**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.

**Листинг Model Company:**

using System.ComponentModel.DataAnnotations;

namespace SurveyPortal.Models.Company

{

public class Company

{

public long CompanyID { get; set; }

[Required(ErrorMessage = "Please enter a company name")]

public string Name { get; set; } = string.Empty;

[Required(ErrorMessage = "Please enter a company description")]

public string? Description { get; set; }

[Required(ErrorMessage = "Please enter a Country of origin.")]

public string? Country { get; set; }

public DateTime CreatedAt { get; set; } = DateTime.UtcNow.AddTicks(-(DateTime.UtcNow.Ticks % TimeSpan.TicksPerSecond));

public ICollection<Survey.Survey.Survey>? Surveys { get; set; } = new List<Survey.Survey.Survey>();

}

}

**Листинг Model Survey(основна модель):**

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace SurveyPortal.Models.Survey.Survey

{

public class Survey

{

[Key]

public long? SurveyID { get; set; }

[Required(ErrorMessage = "Please enter a survey title")]

public string Title { get; set; } = string.Empty;

[Required(ErrorMessage = "Please enter a disription")]

public string Description { get; set; } = string.Empty;

public DateTime CreatedAt { get; set; } = DateTime.UtcNow.AddTicks(-(DateTime.UtcNow.Ticks % TimeSpan.TicksPerSecond));

[Required(ErrorMessage = "Please enter a category")]

public string? Category { get; set; }

public bool IsActive { get; set; } = true;

[Required(ErrorMessage = "Please chose a company of survey")]

public long? CompanyID { get; set; }

public Company.Company? Company { get; set; }

public ICollection<SurveyAnswers> SurveyAnswers { get; set; } = new List<SurveyAnswers>();

}

}

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

1. Реалізувати представлення з формами для операцій Create та Update
2. Для виконання операції Read реалізувати окрему Details сторінку, на якій будуть виводитися всі дані про сутність.
3. Перед виконанням операції Delete запитати користувача, підтвердження видалення запису.

**Листинг HomeController після виконання лабораторної роботи:**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using SurveyPortal.Models.Survey;

using SurveyPortal.Models.Survey.Survey;

using SurveyPortal.Models.ViewModels;

namespace SurveyPortal.Controllers

{

public class HomeController : Controller

{

private ISurveyRepository \_repository;

public const int PageSize = 4;

public HomeController(ISurveyRepository repository)

{

\_repository = repository;

}

[HttpPost]

[ValidateAntiForgeryToken]

public IActionResult Create(Survey model)

{

if (!ModelState.IsValid)

{

return View(model);

}

model.CreatedAt = DateTime.UtcNow.AddTicks(-(DateTime.UtcNow.Ticks % TimeSpan.TicksPerSecond));

model.Title = model.Title.Trim();

model.Description = model.Description.Trim();

model.Category = model.Category?.Trim();

model.IsActive = true;

\_repository.AddSurvey(model);

return RedirectToAction("Details", new { id = model.SurveyID });

}

public IActionResult Create()

{

var companies = \_repository.Companies

.Select(c => new SelectListItem

{

Value = c.CompanyID.ToString(),

Text = c.Name

}).ToList();

ViewBag.Companies = companies;

return View();

}

[HttpDelete]

public IActionResult Delete(long id)

{

var survey = \_repository.Surveys.FirstOrDefault(s => s.SurveyID == id);

if (survey == null)

{

return NotFound();

}

\_repository.DeleteSurvey(survey);

return RedirectToAction("Index");

}

[HttpPost]

[ValidateAntiForgeryToken]

public IActionResult Update(Survey model)

{

if(!ModelState.IsValid)

{

return View(model);

}

var existingSurvey = \_repository.Surveys.FirstOrDefault(s => s.SurveyID == model.SurveyID);

if (existingSurvey == null)

{

return NotFound();

