CHALLENGE

EXAGO

Objective

- Web API project - .NET Core 2.x.
- Create a controller with the HTTP verbs – asynchronously.
- Two entities: Category (id, name) and Product (id, name, price, category).
- Should be used SQL server and Entity framework core.
- At least, apply 2 design patterns.
- Should be made available on GIT (we only need the project URL).
- If needed, documentation with the main justifications.

Summary

I divided the application into 5 layers, using the CQRS pattern to separate responsibilities and generate a low coupling.

Technologies

- ASP NET Core 2.2
- Entity Framework Core.
- FluentValidation.
- MediatR.
- AutoMapper.
- Swagger.
- HelthCheck.
- Sql Server

Patterns and architecture

- DDD (Domain Driven Design)
  https://en.wikipedia.org/wiki/Domain-driven_design

- Bounded Context

- CQRS (Command Query Responsibility Segregation)
  https://cqrs.files.wordpress.com/2010/11/cqrs_documents.pdf
  https://martinfowler.com/bliki/CQRS.html

- Event Sourcing

- Layered architecture

Running Application

- Clone the Project ong Git hub.

  Project link: https://github.com/Juliannotcg/ChallengeZeroToOneSearch

- Create the database in SQL Server database.

  Script: `CREATE DATABASE ChallengeZero.`

- Change the Connection String in the appsettings.json file in the Challenge.Api adn Challenge.Data projects.

  Enter the server name, database name, and password, if any.

- After that, just open the Package Manage Console from Visual Studio or through some terminal and run the command.

  Command: update-database

  If the error happens: `More than one DbContext was found. Specify which one to use. Use the '-Context' parameter for PowerShell commands and the '--context' parameter for dotnet commands.`

  To solve just run the command informing the contexto.
  Command: `update-database -Context ContextEntity`

- After that, excite the project, by default it will open on the Swagger page.

  ### Swagger
  Likn: https://localhost:44388/swagger/index.html
  Swagger is a very easy to use and intuitive API documenter, this is the swagger documentation link https://swagger.io.

  ### HelthCheck
  Likn: https://localhost:44388/healthchecks-ui
  HelthCheck will monitor the health of our application, it is monitoring the API and the database.
  documentation link https://github.com/Xabaril/AspNetCore.Diagnostics.HealthChecks

Front-end Project

I did a small REACT project to consume the API project.

To run the project just run the command NPM INSTALL, to install the packages.

Once complete, simply run the NPM RUN START command.

Technologies

- React.
- Redux.
- Material UI
- JavaScript ES6.