**LAB №01 - ASP.NET. Тришарова архітектура (Data Access Layer): ADO.NET та Dapper.**

1. У відповідності до сформованої теми роботи (технічне завдання) розробити РЕЛЯЦІЙНУ базу даних предметної області.

2. На рівні реляційної бази даних передбачити наявність різного виду зв'язків між таблицями (один до одного, один до багатьох, багато до багатьох).

3. Для однотипних запитів до бази даних передбачити наявність збережуваних процедур (4-6).

4. Використовуючи технологію доступу та керування даними [ADO.NET](https://moodle.chnu.edu.ua/mod/url/view.php?id=108200) розробити репозиторій із CRUD - запитами (1 шт - чистий [ADO.NET](https://moodle.chnu.edu.ua/mod/url/view.php?id=108200)). Врахувати можливість роботи із збережуваними процедурами.

5. Використовуючи поєднання [ADO.NET](https://moodle.chnu.edu.ua/mod/url/view.php?id=108200) та ORM-маппера [Dapper](https://moodle.chnu.edu.ua/mod/url/view.php?id=108209) розробити репозиторій із CRUD - запитами (2 шт - [ADO.NET](https://moodle.chnu.edu.ua/mod/url/view.php?id=108200) + [Dapper](https://moodle.chnu.edu.ua/mod/url/view.php?id=108209)). Врахувати можливість роботи із збережуваними процедурами.

В якості системи контролю версій використати GIT.

**LAB №02 - ASP.NET. Тришарова архітектура (Data Access Layer): Entity Framework та CodeFirst-підхід при проектуванні бази даних. Fluent Api.**

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>).

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.

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

В якості системи контролю версій використати GIT.

№03 - ASP.NET. Тришарова архітектура (Data Access Layer): 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:

1) Реалізувати паттерн Async Generic Repository. Врахувати, що вказана конструкція не повинна містити бізнес-логіки та керування бізнес-процесами.

2) На основі Async Generic Repository розробити (3 шт.) конкретних репозиторії в розрізі яких передбачити виконання ряду специфіxних 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).

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

**№04 - ASP.NET. Тришарова архітектура ( Business Logic Layer/WEB(API)-Layer): DTO, Automapper, End-Points**

*I. У відповідності до сформовованого технічного завдання розробити функціонал шару Business Logic Layer.*

Так, зокрема, для відповідного шару тришарової архітектури:

1) Розробити сукупність сервісів де в розірізі кожного передбачити виконання певних бізнес-задач (<https://emacsway.github.io/ru/service-layer/>);

2) Розробити сукупність моделей - Data Transfer Objects (DTO);

3) Передбачити наявність функціоналу проеціювання даних між моделями Entity та DTO. Для розв'язку даного завдання використати функціонал бібліотеки AutoMapper.

*II. У відповідності до сформовованого технічного завдання розробити функціонал шару WEB/API (контролери).*

Так, зокрема, для відповідного шару тришарової архітектури передбачити:

1) асинхронність контролерів (<https://docs.microsoft.com/ru-ru/aspnet/core/web-api/action-return-types?view=aspnetcore-5.0>)

2) наявність маршрутизації на основі атрибутів (<https://docs.microsoft.com/ru-ru/aspnet/core/mvc/controllers/routing?view=aspnetcore-5.0>). При бажанні можна використати інший підхід.

3) роботу із статусними кодами;

4) описові відомості для сторінок-довідок по веб-API;

III. Додатково:

підключити Swagger (<https://docs.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-5.0>).

**№05 - ASP.NET. Entity Framework. Seeding, Sorting, Filtering, Paging, Validation. FluentValidation.**

I. На основі попередньо розробленого проекту (лабораторні заняття №01 - №04) передбачити коректну реалізацію наступних операцій:

сідінгу бази даних при виконанні Code First Migration;

* пагінації;
* фільтрації;
* пошуку;
* сортування.

ІІ. На основі попередньо розробленого проекту (лабораторні заняття №01 - №04) реалізувати валідування моделей:

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

**№06 - ASP.NET. Authentication and Authorization. Identity Server. ASP.NET Identity. JWT**

В розрізі окремого мікросервісу (проекту) реалізувати концепцію захисту розробляємих мікросервісів (API) аутентифікації/авторизації на базі Identity Server 4 (OpenID Connect and Outh 2.0).

Для цього передбачити:

1) роботу із clients (список дозволених для роботи із вашою системою додатків - Access Control List)

2) роботу із scopes (регламентування списку "дозволеностей" для клієнта)

3) роботу із resources (список API із якими може проводитись робота)

4) роботу із users (користувачі, що будуть взаємодіяти із вашою системою)

В якості сховища даних використати реляційну базу даних (за вибором) та реалізувати роботу із БД на основі пакету Entity Framework та ASP.NET СORE Identity. Предбачити збереження наступної інформації про: clients, resources, scopes, та persisted grants

Логіка роботи:

* Користувач відправляє на сервер запит на вхід з іменем користувача та паролем.
* Сервер аутентифікації бере цю інформацію та здійснює запит до бази даних, поєднує ім'я користувача із секретним ключем та генерує токен.
* Коли користувач аутентифікований, сервер повертає токен, і записує цю інформацію у відповідний стовпець користувача у базі даних.
* Клієнт передає токен JWT при виконанні викликів API у частину заголовку запитів, і сторона серверу кожен раз перевіряє, чи дійсний токен.
* На основі отриманої інформації відбувається обробка запиту.