ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

СОГЛАСОВАНО
Канд. пед наук, доцент департамента программной инженерии факультета компьютерных наук

_____ С. А. Виденин

«_13_» _____мая_____2023 г.

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

_____ В. В. Шилов

«___» _____2023 г.

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

WEB-ПРИЛОЖЕНИЕ «Tamak´»

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.09.09-01 12 01-1-ЛУ

Исполнитель:
Студентка группы БПИ217
_____ / Омирбекова Д.К. /
«_13_» _____мая_____ 2023 г.

Москва 2023

УТВЕРЖДЕН

RU.17701729.09.09-01 12 01-1-ЛУ

WEB-ПРИЛОЖЕНИЕ «Tamak´»

Текст программы

RU.17701729.09.09-01 12 01-1

Листов 96

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

Москва 2023

АННОТАЦИЯ

В настоящем документе «Текст программы» приведен текст «Tamak’». Программа разработана на языке C#.

Среда разработки – Microsoft Visual Studio 2022.

«Tamak´» - Web-приложение предназначено для преждевременного заказа еды в предприятиях общественного питания, располагающихся в корпусах НИУ ВШЭ. Основными клиентами сервиса будут являться студенты и сотрудники НИУ ВШЭ.

Настоящий документ разработан в соответствии с требованиями:

1. ГОСТ 19.101–77 Виды программ и программных документов.
2. ГОСТ 19.102-77 Стадии разработки.
3. ГОСТ 19.103-77 Обозначения программ и программных документов.
4. ГОСТ 19.104-78 Основные надписи.
5. ГОСТ 19.105-78 Общие требования к программным документам.
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом.
7. ГОСТ 19.401-79 Текст программы. Требования к содержанию и оформлению.

Изменения к данному Пояснительной записке оформляются согласно:

8. ГОСТ 19.603-78
9. ГОСТ 19.604-78

СОДЕРЖАНИЕ

1. ТЕКСТ ПРОГРАММЫ

1.1. Контроллеры:

1.1.1. AccountController

```
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;
using Tamak.Service.Interfaces;
using Tamak.ViewModels;

namespace Tamak.Controllers
{
   public class AccountController : Controller
   {
      private readonly IAccountService _accountService;

      public AccountController(IAccountService accountService)
      {
         _accountService = accountService;
      }

      [HttpGet]
      public IActionResult Register() => View();

      [HttpPost]
      public async Task<IActionResult> Register(RegisterViewModel model)
      {
         if (ModelState.IsValid)
```

```
    {
        var response = await _accountService.Register(model);
        if (response.StatusCode == Data.Enum.StatusCode.Success)
        {
            await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,
                new ClaimsPrincipal(response.Data));

            return RedirectToAction("Index", "Home");
        }
        ModelState.AddModelError("", response.Description);
    }
    return View(model);
}

[HttpGet]
public IActionResult Login() => View();

[HttpPost]
public async Task<IActionResult> Login(LoginViewModel model)
{
    if (ModelState.IsValid)
    {
        var response = await _accountService.Login(model);
        if (response.StatusCode == Data.Enum.StatusCode.Success)
        {
            await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,
                new ClaimsPrincipal(response.Data));
```

```
                return RedirectToAction("Index", "Home");
            }
            ModelState.AddModelError("", response.Description);
        }
        return View(model);
    }

    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Logout()
    {
        await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
        return RedirectToAction("Index", "Home");
    }
  }
}
```

1.1.2. **AssortimentController**

```
using Microsoft.AspNetCore.Mvc;
using Tamak.Service.Interfaces;

namespace Tamak.Controllers
{
  public class AssortimentController : Controller
  {
    private readonly IAssortimentService _assortimentService;

    public AssortimentController(IAssortimentService basketService)
    {
      _assortimentService = basketService;
```

```
        }

        public async Task<IActionResult> Detail()
        {
            var response = await _assortimentService.GetItems(User.Identity.Name);
            if (response.StatusCode == Data.Enum.StatusCode.Success)
            {
                return View(response.Data.ToList());
            }
            return RedirectToAction("Index", "Home");
        }

        [HttpGet]
        public async Task<IActionResult> GetItem(long id)
        {
            var response = await _assortimentService.GetItem(User.Identity.Name, id);
            if (response.StatusCode == Data.Enum.StatusCode.Success)
            {
                return PartialView(response.Data);
            }
            return RedirectToAction("Index", "Home");
        }
    }
}
```

1.1.3. **HomeController**

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Reflection.Metadata.Ecma335;
```

```
using Tamak.Data.Interfaces;
using Tamak.Service.Implementations;
using Tamak.Service.Interfaces;
using Tamak.ViewModels;

namespace Tamak.Controllers
{
    public class HomeController : Controller
    {
        private readonly IProductService _productService;
        private readonly IAssortimentService _assortimentService;
        private readonly IUserService _userService;

        public HomeController(IProductService productService, IAssortimentService
assortimentService, IUserService userService)
        {
            _productService = productService;
            _assortimentService = assortimentService;
            _userService = userService;
        }

        [HttpGet]
        public async Task<IActionResult> IndexAsync()
        {
            var response = await _assortimentService.GetItems(User.Identity.Name);

            var response2 = await _userService.GetUsers();
```

```
    //var response3 = await _assortimentService.GetItems(User.Identity.Name);

    if (response.StatusCode == Data.Enum.StatusCode.Success)
    {
        HomeViewModel obj = new HomeViewModel();
        obj.allProducts = response.Data;
        obj.allUsers = response2.Data;
        return View(obj);

    }
    return RedirectToAction("Login", "Account");
}

[HttpGet]
public async Task<ActionResult> GetProduct(int id)
{
    var response = await _productService.GetProduct(id);
    if (response.StatusCode == Data.Enum.StatusCode.Success)
    {
        HomeViewModel obj = new HomeViewModel();
        obj.allProducts = (IEnumerable<Data.Models.Product>)response.Data;
        return View(obj);
    }
    return RedirectToAction("Error");
}

[Authorize(Roles = "Admin")]
public async Task<ActionResult> Delete(int id)
```

```
{
    var response = await _productService.DeleteProduct(id);
    if (response.StatusCode == Data.Enum.StatusCode.Success)
    {
        return RedirectToAction("GetProducts");
    }
    return RedirectToAction("Error");
}

[HttpGet]
public async Task<IActionResult> Save(int id)
{
    if (id == 0)
    {
        return View();
    }

    var response = await _productService.GetProduct(id);
    if (response.StatusCode == Data.Enum.StatusCode.Success)
    {
        HomeViewModel obj = new HomeViewModel();
        obj.allProducts = (IEnumerable<Data.Models.Product>)response.Data;
        return View(obj);
    }

    return RedirectToAction("Error");
}
```

```
[HttpPost]
public async Task<IActionResult> Save(ProductViewModel model)
{
    ModelState.Remove("Avatar");
    ModelState.Remove("Img");
    ModelState.Remove("Category");

    if (ModelState.IsValid)
    {
        var response = await _productService.Save(model);
        if (response.StatusCode == Data.Enum.StatusCode.Success)
        {
            return Json(new { data = response.Description });
        }
    }
    return RedirectToAction("GetProducts");
}

[HttpPost]
public async Task<IActionResult> ChangeAvaliable(ProductViewModel model)
{
    ModelState.Remove("Avatar");
    ModelState.Remove("Img");
    ModelState.Remove("Category");
    ModelState.Remove("Name");
    ModelState.Remove("Description");
    ModelState.Remove("Category");
    if (ModelState.IsValid)
```

```
        {
            var response = await _productService.ChangeAvaliable(model);
            if (response.StatusCode == Data.Enum.StatusCode.Success)
            {
                return Json(new { data = response.Description });
            }
        }
        return RedirectToAction("GetProducts");
    }

