МЕЖГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

РАЗРАБОТКА ИНТЕРНЕТ-МАГАЗИНА

Курсовая работа по дисциплине «Основы алгоритмизации и программирования»

КР.1-53 01 02.№10028398.20

Исполнитель Шамяков К.Е., АСОИ - 181  
 (подпись)  
Руководитель Кашпар А.И.  
 (подпись)

Дата допуска к защите ттт

Дата защиты т

Оценка т

Могилев 2019

ГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ  
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Факультет Электротехнический

«Утверждаю»

Заведующий кафедрой\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

З А Д А Н И Е

на курсовую работу по дисциплине «Основы алгоритмизации и программирования» студенту Шамяков К.Е. гр. АСОИ-181.

1. Тема курсовой работы: Разработка программного модуля.

2. Срок сдачи законченной работы:

3. Исходные данные к курсовой работе:

4. Перечень подлежащих разработке вопросов:

Введение

1 Техническое задание

1.1 Анализ предметной области

1.2 Описание метода решения задачи

2 Описание программы

3 Руководство оператора

4 Тестирование

Заключение

Список использованных источников

5. На проверку предоставляются пояснительная записка, исходные тексты программ и исполняемые файлы на электронном носителе.

Руководитель курсовой работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кашпар А.И.

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шамяков К.Е

Содержание

[1 Техническое задание 5](#_Toc7162508)

[2 Описание программы 6](#_Toc7162509)

[2.1 Общие сведения 6](#_Toc7162510)

[Авторизация реализована через cookie-файлы, что дает возможность оставаться в аккаунте даже после перезапуска проекта. К тому же это добавляет безопасность, т.к. с помощью механизмов рефлексии мы можем проверять имеет ли текущий пользователь доступ к конкретному ресурсу или нет. Каждый новый Http-запрос сервер запрашивает cookie браузера, при помощи метода ниже. 8](#_Toc7162511)

[**2.2 Функциональное назначение** 8](#_Toc7162512)

[**2.3 Описание логической структуры** 8](#_Toc7162513)

[**2.4 Используемые технические средства** 8](#_Toc7162514)

[**2.5 Вызов и загрузка** 8](#_Toc7162515)

[**2.6 Входные данные** 9](#_Toc7162516)

[**2.7 Выходные данные** 9](#_Toc7162517)

[**3.Руководство оператора** 9](#_Toc7162518)

[**3.1 Назначение программы** 9](#_Toc7162519)

[**3.2 Условия выполнения программы** 9](#_Toc7162520)

[**3.3 Выполнение программы** 10](#_Toc7162521)

[**3.4** **Сообщения оператору** 12](#_Toc7162522)

[**4 Тестирование** 13](#_Toc7162523)

Лист

№ докум.

Подпись

Дата

Лист

3

КР.1-53 01 02.№ 10028398.20

Курсовая работа “Разработка программного модуля”

Лит.

Листов

БРУ, гр. АСОИ-181

# 1 Техническое задание

Разработать интернет – магазин, поддерживающий простые операции по покупке\продаже товаров, их редактирование, возможность заходить с разных ролей, имеющих разные возможности. Сделать авторизацию, аутентификацию, регистрацию через cookie-файлы. Работу с файлами производить при помощи языка запросов SQL. Данные хранить в базе данных Ms SQL Server

Изм.

Лист

№ докум.

Подпись

Дата

Лист

5

КР.1-53 01 02.№ 10028398.15.81-01

# 2 Описание программы

## 2.1 Общие сведения

Данный проект представляет интернет-магазин, имеющий следующие особенности:

**Каждый товар в магазине содержит:**

· Id (Для операций с товарами)

· Название

· Описание

· Цена

· Дата создания

· Дата последнего изменения

· Категория

- Название категории

· Контакты автора

- Имя

- Номер телефона

- Email

· Местонахождение товара

· Состояние

**Система предоставляет возможность изменять/добавлять/удалять любую информацию о товарах.**

**Система предоставляет навигацию:**

· По категориям

· По полному или частичному имени товара

Также система содержит страницу поиска по следующим полям: название товара(частичное и полное, без учета регистра), диапазон цен, категория, состояние, местонахождение товара

**Система обладает аутентификацией и авторизацией. Система может работать в 4-х режимах:**

**· Режим гостя. Данный режим разрешает:**

- Пользоваться навигацией

- Пользоваться поиском

- Просматривать товары

**· Режим пользователя. Данный режим разрешает все, что может гость, а также:**

- Просмотр списка своих товаров

- Добавлять/изменять/удалять свои товары

- Редактировать свой профиль

- Покупать товары других пользователей

**· Режим редактора. Данный режим разрешает все, что может пользователь, а также:**

- Создавать/удалять/изменять любые товары

**· Режим администратора. Данный режим разрешает все, что может редактор, а также:**

- Создавать/изменять/удалять пользователей

- Добавлять новые категории

- Добавлять новые состояния товаров

**Каждый пользователь в магазине содержит:**

· Id

· Логин

· Пароль

· Почту

· Номер телефона

· Роль

Авторизация реализована через cookie-файлы, что дает возможность оставаться в аккаунте даже после перезапуска проекта. К тому же это добавляет безопасность, т.к. с помощью механизмов рефлексии мы можем проверять имеет ли текущий пользователь доступ к конкретному ресурсу или нет. Каждый новый Http-запрос сервер запрашивает cookie браузера, при помощи метода ниже.