}

existingSurvey.Title = model.Title;

existingSurvey.Description = model.Description;

existingSurvey.Category = model.Category;

existingSurvey.IsActive = model.IsActive;

existingSurvey.CreatedAt = DateTime.UtcNow;

\_repository.SaveSurvey(existingSurvey);

return RedirectToAction("Details", new { id = model.SurveyID });

}

public IActionResult Update(long id)

{

var survey = \_repository.Surveys.FirstOrDefault(s => s.SurveyID == id);

if (survey == null)

{

return NotFound();

}

return View(survey);

}

private void PopulateNavigationData()

{

var categories = \_repository.Surveys

.Select(x => x.Category)

.Where(x => x != null)

.Distinct()

.OrderBy(x => x)

.ToList();

var companies = \_repository.Companies

.Select(x => new { x.CompanyID, x.Name })

.OrderBy(x => x.Name)

.ToList();

ViewBag.Categories = categories;

ViewBag.Companies = companies;

ViewBag.SelectedCategory = RouteData?.Values["category"] ?? Request.Query["category"].ToString();

ViewBag.SelectedCompany = RouteData?.Values["company"] ?? Request.Query["company"].ToString();

}

public IActionResult Index(string? category, long? company, int page = 1)

{

IQueryable<Survey> query = \_repository.Surveys.Include(s => s.Company);

if (!string.IsNullOrEmpty(category))

{

query = query.Where(s => s.Category == category);

}

if (company.HasValue)

{

query = query.Where(s => s.CompanyID == company.Value);

}

var model = new SurveyListViewModel

{

Surveys = query

.OrderByDescending(s => s.CreatedAt)

.Skip((page - 1) \* PageSize)

.Take(PageSize),

PagingInfo = new PagingInfo

{

CurrentPage = page,

ItemsPerPage = PageSize,

TotalItems = query.Count()

},

CurrentCategory = category,

CurrentCompany = company

};

PopulateNavigationData();

return View(model);

}

public IActionResult Details(long id)

{

var survey = \_repository.Surveys

.Include(s => s.Company)

.FirstOrDefault(s => s.SurveyID == id);

if (survey == null)

{

return NotFound();

}

return View(survey);

}

[HttpPost]

public IActionResult SaveSurveyAnswer(long SurveyID, string Answer)

{

var survey = \_repository.Surveys.FirstOrDefault(s => s.SurveyID == SurveyID);

if (survey == null || string.IsNullOrWhiteSpace(Answer))

{

return NotFound();

}

\_repository.AddAnswer(new SurveyAnswers

{

SurveyID = SurveyID,

Answer = Answer,

AnsweredAt = DateTime.UtcNow

});

TempData["Message"] = "Thank you for your answer!";

return RedirectToAction("Details", new { id = SurveyID });

}

public IActionResult SaveDraftAnswer(long surveyId, [FromBody] DraftDto dto)

{

var key = $"draft\_{surveyId}";

HttpContext.Session.SetString(key, dto.Answer ?? "");

return Ok();

}

public class DraftDto

{

public string? Answer { get; set; }

}

}

}

**Листинг CompanyController після виконання лабораторної роботи:**

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using SurveyPortal.Models.Company;

using SurveyPortal.Models.Survey;

using SurveyPortal.Models.Survey.Survey;

namespace SurveyPortal.Controllers

{

public class CompaniesController : Controller

{

private readonly ISurveyRepository \_repository;

public const int PageSize = 8;

public CompaniesController(ISurveyRepository repository)

{

\_repository = repository;

}

private void PopulateNavigationData()

{

var categories = \_repository.Surveys

.Select(x => x.Category)

.Where(x => x != null)

.Distinct()

.OrderBy(x => x)

.ToList();

var companies = \_repository.Companies

.Select(x => new { x.CompanyID, x.Name })

.OrderBy(x => x.Name)

.ToList();

ViewBag.Categories = categories;

ViewBag.Companies = companies;

ViewBag.SelectedCategory = RouteData?.Values["category"] ?? Request.Query["category"].ToString();