    [HttpPost]
    public async Task<IActionResult> Add()
    {
        var response = await _assortimentService.Add(User.Identity.Name);
        if (response.StatusCode == Data.Enum.StatusCode.Success)
        {
            return RedirectToAction("Index");
        }
        return RedirectToAction("GetProducts");
    }
  }
}
```

1.1.4. **ProductController**

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Reflection.Metadata.Ecma335;
using Tamak.Data.Interfaces;
using Tamak.Data.Models;
```

```
using Tamak.Service.Interfaces;
using Tamak.ViewModels;

namespace Tamak.Controllers
{
    public class ProductController : Controller
    {
        private readonly IProductService _productService;

        public ProductController(IProductService productService)
        {
            _productService = productService;
        }

        [HttpGet]
        public IActionResult GetProducts()
        {
            var response = _productService.GetProducts();
            if (response.StatusCode == Data.Enum.StatusCode.Success)
            {
                HomeViewModel obj = new HomeViewModel();
                obj.allProducts = response.Data;
                return View(obj);
            }
            return RedirectToAction("Error", $"{response.Description}");
        }

        [HttpGet]
```

```
public async Task<ActionResult> GetProduct(int id)
{
    var response = await _productService.GetProduct(id);
    if (response.StatusCode == Data.Enum.StatusCode.Success)
    {
        HomeViewModel obj = new HomeViewModel();
        obj.allProducts = (IEnumerable<Data.Models.Product>)response.Data;
        return View(obj);
    }
    return RedirectToAction("Error");
}

[Authorize(Roles = "Admin")]
public async Task<ActionResult> Delete(int id)
{
    var response = await _productService.DeleteProduct(id);
    if (response.StatusCode == Data.Enum.StatusCode.Success)
    {
        return RedirectToAction("GetProducts");
    }
    return RedirectToAction("Error", $"{response.Description}");
}

public IActionResult Compare() => PartialView();

[HttpGet]
public async Task<IActionResult> Save(int id)
{
```

```
    if (id == 0)
    {
        return PartialView();
    }

    var response = await _productService.GetProduct(id);
    if (response.StatusCode == Data.Enum.StatusCode.Success)
    {
        HomeViewModel obj = new HomeViewModel();
        obj.allProducts = (IEnumerable<Data.Models.Product>)response.Data;
        return View(obj);
    }
    ModelState.AddModelError("", response.Description);
    return PartialView();
}

[HttpPost]
public async Task<IActionResult> Save(ProductViewModel model)
{
    if (ModelState.IsValid)
    {
        _productService.Save(model);
    }
    return RedirectToAction("GetProducts");
}
```

```
        [HttpPost]
        public JsonResult GetCategories()
        {
            var categories = _productService.GetCategories();
            return Json(categories.Data);
        }

        [HttpPost]
        public async Task<IActionResult> GetProduct(string term)
        {
            var response = await _productService.GetProduct(term);
            return Json(response.Data);
        }

        [HttpGet]
        public async Task<IActionResult> GetProduct(int id, bool isJson)
        {
            var response = await _productService.GetProduct(id);
            if (isJson)
            {
                return Json(response.Data);
            }
            return PartialView("GetCar", response.Data);
        }

    }
}
```

1.1.5. **UserController**

```
using Automarket.Service.Implementations;
using Microsoft.AspNetCore.Mvc;
using Tamak.Service.Interfaces;
using Tamak.ViewModels;

namespace Tamak.Controllers
{
    public class UserController : Controller
    {
        private readonly IUserService _userService;

        public UserController(IUserService userService)
        {
            _userService = userService;
        }

        public async Task<IActionResult> GetUsers()
        {
            var response = await _userService.GetUsers();
            if (response.StatusCode == Data.Enum.StatusCode.Success)
            {
                return View(response.Data);
            }
            return RedirectToAction("Index", "Home");
        }

        [HttpGet]
        public async Task<ActionResult> GetUser()
```

```
{
    var userEmail = User.Identity.Name;
    var response = await _userService.GetUser(userEmail);
    if (response.StatusCode == Data.Enum.StatusCode.Success)
    {
        return View(response.Data);
    }
    return RedirectToAction("Error");
}

[HttpPost]
public async Task<IActionResult> Save(UserViewModel model)
{
    ModelState.Remove("Role");
    ModelState.Remove("Id");
    if (User.IsInRole("User"))
    {
        ModelState.Remove("Campus");
    } else if (User.IsInRole("Admin") && model.City != "Moscow")
    {
        model.Campus = "None";
    }
    if (ModelState.IsValid)
    {
        var response = await _userService.Save(model);
        if (response.StatusCode == Data.Enum.StatusCode.Success)
        {
            return Json(new { data = response.Description });
```

```
            }
        }
        return RedirectToAction("GetProducts");
    }
  }
}
```

1.2. Перечисления

1.2.1. Campus

```
using System.ComponentModel.DataAnnotations;

namespace Tamak.Data.Enum
{
  public enum Campus
  {
    [Display(Name = "Покровский бульвар")]
    Pokrovka = 0,
    [Display(Name = "Шаболовка")]
    Shabolovka = 1,
    [Display(Name = "Строгино")]
    Strogino = 2,

    [Display(Name = "Ничего")]
    None = 3
  }
}
```

1.2.2. Category

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

```
namespace Tamak.Data.Enum
{
   public enum Category
   {
      [Display(Name = "Кофе")]
      Coffee = 0,
      [Display(Name = "Какао")]
      Kakao = 1,
   }
}
```

1.2.3. City

```
using System.ComponentModel.DataAnnotations;

namespace Tamak.Data.Enum
{
   public enum City
   {
      [Display(Name = "Москва")]
      Moscow = 0,
      [Display(Name = "Санкт-Петербург")]
      Spb = 1,
      [Display(Name = "Нижний Новгород")]
      Nn = 2,
      [Display(Name = "Пермь")]
      Perm = 3,
   }
}
```

1.2.4. Role

```
using System.ComponentModel.DataAnnotations;


namespace Tamak.Data.Enum
{
   public enum Role
   {
      [Display(Name = "Пользователь")]
      User = 0,
      [Display(Name = "Модератор")]
      Moderator = 1,
      [Display(Name = "Админ")]
      Admin = 2,
   }
}
```

1.2.5. StatusCode

```
namespace Tamak.Data.Enum
{
   public enum StatusCode
   {
      UserNotFound = 0,
      ProductNotFound = 10,
      Success = 200,
      InternalServerError = 500,
   }
}
```

1.3. Расширения

1.3.1. EnumExtension

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

```
using System.Reflection;


namespace Tamak.Data.Extensions
{
   public static class EnumExtension
   {
      public static string GetDisplayName(this System.Enum enumValue)
      {
         return enumValue.GetType()
            .GetMember(enumValue.ToString())
            .First()
            .GetCustomAttribute<DisplayAttribute>()
            ?.GetName() ?? "Неопределенный";
      }
   }
}
```

1.3.2. StringExtension

```
using System.Text;


namespace Tamak.Data.Extensions
{
   public static class StringExtension
   {
      public static string Join(this List<string> words)
      {
         var sb = new StringBuilder();


         for (int i = 0; i < words.Count; i++)
```

```
            {
                sb.Append($"{i + 1}: {words[i]} ");
            }

            return sb.ToString();
        }
    }
}
```

1.4. Hash-функция

```
using System.Security.Cryptography;
using System.Text;

namespace Tamak.Data.Helpers
{
    public static class HashPasswordHelper
    {
        public static string HashPassowrd(string password)
        {
            using (var sha256 = SHA256.Create())
            {
                var hashedBytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(password));
                var hash = BitConverter.ToString(hashedBytes).Replace("-", "").ToLower();

                return hash;
            }
        }
    }
}
```

1.5. Интерфейсы

1.5.1. IBaseRepository

```
using Tamak.Data.Models;


namespace Tamak.Data.Interfaces
{
   public interface IBaseRepository<T>
   {
      Task Create(T entity);


      IQueryable<T> GetAll();


      Task Delete(T entity);


      Task<T> Update(T entity);
   }
}
```

1.6. Модели

1.6.1. Assortiment

```
namespace Tamak.Data.Models
{
   public class Assortiment
   {
      public long Id { get; set; }
      public User User { get; set; }
      public long UserId { get; set; }
      public List<Product> Products { get; set; }
   }
```

```
}
```

1.6.2. Product

```
using Tamak.Data.Enum;

namespace Tamak.Data.Models
{
    public class Product
    {
        public long Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public byte[]? Img { get; set; }
        public decimal Price { get; set; }
        public bool Available { get; set; }
        public int CategoryId { get; set; }
        public Category Category { set; get; }
        public byte[]? Avatar { get; set; }

        public long AssortimentId { set; get; }
        public virtual Assortiment Assortiment { set; get; }

    }
}
```

1.6.3. User

```
using System.Data;
using Tamak.Data.Enum;

namespace Tamak.Data.Models
```

```
{
    public class User
    {
        public long Id { get; set; }

        public string Email { get; set; }

        public string Password { get; set; }

        public string Name { get; set; }

        public City City { get; set; }

        public Campus Campus { get; set; }

        public Role Role { get; set; }

        public Assortiment Assortiment { set; get; }

    }
}
```

1.7. Репозитории

1.7.1. AssortimentRepository

```
using Microsoft.EntityFrameworkCore;
using Tamak.Data.Interfaces;
using Tamak.Data.Models;

namespace Tamak.Data.Repository
```

```
{
    public class AssortimentRepository : IBaseRepository<Assortiment>
    {
        private readonly AppDBContent _db;

        public AssortimentRepository(AppDBContent dbContext)
        {
            _db = dbContext;
        }

        public async Task Create(Assortiment entity)
        {
            await _db.Assortements.AddAsync(entity);
            await _db.SaveChangesAsync();
        }

        public IQueryable<Assortiment> GetAll()
        {
            return _db.Assortements;
        }

        public async Task Delete(Assortiment entity)
        {
            _db.Assortements.Remove(entity);
            await _db.SaveChangesAsync();
        }

        public async Task<Assortiment> Update(Assortiment entity)
```

```
        {
            _db.Assortements.Update(entity);
            await _db.SaveChangesAsync();

            return entity;
        }

    }
}
```