protected void Application\_PostAuthenticateRequest(object sender, EventArgs arg)

{

var cookie = HttpContext.Current.Request.Cookies[FormsAuthentication.FormsCookieName];

if (cookie != null)

{

var ticket = FormsAuthentication.Decrypt(cookie.Value);

var user = JsonConvert.DeserializeObject<User>(ticket.UserData);

var userPrinciple = new UserPrinciple(user.Login)

{

UserId = user.Id,

Role = user.Role

};

HttpContext.Current.User = userPrinciple;

}

}

## **2.2 Функциональное назначение**

Программа предназначена для легкой продажи товаров и быстрого совершения покупок без комиссии.

# **2.3 Описание логической структуры**



Данная схема очень упрощена, но отражает основные меню магазина, у пользователей с ролями спектр возможностей еще шире.

# 2.4 Используемые технические средства

Программа эксплуатируется на персональном компьютере типа PC. Для работы с приложением используются мышь, клавиатура и дисплей монитора.

## 2.5 Вызов и загрузка

Загрузка проекта осуществляется после запуска локальной базы данных. Проект занимает на жёстком диске около 300 мегабайт.

## 2.6 Входные данные

Входными данными являются данные, получаемые из базы данных Ms SQL Server посредством запросов к ней через технологию ADO.NET.

## 2.7 Выходные данные

Выходными данными являются представления, возвращаемые методами контроллеров.

# 3.Руководство оператора

## 3.1 Назначение программы

Программа предназначена для легкой продажи товаров и быстрого совершения покупок без комиссии.

## 3.2 Условия выполнения программы

Для корректной работы приложения требуется следующее аппаратное обеспечение:

ПЭВМ под управление операционной системы Windows 7/8/10 со следующими аппаратными характеристиками:

• процессор с частотой 1,6 ГГц или выше,

• 2048 МБ оперативной памяти,

• 4 ГБ свободного места на диске,

• жесткий диск со скоростью 5400 об/мин,

• видеоадаптер с поддержкой DirectX 9 и разрешением 1280 x 1024 (или более высоким).

Периферийные устройства:

Изм.

Лист

№ докум.

Подпись

Дата

Лист

8

КР.1-53 01 02.№ 10028398.15.81-01

• манипулятор мышь (работа с пользовательским интерфейсом)

• клавиатура (ввод данных)

• монитор (отображение пользовательского интерфейса)

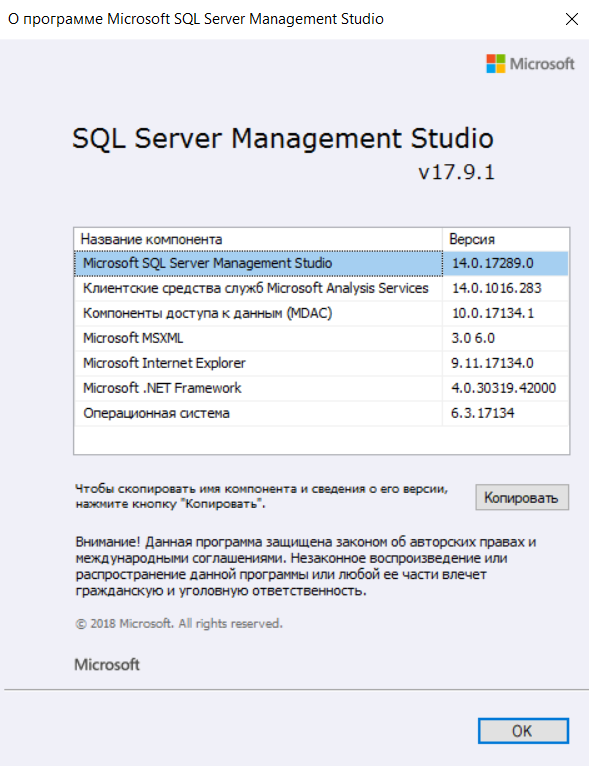
Данный проект работает с MsSQL Server. Его версия, на которой все точно работает указана на рис.1, поэтому для корректной работы программы рекомендую использовать эту версию или более позднюю.

Рисунок 1 – Версия MsSQL Server

## 3.3 Выполнение программы

Если при выполнении описанных ниже шагов будет допущена ошибка, ведущая к некорректной работе приложения - **потребуется очистка cookie браузера**, на котором проект запускается, потому как аутентификация происходит непосредственно через них и в случае некорректного запуска, cookie этого сайта могут сохраниться некорректно, в результате чего цикл аутентификация(сохранение cookie) –> выход из аккаунта (удаление cookie) будет нарушен.

