**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

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

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

дисциплина «Разработка приложений баз данных для информационных систем»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

«Разработка моделей и контроллеров ASP.NET MVC приложения баз данных.»

ВАРИАНТ № 15

Выполнил: студент гр. ИТП-31

Козлов О. В.

Принял:доцент

Асенчик О. Д.

Гомель 2023

**Цель работы:** ознакомиться с возможностями *ASP.NET Core MVC* и *Entity Framework Core* для разработки слоя доступа к данным, хранящимся в базе данных, и обработки запросов пользователя посредством контроллеров.

**Задание:** создать с использованием *ASP.NET Core MVC Web*-приложение, содержащее набор классов, моделирующих предметную область, и осуществляющих генерацию и заполнение тестовыми наборами записей базу данных. Разработать один компонент *middleware*, контроллеры и представления для выборки и отображения информации из не менее чем 3- таблиц базы данных с использованием механизма внедрение зависимостей.

Для выполнения задания необходимо создать:

• Классы, моделирующие не менее чем три таблицы базы данных согласно вашему варианту. Перечень таблиц предварительно согласовывается с преподавателем. Одна из таблиц обязательно должна находиться на стороне отношения «многие» связи с другой таблицей в схеме базы данных.

• Класс контекста данных.

• Другие классы, например, классы *View Model* и т.п. (при необходимости).

• Компонент *middleware*, вызываемый в классе *Startup*, для инициализации базы данных путем заполнения ее таблиц тестовым набором записей.

• Классы контроллеров (по одному на каждую таблицу базы данных) для обработки обращений пользователя, выборки данных из таблиц и вызова соответствующих представлений для отображения выбранных данных.

• Разработать представления для отображения данных из таблиц, выбранных контроллерами. Представления, работающими с таблицами, стоящими на стороне отношения «многие» в схеме базы данных, должны выводить вместо кодов внешних ключей смысловые значения из связанных таблиц, стоящих на стороне отношения «один».

• Используя предварительно созданный и сконфигурированный в классе *Startup* профиль кэширования, подключить кэширование вывода для страниц с использованием атрибута *ResponseCache* для соответствующих методов контроллера. Данные в кэше хранить неизменными в течение 2\**N*+240 секунд, где N- номер вашего варианта.

**Ход работы:** для начала необходимо в созданном приложении разработать класс контекста базы данных, а также все классы, моделирующие предметную область. С целью лучшей структуризации проекта классы моделей были помещены в отдельную папку *Models,* а класс контекста в папку *Data*. На рисунке 1 представлена схема взаимодействия сущностей базы данных, построенная при помощи встроенного конструктора схем.

