МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

ОТЧЁТ ПО ТЕХНОЛОГИЧЕСКОЙ ПРАКТИКЕ ПО СПЕЦИАЛЬНОСТИ «АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ» НА ОАО «МОГИЛЁВЛИФТМАШ»

1-53 01 02.10028326.17.81-01

Исполнитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тищенко Кристина Геннадьевна рррррррррррррррррррррррррррррррррррррррррррррррррррррррАСОИ-181

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тимашкова Людмила Александровна

(подпись)

Дата допуска к защите \_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_

Могилев 2021

Содержание

|  |  |
| --- | --- |
| Введение…………………………………………………………………………….. | 3 |
| 1 Краткая характеристика предприятия………….……………………………….. | 4 |
| 2 Описание выполненной работы…………………………………………………. | 6 |
| Заключение…………………………………………………………………………. | 7 |
| Приложение ……………………………………………............................................ | 8 |

Введение

Технологическая практика была пройдена на предприятии ОАО «Могилёвлифтмаш».

Объект исследования – ОРПО ОАО «Могилёвлифтмаш».

Индивидуальным заданием на практику была поставлена разработка приложения «Номенклатурный справочник» для использования в производстве на ОАО «Магилёвлифтмаш».

Для выполнения данного задания потребовалась следующая информация:

* Какие СУБД используются на предприятии
* Какие поля необходимы для БД и корректного использования на предприятии в работе.
* Какие языки программирования предпочтительнее использовать в ходе работы

Отчёт по технологической практике состоит из введения, двух параграфов, заключения и приложения.

1 Краткая характеристика предприятия

ОАО «Могилёвлифтмаш» — советский и белорусский производитель [лифтов](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%84%D1%82). Предприятие, основанное в [1966 году](https://ru.wikipedia.org/wiki/1966_%D0%B3%D0%BE%D0%B4) (постановлением Совета Министров СССР № 835 "О мероприятиях по увеличению производства лифтов и повышению их качества"), расположено в городе [Могилёв](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B3%D0%B8%D0%BB%D1%91%D0%B2) ([Белоруссия](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D0%BB%D0%BE%D1%80%D1%83%D1%81%D1%81%D0%B8%D1%8F)). С 2014 года ОАО "Могилевлифтмаш" стало управляющей компанией холдинга "Могилевлифтмаш". В состав холдинга входят следующие предприятия: ОАО "Могилевлифтмаш" (г. Могилев); ОАО «[Могилевский завод „Электродвигатель“](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%B4%D0%B2%D0%B8%D0%B3%D0%B0%D1%82%D0%B5%D0%BB%D1%8C_(%D0%B7%D0%B0%D0%B2%D0%BE%D0%B4,_%D0%9C%D0%BE%D0%B3%D0%B8%D0%BB%D1%91%D0%B2))» (г. Могилев); ОАО "Зенит" (г. Могилев);ОАО "Завод "Ветразь" (Витебская обл., Докшицкий район, г.п. Бегомоль).

Продукция включает в себя пассажирские, больничные и грузовые лифты (130 моделей) грузоподъемностью от 100 до 6300 кг, а также различные нестандартные лифты. Кроме лифтов выпускаются платформы различного типа для физически ослабленных лиц, эскалаторы и траволаторы, строительные подъемники, измельчители мелколесья (мульчировщики), потребительские товары.

ОАО «Могилевлифтмаш» является лауреатом следующих Премий в области качества:

* "Лучшие товары Республики Беларусь" 2004, 2006, 2008, 2013, 2014, 2016, 2017, 2018, 2019 год
* «Премия Правительства Республики Беларусь за достижения в области качества» за 2004, 2007, 2012, 2017 годы (специальная награда за 2017 год в номинации «Совершенство менеджмента» за построение в организации систем менеджмента, ориентированных на проектно-процессное управление и позволяющих внедрять современные высокоэффективные, ресурсосберегающие и инновационные технологии, создавать конкурентоспособную продукцию, работы (услуги);
* «Премия Министерства промышленности Республики Беларусь за достижение в области качества» - 2003, 2006, 2009, 2013, 2017 год;
* «Лучшие товары Республики Беларусь» - 2004, 2006, 2008, 2010, 2013, 2014, 2015, 2016, 2017, 2018 год.
* «Лучшие товары Республики Беларусь на рынке Российской Федерации» - 2004, 2006, 2008, 2010, 2013, 2015 год;
* -«Лучший экспортер Республики Беларусь» - 2012 год.

Специальные награды за 2017 год:

* «Лучший в регионе» - Лебедка сельскохозяйственная бытовая ЛС-100А;
* «Лучший в отрасли» - Лифт пассажирский ЛП-0401БМЭ.

Специальные награды за 2018 год:

* -«Лучший в регионе» - Машина деревообрабатывающая бытовая ИЭ-6009А4.2-02;
* -«Стабильное качество» - Машина деревообрабатывающая бытовая ИЭ-6009А4.2-02.
* -«Стабильное качество» - Лифт пассажирский ЛП-0463 БМЭ.

Специальные награды за 2019 год:

* "Лучший в регионе" - Лифт пассажирский ЛП-0401К, Лебедка сельскохозяйственная бытовая ЛС-100А;
* "Лучший в отрасли" - Лифт пассажирский ЛП-0401К, Лебедка сельскохозяйственная бытовая ЛС-100А;
* "Стабильное качество" - Лифт пассажирский ЛП-0401К, Лебедка сельскохозяйственная бытовая ЛС-100А).

2 Описание выполненной работы

В ходе практики было разработано приложение, позволяющее работать с номенклатурой. В полях содержится вся необходимая информация по материалам и деталям, которые используются в производстве.

В результате выполненной работы получили приложение следующего плана:

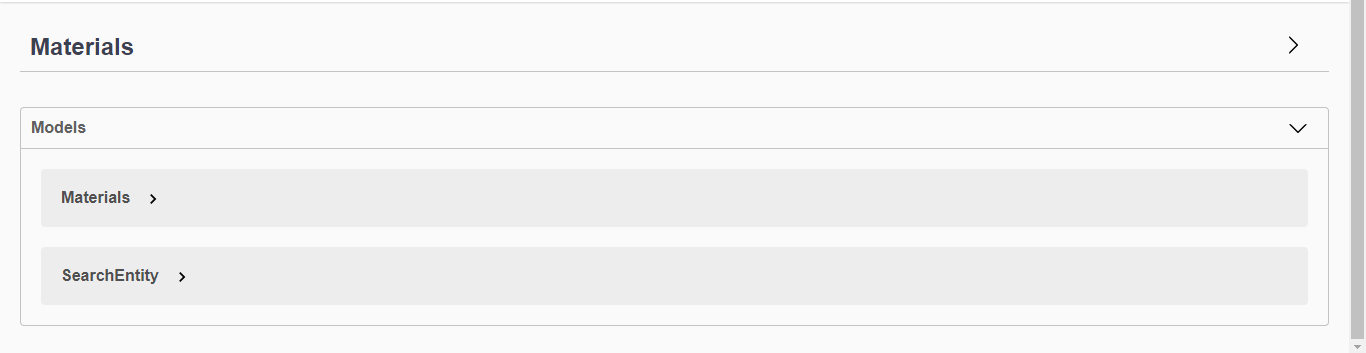


Рисунок 1 – Сущность материалов и деталей, модели всех сущностей в БД

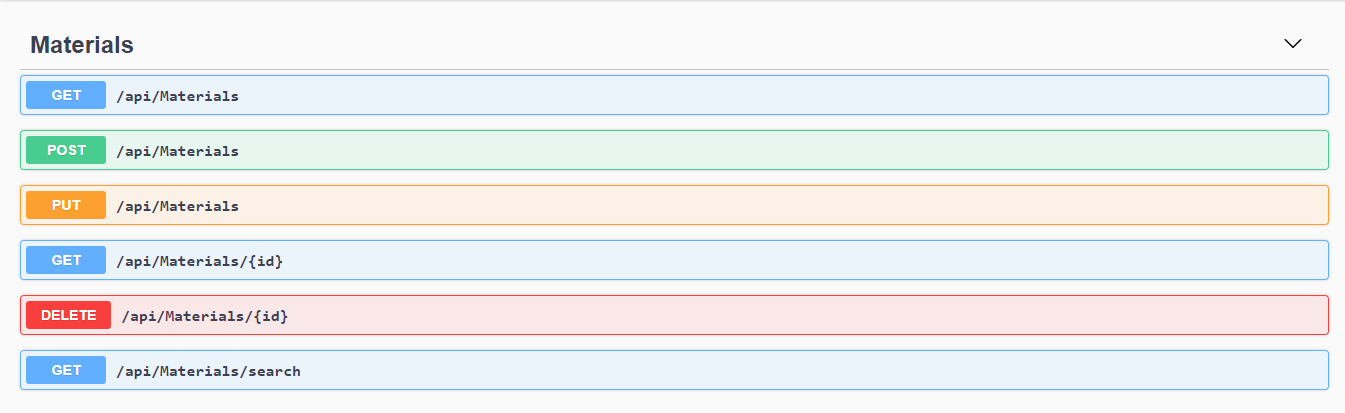


Рисунок 2 – Перечень всех CRUD запросов, с помощью которых можно осуществлять работу с данными.

Код программы представлен в Приложении.

Заключение

В настоящее время становится все труднее и труднее манипулировать окружающими нас вещами. Для этого сегодня и существуют технологии автоматизации. Мало осталось вещей которые нельзя было автоматизировать. Людям все время облегчается работа с информацией. Раньше некоторые могли сидеть за одним делом часами, а то и днями. Но настали времена когда стоит приложить маленькие усилие и все получится даже лучше чем планировалось.

Во время прохождения технологической практики было разработано приложение «Номенклатурный справочник», который предназначен для использования в производстве на ОАО «Могилёвлифтмаш».

При создании ИС использовались: сервер баз данных Microsoft SQL server 2018, язык баз данных T-SQL, среда разработки Visual Studio и язык программирования C#.