ViewBag.SelectedCompany = RouteData?.Values["company"] ?? Request.Query["company"].ToString();

}

public IActionResult Index()

{

var companies = \_repository.Companies

.OrderBy(c => c.Name)

.ToList();

PopulateNavigationData();

return View(companies);

}

public IActionResult Details(long id)

{

var company = \_repository.Companies

.Include(c => c.Surveys)

.FirstOrDefault(c => c.CompanyID == id);

if (company == null)

{

return NotFound();

}

return View(company);

}

public IActionResult Create()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public IActionResult Create(Company company)

{

if (ModelState.IsValid)

{

company.CreatedAt = DateTime.UtcNow.AddTicks(-(DateTime.UtcNow.Ticks % TimeSpan.TicksPerSecond));

company.Name = company.Name.Trim();

company.Description = company.Description?.Trim();

company.Country = company.Country?.Trim();

\_repository.AddCompany(company);

return RedirectToAction(nameof(Details), new { id = company.CompanyID });

}

return View(company);

}

public IActionResult Edit(long id)

{

var company = \_repository.Companies.FirstOrDefault(c => c.CompanyID == id);

if (company == null)

{

return NotFound();

}

return View(company);

}

[HttpPost]

[ValidateAntiForgeryToken]

public IActionResult Edit(Company company)

{

if (ModelState.IsValid)

{

var existingCompany = \_repository.Companies.FirstOrDefault(c => c.CompanyID == company.CompanyID);

if (existingCompany == null)

{

return NotFound();

}

existingCompany.Name = company.Name.Trim();

existingCompany.Description = company.Description?.Trim();

existingCompany.Country = company.Country?.Trim();

\_repository.SaveCompany(existingCompany);

return RedirectToAction(nameof(Details), new { id = company.CompanyID });

}

return View(company);

}

public IActionResult Delete(long id)

{

var company = \_repository.Companies

.Include(c => c.Surveys)

.FirstOrDefault(c => c.CompanyID == id);

if (company == null)

{

return NotFound();

}

return View(company);

}

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public IActionResult DeleteConfirmed(long id)

{

var company = \_repository.Companies.FirstOrDefault(c => c.CompanyID == id);

if (company == null)

{

return NotFound();

}

\_repository.DeleteCompany(company);

return RedirectToAction(nameof(Index));

}

}

}

**Листинг EFSurveyRepository після виконання лабораторної роботи:**

using Microsoft.EntityFrameworkCore;

namespace SurveyPortal.Models.Survey.Survey

{

public class EFSurveyRepository : ISurveyRepository

{

private readonly SurveyDbContext \_context;

public EFSurveyRepository(SurveyDbContext context)

{

\_context = context;

}

public IQueryable<Survey> Surveys => \_context.Surveys.Include(s => s.SurveyAnswers);

public IQueryable<Company.Company> Companies => \_context.Companies.Include(c => c.Surveys);

public void AddAnswer(SurveyAnswers answer)

{

\_context.SurveyAnswers.Add(answer);

\_context.SaveChanges();

}

public void SaveSurvey(Survey survey)

{

\_context.Surveys.Update(survey);

\_context.SaveChanges();

}

public void AddSurvey(Survey survey)

{

\_context.Surveys.Add(survey);

\_context.SaveChanges();

}

public void DeleteSurvey(Survey survey)

{

\_context.Surveys.Remove(survey);

\_context.SaveChanges();

}

public void AddCompany(Company.Company company)

{

\_context.Companies.Add(company);

\_context.SaveChanges();

}

public void SaveCompany(Company.Company company)

{

\_context.Companies.Update(company);

\_context.SaveChanges();

}

public void DeleteCompany(Company.Company company)

{

\_context.Companies.Remove(company);

\_context.SaveChanges();

}

}

}

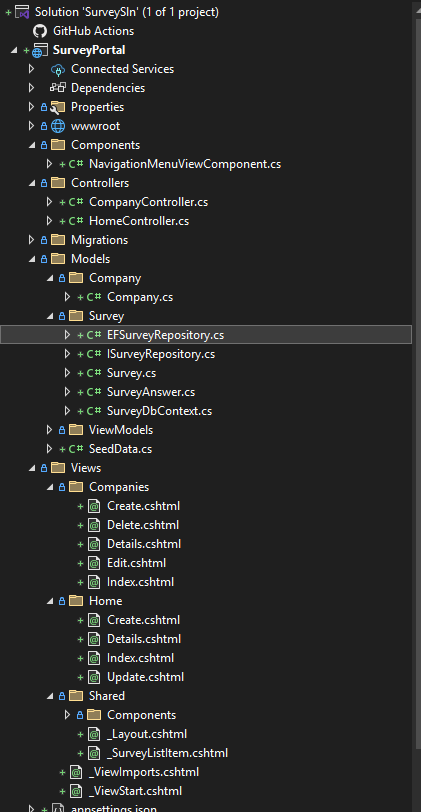


Рис. 1. Структура проекту після виконання лабораторної роботи.

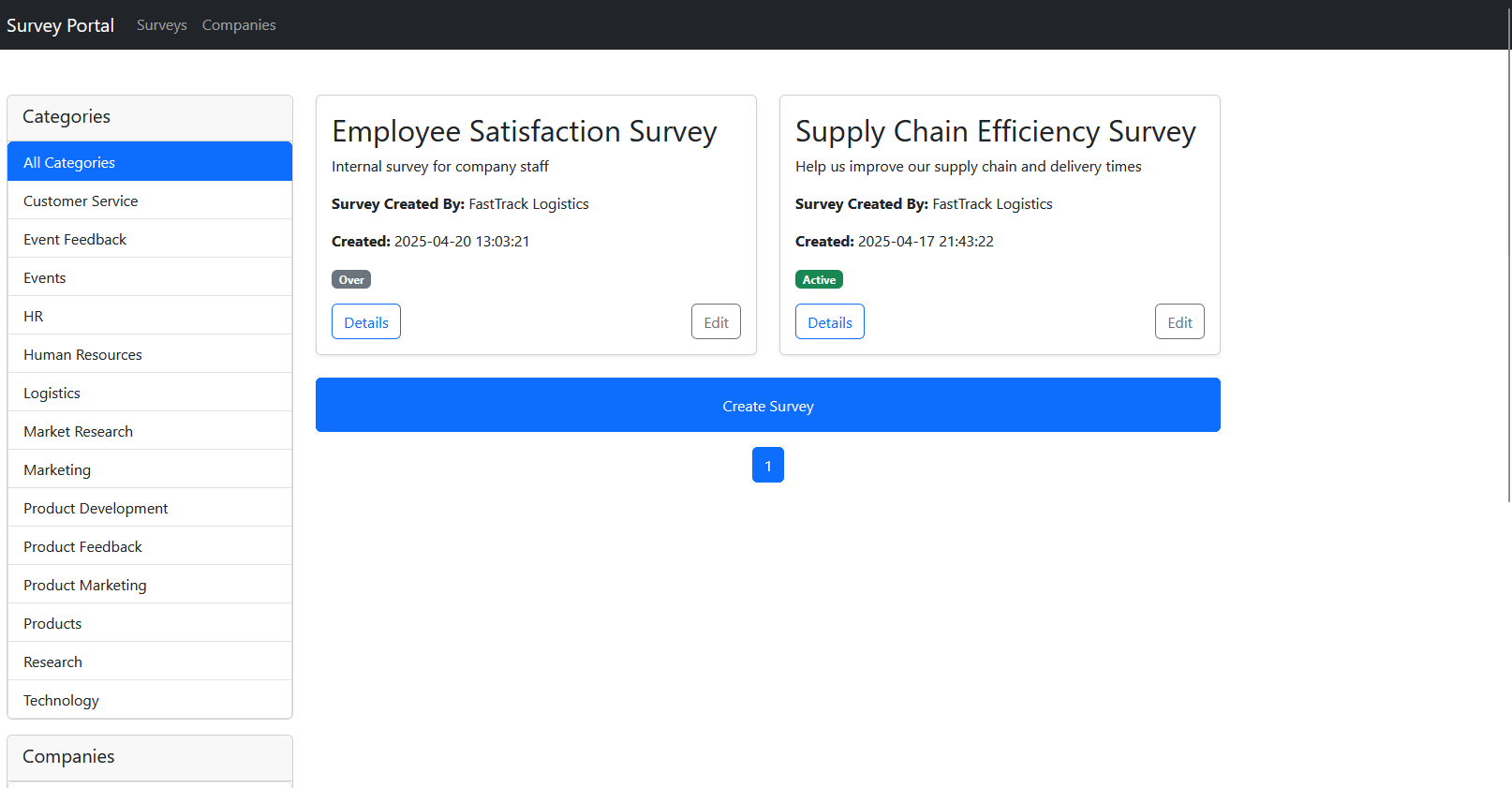