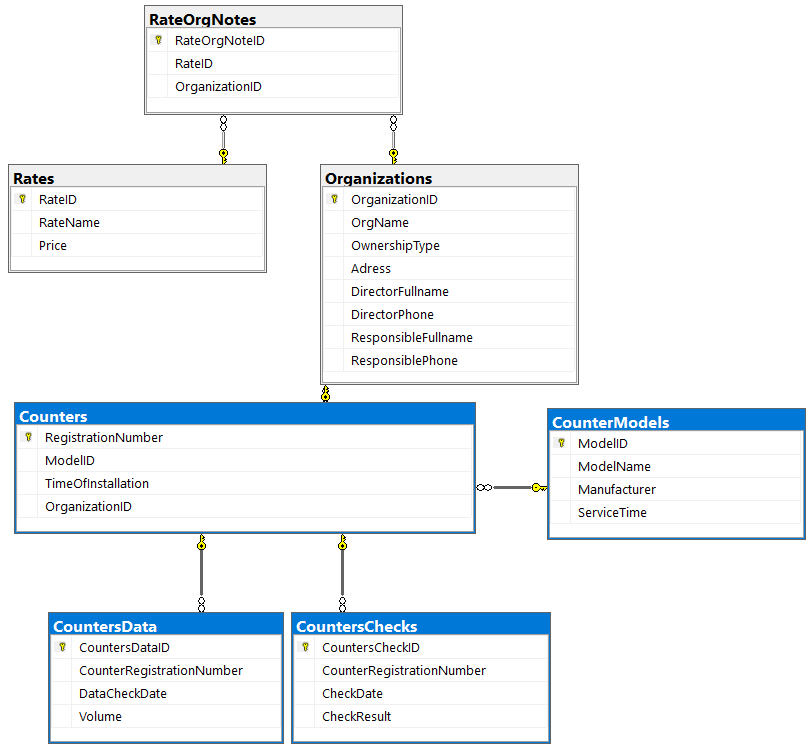


Рисунок 1 – Схема взаимодействия моделей

Как только модели созданы можно приступить к созданию *middleware,* которая позволит заполнять таблицы в базе данных случайными значениями. Для этого были созданы наборы различных данных: фамилии, имена, города и т.д. После при помощи циклов генерируются комбинации этих данных и заносятся в базу данных.

После заполнения базы данных были разработаны контроллеры и представления, который позволят управлять вызовом страниц и отображением данных, которые буду запрашиваться из базы данных. Стоит отметить, что, согласно заданию, контроллеры были созданы так, что данные получаются из кэша, а не на прямую запросом из базы данных.

На рисунке 2 показан пример страницы, содержащей информацию из базы данных.

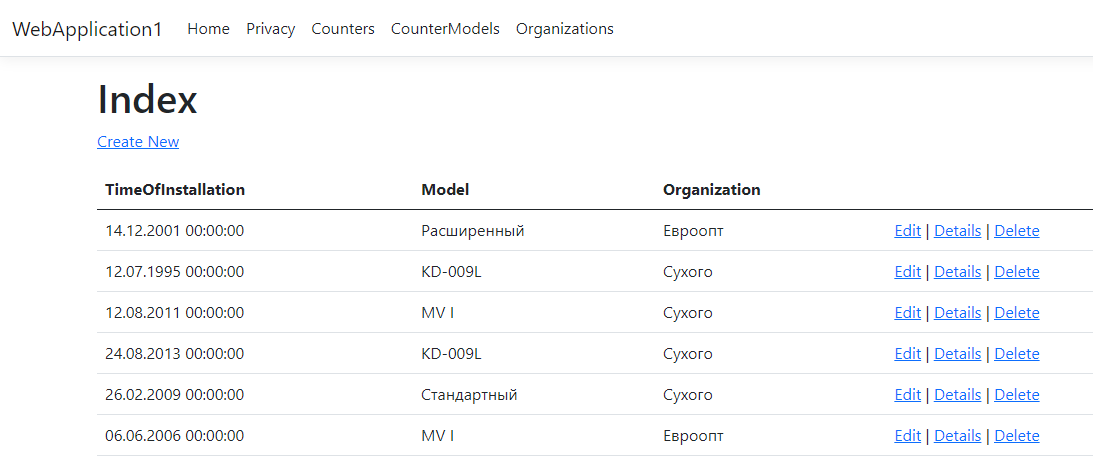


Рисунок 2 – Таблица счетчиков

В приложении А показан листинг контроллеров, а в приложении Б листинг генератора данных.

**Github:** https://github.com/OLagsV/OrganizationsWaterSupplyL4

**Вывод:** в результате выполнения лабораторной работы было разработано *ASP.NET Core MVC Web*-приложение, содержащее набор классов, моделирующих предметную область, и осуществляющих генерацию и заполнение тестовыми наборами записей базу данных с помощью *Entity Framework*.

