

Лабораторна робота №3

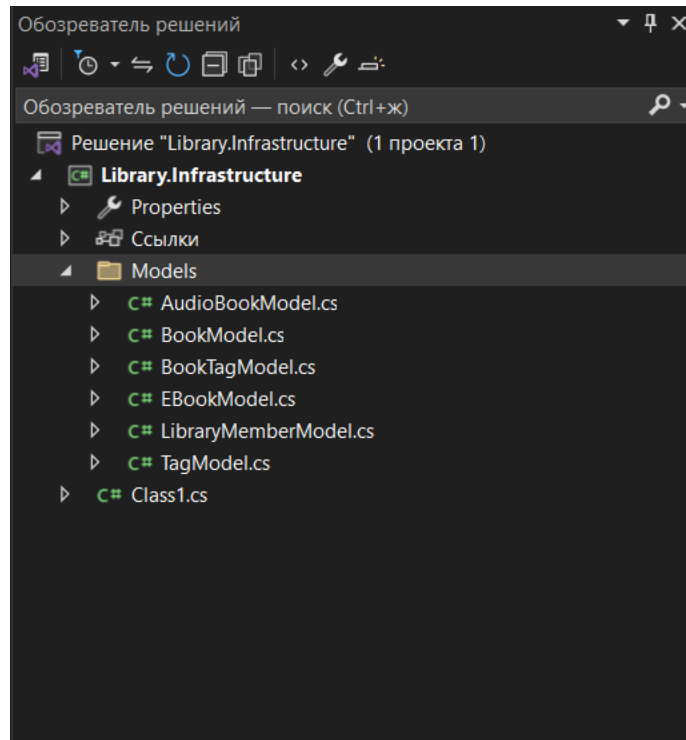
Робота із базами даних в .NET. Використання бібліотеки Entity Framework.

Виконала студентка групи 302-ТК

Варинська Євгенія

1. Створити проєкт {Назва тематики}.Infrastructure. У новоствореному проєкті створити папку Models. У створеній папці додати модель(набір класів), яка відповідає створеній у першій лабораторній роботі моделі, та містить всередині себе властивості. Наприклад, якщо існував у першій лабораторній клас Bus, у папці Models необхідно створити клас BusModel. Отримана модель має обов'язково мати зв'язки один-до-одного та один-до-багатьох. За додавання зв'язків багато-до-багатьох можна отримати додаткові бали. Для наслідування більш пріоритетно застосовувати підхід Таблиця на тип (Table-per-Type).

Структура проєкту:



Клас BookModel:

```

C# Library.Infrastructure Library.Infrastructure.Models.Boo
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Library.Infrastructure.Models
8  {
9      Ссылка: 4
10     public class BookModel
11     {
12         Ссылка: 0
13         public int Id { get; set; }
14         Ссылка: 0
15         public string Title { get; set; }
16         Ссылка: 0
17         public string Author { get; set; }
18
19         // Один-до-багато: Книга належить одному учаснику
20         Ссылка: 0
21         public int LibraryMemberId { get; set; }
22         Ссылка: 0
23         public LibraryMemberModel LibraryMember { get; set; }
24
25         // Багато-до-багато: Книга має багато тегів
26         Ссылка: 0
27         public List<BookTagModel> BookTags { get; set; }
28     }
29 }

```

Клас EBookModel:

```

C# Library.Infrastructure Library.Infrastructure
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Library.Infrastructure.Models
8  {
9      Ссылка: 0
10     public class EBookModel : BookModel
11     {
12         Ссылка: 0
13         public string FileFormat { get; set; }
14         Ссылка: 0
15         public double FileSizeMB { get; set; }
16     }
17 }

```

Клас AudioBookModel:

```

C# Library.Infrastructure Library.Infrastructure.Models.AudioBo
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Library.Infrastructure.Models
8  {
9      Ссылка: 0
10     public class AudioBookModel : BookModel
11     {
12         Ссылка: 0
13         public string AudioFormat { get; set; }
14         Ссылка: 0
15         public double DurationInMinutes { get; set; }
16     }
17 }

