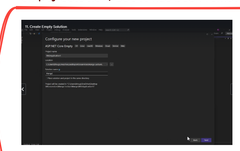
Section 2 - Coupons API (Part1)

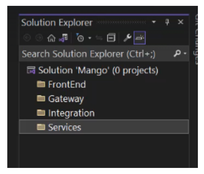
1. Setting Up the Project

Creating a New Project

- Opening Visual Studio
  - Open Visual Studio to start a new project.
- Selecting Project Type
  - Since we are building a microservices architecture, we need to create individual projects.
  - Empty Solution
    - Previously, there was an option for an empty solution, but it has been removed.
    - 
    - Create an ASP.NET Core empty project as a workaround.
- Project Configuration
  - Project Name and Solution Name
    - Project name is not important, but the solution name should be "mango."
    - Framework: .NET 8.
    - Do not use top-level statements.
  - Creating the Project
    - Hit the create button to generate the project.
    - Delete the temporary project to have an empty solution.

2. Organizing the Solution

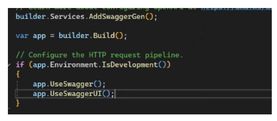
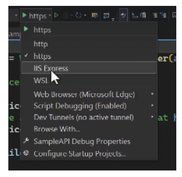
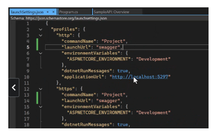
Folder Structure



- Front-End Folder
  - Create a folder for the front-end application, which will be the MVC web application calling the microservices.
- Gateway Folder
  - Create a folder for the gateway, which will be used for advanced concepts with Gateway later in the course.
- Integration Folder
  - Create a folder for integration, which will handle messaging and related functionalities.
- Services Folder
  - Create a folder for services, where all microservices or API endpoints will be developed.

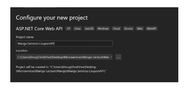
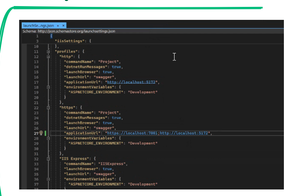
3.Swashbucklel

Installing Swashbuckle

- Creating a New API Project
  - Create a new ASP.NET Core API project named "sample API."
  - 
  - Use .NET 9, enable OpenAPI support, and use controllers.
- 
  - Install Swashbuckle ASP.NET Core via NuGet packages.
- 
  - Open Program.cs and remove the default OpenAPI configuration.
- Configuring Swagger
  - 
  - Add builder.Services.AddSwaggerGen() to the service container.
  - Add app.UseSwagger() and app.UseSwaggerUI() to the pipeline for development.
- Running the Application
  - 
  - Change the launch settings to use IIS Express.
  - 
  - Modify launchSettings.json to set the default path to Swagger.
  - 
  - Change in IISExpress
  - Run the application to ensure Swagger documentation loads by default.

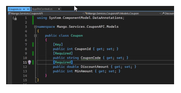
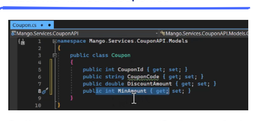
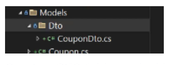
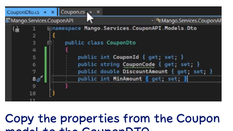
4. Creating the Coupon API

Project Setup

- Creating the Coupon API Project
  - 
  - Create a new ASP.NET Core Web API project named "mango.services.couponAPI."
  - Use .NET 8 and controllers.
- Configuring the Project
  - Run the application to ensure it is set up correctly.
  - 
  - Update the port number to 7001 in launchSettings.json.

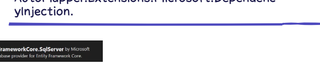
5. Building the Coupon Model

Creating the Model

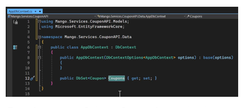
- Coupon Model
  - 
  - Create a new folder for models and add a class file named "Coupon.cs."
  - 
  - Define properties: CouponID, CouponCode, DiscountAmount, and MinimumAmount.
- DTO (Data Transfer Object)
  - 
  - Create a DTO folder and add a class file named "CouponDTO.cs."
  - 
  - Copy the properties from the Coupon model to the CouponDTO.

6. Setting Up the Database

Installing NuGet Packages

- AutoMapper
  - 
  - Install AutoMapper and AutoMapper.Extensions.Microsoft.Dependencyinjection.
- Entity Framework Core
  -
  - Install Microsoft.EntityFrameworkCore.SqlServer.
  -
  - Install Microsoft.EntityFrameworkCore.Tools.
  - Install Microsoft.AspNetCore.Authentication.JwtBearer.

Configuring the Database

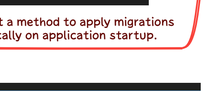
- 
  - Create a new folder for data and add a class file named "AppDbContext.cs."
- 
  - Implement the DbContext class and configure the connection string in appsettings.json.
- 
  - Add the connection string to the service container in Program.cs.

7. Creating the Database and Tables

Adding Migrations

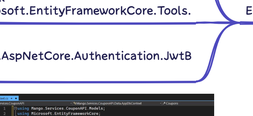
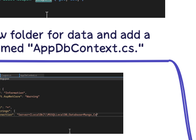
- Initial Migration
  -
  - Use the Package Manager Console to add a migration named "AddCouponToDB."
  -
  - Update the database to create the Coupon table.
-
  - Override the OnModelCreating method in AppDbContext.cs to seed initial data.

Seeding the Database

- 
  - Add a new migration named "SeedCouponTables."
-
  - Implement a method to apply migrations automatically on application startup.

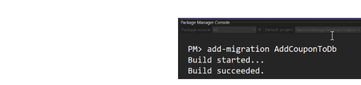
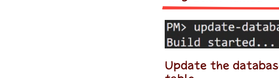
8. Building the Coupon API Controller

Creating the Controller

- CouponAPIController
  -
  - Add a new API controller named "CouponAPIController.cs."
  -
  - Implement endpoints for CRUD operations: GetAll, GetById, GetByCode, Create, Update, and Delete.
- Common Response DTO
  - 
  - Create a class file named "ResponseDTO.cs" to define a common response structure.
  - 
  - Modify the controller to return ResponseDTO for all endpoints.

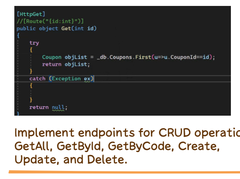
9. Implementing AutoMapper

Configuring AutoMapper

- Mapping Configuration
  - 
  - Create a new class file named "MappingConfig.cs" to define mapping configurations.
  - 
  - Register the mapping configuration in Program.cs.
- Using AutoMapper in the Controller
  - Inject AutoMapper using dependency injection.
  - 
  - Use AutoMapper to map between Coupon and CouponDTO in the controller endpoints.

10. Finalizing the Coupon API

Testing the Endpoints

- Running the Application
  - Run the application and use Swagger to test the endpoints.
  - 
  - Ensure that all CRUD operations (GetAll, GetById, GetByCode, Create, Update, Delete) are working as expected.