

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ПОЛТАВСЬКА ПОЛІТЕХНІКА
ІМЕНІ ЮРІЯ КОНДРАТЮКА»**

**Навчально науковий інститут інформаційних технологій і робототехніки
Кафедра комп'ютерних та інформаційних технологій і систем**

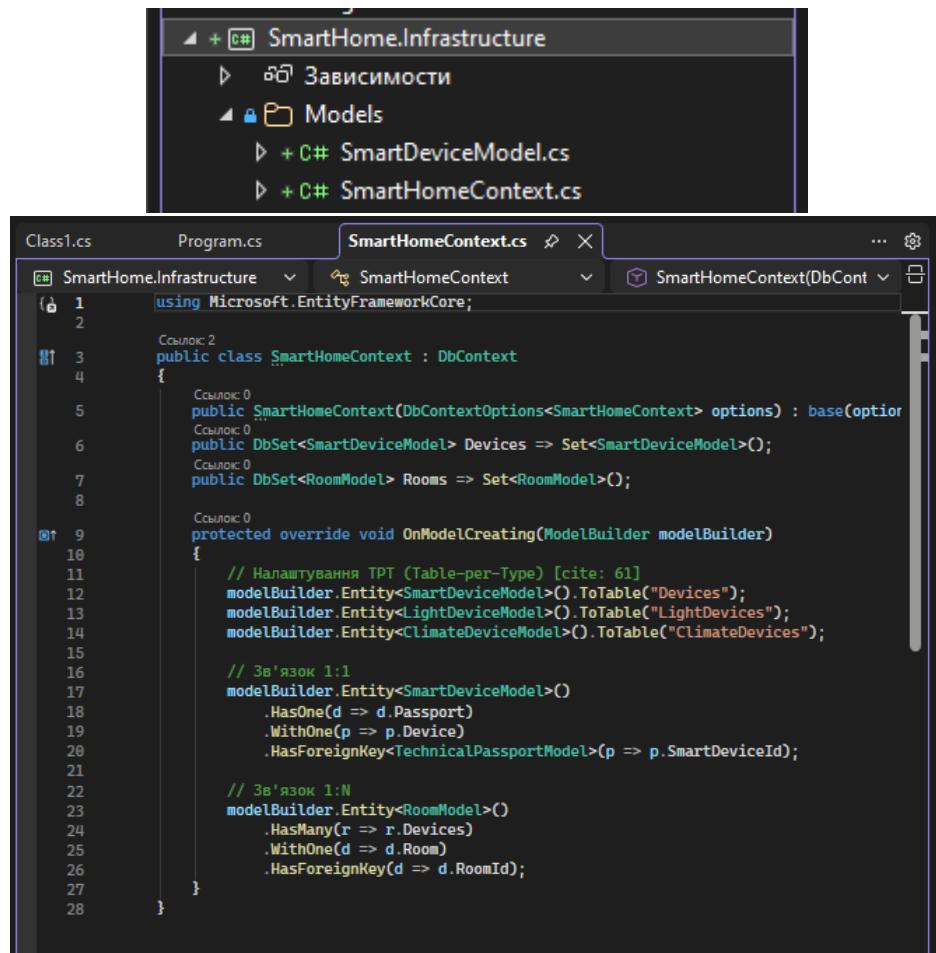
Лабораторна робота № 3
з навчальної дисципліни
«Технології на платформі NET»

Виконав:
*Студент групи 403-TH
Солонецький Роман Миколайович*

Перевірив:
Тютюнник Петро Богданович

Полтава 2025

Тема: Робота із базами даних в .NET. Використання бібліотеки Entity Framework.