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

Листинг контроллеров

***CounterModelsController*** ***.cs:***

вы using System;

using System.Collections.Generic;

using System.Diagnostics.Metrics;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using OrganizationsWaterSupplyL4.Data;

using OrganizationsWaterSupplyL4.Models;

using OrganizationsWaterSupplyL4.Services;

namespace OrganizationsWaterSupplyL4.Controllers

{

public class CounterModelsController : Controller

{

private readonly OrganizationsWaterSupplyContext \_context;

CachedModelsService \_modelsService;

List<CounterModel> \_models;

public CounterModelsController(OrganizationsWaterSupplyContext context)

{

\_context = context;

\_modelsService = (CachedModelsService)context.GetService<ICachedService<CounterModel>>();

\_models = \_modelsService.GetData("models").ToList();

}

// GET: CounterModels

public async Task<IActionResult> Index()

{

\_models = \_modelsService.GetData("models").ToList();

var organizationsWaterSupplyContext = \_models;

return View(organizationsWaterSupplyContext);

/\*return \_context.CounterModels != null ?

View(await \_context.CounterModels.ToListAsync()) :

Problem("Entity set 'OrganizationsWaterSupplyContext.CounterModels' is null.");\*/

}

// GET: CounterModels/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null || \_context.CounterModels == null)

{

return NotFound();

}

var counterModel = await \_context.CounterModels

.FirstOrDefaultAsync(m => m.ModelId == id);

if (counterModel == null)

{

return NotFound();

}

return View(counterModel);

}

// GET: CounterModels/Create

public IActionResult Create()

{

return View();

}

// POST: CounterModels/Create

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("ModelId,ModelName,Manufacturer,ServiceTime")] CounterModel counterModel)

{

if (ModelState.IsValid)

{

\_context.Add(counterModel);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(counterModel);

}

// GET: CounterModels/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null || \_context.CounterModels == null)

{

return NotFound();

}

var counterModel = await \_context.CounterModels.FindAsync(id);

if (counterModel == null)

{

return NotFound();

}

return View(counterModel);

}

// POST: CounterModels/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("ModelId,ModelName,Manufacturer,ServiceTime")] CounterModel counterModel)

{

if (id != counterModel.ModelId)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(counterModel);

await \_context.SaveChangesAsync();

\_modelsService.UpdateData("models");

}

catch (DbUpdateConcurrencyException)

{

if (!CounterModelExists(counterModel.ModelId))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

return View(counterModel);

}

// GET: CounterModels/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null || \_context.CounterModels == null)

{

return NotFound();

}

var counterModel = await \_context.CounterModels

.FirstOrDefaultAsync(m => m.ModelId == id);

if (counterModel == null)

{

return NotFound();

}

return View(counterModel);

}

// POST: CounterModels/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

if (\_context.CounterModels == null)

{

return Problem("Entity set 'OrganizationsWaterSupplyContext.CounterModels' is null.");

}

var counterModel = await \_context.CounterModels.FindAsync(id);

if (counterModel != null)

{

\_context.CounterModels.Remove(counterModel);

}

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool CounterModelExists(int id)

{

return (\_context.CounterModels?.Any(e => e.ModelId == id)).GetValueOrDefault();

}

}

}

***CountersController.cs:***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using OrganizationsWaterSupplyL4.Data;

using OrganizationsWaterSupplyL4.Services;

using OrganizationsWaterSupplyL4.Models;

using Microsoft.EntityFrameworkCore.Infrastructure;

namespace OrganizationsWaterSupplyL4.Controllers

{

public class CountersController : Controller

{

private readonly OrganizationsWaterSupplyContext \_context;

CachedCountersService \_countersService;

List<Counter> \_counters;

public CountersController(OrganizationsWaterSupplyContext context)

{

\_context = context;

\_countersService = (CachedCountersService)context.GetService<ICachedService<Counter>>();

\_counters = \_countersService.GetData("counters").ToList();

}

// GET: Counters

public async Task<IActionResult> Index()

{

\_counters = \_countersService.GetData("counters").ToList();

var organizationsWaterSupplyContext = \_counters;

