# Repository Pattern and Unit of Work C# -MySQL

My goal to write this article is showing how powerful those design patterns are when they work together.

Repository Pattern  
Consider that all the operations for an entity must be separated by Repositories that implement all the Operations that are coming from an interface.

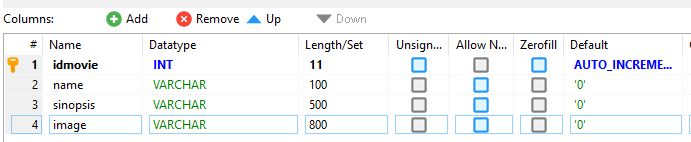
Unit of Work  
Handle all the operations defined in the Repository pattern as a transaction that could be failed or being success.

## Challenge

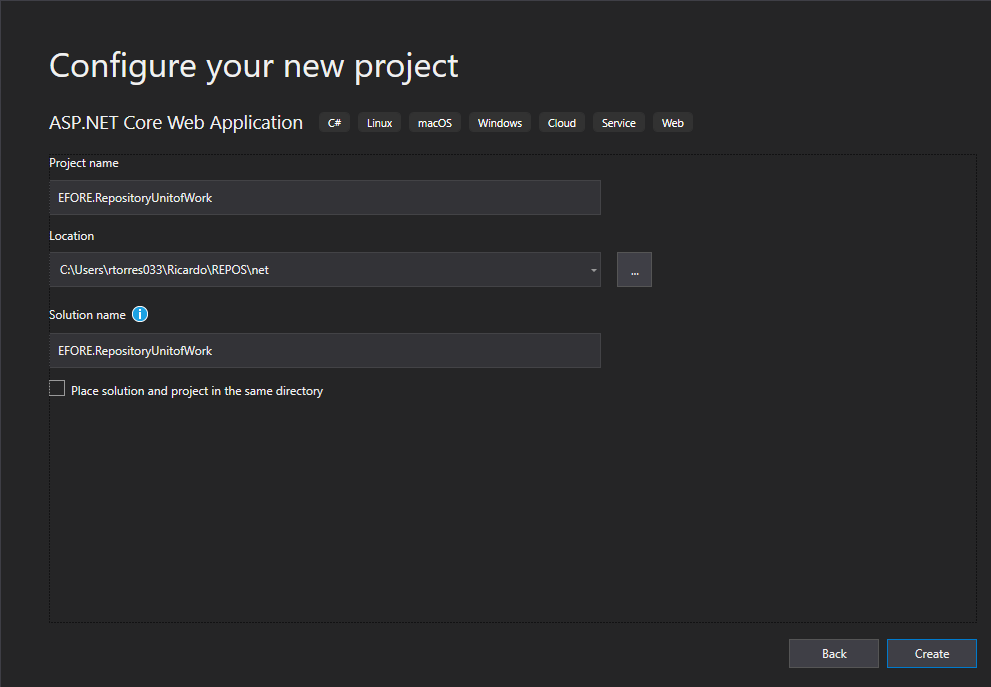
We have a database MySql named Movies and we need to expose an Api with a method *GetAll* that will retrieve data

