ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

"ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"

Факультет ИСП

Кафедра ПИ

Лабораторная работа №4

по курсу: «Профессиональная практика программной инженерии»

по теме: «Создание самодокументирующегося кода»

Выполнил:

ст. гр. ПИ-19а

Истягин Алексей

Проверил:

Филипишин Д.А.

Ищенко А.П.

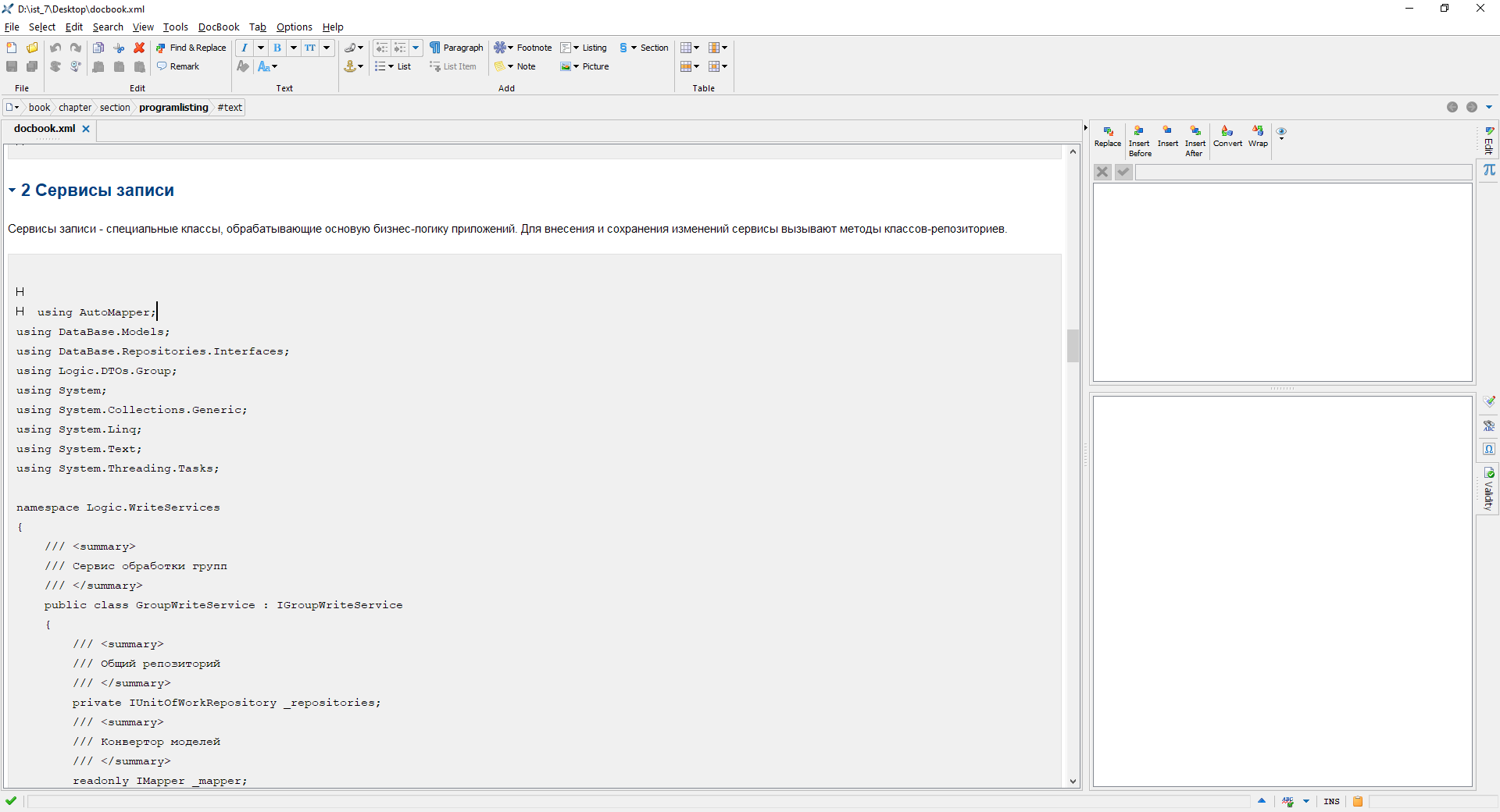
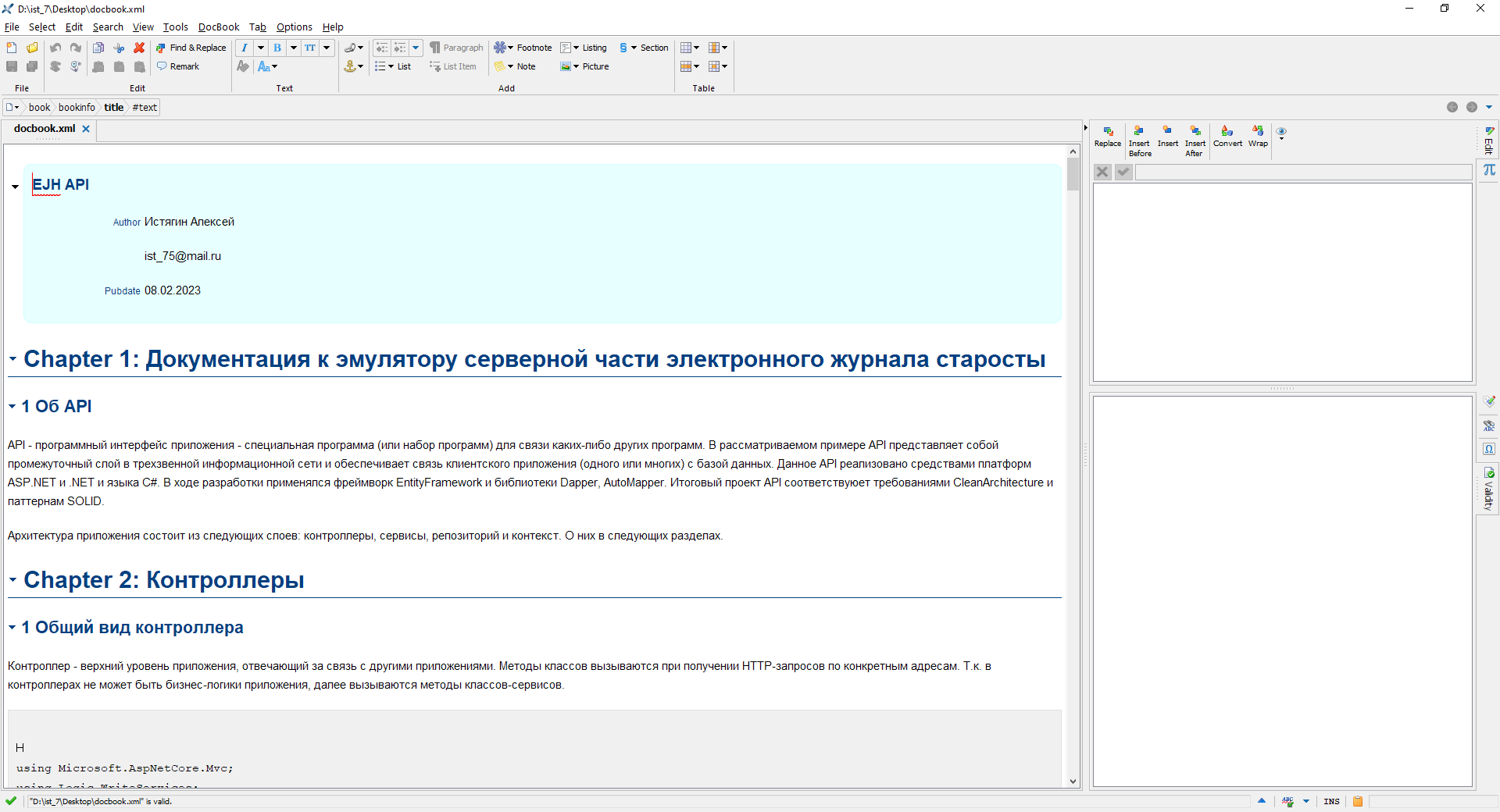
ДОНЕЦК – 2023

Примеры DocBlock:

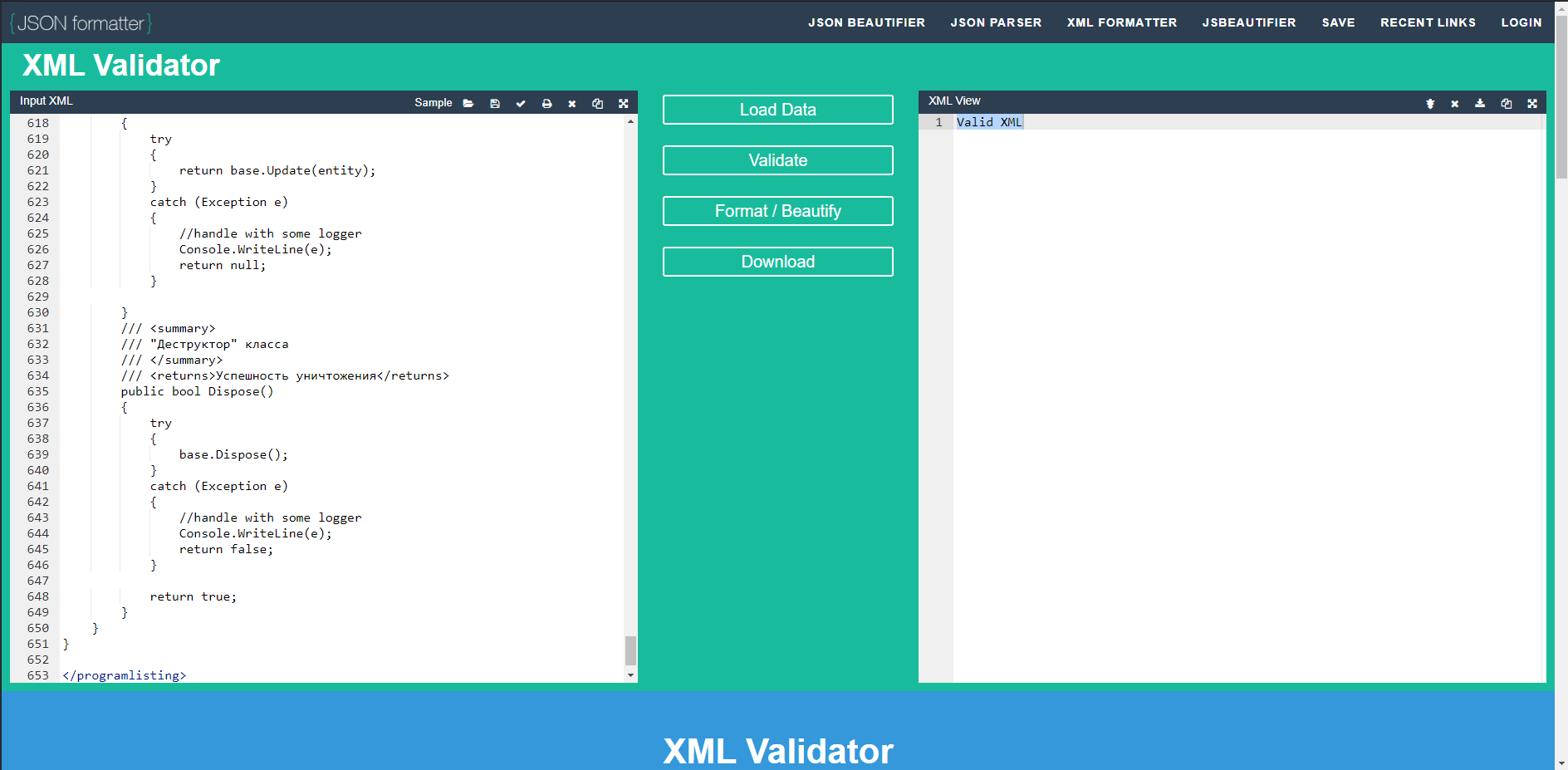
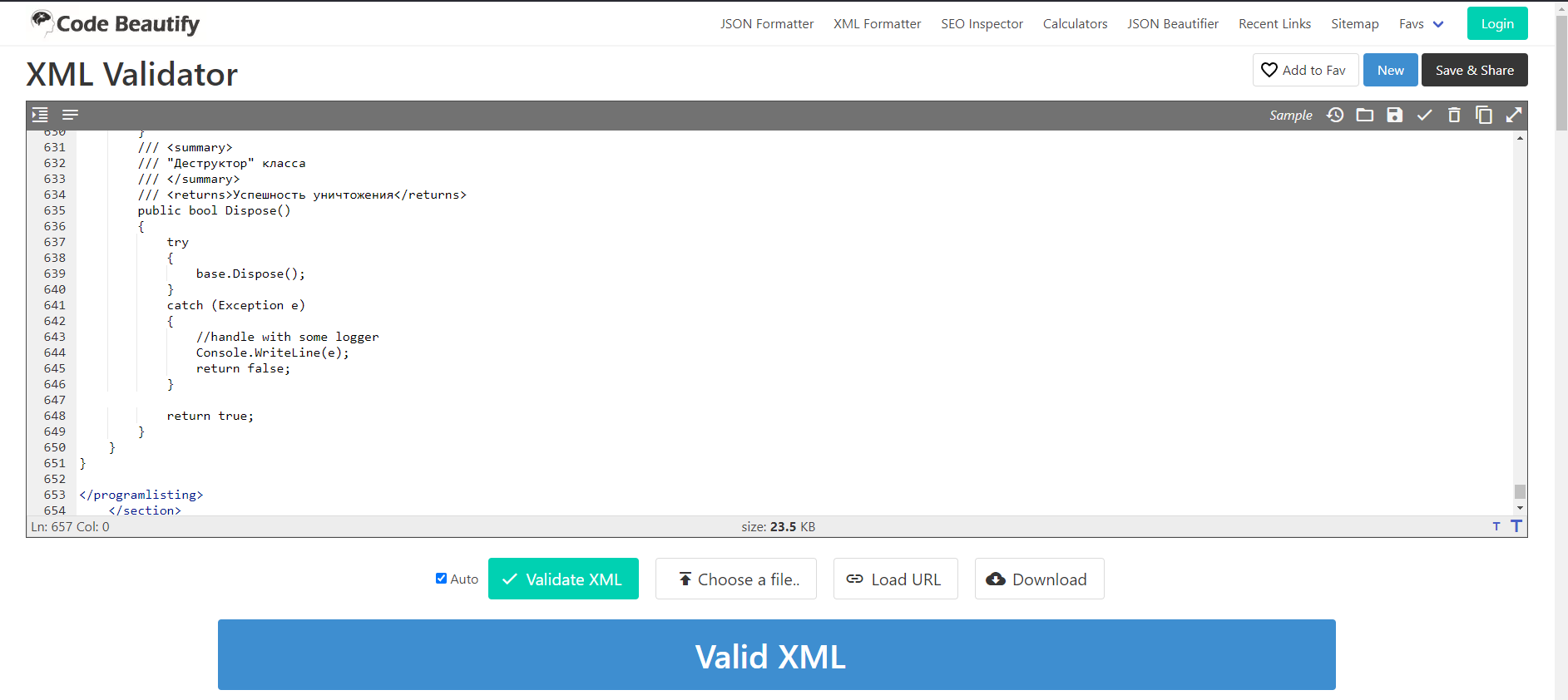