1.7.2. ProductRepository

```
using Microsoft.EntityFrameworkCore;
using Tamak.Data.Interfaces;
using Tamak.Data.Models;

namespace Tamak.Data.Repository
{
    public class ProductRepository : IBaseRepository<Product>
    {
        private readonly AppDBContent appDBContent;

        public ProductRepository(AppDBContent appDBContent)
        {
            this.appDBContent = appDBContent;
        }

        public async Task Create(Product entity)
        {
            await appDBContent.Products.AddAsync(entity);
```

```
            await appDBContent.SaveChangesAsync();
        }

        public async Task Delete(Product entity)
        {
            appDBContent.Products.Remove(entity);
            await appDBContent.SaveChangesAsync();
        }

        public IQueryable<Product> GetAll()
        {
            return appDBContent.Products;
        }

        public async Task<Product> Update(Product entity)
        {
            appDBContent.Products.Update(entity);
            await appDBContent.SaveChangesAsync();
            return entity;
        }
    }
}
```

1.7.3. UserRepository

```
using Tamak.Data.Interfaces;
using Tamak.Data.Models;

namespace Tamak.Data.Repository
{
```

```
public class UserRepository : IBaseRepository<User>
{
    private readonly AppDBContent _database;

    public UserRepository(AppDBContent db)
    {
        _database = db;
    }

    public IQueryable<User> GetAll()
    {
        return _database.Users;
    }

    public async Task Delete(User entity)
    {
        _database.Users.Remove(entity);
        await _database.SaveChangesAsync();
    }

    public async Task Create(User entity)
    {
        await _database.Users.AddAsync(entity);
        await _database.SaveChangesAsync();
    }

    public async Task<User> Update(User entity)
    {
```

```
            _database.Users.Update(entity);
            await _database.SaveChangesAsync();

            return entity;
        }
    }
}
```

1.8. BaseResponse

```
using Microsoft.AspNetCore.Http;
using System.Collections;
using Tamak.Data.Enum;

namespace Tamak.Data.Response
{
    public class BaseResponse<T> : IBaseResponse<T>
    {
        public string Description { get; set; }

        public StatusCode StatusCode { get; set; }

        public T Data { get; set; }
    }

    public interface IBaseResponse<T>
    {
        public string Description { get; set; }
        public T Data { get; set;  }
        public StatusCode StatusCode { get; set; }
```

```
  }
}
```

1.9. AppDBContext

```
using Microsoft.EntityFrameworkCore;
using Tamak.Data.Enum;
using Tamak.Data.Helpers;
using Tamak.Data.Models;

namespace Tamak.Data
{
   public class AppDBContent : DbContext
   {
      public AppDBContent(DbContextOptions<AppDBContent> options) : base(options) {
         Database.EnsureCreated();
      }

      public DbSet<Product> Products { get; set; }
      public DbSet<User> Users { get; set; }
      public DbSet<Assortiment> Assortements { get; set; }


      protected override void OnModelCreating(ModelBuilder modelBuilder)
      {

         modelBuilder.Entity<User>(builder =>
         {
            builder.HasData(new User()
            {
```

```
            Id = 1,
            Name = "Test1",
            Email = "Test@edu.hse.ru",
            Password = HashPasswordHelper.HashPassowrd("Test1"),
            Role = Role.Admin,
            City = City.Moscow,
            Campus = Campus.Pokrovka,
        });
        builder.ToTable("User").HasKey(x => x.Id);
        builder.Property(x => x.Id).ValueGeneratedOnAdd();
        builder.Property(x => x.Password).IsRequired();
        builder.Property(x => x.Name).HasMaxLength(100).IsRequired();
        builder.Property(x => x.Email).HasMaxLength(100).IsRequired();

        builder.HasOne(x => x.Assortiment).WithOne(x => x.User).HasPrincipalKey<User>(x =>
x.Id).OnDelete(DeleteBehavior.Cascade);
    });

    modelBuilder.Entity<Assortiment>(builder =>
    {
        builder.ToTable("Assortiments").HasKey(x => x.Id);

        builder.HasData(new Assortiment()
        {
            Id = 1,
            UserId = 1
        });
    });
```

```
        modelBuilder.Entity<Product>(builder =>
        {
            builder.HasData(new Product()
            {
                Id = 1,
                Name = "Латте",
                Description = "Свежий кофе с молочным оттенком",
                Price = 200,
                Available = true,
                Category = Category.Coffee,
                AssortimentId = 1,
            });
            builder.ToTable("Products").HasKey(x => x.Id);
            builder.Property(x => x.Id).ValueGeneratedOnAdd();
            builder.HasOne(r => r.Assortiment).WithMany(t => t.Products).HasForeignKey(x =>
x.AssortimentId);
        });
    }
  }
}
```

1.10. Представления сервисов

1.10.1. AccountService

```
using Microsoft.EntityFrameworkCore;
using System.Security.Claims;
using Tamak.Data.Enum;
using Tamak.Data.Helpers;
using Tamak.Data.Interfaces;
```

```
using Tamak.Data.Models;
using Tamak.Data.Response;
using Tamak.Service.Interfaces;
using Tamak.ViewModels;

namespace Tamak.Service.Implementations
{
   public class AccountService : IAccountService
   {
      private readonly IBaseRepository<User> _userRepository;
      private readonly ILogger<AccountService> _logger;

      public AccountService(IBaseRepository<User> userRepository,
         ILogger<AccountService> logger)
      {
         _userRepository = userRepository;
         _logger = logger;
      }

      public async Task<BaseResponse<ClaimsIdentity>> Register(RegisterViewModel model)
      {
         try
         {
            var user = await _userRepository.GetAll().FirstOrDefaultAsync(x => x.Email ==
model.Email);
            if (user != null)
            {
               return new BaseResponse<ClaimsIdentity>()
```

```
            {
                Description = "Пользователь с такой почтой уже существует",
            };
        }

        user = new User()
        {
            Name = model.Name,
            Email = model.Email,
            Role = (Role)Enum.Parse(typeof(Role), model.Role),
            City = (City)Enum.Parse(typeof(City), model.City),
            Campus = (Campus)Enum.Parse(typeof(Campus), model.Campus),
            Password = HashPasswordHelper.HashPassowrd(model.Password),
        };

        await _userRepository.Create(user);
        var result = Authenticate(user);

        return new BaseResponse<ClaimsIdentity>()
        {
            Data = result,
            Description = "Объект добавился",
            StatusCode = StatusCode.Success
        };
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"[Register]: {ex.Message}");
```

```
            return new BaseResponse<ClaimsIdentity>()
            {
                Description = ex.Message,
                StatusCode = StatusCode.InternalServerError
            };
        }
    }

    public async Task<BaseResponse<ClaimsIdentity>> Login(LoginViewModel model)
    {
        try
        {
            var user = await _userRepository.GetAll().FirstOrDefaultAsync(x => x.Email ==
model.Email);
            if (user == null)
            {
                return new BaseResponse<ClaimsIdentity>()
                {
                    Description = "Пользователь не найден"
                };
            }

            if (user.Password != HashPasswordHelper.HashPassowrd(model.Password))
            {
                return new BaseResponse<ClaimsIdentity>()
                {
                    Description = "Неверный пароль или логин"
                };
```

```
        }
        var result = Authenticate(user);

        return new BaseResponse<ClaimsIdentity>()
        {
            Data = result,
            StatusCode = StatusCode.Success
        };
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"[Login]: {ex.Message}");
        return new BaseResponse<ClaimsIdentity>()
        {
            Description = ex.Message,
            StatusCode = StatusCode.InternalServerError
        };
    }
}

private ClaimsIdentity Authenticate(User user)
{
    var claims = new List<Claim>
    {
        new Claim(ClaimsIdentity.DefaultNameClaimType, user.Email),
        new Claim(ClaimsIdentity.DefaultRoleClaimType, user.Role.ToString())
    };
    return new ClaimsIdentity(claims, "ApplicationCookie",
```

```
            ClaimsIdentity.DefaultNameClaimType, ClaimsIdentity.DefaultRoleClaimType);
        }
    }
}
```

