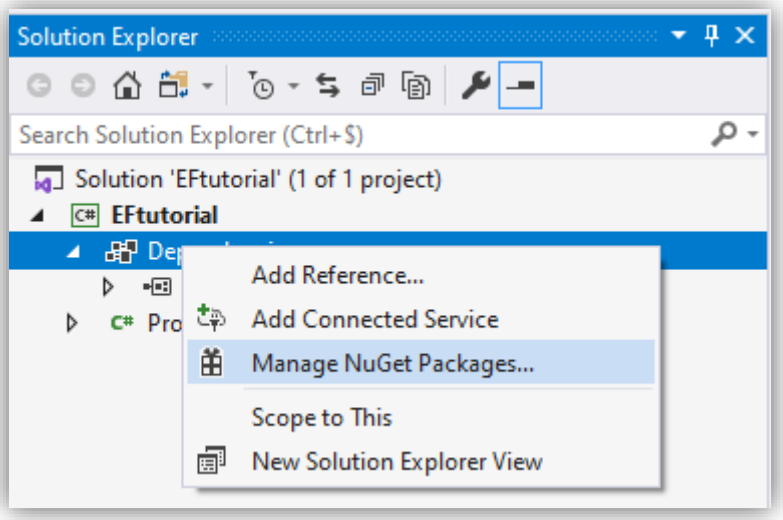
Entity Framework Core

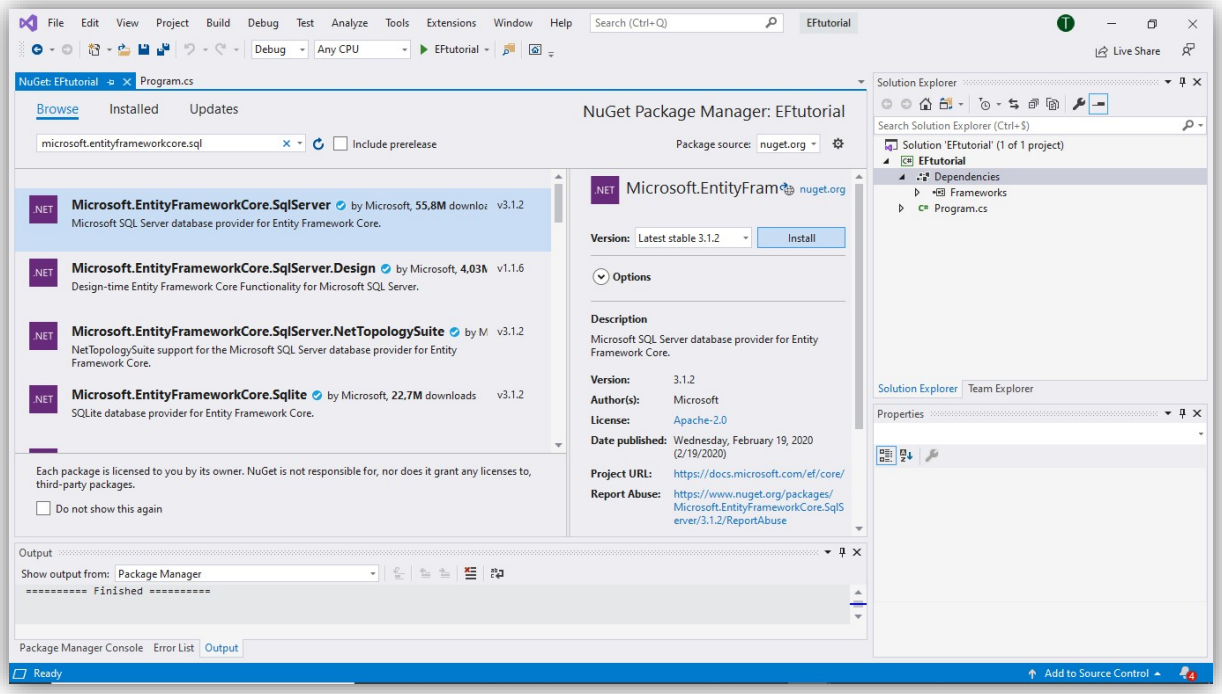
We zullen aan de hand van een voorbeeld een introductie geven over Entity Framework (EF) Core, waarbij we gebruik maken van Visual Studio en SQL Server.

