**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 |  | |  | |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **Агрегатор магазинов с элементами социальной сети**  **Сервер**  **Текст программы**  **RU.17701729.04.01-01 12 01-1**  **Листов 48** | | | | |
|  | |  | | |
|  | | |
|  | | | | |
|  | | | |  |

**Москва 2020**

Содержание

[1 Текст программы 4](#_Toc40204194)

[1.1 Program.cs 4](#_Toc40204195)

[1.2 Startup.cs 4](#_Toc40204196)

[1.3 Buyer.cs 6](#_Toc40204197)

[1.4 BuyerOrderView.cs 8](#_Toc40204198)

[1.5 BuyerPostView.cs 9](#_Toc40204199)

[1.6 CreatePostForm.cs 10](#_Toc40204200)

[1.7 IconPostForm.cs 11](#_Toc40204201)

[1.8 Order.cs 11](#_Toc40204202)

[1.9 Post.cs 12](#_Toc40204203)

[1.10 RegistrationForm.cs 14](#_Toc40204204)

[1.11 Seller.cs 15](#_Toc40204205)

[1.12 SellerOrderView.cs 16](#_Toc40204206)

[1.13 User.cs 17](#_Toc40204207)

[1.14 AuthorizationController.cs 19](#_Toc40204208)

[1.15 OrdersController.cs 22](#_Toc40204209)

[1.16 PostController.cs 25](#_Toc40204210)

[1.17 SearchController.cs 27](#_Toc40204211)

[1.18 SubscribeController.cs 29](#_Toc40204212)

[1.19 DatabaseContext.cs 30](#_Toc40204213)

[2 Список литературы 47](#_Toc40204214)

[3 Лист изменений 48](#_Toc40204215)

