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

МЕЖГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО 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](#_Toc8388046)

[2 Описание программы 6](#_Toc8388047)

[2.1 Общие сведения 6](#_Toc8388048)

[2.2 Функциональное назначение 8](#_Toc8388049)

[2.3 Описание логической структуры 8](#_Toc8388050)

[2.4 Используемые технические средства 9](#_Toc8388051)

[2.5 Вызов и загрузка 9](#_Toc8388052)

[2.6 Входные данные 9](#_Toc8388053)

[2.7 Выходные данные 10](#_Toc8388054)

[3 Руководство оператора 11](#_Toc8388055)

[3.1 Назначение программы 11](#_Toc8388056)

[3.2 Условия выполнения программы 11](#_Toc8388057)

[3.3 Выполнение программы 14](#_Toc8388058)

[3.4 Сообщения оператору 20](#_Toc8388059)

[4 Тестирование 25](#_Toc8388060)

[Заключение 26](#_Toc8388061)

[Литература 27](#_Toc8388062)

[Приложение А 28](#_Toc8388063)

Введение

Необходимо разработать сайт, используя C#, SQL, CSS, HTML, JS. Реализовать авторизацию и регистрацию через cookie-файлы. Сайт должен поддерживает все функции, необходимые для купли-продажи товаров, коммуникации между пользователями, а также разделение возможностей пользователей по ролям.

Целью работы является разработка интернет-магазина, поддерживающего простые операции по покупке и продаже товаров, их редактирование, возможность заходить с разных ролей, имеющих разные возможности.

В данной пояснительной записке представлена демонстрация работы приложения, показаны скриншоты сообщений, появляющихся при работе с приложением, подробно расписаны в виде таблиц основные классы сущностей, показаны логическая структура и структура страниц приложения. Также показаны результаты тестирования приложения при помощи юнит-тестов, представлены скриншоты и подробное описание каждой страницы сайта.

# 1 Техническое задание

Разработать интернет-магазин, поддерживающий простые операции по покупке и продаже товаров, их редактирование, возможность заходить с разных ролей, имеющих разные возможности.

Программный продукт должен реализовывать следующие операции:

- удаление, изменение и добавление товаров.

- удаление, изменение и добавление пользователей.

- возможности модератора и администратора при входе в аккаунт с необходимыми ролями.

- поиск товаров по частичному или полному совпадению имени, по категории и по прочим фильтрам.

- поиск пользователей по частичному или полному имени.

- защищенные от инъекций методы работы с данными.

Так же необходимо делать авторизацию, аутентификацию, регистрацию через cookie-файлы. Работу с файлами производить при помощи языка запросов SQL. Данные хранить в базе данных Ms SQL Server. Спроектировать сайт таким образом, чтобы структура меню была такой, как показано на рисунке 1



Рисунок 1 – Структура меню сайта

# 2 Описание программы

# 2.1 Общие сведения

Программа предназначена для предоставления возможности пользователям очень легко и быстро выкладывать свои товары и продавать их, или же наоборот, найти с удобным поиском все что нужно и приобрести.

Система поддерживает 3 (User, Moder, Admin) роли и режим гостя. Гость может только смотреть товары, страницы пользователей, пользоваться навигацией. User может то же, что и гость, а также добавлять, изменять и удалять свои товары, покупать товары других пользователей. Moder может то же, что и User, а также удалять, изменять товары других пользователей. Admin может то же, что и Moder, а так же удалять, изменять и добавлять пользователей, так же у администратора появляется его специальное меню, в котором он может посмотреть краткую статистику сайта и добавить новые категории и новых пользователей.

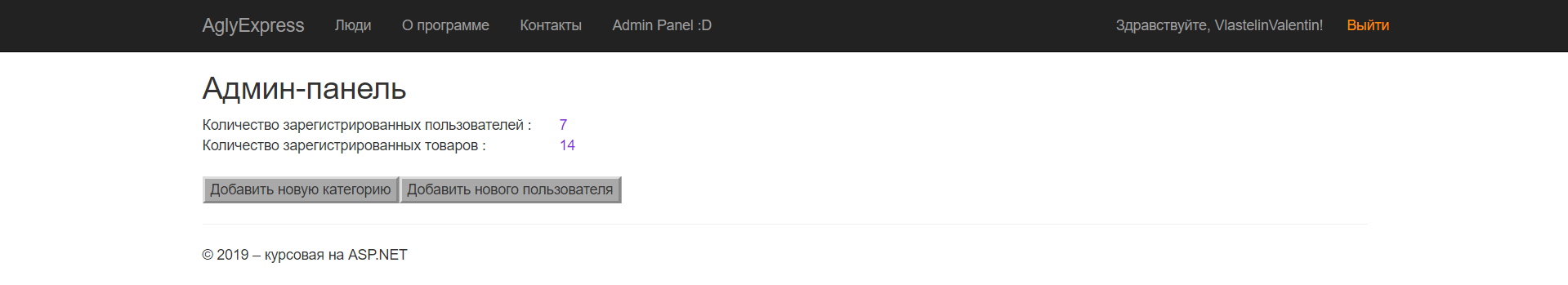


Рисунок 2 – Панель администратора

При покупке товара, он в свою очередь удаляется с базы данных товаров, для того, чтобы данный товар больше ни у кого в списке не появлялся, после чего он переносится в специальную таблицу совершенных покупок - Purchase

Авторизация, регистрация и аутентификация реализованы 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.

Имеется множество способов поиска, недавно был введен новый для товаров, который умеет работать с любыми фильтрами (вообще любыми) код, фильтрующий эти товары на основе этих фильтров можно увидеть на рисунке 3, а само его представление на рисунке 4.

Данное меню имеет несколько фильтров, количество и виды которых можно менять так, как это будет нужно. Не обязательно вписывать что-либо для конкретизации запроса в каждую ячейку, система сама понимает, какие данные введены(по ним происходит фильтрация), а какие остались пустые (они не добавляются в очередь фильтров, и никак не влияют на результат вывода). Например, при вводе значения 10,99 в поле «Минимальная цена» (ничего не вводя в другие поля) выведутся все товары, цена которых больше, либо равна 10,99.

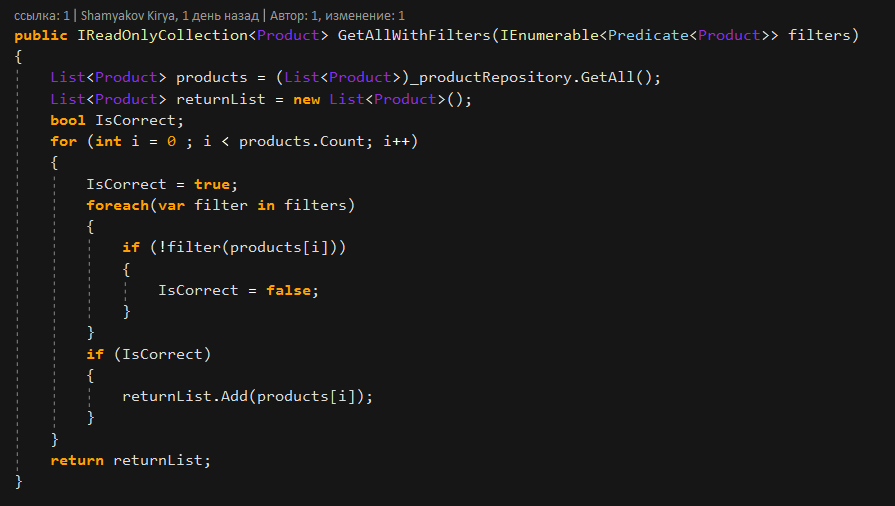


Рисунок 3 – Метод, фильтрующий коллекцию товаров

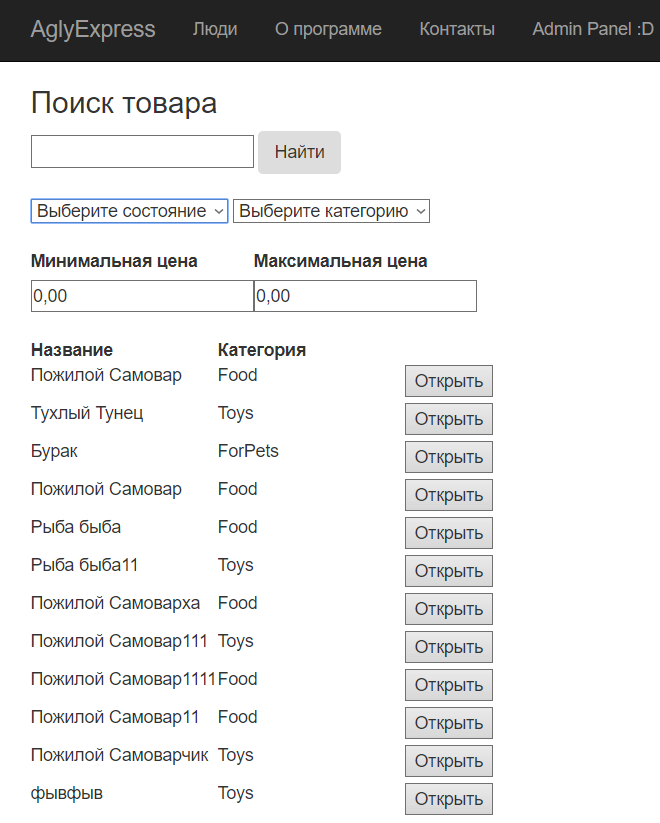


Рисунок 4 – Новый, сложный поиск товаров

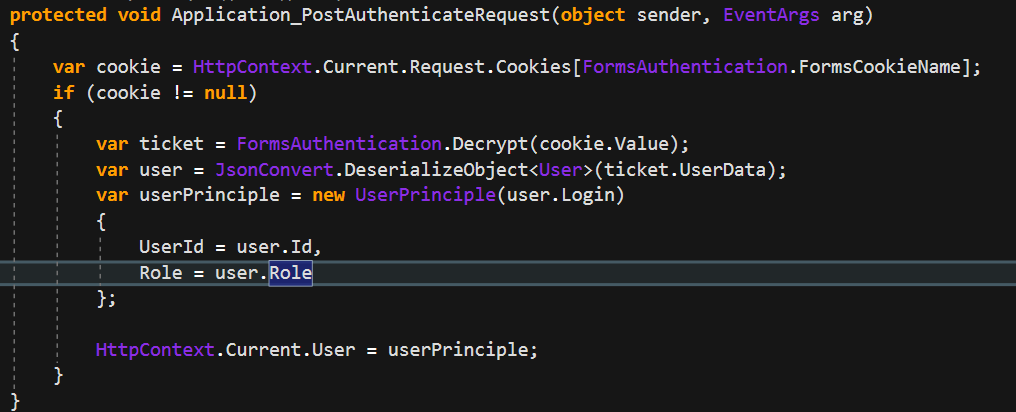


Рисунок 5 – Метод получения cookie браузера

# 2.2 Функциональное назначение

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

# 2.3 Описание логической структуры



Рисунок 6 – Логическая структура приложения

На рисунке 6 показана логическая структура классов, работающих с данными. Все разбито на сервис, репозиторий и контекст для реализации паттерна Repository, что позволяет полностью абстрагироваться от типа подключения к базе данных, в результате чего приложение может работать с любым типом баз данных(Mongo db, Oracle, MySQL).