Перейдем к самим шагам:

1. Перейти в папку /CourseWork/database
2. Запустить скрипт CreatedbWithTables.sql
3. Перейти в папку /CourseWork/database/Inserts
4. Запустить скрипт Insert all.sql

Теперь необходимо войти в проект, но не запускать его

Для этого следует перейти в папку Shop.WebApplication и запустить файл .sln(рис.2)

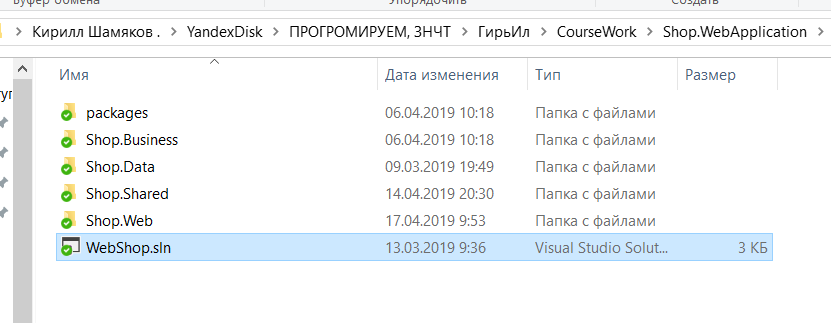


Рисунок 2 – Папка проекта

В пространстве имен Shop.Data.Constants необходимо заменить строку подключения(Рис. 3)

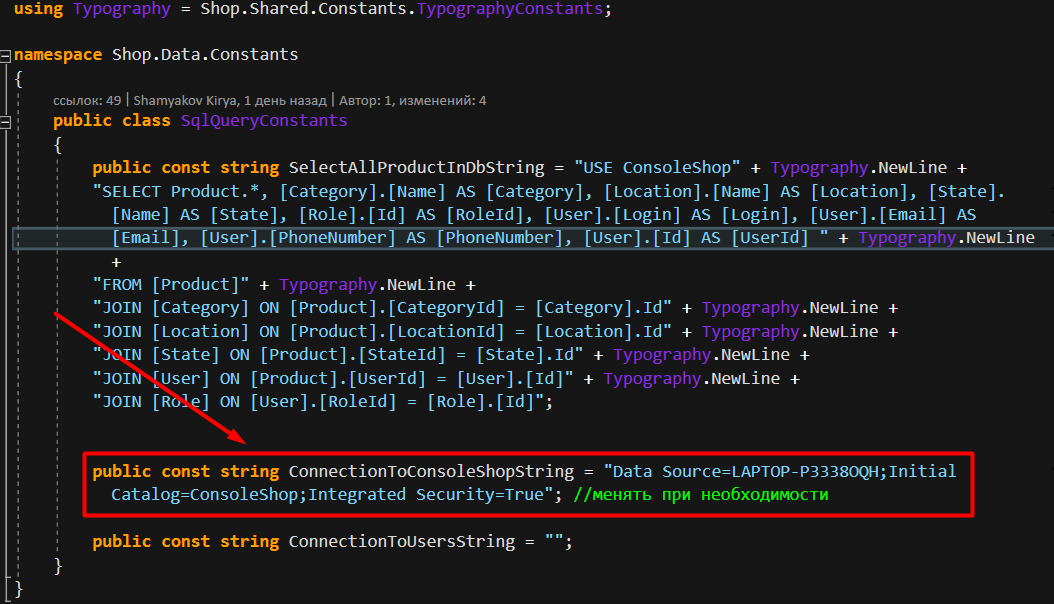


Рисунок 3 – строка подключения

Её можно получить, добавив в обозреватель серверов запущенную по приведенным выше шагам базу данных, после чего открыть окно свойств где собственно строка подключения и находится(рис. 4)

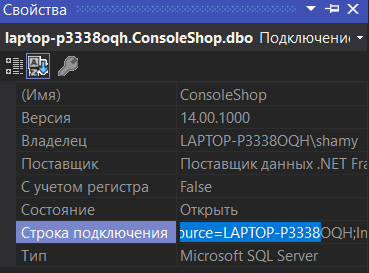
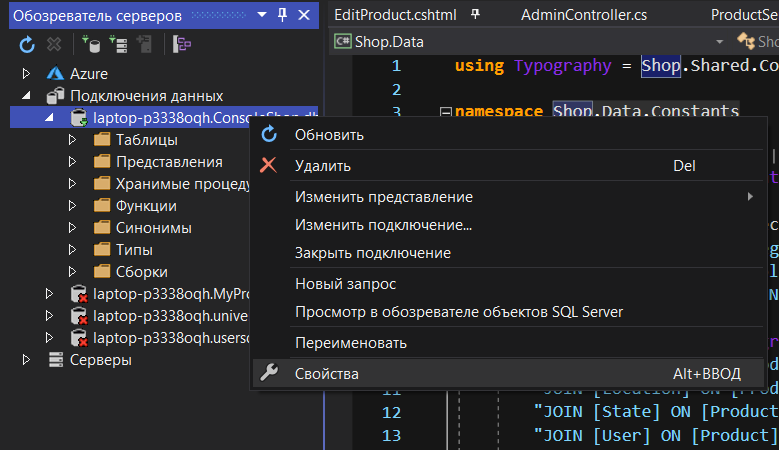


Рисунок 4 – Свойства Базы данных

Если все было сделано правильно, приложение можно запускать.

