Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

WEB-Програмування

Лабораторна робота №4

Тема: EF CRUD

Виконав:
Студент гр. ІТ-83
Чертов М. А.

Перевірив:
Викладач
Вовк Є. А.

Київ – 2020

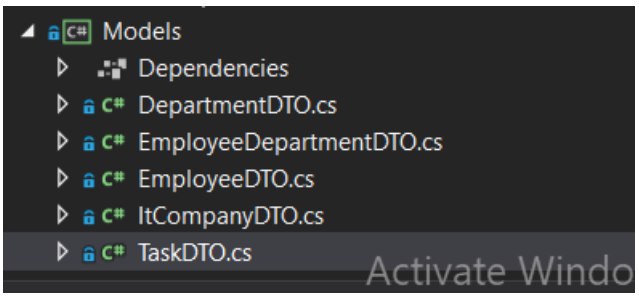
Мета: Розібратись з роботою з базами даних в Asp.net. Розібратись з шаблоном Repository при роботі з базами даних. Навчитись створювати міграції:

- Реалізувати репозиторії які будуть зберігати дані в базі даних. Реалізувати CRUD (Create, Read, Update, Delete) операції відповідно до варіанту.
- Створити кілька Transient і Scoped сервісів. Наприклад Transient сервіс може здійснювати якісь обчислення.
- Створити DTOs (Data transfer objects) для відображення моделі на користувацькому інтерфейсі.
- Строчка з'єднання з базою даних повинна міститись в файлі конфігурації.
- Реалізувати користувацький інтерфейс, що дозволяє виконувати весь круд операцій з сутностями.

Хід роботи

1. Створити DTOs (Data transfer objects) для відображення моделі на користувацькому інтерфейсі.

   DTO об'єкти містять тільки поля, що будуть відображені на View на відміну від сутностей, що містяться в проекті Entities і містять поля, що віповідають за зв’язки між сутностями.




```cs
using System.Collections.Generic;
using System.Text;

namespace Models
{
    25 references
    public class TaskDTO
    {
        7 references
        public int Id { get; set; }

        15 references
        public string Name { get; set; }

        15 references
        public string FullDiscription { get; set; }

        15 references
        public DateTime Start { get; set; }

        15 references
        public DateTime Deadline { get; set; }

        13 references
        public bool Performed { get; set; }
    }
}
```

View не мають доступу до сутностей. Тільки до ДТО об’єктів.

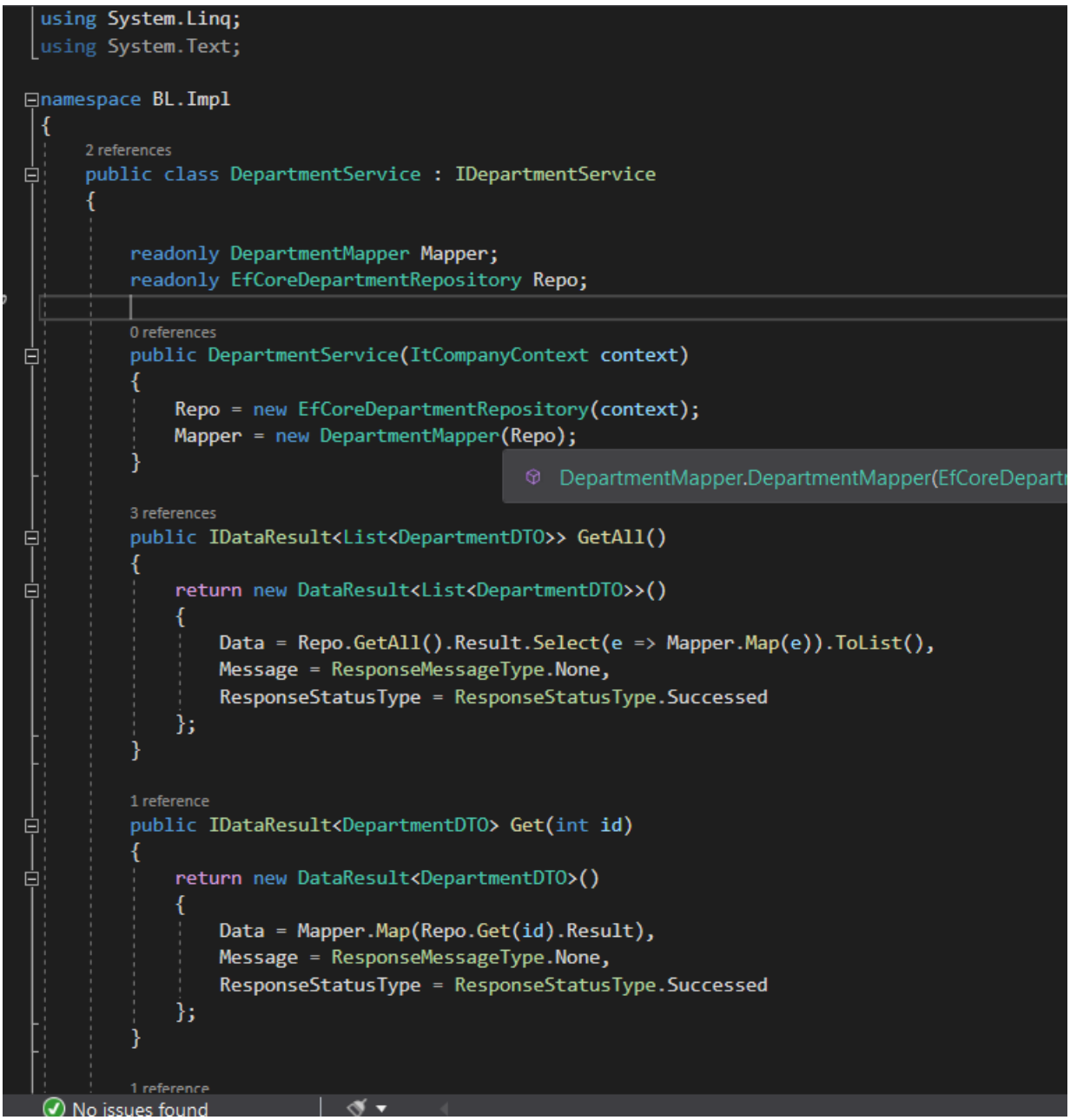
```cshtml
@model Models.TaskDTO

@{
    ViewData["Title"] = "Create";
}

<h1>Create</h1>

<h4>Task</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Name" class="control-label"></label>
                <input asp-for="Name" class="form-control" />
                <span asp-validation-for="Name" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="FullDiscription" class="control-label"></label>
```

Контроллери не мають доступу до контексту даних чи сутностей. Вони використовують сервіси та ДТО об’єкти.

```
public class TasksController : Controller
{

    private readonly ITaskService _TaskService;
    private readonly IEmployeeService _EmployeeService;
    private readonly IDepartmentService _DepartmentService;


    0 references
    public TasksController(ITaskService taskService, IEmployeeService employeeService, IDepartmentService departmentService)
    {
        _TaskService = taskService;
        _EmployeeService = employeeService;
        _DepartmentService = departmentService;
    }

    // GET: Tasks
    3 references
    public ActionResult<IEnumerable<TaskDTO>> Index([FromForm] string[] filters)
    {
        return View(_TaskService.GetAll().Data);
    }

    // GET: [controller]/Details/5
    [HttpGet("[controller]/Details/{id}")]
    0 references
    public ActionResult<TaskDTO> Details(int id)
    {
        var item = _TaskService.Get(id).Data;
        if (item == null)
            return NotFound();
        return View(item);
    }

    // GET: Tasks/Create
    0 references
    public IActionResult Create()
    {
        ConfigureViewData();
        return View();
    }
```

```
2 references
protected void ConfigureViewData()
{
    ViewData[TasksViewDataParams.EmpoyeeId] = new SelectList(
        _EmployeeService.GetAll().Data,
        nameof(EmployeeDTO.Id),
        nameof(Employee.Name));

    ViewData[TasksViewDataParams.DepartmentId] = new SelectList(_DepartmentService.GetAll().Data,
        nameof(DepartmentDTO.Id),
        nameof(DepartmentDTO.Name));
}

2 references
protected void ConfigureViewData(TaskDTO dto)
{
    ViewData[TasksViewDataParams.EmpoyeeId] = new SelectList(
        _EmployeeService.GetAll().Data,
        nameof(EmployeeDTO.Id),
        nameof(Employee.Name), dto);

    ViewData[TasksViewDataParams.DepartmentId] = new SelectList(_DepartmentService.GetAll().Data,
        nameof(DepartmentDTO.Id),
        nameof(DepartmentDTO.Name), dto);
}
```

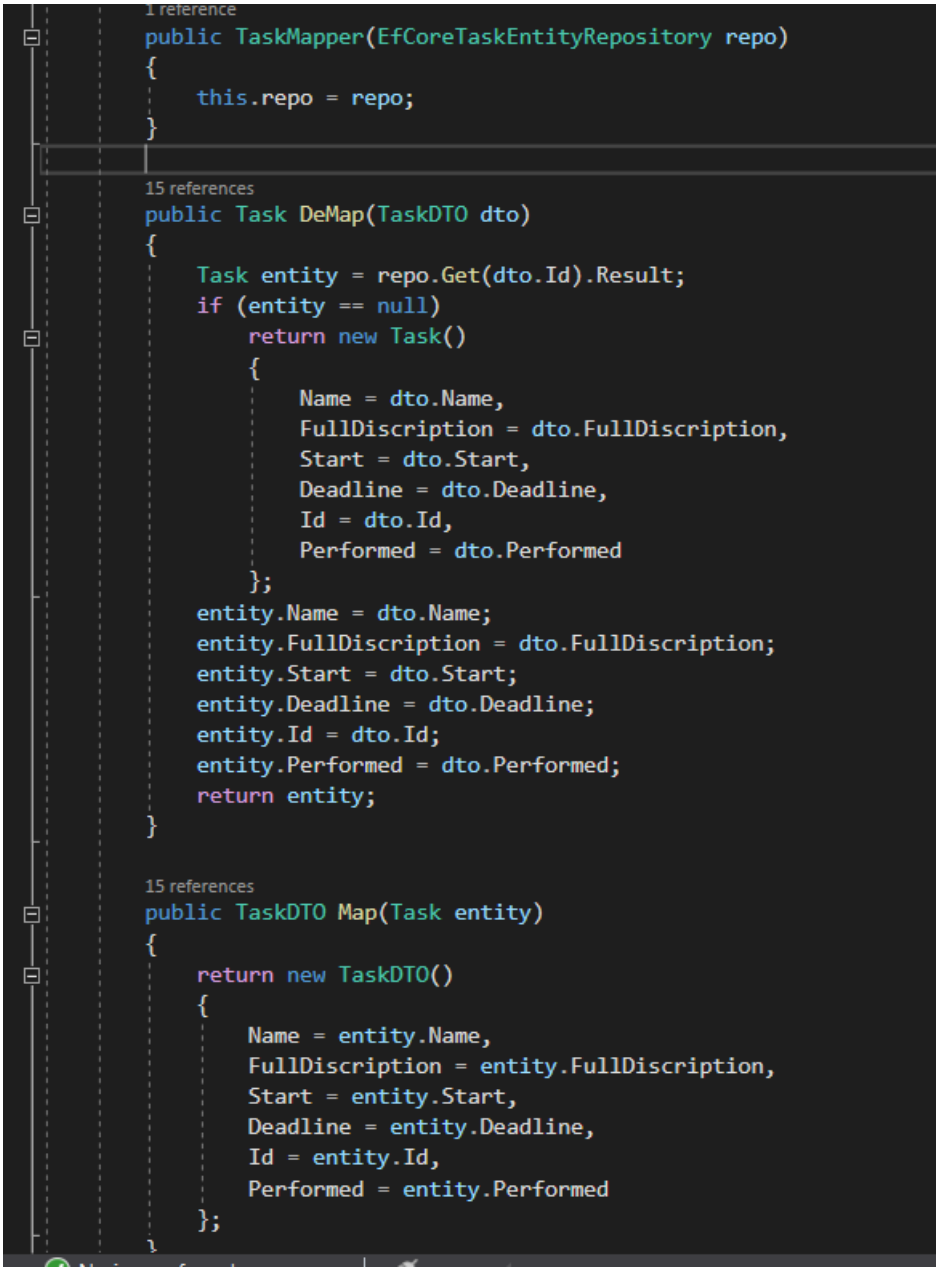
Сервіси у свою чергу мають доступ до маперів та репозиторієв



```
using System.Linq;
using System.Text;

namespace BL.Impl
{
    2 references
    public class DepartmentService : IDepartmentService
    {

        readonly DepartmentMapper Mapper;
        readonly EfCoreDepartmentRepository Repo;

        0 references
        public DepartmentService(ItCompanyContext context)
        {
            Repo = new EfCoreDepartmentRepository(context);
            Mapper = new DepartmentMapper(Repo);
        }

        DepartmentMapper.DepartmentMapper(EfCoreDepartr

        3 references
        public IDataResult<List<DepartmentDTO>> GetAll()
        {
            return new DataResult<List<DepartmentDTO>>()
            {
                Data = Repo.GetAll().Result.Select(e => Mapper.Map(e)).ToList(),
                Message = ResponseMessageType.None,
                ResponseStatusType = ResponseStatusType.Successed
            };
        }

        1 reference
        public IDataResult<DepartmentDTO> Get(int id)
        {
            return new DataResult<DepartmentDTO>()
            {
                Data = Mapper.Map(Repo.Get(id).Result),
                Message = ResponseMessageType.None,
                ResponseStatusType = ResponseStatusType.Successed
            };
        }

        1 reference
```

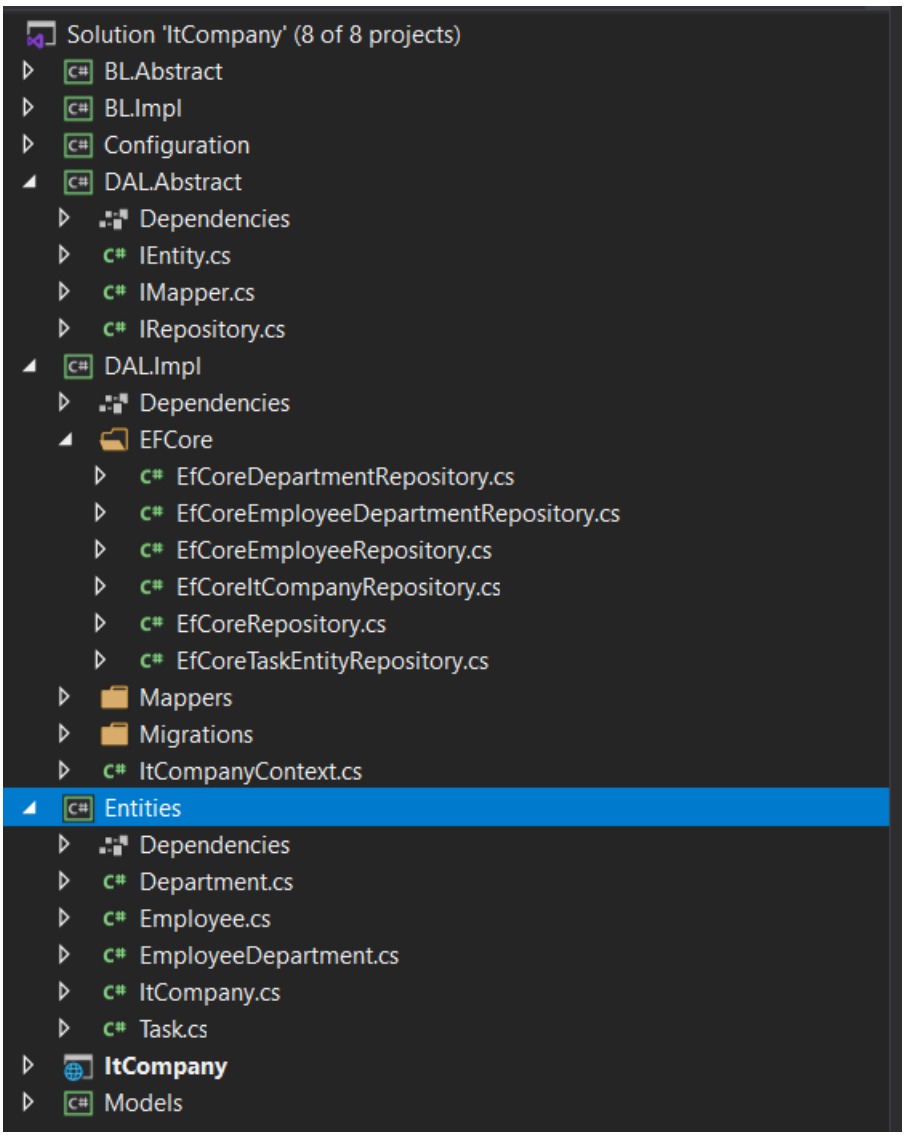
No issues found

Мапери слугують для того щоб перетворювати сутності на ДТО об’єкти та навпаки

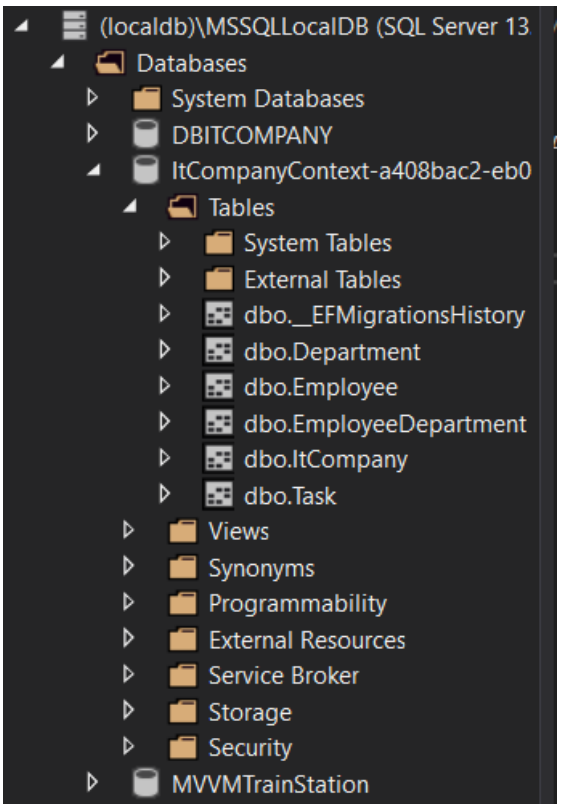


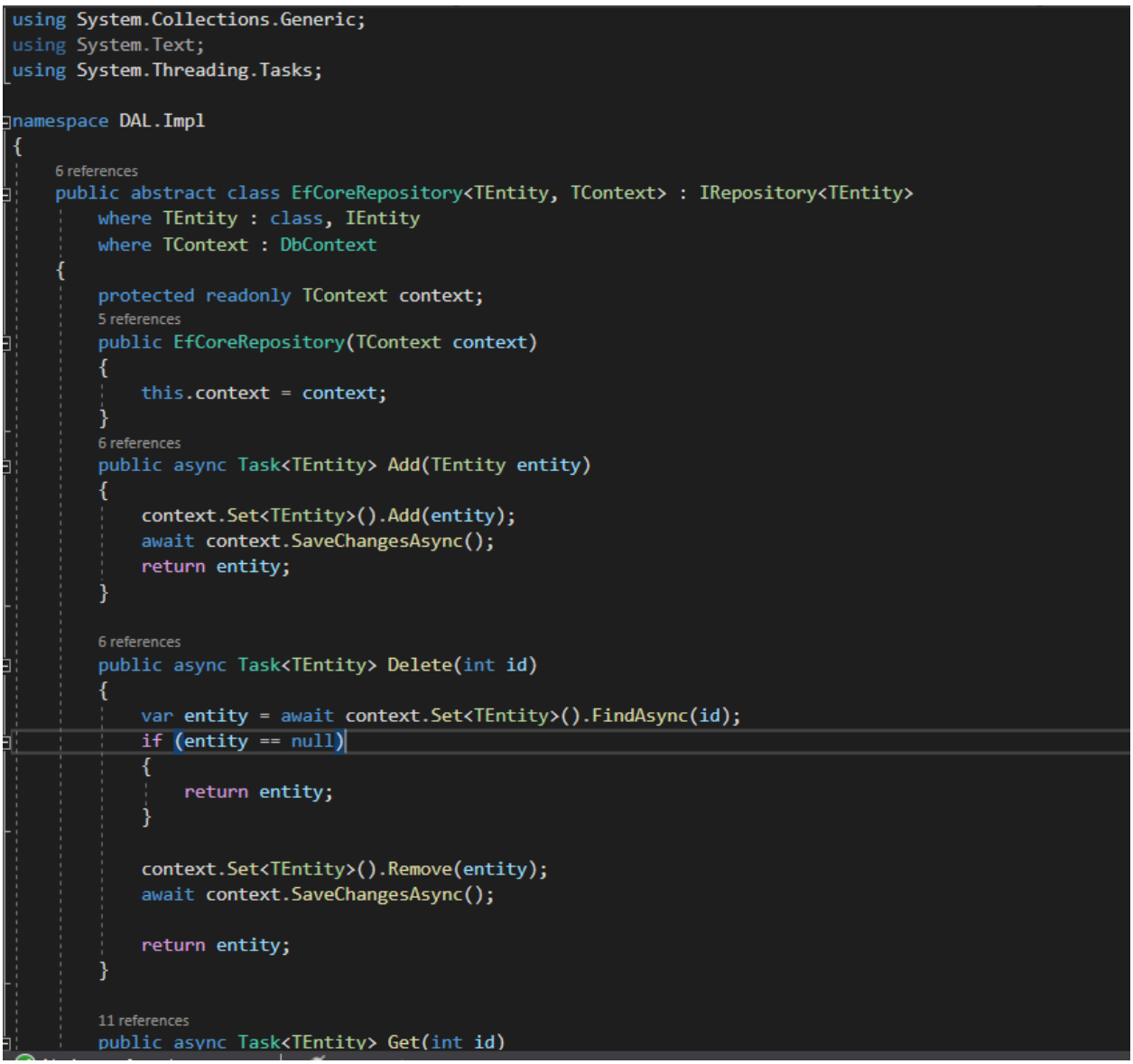
```
1 reference
public TaskMapper(EfCoreTaskEntityRepository repo)
{
    this.repo = repo;
}

15 references
public Task DeMap(TaskDTO dto)
{
    Task entity = repo.Get(dto.Id).Result;
    if (entity == null)
        return new Task()
        {
            Name = dto.Name,
            FullDiscription = dto.FullDiscription,
            Start = dto.Start,
            Deadline = dto.Deadline,
            Id = dto.Id,
            Performed = dto.Performed
        };
    entity.Name = dto.Name;
    entity.FullDiscription = dto.FullDiscription;
    entity.Start = dto.Start;
    entity.Deadline = dto.Deadline;
    entity.Id = dto.Id;
    entity.Performed = dto.Performed;
    return entity;
}

15 references
public TaskDTO Map(Task entity)
{
    return new TaskDTO()
    {
        Name = entity.Name,
        FullDiscription = entity.FullDiscription,
        Start = entity.Start,
        Deadline = entity.Deadline,
        Id = entity.Id,
        Performed = entity.Performed
    };
}
```

2. Реалізація репозиторія та CRUD операцій:
   Бібліотеки классів з інтрефейсами що реалізовані в Entities та EFCoreRepositories.



База данних створена за допомогою міграцій:


(localdb)\MSSQLLocalDB (SQL Server 13
Databases
System Databases
DBITCOMPANY
ItCompanyContext-a408bac2-eb0
Tables
System Tables
External Tables
dbo.__EFMigrationsHistory
dbo.Department
dbo.Employee
dbo.EmployeeDepartment
dbo.ItCompany
dbo.Task
Views
Synonyms
Programmability
External Resources
Service Broker
Storage
Security
MVVMTrainStation



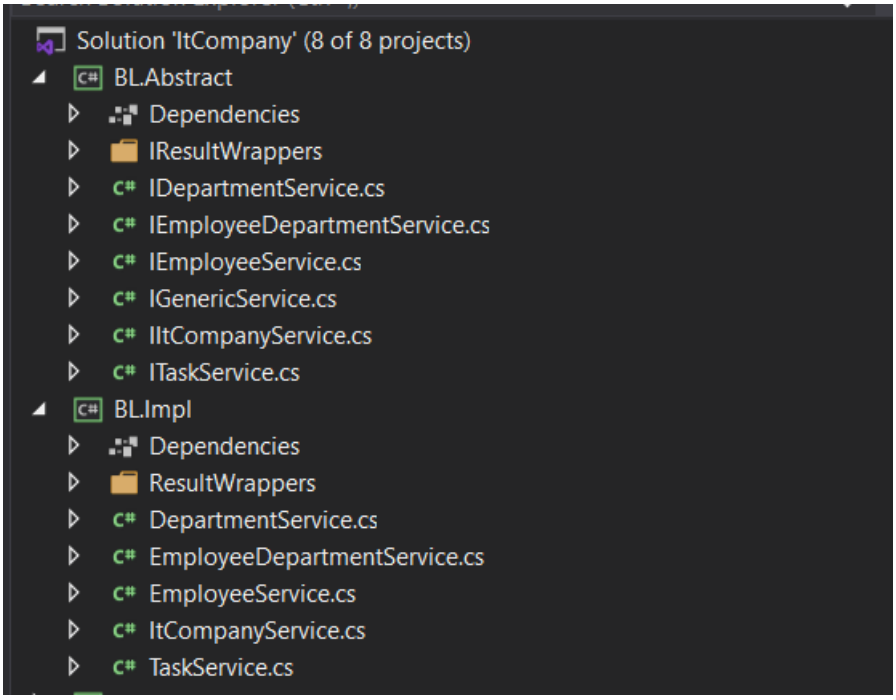
```
using System.Collections.Generic;
using System.Text;
using System.Threading.Tasks;

namespace DAL.Impl
{
    6 references
    public abstract class EfCoreRepository<TEntity, TContext> : IRepository<TEntity>
        where TEntity : class, IEntity
        where TContext : DbContext
    {
        protected readonly TContext context;
        5 references
        public EfCoreRepository(TContext context)
        {
            this.context = context;
        }
        6 references
        public async Task<TEntity> Add(TEntity entity)
        {
            context.Set<TEntity>().Add(entity);
            await context.SaveChangesAsync();
            return entity;
        }

        6 references
        public async Task<TEntity> Delete(int id)
        {
            var entity = await context.Set<TEntity>().FindAsync(id);
            if (entity == null)
            {
                return entity;
            }

            context.Set<TEntity>().Remove(entity);
            await context.SaveChangesAsync();

            return entity;
        }

        11 references
        public async Task<TEntity> Get(int id)
```

```
namespace DAL.Impl
{
    7 references
    public class EfCoreTaskEntityRepository : EfCoreRepository<Entities.Task, ItCompanyContext>
    {
        1 reference
        public EfCoreTaskEntityRepository(ItCompanyContext context) : base(context)
        {

        }
        // We can add new methods specific to the movie repository here in the future
    }

}
```

class DAL.Impl.ItCompanyContext

1 Створити сервіси scoped and transient

Створені сервіси для використання DTO.

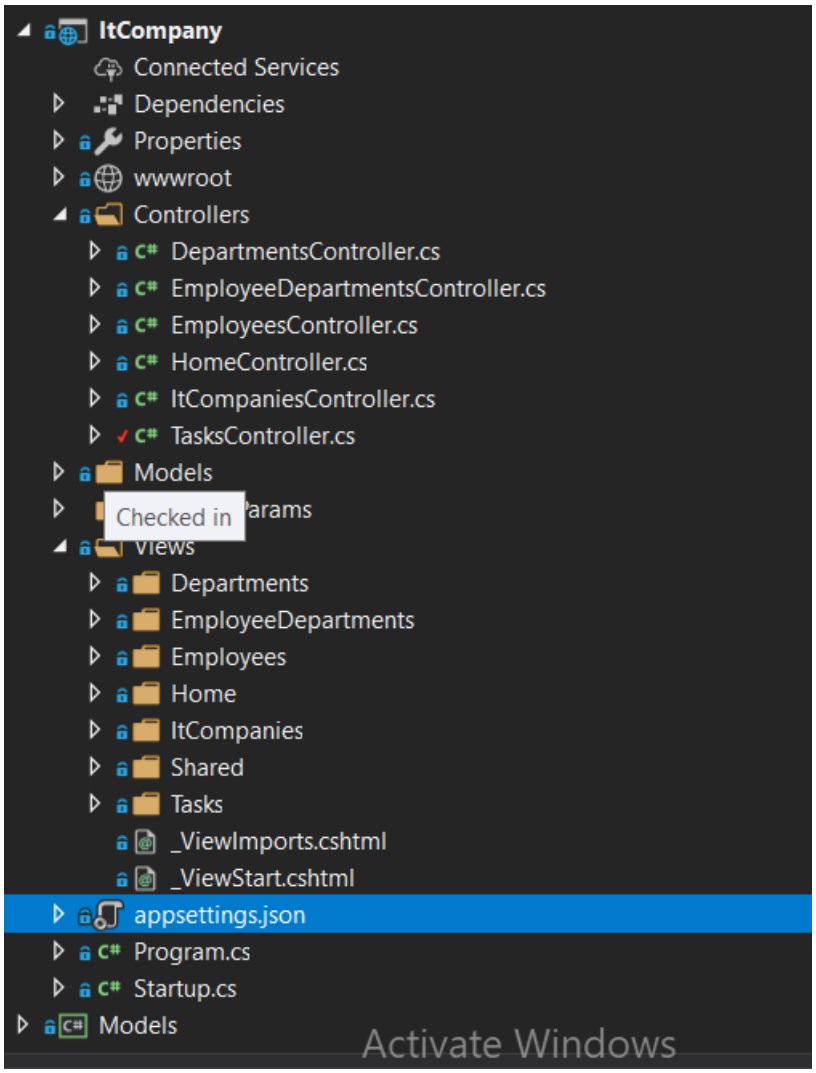


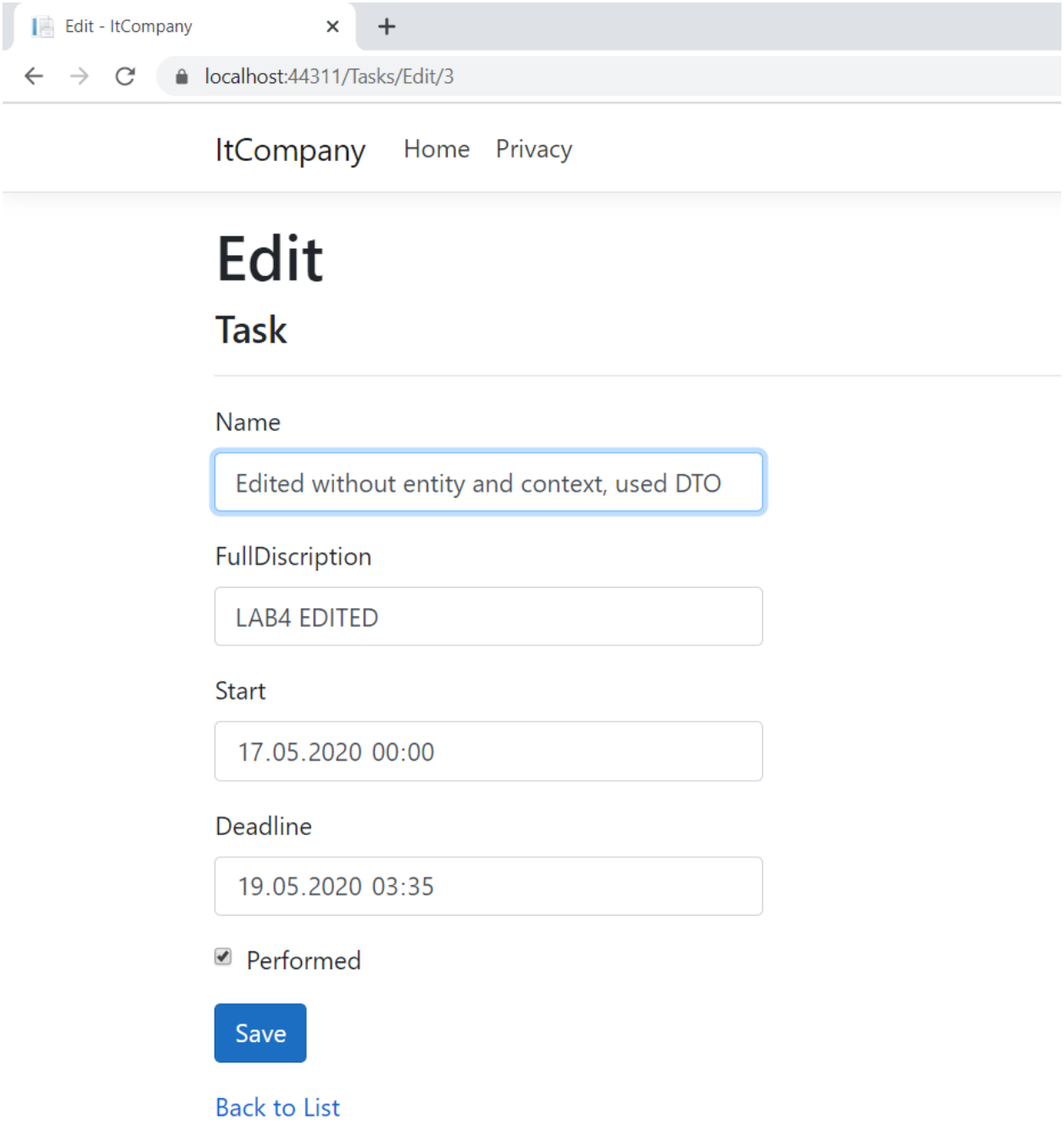

```
services.AddScoped<EfCoreTaskEntityRepository>();
services.AddScoped<EfCoreEmployeeRepository>();
services.AddScoped<EfCoreEmployeeDepartmentRepository>();
services.AddScoped<EfCoreItCompanyReposi
services.AddScoped<EfCoreDepartmentRepos

services.AddTransient<IDepartmentService, DepartmentService>();
services.AddTransient<IEmployeeDepartmentService, EmployeeDepartmentService>();
services.AddTransient<IEmployeeService, EmployeeService>();
services.AddTransient<IItCompanyService, ItCompanyService>();
services.AddTransient<ITaskService, TaskService>();
```

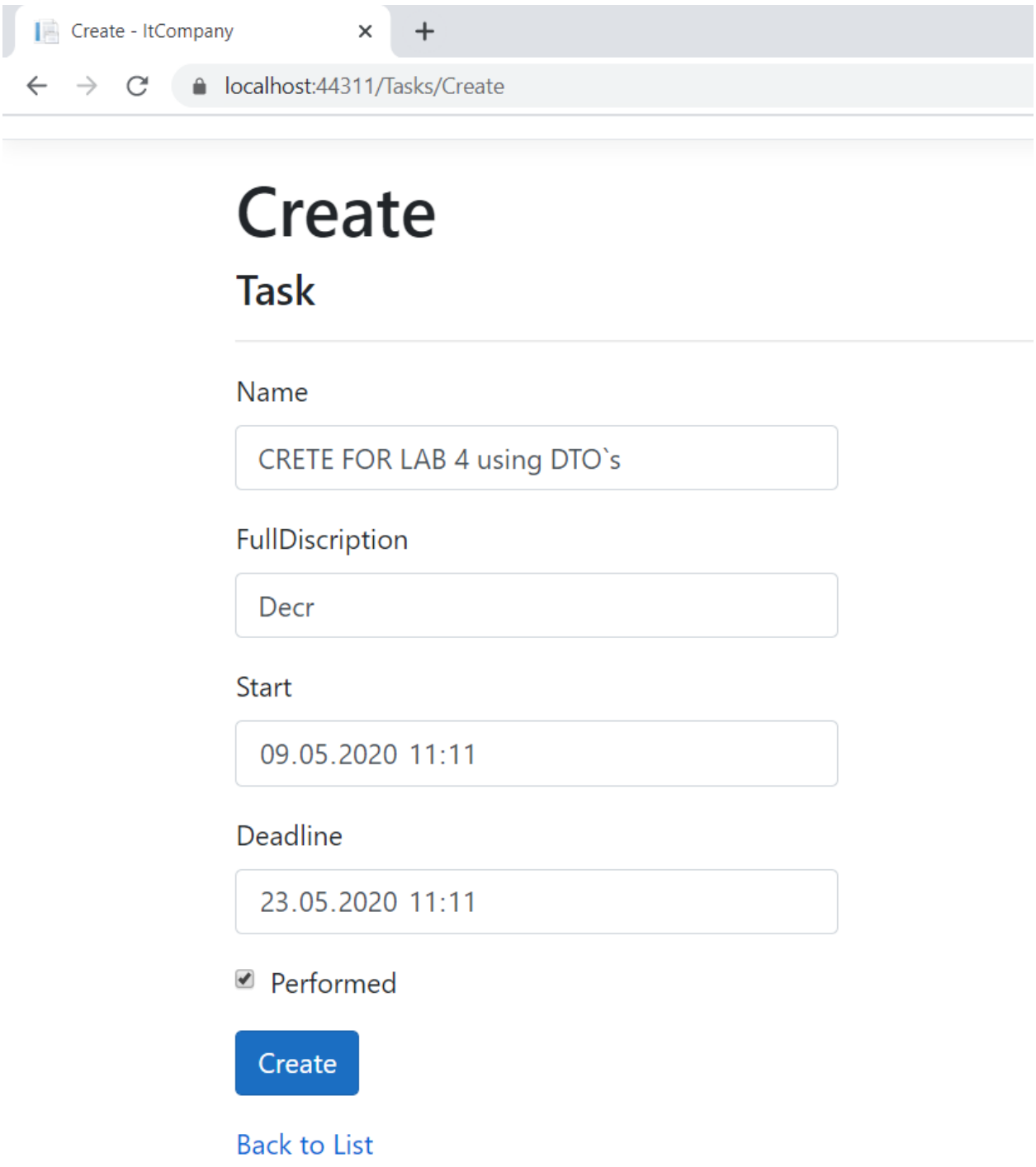
class DAL.Impl.EfCoreEmployeeDepartmentRepository

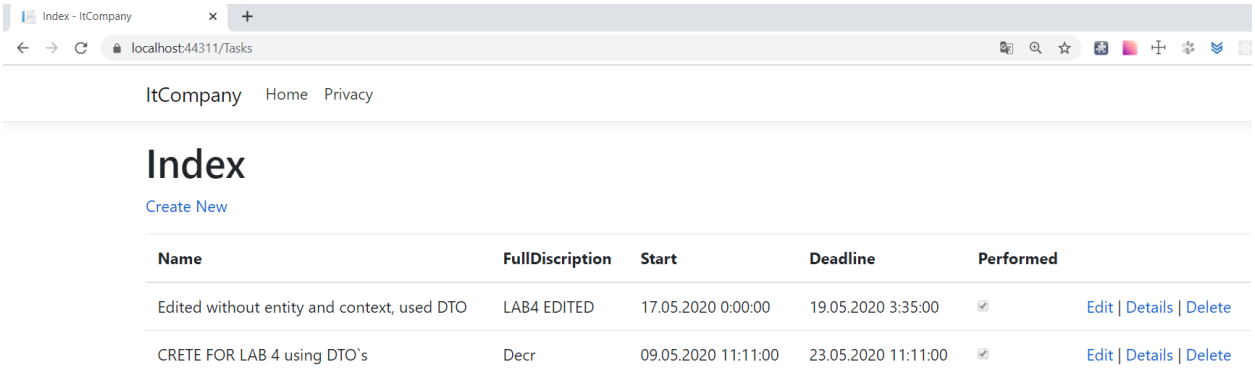
4. Строчка з'єднання з базою даних повинна міститись в файлі конфігурації:

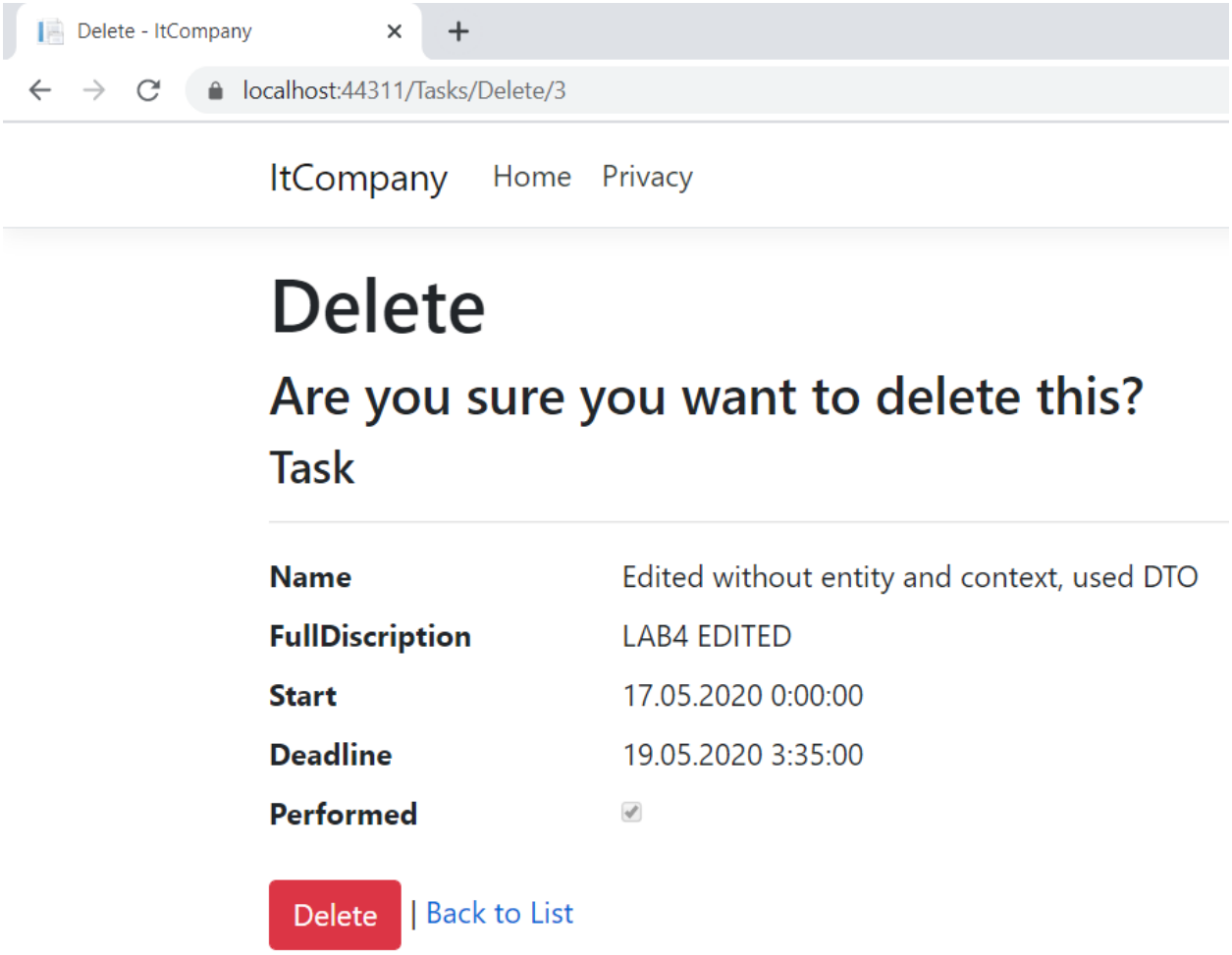
5. Користувацький інтерфейс, що дозволяє виконувати CRUD операції з сутностями:

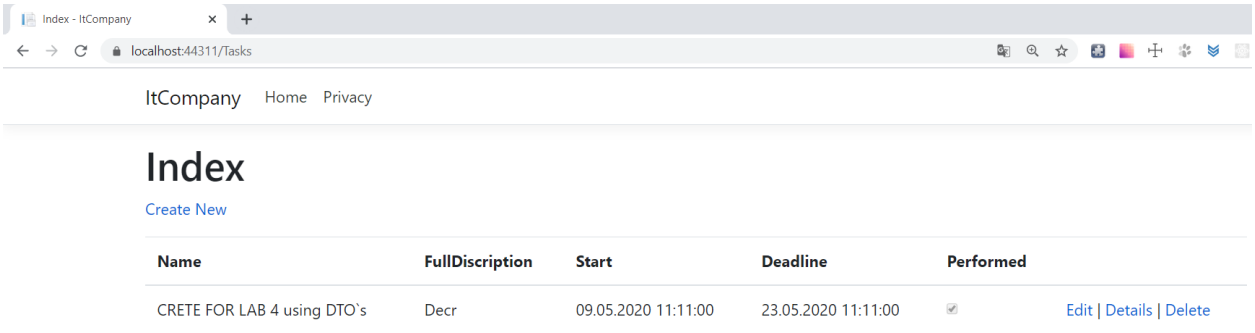



Edit - ItCompany
localhost:44311/Tasks/Edit/3
ItCompany
Home
Privacy
Edit
Task
Name
Edited without entity and context, used DTO
FullDiscription
LAB4 EDITED
Start
17.05.2020 00:00
Deadline
19.05.2020 03:35
Performed
Save
Back to List


Create - ItCompany
localhost:44311/Tasks/Create
Create
Task
Name
CRETE FOR LAB 4 using DTO`s
FullDiscription
Decr
Start
09.05.2020 11:11
Deadline
23.05.2020 11:11
Performed
Create
Back to List


Index - ItCompany
localhost:44311/Tasks
ItCompany
Home
Privacy
Index
Create New
Name
FullDiscription
Start
Deadline
Performed
Edited without entity and context, used DTO
LAB4 EDITED
17.05.2020 0:00:00
19.05.2020 3:35:00
Edit | Details | Delete
CRETE FOR LAB 4 using DTO`s
Decr
09.05.2020 11:11:00
23.05.2020 11:11:00
Edit | Details | Delete





Висновок: Я навчився реалізувати архітектуру паттерна Repository. Повторив Dependency Injection. Освоїв розмежування архітектурних шарів додатку. Навчився ділити рішення на проекти. Навчився створювати бази данних і міграцій. Створив DTO об`єкти, щоб прибрати роботу контроллерів напряму з контекстом бази даних