Описание методов:

GetAll() – получает все сущности и возвращает их.

GetById(int id) – получает сущность по номеру в базе данных, в противном случае возвращает default(T).

Save(T obj) – сохраняет сущность в базе данных.

DeleteById(int id) – удаляет сущность из базы данных.

Edit(T obj) – перезаписывает данные о сущности в базе данных.

Также есть еще множество методов, таких как GetByCategoryId, GetIdByName, но описывать их нету необходимости, так как здесь описаны лишь базовые методы, остальные работают по аналогии.

# 2.4 Используемые технические средства

Данный проект разрабатывался на персональном компьютере со следующими характеристиками:

- операционная система Windows

- процессор Intel Core i5 7300HQ,

- 12288 МБ оперативной памяти,

- 2 ТБ свободного места на диске,

- жесткий диск со скоростью 7200 об/мин,

- видеоадаптер NVIDIA GeForce GTX 1050 Ti 4 ГБ и Intel HD Graphics 630 в качестве встроенного.

Периферийные устройства:

- манипулятор мышь (работа с пользовательским интерфейсом)

- клавиатура (ввод данных)

- монитор (отображение пользовательского интерфейса)

# 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.

Каждый товар в магазине содержит:

Таблица 1 – Поля товара

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Класс | Свойство | Тип |
| 1 | 2 | 3 | 4 |
| Название товара | Product | Name | string |
| Описание товара | Product | Description | string |
| Цена товара | Product | Price | decimal |
| Дата создания товара | Product | CreationDate | DateTime |

Продолжение таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Дата последнего изменения товара | Product | LastModifiedDate | DateTime |
| Категория товара | Product | Category | Category |
| Номер категории | Category | Id | int |
| Название категории | Category | Name | string |
| Автор товара | Product | Author | User |
| Номер пользователя | User | Id | int |
| Логин пользователя | User | Login | string |
| Номер телефона пользователя | User | PhoneNumber | string |
| Почта пользователя | User | Email | string |
| Местонахождение товара | Product | Location | string |
| Состояние товара | Product | State | State |
| Номер состояния | State | Id | int |
| Название состояния | State | Name | string |

# 2.7 Выходные данные

Выходными данными являются представления, созданные контроллерами в зависимости от запросов пользователей.

# 3 Руководство оператора

# 3.1 Назначение программы

Программа предназначена для автоматизации 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,

Периферийные устройства:

- манипулятор мышь (работа с пользовательским интерфейсом)

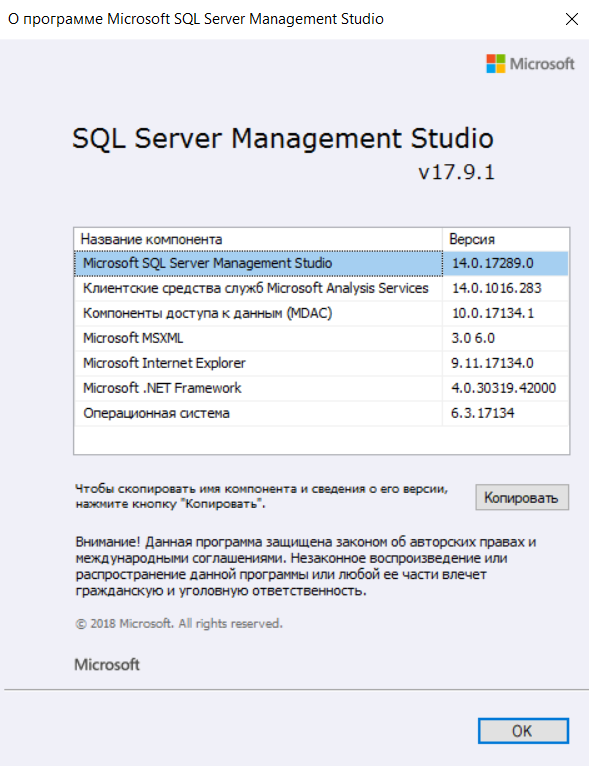
Данный проект работает с MsSQL Server. Его версия, на которой все точно работает указана на рис.7, поэтому для корректной работы программы рекомендую использовать эту версию или более позднюю.

Рисунок 7 – Версия MsSQL Server

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.8)

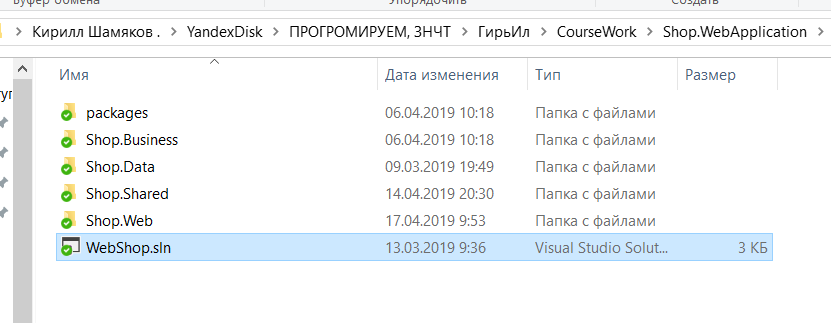


Рисунок 8 – Папка проекта

В пространстве имен Shop.Data.Constants необходимо заменить строку подключения(Рис. 8)

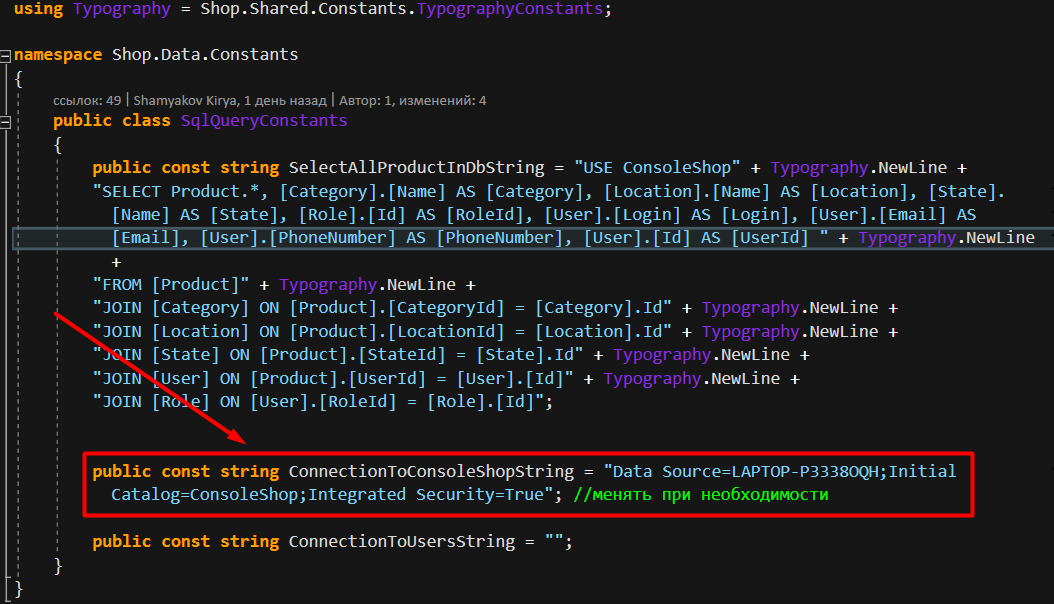


Рисунок 9 – строка подключения

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. 9)

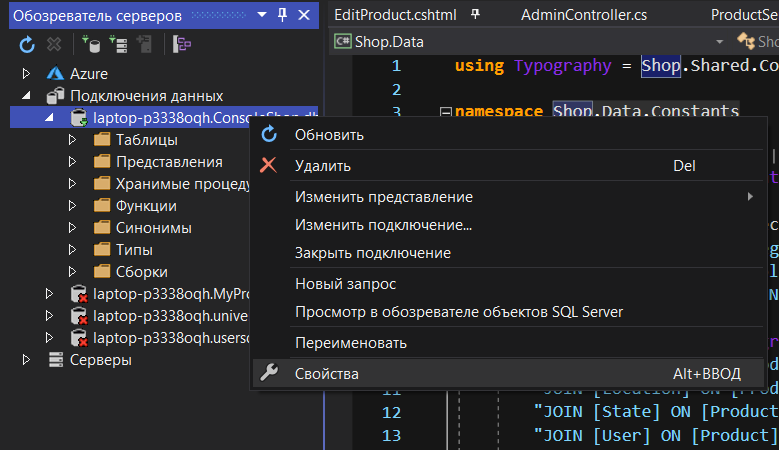
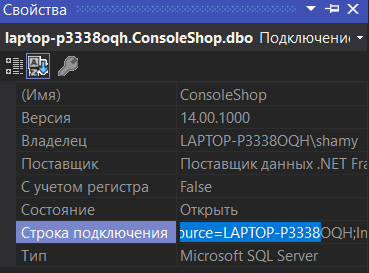


Рисунок 10 – Свойства Базы данных

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)

Таблица 2 – Данные о пользователях

|  |  |  |
| --- | --- | --- |
| Login | Password | Role |
| VlastelinValentin | Павук | Administrator |
| Anatolya | 123 | Moderator |
| Vadimka12 | 343221 | User |
| Valentinka48 | Admin | Moderator |
| Genka15 | 1234 | User |

# 3.3 Выполнение программы

При запуске приложения появится главное меню сайта (рисунок 11), после чего можно воспользоваться поиском товаров (рисунок 12), или просто открыть список всех товаров или товаров в выбранной категории(рисунок 13).

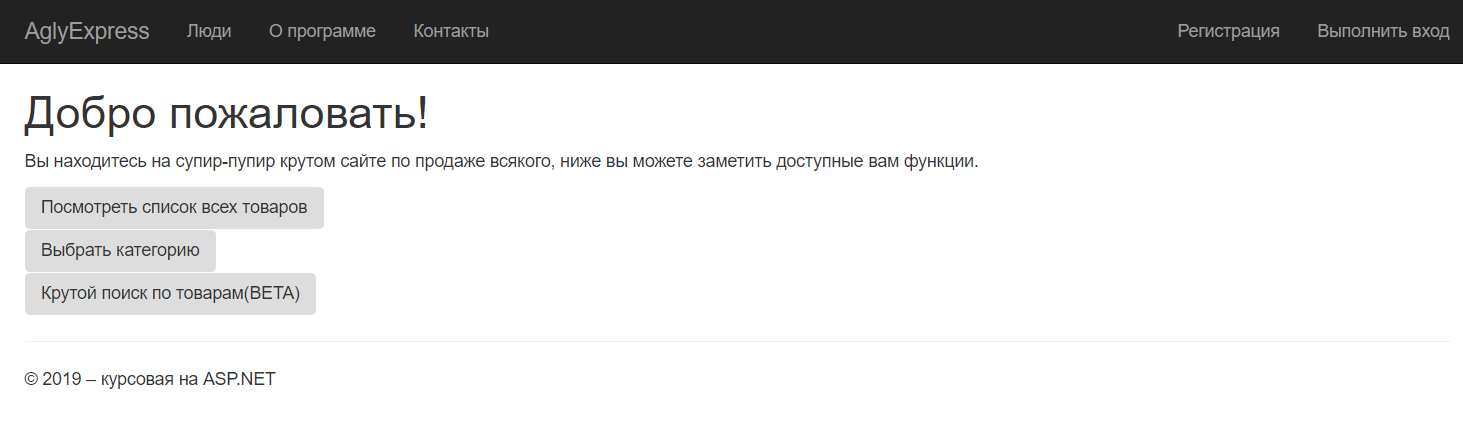


Рисунок 11 – Главная страница сайта

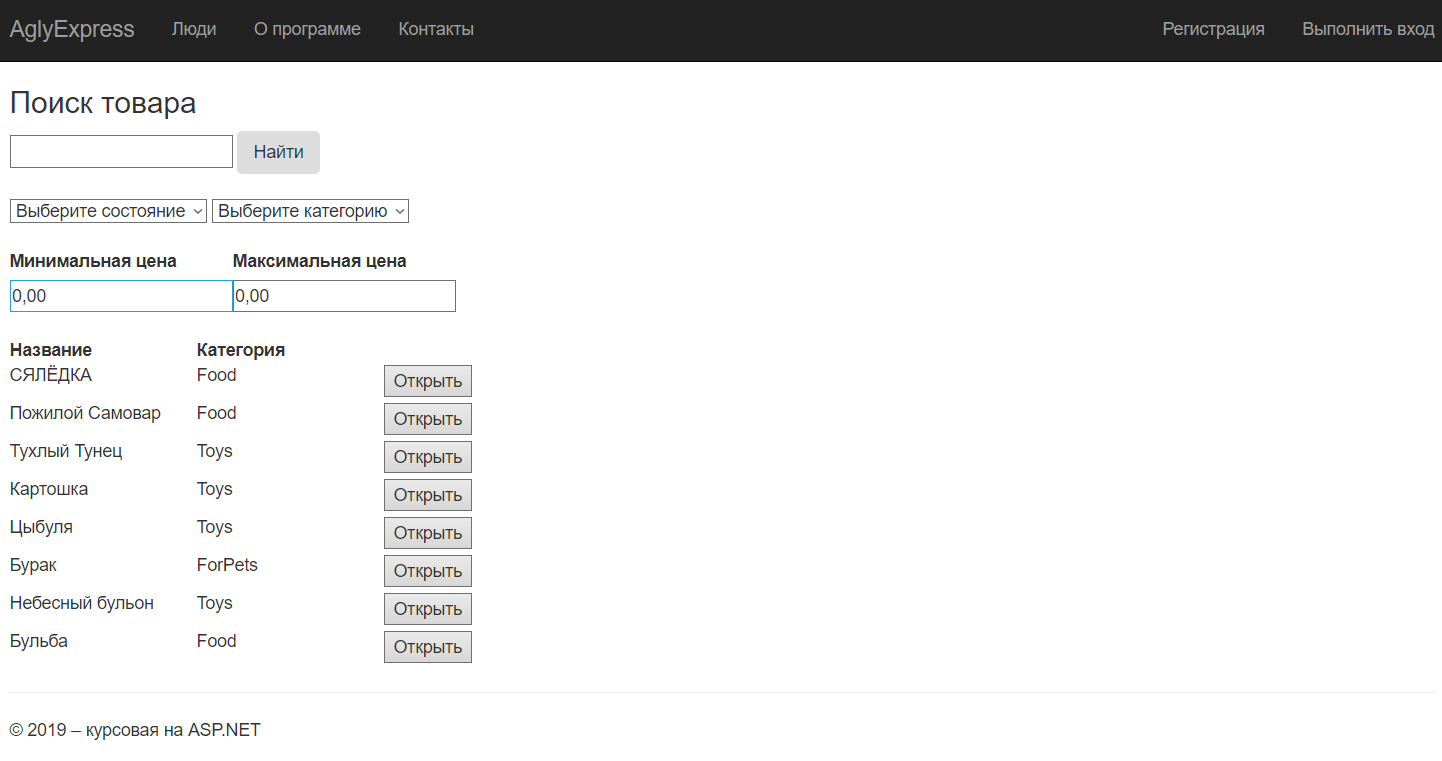


Рисунок 12 – Поиск товара с фильтрами

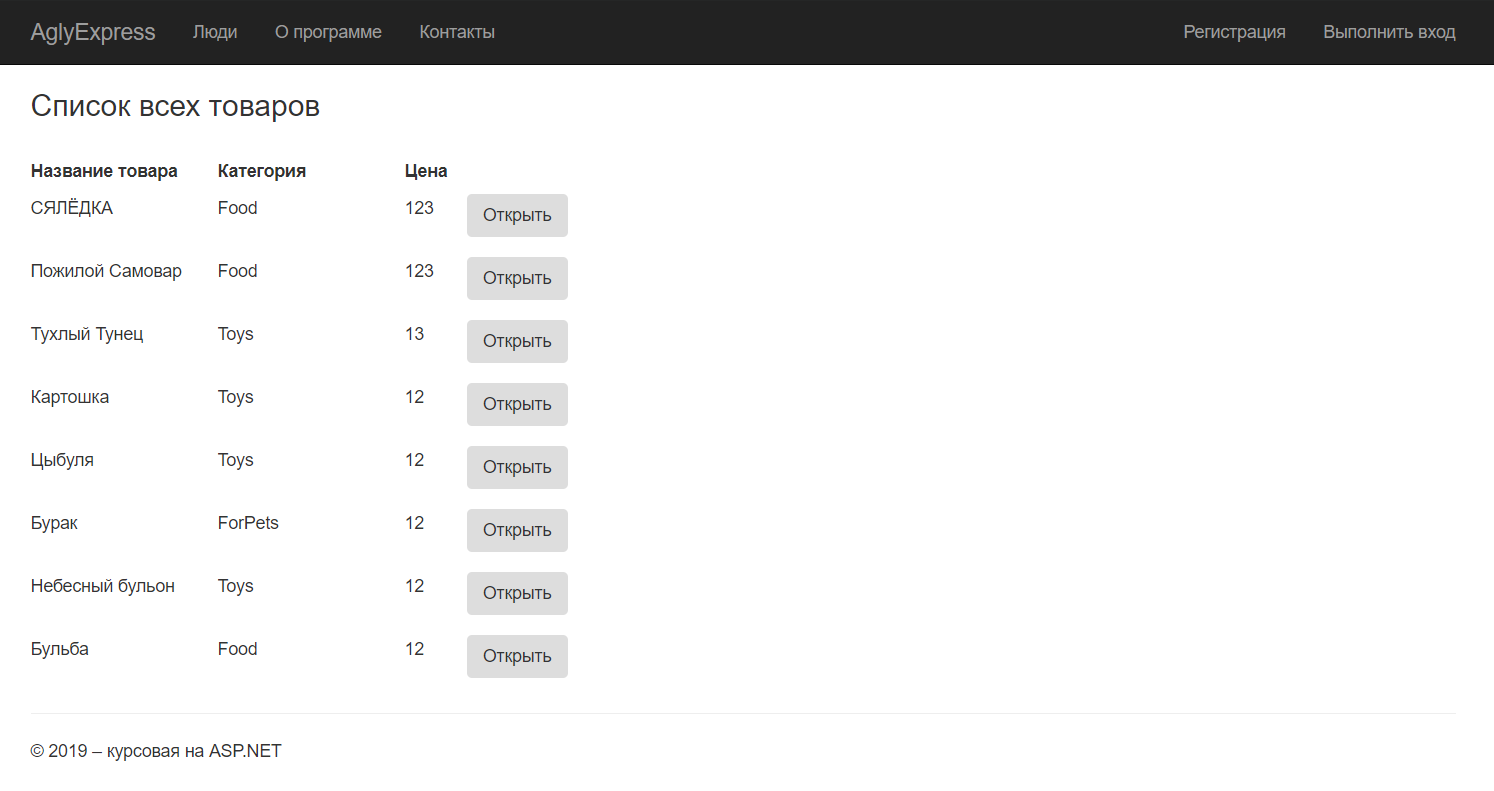


Рисунок 13 – Список всех товаров

Есть раздел «люди» в хедере сайта, в нем можно увидеть меню, показанное на рисунке 14. В этих меню можно открыть профиль пользователя или информацию о товаре(рисунки 15 и 16).

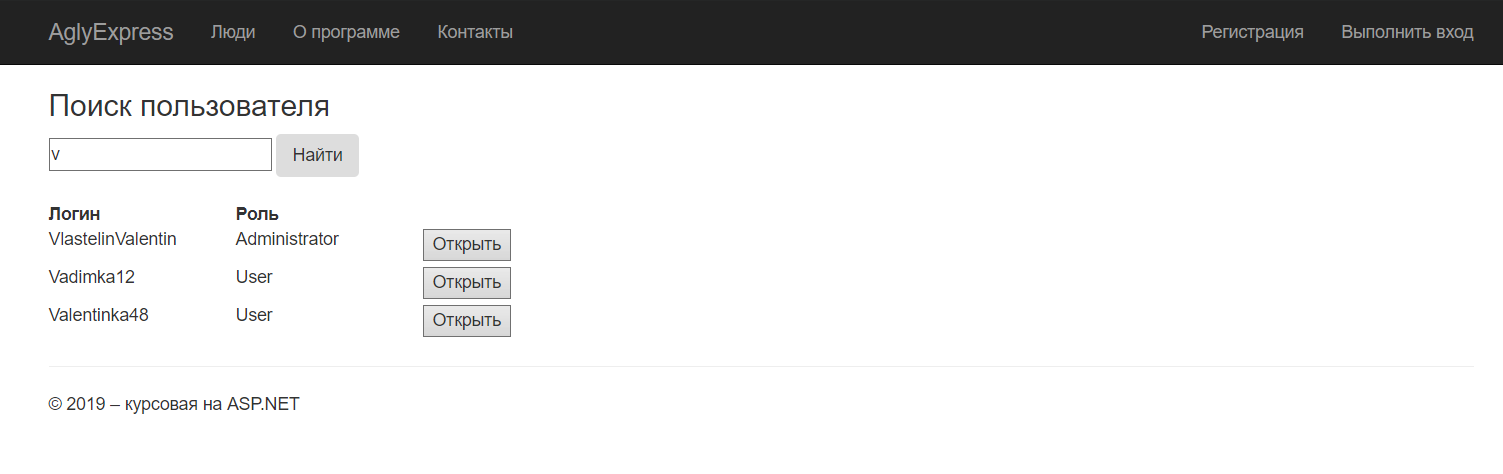


Рисунок 14 – Список пользователей

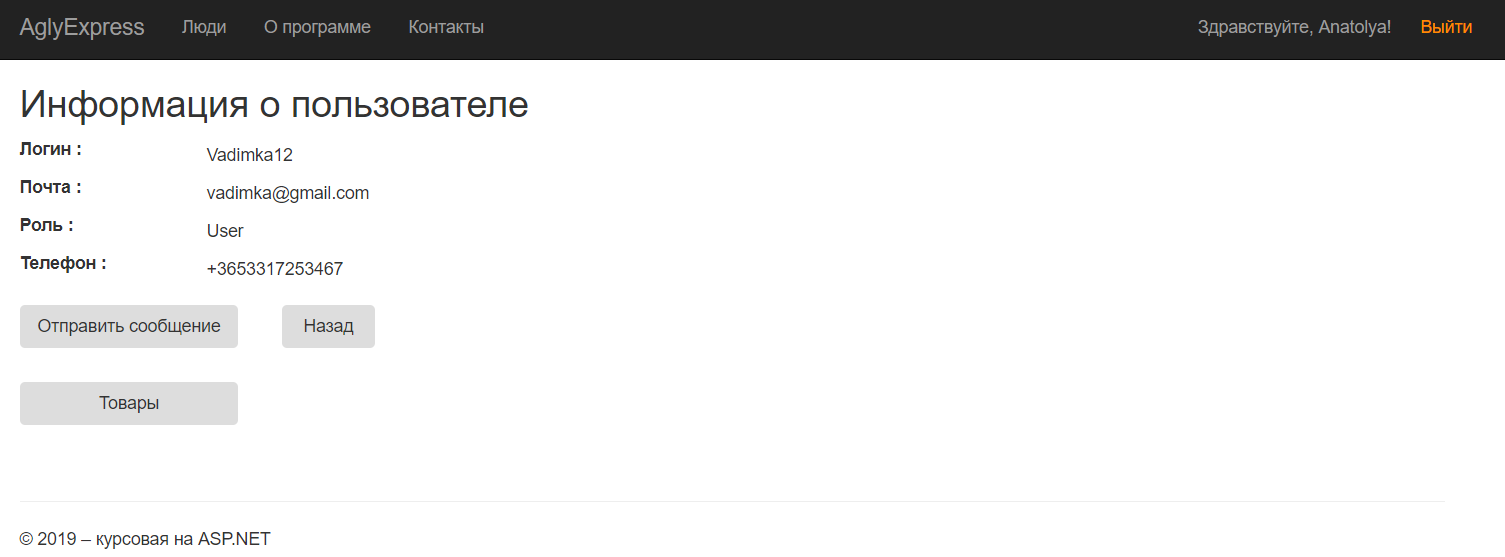


Рисунок 15 – Профиль пользователя

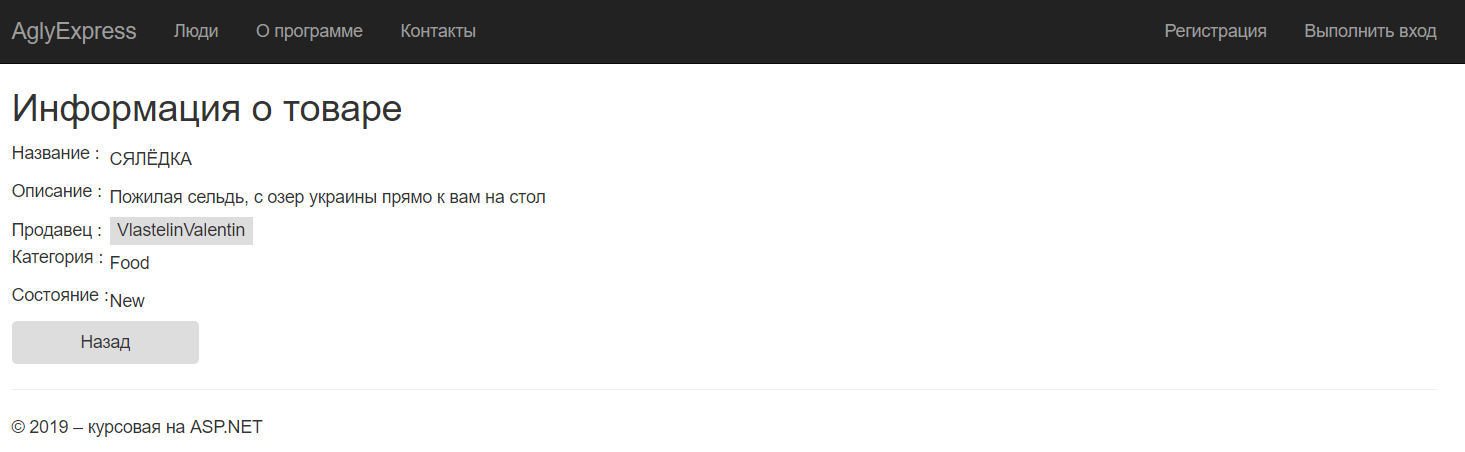


Рисунок 16 – Информация о товаре

В информации о товаре можно перейти на страницу его автора, кликнув по его логину, а также в профиле любого пользователя можно посмотреть все его товары, кликнув по меню «Товары». В хедере сайта доступны меню «Контакты» и «О программе», их внешний вид показан на рисунках 17 и 18.

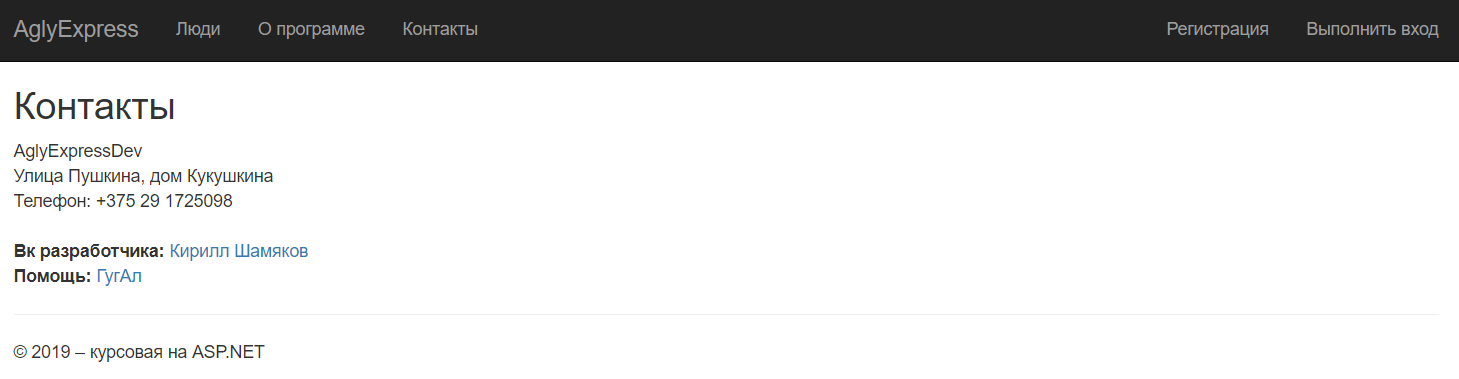


Рисунок 17 – Меню «Контакты»

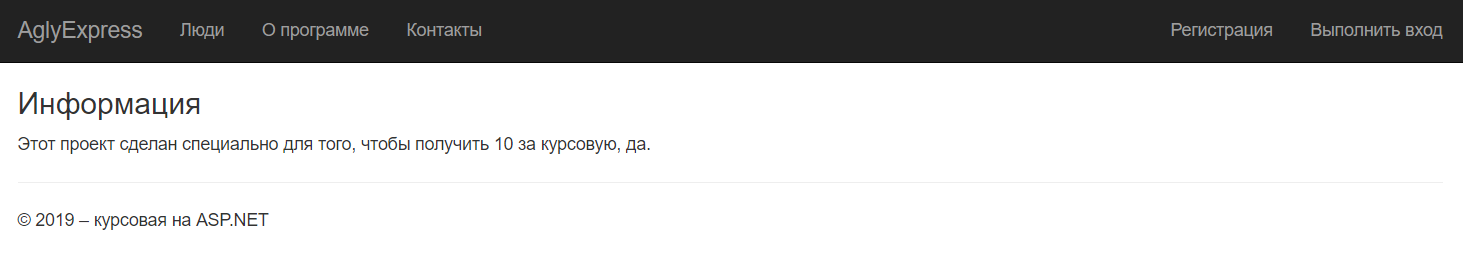


Рисунок 18 – Меню «О программе»

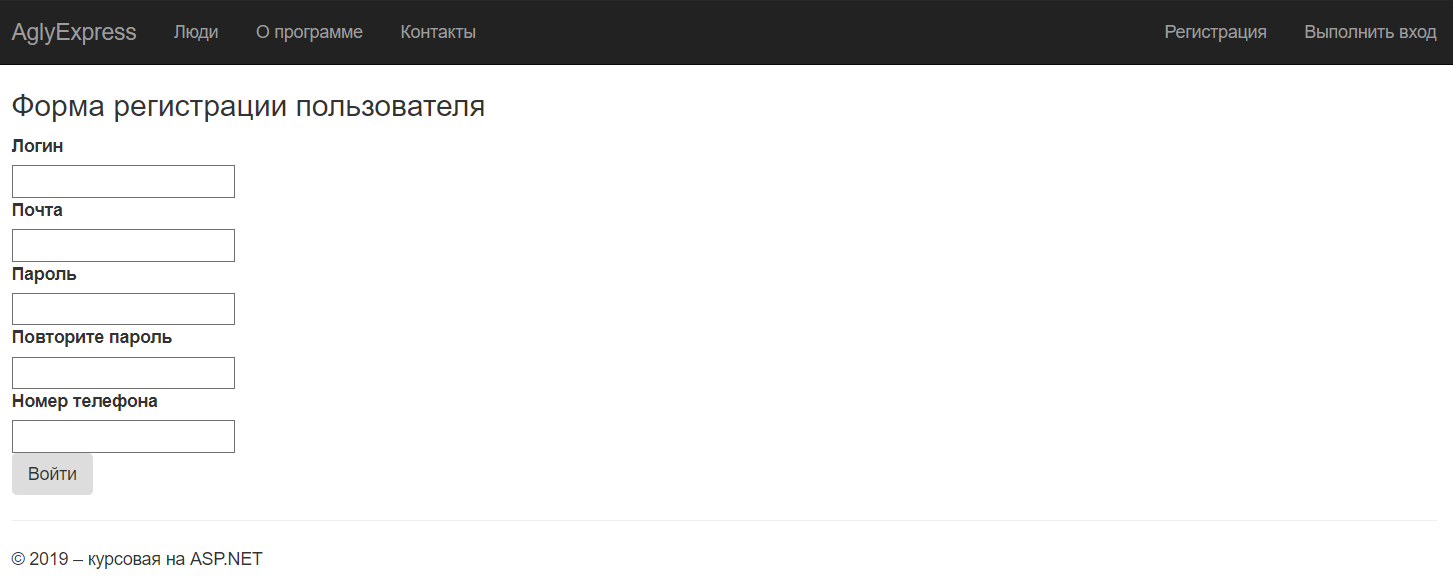


Рисунок 19 – Регистрация

Для более широкого спектра возможностей необходимо произвести аутентификацию, для этого есть разделы «Регистрация» и «Выполнить вход» в хедере. Их отображение можно увидеть на рисунках 19 и 20.

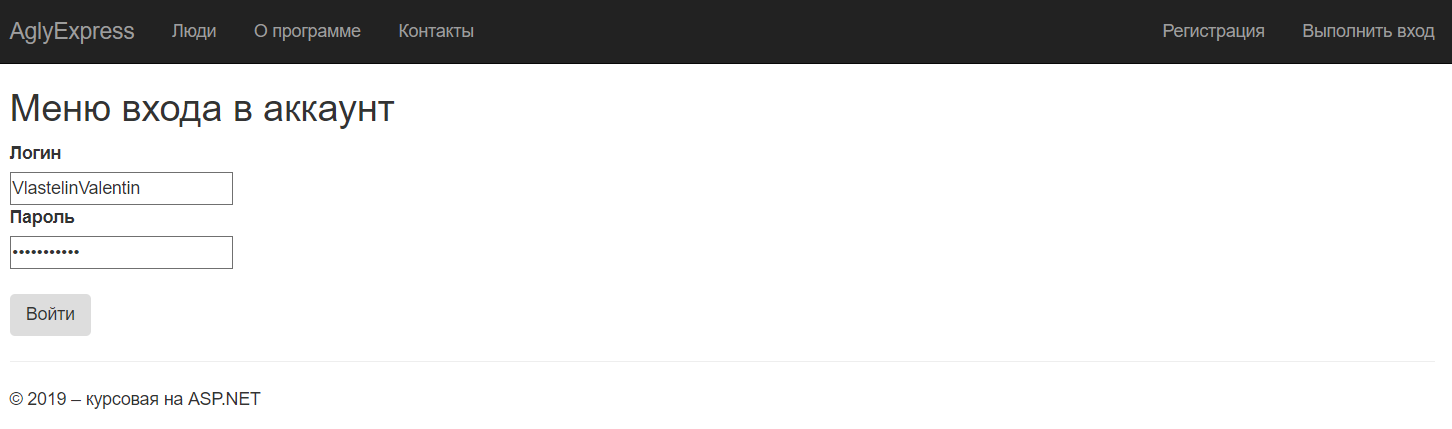


Рисунок 20 – Вход в систему

После входа появляется возможность покупать товары других пользователей(рисунок 21) а также добавлять новые(рисунок 22), нажав на кнопку, находящуюся в личном кабинете(рисунок 23). Добавленные товары можно редактировать или даже удалить. Кнопки дополнительных действий с товарами или пользователями появляются сами, при получении определенных прав, или при входе в систему.

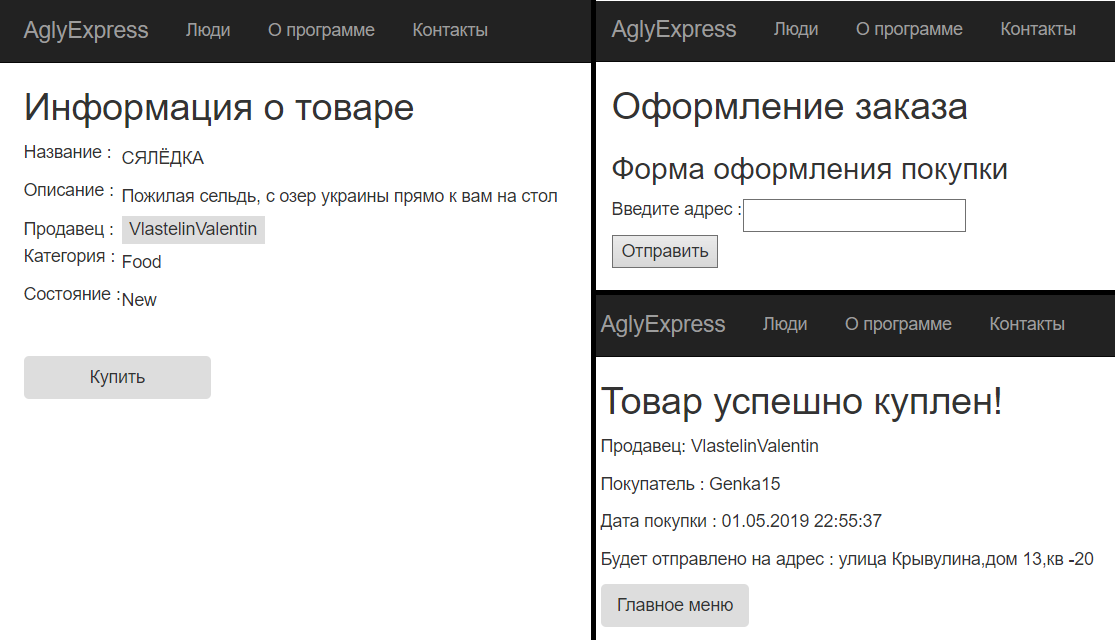


Рисунок 21 – Процесс покупки товара

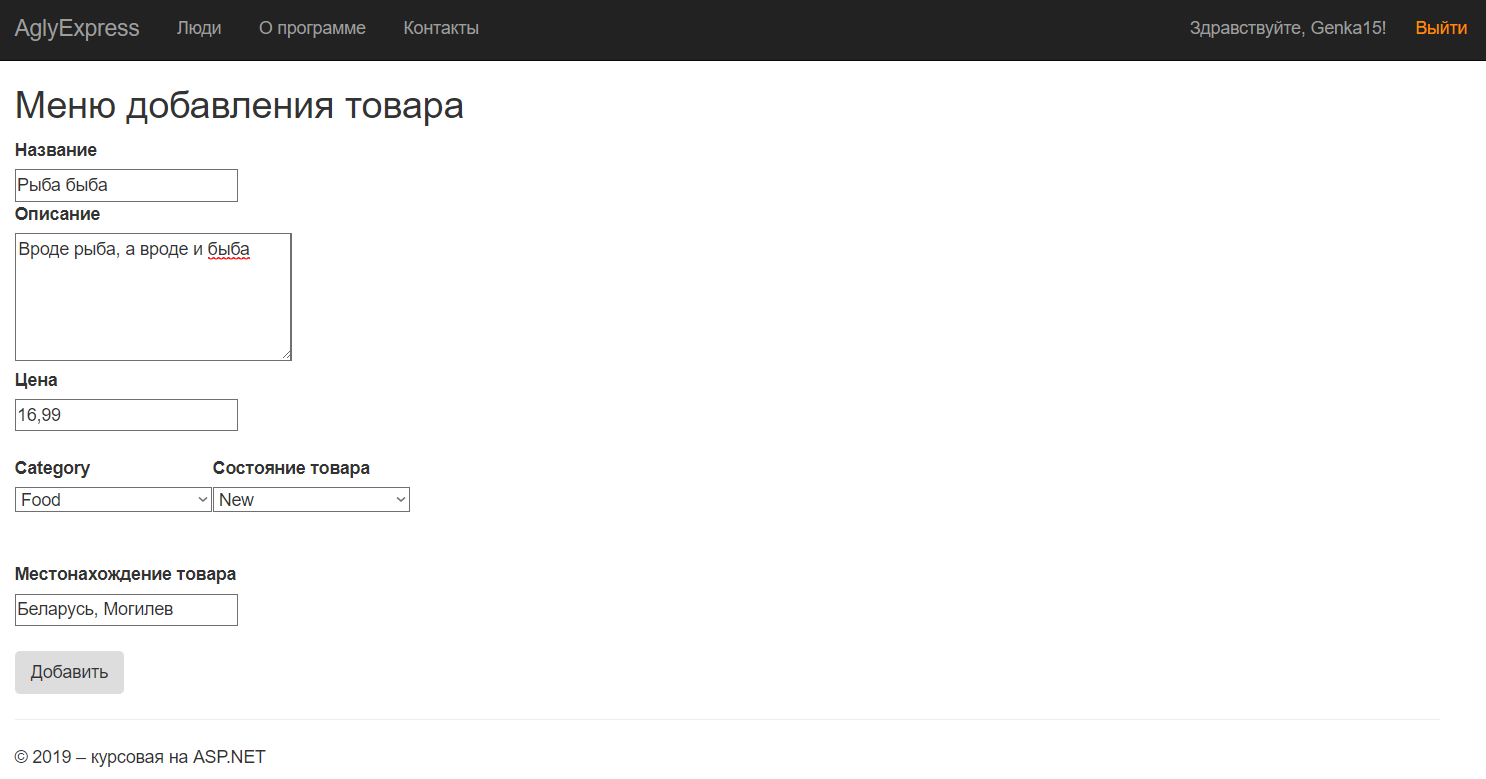


Рисунок 22 – Выставление своего товара

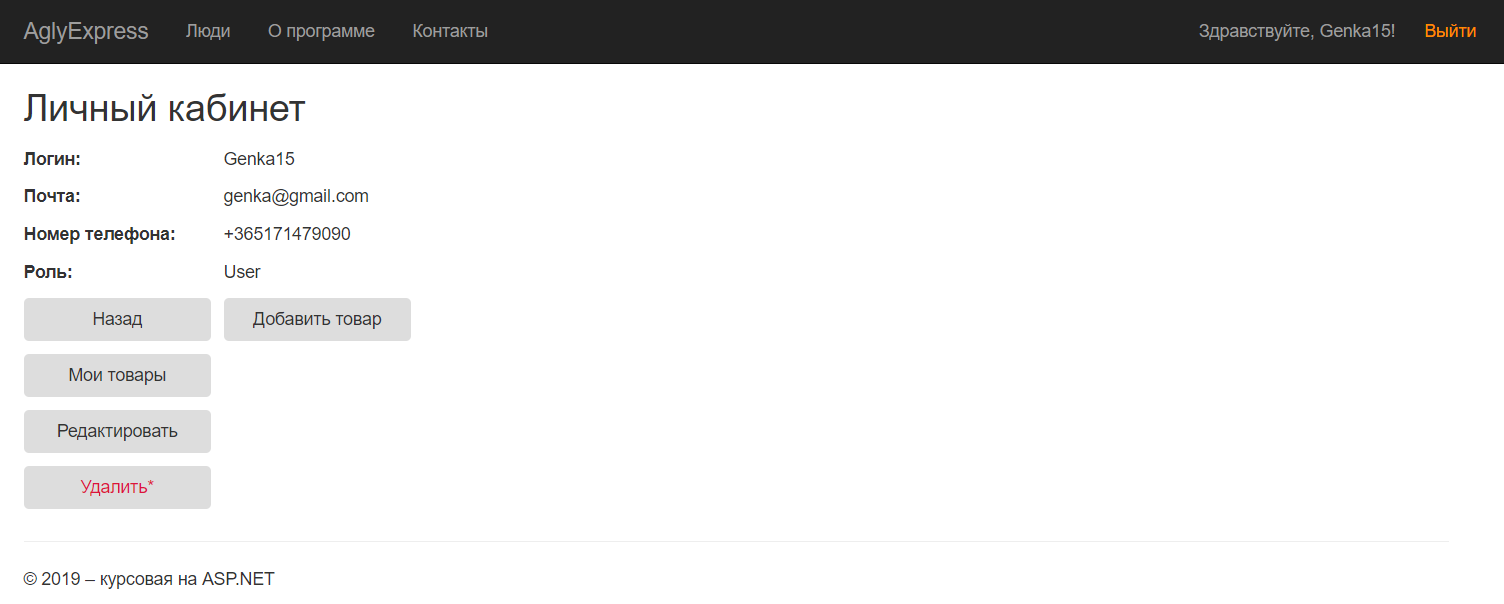


Рисунок 23 – Личный кабинет

При входе в систему за модератора, у товаров других пользователей появится кнопка «Редактировать» и «Удалить»(рисунок 24), а при входе за администратора тоже самое появится и у пользователей(рисунок 28).

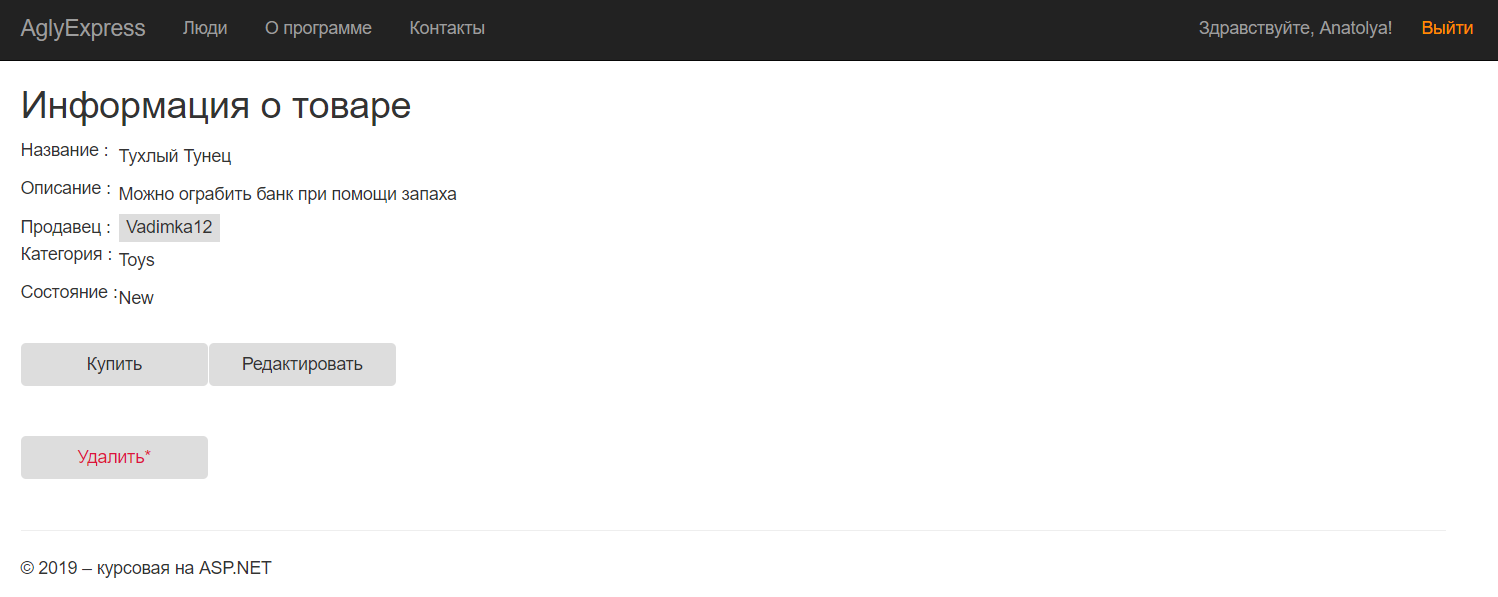


Рисунок 24 – Страница товара у модератора

и администратора

У администратора также есть его личная панель, имеющая краткую статистику сайта и несколько полезных функций(Рисунок 25). К этим функциям относится добавление пользователя(рисунок 26) и новой категории (рисунок 27)

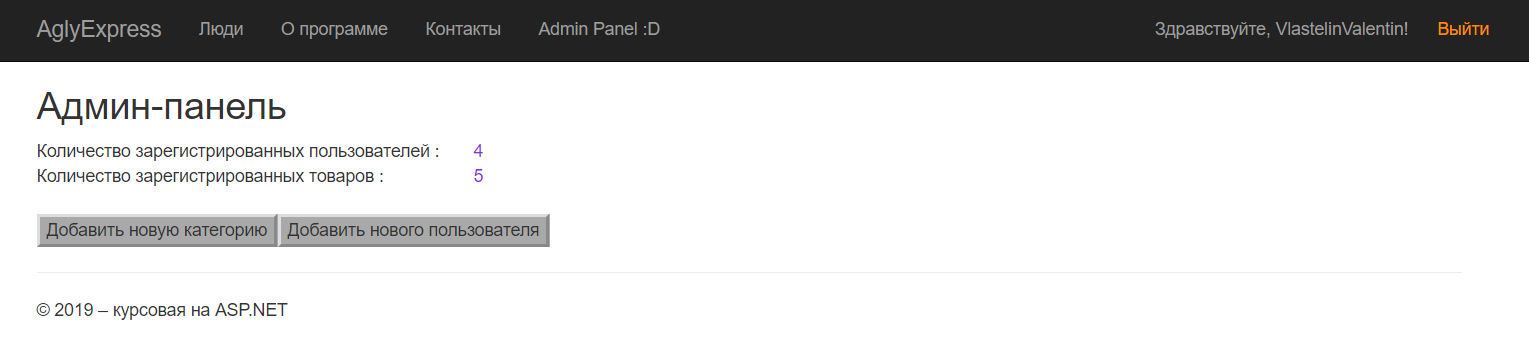


Рисунок 25 – Админ-панель

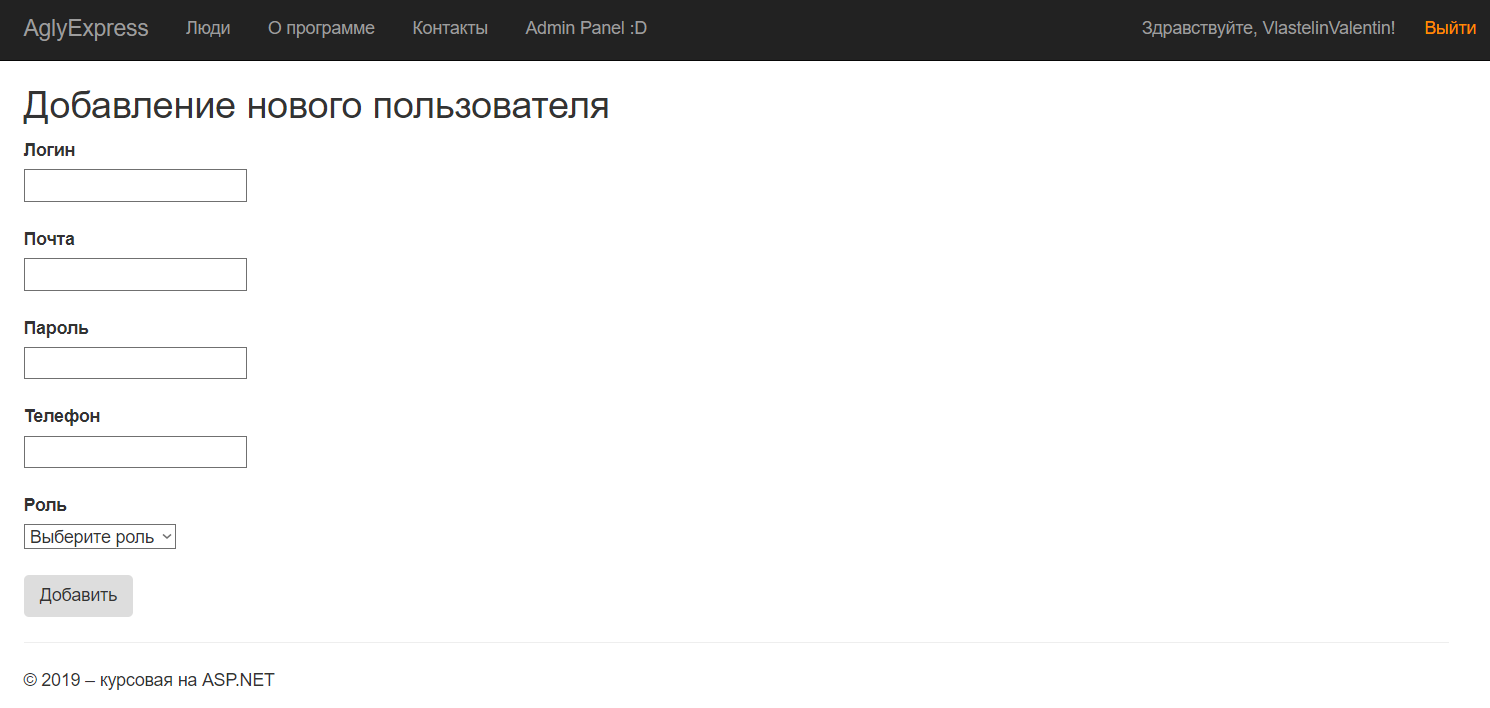


Рисунок 26 – Добавление пользователя

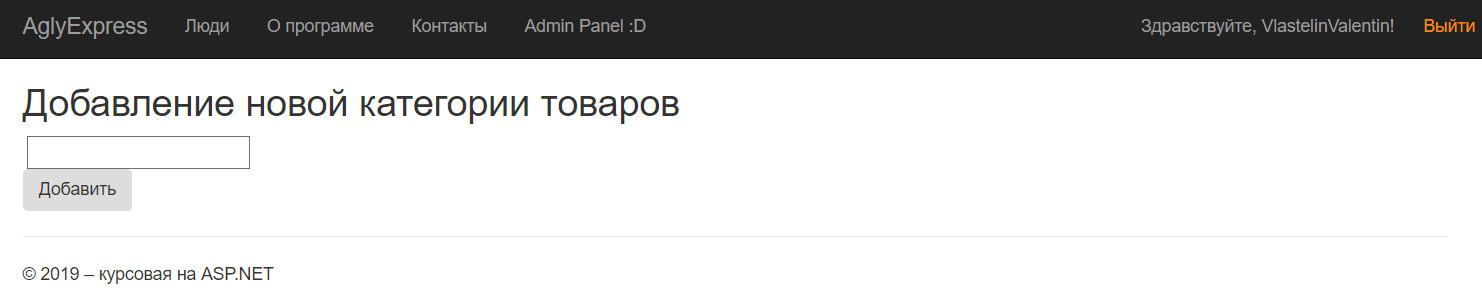


Рисунок 27 – Добавление новой категории

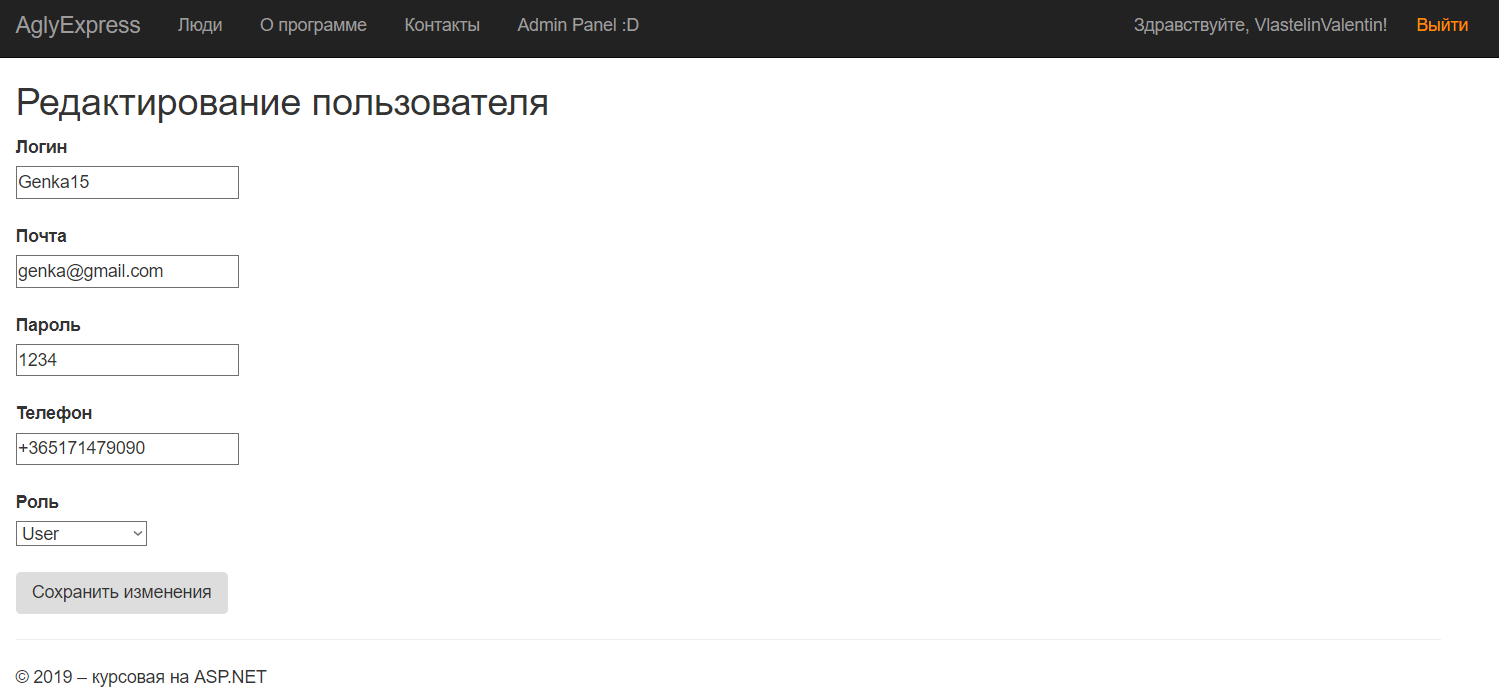


Рисунок 28 – Редактирование пользователя

# 3.4 Сообщения оператору

Большинство сообщений и уведомлений в данном проекте производятся с помощью этого представления. Каждый использующий это представление метод просто дает определенное значение динамической переменной ViewBag.Massage(рис.29).

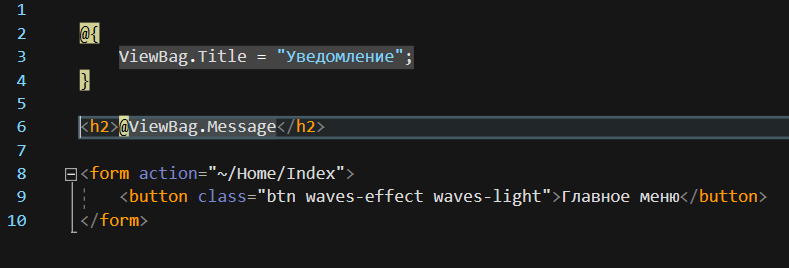
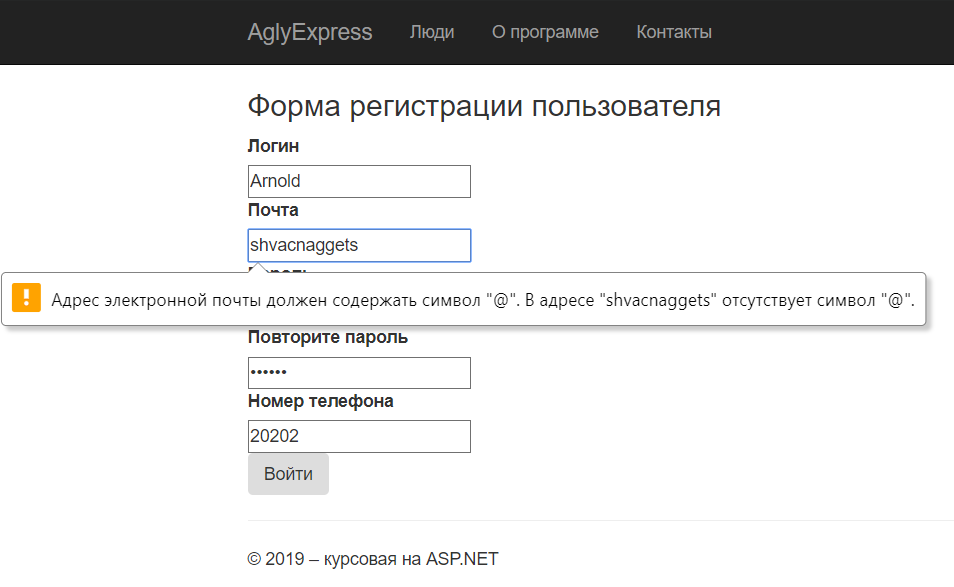


Рисунок 29 – код страницы уведомлений

При использовании приложения вы сможете увидеть следующие уведомления :