**Ниже приведён перечень пользователей(при желании новых можно зарегистрировать, и выдать любую роль с помощью аккаунта администратора)**

|  |  |  |
| --- | --- | --- |
| **Login** | **Password** | **Role** |
| **VlastelinValentin** | **павук** | **Administrator** |
| **Anatolya** | **123** | **Moderator** |
| **Vadimka12** | **343221** | **User** |
| **Valentinka48** | **admin** | **Moderator** |
| **Genka15** | **1234** | **User** |
| **Padinhton** | **padinhton** | **User** |

## 3.4 Сообщения оператору

Все сообщения и уведомления в данном проекте производятся с помощью этого представления.

Каждый использующий это представление метод просто дает определенное значение динамической переменной ViewBag.Massage(рис.5).

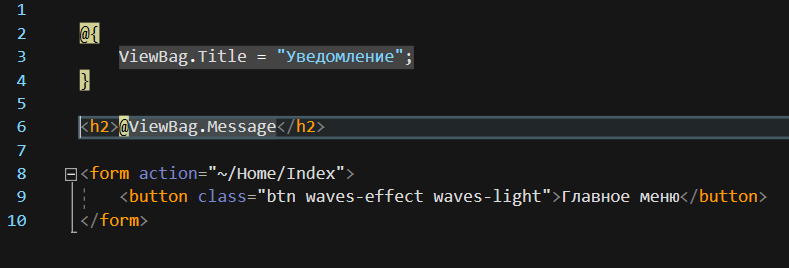


Рисунок 5 – код страницы уведомлений

Так же, при попытке пользователя, не имеющего необходимых прав, вручную написать URL закрытого для него ресурса, он будет перенаправлен на следующую страницу(рис.6)

