**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**
**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**
**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Образовательная программа бакалавриата «Программная инженерия»

СОГЛАСОВАНО
Научный руководитель,
Преподаватель факультета Компьютерных наук Департамента «Программная инженерия»

_____ Н.К.Чуйкин
«___» _____ 2020 г.

УТВЕРЖДАЮ
Академический руководитель образовательной программы «Программная инженерия» профессор департамента программной инженерии, канд. техн. наук

_____ В.В. Шилов
«___» _____ 2020 г.

| | |
|---|---|
| *Подп. и дата* | |
| *Инв. № дубл.* | |
| *Взам. инв. №* | |
| *Подп. и дата* | |
| *Инв. № подл* | |

**Агрегатор магазинов с элементами социальной сети**
**Сервер**

# Текст программы

# ЛИСТ УТВЕРЖДЕНИЯ

# RU.17701729.04.01-01 12 01-1-ЛУ

Исполнитель
студент группы БПИ 194
_____ / Е.В.Аникеев /
«____»_____ 2020 г.

**Москва 2020**

УТВЕРЖДЕН
RU.17701729.04.01-01 12 01-1-ЛУ

**Агрегатор магазинов с элементами социальной сети**
**Сервер**

# Текст программы

**RU.17701729.04.01-01 12 01-1**

**Листов 52**

| Инв. № подл | Подп. и дата | Взам. инв. № | Инв. № дубл. | Подп. и дата |
|---|---|---|---|---|
| | | | | |

**Москва 2020**

# Содержание

# 1 Текст программы

## 1.1Program.cs

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Hosting;

namespace ShopsAggregatorWebApi
{
    /// <summary>
    /// Класс с Main.
    /// </summary>
    public class Program
    {
        /// <summary>
        /// Main метод.
        /// </summary>
        /// <param name="args"></param>
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        /// <summary>
        /// Создает хост.
        /// </summary>
        /// <param name="args">Аргументы.</param>
        /// <returns>Созданый хост.</returns>
        public static IHostBuilder CreateHostBuilder(string[]
args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder => {
webBuilder.UseStartup<Startup>(); });
    }
}
```

## 1.2Startup.cs

```
using System.Globalization;
using System.Net;
using System.Threading;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
```

```
using Microsoft.AspNetCore.HttpOverrides;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using ShopsAggregatorWebApi.Services;

namespace ShopsAggregatorWebApi
{
    /// <summary>
    /// Класс создания сервиса.
    /// </summary>
    public class Startup
    {
        /// <summary>
        /// Конструктор.
        /// </summary>
        /// <param name="configuration">Конфигурации
сервера.</param>
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        /// <summary>
        /// Конфигурация сервера.
        /// </summary>
        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this
method to add services to the container.
        /// <summary>
        /// Добавляет различные сервисы.
        /// </summary>
        /// <param name="services">Сервис для
добавления.</param>
        public void ConfigureServices(IServiceCollection
services)
        {
```

```
            Thread.CurrentThread.CurrentCulture =
Thread.CurrentThread.CurrentCulture = new CultureInfo("en-
US");

services.AddEntityFrameworkNpgsql().AddDbContext<DatabaseConte
xt>(options =>

options.UseNpgsql(Configuration.GetConnectionString("DatabaseC
ontext")));
            services.AddControllers();

services.Configure<ForwardedHeadersOptions>(options =>
            {

options.KnownProxies.Add(IPAddress.Parse("10.0.0.100"));
            });
        }

        /// <summary>
        /// Создает HTTP клиент.
        /// </summary>
        /// <param name="app">Создатель клиента.</param>
        /// <param name="env">Параметры окружения.</param>
        // 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();
            }

            app.UseForwardedHeaders(new
ForwardedHeadersOptions
            {
                ForwardedHeaders =
ForwardedHeaders.XForwardedFor |
ForwardedHeaders.XForwardedProto
            });
```

```
            app.UseAuthentication();

            app.UseRouting();

            app.UseAuthorization();

            app.UseEndpoints(endpoints => {
endpoints.MapControllers(); });
        }
    }
}
```

1.3Buyer.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;

namespace ShopsAggregatorWebApi.Models
{
    /// <summary>
    /// Тип пользователя-покупателя.
    /// </summary>
    [Table("buyers")]
    public class Buyer : User
    {
        /// <summary>
        /// Id понравившихся записей в базе данных.
        /// </summary>
        [Column("likedposts")]
        public List<Int32> LikedPosts { get; set; } = new
List<Int32>();
        /// <summary>
        /// Id пользователей-продавцов, на которых подписан
пользователь.
        /// </summary>
        [Column("subscribed")]
        public List<Int32> Subscribed { get; set; } = new
List<Int32>();

        /// <summary>
```

```
        /// Id записей, которые добавли пользователи-продавцы,
    на которых подписан данный пользователь.
        /// </summary>
        [Column("newposts")]
        public List<Int32> NewPosts { get; set; } = new
    List<Int32>();

        /// <summary>
        /// Конструктор от формы создания пользователя.
        /// </summary>
        /// <param name="data">Форма регистрации.</param>
        public Buyer(RegistrationForm data) : base(data)
        {

        }

        /// <summary>
        /// Пустой конструктор.
        /// </summary>
        public Buyer()
        {

        }
    }
}
```

1.4BuyerOrderView.cs

```
using System;

namespace ShopsAggregatorWebApi.Models
{
    /// <summary>
    /// Модель данных для заказа пользователя-покупателя.
    /// </summary>
    public class BuyerOrderView
    {
        /// <summary>
        /// Id заказа.
        /// </summary>
        public Int32 OrderId { get; set; }
        /// <summary>
```

```
        /// Экземпляр пользователя-покупателя, который создал
запись.
        /// </summary>
        public Seller OrderSeller { get; set; }
        /// <summary>
        /// Запись, которую опубликовал пользователь-продавец.
        /// </summary>
        public Post SellerPost { get; set; }
        /// <summary>
        /// Одобрил ли пользователь-продавец заказ.
        /// </summary>
        public Boolean IsSucceded { get; set; }
        /// <summary>
        /// Отклонил ли пользователь-продавец заказ.
        /// </summary>
        public Boolean IsCanceled { get; set; }

        /// <summary>
        /// Пустой конструкток.
        /// </summary>
        public BuyerOrderView()
        {

        }

        /// <summary>
        /// Конструктор от записи, заказа и пользователя-
продавца.
        /// </summary>
        /// <param name="order">Экземпляр типа заказа.</param>
        /// <param name="post">Экземпляр типа записи.</param>
        /// <param name="seller">Экземпляр типа пользователя-
продавца.</param>
        public BuyerOrderView(Order order, Post post,  Seller
seller)
        {
            OrderSeller = seller;
            SellerPost = post;
            IsSucceded = order.IsSucceded;
            IsCanceled = order.IsCanceled;
```

```
                OrderId = order.Id;
            }
        }
    }
```