Все поставленные задачи были выполнены, программа выполняет поставленную перед ней задачу.

Приложение

using Liftmash.Domain.Abstaractions.Repositories;

using Liftmash.Domain.Entities;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace Liftmash.Infrasructure.Data

{

public class MaterialsRepository : IMaterialsRepository

{

private readonly DataContext dataContext;

public MaterialsRepository(DataContext dataContext)

{

this.dataContext = dataContext ?? throw new ArgumentNullException(nameof(dataContext));

}

public async Task<IList<Materials>> GetAll()

{

return await this.dataContext.Materials.ToListAsync();

}

public async Task<Materials> GetById(long id)

{

return await this.dataContext.Materials.FindAsync(id);

}

public async Task<IList<Materials>> GetByCriteria(SearchEntity material)

{

return await this.dataContext.Materials.Where(b =>

((string.IsNullOrEmpty(material.Name) && string.IsNullOrEmpty(material.Article) && string.IsNullOrEmpty(material.Code)) ||

(!string.IsNullOrEmpty(material.Name) && string.IsNullOrEmpty(material.Article) && string.IsNullOrEmpty(material.Code)) ||

(string.IsNullOrEmpty(material.Name) && !string.IsNullOrEmpty(material.Article) && string.IsNullOrEmpty(material.Code)) ||

(string.IsNullOrEmpty(material.Name) && string.IsNullOrEmpty(material.Article) && !string.IsNullOrEmpty(material.Code))))

.ToListAsync();

}

public async Task Add(Materials materials)

{

this.dataContext.Materials.Add(materials);

await this.dataContext.SaveChangesAsync();

}

public async Task Delete(long id)

{

this.dataContext.Materials.Remove(this.dataContext.Materials.Find(id));

await this.dataContext.SaveChangesAsync();

}

public async Task Update(Materials materials)

{

await Task.Run(() => dataContext.Materials.Attach(materials));

dataContext.Entry(materials).State = EntityState.Modified;

await dataContext.SaveChangesAsync();

}

}

}

using Liftmash.Domain.Entities;

using Microsoft.EntityFrameworkCore;

namespace Liftmash.Infrasructure.Data

{

public class DataContext :DbContext

{

public DataContext(DbContextOptions<DataContext> options)

: base(options)

{

Database.EnsureCreated();