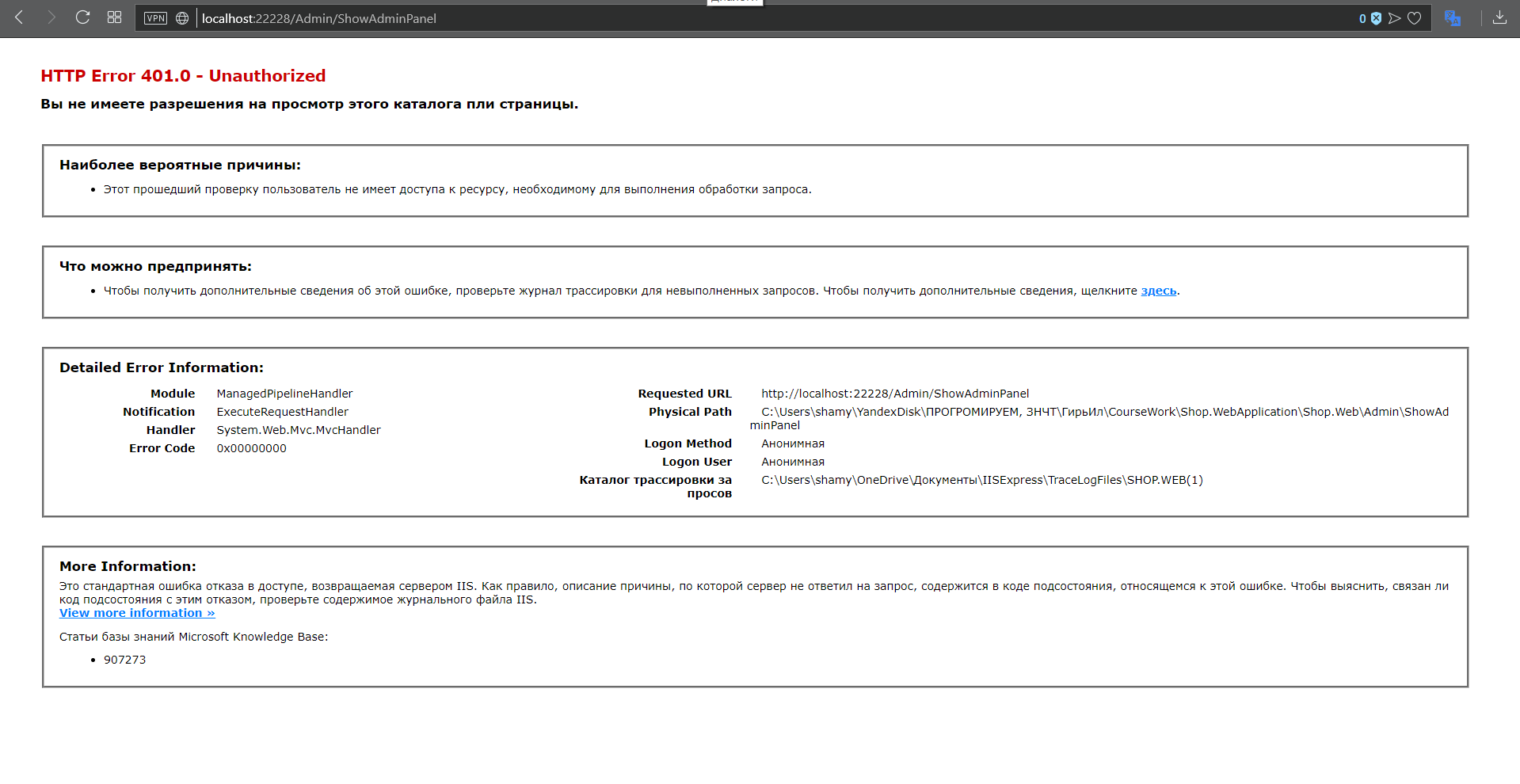


Рисунок – 6 Сообщение системы безопасности

# 4 Тестирование

Для тестирования данного веб-приложения необходимо корректно запустить проект, и если при заходе на сайт вы увидите главное меню – то тестирование завершено успешно, проект работает.

**Заключение**

В результате выполнения данной курсовой работы были улучшены и закреплены навыки создания приложений в среде Visual Studio 2017 на языках C#,

HTML, CSS, SQL с использованием компонентов MVC. Был разработан сайт, поддерживающий регистрацию, аутентификацию, авторизацию. Имеющий множество функций, количество которых зависит от роли текущего пользователя.

Если учитывать что данный проект сделан студентом первого курса, то к достоинствам могу отнести : защита от инъекций при работе с базой данных, наличие аутентификации и авторизации при помощи cookie, что добавляет возможность использовать пространство имен System.Security, а значит и повысить безопасность приложения(Методы предназначенные для администратора никак нельзя использовать пользователям без данной роли, т.к. в каждом методе происходит проверка метаданных, и в последствии роль текущего пользователя), отсутствие необходимости заново авторизироваться, т.к. данные о текущем пользователе сохранятся в cookie браузера и без проблем возьмутся при необходимости. К недостатком можно отнести, то, что дизайн сайта не является приятным для глаз, что отрицательно сказывается на продажах в такого рода магазинах, практические полное отсутствие языка JavaScript, который добавляет много интересных свойств страницам, ну и наконец сильная привязанность к некоторым классам из-за малого количества интерфейсов, что отрицательно сказывается на дальнейшем развитии сайта. Перспективы у сайта я считаю хорошие, так как он разработан таким образом(все разбито на логические части, наличие всех нужных самописных методов для работы с базой данных для каждой из сущностей), что не составляет никакого труда добавить новые функции, менять большие части кода без страха, что весь проект из-за этого больше не вернуть в рабочее состояние.

Литература

1.[Джеффри Рихтер](https://oz.by/people/more901517.html), CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#

2.<https://metanit.com>

**Приложение**

**Класс ProductContext**

Работает с базой данных через технологию ADO.NET. Таких классов несколько, почти на каждый Entity из пространства имен Shop.Shared.Entities (User, Product, Category, Role, Location, Purchase, State)

using Shop.Data.DataContext.Interfaces;

using Shop.Shared.Entities;

using Shop.Shared.Helpers;

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using SqlConst = Shop.Data.Constants.SqlQueryConstants;

using Typography = Shop.Shared.Constants.TypographyConstants;

namespace Shop.Data.DataContext.Realization.MsSql

{

public class ProductContext : IProductContext

{

UserContext \_userContext = new UserContext();

CategoryContext \_categoryContext = new CategoryContext();

StateContext \_stateContext = new StateContext();

LocationContext \_locationContext = new LocationContext();

public IReadOnlyCollection<Product> GetAllByName(string searchParameter, string searchQuery)

{

using (var connection = new SqlConnection(SqlConst.ConnectionToConsoleShopString))

{

connection.Open();

List<Product> products = new List<Product>();

var command = new SqlCommand(SqlConst.SelectAllProductInDbString, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

if (reader[searchParameter].ToString().Contains(searchQuery))

{

products.Add(GetProduct(reader));

}

}

return products;

}

}

public void Edit(Product editedProduct)

{

using (SqlConnection connection = new SqlConnection(SqlConst.ConnectionToConsoleShopString))