```

Клас LibraryMemberModel:

```

Library.Infrastructure Library.Infrastructure.Models.LibraryMemb
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Library.Infrastructure.Models
8  {
9      Ссылка: 1
10     public class LibraryMemberModel
11     {
12         Ссылка: 0
13         public int Id { get; set; }
14         Ссылка: 0
15         public string FullName { get; set; }
16         Ссылка: 0
17         public DateTime JoinDate { get; set; }
18
19         // Один-до-багатох: Один учасник може позичати багато книжок
20         Ссылка: 0
21         public List<BookModel> BorrowedBooks { get; set; }
22     }
23 }

```

Клас TagModel:

```

Library.Infrastructure Library.Infrastructure.Models.TagMod
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Library.Infrastructure.Models
8  {
9      Ссылка: 1
10     public class TagModel
11     {
12         Ссылка: 0
13         public int Id { get; set; }
14         Ссылка: 0
15         public string Name { get; set; }
16
17         // Багато-до-багатох: Один тег - багато книжок
18         Ссылка: 0
19         public List<BookTagModel> BookTags { get; set; }
20     }
21 }

```

Клас BookTagModel: (таблиця зв'язку багато-до-багатох)

```

Library.Infrastructure Library.Infrastructure
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Library.Infrastructure.Models
8  {
9      Ссылка: 2
10     public class BookTagModel
11     {
12         Ссылка: 0
13         public int BookId { get; set; }
14         Ссылка: 0
15         public BookModel Book { get; set; }
16
17         Ссылка: 0
18         public int TagId { get; set; }
19         Ссылка: 0
20         public TagModel Tag { get; set; }
21     }
22 }

```

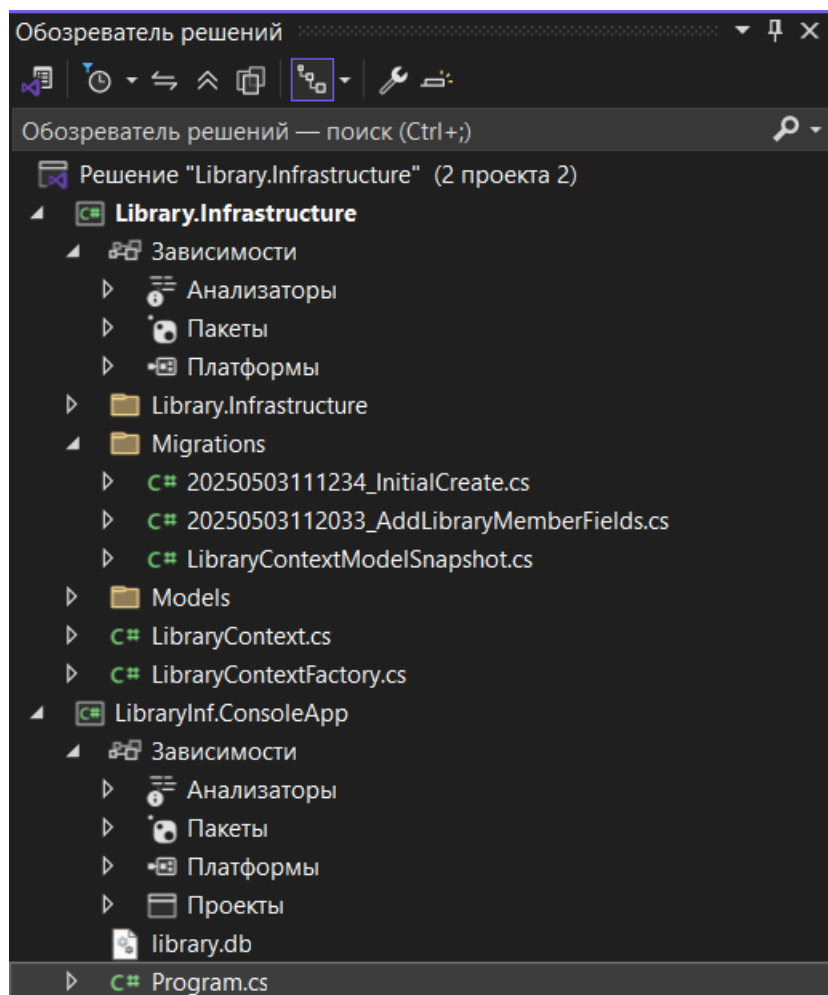
2. Створити клас {Назва тематики}Context, який наслідуватиметься від класу DbContext та опише схему бази даних для обраної тематики застосовуючи Entity Framework та анотації або Fluent API на вибір, Fluent API більш пріоритетно.

```
LibraryContext.cs  + X
Library.Infrastructure  Library.Infrastructure.LibraryContext

1  using Library.Infrastructure.Models;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7  using Microsoft.EntityFrameworkCore;
8
9
10 namespace Library.Infrastructure
11 {
12     Ссылка: 2
13     public class LibraryContext : DbContext
14     {
15         Ссылка: 0
16         public LibraryContext(DbContextOptions<LibraryContext> options)
17             : base(options)
18         {
19             Ссылка: 0
20             public DbSet<BookModel> Books { get; set; }
21             Ссылка: 0
22             public DbSet<AudioBookModel> AudioBooks { get; set; }
23             Ссылка: 0
24             public DbSet<EBookModel> EBooks { get; set; }
25             Ссылка: 0
26             public DbSet<LibraryMemberModel> LibraryMembers { get; set; }
27             Ссылка: 0
28             public DbSet<TagModel> Tags { get; set; }
29             Ссылка: 0
30             public DbSet<BookTagModel> BookTags { get; set; }
31
32         Ссылка: 0
33         protected override void OnModelCreating(ModelBuilder modelBuilder)
34         {
35             // ===== Table-per-Type успадкування =====
36             modelBuilder.Entity<BookModel>().ToTable("Books");
37             modelBuilder.Entity<AudioBookModel>().ToTable("AudioBooks");
38             modelBuilder.Entity<EBookModel>().ToTable("EBooks");
39
40             // ===== Один-до-багатьох: Book → LibraryMember =====
41             modelBuilder.Entity<BookModel>()
42                 .HasOne(b => b.LibraryMember)
43                 .WithMany(l => l.BorrowedBooks)
44                 .HasForeignKey(b => b.LibraryMemberId)
45                 .OnDelete(DeleteBehavior.Cascade);
46
47             // ===== Багато-до-багатьох: Book ↔ Tag через BookTag =====
48             modelBuilder.Entity<BookTagModel>()
49                 .HasKey(bt => new { bt.BookId, bt.TagId });
50
51             modelBuilder.Entity<BookTagModel>()
52                 .HasOne(bt => bt.Book)
53                 .WithMany(b => b.BookTags)
54                 .HasForeignKey(bt => bt.BookId);
55
56             modelBuilder.Entity<BookTagModel>()
57                 .HasOne(bt => bt.Tag)
58                 .WithMany(t => t.BookTags)
59                 .HasForeignKey(bt => bt.TagId);
60
61             base.OnModelCreating(modelBuilder);
62         }
63     }
64 }
```

