoćišta. Online prikazivanje i rezervisanje datih objekata je daleko uprostilo ceo postupak pronalaženja kvalitetnog prostora za boravak. Korisnici, kroz informacioni sistem, registruju sebe kao domaćine sa objektima za izdavanje uz pomoć obilja opisnih informacija za svaki. Za svaki objekat je potrebno definisati lokaciju, cenu, fotografije, razne kućne potrepštine kao što su grejanje, WiFi, besplatan parking, mikrotalasna pećnica itd. Korisnik koji je gost može da izabere željeni apartman, i ukoliko je u navedenom terminu slobodan, da rezerviše i plati online karticom. Pri ovom postupku mora da preda informacije o sebi, kao i o broju gostiju za dat vid rezervacije. Nakon odsedanja, gost ima prava da napiše recenziju za dati prostor. |

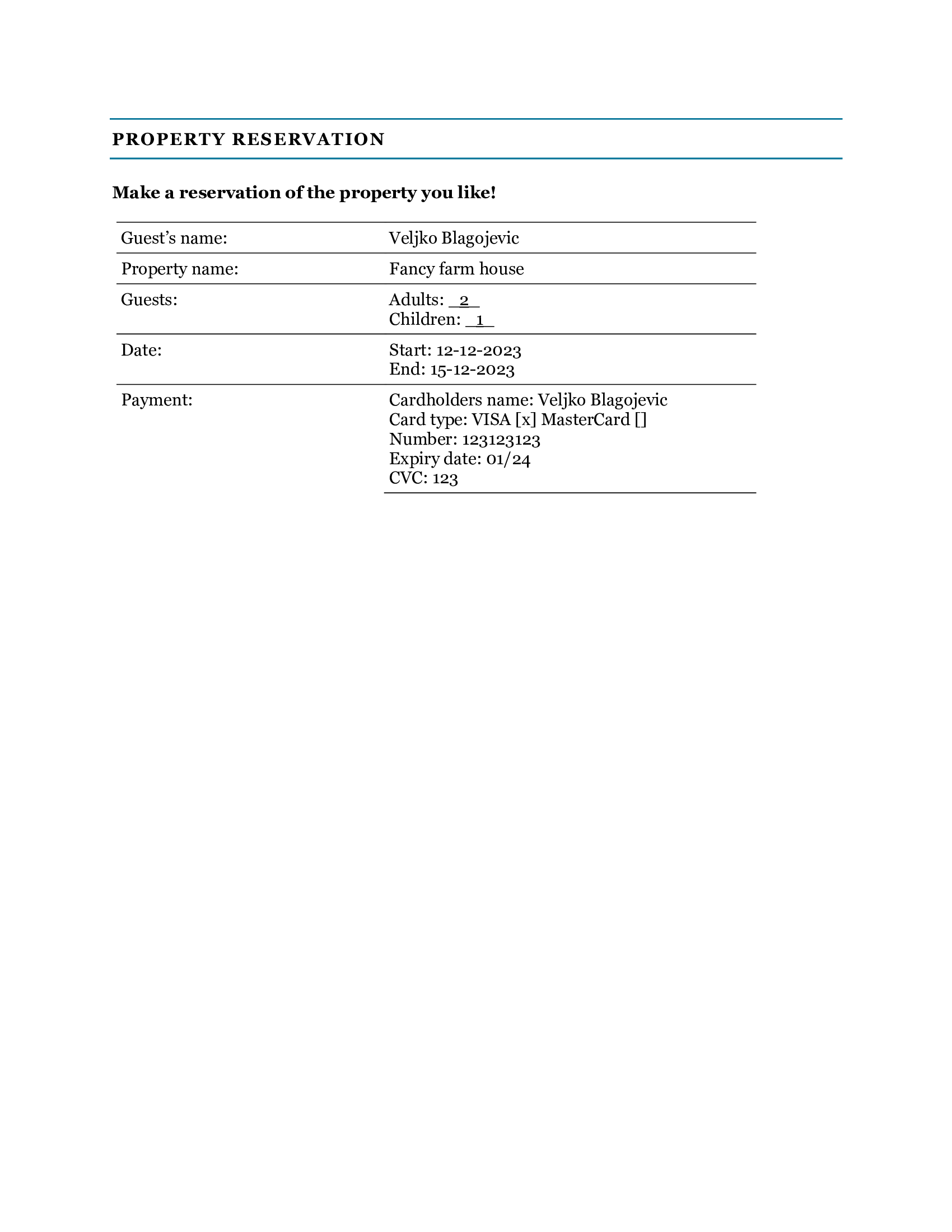
## Poslovna dokumenta koja se koriste u procesu