Project

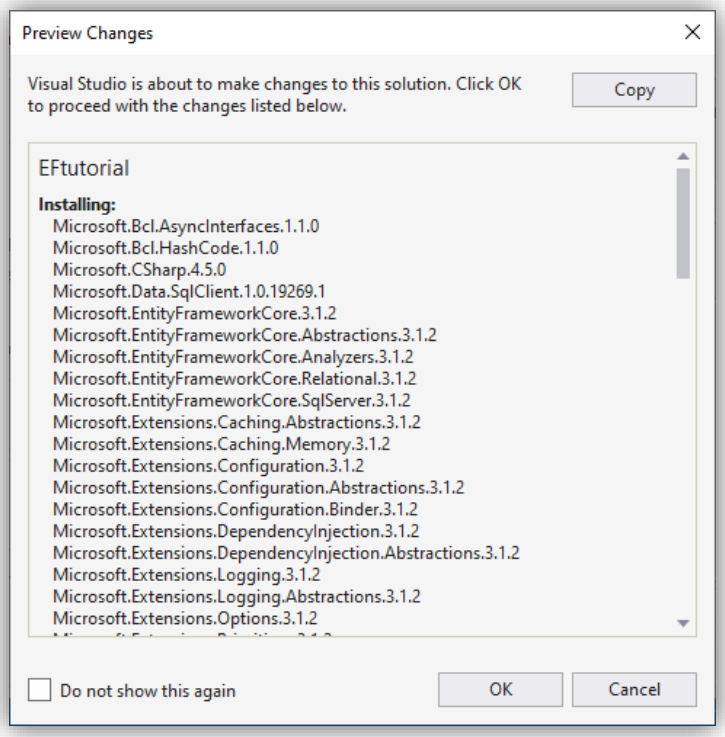
We starten met het aanmaken van een nieuw project EFtutorial van het type Console App (.NET 5.0 of .NET 6.0). Eénmaal dit project aangemaakt installeren we de packages die nodig zijn om EF te gaan gebruiken. We selecteren 'Dependencies' in het project, en via de rechtermuistoets kunnen we een menu laten verschijnen met daarin 'Manage NuGet Packages'.

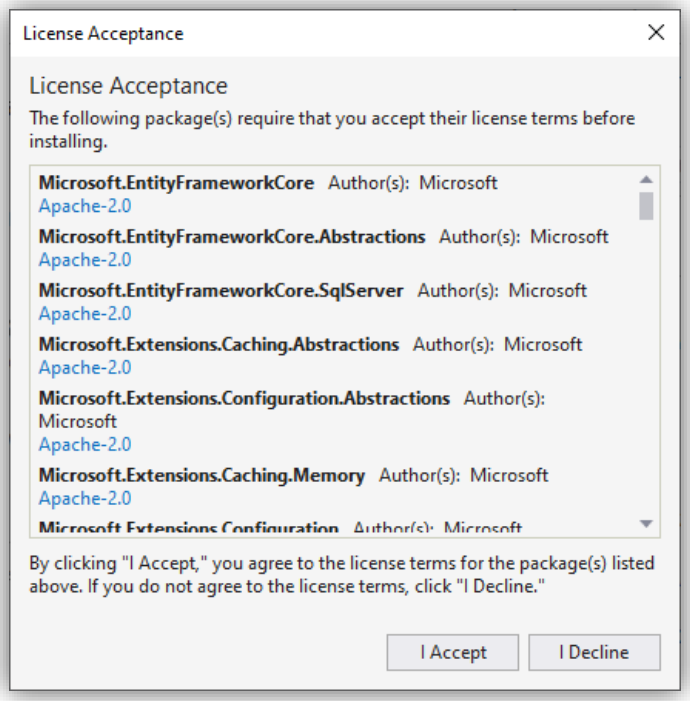


Een venster 'NuGet Package Manager' opent zich en na het selecteren van 'Browse' (links boven) gaan we op zoek naar het package 'Microsoft.EntityFrameworkCore.SqlServer' dat we dan ook installeren.

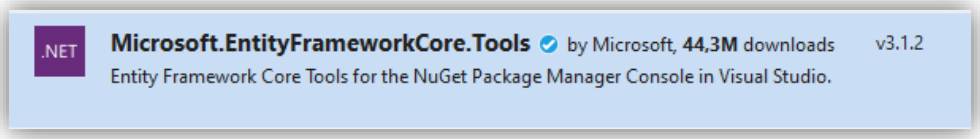


Er verschijnen een aantal vensters die je informeren over de aanpassingen die zullen plaatsvinden en een vraag om aan de licentie-voorwaarden te voldoen.

