**Прізвище:** Варіченко

**Ім'я:** Лідія

**Група:** КН-305

**Варіант:**13\1

**Дисципліна:** Проектування інформаційних систем

**Перевірив:** Файтас О. І.

**ЗВІТ**

до лабораторної роботи №6

на тему "Oснови роботи з ado.net"

**Мета роботи:** здобути теоретичні основи патернів проектування MVC, MVP I MVVM. Реалізувати проект на технології ASP.NET MVC для роботи з базою даних.

**Лабораторне завдання:**

1. Реалізувати проект для обробки даних за допомогою ASP.NET MVC 5.

2. Весь написаний код внести в систему контролю версій.

**Індивідуальне завдання:**

Таблиці:

Країни,

Матчі,

Стадіони.

Зв’язки:

Матчі – Стадіони = N-N

Країни – Стадіони = 1-N

**Результат виконання індивідуального завдання:**

Першим кроком було успадковано «DropCreateDatabaseAlways» щоб програма заповнювала БД при старті. Лістинг наведено нижче:

public class CustomDbInitializer : DropCreateDatabaseAlways<MyDbContext>

{

protected override void Seed(MyDbContext context)

{

context.Countries.Add(new Lab6.Models.Country { name = "Ukraine", capitalCity = "Kyiv" });

context.Countries.Add(new Lab6.Models.Country { name = "Great Britain", capitalCity = "London" });

context.Countries.Add(new Lab6.Models.Country { name = "Canada", capitalCity = "Ottawa" });

context.Countries.Add(new Lab6.Models.Country { name = "Austria", capitalCity = "Vena" });

context.Games.Add(new Lab6.Models.Game { sportType = "baseball", team1 = "White Sox", team2 = "Shahtar" });

context.Games.Add(new Lab6.Models.Game { sportType = "baseball", team1 = "Astros", team2 = "Shahtar" });

context.Games.Add(new Lab6.Models.Game { sportType = "football", team1 = "Mets", team2 = "Dynamo" });

context.Games.Add(new Lab6.Models.Game { sportType = "football", team1 = "Black Fox", team2 = "Vorskla" });

context.Stadia.Add(new Lab6.Models.Stadium { name = "Stadium#1", square = 90.0, countryId = 1 });

context.Stadia.Add(new Lab6.Models.Stadium { name = "Stadium#2", square = 90.0, countryId = 1 });

context.Stadia.Add(new Lab6.Models.Stadium { name = "Stadium#3", square = 90.0, countryId = 1 });

context.Stadia.Add(new Lab6.Models.Stadium { name = "Stadium#4", square = 90.0, countryId = 1 });

context.Schedules.Add(new Lab6.Models.Schedule { countryId = 1, gameId = 2 });

context.Schedules.Add(new Lab6.Models.Schedule { countryId = 2, gameId = 3 });

context.Schedules.Add(new Lab6.Models.Schedule { countryId = 3, gameId = 4 });

context.Schedules.Add(new Lab6.Models.Schedule { countryId = 4, gameId = 1 });

}

}

Наступним кроком було створено контролери (рис. 1).

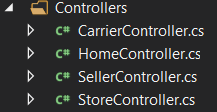


Рис. 1. Створені контролери

Далі для кожного контролера було створено представлення (View). Для кожної сутності було створено два представлення тому, що один з них відповідає за сторінку для створення нової сутністі. Загальне представлення для сутності «Country» зображено на рисунку 2.