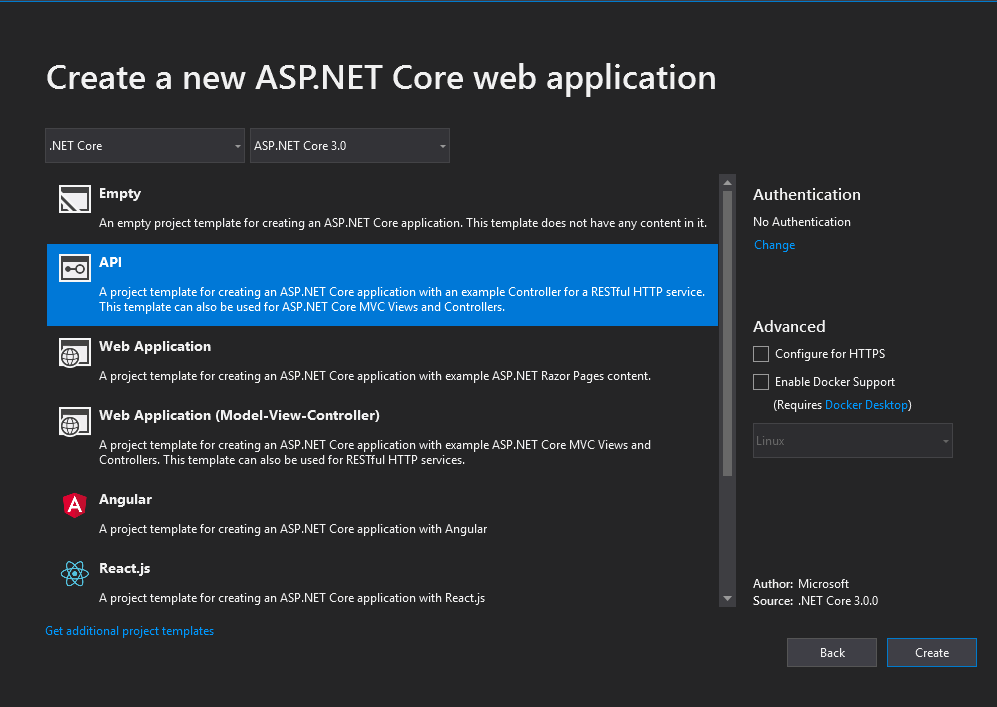
## Database Structure

Database name: movieonline

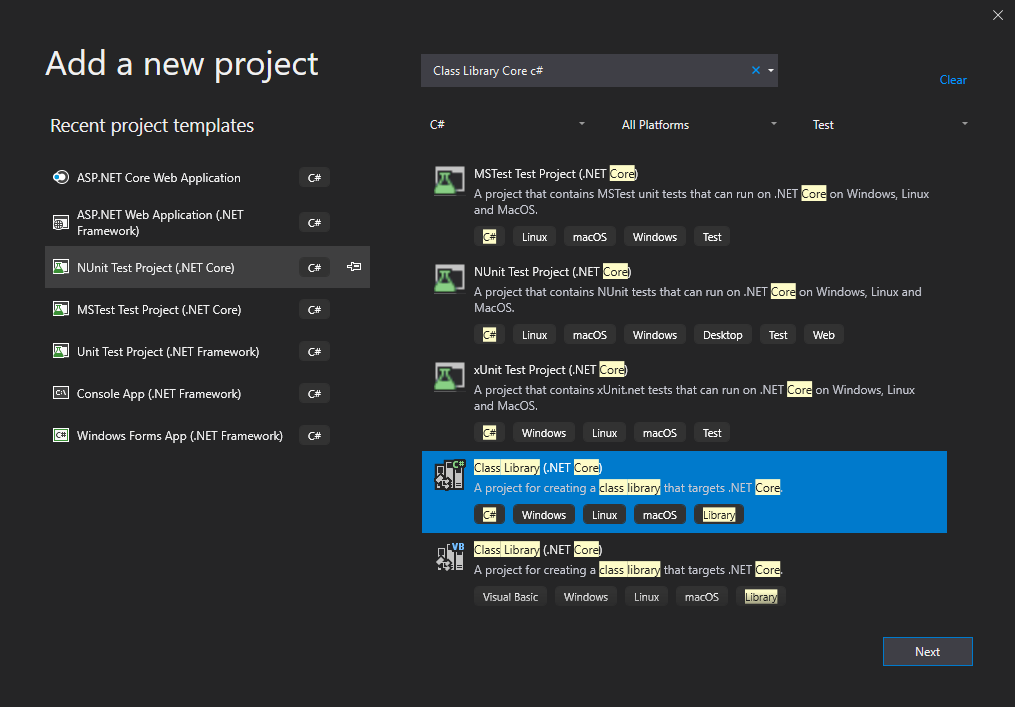


## Steps

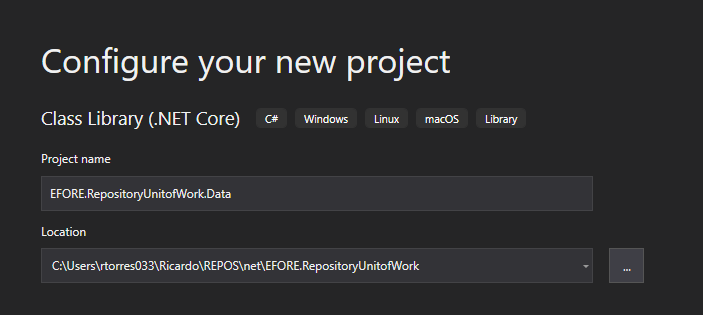
1. Create New Asp.net Core Application with Language C# and name it *EFORE.RepositoryUnitofWork*
2. Select Api.



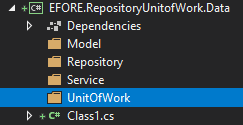
1. Create New Project Library Core in C#.



1. Name it *EFORE.RepositoryUnitofWork.Data*



1. Create an folder structure under this project like this :



1. Meaning of each folder:

Models: Contains all the classes that represent a table in the database.

Repository: Contains all the Repository for each entity that we need for each Service

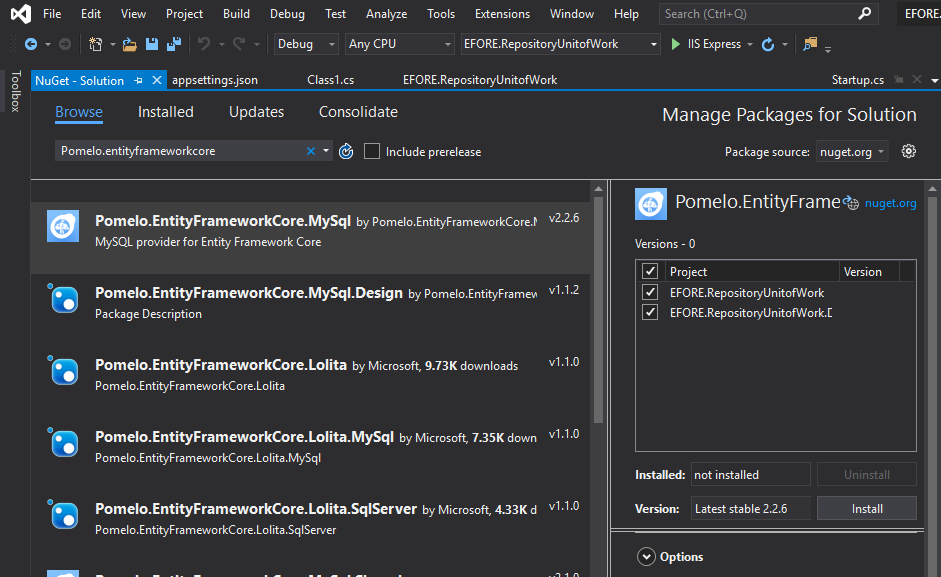
Service: Service is used to inject functionality

UnitOfWork: Contains the Unit of Work pattern to handle all the transactions the happen for each Repository.

1. Go to options -> Nuget Package Manager -> Package Manager Console o type

ALT +T+ N+O

Run this command



<https://codinginfinite.com/entity-framework-core-database-first-net-core-scaffold-mysql-db/>

<https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-3.0&tabs=visual-studio-code>