/\*var organizationsWaterSupplyContext = \_context.Counters.Include(c => c.Model).Include(c => c.Organization)\*/;

return View(organizationsWaterSupplyContext);

}

// GET: Counters/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null || \_context.Counters == null)

{

return NotFound();

}

var counter = await \_context.Counters

.Include(c => c.Model)

.Include(c => c.Organization)

.FirstOrDefaultAsync(m => m.RegistrationNumber == id);

if (counter == null)

{

return NotFound();

}

return View(counter);

}

// GET: Counters/Create

public IActionResult Create()

{

ViewData["ModelId"] = new SelectList(\_context.CounterModels, "ModelId", "ModelId");

ViewData["OrganizationId"] = new SelectList(\_context.Organizations, "OrganizationId", "OrganizationId");

return View();

}

// POST: Counters/Create

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("RegistrationNumber,ModelId,TimeOfInstallation,OrganizationId")] Counter counter)

{

if (ModelState.IsValid)

{

\_context.Add(counter);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

ViewData["ModelId"] = new SelectList(\_context.CounterModels, "ModelId", "ModelId", counter.ModelId);

ViewData["OrganizationId"] = new SelectList(\_context.Organizations, "OrganizationId", "OrganizationId", counter.OrganizationId);

return View(counter);

}

// GET: Counters/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null || \_context.Counters == null)

{

return NotFound();

}

var counter = await \_context.Counters.FindAsync(id);

if (counter == null)

{

return NotFound();

}

ViewData["ModelId"] = new SelectList(\_context.CounterModels, "ModelId", "ModelName", counter.ModelId);

ViewData["OrganizationId"] = new SelectList(\_context.Organizations, "OrganizationId", "OrgName", counter.OrganizationId);

return View(counter);

}

// POST: Counters/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("RegistrationNumber,ModelId,TimeOfInstallation,OrganizationId")] Counter counter)

{

if (id != counter.RegistrationNumber)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(counter);

await \_context.SaveChangesAsync();

\_countersService.UpdateData("counters");

}

catch (DbUpdateConcurrencyException)

{

if (!CounterExists(counter.RegistrationNumber))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

ViewData["ModelId"] = new SelectList(\_context.CounterModels, "ModelId", "ModelId", counter.ModelId);

ViewData["OrganizationId"] = new SelectList(\_context.Organizations, "OrganizationId", "OrganizationId", counter.OrganizationId);

return View(counter);

}

// GET: Counters/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null || \_context.Counters == null)

{

return NotFound();

}

var counter = await \_context.Counters

.Include(c => c.Model)

.Include(c => c.Organization)

.FirstOrDefaultAsync(m => m.RegistrationNumber == id);

if (counter == null)

{

return NotFound();

}

return View(counter);

}

// POST: Counters/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

if (\_context.Counters == null)

{

return Problem("Entity set 'OrganizationsWaterSupplyContext.Counters' is null.");

}

var counter = await \_context.Counters.FindAsync(id);

if (counter != null)

{

\_context.Counters.Remove(counter);

}

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool CounterExists(int id)

{

return (\_context.Counters?.Any(e => e.RegistrationNumber == id)).GetValueOrDefault();

}

}

}

***CountersController.cs:***

using System;

using System.Collections.Generic;

using System.Diagnostics.Metrics;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using OrganizationsWaterSupplyL4.Data;

using OrganizationsWaterSupplyL4.Models;

using OrganizationsWaterSupplyL4.Services;

namespace OrganizationsWaterSupplyL4.Controllers

{

public class OrganizationsController : Controller

{

private readonly OrganizationsWaterSupplyContext \_context;

CachedOrganizationsService \_organizationsService;

List<Organization> \_organizations;

public OrganizationsController(OrganizationsWaterSupplyContext context)

{

\_context = context;

\_organizationsService = (CachedOrganizationsService)context.GetService<ICachedService<Organization>>();

\_organizations = \_organizationsService.GetData("organizations").ToList();

}

// GET: Organizations

public async Task<IActionResult> Index()

{

\_organizations = \_organizationsService.GetData("organizations").ToList();

var organizationsWaterSupplyContext = \_organizations;

return View(organizationsWaterSupplyContext);

/\*return \_context.Organizations != null ?