1. Текст программы
   1. Program.cs
      1. using Microsoft.AspNetCore.Hosting;
      2. using Microsoft.Extensions.Hosting;
      3. namespace ShopsAggregatorWebApi
      4. {
      5. /// <summary>
      6. /// Класс с Main.
      7. /// </summary>
      8. public class Program
      9. {
      10. /// <summary>
      11. /// Main метод.
      12. /// </summary>
      13. /// <param name="args"></param>
      14. public static void Main(string[] args)
      15. {
      16. CreateHostBuilder(args).Build().Run();
      17. }
      18. /// <summary>
      19. /// Создает хост.
      20. /// </summary>
      21. /// <param name="args">Аргументы.</param>
      22. /// <returns>Созданый хост.</returns>
      23. public static IHostBuilder CreateHostBuilder(string[] args) =>
      24. Host.CreateDefaultBuilder(args)
      25. .ConfigureWebHostDefaults(webBuilder => { webBuilder.UseStartup<Startup>(); });
      26. }
      27. }
   2. Startup.cs
      1. using System.Globalization;
      2. using System.Net;
      3. using System.Threading;
      4. using Microsoft.AspNetCore.Builder;
      5. using Microsoft.AspNetCore.Hosting;
      6. using Microsoft.AspNetCore.HttpOverrides;
      7. using Microsoft.EntityFrameworkCore;
      8. using Microsoft.Extensions.Configuration;
      9. using Microsoft.Extensions.DependencyInjection;
      10. using Microsoft.Extensions.Hosting;
      11. using ShopsAggregatorWebApi.Services;
      12. namespace ShopsAggregatorWebApi
      13. {
      14. /// <summary>
      15. /// Класс создания сервиса.
      16. /// </summary>
      17. public class Startup
      18. {
      19. /// <summary>
      20. /// Конструктор.
      21. /// </summary>
      22. /// <param name="configuration">Конфигурации сервера.</param>
      23. public Startup(IConfiguration configuration)
      24. {
      25. Configuration = configuration;
      26. }
      27. /// <summary>
      28. /// Конфигурация сервера.
      29. /// </summary>
      30. public IConfiguration Configuration { get; }
      31. // This method gets called by the runtime. Use this method to add services to the container.
      32. /// <summary>
      33. /// Добавляет различные сервисы.
      34. /// </summary>
      35. /// <param name="services">Сервис для добавления.</param>
      36. public void ConfigureServices(IServiceCollection services)
      37. {
      38. Thread.CurrentThread.CurrentCulture = Thread.CurrentThread.CurrentCulture = new CultureInfo("en-US");
      39. services.AddEntityFrameworkNpgsql().AddDbContext<DatabaseContext>(options =>
      40. options.UseNpgsql(Configuration.GetConnectionString("DatabaseContext")));
      41. services.AddControllers();
      42. services.Configure<ForwardedHeadersOptions>(options =>
      43. {
      44. options.KnownProxies.Add(IPAddress.Parse("10.0.0.100"));
      45. });
      46. }
      47. /// <summary>
      48. /// Создает HTTP клиент.
      49. /// </summary>
      50. /// <param name="app">Создатель клиента.</param>
      51. /// <param name="env">Параметры окружения.</param>
      52. // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
      53. public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
      54. {
      55. if (env.IsDevelopment())
      56. {
      57. app.UseDeveloperExceptionPage();
      58. }
      59. app.UseForwardedHeaders(new ForwardedHeadersOptions
      60. {
      61. ForwardedHeaders = ForwardedHeaders.XForwardedFor | ForwardedHeaders.XForwardedProto
      62. });
      63. app.UseAuthentication();
      64. app.UseRouting();
      65. app.UseAuthorization();
      66. app.UseEndpoints(endpoints => { endpoints.MapControllers(); });
      67. }
      68. }
      69. }
   3. Buyer.cs
      1. using System;
      2. using System.Collections.Generic;
      3. using System.ComponentModel.DataAnnotations.Schema;
      4. namespace ShopsAggregatorWebApi.Models
      5. {
      6. /// <summary>
      7. /// Тип пользователя-покупателя.
      8. /// </summary>
      9. [Table("buyers")]
      10. public class Buyer : User
      11. {
      12. /// <summary>
      13. /// Id понравившихся записей в базе данных.
      14. /// </summary>
      15. [Column("likedposts")]
      16. public List<Int32> LikedPosts { get; set; } = new List<Int32>();
      17. /// <summary>
      18. /// Id пользователей-продавцов, на которых подписан пользователь.
      19. /// </summary>
      20. [Column("subscribed")]
      21. public List<Int32> Subscribed { get; set; } = new List<Int32>();
      23. /// <summary>
      24. /// Id записей, которые добавли пользователи-продавцы, на которых подписан данный пользователь.
      25. /// </summary>
      26. [Column("newposts")]
      27. public List<Int32> NewPosts { get; set; } = new List<Int32>();
      28. /// <summary>
      29. /// Конструктор от формы создания пользователя.
      30. /// </summary>
      31. /// <param name="data">Форма регистрации.</param>
      32. public Buyer(RegistrationForm data) : base(data)
      33. {
      35. }
      36. /// <summary>
      37. /// Пустой конструктор.
      38. /// </summary>
      39. public Buyer()
      40. {
      42. }
      43. }
      44. }
   4. BuyerOrderView.cs
      1. using System;
      2. namespace ShopsAggregatorWebApi.Models
      3. {
      4. /// <summary>
      5. /// Модель данных для заказа пользователя-покупателя.
      6. /// </summary>
      7. public class BuyerOrderView
      8. {
      9. /// <summary>
      10. /// Id заказа.
      11. /// </summary>
      12. public Int32 OrderId { get; set; }
      13. /// <summary>
      14. /// Экземпляр пользователя-покупателя, который создал запись.
      15. /// </summary>
      16. public Seller OrderSeller { get; set; }
      17. /// <summary>
      18. /// Запись, которую опубликовал пользователь-продавец.
      19. /// </summary>
      20. public Post SellerPost { get; set; }
      21. /// <summary>
      22. /// Одобрил ли пользователь-продавец заказ.
      23. /// </summary>
      24. public Boolean IsSucceded { get; set; }
      25. /// <summary>
      26. /// Отклонил ли пользователь-продавец заказ.
      27. /// </summary>
      28. public Boolean IsCanceled { get; set; }
      29. /// <summary>
      30. /// Пустой конструкток.
      31. /// </summary>
      32. public BuyerOrderView()
      33. {
      35. }
      36. /// <summary>
      37. /// Конструктор от записи, заказа и пользователя-продавца.
      38. /// </summary>
      39. /// <param name="order">Экземпляр типа заказа.</param>
      40. /// <param name="post">Экземпляр типа записи.</param>
      41. /// <param name="seller">Экземпляр типа пользователя-продавца.</param>
      42. public BuyerOrderView(Order order, Post post, Seller seller)
      43. {
      44. OrderSeller = seller;
      45. SellerPost = post;
      46. IsSucceded = order.IsSucceded;
      47. IsCanceled = order.IsCanceled;
      48. OrderId = order.Id;
      49. }
      50. }
      51. }
   5. BuyerPostView.cs
      1. using System;
      2. using System.Collections.Generic;
      3. namespace ShopsAggregatorWebApi.Models
      4. {
      5. /// <summary>
      6. /// Модель данных записи для пользователя-покупателя.
      7. /// </summary>
      8. public class BuyerPostView
      9. {
      10. /// <summary>
      11. /// Имя пользователя-продавца, который опубликовал запись.
      12. /// </summary>
      13. public String Username { get; set; }
      14. /// <summary>
      15. /// Id записи.
      16. /// </summary>
      17. public Int32 PostId { get; set; }
      18. /// <summary>
      19. /// Информация о записи.
      20. /// </summary>
      21. public String Info { get; set; }
      22. /// <summary>
      23. /// Пользователи-покупатели, которым понравилась запись.
      24. /// </summary>
      25. public List<Int32> Likers { get; set; } = new List<Int32>();
      26. /// <summary>
      27. /// Конструктор от записи и пользователя-продавца.
      28. /// </summary>
      29. /// <param name="post">Экземпляр типа записи.</param>
      30. /// <param name="creator">Экземпляр типа пользователя-продавца.</param>
      31. public BuyerPostView(Post post, Seller creator)
      32. {
      33. Username = creator.Username;
      34. PostId = post.Id;
      35. Info = post.Info;
      36. Likers = post.Likers;
      37. }
      38. }
      39. }
   6. CreatePostForm.cs
      1. using System;
      2. using System.Collections.Generic;
      3. namespace ShopsAggregatorWebApi.Models
      4. {
      5. /// <summary>
      6. /// Модель для создания записи, которая отправляется на сервер.
      7. /// </summary>
      8. public class CreatePostForm
      9. {
      10. /// <summary>
      11. /// Id пользователя-продавца, который создает запись.
      12. /// </summary>
