

Laboratorijska Vježba 8. ASP.NET Core MVC Perzistencija podataka

Cilj vježbe:

- Upoznavanje sa načinom korištenja različitih besplatnih *cloud* baza podataka (Azure i *FreeSQLDatabase*);
- Konfiguracija ASP.NET aplikacije za korištenje baze podataka;
- Migracija modela podataka iz ASP.NET aplikacije u bazu podataka.

1. Konfiguracija *cloud* baze podataka

U prethodnim laboratorijskim vježbama kreirana je ASP.NET Core aplikacija koristeći MVC arhitekturni patern. U toj aplikaciji definisan je model podataka, a zatim su automatski generisani kontroleri i pogledi za koje je zatim definisan način rutiranja i način prikaza formi. Sljedeći korak koji je potrebno izvršiti je **omogućavanje perzistencije podataka**, odnosno povezivanje aplikacije sa izvorom podataka u obliku baze podataka. Prvo je potrebno odabrati adekvatnu bazu podataka koja će se koristiti kao izvor – lokalnu ili *cloud* bazu podataka.

Ukoliko je moguće, bolje je odabrati *cloud* bazu podataka kao izvor podataka. Ovakva baza podataka dostupna je svim osobama koje razvijaju aplikaciju i ako se aplikacija pokreće lokalno. Promjene nad bazom podataka vidljive su svima bez potrebe za manuelnom sinhronizacijom. Bez *cloud* baze podataka nije moguće izvršiti *deployment* aplikacije na udaljeni server. Alternativno, može se koristiti lokalni računar ili neki svoj server za instancu sistema za upravljanje bazama podataka. Ti sistemi mogu biti *Microsoft SQL server*, *Oracle Database*, *PostgreSQL*, *MySQL*, *MariaDB* i drugi – većina ovakvih rješenja podržana je za rad sa .NET Core aplikacijama. U ovoj laboratorijskoj vježbi demonstrirati će se korištenje dvije *cloud* baze podataka, a studentima se ne preporučuje korištenje lokalnih baza podataka.

1.1. *FreeSQLDatabase cloud* baza podataka

FreeSQLDatabase je besplatna relacionalna baza podataka koja se može koristiti kao izvor podataka za ASP.NET aplikacije. Dostupna je na sljedećem linku: <https://www.freesqldatabase.com>, gdje je potrebno odabrati besplatni **MySQL Free** plan i registrovati novi korisnički račun. Nakon toga korisnik biva preusmjeren na početni ekran koji je prikazan na Slici 1. Prvo je potrebno odabrati lokaciju baze podataka (najbolje je odabrati **Europe (Mainland)** jer je to najbliža lokacija koja omogućava najbrži pristup), a zatim odabrati opciju **Start new database** kako bi se mogla kreirati nova baza podataka koja će se zatim koristiti kao izvor podataka za prethodno kreiranu ASP.NET Core aplikaciju.

Objektno Orijentisana Analiza i Dizajn

Server Location

Select where you would like your database located.

Please Select

Save Location

Account Details

Account Number	Available Databases	Available Space
490275	1 of 1	0% of 5.00MB

Database Details

Database Host	Database Name	Database Username	Database Password	Size	Status	Delete
---------------	---------------	-------------------	-------------------	------	--------	--------

Start new database

phpMyAdmin for database administration

► Follow this link for phpMy Admin

Slika 1. Početni ekran za Free SQL Database korisnički račun

Nakon toga kreira se baza podataka i na početnom ekranu se prikazuju njene osnovne informacije. Potrebno je sačekati nekoliko minuta kako bi se baza podataka osposobila i na početnom ekranu prikazali podaci koji su neophodni kako bi se izvršila prijava na bazu podataka (korisničko ime i šifra baze podataka, koja je poslana na email adresu korisnika). Izgled informacija o bazi podataka nakon njenog osposobljavanja prikazani su na Slici 2.

Account Details

Account Number	Available Databases	Available Space
490275	0 of 1	0% of 5.00MB

Database Details

Database Host	Database Name	Database Username	Database Password	Size	Status	Delete
sql11.freessqlatabase.com	sql11392600	sql11392600	Check your emails	0.00MB	Live	<input type="checkbox"/>