1.5BuyerPostView.cs

```
    using System;
    using System.Collections.Generic;

    namespace ShopsAggregatorWebApi.Models
    {
        /// <summary>
        /// Модель данных записи для пользователя-покупателя.
        /// </summary>
        public class BuyerPostView
        {
            /// <summary>
            /// Имя пользователя-продавца, который опубликовал
    запись.
            /// </summary>
            public String Username { get; set; }
            /// <summary>
            /// Id записи.
            /// </summary>
            public Int32 PostId { get; set; }
            /// <summary>
            /// Информация о записи.
            /// </summary>
            public String Info { get; set; }

            /// <summary>
            /// Пользователи-покупатели, которым понравилась
    запись.
            /// </summary>
            public List<Int32> Likers { get; set; } = new
    List<Int32>();

            /// <summary>
            /// Конструктор от записи и пользователя-продавца.
            /// </summary>
            /// <param name="post">Экземпляр типа записи.</param>
```

```
        /// <param name="creator">Экземпляр типа пользователя-
продавца.</param>
        public BuyerPostView(Post post, Seller creator)
        {
            Username = creator.Username;
            PostId = post.Id;
            Info = post.Info;
            Likers = post.Likers;
        }
    }
}
```

1.6CreatePostForm.cs

```
using System;
using System.Collections.Generic;

namespace ShopsAggregatorWebApi.Models
{
    /// <summary>
    /// Модель для создания записи, которая отправляется на
сервер.
    /// </summary>
    public class CreatePostForm
    {
        /// <summary>
        /// Id пользователя-продавца, который создает запись.
        /// </summary>
        public Int32 CreatorId { get; set; }
        /// <summary>
        /// Информация о записи.
        /// </summary>
        public String Info { get; set; }
        /// <summary>
        /// Массив байтов картинки записи.
        /// </summary>
        public List<Int32> ImageBytes { get; set; } = new
List<Int32>();
    }
}
```

1.7IconPostForm.cs

```
using System;
using System.Collections.Generic;

namespace ShopsAggregatorWebApi.Models
{
    /// <summary>
    /// Экземпляр типа установки иконки пользователю.
    /// </summary>
    public class IconPostForm
    {
        /// <summary>
        /// Байты фотографии, выбранной пользователем.
        /// </summary>
        public List<Int32> IconBytesArr { get; set; }

        /// <summary>
        /// Id пользователя, который изменяет иконку.
        /// </summary>
        public Int32 ToId { get; set; }
    }
}
```

1.8Order.cs

```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace ShopsAggregatorWebApi.Models
{
    /// <summary>
    /// Экземпляр типа заказ.
    /// </summary>
    [Table("orders")]
    public class Order
    {
        /// <summary>
        /// Id заказа.
        /// </summary>

[Key,DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        [Column("id")]
```

```
            public Int32 Id { get; set; }
            /// <summary>
            /// Id записи, на которую создали запись.
            /// </summary>
            [Column("postid")]
            public Int32 PostId { get; set; }
            /// <summary>
            /// Id пользователя-покупателя, который создал заказ.
            /// </summary>
            [Column("customerid")]
            public Int32 BuyerId { get; set; }
            /// <summary>
            /// Подтвердил ли пользователь-продавец заказ.
            /// </summary>
            [Column("issucceded")]
            public Boolean IsSucceded { get; set; } = false;
            /// <summary>
            /// Отклонил ли пользователь-продавец заказ.
            /// </summary>
            [Column("iscanceled")]
            public Boolean IsCanceled { get; set; }= false;
        }
    }
```

1.9Post.cs

```
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;
    using System.ComponentModel.DataAnnotations.Schema;
    using System.IO;

    namespace ShopsAggregatorWebApi.Models
    {
        /// <summary>
        /// Тип записи, которую создает пользователь-продавец
        /// </summary>
        [Table("posts")]
        public class Post
        {
            /// <summary>
            /// Путь для сохранения картинок.
```

```
        /// </summary>
        private const String PostsPhotoDirectory =
"../images";
        /// <summary>
        /// Id записи.
        /// </summary>
        [Key,
DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        [Column("id")]
        public Int32 Id { get; set; }
        /// <summary>
        /// Id создателя записи - пользователя-продавца.
        /// </summary>
        [Column("creatorid")]
        public Int32 CreatorId { get; set; }
        /// <summary>
        /// Информации о записи.
        /// </summary>
        [Column("info")]
        public String Info { get; set; }

        /// <summary>
        /// Id пользователей-покупателей, которым понравилась
эта запись.
        /// </summary>
        [Column("likers")]
        public List<Int32> Likers { get; set; } = new
List<Int32>();

        /// <summary>
        /// Ставит полям значения из формы и вызывает метод
добавления фотографии.
        /// </summary>
        /// <param name="form">Форма создания записи.</param>
        public void CreatePostFromForm(CreatePostForm form)
        {
            this.CreatorId = form.CreatorId;
            this.Info = form.Info;
            AddPostImage(form);
        }
```

```
        /// <summary>
        /// Пустой конструктор.
        /// </summary>
        public Post()
        {

        }

        /// <summary>
        /// Добавляет фотографию для записи из формы.
        /// </summary>
        /// <param name="form">Форма записи.</param>
        private void AddPostImage(CreatePostForm form)
        {
            if (!Directory.Exists(PostsPhotoDirectory))

Directory.CreateDirectory(PostsPhotoDirectory);
            using (FileStream fs = new
FileStream(PostsPhotoDirectory + $"/{this.Id}photo.jpeg",
                FileMode.OpenOrCreate,
                FileAccess.Write))
            {
                foreach (Int32 imageByte in form.ImageBytes)
                {
                    fs.WriteByte((Byte) imageByte);
                }
            }
        }
    }
}
```

1.10 RegistrationForm.cs

```
using System;

namespace ShopsAggregatorWebApi.Models
{
    /// <summary>
    /// Форма для регистрации пользователя в сервисе.
    /// </summary>
    public class RegistrationForm
```

```
        {
            /// <summary>
            /// Имя пользователя.
            /// </summary>
            public String Username { get; set; }
            /// <summary>
            /// Email пользователя.
            /// </summary>
            public String Email { get; set; }
            /// <summary>
            /// Пароль пользователя.
            /// </summary>
            public String Password { get; set; }
            /// <summary>
            /// Является ли пользователь пользователем-продавцом.
            /// </summary>
            public Boolean IsSeller { get; set; }
        }
    }
```

1.11 Seller.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;

namespace ShopsAggregatorWebApi.Models
{
    /// <summary>
    /// Тип пользователя-продавца.
    /// </summary>
    [Table("sellers")]
    public class Seller : User
    {
        /// <summary>
        /// Id записей, созданных пользователем.
        /// </summary>
        [Column("items")]
        public List<Int32> Items { get; set; } = new
List<Int32>();
        /// <summary>
```