1.10.2. AssortimentService

```
using Microsoft.EntityFrameworkCore;
using Tamak.Data.Enum;
using Tamak.Data.Interfaces;
using Tamak.Data.Models;
using Tamak.Data.Response;
using Tamak.Service.Interfaces;
using Tamak.ViewModels;

namespace Tamak.Service.Implementations
{
    public class AssortimentService : IAssortimentService
    {
        private readonly IBaseRepository<User> _userRepository;
        private readonly IBaseRepository<Product> _productRepository;

        public AssortimentService(IBaseRepository<User> userRepository,
IBaseRepository<Product> productRepository)
        {
            _userRepository = userRepository;
            _productRepository = productRepository;
        }

        public async Task<IBaseResponse<IEnumerable<Product>>> GetItems(string userName)
```

```
{
    try
    {
        var user = await _userRepository.GetAll()
            .Include(x => x.Assortiment)
            .ThenInclude(x => x.Products)
            .FirstOrDefaultAsync(x => x.Email == userName);

        if (user == null)
        {
            return new BaseResponse<IEnumerable<Product>>()
            {
                Description = "Пользователь не найден",
                StatusCode = StatusCode.UserNotFound
            };
        }
        var products = user.Assortiment?.Products;

        return new BaseResponse<IEnumerable<Product>>()
        {
            Data = products,
            StatusCode = StatusCode.Success
        };
    }
    catch (Exception ex)
    {
        return new BaseResponse<IEnumerable<Product>>()
        {
```

```
            Description = ex.Message,
            StatusCode = StatusCode.InternalServerError
        };
    }
}

public async Task<IBaseResponse<Product>> GetItem(string userName, long id)
{
    try
    {
        var user = await _userRepository.GetAll()
            .Include(x => x.Assortiment)
            .ThenInclude(x => x.Products)
            .FirstOrDefaultAsync(x => x.Email == userName);

        if (user == null)
        {
            return new BaseResponse<Product>()
            {
                Description = "Пользователь не найден",
                StatusCode = StatusCode.UserNotFound
            };
        }

        var orders = user.Assortiment.Products.Where(x => x.Id == id).ToList();
        if (orders == null || orders.Count == 0)
        {
            return new BaseResponse<Product>()
```

```
        {
            Description = "Заказов нет",
            StatusCode = StatusCode.ProductNotFound
        };
    }

    var response = (from p in orders
                    join c in _productRepository.GetAll() on p.Id equals c.Id
                    select new Product()
                    {
                        Id = p.Id,
                        Name = c.Name,
                        Price = c.Price,
                        Description = c.Description,
                        Available = c.Available,
                    }).FirstOrDefault();

    return new BaseResponse<Product>()
    {
        Data = response,
        StatusCode = StatusCode.Success
    };
}
catch (Exception ex)
{
    return new BaseResponse<Product>()
    {
        Description = ex.Message,
```

```
            StatusCode = StatusCode.InternalServerError
        };
    }
}

public async Task<IBaseResponse<Product>> Add(string userName)
{
    try
    {
        var user = await _userRepository.GetAll()
            .Include(x => x.Assortiment)
            .ThenInclude(x => x.Products)
            .FirstOrDefaultAsync(x => x.Email == userName);

        if (user == null)
        {
            return new BaseResponse<Product>()
            {
                Description = "Пользователь не найден",
                StatusCode = StatusCode.UserNotFound
            };
        }

        var product = new Product()
        {
            Name = "кофе",
            Description = "кофе",
            Price = 0,
```

```
                Available = true,
                AssortimentId = user.Assortiment.Id,
                Assortiment = user.Assortiment,
                Category = 0,
                CategoryId = 0,
            };
            var productRep = _productRepository.Create(product);
            user.Assortiment.Products.Add(product);

            return new BaseResponse<Product>()
            {
                Data = product,
                StatusCode = StatusCode.Success
            };
        }
        catch (Exception ex)
        {
            return new BaseResponse<Product>()
            {
                Description = ex.Message,
                StatusCode = StatusCode.InternalServerError
            };
        }
    }
  }
}
```

1.10.3. ProductService

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Runtime.ConstrainedExecution;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Cors.Infrastructure;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Tamak.Data.Enum;
using Tamak.Data.Extensions;
using Tamak.Data.Interfaces;
using Tamak.Data.Models;
using Tamak.Data.Repository;
using Tamak.Data.Response;
using Tamak.Service.Interfaces;
using Tamak.ViewModels;

namespace Automarket.Service.Implementations
{
    public class ProductService : IProductService
    {
        private readonly IBaseRepository<Product> _productRepository;

        public ProductService(IBaseRepository<Product> productRepository)
        {
            _productRepository = productRepository;
        }

        public BaseResponse<Dictionary<long, string>> GetCategories()
```

```
{
    try
    {
        var types = ((Category[])Enum.GetValues(typeof(Category)))
            .ToDictionary(k => (long)k, t => t.GetDisplayName());

        return new BaseResponse<Dictionary<long, string>>()
        {
            Data = types,
            StatusCode = StatusCode.Success
        };
    }
    catch (Exception ex)
    {
        return new BaseResponse<Dictionary<long, string>>()
        {
            Description = ex.Message,
            StatusCode = StatusCode.InternalServerError
        };
    }
}

public async Task<BaseResponse<Dictionary<long, string>>> GetProduct(string term)
{
    var baseResponse = new BaseResponse<Dictionary<long, string>>();
    try
    {
        var products = await _productRepository.GetAll()
```

```
            .Select(x => new ProductViewModel()
            {
                Id = x.Id,
                Name = x.Name,
                Description = x.Description,
                Price = x.Price,
                Available= x.Available,
                Category = x.Category.GetDisplayName()
            })
            .Where(x => EF.Functions.Like(x.Name, $"%{term}%"))
            .ToDictionaryAsync(x => x.Id, t => t.Name);

        baseResponse.Data = products;
        return baseResponse;
    }
    catch (Exception ex)
    {
        return new BaseResponse<Dictionary<long, string>>()
        {
            Description = ex.Message,
            StatusCode = StatusCode.InternalServerError
        };
    }
}

public async Task<IBaseResponse<ProductViewModel>> GetProduct(long id)
{
    try
```

```
{
    var product = await _productRepository.GetAll().FirstOrDefaultAsync(x => x.Id == id);
    if (product == null)
    {
        return new BaseResponse<ProductViewModel>()
        {
            Description = "Пользователь не найден",
            StatusCode = StatusCode.UserNotFound
        };
    }

    var data = new ProductViewModel()
    {
        Description = product.Description,
        Category = product.Category.GetDisplayName(),
        Img = product.Avatar,
    };

    return new BaseResponse<ProductViewModel>()
    {
        StatusCode = StatusCode.Success,
        Data = data
    };
}
catch (Exception ex)
{
    return new BaseResponse<ProductViewModel>()
    {
```

```
            Description = $"[GetProduct] : {ex.Message}",
            StatusCode = StatusCode.InternalServerError
        };
    }
}

public async Task<IBaseResponse<Product>> Create(long assortimentId, Assortiment assortiment)
{
    try
    {
        var product = new Product()
        {
            Name = "кофе",
            Description = "кофе",
            Price = 0,
            Available = true,
            AssortimentId = assortimentId,
            Assortiment = assortiment,
            Category = 0,
            CategoryId = 0,
        };
        await _productRepository.Create(product);

        return new BaseResponse<Product>()
        {
            StatusCode = StatusCode.Success,
            Data = product
```

```
        };
    }
    catch (Exception ex)
    {
        return new BaseResponse<Product>()
        {
            Description = $"[Create] : {ex.Message}",
            StatusCode = StatusCode.InternalServerError
        };
    }
}

public async Task<IBaseResponse<bool>> DeleteProduct(long id)
{
    try
    {
        var product = await _productRepository.GetAll().FirstOrDefaultAsync(x => x.Id == id);
        if (product == null)
        {
            return new BaseResponse<bool>()
            {
                Description = "User not found",
                StatusCode = StatusCode.UserNotFound,
                Data = false
            };
        }

        await _productRepository.Delete(product);
```

```
            return new BaseResponse<bool>()
            {
                Data = true,
                StatusCode = StatusCode.Success
            };
        }
        catch (Exception ex)
        {
            return new BaseResponse<bool>()
            {
                Description = $"[DeleteProduct] : {ex.Message}",
                StatusCode = StatusCode.InternalServerError
            };
        }
    }

    public async Task<IBaseResponse<Product>> Edit(long id, ProductViewModel model)
    {
        try
        {
            var product = await _productRepository.GetAll().FirstOrDefaultAsync(x => x.Id == id);
            if (product == null)
            {
                return new BaseResponse<Product>()
                {
                    Description = "Product not found",
                    StatusCode = StatusCode.ProductNotFound
```

```
            };
        }

        product.Description = model.Description;
        product.Name = model.Name;
        product.Price = model.Price;
        product.Available = model.Available;

        await _productRepository.Update(product);



        return new BaseResponse<Product>()
        {
            Data = product,
            StatusCode = StatusCode.Success,
        };
        // TypeCar
    }
    catch (Exception ex)
    {
        return new BaseResponse<Product>()
        {
            Description = $"[Edit] : {ex.Message}",
            StatusCode = StatusCode.InternalServerError
        };
    }
}
```

```
public IBaseResponse<List<Product>> GetProducts()
{
    try
    {
        var products = _productRepository.GetAll().ToList();
        if (!products.Any())
        {
            return new BaseResponse<List<Product>>()
            {
                Description = "Найдено 0 элементов",
                StatusCode = StatusCode.Success
            };
        }

        return new BaseResponse<List<Product>>()
        {
            Data = products,
            StatusCode = StatusCode.Success
        };
    }
    catch (Exception ex)
    {
        return new BaseResponse<List<Product>>()
        {
            Description = $"[GetProducts] : {ex.Message}",
            StatusCode = StatusCode.InternalServerError
        };
    }
```