      13. public Int32 CreatorId { get; set; }
      14. /// <summary>
      15. /// Информация о записи.
      16. /// </summary>
      17. public String Info { get; set; }
      18. /// <summary>
      19. /// Массив байтов картинки записи.
      20. /// </summary>
      21. public List<Int32> ImageBytes { get; set; } = new List<Int32>();
      22. }
      23. }
   7. IconPostForm.cs
      1. using System;
      2. using System.Collections.Generic;
      3. namespace ShopsAggregatorWebApi.Models
      4. {
      5. /// <summary>
      6. /// Экземпляр типа установки иконки пользователю.
      7. /// </summary>
      8. public class IconPostForm
      9. {
      10. /// <summary>
      11. /// Байты фотографии, выбранной пользователем.
      12. /// </summary>
      13. public List<Int32> IconBytesArr { get; set; }
      15. /// <summary>
      16. /// Id пользователя, который изменяет иконку.
      17. /// </summary>
      18. public Int32 ToId { get; set; }
      19. }
      20. }
   8. Order.cs
      1. using System;
      2. using System.ComponentModel.DataAnnotations;
      3. using System.ComponentModel.DataAnnotations.Schema;
      4. namespace ShopsAggregatorWebApi.Models
      5. {
      6. /// <summary>
      7. /// Экземпляр типа заказ.
      8. /// </summary>
      9. [Table("orders")]
      10. public class Order
      11. {
      12. /// <summary>
      13. /// Id заказа.
      14. /// </summary>
      15. [Key,DatabaseGenerated(DatabaseGeneratedOption.Identity)]
      16. [Column("id")]
      17. public Int32 Id { get; set; }
      18. /// <summary>
      19. /// Id записи, на которую создали запись.
      20. /// </summary>
      21. [Column("postid")]
      22. public Int32 PostId { get; set; }
      23. /// <summary>
      24. /// Id пользователя-покупателя, который создал заказ.
      25. /// </summary>
      26. [Column("customerid")]
      27. public Int32 BuyerId { get; set; }
      28. /// <summary>
      29. /// Подтвердил ли пользователь-продавец заказ.
      30. /// </summary>
      31. [Column("issucceded")]
      32. public Boolean IsSucceded { get; set; } = false;
      33. /// <summary>
      34. /// Отклонил ли пользователь-продавец заказ.
      35. /// </summary>
      36. [Column("iscanceled")]
      37. public Boolean IsCanceled { get; set; }= false;
      38. }
      39. }
   9. Post.cs
      1. using System;
      2. using System.Collections.Generic;
      3. using System.ComponentModel.DataAnnotations;
      4. using System.ComponentModel.DataAnnotations.Schema;
      5. using System.IO;
      6. namespace ShopsAggregatorWebApi.Models
      7. {
      8. /// <summary>
      9. /// Тип записи, которую создает пользователь-продавец
      10. /// </summary>
      11. [Table("posts")]
      12. public class Post
      13. {
      14. /// <summary>
      15. /// Путь для сохранения картинок.
      16. /// </summary>
      17. private const String PostsPhotoDirectory = "../images";
      18. /// <summary>
      19. /// Id записи.
      20. /// </summary>
      21. [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
      22. [Column("id")]
      23. public Int32 Id { get; set; }
      24. /// <summary>
      25. /// Id создателя записи - пользователя-продавца.
      26. /// </summary>
      27. [Column("creatorid")]
      28. public Int32 CreatorId { get; set; }
      29. /// <summary>
      30. /// Информации о записи.
      31. /// </summary>
      32. [Column("info")]
      33. public String Info { get; set; }
      35. /// <summary>
      36. /// Id пользователей-покупателей, которым понравилась эта запись.
      37. /// </summary>
      38. [Column("likers")]
      39. public List<Int32> Likers { get; set; } = new List<Int32>();
      41. /// <summary>
      42. /// Ставит полям значения из формы и вызывает метод добавления фотографии.
      43. /// </summary>
      44. /// <param name="form">Форма создания записи.</param>
      45. public void CreatePostFromForm(CreatePostForm form)
      46. {
      47. this.CreatorId = form.CreatorId;
      48. this.Info = form.Info;
      49. AddPostImage(form);
      50. }
      51. /// <summary>
      52. /// Пустой конструктор.
      53. /// </summary>
      54. public Post()
      55. {
      57. }
      59. /// <summary>
      60. /// Добавляет фотографию для записи из формы.
      61. /// </summary>
      62. /// <param name="form">Форма записи.</param>
      63. private void AddPostImage(CreatePostForm form)
      64. {
      65. if (!Directory.Exists(PostsPhotoDirectory))
      66. Directory.CreateDirectory(PostsPhotoDirectory);
      67. using (FileStream fs = new FileStream(PostsPhotoDirectory + $"/{this.Id}photo.jpeg",
      68. FileMode.OpenOrCreate,
      69. FileAccess.Write))
      70. {
      71. foreach (Int32 imageByte in form.ImageBytes)
      72. {
      73. fs.WriteByte((Byte) imageByte);
      74. }
      75. }
      76. }
      77. }
      78. }
   10. RegistrationForm.cs
       1. using System;
       2. namespace ShopsAggregatorWebApi.Models
       3. {
       4. /// <summary>
       5. /// Форма для регистрации пользователя в сервисе.
       6. /// </summary>
       7. public class RegistrationForm
       8. {
       9. /// <summary>
       10. /// Имя пользователя.
       11. /// </summary>
       12. public String Username { get; set; }
       13. /// <summary>
       14. /// Email пользователя.
       15. /// </summary>
       16. public String Email { get; set; }
       17. /// <summary>
       18. /// Пароль пользователя.
       19. /// </summary>
       20. public String Password { get; set; }
       21. /// <summary>
       22. /// Является ли пользователь пользователем-продавцом.
       23. /// </summary>
       24. public Boolean IsSeller { get; set; }
       25. }
       26. }
   11. Seller.cs
       1. using System;
       2. using System.Collections.Generic;
       3. using System.ComponentModel.DataAnnotations.Schema;
       4. namespace ShopsAggregatorWebApi.Models
       5. {
       6. /// <summary>
       7. /// Тип пользователя-продавца.
       8. /// </summary>
       9. [Table("sellers")]
       10. public class Seller : User
       11. {
       12. /// <summary>
       13. /// Id записей, созданных пользователем.
       14. /// </summary>
       15. [Column("items")]
       16. public List<Int32> Items { get; set; } = new List<Int32>();
       17. /// <summary>
       18. /// Id подписчиков, которые подписались на данного пользователя.
       19. /// </summary>
       20. [Column("subscribers")]
       21. public List<Int32> Subscribers { get; set; } = new List<Int32>();
       23. /// <summary>
       24. /// Пустой конструктор.
       25. /// </summary>
       26. public Seller() : base()
       27. {
       29. }
       30. /// <summary>
       31. /// Конструктор от формы создания пользователя.
       32. /// </summary>
       33. /// <param name="data">Форма регистрации.</param>
       34. public Seller(RegistrationForm data) : base(data)
       35. {
       37. }
       38. }
       39. }
   12. SellerOrderView.cs
       1. using System;
       2. namespace ShopsAggregatorWebApi.Models
       3. {
       4. /// <summary>
       5. /// Модель заказа для пользователя-покупателя.
       6. /// </summary>
       7. public class SellerOrderView
       8. {
       9. /// <summary>
       10. /// Id заказа.
       11. /// </summary>
       12. public Int32 OrderId { get; set; }
       13. /// <summary>
       14. /// Экземпляр типа пользователя, который сделал заказ.
       15. /// </summary>
       16. public Buyer OrderBuyer { get; set; }
       17. /// <summary>
       18. /// Экземпляр типа записи, которую заказал пользователь.
       19. /// </summary>
       20. public Post CurrentPost { get; set; }
       21. /// <summary>
       22. /// Одобрил ли пользователь-продавец заказ.
       23. /// </summary>
       24. public Boolean IsSucceded { get; set; }
       25. /// <summary>
       26. /// Отклонил ли пользователь-продавец заказ.
       27. /// </summary>
       28. public Boolean IsCanceled { get; set; }
       29. /// <summary>
       30. /// Пустой конструктор.
       31. /// </summary>
       32. public SellerOrderView()
       33. {
       35. }
       36. /// <summary>
       37. /// Конструктор от заказа, записи и пользователя-покупателя.
       38. /// </summary>
       39. /// <param name="order">Экземпляр типа заказа.</param>
       40. /// <param name="post">Экземпляр типа записи.</param>
       41. /// <param name="buyer">Экземпляр типа пользователя-покупателя.</param>
       42. public SellerOrderView(Order order, Post post, Buyer buyer)
       43. {
       44. OrderBuyer = buyer;
       45. CurrentPost = post;
       46. IsSucceded = order.IsSucceded;
       47. IsCanceled = order.IsCanceled;
