OOP/C# - Opdrachten (Razor Pages)

Opdrachten bij het vak OOP/C# voor 2e jaars MBO Software Development

Auteur:

René Melenhorst

rgj.melenhorst@noorderpoort.nl

Inhoud

1 Razor Pages - Web app with ASP.NET (Introduction) 3

GitHub repo en MS Visual Studio 3

1.1 Get started 4

1.2 Add a model 4

1.3 Scaffolding 6

1.4 Work with a database 7

1.5 Update the pages 8

1.6 Add search functionality 9

1.7 Add a new field 10

1.8 Add validation 11

2 Razor Pages - Games Database web app with ASP.NET 13

GitHub repo en MS Visual Studio 13

2.1 Backend 13

2.2 Frontend 14

2.3 Implementatie 15

3 Razor Pages - Using Entity Framework in ASP.NET web app 16

GitHub repo en MS Visual Studio 16

3.1 Get started 16

3.2 Create, Read, Update Delete 17

3.3 TBD 18

1 Razor Pages - Web app with ASP.NET (Introduction)

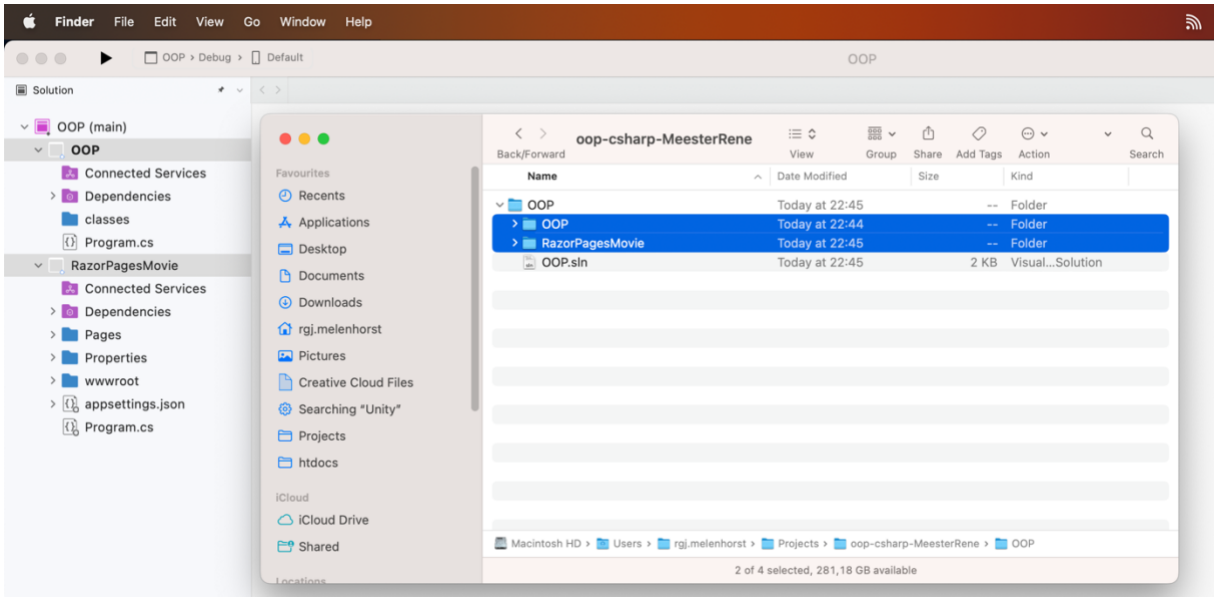
We gaan voor onze *Games Database Applicatie* een web based frontend bouwen en wel in *C#/ASP.NET* waarbij we gebruik gaan maken van de *Razor* syntax.

Dat doen we in twee stappen. We beginnen eerst met een tutorial om kennis te maken met Razor en ASP.NET. Daarna gaan we de frontend maken voor de games applicatie die je in de eerste hoofdstukken hebt gemaakt, zie hiervoor het volgende hoofdstuk. Hieronder een link naar de online tutorial waar we mee starten:
https://learn.microsoft.com/en-us/aspnet/core/tutorials/razor-pages

GitHub repo en MS Visual Studio

Je blijft werken in de repo waarin je de *Games Database Applicatie* hebt gemaakt, maar we maken voor de overzichtelijkheid wel **in de bestaande solution** een nieuw C# project aan op de manier zoals staat beschreven in de eerste stap van de online tutorial (zie verderop in 1.1).