```
}

public async Task<BaseResponse<Product>> Save(ProductViewModel model)
{
    try
    {
        var product = await _productRepository.GetAll()
            .FirstOrDefaultAsync(x => x.Id == model.Id);

        product.Name = model.Name;
        /*product.Category = (Category)Enum.Parse(typeof(Category), model.Category);*/
        product.Description = model.Description;
        product.Price = model.Price;
        product.Available = model.Available;

        await _productRepository.Update(product);

        return new BaseResponse<Product>()
        {
            Data = product,
            Description = "Данные обновлены",
            StatusCode = StatusCode.Success
        };
    }
    catch (Exception ex)
    {
        return new BaseResponse<Product>()
        {
```

```
            StatusCode = StatusCode.InternalServerError,
            Description = $"Внутренняя ошибка: {ex.Message}"
        };
    }
}

public async Task<BaseResponse<Product>> ChangeAvaliable(ProductViewModel model)
{
    try
    {
        var product = await _productRepository.GetAll()
            .FirstOrDefaultAsync(x => x.Id == model.Id);

        product.Available = !model.Available;

        await _productRepository.Update(product);

        return new BaseResponse<Product>()
        {
            Data = product,
            Description = "Данные обновлены",
            StatusCode = StatusCode.Success
        };
    }
    catch (Exception ex)
    {
        return new BaseResponse<Product>()
        {
```

```
                StatusCode = StatusCode.InternalServerError,
                Description = $"Внутренняя ошибка: {ex.Message}"
            };
        }
    }
}
```

1.10.4. UserService

```
using Microsoft.EntityFrameworkCore;
using Tamak.Data.Enum;
using Tamak.Data.Extensions;
using Tamak.Data.Interfaces;
using Tamak.Data.Models;
using Tamak.Data.Repository;
using Tamak.Data.Response;
using Tamak.Service.Interfaces;
using Tamak.ViewModels;

namespace Tamak.Service.Implementations
{
    public class UserService : IUserService
    {
        private readonly IBaseRepository<User> _userRepository;

        public UserService(IBaseRepository<User> userRepository)
        {
            _userRepository = userRepository;
        }
```

```
public async Task<BaseResponse<IEnumerable<User>>> GetUsers()
{
    try
    {
        var users = _userRepository.GetAll()
            .Select(x => new User()
            {
                Id = x.Id,
                Email = x.Email,
                Name = x.Name,
                Role = x.Role,
                City = x.City,
                Campus = x.Campus
            });

        return new BaseResponse<IEnumerable<User>>()
        {
            Data = users,
            StatusCode = StatusCode.Success
        };
    }
    catch (Exception ex)
    {
        throw;
    }
}
```

```
public async Task<IBaseResponse<UserViewModel>> GetUser(string email)
{
    try
    {
        var product = await _userRepository.GetAll().FirstOrDefaultAsync(x => x.Email ==
email);
        if (product == null)
        {
            return new BaseResponse<UserViewModel>()
            {
                Description = "Пользователь не найден",
                StatusCode = StatusCode.UserNotFound
            };
        }

        var data = new UserViewModel()
        {
            Email = product.Email,
            Name = product.Name,
            City = product.City.GetDisplayName(),
            Campus = product.Campus.GetDisplayName(),

        };

        return new BaseResponse<UserViewModel>()
        {
            StatusCode = StatusCode.Success,
            Data = data
```

```
        };
    }
    catch (Exception ex)
    {
        return new BaseResponse<UserViewModel>()
        {
            Description = $"[GetProduct] : {ex.Message}",
            StatusCode = StatusCode.InternalServerError
        };
    }
}

public async Task<BaseResponse<User>> Save(UserViewModel model)
{
    try
    {
        var user = await _userRepository.GetAll()
            .FirstOrDefaultAsync(x => x.Email == model.Email);

        user.Name = model.Name;
        user.City = (City)Enum.Parse(typeof(City), model.City);
        if (model.Campus != null)
        {
            user.Campus = (Campus)Enum.Parse(typeof(Campus), model.Campus);
        }

        await _userRepository.Update(user);
```

```
            return new BaseResponse<User>()
            {
                Data = user,
                Description = "Данные обновлены",
                StatusCode = StatusCode.Success
            };
        }
        catch (Exception ex)
        {
            return new BaseResponse<User>()
            {
                StatusCode = StatusCode.InternalServerError,
                Description = $"Внутренняя ошибка: {ex.Message}"
            };
        }
    }
}
```

1.11. Представления интерфейсов

1.11.1. IAccountService

```
using System.Security.Claims;
using Tamak.Data.Response;
using Tamak.ViewModels;

namespace Tamak.Service.Interfaces
{
    public interface IAccountService
    {
```

```
        Task<BaseResponse<ClaimsIdentity>> Register(RegisterViewModel model);

        Task<BaseResponse<ClaimsIdentity>> Login(LoginViewModel model);
    }
}
```

1.11.2. IAssortimentService

```
using Tamak.Data.Models;
using Tamak.Data.Response;
using Tamak.ViewModels;

namespace Tamak.Service.Interfaces
{
    public interface IAssortimentService
    {
        Task<IBaseResponse<IEnumerable<Product>>> GetItems(string userName);

        Task<IBaseResponse<Product>> GetItem(string userName, long id);

        public Task<IBaseResponse<Product>> Add(string userName);
    }
}
```

1.11.3. IProductService

```
using Tamak.Data.Interfaces;
using Tamak.Data.Models;
using Tamak.Data.Response;
using Tamak.ViewModels;

namespace Tamak.Service.Interfaces
```

```
{
    public interface IProductService
    {
        BaseResponse<Dictionary<long, string>> GetCategories();
        public IBaseResponse<List<Product>> GetProducts();

        public Task<IBaseResponse<ProductViewModel>> GetProduct(long id);

        public Task<BaseResponse<Dictionary<long, string>>> GetProduct(string term);

        public Task<IBaseResponse<Product>> Create(long assortimentId, Assortiment assortiment);

        public Task<IBaseResponse<bool>> DeleteProduct(long id);

        public Task<IBaseResponse<Product>> Edit(long id, ProductViewModel model);

        public Task<BaseResponse<Product>> Save(ProductViewModel model);

        public Task<BaseResponse<Product>> ChangeAvaliable(ProductViewModel model);
    }
}
```

1.11.4. IUserService

```
using Tamak.Data.Models;
using Tamak.Data.Response;
using Tamak.ViewModels;

namespace Tamak.Service.Interfaces
{
```

```
    public interface IUserService
    {
        Task<BaseResponse<IEnumerable<User>>> GetUsers();

        public Task<IBaseResponse<UserViewModel>> GetUser(string id);

        public Task<BaseResponse<User>> Save(UserViewModel model);
    }
}
```

1.12. Модели представления

1.12.1. HomeViewModel

```
using Tamak.Data.Models;
using Tamak.Data.Response;

namespace Tamak.ViewModels
{
    public class HomeViewModel
    {
        public IEnumerable<Product> allProducts { get; set; }

        public IEnumerable<Assortiment> allAssortiments { get; set; }

        public IEnumerable<User> allUsers { get; set; }

        public Assortiment assortiment { get; set; }

        public LoginViewModel loginViewModel { get; set; }
    }
```

```
}
```

1.12.2. LoginViewModel

```
using System.ComponentModel.DataAnnotations;

namespace Tamak.ViewModels
{
    public class LoginViewModel
    {
        [Required(ErrorMessage = "Введите почту")]
        public string Email { get; set; }

        [Required(ErrorMessage = "Введите пароль")]
        [DataType(DataType.Password)]
        [Display(Name = "Пароль")]
        public string Password { get; set; }
    }
}
```

1.12.3. ProductViewModel

```
using System.ComponentModel.DataAnnotations;
using Tamak.Data.Models;

namespace Tamak.ViewModels
{
    public class ProductViewModel
    {
        public long Id { get; set; }
```

```
        [Display(Name = "Название")]
        [Required(ErrorMessage = "Введите название продукта")]
        [MinLength(1, ErrorMessage = "Минимальная длина должна быть больше одного символа")]
        public string Name { get; set; }


        [Display(Name = "Описание")]
        [MinLength(1, ErrorMessage = "Минимальная длина должна быть больше одного символа")]
        public string Description { get; set; }


        [Display(Name = "Категория продукта")]
        [Required(ErrorMessage = "Выберите категорию")]
        public string Category { set; get; }


        [Display(Name = "Стоимость")]
        [Required(ErrorMessage = "Укажите стоимость")]
        public decimal Price { get; set; }


        public byte[]? Img { get; set; }
        public bool Available { get; set; }
        public IFormFile Avatar { get; set; }
    }
}
```

1.12.4. RegisterViewModel

```
using System.ComponentModel.DataAnnotations;

namespace Tamak.ViewModels
{
    public class RegisterViewModel
    {
        [Required(ErrorMessage = "Укажите имя")]
        public string Name { get; set; }

        [Required(ErrorMessage = "Укажите почту")]
        public string Email { get; set; }

