Razvoj Web Aplikacija

Predavanje 5

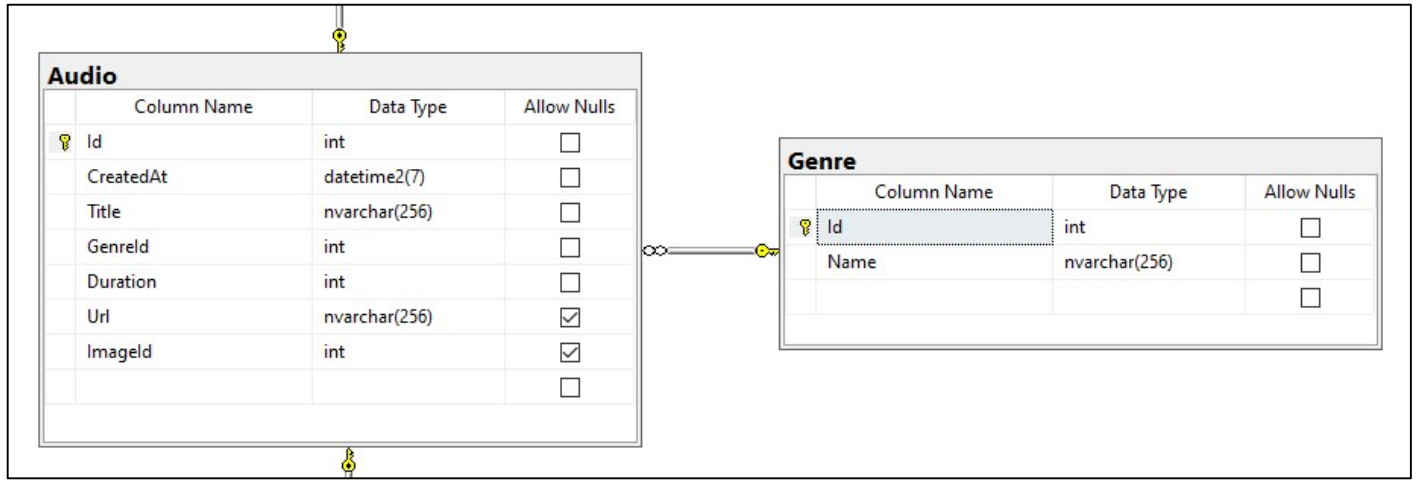
Ponavljanje - pristup bazi podataka (Entity Framework)

- Rad u "odspojenom" (disconnected) načinu
- Entitet i database context
- Klasa entiteta je POCO ➔ Plain Old Class Object
- "Database first" nasuprot "code first" način rada
- Što je troslojna arhitektura?
- Poslovni sloj (business layer, BL) i sloj pristupa podacima (data access layer, DAL)
- Mapiranje između BL-a i DAL-a