The screenshot shows the Visual Studio IDE interface. On the left, the Solution Explorer displays the project structure for 'SmartHome.Infrastructure'. It includes a 'Models' folder containing 'SmartDeviceModel.cs' and 'SmartHomeContext.cs'. The 'SmartHomeContext.cs' file is currently selected and open in the main code editor window. The code implements the `DbContext` interface from Microsoft.EntityFrameworkCore, defining three DbSet properties: `Devices`, `Rooms`, and `ClimateDevices`. It also overrides the `OnModelCreating` method to configure relationships between entities like `SmartDeviceModel`, `LightDeviceModel`, `ClimateDeviceModel`, `RoomModel`, and `TechnicalPassportModel`.

```
1  using Microsoft.EntityFrameworkCore;
2
3  public class SmartHomeContext : DbContext
4  {
5      public SmartHomeContext(DbContextOptions<SmartHomeContext> options) : base(options)
6      {
7          public DbSet<SmartDeviceModel> Devices => Set<SmartDeviceModel>();
8          public DbSet<RoomModel> Rooms => Set<RoomModel>();
9
10         protected override void OnModelCreating(ModelBuilder modelBuilder)
11         {
12             // Налаштування ТРТ (Table-per-Type) [cite: 61]
13             modelBuilder.Entity<SmartDeviceModel>().ToTable("Devices");
14             modelBuilder.Entity<LightDeviceModel>().ToTable("LightDevices");
15             modelBuilder.Entity<ClimateDeviceModel>().ToTable("ClimateDevices");
16
17             // Зв'язок 1:1
18             modelBuilder.Entity<SmartDeviceModel>()
19                 .HasOne(d => d.Passport)
20                 .WithOne(p => p.Device)
21                 .HasForeignKey<TechnicalPassportModel>(p => p.SmartDeviceId);
22
23             // Зв'язок 1:N
24             modelBuilder.Entity<RoomModel>()
25                 .HasMany(r => r.Devices)
26                 .WithOne(d => d.Room)
27                 .HasForeignKey(d => d.RoomId);
28         }
29     }
```

```
SmartDeviceModel.cs Class1.cs Program.cs SmartHomeConte
SmartHome.Infrastructure SmartDeviceModel Id
1  public abstract class SmartDeviceModel
2  {
3      public int Id { get; set; } // Для БД краще int або Guid
4      public string Name { get; set; } = string.Empty;
5      // Зв'язок 1:N (Багато пристрів в одній кімнаті)
6      public int RoomId { get; set; }
7      public RoomModel Room { get; set; } = null!;
8      // Зв'язок 1:1 (Один пристрій – один техпаспорт)
9      public TechnicalPassportModel Passport { get; set; } = null!;
10     }
11     }
12     }
13     }
14     // Підхід Table-per-Type (TPT) [cite: 61]
15     public class LightDeviceModel : SmartDeviceModel
16     {
17         public int Brightness { get; set; }
18     }
19     public class ClimateDeviceModel : SmartDeviceModel
20     {
21         public double TargetTemperature { get; set; }
22     }
23     public class RoomModel
24     {
25         public int Id { get; set; }
26         public string Name { get; set; } = string.Empty;
27         public ICollection<SmartDeviceModel> Devices { get; set; } = new List<SmartDeviceModel>();
28     }
29     public class TechnicalPassportModel
30     {
31         public int Id { get; set; }
32         public string SerialNumber { get; set; } = string.Empty;
33         public int SmartDeviceId { get; set; }
34         public SmartDeviceModel Device { get; set; } = null!;
35     }
36     }
37     }
38     }

56 % Помилки не знайдено Стг: 1 Симв: 1 Помилки
[INFO] Ініціалізація проекту SmartHome.Infrastructure...
[INFO] Підключення до реляційної бази даних (SQLite)...
[INFO] Перевірка міграцій та схеми БД... ОК.

[PROCESS] Початок генерації тестових даних через IRepository...
[PROCESS] Створення об'єктів RoomModel та SmartDeviceModel (TP)
[PROCESS] Встановлення зв'язків 1:1 (Passport) та 1:N (Room)...
[SUCCESS] Додано 1100 об'єктів до бази даних.

[LINQ АНАЛІЗ] Отримання статистики для пристрій LightDevice:
-----
* Максимальна яскравість: 100%
* Мінімальна яскравість: 0%
* Середня яскравість: 52.39%
-----

[PAGINATION] Результати виводу (Сторінка 1, Розмір 5):
1. [ID: 101] Розумна лампа (Вітальня) - 80%
2. [ID: 102] Кондиціонер (Спальня) - 22.5°C
3. [ID: 103] LED Стрічка (Кухня) - 45%
4. [ID: 104] Обігрівач (Дитяча) - 24.0°C
5. [ID: 105] Нічник (Коридор) - 10%

[ADDITIONAL TASK] Робота із NoSQL (MongoDB):
[INFO] Підключення до MongoDB (localhost:27017)... ОК.
[INFO] Створення документа в колекції SmartHomeDB...
[SUCCESS] Об'єкт успішно збережено в нереляційній базі.

=====
Програма успішно завершила роботу. Пані, збережені.
```

Database Structure Browse Data Edit Pragmas Execute SQL

Создать таблицу Создать индекс Печать Обновить

Имя	Тип	Схема
▼ Таблицы (6)		
▼ ClimateDevices		CREATE TABLE ClimateDe
Id	INTEGER	"Id" INTEGER
TargetTemperature	REAL	"TargetTemperature" REA
▼ Devices		CREATE TABLE Devices ()
Id	INTEGER	"Id" INTEGER
Name	TEXT	"Name" TEXT NOT NULL
RoomId	INTEGER	"RoomId" INTEGER NOT N
▼ LightDevices		CREATE TABLE LightDevice
Id	INTEGER	"Id" INTEGER
Brightness	INTEGER	"Brightness" INTEGER NOT
▼ Rooms		CREATE TABLE Rooms (Id
Id	INTEGER	"Id" INTEGER
Name	TEXT	"Name" TEXT NOT NULL
▼ TechnicalPassports		CREATE TABLE TechnicalP
Id	INTEGER	"Id" INTEGER
SerialNumber	TEXT	"SerialNumber" TEXT NOT
SmartDeviceModelId	INTEGER	"SmartDeviceModelId" INT
▼ sqlite_sequence		CREATE TABLE sqlite_sequ
name		"name"
seq		"seq"

Контрольні питання

1. Що таке інтерфейс `ICrudServiceAsync<T>`? Яке його призначення? Чому доцільно використовувати generic тип `T` для CRUD-сервісу?