```
        /// Id подписчиков, которые подписались на данного
    пользователя.
        /// </summary>
        [Column("subscribers")]
        public List<Int32> Subscribers { get; set; } = new
    List<Int32>();

        /// <summary>
        /// Пустой конструктор.
        /// </summary>
        public Seller() : base()
        {

        }

        /// <summary>
        /// Конструктор от формы создания пользователя.
        /// </summary>
        /// <param name="data">Форма регистрации.</param>
        public Seller(RegistrationForm data) : base(data)
        {

        }
    }
}
```

1.12 SellerOrderView.cs

```
using System;

namespace ShopsAggregatorWebApi.Models
{
    /// <summary>
    /// Модель заказа для пользователя-покупателя.
    /// </summary>
    public class SellerOrderView
    {
        /// <summary>
        /// Id заказа.
        /// </summary>
        public Int32 OrderId { get; set; }
        /// <summary>
```

```
        /// Экземпляр типа пользователя, который сделал заказ.
        /// </summary>
        public Buyer OrderBuyer { get; set; }
        /// <summary>
        /// Экземпляр типа записи, которую заказал
пользователь.
        /// </summary>
        public Post CurrentPost { get; set; }
        /// <summary>
        /// Одобрил ли пользователь-продавец заказ.
        /// </summary>
        public Boolean IsSucceded { get; set; }
        /// <summary>
        /// Отклонил ли пользователь-продавец заказ.
        /// </summary>
        public Boolean IsCanceled { get; set; }

        /// <summary>
        /// Пустой конструктор.
        /// </summary>
        public SellerOrderView()
        {

        }

        /// <summary>
        /// Конструктор от заказа, записи и пользователя-
покупателя.
        /// </summary>
        /// <param name="order">Экземпляр типа заказа.</param>
        /// <param name="post">Экземпляр типа записи.</param>
        /// <param name="buyer">Экземпляр типа пользователя-
покупателя.</param>
        public SellerOrderView(Order order, Post post, Buyer
buyer)
        {
            OrderBuyer = buyer;
            CurrentPost = post;
            IsSucceded = order.IsSucceded;
            IsCanceled = order.IsCanceled;
```

```
                OrderId = order.Id;
            }
        }
    }
```

1.13 User.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace ShopsAggregatorWebApi.Models
{
    /// <summary>
    /// Общий тип пользователей.
    /// </summary>
    public class User
    {
        /// <summary>
        /// Id пользователя.
        /// </summary>

[Key,DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        [Column("id")]
        public Int32 Id { get; set; }
        /// <summary>
        /// Имя пользователя.
        /// </summary>
        [Column("username")]
        public String Username { get; set; }
        /// <summary>
        /// Email пользователя.
        /// </summary>
        [Column("email")]
        public String Email { get; set; }
        /// <summary>
        /// Пароль пользователя.
        /// </summary>
        [Column("password")]
        public String Password { get; set; }
```

```
        /// <summary>
        /// Имя файла с иконкой пользователя.
        /// </summary>
        [Column("iconpath")]
        public String IconPath { get; set; } =
"standarticon.jpeg";
        /// <summary>
        /// Информация о пользователе.
        /// </summary>
        [Column("info")]
        public String Info { get; set; }

        /// <summary>
        /// Id заказов пользователя.
        /// </summary>
        [Column("orders")]
        public List<Int32> Orders { get; set; } = new
List<Int32>();

        /// <summary>
        /// Пустой конструктор.
        /// </summary>
        public User()
        {

        }
        /// <summary>
        /// Конструктор от формы создания пользователя.
        /// </summary>
        /// <param name="data">Форма регистрации.</param>
        public User(RegistrationForm data)
        {
            this.Email = data.Email;
            this.Username = data.Username;
            this.Password = data.Password;
        }
    }
}
```

## 1.14 AuthorizationController.cs

```
using System;
```

```
using System.Collections.Generic;
using Microsoft.AspNetCore.Mvc;
using ShopsAggregatorWebApi.Models;
using ShopsAggregatorWebApi.Services;

namespace ShopsAggregatorWebApi.Controllers
{
    /// <summary>
    /// Контроллер взаимодействия с пользователями.
    /// </summary>
    [Route("api/users")]
    [ApiController]
    public class AuthorizationContoller : ControllerBase
    {
        /// <summary>
        /// Сервис базы данных.
        /// </summary>
        private readonly DatabaseContext _service;
        /// <summary>
        /// Конструктор контроллера.
        /// </summary>
        /// <param name="service">Сервис баз данных.</param>
        public AuthorizationContoller(DatabaseContext service)
        {
            _service = service;
        }

        /// <summary>
        /// Метод получения всех пользователей.
        /// </summary>
        /// <returns>Список всех пользователей.</returns>
        [HttpGet]
        public ActionResult<List<User>> GetAllUsers() =>
_service.GetAllUsers();

        /// <summary>
        /// Регистрирует нового пользователя.
        /// </summary>
        /// <param name="user">Форма регистрации
пользователя.</param>
```

```
        /// <returns>Результат регистрации.</returns>
        [HttpPost("reg")]
        public ActionResult
RegistrateNewUser([FromBody]RegistrationForm user)
        {
            User registeredUser = null;
            if (user.IsSeller)
                registeredUser = _service.CreateUser(new
Seller(user));
            else
               registeredUser = _service.CreateUser(new
Buyer(user));
            if (registeredUser != null)
            {
                return Ok("Аккаунт успешно создан");
            }

            return BadRequest("Такой аккаунт уже
существует!");
        }

        /// <summary>
        /// Аутентификация пользователя-покупателя.
        /// </summary>
        /// <param name="login">Логин для входа.</param>
        /// <param name="password">Пароль для входа.</param>
        /// <returns>Экземпляр пользователя.</returns>
        [HttpGet("authBuyer")]
        public ActionResult<User>TryAuthBuyer(String login,
String password)
        {
            return _service.GetBuyer(login, password);
        }

        /// <summary>
        /// Аутентификация пользователя-продавца.
        /// </summary>
        /// <param name="login">Логин для входа.</param>
        /// <param name="password">Пароль для входа.</param>
        /// <returns>Экземпляр пользователя.</returns>
```

```
        [HttpGet("authSeller")]
        public ActionResult<User> TryAuthSeller(String login,
String password)
        {
            return _service.GetSeller(login, password);
        }

        /// <summary>
        /// Добавление иконки пользователю-продавцу.
        /// </summary>
        /// <param name="icon">Форма добавления
иконки.</param>
        /// <returns>Результат добавления.</returns>
        [HttpPut("addSellerIcon")]
        public IActionResult UpdateSellerIcon(IconPostForm
icon)
        {
            if (_service.AddIconToSeller(icon, icon.ToId))
                return Ok();
            return BadRequest();
        }

        /// <summary>
        /// Добавление иконки пользователю-покупателю.
        /// </summary>
        /// <param name="icon">Форма добавления
иконки.</param>
        /// <returns>Результат добавления.</returns>
        [HttpPut("addBuyerIcon")]
        public IActionResult
UpdateBuyerIcon([FromBody]IconPostForm icon)
        {
            if (_service.AddIconToBuyer(icon, icon.ToId))
                return Ok();
            return BadRequest();
        }

        /// <summary>
        /// Обновляют информацию о пользователе-покупателе.
        /// </summary>
```

```
        /// <param name="buyerId">Id пользователя-
покупателя.</param>
        /// <param name="info">Новая информация о
пользователе.</param>
        /// <returns>Результат обновления.</returns>
        [HttpPut("setBuyerInfo")]
        public IActionResult SetBuyerInfo(Int32 buyerId,
String info)
        {
            _service.SetBuyerInfo(buyerId, info);
            return Ok();
        }

        /// <summary>
        /// Обновляют информацию о пользователе-продавце.
        /// </summary>
        /// <param name="sellerId">Id пользователя-
продавца.</param>
        /// <param name="info">Новая информация о
пользователе.</param>
        /// <returns>Результат обновления.</returns>
        [HttpPut("setSellerInfo")]
        public IActionResult SetSellerInfo(Int32 sellerId,
String info)
        {
            _service.SetSellerInfo(sellerId, info);
            return Ok();
        }
    }
}
```

