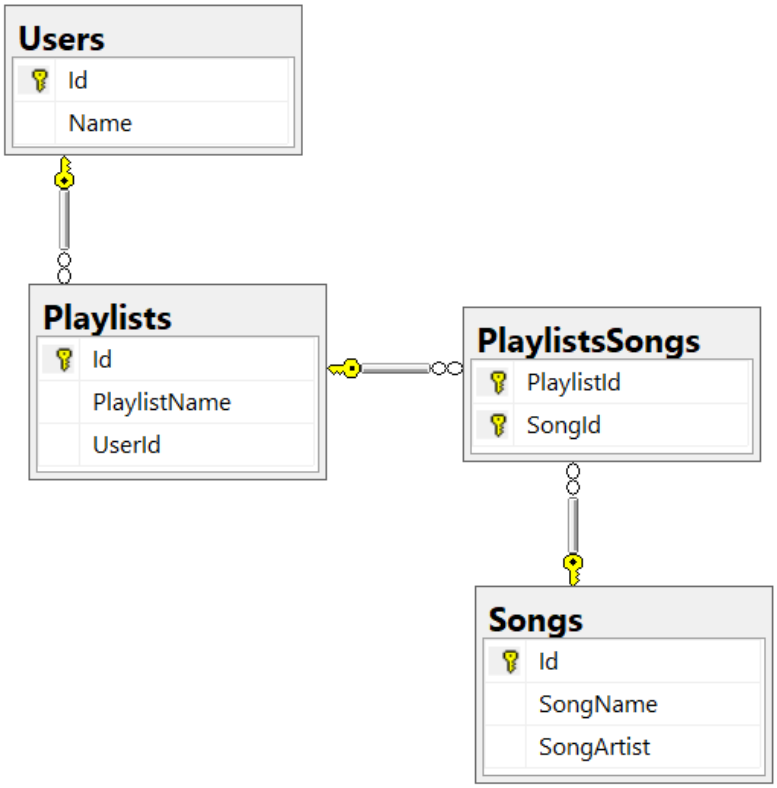
Mini-Exam: Entity Relations and LINQ

You can check your solutions here: https://judge.softuni.bg/Contests/3202/Additional-Exercises.

You will be given a **skeleton** for your **tasks** solutions. Do not change the skeleton.

14. Music Provider

Your task is to create a database for the **Music Provider System**, using the **EF Core Code First** approach. It should look like this:



Constraints

Your **folders**:

- **`MusicProvider.Data`** – for your DbContext and Configuration
- **`MusicProvider.Data.Models`** – for your models

Your **models** should be:

- **`User`**:
  - Id
  - Name – up to 100 characters
- **`Playlist`**:
  - Id
  - Name – up to 100 characters
  - UserId
- **`Song`**:
  - Id
  - SongName – up to 120 characters

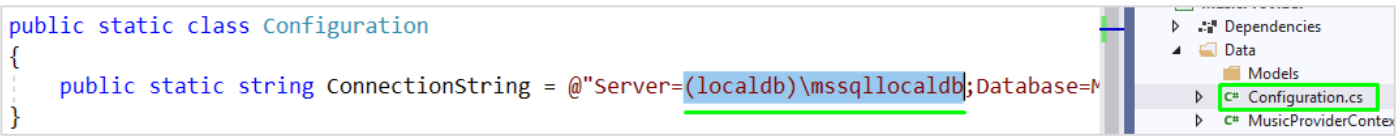
    - SongArtist – up to 150 characters
- **`PlaylistSong`** mapping class between **Playlist** and **Song**

Table relations:

- **One User** can have **many Playlist**
- **One Playlist** can have **one User**
- **One Playlist** can have **many Song**
- **One Song** can have **many Playlist**

Hints:

If you are using a different SQL Server than localdb, don't forget to change your server connection.



15. Characters Information

Use the project **Diablo** in the skeleton.

You need to write your solution in the method `CharactersInformation(DiabloContext context, int luck)` in the **StartUp** class that receives a **luck value**. Export all the **characters** which have **luck more than** the received. For each **Character**, get the **Name, the count of Games** and the **Name** of each **Game. Sort** the **Characters** by **count of Games.**

Print the result in the following format:

`"Name:{Characters Name}"`

`"Games: {Count Games}"`

and for each game:

`"Game name: {Game Name}"`

Example

Output(luck = 17)

`Name: Necromancer`
`Games: 26`
`Game name: Gerbera Ruby Red`
`Game name: Chicago`
`Game name: Houston`
`Game name: Chicago`
`Game name: Copenhagen`
`Game name: Love in a mist`
`Game name: London`
…

16. Types Information

You need to write your solution in the method `GameTypesInformation(DiabloContext context, int idGameType)` in the **StartUp** class that receives a **GameType Id**. Export all the **GameTypes** which are with the received Id. For each **GameType**, get the **Name** and the **Name** of each **Game. Sort** them by **Game Type Name.**

Print the result in the following format:

"Name: {GameType Name}"

and for each game:

"Game name: {Game Name}"

Example

Output(GameType Id = 5)
Name: Funny Game name: Acid green Game name: Broadway Game name: Ancalagon Game name: Acaeria Game name: Daffodil Game name: Freesia …

17. User Games Information

You need to write your solution in the method `UserGamesInformation(DiabloContext context, int userId)` in the **StartUp** class that receives a **User Id**. Export all the **Games** of the user with the received Id. For each user's **Game**, get the Game **Name, the Character Name** and the **Names** of the all **Items used** in the game**. Sort** them by **Items Count** and **by** Game **Name.**

Print the result in the following format:

"Game:{Game Name}");

" Character Name: {Character Name}");

"   Items:"

and for each **Item**:

"   -{g.ItemName}"

Example

Output(User Id = 10)

Game:Vancouver
  Character Name: Necromancer
  Items:
  -Death Watch Mantle
  -Fragment of Destiny
Game:Pincushion flower annual
  Character Name: Demon Hunter
  Items:
  -Corrupted Ashbringer
  -Fire Brand
  -Invigorating Gemstone
  -Mutilation Guard
  -Puzzle Ring

…

Hints:

If you are using a different SQL Server than localdb, don't forget to change your server connection.

```
public static class Configuration
{
    public static string ConnectionString = @"Server=(localdb)\mssqllocaldb;Databa
}
```