        [Required(ErrorMessage = "Укажите город")]
        public string City { get; set; }

        [DataType(DataType.Password)]
        [Required(ErrorMessage = "Укажите пароль")]
        [MinLength(6, ErrorMessage = "Пароль должен иметь длину больше 6 символов")]
        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Required(ErrorMessage = "Подтвердите пароль")]
        [Compare("Password", ErrorMessage = "Пароли не совпадают")]
        public string PasswordConfirm { get; set; }

        [Required(ErrorMessage = "Укажите корпус, в котором располагается ваша
организация")]
        public string Campus { get; set; }
```

```
        public string Role { get; set; }
    }
}
```

1.12.5. UserViewModel

```
using System.ComponentModel.DataAnnotations;
using Tamak.Data.Enum;

namespace Tamak.ViewModels
{
    public class UserViewModel
    {
        [Display(Name = "Id")]
        public long Id { get; set; }

        [Display(Name = "Роль")]
        public string Role { get; set; }

        [Display(Name = "Имя")]
        public string Name { get; set; }

        [Display(Name = "Почта")]
        public string Email { get; set; }

        [Display(Name = "Корпус")]
        public string Campus { get; set; }

        [Display(Name = "Город")]
```

```
        public string City { get; set; }
    }
}
```

1.13. Представления

1.13.1. Register

```
@model RegisterViewModel


<div class="container col-sm-5 col-md-4 col-lg-3 d-flex align-items-center">
    <div class="form-signin w-100 m-auto">
        <h1 style="font-weight:700;" class="fs-0 text-primary mb-4 text-center">Tamak</h1>


        <form asp-controller="Account" asp-action="Register">
            <div class="row g-3">


                <div class="my-3">
                    <div class="form-check form-check-inline">
                        <input asp-for="Role" value="Admin" id="Admin" name="Role" type="radio" class="form-check-input" required="" onchange='showOrHide("Admin");'>
                        <label class="form-check-label" for="admin">Предприятие</label>
                    </div>
                    <div class="form-check form-check-inline">
                        <input asp-for="Role" value="User" id="User" name="Role" type="radio" class="form-check-input" checked="" required="" onchange='showOrHide("Admin");'>
                        <label class="form-check-label" for="user">Покупатель</label>
                    </div>
                </div>


                <div class="col-12">
                    <label for="City" class="form-label">Город</label>
```

```
            <select asp-for="City" class="form-select" id="city" required="">
                <option value="">Выберите...</option>
                <option value="Moscow">Москва</option>
                <option value="Spb">Санкт-Петербург</option>
                <option value="Nn">Нижний Новгород</option>
                <option value="Perm">Пермь</option>
            </select>
            <div class="invalid-feedback">
                Пожалуйста, выберите город
            </div>
        </div>

        <div asp-validation-summary="All"></div>
        <div class="col-12">
            <label for="Name" class="form-label" id="userBlock">Имя</label>
            <label for="Name" class="form-label" id="adminBlock1" style="display:
none;">Название организации</label>

            <input asp-for="Name" type="name" class="form-control" id="name"
placeholder="Михаил">
            <div class="invalid-feedback">
                Вы ввели неверное имя (такое вообще бывает???)
            </div>
        </div>

        <div class="col-12" id="adminBlock2" style="display: none;">
            <label for="Campus" class="form-label">Корпус</label>
            <select asp-for="Campus" class="form-select" id="campus" required="">
                <option value="None" id="noneOption">Выберите...</option>
```

```
                <option value="Pokrovka">Покровский бульвар</option>
                <option value="Shabolovka">Шаболовка</option>
                <option value="Strogino">Строгино</option>
            </select>
            <div class="invalid-feedback">
                Пожалуйста, выберите корпус
            </div>
        </div>


        <div class="col-12">
            <label for="Email" class="form-label">Почта</label>
            <input asp-for="Email" type="email" class="form-control" id="email"
placeholder="vpupkin@edu.hse.ru">
            <div class="invalid-feedback">
                Вы ввели неверную почту
            </div>
        </div>


        <div class="col-12">
            <label for="email" class="form-label">Пароль</label>

            <input autocomplete="new-password" asp-for="Password" type="password"
class="form-control" placeholder="Пароль" id="password">
            <input autocomplete="new-password" asp-for="PasswordConfirm"
placeholder="Подтвердите пароль" class="form-control" type="password" id="password">

            <div class="invalid-feedback">
```

```
                Вы ввели неверный пароль
            </div>
        </div>

        <a href="/" type="button" class="float-left">Логин</a>
        <button class="w-100 btn btn-primary" type="submit">Зарегистрироваться</button>
      </div>
    </form>
  </div>
</div>

<script>
  function showOrHide(cb) {
    cb = document.getElementById(cb);
    if (cb.checked){
      document.getElementById("adminBlock1").style.display = "block";
      document.getElementById("adminBlock2").style.display = "block";
      document.getElementById("userBlock").style.display = "none";
      document.getElementById("noneOption").value = "";
    }
    else{
      document.getElementById("adminBlock1").style.display = "none";
      document.getElementById("adminBlock2").style.display = "none";
      document.getElementById("userBlock").style.display = "block";
      document.getElementById("noneOption").value = "None";
    }
  }
</script>
```

1.13.2. Index

```
@model HomeViewModel


@if (User.Identity.IsAuthenticated) {
   <div class="container">
   <header class="d-flex flex-wrap justify-content-center py-3">
      <a href="/" class="d-flex align-items-center mb-3 mb-md-0 me-md-auto link-body-emphasis text-decoration-none">
         <span class="fs-2 text-primary" style="font-weight:700">Tamak</span>
      </a>


      <ul class="nav nav-pills">
            <li class="nav-item"><a href="/User/GetUser" class="nav-link text-dark">Профиль</a></li>
            <li class="nav-item">
               @if (User.IsInRole("User"))
               {
                  <a href="#" class="nav-link text-dark">Корзина</a>
               } else
               {
                  <a href="#" class="nav-link text-dark">Список заказов</a>
               }
            </li>
            <li class="nav-item">
            <form method="post" asp-controller="Account" asp-action="Logout">
               <input class="nav-link text-dark" type="submit" value="Выйти"/>
            </form>
         </li>
      </ul>
```

```
    </header>
</div>


    <div style="background-color: #f2f2f2; min-height: 620px;" class="album py-5">
    <div class="container">
        @{
            if (User.IsInRole("Admin"))
            {
                @if (Model.allProducts == null || !Model.allProducts.Any())
                {
                    <center>
                        <div class="d-flex align-items-center">
                            <h3>Упс, а еда тут еще не появилась...</h3>
                        </div>
                    </center>
                }
                else
                {
                        <div class="row row-cols-1 row-cols-sm-2 row-cols-md-3 g-3">
                            @foreach (Product product in Model.allProducts)
                            {
                                @Html.Partial("AllProducts", product)
                            }
                        </div>

                }
```

```
        }
        else
        {
            <div class="row rounded-4 w-100" style="background-color: white">
                <div class="container-fluid py-3">
                    <center>
                        <h3 class="fs-3 text-primary" style="font-weight:700">Вы еще ничего не
заказали</h3>
                    </center>
                </div>
            </div>

            bool flag = false;
            foreach (User user in Model.allUsers)
            {
                if (user.Role.ToString() == "Admin")
                {
                    flag = true;

                    <div class="row rounded-4 w-100 my-4" style="background-color: white">
                        <div class="container-fluid p-3">
                            <h3 class="fs-3 mx-5" style="font-
weight:700">@user.Name.ToString()</h3>

                            @if (user.City.ToString() == "Москва")
                            {
                                <p class="mx-5">@user.City.ToString(), @user.Campus.ToString()</p>
                            } else
                            {
```

```
                        <p class="mx-5">@user.City.ToString()</p>
                    }


                </div>
            </div>
        }
    }
    if (!flag)
    {
        <div class="col-lg-3 d-flex align-items-center">
            <h3>Упс, а еда тут еще не появилась....</h3>
        </div>
    }


    <div class="row row-cols-1 row-cols-sm-2 row-cols-md-3 g-3"></div>
  }
}
@if (User.IsInRole("Admin"))
{
  <form asp-controller="Home" asp-action="Add">
    <button class="w-100 btn btn-primary mt-4" type="submit">Добавить
продукт</button>
  </form>
}
</div>
</div>
```

```
}
else
{
    @Html.Partial("Login", Model.loginViewModel)
}

@section pageScripts {
    <script>
        function ckick(str) {
            const data = $(str).serialize();
            $.ajax({
                url: '@Url.Action("Save")',
                type: 'POST',
                data: data,

            });
        }

        function clickAvaliable(str, str2, str3, str4) {
            const data = $(str).serialize();
            $.ajax({
                url: '@Url.Action("ChangeAvaliable")',
                type: 'POST',
                data: data,

            });
```

```
            $(str3).css('opacity', $(str3).css('opacity') === '1' ? '0.6' : '1');

