# **Unit Testen Repository in ASP.NET Core 3.1**

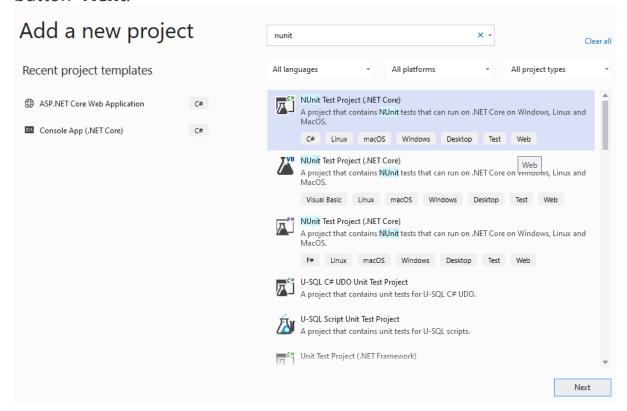
Dit document bevat een beschrijving hoe je NUnit test project toevoegt voor het testen van **ASP.NET Core 3.1** projecten, hoe je unit testen schrijft voor Repository classes en hoe je objecten kan mocken

# Creëer een NUnit test project met Moq

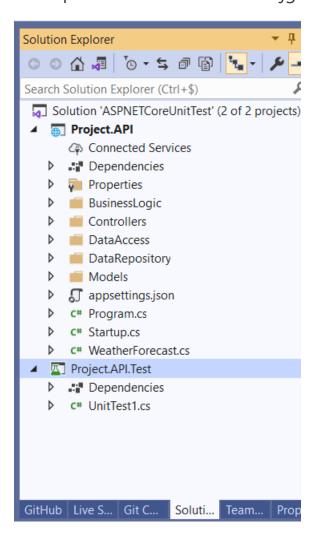
Stap 1: Rechts-klik op de project solution en selecteer de optie

## Add -> New Project

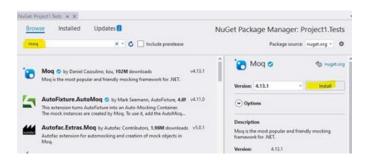
**Stap 2**: Selecteer **NUnit test project (.NET Core)** en klik op de button **Next**:



**Stap 3**: Vul de naam van het project in met *Project.API.Test* en klik dan op de button **Create**. Je krijgt het volgende te zien:



**Stap 4**: Open *NuGet* manager voor **Project.API.Test** en zoek naar de package *Moq*. Selecteer **Moq** en installeer de laatste stabiele versie in het test project:



**Stap 5:** Controleer onder Dependencies/Packages dat de Moq NuGet package is geïnstalleerd

# Maak je eerste unit test case:

We schrijven een test case voor een methode die de som van 2 gehele getallen teruggeeft.

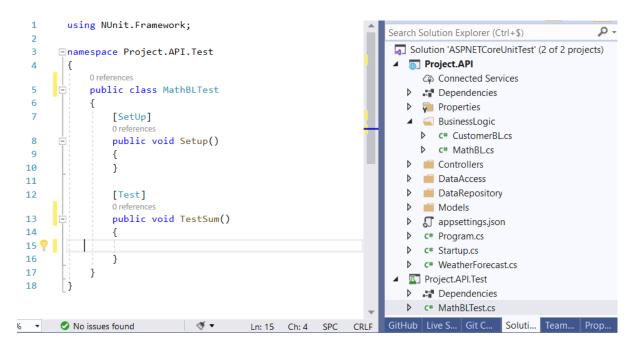
**Stap 1:** Er is reeds een class file met de naam **MathBL.cs** in de folder BusinessLogic. Deze heeft een methode **Sum()** in **Project.API**.

```
namespace Project.API.BusinessLogic
{
    Oreferences
    public class MathBL
    {
        Oreferences
        public int Sum(int num1, int num2)
        {
            int sum = num1 + num2;
            return sum;
        }
    }
}
```

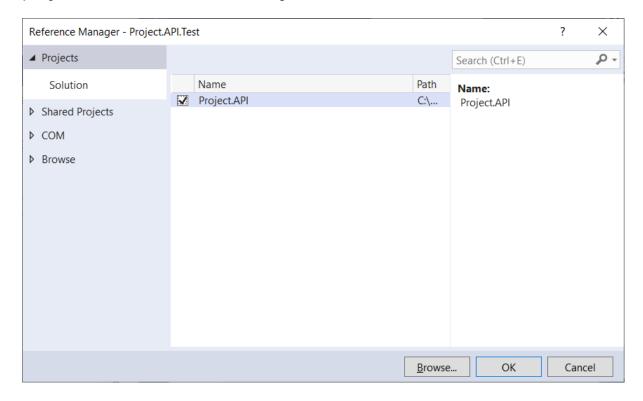
We maken een Test class en test methode aan voor MathBL:

**Stap 2**: Selecteer het **Project.API.Test** project en voeg de class file **MathBLTest.cs** toe:

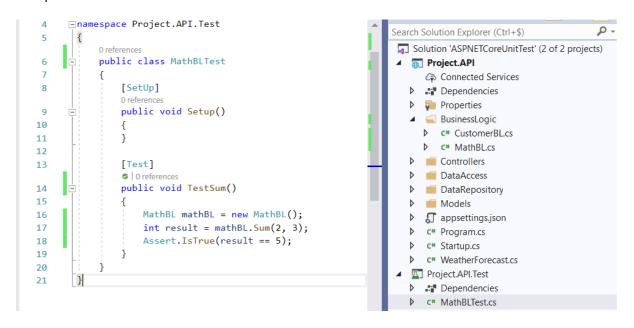
**Stap 3**: Voeg de methoden **Setup()** en **TestSum()** toe aan de MathBLTest.cs class:



**Stap 4**: Rechtsklik op de **Project.API.Test** dependencies en voeg een project referentie toe naar **Project.API**:

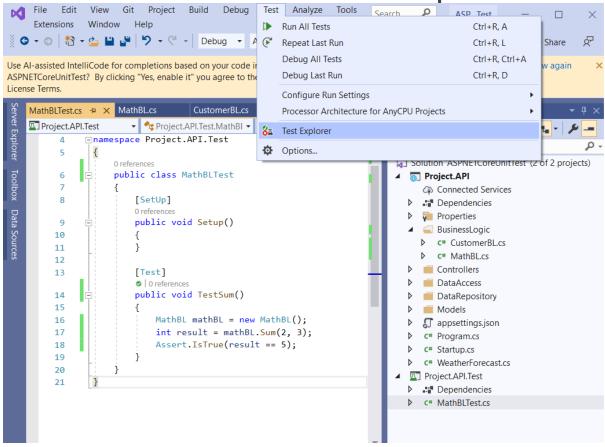


**Stap 5**: schrijf de volgende code om de **Sum()** methode aan te roepen :

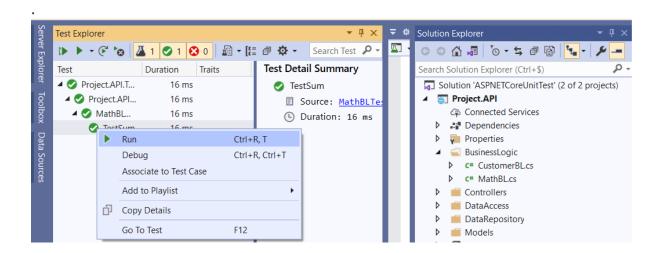


Stap 6: Build het project en controleer of de build succesvol is.

Selecteer dan van het hoofdmenu View -> Test Explorer.



# **Stap 7**: Je krijgt de lijst van Tests te zien. Rechtsklik op **MathTest.BL** en klik op **Run:**



Als de test groen kleurt, is de test succesvol verlopen.

# Unit test case voor mock database en data access service aanroep

## Implementeer business logica en database context

Neem het volgende project met code-first aanpak.

Genereer eerst de database via de volgende instructive in de Package Manager Console:

### PM> update-database

We bekijken eerst de code in het te testen project:

#### **Modal class Customer:**

```
Customer.cs ≠ × EntityRepository.cs
                    ▼ Project.API.Models.Cus ▼ 🔑 Id
                                                                        using System.Collections.Generic;
                                                                        Search Solution Explorer (Ctrl+$)
            using System.ComponentModel.DataAnnotations;
                                                                         Solution 'ASPNETCoreUnitTest' (2 of 2 projects)
      4
            using System.ComponentModel.DataAnnotations.Schema;
                                                                           Project.API
            using System.Linq;
            using System.Threading.Tasks;
                                                                              Connected Services
                                                                           Dependencies
           pnamespace Project.API.Models
      8
                                                                              Properties
            {
                                                                               BusinessLogic
                                                                              ▶ c# CustomerBL.cs
                 public class Customer
     10
                                                                                c# MathBL.cs
                     public int Id { get; set; }
                                                                              C# WeatherForecastController.cs
     12
                                                                              DataAccess
     13
                     [Required]
     14
                     [Column(TypeName = "nvarchar(20)")]
                                                                                c# EntityRepository.cs
                                                                                c# IEntityRepository.cs
                     public string FirstName { get; set; }
     15
                                                                             DataRepository
                     [Required]
                                                                              ▶ c# CustomerDbContext.cs
                     [Column(TypeName = "nvarchar(25)")]
     17
                                                                              ▶ c# Customer.cs
     18
                     public string LastName { get; set; }
                                                                              appsettings.json
                     [Required]
     19
                                                                               c# Program.cs
     20
                     [Column(TypeName = "nvarchar(5)")]
                                                                              c# Startup.cs
     21
                     public string CustomerId { get; set; }
                                                                              c# WeatherForecast.cs
                                                                         Project.API.Test
     22
                     public bool IsDeleted { get; set; }
                                                                           Dependencies
     23
                                                                           ▶ C# MathBLTest.cs
     24
     25
     26
```

#### database context class CustomerDbContext:

```
CustomerDbContext.cs + × Customer.cs
                                     EntityRepository.cs
                                                                                           Solution Explorer
                          → 🔩 Project.API.DataRepository.Custor → 🔮 CustomerDbContext(DbContextC → 💠
Project.API
                                                                                           ⊡using Microsoft.EntityFrameworkCore;
                                                                                           Search Solution Explorer (Ctrl+$)
     2
           using Project.API.Models;
                                                                                           Solution 'ASPNETCoreUnitTest' (2 of 2 p
            using System;
                                                                                           using System.Collections.Generic;
            using System.Linq;
                                                                                                Connected Services
           using System.Threading.Tasks;
                                                                                              Dependencies
                                                                                             Properties
          -namespace Project.API.DataRepository

■ BusinessLogic

           {
                                                                                                ▶ c# CustomerBL.cs
                                                                                                ▶ c# MathBL.cs
               public class CustomerDbContext:DbContext
    10
                                                                                              11
                                                                                                ▶ C# WeatherForecastController.cs
    12
                   public CustomerDbContext(DbContextOptions options) : base(options)
                                                                                               DataAccess
    13
                   { }
                                                                                                ▶ c# EntityRepository.cs
                                                                                                ▶ c# IEntityRepository.cs
    14
                   DbSet<Customer> Customers { get; set; }
                                                                                              ■ DataRepository
    15
                                                                                                ▶ C# CustomerDbContext.cs
    16
                                                                                                Models
    17
                                                                                                C# Customer.cs
                                                                                             c# Program.cs
                                                                                             ▶ c# Startup.cs
                                                                                                c# WeatherForecast.cs
                                                                                           ▲ Project.API.Test
                                                                                             Dependencies
                                                                                             ▶ c# MathBLTest.cs
```

### **Data access logica**

De **CustomerDbContext** class wordt aangesproken via een Repository class **EntityRepository** die de business logica bevat:

```
EntityRepository.cs → ×
IEntityRepository.cs
Project.API
                             ▼ Project.API.DataAccess.EntityReposit ▼ 🐾 _dbContext
                                                                                             - ‡
                                                                                                  Search Solution Explorer (Ctrl+$)
          -namespace Project.API.DataAccess
                                                                                                   Project.API
                public class EntityRepository<T>: IEntityRepository<T> where T:class,new()
    10
                                                                                                        Connected Services
    11

    □ Dependencies

                    private CustomerDbContext _dbContext;
                                                                                                     Properties
                    private DbSet<T> _dbSet;
                                                                                                        BusinessLogic
                    public EntityRepository(CustomerDbContext dbContext)
                                                                                                           c# MathBL.cs
    15
                                                                                                        Controllers
    16
                         dbContext = dbContext;
                                                                                                           c# WeatherForecastController.c:
    17
                       _dbSet = _dbContext.Set<T>();
    18
    19
                                                                                                       c# IEntityRepository.cs
    20
                   public virtual void Insert(T entity)
                                                                                                        DataRepository
    21
                                                                                                          c# CustomerDbContext.cs
    22
                       _dbSet.Add(entity);
                                                                                                        Models
    23
                       _dbContext.SaveChanges();
                                                                                                          c# Customer.cs
    24
                                                                                                       appsettings.json
    25
                                                                                                        c# Program.cs
                    public virtual IQueryable<T> GetAllQueryable()
                                                                                                       c# Startup.cs
                                                                                                       c# WeatherForecast.cs
    27
                       return _dbSet;
                                                                                                   ▲ Project.API.Test
    28
    29
                                                                                                     Dependencies
    30
                                                                                                       c# MathBLTest.cs
    31
    32
```

Deze implementeert een interface IEntityRepository:

```
IEntityRepository.cs → X CustomerDbContext.cs
                                                                                                Solution Explorer
Project.API
                              •• Project.API.DataAccess.IEntityRep ▼ ፡ Insert(T entity)
                                                                                           - ‡
                                                                                                 _using System;
                                                                                                 Search Solution Explorer (Ctrl+$)
            using System.Collections.Generic;

    Solution 'ASPNETCoreUnitTest' (2 of 2 p

            using System.Ling;
            using System.Threading.Tasks;
                                                                                                   Project.API
                                                                                                      Connected Services
           -namespace Project.API.DataAccess
                                                                                                    Dependencies
                                                                                                    Properties
                                                                                                      BusinessLogic
                public interface IEntityRepository⟨T> where T : class, new()
                                                                                                        c# CustomerBL.cs
                                                                                                      b c# MathBL.cs
                                                                                                      Controllers
     10
                    void Insert(T entity);
                                                                                                      ▶ c# WeatherForecastController.cs
                     IQueryable<T> GetAllQueryable();
                                                                                                      DataAccess
                                                                                                         c# EntityRepository.cs
     13
                                                                                                      ▶ c# IEntityRepository.cs
                                                                                                      DataRepository
                                                                                                        c# CustomerDbContext.cs
                                                                                                      Models
                                                                                                      ▶ c# Customer.cs
                                                                                                      c# Program.cs
                                                                                                      c# Startup.cs
                                                                                                      c# WeatherForecast.cs
                                                                                                   Project.API.Test

■ Dependencies

                                                                                                       c# MathBLTest.cs
```

# **Business logica**

De class **CustomerBL** bevat de volgende methoden:

- GetActiveCustomer()
- SaveCustomer()

```
IEntityRepository.cs
                                                Project.API

→ __customerRepository
                                            ♦ Project.API.BusinessLogic.CustomerBL
                                                                                                                                    + ‡
                                                                                                                                          public class CustomerBL
                                                                                                                                           Search Solution Explorer (Ctrl+$)
                                                                                                                                           Solution 'ASPNETCoreUnitTest' (2 of 2 p
                                                                                                                                           ▲ ■ Project.API
                    public IEntityRepository<Customer> _customerRepository { get; set; }
                                                                                                                                                Connected Services
                    public CustomerBL(IEntityRepository<Customer> customerRepository)
                                                                                                                                                ■ Dependencies
                                                                                                                                                Properties

BusinessLogic

C= CustomerBL.cs
                        _customerRepository = customerRepository;
    15
16
                     public List<Customer> GetActiveCustomers()
                                                                                                                                                   c= MathBL.cs
    17
18
19
20
21
22
                                                                                                                                                Controllers
                                                                                                                                                   c= WeatherForecastController.cs
                        result = _customerRepository.GetAllQueryable().Where(s => s.IsDeleted == false).ToList(); return result;
                                                                                                                                                DataAccess
                                                                                                                                                D C# EntityRepository.cs
                                                                                                                                                   c= IEntityRepository.cs
                    public bool GetActiveCustomers(Customer customer)
    23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
                                                                                                                                                ☐ DataRepository

▶ c= CustomerDbContext.cs
                        var isAdded = false;
                                                                                                                                                Models
                                                                                                                                                customerRepository.Insert(customer);
                                                                                                                                                c# Program.cs
c# Startup.cs
                         catch (Exception ex)
                                                                                                                                           ▲ ☑ Project.API.Test
                                                                                                                                             Dependencies
                            throw new Exception("GetActiveCustomers",ex);
                                                                                                                                                c# MathBLTest.cs
                         return isAdded;
```

CustomerBL.cs

## Unit test voor class met business logica

Maak een class **CustomerBLTest** in het NUnit test project en implementeer de test methode **GetActiveCustomersTest**()

```
using Moq;
using Project.API.DataAccess;
using Project.API.Models;
using System;
using System.Collections.Generic;
using System.Text;
using NUnit.Framework;
using System.Linq;
using Project.API.BusinessLogic;
namespace Project.API.Tests.BusinessLogicTests
    public class CustomerTestBL
        private Mock<IEntityRepository<Customer>> _customerRepository;
        private List<Customer> _customers;
        [SetUp]
        public void Setup()
            //Set up de mock
            _customerRepository = new Mock<IEntityRepository<Customer>>();
            _customers = new List<Customer>();
             _customers.Add(new Customer() { Id = 1, FirstName = "Cus1", CustomerId =
"COO1", IsDeleted = false });
             customers.Add(new Customer() { Id = 2, FirstName = "Cus2", CustomerId =
"C002", IsDeleted = true });
             customers.Add(new Customer() { Id = 3, FirstName = "Cus3", CustomerId =
"C003", IsDeleted = false });
        }
        [Test]
        public void GetActiveRecordsTest()
            _customerRepository.<mark>Setup</mark>(a =>
a.GetAllQueryable()).Returns(_customers.AsQueryable());
            var customerBL = new CustomerBL( customerRepository.Object);
            var customerList = customerBL.GetActiveCustomers();
            Assert.IsTrue(customerList.Count == 2);
            Assert.IsTrue(customerList.All(s => s.IsDeleted == false));
        }
    }
```

## Arrange/Act/Assert wordt het **AAA** of 3A design pattern genoemd

## **Uitleg over Mock en test asserts**

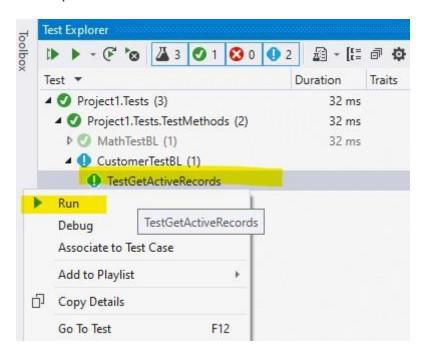
- Arrange: De nodige voorbereidingen (insitialisaties en mock training gebeuren voor de dependency-objecten): Een Mock instantie customerRepository wordt aangemaakt voor IEntityRepository < Customer > in de Setup method en statische test data wordt gedefinieerd voor customers List.
  - In de GetActiveRecordsTest method wordt de Mock wordt getrained dmv van de Setup methode van de Mock customerRepository. Wanneer de methode GetAllQueryable() wordt aangeroepen, zal de statische test data gedefineerd voor customers worden teruggegeven door de Mock.
- 2. **Act**: De te testen methode wordt hier effectief aangeroepen. Hier is dit **GetActiveCustomers()**; **deze** wordt aangeroepen en het resultaat wordt in de variabele customerList gezet.
- 3. **Assert**: via Assert statements worden de verwachte resultaten vergeleken met de teruggegeven waarden van de te testen methode. In dit voorbeeld is het te verwachten resultaat dat er 2 elementen in de customerList staan en enkel Customer elementen waarvan IsDeleted==false worden teruggegeven door de methode GetActiveCustomers().

```
public class CustomerTestBL
     private Mock<IEntityRepository<Customer>> _customerRepository;
     private List<Customer> customers;
     [SetUp]
    O references
public void Setup()
          //Set up de mock
          _customerRepository = new Mock<IEntityRepository<Customer>>();
         _customers = new List<Customer>();
         _customers.Add(new Customer() { Id = 1, FirstName = "Cus1", CustomerId = "C001", IsDeleted = false });
_customers.Add(new Customer() { Id = 2, FirstName = "Cus2", CustomerId = "C002", IsDeleted = true });
_customers.Add(new Customer() { Id = 3, FirstName = "Cus3", CustomerId = "C003", IsDeleted = false });
     [Test]
     public void GetActiveRecordsTest()
          _customerRepository.Setup(a => a.GetAllQueryable()).Returns(_customers.AsQueryable());
          //Act
          var customerBL = new CustomerBL(_customerRepository.Object);
          var customerList = customerBL.GetActiveCustomers();
          //Assert
          Assert.IsTrue(customerList.Count == 2);
          Assert.IsTrue(customerList.All(s => s.IsDeleted == false));
```

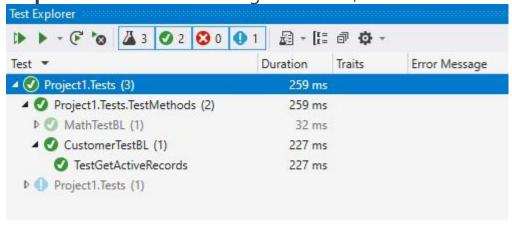
#### **Run test**

**Stap 1**: Selecteer **View -> Test Explorer**. Deze geeft de lijst van test case methoden.

**Stap 2**: Rechtsklik op de test methode **TestGetActiveCustomer()** en klik op **Run**.



Stap 3: Als de test methode groen kleurt, is de test succesvol.



# Referenties

https://www.syncfusion.com/blogs/post/how-to-integrate-unittesting-with-asp-net-core-3-1.aspx

# Moq voorbeeld