SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

Seminar iz kolegija:

**Objektno oblikovanje**

TUSK

*O/R Mapiranje*

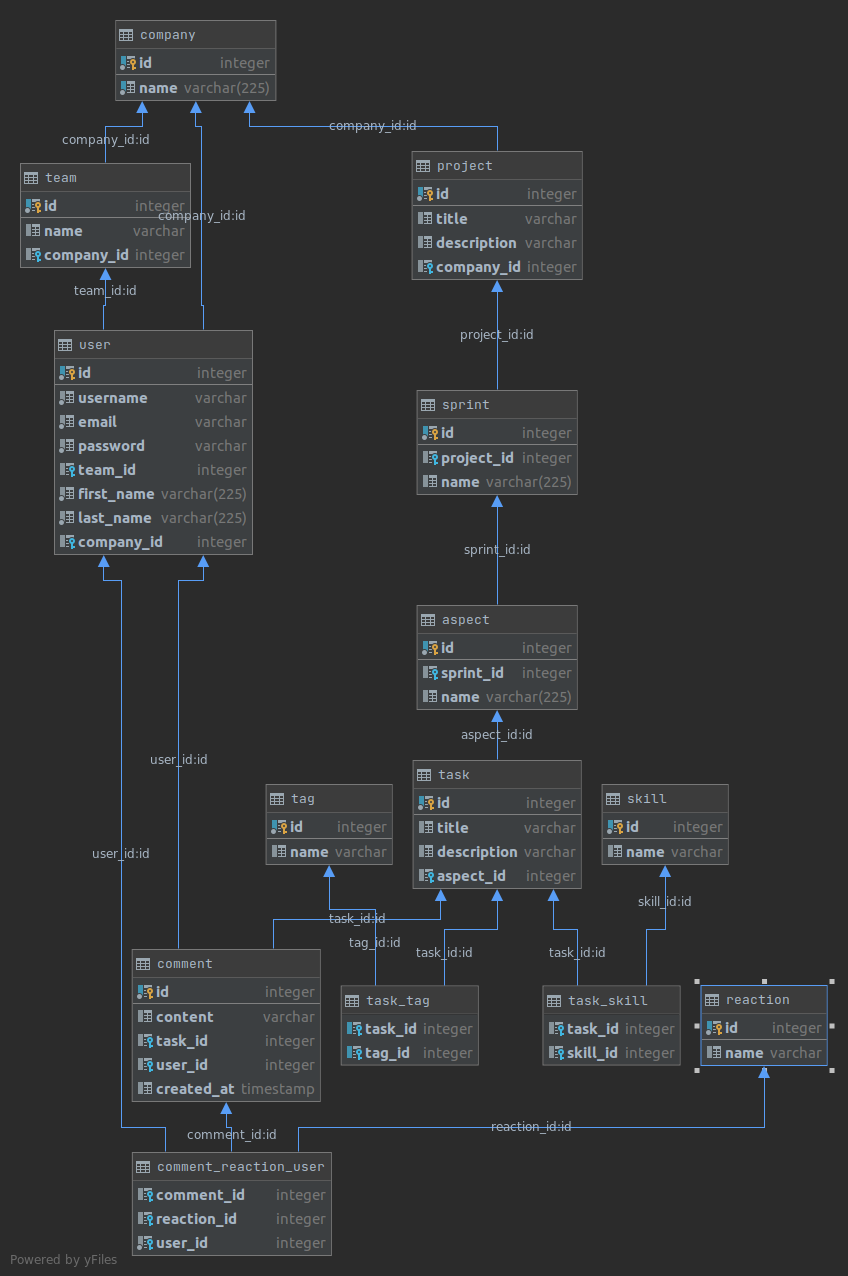
## Bartol Bilić Luka Dumančić Luka Lenđel Karlo Ježić Jurica Kenda

Zagreb, siječanj, 2021.

