SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Tracktor – opis O/R mapiranja

Leonard Volarić Horvat, Jan Kelemen, Marko Stanić, Roko Zubčić

Voditelj: Matija Herceg

Zagreb, veljača, 2017

**Sadržaj**

[1. Opis domene 1](#_Toc473906158)

[2. Model domene 3](#_Toc473906159)

[3. Objektni model 4](#_Toc473906160)

[4. Relacijski model 5](#_Toc473906161)

[5. O/R mapiranje 6](#_Toc473906162)

[5.1 Problem O/R mapiranja 6](#_Toc473906163)

[5.2 Infrastruktura 6](#_Toc473906164)

[5.3 Unit of Work 9](#_Toc473906165)

[5.4 Entity framework 9](#_Toc473906166)

# Opis domene

Sustav za dijeljenje informacija baziranih na trenutnoj lokaciji omogućava korisniku unos podataka o nekom događaju. Za korištenje aplikacije korisnici moraju kontaktirati administratora kako bi dobili podatke za prijavu. Aplikacija podržava dvije vrste korisnika: regularne i *premium*.

Regularni korisnici aplikaciju koriste preko web i mobilnog sučelja. Za korištenje je nužna autentikacija korisničkim imenom i lozinkom. Regularni korisnici mogu unositi podatke o trenutnim događajima na njihovim lokacijama. Događaji mogu pripadati jednoj od kategorija koje su definirane u sustavu. Za svaki se događaj bilježi lokacija u obliku koordinata - zemljopisna širina i dužina - te naziv mjesta ili objekta u kojem se događaj zbiva. Također, za svaki se događaj bilježi preostalo vrijeme trajanja, kao i informacija u obliku teksta koja pobliže opisuje isti. Korisnici mogu pretraživati događaje za odabranu lokaciju te mogu koristiti uvjete pretrage za filtriranje sadržaja u vidu odabira jedne ili više kategorija kojima mogu pripadati događaji koji ih zanimaju. Također mogu komentirati i ocjenjivati događaje te mogu ocjenjivati i komentare ostalih korisnika. U nekom trenutku korisniku su vidljivi isključivo podaci o događajima koje su unijeli drugi regularni korisnici, a koji su aktivni, odnosno, kojima nije isteklo vrijeme trajanja, te podaci o onim događajima koji još nisu aktivni, ali su ih unijeli premium korisnici te su kao takvi u najavi. Korisniku se geografski, na karti, prikazuju lokacije događaja koji zadovoljavaju zadani uvjet pretrage. Korisnik ima mogućnost kreiranja liste svojih najdražih mjesta.

*Premium* korisnici aplikaciju koriste preko web i mobilnog sučelja te se trebaju autenticirati korisničkim imenom i lozinkom. Po uspješnoj prijavi na sustav, premium korisnik može aplikaciju koristiti na način koji je koristi i regularni korisnik uz dodatne mogućnosti - *premium* korisnik može:

* najavljivati događaje, tj. unositi informacije o događajima koji se tek trebaju dogoditi u budućnosti,
* unositi podatke o pokroviteljstvu nad nekim mjestima.

Administrator aplikaciju koristi isključivo preko desktop sučelja. Za uspješno korištenje aplikacije treba se autenticirati korisničkim imenom i lozinkom. Administrator može:

* mijenjati i brisati informacije o događajima (neovisno o vrsti korisnika koji ih je unio),
* mijenjati i brisati komentare na događaje,
* unositi informacije o događajima, komentirati i ocjenjivati ostale događaje unesene u sustav,
* za svakog korisnika ažurirati njegov tip,
* uređivati podatke o kategorijama događaja.

# Model domene

U modelu domene nalaze se klase koje opisuju objekte iz domene problema. To su uglavnom entiteti koji predstavljaju korisnike, informacije, mjesta, komentare i ostale relevantne objekte. Svaki od njih ima svoj jedinstveni ID, a međusobno su povezani, gdje je to potrebno, stranim ključevima.



Modeli za korisnike su UserEntity i UserTypeEntity, klasa UserTypeEntity služi za određivanje tipa korisnika (administrator, regularni ili premium), a klasa UserEntity sadrži sve podatke o korisniku. Modeli koji se koriste za mjesta su PlaceEntity te GeoCoordinate, klasa PlaceEntity modelira mjesto, a klasa GeoCoordinate je value objekt koji predstavlja lokaciju mjesta kao geografske koordinate. Modeli koji se koriste za informacije su InfoEntity i CommentEntity koji modeliraju informaciju i komentare za informaciju, uz ova dva modela postoje klase ReputationInfoEntity i ReputationCommentEntity koji služe za pamćenje glasova za informaciju ili komentar na informaciju.

# Objektni model

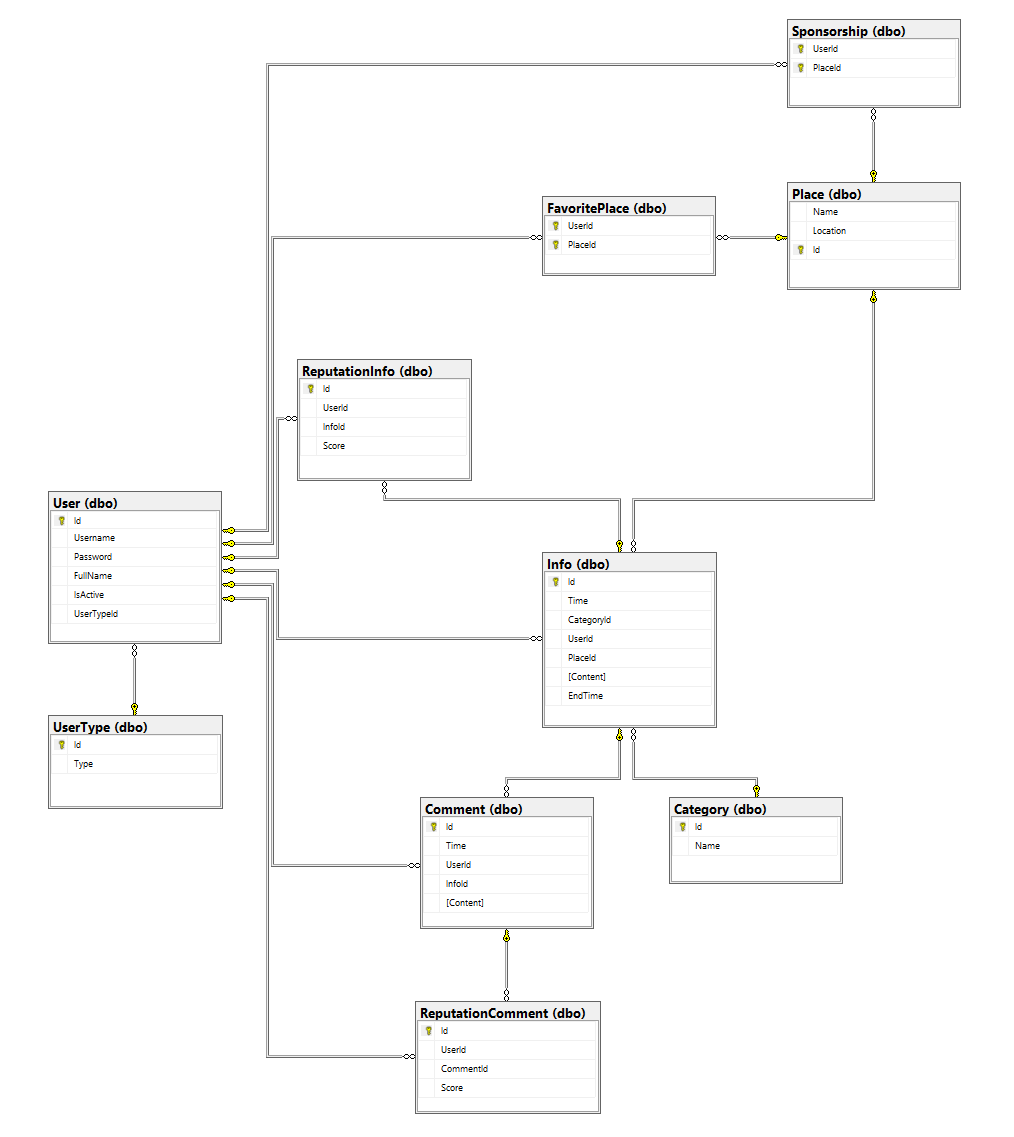
Objekti modela domene stvaraju se u sloju za pristup podacima od objektnog modela baze podataka, klase tog objektnog modela prikazane su u nastavku. Jedna klasa ovog modela u pravilu predstavlja jednu tablicu iz baze podataka.



Objektni model baze podataka generiran je iz sheme baze podataka pomoću čarobnjaka. Ovaj model sličan je modelu domene uz neka manja odstupanja, na primjer ono što je u modelu domene klasa GeoCoordinate to je u ovom modelu jedan atribut klase Place čiji je tip DbGeography, zbog ovakvih pojedinosti koje su vezane uz sam način perzistencije, dobro je da je model domene odvojen od načina perzistencije jer se model domene ne opterećuje detaljima perzistencije podataka.

# Relacijski model

Relacijski model podataka dobiven je analizom modela domene, odnosno zahtjeva za sustav i iz toga je napravljena relacijska shema baze podatka. Relacijska shema baze podataka prikazana je ER dijagramom u nastavku.



# O/R mapiranje

O/R mapiranje implementirano je u Data Access sloju (DAL) aplikacije u ovome sloju nalazi se objektni model i popratna infrastruktura sloja za pristup podacima, odnosno repozitoriji i UnitOfWork klasa koja implementira Unit of Work oblikovni obrazac.

## Problem O/R mapiranja

Problemi s O/R mapiranjem uglavnom proizlaze iz različitih teorijskih temelja objektne i relacijske paradigme. Neki od problema su problem granularnosti, problem mapiranja hijerarhije klasa, problem identiteta. Kako bi izbjegli neke od ovih problema i olakšali rad s bazom podataka koristimo neki od O/R mapping frameworka, u ovome projektu korišten je Entity Framework.

## Infrastruktura

Kako bi se višim slojevima aplikacije omogućilo da rade isključivo s modelima domene, u sloju za pristup podacima postoje repozitoriji koji u svojem sučelju rade samo s modelima domene.



Svi repozitoriji su izvedeni iz baznog razreda *EFRepository* i njih kreira *UnitOfWork* objekt te im kao parametar predaje *DbContext* objekt. Repozitoriji primaju objekte iz domenskog sloja koje pomoću *ModelMapper* objekta preslikavaju u njihove reprezentacije u DAL sloju. Preslikavanje je nužno jer razredi u domenskom sloju nisu anemični već posjeduju i nekakve metode (npr. za izračunavanje reputacije nekog događa ili komentara). Objekti razreda koji implemetiraju *DbContext* sučelje (a u našem je to slučaju instanca razreda *TracktorDb*) prate promjene na svim objektima u objektnom modelu te se prilikom poziva metode *SaveChanges*() te promjene evidentiraju u bazi podataka.

Primjeri nekih metoda razreda PlaceRepository dani su u nastavku.

public int Insert(PlaceEntity placeDomain, Action saveChanges)

{

Place placeDAL = Mapper.ToDALModel(placeDomain);

DbSet.Add(placeDAL);

saveChanges();

return placeDAL.Id;

}

Metoda *Insert()* prima objekt domenskog sloja (parametar *placeDomain*) i delegat na void funkciju (parametar *saveChanges*) za spremanje promjena u bazu koje *DbContext* objekt prati. Drugi parametar je potreban jer na taj način *Entity* *Framework* osvježi *placeDAL* objekt, odnosno dodijeli mu iz baze generirani *Id*. Taj *Id* je ujedno i povratna vrijednost ove funkcije.

public IEnumerable<PlaceEntity> GetMany(IDictionary<string, bool> filters,

bool active,

bool future)

{

//Dohvat svih aktivnih i buducih dogadjaja

List<Info> activeInfos = this.Context.Set<Info>().Where(i => i.Time < DateTime.Now && i.EndTime > DateTime.Now).ToList();

List<Info> futureInfos = this.Context.Set<Info>().Where(i => i.Time > DateTime.Now).ToList();

//filter po active i future uvjetima

List<Info> filterInfo = new List<Info>();

if (active)

filterInfo.AddRange(activeInfos);

if (future)

filterInfo.AddRange(futureInfos);

//filter po kategorijama

List<Info> filtered = new List<Info>();

foreach(var category in filters)

{

if(category.Value)

filtered.AddRange(

(from info in filterInfo

where info.Category.Name.Equals(category.Key)

select info).ToList()

);

}

List<Place> placesDAL = filtered.Select(i => i.Place).ToList();

placesDAL = placesDAL.GroupBy(p => p.Id).Select(grp => grp.First()).ToList();

List<PlaceEntity> placesDomain = new List<PlaceEntity>();

foreach (var place in placesDAL)

{

placesDomain.Add(Mapper.ToDomainModel(place));

}

return placesDomain;

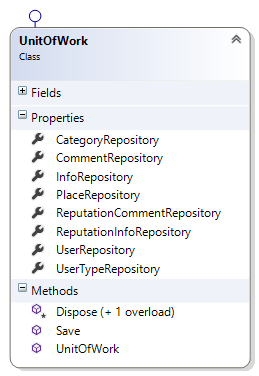
}

Metoda *GetMany()* kao parametar prima rječnik uvjeta za filtriranje. Svaki uvjet je predstavljen kategorijom i *bool* vrijednošću koja označava uzimaju li se u obzir događaji koji pripadaju toj kategoriji ili ne. Parametri *active* i *future* označavaju koje događaje treba uzeti u obzir (one koji se trenutno odvijaju ili događaje u najavi). Iz konteksta se dohvaćaju događaji te se za svaki provjerava zadovoljava li uvjet filtriranja po kategorijama. Za svaki od događaja koji zadovoljava uvjet, njegovo mjesto događanja se sprema u *placeDAL* varijablu. Ta mjesta se uz pomoć *Mapper* objekta preslikavaju u entitete iz domenskog sloja i spremaju u listu (*placesDomain*). Ta lista je ujedno i povratna vrijednost iz ove metode.

## Unit of Work

Servisi iz poslovnog sloja komuniciraju s repozitorijima preko razreda *UnitOfWork*. *UnitOfWork* ima referencu na svaki od repozitorija u vidu privatnih članskih varijabli. Njima se pristupa preko sučelja odnosno javnih svojstava. Repozitoriji se instanciraju samo prvi put kad ih neki drugi objekt zatraži. Prilikom kreiranja repozitorija, njemu se u konstruktor prosljeđuje objekt tipa *TracktorDb* odnosno *DbContext*. *UnitOfWork* može biti instanciran na način da mu se kroz konstruktor proslijedi *DbContext* ili da ga on sam kreira. Također, razred *UnitOfWork* implementira *IDisposable* sučelje.

Svaki od kreiranih repozitorija ima referencu na objekt tipa *TracktorDb* s kojim rade. Sve promjene koje pojedini repozitoriji naprave se pohranjuju u bazu podataka na način da se pozove metoda *Save()* klase *UnitOfWork* koja zatim na *TracktorDb* kontekstu poziva metodu *SaveChanges()*. Najčešće se poziv te metode delegira u konkretne metode repozitorija kako bi se od baze podataka mogao dobiti identifikator (*Id*) novog objekta koji se pohranjuje.



## Entity framework

*TracktorDb* razred nasljeđuje *DbContext*. Kao javno sučelje pruža javna svojstva za dohvat *DbSet* kolekcija određenih tipova. Tipovi su objekti koje je *Entity* *Framework* mapirao iz relacijske sheme baze podataka. Veze, predstavljene relacijama u relacijskom modelu, su mapirane kao svojstva, odnosno virtualne kolekcije. Prilikom dohvaćanja bilo kojeg entiteta iz baze podataka koristi se *lazy loading* tehnika, što znači da se entiteti povezani s entitetom kojeg dohvaćamo učitavaju tek kada im pristupamo.