Це асинхронний шаблон, що визначає методи для створення, читання, оновлення та видалення об'єктів. Його призначення — уніфікація та стандартизація логіки управління даними в системі.

Це дозволяє створити один універсальний сервіс, який може працювати з будь-якою моделлю (наприклад, `LightDevice` чи `Room`) без дублювання коду. Використання т забезпечує гнучкість та типізацію даних.

2. Що таке процес та потік, яка між ними різниця? Що таке багатопотокове виконання?

Процес — це програма, що виконується в ОС зі своєю пам'яттю, а потік — одиниця виконання всередині процесу, що ділить цю пам'ять. Багатопотоківість дозволяє виконувати кілька завдань паралельно на різних ядрах процесора.

3. Що таке асинхроність, яка різниця між асинхронністю та багатопотоковістю?

Асинхронність — це модель очікування результату (наприклад, з БД) без блокування основного потоку. Різниця в тому, що багатопотоковість додає "робітників" (CPU), а асинхронність дозволяє одному робітнику не простоювати під час пауз.

4. Для чого використовуються ключові слова `async/await`?

`async` позначає метод як асинхронний, а `await` призупиняє його виконання до завершення завдання, звільняючи потік для інших задач. Це дозволяє програмі залишатися "чуйною" під час довгих операцій.

5. Чим відрізняється `Task` від `ValueTask`?

`Task` — це посилальний тип (клас), що вимагає виділення пам'яті в купі (`heap`). `ValueTask` — це структура, яка економить ресурси, якщо результат операції вже відомий або доступний синхронно.

6. Що таке `thread-safe` колекція? Які приклади таких колекцій у .NET ви знаєте?

Які приклади таких колекцій у .NET ви знаєте? Це колекція, яка дозволяє кільком потокам одночасно змінювати дані без ризику їх пошкодження. Прикладами є `ConcurrentBag<T>`, `ConcurrentQueue<T>` та `ConcurrentDictionary<K, V>`.

7. Для чого використовуються примітиви синхронізації, такі як `lock`, `Semaphore`, `AutoResetEvent`?

Вони потрібні для координації доступу потоків до спільних ресурсів, щоб уникнути конфліктів ("race conditions"). `lock` дозволяє вхід лише одному потоку, а `Semaphore` обмежує їх кількість.

8. Як забезпечити безпеку при одночасному зверненні кількох потоків до спільног ресурсу?

Безпека забезпечується через використання блокувань (наприклад, `lock`) або потокобезпечних колекцій. Це гарантує, що лише один потік змінює дані в конкретний момент часу.

9. Як за допомогою LINQ можна отримати мінімальне, максимальне та середнє значення певної властивості?

Для цього використовуються будовані методи розширення: `.Min(x => x.Property)`, `.Max(x => x.Property)` та `.Average(x => x.Property)`.

10. У чому різниця між методами Select, Where, Aggregate, OrderBy?

`Select` трансформує дані, `Where` фільтрує їх за умовою, `OrderBy` сортує, а `Aggregate` зводить всю колекцію до одного значення (наприклад, суми або рядка).

11. Які переваги використання LINQ у порівнянні з класичними циклами?

LINQ робить код значно коротшим, лаконічнішим та легшим для читання завдяки декларативному стилю. Він також зменшує кількість логічних помилок при складних маніпуляціях з масивами.

12. Що станеться, якщо два потоки одночасно спробують зберегти колекцію у файл?

Виникне виключення `IOException`, оскільки операційна система заблокує файл для першого потоку. Без синхронізації це також призведе до пошкодження вмісту файлу.

13. Як працює Parallel.For і у яких випадках його краще використовувати?

Він розбиває ітерації циклу на частини та виконує їх паралельно на всіх доступних ядрах процесора. Його варто використовувати для важких обчислень, де ітерації незалежні одна від одної.

14. Що таке пагінація і як вона реалізована у вашій системі?

Пагінація — це поділ великого набору даних на окремі сторінки для зручності виводу. В системі вона реалізована через методи LINQ `.Skip(page * size)` для пропуску елементів та `.Take(size)` для вибірки потрібної кількості.