# Exercises: Unit Testing

Problems for exercises and homework for the [“C# MVC Frameworks - ASP.NET Core” course @ SoftUni](https://softuni.bg/courses/asp-net-mvc).

## News REST Service

Design and implement a **REST service** based on **ASP.NET Core Web API**, **Entity Framework Core** and **SQL Server** to hold news.

**News** items should have **title**, **content** and **publish date**.

Your service should implement the following endpoints:

* GET /api/news – returns all news ordered by publish date (from the latest) as JSON array.
* POST /api/news – creates a news item by given **title**, **content** and **publish date** (given in the request body).
* PUT /api/news/{id} – updates an existing news item by given **id** (given in the request URL) and **title**, **content** and **publish date** (given in the request body).
* DELETE /api/news/{id} – deletes an existing news item by given **id**.

Structure your application into layers:

* **Data models layer** – to hold the entity classes.
* **Repository layer** – to hold your EF Core data context and repositories (**if you have such**).
* **Web API services** – ASP.NET Core Web API application that holds your REST services.

## Unit Testing the Web API Controllers

Write **unit tests** to test your **Web API controllers**.Use **in-memory** **Database** to test the **DbContext**. Test all **service endpoints**. Your tests should run without a database. Test at least the following scenarios:

1. List all news items 🡪 200 (OK) + returns the news items correctly.
2. Create a news item with correct data 🡪 201 (Created) + creates a news item + returns the created item.
3. Create a news item with incorrect data 🡪 400 (Bad Request).
4. Modify existing news item with correct data 🡪 200 (OK) + modifies the news item.
5. Modify existing news item with incorrect data 🡪 400 (Bad Request).
6. Modify non-existing news item 🡪 400 (Bad Request).
7. Delete existing news item 🡪 200 (OK) + deletes the item.
8. Delete non-existing news item 🡪 400 (Bad Request).