View(await \_context.Organizations.ToListAsync()) :

Problem("Entity set 'OrganizationsWaterSupplyContext.Organizations' is null.");\*/

}

// GET: Organizations/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null || \_context.Organizations == null)

{

return NotFound();

}

var organization = await \_context.Organizations

.FirstOrDefaultAsync(m => m.OrganizationId == id);

if (organization == null)

{

return NotFound();

}

return View(organization);

}

// GET: Organizations/Create

public IActionResult Create()

{

return View();

}

// POST: Organizations/Create

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("OrganizationId,OrgName,OwnershipType,Adress,DirectorFullname,DirectorPhone,ResponsibleFullname,ResponsiblePhone")] Organization organization)

{

if (ModelState.IsValid)

{

\_context.Add(organization);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(organization);

}

// GET: Organizations/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null || \_context.Organizations == null)

{

return NotFound();

}

var organization = await \_context.Organizations.FindAsync(id);

if (organization == null)

{

return NotFound();

}

return View(organization);

}

// POST: Organizations/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("OrganizationId,OrgName,OwnershipType,Adress,DirectorFullname,DirectorPhone,ResponsibleFullname,ResponsiblePhone")] Organization organization)

{

if (id != organization.OrganizationId)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(organization);

await \_context.SaveChangesAsync();

\_organizationsService.UpdateData("organizations");

}

catch (DbUpdateConcurrencyException)

{

if (!OrganizationExists(organization.OrganizationId))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

return View(organization);

}

// GET: Organizations/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null || \_context.Organizations == null)

{

return NotFound();

}

var organization = await \_context.Organizations

.FirstOrDefaultAsync(m => m.OrganizationId == id);

if (organization == null)

{

return NotFound();

}

return View(organization);

}

// POST: Organizations/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

if (\_context.Organizations == null)

{

return Problem("Entity set 'OrganizationsWaterSupplyContext.Organizations' is null.");

}

var organization = await \_context.Organizations.FindAsync(id);

if (organization != null)

{

\_context.Organizations.Remove(organization);

}

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool OrganizationExists(int id)

{

return (\_context.Organizations?.Any(e => e.OrganizationId == id)).GetValueOrDefault();

}

}

}

**ПРИЛОЖЕНИЕ Б**

**(обязательное)**

Листинг *DbInitializer*

using System;

using System.Diagnostics;

using System.Diagnostics.Metrics;

using System.Drawing.Text;

using OrganizationsWaterSupplyL4.Data;

using OrganizationsWaterSupplyL4.Models;

namespace OrganizationsWaterSupplyL4.Middleware

{

public class DbInitializer

{

public static void Initialize(OrganizationsWaterSupplyContext db)

{

db.Database.EnsureCreated();

int countermodel\_num = 50;

int organization\_num = 50;

int counters\_num = 100;

int rates\_num = 50;

int models\_num = 20;

Random rand = new Random(1);

string[] org\_names = { "ГомСельМаш", "Брилево", "Сухого", "Евроопт", "Кари", "Platoon" };

string[] ownership\_types = { "Индивидуальный предприниматель", "Унитарное предприятие", "ОАО", "ООО", "ОДО" };

string[] countries = { "Беларусь", "Германия", "Польша", "Япония", "Франция", "США" };

string[] towns = { "Гомель", "Минск", "Дрезден", "Токио", "Париж", "Вашингтон", "Новиград" };

string[] streets = { "Чечерская", "Фадеева", "Яблочная", "Пенязькова", "Советская" };

string[] first\_names = {"Владимир", "Михаил" , "Григорий" , "Анжела" , "Виктория" , "Олег" , "Дмитрий" , "Крубер"};