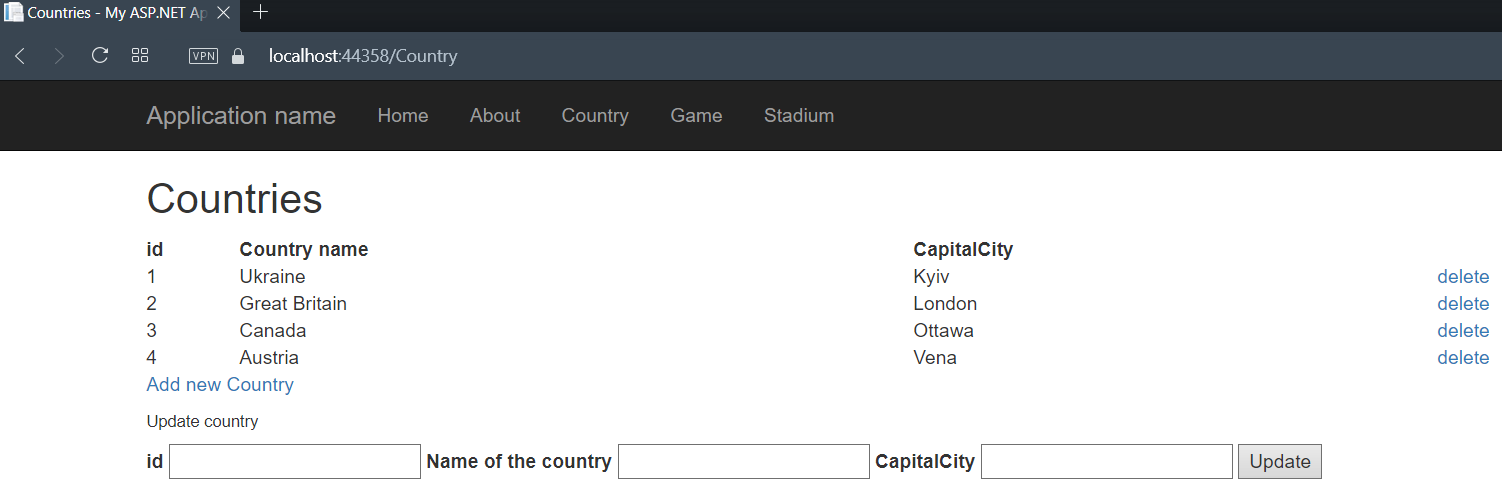


Рис. 2. Представлення для сутності Country.

Як видно з рисунку 2 – на сторінці відображається інформація про всі країни, які збережені в системі. Якщо користувач натисне на «Add new country» - він буде перенаправлений на сторінку для зберігання інформації про нову країну (рис. 3).

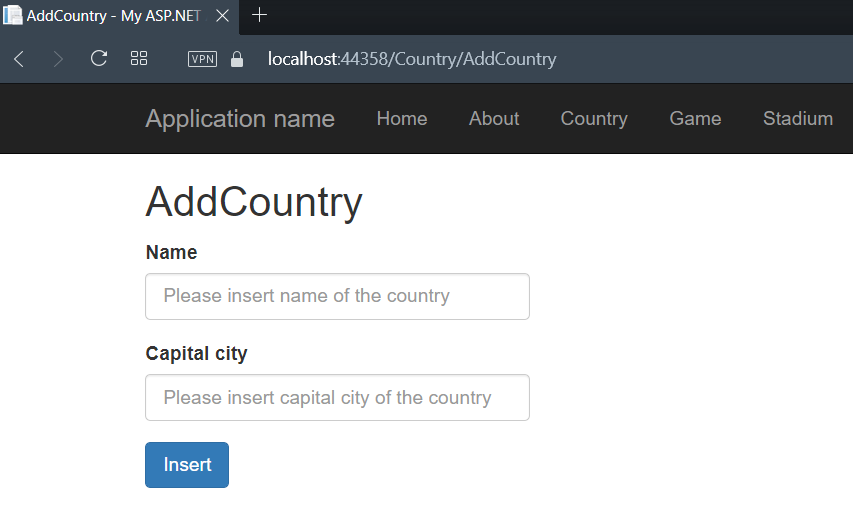


Рис. 3. Сторінка для збереження інформації про нову країну

Якщо користувач натисне на «delete» - країна, навпроти якої було натиснено кнопку, буде видалена з бази даних.

Для того щоб відредагувати інформацію про країну – необхідно ввести його id і нові дані (рис. 4).

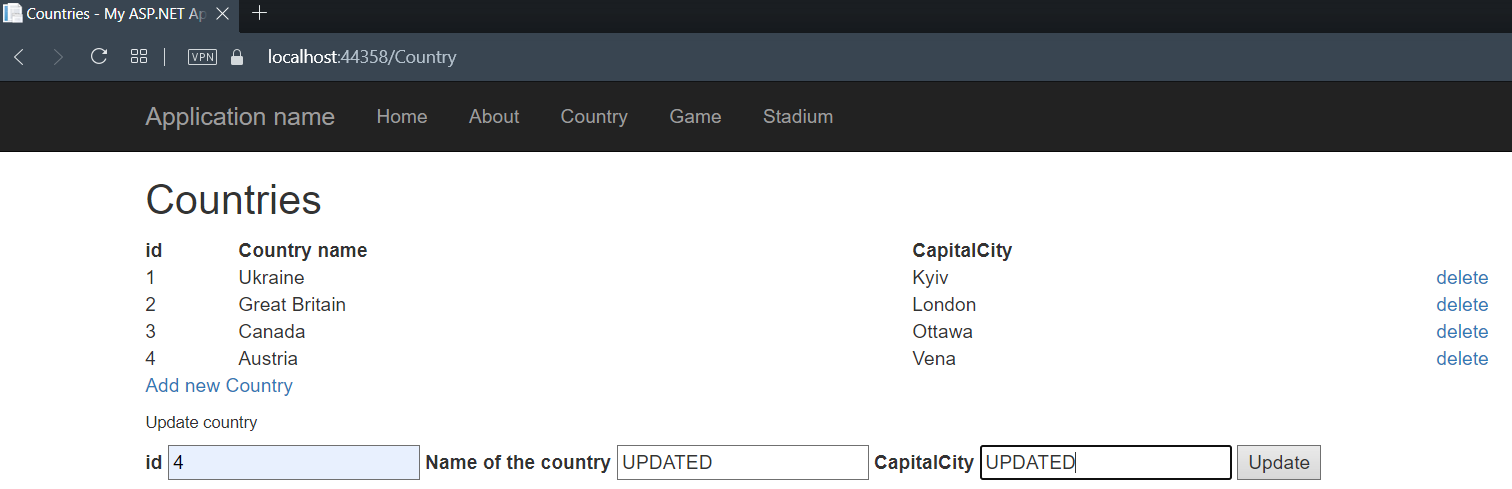


Рис. 4. Ввід даних для оновлення інформації про країну

Результат оновлення зображено на рисунку 5.

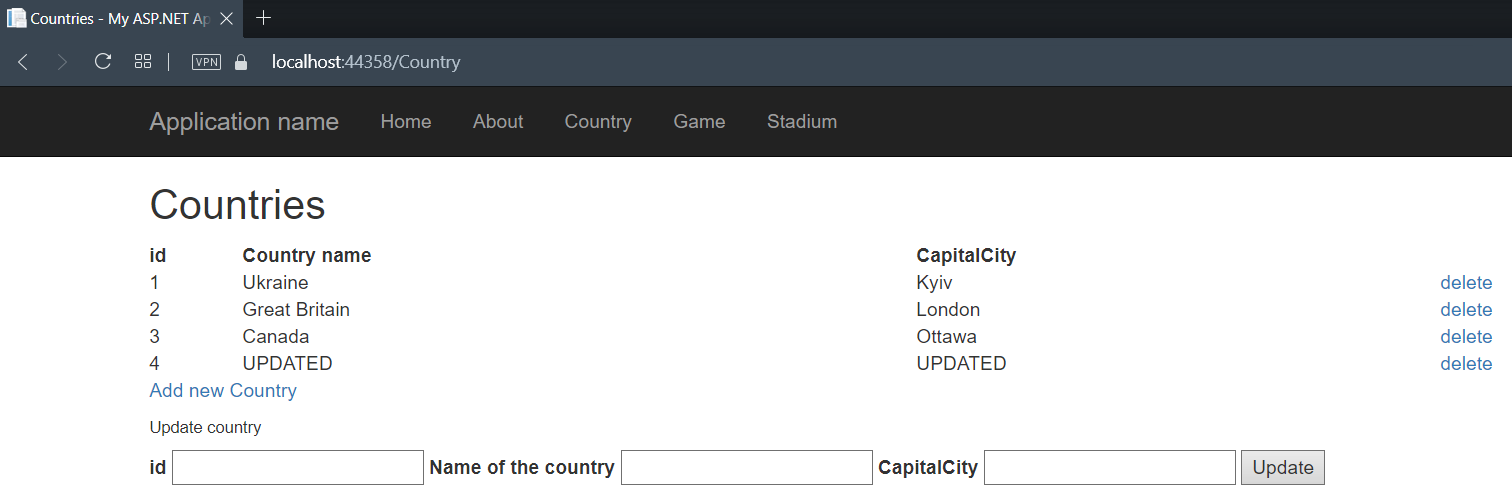


Рис. 5. Результат додавання нового постачальника

Для кожної з сутностей було реалізовано цей функціонал.

На вкладці «About» - інформація про того, хто виконав роботу (рис. 6).

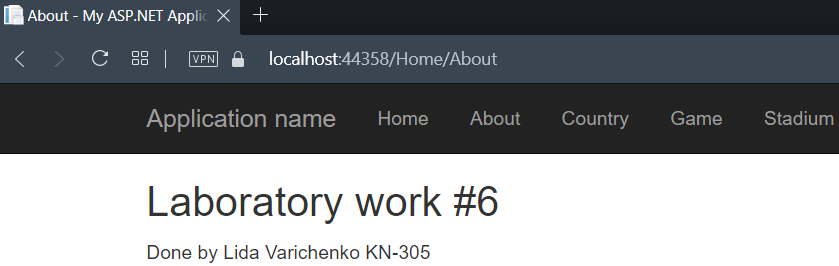


Рис. 6. Вкладка About

Для реалізації виводу всіх країн на сторінці - для представлення було передано DbContext: return View(context);.

В коді представлення, DbContext приймаєтсья за допомогою стрічки: @model Lab6.App\_Start.MyDbContext

В представленні відбувається ітерація по всіх сутностях які отримуються за допомогою методу Model.Countries.ToList() для побудови таблиці.

<div>

<table style="width: 100%">

<tr>

<th>id</th>

<th>Country name</th>

<th>CapitalCity</th>

</tr>

@foreach (var country in Model.Countries.ToList())

{

<tr>

<td>@country.id</td>

<td>@country.name</td>

<td>@country.capitalCity</td>

<td><a href="/Country/Delete/@country.id/" role="button"><span aria-hidden="true">delete</span></a></td>

</tr>

}

</table>

</div>

З лістингу видно, що остання колонка таблиці має в собі посилання. Це посилання на контролер «Country» метод «DeleteCountry». Коли користувач натискає на це посилання, відбувається HTTP метод GET на конкретний метод:

[HttpGet]

public ActionResult Delete(int id)

Як видно метод приймає id країни для якого необхідно здійснити видалення. Id країни отримується з стрічки @carrier.id. Цей id передається в метод і програма опрацьовує його:

Country c = new Country { id = id };

context.Countries.Attach(c);

context.Countries.Remove(c);

context.SaveChanges();

Спочатку відбувається пошук в БД країни за id.

Метод Attach використовується для того, щоб контекст почав відслідковувати задану сутність. Оскільки контекст містить сутність з заданим id – то відбувається прив’язка нової сутності до сутності яка наявна в контексті. Таким чином контекст буде знати яку конкретну сутність я хочу видалити.

Створення нової країни відбувається за допомогою іншої сторінки. Коли користувач ввів всі необхідні дані і натиснув кнопку «Insert» - відбувається HTTP метод POST (метод вказано атрибутом method тегу form). Оскільки атрибут action відсутній, то метод POST відбувається на той же метод контроллера (AddCountry).

[HttpPost]

public ActionResult AddCountry(Country country)

Цей метод приймає сутність «Country», яку MVC створює на основі данних, які були введені користувачем. Саме тому в тега input атрибут name=”name”. Якщо прирівняти до атрибуту якесь інше значення, то поле name сутності Country буде null. Тобто атрибут тегу input повинен відповідати полю сутності.

Аналогічні контролери і представлення було створено і для інших сутностей.

Цей проект було додано в систему контролю версій і завантажено на віддалений репозиторій: <https://github.com/lida-kn-305/lab6>

Прив’язав github до цього електронного адресу

email: lidainki@protonmail.com

pass: root

username: lida-kn-305

pass: rootpassword12345

**Висновок:** протягом виконання цієї роботи, я здобула теоретичні основи патернів проектування, а саме ознайомилася з патернами «MVC», «MVP», «MVVM». Реалізувала проект за допомогою технології ASP.NET MVC. Реалізувала доступ до БД. Додала проект в систему контролю версій.