Delete database

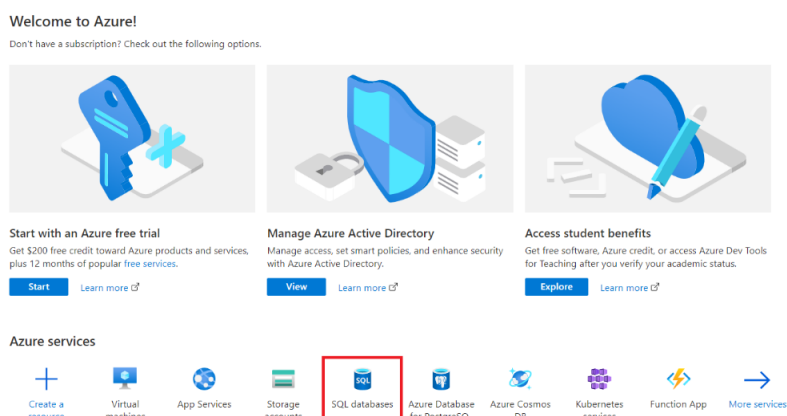
Slika 2. Prikaz informacija o bazi podataka

Konekcijski string za ovako kreiranu bazu podataka prikazan je u isječku koda ispod, gdje je **<host>** potrebno zamijeniti sa web-adresom baze, **<port>** potrebno zamijeniti sa brojem porta baze podataka, **<baza>** potrebno zamijeniti s imenom baze podataka, **<user>** potrebno zamijeniti s korisničkim imenom, a **<šifra>** potrebno zamijeniti sa korisničkom šifrom.

```
Server=<host>;Port=<port>;Database=<baza>;User=<user>;Password=<šifra>;
```

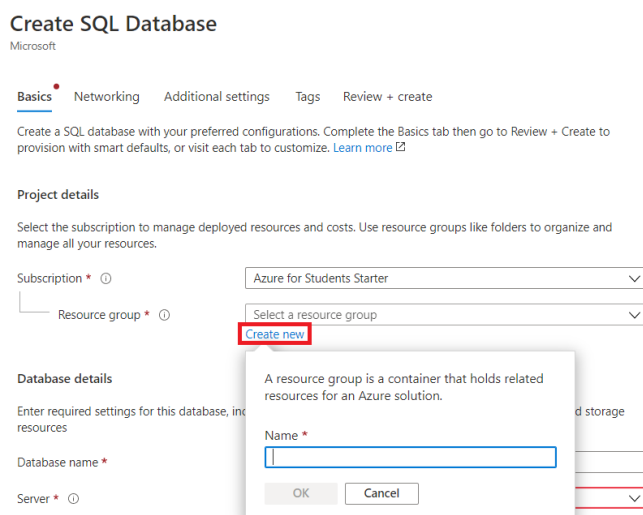
1.2. Microsoft Azure cloud baza podataka

Microsoft Azure je .NET cloud platforma sa velikim brojem servisa koje je moguće aktivirati i koristiti. Između ostalog, platforma nudi i korištenje baze podataka i vršenje *deploymenta* ASP.NET aplikacija. Novi račun moguće je registrovati putem sljedećeg linka: <https://azure.microsoft.com/en-us/>, kao i koristeći studentski *GitHub Education* korisnički račun i njegove dodatne servise¹. Nakon kreiranja korisničkog računa, na početnom ekranu (koji je prikazan na Slici 3) korisniku se prikazuje opcija **SQL Databases** koju je potrebno odabrati kako bi se kreirala nova baza podataka.



Slika 3. Prikaz početnog ekrana Azure portala

Nakon toga prikazuje se ekran sa svim bazama podataka, odakle je vidljivo da ne postoji nijedna predefinisana baza podataka. Potrebno je odabrati opciju **Add**, nakon čega je potrebno dodati osnovne informacije o novoj bazi podataka. Prvo je potrebno dodati novu grupu resursa odabirom opcije **Create new** ispod ovog polja (na način prikazan na Slici 4). Dozvoljeno je odabrati proizvoljno ime grupe resursa.

