Exercise Guidelines

Please read carefully all the bullet points and ask questions if they are not clear enough.

All code exercises are expected to comply to this guideline.

If any the code delivered does not meet the guideline a revision/correction will be requested.

Deliverable artifact

* 1. Use git (share the repo with us, use git private repos or bitbucket)
  2. Use .gitignore to ignore bin/obj/npm\_packages/nuget folders.

Project structure

* The project should compile and run with no other changes than the ones in the Web.config / Appsetting.json files
* Use Entity Framework (Code First) to generate the database.
* The project should not depend on external webservices or any service that is not part of the source code delivered.
* Create your solution as an “empty project” or remove any boilerplate code or files autogenerated by Visual Studio that are not part of the exercise.

Application Requirements

Create a stock management web tool for “Toys and Games” store.

* Should be able to list the available products in a grid
* Should be able to Create, Update, Delete products
* Should provide a simple Form when creating or updating products
* Should provide user confirmation for product Deletion
* Should display product images

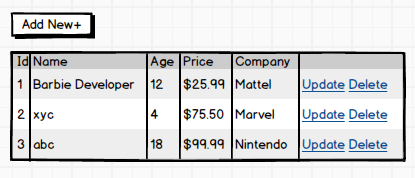
Product data dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Optional** | **Constrains** |
| Id | Int | No | Unique |
| Name | string | No | Max length 50 |
| Description | string | Yes | Max length 100 |
| AgeRestriction | int | Yes | 0 to 100 |
| CompanyId | Int | No | FK\_Company |
| Price | decimal | No | 1$ to $1000 |

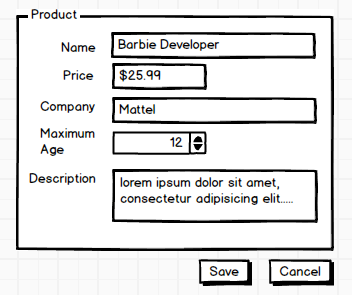
UI mocks

Feel free to improve or add any extra elements that help usability or user experience, use these mocks as reference

* Product Listing



* Input Form



**Architecture constrains**

1. Data Persistence
2. Use Entity Framework Code First
3. Use seed data🡪 EF HasData() Method
4. Server Side
   1. Use C# with .Net Core 6
   2. Use ASP.NET Core
   3. Communicate the UI and the backend with a REST API using Web API
   4. The Angular and WebAPI code can share the same project or live as separate projects
   5. Favor IIS express or Kestrel over IIS webapp or website
   6. Model/Entity validation// FluentValidations
   7. Use Dependency Injection
   8. Add a Unit test project with at least 5-unit tests using either xUnit or MS Test
      1. Use Mocking Framework
   9. Add at least 3 Integration Tests
5. Client Side
6. Use framework Angular 12 or Above
7. Use simple CSS or a CSS framework like Bootstrap to provide a simple UI experience
8. Input form validation
9. Use Reactive Forms
10. Use Observables
11. Use NgRx store and effects (Optional)
12. Use granular components
13. Use SASS or LESS
14. Add Karma / Jasmine Unit test
15. Use Angular Material or Material Design