# Opis objektnog modela

## 

## 1.1 Model baze podataka



### 1.2. Dijagram klasa kontrolera

### 1.3 Opis implementacije perzistencije

Svaki entitet u bazi podataka ima 1-1 mapiranje s kontrolerom i repozitorijem. Ako imamo entitet *user* to znači da postoji *UsersController* i *UsersRepository* koji izlažu sučelje prema kontroleru odnosno klijentu.

Jednostavnim grafom tu vezu možemo prikazati na idući način:

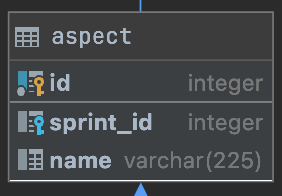


### Aspect

**Dijagram razreda**

****

**Tablica u bazi podataka**

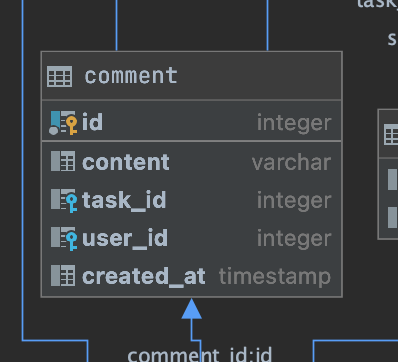
****

Comment

Dijagram razreda:

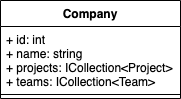


Tablica u bazi podataka:

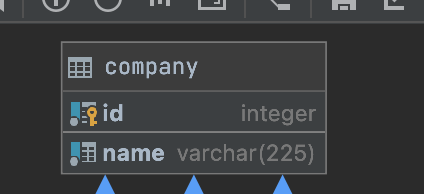


Company

Dijagram razreda:

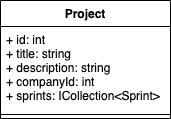


Tablica u bazi podataka:

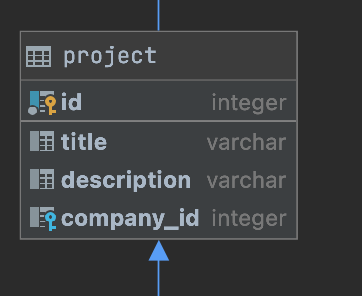


Project

Dijagram razreda:

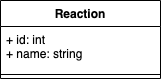


Tablica u bazi podataka:

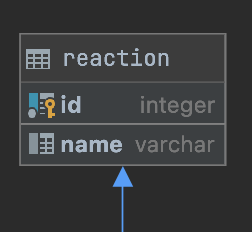


Reaction

Dijagram razreda:



Tablica u bazi podataka:

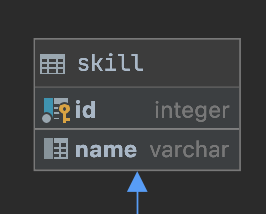


Skill

Dijagram razreda:

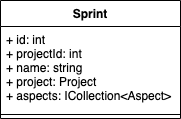


Tablica u bazi podataka:

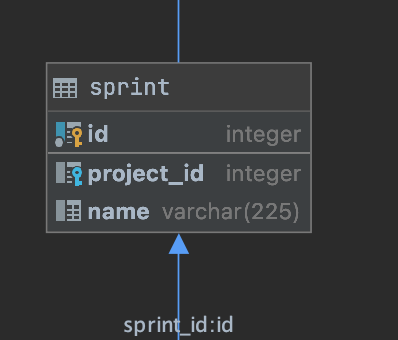


Sprint

Dijagram razreda:



Tablica u bazi podataka:

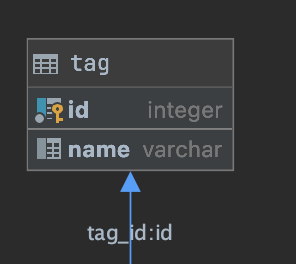


Tag

Dijagram razreda:



Tablica u bazi podataka:

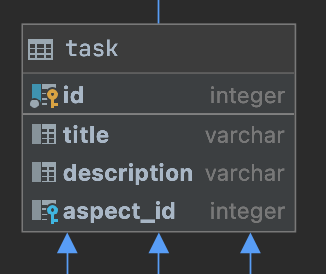


Task

Dijagram razreda:



Tablica u bazi podataka:

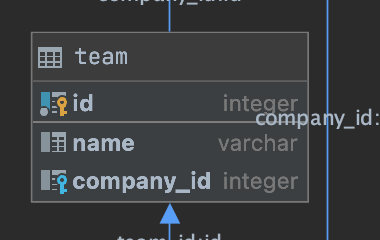


Team

Dijagram razreda:

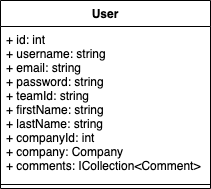


Tablica u bazi podataka:

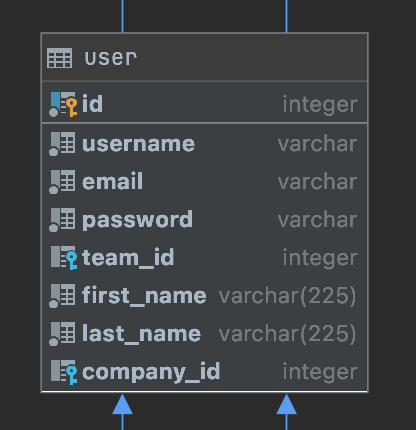


User

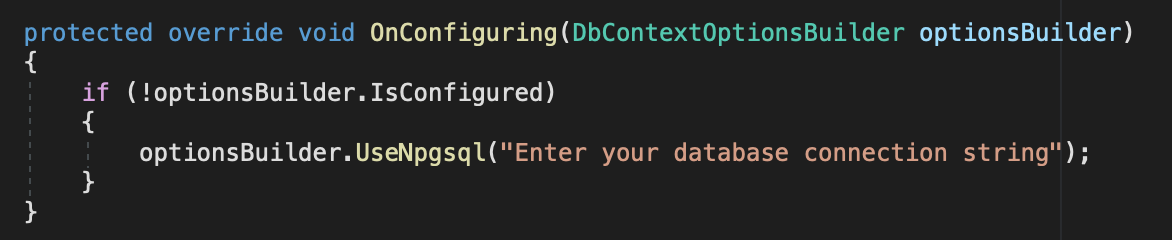
Dijagram razreda:



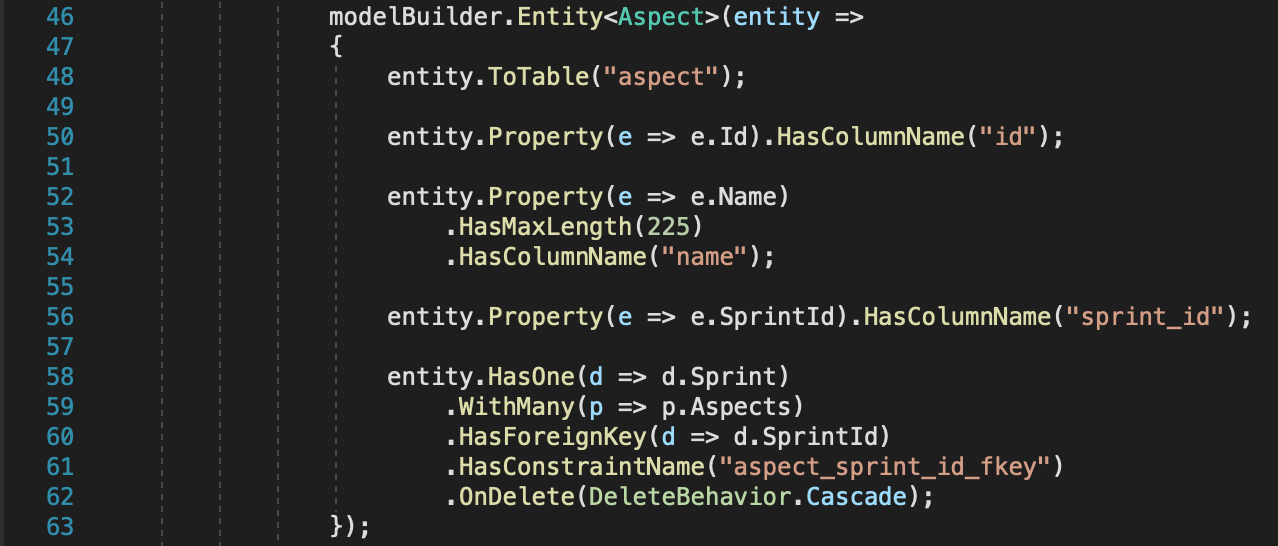
Tablica u bazi podataka:



U Entity Frameworku se O/R mapiranje radi na idući način: Definiramo poseban razred *DatabaseContext* koji nasljeđuje *Microsoft.EntityFrameworkCore.DbContext* i overrideamo metodu *OnConfiguring(DbContextOptionsBuilder optionsBuilder)* i *OnModelCreating(ModelBuilder modelBuilder).*U metodi *OnConfiguring()* se moramo spojiti na bazu podataka i odabrati koju bazu podataka koristimo. U našem slučaju koristimo PostgreSQL.



U metodi *OnModelCreating()* definiramo O/R mapiranja. U nastavku je dan primjer mapiranja entiteta *Aspect*.

Linija 48: entity.toTable(“aspect”);  
Mapiramo klasu *Aspect* na tablicu *“aspect”* u bazi podataka.

Linija 52:: entity.Property(e => e.Id).HasColumnName(“id”);  
Mapiramo field “name” na stupac *“*name”u tablici “aspect” u bazi podataka. Također možemo definirati i dodatna ograničenja kao što su duljina naziva.  
  
Linija 59:

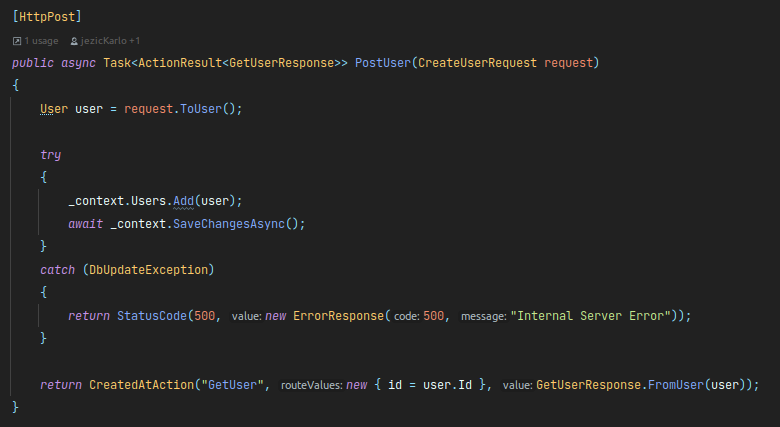
Definiramo 1-N mapiranje između tablice “sprint” i “aspect” što znači da 1 sprint sadrži više aspekata. Također dodajemo Foreign Key koji znači da se “sprintId” u tablici “aspect” odnosi na “id” u tablici “sprint”. Također definiramo ponašanje kada se obriše “roditeljski” entitet. U našem slučaju to je CASCADE i tada će se obrisati i svi entiteti djeca.

### 1.4. Transakcije i unit of work

Transakcije omogućuju da se više operacija nad bazom podataka odviju atomarno. To znači da su ili sve operacije unutar jedne transakcije uspješno provedene ili niti jedna nije provedena.

U našem slučaju jedna transakcija je sve ono što se odvije prilikom jednog korisničkog zahtjeva te završava naredbom SaveChanges(). SaveChanges() sama po sebi osigurava atomarnost, no mi, da bi bili još sigurniji pozivamo metodu SaveChanges() nakon svake promjene u bazi podataka jer smo zaključili da je takvih operacija malo i da nam taj tradeoff najbolje ide u korist.