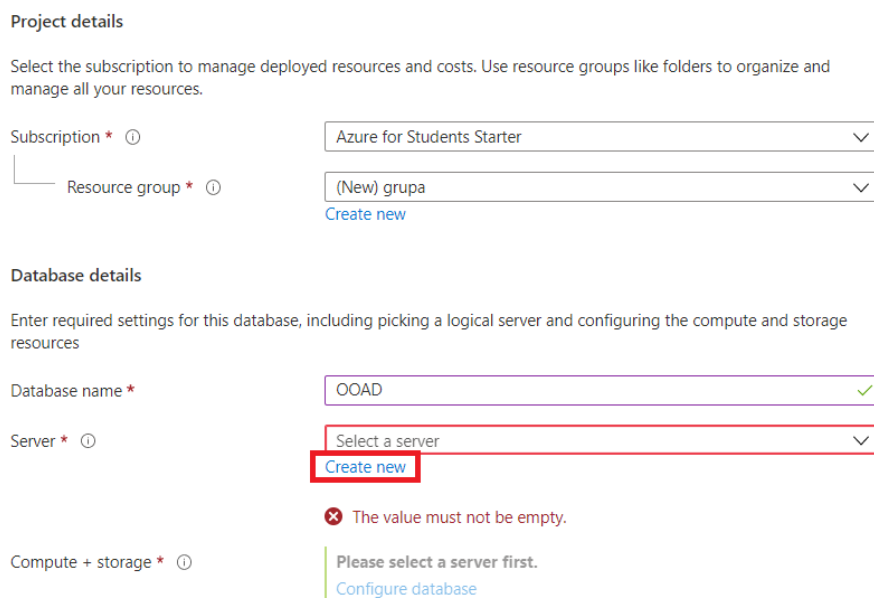


Slika 4. Odabir opcije za kreiranje grupe resursa

¹ Za korištenje ovog načina aktivacije mogu se koristiti uputstva na sljedećem linku: <https://manojbce.medium.com/activate-azure-for-students-using-github-student-developer-pack-389238de4f8>.

Objektno Orijentisana Analiza i Dizajn

Nakon toga potrebno je dodati **ime baze podataka**, a zatim kreirati novi server odabirom opcije **Create new** ispod ovog polja (prikazano na Slici 5). Specifikacija informacija o serveru vrši se sa desne strane ekrana i dozvoljen je odabir proizvoljnih vrijednosti za server i korisničke podatke.



Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ Azure for Students Starter

Resource group * ⓘ (New) grupa
[Create new](#)

Database details

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

Database name * OOAD ✓

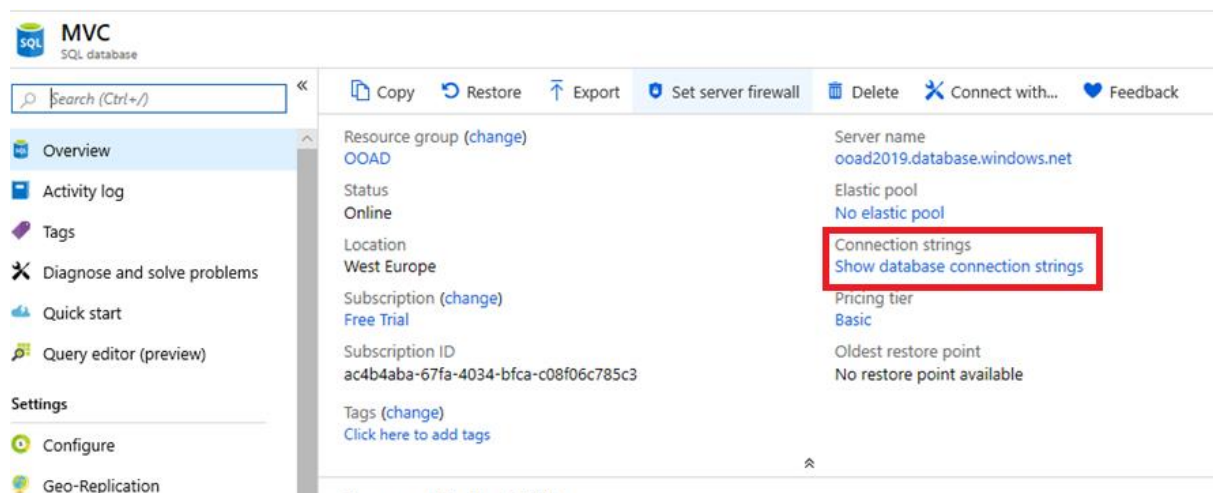
Server * ⓘ Select a server
[Create new](#)

✖ The value must not be empty.

Compute + storage * ⓘ Please select a server first.
[Configure database](#)

Slika 5. Odabir opcije za kreiranje novog servera

Nakon toga specifikacija informacija o bazi podataka je završena i potrebno je kreirati bazu odabirom opcije **Review + Create**, a zatim **Create**. Nakon toga potrebno je sačekati nekoliko minuta kako bi se baza podataka osposobila, a zatim se može početi sa njenim korištenjem. Konekcijski string za ovakvu bazu podataka moguće je preuzeti sa *Azure* portala, pristupom samoj bazi podataka nakon njenog kreiranja (baza podataka nalazi se na listi svih SQL baza podataka). U *Overview* prozoru potrebno je odabrati opciju **Show database connection strings** (prikazano na Slici 6), nakon čega se prikazuju različiti konekcijski stringovi za različite platforme.



