**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 5**

по дисциплине «Введение в облачные вычисления»

на тему: «Работа с *Windows Azure Table* через веб-интерфейс»

Выполнил: студент гр. ИТП-31

Коркуц С. И.

Принял: преподаватель

Гуменников Е.Д.

Гомель 2020

**Цель работы:** освоить разработку веб-интерфейса для взаимодействия с хранилищем *Windows Azure Table*.

**Задание**

Разработать веб-приложение, реализующее *CRUD* операции над хранилищем, разработанным в 4 лабораторной работе.

Вариант задания:



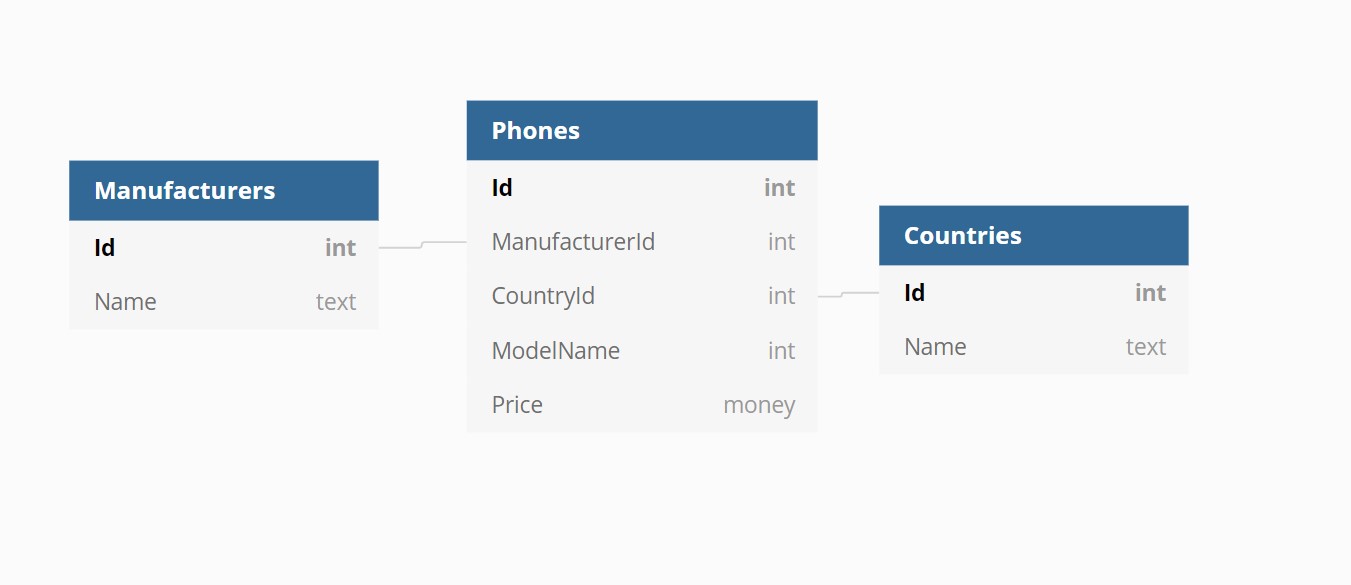
****

Рисунок 1 – Схема базы данных

**Ход работы**

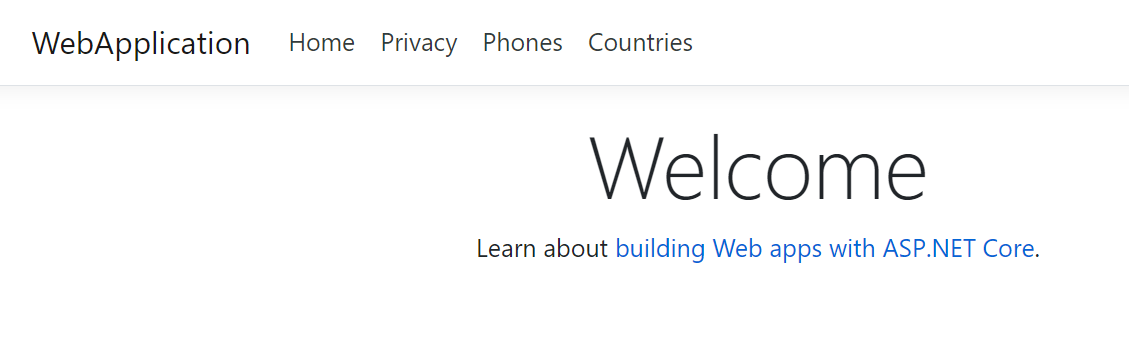


Рисунок 2 ­– Начальная страница веб-приложения

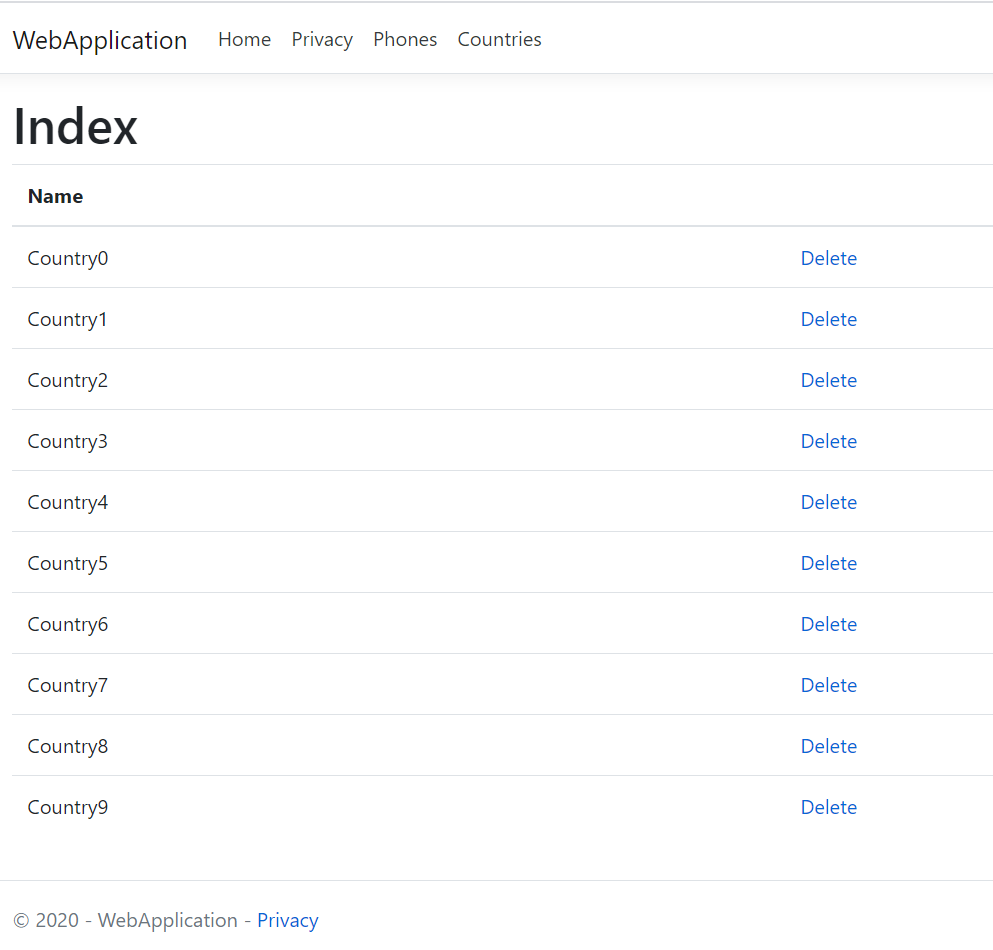


Рисунок 3 – Веб-страница с данными таблицы «*Countries*»

На рисунке 4 изображен результат каскадного удаления записи «*Country1*». При переходе на другие страницы этой записи не будет, в качестве примера приведена таблица «*Phones*».