       48. OrderId = order.Id;
       49. }
       50. }
       51. }
   13. User.cs
       1. using System;
       2. using System.Collections.Generic;
       3. using System.ComponentModel.DataAnnotations;
       4. using System.ComponentModel.DataAnnotations.Schema;
       5. namespace ShopsAggregatorWebApi.Models
       6. {
       7. /// <summary>
       8. /// Общий тип пользователей.
       9. /// </summary>
       10. public class User
       11. {
       12. /// <summary>
       13. /// Id пользователя.
       14. /// </summary>
       15. [Key,DatabaseGenerated(DatabaseGeneratedOption.Identity)]
       16. [Column("id")]
       17. public Int32 Id { get; set; }
       18. /// <summary>
       19. /// Имя пользователя.
       20. /// </summary>
       21. [Column("username")]
       22. public String Username { get; set; }
       23. /// <summary>
       24. /// Email пользователя.
       25. /// </summary>
       26. [Column("email")]
       27. public String Email { get; set; }
       28. /// <summary>
       29. /// Пароль пользователя.
       30. /// </summary>
       31. [Column("password")]
       32. public String Password { get; set; }
       34. /// <summary>
       35. /// Имя файла с иконкой пользователя.
       36. /// </summary>
       37. [Column("iconpath")]
       38. public String IconPath { get; set; } = "standarticon.jpeg";
       39. /// <summary>
       40. /// Информация о пользователе.
       41. /// </summary>
       42. [Column("info")]
       43. public String Info { get; set; }
       44. /// <summary>
       45. /// Id заказов пользователя.
       46. /// </summary>
       47. [Column("orders")]
       48. public List<Int32> Orders { get; set; } = new List<Int32>();
       50. /// <summary>
       51. /// Пустой конструктор.
       52. /// </summary>
       53. public User()
       54. {
       55. }
       56. /// <summary>
       57. /// Конструктор от формы создания пользователя.
       58. /// </summary>
       59. /// <param name="data">Форма регистрации.</param>
       60. public User(RegistrationForm data)
       61. {
       62. this.Email = data.Email;
       63. this.Username = data.Username;
       64. this.Password = data.Password;
       65. }
       66. }
       67. }
   14. AuthorizationController.cs
       1. using System;
       2. using System.Collections.Generic;
       3. using Microsoft.AspNetCore.Mvc;
       4. using ShopsAggregatorWebApi.Models;
       5. using ShopsAggregatorWebApi.Services;
       6. namespace ShopsAggregatorWebApi.Controllers
       7. {
       8. /// <summary>
       9. /// Контроллер взаимодействия с пользователями.
       10. /// </summary>
       11. [Route("api/users")]
       12. [ApiController]
       13. public class AuthorizationContoller : ControllerBase
       14. {
       15. /// <summary>
       16. /// Сервис базы данных.
       17. /// </summary>
       18. private readonly DatabaseContext \_service;
       19. /// <summary>
       20. /// Конструктор контроллера.
       21. /// </summary>
       22. /// <param name="service">Сервис баз данных.</param>
       23. public AuthorizationContoller(DatabaseContext service)
       24. {
       25. \_service = service;
       26. }
       27. /// <summary>
       28. /// Метод получения всех пользователей.
       29. /// </summary>
       30. /// <returns>Список всех пользователей.</returns>
       31. [HttpGet]
       32. public ActionResult<List<User>> GetAllUsers() => \_service.GetAllUsers();
       34. /// <summary>
       35. /// Регистрирует нового пользователя.
       36. /// </summary>
       37. /// <param name="user">Форма регистрации пользователя.</param>
       38. /// <returns>Результат регистрации.</returns>
       39. [HttpPost("reg")]
       40. public ActionResult RegistrateNewUser([FromBody]RegistrationForm user)
       41. {
       42. User registeredUser = null;
       43. if (user.IsSeller)
       44. registeredUser = \_service.CreateUser(new Seller(user));
       45. else
       46. registeredUser = \_service.CreateUser(new Buyer(user));
       47. if (registeredUser != null)
       48. {
       49. return Ok("Аккаунт успешно создан");
       50. }
       52. return BadRequest("Такой аккаунт уже существует!");
       53. }
       55. /// <summary>
       56. /// Аутентификация пользователя-покупателя.
       57. /// </summary>
       58. /// <param name="login">Логин для входа.</param>
       59. /// <param name="password">Пароль для входа.</param>
       60. /// <returns>Экземпляр пользователя.</returns>
       61. [HttpGet("authBuyer")]
       62. public ActionResult<User>TryAuthBuyer(String login, String password)
       63. {
       64. return \_service.GetBuyer(login, password);
       65. }
       67. /// <summary>
       68. /// Аутентификация пользователя-продавца.
       69. /// </summary>
       70. /// <param name="login">Логин для входа.</param>
       71. /// <param name="password">Пароль для входа.</param>
       72. /// <returns>Экземпляр пользователя.</returns>
       73. [HttpGet("authSeller")]
       74. public ActionResult<User> TryAuthSeller(String login, String password)
       75. {
       76. return \_service.GetSeller(login, password);
       77. }
       79. /// <summary>
       80. /// Добавление иконки пользователю-продавцу.
       81. /// </summary>
       82. /// <param name="icon">Форма добавления иконки.</param>
       83. /// <returns>Результат добавления.</returns>
       84. [HttpPut("addSellerIcon")]
       85. public IActionResult UpdateSellerIcon(IconPostForm icon)
       86. {
       87. if (\_service.AddIconToSeller(icon, icon.ToId))
       88. return Ok();
       89. return BadRequest();
       90. }
       91. /// <summary>
       92. /// Добавление иконки пользователю-покупателю.
       93. /// </summary>
       94. /// <param name="icon">Форма добавления иконки.</param>
       95. /// <returns>Результат добавления.</returns>
       96. [HttpPut("addBuyerIcon")]
       97. public IActionResult UpdateBuyerIcon([FromBody]IconPostForm icon)
       98. {
       99. if (\_service.AddIconToBuyer(icon, icon.ToId))
       100. return Ok();
       101. return BadRequest();
       102. }
       103. /// <summary>
       104. /// Обновляют информацию о пользователе-покупателе.
       105. /// </summary>
       106. /// <param name="buyerId">Id пользователя-покупателя.</param>
       107. /// <param name="info">Новая информация о пользователе.</param>
       108. /// <returns>Результат обновления.</returns>
       109. [HttpPut("setBuyerInfo")]
       110. public IActionResult SetBuyerInfo(Int32 buyerId, String info)
       111. {
       112. \_service.SetBuyerInfo(buyerId, info);
       113. return Ok();
       114. }
       116. /// <summary>
       117. /// Обновляют информацию о пользователе-продавце.
       118. /// </summary>
       119. /// <param name="sellerId">Id пользователя-продавца.</param>
       120. /// <param name="info">Новая информация о пользователе.</param>
       121. /// <returns>Результат обновления.</returns>
       122. [HttpPut("setSellerInfo")]
       123. public IActionResult SetSellerInfo(Int32 sellerId, String info)
       124. {
       125. \_service.SetSellerInfo(sellerId, info);
       126. return Ok();
       127. }
       128. }
       129. }
   15. OrdersController.cs
       1. using System;
       2. using System.Collections.Generic;
       3. using Microsoft.AspNetCore.Mvc;
       4. using ShopsAggregatorWebApi.Models;
       5. using ShopsAggregatorWebApi.Services;
       6. namespace ShopsAggregatorWebApi.Controllers
       7. {
       8. /// <summary>
       9. /// Контроллер взаимодействия с заказами.
       10. /// </summary>
       11. [Route("api/orders")]
       12. public class OrdersController : ControllerBase
       13. {
       14. /// <summary>
       15. /// Сервис базы данных.
       16. /// </summary>
       17. private readonly DatabaseContext \_service;
       18. /// <summary>
       19. /// Конструктор контроллера.
       20. /// </summary>
       21. /// <param name="service">Сервис баз данных.</param>
       22. public OrdersController(DatabaseContext service)
       23. {
       24. \_service = service;
       25. }
       27. /// <summary>
       28. /// Добавляет новую запись.
       29. /// </summary>
       30. /// <param name="buyerId">Id пользователя-покупателя.</param>
       31. /// <param name="postId">Id записи.</param>
       32. /// <returns>Резульат добавления.</returns>
       33. [HttpPost("addOrder")]
       34. public IActionResult AddOrder(Int32 buyerId, Int32 postId)
       35. {
       36. \_service.AddOrder(buyerId, postId);
       37. return Ok();
       38. }
       39. /// <summary>
       40. /// Получает список заказов пользователя-покупателя.
