FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

Labo frameworks voor serverapplicaties
ASP.NET MVC

8 & 11 december 2025

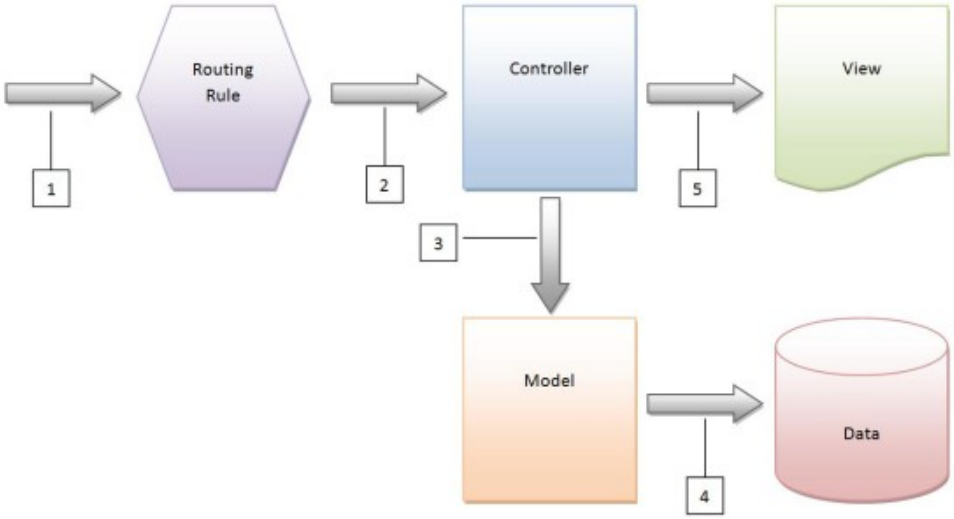
1 Inleiding

In deze reeks maken we kennis met een nieuw framework dat gebruikt kan worden voor het maken van complexe webapplicaties: ASP.NET MVC. Zoals de naam al doet vermoeden is het een implementatie van het MVC (Model-View-Controller) design pattern voor .NET webapplicaties. We maken gebruik van Visual Studio 2022 als IDE.
In dit labo maken we een webapplicatie voor een theateracademie. We maken hiervoor gebruik van een MVC-webapplicatie met ASP.NET Core. Dit is een open-source framework en werkt het cross-platform.

2 De ASP MVC structuur in het kort (herhaling/toepassing theorie)

Uit de naam kan je al afleiden dat een ASP.NET Core MVC applicatie uit drie soorten componenten bestaat:

- Model: Bevat de businesslogica van de applicatie, zorgt voor de opslag van gegevens.
- Controller: Verantwoordelijk voor het afhandelen van requests, aanpassingen doorgeven aan het model en returnt een view aan de gebruiker.
- View: Het visuele aspect van de applicatie.



Figuur 1: ASP.NET MVC model

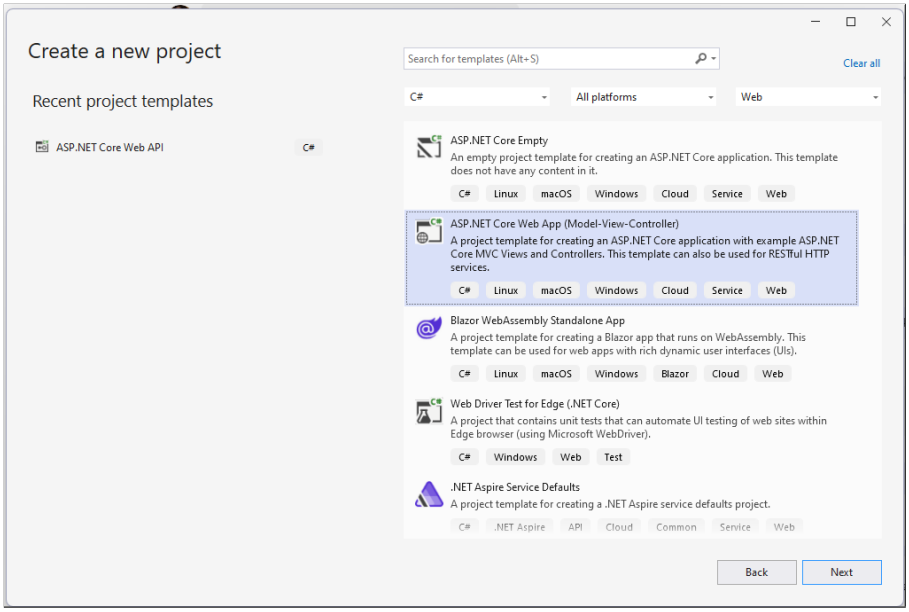
Op basis van de url wordt er nagegaan welke methode van welke controller de request moet afhandelen. Deze methode communiceert met het model en roept een view op. Eventueel kunnen er extra gegevens aan deze view doorgegeven worden.

De interactie in code tussen deze componenten gaat als volgt:

- De methodes (bv. *Index*) van een Controller (bv. *ExampleController*) zijn gekoppeld aan de url's van de webapplicatie. Optioneel hebben deze methodes een aantal parameters die overeenkomen met data in een HTTP bericht.
- Binnen deze methodes van de Controller staat de code die communiceert met het Model.
- De Views hebben dezelfde naam als de methodenaam van de Controller. Een View is een .cshtml-bestand dat je terug vindt in een submap van "Views" met dezelfde naam als de Controller (vb *Views/Example/Index.cshtml*).
- Deze methodes in de Controller hebben als returntype vaak een "Task", "ActionResult" of "IActionResult". Typisch wordt een "View()"-methode aangeroepen om de cshtml-code van de View naar een effectieve HTML pagina te vertalen en naar de gebruiker te sturen. Je kan variabelen meegeven aan de "View()"-methode om zo de gewenste data in de View te injecteren.

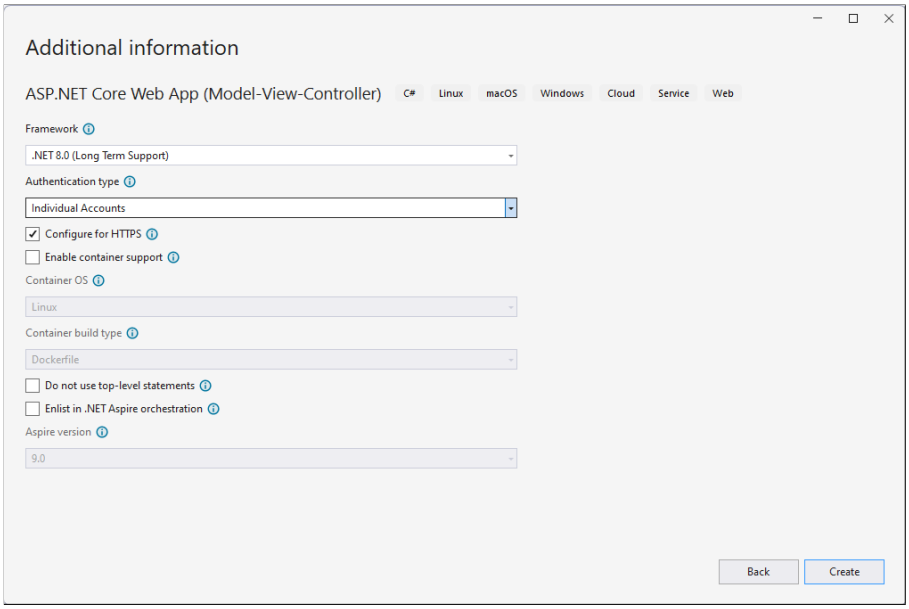
3 Nieuw project

- Maak in Visual Studio een nieuw "Visual C#" project van het type "ASP.NET Core Web App (Model-View-Controller)", zie figuur 2. Kies als locatie voor je project een plek op je lokale schijf.



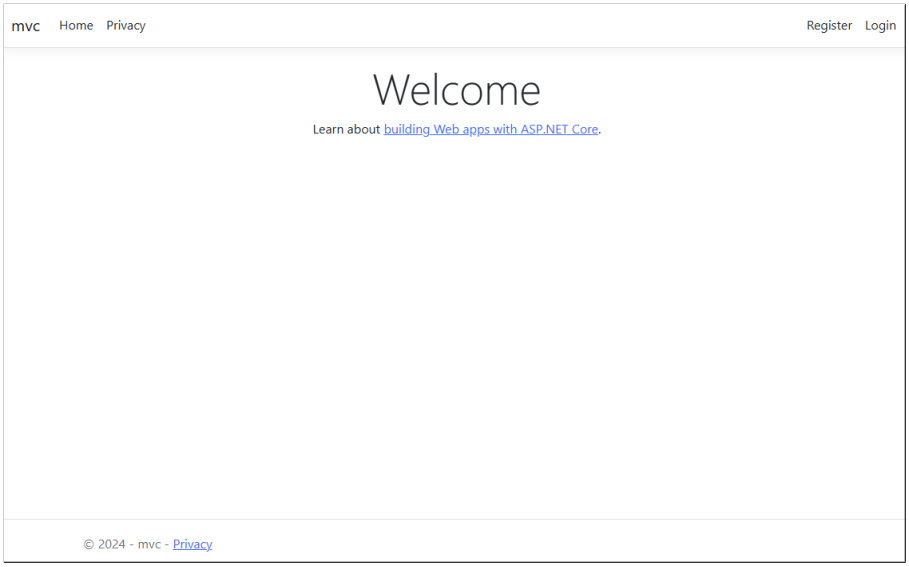
Figuur 2: Een nieuw "ASP.NET Core Web App (Model-View-Controller)" project.

- Zorg dat Authentication op "Individual Accounts" ingesteld staat, aan de andere checkboxes moet je niets veranderen, zie figuur 3.



Figuur 3: De "Web Application (MVC)" template met authenticatie

- Je kan de applicatie al eens uitproberen door deze te deployen ("play"-knop). Indien je een waarschuwing krijgt over de SSL-certificaten mag je die aanvaarden en het self-signed certificaat installeren. Je zal de website zoals in figuur 4 bekomen. Bekijk de code van de Controller en Views eens goed, alsook de gegenereerde HTML-code aan client side in je browser. Let ook op de URLs van de verschillende pagina's.

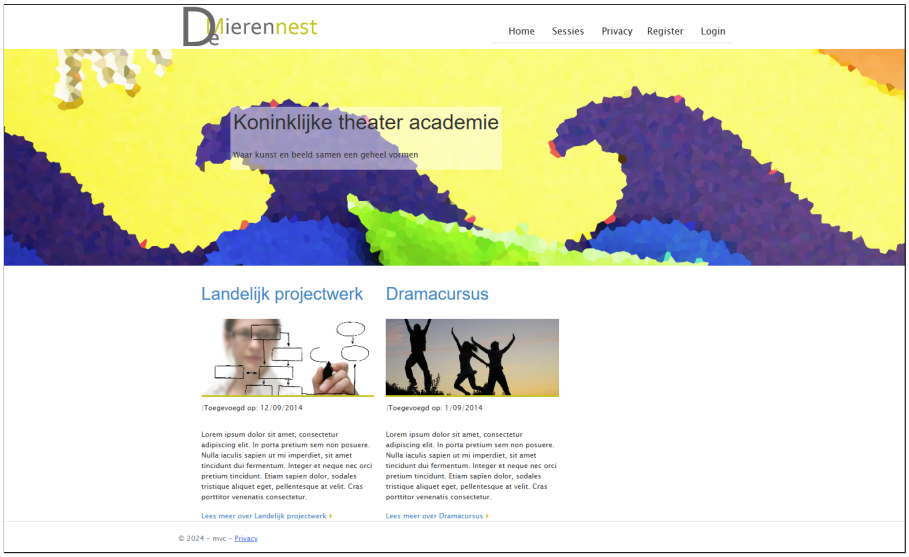


Figuur 4: De standaard indexpagina van een “ASP.NET Core Web App (Model-View-Controller)”

- Komt de layout je bekend voor ... ? Waar vind je de library hiervoor terug in je project?
- Bekijk de ‘Privacy’-pagina. Waar wordt in het project de titel voor deze pagina gedefinieerd? En waar de ondertitel?
- Waar vind je de balk met de navigatielinks terug in de code? Vergelijk de cshtml-files met de gegenereerde HTML code in je browser. Kijk bijvoorbeeld eens naar de navbar. Zie je de verschillen?

4 Home Page

- Plaats de code uit de gegeven bestanden (uit de map startcode van Ufora) op de correcte plaats in je project, hiervoor zal je manueel ook code moeten kopiëren. Zorg ervoor dat je Home pagina eruit ziet als op de onderstaande afbeelding.



Figuur 5: De vernieuwde index pagina.

4.1 _layout

Via _ViewStart.cshtml wordt bij een basis project steeds doorverwezen naar _Layout.cshtml. Dit bestand kan gezien worden als een Masterpage die alle gemeenschappelijke code bevat voor de verschillende Views van je project.

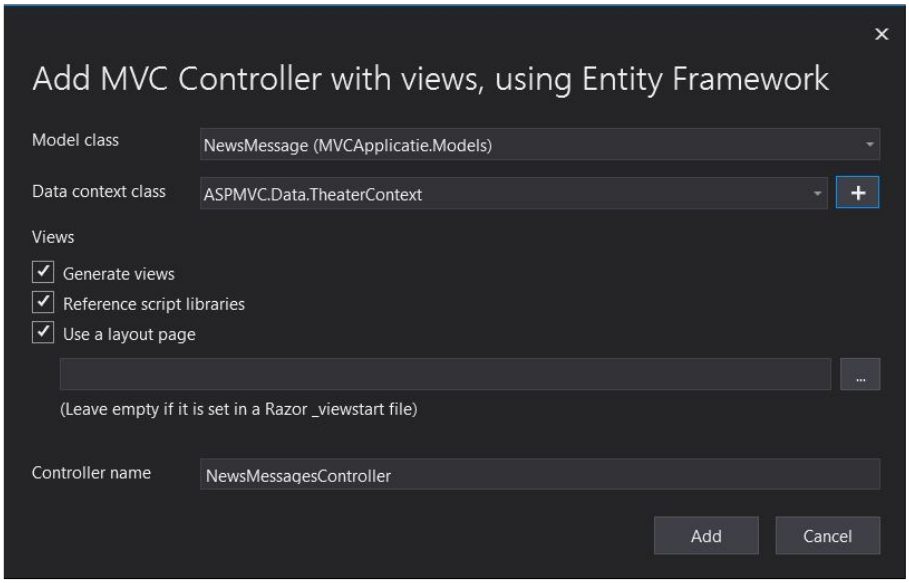
- Vervang de hard-coded links door gebruik te maken van de MVC controls om de navigatie correct in te stellen. Plaats de navigatie op de correcte plaats in het project. (Hint: Actionlinks of tag-helpers)

5 CRUD scaffolded controller

Een webapplicatie die een front-end is voor een achterliggende dataset zal typisch volgende zaken bevatten: een overzichtslijst, een pagina voor aanmaak van nieuwe data, de mogelijkheid om data aan te passen of te verwijderen, etc. Dit zijn de typische CRUD (Create, Update, Delete) operaties. Dit betekent dat hiervoor verschillende C#-methodes in een controller aangemaakt moeten worden, alsook de bijhorende cshtml views.

Omdat dit vaak voorkomende code is, bestaan er in Visual Studio Code Generation tools om deze code automatisch te laten genereren (scaffolding). We hoeven die dan enkel nog aan te vullen met de juiste data.

- Als model gebruiken we hetzelfde “NieuwsBericht” als in vorig labo. Voeg de meegegeven NewsMessage.cs klasse toe aan je project (op de correcte locatie). Belangrijk: pas namespace aan naar dezelfde als rest van de applicatie.
- Maak een nieuwe NewsMessagesController met scaffolding, via rechtermuisknop in de solution overview: Add - New Scaffolded item. Selecteer de "MVC Controller with views, using Entity Framework". Als Model selecteer je de NewsMessages.cs klasse. Bij ‘Data context class’ moet je een nieuwe ‘TheaterContext’ context aanmaken (deze zal data persistentie verzorgen). De checkboxes voor de aanmaak van views mogen geselecteerd zijn (zie figuur 7). Indien je een error krijgt over code-generation, probeer opnieuw. Als dat niet werkt, verwijder en voeg de package “CodeGeneration.Design” eens opnieuw toe via NuGet Package Manager.



Figuur 6: Aanmaak van een Controller met scaffolding en Entity Framework

5.1 Koppeling Database (idem vorige labo)

Om de koppeling tussen model en database compleet te maken moeten we wel nog een aantal extra stappen ondernemen.

In appsettings.json zien we dat de 'localdb' dataserver gebruikt wordt.

- Je kan deze localdb SQL server bekijken via menu-component View - 'SQL Server Object Explorer'. Momenteel vind je hier nog geen database voor ons project terug.

We moeten de database nog aanmaken met de EF Core Migrations feature van Visual Studio. Migrations maakt een database die matcht met het datamodel.

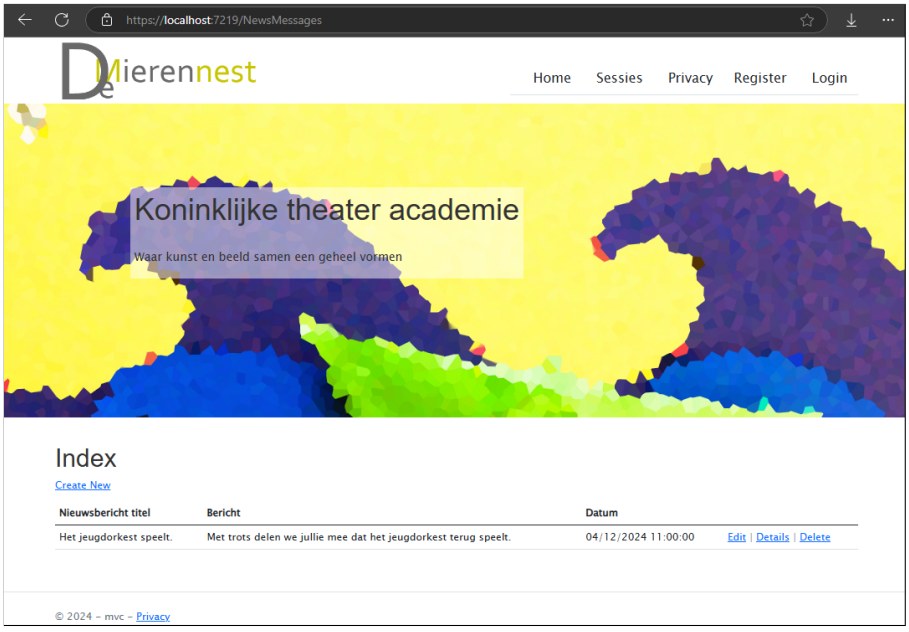
- In het Tools menu ga je naar 'NuGet Package Manager' - 'Package Manager Console'. Typ daar volgende commando's in:
  `Add-Migration Initial -Context TheaterContext`
  `Update-Database -Context TheaterContext`

Na een refresh zou je nu in localdb een database moeten zien met dezelfde naam als de connectiestring in de appsettings.json.

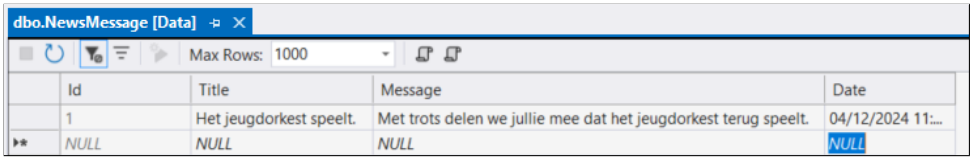
- Bekijk de table 'dbo.NewsMessage' in de database. Deze is de omzetting van je Model-klasse naar een relationele database. Via rechtermuisknop kan je ook de aanwezige data bekijken.

5.2 Run applicatie

- Run je project en browse naar de URL die naar de start-view van je nieuwe controller verwijst. Welke URL is dat?
- Voeg via deze webinterface nu enkele nieuwsberichten toe, pas deze aan of verwijder ze weer en bekijk de opslag van deze data in de database (je moet deze manueel refreshen). Zie figuur 7 en 8



Figuur 7: Web view van scaffolded controller.



Figuur 8: Database view van data

6 Views aanpassen

Onze index pagina toont nu een overzicht van alle (hard-gecodeerde) nieuwsberichten. Deze willen we nu aanpassen zodat de nieuwsberichten uit de database gebruikt worden.

- Pas de index view (van NewsMessages) aan zodat deze dezelfde layout heeft als de huidige index view (van Home). Alle berichten die nu op het overzicht getoond worden zijn afkomstig uit de database die in de vorige stap is aangemaakt. Zorg er ook voor dat indien de ID van een nieuwsbericht te hoog is, de dummyimage getoond wordt in plaats van een image gebaseerd op de ID van het bericht.
- Pas de routing aan zodat default de Index pagina wordt getoond van de NewsMessagesController en niet die van de HomeController. Ook bij de navigatie zal de link moeten worden aangepast.
- Zorg voor een gepaste foutboodschap indien er geen nieuwsberichten zijn om weer te geven. Waar dien je dit te implementeren? In de Model, de View of de Controller? Test je implementatie uit door een lege lijst van nieuwsberichten aan de view mee te geven.
- Zorg ervoor dat je de mogelijkheid voorziet op de indexpagina om een nieuwsbericht toe te voegen.
- Voorzie bij de Detailpagina van een nieuwsbericht de mogelijkheid om een nieuwsbericht te verwijderen of om het aan te passen.
- Zorg dat je na aanmaak van een nieuw nieuwsbericht naar de detailpagina van het toegevoegde bericht gaat, in plaats van naar de Index view. Waar dien je dit aan te passen?
- Gebruik annotaties in je model om er voor te zorgen dat ‘Title’, ‘Message’ en ‘Date’ niet leeg kunnen zijn. Voorzie een foutboodschap. Pas ook de labels bij de input velden aan zoals in Figuur 9.

Figuur 9: Create pagina van een nieuws bericht.

7 Internationalisatie

Internationalisatie (I18N) beschrijft het proces om een applicatie meerdere verschillende talen en regio's te laten ondersteunen (globalisatie, G11N), en deze applicatie hiervoor aan te passen (localisatie, L10N) [1]. Hiertoe zal de applicatie meerdere 'cultures' (synoniem: 'locales') ondersteunen. Deze kunnen neutraal (een specifieke taal zonder regio, bv. 'en' voor Engels, 'es' voor Spaans, enz.) of specifiek (een specifieke taal met een bepaalde regio, bv. 'en-US' voor Amerikaans Engels, 'en-GB' voor Brits Engels, 'es-CL' voor Chileens Spaans, enz.) zijn.

We willen beginnen met het internationaliseren van de boodschap die wordt weergegeven wanneer geen nieuwsberichten beschikbaar zijn.

- Wanneer er geen nieuwsberichten beschikbaar zijn, diende je eerder een aangepaste boodschap weer te geven in je Index View. Zorg ervoor -indien je dit nog niet reeds deed- dat de tekst van deze (Nederlandstalige) boodschap in je NewsMessagesController gedefinieerd wordt via `ViewData`. Gebruik dan de `ViewData` in de Index View, zoals we reeds eerder deden.
- Voeg een `IStringLocalizer _localizer` toe als instantievariabele in de Controller die je wil internationaliseren. In ons geval is dit dus de NewsMessagesController. (Je kan de HomeController later zelf ook nog internationaliseren.) Via constructor based Dependency Injection, injecteer je deze Localizer in je controller (net zoals je met de Context voor object-database mapping eerder automatisch hebt laten doen). Bekijk het voorbeeld op https://docs.microsoft.com/en-us/aspnet/core/fundamentals/localization.
- We moeten deze localisatieservice nu wel expliciet gaan definiëren in onze webapplicatie opdat de webapplicatie de juiste injectie in de controller kan uitvoeren. Voeg daartoe volgende

[1] https://docs.microsoft.com/en-us/aspnet/core/fundamentals/localization

opdracht toe in Program.cs:
`builder.Services.AddLocalization(options => options.ResourcesPath = "Resources");`

- Zorg er vervolgens voor dat je boodschap (die wordt weergegeven wanneer geen berichten beschikbaar zijn) kan geïnternationaliseed worden door deze als key van **`_localizer`** te gebruiken. Start je webapplicatie en bekijk deze boodschap wanneer geen berichten aanwezig zijn. Omdat we nog nergens een value voor deze key gedefinieerd hebben, zal de key zelf gebruikt worden.

Momenteel werkt onze applicatie nog steeds hetzelfde zoals voor het toevoegen van de localization, maar nu kunnen we eenvoudig vertalingen toevoegen.

- In de opties van onze lokalisatieservice hebben we verwezen naar 'Resources' als path voor alle bestanden die de verschillende vertalingen zullen bevatten. Maak daarom een 'Resources' map aan binnen je project. Dit kan door met de rechtermuisknop op je project te klikken en te kiezen voor 'Add > New Folder'.

- Deze Resources folder zal dezelfde subfolders bevatten zoals in de root van je project voor de componenten waar je vertaling wil voorzien, tenminste indien ook je namespace hiermee overeenkomt.
  De namespace van je controller zal bijvoorbeeld **namespace myproject.Controllers** zijn. De vertalingen voor deze controller zullen dus in de subfolder 'Controllers' binnen 'Resources' moeten komen. Maak deze subfolder nu aan.

- Maak hierin een NewsMessagesController.en.resx resource file en een NewsMessagesController.fr.resx resource file aan (gebruik dezelfde naam als je controller). Je kan deze files aanmaken via rechtermuisknop 'New Item > Resources file'.

- Maak nu een Engelse en Franse vertaling van je boodschap in deze resources file.

Nu moeten we nog bepalen welke vertaling naar de client gestuurd moet worden.

- We moeten eerst in onze webapplicatie configureren welke 'cultures' we effectief willen ondersteunen. Daartoe moeten we via **`UseRequestLocalization`** de juiste opties meegeven aan onze ApplicationBuilder. Dit gebeurt in Program.cs . Gebruik hiervoor onderstaand stukje code. Zorg dat dit als eerste opgeroepen wordt op het app-object.

```
var supportedCultures = new[] {"nl","en","en-US","fr","fr-FR"};
var localizationOptions = new RequestLocalizationOptions()
    .SetDefaultCulture(supportedCultures[0])
    .AddSupportedCultures(supportedCultures)
    .AddSupportedUICultures(supportedCultures);
app.UseRequestLocalization(localizationOptions);
```

- Test je vertaling door de culture expliciet mee te geven in de URL als parameter. Bv. surf naar https://localhost:xxxxx/NewsMessages?culture=fr

- Wanneer geen URL-parameter wordt meegegeven, zal er gekeken worden naar de **`accept-language`** header field van de HTTP-request. Immers, je browser zal een voorkeur voor een bepaalde culture automatisch naar elke web server sturen, op basis van de ingestelde taal van je browser en/of besturingssysteem. Bekijk met de developer tools van Chrome welke culture wordt aangegeven in je HTTP-request wanneer je surft naar je webapplicatie.

Nu heb je de basis van internationalisatie werkende gekregen voor inhoud die vanuit je Controller wordt aangemaakt. We zorgen in de volgende stappen dat de tekst in de Views zelf

ook kan geïnternationaliseerd worden. Beschouw bijvoorbeeld in eerste instantie de 'Voeg een nieuwsbericht toe' button in de Index.cshtml View file van onze NewsMessagesController.

- Voeg volgende toe aan je Index.cshtml:

```
@using Microsoft.AspNetCore.Mvc.Localization
@inject IViewLocalizer Localizer
```

- Gebruik vervolgens **`@Localizer`** op een gelijkaardige manier zoals je in je Controller **`_localizer`** gebruikte. https://docs.microsoft.com/en-us/aspnet/core/fundamentals/localization.
- Maak nu ook resx Resources files voor Engels en Frans voor deze Index View.
- Om dit werkende te krijgen, moeten we nu ook nog specifiëren in onze webapplicatie dat de Views ook 'gelocaliseerd' kunnen worden. Vervang hiertoe in Program.cs:
  `builder.Services.AddControllersWithViews();`
  door
  `builder.Services.AddControllersWithViews().AddViewLocalization(`
  `LanguageViewLocationExpanderFormat.Suffix);`

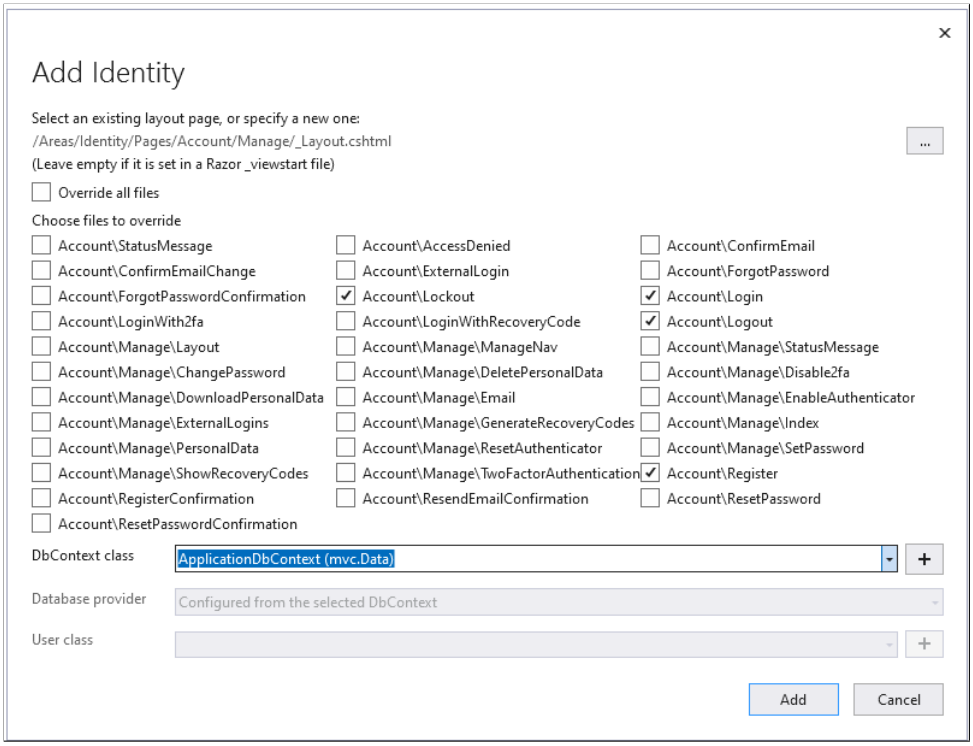
Ten slotte willen we ook de zaken die in ons Model staan, gaan vertalen. Dit is alles wat we via DataAnnotations gedefinieerd hebben, zoals de DisplayName en error messages.

- Indien je dit nog niet gedaan had, zorg dan voor een DisplayName voor de velden 'title' en 'place'. Zorg ook via annotatie voor een foutboodschap wanneer deze velden leeg zijn bij aanmaak/wijzigen van een nieuwsbericht.
- Zorg voor passende Resources files in Engels en Frans.
- Om vertaling van de annotaties mogelijk te maken is geen equivalent nodig als de **`_localizer`** in Controller of **`@Localizer`** in View. Het enige wat nog moet gebeuren is in Program.cs is het volgende toe te voegen
  `builder.Services.AddControllersWithViews()` **`AddDataAnnotationsLocalization();`**

8 Internal authenticatie en autorisatie

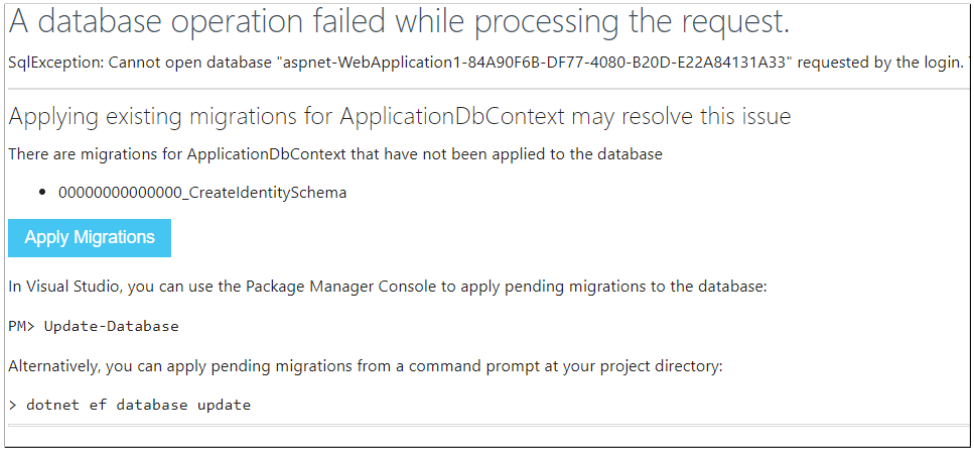
We zouden er graag voor zorgen dat alleen een ingelogde gebruiker nieuwe berichten kan toevoegen in de applicatie. Hiervoor zullen we de autorisatie moeten instellen voor de gewenste acties. We hebben bij de start van dit project gekozen voor individual user accounts als authenticatiemethode. Hierdoor werd voor ons al wat configuratie toegevoegd aan het project.

- Het ASP.NET framework heeft al enkele bestanden voor ons gegenereerd zoals views (onder Views/Shared en Areas/Identity). We voegen nog enkele pagina's toe via rechtermuisknop op het project > add new scaffolded item en vervolgens Identity te selecteren. Deze voeg je toe en je selecteert de gewenste pagina's om te overschrijven. (Zie figuur 10.) Als Data Context Class selecteer je de ApplicationDbContext. Meer info op https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity



Figuur 10: Scaffold Identity

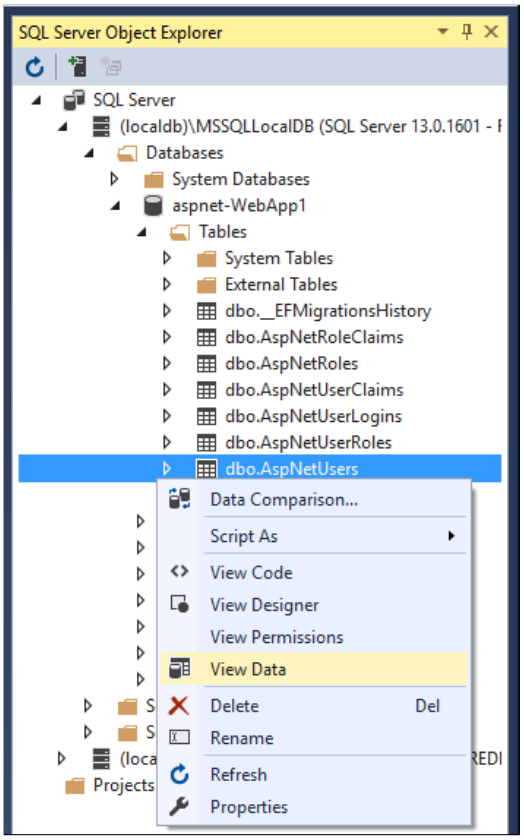
- Indien je tijdens vorig labo de _loginPartial hebt verwijderd uit de _layout.cshtml, dan voeg je hem nu weer toe. Zo stellen we de login en registreer mogelijkheden beschikbaar aan de gebruiker. Hiervoor voeg je de `<partial name="_LoginPartial/>` toe aan de navigatie van het project. Dit komt na het sluiten van de ul tag.
- Start de applicatie en probeer een account te registreren. Bij het uittesten van het inloggen kan je een bepaalde foutmelding krijgen over het toepassen van de database migrations (zie afbeelding 11). Als je deze melding ontvangt klik je op de "Apply Migrations" button en refresh je de pagina. Hierna zal de functionaliteit beschikbaar zijn voor de applicatie.



Figuur 11: Add migrations

- We moeten er nu eerst voor zorgen dat een gebruiker die niet is ingelogd geen nieuw bericht kan toevoegen aan de pagina. Hij mag ook geen berichten kunnen aanpassen en verwijderen. Voeg hiervoor de juiste annotaties toe op de correcte locaties in de controller van de NewsMessages en test uit of je de pagina's nog kan bezoeken. Wat is het verschil wanneer je de pagina's bezoekt terwijl je wel of niet bent ingelogd? Meer info op https://docs.microsoft.com/en-us/aspnet/core/security/authorization/simple

- Bekijk welke data er werd opgeslaan in de database bij het registreren van een nieuw account. Gebruik opnieuw de "SQL Server Object Explorer" om de database te inspecteren, zie afbeelding 12.



Figuur 12: Bekijk de data in de Identity Database.

- Om een gebruiker te laten registreren zouden we graag enkele van de default regels aanpassen.

Deze configuratie komt terecht in Program.cs. Standaard moet een wachtwoord een minimum lengte van 6 karakters hebben bestaande uit zowel kleine als hoofdletters, een cijfer en een "ander" teken. Pas de standaardinstellingen aan zodat een wachtwoord minimum een lengte van 8 karakters moet hebben en minstens vijf verschillende letters. We willen er ook voor zorgen dat het aantal pogingen voor het ingeven van een foutief wachtwoord beperkt wordt tot 3 keer voordat een account 5 minuten wordt geblokkeerd. Pas de correcte configuratie toe. Meer info op https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity-configuration.

Bekijk ook de volgende code. Pas aan, indien nodig.

```
// Areas/Identity/Pages/Account/Login.cshtml.cs
var result = await _signInManager.PasswordSignInAsync(Input.Email,
    Input.Password, Input.RememberMe, lockoutOnFailure: true);
```

- Stel ook in dat de email die gebruikt wordt om te registreren bij de applicatie uniek is.
- Zorg ervoor dat de knoppen voor het toevoegen van een nieuw bericht op de index-pagina, en het aanpassen of verwijderen van een bestaand bericht op de detail-pagina, verborgen zijn voor gebruikers die niet zijn ingelogd.
  Hint: Bekijk de code van LoginPartial waarin de knoppen ook alleen worden getoond als de gebruiker nog niet is ingelogd.

9 External authentication

Vaak willen gebruikers liever geen nieuw account aanmaken voor elke website die ze gebruiken maar inloggen via een door hun vertrouwde identity provider. Wij gaan zorgen dat gebruikers in onze applicatie ook kunnen registreren met hun Google account. Indien je zelf geen persoonlijk Google account hebt, kun je de instructies op volgende pagina volgen voor een andere identity provider: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/social/#setup-login-providers-required-by-your-application.

- Voeg de package "Microsoft.AspNetCore.Authentication.Google" toe aan je project via de NuGetPackage Manager.
- Maak een nieuw project aan met de naam "Frameworks" op het Google Cloud Platform
- Vervolgens configureren we het "OAuth consent screen" op Google Cloud. Kies voor *User Type* External, kies een naam voor jouw applicatie en vul jouw gmail adres in als *user support email* en *developer contact information*. Ga door de andere stappen maar behoud de default instellingen.
- Nu kunnen we OAuth credentials genereren voor onze applicatie. Ga hiervoor naar credentials en klik op *Create Credentials > OAuth client ID*. We kiezen voor "Web application" als type en voeg de correcte redirect URI toe (https://localhost:<port>/signin-google). Je kan de poort terugvinden bij starten van jouw applicatie of in de launchSettings.json file. Kopieer de ClientID en ClientSecret.
- Voeg onderstaande code toe in Program.cs om je applicatie te laten authenticeren bij Google. Test eerst je applicatie uit met je secrets in plain text, in volgend puntje zullen we een secret manager gebruiken.

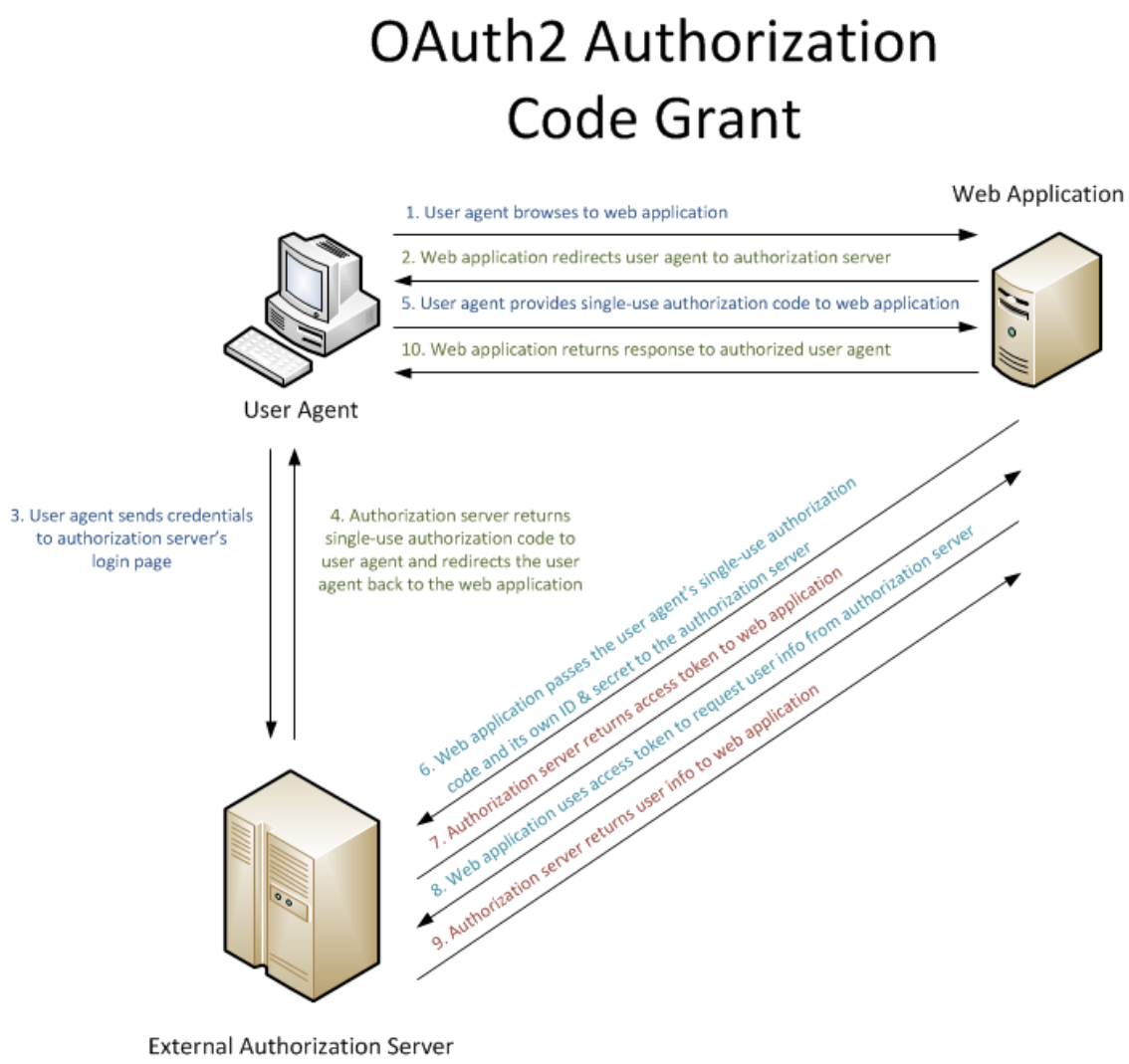
```
builder.Services.AddAuthentication().AddGoogle(
```

```
googleOptions  =>
{
    googleOptions.ClientId = <client_id>;
    googleOptions.ClientSecret = <secret_value>;
});
```

- De secrets toevoegen als plain text in de code is een bad practice. Binnen het .NET framework bestaat er een secret manager die de secrets apart van de code opslaat en toe laat om deze te injecteren in de applicatie. Open een terminal in de folder van jouw project en voer onderstaande commands uit om de secret store te initialiseren en het clientid en secret toe te voegen. https://docs.microsoft.com/en-us/aspnet/core/security/authentication/social/microsoft-logins

```
dotnet user-secrets init
dotnet user-secrets set "Authentication:Google:ClientId"
"<client-id>"
dotnet user-secrets set "Authentication:Google:ClientSecret"
"<client-secret>"
```

- Verwijder de plain text secrets die je initieel had toegevoegd uit de code en vervang deze door de secrets op te halen uit de secret manager.
- Figuur 13 geeft de verschillende stappen weer uit de OAuth 2.0 flow. Open in de browser de network tab in de Developer Tools. Welke berichten zie je passeren?
  - In stap 2 redirects onze web applicatie de gebruiker naar de authorisatie server, in dit geval Google. Wat is de gevraagde OAuth scope? Wat vind je nog terug in dit bericht?
  - Nadat de gebruiker correcte credentials heeft gestuurd naar de authorisatie server stuurt deze de gebruiker terug naar de redirect-url. Aan deze redirect-url wordt de authorisatie code toegevoegd, vind je deze terug?



Figuur 13: De OAuth2 code authorisatie flow.