{

connection.Open();

var command = new SqlCommand($"UPDATE [Product]{Typography.NewLine}SET [Name] = @name, [Description] = @description, [CategoryId] = @categoryId, [LastModifiedDate] = @lastModifiedDate, [LocationId] = @locationId, [Price] = @price, [StateId] = @stateId, [UserId] = @userId {Typography.NewLine}WHERE Id = {editedProduct.Id}", connection);

command.Parameters.AddWithValue("@name", editedProduct.Name);

command.Parameters.AddWithValue("@description", editedProduct.Description);

command.Parameters.AddWithValue("@categoryId", editedProduct.Category.Id);

command.Parameters.AddWithValue("@lastModifiedDate", editedProduct.LastModifiedDate);

command.Parameters.AddWithValue("@locationId", \_locationContext.GetIdByName(editedProduct.LocationOfProduct));

command.Parameters.AddWithValue("@price", editedProduct.Price);

command.Parameters.AddWithValue("@stateId", editedProduct.State.Id);

command.Parameters.AddWithValue("@userId", editedProduct.Author.Id);

command.ExecuteNonQuery();

}

}

public IReadOnlyCollection<Product> GetByCategoryId(int categoryId)

{

List<Product> products = new List<Product>();

using (var connection = new SqlConnection(SqlConst.ConnectionToConsoleShopString))

{

connection.Open();

using (var command = new SqlCommand(SqlConst.SelectAllProductInDbString, connection))

{

using (SqlDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

if ((int)reader["CategoryId"] == categoryId)

{

products.Add(GetProduct(reader));

}

}

}

}

return products;

}

}

public IReadOnlyCollection<Product> GetByUserId(int userId)

{

List<Product> products = new List<Product>();

using (var connection = new SqlConnection(SqlConst.ConnectionToConsoleShopString))

{

connection.Open();

using (var command = new SqlCommand(SqlConst.SelectAllProductInDbString, connection))

{

using (SqlDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

if ((int)reader["UserId"] == userId)

{

products.Add(GetProduct(reader));

}

}

}

}

return products;

}

}

public Product GetProduct(SqlDataReader reader)

{

return new Product

{

Id = (int)reader["Id"],

Name = (string)reader["Name"],

Description = (string)reader["Description"],

Price = (decimal)reader["Price"],

CreationDate = (DateTime)reader["CreationDate"],

LastModifiedDate = (DateTime)reader["LastModifiedDate"],

Category = new Category

{

Id = (int)reader["CategoryId"],

Name = (string)reader["Category"]

},

Author = new User

{

Id = (int)reader["UserId"],

Login = (string)reader["Login"],

Email = (string)reader["Email"],

PhoneNumber = (string)reader["PhoneNumber"],

Role = RoleHelper.ConvertToRoleType((int)reader["RoleId"])

},

LocationOfProduct = (string)reader["Location"],

State = new State

{

Id = (int)reader["StateId"],

Name = (string)reader["State"]

}

};

}

public Product GetById(int id)

{

using (var connection = new SqlConnection(SqlConst.ConnectionToConsoleShopString))

{

connection.Open();

List<Product> products = new List<Product>();

string query = SqlConst.SelectAllProductInDbString + Typography.NewLine + $"WHERE [Product].[Id] = @id";

var command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@id", id);

SqlDataReader reader = command.ExecuteReader();

reader.Read();

return GetProduct(reader);

}

}

public IReadOnlyCollection<Product> GetAll()

{

List<Product> returnProducts = new List<Product>();

using (var connection = new SqlConnection(SqlConst.ConnectionToConsoleShopString))

{

connection.Open();

List<Product> products = new List<Product>();

var command = new SqlCommand(SqlConst.SelectAllProductInDbString, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

returnProducts.Add(GetProduct(reader));

}

return returnProducts;

}

}

public void DeleteById(int id)

{

using (var connection = new SqlConnection(SqlConst.ConnectionToConsoleShopString))

{

connection.Open();

var command = new SqlCommand($"DELETE [Product] WHERE [Id] = {id}", connection);

command.ExecuteNonQuery();

}

}

public void Save(Product product)

{

using (var connection = new SqlConnection(SqlConst.ConnectionToConsoleShopString))

{

connection.Open();

var command = new SqlCommand($"INSERT INTO [dbo].[Product]([CategoryId],[LocationId],[StateId],[UserId],[Name],[Description],[Price],[CreationDate],[LastModifiedDate]) VALUES(@categoryId, @locationId, @stateId, @authorId, @productName, @description, @price, @creationDate, @lastModifiedDate)", connection);

command.Parameters.AddWithValue("@categoryId", product.Category.Id);

command.Parameters.AddWithValue("@locationId", \_locationContext.GetIdByName(product.LocationOfProduct)); //изменить после того, как сделаю наконец эту систему сложной локации с составными ключами

command.Parameters.AddWithValue("@stateId", product.State.Id);

command.Parameters.AddWithValue("@authorId", product.Author.Id);

command.Parameters.AddWithValue("@productName", product.Name);

command.Parameters.AddWithValue("@description", product.Description);

command.Parameters.AddWithValue("@price", product.Price);

command.Parameters.AddWithValue("@creationDate", product.CreationDate);

command.Parameters.AddWithValue("@lastModifiedDate", product.LastModifiedDate);

command.ExecuteNonQuery();