Web API i relacije u bazi podataka

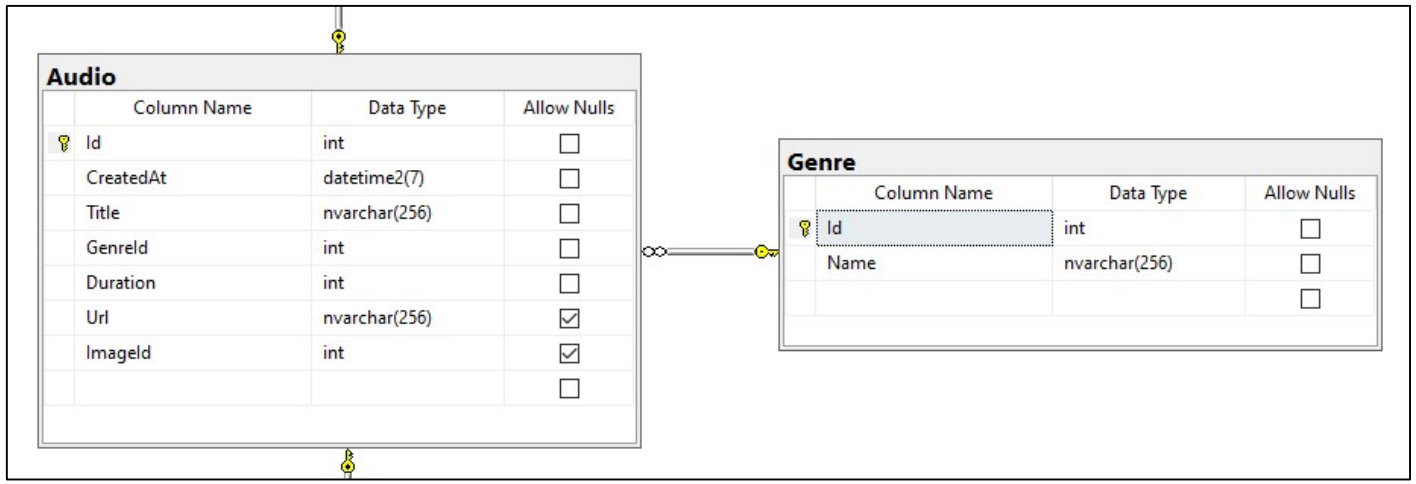
Vrste relacija u bazi podataka

- 1-prema-više ("1-to-many")
- Najčešća vrsta odnosa između entiteta
- Bilo koji žanr (**Genre**) može biti povezan s više pjesama (**Audio**)
- U EF to je predstavljeno pomoću navigacijskog svojstva (Navigation Property) - *virtual*
  - **Audio** ima virtualnog člana tipa **Genre**
  - **Genre** ima virtualnog člana tipa **ICollection<Audio>**
- "Virtual" omogućava "lazy loading" (*lijeno učitavanje?!*)



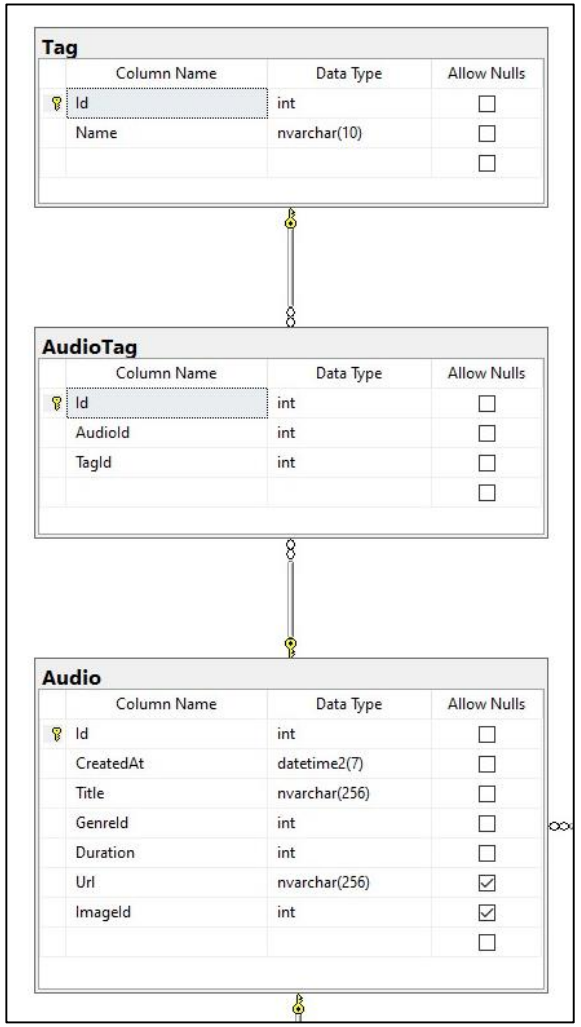
Vrste relacija u bazi podataka

- 1-prema-više: DEMO



Vrste relacija u bazi podataka

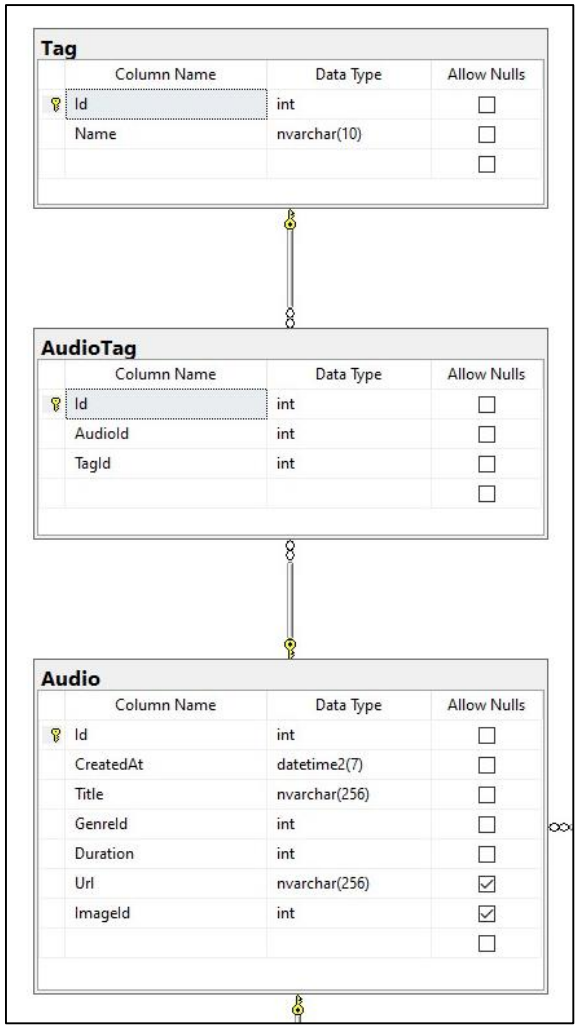
- Više-prema-više ("many-to-many")
- Koristi *asocijativnu tablicu* ili *tablicu za premošćivanje (bridge table)*
- Bilo koja oznaka (**Tag**) može biti povezana s više entiteta pjesme (**Audio**), a bilo koja pjesma može biti povezana s više entiteta oznake
- Također se implementira putem navigacijskog svojstva (virtual, *lazy loading*)
  - **AudioTag** ima članove tipa **Audio** i **Tag**
  - **Audio** ima virtualni član tipa **ICollection<AudioTag>**
  - **Tag** ima virtualni član tipa **ICollection<AudioTag>**
  - *Napomena: najnoviji EF scafolding NE pretpostavlja da će kolekcija AudioTag biti mijenjana na mjestu povezanih entiteta, pa morate sami dodati "setter"*




```
public partial class Audio
{
    public int Id { get; set; }
    …
    public virtual ICollection<AudioTag> AudioTags { get; } = new List<AudioTag>();
}
```

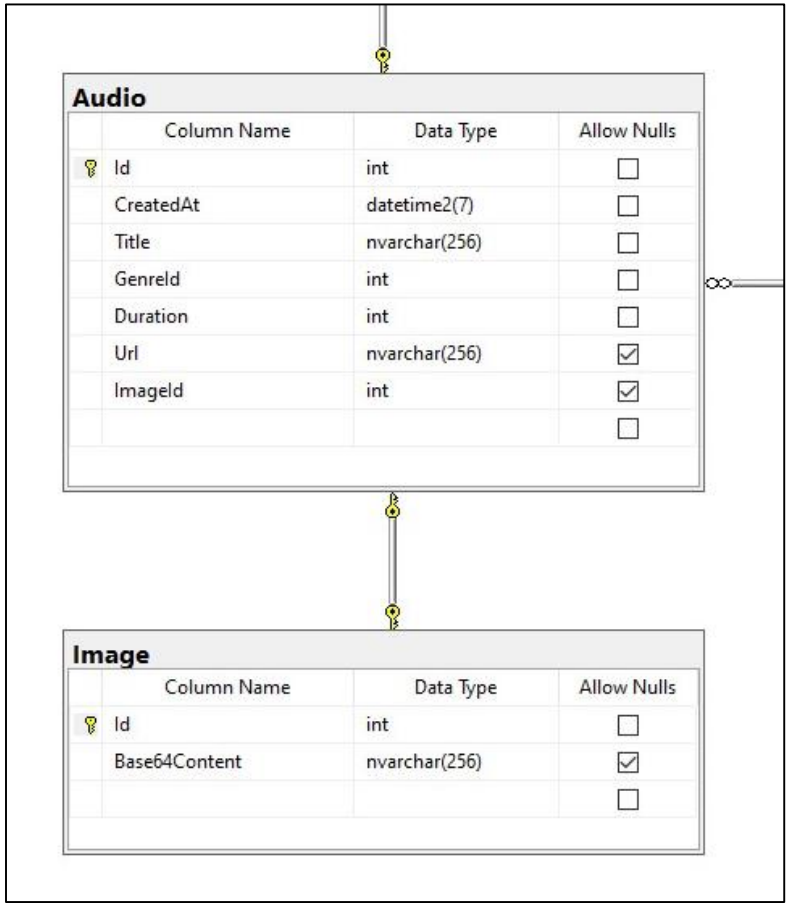
Vrste relacija u bazi podataka

- Više-prema-više DEMO



Vrste relacija u bazi podataka

- 1-prema-1 (1-to-1) relacija
- Uspostavljeno između primarnih ključeva
- Jedan entitet povezan je s jednim (ili nula) entiteta



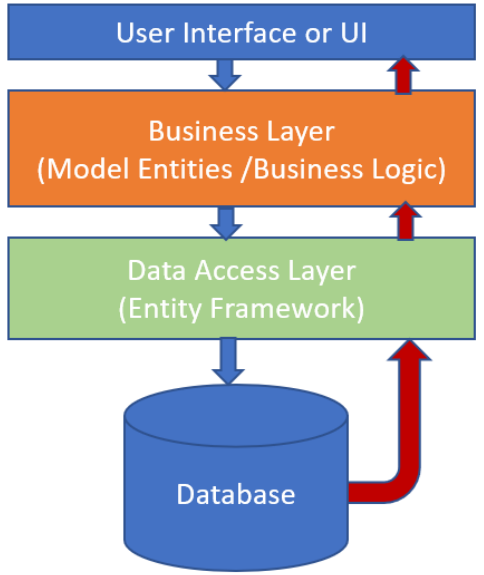
Vrste relacija u bazi podataka

- Kontekst baze podataka (database context) opisuje odnos između tablica
  - Pogledati OnModelCreating()
- Obično proces pripreme koda (scaffolder) za svaku relaciju između entiteta opisuje smjer prema ciljnoj tablici (onoj s PK, relacija "prema jedan") i natrag ako je potrebno (relacija "prema više")
- Relacija „prema jedan”: someEntity.HasOne(d => d.TargetEntity)
- Relacija „prema više”: someEntity.WithMany(d => d.TargetEntity)
- Također, potrebno je navesti strani ključ i ponašanje kod brisanja

Web API, baza podataka i troslojna arhitektura

Troslojna arhitektura

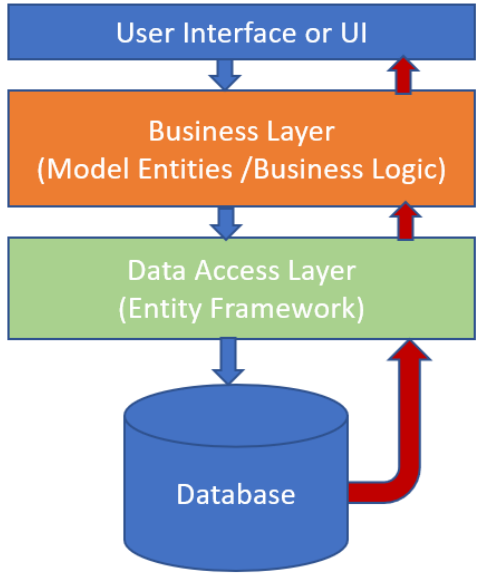
Postavljanje troslojne arhitekture u Web API + EF - kako se to radi?



- Za troslojnu arhitekturu moramo podijeliti naš Web API projekt na 2 dijela
  - DAL
  - BL
- Obično to činimo stvaranjem novog projekta knjižnice (library project) koji predstavlja DAL, a Web API projekt tada predstavlja BL
  - Web API treba referencu na DAL projekt
  - Ne zaboravite instalirati EF podršku u DAL projekt

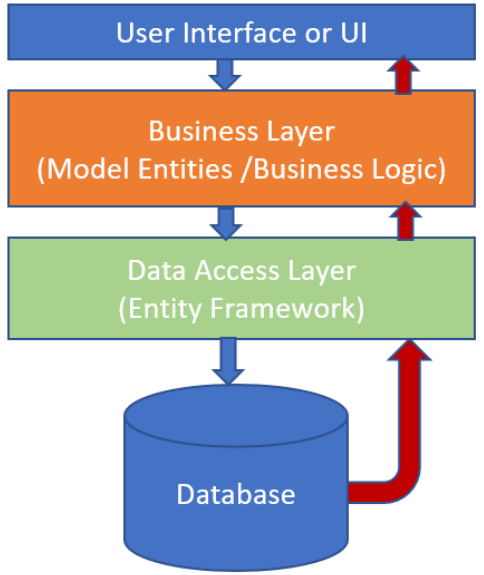
Troslojna arhitektura

- Novi DAL projekt tada je odgovoran za DAL modele, pa bi ti modeli trebali biti unutar tog projekta
  - Sasvim je u redu ponovno pokrenuti pripremu koda za bazu podataka (*scaffolding, reverse engineering*), ovaj put u DAL projektu
  - Ne zaboravite da DAL projekt sadrži db kontekst
- U tom konceptu projekt Web API koristi DAL biblioteku
  - Registracija DI spremnika nalazi se u projektu Web API
  - Razrješenje (resolving) pojedinog tipa iz DI kontejnera moguće je u bilo kojem projektu
  - Web API projekt pokreće DI kontejner ➔ da bi stvorio instancu db konteksta, mora pronaći tip u projektu DAL biblioteke
  - Konfiguracija je još uvijek u BL projektu, ali joj se može pristupiti i u DAL projektu



Troslojna arhitektura

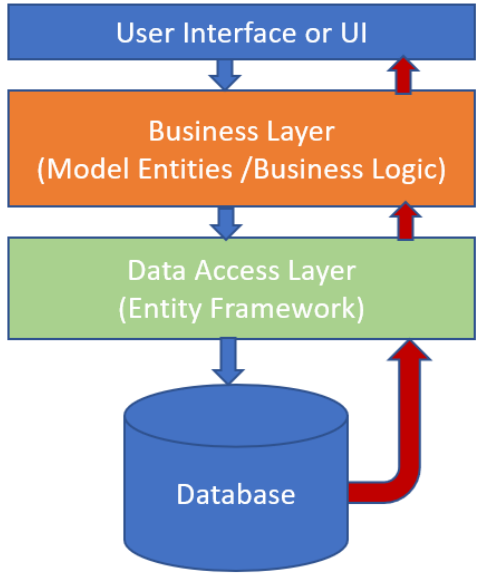
Implementacija troslojne arhitekture u Web API + EF - kako se to radi?



- Kontroler bi trebao biti **tanak** – trebali bismo premjestiti što više koda u specifične enkapsulacije (klase) koje ispunjavaju određene zadatke
- **Klasa za mapiranje** – mapira BL ←→ DAL, jednostavnim kopiranjem ili jednostavnom konverzijom/deriviranjem vrijednosti između BL i DAL-a
- **Klasa repozitorija** – zapravo usluga, obavlja složene operacije na DAL-u, obično potrebne za CRUD

Troslojna arhitektura

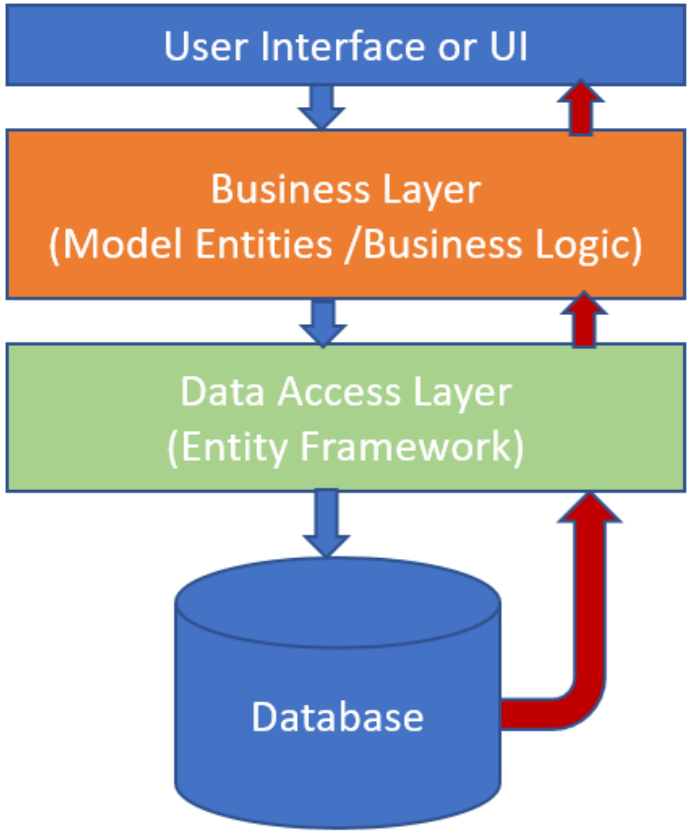
Popis za implementaciju za resurs dostupan u DAL-u



- Imamo li kontroler?
- Imamo li BL model?
- Postoji li klasa repozitorija u DAL-u registrirana u DI, koja je se koristi u BL-u (kontroleru)?
- Imamo li db kontekst u tom repozitoriju?
- Implementirajte jednu akciju HTTP metode koja koristi metodu repozitorija - ovdje trebate stvoriti akciju, repozitorijsku metodu i maper
- Implementirajte ostale akcije

Troslojna arhitektura

- DEMO implementacije



Hvala na pažnji!