Лабораторна робота №4

CRUD-операції в EF Core. Валідація даних.

Мета: набути навичок роботи з ORM, навчитися реалізовувати операції створення, читання, оновлення та видалення даних з використанням EF Core, набути навичок роботи з механізмами валідації даних.

Хід роботи

Завдання 1. Використовуючи підхід Code First, створити модель, яка разом з існуючою моделлю, утворюватиме зв'язок один-до-багатьох.

Лістинг моделі Application (заявка на вакансію):

```
using JobPortal.Models;
using Microsoft.AspNetCore.Mvc.ModelBinding.Validation;
using System.ComponentModel.DataAnnotations;

namespace JobPortal.Models
{
    public class Application
    {
        public int Id { get; set; }

        public string FullName { get; set; }

        public string Email { get; set; }

        public string ResumeText { get; set; }

        public DateTime DateApplied { get; set; } = DateTime.Now;

        public int JobId { get; set; }

        public Job Job { get; set; }
    }
}
```

Лістинг контролера ApplicationController:

```
using JobPortal.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Newtonsoft.Json;

public class ApplicationController : Controller
{
    private readonly IPortalRepository _repository;
```

```
public ApplicationController(IPortalRepository repo)
{
    _repository = repo;
}

// READ: список
public async Task<IActionResult> Index()
{
    var applications = _repository.Applications
        .Include(a => a.Job); //
    return View(await applications.ToListAsync());
}

// READ: деталі
public async Task<IActionResult> Details(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var application = await _repository.Applications
        .Include(a => a.Job)
        .FirstOrDefaultAsync(m => m.Id == id);

    if (application == null)
    {
        return NotFound();
    }

    return View(application);
}


// CREATE (GET)
public IActionResult Create()
{
    ViewBag.Jobs = _repository.Jobs.ToList();
    return View("Edit", new Application());
}

// CREATE (POST)
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Create(Application app)
{
    if (ModelState.IsValid)
    {
        _repository.CreateApplication(app);
        return RedirectToAction("Index");
    }
    ViewBag.Jobs = _repository.Jobs.ToList();
    return View("Edit", app);
}

// UPDATE (GET)
public IActionResult Edit(int id)
{
    var app = _repository.GetApplicationById(id);
    if (app == null) return NotFound();
    ViewBag.Jobs = _repository.Jobs.ToList();
    return View(app);
}

// UPDATE (POST)
[HttpPost]
```

```
        [ValidateAntiForgeryToken]
        public IActionResult Edit(Application app)
        {
            if (ModelState.IsValid)
            {
                _repository.UpdateApplication(app);
                return RedirectToAction("Index");
            }
            ViewBag.Jobs = _repository.Jobs.ToList();
            return View(app);
        }

        // DELETE (GET)
        public IActionResult Delete(int id)
        {
            var app = _repository.Applications
                .Include(a => a.Job)
                .FirstOrDefault(a => a.Id == id);

            if (app == null) return NotFound();

            return View(app);
        }

        // DELETE (POST)
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public IActionResult DeleteConfirmed(int id)
        {
            var app = _repository.GetApplicationById(id);
            if (app != null)
            {
                _repository.DeleteApplication(app);
            }
            return RedirectToAction("Index");
        }


        private const string SessionKey = "ApplicationForm";

        public IActionResult Apply()
        {
            // Якщо є дані в сесії — відновлюємо
            var json = HttpContext.Session.GetString(SessionKey);
            ApplicationForm model = string.IsNullOrEmpty(json)
                ? new ApplicationForm()
                : JsonConvert.DeserializeObject<ApplicationForm>(json);

            return View(model);
        }

        [HttpPost]
        public IActionResult Apply(ApplicationForm model)
        {
            // Зберігаємо поточний стан у сесії
            var json = JsonConvert.SerializeObject(model);
            HttpContext.Session.SetString(SessionKey, json);

            ViewBag.Message = "Заявку збережено в сесії. Можна продовжити пізніше.";
            return View(model);
        }

        public IActionResult Submit()
        {
            // Отримуємо фінальні дані
            var json = HttpContext.Session.GetString(SessionKey);
```

```
        if (json == null)
        {
            return RedirectToAction("Apply");
        }

        var model = JsonConvert.DeserializeObject<ApplicationForm>(json);

        // очищаємо сесію після відправки
        HttpContext.Session.Remove(SessionKey);

        ViewBag.Message = "Заявку успішно подано!";
        return View("Result", model);
    }
}
```

Завдання 2. Реалізувати CRUD (Create, Read, Update, Delete) операції для обох моделей, враховуючи наступні вимоги:

- Реалізувати представлення з формами для операцій Create та Update.

  Лістинг представлення для Application, яке відповідає за створення або редагування заявки:

```
@model JobPortal.Models.Application
@{
    ViewData["Title"] = Model.Id == 0 ? "Нова заявка" : "Редагування заявки";
}

<h2 class="mb-4">@ViewData["Title"]</h2>

<form asp-action="@(Model.Id == 0 ? "Create" : "Edit")" method="post" class="p-4
border rounded bg-light shadow-sm">
    <div asp-validation-summary="All" class="text-danger mb-3"></div>

    <div class="mb-3">
        <label asp-for="FullName" class="form-label fw-bold"></label>
        <input asp-for="FullName" class="form-control" />
        <span asp-validation-for="FullName" class="text-danger"></span>
    </div>

    <div class="mb-3">
        <label asp-for="Email" class="form-label fw-bold"></label>
        <input asp-for="Email" class="form-control" />
        <span asp-validation-for="Email" class="text-danger"></span>
    </div>

    <div class="mb-3">
        <label asp-for="ResumeText" class="form-label fw-bold"></label>
        <textarea asp-for="ResumeText" class="form-control" rows="4"></textarea>
        <span asp-validation-for="ResumeText" class="text-danger"></span>
    </div>

    <div class="mb-3">
        <label class="form-label fw-bold">Вакансія</label>
        <select  asp-for="JobId"  asp-items="@(new  SelectList(ViewBag.Jobs,  "Id",
"Title"))" class="form-select"></select>
        <span asp-validation-for="JobId" class="text-danger"></span>
    </div>

    <button type="submit" class="btn btn-success me-2">Зберегти</button>
    <a asp-action="Index" class="btn btn-secondary">Назад</a>
</form>
```

```
@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}
```

- Для виконання операції Read реалізувати окрему Details сторінку, на якій будуть виводитися всі дані про сутність.

  Лістинг Application/Details:

```
@model JobPortal.Models.Application

@{
    ViewData["Title"] = "Деталі заявки";
}

<h1>Деталі заявки</h1>

<div>
    <h4>Заявка</h4>
    <hr />
    <dl class="row">
        <dt class="col-sm-2">
            Ім’я заявника
        </dt>
        <dd class="col-sm-10">
            @Model.FullName
        </dd>

        <dt class="col-sm-2">
            Email
        </dt>
        <dd class="col-sm-10">
            @Model.Email
        </dd>

        <dt class="col-sm-2">
            Назва вакансії
        </dt>
        <dd class="col-sm-10">
            @Model.Job?.Title
        </dd>
    </dl>
</div>

<div>
    <a      asp-action="Edit"      asp-route-id="@Model.Id"      class="btn      btn-
warning">Редагувати</a> |
    <a asp-action="Index" class="btn btn-secondary">Повернутись до списку</a>
</div>
```

- Перед виконанням операції Delete запитати користувача, підтвердження видалення запису.

  Лістинг Application/Delete (з підтвердженням відправки змін):

```
@model JobPortal.Models.Application

@{
    ViewData["Title"] = "Видалення заявки";
}

<h2 class="mb-3">Підтвердження видалення</h2>
```

```
<div class="alert alert-warning">
    Ви дійсно хочете видалити заявку від
    <strong>@Model.FullName</strong>
    на вакансію <strong>@Model.Job?.Title</strong>?
</div>

<form asp-action="Delete" method="post">
    <input type="hidden" asp-for="Id" />
    <button type="submit" class="btn btn-danger">Видалити</button>
    <a asp-action="Index" class="btn btn-secondary">Скасувати</a>
</form>
```

Результат виконання:

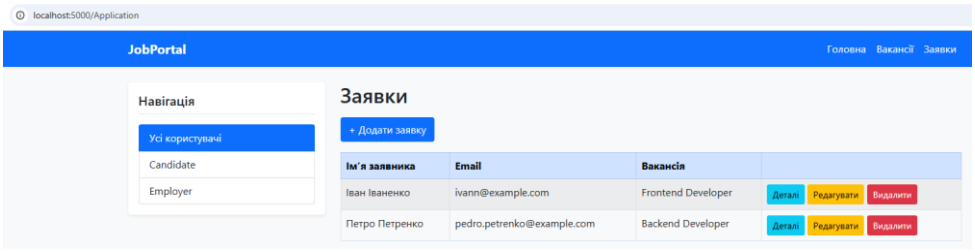

Рис. 4.2.1. Сторінка заявок на вакансії

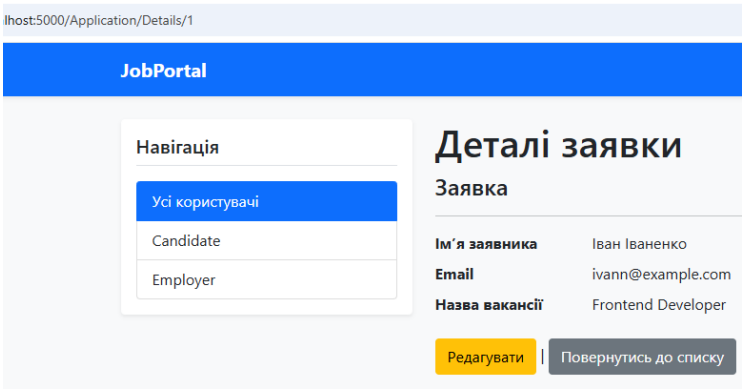

Рис. 4.2.2. Сторінка деталей конкретної заявки

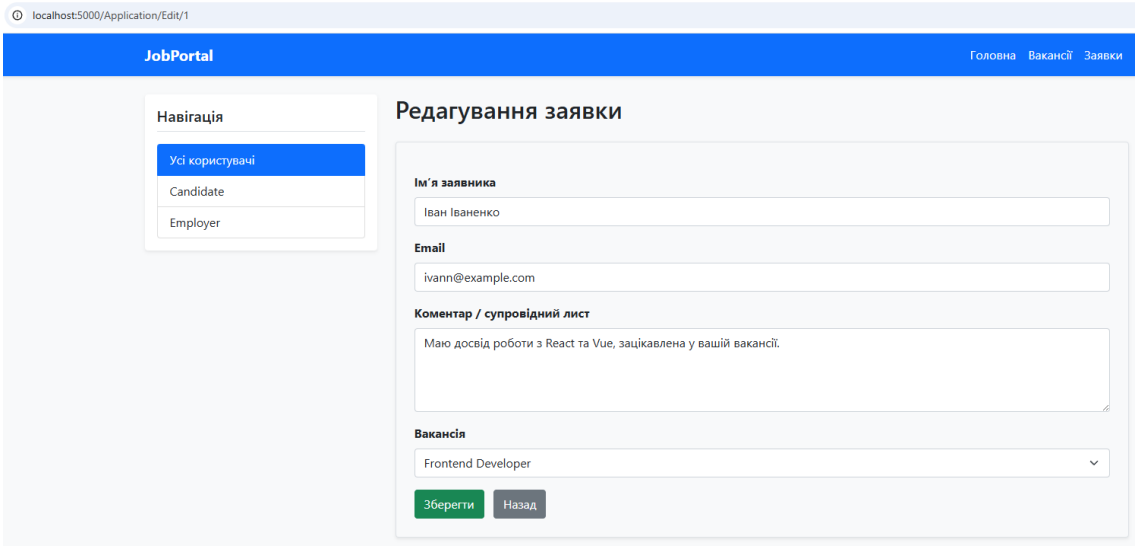

Рис. 4.2.3. Сторінка редагування даних заявки

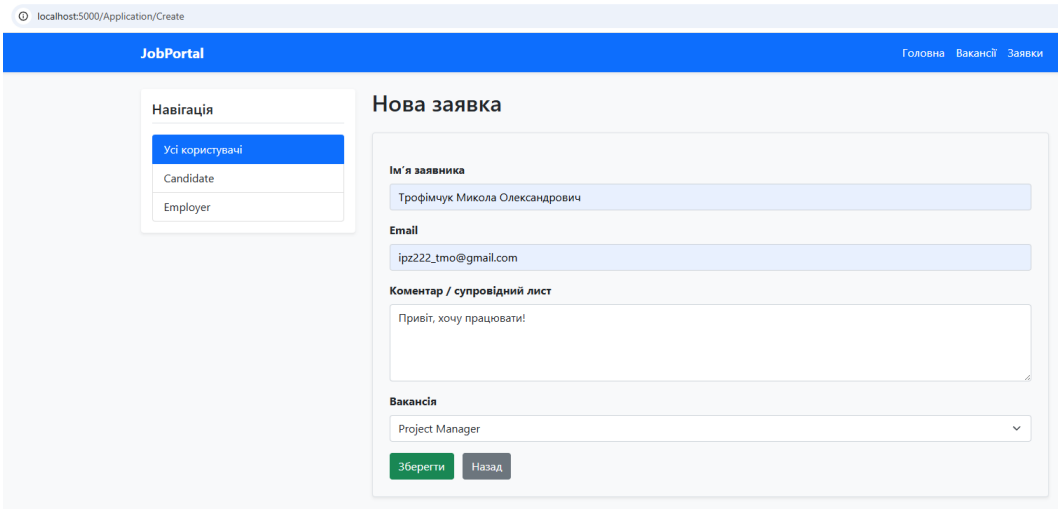

Рис. 4.2.4. Сторінка створення нової заявки

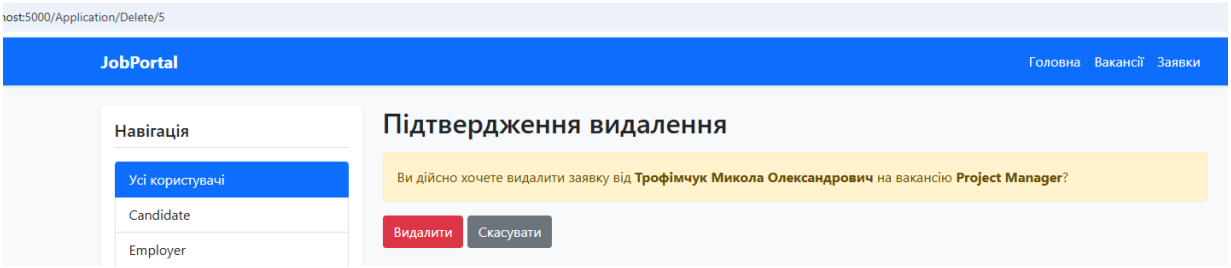

Рис. 4.2.5. Сторінка видалення заявки (підтвердження)

Завдання 3. Додати серверну валідацію для перевірки введених користувачем даних перед виконанням CRUD-операцій. Використати DataAnnotations для реалізації перевірки введених даних.

Лістинг Models/Application з реалізованою валідацією:

```
using JobPortal.Models;
using Microsoft.AspNetCore.Mvc.ModelBinding.Validation;
using System.ComponentModel.DataAnnotations;

namespace JobPortal.Models
{
    public class Application
    {
        public int Id { get; set; }

        [Required(ErrorMessage = "Вкажіть ім'я заявника")]
        [StringLength(100, ErrorMessage = "Ім'я не може перевищувати 100 символів")]
        [Display(Name = "Ім'я заявника")]
        public string FullName { get; set; }

        [Required(ErrorMessage = "Вкажіть адресу електронної пошти")]
        [EmailAddress(ErrorMessage = "Невірний формат електронної пошти")]
        [Display(Name = "Email")]
        public string Email { get; set; }

        [StringLength(500, ErrorMessage = "Коментар не може перевищувати 500
символів")]
        [Display(Name = "Коментар / супровідний лист")]
```

```
        public string ResumeText { get; set; }

        public DateTime DateApplied { get; set; } = DateTime.Now;

        [Required(ErrorMessage = "Оберіть вакансію")]
        [Display(Name = "Вакансія")]
        public int JobId { get; set; }

        [ValidateNever]
        public Job Job { get; set; }
    }
}
```

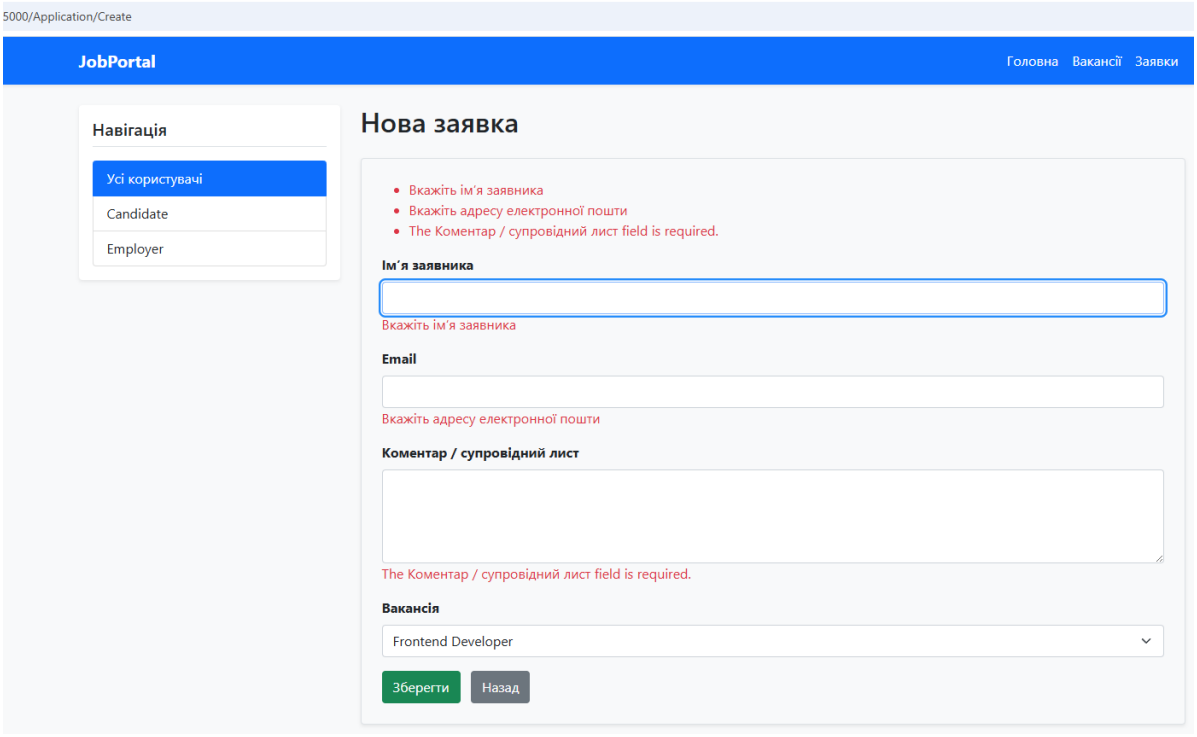

Рис. 4.3.1. Валідація форми створення заявки

Висновок: в ході виконання лабораторної роботи було набуто навичок роботи з ORM, навчено реалізовувати операції створення, читання, оновлення та видалення даних з використанням EF Core, набуто навичок роботи з механізмами валідації даних. Роботу виконано в повному обсязі.