Aan het einde van stap 1.1 zou de structuur in Visual Studio er als volgt moeten uitzien, met 2 projecten in 1 solution:

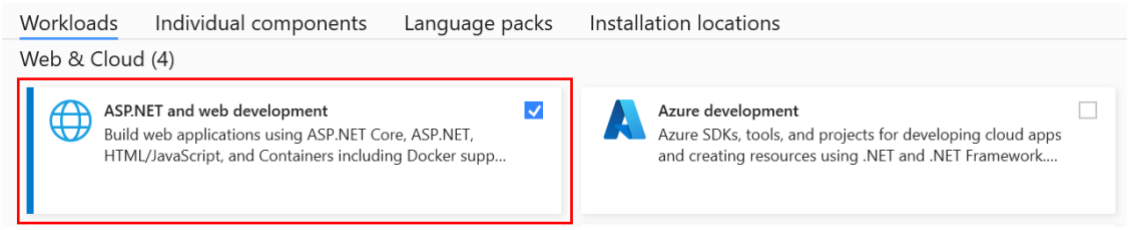


1.1 Get started

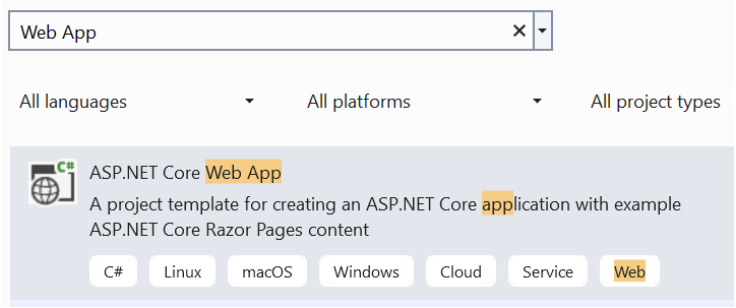
Lees het eerste deel van deze tutorial genaamd *Get started with Razor Pages in ASP.NET*:

https://learn.microsoft.com/en-us/aspnet/core/tutorials/razor-pages/razor-pages-start

Vergeet niet om zoals beschreven het component *ASP.NET and web development* te installeren door de installer van *Microsoft Visual Studio* opnieuw uit te voeren.



En eenmaal geïnstalleerd is het type project wat je in de bestaande solution moet aanmaken dus een *ASP.NET Core Web App*:



Zorg dat aan het einde van deze stap je de Razor page in je webbrowser op de local host hebt draaien. Indien het geval, *commit* en *push* je de code met als message: **Opdracht 9.1**.

1.2 Add a model

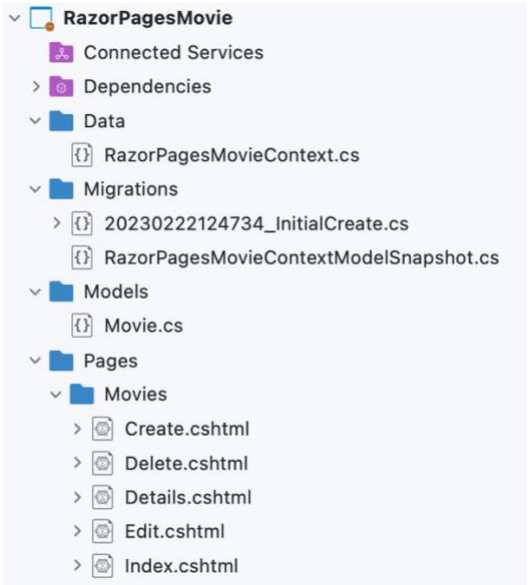
Lees het volgende deel van deze tutorial genaamd *Add a model to a Razor Pages app*:

https://learn.microsoft.com/en-us/aspnet/core/tutorials/razor-pages/model

Aan het einde van deze stap heb je:

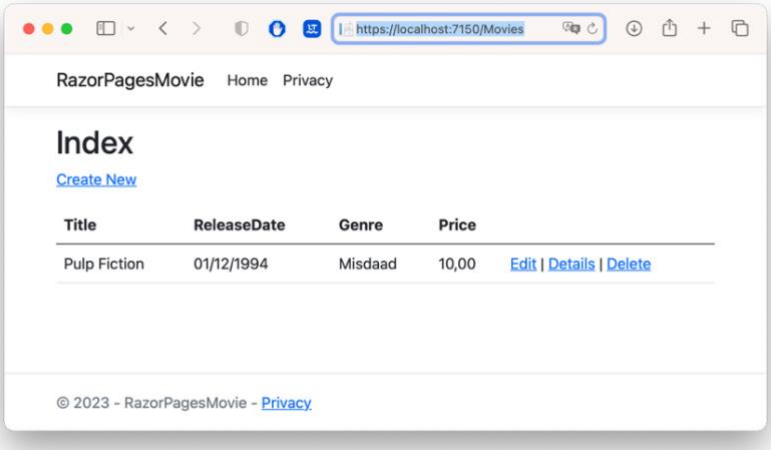
1. Zoals beschreven de class **Movie** toe aan de nieuwe **Models** folder toegevoegd. De properties die je hebt toegevoegd vormen de basis voor de kolommen in de *Movie* database tabel. Controleer of je project nog goed bouwt.
2. Zoals beschreven een zogenaamd *Scaffolded Item* toegevoegd aan de nieuwe **Pages/Movies** folder. Op basis van de *Movie* class en zijn properties (oftewel de database kolommen) worden CRUD pagina's gegenereerd in de *Movies* folder. Controleer de inhoud van je *Movies* folder en of je project steeds nog goed bouwt.
3. De beschreven *initial database schema creation* stappen uitgevoerd die o.a. resulteren in een nieuwe **Migrations** folder in je project. Op basis van de *Movie* class en zijn properties wordt een migration file gegenereerd in de *Migrations* folder. De migration file wordt gebruikt om de daadwerkelijke database tabel en bijbehorende kolommen te maken. Controleer de inhoud van je *Migrations* folder en of je project nog steeds goed bouwt.

Na deze stappen zou je project er als volgt moeten uitzien, met o.a. de *Movie* class, de gegeneerde CRUD pagina's in de *Movies* folder, en de gegeneerde *Migrations* folder met de eerste migration file:



Voer de in de tutorial beschreven test stappen uit (sectie 'Test the app'). Je zou de Index page in je webbrowser moeten kunnen zien, de data van een nieuwe *Movie* moeten kunnen invoeren (via de *Create New* link: de [C]reate van CRUD), de record moeten kunnen aanpassen (*Edit*: de [U]pdate van CRUD), de details moeten kunnen bekijken (*Details*: de [R]ead van CRUD) en het record moeten kunnen verwijderen (*Delete*: de [D]elete van CRUD).

In de browser zou je web applicatie er als volgt moeten uitzien. Zorg dat je na het testen van de CRUD acties tenminste 1 movie in de lijst achterlaat:



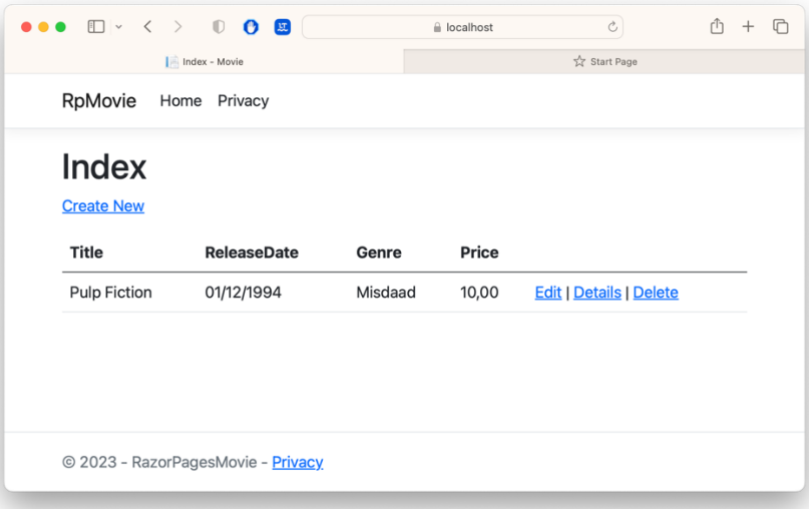
Als je al deze test stappen succesvol hebt uitgevoerd, *commit* en *push* je de code met als message: **Opdracht 9.2**.