MVC SQL database

Search (Ctrl+/)

Copy Restore Export Set server firewall Delete Connect with... Feedback

Resource group (change) OOAD

Status Online

Location West Europe

Subscription (change) Free Trial

Subscription ID ac4b4aba-67fa-4034-bfca-c08f06c785c3

Tags (change) Click here to add tags

Server name ooad2019.database.windows.net

Elastic pool No elastic pool

Connection strings Show database connection strings

Pricing tier Basic

Oldest restore point No restore point available

Slika 6. Odabir opcije za prikaz konekcijskih stringova

Konekcijski string koji će se koristiti odnosi se na ADO.NET platformu, a njegov format prikazan je u isječku koda ispod, gdje je **<host>** potrebno zamijeniti sa web-adresom baze, **<port>** potrebno zamijeniti sa brojem porta baze podataka, **<baza>** potrebno zamijeniti s imenom baze podataka, **<user>** potrebno zamijeniti s korisničkim imenom, a **<sifra>** potrebno zamijeniti sa korisničkom šifrom.

```
Server=tcp: <host>,<port>;Initial Catalog= <baza>;Persist Security Info=False;User ID=<user>;Password=<sifra>;MultipleActiveResultSets=False;Encrypt=True;TrustServer Certificate=False;Connection Timeout=30;
```

2. Konfiguracija ASP.NET aplikacije za korištenje baze podataka

Prethodno kreiranu *cloud* bazu podataka sada je moguće iskoristiti kao izvor podataka za ASP.NET web-aplikaciju koja je kreirana u prethodnim laboratorijskim vježbama. Prvo je potrebno dodati konekcijski string za bazu podataka u **appsettings.json** file. Kako je u prethodno kreiranoj aplikaciji pri *scaffold*-anju kontrolera bilo neophodno dodati kontekst podataka, u listi već postoji definisan jedan konekcijski string koji se odnosi na taj kontekst podataka, koji je potrebno promijeniti tako da odgovara konekcijskom stringu kreiranom u prethodnom poglavlju. U slučaju da kontekst nije prethodno definisan, moguće je dodati novi string na način prikazan u isječku koda ispod (pri čemu je prikazani konekcijski string potrebno zamijeniti konekcijskim stringom korisnika):

```
"ConnectionStrings": {  
  "StudentskaSluzbaContext":  
    "Server=(localdb)\\mssqllocaldb;Database=StudentskaSluzbaContext-c33f65f8-bcfa-4dfb-b638-026b61d068ff;Trusted_Connection=True;MultipleActiveResultSets=true"  
}
```

Nakon toga potrebno je u **Startup.cs** file-u provjeriti da li je kontekst ispravno definisan u funkciji **ConfigureServices**. U isječku koda ispod prikazan je programski kod ove metode, koji treba pozvati funkciju *AddDbContext* pri čemu se kao parametar šalje konekcijski string istog imena koje je u prošlom koraku definisan u *appsettings* file-u.

```
// This method gets called by the runtime. Use this method to add services  
to the container.  
public void ConfigureServices(IServiceCollection services)  
{  
    services.AddControllersWithViews();  
  
    services.AddDbContext<StudentskaSluzbaContext>(options =>  
options.UseSqlServer(Configuration.GetConnectionString("StudentskaSluzbaContext"))  
);  
}
```

Sljedeće što je potrebno uraditi je **definicija konteksta podataka**. Automatski generisani kontekst podataka nalazi se u folderu **Data** i posjeduje samo jedan atribut tipa klase *DbSet<Predmet>* (sama klasa prikazana je u isječku koda ispod), jer je ovaj kontekst automatski generisan u svrhu *scaffold*-anja kontrolera za klasu *Predmet*. Ovaj kontekst sada je neophodno generalizirati kako bi se mogao koristiti za sve klase iz baze podataka.

```
public class StudentskaSluzbaContext : DbContext
{
    public StudentskaSluzbaContext (DbContextOptions<StudentskaSluzbaContext>
options)
        : base(options)
    {
    }

    public DbSet<StudentskaSluzba.Models.Predmet> Predmet { get; set; }
}
```

U kontekst podataka potrebno je dodati ostale klase modela podataka koje se žele sačuvati u bazu podataka (to su klase *Student* i *UpisNaPredmet*), a potrebno je i definisati metodu **OnModelCreating** u okviru koje se specificiraju proizvoljni nazivi tabela podataka. Kako se *default* nazivi tabela kreiraju na engleskom jeziku, često pristup tabelama postaje zbunjujući ukoliko se imena tabela eksplicitno ne definišu. Na ovaj način dobiva se definicija konteksta podataka prikazana u isječku koda ispod.

```
public class StudentskaSluzbaContext : DbContext
{
    public StudentskaSluzbaContext (DbContextOptions<StudentskaSluzbaContext>
options)
        : base(options)
    {
    }

    public DbSet<Student> Student { get; set; }
    public DbSet<UpisNaPredmet> UpisNaPredmet { get; set; }
    public DbSet<Predmet> Predmet { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Student>().ToTable("Student");
        modelBuilder.Entity<UpisNaPredmet>().ToTable("UpisNaPredmet");
        modelBuilder.Entity<Predmet>().ToTable("Predmet");
    }
}
```

Na ovaj način proces konfiguracije ASP.NET aplikacije za konekciju na bazu podataka je završen i sada se može izvršiti migracija na bazu podataka i početak korištenja aplikacije sa perzistencijom podataka.

3. Migracija modela podataka iz ASP.NET aplikacije u bazu podataka

Pri kreiranju ASP.NET aplikacija, postoje tri različita pristupa za objektno-relaciono mapiranje:

1. **Database-first**, pri čemu se prvo koristeći SQL kreira baza podataka sa svim tabelama, a zatim se vrši automatsko kreiranje modela podataka (klasa u C# programskom jeziku) na osnovu specifikacije kolona u tabelama podataka.

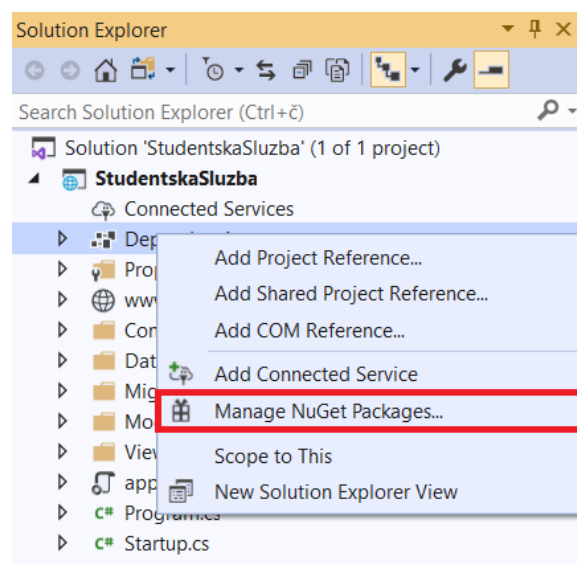
2. **Model-first**, pri čemu se koristeći neki dizajner šeme baze podataka vrši kreiranje šeme baze podataka, na osnovu čega se generiše struktura baze podataka i model podataka (klasa u C# programskom jeziku).
3. **Code-first**, pri čemu se prvo koristeći C# kreira model klasa podataka, a zatim se vrši automatsko kreiranje tabela u bazi podataka na osnovu specifikacije atributa u klasama modela podataka.

Kroz prethodne laboratorijske vježbe korišten je *code-first* pristup, jer je prvo kreiran model klasa podataka sa svim pripadajućim atributima i validacijskim ograničenjima, na osnovu čega se želi omogućiti perzistencija podataka. Nakon što je definisana konekcija na bazu podataka i kontekst podataka kojim se model podataka mapira u bazu podataka, potrebno je izvršiti **migraciju** ovih podataka u bazu podataka. Ovo zapravo znači da će se klase kojima je predstavljen model podataka automatski pretvoriti u tabele podataka u bazi podataka, na način da će se svi atributi (osim onih koji su označeni stereotipom *NotMapped*) konvertovati u kolone odgovarajućih tipova sa specificiranim ograničenjima i ključevima na način specificiran pri kreiranju modela podataka. Na ovaj način ostvaruje se konekcija od modela podataka ka bazi podataka.

Dodatni korak u slučaju korištenja *FreeSQLDatabase cloud* baze podataka

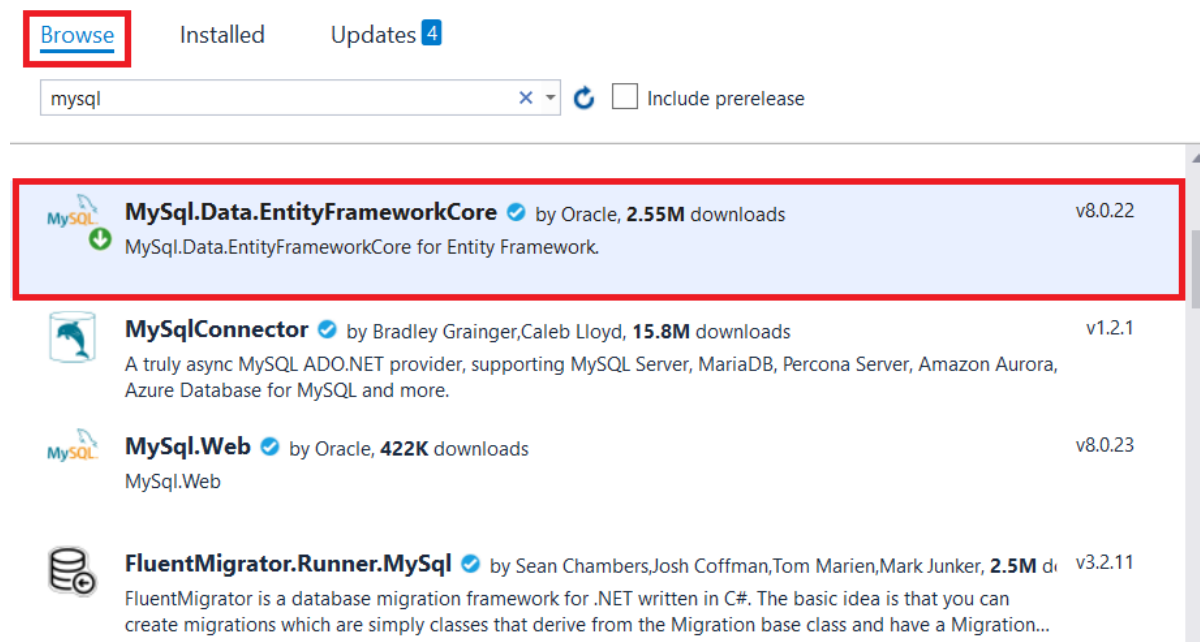
Kako je *FreeSQLDatabase* baza podataka koja po tipu nije *SQL Server* (baza podataka namijenjena za .NET platformu), potrebno je koristiti adapter za *MySQL* tip baze podataka kojoj ova baza podataka (i mnoge druge besplatne *cloud* baze podataka) pripadaju. **Ovaj korak ne treba se vršiti u slučaju korištenju *Microsoft Azure* baze podataka, jer je ona po tipu *SQL Server* baza podataka!**

Prvenstveno je potrebno izvršiti instalaciju adaptera na bazu podataka. Potrebno je izvršiti desni klik na *Dependencies* opciju prikazanu u hijerarhiji programskog rješenja u *Solution Explorer* prozoru u *Visual Studio* okruženju, na način prikazan na Slici 7. Nakon toga potrebno je odabrati opciju **Manage NuGet Packages...** koja omogućava instalaciju eksternih paketa i koja će biti korištena u narednim laboratorijskim vježbama za druge funkcionalnosti.



Slika 7. Pristup upravljanju NuGet paketa

Nakon toga u centralnom prozoru prikazuje se upravljač NuGet paketima. Potrebno je odabrati opciju **Browse**, a zatim se pretragom za pojam *mysql* treba pronaći paket **MySQL.Data.EntityFrameworkCore**, kao što je prikazano na Slici 8. Potrebno je obratiti pažnju da se ne instalira neki drugi paket sličnog imena. Odabirom opcije **Install** s desne strane započinje instalacija ovog paketa, nakon čega je potrebno pristati na sve tražene opcije kako bi se instalacija uspješno završila.



Slika 8. Odabir paketa adaptera za MySQL baze podataka

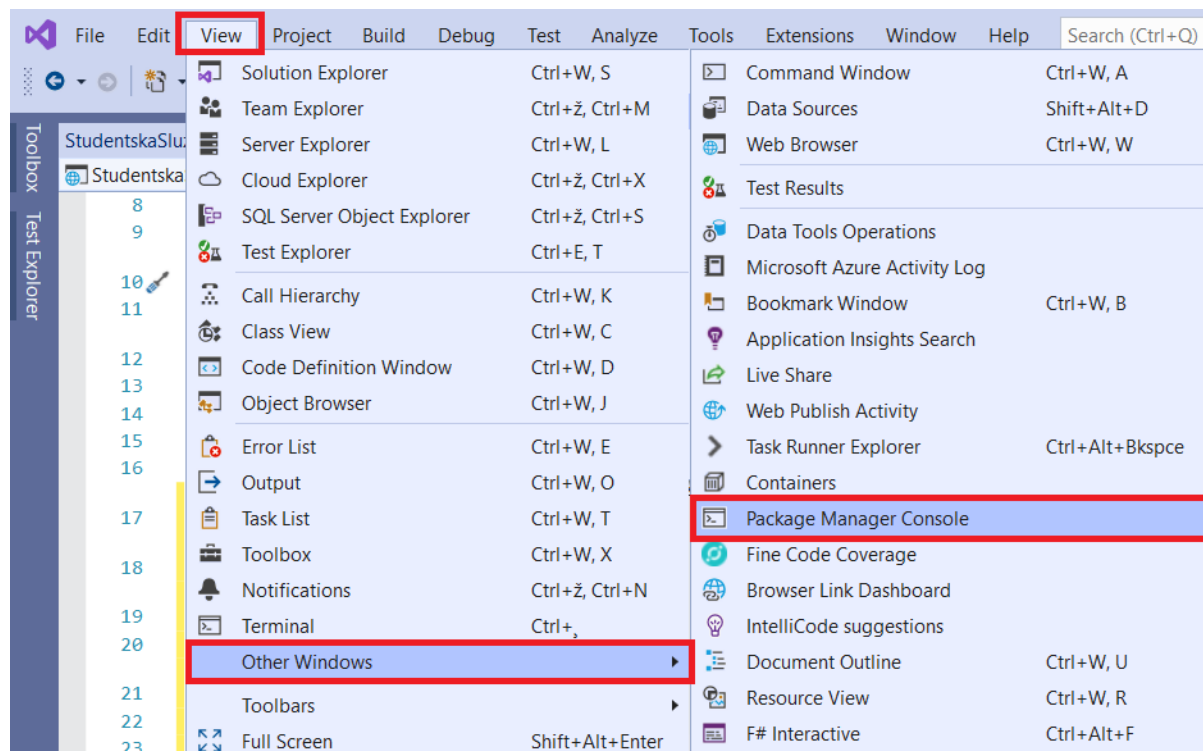
Nakon ovoga potrebno je u **Startup.cs** file-u izvršiti malu modifikaciju dodane registracije servisa za kontekst podataka u metodi **ConfigureServices**. Pri registraciji ovog servisa, poziva se metoda *UseSqlServer* za specifikaciju opcija. Potrebno je promijeniti to u metodu **UseMySQL**, jer se u suprotnom neće moći izvršiti konekcija na bazu. Prethodno instalirani paket neophodan je kako bi se mogla koristiti ova funkcija. Na ovaj način dodatni korak prilagođavanja za *FreeSQLDatabase cloud* bazu podataka je završen² i može se nastaviti sa postupkom migracije na bazu podataka.

```
// This method gets called by the runtime. Use this method to add services
to the container.
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();

    services.AddDbContext<StudentskaSluzbaContext>(options =>
options.UseMySQL(Configuration.GetConnectionString("StudentskaSluzbaContext")));
}
```

² Za direktnu manipulaciju nad ovom bazom podataka potrebno je koristiti *PhpMyAdmin*, prema instrukcijama iz emaila sa korisničkim podacima. Pri unosu korisničkih podataka, u polje **server** potrebno je unijeti **host port** (web-adresu servera, razmak, broj porta), kao i korisničko ime i korisničku šifru da bi se dobio pristup interfejsu za rad s bazom podataka.

Kako bi se izvršila migracija na bazu podataka, potrebno je koristiti **Package Manager Console**, kojoj se može pristupiti kroz *Visual Studio* okruženje, pristupom glavnom meniju i opcijama *View* → *Other Windows* → *Package Manager Console* (prikazano na Slici 9). Nakon toga u donjem dijelu ekrana (na mjestu gdje se prikazuju informacije o greškama) omogućava se unos komandi u konzolu. Ova konzola iskoristiti će se kako bi se unijele komande za migraciju na bazu podataka.



Slika 9. Pristup Package Manager konzoli

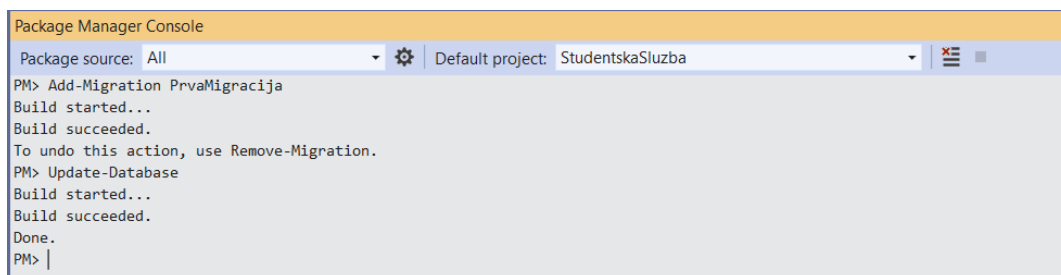
Za vršenje migracije, potrebno je izvršiti dvije komande u konzoli (primjer prikaza za uspješno izvršenu migraciju prikazan je na Slici 10):

1. Dodavanje nove migracije, čime će se generisati SQL kod za kreiranje tabela i njihovih kolona u folderu **Migrations**. U ovom koraku provjerava se ispravnost konekcijskog stringa i mogućnost konekcije na bazu podataka, a ukoliko se migracija ne bude mogla izvršiti u sljedećem koraku, sadržaj ovog foldera može imati važnu ulogu u razumijevanju razloga za to. Komanda za dodavanje nove migracije prikazana je u isječku koda ispod, pri čemu je **<Ime>** potrebno zamijeniti sa proizvoljnim imenom migracije.

```
Add-Migration <Ime>
```

2. Izvršavanje migracije na bazu podataka, čime će se generisani SQL kod izvršiti u bazi podataka. Greške u ovom koraku najčešće nastaju zbog neispravno definisanih stereotipa, nepostojanja primarnih ključeva i drugih problema sa automatski generisanim SQL kodom. Komanda za izvršavanje migracije prikazana je u isječku koda ispod.

Update-Database

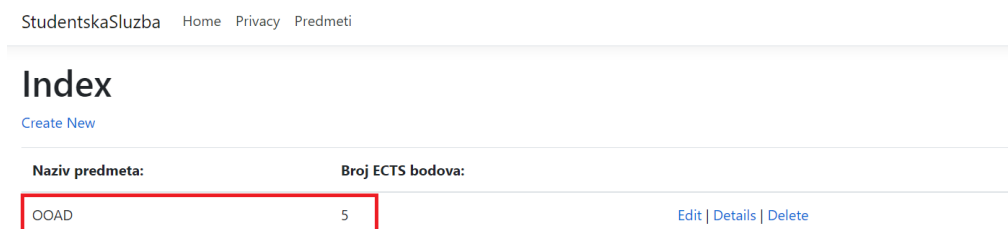


```
Package Manager Console
Package source: All | Default project: StudentskaSluzba
PM> Add-Migration PrvaMigracija
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
PM> Update-Database
Build started...
Build succeeded.
Done.
PM> |
```

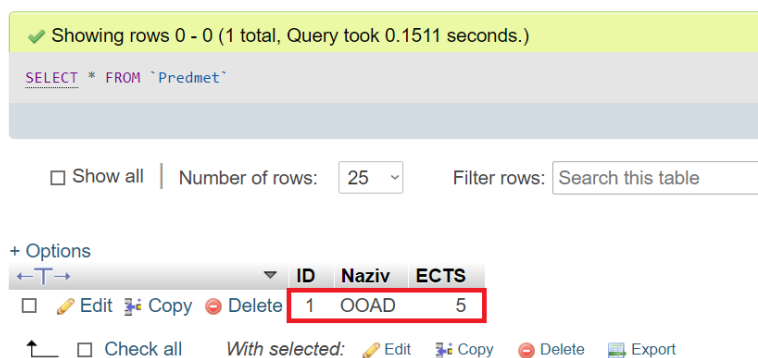
Slika 10. Prikaz izlaza u konzoli tokom vršenja prve migracije

Na ovaj način završen je postupak pripremanja baze podataka za korištenje u prethodno kreiranoj ASP.NET aplikaciji, jer je model klasa podataka preslikan u bazu podataka koja sada posjeduje sve željene tabele (*Predmet*, *Student* i *UpisNaPredmet*). Kako je prethodno automatski generisani kontroler modificiran tako da koristi statičku listu bez perzistencije podataka, taj kontroler (i pripadajući folder pogleda) potrebno je izbrisati i nanovo automatski generisati koristeći *scaffold*. Nakon automatskog generisanja, nije potrebno vršiti nikakve izmjene nad automatski generisanim kodom, već je odmah moguće pokrenuti aplikaciju i početi sa njenim korištenjem.

Pri prvom pokretanju, u listi predmeta ne nalazi se nijedan predmet. Nakon dodavanja novog predmeta, on se prikazuje u listi predmeta. Nakon što aplikacija završi s radom, bez perzistencije podataka uneseni predmet nestaje i pri ponovnom pokretanju ne nalazi se u listi predmeta. No, uz korištenje prethodno opisanih principa perzistencije podataka, nakon ponovnog pokretanja predmet se i dalje nalazi u listi podataka (prikazano na Slici 11), kao i u pripadajućoj tabeli baze podataka (prikazano na Slici 12). Na ovaj način proces omogućavanja perzistencije podataka je uspješno završen.



Slika 11. Prikaz dodanog predmeta nakon ponovnog pokretanja aplikacije



Slika 12. Prikaz predmeta koji se nalazi u bazi podataka

4. Zadaci za samostalni rad

Programski kod aplikacije koja je kreirana u vježbi dostupan je u repozitoriju na sljedećem linku: <https://github.com/ehlymana/OOADVjezbe>, na *branchu* **lv8**.

1. Napraviti *cloud* bazu podataka po želji. Preporučeno je koristiti *FreeSQLDatabase* ili *Microsoft Azure* zbog mogućnosti nastanka problema za druge platforme koje neke od baza podataka koriste.
2. Postojeći konekcijski string u aplikaciji zamijeniti sa vlastitim. Izvršiti komandu *Add-Migration* u svrhu provjere da li je konekcijski string ispravno definisan.
3. Izvršiti migraciju na bazu podataka koristeći *Update-Database* komandu. Ulogovati se na bazu podataka i provjeriti da li se u bazi podataka zaista nalaze tabele *Predmet*, *Student* i *UpisNaPredmet*.
4. Obrisati i ponovo generisati kontrolere i poglede. Pokrenuti aplikaciju i provjeriti da li zaista postoji perzistencija podataka. Automatski generisati preostala dva kontrolera i provjeriti da li aplikacija radi bez ikakvih problema.