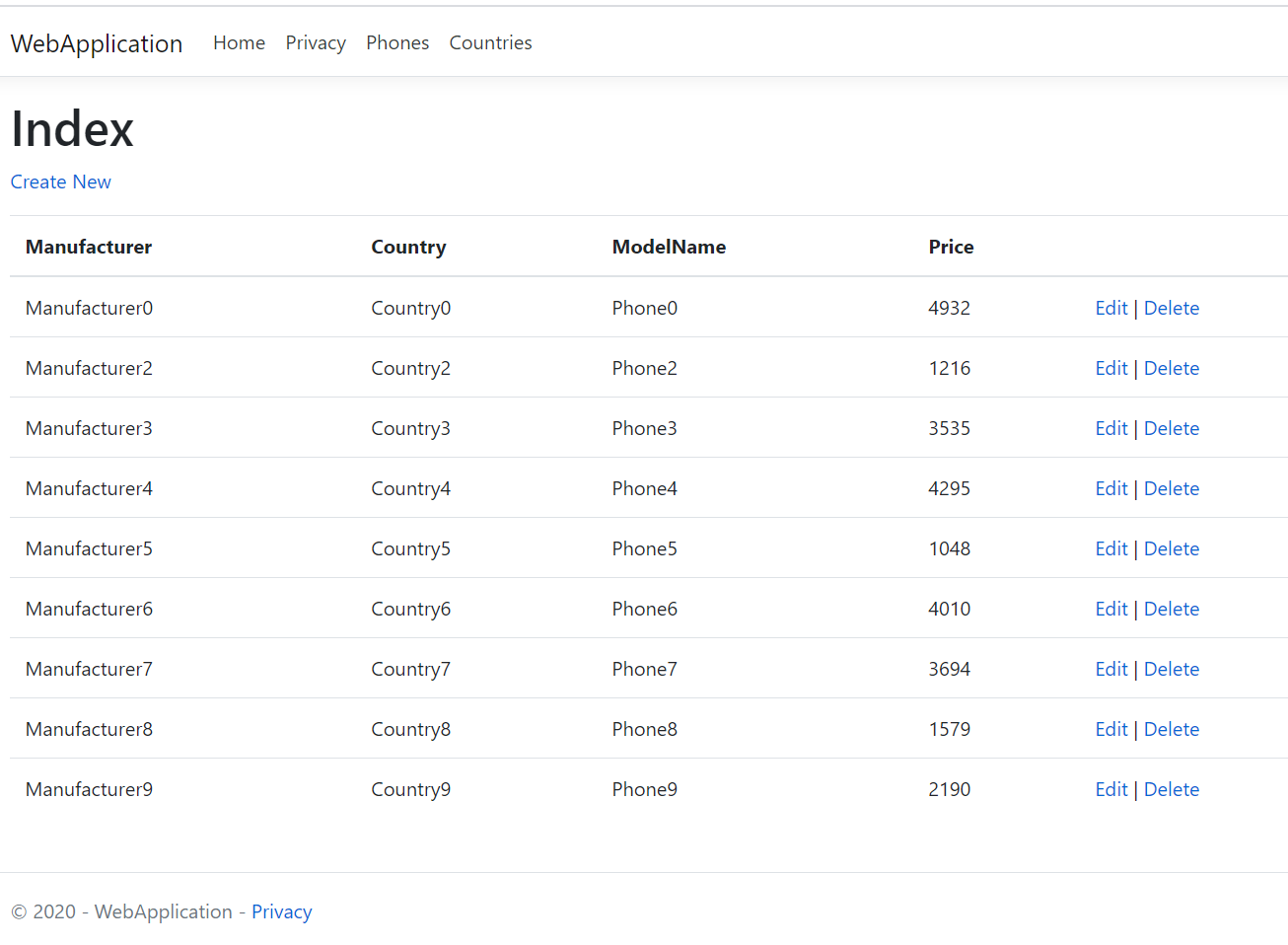


Рисунок 4 – Результат каскадного удаления записи «*Country1*» из таблицы «*Countries*»

На рисунках 5 – 9 приведены *CRUD* операции.

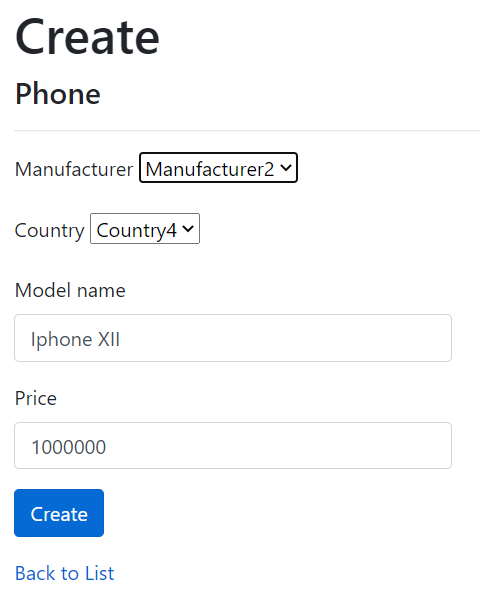


Рисунок 5 – Заполненная форма создаваемого телефона

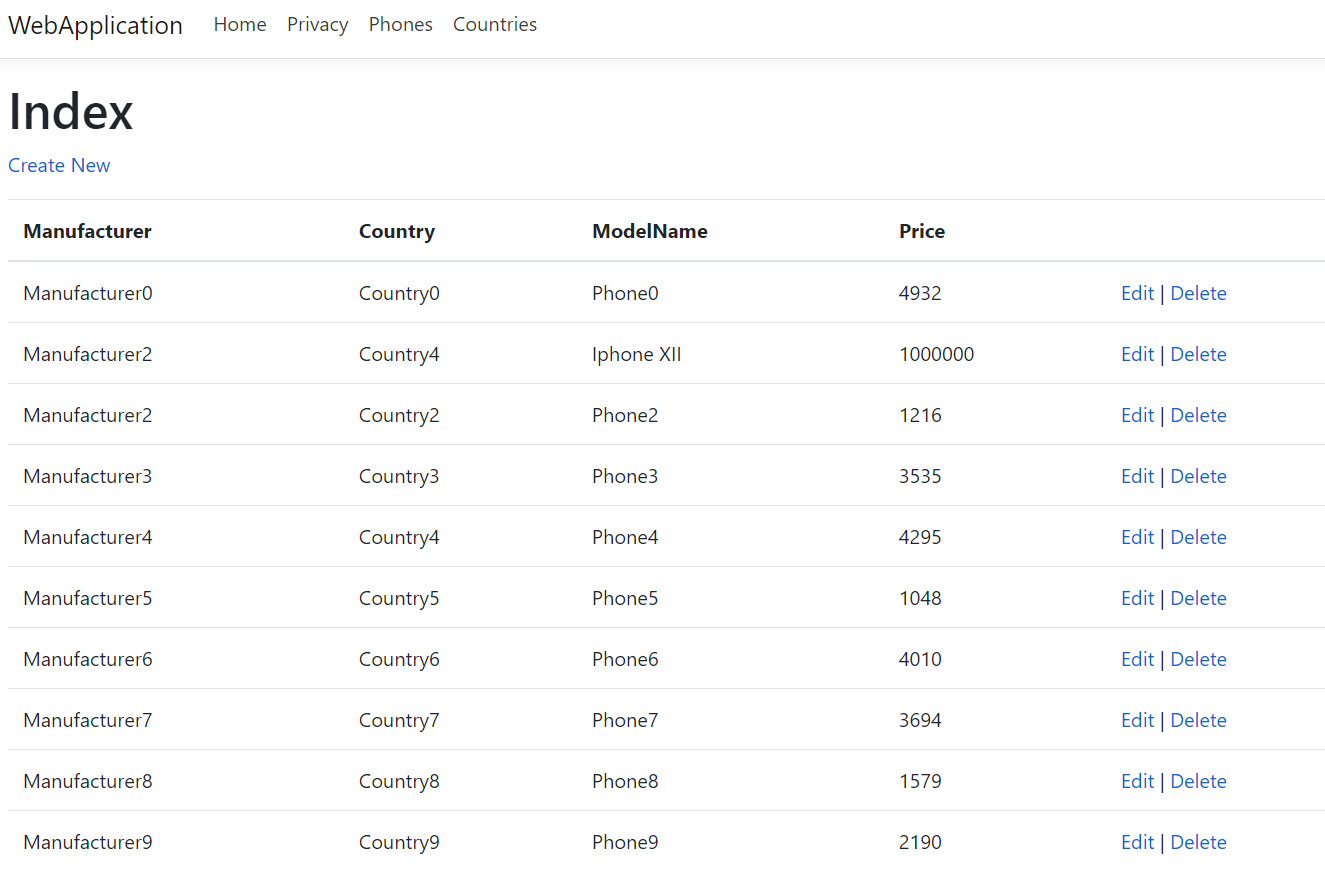


Рисунок 6 – Результат создания телефона

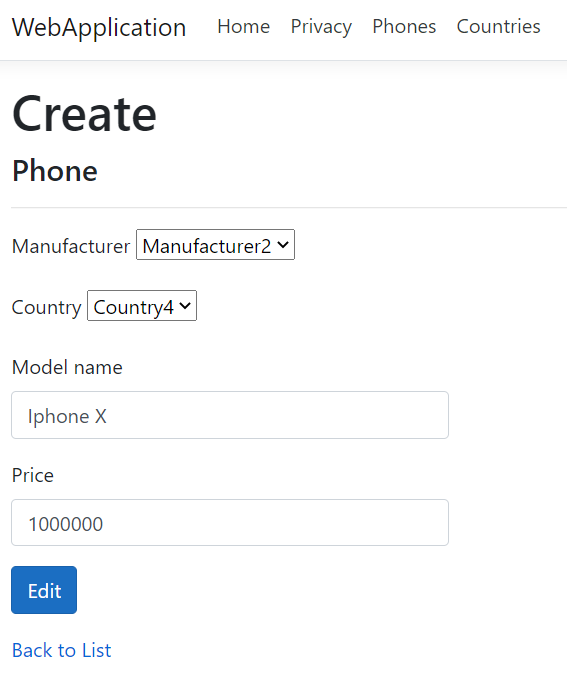


Рисунок 7 – Заполненная форма изменения телефона

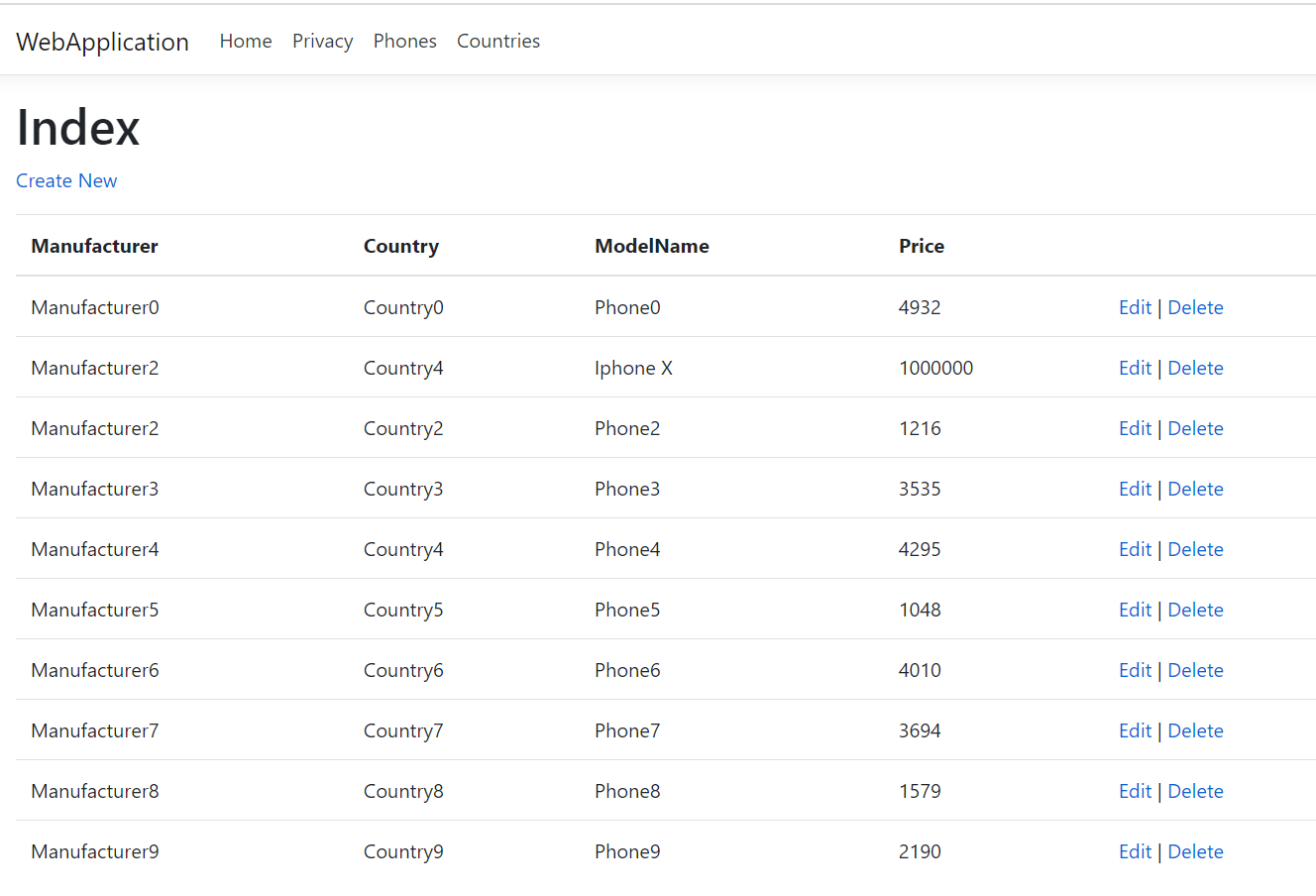


Рисунок 8 – Результат изменения телефона

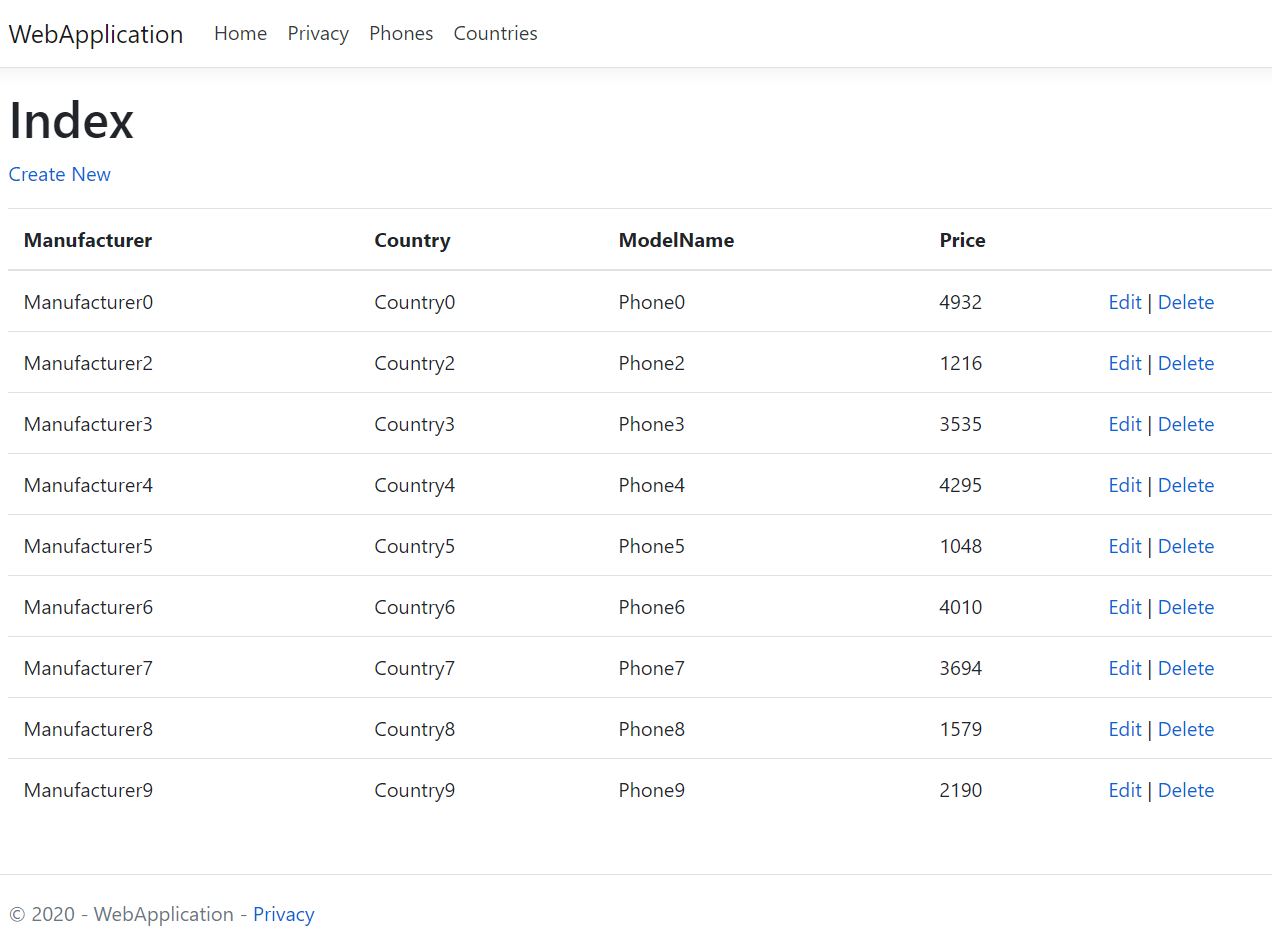


Рисунок 9 – Результат удаления телефона

**Вывод:** в результате лабораторной работы была освоена разработка веб интерфейса для взаимодействия с хранилищем *Windows Azure Table*.

**ПРИЛОЖЕНИЕ А**

**Листинг созданных классов**

using System;

using CosmosTableSamples;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Azure.Cosmos.Table;

namespace WebApplication.Controllers

{

public class PhoneController : Controller

{

private readonly StorageOperations \_so;

private readonly EntityOperations \_eo;

public PhoneController()

{

\_so = new StorageOperations();

\_eo = new EntityOperations();

}

// GET: PhoneController

public async Task<ActionResult> Index()

{

var phonesTable = await \_so.CreateTableAsync("phones");

var manufacturersTable = await \_so.CreateTableAsync("manufacturers");

var countriesTable = await \_so.CreateTableAsync("countries");

var phones = \_eo.GetPhones(phonesTable);

var manufacturers = \_eo.GetManufacturers(manufacturersTable);

var countries = \_eo.GetCountries(countriesTable);

ViewModels.IndexPhoneViewModel model = new ViewModels.IndexPhoneViewModel

{

Phones = phones,

Manufacturers = new Dictionary<string, string>(manufacturers.Select(m => new KeyValuePair<string, string>(m.RowKey, m.Name)).ToList()),

Countries = new Dictionary<string, string>(countries.Select(m => new KeyValuePair<string, string>(m.RowKey, m.Name)).ToList())

};

return View(model);

}

// GET: PhoneController/Details/5

public ActionResult Details(int id)

{

return View();

}

// GET: PhoneController/Create

public async Task<ActionResult> Create()

{

var manufacturersTable = await \_so.CreateTableAsync("manufacturers");

var countriesTable = await \_so.CreateTableAsync("countries");

var manufacturers = \_eo.GetManufacturers(manufacturersTable);

var countries = \_eo.GetCountries(countriesTable);

ViewModels.CreatePhoneViewModel model = new ViewModels.CreatePhoneViewModel

{

Manufacturers = new Microsoft.AspNetCore.Mvc.Rendering.SelectList(manufacturers, "RowKey", "Name"),

Countries = new Microsoft.AspNetCore.Mvc.Rendering.SelectList(countries, "RowKey", "Name")

};

return View(model);

}

// POST: PhoneController/Create

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Create(int? countryId, int? manufacturerId, string modelName, int? price)

{

try

{

var phonesTable = await \_so.CreateTableAsync("phones");

int maxId = \_eo.GetPhones(phonesTable).Select(p => int.Parse(p.RowKey)).Max();

CosmosTableSamples.Models.Phone phone = new CosmosTableSamples.Models.Phone(maxId + 1)

{

ModelName = modelName,

ManufacturerId = manufacturerId ?? 1,

CountryId = countryId ?? 1,

Price = price ?? 100

};

await \_eo.InsertOrMergePhoneAsync(phonesTable, phone);

return RedirectToAction(nameof(Index));

}

catch

{

var manufacturersTable = await \_so.CreateTableAsync("manufacturers");

var countriesTable = await \_so.CreateTableAsync("countries");

var manufacturers = \_eo.GetManufacturers(manufacturersTable);

var countries = \_eo.GetCountries(countriesTable);

ViewModels.CreatePhoneViewModel model = new ViewModels.CreatePhoneViewModel

{

Manufacturers = new Microsoft.AspNetCore.Mvc.Rendering.SelectList(manufacturers, "RowKey", "Name"),

Countries = new Microsoft.AspNetCore.Mvc.Rendering.SelectList(countries, "RowKey", "Name")

};

return View(model);

}

}

// GET: PhoneController/Edit/5

public async Task<ActionResult> Edit(int id)

{

var manufacturersTable = await \_so.CreateTableAsync("manufacturers");

var countriesTable = await \_so.CreateTableAsync("countries");

var manufacturers = \_eo.GetManufacturers(manufacturersTable);

var countries = \_eo.GetCountries(countriesTable);

var phonesTable = await \_so.CreateTableAsync("phones");

var phone = \_eo.GetPhones(phonesTable).FirstOrDefault(p => int.Parse(p.RowKey) == id);

ViewModels.CreatePhoneViewModel model = new ViewModels.CreatePhoneViewModel

{

Manufacturers = new Microsoft.AspNetCore.Mvc.Rendering.SelectList(manufacturers, "RowKey", "Name", phone.ManufacturerId.ToString()),

Countries = new Microsoft.AspNetCore.Mvc.Rendering.SelectList(countries, "RowKey", "Name", phone.CountryId.ToString())

};

ViewBag.Phone = phone;

return View(model);

}

// POST: PhoneController/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Edit(int id, int? countryId, int? manufacturerId, string modelName, int? price)

{

try

{

CosmosTableSamples.Models.Phone phone = new CosmosTableSamples.Models.Phone(id)

{

ModelName = modelName,

ManufacturerId = manufacturerId ?? 1,

CountryId = countryId ?? 1,

Price = price ?? 100

};

var phonesTable = await \_so.CreateTableAsync("phones");

await \_eo.InsertOrMergePhoneAsync(phonesTable, phone);

return RedirectToAction(nameof(Index));

}

catch

{

var manufacturersTable = await \_so.CreateTableAsync("manufacturers");

var countriesTable = await \_so.CreateTableAsync("countries");

var manufacturers = \_eo.GetManufacturers(manufacturersTable);

var countries = \_eo.GetCountries(countriesTable);

var phonesTable = await \_so.CreateTableAsync("phones");

var phone = \_eo.GetPhones(phonesTable).FirstOrDefault(p => int.Parse(p.RowKey) == id);

ViewModels.CreatePhoneViewModel model = new ViewModels.CreatePhoneViewModel

{

Manufacturers = new Microsoft.AspNetCore.Mvc.Rendering.SelectList(manufacturers, "RowKey", "Name", phone.ManufacturerId.ToString()),

Countries = new Microsoft.AspNetCore.Mvc.Rendering.SelectList(countries, "RowKey", "Name", phone.CountryId.ToString())

};

ViewBag.Phone = phone;

return View();

}

}

// GET: PhoneController/Delete/5

public async Task<ActionResult> Delete(int id)

{

var phonesTable = await \_so.CreateTableAsync("phones");

await \_eo.DeletePhoneAsync(phonesTable, new CosmosTableSamples.Models.Phone(id) { ETag = "\*" });

return RedirectToAction(nameof(Index));

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using CosmosTableSamples;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace WebApplication.Controllers

{

public class CountriesController : Controller

{

private readonly StorageOperations \_so;

private readonly EntityOperations \_eo;

public CountriesController()

{

\_so = new StorageOperations();

\_eo = new EntityOperations();

}

// GET: CountriesController

public async Task<ActionResult> Index()

{

var countriesTable = await \_so.CreateTableAsync("countries");

var countries = \_eo.GetCountries(countriesTable);

return View(countries);

}

// GET: CountriesController/Delete/5

public async Task<ActionResult> Delete(int id)

{

var phonesTable = await \_so.CreateTableAsync("phones");

var countriesTable = await \_so.CreateTableAsync("countries");

await \_eo.CascadeDeleteCountry(phonesTable, countriesTable, new CosmosTableSamples.Models.Country(id) { ETag = "\*" });

return RedirectToAction(nameof(Index));

}

}

}

using System;

using CosmosTableSamples;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace WebApplication.Data

{

public class StorageInitializer

{

StorageOperations \_so;

EntityOperations \_eo;

public StorageInitializer()

{

\_so = new StorageOperations();

\_eo = new EntityOperations();

}

public async Task Initialize()

{

var manufacturers = await \_so.CreateTableAsync("manufacturers");

var countries = await \_so.CreateTableAsync("countries");

var phones = await \_so.CreateTableAsync("phones");

for (int i = 0; i < 10; i++)

{

var country = new CosmosTableSamples.Models.Country(i) { Name = $"Country{i}" };

await \_eo.InsertOrMergeCountryAsync(countries, country);

}

for (int i = 0; i < 10; i++)

{

var manufacturer = new CosmosTableSamples.Models.Manufacturer(i) { Name = $"Manufacturer{i}" };

await \_eo.InsertOrMergeManufacturerAsync(manufacturers, manufacturer);

}

Random random = new Random();

for (int i = 0; i < 10; i++)

{

var phone = new CosmosTableSamples.Models.Phone(i) { ModelName = $"Phone{i}", Price = random.Next(1000, 5000), ManufacturerId = i, CountryId = i};

await \_eo.InsertOrMergePhoneAsync(phones, phone);

}

}

}

}

using Microsoft.AspNetCore.Mvc.Rendering;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace WebApplication.ViewModels

{

public class CreatePhoneViewModel

{

public SelectList Manufacturers { get; set; }

public SelectList Countries { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace WebApplication.ViewModels

{

public class IndexPhoneViewModel

{

public List<CosmosTableSamples.Models.Phone> Phones { get; set; }

public Dictionary<string, string> Countries { get; set; }

public Dictionary<string, string> Manufacturers { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

namespace WebApplication

{

public class Program

{

public static async Task Main(string[] args)

{

var host = CreateHostBuilder(args).Build();

using (var scope = host.Services.CreateScope())

{

var dbInitializer = scope.ServiceProvider.GetRequiredService<Data.StorageInitializer>();

await dbInitializer.Initialize();

}

host.Run();

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using CosmosTableSamples;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

namespace WebApplication

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddTransient<Data.StorageInitializer>();

services.AddControllersWithViews().AddRazorRuntimeCompilation();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

});

}

}

}