Ecommerce API

In your first ASP.NET Core Web API project you had a first taste of creating, configuring and exposing endpoints and creating a client to consume them. We will now expand on this knowledge with requirements that are aligned with the demands of enterprise development. So far we have worked predominantly in personal apps. This time we will dive into a common business case in enterprise: an ecommerce API that supports a retail business..

The business case itself will be fairly loose, you can create an ecommerce for whatever type of product you want, which will help you exercise your creativity and focus on the technical specifications.

- ✓ Your project needs to be an ASP.NET Core Web API, with Entity Framework and your choice between SQL Server and Sqlite.
- ✓ Your api needs to use Dependency Injection.
- ✓ You should have at least three tables: Products, Categories and Sales.
- ✓ Products and Sales need to have a many-to-many relationship, meaning products can have multiple sales, and sales can have multiple products.
- ✓ Products need to have a price. Multiple products can be sold in the same sale.
- ✓ You need to provide a Postman Collection with all possible requests for your API. It's a json file that needs to be included in your PR.

Logout

- ✓ You don't need to create an UI to consume your API.
- ✓ Your GetProducts and GetSales endpoints need to have pagination capabilities.
- ✓ In retail it's good practice to prevent deletion of records. Feel free to add soft-deletes.
- ✓ You shouldn't update products prices. What would happen if you made a sale and later updated the price of that product?

Resources

Here are a few resources that might be helpful.

- Many to Many tutorial Microsoft
- Dependency Injection Tutorial The C# Academy
- Dependency Injection Tutorial Microsoft
- Pagination Tutorial C# Corner
- Rest APIs Article

Tips

- ✓ Seed data from the beginning so you don't need to create records manually.
- ✓ This project can get overwhelming very quickly. Use a step-by-step approach, creating one end-point at a time and making sure everything works before moving to the next one.
- ✓ Start using Postman from the start to tests your requests. It will save you a lot of time.
- ✓ Think about the business case and what endpoints it will actually. Not all tables need all CRUD operations. For example, you might not need to delete or update products, but you will need to create them and read them.

Challenges

- ✓ Add filtering and sorting capabilities to your endpoints

- ✓ Create a Console UI to consume your Web API.
- ✓ After completing the project, read the **Rest APIs Article** listed in the resources and check what could be missing. Pay particular attention to status codes and endpoints signatures.

Review Repository

https://github.com/TheCSharpAcademy/CodeReviews.Console.EcommerceAp