Скриншоты DocBook:



Проверка корректности:



Листинг DocBook:

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.5//EN"

"http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd">

<book>

<bookinfo>

<title>EJH API</title>

<author>

<firstname>Истягин</firstname>

<surname>Алексей</surname>

<affiliation>

<orgname>ist\_75@mail.ru</orgname>

</affiliation>

</author>

<pubdate>08.02.2023</pubdate>

</bookinfo>

<chapter>

<title>Документация к эмулятору серверной части электронного журнала

старосты</title>

<section>

<title>Об API</title>

<para>API - программный интерфейс приложения - специальная программа

(или набор программ) для связи каких-либо других программ. В

рассматриваемом примере API представляет собой промежуточный слой в

трехзвенной информационной сети и обеспечивает связь клиентского

приложения (одного или многих) с базой данных. Данное API реализовано

средствами платформ ASP.NET и .NET и языка C#. В ходе разработки

применялся фреймворк EntityFramework и библиотеки Dapper, AutoMapper.

Итоговый проект API соответствуюет требованиями CleanArchitecture и

паттернам SOLID.</para>

<para>Архитектура приложения состоит из следующих слоев: контроллеры,

сервисы, репозиторий и контекст. О них в следующих разделах.</para>

</section>

</chapter>

<chapter>

<title>Контроллеры</title>

<section>

<title>Общий вид контроллера</title>

<para>Контроллер - верхний уровень приложения, отвечающий за связь с

другими приложениями. Методы классов вызываются при получении

HTTP-запросов по конкретным адресам. Т.к. в контроллерах не может быть

бизнес-логики приложения, далее вызываются методы

классов-сервисов.</para>

<programlisting>

using Microsoft.AspNetCore.Mvc;

using Logic.WriteServices;

using Logic.DTOs.Group;

using Logic.ReadServices.Interfaces;

using DataBase.Contexts;

// For more information on enabling Web API for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=397860

namespace API.Controllers

{

/// ˂summary˃

/// HTTP-контроллер для групп

/// ˂/summary˃

[Authorization.Attributes.Authorize( Roles.Admin )]

[Route("api/[controller]")]

[ApiController]

public class GroupController : ControllerBase

{

/// ˂summary˃

/// Сервис записи групп

/// ˂/summary˃

IGroupWriteService \_groupWriteService;

/// ˂summary˃

/// Сервис чтения групп

/// ˂/summary˃

IGroupReadService \_groupReadService;

/// ˂summary˃

/// Конструктор с DI

/// ˂/summary˃

/// ˂param name="groupWriteService"˃Сервис записи групп˂/param˃

/// ˂param name="groupReadService"˃Сервис чтения групп˂/param˃

public GroupController(IGroupWriteService groupWriteService, IGroupReadService groupReadService)

{

\_groupWriteService = groupWriteService;

\_groupReadService = groupReadService;

}

/// ˂summary˃

/// Получение всех групп

/// ˂/summary˃

/// ˂returns˃Все группы˂/returns˃

// GET api/˂GroupController˃

[HttpGet]

[Authorization.Attributes.Authorize( Roles.Student)]

public async Task˂ActionResult˂List˂GetGroupResponse˃˃˃ Get()

{

try

{

return Ok(\_groupReadService.GetAll());

}

catch (Exception ex)

{

return Conflict(ex.Message);

}

}

/// ˂summary˃

/// Получение группы по id

/// ˂/summary˃

/// ˂param name="id"˃id группы˂/param˃

/// ˂returns˃Группа по id˂/returns˃

// GET api/˂GroupController˃

[HttpGet("{id}")]

public async Task˂ActionResult˂GetGroupResponse˃˃ Get(Guid id)

{

try

{

return Ok(\_groupReadService.Get(id));

}

catch (Exception ex)

{

return Conflict(ex.Message);

}

}

/// ˂summary˃

/// Добавить группу

/// ˂/summary˃

/// ˂param name="createGroupRequest"˃Модель группы˂/param˃

/// ˂returns˃Статус запроса˂/returns˃

// POST api/˂GroupController˃

[HttpPost]

public async Task˂ActionResult˃ Post([FromBody] CreateGroupRequest createGroupRequest)

{

try

{

\_groupWriteService.Add(createGroupRequest);

return Ok();

}

catch (Exception ex)

{

return Conflict(ex.Message);

}

}

/// ˂summary˃

/// Обновление группы по id

/// ˂/summary˃

/// ˂param name="id"˃id группы˂/param˃

/// ˂param name="updateGroupRequest"˃Новые данные˂/param˃

/// ˂returns˃Статус запроса˂/returns˃

// PUT api/˂GroupController˃

[HttpPut("{id}")]

public async Task˂ActionResult˃ Put(Guid id, [FromBody] UpdateGroupRequest updateGroupRequest)

{

try

{

\_groupWriteService.Update(id, updateGroupRequest);

return Ok();

}

catch (Exception ex)

{

return Conflict(ex.Message);

}

}

/// ˂summary˃

/// Удалить группу по id

/// ˂/summary˃

/// ˂param name="id"˃id группы˂/param˃

/// ˂returns˃Статус запроса˂/returns˃

// DELETE api/˂GroupController˃

[HttpDelete("{id}")]

public async Task˂ActionResult˃ Delete(Guid id)

{

try

{

\_groupWriteService.Delete(id);

return Ok();

}

catch (Exception ex)

{

return Conflict(ex.Message);

}

}

}

}

</programlisting>

</section>

<section>

<title>Сервисы записи</title>

<para>Сервисы записи - специальные классы, обрабатывающие основую

бизнес-логику приложений. Для внесения и сохранения изменений сервисы

вызывают методы классов-репозиториев.</para>

<programlisting>

using AutoMapper;

using DataBase.Models;

using DataBase.Repositories.Interfaces;

using Logic.DTOs.Group;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Logic.WriteServices