Рис. 2. Вигляд сайту після виконання лабораторної роботи.

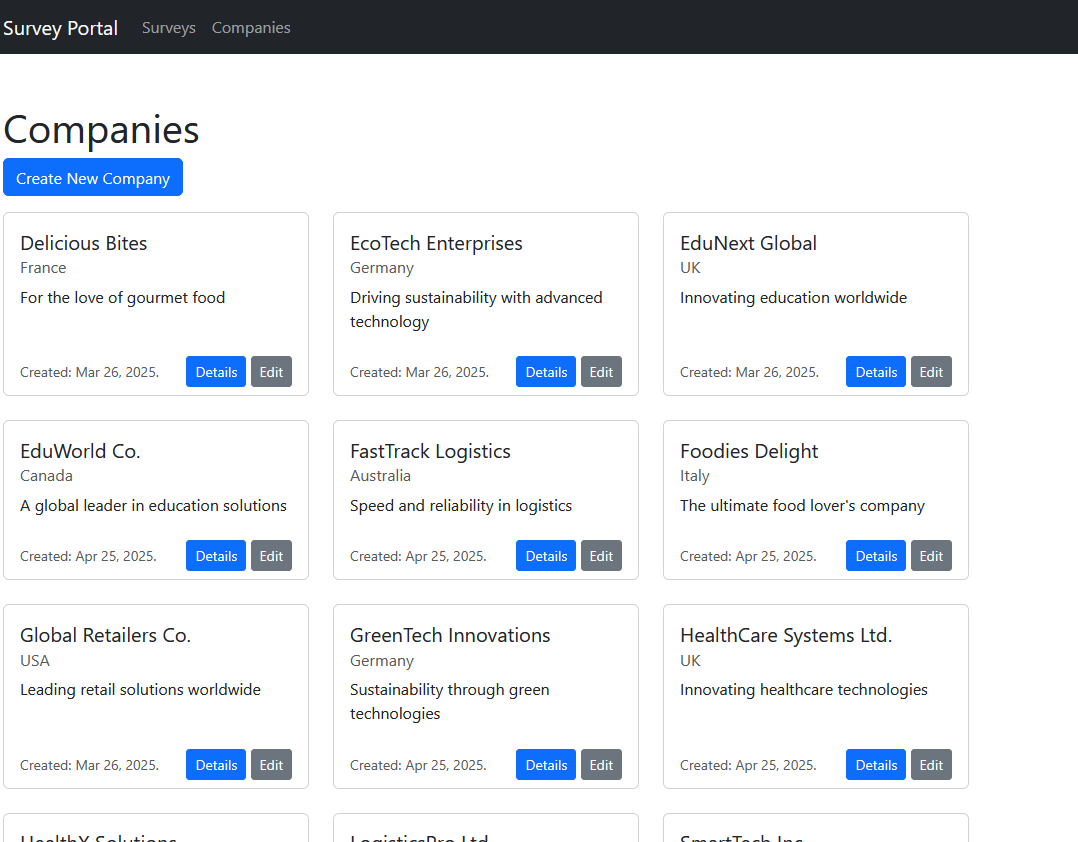


Рис. 3. View Companies/Index.

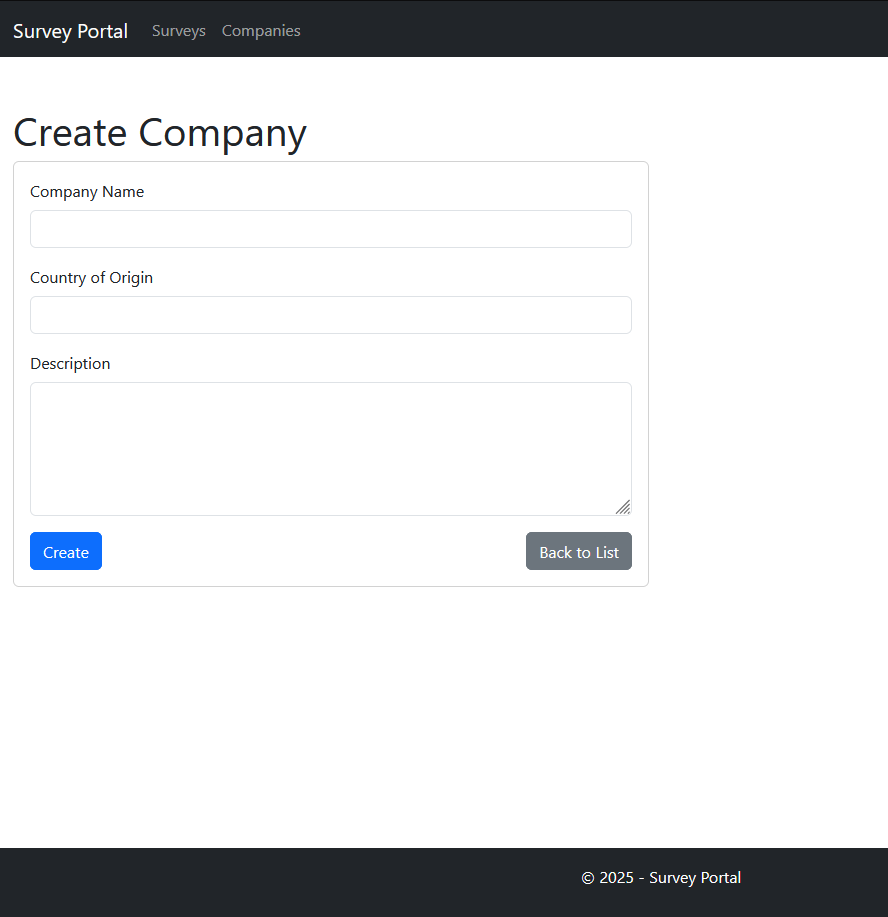


Рис. 4. View Companies/Create.

