ПРАКТИЧНЕ ЗАНЯТТЯ № 03

ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

Тема: **ASP.NET. Тришарова архітектура на базі ENTITY FRAMEWORK.**

Мета: дослідження практичних аспектів реалізації веб-додатків на основі платформи ASP.NET на базі функціональних можливостей Entity Framework.

Програмне забезпечення: ОС Windows/Linux, Visual Studio, Visual Studio Code

Порядок виконання роботи:

Частина 1: Entity Framework та CodeFirst-підхід при проектуванні бази даних. Fluent Api.

1.1. У відповідності до сформованої теми роботи (технічне завдання) розробити РЕЛЯЦІЙНУ базу даних предметної області, використовуючи інструментарій Entity Framework:

- за основу можна використати реляційну базу даних, що була розроблена у попередній роботі;
- використовуючи інструментарій Entity Framework Core та Fluent Api (https://docs.microsoft.com/en-us/ef/core/modeling/)
- розробити моделі (передбачити налаштування стовпців, таблиць, ключей, індексів, генерації значень стовпців тощо);
- врахувати присутність співвідношень між моделями (один-до-одного, один-до-багатьох, багато-до-багатьох) (https://docs.microsoft.com/en-us/ef/core/modeling/relationships?tabs=fluent-api%2Cfluent-api-simple-key%2Csimple-key), сконфігурувати всі взаємозв'язки, вводячи при необхідності проміжні сутності, на базі Fluent Api (https://metanit.com/sharp/entityframeworkcore/3.6.php);
- передбачити можливість конфігрування окремих таблиць/сутностей у розрізі окремих класів кофігурації (https://metanit.com/sharp/entityframeworkcore/2.13.php).

1.2. Використовуючи інструментарій "міграцій бази данних" (https://docs.microsoft.com/en-us/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli) привести реляційну базу даних до бажаного стану.

1.3. Порівняти отриманий результат із структурою бази даних, що використовувалась для попередньої роботи. Усунути всі неточності опираючись на інструментарій "міграцій".

Частина 2: Entity Framework. LINQ to Entities. Паттерни Generic Repository/Unit of Work

У відповідності до сформовованого технічного завдання розробити функціонал шару Data Access Layer. В якості бази даних використати розроблену базу даних на попередньому лабораторному занятті.

Зокрема:
2.1. Реалізувати паттерн Async Generic Repository. Врахувати, що вказана конструкція не повинна містити бізнес-логіки та керування бізнес-процесами.
2.2. На основі Async Generic Repository розробити (3 шт.) конкретних репозиторії в розрізі яких передбачити виконання ряду специфіхних запитів (не є частиною Generic Repository), що повинні бути реалізовані на основі:

- Eager Loading;
- Explicit Loading;
- інструментарію Linq To Entities (багато до багатьох із використанням проміжної сутності).

2.3. Реалізувати патерн Unit of Work. Повинен містити в собі весь набір репозиторіїв. Перебачити у UOW функціонал збереження всіх змін на рівні бази даних.

Частина 3: DTO, Automapper, End-Points
3.1. У відповідності до сформовованого технічного завдання розробити функціонал шару Business Logic Layer.
Так, зокрема, для відповідного шару тришарової архітектури:

- Розробити сукупність сервісів де в розірізі кожного передбачити виконання певних бізнес-задач (https://emacsway.github.io/ru/service-layer/);
- Розробити сукупність моделей - Data Transfer Objects (DTO);
- Передбачити наявність функціоналу проеціювання даних між моделями Entity та DTO. Для розв'язку даного завдання використати функціонал бібліотеки AutoMapper.

3.2. У відповідності до сформовованого технічного завдання розробити функціонал шару WEB/API (контролери).
Так, зокрема, для відповідного шару тришарової архітектури передбачити:

- асинхронність контролерів (https://docs.microsoft.com/ru-ru/aspnet/core/web-api/action-return-types?view=aspnetcore-6.0)
- наявність маршрутизації на основі атрибутів (https://docs.microsoft.com/ru-ru/aspnet/core/mvc/controllers/routing?view=aspnetcore-6.0). При бажанні можна використати інший підхід.
- роботу із статусними кодами;
- описові відомості для сторінок-довідок по веб-API;

3.3. Додатково:
підключити Swagger (https://docs.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-6.0).

Частина 4: Частина 3: Seeding, Sorting, Filtering, Paging, Validation. FluentValidation.

4.1. На основі попередньо розробленого проекту (частина №01-03) передбачити коректну реалізацію наступних операцій:

- сідінгу бази даних при виконанні Code First Migration;
- пагінації;
- фільтрації;
- пошуку;
- сортування.

4.2. На основі попередньо розробленого проекту (частина №01-03) реалізувати валідування моделей:

- на основі атрибутів (тільки на основі 1 моделі).
- на основі функціоналу бібліотеки FluentValidation.

Частина 5: В якості системи контролю версій використати GIT.

Контрольні питання:

1. https://www.interviewbit.com/entity-framework-interview-questions/
2. https://www.fullstack.cafe/blog/entity-framework-interview-questions
3. http://a4academics.com/interview-questions/52-dot-net-interview-questions/973-entity-framework