{

/// ˂summary˃

/// Сервис обработки групп

/// ˂/summary˃

public class GroupWriteService : IGroupWriteService

{

/// ˂summary˃

/// Общий репозиторий

/// ˂/summary˃

private IUnitOfWorkRepository \_repositories;

/// ˂summary˃

/// Конвертор моделей

/// ˂/summary˃

readonly IMapper \_mapper;

/// ˂summary˃

/// Конструктор с DI

/// ˂/summary˃

/// ˂param name="repositories"˃Общий репозиторий˂/param˃

/// ˂param name="mapper"˃Конвертор моделей˂/param˃

public GroupWriteService(IUnitOfWorkRepository repositories, IMapper mapper)

{

\_repositories = repositories;

\_mapper = mapper;

}

/// ˂summary˃

/// Создать группу

/// ˂/summary˃

/// ˂param name="createGroupRequest"˃Модель создания группы˂/param˃

public void Add(CreateGroupRequest createGroupRequest)

{

\_repositories.Groups.Add(\_mapper.Map˂Group˃(createGroupRequest));

\_repositories.SaveChanges();

}

/// ˂summary˃

/// Обновить группу по id

/// ˂/summary˃

/// ˂param name="id"˃id группы для обновления˂/param˃

/// ˂param name="updateGroupRequest"˃Модель для обновления˂/param˃

public void Update(Guid id, UpdateGroupRequest updateGroupRequest)

{

Group group = \_repositories.Groups.Get(id);

\_mapper.Map˂UpdateGroupRequest, Group˃(updateGroupRequest, group);

\_repositories.Groups.Update(group);

\_repositories.SaveChanges();

}

/// ˂summary˃

/// Удалить группу по id

/// ˂/summary˃

/// ˂param name="id"˃id группы для удаления˂/param˃

public void Delete(Guid id)

{

Group group = \_repositories.Groups.Get(id);

\_repositories.Groups.Delete(group);

\_repositories.SaveChanges();

}

}

}

</programlisting>

</section>

<section>

<title>Сервисы чтения</title>

<para>Сервисы чтения - специальные классы, читающие данные из базы

данных. Для ускорения работы они вызывают методы классов-запросов, в

которых, в свою очередь, хранятся "сырые" SQL-запросы.</para>

<programlisting>

using Logic.DTOs.Group;

using Logic.Queries.Interfaces;

using Logic.ReadServices.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Logic.ReadServices

{

/// ˂summary˃

/// Сервис чтения групп

/// ˂/summary˃

public class GroupReadService : IGroupReadService

{

/// ˂summary˃

/// Запросы групп

/// ˂/summary˃

IGroupQuery \_groupQuery;

/// ˂summary˃

/// Конструктор с DI

/// ˂/summary˃

/// ˂param name="query"˃Запросы групп˂/param˃

public GroupReadService(IGroupQuery query)

{

\_groupQuery = query;

}

/// ˂summary˃

/// Получить все группы

/// ˂/summary˃

/// ˂returns˃Список групп˂/returns˃

public List˂GetGroupResponse˃ GetAll()

{

return \_groupQuery.GetAll();

}

/// ˂summary˃

/// Получить группу по id

/// ˂/summary˃

/// ˂param name="id"˃id группы˂/param˃

/// ˂returns˃Группа по id˂/returns˃

public GetGroupResponse Get(Guid id)

{

return \_groupQuery.Get(id);

}

}

}

</programlisting>

</section>

<section>

<title>Репозитории</title>

<para>Репозитории - классы, в которых хранятся списки моделей с

"ленивой" подгрузкой. Они объединены в единый репозиторий UnitOfWork,

реализуя одноименный паттерн. В репозиториях допустима минимальная

логика (логгирование, кеширование, сбор статистики), далее они вызывают

методы класса-контекста</para>

<programlisting>

using System.Linq.Expressions;

using DataBase.Contexts;

using DataBase.Models;

using DataBase.Repositories.Interfaces;

using Microsoft.EntityFrameworkCore;

namespace DataBase.Repositories

{

/// ˂summary˃

/// Базовый репозиторий

/// ˂/summary˃

/// ˂typeparam name="TEntity"˃Тип хранимых сущностей˂/typeparam˃

public class BaseRepository˂TEntity˃ : IBaseRepository˂TEntity˃ where TEntity : BaseModel

{

/// ˂summary˃

/// Контекст БД

/// ˂/summary˃

private readonly Context \_context;

/// ˂summary˃

/// Список хранимых сущностей

/// ˂/summary˃

private readonly DbSet˂TEntity˃ \_dbSet;

/// ˂summary˃

/// Конструктор с DI

/// ˂/summary˃

/// ˂param name="context"˃Контекст БД˂/param˃

public BaseRepository(Context context)

{

\_context = context;

\_dbSet = context.Set˂TEntity˃();

}

/// ˂summary˃

/// Добавить модель

/// ˂/summary˃

/// ˂param name="entity"˃Модель˂/param˃

/// ˂returns˃Добавленная модель с трекингом˂/returns˃

public TEntity Add(TEntity entity)

{

entity.DateCreate = DateTime.Now;

entity.DateUpdate = DateTime.Now;

return \_dbSet.Add(entity).Entity;

}

/// ˂summary˃

/// Найти модели по условию

/// ˂/summary˃

/// ˂param name="expression"˃Условие поиска (выражение)˂/param˃

/// ˂returns˃Список найденных моделей˂/returns˃

public IQueryable˂TEntity˃ Where(Expression˂Func˂TEntity, bool˃˃ expression)

{

return \_dbSet.Where(expression);

}

/// ˂summary˃

/// Добавить список моделей

/// ˂/summary˃

/// ˂param name="items"˃Список моделей для добавления˂/param˃

public void AddRange(IEnumerable˂TEntity˃ items)

{

\_dbSet.AddRange(items);

}

/// ˂summary˃

/// Получить все модели

/// ˂/summary˃

/// ˂returns˃Список всех моделей˂/returns˃

public List˂TEntity˃ GetAll()

{

return \_dbSet.ToList();

}

/// ˂summary˃

/// Получить модель по id

/// ˂/summary˃

/// ˂param name="id"˃id модели˂/param˃

/// ˂returns˃Модель по id˂/returns˃

public TEntity? Get(Guid id)

{

return \_dbSet.Where(i =˃ i.Id == id).FirstOrDefault();

}

/// ˂summary˃

/// Обновить модель с трекингом

/// ˂/summary˃

/// ˂param name="entity"˃Модель для обновления˂/param˃

/// ˂returns˃Обновленная модель с трекингом˂/returns˃

public TEntity Update(TEntity entity)

{

entity.DateUpdate = DateTime.Now;

return (TEntity)\_context.Update(entity).Entity;

}

public TEntity Attach(TEntity entity)

{

entity = \_dbSet.Attach(entity).Entity;

\_context.Entry(entity).State = EntityState.Modified;

;

return entity;

}

/// ˂summary˃

/// Удалить модель

/// ˂/summary˃

/// ˂param name="entity"˃Модель для удаления˂/param˃

/// ˂returns˃Удаленная модель˂/returns˃

public TEntity Delete(TEntity entity)

{

return \_context.Remove(entity).Entity;

}

/// ˂summary˃

/// Асинхронное сохранение изменений

/// ˂/summary˃

/// ˂returns˃Результат сохранения˂/returns˃

public Task SaveChangesAsync()

{

return \_context.SaveChangesAsync();

}

/// ˂summary˃

/// Сохранить изменения

/// ˂/summary˃

public void SaveChanges()

{

\_context.SaveChanges();

}

}

}

</programlisting>

</section>

<section>

<title>Контекст</title>

<para>Контекст - класс, имеющий непосредственный доступ к базе данных и

отвечающий за внесение и сохранение изменений.</para>

<programlisting>

using DataBase.Models;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.ChangeTracking;

using Microsoft.Extensions.Configuration;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataBase.Contexts

{

/// ˂summary˃

/// Контекст БД

/// ˂/summary˃

public class Context : DbContext, IWebContext

{

/// ˂summary˃

/// Строка подключения к БД

/// ˂/summary˃

string \_connectionString;

/// ˂summary˃

/// Конструктор с DI

/// ˂/summary˃

/// ˂param name="configuration"˃Файл конфигураций˂/param˃

public Context(IConfiguration configuration)

{

\_connectionString = configuration["ConnectionStrings:DefaultConnection"];

}

/// ˂summary˃

/// Конструктор с DI

/// ˂/summary˃

/// ˂param name="options"˃Параметры контекста˂/param˃

/// ˂param name="configuration"˃Файл конфигураций˂/param˃

public Context(DbContextOptions options, IConfiguration configuration) : base(options)

{

\_connectionString = configuration["ConnectionStrings:DefaultConnection"];

}

public DbSet˂Discipline˃ Disciplines { get; set; }

public DbSet˂Assistant˃ Assistants { get; set; }

public DbSet˂Group˃ Groups { get; set; }

public DbSet˂LessonVisitor˃ LessonsVisitors { get; set; }

public DbSet˂Lesson˃ Lessons { get; set; }

public DbSet˂LessonType˃ LessonTypes { get; set; }

public DbSet˂Person˃ Persons { get; set; }

public DbSet˂StatusOnLesson˃ StatusOnLessons { get; set; }

public DbSet˂Student˃ Students { get; set; }

public DbSet˂Teacher˃ Teachers { get; set; }

public DbSet˂User˃ Users { get; set; }

public DbSet˂Role˃ Roles { get; set; }

public DbSet˂UserRole˃ UserRoles { get; set; }

/// ˂summary˃

/// Событие "при конфигурации"

/// ˂/summary˃

/// ˂param name="optionsBuilder"˃Параметры конфигурации˂/param˃

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseNpgsql(\_connectionString);

}

/// ˂summary˃

/// Событие "при создании моделей"

/// ˂/summary˃

/// ˂param name="modelBuilder"˃Генератор моделей˂/param˃

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

SeedDB.UpdateTables(modelBuilder);

modelBuilder.Entity˂LessonVisitor˃()

.HasOne˂Lesson˃(v =˃ v.Lesson)

.WithMany(l =˃ l.LessonVisitors)

.HasForeignKey(x =˃ x.LessonId);

modelBuilder.Entity˂Lesson˃()

.HasOne˂Discipline˃(x =˃ x.Discipline)

.WithMany(i =˃ i.Lessons)

.HasForeignKey(x =˃ x.DisciplineId);

modelBuilder.Entity˂Student˃()

.HasOne˂Group˃(s =˃ s.Group)

.WithMany(i =˃ i.Studnets)

.HasForeignKey(x =˃ x.GroupId);

modelBuilder.Entity˂Discipline˃()

.HasOne˂Group˃(d =˃ d.Group)

.WithMany(i =˃ i.Disciplines)

.HasForeignKey(x =˃ x.GroupId);

}

/// ˂summary˃

/// Асинхронное сохранение изменений

/// ˂/summary˃

/// ˂param name="cancellationToken"˃Токен отмены˂/param˃

/// ˂returns˃Результат сохранения˂/returns˃

public async Task˂bool˃ SaveChangesAsync(CancellationToken cancellationToken = default)

{

try

{

await base.SaveChangesAsync(cancellationToken);

}

catch (Exception)

{

//handle with some logger

return false;

}

return true;

}

/// ˂summary˃

/// Сохранить изменения

/// ˂/summary˃

/// ˂returns˃Успешность сохранения˂/returns˃

public bool SaveChanges()

{

try

{

base.SaveChanges();

}

catch (Exception e)

{

//handle with some logger

Console.WriteLine(e);

return false;

}

return true;

}

/// ˂summary˃

/// Обновить модель

/// ˂/summary˃

/// ˂typeparam name="TEntity"˃Тип модели˂/typeparam˃

/// ˂param name="entity"˃Модель для обновления˂/param˃

/// ˂returns˃Обновленная модель˂/returns˃

public EntityEntry? Update˂TEntity˃(TEntity entity)

{

try

{

return base.Update(entity);

}

catch (Exception e)

{

//handle with some logger

Console.WriteLine(e);

return null;

}

}

/// ˂summary˃

/// "Деструктор" класса

/// ˂/summary˃

/// ˂returns˃Успешность уничтожения˂/returns˃

public bool Dispose()

{

try

{

base.Dispose();

}

catch (Exception e)

{

//handle with some logger

Console.WriteLine(e);

return false;

}

return true;

}

}

}

</programlisting>

</section>

</chapter>

</book>