Za implementaciju smo odabrali kreacijske patern builder i singleton.

Singleton

Singleton patern je kreacijski obrazac koji osigurava da određena klasa može imati samo jednu instancu u okviru aplikacije. Kroz globalnu tačku pristupa omogućava kontrolisano i dosljedno korištenje te instance. Ovaj pristup se koristi u slučajevima kada bi višestruke instance dovele do nelogičnosti u radu sistema, kao što su servisi za logovanje, slanje notifikacija ili upravljanje konfiguracijom.

Potreba u našem sistemu?

U sistemu za rentanje vozila, Singleton patern se koristi za servis za slanje notifikacija i za upravljanje konekcijom prema bazi podataka. Servis za notifikacije treba biti jedinstven kako bi se centralizovalo slanje poruka iz različitih dijelova sistema i osigurala dosljednost. Slično tome, konekcija prema bazi zahtijeva kontrolisan pristup kroz jednu instancu kako bi se spriječilo prekomjerno otvaranje veza i omogućila efikasna komunikacija sa bazom.

Prototype

Prototype je kreacijski dizajn šablon koji omogućava kreiranje novih objekata kloniranjem postojećih, umjesto njihovog ponovnog instanciranja pomoću konstruktora. Ovaj pristup je posebno koristan kada je inicijalizacija objekta složena ili kada je potrebno više objekata sa sličnim svojstvima. Korištenjem Prototype šablona povećava se efikasnost, smanjuje dupliranje koda i omogućava jednostavno kreiranje kopija objekata s minimalnim izmjenama.

Potreba u našem sistemu?

U sistemu za rentanje vozila, Prototype patern se može primijeniti na klasu ZahtjevZaRentanje kako bi se omogućilo kloniranje postojećeg zahtjeva prilikom kreiranja novog. Na taj način korisnik može brzo napraviti novi zahtjev sličan prethodnom, bez ponovnog unosa svih podataka, već uz minimalne izmjene. Ovo rješenje povećava efikasnost i pojednostavljuje rad sa zahtjevima.

Factory Method

Factory Method definiše interfejs za kreiranje objekta, ali odluku o tome koji konkretan objekat će biti instanciran prepušta podklasama. Na taj način se postiže fleksibilnost i olakšava proširivanje sistema novim tipovima objekata.

Potreba u našem sistemu?

U sistemu za rentanje vozila, Factory Method patern se može koristiti za kreiranje različitih tipova vozila, kao što su SUV, ATV ili Buggy. Umjesto da se objekti vozila instanciraju direktno, koristi se fabrika koja, na osnovu zadatog tipa, vraća odgovarajući objekat. Ovo olakšava dodavanje novih tipova vozila bez izmjene postojećeg koda.

Abstract Factory

Abstract Factory omogućava kreiranje grupa povezanih objekata bez navođenja konkretnih klasa. Umjesto toga, koristi se apstraktna fabrika koja definira interfejse za kreiranje, a konkretne fabrike određuju koje tačno objekte vraćaju.

Potreba u našem sistemu?

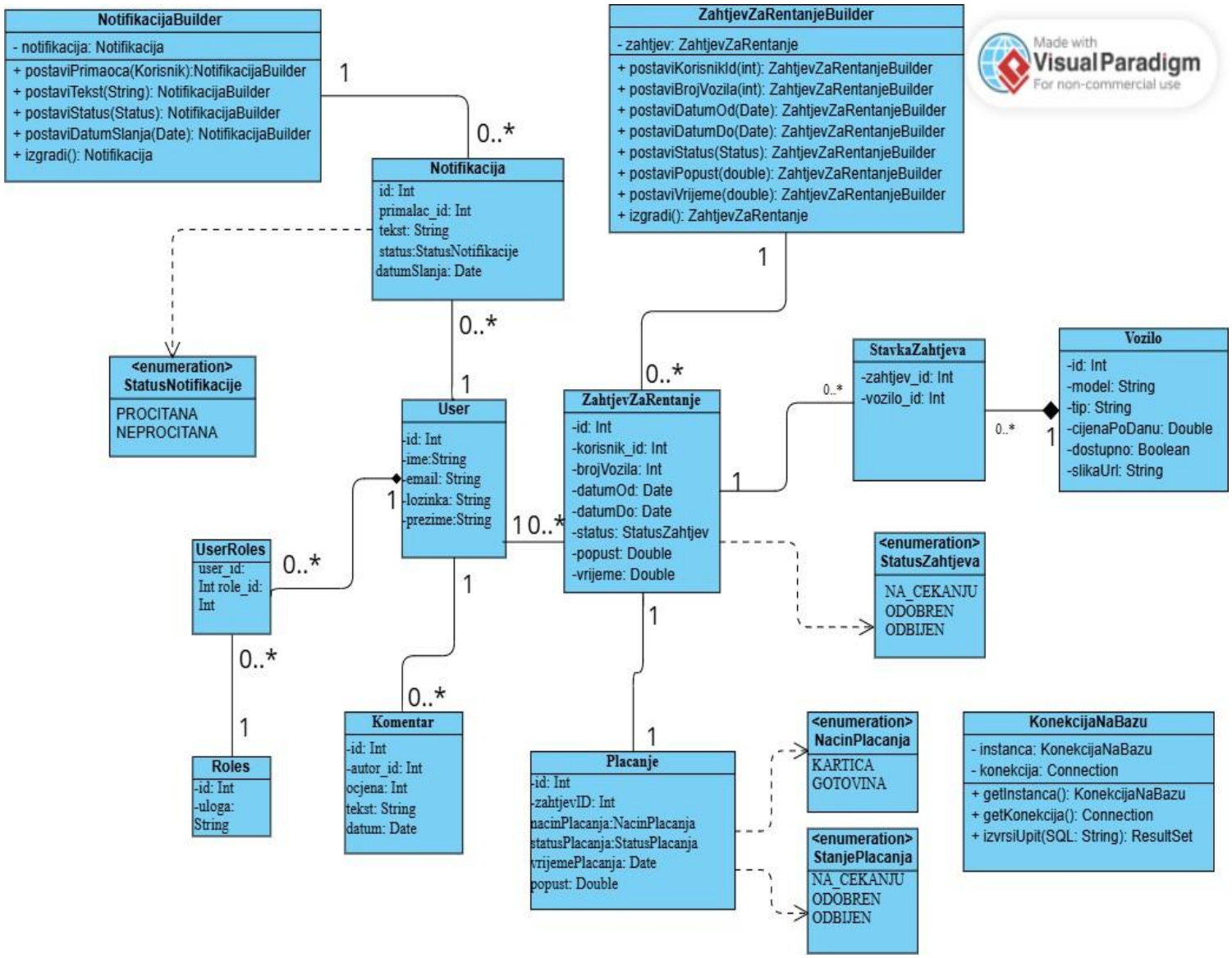
U sistemu za rentanje vozila, Abstract Factory se može primijeniti prilikom kreiranja različitih paketa ponuda, kao što su standardni i premium paket. Svaka fabrika može kreirati grupu međusobno povezanih objekata – vozilo, dodatnu opremu i popust – bez navođenja konkretnih klasa. Na ovaj način se postiže dosljednost među objektima unutar jednog paketa i olakšava proširivanje sistema dodavanjem novih vrsta ponuda.