3. Створити міграцію використовуючи Entity Framework. Створену міграцію застосувати до реляційної бази даних. Можна використовувати будь-яку реляційну СУБД, як MS SQL, PostgreSQL, MySQL, тощо. Також слід використати SQLite, коли немає можливості розробляти та здавати Лабораторну роботу на одній системі. При використанні SQLite файл бази даних необхідно додати до гіт репозиторія.

Структура:



Program.cs:

```

LibraryMemberModel.cs  BookModel.cs  Program.cs  Librar
C# LibraryInf.ConsoleApp
1  using Library.Infrastructure;
2  using Library.Infrastructure.Models;
3  using Microsoft.EntityFrameworkCore;
4
5  var options = new DbContextOptionsBuilder<LibraryContext>()
6      .UseSqlite("Data Source=library.db")
7      .Options;
8
9  using var context = new LibraryContext(options);
10
11  // Застосовуємо міграції
12  context.Database.Migrate();
13
14  // Додаємо тестового учасника, якщо відсутній
15  var member = context.LibraryMembers.FirstOrDefault();
16  if (member == null)
17  {
18      member = new LibraryMemberModel
19      {
20          Name = "Тестовий користувач",
21          BorrowedBooks = new List<BookModel>()
22      };
23      context.LibraryMembers.Add(member);
24      context.SaveChanges();
25      Console.WriteLine("Додано тестового учасника.");
26  }
27
28  // Додаємо тестову книгу, якщо відсутня
29  if (!context.Books.Any(b => b.Title == "Test Book"))
30  {
31      var testBook = new BookModel
32      {
33          Title = "Test Book",
34          Author = "Test Author",
35          LibraryMemberId = member.Id,
36          BookTags = new List<BookTagModel>()
37      };
38
39      context.Books.Add(testBook);
40      context.SaveChanges();
41      Console.WriteLine("Додано тестову книгу.");
42  }
43  else
44  {
45      Console.WriteLine("Тестова книга вже існує.");
46  }
47
48  Console.WriteLine("Готово.");
49