1.3 Scaffolding

Lees het volgende deel van deze tutorial genaamd *Scaffolded Razor Pages in ASP.NET*:

https://learn.microsoft.com/en-us/aspnet/core/tutorials/razor-pages/page

Aan het einde van deze stap heb je zoals beschreven de code van de Pages/Shared/**_Layout** page aangepast en je aanpassingen getest in de browser. Je web applicatie zou er als volgt moeten uitzien (let ook op de titel van het tabblad):



Als dit het geval is, *commit* en *push* je de code met als message: **Opdracht 9.3**.

1.4 Work with a database

Lees het volgende deel van deze tutorial genaamd *Work with a database in ASP.NET*:

https://learn.microsoft.com/en-us/aspnet/core/tutorials/razor-pages/sql

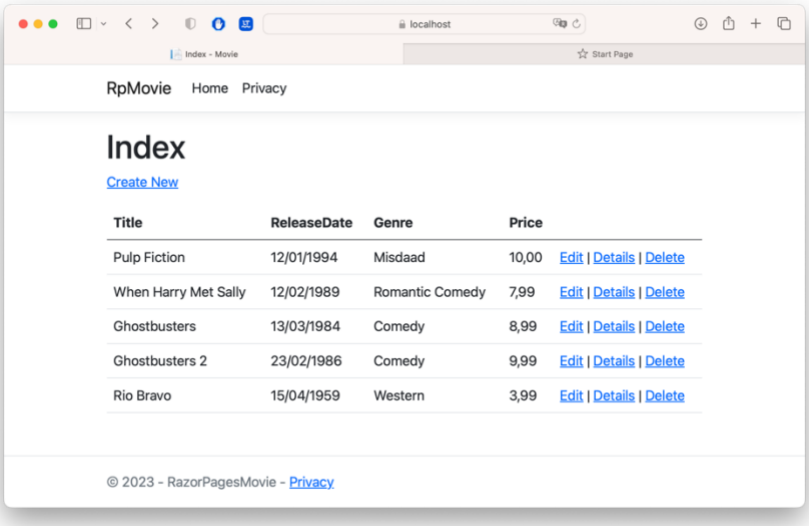
Aan het einde van deze stap heb je:

1. In MS Visual Studio (voor Windows [1]) de *SQL Server Object Explorer* geopend en de data in je lokale database bekeken die je in de vorige stappen hebt aangemaakt. Bekijk niet alleen de data maar ook het design van de *Movie* tabel en vergelijk deze met de inhoud van je *Movie* class in de *Models* folder.
2. Zoals beschreven de zogenaamde *seeding* code toegevoegd aan de nieuwe Models/**SeedData** class. Voeg je eigen, in de web applicatie, handmatig toegevoegde movie(s) toe aan de *seeding* code omdat je vóór het *seeden* met een lege database moet beginnen.
3. De **Program** class met de *seed initializer* code ge-update.

[1] Heb je een Apple MacBook volg dan de beschreven stappen om een (externe) SQLite client te installeren, en bekijk daarin de inhoud van je database.

4. De in de tutorial beschreven test stappen uitgevoerd (sectie 'Test the app'). Vergeet niet om zoals aangegeven je eigen toegevoegde movies te deleten uit de database anders zal het *seeden* niet in werking treden.

Je web applicatie zou er als volgt moeten uitzien, met in de lijst de data van de movies die in de originele *seeding* code stond, en in dit geval één zelf (in de seeder code) toegevoegde film.



Als dit het geval is, *commit* en *push* je de code met als message: **Opdracht 9.4**.

1.5 Update the pages

Lees het volgende deel van deze tutorial genaamd *Update the generated pages in ASP.NET*:

https://learn.microsoft.com/en-us/aspnet/core/tutorials/razor-pages/da1

Aan het einde van deze stap heb je:

1. Zoals beschreven de implementatie van de Models**Movie** class aangepast, gerelateerd aan de gebruikte *attributes*.
2. Zoals beschreven de *page directive* van de Pages\Movies**Edit, Details,** en **Delete** Pages aangepast, gerelateerd aan de `{id:int}` route template.