            $(str2).css('opacity', $(str2).css('opacity') === '1' ? '0.5' : '1');


            if (document.getElementById(str4).textContent === "Нет в наличии") {
                document.getElementById(str4).textContent  = "Есть в наличии";
            } else {
                document.getElementById(str4).textContent = "Нет в наличии";
            }
        }


    </script>
}
```

1.13.3. _Layout

```
<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Tamak</title>
    <link href="/css/bootstrap.min.css" rel="stylesheet" type="text/css">
    <link href="/css/style.css" rel="stylesheet" type="text/css">
</head>
<body>
    @RenderBody()
```

```
    <footer class="container mb-4 mt-4 fixed-bottom">
        <p class="float-end"><a href="#">Наверх</a></p>
        <p>© Курсовая работа 2023 Омирбековой Дарии · </p>
    </footer>

    <script src="/js/bootstrap.min.js"></script>

    <script src="//cdn.jsdelivr.net/npm/sweetalert2@11"></script>
<script src="~/lib/jquery/dist/jquery.min.js"></script>
<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
<script src="~/js/site.js" asp-append-version="true"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/select2/4.0.13/js/select2.min.js"></script>
<script src="~/js/modal.js"></script>
@await RenderSectionAsync("pageScripts", required: false)
</body>
</html>
```

1.13.4. AllProducts

```
@model Product

@if (Model.Available.ToString() == "True")
{
    <div class="col">
        <div class="card shadow-sm" id="@Model.Id-backCard" style="opacity: 1">
            <img class="bd-placeholder-img mx-auto" width="210" height="225" role="img"
preserveAspectRatio="xMidYMid slice" focusable="false" src="https://dodopizza-
a.akamaihd.net/static/Img/Products/d45790074f574ccfa9c75884dfe55f09_584x584.webp" />
            <div class="card-body">
                @if (User.IsInRole("User"))
                {
```

```
            <h3 class="fs-2">@Model.Name</h3>
            <p class="text-body-secondary">@Model.Description</p>
            <div class="d-flex justify-content-between align-items-center">
              <p class="card-text">@Model.Price.ToString("c")</p>
              <div class="btn-group">
                <button asp-controller="ShopCart" asp-action="Add" asp-route-id="@Model.Id" type="button" class="btn btn-sm btn-outline-primary">Добавить</button>
              </div>
            </div>
          }
          @if (User.IsInRole("Admin"))
          {
            <form id="@Model.Id-productCardId">
              <input asp-for="Available" value="@Model.Available" name="Available" type="radio" class="form-check-input" checked="" required="" style="display:none">
              <input asp-for="Id" value="@Model.Id" name="Id" type="radio" class="form-check-input" checked="" required="" style="display:none">
              <input type="text" class="form-control fs-2 @Model.Id-formColor" asp-for="Name" style="opacity: 1">
              <textarea type="text" class="form-control text-body-secondary @Model.Id-formColor" asp-for="Description" style="opacity: 1"></textarea>
              <div class="justify-content-between align-items-center">
                <input type="text" class="form-control card-text @Model.Id-formColor" asp-for="Price" style="opacity: 1">
              </div>

              <button type="button" class="col-12 btn btn-sm btn-outline-primary" id="@Model.Id-saveProductId" onclick='ckick("#@Model.Id-productCardId");'>Сохранить</button>

            </form>
```

```
                <form id="@Model.Id-productAvaliable">
                    <input asp-for="Id" value="@Model.Id" name="Id" type="radio" class="form-check-input" checked="" required="" style="display:none">
                    <input id="avaliableInput" asp-for="Available" value="@Model.Available" name="Available" type="radio" class="form-check-input" checked="" required="" style="display:none">
                    <button id="@Model.Id-changeAvaliable" type="button" class="col-12 btn btn-sm btn-outline-primary" onclick='clickAvaliable("#@Model.Id-productAvaliable", "#@Model.Id-backCard", ".@Model.Id-formColor", "@Model.Id-changeAvaliable");'>Есть в наличии</button>
                </form>
            }


        </div>
    </div>
</div>
} else
{
<div class="col">
    <div class="card shadow-sm" id="@Model.Id-backCard" style="opacity: 0.5">
        <img class="bd-placeholder-img mx-auto" width="210" height="225" role="img" preserveAspectRatio="xMidYMid slice" focusable="false" src="https://dodopizza-a.akamaihd.net/static/Img/Products/d45790074f574ccfa9c75884dfe55f09_584x584.webp" />
        <div class="card-body">
            @if (User.IsInRole("User"))
            {
                <h3 class="fs-2">@Model.Name</h3>
                <p class="text-body-secondary">@Model.Description</p>
                <div class="d-flex justify-content-between align-items-center">
                    <p class="card-text">@Model.Price.ToString("c")</p>
                    <div class="btn-group">
```

```
                    <button asp-controller="ShopCart" asp-action="Add" asp-route-id="@Model.Id"
type="button" class="btn btn-sm btn-outline-primary" style="display: none;">Добавить</button>
                </div>
            </div>
        }
        @if (User.IsInRole("Admin"))
        {
            <form id="@Model.Id-productCardId">
                <input asp-for="Available" value="@Model.Available" name="Available"
type="radio" class="form-check-input" checked="" required="" style="display:none">
                <input asp-for="Id" value="@Model.Id" name="Id" type="radio" class="form-
check-input" checked="" required="" style="display:none">
                <input type="text" class="form-control fs-2 @Model.Id-formColor" asp-
for="Name" style="opacity: 0.6">
                <textarea type="text" class="form-control text-body-secondary
@Model.Id=formColor" asp-for="Description" style="opacity: 0.6"></textarea>
                <div class="justify-content-between align-items-center">
                    <input type="text" class="form-control card-text @Model.Id-formColor" asp-
for="Price" style="opacity: 0.6">
                </div>

                <button type="button" class="col-12 btn btn-sm btn-outline-primary"
id="@Model.Id-saveProductId">Сохранить</button>

            </form>
            <form id="@Model.Id-productAvaliable">
                <input asp-for="Id" value="@Model.Id" name="Id" type="radio" class="form-
check-input" checked="" required="" style="display:none">
                <input id="avaliableInput" asp-for="Available" value="@Model.Available"
name="Available" type="radio" class="form-check-input" checked="" required=""
style="display:none">
```

```
                    <button id="@Model.Id-changeAvaliable" type="button" class="col-12 btn btn-sm
btn-outline-primary" onclick='clickAvaliable("#@Model.Id-productAvaliable", "#@Model.Id-
backCard", ".@Model.Id-formColor", "@Model.Id-changeAvaliable");'>Есть в наличии</button>


                </form>
            }


        </div>
    </div>
</div>
}


<script>
    function ChangeColor(avaliableInput) {
        avaliableInput = document.getElementById("avaliableInput");
        if (avaliableInput.Value) {
            document.getElementById("changeAvaliable").textContent = 'Есть в наличии';


            document.getElementById("form1").style.opacity = "0.6";
            document.getElementById("form2").style.opacity = "0.6";
            document.getElementById("form3").style.opacity = "0.6";
            document.getElementById("backCard").style.opacity = "0.4";
            document.getElementById("backCard").style.backgroundColor = "#e3e3e3;";
        }
        else {
            document.getElementById("changeAvaliable").textContent = 'Нет в наличии';


            document.getElementById("form1").style.opacity = "1";
            document.getElementById("form2").style.opacity = "1";
```

```
        document.getElementById("form3").style.opacity = "1";


        document.getElementById("backCard").style.opacity = "1";
        document.getElementById("backCard").style.backgroundColor = "white";
      }
   }
</script>
```

1.13.5. Login

```
@model LoginViewModel


<div style="height: 600px;" class="container col-3 d-flex align-items-center">
   <div class="form-signin w-100 m-auto">
      <h1 style="font-weight:700;" class="fs-0 text-primary mb-4 text-center">Tamak</h1>


      <form autocomplete="on" method="post" asp-controller="Account" asp-action="Login"
class="needs-validation" novalidate="">
         <div class="row g-3">
            <div asp-validation-summary="All"></div>
            <div class="col-12">
               <label for="email" class="form-label">Почта</label>
               <input asp-for="Email" type="email" class="form-control" id="email"
placeholder="vpupkin@edu.hse.ru">
               <div class="invalid-feedback">
                  Вы ввели неверную почту
               </div>
            </div>


            <div class="col-12">
               <label for="email" class="form-label">Пароль</label>
```

```
            <input input autocomplete="new-password" asp-for="Password" type="password" class="form-control" id="email" placeholder="Пароль">
            <div class="invalid-feedback">
                Вы ввели неверный пароль
            </div>
        </div>
        <a href="/Account/Register" type="button" class="float-left">Регистрация</a>
        <button class="w-100 btn btn-primary" type="submit">Войти</button>
    </div>
</form>
</div>
</div>
```

1.13.6. GetUser

```
@model UserViewModel


<div class="container">
  <header class="d-flex flex-wrap justify-content-center py-3">
    <a href="/" class="d-flex align-items-center mb-3 mb-md-0 me-md-auto link-body-emphasis text-decoration-none">
      <span class="fs-2 text-primary" style="font-weight:700">Tamak</span>
    </a>