}

public DbSet<Materials> Materials { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Materials>(e => e.HasIndex(p => p.Name));

modelBuilder.Entity<Materials>(e => e.HasIndex(p => p.Article));

modelBuilder.Entity<Materials>(e => e.HasIndex(p => p.Code));

}

}

}

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace Liftmash.Domain.Entities

{

public class SearchEntity

{

public string Name { get; set; }

public string Code { get; set; }

public string Article { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace Liftmash.Domain.Entities

{

public class Materials

{

public long Id { get; set; }

[Column("Код")]

public string Code { get; set; }

[Column("Артикул")]

public string Article { get; set; }

[StringLength(200)]

[Column("Наименование")]

public string Name { get; set; }

[Column("Единица измерения")]

public string Unit { get; set; }

[Column("Количество")]

public long Count { get; set; }

[Column("Цена",TypeName = "money")]

public decimal Price { get; set; }

}

}

using Liftmash.Domain.Entities;

using System.Collections.Generic;

using System.Threading.Tasks;

namespace Liftmash.Domain.Abstaractions.Repositories

{

public interface IMaterialsRepository

{

public Task<IList<Materials>> GetAll();

public Task<Materials> GetById(long id);

public Task<IList<Materials>> GetByCriteria(SearchEntity material);

public Task Add(Materials materials);

public Task Update(Materials materials);

public Task Delete(long id);

}

}

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Liftmash.Infrasructure.Data;

using Microsoft.EntityFrameworkCore;

using Liftmash.Domain.Abstaractions.Repositories;

namespace Liftmash.Api

{

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.AddCors();

services.AddControllersWithViews();

services.AddScoped<IMaterialsRepository, MaterialsRepository>();

services.AddDbContext<DataContext>(options => options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

services.AddSwaggerDocument(config =>

{

config.PostProcess = document =>

{

document.Info.Version = "v1";

document.Info.Title = "ToDo API";

document.Info.Description = "A simple ASP.NET Core web API";

document.Info.TermsOfService = "None";

document.Info.Contact = new NSwag.OpenApiContact

{

Name = "Shayne Boyer",

Email = string.Empty,

Url = "https://twitter.com/spboyer"

};

document.Info.License = new NSwag.OpenApiLicense

{

Name = "Use under LICX",

Url = "https://example.com/license"

};

};

});

}

// 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("/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.UseOpenApi();

app.UseSwaggerUi3();

app.UseRouting();

app.UseCors(builder => builder

.AllowAnyOrigin()

.AllowAnyHeader()

.AllowAnyMethod()

);

app.UseHttpsRedirection();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

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

});

}

}

}

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace Liftmash.Api

{

public class Program

{

public static void Main(string[] args)

{

CreateHostBuilder(args).Build().Run();

}

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

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

}

}

using Liftmash.Domain.Entities;

using Liftmash.Domain.Abstaractions.Repositories;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Threading.Tasks;

namespace Liftmash.Api.Controllers

{

[Route("api/Materials")]

[ApiController]

public class MaterialsController : Controller

{

private readonly IMaterialsRepository materialsRepository;

public MaterialsController(IMaterialsRepository materialsRepository)

{

this.materialsRepository = materialsRepository ?? throw new ArgumentNullException(nameof(materialsRepository));

}

[HttpGet]

public async Task<IList<Materials>> GetAll()

{

return await this.materialsRepository.GetAll();

}

[HttpGet("{id:long}")]

[ProducesResponseType(StatusCodes.Status200OK)]

public async Task<ActionResult<Materials>> GetById([FromRoute] long id)

{

return this.Ok(await this.materialsRepository.GetById(id));

}

[HttpGet]

[Route("search")]

public async Task<IList<Materials>> Search([FromBody] SearchEntity parametr)

{

return await this.materialsRepository.GetByCriteria(parametr);

}

[HttpPost]

public async Task Post([FromBody] Materials materials)

{

await this.materialsRepository.Add(materials);

}

[HttpPut]

public async Task Put(Materials materials)

{

await this.materialsRepository.Update(materials);

}

[HttpDelete("{id:long}")]

public async Task Delete([FromRoute] long id)

{

await this.materialsRepository.Delete(id);

}

}

}