Unit of work je pattern koji nam omogućuje da se više operacija nad bazom podataka (insert/update/delete) odvija kao da je u pitanju jedna transakcija. U našem slučaju svaka je transakcija sama po sebi već napravljena prema Unit of work pattern-u jer je nakon svake operacije nad bazom podataka (koja nije dohvaćanje) pozvana metoda SaveChanges() kojom se završava transakcija.



Kao primjer možemo vidjeti PostUser() methodu UsersController-a (slika iznad) koja služi za registraciju korisnika u sustav. Vidimo da je nakon metode Add(), koja unosi novog korisnika u bazu podataka, pozvana metoda SaveChangesAsync() koja zapravo provodi to što je napravljeno liniju prije.

Proučavanjem ovog primjera vidljivo je da je u našem slučaju 1 request = 1 transaction = 1 unit of work.

## 2. NHibernate vs Entity Framework

Ovaj dio dokumenta bavi se proučvanjem sličnosti i razlika između ova dva radna okriva. Cilj je dobiti što veći uvid u trade-offe koji se rade odabirom jedne od ovih tehnologija nad drugom. Kao stupove usporedbe odlučili smo uzeti: distributere, podržane baze podataka, korisne značajke, upravljanje konkurentnim pristupom podacima..

## Distributeri

NHibernate je radni okvir distribuiran i održavan od strane zajednice, bez ikakvih sponzora ili sponzorskih kompanija. Entity Framework je radni okvir usko vezan uz tvrtku Microsoft. To generalno nije najbolji indikator, obzirom da ta tvrtka esencijalno drži monopol nad tehnologijom i donosi relevantne odluke koje utječu na naš kod. S druge strane, ako je tvrtka pouzdana i pokazuje da su politike odluka u skladu s dobrim principima poput “backwards compatibilitya”, nije loše imati ju kao distributera koji će kontinuirano održavati taj kod.

## Podržane baze podataka

Obzirom da koristimo bazu podataka za perzistenciju, svakako želimo analizirati koja tehnologija podržava koji stack. NHibernate je u ovom pogledu svakako daleko ispred Entity Frameworka. Podržava PostgreSQL, SQL Server, SQL Lite, My SQL , DB2, Ingres … dok EF podržava mnogo manje. Neke koje EF podržava su SQL Server, PostgreSQL, te SQL Lite.

## Upravljanje konkurentnim pristupom podacima

Po oba aspekta, obje tehnologije pružaju zadovoljavajuću podršku. Iako, naizgled NHibernate pruža više potencijalno korisnih značajki u ovom aspektu poput eksplicitnog zaključavanja n-torki. Nama to ne donosi nikakvu vrijednost, tako da ovo nije komponenta koja nas je nagnala na odluku.

## Korisne značajke

### Kolekcije

NHibernate nudi mnogo veću podršku za kolekcije nego Entity Framework. Za razliku od EF koji nudi podršku samo za liste, NHibernate nudi neporedane kolekcije, setove, mape, primitivne kolekcije..

Iz tog aspekta NHibernate pruža značajno više nego EF, te ako imamo neke aplikacijski specifične potrebe, ovo može biti komponenta koja nam utemelji konačnu odluku.

### Lazy loading

NHibernate podržava lazy loading, za razliku od Entity Frameworka. Iako je to veoma koristan feature, nama ne utječe na odluku obzirom da ne očekujemo mnogo podataka u aplikaciji. Kada bi aplikaciju skalirali na produkcijsku razinu, svakako je ovo točka koja bi nam mnogo značila u procjeni.

### Infrastrukturalna podrška

Jedna od najvećih podrški u Entity Frameworku jest autogeneriranje entiteta iz baze podataka. Ovo je vrlo važna stavka jer može inženjerima uštediti velike količine vremena koje bi inače uložili u pisanje boilerplate koda. Također, najviše grešaka nalazi se upravo u ovakvom kodu. On je repetitivan, te developeri često posežu za CPP (copy-paste-programming) metodologijom :) .

### Iskoristivost

Obzirom na mnoge feature koje pruža NHibernate, learning curve se može činiti poprilično strm. Za razliku od njega, Entity Framework je lak za uporabu i temeljito dokumentiran.