    <ul class="nav nav-pills">
      <li class="nav-item"><a href="/User/GetUser" class="nav-link text-dark">Профиль</a></li>
      <li class="nav-item"><a href="#" class="nav-link text-dark">Корзина</a></li>
      <li class="nav-item">
        <form method="post" asp-controller="Account" asp-action="Logout">
          <input class="nav-link text-dark" type="submit" value="Выйти" />
```

```
            </form>
          </li>
        </ul>
    </header>
</div>


<div style="background-color: #f2f2f2" class="album py-5">
    <div style="height: 570px;" class="container col-3 d-flex align-items-center">
        <div class="form-signin w-100 m-auto">

            <form id="UserId">
                <p id="vis1" style="display: block; visibility: hidden;">@Model.City</p>
                <p id="vis2" style="display: block; visibility: hidden;">@Model.Campus</p>

                <div class="col-12">
                    <label for="Email" class="form-label">Почта</label>
                    <input asp-for="Email" type="email" class="form-control" id="email"
placeholder="@Model.Email" readonly>
                </div>

                <div class="col-12">
                    @if (User.IsInRole("User"))
                    {
                        <label for="Name" class="form-label">Имя</label>
                    } else
                    {
                        <label for="Name" class="form-label">Название организации</label>
```

```
            }
            <input type="text" class="form-control" asp-for="Name"
placeholder="@Model.Name">
            <div class="invalid-feedback">
                Вы ввели неверное имя (такое вообще бывает???)
            </div>
        </div>

        <div class="col-12">
            <label for="City" class="form-label">Город</label>
            <select asp-for="City" class="form-select" id="city" required=""
onchange='showOrHideCampus("city");'>
                <option value="Moscow">Москва</option>
                <option value="Spb">Санкт-Петербург</option>
                <option value="Nn">Нижний Новгород</option>
                <option value="Perm">Пермь</option>
            </select>
            <div class="invalid-feedback">
                Пожалуйста, выберите город
            </div>
        </div>

        @if (User.IsInRole("Admin"))
        {
            <div class="col-12" id="adminBlockMoscow">
                <label for="Campus" class="form-label">Корпус</label>
                <select asp-for="Campus" class="form-select" id="campus" required="">
                    <option value="Pokrovka">Покровский бульвар</option>
                    <option value="Shabolovka">Шаболовка</option>
```

```
                    <option value="Strogino">Строгино</option>
                </select>
                <div class="invalid-feedback">
                    Пожалуйста, выберите корпус
                </div>
            </div>
        }



        <button type="button" class="col-12 btn btn-sm btn-outline-primary mt-4"
id="saveUserId">Сохранить</button>


    </form>


</div>
</div>


@section pageScripts {
    <script>
        $('#saveUserId').on('click', function () {
            const data = $('#UserId').serialize();
            $.ajax({
                url: '@Url.Action("Save")',
                type: 'POST',
                data: data,


            });
```

```
});

    if (document.getElementById("vis1").style.display === "block" &&
document.getElementById("vis2").style.display === "block") {
        selectCampus(document.getElementById("vis2").textContent);
        selectCity(document.getElementById("vis1").textContent);
        showOrHideCampus("city");

        document.getElementById("vis1").style.display = "none";
        document.getElementById("vis2").style.display = "none";
    }

    function showOrHideCampus(cb) {
        cb = document.getElementById(cb);
        const option = cb.querySelector(`option[value="${cb.value}"]`)
        if (option.value === "Moscow") {
            document.getElementById("adminBlockMoscow").style.display = "block";
        }
        else {
            document.getElementById("adminBlockMoscow").style.display = "none";
        }
    }

    function selectCity(str) {
        if (str === "Москва") {
            str = "Moscow";
        }
        if (str === "Санкт-Петербург") {
```

```
        str = "Spb";
    }
    if (str === "Нижний Новгород") {
        str = "Nn";
    }
    if (str === "Пермь") {
        str = "Perm";
    }
    const select = document.querySelector('#city').getElementsByTagName('option');
    for (let i = 0; i < select.length; i++) {
        if (select[i].getAttribute('selected') === 'selected') {
            select[i].removeAttribute('selected');
        }
        if (select[i].value === str) select[i].selected = true;
    }
}

function selectCampus(str) {
    if (str === "Покровский бульвар") {
        str = "Pokrovka";
    }
    if (str === "Шаболовка") {
        str = "Shabolovka";
    }
    if (str === "Строгино") {
        str = "Strogino";
    }
    const select = document.querySelector('#campus').getElementsByTagName('option');
```

```
            for (let i = 0; i < select.length; i++) {
                if (select[i].getAttribute('selected') === 'selected') {
                    select[i].removeAttribute('selected');
                }
                if (select[i].value === str) select[i].selected = true;
            }
        }
    </script>
}
```

1.13.7. _ViewImports

```
@using Tamak.ViewModels
@using Tamak.Data.Models
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

1.13.8. _ViewStart

```
@{
    Layout = "_Layout";
}
```

1.14. appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
```

"DefaultConnection":
"Server=(localdb)\\MSSQLLocalDB;Database=Tamak;Trusted_Connection=True;MultipleActiveResultSets=true"

}

}

1.15. dbsettiong.json

{

"ConnectionStrings": {

"DefaultConnection":
"Server=(localdb)\\MSSQLLocalDB;Database=Tamak;Trusted_Connection=True;MultipleActiveResultSets=true"

}

}

1.16. Initializer

```
using Automarket.Service.Implementations;
using Microsoft.AspNetCore.Cors.Infrastructure;
using Tamak.Data.Interfaces;
using Tamak.Data.Models;
using Tamak.Data.Repository;
using Tamak.Service.Implementations;
using Tamak.Service.Interfaces;


namespace Tamak
{
   public static class Initializer
   {
      public static void InitializeRepositories(this IServiceCollection services)
      {
         services.AddScoped<IBaseRepository<Product>, ProductRepository>();
```

```
            services.AddScoped<IBaseRepository<User>, UserRepository>();
            services.AddScoped<IBaseRepository<Assortiment>, AssortimentRepository>();
            /*services.AddScoped<IBaseRepository<Basket>, BasketRepository>();
            services.AddScoped<IBaseRepository<Order>, OrderRepository>();*/
        }

        public static void InitializeServices(this IServiceCollection services)
        {
            services.AddScoped<IProductService, ProductService>();
            services.AddScoped<IAccountService, AccountService>();
            services.AddScoped<IUserService, UserService>();
            services.AddScoped<IAssortimentService, AssortimentService>();
            /*services.AddScoped<IBasketService, BasketService>();
            services.AddScoped<IOrderService, OrderService>();*/
        }
    }
}
```

1.17. Program

```
using Tamak;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.EntityFrameworkCore;
using Tamak.Data;
using Microsoft.AspNetCore.Identity;

var builder = WebApplication.CreateBuilder(args);
builder.Services.AddControllersWithViews().AddRazorRuntimeCompilation();

var connection = builder.Configuration.GetConnectionString("DefaultConnection");
```

```
builder.Services.AddDbContext<AppDBContent>(options =>
    options.UseSqlServer(connection));

builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
    .AddCookie(options =>
    {
        options.LoginPath = new Microsoft.AspNetCore.Http.PathString("/Account/Login");
        options.AccessDeniedPath = new Microsoft.AspNetCore.Http.PathString("/Account/Login");
    });

builder.Services.InitializeRepositories();
builder.Services.InitializeServices();

var app = builder.Build();

if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();
app.UseAuthorization();
```

```
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");
app.Run();
```

ПРИЛОЖЕНИЕ 1

ТЕРМИНОЛОГИЯ

1. **База данных** – структура хранения данных, которая задает определенные правила для способа хранения информации, а также методы взаимодействия данных между собой.

2. **Веб-приложение/Веб-сервис** – программное обеспечение, функционирующее в сети Интернет.

3. **Покупатель** – в контексте данного веб-приложения покупателем является пользователь, который намерен купить продукты питания в предприятиях питания НИУ ВШЭ через предоставленный веб-сервис.

4. **Предприятие –** в контексте данного веб-приложения предприятием является пользователь, который намерен продавать продукцию предприятий питания НИУ ВШЭ через предоставленный веб-сервис.

5. **Хеширование** – процесс, производящийся в хеш-функции.

6. **Хеш-функция** – это функция, которая производит шифрование входных данных путем преобразования поступившей строки в битовую строку.

7. **Bootstrap** – фреймворк языка разметки HTML, который упрощает использование стилей CSS.

8. **JQuery** – библиотека языка Java Script, которая упрощает взаимодействие Java Script и HTML, а также предоставляет удобный API для работы с AJAX.

9. **Entity** – фреймворк предназначенный для упрощенного взаимодействия с базой данных**.**

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц в докум.)	№ документа	Входящий № сопроводительного документа и дата	Подп.	Дата
Изм.	Измененных	Замен-ных	Новых	Аннулированных					

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.15-01 81				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата