# QA Back-End Technologies Basics

# Exam Prep II

## Products

You are given a .NET solution that implements a console application for simulating a product managing application. Your task is to create and implement **tests** using the **Integration** **Tests** **project** and **NUnit**.

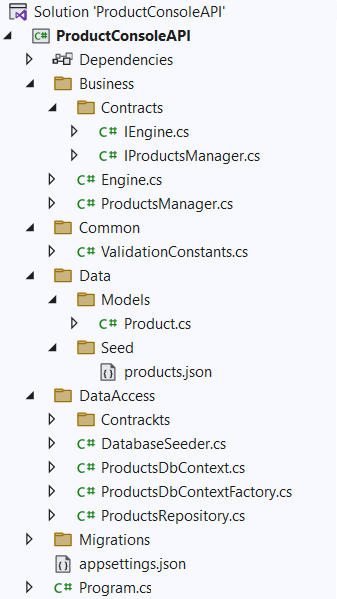
### Introduction

**ProductConsoleAPI** is a console-based application built using the .NET Framework and the app manages a collection of Products.

It allows users to perform various operations, such as searching, adding, reading, updating and deleting products from database.

This application is designed to showcase **essential software development concepts and practices**.

### Architecture Overview



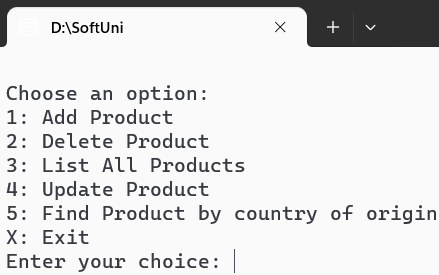
### Data Validation Overview

The **Common** folder in the application contains shared constants that can be used across multiple layers of the application. In particular, the **ValidationConstants** class provides a centralized place to define the rules and constraints related to data validation for the domain models, particularly products in this context.

Don't forget to use the data in this class while writing your tests.

### Functionality Overview

When the application is executed, it presents a menu with options that the user can select from, as shown below:

****

**Users interact with the application** by entering **commands** or **data** into the console. The application processes these inputs and performs the corresponding actions.

For example, when listing all the products, the application, through **the ProductsManager**, calls the **GetAllProductsAsync method** on the **ProductsRepository**. This **method is responsible for retrieving all products records from the database**.

This retrieval is performed **asynchronously**, ensuring that the **application remains responsive**. Once the data is fetched, the **method iterates over the collection of Product objects and prints their details** to the console.

#### Add Product

The application promts the user to enter data for the details below:

* Code
* Name
* Quantity
* Price
* Country of origin
* Description

**Input data** is **validated** according to the **rules**, defined in the **business** **layer** and the **model's** **data** **annotations**.

#### Delete Product

Deletes the input product from the database.

#### List All Products

Retrieves a list with all products, present in the database.

#### Update Product

Updates a product's information.

#### Find Product by Country of Origin

Finds a product in the database by the country of origin, that the user types in the console.

## Integration Tests

Your task is to **implement the test methods in the provided test project**. These methods are placeholders that represent the **various scenarios** you will test within the application.

Navigate to **ProductConsoleAPI.IntergrationTests.NUnit** and use the **IntegrationTests.cs** to start writing your tests.

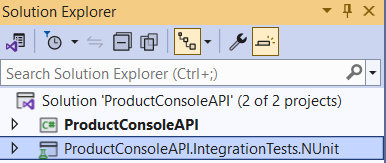
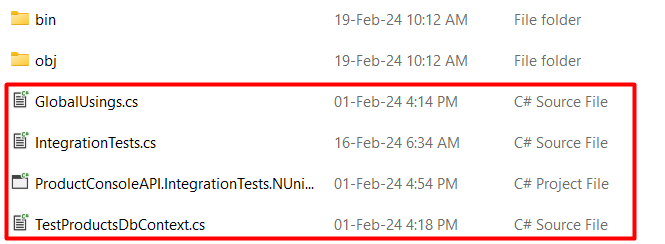
The whole solution is already configured with the necessary packages and references and works seamlessly.

#### Additional Notes

* The test should be isolated, meaning it shouldn't be affected by or affect other tests.
* Integration tests often involve asynchronous operations, especially when dealing with database operations. Using await ensures that these asynchronous operations complete before assertions are made.
* While the data used in the test is arbitrary, it should resemble realistic data that the application might handle.

Good luck! 😊

#### How to submit your work

* From the Solution Explorer in Visual Studio, right click on your **IntegrationTest Project and choose Open Folder in File Explorer**
* **Choose Open Folder in File Explorer**
* Create a **.zip archive** with the following files:  
  

**NB!:** ZIP archive (**NOT rar**, or any other)! **Don't include** **bin and obj folders**.

* Attach your archive in the Judge system.
* Don't mind that it will give you "Compile Time Error". Third task of your exam will be checked manualy.