       41. /// </summary>
       42. /// <param name="buyerId">Id пользователя-покупателя.</param>
       43. /// <returns>Список заказов.</returns>
       44. [HttpGet("getBuyerOrders")]
       45. public List<BuyerOrderView> GetBuyerOrders(Int32 buyerId)
       46. {
       47. return \_service.GetBuyerOrders(buyerId);
       48. }
       49. /// <summary>
       50. /// Получает список заказов пользователя-продавца.
       51. /// </summary>
       52. /// <param name="sellerId">Id пользователя-продавца.</param>
       53. /// <returns>Список заказов.</returns>
       54. [HttpGet("getSellerOrders")]
       55. public List<SellerOrderView> GetSellerorders(Int32 sellerId)
       56. {
       57. return \_service.GetSellerOrders(sellerId);
       58. }
       59. /// <summary>
       60. /// Ставит одобренный статус заказу.
       61. /// </summary>
       62. /// <param name="orderId">Id заказа, которому устанавливается статус.</param>
       63. /// <returns>Резльтат изменения статуса.</returns>
       64. [HttpPut("setOrderSuccess")]
       65. public IActionResult SetOrderSuccess(Int32 orderId)
       66. {
       67. \_service.SetOrderSuccess(orderId);
       68. return Ok();
       69. }
       71. /// <summary>
       72. /// Ставит отклоненный статус заказу.
       73. /// </summary>
       74. /// <param name="orderId">Id заказа, которому устанавливается статус.</param>
       75. /// <returns>Резльтат изменения статуса.</returns>
       76. [HttpPut("setOrderCanceled")]
       77. public IActionResult SetOrderCanceled(Int32 orderId)
       78. {
       79. \_service.SetOrderCanceled(orderId);
       80. return Ok();
       81. }
       82. /// <summary>
       83. /// Удаляет заказ.
       84. /// </summary>
       85. /// <param name="orderId">Id заказа, который удаляют.</param>
       86. /// <returns>Результат удаления.</returns>
       87. [HttpDelete("deleteOrder")]
       88. public IActionResult DeleteOrder(Int32 orderId)
       89. {
       90. \_service.DeleteOrder(orderId);
       91. return Ok();
       92. }
       93. }
       94. }
   16. PostController.cs
       1. using System;
       2. using System.Collections.Generic;
       3. using Microsoft.AspNetCore.Mvc;
       4. using ShopsAggregatorWebApi.Models;
       5. using ShopsAggregatorWebApi.Services;
       6. namespace ShopsAggregatorWebApi.Controllers
       7. {
       8. /// <summary>
       9. /// Контроллер взаимодействия с записями.
       10. /// </summary>
       11. [Route("api/posts")]
       12. public class PostController : ControllerBase
       13. {
       14. /// <summary>
       15. /// Сервис базы данных.
       16. /// </summary>
       17. private readonly DatabaseContext \_service;
       18. /// <summary>
       19. /// Конструктор контроллера.
       20. /// </summary>
       21. /// <param name="service">Сервис баз данных.</param>
       22. public PostController(DatabaseContext service)
       23. {
       24. \_service = service;
       25. }
       26. /// <summary>
       27. /// Создает новую запись.
       28. /// </summary>
       29. /// <param name="postForm">Форма для создания записи.</param>
       30. /// <returns>Результат создания записи.</returns>
       31. [HttpPost("create")]
       32. public IActionResult CreatePost([FromBody]CreatePostForm postForm)
       33. {
       34. if (\_service.CreatePost(postForm))
       35. return Ok();
       36. return BadRequest();
       37. }
       38. /// <summary>
       39. /// Получает запись по id.
       40. /// </summary>
       41. /// <param name="id">Id записи.</param>
       42. /// <returns>полученная запись.</returns>
       43. [HttpGet("getPost")]
       44. public Post GetPostById(Int32 id)
       45. {
       46. return \_service.GetPostById(id);
       47. }
       48. /// <summary>
       49. /// Получает записи, опубликованные пользователем-продавцом.
       50. /// </summary>
       51. /// <param name="sellerName">Имя пользователя-продавца.</param>
       52. /// <returns>Список записей пользователя-продавца.</returns>
       53. [HttpGet("getSellerPosts")]
       54. public List<Post> GetPostsByCreatorId(String sellerName)
       55. {
       56. return \_service.GetPostsByCreatorName(sellerName);
       57. }
       58. /// <summary>
       59. /// Получает новые записи пользователя-покупателя.
       60. /// </summary>
       61. /// <param name="buyerId">Id пользователя-покупателя.</param>
       62. /// <returns>Список новых записей пользователя-покупателя.</returns>
       63. [HttpGet("getBuyerNewPost")]
       64. public List<BuyerPostView> GetBuyerNewPosts(Int32 buyerId)
       65. {
       66. return \_service.GetBuyerNewPosts(buyerId);
       67. }
       68. /// <summary>
       69. /// Добавляет лайк пользователя-покупателя к записи.
       70. /// </summary>
       71. /// <param name="likerId">Id пользователя-покупателя.</param>
       72. /// <param name="postId">Id записи.</param>
       73. /// <returns>Результат добавления лайка.</returns>
       74. [HttpPut("addLike")]
       75. public IActionResult AddLikeToPost(Int32 likerId, Int32 postId)
       76. {
       77. \_service.AddLikeToPost(likerId, postId);
       78. return Ok();
       79. }
       80. /// <summary>
       81. /// Удаляет лайк пользователя-покупателя у записи.
       82. /// </summary>
       83. /// <param name="likerId">Id пользователя-покупателя.</param>
       84. /// <param name="postId">Id записи.</param>
       85. /// <returns>Результат удаления лайка.</returns>
       86. [HttpPut("removeLike")]
       87. public IActionResult RemoveLikeFromPost(Int32 likerId, Int32 postId)
       88. {
       89. \_service.RemoveLikeFromPost(likerId, postId);
       90. return Ok();
       91. }
       92. /// <summary>
       93. /// Получает записи, которые понравились пользователю-покупателю.
       94. /// </summary>
       95. /// <param name="buyerId">Id пользователя-покупателя.</param>
       96. /// <returns>Список понравившихся записей.</returns>
       97. [HttpGet("getBuyerLikedPosts")]
       98. public List<BuyerPostView> GetBuyerLikedPosts(Int32 buyerId)
       99. {
       100. return \_service.GetBuyerLikedPosts(buyerId);
       101. }
       102. }
       103. }
   17. SearchController.cs
       1. using System;
       2. using System.Collections.Generic;
       3. using Microsoft.AspNetCore.Mvc;
       4. using ShopsAggregatorWebApi.Models;
       5. using ShopsAggregatorWebApi.Services;
       6. namespace ShopsAggregatorWebApi.Controllers
       7. {
       8. /// <summary>
       9. /// Контроллер взаимодействия с записями.
       10. /// </summary>
       11. [Route("api/search")]
       12. public class SearchController : Controller
       13. {
       14. /// <summary>
       15. /// Сервис базы данных.
       16. /// </summary>
       17. private readonly DatabaseContext \_service;
       18. /// <summary>
       19. /// Конструктор контроллера.
       20. /// </summary>
       21. /// <param name="service">Сервис баз данных.</param>
       22. public SearchController(DatabaseContext service)
       23. {
       24. \_service = service;
       25. }
       26. /// <summary>
       27. /// Получает пользователей-продавцов, которые содержат введенную строку в имени.
       28. /// </summary>
       29. /// <param name="q">Строка поиска.</param>
       30. /// <returns>Список полученных пользователе-продавцов.</returns>
       31. [HttpGet]
       32. public ActionResult<List<String>> SearchSellers(String q)
       33. {
       34. var users = \_service.SearchSellers(q);
       35. return users;
       36. }
       37. /// <summary>
       38. /// Получает пользователя-продавца по имени.
       39. /// </summary>
       40. /// <param name="sellerName">Имя пользователя-продавца.</param>
       41. /// <returns>Полученный пользователь-продавец.</returns>
       42. [HttpGet("getSeller")]
       43. public Seller GetSellerByName(String sellerName)
       44. {
       45. return \_service.GetSeller(sellerName);
       46. }
       47. }
       48. }
   18. SubscribeController.cs
       1. using System;
       2. using Microsoft.AspNetCore.Mvc;
       3. using ShopsAggregatorWebApi.Services;
       4. namespace ShopsAggregatorWebApi.Controllers
       5. {
       6. /// <summary>
       7. /// Контроллер взаимодействия с подписками.
       8. /// </summary>
       9. [Route("api/sub")]
       10. public class SubscribeController : Controller
       11. {
       12. /// <summary>
       13. /// Сервис базы данных.
       14. /// </summary>
       15. private readonly DatabaseContext \_service;
       16. /// <summary>
       17. /// Конструктор контроллера.
       18. /// </summary>
       19. /// <param name="service">Сервис баз данных.</param>
       20. public SubscribeController(DatabaseContext service)
       21. {
       22. \_service = service;
       23. }
       24. /// <summary>
       25. /// Добавляет подписчика пользователя-покупателя к пользователю-продавцу.
       26. /// </summary>
       27. /// <param name="buyerId">Id пользователя-покупателя.</param>
       28. /// <param name="sellerName">Id пользователя-продавца.</param>
       29. /// <returns>Результат подписки.</returns>
       30. [HttpPut("addSub")]
       31. public IActionResult AddSubscriberToSeller(Int32 buyerId, String sellerName)
       32. {
       33. if (\_service.AddSubscriber(buyerId, sellerName))
       34. return Ok();
       35. return BadRequest();
       36. }
       37. /// <summary>
       38. /// Удаляет из подписчиков пользователя-покупателя у пользователю-продавцу.
       39. /// </summary>
       40. /// <param name="buyerId">Id пользователя-покупателя.</param>
       41. /// <param name="sellerName">Id пользователя-продавца.</param>
       42. /// <returns>Результат отписки.</returns>
       43. [HttpPut("rmSub")]
       44. public IActionResult RemoveSellerSubscriber(Int32 buyerId, String sellerName)
       45. {
       46. if (\_service.RemoveSubscriber(buyerId, sellerName))
       47. return Ok();
       48. return BadRequest();
       49. }
       51. }
       52. }
   19. DatabaseContext.cs
       1. using System;
       2. using System.Collections.Generic;
       3. using System.IO;
       4. using System.Linq;
       5. using Microsoft.EntityFrameworkCore;
       6. using ShopsAggregatorWebApi.Models;
       7. namespace ShopsAggregatorWebApi.Services
       8. {
       9. /// <summary>
       10. /// Сервис работы с базой данных.
       11. /// </summary>
       12. public class DatabaseContext : DbContext
       13. {
       14. /// <summary>
       15. /// Путь для сохранения картинок.
       16. /// </summary>
       17. private const String UsersIconsDirPath = "../images";
       18. /// <summary>
       19. /// База данных пользователей-продавцов.
       20. /// </summary>
       21. private DbSet<Seller> Sellers { get; set; }
       22. /// <summary>
       23. /// База данных пользователкей-покупателей.
       24. /// </summary>
       25. private DbSet<Buyer> Buyers { get; set; }
       27. /// <summary>
       28. /// База данных записей пользователей-продавцов.
       29. /// </summary>
       30. private DbSet<Post> Posts { get; set; }
       31. /// <summary>
       32. /// База данных заказов пользователей-покупателей.
       33. /// </summary>
       34. private DbSet<Order> Orders { get; set; }
       36. /// <summary>
       37. /// Конструктор.
       38. /// </summary>
       39. /// <param name="options">Опции для сервиса.</param>
       40. public DatabaseContext(DbContextOptions<DatabaseContext> options) : base(options)
       41. {
       43. }
       44. /// <summary>
       45. /// Получает всех пользователей, зарегестрированных в системе.
       46. /// </summary>
       47. /// <returns>Список пользователей</returns>
       48. public List<User> GetAllUsers()
       49. {
       50. List<User> users = Sellers.ToList().ConvertAll(x => (User) x);
       51. users.AddRange(Buyers.ToList().ConvertAll(x => (User)x));
       52. return users;
       53. }
       54. /// <summary>
       55. /// Создает нового пользователя-покупателя.
       56. /// </summary>
       57. /// <param name="user">Экземпляр пользователя-покупателя.</param>
       58. /// <returns>Созданный пользователь.</returns>
       59. public User CreateUser(Buyer user)
       60. {
       61. if (IsSameSellerCreated(user) || IsSameBuyerCreated(user))
       62. return null;
       63. Buyers.Add(user);
       64. SaveChangesAsync();
       65. return user;
       66. }
       67. /// <summary>
       68. /// Создает нового пользователя-продавца.
       69. /// </summary>
       70. /// <param name="user">Экземпляр пользователя-продавца.</param>
       71. /// <returns>Созданный пользователь.</returns>
       72. public User CreateUser(Seller user)
       73. {
       74. if (IsSameSellerCreated(user) || IsSameBuyerCreated(user))
       75. return null;
       76. Sellers.Add(user);
       77. SaveChangesAsync();
       78. return user;
       79. }
       80. /// <summary>
       81. /// Проверяет создан ли пользователь-продавец с таким же email или username.
       82. /// </summary>
       83. /// <param name="user">Экземпляр проверяемого пользователя-продавца.</param>
       84. /// <returns>Результат проверки.</returns>
       85. private Boolean IsSameSellerCreated(User user)
       86. {
       87. return (from seller in Sellers.ToList() where seller.Username == user.Username
       88. || seller.Email == user.Email select seller)
       89. .FirstOrDefault() != null;
       90. }
       91. /// <summary>
       92. /// Проверяет создан ли пользователь-покупатель с таким же email или username.
       93. /// </summary>
       94. /// <param name="user">Экземпляр проверяемого пользователя-покупатель.</param>
       95. /// <returns>Результат проверки.</returns>
       96. private Boolean IsSameBuyerCreated(User user)
       97. {
       98. return (from buyer in Buyers.ToList() where buyer.Username == user.Username
       99. || buyer.Email == user.Email select buyer)
       100. .FirstOrDefault() != null;
       101. }
       102. /// <summary>
       103. /// Получает пользователя по логину и паролю.
       104. /// </summary>
       105. /// <param name="login">Логин.</param>
       106. /// <param name="password">Пароль.</param>
       107. /// <returns>Полученный пользователь.</returns>
       108. public Buyer GetBuyer(String login, String password)
       109. {
       110. return (from buyer in Buyers.ToList()
       111. where (buyer.Username == login || buyer.Username == login)
       112. && buyer.Password == password
       113. select buyer).FirstOrDefault();
       114. }
       115. /// <summary>
       116. /// Получает пользователя-покупателя по его id.
       117. /// </summary>
       118. /// <param name="id">Id пользователя.</param>
       119. /// <returns>Экземпляр типа пользователя-покупателя.</returns>
       120. private Buyer GetBuyer(Int32 id)
       121. {
       122. return (from buyer in Buyers.ToList() where buyer.Id == id select buyer).FirstOrDefault();
       123. }
       124. /// <summary>
       125. /// Получает пользователя по логину и паролю.
       126. /// </summary>
       127. /// <param name="login">Логин.</param>
       128. /// <param name="password">Пароль.</param>
       129. /// <returns>Полученный пользователь.</returns>
       130. public Seller GetSeller(string login, string password)
       131. {
       132. return (from seller in Sellers.ToList()
       133. where (seller.Username == login || seller.Username == login)
       134. && seller.Password == password
       135. select seller).FirstOrDefault();
       136. }
       137. /// <summary>
       138. /// Получает пользователя-продавца по его id.
       139. /// </summary>
       140. /// <param name="id">Id пользователя.</param>
       141. /// <returns>Экземпляр типа пользователя-продавца.</returns>
       142. private Seller GetSeller(Int32 id)
       143. {
       144. return (from seller in Sellers.ToList() where seller.Id == id select seller).FirstOrDefault();
       145. }
       146. /// <summary>
       147. /// Получает пользователя по имени.
       148. /// </summary>
       149. /// <param name="sellerName">Имя пользователя.</param>
       150. /// <returns>Полученный пользователь.</returns>
       151. public Seller GetSeller(String sellerName)
       152. {
       153. Seller currentSeller = (from seller in Sellers.ToList() where seller.Username == sellerName select seller)
       154. .FirstOrDefault();
       155. if (currentSeller == null)
       156. return null;
       157. currentSeller.Id = 0;
       158. currentSeller.Password = "";
       159. return currentSeller;
       160. }
       161. /// <summary>
       162. /// Добавляет иконку пользователю-продавцу.
       163. /// </summary>
       164. /// <param name="icon">Форма с информацией об изображении.</param>
       165. /// <param name="sellerId">Id пользователя-продавца.</param>
       166. /// <returns>Результат добавления.</returns>
       167. public Boolean AddIconToSeller(IconPostForm icon, Int32 sellerId)
       168. {
       169. Seller toSeller =
       170. (from seller in Sellers.ToList() where seller.Id == sellerId select seller).FirstOrDefault();
       171. if (toSeller == null)
       172. return false;
       173. try
       174. {
       175. AddIconToUser(icon, toSeller);
       176. }
       177. catch (Exception)
       178. {
       179. return false;
       180. }
       181. toSeller.IconPath = $"{toSeller.Username}icon.jpeg";
       182. SaveChangesAsync();
       183. return true;
       184. }
       185. /// <summary>
       186. /// Добавляет иконку пользователю-покупателю.
       187. /// </summary>
       188. /// <param name="icon">Форма с информацией об изображении.</param>
       189. /// <param name="buyerId">Id пользователя-покупателя.</param>
       190. /// <returns>Результат добавления.</returns>
       191. public Boolean AddIconToBuyer(IconPostForm icon, Int32 buyerId)
       192. {
       193. Buyer toBuyer = (from buyer in Buyers.ToList() where buyer.Id == buyerId select buyer).FirstOrDefault();
       194. if (toBuyer == null)
       195. return false;
       196. try
       197. {
       198. AddIconToUser(icon, toBuyer);
       199. }
       200. catch (Exception)
       201. {
       202. return false;
       203. }
       204. toBuyer.IconPath = $"{toBuyer.Username}icon.jpeg";
       205. SaveChangesAsync();
       206. return true;
       207. }
       208. /// <summary>
       209. /// Добавляет иконку пользователю.
       210. /// </summary>
       211. /// <param name="icon">Форма с информацией об изображении.</param>
       212. /// <param name="toUser">Экземпляр пользователя.</param>
       213. /// <returns>Результат добавления.</returns>
       214. private void AddIconToUser(IconPostForm icon, User toUser)
       215. {
       216. if (!Directory.Exists(UsersIconsDirPath))
       217. Directory.CreateDirectory(UsersIconsDirPath);
       218. using (FileStream fs = new FileStream(UsersIconsDirPath + $"/{toUser.Username}icon.jpeg",
       219. FileMode.OpenOrCreate,
       220. FileAccess.Write))
       221. {
       222. foreach (Int32 iconByte in icon.IconBytesArr)
       223. {
       224. fs.WriteByte((Byte) iconByte);
       225. }
       226. }
       227. }
       228. /// <summary>
       229. /// Ищет пользователя-покупателя, который содержит в username поисковую строку.
       230. /// </summary>
       231. /// <param name="s">Поисковая строка.</param>
       232. /// <returns>Результат поиска.</returns>
       233. public List<String> SearchSellers(String s)
       234. {
       235. return (from seller in Sellers.ToList() where seller.Username.Contains(s) select seller.Username).ToList();
       236. }
       237. /// <summary>
       238. /// Добваляет подписчика.
       239. /// </summary>
       240. /// <param name="buyerId">Id пользователя-покупателя.</param>
       241. /// <param name="sellerName">Имя пользователя-продавца.</param>
       242. /// <returns>Результат добавления.</returns>
       243. public Boolean AddSubscriber(Int32 buyerId, String sellerName)
       244. {
       245. Seller currentSeller = (from seller in Sellers.ToList() where seller.Username == sellerName select seller)
       246. .FirstOrDefault();
       247. if (currentSeller == null)
       248. return false;
       249. return AddSubscriber(buyerId, currentSeller.Id);
       250. }
       252. /// <summary>
       253. /// Добваляет подписчика.
       254. /// </summary>
       255. /// <param name="buyerId">Id пользователя-покупателя.</param>
       256. /// <param name="sellerId">Id пользователя-продавца.</param>
       257. /// <returns>Результат добавления.</returns>
       258. private Boolean AddSubscriber(Int32 buyerId, Int32 sellerId)
       259. {
       260. Buyer buyer = GetBuyer(buyerId);
       261. Seller seller = GetSeller(sellerId);
       262. if(buyer == null || seller == null)
       263. return false;
       264. if(buyer.Subscribed == null)
       265. buyer.Subscribed = new List<Int32>();
       266. if(seller.Subscribers == null)
       267. seller.Subscribers = new List<Int32>();
       268. buyer.Subscribed.Add(sellerId);
       269. seller.Subscribers.Add(buyerId);
       270. SaveChangesAsync();
       271. return true;
       272. }
       273. /// <summary>
       274. /// Удаляет подписчика.
       275. /// </summary>
       276. /// <param name="buyerId">Id пользователя-покупателя.</param>
       277. /// <param name="sellerName">Имя пользователя-продавца.</param>
       278. /// <returns>Результат добавления.</returns>
       279. public Boolean RemoveSubscriber(Int32 buyerId, String sellerName)
       280. {
       281. Seller currentSeller = (from seller in Sellers.ToList() where seller.Username == sellerName select seller)
       282. .FirstOrDefault();
       283. if (currentSeller == null)
       284. return false;
       285. return RemoveSubscriber(buyerId, currentSeller.Id);
       286. }
       287. /// <summary>
       288. /// Удаляет подписчика.
       289. /// </summary>
       290. /// <param name="buyerId">Id пользователя-покупателя.</param>
       291. /// <param name="sellerId">Id пользователя-продавца.</param>
       292. /// <returns>Результат добавления.</returns>
       293. private Boolean RemoveSubscriber(Int32 buyerId, Int32 sellerId)
       294. {
       295. Buyer buyer = GetBuyer(buyerId);
       296. Seller seller = GetSeller(sellerId);
       297. if(buyer == null || seller == null)
       298. return false;
       299. if (buyer.Subscribed == null || seller.Subscribers == null)
       300. return false;
       301. buyer.Subscribed.Remove(sellerId);
       302. seller.Subscribers.Remove(buyerId);
       303. SaveChangesAsync();
       304. return true;
       305. }
       306. /// <summary>
       307. /// Создает новую запись и добавляет ее в базу данных.
       308. /// </summary>
       309. /// <param name="postForm">Форма новой записи.</param>
       310. /// <returns>Результат создания записи</returns>
       311. public Boolean CreatePost(CreatePostForm postForm)
       312. {
       313. if (postForm.ImageBytes == null || postForm.ImageBytes.Count == 0 || postForm.CreatorId < 0)
       314. return false;
       315. Post newPost = new Post();
       316. Posts.Add(newPost);
       317. SaveChanges();
       318. try
       319. {
       320. newPost.CreatePostFromForm(postForm);
       321. }
       322. catch (Exception)
       323. {
       324. Posts.Remove(newPost);
       325. return false;
       326. }
       327. Seller creator = GetSeller(postForm.CreatorId);
       328. if (creator == null)
       329. return false;
       330. if(creator.Items == null)
       331. creator.Items = new List<Int32>();
       332. creator.Items.Insert(0,newPost.Id);
       333. AddPostForSubscribers(creator, newPost.Id);
       334. SaveChangesAsync();
       335. return true;
       336. }
       337. /// <summary>
       338. /// Добавляет новую запись всем подписчикам пользователя-продавца.
       339. /// </summary>
       340. /// <param name="creator">Пользователь-создатель.</param>
       341. /// <param name="postId">Id записи.</param>
       342. private void AddPostForSubscribers(Seller creator, Int32 postId)
       343. {
       344. if(creator.Subscribers == null)
       345. creator.Subscribers = new List<Int32>();
       346. foreach (Int32 buyerId in creator.Subscribers)
       347. {
       348. Buyer subscriber = (from buyer in Buyers.ToList() where buyer.Id == buyerId select buyer).FirstOrDefault();
       349. if(subscriber == null)
       350. continue;
       351. if(subscriber.NewPosts == null)
       352. subscriber.NewPosts = new List<Int32>();
       353. subscriber.NewPosts.Insert(0,postId);
       354. }
       355. }
       356. /// <summary>
       357. /// Получает экземпляр записи по id.
       358. /// </summary>
       359. /// <param name="id">Id записи.</param>
       360. /// <returns>Экземпляр типа записи.</returns>
       361. public Post GetPostById(Int32 id)
       362. {
       363. return (from post in Posts.ToList() where post.Id == id select post).FirstOrDefault();
       364. }
       366. /// <summary>
       367. /// Получаем записи пользователя-продавца по имени.
       368. /// </summary>
       369. /// <param name="sellerName">Имя пользователя-продавца</param>
       370. /// <returns>Список записей.</returns>
       371. public List<Post> GetPostsByCreatorName(String sellerName)
       372. {
       373. Int32 creatorId = (from seller in Sellers.ToList() where seller.Username == sellerName select seller.Id)
       374. .FirstOrDefault();
       375. return (from post in Posts.ToList() where post.CreatorId == creatorId select post).ToList();
       376. }
       377. /// <summary>
       378. /// Получает новые записи пользователя-покупателя по его id.
       379. /// </summary>
       380. /// <param name="buyerId">Id пользователя-покупателя.</param>
       381. /// <returns>Список новых записей.</returns>
       382. public List<BuyerPostView> GetBuyerNewPosts(Int32 buyerId)
       383. {
       384. Buyer buyer = GetBuyer(buyerId);
