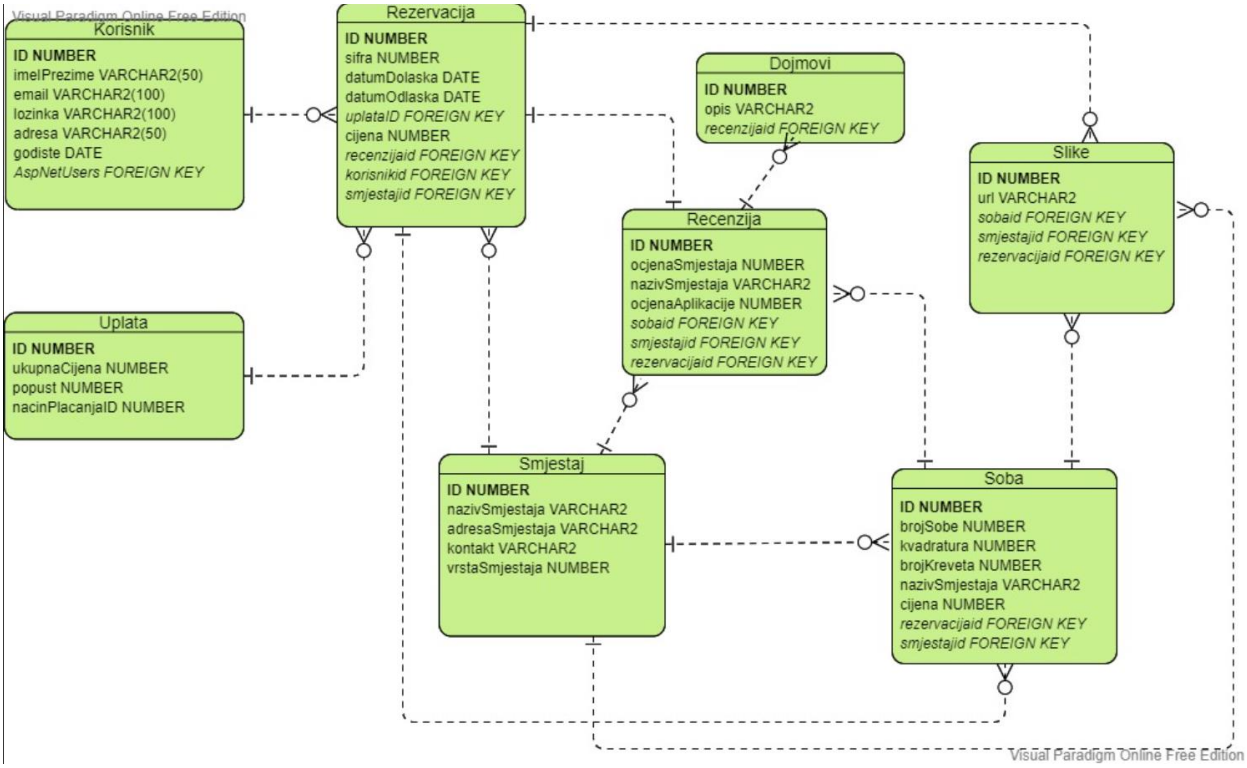
Univerzitet u Sarajevo
Elektrotehnički fakultet
Predmet: Objektno-orjentirana analiza I dizajn

REFACTORING RJEŠENJA (DOKUMENTACIJA)

Naziv grupe: AAEInvest
Radili: Brulić Almina, Hajro Adnan
Brojevi indexa: 18962, 18905

1. *Prepravljanje ER dijagrama kroz kod*

Nakon što smo uradili prvu migraciju našeg projekta shvatili smo da na osnovu našeg ER dijgrama primijetili smo da kompajler prijavljuje greške koje upućuju na to da se preko jedne tabele mogu obrisati I ažurirati redovi iz druge tabele u bazi podataka.
Razlog tome je bio što smo imali ForeignKey-eve npr. u 3 tabele, gdje su u prve dvije tabele bila dva ista ForeignKey-a, a u drugoj I trećoj tabeli opet dva ista, ali je jedan bio PrimaryKey u prvoj tabela.
Problem smo riješili tako što smo uradili sljedeće prepravke:



U klasi “Recenzija” smo obrisali ForeignKey-eve navedene na slici iznad u toj klasi, a umjesto njih smo ubacili ForeignKey na tabelu “Dojmovi”. U klasi Rezervacija smo umjesto ForeignKey-a na klasu(entitet) “Recenzija” ubacili ForeignKey na tabelu “Uplata”. Iz klase “Slike” smo obrisali sve ForeignKey-eve, a umjesto toga smo u klasi “Smještaj” dodali ForeignKey na tabelu(klasu) “Slike”. Kako klasa “Soba” ima ForeignKey na “Smještaj” onda preko klase “Smještaj” može pristupiti I odgovarajućim slikama, te bi zbog toga bilo pogrešno da smo u “Soba” stavili ForeignKey na klasu(entitet) “Slike”. Neke od prepravki modela se mogu vidjeti na narednim slikama:

```
public class Recenzija
{
    [Key]
    0 references
    public int Id { get; set; }
    0 references
    public int OcjenaSmjestaja { get; set; }
    0 references
    public string NazivSmjestaja { get; set; }
    0 references
    public int OcjenaAplikacije { get; set; }

    [ForeignKey("Dojmovi")]
    0 references
    public int DojamId { get; set; }
    0 references
    public Dojmovi Dojmovi { get; set; }

    0 references
    public Recenzija() { }
}
```

```
public class Slike
{
    [Key]
    5 references
    public int Id { get; set; }
    0 references
    public string Url { get; set; }

    0 references
    public Slike() { }
}
```

2. *Refactoring na nivou podataka*

Pošto u nekim našim klasama iz modela projekta imamo jako puno atributa koji su tip int, string DateTime i slično, bilo bi poželjno da napravimo novu klasu koja će imati iste te tipove i da naslijedimo tu klasu iz naše klase modela za koju pravimo ispravku. Međutim, primijetili smo da to nije baš najbolja ideja u našem slučaju iz razloga što nemamo nikakvih metoda u tim klasama pa ne bismo ispunili koncept nasljeđivanja(dva ključna pitanja na koja se treba odgovoriti kada govorimo o nasljeđivanju).

```
public class Korisnik
{
    [Key]
    12 references
    public int KorisnikId { get; set; }
    [DisplayName("Ime i prezime")]
    [Required]
    13 references
    public string KorisnikImeIPrezime { get; set; }
    [Required]
    [RegularExpression(@"^[^@\s]+@[^@\s]+\.(com|net|org|gov|ba)$", ErrorMessage = "Invalid pattern.")]
    12 references
    public string Email { get; set; }
    [Required]
    12 references
    public string Lozinka { get; set; }

    [StringLength(maximumLength: 50, MinimumLength = 3, ErrorMessage ="Nevalidna adresa!")]
    14 references
    public string Adresa { get; set; }
    [ValidateDate2]
    [DataType(DataType.Date)]
    12 references
    public DateTime Godiste { get; set; }

    0 references
    public Korisnik() { }
}
```

3. Refactoring na nivou klase

U klasama “Korisnik” I “Rezervacij” smo imali preklapanje metode “ValidationResult” za validaciju datuma. Prvi put ovu metodu smo preklopili u imeniku “OOAD2022.Models”, međutim pošto su nam trebale 2 različite implementacije, ono što smo uradili jeste da smo klasu, koja implementira metodu za validaciju, ubacili u klasu “Korisnik” I “Rezervaciju” I proglasili je privatnom klasom. Ispravke se mogu vidjeti na slikama ispod:

```
public class Korisnik
{
    1 reference
    private class ValidateDate2 : ValidationAttribute
    {
        0 references
        protected override ValidationResult IsValid
        (object date, ValidationContext validationContext)
        {
            return ((DateTime)date < DateTime.Now)
            ? ValidationResult.Success
            : new ValidationResult("Validan je datum prije danasnjeg datuma");
        }
    }
```

```
public class Rezervacija
{
    2 references
    private class ValidateDate : ValidationAttribute
    {
        0 references
        protected override ValidationResult IsValid
        (object date, ValidationContext validationContext)
        {
            return ((DateTime)date >= DateTime.Now)
            ? ValidationResult.Success
            : new ValidationResult("Validan je datum od datuma danasnjeg dana");
        }
    }
```