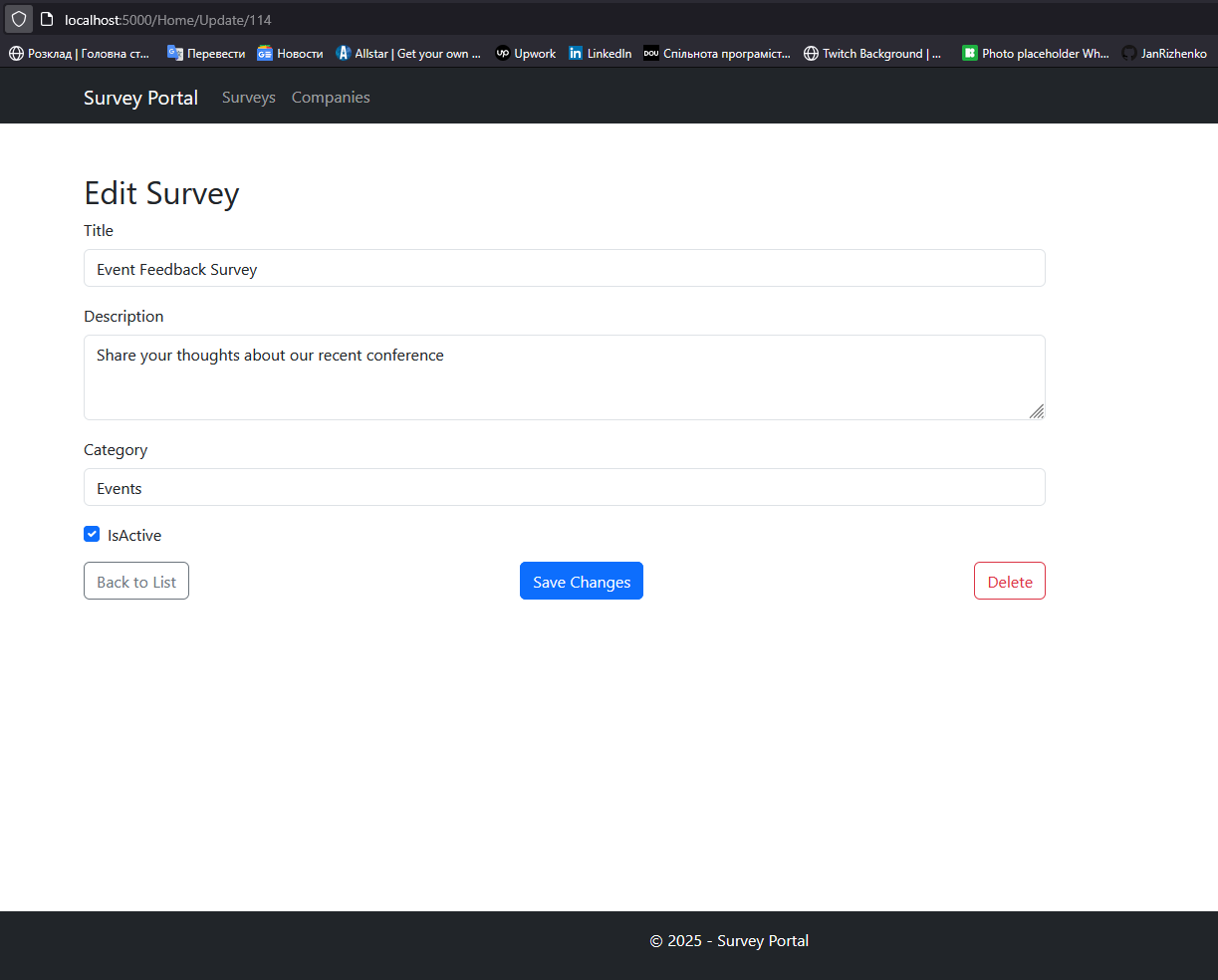


Рис. 5. View Home/Edit.

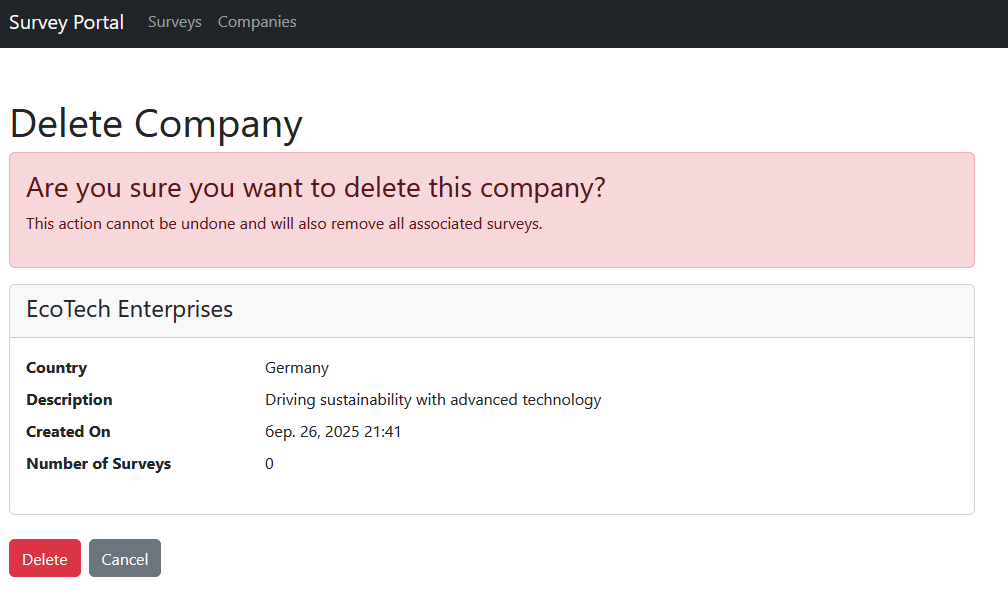


Рис. 6. View Companies/Delete.

**Завдання 3.**

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

**Листинг Model Survey після виконання лабораторної роботи:**

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace SurveyPortal.Models.Survey.Survey

{

public class Survey

{

[Key]

public long? SurveyID { get; set; }

[Required(ErrorMessage = "Please enter a survey title")]

public string Title { get; set; } = string.Empty;

[Required(ErrorMessage = "Please enter a disription")]

public string Description { get; set; } = string.Empty;

public DateTime CreatedAt { get; set; } = DateTime.UtcNow.AddTicks(-(DateTime.UtcNow.Ticks % TimeSpan.TicksPerSecond));

[Required(ErrorMessage = "Please enter a category")]

public string? Category { get; set; }

public bool IsActive { get; set; } = true;

[Required(ErrorMessage = "Please chose a company of survey")]

public long? CompanyID { get; set; }

public Company.Company? Company { get; set; }

public ICollection<SurveyAnswers> SurveyAnswers { get; set; } = new List<SurveyAnswers>();

}

}

**Листинг Model Company після виконання лабораторної роботи:**

using System.ComponentModel.DataAnnotations;

namespace SurveyPortal.Models.Company

{

public class Company

{

public long CompanyID { get; set; }

[Required(ErrorMessage = "Please enter a company name")]

public string Name { get; set; } = string.Empty;

[Required(ErrorMessage = "Please enter a company description")]

public string? Description { get; set; }

[Required(ErrorMessage = "Please enter a Country of origin.")]

public string? Country { get; set; }

public DateTime CreatedAt { get; set; } = DateTime.UtcNow.AddTicks(-(DateTime.UtcNow.Ticks % TimeSpan.TicksPerSecond));

public ICollection<Survey.Survey.Survey>? Surveys { get; set; } = new List<Survey.Survey.Survey>();

}

}

**Листинг Model SurveyAnswer після виконання лабораторної роботи:**

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace SurveyPortal.Models.Survey.Survey

{

public class SurveyAnswers

{

[Key]

public long SurveyAnswerID { get; set; }

[ForeignKey("Survey")]

public long SurveyID { get; set; }

public Survey Survey { get; set; }

[Required(ErrorMessage ="Please enter an answer")]

public string Answer { get; set; } = string.Empty;

public DateTime AnsweredAt { get; set; } = DateTime.UtcNow.AddTicks(-(DateTime.UtcNow.Ticks % TimeSpan.TicksPerSecond));

}

}

Посилання на репозиторій: <https://github.com/JanRizhenko/ASP.NET>

***Висновок:*** У результаті виконання завдання було набуте практичне розуміння використання ORM EF Core для реалізації CRUD-операцій, а також засвоєно основи валідації даних для забезпечення коректності введеної інформації..