In je web applicatie is de aangepaste column header *Release Date* zichtbaar. Als dit het geval is, en je de andere stappen ook succesvol hebt uitgevoerd, *commit* en *push* je de code met als message: **Opdracht 9.5**.

1.6 Add search functionality

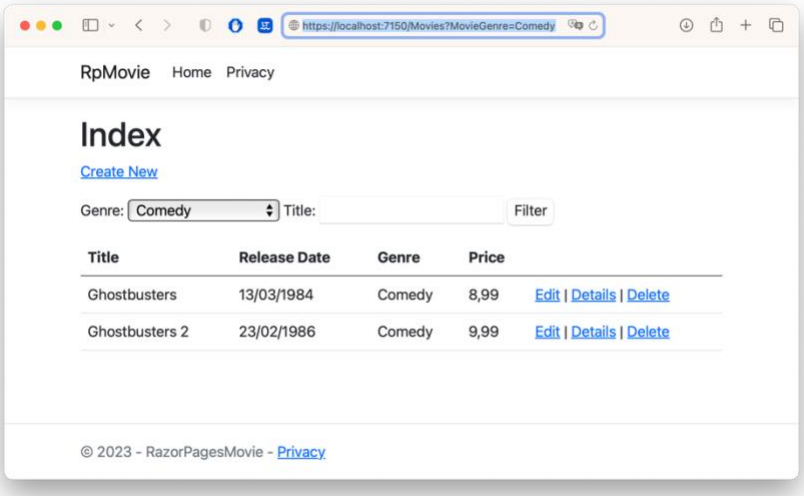
Lees het volgende deel van deze tutorial genaamd *Add search to ASP.NET Razor Pages*:

https://learn.microsoft.com/en-us/aspnet/core/tutorials/razor-pages/search

Aan het einde van deze stap heb je:

1. Zoals beschreven de implementatie van de Pages\Movies**Index** page *code behind* file aangepast, gerelateerd aan de *Search by Title* functionaliteit, en de aanpassing in de browser getest m.b.v. de genoemde *query string*.
2. Zoals beschreven de implementatie van de **Index** HTML page aangepast, gerelateerd aan het search input veld, en de aanpassing in de browser getest m.b.v. dit nieuwe veld.
3. Zoals beschreven de implementatie van de **Index** page *code behind* file aangepast, gerelateerd aan de *Search by Genre* functionaliteit, en de aanpassing in de browser getest met een *query string* net zoals je bij de film titel hebt gedaan.
4. Zoals beschreven de implementatie van de **Index** HTML page aangepast, gerelateerd aan de genre selection box, en de aanpassing in de browser getest m.b.v. dit nieuwe veld.
5. De search functionaliteit ook met combinaties van *Title* en *Genre* getest, en na elke zoekopdracht de *query string* in de browser geanalyseerd.

Je web applicatie zou er als volgt moeten uitzien, met boven de lijst met films de *Genre* selection box en *Title* input veld, en een (met de zoekopdracht overeenkomstige) *query string* in de browser URL.



Als dit het geval is, *commit* en *push* je de code met als message: **Opdracht 9.6**.

1.7 Add a new field

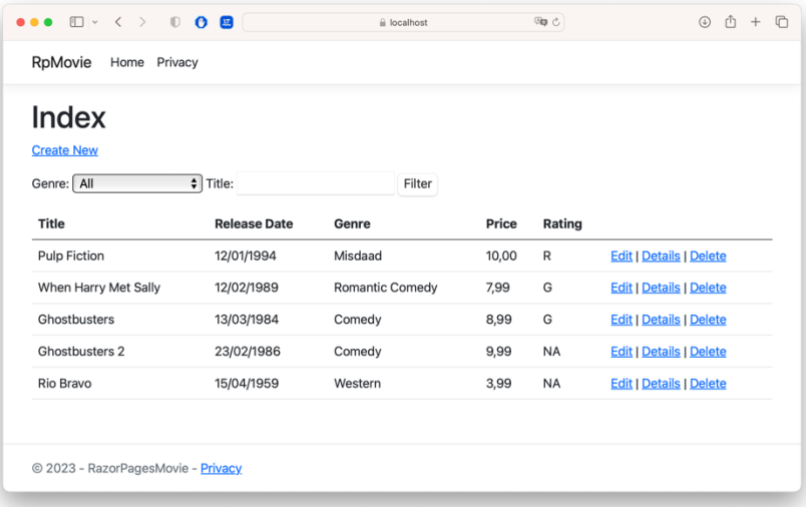
Lees het volgende deel van deze tutorial genaamd *Add a new field to a ASP.NET Razor Page*:

https://learn.microsoft.com/en-us/aspnet/core/tutorials/razor-pages/new-field

Aan het einde van deze stap heb je:

1. Zoals beschreven de implementatie van de Models**Movie** class aangepast, gerelateerd aan de nieuwe **Rating** property.
2. Zoals beschreven de implementatie van de Pages\Movies**Index**, **Create**, **Edit, Details,** en **Delete** HTML Pages aangepast, gerelateerd aan de nieuwe **Rating** field.
3. De *seeding* code in de Models/**SeedData** class ge-update door voor elke Movie de nieuwe Rating property en een bijbehorende waarde toe te voegen.
4. De in de tutorial beschreven migratie stappen uitgevoerd (sectie 'Add a migration for rating'). Vergeet niet om zoals aangegeven alle movies te deleten uit de database anders zal het *seeden* niet in werking treden.

Je web applicatie zou er als volgt moeten uitzien, met de nieuwe Rating kolom inclusief de waardes die je hebt gekozen.



Als dit het geval is, *commit* en *push* je de code met als message: **Opdracht 9.7**.

1.8 Add validation

Lees het laatste deel van deze tutorial genaamd *Add validation to a ASP.NET Razor Page*:

https://learn.microsoft.com/en-us/aspnet/core/tutorials/razor-pages/validation

Aan het einde van deze stap heb je:

1. Zoals beschreven de implementatie van de Models**Movie** class aangepast, gerelateerd aan de *validation attributes*, en de *client-side validation* in de *Create* en *Edit* pages van de web app getest.
2. Deze aanpassing ook met de debugger getest door zoals aangegeven een breakpoint te zetten in de *OnPostAsync()* methods van beide pagina's en heb je geanalyseerd wanneer het breakpoint wel en niet wordt geraakt i.g.v. een submit van de form.
3. Zoals beschreven de aanpassing nogmaals met de debugger getest, maar ditmaal heb je voor een *server-side validation* gezorgd door de JavaScript in je browser tijdelijk uit te zetten. Analyseer de verschillen tussen deze *server-side* en de voorgaande *client-side validatie*.
4. Zoals beschreven een aantal *validation attributes* gecombineerd zodat ze nog maar op 1 code regel staan.

5. De in de tutorial beschreven migratie stappen uitgevoerd (sectie 'Apply migrations') i.v.m. de toegevoegde *validation attributes*, en tevens de **Up()** method in de gegenereerde migratie file geanalyseerd. Als het goed is herken je in de code van deze method de code aanpassingen die je in de Models**Movie** class hebt gedaan.

Als je al deze stappen succesvol hebt uitgevoerd, *commit* en *push* je de code met als message: **Opdracht 9.8**.

2 Razor Pages - Games Database web app with ASP.NET

Nu je kennis hebt gemaakt met Razor en C#/ASP.NET gaan we van onze bestaande *Games Database (console) Applicatie* een web applicatie versie maken. We volgen daarvoor min-of-meer dezelfde stappen als uit het vorige hoofdstuk, echter alle movie data vervangen we door game en publisher data.

GitHub repo en MS Visual Studio

Je blijft werken in de repo waarin je de vorige twee applicaties hebt gemaakt, en we maken voor de overzichtelijkheid wederom **in de bestaande solution** een nieuw C# project aan op de manier zoals staat beschreven in de eerste stap van de online tutorial uit het vorige hoofdstuk (zie 1.1).

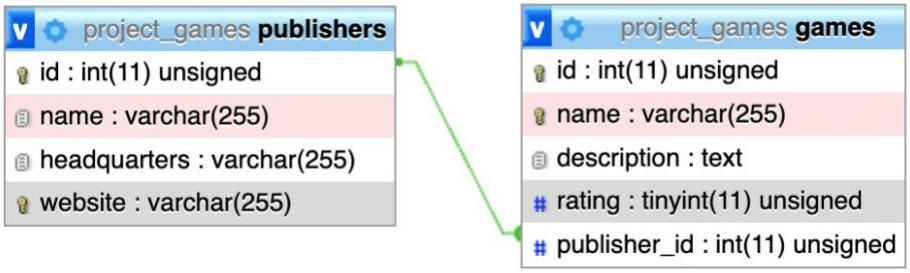
Kies ditmaal de naam **RazorPagesGame** als project naam. Aan het einde van stap 1.1 zou de structuur in Visual Studio er als volgt moeten uitzien, met 3 projecten in 1 solution:

- OOP
- RazorPagesMovie
- RazorPagesGame

2.1 Backend

Voor de **backend** zorg je dat de data die in de database komt te staan voldoet aan het onderstaande *Entity Relationship Model* die we al eerder hebben gehanteerd. Groot verschil met de vorige keer is dat we ditmaal de tabellenstructuur in de code definiëren (in de *Model*) en we d.m.v. een *migratie* de database tabellen laten genereren (i.p.v. handmatig aanmaken).

Hetzelfde geldt voor de inhoud van de database tabellen: definitie in de code en het vullen van de tabellen d.m.v. een migratie (het *seeden*).



2.2 Frontend

Voor de **frontend** moet de web applicatie voldoen aan de volgende requirements:

1. Er zijn twee *index pages* te selecteren via het hoofdmenu, waarvan die voor de Games de eerste is. Deze web pagina geeft een overzicht van alle Games.
2. De tweede index page is die voor de Publishers. Deze web pagina geeft een overzicht van alle Publishers.
3. Voor elke Game en Publisher zijn de bekende CRUD acties **C**reate, **U**pdate en **D**elete acties beschikbaar. Deze zijn te selecteren via links in de *index page*.
4. Voor elke Game en Publisher is de CRUD actie **R**ead beschikbaar, d.w.z. het inzien van de details d.m.v. een *details page*. Deze is te selecteren via een link in de *index page*.
5. Op de index pagina van de Games, en tevens op de detail pagina van een enkele Game, is een link aanwezig naar de details pagina van de desbetreffende Publisher van de Game.

   Dit is dus in feite de implementatie van de relatie uit het bovenstaande schema: het groene lijntje tussen de twee tabelvelden, en wel de 1-op-1 relatie van de Games tabel richting de Publisher tabel.
6. Op de detail pagina van een enkele Publisher, wordt behalve de details ook een lijst met games getoond die door deze publisher zijn uitgebracht.

   Dit is dus in feite de implementatie van de relatie uit het bovenstaande schema: het groene lijntje tussen de twee tabelvelden, en wel de 1-op-veel relatie van de Publisher tabel richting de Games tabel.

   Voor elke Game in de lijst is een link aanwezig die leidt naar de details pagina van de Game.

2.3 Implementatie

Implementeer de hierboven beschreven backend en frontend requirements in een nieuwe web applicatie. Geen detailstappen ditmaal, maak gebruik van je opgedane kennis uit de tutorial van het vorige hoofdstuk en raadpleeg Google daar waar je iets nieuws implementeert.

Vergeet niet om tussendoor na elke succesvolle deelstap je code te *committen* en *pushen* naar GitHub, en gebruik als commit message: **Opdracht 10.1**, **Opdracht 10.2**, etc.

3 Razor Pages - Using Entity Framework in ASP.NET web app

We gaan ons wat verder verdiepen in C#/ASP.NET en leggen wat meer nadruk op het Entity Framework (EF), een zogenaamd Object Relational Mapping (ORM) tool. Dit is een manier om een database met gegevens te modelleren in code, in dit geval C#, en vervolgens de code de database tabellen te laten genereren én eventueel te vullen met gegevens. Misschien herken je deze methodiek wel uit de Laravel cursus waarbij je de database-structuur ook modelleert in code, maar dan m.b.v. de PHP programmeertaal en de *Eloquent* ORM.