```

Результат:

```

>> dotnet run --project ./LibraryInf.ConsoleApp
>>
Додано тестового учасника.
Додано тестову книгу.
Готово.

```

4. Реалізувати шаблон проектування Репозиторій, {Назва тематики} Context який буде використовувати клас {Назва тематики} Context для досьупу до даних із БД:

```

public interface IRepository<T> where T : class
{
    Task<T> GetByIdAsync(int id);
    Task<IEnumerable<T>> GetAllAsync();
    Task AddAsync(T entity);
}

```

```
Task Update(T entity);  
Task Delete(T entity);  
}
```

```
Repository.cs*  IRepository.cs*  Program.cs*  
Library.Infrastructure  Library.Infra  
{  
1  using System.Collections.Generic;  
2  using System.Threading.Tasks;  
3  
4  namespace Library.Infrastructure.Repositories  
5  {  
6  public interface IRepository<T> where T : class  
7  {  
8      Task<T?> GetByIdAsync(int id);  
9      Task<IEnumerable<T>> GetAllAsync();  
10     Task AddAsync(T entity);  
11     Task UpdateAsync(T entity);  
12     Task DeleteAsync(T entity);  
13 }  
14 }
```

```

Repository.cs*  IRepository.cs*  Program.cs*
C# Library.Infrastructure  Library.Infrastructure.Repo
3  using System.Linq.Tasks;
4
5  namespace Library.Infrastructure.Repositories
6  {
7      Ссылки 2
8      public class Repository<T> : IRepository<T> where T : class
9      {
10         private readonly LibraryContext _context;
11         private readonly DbSet<T> _dbSet;
12
13         Ссылки 1
14         public Repository(LibraryContext context)
15         {
16             _context = context;
17             _dbSet = context.Set<T>();
18         }
19
20         Ссылки 1
21         public async Task<T?> GetByIdAsync(int id)
22         {
23             return await _dbSet.FindAsync(id);
24         }
25
26         Ссылки 1
27         public async Task<IEnumerable<T>> GetAllAsync()
28         {
29             return await _dbSet.ToListAsync();
30         }
31
32         Ссылки 2
33         public async Task AddAsync(T entity)
34         {
35             await _dbSet.AddAsync(entity);
36             await _context.SaveChangesAsync();
37         }
38
39         Ссылки 1
40         public async Task UpdateAsync(T entity)
41         {
42             _dbSet.Update(entity);
43             await _context.SaveChangesAsync();
44         }
45
46         Ссылки 1
47         public async Task DeleteAsync(T entity)
48         {
49             _dbSet.Remove(entity);
50             await _context.SaveChangesAsync();
51         }
52     }
53 }

```

5. Оновити асинхронну версію дженерік CRUD сервісу, щоб він використовував репозиторій для доступу до даних:

```

public interface ICrudServiceAsync<T>
{
    public Task<bool> CreateAsync(T element);
    public Task<T> ReadAsync(Guid id);
    public Task<IEnumerable<T>> ReadAllAsync();
    public Task<IEnumerable<T>> ReadAllAsync(int page, int amount);
    public Task<bool> UpdateAsync(T element);
    public Task<bool> RemoveAsync(T element);
    public Task<bool> SaveAsync();
}

