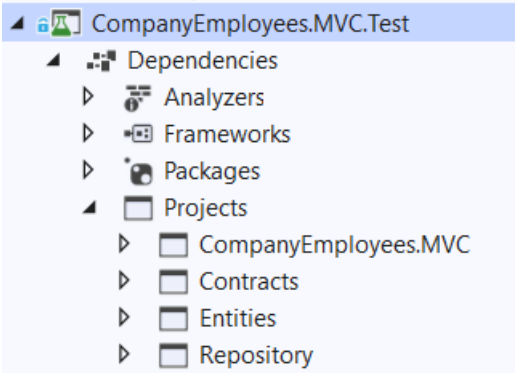
Repository en Controller Tests met Nunit en ASP.NET Core 3.1

Start project :

https://github.com/CSharpSyntraWest/CompaniesEmployees5

1. Opzetten van Nunit Test project

1. **Kloon het startproject en voeg aan dezelfde solution een nieuw NUnit 3 Test project aan met naam CompanyEmployees.MVC.Test**
2. **Voeg project referenties toe naar CompanyEmployees.MVC, Contracts, Entities en Repository**



3. **Voeg de volgende hulpklasse TestRepositoryContextFactory toe aan het Test project :**

```
namespace CompanyEmployees.MVC.Test
{
    public class TestRepositoryContextFactory : IDisposable
    {
        private DbConnection _connection;

        private DbContextOptions<RepositoryContext> CreateOptions()
        {
            return new DbContextOptionsBuilder<RepositoryContext>()
                .UseSqlite(_connection).Options;
        }

        public RepositoryContext CreateContext()
        {
            if (_connection == null)
            {
                _connection = new SqliteConnection("DataSource=:memory:");
                _connection.Open();

                var options = CreateOptions();
                using (var context = new RepositoryContext(options))
                {
                    context.Database.EnsureCreated();
                }
            }

            return new RepositoryContext(CreateOptions());
        }

        public void Dispose()
        {
            if (_connection != null)
            {
                _connection.Dispose();
                _connection = null;
            }
        }
    }
}
```

4. Voeg de volgende Test class toe om de TestRepositoryContextFactory te testen :

```
    public class TestRepositoryContextFactoryTests
    {
        [Test]
        public void Add_ShouldAddNewCompanyAndNewEmployee()
        {
            using (var factory = new TestRepositoryContextFactory())
            {
                Guid testEmployeeId = Guid.NewGuid();
                Guid testCompanyId = Guid.NewGuid();
                Company testCompany = new Company()
                {
                    Id = testCompanyId,
                    Name = "Test bedrijf",
                    Country = "Test land",
                    Description = "Test beschrijving",
                    Size = CompanySize.Small,
                    LaunchDate = DateTime.Today,
                    Address = "Test adres"
                };

                Employee testEmployee = new Employee()
                {
                    Id = testEmployeeId,
                    CompanyId = testCompanyId,
                    Name = "Jos",
                    Description = "Test employee",
                    Age = 45,
                    Gender = GeslachtType.Man,
                    Position = "Developer"
                };
                // Get a context
                using (var context = factory.CreateContext())
                {
                    context.Companies.Add(testCompany);
                    context.Employees.Add(testEmployee);
                    context.SaveChanges();
                }

                // Get another context using the same connection
                using (var context = factory.CreateContext())
                {
                    var count = context.Employees.Count();
                    Assert.AreEqual(4, count);

                    var emp = context.Employees.FirstOrDefault(e => e.Id == testEmployeeId);
                    Assert.IsNotNull(emp);
                }
            }
        }

    }
}
```

5. **Run de test methode en controleer of deze groen kleurt**
6. **Voeg de hulpclass SeedData toe die methoden heeft met test gegevens:**

```
public class SeedTestData
{
    public static void PopulateTestData(RepositoryContext dbContext)
    {
        Employee testEmployee = new Employee()
        {
            Id = Guid.NewGuid(),
            CompanyId = Guid.NewGuid(),
            Name = "Jos"
        };
        dbContext.Employees.Add(testEmployee);
        dbContext.SaveChanges();
    }
    public static IEnumerable<Employee> GetTestEmployees()
    {
        return new List<Employee>()
            {
                new Employee()
                {
                    Id = new Guid("80abbca8-664d-4b20-b5de-024705497d4a"),
                    CompanyId = Guid.NewGuid(),
                    Name = "John",
                    Position = "Developer"
                },
                new Employee()
                {
                    Id = new Guid("86dba8c0-d178-41e7-938c-ed49778fb52a"),
                    CompanyId = Guid.NewGuid(),
                    Position = "Analyst",
                    Name = "Doe"
                }
            };
    }
    public static Employee GetTestEmployee()
    {
        return SeedTestData.GetTestEmployees().FirstOrDefault();
    }

    public static IEnumerable<Company> GetTestCompanies()
    {
        return new List<Company>()
        {
            new Company
            {
                Id = new Guid("c9d4c053-49b6-410c-bc78-2d54a9991870"),
                Name = "IT_Solutions Ltd",
                Address = "583 Wall Dr. Gwynn Oak, MD 21207",
                Country = "USA"
            },
             new Company
             {
                 Id = new Guid("3d490a70-94ce-4d15-9494-5248280c2ce3"),
                 Name = "Admin_Solutions Ltd",
                 Address = "312 Forest Avenue, BF 923",
                 Country = "USA"
             }
        };
    }
}
```

2. Testen van Repository classes

1. Voeg de class RepositoryManagerTests toe met de volgende Test methoden :

```
public class RepositoryManagerTests
{

    #region EmployeesRepoTests//EMPLOYEES TESTS

    [Test]
    0 references
    public void GetAllEmployees_ShouldReturnAllEmployeesFromContext()
    {
        using (var factory = new TestRepositoryContextFactory())
        {
            using (var context = factory.CreateContext())
            {
                var countEmployeesInDb = context.Employees.Count();

                var repository = new RepositoryManager(context);
                var emp = repository.Employee.GetAllEmployees(false);
                Assert.IsNotNull(emp);
                Assert.AreEqual(countEmployeesInDb, emp.Count());
            }
        }
    }
```

```
    [Test]
    0 references
    public void GetEmployee_ShouldReturnEmployee()
    {
        //Arrange
        Guid testEmployeeId;
        using (var factory = new TestRepositoryContextFactory())
        {
            using (var context = factory.CreateContext())
            {
                var testEmployee = context.Employees.FirstOrDefault();
                testEmployeeId = testEmployee.Id;
                var repository = new RepositoryManager(context);
                //Act
                var empl = repository.Employee.GetEmployee(testEmployeeId, false);
                Assert.IsNotNull(empl);
                Assert.AreEqual(testEmployeeId, empl.Id);
            }
        }
    }
```

```
[Test]
0 references
public void CreateEmployeeForExistingCompany_ShouldAddNewEmployeeToContextForCompany()
{
    using (var factory = new TestRepositoryContextFactory())
    {
        //Arrange
        int count = 0;

        Company testCompany = null;
        Guid testEmployeeId = Guid.NewGuid();
        Employee testEmployee = new Employee()
        {
            Id = testEmployeeId,
            Name = "Jos",
            Description = "Test employee",
            Age = 45,
            Gender = GeslachtType.Man,
            Position = "Developer"
        };
        using (var context = factory.CreateContext())
        {
            count = context.Employees.Count();
            testCompany = context.Companies.FirstOrDefault();
            testEmployee.CompanyId = testCompany.Id;
            var repository = new RepositoryManager(context);
            //Act
            repository.Employee.CreateEmployeeForCompany(testCompany.Id, testEmployee);
            repository.Save();
        }
        //Assert
        using (var context = factory.CreateContext())
        {
            Assert.AreEqual(count + 1, context.Employees.Count());
            var addedEmployee = context.Employees.Find(testEmployeeId);
            Assert.IsNotNull(addedEmployee);
            Assert.AreEqual(testEmployeeId, addedEmployee.Id);
        }
    }
}
```

```
[Test]
0 references
public void SaveChangesGetEmployeeTrackChangesTrue_ShouldChangeEmployeeInContext()
{
    using (var factory = new TestRepositoryContextFactory())
    {
        //Arrange
        Guid testCompanyId;
        Guid testEmployeeId;
        Employee testEmployee;

        using (var context = factory.CreateContext())
        {
            var repository = new RepositoryManager(context);
            var firstCompany = context.Companies.FirstOrDefault();
            testCompanyId = firstCompany.Id;
            var firstEmployee = context.Employees.FirstOrDefault();
            testEmployeeId = firstEmployee.Id;
            //Act
            testEmployee = repository.Employee.GetEmployee(testEmployeeId, true);

            testEmployee.Name = "gewijzigde naam Joke";
            testEmployee.Age = 18;
            testEmployee.CompanyId =testCompanyId;
            testEmployee.Description = "gewijzigde beschrijving";
            testEmployee.Gender = GeslachtType.Vrouw;
            testEmployee.Position = "gewijzigde positie";

            repository.Save();
        }
        //Assert
        using (var context = factory.CreateContext())
        {
            var changedEmployee = context.Employees.FirstOrDefault(e => e.Id == testEmployeeId);
            Assert.IsNotNull(changedEmployee);
            Assert.AreEqual(testEmployee.Id, changedEmployee.Id);
            Assert.AreEqual(testEmployee.Name, changedEmployee.Name);
            Assert.AreEqual(testEmployee.Age, changedEmployee.Age);
            Assert.AreEqual(testEmployee.CompanyId, changedEmployee.CompanyId);
            Assert.AreEqual(testEmployee.Description, changedEmployee.Description);
            Assert.AreEqual(testEmployee.Gender, changedEmployee.Gender);
            Assert.AreEqual(testEmployee.Position, changedEmployee.Position);
        }
    }
}
```

```
[Test]
0 references
public void DeleteEmployee_ShouldRemoveEmployeeFromContext()
{
    using (var factory = new TestRepositoryContextFactory())
    {
        //Arrange
        Guid testEmployeeId;
        int count;
        using (var context = factory.CreateContext())
        {
            count = context.Employees.Count();
            var repository = new RepositoryManager(context);
            var firstEmployee = context.Employees.FirstOrDefault();
            testEmployeeId = firstEmployee.Id;
            //Act
            repository.Employee.DeleteEmployee(firstEmployee);

            repository.Save();
        }
        //Assert
        using (var context = factory.CreateContext())
        {
            Assert.AreEqual(count - 1, context.Employees.Count());
            Assert.IsFalse(context.Employees.Where(c => c.Id == testEmployeeId).Any());
        }
    }
}
```

COMPANIES TESTEN:

```
    #endregion //EMPLOYEES TESTS
    #region CompaniesRepoTests//COMPANIES TESTS
    [Test]
    0 references
    public void GetAllCcompanies_ShouldReturnAllCompaniesFromContext()...
    [Test]
    0 references
    public void GetCompany_ShouldReturnCompany()...
    [Test]
    0 references
    public void CreateCompany_ShouldAddNewCompanyToContext()...
    [Test]
    0 references
    public void SaveChangesGetCompanyTrackChangesTrue_ShouldChangeCompanyInContext()...
    [Test]
    0 references
    public void DeleteCompany_ShouldRemoveCompanyFromContext()...
    #endregion //COMPANIES TESTS
}
```

Oefening: Vul de bovenstaande Test methoden aan voor de COMPANIES TESTS

3. Testen van Controller classes

3.1 Testen van EmployeeManagerController

1. Voeg de test class EmployeeManagerControllerTests toe:

```
public class EmployeeManagerControllerTests
 {
     private Mock<IRepositoryManager> mockRepo;
     [SetUp]
     public void Initialize()
     {
         mockRepo = new Mock<IRepositoryManager>();
         mockRepo.Setup(repo => repo.Company.GetAllCompanies(false))
            .Returns(SeedTestData.GetTestCompanies());
     }

     [Test]
     public void Index_ReturnsAViewResult_WithAListOfEmployees()
     {
         // Arrange

         mockRepo.Setup(repo => repo.Employee.GetAllEmployees(false))
             .Returns(SeedTestData.GetTestEmployees());
         var controller = new EmployeeManagerController(mockRepo.Object);

         // Act
         var result = controller.Index();

         // Assert
         Assert.IsInstanceOf<ViewResult>(result);
         var viewResult = result as ViewResult;
         Assert.IsAssignableFrom<List<Employee>>(
              viewResult.ViewData.Model);
         var model = viewResult.ViewData.Model as List<Employee>;
         Assert.AreEqual(2, model.Count());
     }


     [Test]
     public void Insert_InsertsEmployeeAndReturnsAViewResult_WithAnEmployee()
     {
         // Arrange

         mockRepo.Setup(repo => repo.Employee.CreateEmployeeForCompany(It.IsAny<Guid>(), It.IsAny<Employee>()))
             .Verifiable();
         var controller = new EmployeeManagerController(mockRepo.Object);
         var newEmployee = SeedTestData.GetTestEmployee();

         // Act
         var result = controller.Insert(newEmployee);

         // Assert
         Assert.IsInstanceOf<ViewResult>(result);
         var viewResult = result as ViewResult;
         Assert.AreEqual(viewResult.Model, newEmployee);
         mockRepo.Verify();
     }
     [Test]
     public void Delete_DeletesEmployeeAndReturnsRedirectToActionResult()
     {
         // Arrange
         var httpContext = new DefaultHttpContext();
         var tempData = new TempDataDictionary(httpContext, Mock.Of<ITempDataProvider>());
         tempData["Message"] = "Werknemer verwijderd";
         var testDeleteEmployee = SeedTestData.GetTestEmployee();
         mockRepo.Setup(repo => repo.Employee.GetEmployee(testDeleteEmployee.Id, false))
             .Returns(testDeleteEmployee);

         var controller = new EmployeeManagerController(mockRepo.Object) { TempData = tempData};

         // Act
         var result = controller.Delete(testDeleteEmployee.Id);

         // Assert
         Assert.IsInstanceOf<RedirectToActionResult>(result);
         var redirectToActionResult = result as RedirectToActionResult;
         Assert.Null(redirectToActionResult.ControllerName);
         Assert.AreEqual("Index", redirectToActionResult.ActionName);
     }
     [Test]
     public void Employee_Details_ReturnsEmployee()
     {
```

```
            // arrange
            Guid testEmployeeId = new Guid("80abbca8-664d-4b20-b5de-024705497d4a");
            var testEmployees = SeedTestData.GetTestEmployees();
            var firstEmpl = SeedTestData.GetTestEmployee();
            mockRepo.Setup(x => x.Employee.GetEmployee(testEmployeeId, It.IsAny<bool>())).Returns(firstEmpl);

            var controller = new EmployeeManagerController(mockRepo.Object);

            // act
            var result = controller.Details(testEmployeeId);
            Assert.IsInstanceOf<ViewResult>(result);
            var viewResult = result as ViewResult;
            Assert.That(viewResult.Model, Is.TypeOf<Employee>());
            var employee = viewResult.Model as Employee;
            // assert
            Assert.AreEqual(testEmployeeId, employee.Id);
        }
    }
```

3.2 Testen van CompanyManagerController : oefening: maak de CompanyManagerControllerTests class aan en voeg de nodig Test methoden toe