Hieronder een link naar de online tutorial:
https://learn.microsoft.com/en-us/aspnet/core/data/ef-rp/intro

GitHub repo en MS Visual Studio

Je blijft werken in de repo waarin je de vorige applicaties hebt gemaakt, en we maken voor de overzichtelijkheid wederom **in de bestaande solution** een nieuw C# project aan.

Zoals staat beschreven in de tutorial gebruiken we ditmaal de naam **ContosoUniversity** als project naam. Aan het einde van stap 3.1 zou de structuur in Visual Studio er als volgt moeten uitzien, met 4 projecten in 1 solution:

- OOP
- RazorPagesMovie
- RazorPagesGame
- ContosoUniversity

Je mag, in tegenstelling tot wat in de tutorial staat, ook het project de naam **RazorPagesContosoUniversity** geven, een wat lange naam maar zodoende staan wel al je RazorPages projecten bij elkaar in je solution (of gebruik het kortere *RPContosoUniversity*).

3.1 Get started

Volg het eerste deel van deze tutorial genaamd *Get Started*:
https://learn.microsoft.com/en-us/aspnet/core/data/ef-rp/intro

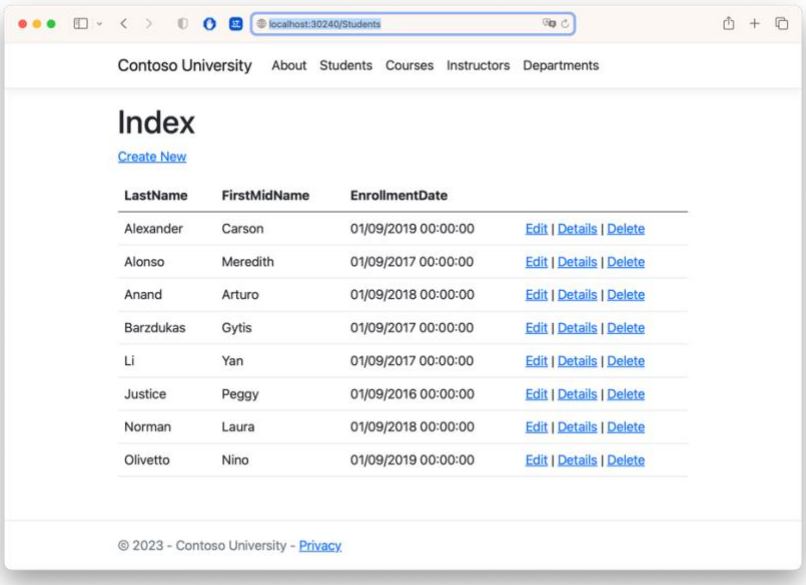
In dit eerste deel van de tutorial begin je dus met het maken van een nieuw project. Ook in dit geval is het type project wat je in de bestaande solution moet aanmaken een *ASP.NET Core Web App*. Zie ook het screenshot in 1.1.

Aan de beschreven *Prerequisites* (voorwaardes) heb je al voldaan als je de vorige Razor Pages hoofdstukken hebt gevolgd. De instructies in dit kopje kun je dus overslaan.

De instructies in het kopje *Optional: Build the sample download* kun je ook overslaan.

Zorg dat aan het einde van de stap *Create the web app project*, en dus vóórdat je de eerste code aanpassingen doet, dat je project goed bouwt en dat je de Razor page in je webbrowser op de local host hebt draaien. Als dit het geval is, *commit* en *push* je de code met als message: **Opdracht 11.1a**.

Aan het einde van de laatste stap zou je web applicatie er als volgt moeten uitzien, met op de Students page de lijst met students die je ook in de *seeding* code kunt terugvinden (zie class DbInitializer).



Als dit het geval is, *commit* en *push* je de code met als message: **Opdracht 11.1b**.

3.2 Create, Read, Update Delete

Lees het volgende deel van deze tutorial genaamd *Create, Read, Update Delete*:

https://learn.microsoft.com/en-us/aspnet/core/data/ef-rp/crud

3.3 TBD

<To Be Done>