horizontal line

**Fakultet elektrotehnike i računarstva**

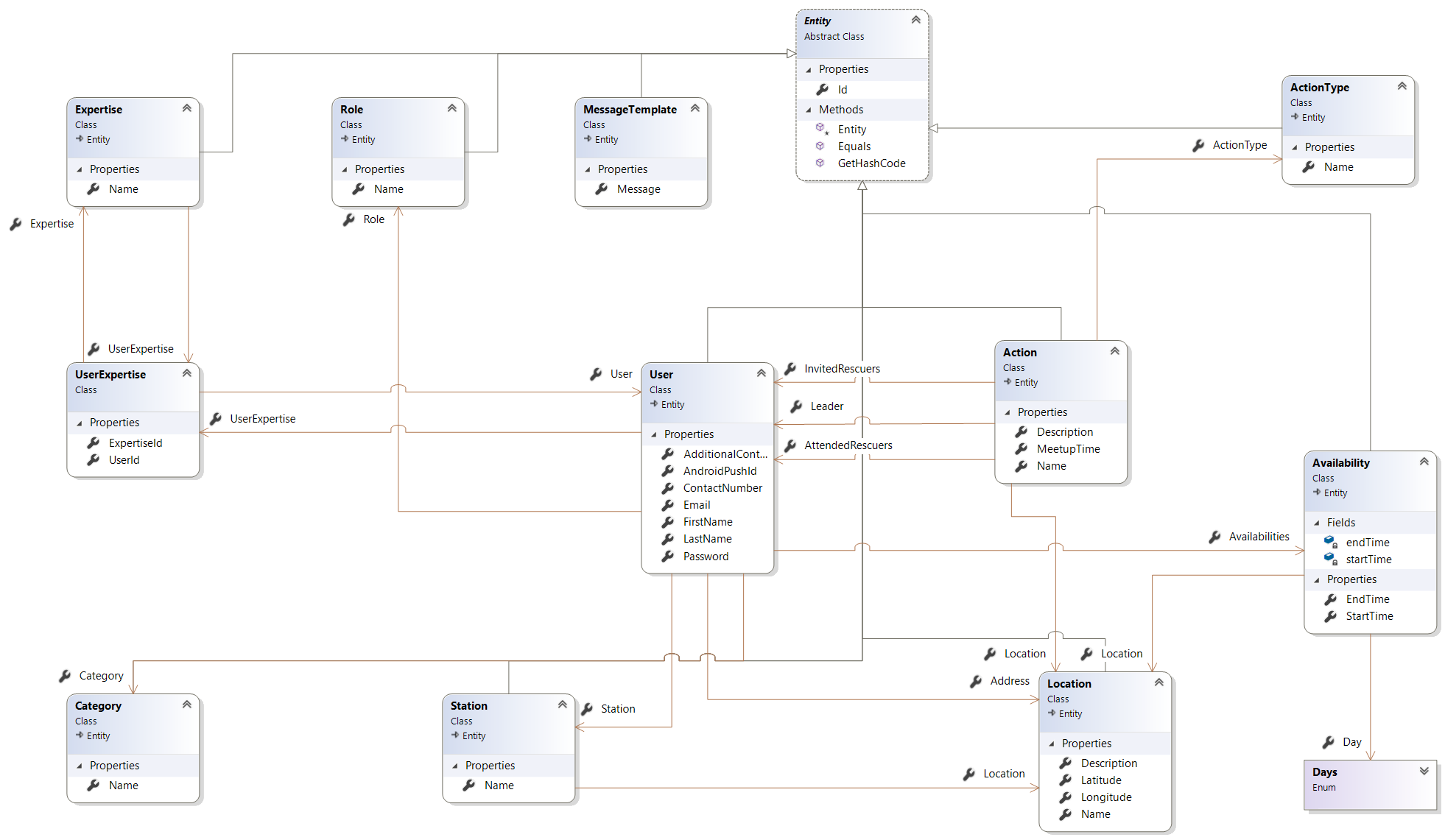
Unska 3

10000 Zagreb

Laboratorij profila

**5. Veljače 2018**

# OPIS MODELA



*Slika 1.* Dijagram razreda objektnog modela

Dostupnost korisnika se definira na tjednoj bazi, stoga entitet Availability opisuje pojedini period dostupnosti korisnika aplikacije tako da definira vrijeme početka dostupnosti, vrijeme kraja dostupnosti i dan u tjednu, te lokaciju na kojoj je korisnik u tom vremenskom rasponu dostupan.

Entitet korisnika (User) sadrži metode za provjeru dostupnosti prema određenom vremenu tako da ako korisnik ima definiran period dostupnosti za to vrijeme lokacija na kojoj je dostupan je lokacija definirana na toj instanci dostupnosti, a ako nije, pretpostavlja se njegova lokacija prebivanja koja se definira pri stvaranju samog korisnika i uređuje naknadno po potrebi.

Perzistencija je ostvarena pomoću PostgreSQL baze podataka. Korišten je besplatni javni server Elephant SQL.

# MAPIRANJE

Mapiranje klasa napravljeno je pomoću Entity Framework Core radnog okvira.[[1]](#footnote-1) Definicije mapiranja navedene su unutar ApplicationDbContext klase. Mapiranja su definirana pomoću Entity Framework Core Fluent API i definiraju ograničenja nad pojedinim svojstvima modela domene i odnosima među modelima. Unit Of Work obrazac je implementiran u instanci DbContext odnosno ApplicationDbContext. Entity Framework Core brine o provođenju svih transakcija i poništavanju neuspješnih.

Prema konvenciji, svaki entitet biti će postavljen za mapiranje s tablicom u bazi podataka s nazivom koji odgovara DbSet<TEntity> svojstvu koje je definirano na klasi izvedenoj iz DbContext klase sadržane u Entity Framework Coreu.

public DbSet<User> Users { get; set; }

Za konfiguraciju tablice na koju se mapira određeni tip moguće je koristiti anotacije ili Entity Framework Core Fluent API.

Za mapiranje stupaca u pojedinoj tablici baze podataka, svako svojstvo će na pojedinom entitetu definirati stupac u tablici na koju se mapira uz odgovarajuću definiciju u Fluent API-ju, npr.

modelBuilder.Entity<User>()  
 .Property(u => u.FirstName)  
 .HasColumnName("UserFirstName"); // proizvoljni naziv stupca

Tipovi podataka u stupcima konfigurirani su automatski, sustav baze podataka određuje tip podataka prema CLR (*Common Language Runtime*) tipovima. Uz definiciju tipa prihvaća i ograničenja kao što su npr. najveća dozvoljena duljina, je li stupac primarni ključ itd.

Primarni ključevi određeni su kroz Fluent API metodom HasKey, a tip podataka metodom HasColumnType.

modelBuilder.Entity<HGSSSARAssistant.Core.User>(user =>  
 {  
 user.HasKey(u => u.Id);  
 user.Property(u => u.FirstName).HasColumnType("nvarchar(40)");  
 user.Property(u => u.LastName).HasColumnType("nvarchar(40)");  
 });

Prema konvenciji, strani ključevi imenuju se prema slijedećem predlošku:

FK\_<ime zavisne tablice>\_<ime temeljne tablice>\_<naziv svojstva stranog ključa>

Fluent API koristi ovaj predložak, no moguće je proizvoljno imenovati ograničenja stranih ključeva pozivom metode HasConstraintName u lancu metoda.

modelBuilder.Entity<UserExpertise>()  
 .HasOne(ue => ue.Expertise)  
 .WithMany(e => e.UserExpertise)  
 .HasForeignKey(ue => ue.ExpertiseId);

Mapiranje nasljeđivanja u Entity Framework Core je dostupno samo za strategiju *Table-Per-Hierarchy*. Ostale strategije kao što su *Table-Per-Type* i *Table-Per-Concrete-Type* nisu još dostupne.[[2]](#footnote-2)

Definiranje diskriminatorskog stupca pomoću metode HasDiscriminator<T> u Fluent API lancu.

modelBuilder.Entity<Blog>()  
 .HasDiscriminator<string>("blog\_type")  
 .HasValue<Blog>("blog\_base")  
 .HasValue<RssBlog>("blog\_rss");  
 }

1. <https://github.com/nhibernate/nhibernate-core/issues/954> [↑](#footnote-ref-1)
2. <https://docs.microsoft.com/en-us/ef/core/modeling/relational/inheritance> [↑](#footnote-ref-2)