1.15 OrdersController.cs

```
using System;
using System.Collections.Generic;
using Microsoft.AspNetCore.Mvc;
using ShopsAggregatorWebApi.Models;
using ShopsAggregatorWebApi.Services;

namespace ShopsAggregatorWebApi.Controllers
{
    /// <summary>
```

```
    /// Контроллер взаимодействия с заказами.
    /// </summary>
    [Route("api/orders")]
    public class OrdersController : ControllerBase
    {
        /// <summary>
        /// Сервис базы данных.
        /// </summary>
        private readonly DatabaseContext _service;
        /// <summary>
        /// Конструктор контроллера.
        /// </summary>
        /// <param name="service">Сервис баз данных.</param>
        public OrdersController(DatabaseContext service)
        {
            _service = service;
        }

        /// <summary>
        /// Добавляет новую запись.
        /// </summary>
        /// <param name="buyerId">Id пользователя-
покупателя.</param>
        /// <param name="postId">Id записи.</param>
        /// <returns>Резульат добавления.</returns>
        [HttpPost("addOrder")]
        public IActionResult AddOrder(Int32 buyerId, Int32
postId)
        {
            _service.AddOrder(buyerId, postId);
            return Ok();
        }

        /// <summary>
        /// Получает список заказов пользователя-покупателя.
        /// </summary>
        /// <param name="buyerId">Id пользователя-
покупателя.</param>
        /// <returns>Список заказов.</returns>
        [HttpGet("getBuyerOrders")]
```

```
        public List<BuyerOrderView> GetBuyerOrders(Int32
buyerId)
        {
            return _service.GetBuyerOrders(buyerId);
        }

        /// <summary>
        /// Получает список заказов пользователя-продавца.
        /// </summary>
        /// <param name="sellerId">Id пользователя-
продавца.</param>
        /// <returns>Список заказов.</returns>
        [HttpGet("getSellerOrders")]
        public List<SellerOrderView> GetSellerorders(Int32
sellerId)
        {
            return _service.GetSellerOrders(sellerId);
        }

        /// <summary>
        /// Ставит одобренный статус заказу.
        /// </summary>
        /// <param name="orderId">Id заказа, которому
устанавливается статус.</param>
        /// <returns>Резльтат изменения статуса.</returns>
        [HttpPut("setOrderSuccess")]
        public IActionResult SetOrderSuccess(Int32 orderId)
        {
            _service.SetOrderSuccess(orderId);
            return Ok();
        }

        /// <summary>
        /// Ставит отклоненный статус заказу.
        /// </summary>
        /// <param name="orderId">Id заказа, которому
устанавливается статус.</param>
        /// <returns>Резльтат изменения статуса.</returns>
        [HttpPut("setOrderCanceled")]
        public IActionResult SetOrderCanceled(Int32 orderId)
```

```
        {
            _service.SetOrderCanceled(orderId);
            return Ok();
        }

        /// <summary>
        /// Удаляет заказ.
        /// </summary>
        /// <param name="orderId">Id заказа, который
удаляют.</param>
        /// <returns>Результат удаления.</returns>
        [HttpDelete("deleteOrder")]
        public IActionResult DeleteOrder(Int32 orderId)
        {
            _service.DeleteOrder(orderId);
            return Ok();
        }
    }
}
```

1.16 PostController.cs

```
using System;
using System.Collections.Generic;
using Microsoft.AspNetCore.Mvc;
using ShopsAggregatorWebApi.Models;
using ShopsAggregatorWebApi.Services;

namespace ShopsAggregatorWebApi.Controllers
{
    /// <summary>
    /// Контроллер взаимодействия с записями.
    /// </summary>
    [Route("api/posts")]
    public class PostController  : ControllerBase
    {
        /// <summary>
        /// Сервис базы данных.
        /// </summary>
        private readonly DatabaseContext _service;
        /// <summary>
        /// Конструктор контроллера.
```

```
        /// </summary>
        /// <param name="service">Сервис баз данных.</param>
        public PostController(DatabaseContext service)
        {
            _service = service;
        }

        /// <summary>
        /// Создает новую запись.
        /// </summary>
        /// <param name="postForm">Форма для создания
записи.</param>
        /// <returns>Результат создания записи.</returns>
        [HttpPost("create")]
        public IActionResult
CreatePost([FromBody]CreatePostForm postForm)
        {
            if (_service.CreatePost(postForm))
                return Ok();
            return BadRequest();
        }

        /// <summary>
        /// Получает запись по id.
        /// </summary>
        /// <param name="id">Id записи.</param>
        /// <returns>полученная запись.</returns>
        [HttpGet("getPost")]
        public Post GetPostById(Int32 id)
        {
            return _service.GetPostById(id);
        }

        /// <summary>
        /// Получает записи, опубликованные пользователем-
продавцом.
        /// </summary>
        /// <param name="sellerName">Имя пользователя-
продавца.</param>
```

```
        /// <returns>Список записей пользователя-
продавца.</returns>
        [HttpGet("getSellerPosts")]
        public List<Post> GetPostsByCreatorId(String
sellerName)
        {
            return _service.GetPostsByCreatorName(sellerName);
        }

        /// <summary>
        /// Получает новые записи пользователя-покупателя.
        /// </summary>
        /// <param name="buyerId">Id пользователя-
покупателя.</param>
        /// <returns>Список новых записей пользователя-
покупателя.</returns>
        [HttpGet("getBuyerNewPost")]
        public List<BuyerPostView> GetBuyerNewPosts(Int32
buyerId)
        {
            return _service.GetBuyerNewPosts(buyerId);
        }

        /// <summary>
        /// Добавляет лайк пользователя-покупателя к записи.
        /// </summary>
        /// <param name="likerId">Id пользователя-
покупателя.</param>
        /// <param name="postId">Id записи.</param>
        /// <returns>Результат добавления лайка.</returns>
        [HttpPut("addLike")]
        public IActionResult AddLikeToPost(Int32 likerId,
Int32 postId)
        {
            _service.AddLikeToPost(likerId, postId);
            return Ok();
        }

        /// <summary>
        /// Удаляет лайк пользователя-покупателя у записи.
```

```
        /// </summary>
        /// <param name="likerId">Id пользователя-
покупателя.</param>
        /// <param name="postId">Id записи.</param>
        /// <returns>Результат удаления лайка.</returns>
        [HttpPut("removeLike")]
        public IActionResult RemoveLikeFromPost(Int32 likerId,
Int32 postId)
        {
            _service.RemoveLikeFromPost(likerId, postId);
            return Ok();
        }

        /// <summary>
        /// Получает записи, которые понравились пользователю-
покупателю.
        /// </summary>
        /// <param name="buyerId">Id пользователя-
покупателя.</param>
        /// <returns>Список понравившихся записей.</returns>
        [HttpGet("getBuyerLikedPosts")]
        public List<BuyerPostView> GetBuyerLikedPosts(Int32
buyerId)
        {
            return _service.GetBuyerLikedPosts(buyerId);
        }
    }
}
```

1.17 SearchController.cs

```
using System;
using System.Collections.Generic;
using Microsoft.AspNetCore.Mvc;
using ShopsAggregatorWebApi.Models;
using ShopsAggregatorWebApi.Services;

namespace ShopsAggregatorWebApi.Controllers
{
    /// <summary>
    /// Контроллер взаимодействия с записями.
    /// </summary>
```

```
    [Route("api/search")]
    public class SearchController : Controller
    {
        /// <summary>
        /// Сервис базы данных.
        /// </summary>
        private readonly DatabaseContext _service;
        /// <summary>
        /// Конструктор контроллера.
        /// </summary>
        /// <param name="service">Сервис баз данных.</param>
        public SearchController(DatabaseContext service)
        {
            _service = service;
        }

        /// <summary>
        /// Получает пользователей-продавцов, которые содержат
введенную строку в имени.
        /// </summary>
        /// <param name="q">Строка поиска.</param>
        /// <returns>Список полученных пользователе-
продавцов.</returns>
        [HttpGet]
        public ActionResult<List<String>> SearchSellers(String
q)
        {
            var users = _service.SearchSellers(q);
            return users;
        }

        /// <summary>
        /// Получает пользователя-продавца по имени.
        /// </summary>
        /// <param name="sellerName">Имя пользователя-
продавца.</param>
        /// <returns>Полученный пользователь-
продавец.</returns>
        [HttpGet("getSeller")]
        public Seller GetSellerByName(String sellerName)
```

```
        {
            return _service.GetSeller(sellerName);
        }
    }
}
```

1.18 SubscribeController.cs

```
using System;
using Microsoft.AspNetCore.Mvc;
using ShopsAggregatorWebApi.Services;

namespace ShopsAggregatorWebApi.Controllers
{
    /// <summary>
    /// Контроллер взаимодействия с подписками.
    /// </summary>
    [Route("api/sub")]
    public class SubscribeController : Controller
    {
        /// <summary>
        /// Сервис базы данных.
        /// </summary>
        private readonly DatabaseContext _service;
        /// <summary>
        /// Конструктор контроллера.
        /// </summary>
        /// <param name="service">Сервис баз данных.</param>
        public SubscribeController(DatabaseContext service)
        {
            _service = service;
        }

        /// <summary>
        /// Добавляет подписчика пользователя-покупателя к
пользователю-продавцу.
        /// </summary>
        /// <param name="buyerId">Id пользователя-
покупателя.</param>
        /// <param name="sellerName">Id пользователя-
продавца.</param>
        /// <returns>Результат подписки.</returns>
```

```
        [HttpPut("addSub")]
        public IActionResult AddSubscriberToSeller(Int32
buyerId, String sellerName)
        {
            if (_service.AddSubscriber(buyerId, sellerName))
                return Ok();
            return BadRequest();
        }

        /// <summary>
        /// Удаляет из подписчиков пользователя-покупателя у
пользователю-продавцу.
        /// </summary>
        /// <param name="buyerId">Id пользователя-
покупателя.</param>
        /// <param name="sellerName">Id пользователя-
продавца.</param>
        /// <returns>Результат отписки.</returns>
        [HttpPut("rmSub")]
        public IActionResult RemoveSellerSubscriber(Int32
buyerId, String sellerName)
        {
            if (_service.RemoveSubscriber(buyerId,
sellerName))
                return Ok();
            return BadRequest();
        }

    }
}
```

1.19 DatabaseContext.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using Microsoft.EntityFrameworkCore;
using ShopsAggregatorWebApi.Models;

namespace ShopsAggregatorWebApi.Services
{
```

```
        /// <summary>
        /// Сервис работы с базой данных.
        /// </summary>
        public class DatabaseContext : DbContext
        {
            /// <summary>
            /// Путь для сохранения картинок.
            /// </summary>
            private const String UsersIconsDirPath = "../images";
            /// <summary>
            /// База данных пользователей-продавцов.
            /// </summary>
            private DbSet<Seller> Sellers { get; set; }
            /// <summary>
            /// База данных пользователкей-покупателей.
            /// </summary>
            private DbSet<Buyer> Buyers { get; set; }

            /// <summary>
            /// База данных записей пользователей-продавцов.
            /// </summary>
            private DbSet<Post> Posts { get; set; }
            /// <summary>
            /// База данных заказов пользователей-покупателей.
            /// </summary>
            private DbSet<Order> Orders { get; set; }

            /// <summary>
            /// Конструктор.
            /// </summary>
            /// <param name="options">Опции для сервиса.</param>
            public
    DatabaseContext(DbContextOptions<DatabaseContext> options) :
    base(options)
            {

            }

            /// <summary>
```

```
        /// Получает всех пользователей, зарегестрированных в
системе.
        /// </summary>
        /// <returns>Список пользователей</returns>
        public List<User> GetAllUsers()
        {
            List<User> users = Sellers.ToList().ConvertAll(x
=> (User) x);
            users.AddRange(Buyers.ToList().ConvertAll(x =>
(User)x));
            return users;
        }

        /// <summary>
        /// Создает нового пользователя-покупателя.
        /// </summary>
        /// <param name="user">Экземпляр пользователя-
покупателя.</param>
        /// <returns>Созданный пользователь.</returns>
        public User CreateUser(Buyer user)
        {
            if (IsSameSellerCreated(user) ||
IsSameBuyerCreated(user))
                return null;
            Buyers.Add(user);
            SaveChangesAsync();
            return user;
        }

        /// <summary>
        /// Создает нового пользователя-продавца.
        /// </summary>
        /// <param name="user">Экземпляр пользователя-
продавца.</param>
        /// <returns>Созданный пользователь.</returns>
        public User CreateUser(Seller user)
        {
            if (IsSameSellerCreated(user) ||
IsSameBuyerCreated(user))
                return null;
```

```
            Sellers.Add(user);
            SaveChangesAsync();
            return user;
        }

        /// <summary>
        /// Проверяет создан ли пользователь-продавец с таким
же email или username.
        /// </summary>
        /// <param name="user">Экземпляр проверяемого
пользователя-продавца.</param>
        /// <returns>Результат проверки.</returns>
        private Boolean IsSameSellerCreated(User user)
        {
            return (from seller in Sellers.ToList() where
seller.Username == user.Username
                                                          ||
seller.Email == user.Email select seller)
                .FirstOrDefault() != null;
        }
        /// <summary>
        /// Проверяет создан ли пользователь-покупатель с
таким же email или username.
        /// </summary>
        /// <param name="user">Экземпляр проверяемого
пользователя-покупатель.</param>
        /// <returns>Результат проверки.</returns>
        private Boolean IsSameBuyerCreated(User user)
        {
            return (from buyer in Buyers.ToList() where
buyer.Username == user.Username
                                                    ||
buyer.Email == user.Email select buyer)
                .FirstOrDefault() != null;
        }

        /// <summary>
        /// Получает пользователя по логину и паролю.
        /// </summary>
        /// <param name="login">Логин.</param>
```

```
        /// <param name="password">Пароль.</param>
        /// <returns>Полученный пользователь.</returns>
        public Buyer GetBuyer(String login, String password)
        {
            return (from buyer in Buyers.ToList()
                where (buyer.Username == login ||
buyer.Username == login)
                      && buyer.Password == password
                select buyer).FirstOrDefault();
        }

        /// <summary>
        /// Получает пользователя-покупателя по его id.
        /// </summary>
        /// <param name="id">Id пользователя.</param>
        /// <returns>Экземпляр типа пользователя-
покупателя.</returns>
        private Buyer GetBuyer(Int32 id)
        {
            return (from buyer in Buyers.ToList() where
buyer.Id == id select buyer).FirstOrDefault();
        }

        /// <summary>
        /// Получает пользователя по логину и паролю.
        /// </summary>
        /// <param name="login">Логин.</param>
        /// <param name="password">Пароль.</param>
        /// <returns>Полученный пользователь.</returns>
        public Seller GetSeller(string login, string password)
        {
            return (from seller in Sellers.ToList()
                where (seller.Username == login ||
seller.Username == login)
                      && seller.Password == password
                select seller).FirstOrDefault();
        }

        /// <summary>
        /// Получает пользователя-продавца по его id.
```

```
        /// </summary>
        /// <param name="id">Id пользователя.</param>
        /// <returns>Экземпляр типа пользователя-
продавца.</returns>
        private Seller GetSeller(Int32 id)
        {
            return (from seller in Sellers.ToList() where
seller.Id == id select seller).FirstOrDefault();
        }

        /// <summary>
        /// Получает пользователя по имени.
        /// </summary>
        /// <param name="sellerName">Имя пользователя.</param>
        /// <returns>Полученный пользователь.</returns>
        public Seller GetSeller(String sellerName)
        {
            Seller currentSeller = (from seller in
Sellers.ToList() where seller.Username == sellerName select
seller)
                .FirstOrDefault();
            if (currentSeller == null)
                return null;
            currentSeller.Id = 0;
            currentSeller.Password = "";
            return currentSeller;
        }

        /// <summary>
        /// Добавляет иконку пользователю-продавцу.
        /// </summary>
        /// <param name="icon">Форма с информацией об
изображении.</param>
        /// <param name="sellerId">Id пользователя-
продавца.</param>
        /// <returns>Результат добавления.</returns>
        public Boolean AddIconToSeller(IconPostForm icon,
Int32 sellerId)
        {
            Seller toSeller =
```

```
                (from seller in Sellers.ToList() where
seller.Id == sellerId select seller).FirstOrDefault();
            if (toSeller == null)
                return false;
            try
            {
                AddIconToUser(icon, toSeller);
            }
            catch (Exception)
            {
                return false;
            }
            toSeller.IconPath =
$"{toSeller.Username}icon.jpeg";
            SaveChangesAsync();
            return true;
        }

        /// <summary>
        /// Добавляет иконку пользователю-покупателю.
        /// </summary>
        /// <param name="icon">Форма с информацией об
изображении.</param>
        /// <param name="buyerId">Id пользователя-
покупателя.</param>
        /// <returns>Результат добавления.</returns>
        public Boolean AddIconToBuyer(IconPostForm icon, Int32
buyerId)
        {
            Buyer toBuyer = (from buyer in Buyers.ToList()
where buyer.Id == buyerId select buyer).FirstOrDefault();
            if (toBuyer == null)
                return false;
            try
            {
                AddIconToUser(icon, toBuyer);
            }
            catch (Exception)
            {
                return false;
```

```
            }

            toBuyer.IconPath = $"{toBuyer.Username}icon.jpeg";
            SaveChangesAsync();
            return true;
        }

        /// <summary>
        /// Добавляет иконку пользователю.
        /// </summary>
        /// <param name="icon">Форма с информацией об
изображении.</param>
        /// <param name="toUser">Экземпляр
пользователя.</param>
        /// <returns>Результат добавления.</returns>
        private void AddIconToUser(IconPostForm icon, User
toUser)
        {
            if (!Directory.Exists(UsersIconsDirPath))
                Directory.CreateDirectory(UsersIconsDirPath);
            using (FileStream fs = new
FileStream(UsersIconsDirPath + $"/{toUser.Username}icon.jpeg",
                FileMode.OpenOrCreate,
                FileAccess.Write))
            {
                foreach (Int32 iconByte in icon.IconBytesArr)
                {
                    fs.WriteByte((Byte) iconByte);
                }
            }
        }

        /// <summary>
        /// Ищет пользователя-покупателя, который содержит в
username поисковую строку.
        /// </summary>
        /// <param name="s">Поисковая строка.</param>
        /// <returns>Результат поиска.</returns>
        public List<String> SearchSellers(String s)
        {
```

```
            return (from seller in Sellers.ToList() where
seller.Username.Contains(s) select seller.Username).ToList();
        }

        /// <summary>
        /// Добваляет подписчика.
        /// </summary>
        /// <param name="buyerId">Id пользователя-
покупателя.</param>
        /// <param name="sellerName">Имя пользователя-
продавца.</param>
        /// <returns>Результат добавления.</returns>
        public Boolean AddSubscriber(Int32 buyerId, String
sellerName)
        {
            Seller currentSeller = (from seller in
Sellers.ToList() where seller.Username == sellerName select
seller)
                .FirstOrDefault();
            if (currentSeller == null)
                return false;
            return AddSubscriber(buyerId, currentSeller.Id);
        }

        /// <summary>
        /// Добваляет подписчика.
        /// </summary>
        /// <param name="buyerId">Id пользователя-
покупателя.</param>
        /// <param name="sellerId">Id пользователя-
продавца.</param>
        /// <returns>Результат добавления.</returns>
        private Boolean AddSubscriber(Int32 buyerId, Int32
sellerId)
        {
            Buyer buyer = GetBuyer(buyerId);
            Seller seller = GetSeller(sellerId);
            if(buyer == null || seller == null)
                return false;
            if(buyer.Subscribed == null)
```

```
                buyer.Subscribed = new List<Int32>();
            if(seller.Subscribers == null)
                seller.Subscribers = new List<Int32>();
            buyer.Subscribed.Add(sellerId);
            seller.Subscribers.Add(buyerId);
            SaveChangesAsync();
            return true;
        }

        /// <summary>
        /// Удаляет подписчика.
        /// </summary>
        /// <param name="buyerId">Id пользователя-
покупателя.</param>
        /// <param name="sellerName">Имя пользователя-
продавца.</param>
        /// <returns>Результат добавления.</returns>
        public Boolean RemoveSubscriber(Int32 buyerId, String
sellerName)
        {
            Seller currentSeller = (from seller in
Sellers.ToList() where seller.Username == sellerName select
seller)
                .FirstOrDefault();
            if (currentSeller == null)
                return false;
            return RemoveSubscriber(buyerId,
currentSeller.Id);
        }

        /// <summary>
        /// Удаляет подписчика.
        /// </summary>
        /// <param name="buyerId">Id пользователя-
покупателя.</param>
        /// <param name="sellerId">Id пользователя-
продавца.</param>
        /// <returns>Результат добавления.</returns>
        private Boolean RemoveSubscriber(Int32 buyerId, Int32
sellerId)
```

```
        {
            Buyer buyer = GetBuyer(buyerId);
            Seller seller = GetSeller(sellerId);
            if(buyer == null || seller == null)
                return false;
            if (buyer.Subscribed == null || seller.Subscribers
== null)
                return false;
            buyer.Subscribed.Remove(sellerId);
            seller.Subscribers.Remove(buyerId);
            SaveChangesAsync();
            return true;
        }

        /// <summary>
        /// Создает новую запись и добавляет ее в базу данных.
        /// </summary>
        /// <param name="postForm">Форма новой записи.</param>
        /// <returns>Результат создания записи</returns>
        public Boolean CreatePost(CreatePostForm postForm)
        {
            if (postForm.ImageBytes == null ||
postForm.ImageBytes.Count == 0 || postForm.CreatorId < 0)
                return false;
            Post newPost = new Post();
            Posts.Add(newPost);
            SaveChanges();
            try
            {
                newPost.CreatePostFromForm(postForm);
            }
            catch (Exception)
            {
                Posts.Remove(newPost);
                return false;
            }

            Seller creator = GetSeller(postForm.CreatorId);
            if (creator == null)
                return false;
```

```
            if(creator.Items == null)
                creator.Items = new List<Int32>();
            creator.Items.Insert(0,newPost.Id);
            AddPostForSubscribers(creator, newPost.Id);
            SaveChangesAsync();
            return true;
        }

        /// <summary>
        /// Добавляет новую запись всем подписчикам
пользователя-продавца.
        /// </summary>
        /// <param name="creator">Пользователь-
создатель.</param>
        /// <param name="postId">Id записи.</param>
        private void AddPostForSubscribers(Seller creator,
Int32 postId)
        {
            if(creator.Subscribers == null)
                creator.Subscribers = new List<Int32>();
            foreach (Int32 buyerId in creator.Subscribers)
            {
                Buyer subscriber = (from buyer in
Buyers.ToList() where buyer.Id == buyerId select
buyer).FirstOrDefault();
                if(subscriber == null)
                    continue;
                if(subscriber.NewPosts == null)
                    subscriber.NewPosts = new List<Int32>();
                subscriber.NewPosts.Insert(0,postId);
            }
        }

        /// <summary>
        /// Получает экземпляр записи по id.
        /// </summary>
        /// <param name="id">Id записи.</param>
        /// <returns>Экземпляр типа записи.</returns>
        public Post GetPostById(Int32 id)
        {
```

```
            return (from post in Posts.ToList() where post.Id
        == id select post).FirstOrDefault();
        }

        /// <summary>
        /// Получаем записи пользователя-продавца по имени.
        /// </summary>
        /// <param name="sellerName">Имя пользователя-
        продавца</param>
        /// <returns>Список записей.</returns>
        public List<Post> GetPostsByCreatorName(String
        sellerName)
        {
            Int32 creatorId = (from seller in Sellers.ToList()
        where seller.Username == sellerName select seller.Id)
                .FirstOrDefault();
            return (from post in Posts.ToList() where
        post.CreatorId == creatorId select post).ToList();
        }

        /// <summary>
        /// Получает новые записи пользователя-покупателя по
        его id.
        /// </summary>
        /// <param name="buyerId">Id пользователя-
        покупателя.</param>
        /// <returns>Список новых записей.</returns>
        public List<BuyerPostView> GetBuyerNewPosts(Int32
        buyerId)
        {
            Buyer buyer = GetBuyer(buyerId);
            if(buyer.NewPosts == null)
                buyer.NewPosts = new List<Int32>();
            List<BuyerPostView> newPosts = new
        List<BuyerPostView>();
            foreach (Int32 postId in buyer.NewPosts)
            {
                Post post = GetPostById(postId);
                Seller seller = GetSeller(post.CreatorId);
```

```
                newPosts.Insert(0, new BuyerPostView(post,
seller));
            }
            return newPosts;
        }

        /// <summary>
        /// Добавляет лайк пользователя-покупателя к записи.
        /// </summary>
        /// <param name="likerId">Id пользователя-
покупателя.</param>
        /// <param name="postId">Id записи.</param>
        public void AddLikeToPost(Int32 likerId, Int32 postId)
        {
            Post post = GetPostById(postId);
            Buyer liker = GetBuyer(likerId);
            if (liker == null)
                return;
            if(post.Likers == null)
                post.Likers = new List<Int32>();
            if(liker.LikedPosts == null)
                liker.LikedPosts = new List<Int32>();
            if(!liker.LikedPosts.Contains(postId))
                liker.LikedPosts.Add(postId);
            if(!post.Likers.Contains(likerId))
                post.Likers.Add(likerId);
            SaveChangesAsync();
        }

        /// <summary>
        /// Удаляет лайк пользователя-покупателя к записи.
        /// </summary>
        /// <param name="likerId">Id пользователя-
покупателя.</param>
        /// <param name="postId">Id записи.</param>
        public void RemoveLikeFromPost(Int32 likerId, Int32
postId)
        {
            Post post = GetPostById(postId);
            Buyer liker = GetBuyer(likerId);
```

```
            if (liker == null)
                return;
            if(post.Likers == null)
                post.Likers = new List<Int32>();
            if(liker.LikedPosts == null)
                liker.LikedPosts = new List<Int32>();
            liker.LikedPosts.Remove(postId);
            post.Likers.Remove(likerId);
            SaveChangesAsync();
        }

        /// <summary>
        /// Получает записи, понравившиеся пользователю
записи.
        /// </summary>
        /// <param name="buyerId">Id пользователя.</param>
        /// <returns>Список записей.</returns>
        public List<BuyerPostView> GetBuyerLikedPosts(Int32
buyerId)
        {
            Buyer buyer = GetBuyer(buyerId);
            if (buyer == null)
                return null;
            if(buyer.LikedPosts == null)
                buyer.LikedPosts = new List<Int32>();
            List<BuyerPostView> posts = new
List<BuyerPostView>();
            foreach (Int32 postId in buyer.LikedPosts)
            {
                Post post = GetPostById(postId);
                Seller seller = GetSeller(post.CreatorId);
                posts.Add(new BuyerPostView(post, seller));
            }

            return posts;
        }

        /// <summary>
        /// Создает новый заказ и добавляет его в базу данных.
        /// </summary>
```

```
        /// <param name="buyerId">Id пользователя-покупателч,
который создает заказ.</param>
        /// <param name="postId">Id записи, по которой создают
заказ.</param>
        public void AddOrder(Int32 buyerId, Int32 postId)
        {
            Buyer buyer = GetBuyer(buyerId);
            Post post = GetPostById(postId);
            if (buyer == null || post == null)
                return;
            Seller seller = GetSeller(post.CreatorId);
            if (seller == null)
                return;
            Order newOrder = new Order
            {
                PostId = post.Id,
                BuyerId = buyer.Id
            };
            Orders.Add(newOrder);
            SaveChanges();
            if(buyer.Orders == null)
                buyer.Orders = new List<Int32>();
            buyer.Orders.Add(newOrder.Id);
            if(seller.Orders == null)
                seller.Orders = new List<Int32>();
            seller.Orders.Add(newOrder.Id);
            SaveChangesAsync();
        }

        /// <summary>
        /// Получает заказы пользователя-покупателя.
        /// </summary>
        /// <param name="buyerId">Id пользователя-
покупателя.</param>
        /// <returns>Список заказов.</returns>
        public List<BuyerOrderView> GetBuyerOrders(Int32
buyerId)
        {
```

| | | | | |
|---|---|---|---|---|
| Изм. | Лист | № док. | Подп. | Дата |
| RU.17701729.04.01-01 ТЗ 01-1 | | | | |
| Инв. № подл. | Подп. и дата | Взам инв. № | Инв. № дубл. | Подп. и дата |

```
            List<Order> orders = (from order in
Orders.ToList() where order.BuyerId == buyerId select
order).ToList();
            List<BuyerOrderView> views = new
List<BuyerOrderView>();
            foreach (Order order in orders)
            {
                Post post = GetPostById(order.PostId);
                Seller seller = GetSeller(post.CreatorId);
                views.Add(new BuyerOrderView(order, post,
seller));
            }

            return views;
        }

        /// <summary>
        /// Получает заказы пользователя-продавца.
        /// </summary>
        /// <param name="sellerId">Id пользователя-
продавца.</param>
        /// <returns>Список заказов.</returns>
        public List<SellerOrderView> GetSellerOrders(Int32
sellerId)
        {
            Seller seller = GetSeller(sellerId);
            List<Order> orders = (from order in
Orders.ToList() where seller.Orders.Contains(order.Id) select
order)
                .ToList();
            List<SellerOrderView> views = new
List<SellerOrderView>();
            foreach (Order order in orders)
            {
                Buyer buyer = GetBuyer(order.BuyerId);
                Post post = GetPostById(order.PostId);
                views.Add(new SellerOrderView(order, post,
buyer));
            }
```

```
            return views;
        }

        /// <summary>
        /// Получает заказ по Id.
        /// </summary>
        /// <param name="orderId">Id заказа.</param>
        /// <returns>Экземпляр типа заказа.</returns>
        private Order GetOrder(Int32 orderId)
        {
            return (from order in Orders.ToList() where
order.Id == orderId select order).FirstOrDefault();
        }

        /// <summary>
        /// Устанавливает одобренный статус заказа.
        /// </summary>
        /// <param name="orderId">Id заказа.</param>
        public void SetOrderSuccess(Int32 orderId)
        {
            Order current = GetOrder(orderId);
            if (current == null)
                return;
            current.IsSucceded = true;
            current.IsCanceled = false;
            SaveChangesAsync();
        }

        /// <summary>
        /// Устанавливает отклоненный статус заказа.
        /// </summary>
        /// <param name="orderId">Id заказа.</param>
        public void SetOrderCanceled(Int32 orderId)
        {
            Order current = GetOrder(orderId);
            if (current == null)
                return;
            current.IsSucceded = false;
            current.IsCanceled = true;
            SaveChangesAsync();
```

```
        }

        /// <summary>
        /// Удаляет заказ.
        /// </summary>
        /// <param name="orderId">Id заказа.</param>
        public void DeleteOrder(Int32 orderId)
        {
            Order current = GetOrder(orderId);
            if (current == null)
                return;
            Orders.Remove(current);
            SaveChangesAsync();
        }

        /// <summary>
        /// Изменяет информацию о пользователе-покупателе.
        /// </summary>
        /// <param name="buyerId">Id пользователя-
покупателя.</param>
        /// <param name="info">Новая информация.</param>
        public void SetBuyerInfo(Int32 buyerId, String info)
        {
            Buyer buyer = GetBuyer(buyerId);
            if (buyer == null)
                return;
            buyer.Info = info;
            SaveChangesAsync();
        }

        /// <summary>
        /// Изменяет информацию о пользователе-продавце.
        /// </summary>
        /// <param name="sellerId">Id пользователя-
покупателя.</param>
        /// <param name="info">Новая информация.</param>
        public void SetSellerInfo(Int32 sellerId, String info)
        {
            Seller seller = GetSeller(sellerId);
            if (seller == null)
```

```
                return;
            seller.Info = info;
            SaveChangesAsync();
        }
    }
}
```

# 2 Список литературы

2.1 ГОСТ 19.401-78. ЕСПД. Текст программы. Требования к содержанию и оформлению. – М.: ИПК Издательство стандартов, 2001.

2.2 Документация по C# [Электронный ресурс]\\ - URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/

# 3 Лист изменений

| Изм. | Номера листов (страниц) | | | | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
|---|---|---|---|---|---|---|---|---|
| | Измененных | Замененных | новых | Аннулированных | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |

| | | | | |
|---|---|---|---|---|
| Изм. | Лист | № док. | Подп. | Дата |
| RU.17701729.04.01-01 ТЗ 01-1 | | | | |
| Инв. № подл. | Подп. и дата | Взам инв. № | Инв. № дубл. | Подп. и дата |