MEĐIMURSKO VELEUČILIŠTE ČAKOVEC

RAČUNALSTVO

PATRIK MIHOCI

Izrada modela za objektno relacijsko povezivanje podataka u okviru aplikacije za opis predmeta

ZAVRŠNI RAD

Mentor:

dr.sc. Mihael Kukec, prof.v.š.

ČAKOVEC, 2017.

ZAHVALA

Zahvaljujem se na stručnom vodstvu mentora dr.sc. Mihaela Kukeca koji mi je predložio i olakšao izradu ovog rada svojim znanjem i prijedlozima. Također se zahvaljujem prijateljima koji su mi pružili podršku tijekom izrade.

SAŽETAK

Tema ovog rada je izrada modela za objektno relacijsko povezivanje podataka koji će kasnije biti korišteni za pohranu podataka aplikacije za opis predmeta. U današnje vrijeme većina aplikacija koristi bazu podataka pa je cilj ovog završnog rada prikazati na koji način se stvara baza tako da se koristi pristup „Model First“ što znači da se baza generira na temelju modela koje koristi aplikacija. Okvir (*engl. Framework*) koji nam pruža tu mogućnost je Entity Framework za platformu .NET Core. Opisane su i CRUD (*engl. create, read, update, delete*) metode LINQ upitima koje se koriste za interakciju aplikacije s bazom podataka. Čitanje, pisanje, uređivanje i brisanje podataka iz baze prikazano je kroz aplikaciju opisanu u četvrtom poglavlju.

Ključne riječi: .NET Core, Entitiy Framework, LINQ, Model First

SADRŽAJ

Sažetak

1. UVOD

Cilj ovog rada je izrada modela za objektno relacijsko povezivanje podataka u okviru aplikacije za opis predmeta. Prvi korak je odabir alata kojim će se realizirati ovaj rad. Za upravljanje bazom podataka koristi se Microsoft SQL Server, za izradu modela i aplikacije koristi se .NET Core platforma tvrtke Microsoft, za objektno relacijsko povezivanje koristi se Entity Framework (EF), a razvojni alat korišten je Visual Studio 2017 također od tvrtke Microsoft. Nakon odabira alata za rad slijedi izrada modela prema kome će EF generirati tablice za bazu podataka. Ovaj pristup izrade baze podataka naziva se „Model First“. Uz to biti će demonstrirane sposobnosti .NET Core platforme kroz manju aplikaciju koja koristi LINQ upite za komuniciranje s bazom. Aplikacija će imati funkcionalnost za stvaranje, čitanje, izmjene i brisanje podataka (Create, Read, Update, Delete - CRUD).

1.1 Priprema

Prije početka rada potrebno je pripremiti razvojnu okolinu. Treba instalirati sve spomenute programe. Važno je napomenuti da aplikacija zahtjeva .NET Core verziju 2.0 i više. [1] Za početak u konzolu upisujemo naredbu:

```
dotnet new console
```

Ovom naredbom generira se novi projekt, nadalje potrebni su paketi za rad s bazom podataka koji se dodaju upisom naredbi:

```
dotnet add package Microsoft.EntityFrameworkCore.SQLServer
dotnet add package Microsoft.EntityFrameworkCore.Design
```

Također treba urediti .csproj datoteku koje se nalazi u mapi generiranog projekta tako da se doda sljedeći kod unutar ItemGroup oznaka.

```
  <DotNetCliToolReference
Include="Microsoft.EntityFrameworkCore.Tools.DotNet"
Version="2.0.0" />
```

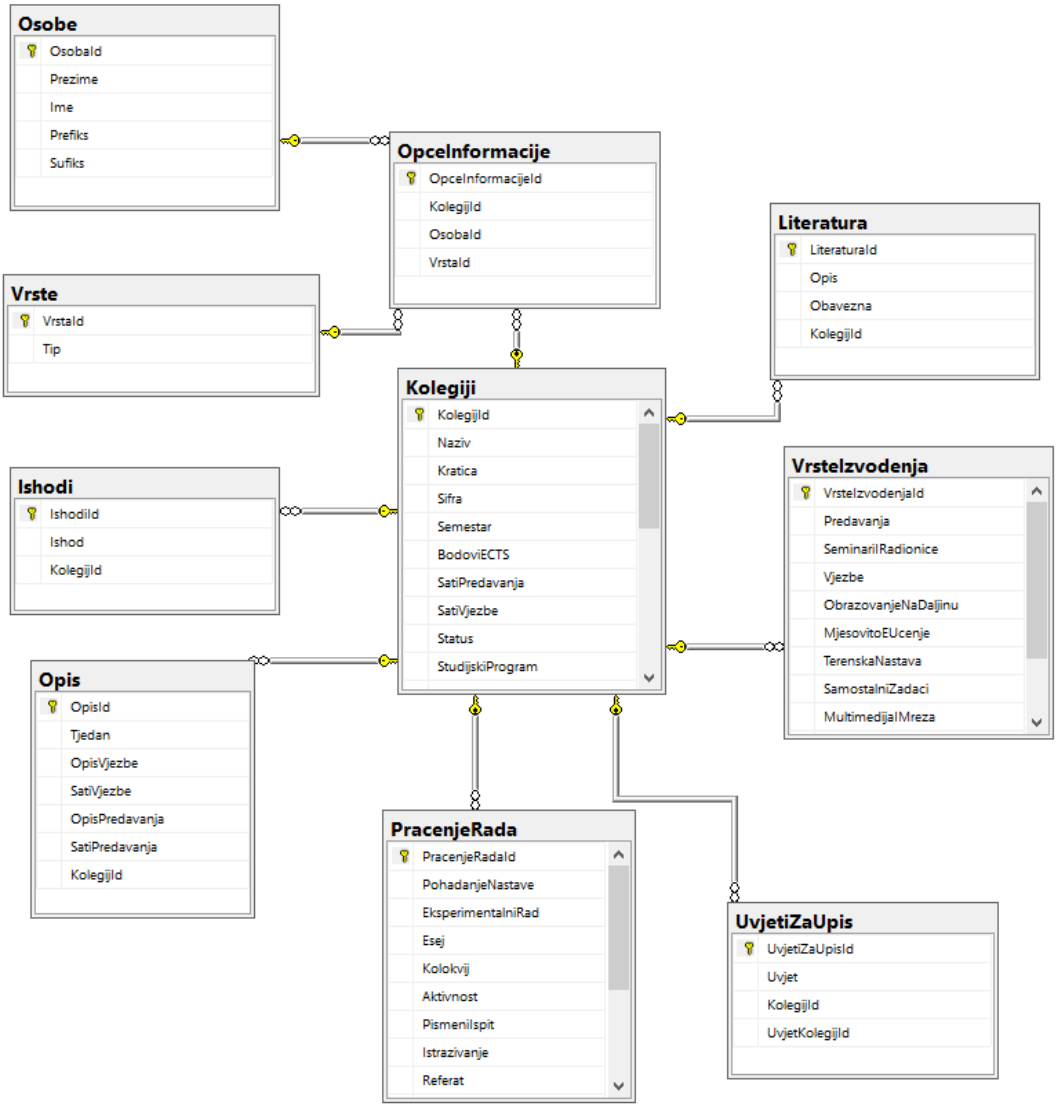
Ukoliko brojevi verzije iznad nisu točni mogu se instalirati nove naredbom

```
dotnet restore
```

U mapi u kojoj se nalazi projekt potrebno je dodati novu datoteku s nazivom „Models.cs“ koja će se koristiti za izradu modela.

2. MODEL

Model je razred koji predstavlja tablicu u bazi podataka, a budući da se koristi pristup „Model First“ znači da vrste podataka koji će se nalaziti u bazi podataka definiramo u modelu [2]. Podaci korišteni za izradu modela bazirani su na već postojećoj aplikaciji. Na slici 1 nalazi se dijagram već gotove baze podataka.



Slika 1. Dijagram baze podataka
Izvor: Autor

2.1 Kontekst razred

Prije nego što se počne s izradom modela potreban je kontekst koji će EF koristi za komunikaciju između baze i aplikacije. Kontekst dodajemo tako da se doda novi razred koji nasljeđuje *DbContext*.

```
public class OpisKolegijaContext : DbContext
{
    public DbSet<Kolegij> Kolegiji { get; set; }
    public DbSet<Osoba> Osobe { get; set; }
    public DbSet<Vrsta> Vrste { get; set; }
    public DbSet<OpceInformacije> OpceInformacije { get; set; }
    public DbSet<PracenjeRada> PracenjeRada { get; set; }
    public DbSet<UvjetiZaUpis> UvjetiZaUpis { get; set; }
    public DbSet<Opis> Opis { get; set; }
    public DbSet<VrsteIzvodenja> VrsteIzvodenja { get; set; }
    public DbSet<Literatura> Literatura { get; set; }
    public DbSet<Ishodi> Ishodi { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlServer("Data Source=dev1.mev.hr;Initial Catalog = " +
            "MihociPatrikDB1;User ID=mihocip;Password=r5kB8vx2jtJdWDGk;Connect Timeout=60");
    }
}
```

Slika 2. Razred *OpisKolegijaContext* koji nasljeđuje *DbContext*;
Izvor: Autor

Razrede koji nasljeđuju *DbContext* koriste se kao veza između modela i baze podataka. Metoda *OnConfiguring* služi za postavljanje veze sa serverom na kojem se nalazi baza podataka. Svaka od *DbSet* svojstva sa slike 2 predstavlja vezu između modela i tablice u bazi podataka pa se iz toga može zaključiti da se za ovaj projekt koristi 10 modela ili tablica. *DbSet* se koristi kao metoda otkrivanja modela. Modeli se mogu otkriti na 3 načina, pomoću već spomenutog *DbSet*, tako da u modelu koji je već otkriven spomenemo neki drugi model, ili tako da ga spomenemo u *OnModelCreating* metodi kao što je prikazano na slici 3.

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Kolegij>();
}
```

Slika 3. Otkrivanje modela „Kolegij“ pomoću OnModelCreating metode
Izvor: Autor

2.2 Modeli Osoba i Vrsta

```
public class Osoba
{
    [Key]
    public int OsobaId { get; set; }
    public string Prezime { get; set; }
    public string Ime { get; set; }
    public string Prefiks { get; set; }
    public string Sufiks { get; set; }
}

public class Vrsta
{
    [Key]
    public int VrstaId { get; set; }
    public string Tip { get; set; }
}
```

Slika 4. Model Osoba i Model Vrsta
Izvor: Autor

EF koristi konvencije kako bi odredio koje svojstvo je primarni ključ [3]. Primarni ključ se određuje tako da se svojstvu da naziv „Id“ ili „<NazivModela>Id“ kao što je prikazano na slici 4. Isto tako može se primijetiti podatkovna anotacija *Key* koja isto označava primarni ključ. Kako bi omogućili korištenje ovakve anotacije potrebno je uključiti polje imena (*engl. namespace*) kodom:

```
using System.ComponentModel.DataAnnotations;
```

Još jedan način na koji se može odrediti primarni ključ prikazan je primjerom na slici 5 gdje je postavljen u metodi *OnModelCreating* u kontekst razredu.

```
public class OpisKolegijaContext : DbContext
{
    public DbSet<Vrste> Vrste { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Vrste>()
            .HasKey(c => c.PrimarniKljucVrste);
    }
}

public class Vrste
{
    [Key]
    public int PrimarniKljucVrste { get; set; }
    public string Vrsta { get; set; }
}
```

Slika 5. Model Vrste s primjerom postavljanja primarnog ključa
Izvor: Autor

2.3 Model Kolegij

```
public class Kolegij
{
    [Key]
    public int KolegijId { get; set; }
    public string Naziv { get; set; }
    public string Kratica { get; set; }
    public int Sifra { get; set; }
    public int Semestar { get; set; }
    public int BodoviECTS { get; set; }
    public int SatiPredavanja { get; set; }
    public int SatiVjezbe { get; set; }
    public string Status { get; set; }
    public string StudijskiProgram { get; set; }
    [Column(TypeName = "ntext")]
    public string Ciljevi { get; set; }
    [Column(TypeName = "ntext")]
    public string Ishodi { get; set; }
    [Column(TypeName = "ntext")]
    public string Komentari { get; set; }
    [Column(TypeName = "ntext")]
    public string ObavezeStudenata { get; set; }
    [Column(TypeName = "ntext")]
    public string RadnoOpterecenje { get; set; }
    [Column(TypeName = "ntext")]
    public string Ocjenjivanje { get; set; }
    [Column(TypeName = "ntext")]
    public string Pohadanje { get; set; }
    [Column(TypeName = "ntext")]
    public string Kontaktiranje { get; set; }
    [Column(TypeName = "ntext")]
    public string Informacije { get; set; }
    [Column(TypeName = "ntext")]
    public string PisaniRadovi { get; set; }

}
```

Slika 6. Model Kolegij
Izvor: Autor

U modelu *Kolegij* se nalaze podaci koji su specifični samo za taj određeni kolegij to jest podaci koji ne trebaju veze jedan na više (1:n) ili više na više (n:n). Za brojevne podatke korišten je tip podataka *int*, a za tekstovne podatke korišten je tip *string*. Kod nekih podataka tipa *string* može se vidjeti podatkovna anotacija

```
[Column(TypeName = "ntext")]
```

Ona se koristi kod tekstovnih podataka za koje je potrebno više prostora, njome se u bazi podataka tip podatka postavlja u tip *text*, gdje se tip *string* bez anotacije postavlja u tip

varchar. Kako bi se anotacije mogle koristiti potrebno je uključiti novo polje imena kodom:

```
using System.ComponentModel.DataAnnotations.Schema;
```

2.4 Model OpceInformacije

```
public class OpceInformacije
{
    [Key]
    public int OpceInformacijeId { get; set; }
    [ForeignKey("KolegijForeignKey")]
    public int KolegijId { get; set; }
    public Kolegij Kolegij { get; set; }
    [ForeignKey("OsobaForeignKey")]
    public int OsobaId { get; set; }
    public Osoba Osoba { get; set; }
    [ForeignKey("VrstaForeignKey")]
    public int VrstaId { get; set; }
    public Vrsta Vrsta { get; set; }
}
```

Slika 7. Model OpceInformacije
Izvor: Autor

Model *OpceInformacije* je „spojni“ model koji spaja modele *Kolegij*, *Osoba* i *Vrste*, što znači da ovaj model sam po sebi nema značenje to jest kako bi ovaj model imao značenje potrebni su nam drugi modeli. Ovaj model je potreban kako bi ostvarili relacije više na više (n:n), pošto jedan kolegij može imati više osoba povezanih s njime, isto tako jedna osoba može biti povezana s više kolegija i imati više uloga ili „Vrsta“ za svaki kolegij. U ovom primjeru može se vidjeti korištenja konvencija za određivanje stranog ključa na primjer svojstva:

```
public int KolegijId { get; set; }
public Kolegij Kolegij { get; set; }
```

Ova dva svojstva zajedno stvaraju strani ključ koji pokazuje na model *Kolegij*. Svojstvo „KolegijId“ služi za spremanje primarnog ključa nekog kolegija dok svojstvo „Kolegij“ EF koristi kako bi olakšao neke operacije npr. ako treba obrisati neki kolegij EF će obrisati sve relacije koje sadrže strani ključ tog kolegija. Za određivanje stranog

ključa u modelu mogu se koristi i podatkovne anotacije koje u ovim primjerima nisu potrebne ali su uključene zbog preglednosti:

```
[ForeignKey("KolegijForeignKey")]
```

2.5 Model PracenjeRada

```
public class PracenjeRada
{
    [Key]
    public int PracenjeRadaId { get; set; }
    public decimal PohadanjeNastave { get; set; }
    public decimal EksperimentalniRad { get; set; }
    public decimal Esej { get; set; }
    public decimal Kolokvij { get; set; }
    public decimal Aktivnost { get; set; }
    public decimal PismeniIspit { get; set; }
    public decimal Istrazivanje { get; set; }
    public decimal Referat { get; set; }
    public decimal SeminarskiRad { get; set; }
    public decimal UsmeniIspit { get; set; }
    public decimal Projekt { get; set; }
    public decimal PrakticniRad { get; set; }
    public decimal KontinuiranaProvjera { get; set; }
    [ForeignKey("KolegijForeignKey")]
    public int KolegijId { get; set; }
    public Kolegij Kolegij { get; set; }
}
```

Slika 8. Model PracenjeRada
Izvor: Autor

Model *PracenjeRada* predstavlja vrijednost u postocima koja određuje koliko je svaka od aktivnosti kolegija vrijedna. Model ima primarni ključ kojim se referencira objekt s kojim se želi raditi i strani ključ koji pokazuje na neki od kolegija koji se nalaze u bazi podataka. Ovaj model zahtjeva upis vrijednosti tipa *decimal* s preciznošću od 3 decimale. To postižemo kodom na slici 9.

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<PracenjeRada>(eb =>
    {
        eb.Property(p => p.PohadanjeNastave).HasColumnType("decimal(4, 3)");
        eb.Property(p => p.EksperimentalniRad).HasColumnType("decimal(4, 3)");
        eb.Property(p => p.Esej).HasColumnType("decimal(4, 3)");
        eb.Property(p => p.Kolokvij).HasColumnType("decimal(4, 3)");
        eb.Property(p => p.Aktivnost).HasColumnType("decimal(4, 3)");
        eb.Property(p => p.PismeniIspit).HasColumnType("decimal(4, 3)");
        eb.Property(p => p.Istrazivanje).HasColumnType("decimal(4, 3)");
        eb.Property(p => p.Referat).HasColumnType("decimal(4, 3)");
        eb.Property(p => p.SeminarskiRad).HasColumnType("decimal(4, 3)");
        eb.Property(p => p.UsmeniIspit).HasColumnType("decimal(4, 3)");
        eb.Property(p => p.Projekt).HasColumnType("decimal(4, 3)");
        eb.Property(p => p.PrakticniRad).HasColumnType("decimal(4, 3)");
        eb.Property(p => p.KontinuiranaProvjera).HasColumnType("decimal(4, 3)");

    });
}
```

Slika 9. Postavljanje preciznosti u metodi OnModelCreating
Izvor: Autor

2.6 Model UvjetiZaUpis

```
public class UvjetiZaUpis
{
    [Key]
    public int UvjetiZaUpisId { get; set; }
    public string Uvjet { get; set; }
    [ForeignKey("KolegijForeignKey")]
    public int KolegijId { get; set; }
    public Kolegij Kolegij { get; set; }
    public int UvjetKolegijId { get; set; }
}
```

Slika 10. Model UvjetiZaUpis
Izvor: Autor

Neki kolegiji se ne mogu upisati ukoliko nisu ostvareni neki uvjeti. Zbog toga postoji ovaj model. U njega se može upisati uvjet koji želimo, potrebno je upisati i strani ključ u „KolegijId“. Ukoliko se uvjet odnosi na neki drugi kolegij (npr. nije položen neki predmet) tada se može popuniti polje „UvjetKolegijaId“ s primarnim ključem kolegija na koji se taj uvjet odnosi.

2.7 Model VrsteIzvodenja

Model *VrsteIzvođenja* sadrži tipove nastave za svaki kolegij vezom jedan na jedan (1:1). Podaci ovog model mogli su se nalaziti u modelu *Kolegij*, no zbog već velikog broja podataka u modelu *Kolegij* i preglednosti napravljen je novi.

```
public class VrsteIzvodenja
{
    [Key]
    public int VrsteIzvodenjaId { get; set; }
    public bool Predavanja { get; set; }
    public bool SeminariIRadionice { get; set; }
    public bool Vjezbe { get; set; }
    public bool ObrazovanjeNaDaljinu { get; set; }
    public bool MjesovitoEUcenje { get; set; }
    public bool TerenskaNastava { get; set; }
    public bool SamostalniZadaci { get; set; }
    public bool MultimedijaIMreza { get; set; }
    public bool Laboratorij { get; set; }
    public bool MentorskiRad { get; set; }
    [ForeignKey("KolegijForeignKey")]
    public int KolegijId { get; set; }
    public Kolegij Kolegij { get; set; }
}
```

Slika 11. Model VrsteIzvođenja
Izvor: Autor

2.8 Model Literatura

Model *Literatura* koristi vezu više na jedan (n:1) s modelom *Kolegij*, što znači da jedan kolegij može imati više literatura ali ne i obrnuto. Svojstvo „Opis“ koristi se za zapisivanje naziva i autora literature dok svojstvo „Obavezna“ sprema *bool* vrijednosti o tome da li je literatura obavezna (*true*) ili ne (*false*).

```
public class Literatura
{
    [Key]
    public int LiteraturaId { get; set; }
    public string Opis { get; set; }
    public bool Obavezna { get; set; }
    [ForeignKey("KolegijForeignKey")]
    public int KolegijId { get; set; }
    public Kolegij Kolegij { get; set; }
}
```

Slika 12. Model Literatura
Izvor: Autor

2.9 Model Ishodi

Model *Ishodi* koristi vezu više na jedan (n:1) s modelom *Kolegij* kao i model *Literatura*. Koristi se za spremanje ishoda koji se upisuje u svojstvo ishod.

```
public class Ishodi
{
    [Key]
    public int IshodiId { get; set; }
    [Column(TypeName = "ntext")]
    public string Ishod { get; set; }
    [ForeignKey("KolegijForeignKey")]
    public int KolegijId { get; set; }
    public Kolegij Kolegij { get; set; }
}
```

Slika 13. Model Ishodi
Izvor: Autor

2.10 Model Opis

Model *Opis* koristi se za upis tjednog plana za kolegij. Upis se vrši za svaki od 15 tjedana. Sadrži broj sati predavanja i vježbi te opis predavanja i vježbi.

```
public class Opis
{
    [Key]
    public int OpisId { get; set; }
    public int Tjedan { get; set; }
    [Column(TypeName = "ntext")]
    public string OpisVjezbe { get; set; }
    public int SatiVjezbe { get; set; }
    [Column(TypeName = "ntext")]
    public string OpisPredavanja{ get; set; }
    public int SatiPredavanja { get; set; }
    [ForeignKey("KolegijForeignKey")]
    public int KolegijId { get; set; }
    public Kolegij Kolegij { get; set; }
}
```

Slika 14. Model Opis
Izvor: Autor

2.11 Verzija

Kada su svi modeli definirani može se odabrati jedan koji će biti korišten za spremanje verzije. Pogledom na dijagram baze na slici 1 može se primijetiti da sve tablice imaju vezu na tablicu *Kolegij*. Po tome se može zaključiti da bi podaci za verziju trebali biti u tablici *Kolegij*, pa je u model *Kolegij* dodan kod:

```
public int Verzija { get; set; }
public DateTime Stvoren { get; set; }
public DateTime Promjenjen { get; set; }
[Column(TypeName = "ntext")]
public string Opis { get; set; }
```

Svojstvo „Verzija“ koristi se za spremanje brojčane vrijednosti verzije, svojstvo „Stvoren“ i „Promijenjen“ koriste tip podatka *DateTime* se za spremanje datuma stvaranja i promjene verzije, svojstvo „Opis“ koristi se za tekstualni opis verzije. Važno je napomenuti da je ovo samo jedan od načina realiziranja verzije podataka.

3. MIGRACIJA

Nakon što su svi modeli završeni, prema njima će se pomoću EF generirati baza. Ako još nije, potrebno je postaviti vezu do baze podataka spomenutu u poglavlju 2.1. Za stvaranje i promjenu baze podataka koriste se migracije [4]. Kako bi se napravila migracija, potrebno je otvoriti konzolu u direktoriju projekta i upisati:

```
dotnet ef migrations add InitialCreate
```

Ovom naredbom dodaje se novu migraciju koja se zove „InitialCreate“, ako se dodaje nova migracija potrebno ju je nazvati drugačije jer samo jedna migracija može imati određeno ime.

```
dotnet ef database update
```

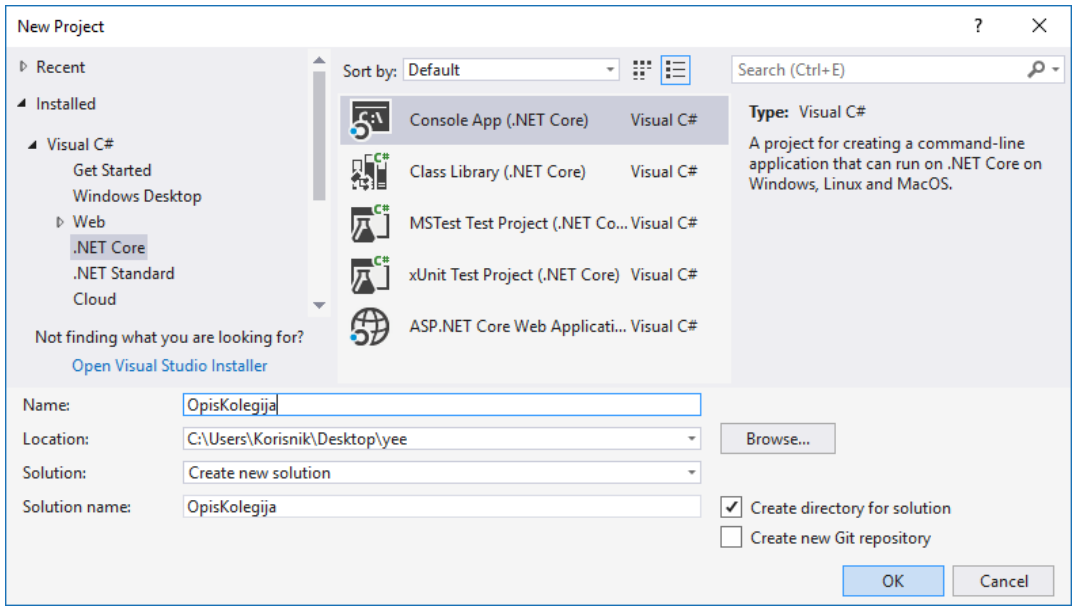
Naredbom iznad stvaramo ili primjenjujemo promjene sa migracije na bazu podataka. Ukoliko ne želimo primijeniti promjene na bazu migraciju možemo obrisati naredbom:

```
dotnet ef migrations remove
```

S migracijama ne treba pretjerivati jer se može desiti da se zbog nekog razloga baza desinkronizira s modelom.

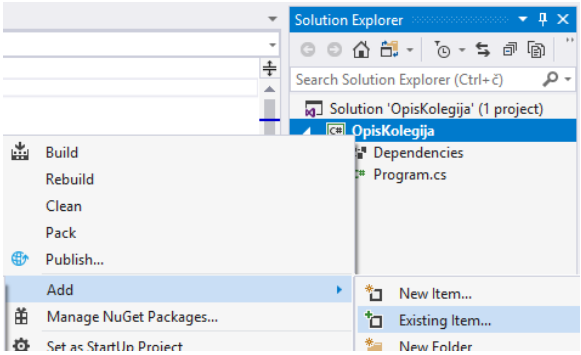
3.1 Prijelaz na Visual Studio

Kako bi olakšali izradu aplikacije korištena je razvojna okolina Visual Studio. U programu Visual Studio klikom na gumb *File>New>Project* otvara se sljedeći prozor.



Slika 15. Novi projekt
Izvor: Autor

Kako bi aplikacija radila ispravno potrebno je odabrati Visual C# > .NET Core. Klikom na gumb „OK“ stvara se novi projekt na odabranoj lokaciji. Na tu lokaciju potrebno je premjestiti datoteku „Models.cs“. Nakon toga datoteku dodajemo u projekt desnim klikom na naziv projekta i odaberemo *Add>Existing Item…* Kao što je prikazano na slici ispod.



Slika 16. Novi projekt
Izvor: Autor

4. APLIKACIJA

Kod izrade aplikacije koriste se LINQ (.NET Language-Integrated Query) upiti [5]. U narednim poglavljima biti će prikazan i objašnjen kod kojim se može upisivati, brisati, mijenjati i čitati podatke iz baze podataka. U glavnom programu koji se nalazi u datoteci „Program.cs“ potrebno je napraviti vezu do kontekst razreda kodom:

```
using (var db = new OpisKolegijaContext())
{


}
```

Varijabla *db* je isntanca razreda *OpisKolegijaContext* koji nasljeđuje razred *DbContext* koji sadrži metode za komuniciranje s bazom. Sav kod biti će pisan unutar vitičastih zagrada. Kod pokretanja aplikacije otvara se glavni izbornik koji izgleda ovako:

```
OPIS KOLEGIJA
1 - Ispis
2 - Uređivanje
3 - Brisanje
4 - Upis
5 - DEMO upis prvog semestra
6 - Tablični ispis prvog semestra
0 - Kraj
```

Upisom broja u konzolu i pritiskom na tipku „ENTER“ korisnik odabire koju operaciju želi izvršavati. Nakon odabira otvara se novi pod izbornik s preciznijim operacijama npr. ako korisnik odabere „Brisanje“ otvara se izbornik gdje korisnik odabire što točno želi obrisati. Upisom broja 0 izlazi se iz programa. Izbornik je realiziran *while* petljom koje ispisuje izbornik i čeka upis. *If* izraz provjerava ako je korisnik unio neki od ponuđenih brojeva s izbornika, ako nije petlja se ponavlja. Pošto velik broj opcija zahtjeva odabir kolegija napisana je metoda *OdaberiKolegij* prikazana na slici 17.

```
public static Kolegij OdaberiKolegij(List<Kolegij> kolegiji) {
    Console.Clear();
    Console.WriteLine("ODABERI KOLEGIJ");
    if (kolegiji.Count == 0) {
        Console.WriteLine("Nema upisanih kolegija, pritisnite " +
            "tipku za nastavak");
        Console.ReadKey();
        return null;
    }

    int i = 1;

    foreach (Kolegij kol in kolegiji)
        Console.WriteLine(i++ + " - " + kol.Naziv + " (Verzija: "
            + kol.Verzija + ")");
    int upis;
    try { upis = Int32.Parse(Console.ReadLine()); }
    catch (Exception e) { return null; }
    if (upis-1 > kolegiji.Count || upis<1)
    {
        Console.Clear();
        return null;
    }
    return kolegiji.ElementAt(upis-1);
}
```

Slika 17. Metoda OdaberiKolegij
Izvor: Autor

Metoda prima listu *kolegiji* kojoj provjerava broj elemenata tj. kolegija. Ukoliko je lista prazna ispisuje se prikladna poruka i čeka se potvrda od korisnika nakon čega metoda vraća vrijednost *null*. Ako se uspostavi da lista nije prazna ona se ispisuje kao u primjeru ispod

```
ODABERI KOLEGIJ
1 - Engleski jezik 1 (Verzija: 1)
2 - Tjelesna i zdravstvena kultura I (Verzija: 1)
3 - FIZIKA (Verzija: 1)
4 - Matematika 1 (Verzija: 1)
5 - EKONOMIKA I ORGANIZACIJA POSLOVNIH SUSTAVA (Verzija: 1)
6 - Osnove elektrotehnike i elektronike (Verzija: 1)
```

Nakon ispisa čeka se upis korisnika. Ako korisnik ne upiše valjanu vrijednost (npr. slova, znakove ili u ovom slučaju broj koji je manji od 1 ili veći od 6) metoda vraća *null* vrijednost. Na kraju ako sve dobro prođe funkcija bi trebala vratiti odabrani kolegij. Naredna poglavlja pojašnjavaju izvedbu svaku do opcija s glavnog izbornika.

4.1 Ispis

Opcija „Ispis“ sadrži metode za koje dohvaćaju i ispisuju podatke iz baze podataka. Izbornik ispisa izgleda ovako:

```
ISPIS
1 - Ispis svih kolegija
2 - Ispis kolegija za određeni semestar
3 - Zbroj ECTS bodova za semestar
4 - Ispis plana nastave
5 - Ispis predmeta za osobu
6 - Ispis literature za kolegij
7 - Ispis nastave
8 - Ispis osoba povezanih s predmetom
9 - Ispis ishoda za kolegij
10 - Ispis satnice kolegija
11 - Ispis uvjeta za kolegij
12 - Ispis vrsta izvođenja
13 - Udio ECTS bodova
0 - NATRAG
```

```
int i = 1;
var kolegiji = db.Kolegiji.ToList();
foreach (Kolegij kol in kolegiji)
{
    Console.WriteLine(i + " - " + kol.Naziv + " (Verzija: " + kol.Verzija + ")");
    i++;
}
```

Slika 18. Ispis svih kolegija
Izvor: Autor

Opcija „Ispis svih kolegija“ ispisuje sve kolegije koji se nalaze u bazi podataka. Kod ispisa koristi se jedna varijabla koja je u ovom slučaju *kolegiji* u koju se dohvaćaju svi kolegiji koji se nalaze u bazi. *Foreach* petlja prolazi se po listi i ispisuje nazive kolegija i njihove verzije u zagradama:

```
Ispis svih kolegija

1 - Engleski jezik 1 (Verzija: 1)

2 - Tjelesna i zdravstvena kultura I (Verzija: 1)
```

```
3 - FIZIKA (Verzija: 1)

4 - Tjelesna i zdravstvena kultura I (Verzija: 2)
```

Opcija „Ispis kolegija za određeni semestar“ ispisuje kolegije za upisani semestar, opcija „Zbroj ECTS bodova za semestar“ zbraja ECTS bodove koji se nalaze u tablici *Kolegij* za upisani semestar, opcija „Ispis plana nastave“ ima svoj podizbornik koji se otvara nakon odabira kolegija koji izgleda ovako:

```
PLAN NASTAVE ZA Engleski jezik 1
1 - Ispis 15 tjednog plana nastave
2 - Ispis 15 tjednog plana nastave samo predavanja
3 - Ispis 15 tjednog plana nastave samo vježbe
4 - Ispis teme predavanja za točno određeni tjedan
0 - NATRAG
```

Svaka od ovih opcija ispisuje tjedni plan na drugačiji način. Opcija „Ispis predmeta za osobu“ nakon odabira osobe ispisuje sve kolegije sa svim tipovima veza (nositelj, predavač, suradnik) povezane s tom osobom, opcija „Ispis literature za kolegij“ ispisuje obaveznu i neobaveznu literaturu za odabrani kolegij, opcija „Ispis nastave“ ispisuje sve veze koje se nalaze u tablici *OpceInformacije*.

```
var kolegij = OdaberiKolegij(db.Kolegiji.ToList());
if (kolegij == null)
    continue;
var oi = db.OpceInformacije.Where(x => x.KolegijId == kolegij.KolegijId).ToList();
Console.Clear();
Console.WriteLine("KOLEGIJ: " + kolegij.Naziv);
Console.WriteLine();
foreach (OpceInformacije o in oi)
{
    var osobe = db.Osobe.Single(x => x.OsobaId == o.OsobaId);
    var vrsta = db.Vrste.Single(x => x.VrstaId == o.VrstaId);
    Console.WriteLine(vrsta.Tip + ": " + osobe.Prefiks + " " + osobe.Ime + " " +
        osobe.Prezime + " " + osobe.Sufiks);
}
```

Slika 19. Ispis osoba povezanih s predmetom
Izvor: Autor

Opcija „Ispis osoba povezanih s predmetom“ ispisuje sve osobe koje su u tablici *OpceInformacije* povezane s odabranim kolegijem. Koristeći LINQ upite dobiju se filtrirane upite od baze podataka. Na slici 19 prikazan je kod koji ispisuje sve osobe koje su povezane s odabranim predmetom. Metoda *OdaberiKolegij* prima listu kolegija, ispiše

ih i vraća onog koji korisnik odabere. Koristeći upit *Where* u varijablu *oi* spremaju se sve *OpceInformacije* koje sadrže vezu s odabranim kolegijem. *OpceInformacije* uz vezu na kolegij još sadrže vezu na osobu i vezu na tip osobe (nositelj, suradnik, predavač). *Foreach* petlja prolazi po listi *oi*, za svaki element dohvaća osobu i tip osobe upitom *Single* koji vraća samo jedan rezultat i ispisuje ga. Ispis osoba vezanih za engleski jezik izgleda ovako:

```
KOLEGIJ: Engleski jezik 1

Nositelj: prof. Profesor nositelj
Predavač: prof. Profesor predavač
Suradnik: dr.sc. Profesor suradnik
```

Opcija „Ispis ishoda za kolegij“ ispisuje ishode za odabrani kolegij, opcija „Ispis satnice kolegija“ za odabrani kolegij ispisuje broj sati predavanja i vježbi koje su zapisane u tablici *Kolegij* kao i zbroj sati iz tablice *Opis* (tablica koja sadrži tjedni plan), opcija „Ispis uvjeta za kolegij“ ispisuje uvjete potrebne za upis na odabrani kolegij, opcija „Ispis vrsta izvođenja“ ispisuje sve vrste nastave (predavanja, seminari, vježbe, laboratorij…) za odabrani kolegij, opcija „Udio ECTS bodova“ ispisuje udio ECTS bodova u postocima za odabrani kolegij.

4.2 Uređivanje

Opcija „Uređivanje“ sadrži metode za koje dohvaćaju, mijenjaju i spremaju podatke iz baze podataka. Izbornik uređivanja izgleda ovako:

```
UREĐIVANJE
1 - Uredi tjedni program
2 - Uredi literaturu
3 - Uredi ishode
4 - Uredi osobe na predmetu
5 - Uredi uvjet za upis
6 - Promjena kolegija
7 - Upisi sve vrste nastave za kolegij
8 - Upisi sve vrste PracenjeRada
0 - NATRAG
```

Opcija „Uredi tjedni program“ za odabrani kolegij i tjedan omogućava promjenu satnice i opisa tjednog programa. Opis sadrži podatke o petnaest tjednom rasporedu nastave što znači da svaki kolegij ima petnaest opisa. Princip rada je sljedeći.

```
var opisi = db.Opis.Where(x => x.KolegijId == 1).ToList();
foreach (Opis o in opisi) {
    o.SatiPredavanja = 0;
    o.SatiVjezbe = 4;
    db.Opis.Update(o);
}
db.SaveChanges();
```

Slika 20. Uređivanje opisa
Izvor: Autor

Za dohvat opisa na slici 20 korišten je upit *Where* koji vraća više rezultata za koje vrijedi uvjet u zagradi što znači da će upit vratiti sve rezultate kojima je strani ključ „KolegijId“ jednak jedan. Metodom *ToList* dobiveni rezultate stavljaju se u listu koja se sprema u varijablu opisi. *Foreach* petlja prolazi po listi i mjenja svakome opisu u listi broj sati predavanja u nula i broj sati vježbe u 4. Promijenjeni opis stavljamo kao argument u metodu *Update*. Na kraju se promjene spremaju u bazu podataka metodom *SaveChanges*.

Opcija „Uredi literaturu“ omogućava korisniku promjenu opisa literature za odabrani kolegij, opcija „Uredi ishode“ slično kao i kod literature omogućava uređivanje ishoda za odabrani kolegij, opcija „Uredi osobe na predmetu“ omogućava izmjenu podataka o vezi između kolegija i osobe tj. mijenja podatke u tablici *OpceInformacije*, opcija „Uredi uvjet za upis“ omogućava izmjenu uvjeta potrebnih za upis u odabrani kolegij, opcija „Promjena kolegija“ mijenja podatke o opisu verzije kolegija, opcija „Upisi sve vrste nastave za kolegij“ za odabrani kolegij u tablici *VrsteIzvodenja* postavlja sve vrijednosti na *true*, opcija „Upisi sve vrste PracenjeRada“ u tablici *PracenjeRada* postavlja sve vrijednosti na 0.076 kako bi zbroj bio približno 1.

4.3 Brisanje

Opcija „Brisanje“ sadrži metode za koje brišu podatke iz baze podataka. Izbornik brisanja izgleda ovako:

```
BRISANJE
1 - Briši kolegij
2 - Brisanje literature
3 - Brisanje nastave
4 - Brisanje ishoda
5 - Brisanje uvjeta
6 - Obrisi sve vrste nastave za kolegij
7 - Obrisi sve vrste PracenjeRada
0 - NATRAG
```

Opcija „Briši kolegij“ briše odabrani kolegij i sve njegove poveznice iz baze podataka. Brisanje se radi tako da prvo se prvo dohvati podatak iz baze i onda se taj podatak daje metodi za brisanje.

```
var kolegiji = db.Kolegiji.ToList();

var kolegij = OdaberiKolegij(kolegiji);
if (kolegij == null)
    continue;
db.Kolegiji.Remove(kolegij);
db.SaveChanges();
Console.WriteLine("Uspješno obrisano");
Console.ReadKey();
```

Slika 21. Brisanje kolegija
Izvor: Autor

Ovim kodom u varijablu *kolegij* dohvaćen je jedan rezultat kojemu je primarni ključ jednak jedan i ta varijabla stavlja se kao argument metodi *Remove*. Nakon toga metodom *SaveChanges* spremaju se promjene. Važno je napomenuti da će se na ovaj način obrisati svi podaci koji sadrže strani ključ kolegija koji se briše.

Opcije „Brisanje literature“, „Brisanje ishoda“, „Brisanje uvjeta“ za odabrani kolegij brišu odabranu literaturu, ishod ili uvjet ovisno o odabranoj opciji. Opcija „Obrisi sve vrste nastave za kolegij“ postavlja sve vrijednosti u tablici *VrsteIzvodenja* na *false*, opcija „Obrisi sve vrste PracenjeRada“ postavlja sve vrijednosti u tablici *PracenjeRada* na 0.

4.4 Upis

Opcija „Upis“ sadrži metode koje upisuju podatke u bazu podataka. Izbornik upisa izgleda ovako:

```
UPIS
1 - Upis literature
2 - Upis osobe
3 - Povezivanje osobe s kolegijem
4 - Upis novog ishoda
5 - Upis uvjeta za upis
6 - Upis nove verzije kolegija
7 - Upis vrste
0 - NATRAG
```

Opcija „Upis literature“ omogućava upis obavezne i neobavezne literature za odabrani kolegij, opcija „Upis osobe“ omogućava upis u tablicu *Osobe*.

```
Osoba o = new Osoba();
Console.Write("Upisi prefiks: ");
o.Prefiks = Console.ReadLine();
Console.Write("Upisi sufiks: ");
o.Sufiks = Console.ReadLine();
Console.Write("Upisi ime: ");
o.Ime = Console.ReadLine();
Console.Write("Upisi prezime: ");
o.Prezime = Console.ReadLine();

db.Osobe.Add(o);
db.SaveChanges();
Console.WriteLine("Spremljeno");
```

Slika 22. Upis osobe
Izvor: Autor

Kod upisa podataka u bazu preko aplikacije prvo je potrebno stvoriti novu instancu modela koji se želi upisati. U slučaju prikazanom na slici 22 to je model *Osoba*. Kod popunjavanja svojstva modela, moraju se popuniti samo ona koja su obavezna, a takvih ovdje nema. Iako *OsobaId* treba obavezno popuniti (jer je to primarni ključ) ovdje se to ne radi jer baza tijekom upisa sama zadaje primarni ključ. Ako se želi postaviti svojstvo da bude obavezno u modelu možemo koristiti anotaciju *Required*

```
[Required]
public string Prijmer { get; set; }
```

Zadnja dva reda koda sa slike 22 koriste nam za spremanje podataka u bazu. Metoda *Add* dodaje , a *SaveChanges* sprema podatke u bazu. Kad se jednom osoba spremi u bazu to jest nakon izvršavanja metode *SaveChanges* u varijablu *o* se sprema primarni ključ koji je zadala baza. Opcija „Povezivanje osobe s kolegijem“ omogućava upis u tablicu *OpceInformacije* tj. spajanje kolegija s osobama i tipom osoba.

```
OpceInformacije oi = new OpceInformacije();
oi.KolegijId = k.KolegijId;
oi.VrstaId = db.Vrste.Single(x => x.VrstaId == 1).VrstaId;
oi.OsobaId = db.Osobe.Single(x => x.OsobaId == 1).OsobaId;
db.OpceInformacije.Add(oi);
db.SaveChanges();
```

Slika 23. Upis u OpceInformacije
Izvor: Autor

U *OpceInformacije* upisuju se 3 strana ključa, prije ovog upisa potrebno je upisati podatke u tablicu *Osoba* i *Vrsta* kako bi mogli dobiti njihove primarne ključeve. U primjeru na slici 17 ključeve za osobu i vrstu dohvaćaju se iz baze preko instance *OpisKolegijaContext db* koristeći metodu *Single* koja vraća jedan rezultat iz baze s primarnim ključ koji je jednak jedan. Dobiveni primarni ključevi spremaju se u svojstva modela *OpceInformacije* kao strani ključevi. Na kraju se koriste metode *Add* i *SaveChanges* kako bi spremili podatke u bazu.

Opcija „Upis novog ishoda“ omogućava korisniku upis novog ishoda za odabrani kolegij, opcija „Upis uvjeta za upis“ za odabrani kolegij omogućava upis novog uvjeta potrebnog za upis kao i upis kolegija na koji se taj uvjet odnosi npr. kolegij „Engleski Jezik II“ ima uvjet „Položen“ za kolegij Engleski jezik I, opcija „Upis nove verzije kolegija“ ponovo upisuje odabrani kolegij s novom verzijom, opcija „Upis vrste“ upisuje novi tip u tablicu *Vrste*.

4.5 Upis nove verzije kolegija

```
Kolegij kolegij = OdaberiKolegij(db.Kolegiji.ToList());
if (kolegij == null)
    continue;

db.Entry(kolegij).State = Microsoft.EntityFrameworkCore.EntityState.Added;
Console.Clear();
Console.WriteLine("Nova verzija kolegija " + kolegij.Naziv);
Console.WriteLine("Upiši opis verzije:");
kolegij.KolegijId = 0;
kolegij.Verzija += 1;
kolegij.Opis = Console.ReadLine();
kolegij.Stvoren = DateTime.Now;


db.SaveChanges();
Console.WriteLine("Spremljeno");

Console.ReadKey();
```

Slika 24. Upis nove verzije kolegija
Izvor: Autor

Na slici 24 prikazan je kod koji se izvršava ukoliko korisnik odabere opciju 6 s izbornika „Upis“ koje stvara novu verziju postojećeg kolegija. U varijablu *kolegij* sprema se kolegij odabrani s izbornika. Slijedi kod

```
db.Entry(kolegij).State =
Microsoft.EntityFrameworkCore.EntityState.Added;
```

Kojim postavljamo status varijable *kolegij* u *Added* čime EF zna da je ovo novi entitet. Metoda *SaveChanges* s entitetom radi različite stvari ovisno u kojem je stanju [6].

- *Unchanged* – metoda ignorira ovakve entitete
- *Added* – metoda sprema entitete u bazu i postavlja ih u *Unchanged*
- *Modified* – metoda ažurira entitete i postavlja ih u *Unchanged*
- *Deleted* – metoda briše ovakve podatke iz baze podataka

Prije spremanja svojstvo *Verzija* postavlja se za 1 veće od prijašnje, svojstvo *Stvoren* postavlja se na trenutni datum i u *Opis* potrebno je upisati neki tekst koji opisuje razlog stvaranja nove verzije.

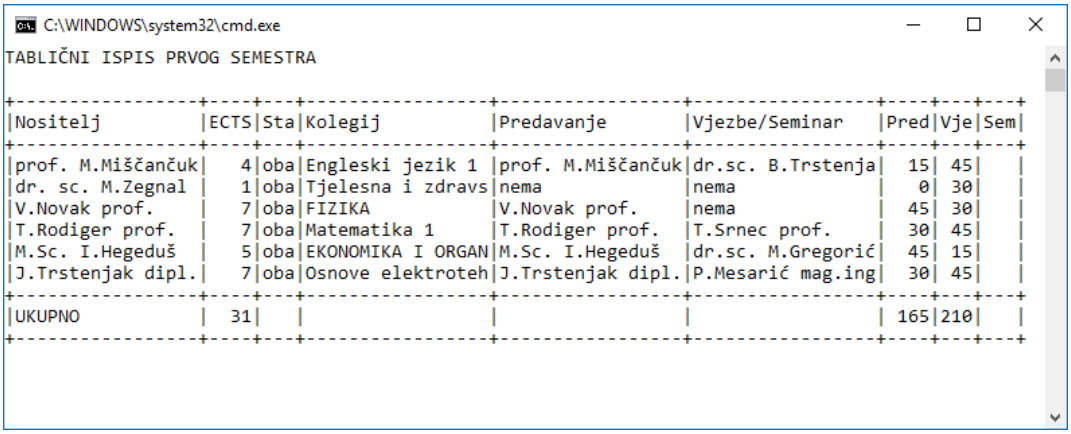
4.6 Demo upis prvog semestra

Opcija „DEMO upis prvog semestra“ poziva metodu *UpisPrvogSemestra* iz razreda *DemoUpis* koja u bazu podataka upisuje sve kolegije i ostale podatke vezene uz kolegij kao što su literatura, opis, ishodi itd. Kako se ne bi narušila preglednost glavnog programa zbog velike količine koda i podataka koje ova opcija sadrži stvoren je novi razred s nazivom *DemoUpis*. Kod odabira ove opcije ispis u konzoli izgleda ovako:

```
Upisivanje:
(0/6)
(1/6) Neuspješno upisan Engleski Jezik 1
(2/6) Neuspješno upisan Tjelesna i zdravstvena kultura I
(3/6) Uspješno upisan Matematika 1
(4/6) Uspješno upisan EKONOMIKA I ORGANIZACIJA POSLOVNIH…
(5/6) Neuspješno upisan FIZIKA
(6/6) Uspješno upisan Osnove elektrotehnike i elektronike
Upis završen
```

Metoda *UpisPrvogSemestra* za svaki od kolegija poziva metodu koja upisuje taj kolegij i vraća podatak tip *string* koji ispisuje u konzolu. Ukoliko je upis kolegija bio uspješan metoda vraća „Uspješno upisan <Naziv Kolegija>“, a ukoliko je upis neuspješan metoda vraća „Neuspješno upisan <Naziv Kolegija>“. U slučaju primjera navedenog gore 3 od 6 kolegija već su bila u bazi pa ih metoda nije uspjela upisati dok su ostali uspješno upisani.

4.7 Tablični ispis prvog semestra



Slika 25. Tablični ispis prvog semestra
Izvor: Autor

Opcija „Tablični ispis prvog semestra“ ispisuje podatke prvog semestra u obliku tablice kao što je prikazano na slici 25. Prije ispisa tablice potrebno je dohvatiti sve podatke i odrediti dužinu za svaki stupac. Na slici 26 prikazan je kod kojim je određen sadržaj i dužina stupca „Nositelj“. Za spremanje sadržaja stupca tj. imena nositelja korištena je lista s nazivom *nositelji*, za spremanje najveće duljine stupca korištena je varijabla tipa *int* s nazivom *nositelj* s početnom vrijednošću 17. Nakon dohvata podataka korištenjem *Where* upita u varijablu *kolegiji*, *foreach* petljom prolazi se po listi kolegija i za svaki u varijablu *nos* dohvaćaju se podaci iz tablice *OpceInformacije* koji imaju jednak id kolegija i id vrste jednak 1(tip osobe pod id 1 je nositelj). *If* izraz provjerava da li u varijabli *nos* postoje dohvaćeni podaci, ako ne postoje u varijablu *n* sprema se *string* „nema“. Varijabla *n* koristi za spajanje imena, prezimena, prefiksa i sufiksa nositelja koji se na kraju sprema u listu. Sljedeća *foreach* petlja dohvaća osobe, formatira ih i provjerava da li je duljina veća od one koja se nalazi u varijabli *nositelj*. Na kraju *n* se sprema u listu nositelji. Isti princip korišten je i za stupce „Predavaci“ i „Vjezbe/seminar“, a ostali stupci koriste statičke duljine.

```
List<string> nositelji = new List<string>();
int nositelj = 17;
var kolegiji = db.Kolegiji.Where(x=>x.Semestar==1).ToList();
foreach (Kolegij k in kolegiji) {

    //dodavanje nositelja u listu
    var nos = db.OpceInformacije.Where(x => x.KolegijId == k.KolegijId
        && x.VrstaId==1);
    string n="";
    if (nos.Count() == 0) n = "nema";
    foreach (OpceInformacije o in nos) {
        Osoba osoba = db.Osobe.Single(x => x.OsobaId == o.OsobaId);
        if (osoba.Prefiks == "" || osoba.Prefiks == null)
        {
            n += String.Format("{0}.{1} {2}", osoba.Ime.Remove(1),
                osoba.Prezime, osoba.Sufiks);
        }
        else {
            n += String.Format("{0} {1}.{2} {3}", osoba.Prefiks,
                osoba.Ime.Remove(1), osoba.Prezime, osoba.Sufiks);
        }

        if (nositelj < n.Length)
           nositelj = n.Length;
        nositelji.Add(n);
    }
```

Slika 26. Određivanje duljine stupca „Nositelj“
Izvor: Autor

Prije ispisivanja podataka potrebno je ispisati okvir tablice. To radi metoda *ispisOkvira*. Metoda prima 4 parametra koji joj služe kako bi znala dužinu svakog polja.

```
public static void ispisOkvira(int nositelj, int kolegij, int predavac, int vjezbe) {
    Console.Write("+");
    for (int i = 0; i < nositelj; i++)
    {
        Console.Write("-");
    }
    Console.Write("+----+---+");
    for (int i = 0; i < kolegij; i++)
    {
        Console.Write("-");
    }
    Console.Write("+");
    for (int i = 0; i < predavac; i++)
    {
        Console.Write("-");
    }
    Console.Write("+");
    for (int i = 0; i < vjezbe; i++)
    {
        Console.Write("-");
    }
    Console.Write("+----+---+---+");
    Console.Write("\n");
```

Slika 27. Metoda za iscrtavanje ruba tablice
Izvor: Autor

5. ZAKLJUČAK

Izradom ovog završnog rada pokazani su koraci potrebni za izradu jedne aplikacije kao i baze podataka korištenjem pristupa „Model First“. Glavna tema bila je izrada modela no nastojalo se prikazati korištenje .NET Core-a i Entity Framework-a kako bi postigli i jednu funkcionalnu aplikaciju. Rad prikazuje postupak izrade modela, stvaranje baze, postavljanje projekta, izrada aplikacije, spajanje s bazom i operacije nad bazom. Rezultat ovog rada tj.baza i modeli mogu se koristi kao poboljšanja za već postojeću aplikaciju na čijoj je bazi bio i baziran ovaj rad.

6. POPIS LITERATURE

[1] .NET Core – New Database - https://docs.microsoft.com/en-us/ef/core/get-started/netcore/new-db-sqlite (17.6.2018)

[2]ModelFirst - https://docs.microsoft.com/en-us/ef/ef6/modeling/designer/workflows/model-first (17.6.2018)

[3] Creating a Model – Keys (primary) - https://docs.microsoft.com/en-us/ef/core/modeling/keys (17.6.2018)

[4] Migrations - https://docs.microsoft.com/en-us/ef/core/managing-schemas/migrations/ (17.8.2018)

[5] LINQ - https://msdn.microsoft.com/en-us/library/bb308959.aspx (17.8.2018)

[6] Entity States - https://docs.microsoft.com/en-us/ef/ef6/saving/change-tracking/entity-state (17.8.2018)

[7] John Kocer (2018.) Professional Entity Framework Core 2.0 & 6.x with Examples: .NET Core, C#, Entity Framework Core