Builder

Razdvaja konstrukciju kompleksnog objekta od njegove reprezentacije tj. omogućava pravljenje objekta korak po korak.

Potreba u našem sistemu?

Builder patern je u ovom sistemu primijenjen na klase Notifikacija i ZahtjevZaRentanje, gdje je bilo potrebno odvojiti proces kreiranja objekta od njegove reprezentacije. Oba objekta sadrže više polja koja se ne popunjavaju uvijek u istom trenutku i istim redoslijedom.
Korištenjem Builder šablona omogućeno je postepeno i fleksibilno sastavljanje ovih objekata, bez potrebe za velikim konstruktorima ili višestrukim set metodama. Na ovaj način povećana je čitljivost i pouzdanost koda, a kreiranje složenih objekata je postalo jednostavnije i
kontrolisanije.


NotifikacijaBuilder
- notifikacija: Notifikacija
+ postaviPrimaoca(Korisnik):NotifikacijaBuilder
+ postaviTekst(String): NotifikacijaBuilder
+ postaviStatus(Status): NotifikacijaBuilder
+ postaviDatumSlanja(Date): NotifikacijaBuilder
+ izgradi(): Notifikacija
Notifikacija
id: Int
primalac_id: Int
tekst: String
status:StatusNotifikacije
datumSlanja: Date
<enumeration>
StatusNotifikacije
PROCITANA
NEPROCITANA
ZahtjevZaRentanjeBuilder
- zahtjev: ZahtjevZaRentanje
+ postaviKorisnikId(int): ZahtjevZaRentanjeBuilder
+ postaviBrojVozila(int): ZahtjevZaRentanjeBuilder
+ postaviDatumOd(Date): ZahtjevZaRentanjeBuilder
+ postaviDatumDo(Date): ZahtjevZaRentanjeBuilder
+ postaviStatus(Status): ZahtjevZaRentanjeBuilder
+ postaviPopust(double): ZahtjevZaRentanjeBuilder
+ postaviVrijeme(double): ZahtjevZaRentanjeBuilder
+ izgradi(): ZahtjevZaRentanje
Made with VisualParadigm For non-commercial use
User
-id: Int
-ime:String
-email: String
-lozinka: String
-prezime:String
ZahtjevZaRentanje
-id: Int
-korisnik_id: Int
-brojVozila: Int
-datumOd: Date
-datumDo: Date
-status: StatusZahtjev
-popust: Double
-vrijeme: Double
StavkaZahtjeva
-zahtjev_id: Int
-vozilo_id: Int
Vozilo
-id: Int
-model: String
-tip: String
-cijenaPoDanu: Double
-dostupno: Boolean
-slikaUrl: String
<enumeration>
StatusZahtjeva
NA_CEKANJU
ODOBREN
ODBIJEN
UserRoles
user_id:
Int role_id:
Int
Roles
-id: Int
-uloga:
String
Komentar
-id: Int
-autor_id: Int
ocjena: Int
tekst: String
datum: Date
Placanje
-id: Int
-zahtjevID: Int
nacinPlacanja:NacinPlacanja
statusPlacanja:StatusPlacanja
vrijemePlacanja: Date
popust: Double
<enumeration>
NacinPlacanja
KARTICA
GOTOVINA
<enumeration>
StanjePlacanja
NA_CEKANJU
ODOBREN
ODBIJEN
KonekcijaNaBazu
- instanca: KonekcijaNaBazu
- konekcija: Connection
+ getInstanca(): KonekcijaNaBazu
+ getKonekcija(): Connection
+ izvrsiUpit(SQL: String): ResultSet