string[] last\_names = { "Петрик", "Никитин", "Гордеев", "Иващенко", "Бобров", "Гордеев", "Богданов" };

string[] model\_names = { "MK II", "Стандартный", "Расширенный", "MV I", "KD-009L" };

string[] manufacturers = { "ГомельАква", "МосВодоКанал", "Сож", "Японская ГЭС"};

string[] rates = { "Субсидируемый", "Полный" };

string[] results = { "Задолженность", "Оплачено", "Нет данных" };

string RandomDigits(int length)

{

var random = new Random();

string s = string.Empty;

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

s = String.Concat(s, random.Next(10).ToString());

return s;

}

DateTime RandomDay()

{

DateTime start = new DateTime(1995, 1, 1);

int range = (DateTime.Today - start).Days;

return start.AddDays(rand.Next(range));

}

if (!db.CounterModels.Any())

{

for (int i = 0; i < countermodel\_num; i++)

{

db.CounterModels.Add(new CounterModel {

ModelName = model\_names[rand.Next(0,model\_names.Length)],

Manufacturer = manufacturers[rand.Next(0, manufacturers.Length)],

ServiceTime = rand.Next(4, 18)

});

}

db.SaveChanges();

}

if (!db.Organizations.Any())

{

for (int i = 0; i < organization\_num; i++)

{

db.Organizations.Add(new Organization

{

OrgName = org\_names[rand.Next(0, org\_names.Length)],

OwnershipType = ownership\_types[rand.Next(0, ownership\_types.Length)],

Adress = countries[rand.Next(0, countries.Length)] + ", " + towns[rand.Next(1, towns.Length)] + ", " + streets[rand.Next(0, streets.Length)] + " " + rand.Next(1, 120),

DirectorFullname = first\_names[rand.Next(0, first\_names.Length)] + " " + last\_names[rand.Next(0, last\_names.Length)],

DirectorPhone = RandomDigits(10),

ResponsibleFullname = first\_names[rand.Next(0, first\_names.Length)] + " " + last\_names[rand.Next(0, last\_names.Length)],

ResponsiblePhone = RandomDigits(10),

});

}

db.SaveChanges();

}

if (!db.Rates.Any())

{

for (int i = 0; i < rates\_num; i++)

{

db.Rates.Add(new Rate

{

RateName = rates[rand.Next(0, rates.Length)] + ", " + towns[rand.Next(0, towns.Length)],

Price = rand.Next(15,60)

});

}

db.SaveChanges();

}

if (!db.Counters.Any())

{

for (int i = 0; i < counters\_num; i++)

{

db.Counters.Add(new Counter

{

ModelId = rand.Next(1, models\_num),

TimeOfInstallation = RandomDay(),

OrganizationId = rand.Next(1, organization\_num),

});

}

db.SaveChanges();

}

if (!db.CountersChecks.Any())

{

for (int i = 0; i < counters\_num; i++)

{

db.CountersChecks.Add(new CountersCheck

{

CounterRegistrationNumber = rand.Next(1, counters\_num),

CheckDate = RandomDay(),

CheckResult = results[rand.Next(0, results.Length)],

});

}

db.SaveChanges();

}

if (!db.CountersData.Any())

{

for (int i = 0; i < counters\_num; i++)

{

db.CountersData.Add(new CountersDatum

{

CounterRegistrationNumber = rand.Next(1, counters\_num),

DataCheckDate = RandomDay(),

Volume = rand.Next(400, 1200)

});

}

db.SaveChanges();

}

if (!db.RateOrgNotes.Any())

{

for (int i = 0; i < organization\_num; i++)

{

db.RateOrgNotes.Add(new RateOrgNote

{

RateId = rand.Next(1, rates\_num),

OrganizationId = rand.Next(1, organization\_num),

});

}

db.SaveChanges();

}

}

}

}