}

}

}

}

**AdminController(Название говорит само за себя)**

using Shop.Business.Services;

using Shop.Shared.Entities;

using Shop.Web.Attributes;

using Shop.Web.Models;

using Shop.Web.Models.ProductViewModels;

using System.Web.Mvc;

using Shop.Shared.Helpers;

using Shop.Shared.Entities.Enums;

using System;

namespace Shop.Web.Controllers

{

public class AdminController : Controller

{

private UserService \_userService = new UserService();

private ProductService \_productService = new ProductService();

private CategoryService \_categoryService = new CategoryService();

private RoleService \_roleService = new RoleService();

private StateService \_stateService = new StateService();

private LocationService \_locationService = new LocationService();

[Admin]

public ActionResult ShowAdminPanel()

{

var model = new SiteStatViewModel()

{

ProductCount = \_productService.GetAll().Count,

UserCount = \_userService.GetAll().Count

};

return View(model);

}

[Admin]

public ActionResult AddNewCategory()

{

return View(new CategoryViewModel());

}

[Admin]

[HttpPost]

public ActionResult AddNewCategory(CategoryViewModel model)

{

if (!ModelState.IsValid)

{

return View(model);

}

\_categoryService.Save(new Category { Name = model.CategoryName });

ViewBag.Message = $"Категория {model.CategoryName} успешно добавлена!";

return View("~/Views/Shared/Notification.cshtml");

}

[Admin]

public ActionResult EditUser(int id)

{

var user = \_userService.GetById(id);

ViewBag.Roles = \_roleService.GetAll();

return View(new EditUserViewModel

{

Id = user.Id,

Login = user.Login,

Email = user.Email,

Password = user.Password,

PhoneNumber = user.PhoneNumber,

Role = user.Role.ToString()

});

}

[Admin]

[HttpPost]

public ActionResult EditUser(EditUserViewModel model)

{

if (!ModelState.IsValid)

{

ViewBag.Roles = \_roleService.GetAll();

return View(model);

}

\_userService.EditUser(new User

{

Id = model.Id,

Login = model.Login,

Email = model.Email,

Password = model.Password,

PhoneNumber = model.PhoneNumber,

Role = EnumHelper.ParseEnum<RoleType>(model.Role)

});

ViewBag.Message = $"Пользователь {model.Login} изменён успешно!";

return View("~/Views/Shared/Notification.cshtml");

}

[Moder]

public ActionResult EditProduct(int id)

{

var product = \_productService.GetProductById(id);

ViewBag.Categories = \_categoryService.GetAll();

ViewBag.States = \_stateService.GetAll();

return View(new EditProductViewModel

{

Id = id,

Name = product.Name,

Description = product.Description,

Category = product.Category,

Author = product.Author,

Price = product.Price,

State = product.State,

LocationOfProduct = product.LocationOfProduct

});

}

[Moder]

[HttpPost]

public ActionResult EditProduct(EditProductViewModel model)

{

if(!ModelState.IsValid)

{

ViewBag.Categories = \_categoryService.GetAll();

ViewBag.States = \_stateService.GetAll();

return View(model);

}

if (!\_locationService.IsExists(model.LocationOfProduct))

{

\_locationService.Save(model.LocationOfProduct);

}

\_productService.Edit(new Shared.Entities.Product

{

Id = model.Id,

Name = model.Name,

Description = model.Description,

Price = model.Price,

LocationOfProduct = model.LocationOfProduct,

LastModifiedDate = DateTime.Now,

State = model.State,

Category = model.Category,

Author = model.Author

});

ViewBag.Message = $"Товар \"{model.Name}\" изменён успешно!";

return View("~/Views/Shared/Notification.cshtml");

}

}

}

**User Principle(Класс текущего пользователя)**

using Shop.Shared.Entities.Enums;

using System.Security.Principal;

namespace Shop.Shared.Entities.Authorize

{

public class UserPrinciple : IPrincipal

{

public UserPrinciple(string username)

{

this.Identity = new GenericIdentity(username);

}

public decimal Balance { get; set; }

public int UserId { get; set; }

public string Name => this.Identity.Name;

public RoleType Role { get; set; }

public IIdentity Identity { get; private set; }

public bool IsInRole(string role)

{

return this.Role.ToString() == role;

}

}

}

**LoginService**

using System;

using Shop.Shared.Entities;

using Shop.Data.Repositories;

using Shop.Data.DataContext.Realization.MsSql;

using System.Web.Security;

using System.Web;

using Newtonsoft.Json;

namespace Shop.Business.Services.Auth

{

public class LoginService

{

private readonly UserRepository \_userRepository = new UserRepository(new UserContext());

private const int VERSION = 1;

public User Login(string login, string password)

{

if (this.GetUserByLoginAndPassword(login, password) is User user)