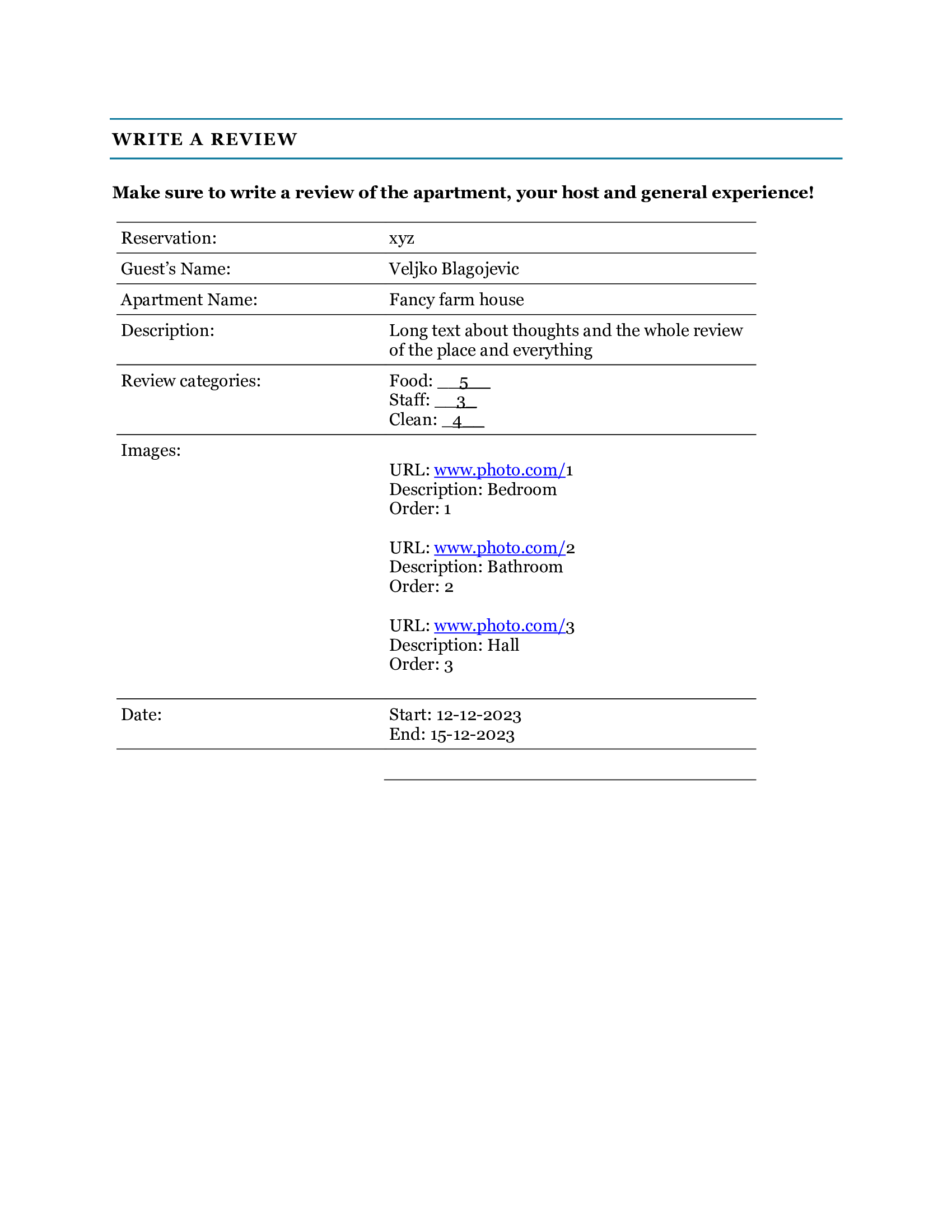
Dokument 1 Registracija korisnika



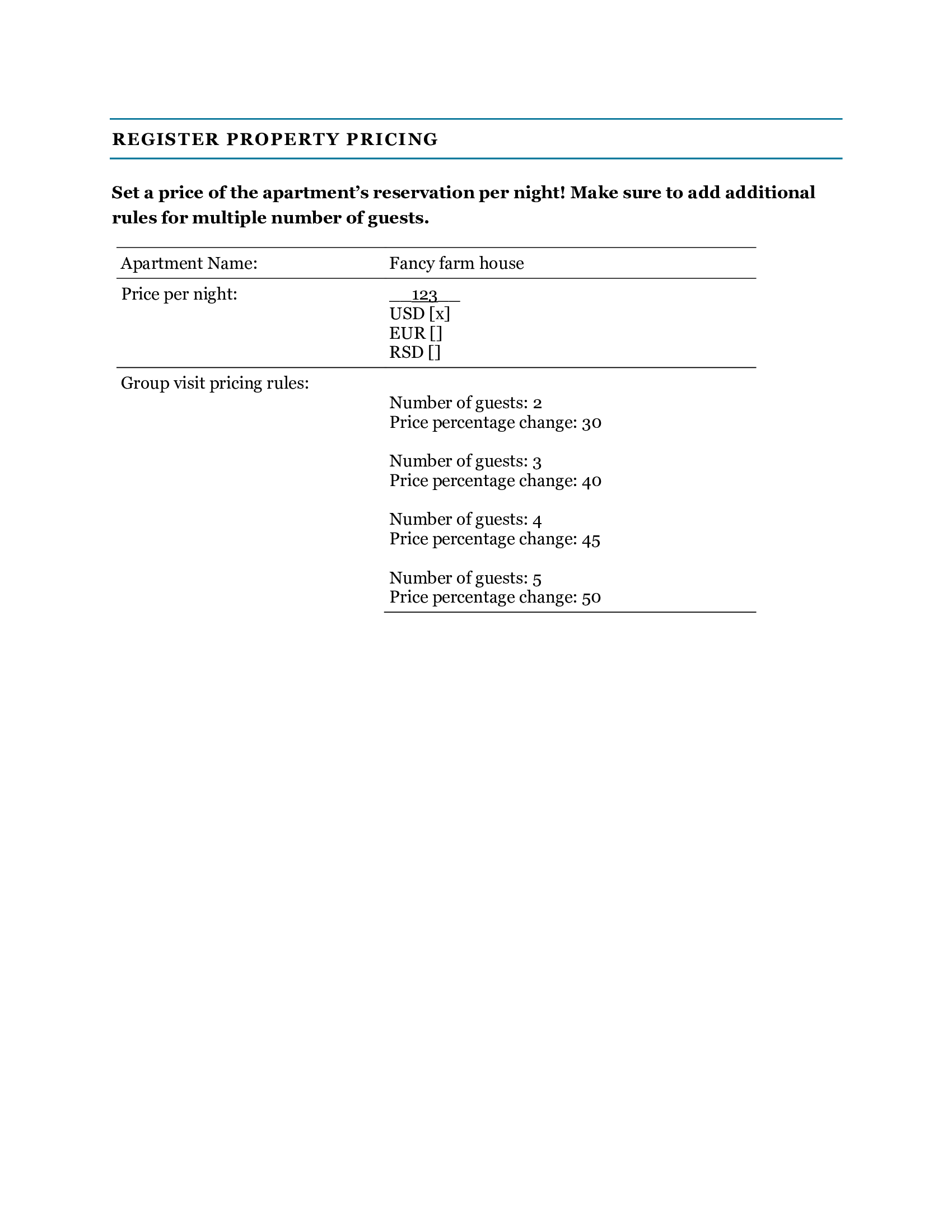
Dokument 2 Registracija smeštaja



Dokument 3 Rezervacija smeštaja



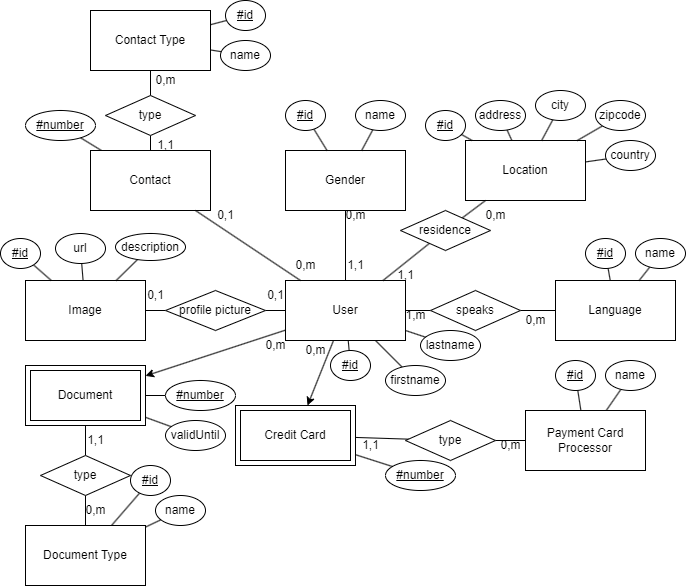
Dokument 4 Pisanje recenzije



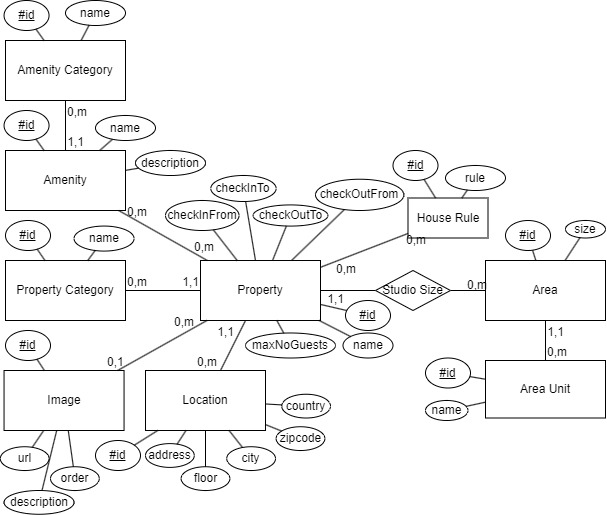
Dokument 5 Definisanje cene smeštaja

# PROJEKTOVANJE BAZE PODATAKA

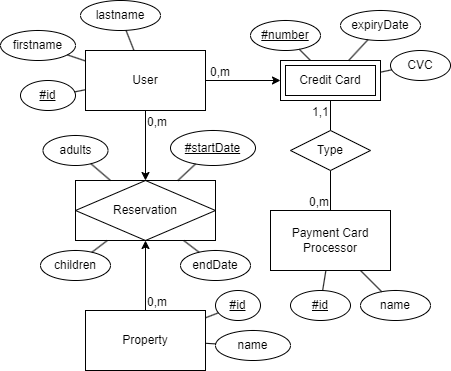
## Model podatka



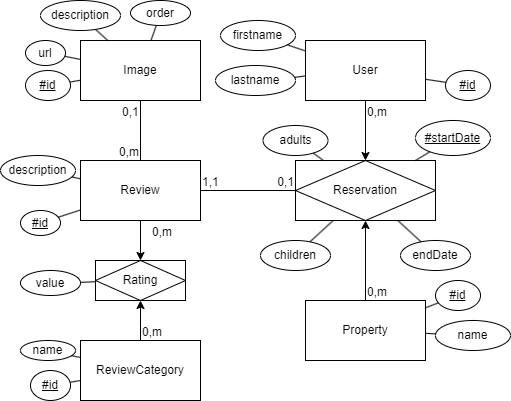
Model podataka Registracija korisnika



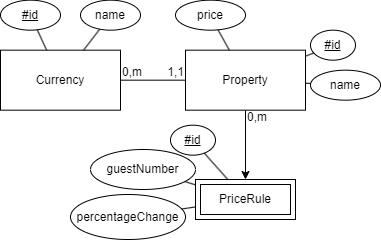
Model podataka Registracija smeštaja



Model podataka Rezervacija smeštaja



Model podataka Pisanje recenzije



Model podataka Definisanje cene smeštaja

## Relacioni model

**AmenityCategory** (amenityCategoryId, name)

**Amenity** (amenityId, name, description, *amenityCategoryId*)

**AmenityProperty** (*amenityId, propertyId*)

**PropertyCategory** (propertyCategoryId, name)

**AreaUnit** (areaUnitId, name)

**Area** (areaId, size, *areaUnitId*)

**Currency** (currencyId, name)

**PriceRule** (*propertyId*, priceRuleId, guestNumber, percentageChange)

**HouseRule** (houseRuleId, rule)

**PropertyHouseRule** (*propertyId, houseRuleId*)

**ContactType** (contactTypeId, name)

**Contact** (number, *userId*, *contactTypeId*)

**Gender** (genderId, name)

**Location** (locationId, country, city, zipcode, address, floor)

**Language** (languageId, name)

**UserSpeaksLanguage** (*userId, languageId*)

**PaymentCardProcessor** (paymentCardProcessorId, name)

**CreditCard** (creditCardNumber, *userId*, expiryDate, CVC)

**DocumentType** (documentTypeId, name)

**Document** (documentNumber, *userId*, validUntil, *documentTypeId*)

**Image** (imageId, url, description, order)

**ProfilePicture** (*userId, imageId*)

**PropertyImage** (*imageId, propertyId*)

**ReviewCategory** (reviewCategoryId, name)

**Property** (propertyId, name, price, *propertyCategoryId, currencyId*, *areaId, hostId, locationId*)

**User** (userId, name, *genderId*, *residenceId*, email)

**Reservation** (startDate*, guestId, propertyId*, adults, children, endDate)

**Review** (reviewId, description, *startDate, guestId, propertyId*)

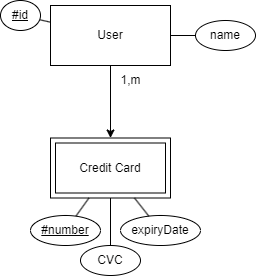
**ReviewImage** (*imageId, reviewId*)

**Rating** (*reviewId, reviewCategoryId*, value)

# DENORMALIZACIJA RELACIJA

## Denormalizacija relacija uz narušavanje 2NF (Pre-joining)

Pre denormalizacije:



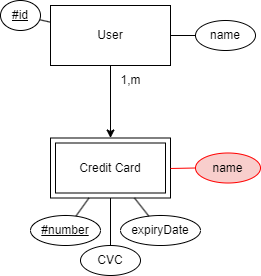
**User** (userId, name, *genderId*, *residenceId*)

**CreditCard** (creditCardNumber, *userId*, expiryDate, CVC)

**Funkcionalna zavisnost:**

userId, creditCardNumber -> CVC, expiryDate

Posle denormalizacije:



**User** (userId, name, *genderId*, *residenceId*)

**CreditCard** (creditCardNumber, *userId*, expiryDate, CVC, name)

**Funkcionalna zavisnost:**

userId, creditCardNumber -> CVC, expiryDate, name

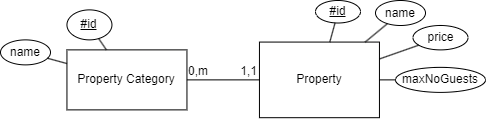
userId -> name

Specifikacija trigera:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabela** | **Tip trigera** | **Kolona** | **Potreban** | **Šta treba da uradi?** |
| **User** | Insert |  | NE |  |
| Update | name | DA | Prilikom izmene vrednosti name u tabeli User, pokreće se triger koji izmenjenu vrednost ažurira i u tabeli CreditCard. |
| Delete |  | NE |  |
| **CreditCard** | Insert |  | DA | Triger ažurira vrednost kolone name na osnovu unete vrednosti userId. |
| Update | userId | DA | Zabraniti direktno ažuriranje ove kolone. |
| name | DA | Zabraniti direktno ažuriranje ove kolone. |
| Delete |  | NE |  |

## Denormalizacija relacija uz narušavanje 3NF (Pre-joining)

Pre denormalizacije:



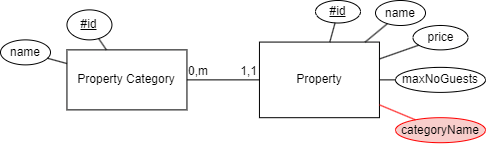
**Property** (propertyId, name, price, *propertyCategoryId, currencyId*, *areaId, hostId, locationId*)

**PropertyCategory** (propertyCategoryId, name)

**Funkcionalna zavistnost:**

propertyId -> name, price, maxNoGuests , categoryId

Posle denormalizacije:



**Property** (propertyId, name, price, *propertyCategoryId, currencyId*, *areaId, hostId, locationId,* categoryName)

**PropertyCategory** (propertyCategoryId, name)

**Funkcionalna zavisnost:**

propertyId -> name, maxNoGuests, categoryName, categoryId

propertyCategoryId -> categoryName

Specifikacija trigera:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabela** | **Tip trigera** | **Kolona** | **Potreban** | **Šta treba da uradi?** |
| **PropertyCategory** | Insert |  | NE |  |
| Update | categoryName | DA | Prilikom izmene vrednosti polja categoryName u tabeli PropertyCategory, pokreće se triger koji izmenjenu vrednost ažurira i u tabeli Property. |
| Delete |  | NE |  |
| **Property** | Insert |  | DA | Triger ažurira vrednost kolone categoryName na osnovu unete vrednosti atributa propertyCategoryId. |
| Update | propertyCategoryId | DA | Triger vrši ažuriranje kolone categoryName na osnovu izmenjene vrednosti kolone propertyCategoryId |
| categoryName | DA | Zabraniti direktno ažuriranje ove kolone. |
| Delete |  | NE |  |

# KORISNIČKI DEFINISANI TIPOVI I TRIGERI

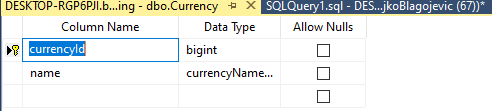
## Definicija korisničkom tipa i njegovo korišćenje

### Distinkt tip

Za primer distinkt tipa kreiran je tip CurrencyName koji se koristi unutar tabele Currency koja je data kroz sledeći relacioni model:

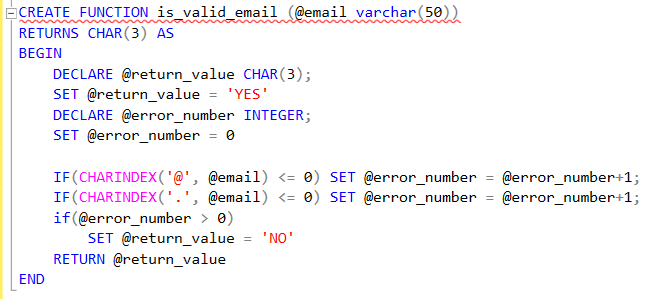
**Currency** (currencyId, name).





Pored toga je i dodata logika za validaciju email adrese koja se nalazi unutar relacije User:

**User** (userId, name, *genderId*, *residenceId*, email)



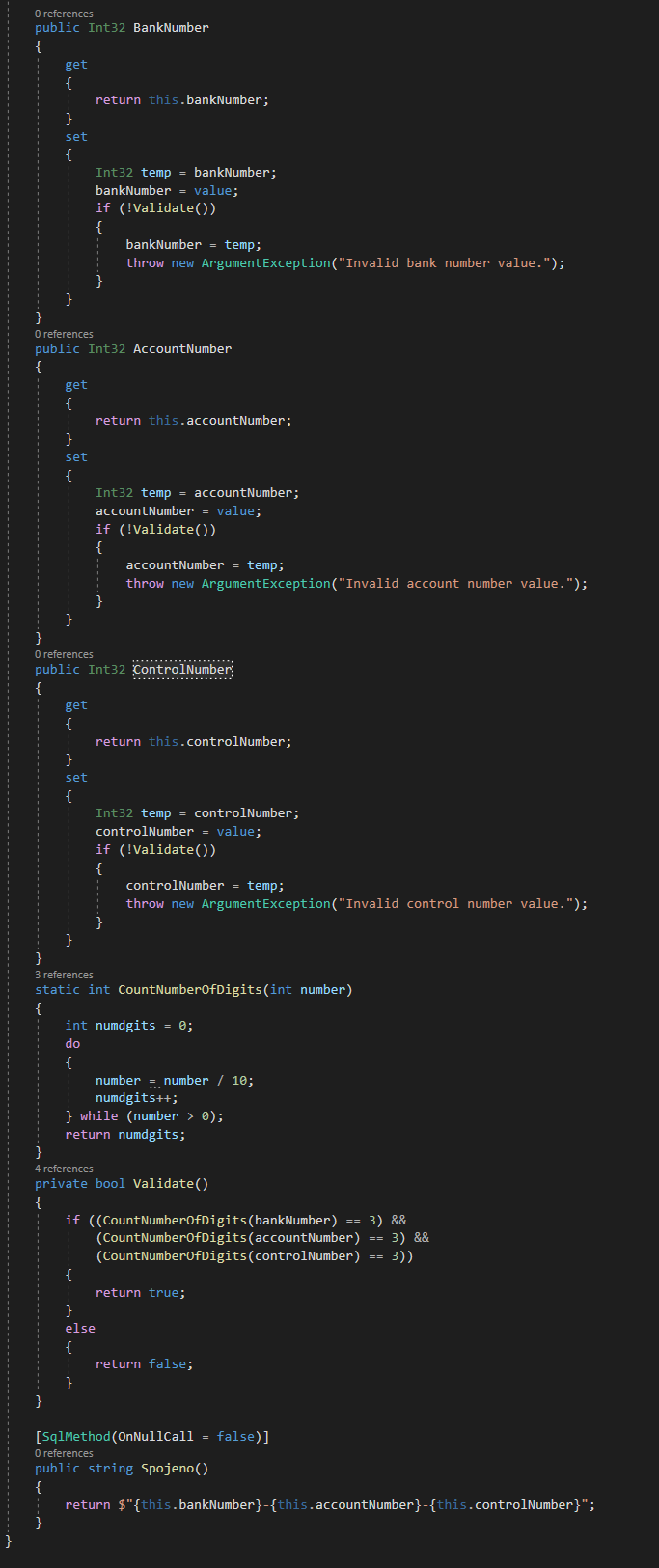
### Struktuirani tip

Za struktuirani tip je odabran broj kreditne kartice iz relacije CreditCard:

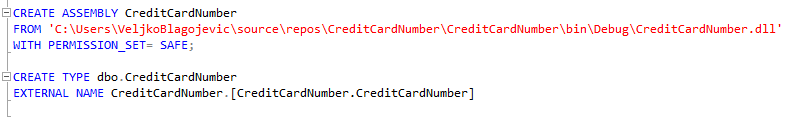
**CreditCard** (creditCardNumber, *userId*, expiryDate, CVC, placeholderName)

Prvo je potrebno kreirati strukturu kao .cs struct unutar .NET okruženja:





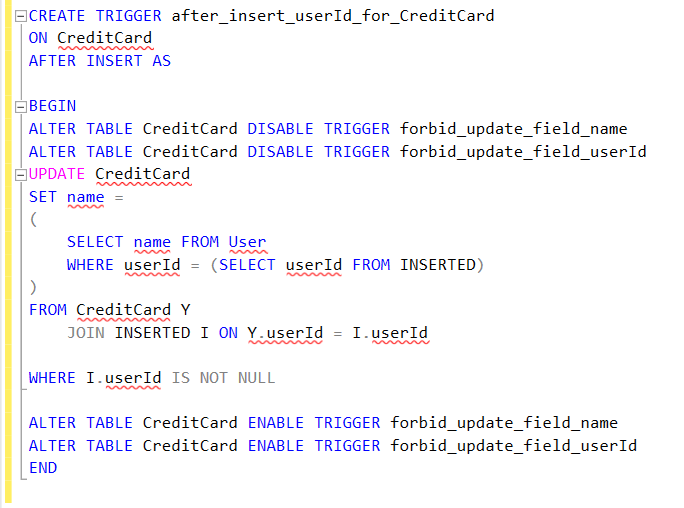
Posle toga se Class Library projekat builduje kako bi se kreirali .dll fajlovi, nakon čega se putem sledeće naredne u Microsoft SQL Server Management Studio programu kreira assembly i uvozi novi tip.

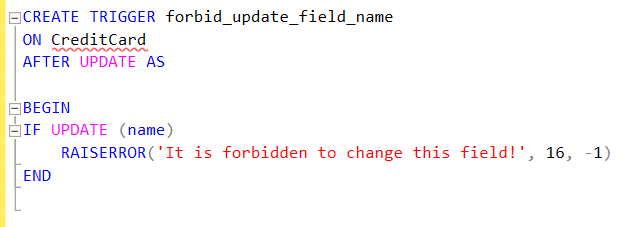


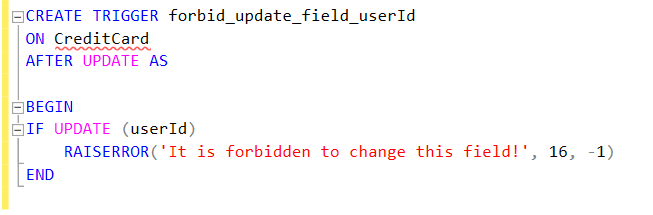
Naime je potrebno voditi računa o konfiguracionim parametrima baze kao što su clr strict security i njihova manipulacija sa sp\_configure.

## Realizacija proceduralne logike uz pomoć trigera

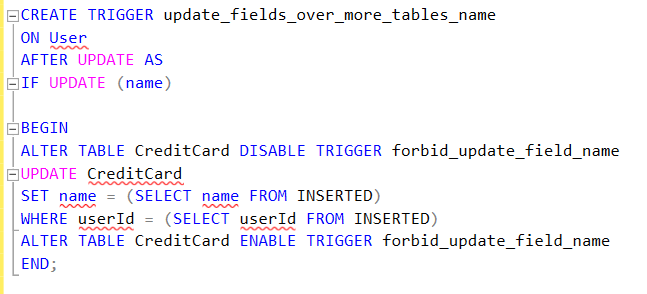
### Tabela CreditCard



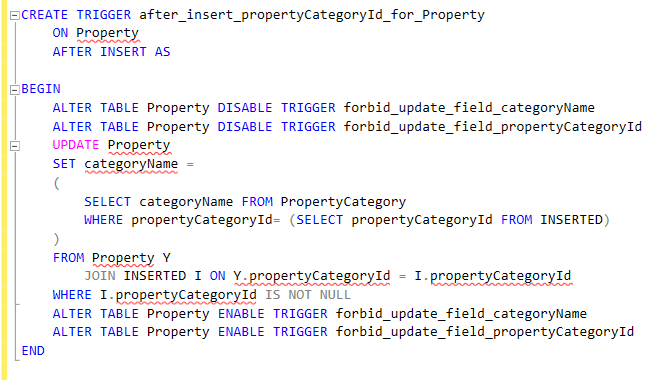


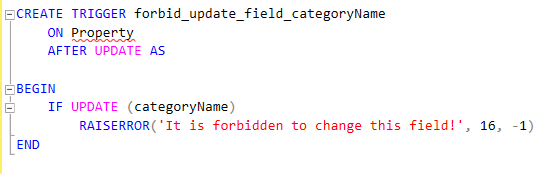


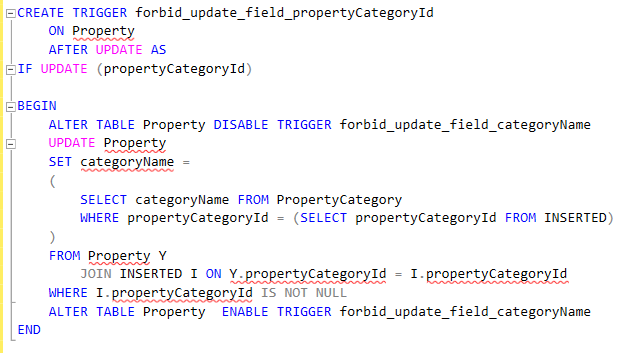
### Tabela User



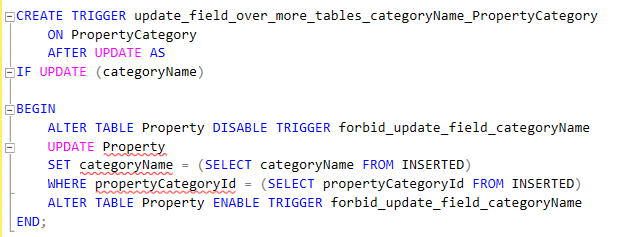
### Tabela Property







### Tabela PropertyCategory

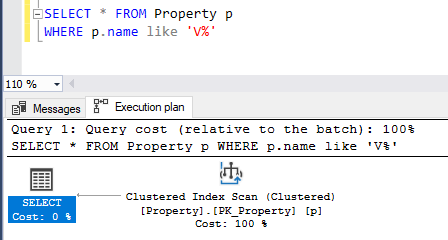


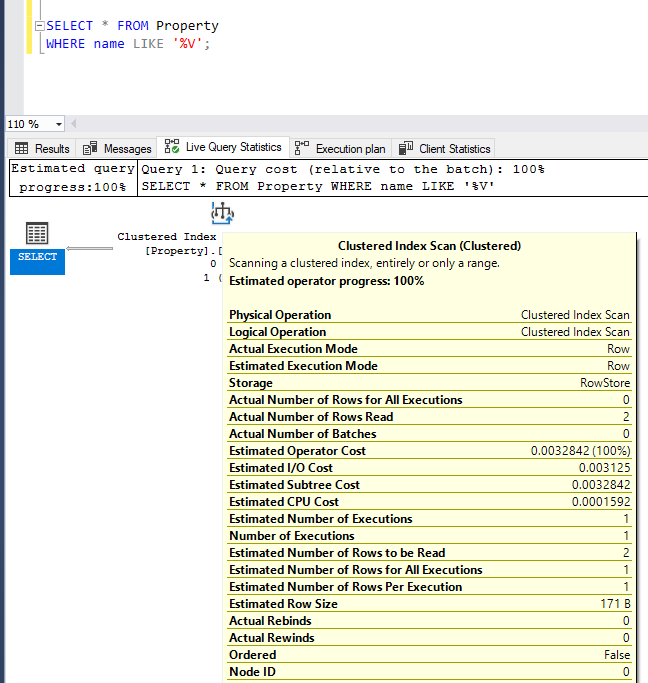
# OPTIMIZACIJA BAZE PODATAKA

## Definisanje i analiza indeksa

Kreiranje indeksa će se izvršiti nad atributom naziva unutar imovine jer će se po tome dosta pretraživati unutar same aplikacije.

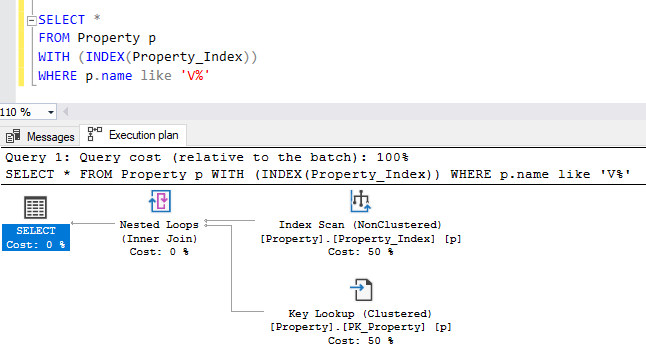
Na sledećoj slici je prikazan plan izvršavanja pre dodavanja bilo kakvog custom indeks rešenja.

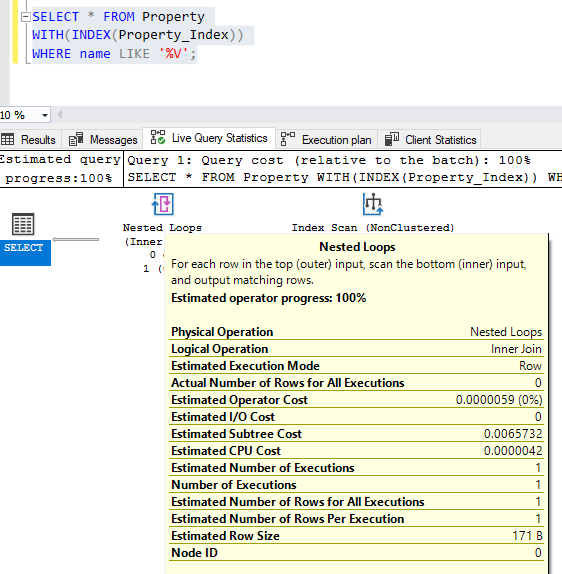




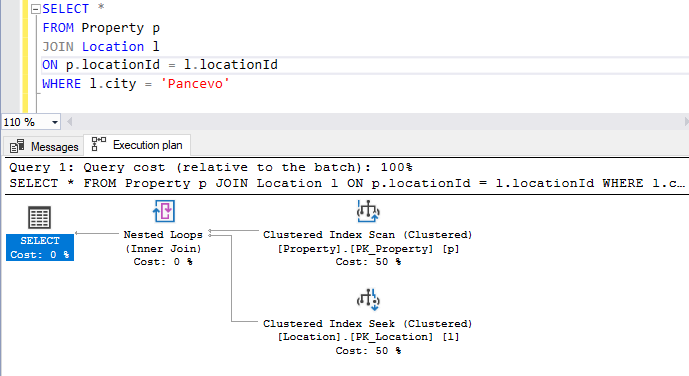


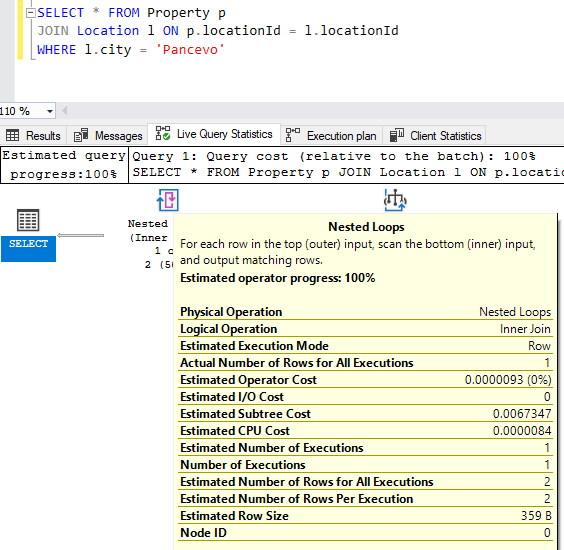
Dok je na sledećoj slici prikazan plan izvršenja nakon dodavanja indeksa.



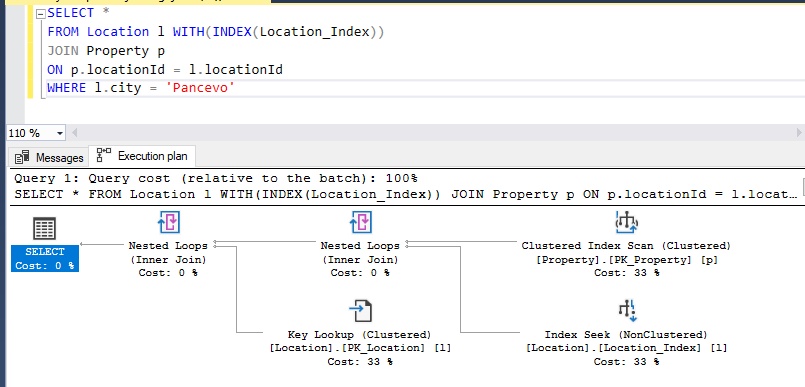


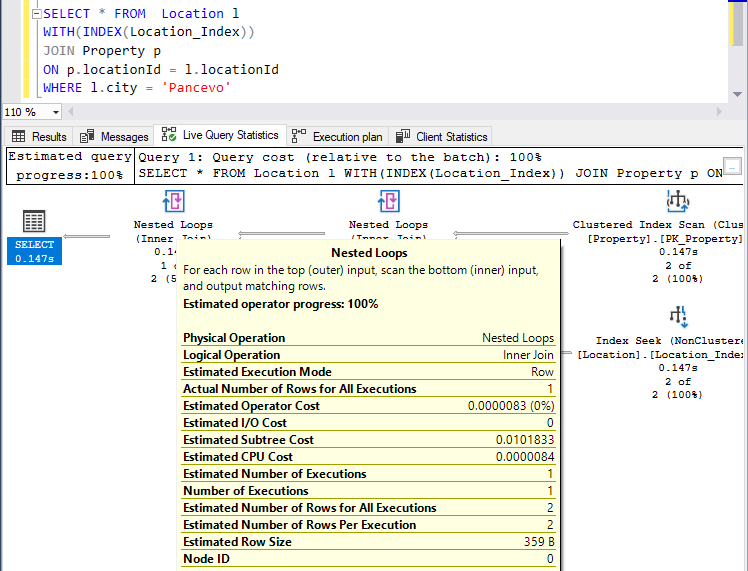
Takođe, drugo mesto na kome bi bilo korisno da se optimizuje baza podataka putem indeksa je u tabeli za Lokaciju. Naime, ova tabela, sa svojim primarnim ključem locationId, se javlja kao spoljni ključ u velikom broju ostalih često korišćenih tabela. A pogotovo pri pretrazi imovina gde korisnik može da filtrira po željenoj lokaciji.





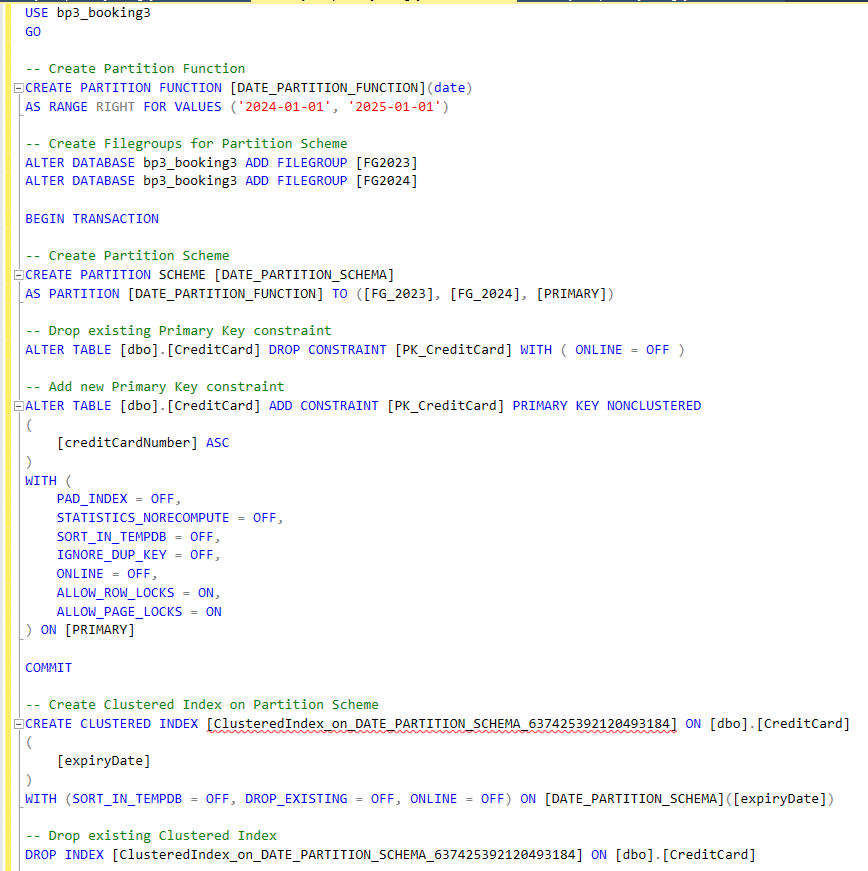




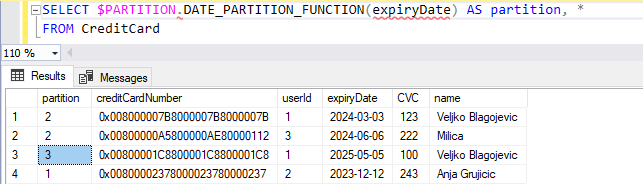


## Horizontalno particionisanje

Izvršava se horizontalno particionisanje nad tabelom CreditCard nad kolonom expiryDate gde se particije račvaju na osnovu godine.



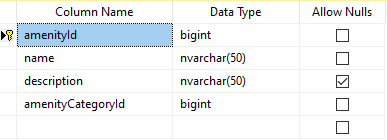
Pregled particija nad tabelom CreditCard se može videti sledećom naredbom



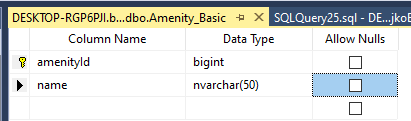
## Vertikalno particionisanje

Izvršava se vertikalno particionisanje nad tabelom Amenity:

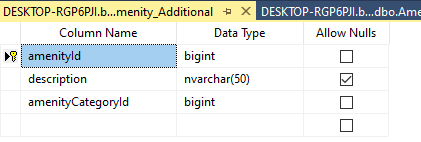
**Amenity** (amenityId, name, description, *amenityCategoryId*)



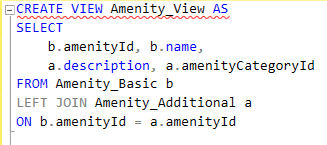
Delimo je na dve tabele, prve Basic sa osnovnim informacijama.



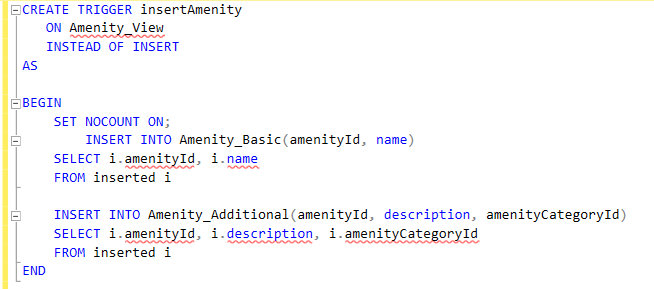
I na drugu Additional sa dodatnim podacima

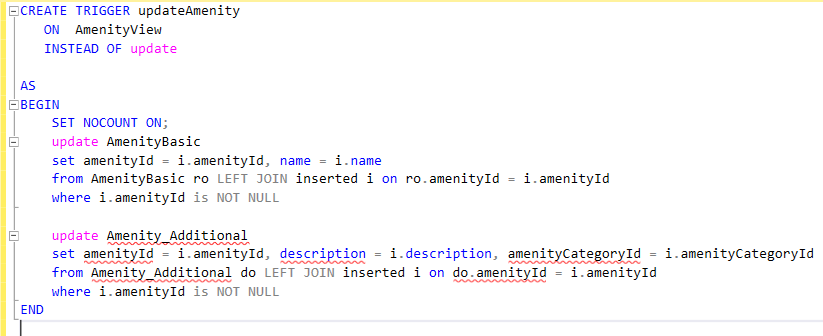


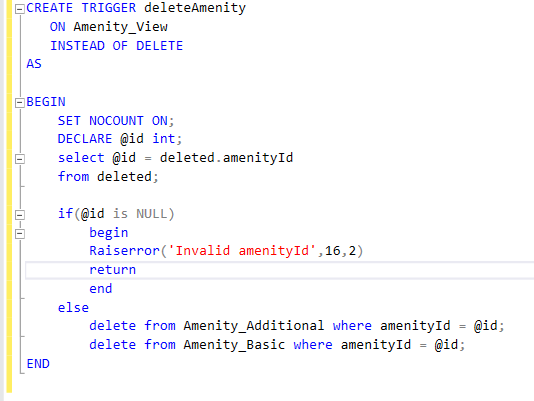
Pogled koji vraća sve podatke iz obe tabele kombinovano se kreira na sledeći način:



U nastavku su dati neophodni trigeri za rad sa tabelama nad kojima je urađen proces podele na vertikalne particije.



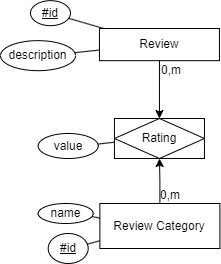




## Primena drugih optimizacionih tehnika

### Storing Derivable Values

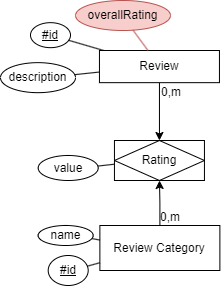
Pre optimizacije:



**Review** (reviewId, description, *startDate, guestId, propertyId*)

**Rating** (*reviewId, reviewCategoryId*, value)

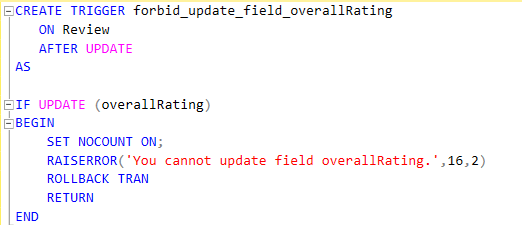
Posle optimizacije:

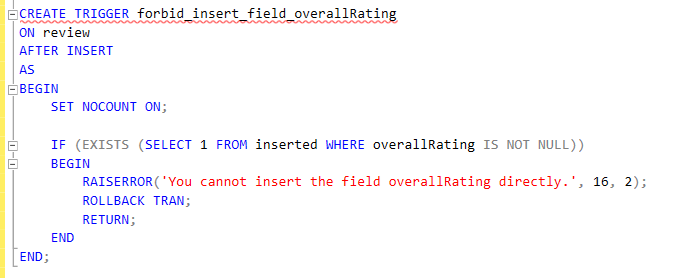


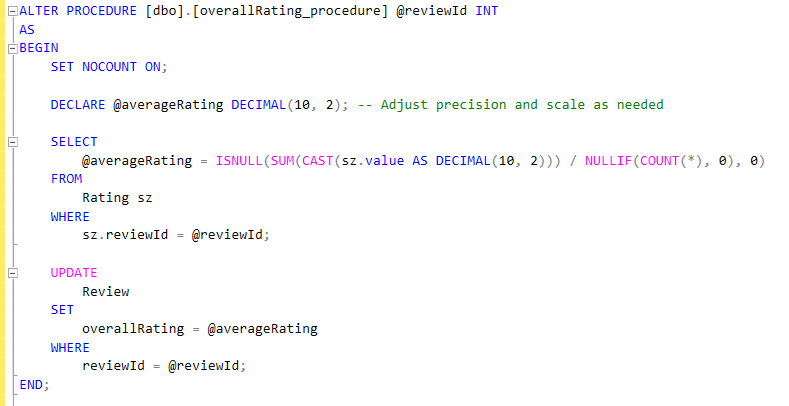
**Review** (reviewId, description, *startDate, guestId, propertyId,* overallRating)

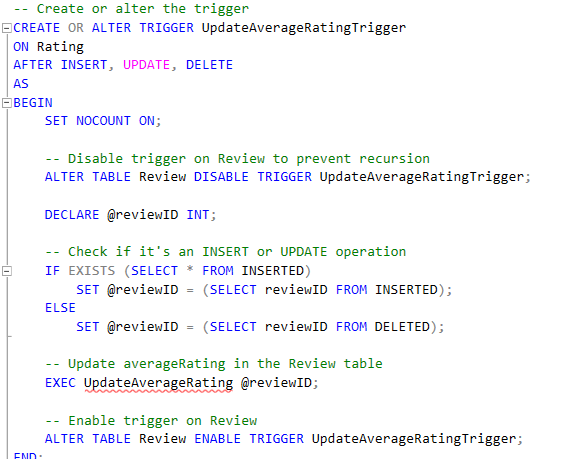
**Rating** (*reviewId, reviewCategoryId*, value)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabela** | **Tip trigera** | **Kolona** | **Potreban** | **Šta treba da uradi?** |
| **Review** | Insert |  | NE |  |
| Update | overallRating | DA | Zabraniti direktno ažuriranje ove kolone. |
| Delete |  | NE |  |
| **Rating** | Insert |  | DA | Prilikom unosa nove stavke okida se triger koji poziva procedure za ažuriranje polja overallRating unutar tabele Review. |
| Update | value | DA | Prilikom ažuriranja kolone value okida se triger koji poziva proceduru za ažuriranje polja overallRating tabele Review. |
| Delete |  | DA | Prilikom brianja stavke okida se triger koji poziva proceduru za ažuriranje polja overallRating tabele Review. |



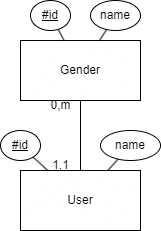






### Hard Coded Values

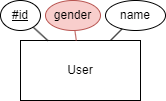
Pre optimizacije:



**Gender** (genderId, name)

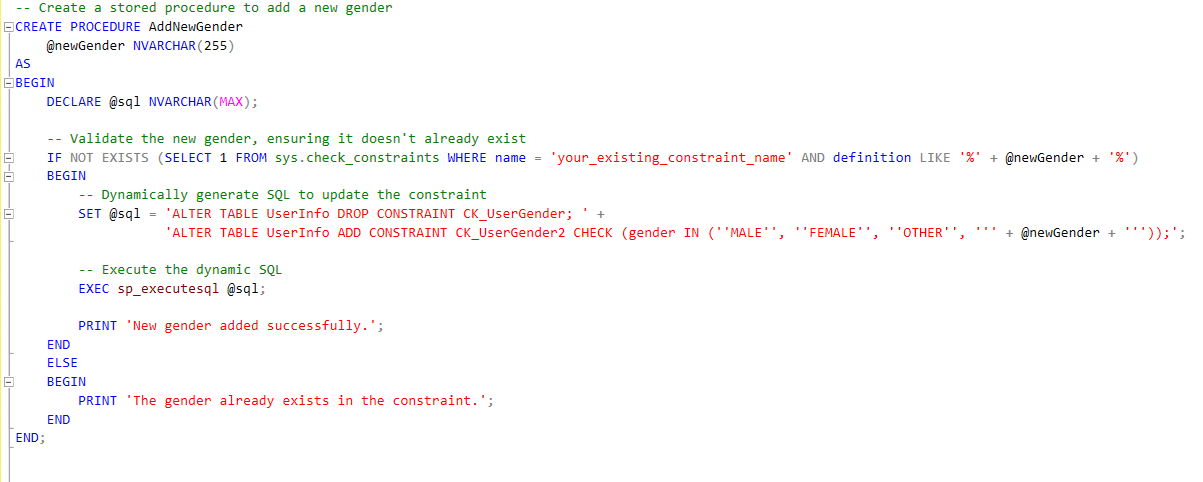
**User** (userId, name, *genderId*, *residenceId*, email)

Posle optimizacije:



**User** (userId, name, *genderId*, *residenceId*, email, gender)

Sa CHECK CONSTRAINT ograničenjem za dozvoljene vrednosti MALE, FEMALE, OTHER. Pri čemu je i napisana uskladištenja procedura čime se dozvoljava korisniku da prošiti ograničenje sa novim polovima.



# TEHNOLOGIJE ZA IMPLEMENTACIJU PROJEKTA

## SUBP korišćen za implementaciju baze podatka (kratak opis)

SQL Server Management Studio (SSMS) je alat koji pruža interfejs za upravljanje i administraciju Microsoft SQL Server bazama podataka. Ovaj alat omogućava različite zadatke, uključujući upravljanje bazama podataka, izvršavanje T-SQL upita, dizajniranje baza podataka, upravljanje sigurnošću i još mnogo toga. Evo nekoliko ključnih karakteristika i funkcija koje možete pronaći u SSMS:

1. **Povezivanje sa Serverom** SSMS vam omogućava povezivanje sa SQL Server instancama. Možete se povezati lokalno ili udaljeno, koristeći Windows autentifikaciju ili SQL Server autentifikaciju.

2. **Object Explorer:** Object Explorer prikazuje strukturu SQL Servera, uključujući baze podataka, tabele, procedure, funkcije, indekse i druge objekte. Odatle možete lako upravljati i pregledavati objekte.

3. **Query Editor**: Query Editor omogućava pisanje, testiranje i izvršavanje T-SQL upita. Pruža bogatu funkcionalnost za rad s upitima, uključujući bojenje sintakse, podršku za IntelliSense i praćenje izvršenja upita.

4. **Dizajner baze podataka:** SSMS uključuje vizuelni alat za dizajniranje baza podataka. Ovaj alat omogućava kreiranje tabela, veza između tabela, indeksa i drugih struktura baza podataka pomoću grafičkog korisničkog interfejsa.

5. **Sigurnosne opcije:** SSMS omogućava upravljanje sigurnošću SQL Servera, uključujući dodavanje i upravljanje korisničkim nalozima, dodelu prava i praćenje sigurnosnih događaja.

6. **Plan izvršavanja upita:** Prilikom izvršavanja upita, SSMS vam omogućava pregled plana izvršavanja upita, što vam pomaže u analizi performansi upita.

7. **Upravljanje SQL Server Agentom:** SQL Server Agent omogućava zakazivanje i automatizaciju zadataka, poput izvršavanja skladištenih procedura ili backup-a baze podataka.

SSMS je ključan alat za svakog SQL Server administratora ili developera koji radi s Microsoft SQL Server bazama podataka. Ove funkcije čine SSMS snažnim i fleksibilnim alatom za efikasno upravljanje SQL Server okolinom.

## Programsko okruženje za razvoj korisničkog interfejsa (kratak opis)

Spring Boot, JdbcTemplate i React su popularni alati i tehnologije u svetu razvoja softvera, posebno za izradu web aplikacija. Evo kratak pregled svake od ovih tehnologija:

1. **Spring Boot:** radni okvir (framework) za razvoj Java aplikacija koji pojednostavljuje proces konfiguracije i izgradnje Spring aplikacija.

- Brza konfiguracija: Automatsko konfigurisanje zavisnosti, što olakšava početak rada.

- Ugrađeni server: Dolazi s ugrađenim servlet kontejnerom (npr. Tomcat), što smanjuje potrebu za eksternim serverom.

- Mikroservisi: Podržava razvoj mikroservisa i jednostavno upravljanje konfiguracijom.

2. **JdbcTemplate:** deo Spring Framework-a i pruža jednostavan način za izvršavanje SQL upita i ažuriranje baza podataka.

- Jednostavnost: Smanjuje složenost JDBC (Java Database Connectivity) koda, pružajući šablone i upitnike.

- Bezbednost: Pomaže u prevenciji SQL injekcija i smanjuje potrebu za ručnim upravljanjem resursima.

3. **React:** React je JavaScript biblioteka za izgradnju korisničkih interfejsa, posebno za jednostrano (single-page) web aplikacije.

- Virtuelni DOM: Efikasno upravljanje prikazom putem virtuelnog DOM-a, što dovodi do poboljšane performanse.

- Komponentna arhitektura: Razdvajanje korisničkog interfejsa u ponovno upotrebljive komponente olakšava održavanje i razvoj.

a. **REST API**: Spring Boot može se koristiti za kreiranje RESTful API-ja koji komunicira s bazom podataka pomoću JdbcTemplate-a.

b. **React Frontend:** React se može koristiti za izgradnju korisničkog interfejsa koji konzumira podatke putem API-ja koje pruža Spring Boot.

c. **Komunikacija:** Frontend i backend komuniciraju putem HTTP zahteva, obično u JSON formatu.

- React komponente mogu se dinamički osvežavati sa podacima dobijenim od Spring Boota.

d. **Alati za izgradnju:** Za upravljanje zavisnostima i izgradnju projekta možete koristiti alate poput Mavena (za backend) i npm-a (za frontend).

Integracija Spring Boot i JdbcTemplate sa React-om pruža snažan skup alata za razvoj modernih web aplikacija, omogućavajući efikasan rad sa serverom, bazom podataka i korisničkim interfejsom.

Ceo izvorni kod projekta se može naći na sledećem GitHub repozitorijumu:  
https://github.com/VeljkoBlagojevic/airbnb