rExercise 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.

Anything marked as OPTIONAL will provide extra points.

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 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
* (OPTIONAL) display product images

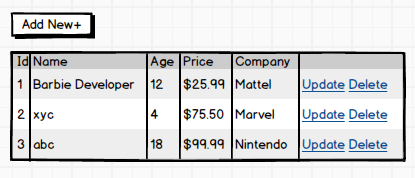
Product data dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |
| Company | string | No | Max length 50 |
| Price | decimal | No | 1$ to $1000 |

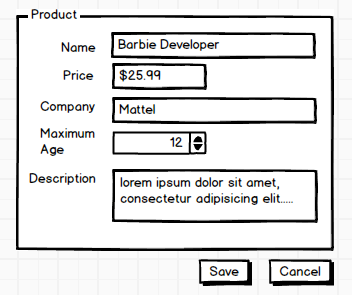
UI mocks

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 simple file storage (EF In memory database, JSON, Redis, etc)
3. Implement IRepository pattern
4. (OPTIONAL) Use seed data
5. (OPTIONAL) Use Code First
6. Server Side
   1. Use C# with .Net Framework 4.5 (or latest)
   2. Use ASP.NET MVC 5 (or latest)
   3. Communicate the UI and the backend with a REST API using Web API 2 (or latest)
   4. The MVC 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
   7. Use Dependency Injection
   8. (OPTIONAL)Add a Unit test project with at least 5 unit tests using either Xunit, MSTest
   9. (OPTIONAL) Use mocking framework
   10. (OPTIONAL) User .Net Core 2.0 and ASP.Net Core 2.0
7. Client Side
8. Use framework Angular 4 or Above
9. Use simple CSS or a CSS framework like Bootstrap to provide a simple UI experience
10. Input form validation
11. (OPTIONAL) Use granular components
12. (OPTIONAL) Use SASS or LESS
13. (OPTIONAL) Use responsive design
14. (OPTIONAL) Add JS unit test