{

this.SetCookies(user);

return user;

}

return default(User);

}

public User Register(string login, string password, string email, string phone)

{

var newUser = new User

{

Login = login,

Email = email,

Password = password,

PhoneNumber = phone,

};

\_userRepository.Save(newUser);

var user = \_userRepository.GetUserByLogin(login);

return user ?? null;

}

public void Logout()

{

FormsAuthentication.SignOut();

}

private User GetUserByLoginAndPassword(string login, string password)

{

return \_userRepository.GetUserByLoginAndPassword(login, password);

}

private void SetCookies(User user)

{

var userData = JsonConvert.SerializeObject(user);

var ticket = new FormsAuthenticationTicket(VERSION, user.Login, DateTime.Now, DateTime.Now.AddMinutes(15), false, userData);

var encryptTicket = FormsAuthentication.Encrypt(ticket);

var cookie = new HttpCookie(FormsAuthentication.FormsCookieName, encryptTicket);

HttpContext.Current.Response.Cookies.Add(cookie);

}

}

}

**MvcApplication**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

26

КР.1-53 01 02.№ 10028398.15.81-01

using Newtonsoft.Json;

using Shop.Shared.Entities;

using Shop.Shared.Entities.Authorize;

using System;

using System.Web;

using System.Web.Mvc;

using System.Web.Optimization;

using System.Web.Routing;

using System.Web.Security;

namespace Shop.Web

{

public class MvcApplication : HttpApplication

{

protected void Application\_Start()

{

AreaRegistration.RegisterAllAreas();

RouteConfig.RegisterRoutes(RouteTable.Routes);

BundleConfig.RegisterBundles(BundleTable.Bundles);

}

protected void Application\_PostAuthenticateRequest(object sender, EventArgs arg)

{

var cookie = HttpContext.Current.Request.Cookies[FormsAuthentication.FormsCookieName];

if (cookie != null)

{

var ticket = FormsAuthentication.Decrypt(cookie.Value);

var user = JsonConvert.DeserializeObject<User>(ticket.UserData);

var userPrinciple = new UserPrinciple(user.Login)

{

UserId = user.Id,

Role = user.Role

};

HttpContext.Current.User = userPrinciple;

}

}

}

}

**ProductRepository**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

27

КР.1-53 01 02.№ 10028398.15.81-01

using Shop.Data.DataContext.Realization.MsSql;

using Shop.Data.Repositories;

using Shop.Shared.Entities;

using System;

using System.Collections.Generic;

namespace Shop.Business.Services

{

public class ProductService

{

private ProductRepository \_productRepository = new ProductRepository(new ProductContext());

public IReadOnlyCollection<Product> GetAll()

{

return \_productRepository.GetAll();

}

public IReadOnlyCollection<Product> GetSearchList(string searchParameter, string searchQuery)

{

return \_productRepository.GetAllByName(searchParameter, searchQuery);

}

public IReadOnlyCollection<Product> GetProductsByCategoryId(int categoryId)

{

return \_productRepository.GetAllByCategoryId(categoryId);

}

public IReadOnlyCollection<Product> GetByUserId(int userId)

{

return \_productRepository.GetByUserId(userId);

}

public Product GetProductById(int productId)

{

return \_productRepository.GetById(productId);

}

public void DeleteById(int id)

{

\_productRepository.DeleteById(id);

}

public void Save(Product product)

{

\_productRepository.Save(product);

}

public void Edit(Product editedProduct)

{

\_productRepository.Edit(editedProduct);

}

public IReadOnlyCollection<Product> GetAllWithFilters(IEnumerable<Predicate<Product>> filters)

{

List<Product> products = (List<Product>)\_productRepository.GetAll();

List<Product> returnList = new List<Product>();

bool IsCorrect;

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

{

IsCorrect = true;

foreach(var filter in filters)

{

if (!filter(products[i]))

{

IsCorrect = false;

}

}

if (IsCorrect)

{

returnList.Add(products[i]);

}

}

return returnList;

}

public IReadOnlyCollection<Product> GetAllByFilterParameters(ProductFilterParameters parameters)

{

List<Predicate<Product>> filters = new List<Predicate<Product>>();

if (parameters.MaxPrice != default(decimal))

{

filters.Add(product => product.Price <= parameters.MaxPrice);

}

if (parameters.MinPrice != default(decimal))

{

filters.Add(product => product.Price >= parameters.MinPrice);

}

if (parameters.Name != default(string))

{

filters.Add(product => product.Name.IndexOf(parameters.Name, StringComparison.OrdinalIgnoreCase) >= 0);

}

if (parameters.State.Id != default(int))

{

filters.Add(product => product.State.Id == parameters.State.Id);

}

if (parameters.Category.Id != default(int))

{

filters.Add(product => product.Category.Id == parameters.Category.Id);

}

if(filters == null)

{

return GetAll();

}

return GetAllWithFilters(filters);

}

}

}