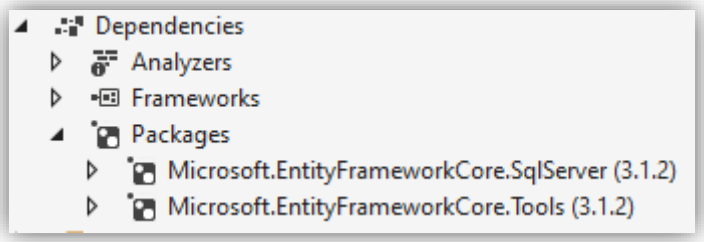




Daarna installeren we nog een tweede package, namelijk Microsoft.EntityFrameworkCore.Tools, dat we zullen gebruiken voor de aanmaak/update van de databank-tabellen.

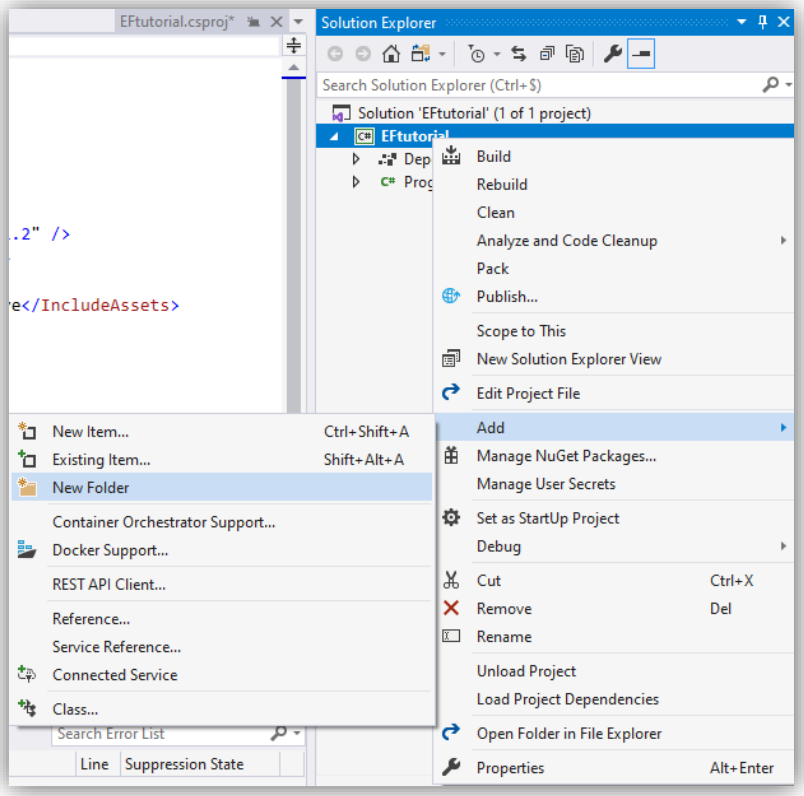


Wanneer beide packages zijn geïnstalleerd zijn deze ook te zien als ‘Dependencies’ in de submap ‘Packages’.



Gegevensmodel

In deze tweede stap gaan we het gegevensmodel aanmaken, we opteren namelijk voor een ‘Code First’- aanpak. De klassen die we gaan gebruiken groeperen we in een aparte folder ‘Model’. Een nieuwe folder kan worden aangemaakt door met de rechtermuistoets te klikken op het project en dan via demenus ‘Add’ en ‘New Folder’.



De eerste klasse die we aanmaken is de klasse Auteur. Belangrijk hierbij is dat we een property ‘Id’ aanmaken, deze zal later door EF automatisch worden gebruikt als ‘Primary Key’.

```
public class Auteur
{
    0 references
    public Auteur(string naam, string emailContact)
    {
        Naam = naam;
        EmailContact = emailContact;
    }
    0 references
    public int Id { get; set; }
    1 reference
    public string Naam { get; set; }
    1 reference
    public string EmailContact { get; set; }
}
```

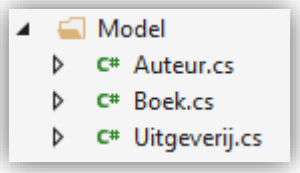
Een tweede klasse is Uitgeverij en ook hier voorzien we een Id property.

```
public class Uitgeverij
{
    0 references
    public Uitgeverij(string naam, string adres, string emailContact)
    {
        Naam = naam;
        Adres = adres;
        EmailContact = emailContact;
    }
    0 references
    public int Id { get; set; }
    1 reference
    public string Naam { get; set; }
    1 reference
    public string Adres { get; set; }
    1 reference
    public string EmailContact { get; set; }
}
```

Onze derde klasse is Boek en deze klasse verwijst naar een uitgeverij en bevat ook een lijst van auteurs. Belangrijk bij deze klasse is de constructor, voor EF is het noodzakelijk dat er een constructor is die geen verwijzingen heeft naar andere klassen, vandaar dat we de eerste eenvoudige constructor toevoegen die enkel een titel en beschrijving als parameters heeft.

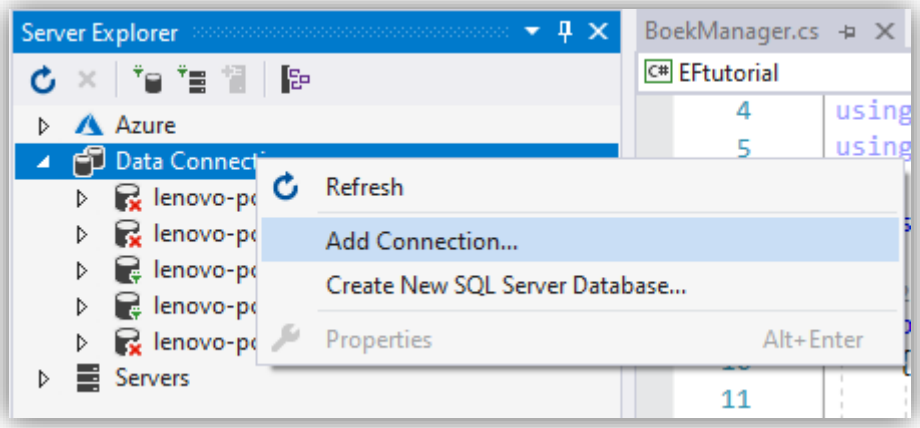
```
public class Boek
{
    0 references
    public Boek(string titel, string beschrijving)
    {
        Titel = titel;
        Beschrijving = beschrijving;
    }

    0 references
    public Boek(string titel, string beschrijving, Uitgeverij uitgeverij, List<Auteur> auteurs)
    {
        Titel = titel;
        Beschrijving = beschrijving;
        Uitgeverij = uitgeverij;
        Auteurs = auteurs;
    }
    0 references
    public int Id { get; set; }
    2 references
    public string Titel { get; set; }
    2 references
    public string Beschrijving { get; set; }
    1 reference
    public Uitgeverij Uitgeverij { get; set; }
    1 reference
    public List<Auteur> Auteurs { get; set; } = new List<Auteur>();
}
```

Het resultaat is dan een folder met onze drie klassen.

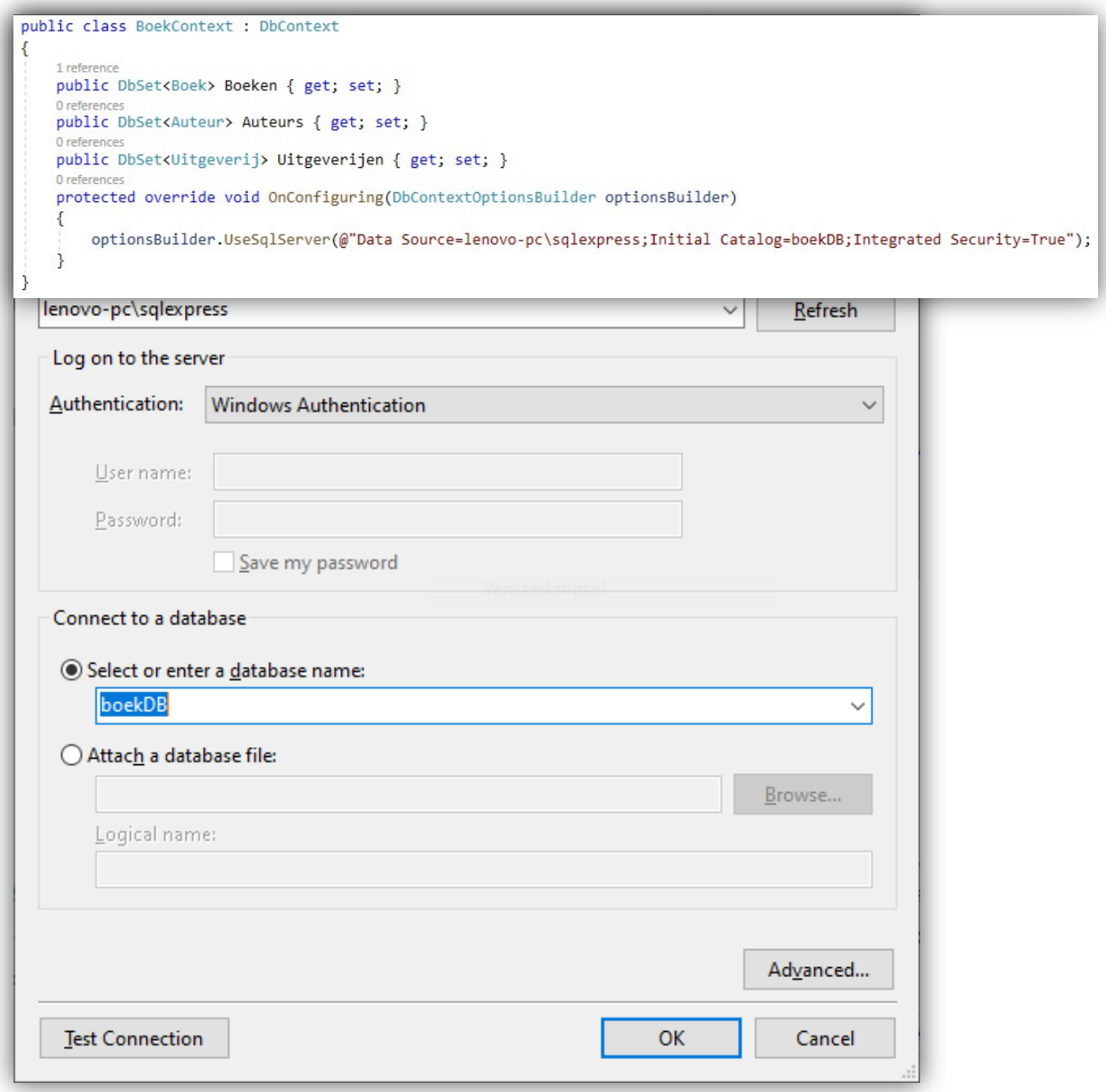


Databank migration

In de derde stap gaan we nu onze klassen laten vertalen naar een databankmodel. We beginnen met het toevoegen van een connectie naar de databank in het Server Explorer venster.



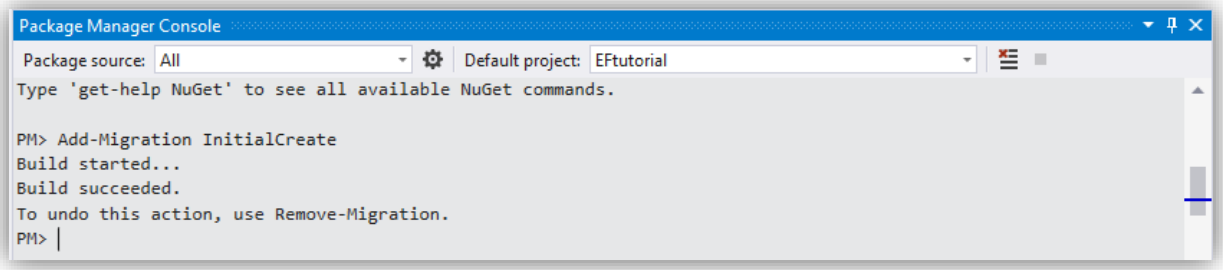
Voor het toevoegen van een connectie hebben we een 'data source' nodig, dit verwijst naar onze databank-server (analoog als bij ADO.NET). We kiezen ook voor een databank, in dit geval werd er reeds een databank boekDB aangemaakt, die we nu selecteren.

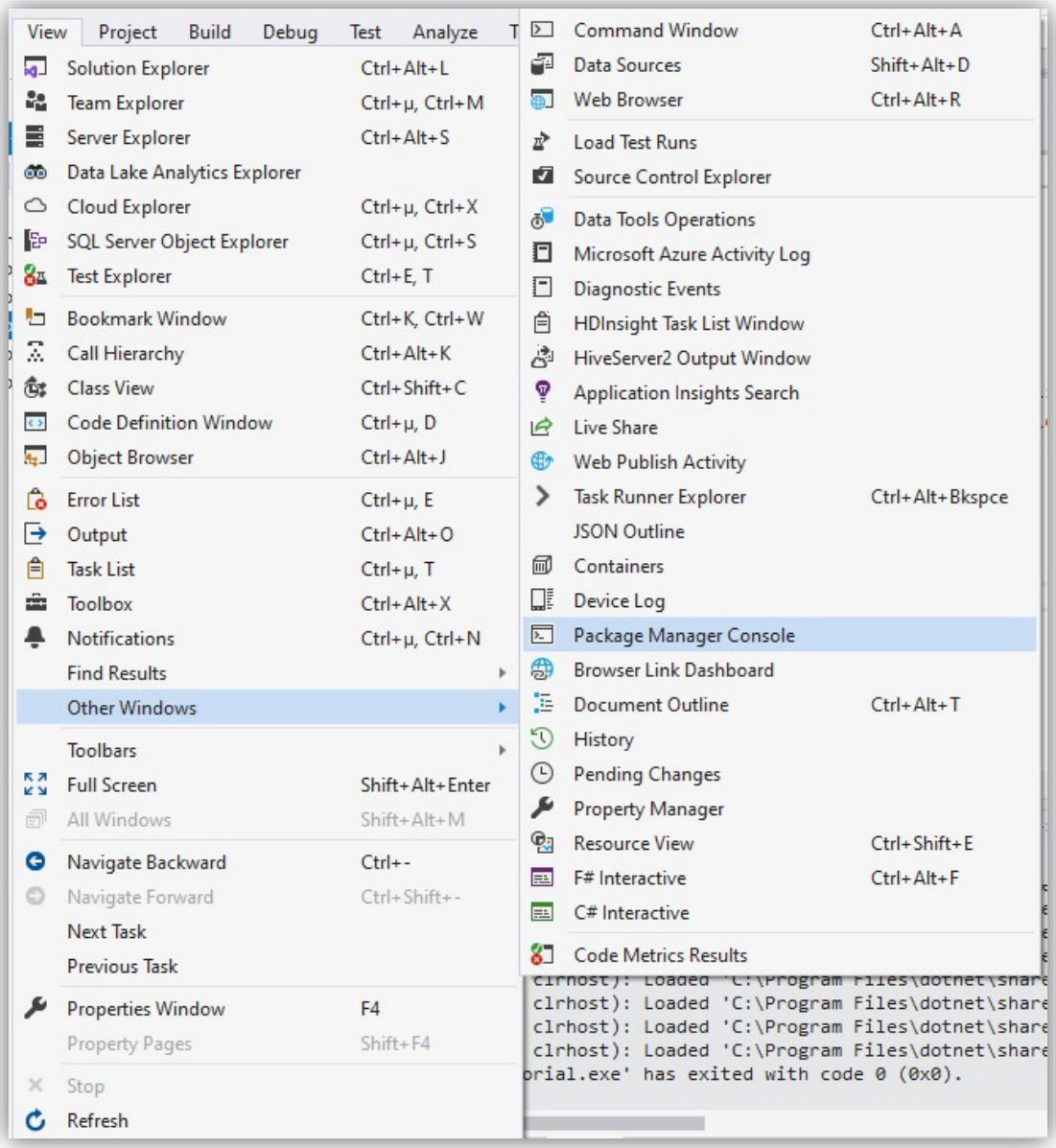


Eénmaal de connectie naar de databank gelegd kunnen we ook de connectiestring opvragen bij de properties die bij de connection horen.

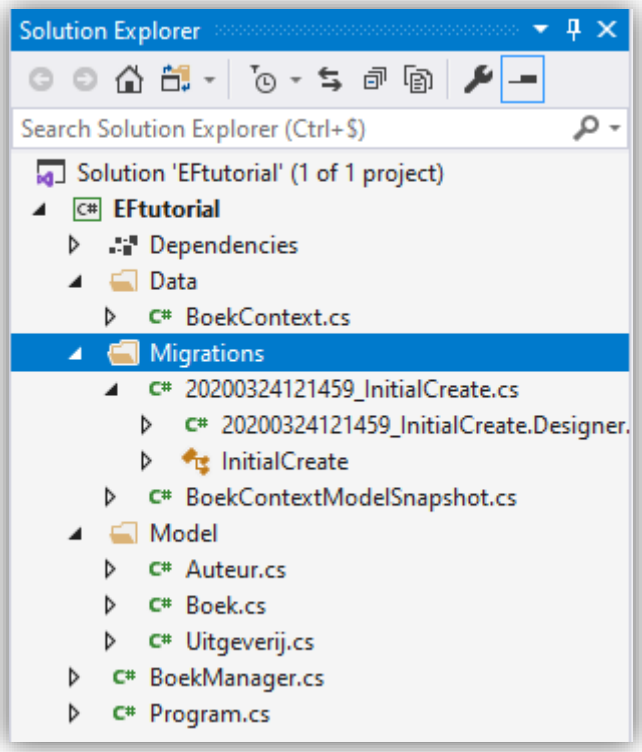
De belangrijkste klasse bij het gebruik van EF is de context klasse die we overerven van DbContext. In deze klasse geven we aan welke objecten/tabellen we gebruiken door middel van DbSets. Zo verwijst DbSet<Boek> naar de boeken in de databank. Belangrijk is om zeker niet de getter en setters te vergeten, die zijn noodzakelijk om straks de tabellen aan te maken in de databank. Naast de sets met de objecten hebben we ook een functie OnConfiguration waar we kunnen instellen met welke databank we gaan werken, in dit geval SQL Server die via de meegegeven connectiestring is beschreven.

We kunnen nu door gebruik te maken van de EntityFrameworkCore Tools de tabellen automatisch laten aanmaken. Dit doen we door in het venster Package Manager Console het commando Add-Migration uit te voeren. Indien het venster niet zou open zijn, dan kan je dat doen door in het menu 'View' te kiezen voor 'Other windows' en daarin het venster te selecteren.

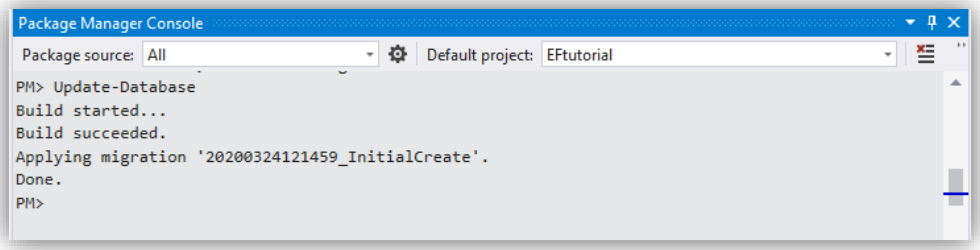




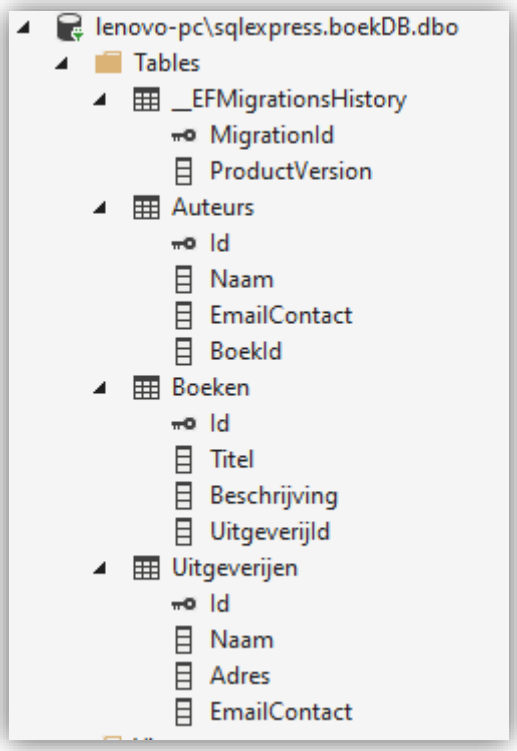
Bij het uitvoeren van het commando Add-Migration geef je ook een naam mee die je aan deze actie wil koppelen. Na het uitvoeren van deze instructie zal er een nieuwe folder verschijnen in het project met de naam Migrations. In deze folder vind je dan enerzijds de klasse terug met de naam die je aan de Add-Migration als parameter hebt meegegeven en anderzijds ook een klasse die verwijst naar je DbContext klasse met als achtervoegsel Snapshot. In de eerste klasse vind je een beschrijving in code van je tabellen, de tweede geeft de huidige toestand weer van het model en zal bij nieuwe migrations (updates) steeds worden aangepast.



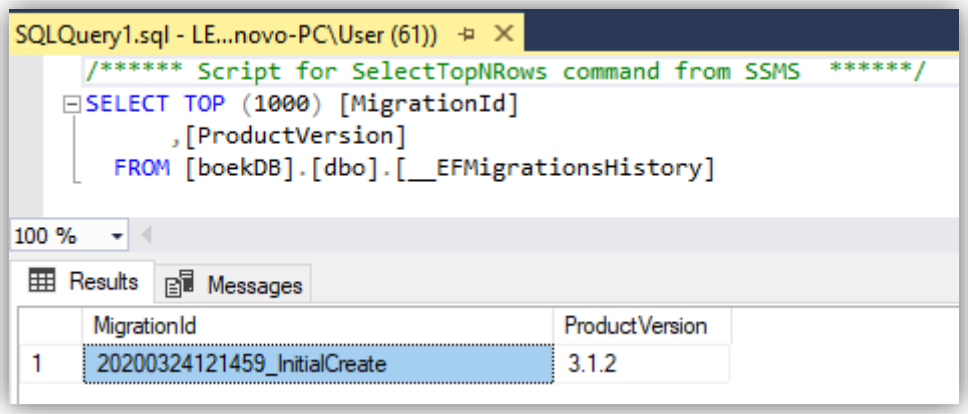
Om nu het datamodel effectief aan te maken in de databank geef je het commando Update-Database in het Package Manager Console venster.



Het resultaat zal dan zijn de tabellen waar onze objecten in worden opgeslagen en een tabel __EFMigrationsHistory die de verschillende migraties weergeeft.



Bekijken we de inhoud van de migrations tabel dan vinden we hier een korte beschrijving die verwijst naar onze aangemaakt 'InitialCreate' migratie.



Test run

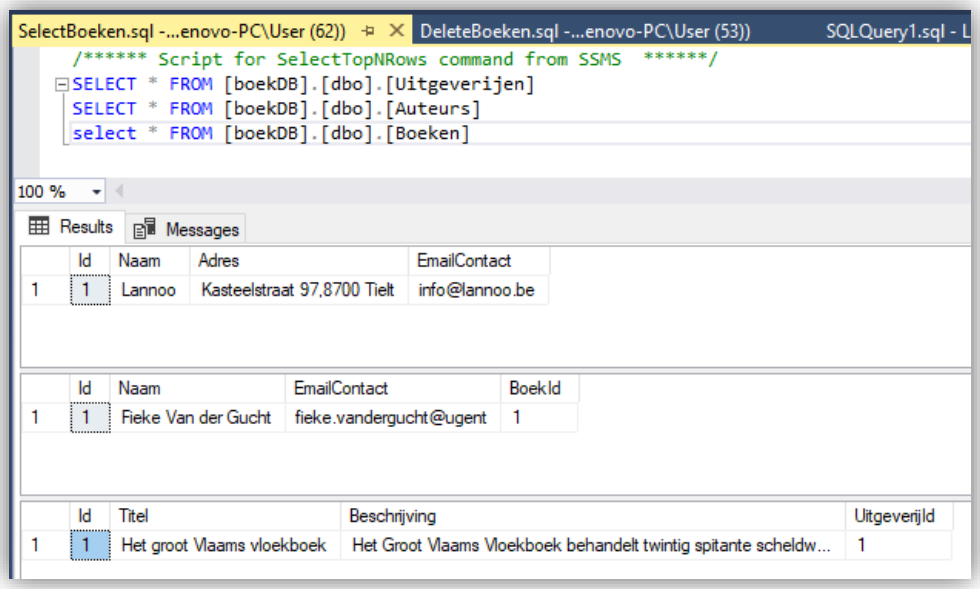
Het gegevensmodel is nu aangemaakt. In de volgende stap schrijven we gegevens weg naar de databank. We maken daartoe een klasse BoekManager die beschikt over onze DbContext en die een moethode VoegBoekToe() implementeert. Het toevoegen van een nieuw object boek kan vrij eenvoudig door gebruik te maken van de Add()-methode van DbSet<Boek> uit de contextklasse. Om de aanpassingen door te voeren in de databank moeten we nu enkel nog de veranderingen opslaan en dat kan door middel van de functie SaveChanges() die we overerven van DbContext.

```
public class BoekManager
{
    private BoekContext ctx = new BoekContext();
    1 reference
    public void VoegBoekToe(Boek boek)
    {
        ctx.Boeken.Add(boek);
        ctx.SaveChanges();
    }
}
```

Het uitvoeren van de methode VoegBoekToe() kan er als volgt uit zien :

```
static void Main(string[] args)
{
    Console.WriteLine("Hello World!");
    Uitgeverij u1 = new Uitgeverij("Lannoo", "Kasteelstraat 97,8700 Tielt", "info@lannoo.be");
    Auteur a1 = new Auteur("Fieke Van der Gucht", "fieke.vandergucht@ugent");
    Boek b1 = new Boek("Het groot Vlaams vloekboek", "Het Groot Vlaams Vloekboek behandelt twintig
    b1.Uitgeverij = u1;
    b1.Auteurs.Add(a1);

    BoekManager BM = new BoekManager();
    BM.VoegBoekToe(b1);

}
```

Het resultaat kunnen we bekijken in SSMS, waarbij we zien dat niet enkel de tabel boeken is ingevuld, maar ook auteurs en uitgeverij.



Referenties

https://docs.microsoft.com/en-us/ef/core/

https://docs.microsoft.com/en-us/ef/core/modeling/constructors

https://docs.microsoft.com/en-us/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli#create-a-migration

https://docs.microsoft.com/en-us/ef/core/miscellaneous/cli/powershell

https://www.youtube.com/watch?v=PpqdsJDvcxY

https://www.entityframeworktutorial.net/efcore/install-entity-framework-core.aspx