```



```
CrudServiceAsync.cs*  ICrudServiceAsync.cs*  Program.cs*
Library.Infrastructure  Library.Infrastructure.S
{
1  using System;
2  using System.Collections.Generic;
3  using System.Threading.Tasks;
4
5  namespace Library.Infrastructure.Services
6  {
7      Ссылка: 0
8      public interface ICrudServiceAsync<T>
9      {
10         Ссылка: 0
11         Task<bool> CreateAsync(T element);
12         Ссылка: 0
13         Task<T?> ReadAsync(Guid id);
14         Ссылка: 0
15         Task<IEnumerable<T>> ReadAllAsync();
16         Ссылка: 0
17         Task<IEnumerable<T>> ReadAllAsync(int page, int amount);
18         Ссылка: 0
19         Task<bool> UpdateAsync(T element);
20         Ссылка: 1
21         Task<bool> RemoveAsync(T element);
22         Ссылка: 1
23         Task<bool> SaveAsync();
24     }
25 }
```

```
CrudServiceAsync.cs*  ICrudServiceAsync.cs*  Program.cs*
Library.Infrastructure  Library.Infrastructure.Services.CrudServiceAsync

{ 1  using Library.Infrastructure.Repositories;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Threading.Tasks;
6  using Microsoft.EntityFrameworkCore;
7
8  namespace Library.Infrastructure.Services
9  {
10     Ссылка: 1
11     public class CrudServiceAsync<T> : ICrudServiceAsync<T> where T : class
12     {
13         private readonly IRepository<T> _repository;
14
15         Ссылка: 0
16         public CrudServiceAsync(IRepository<T> repository)
17         {
18             _repository = repository;
19
20         Ссылка: 1
21         public async Task<bool> CreateAsync(T element)
22         {
23             await _repository.AddAsync(element);
24             return true;
25
26         Ссылка: 1
27         public async Task<T?> ReadAsync(Guid id)
28         {
29             var entity = await _repository.GetAllAsync();
30             return entity.FirstOrDefault(e => (e as dynamic).Id == id);
31
32         Ссылка: 1
33         public async Task<IEnumerable<T>> ReadAllAsync()
34         {
35             return await _repository.GetAllAsync();
36
37         Ссылка: 1
38         public async Task<IEnumerable<T>> ReadAllAsync(int page, int amount)
39         {
40             var all = await _repository.GetAllAsync();
41             return all.Skip((page - 1) * amount).Take(amount);
42
43         Ссылка: 1
44         public async Task<bool> UpdateAsync(T element)
45         {
46             await _repository.UpdateAsync(element);
47             return true;
48
49         Ссылка: 1
50         public async Task<bool> RemoveAsync(T element)
51         {
52             await _repository.DeleteAsync(element);
53             return true;
54
55         Ссылка: 1
56         public Task<bool> SaveAsync()
57         {
58             return Task.FromResult(true);
59         }
60     }
61 }
```

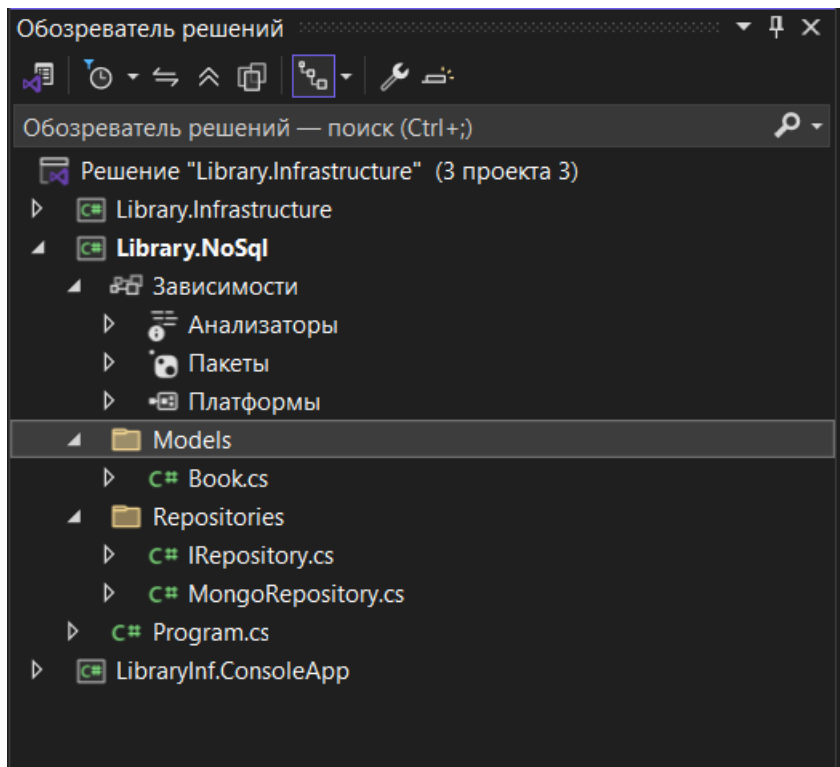
```
45         return true;
46     }
47
48     Ссылка: 1
49     public async Task<bool> RemoveAsync(T element)
50     {
51         await _repository.DeleteAsync(element);
52         return true;
53
54     Ссылка: 1
55     public Task<bool> SaveAsync()
56     {
57         return Task.FromResult(true);
58     }
59 }
```

6. Модифікуйте консольний застосунок, щоб він використовував оновлену версію CRUD сервісу та дані із бази даних.

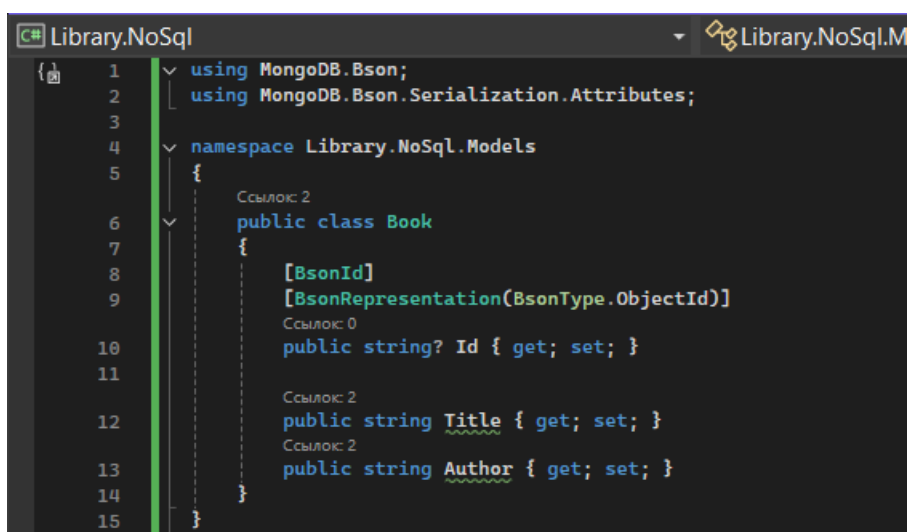
Додаткове завдання

Створіть проєкт {Назва тематики}.Nosql. У цьому проєкті реалізуйте репозиторій із 4 завдання, який буде використовувати нереляційну базу даних, як наприклад MongoDB. У звіт додайте скріншоти об'єктів із нереляційної бази даних. Використовувати Entity Framework для нереляційної БД необов'язково.

Структура:



Клас Book:



Клас IRepository:

```

1  using System.Collections.Generic;
2  using System.Threading.Tasks;
3
4  namespace Library.NoSql.Repositories
5  {
6      Ссылка: 0
7      public interface IRepository<T> where T : class
8      {
9          Ссылка: 0
10         Task<T?> GetByIdAsync(string id);
11         Ссылка: 0
12         Task<IEnumerable<T>> GetAllAsync();
13         Ссылка: 0
14         Task AddAsync(T entity);
15         Ссылка: 0
16         Task UpdateAsync(T entity);
17         Ссылка: 0
18         Task DeleteAsync(string id);
19     }
20 }

```

Клас MongoRepository:

```

1  using MongoDB.Driver;
2  using System.Collections.Generic;
3  using System.Threading.Tasks;
4
5  namespace Library.NoSql
6  {
7      Ссылка: 2
8      public class MongoRepository<T> where T : class
9      {
10         private readonly IMongoCollection<T> _collection;
11
12         Ссылка: 1
13         public MongoRepository(string connectionString, string dbName, string collectionName)
14         {
15             var client = new MongoClient(connectionString);
16             var database = client.GetDatabase(dbName);
17             _collection = database.GetCollection<T>(collectionName);
18         }
19
20         Ссылка: 1
21         public async Task AddAsync(T item) => await _collection.InsertOneAsync(item);
22
23         Ссылка: 1
24         public async Task<List<T>> GetAllAsync() =>
25             await _collection.Find(_ => true).ToListAsync();
26     }
27 }

```

Клас Program:

```

Program.cs  Book.cs  MongoRepository.cs  IRepository.cs  CrudServiceAsync.cs  ICrudServiceAsync.cs
Library.NoSql  Program  Main0
1  using Library.NoSql;
2  using Library.NoSql.Models;
3  using Library.NoSql.Repositories;
4  using System;
5  using System.Threading.Tasks;
6
7  class Program
8  {
9      Ссылка: 0
10     static async Task Main()
11     {
12         var connectionString = "mongodb+srv://ledimoana:Test1234@cluster0.jgfyrtu.mongodb.net/?retryWrites=true&w=majority";
13         var repository = new MongoRepository<Book>(connectionString, "LibraryDb", "Books");
14
15         var newBook = new Book
16         {
17             Title = "MongoDB Test Book",
18             Author = "Test Author"
19         };
20
21         await repository.AddAsync(newBook);
22         Console.WriteLine("Книгу добавлено.");
23
24         var books = await repository.GetAllAsync();
25         Console.WriteLine("Список книг:");
26         foreach (var book in books)
27         {
28             Console.WriteLine($"{book.Title} від {book.Author}");
29         }
30     }
31 }

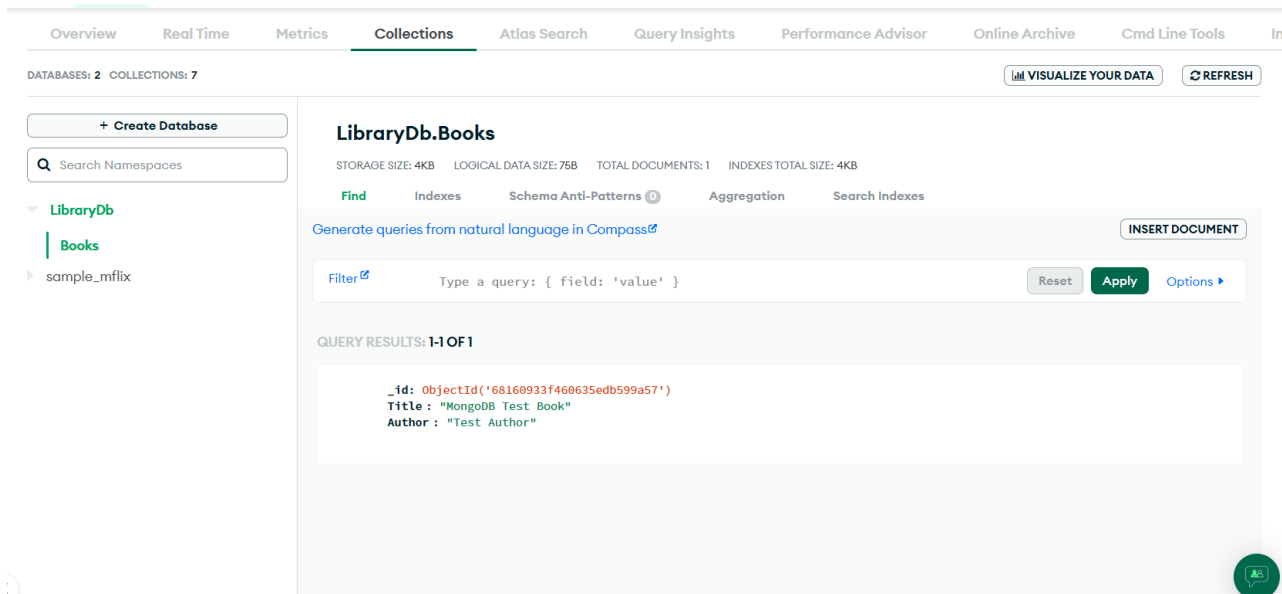
```

Результат в консоли:

```
Консоль отладки Microsoft Visual Studio
Книгу добавлено.
Список книг:
- MongoDB Test Book в?д Test Author

E:\project\Library.NoSql\bin\Debug\net9.0\Library.NoSql.exe (процесс 6908) завершил работу с кодом 0 (0x0).
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно: _
```

Результат в БД:



БД створена через сайт

<https://cloud.mongodb.com/v2#/org/68160468d36ccf1679644c89/projects>

Посилання на GitHub:

https://github.com/varinskaevg/NUPP_NET_2025_302_TK_Varinska_Lab/tree/Lab3