       385. if(buyer.NewPosts == null)
       386. buyer.NewPosts = new List<Int32>();
       387. List<BuyerPostView> newPosts = new List<BuyerPostView>();
       388. foreach (Int32 postId in buyer.NewPosts)
       389. {
       390. Post post = GetPostById(postId);
       391. Seller seller = GetSeller(post.CreatorId);
       392. newPosts.Insert(0, new BuyerPostView(post, seller));
       393. }
       394. return newPosts;
       395. }
       396. /// <summary>
       397. /// Добавляет лайк пользователя-покупателя к записи.
       398. /// </summary>
       399. /// <param name="likerId">Id пользователя-покупателя.</param>
       400. /// <param name="postId">Id записи.</param>
       401. public void AddLikeToPost(Int32 likerId, Int32 postId)
       402. {
       403. Post post = GetPostById(postId);
       404. Buyer liker = GetBuyer(likerId);
       405. if (liker == null)
       406. return;
       407. if(post.Likers == null)
       408. post.Likers = new List<Int32>();
       409. if(liker.LikedPosts == null)
       410. liker.LikedPosts = new List<Int32>();
       411. if(!liker.LikedPosts.Contains(postId))
       412. liker.LikedPosts.Add(postId);
       413. if(!post.Likers.Contains(likerId))
       414. post.Likers.Add(likerId);
       415. SaveChangesAsync();
       416. }
       417. /// <summary>
       418. /// Удаляет лайк пользователя-покупателя к записи.
       419. /// </summary>
       420. /// <param name="likerId">Id пользователя-покупателя.</param>
       421. /// <param name="postId">Id записи.</param>
       422. public void RemoveLikeFromPost(Int32 likerId, Int32 postId)
       423. {
       424. Post post = GetPostById(postId);
       425. Buyer liker = GetBuyer(likerId);
       426. if (liker == null)
       427. return;
       428. if(post.Likers == null)
       429. post.Likers = new List<Int32>();
       430. if(liker.LikedPosts == null)
       431. liker.LikedPosts = new List<Int32>();
       432. liker.LikedPosts.Remove(postId);
       433. post.Likers.Remove(likerId);
       434. SaveChangesAsync();
       435. }
       436. /// <summary>
       437. /// Получает записи, понравившиеся пользователю записи.
       438. /// </summary>
       439. /// <param name="buyerId">Id пользователя.</param>
       440. /// <returns>Список записей.</returns>
       441. public List<BuyerPostView> GetBuyerLikedPosts(Int32 buyerId)
       442. {
       443. Buyer buyer = GetBuyer(buyerId);
       444. if (buyer == null)
       445. return null;
       446. if(buyer.LikedPosts == null)
       447. buyer.LikedPosts = new List<Int32>();
       448. List<BuyerPostView> posts = new List<BuyerPostView>();
       449. foreach (Int32 postId in buyer.LikedPosts)
       450. {
       451. Post post = GetPostById(postId);
       452. Seller seller = GetSeller(post.CreatorId);
       453. posts.Add(new BuyerPostView(post, seller));
       454. }
       455. return posts;
       456. }
       457. /// <summary>
       458. /// Создает новый заказ и добавляет его в базу данных.
       459. /// </summary>
       460. /// <param name="buyerId">Id пользователя-покупателч, который создает заказ.</param>
       461. /// <param name="postId">Id записи, по которой создают заказ.</param>
       462. public void AddOrder(Int32 buyerId, Int32 postId)
       463. {
       464. Buyer buyer = GetBuyer(buyerId);
       465. Post post = GetPostById(postId);
       466. if (buyer == null || post == null)
       467. return;
       468. Seller seller = GetSeller(post.CreatorId);
       469. if (seller == null)
       470. return;
       471. Order newOrder = new Order
       472. {
       473. PostId = post.Id,
       474. BuyerId = buyer.Id
       475. };
       476. Orders.Add(newOrder);
       477. SaveChanges();
       478. if(buyer.Orders == null)
       479. buyer.Orders = new List<Int32>();
       480. buyer.Orders.Add(newOrder.Id);
       481. if(seller.Orders == null)
       482. seller.Orders = new List<Int32>();
       483. seller.Orders.Add(newOrder.Id);
       484. SaveChangesAsync();
       485. }
       486. /// <summary>
       487. /// Получает заказы пользователя-покупателя.
       488. /// </summary>
       489. /// <param name="buyerId">Id пользователя-покупателя.</param>
       490. /// <returns>Список заказов.</returns>
       491. public List<BuyerOrderView> GetBuyerOrders(Int32 buyerId)
       492. {
       493. List<Order> orders = (from order in Orders.ToList() where order.BuyerId == buyerId select order).ToList();
       494. List<BuyerOrderView> views = new List<BuyerOrderView>();
       495. foreach (Order order in orders)
       496. {
       497. Post post = GetPostById(order.PostId);
       498. Seller seller = GetSeller(post.CreatorId);
       499. views.Add(new BuyerOrderView(order, post, seller));
       500. }
       501. return views;
       502. }
       503. /// <summary>
       504. /// Получает заказы пользователя-продавца.
       505. /// </summary>
       506. /// <param name="sellerId">Id пользователя-продавца.</param>
       507. /// <returns>Список заказов.</returns>
       508. public List<SellerOrderView> GetSellerOrders(Int32 sellerId)
       509. {
       510. Seller seller = GetSeller(sellerId);
       511. List<Order> orders = (from order in Orders.ToList() where seller.Orders.Contains(order.Id) select order)
       512. .ToList();
       513. List<SellerOrderView> views = new List<SellerOrderView>();
       514. foreach (Order order in orders)
       515. {
       516. Buyer buyer = GetBuyer(order.BuyerId);
       517. Post post = GetPostById(order.PostId);
       518. views.Add(new SellerOrderView(order, post, buyer));
       519. }
       520. return views;
       521. }
       522. /// <summary>
       523. /// Получает заказ по Id.
       524. /// </summary>
       525. /// <param name="orderId">Id заказа.</param>
       526. /// <returns>Экземпляр типа заказа.</returns>
       527. private Order GetOrder(Int32 orderId)
       528. {
       529. return (from order in Orders.ToList() where order.Id == orderId select order).FirstOrDefault();
       530. }
       532. /// <summary>
       533. /// Устанавливает одобренный статус заказа.
       534. /// </summary>
       535. /// <param name="orderId">Id заказа.</param>
       536. public void SetOrderSuccess(Int32 orderId)
       537. {
       538. Order current = GetOrder(orderId);
       539. if (current == null)
       540. return;
       541. current.IsSucceded = true;
       542. current.IsCanceled = false;
       543. SaveChangesAsync();
       544. }
       545. /// <summary>
       546. /// Устанавливает отклоненный статус заказа.
       547. /// </summary>
       548. /// <param name="orderId">Id заказа.</param>
       549. public void SetOrderCanceled(Int32 orderId)
       550. {
       551. Order current = GetOrder(orderId);
       552. if (current == null)
       553. return;
       554. current.IsSucceded = false;
       555. current.IsCanceled = true;
       556. SaveChangesAsync();
       557. }
       558. /// <summary>
       559. /// Удаляет заказ.
       560. /// </summary>
       561. /// <param name="orderId">Id заказа.</param>
       562. public void DeleteOrder(Int32 orderId)
       563. {
       564. Order current = GetOrder(orderId);
       565. if (current == null)
       566. return;
       567. Orders.Remove(current);
       568. SaveChangesAsync();
       569. }
       570. /// <summary>
       571. /// Изменяет информацию о пользователе-покупателе.
       572. /// </summary>
       573. /// <param name="buyerId">Id пользователя-покупателя.</param>
       574. /// <param name="info">Новая информация.</param>
       575. public void SetBuyerInfo(Int32 buyerId, String info)
       576. {
       577. Buyer buyer = GetBuyer(buyerId);
       578. if (buyer == null)
       579. return;
       580. buyer.Info = info;
       581. SaveChangesAsync();
       582. }
       583. /// <summary>
       584. /// Изменяет информацию о пользователе-продавце.
       585. /// </summary>
       586. /// <param name="sellerId">Id пользователя-покупателя.</param>
       587. /// <param name="info">Новая информация.</param>
       588. public void SetSellerInfo(Int32 sellerId, String info)
       589. {
       590. Seller seller = GetSeller(sellerId);
       591. if (seller == null)
       592. return;
       593. seller.Info = info;
       594. SaveChangesAsync();
       595. }
       596. }
       597. }
2. Список литературы
   1. ГОСТ 19.401-78. ЕСПД. Текст программы. Требования к содержанию и оформлению. – М.: ИПК Издательство стандартов, 2001.
   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 |
| Изменен  ных | Заменен  ных | новых | Аннулиро  ванных |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |