

# Day 1

? Day 1 ? API Setup, Database, and Product CRUD with Pagination

## Objective

Lay the foundational structure of the application by creating a minimal API, configuring database connectivity using Docker, and implementing basic CRUD functionality for product management. Introduce pagination on the read endpoint.

## Tasks

### 1. Set Up a Minimal Web API

- ~~Create a new ASP.NET Core Web API project using the Minimal API approach.~~
- ~~Ensure it targets .NET 8 or the latest stable version.~~
- Configure the project to use dependency injection and logging services.
- ~~Make sure Swagger (OpenAPI) is enabled for API testing and documentation.~~

### 2. Dockerize and Set Up PostgreSQL

- ~~Use the provided docker-compose.yml to spin up a PostgreSQL instance as your application's database.~~
- Ensure the application connects to this database using the following environment variables:
  - ~~POSTGRES\_DB=prac-day~~
  - ~~POSTGRES\_USER=postgres~~
  - ~~POSTGRES\_PASSWORD=Password1~~
- Validate the connection by ensuring the application can apply migrations and query the database.

### 3. Implement Product CRUD Functionality

# Day 1

- Create a Product entity with the following fields:
  - Id (GUID)
  - Name (string)
  - Description (string)
  - Category (use a Smart Enum for this field)
  - Price (decimal, in Rands)
  - StockQuantity (int)
- ~~Add endpoints to perform the following operations:~~
  - Create a new product
  - ~~Get a product by ID~~
  - ~~Update an existing product~~
  - ~~Delete a product~~
  - ~~List products with optional filtering~~

## ~~4. Implement Paging on Product Retrieval~~

- ~~When fetching a list of products, support query parameters like:~~
  - ~~pageNumber~~
  - ~~pageSize~~
- ~~Return pagination metadata (e.g., TotalCount, PageSize, CurrentPage, TotalPages) in the response.~~
- ~~Ensure the paging logic is efficient and uses server-side skip/take.~~

## Notes

- Store the data in PostgreSQL and use Entity Framework Core to manage your models and migrations.

## Day 1

- The Smart Enum should represent different product categories like: Electronics, Books, Clothing, etc.
- Ensure proper model validation on the API endpoints.