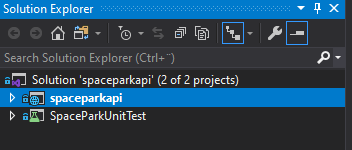
Hur delade vi upp våra projekt?

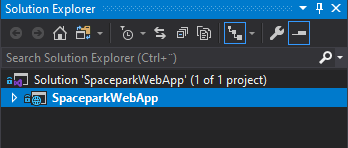
Vi diskuterade och funderade kring hur vi skulle skapa våra Frontend och Backend projekt.

Sen har vi bestämt att skapa tre projekt med två solutioner:

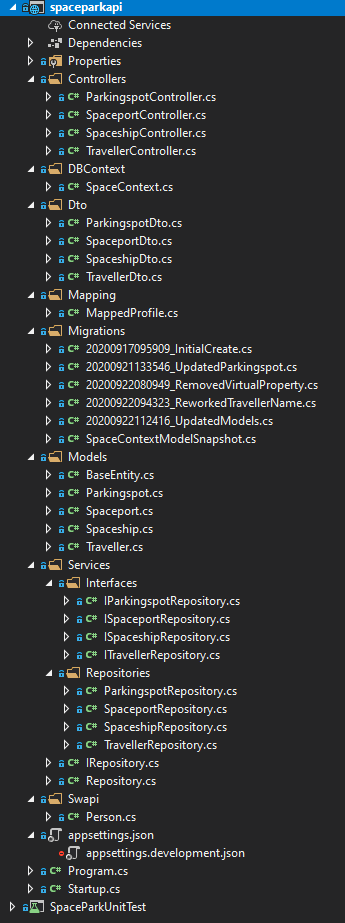
* Backend solution har två Projekt (SpaceParkAPI + SpaceParkUnitTest):



* Frontend solution har ett projekt:



Förklaring av kod:

**SpaceparkAPI:**

För vår API projektet valde vi att göra ett .NET Core API projekt som använder Repository mönster samt använde vi oss DTO (Data Transfer Objekt) mönster som har flera fördelar i API projekt som tar bort cirkulära referenser och dölj vissa egenskaper som klienter inte ska visa och mycket mer.

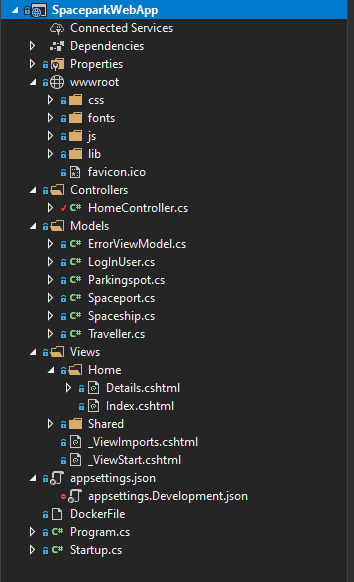
Vi valde att använda Entity Framworks med kod först, så skapade vi modell klasser som entitets .

Alla metoder som pratar med databasen och med SWAPI API står i Controller (Get och Post metoder..).

Vi använde oss Swapi/Person klass som entity-modell när vi kallar Travellers från Swapi API.

Vi hade två appsettings.json filer:

* En vanlig appsetting fil son har moln ConnectionString.
* En development appsetting fil som local sql-databas ConnectionString.

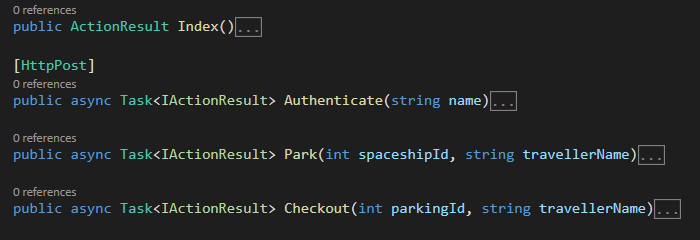
**SpaceparkWebApp:**

Vi diskuterade kring hur vi skulle skapa våran Frontend och funderade i början på att skapa *Razor Pages därför att vi har en enkel Frontend applikation. Men tyvärr lyckas vi inte att hitta tillräckligt material för att lära oss Razor Pages tekniken och kan man använda dem med Web API’s applikationer. Så vi tog äntligen beslutet att använda ett web applikation med MVC mönster som använder Razor Pages i viewen.*

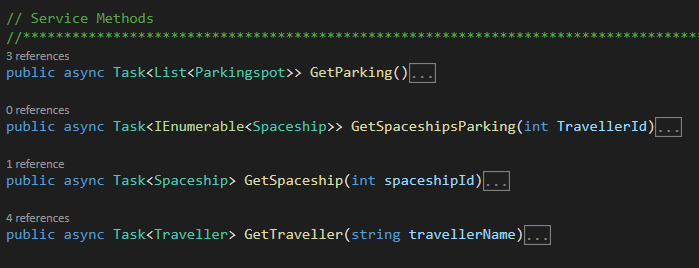
I Frontend projektet handlade inte direkt med databasen eller med Swapi API.

Vi har skapat alla metoder som skickar *http-requests* till vår API projekt och får resposes från där i HomeController klass.

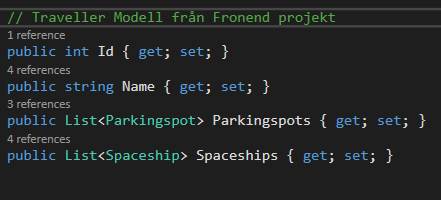
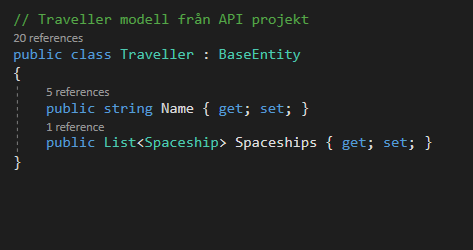
I *HomeController* har vi två typer av metoder:

1. ActionResults som returnerar en View sida i slutet.

1. Vanliga Metoder som returnerar ett objekt från våra modell klasser.



I *Models* mappen har vi alla View modeller som har ibland mer properties eller mindre än samma modeller som använde oss i vår API projektet. Det berör på hur skulle använda dem i vårt Frontend projekt.



I *View* mappen har vi bara två view sidor som skulle visas till användaren när vi kör programmet:

1. Index sida var vi matar in StarWars traveller namn som skulle parkera sina skeppar i vår parkering.



1. Details sida var visar vi Traveller namn och sina skeppar infos samt parkeringsstatus för varje skepp. Sen användare kan parkera eller avparkerea skeppen.