Рисунок 30 – Уведомление о некорректном адресе электронной почты

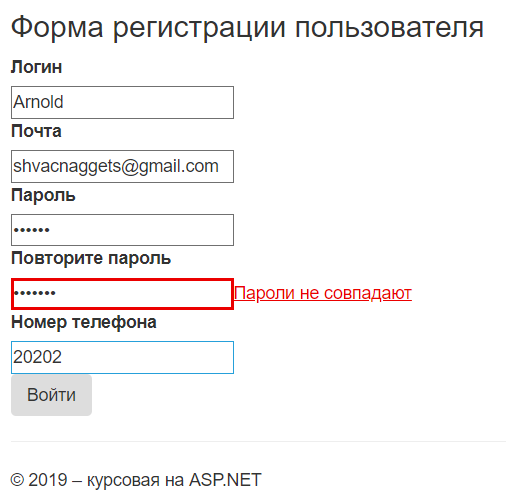


Рисунок 31 – Уведомление при несовпадении паролей

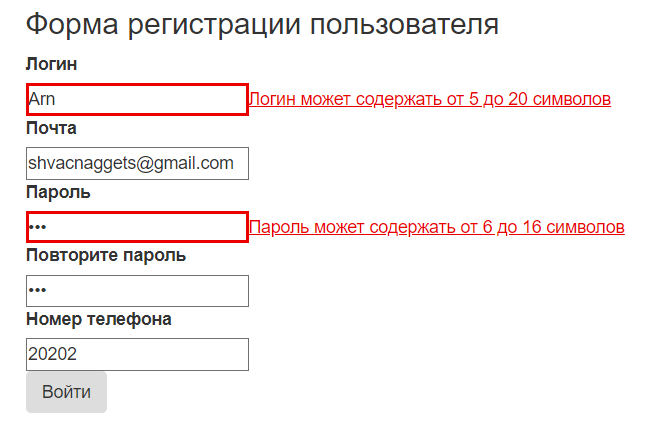


Рисунок 32 – Уведомления при некорректной длине строк

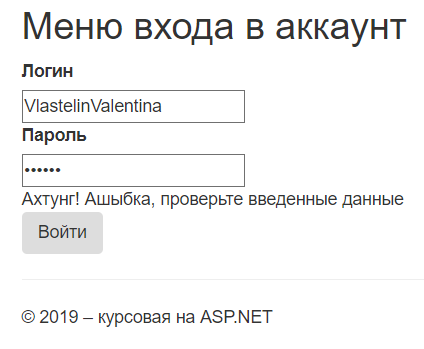


Рисунок 33 – Уведомление при неправильном вводе данных для входа в систему

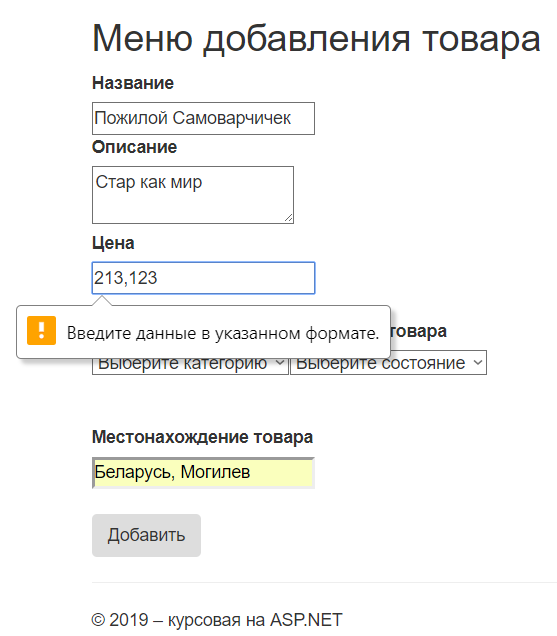


Рисунок 34 – Уведомление при вводе некорректной цены

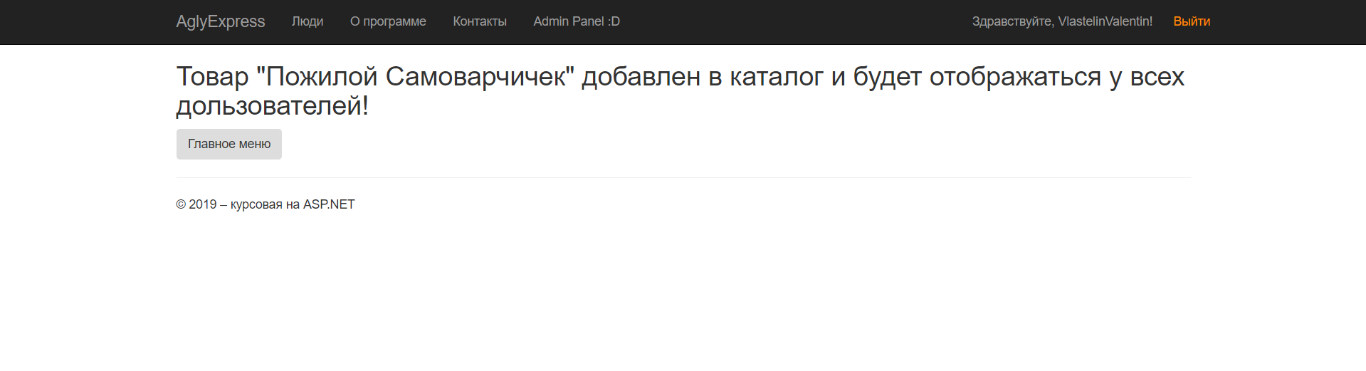


Рисунок 35 – Уведомление при добавлении товара

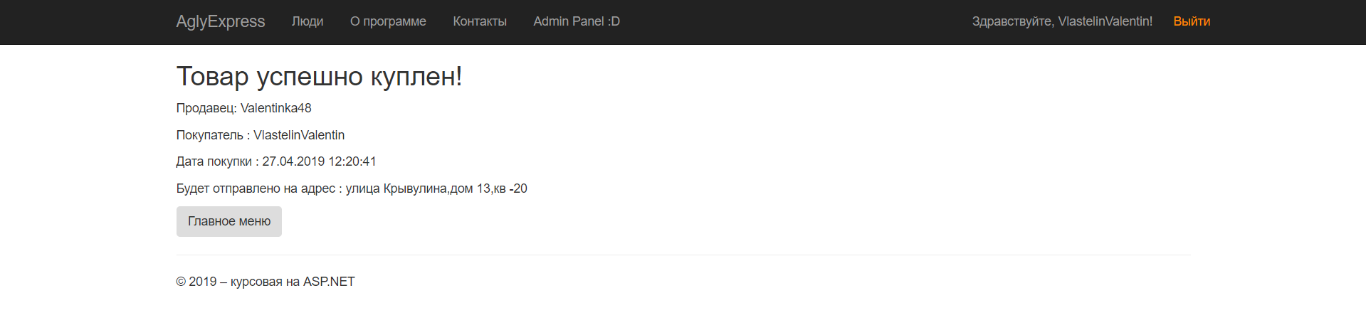


Рисунок 36 – Уведомление при покупке товара

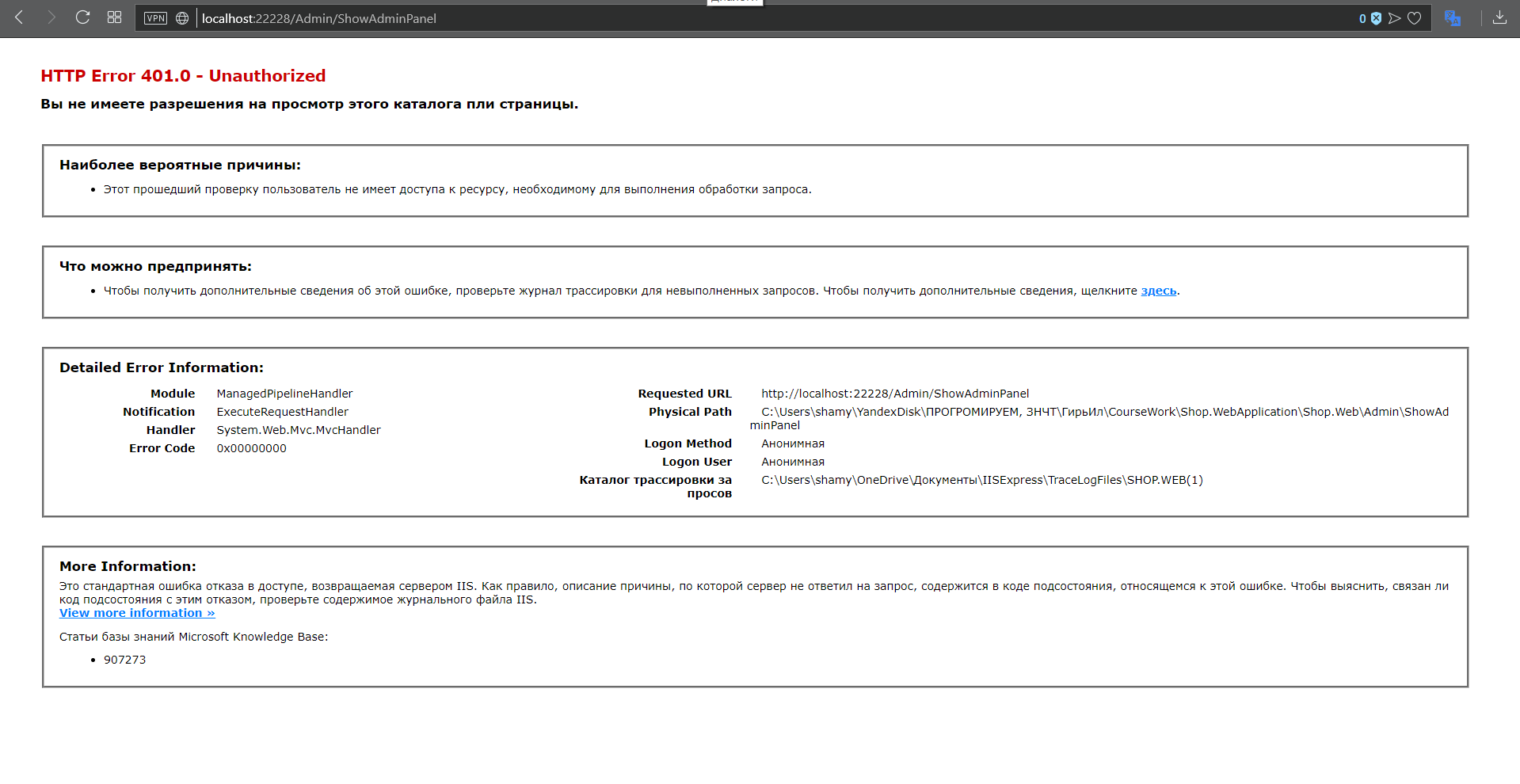
Так же, при попытке пользователя, не имеющего необходимых прав, вручную написать URL закрытого для него ресурса, он будет перенаправлен на следующую страницу(рис.18)

Рисунок 37 - Сообщение системы безопасности

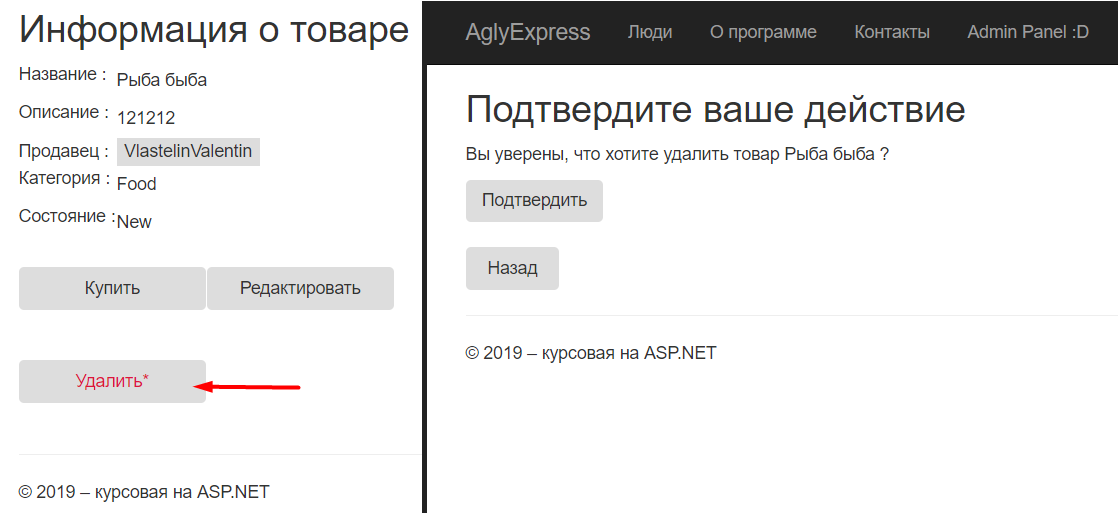


Рисунок 38 – Подтверждение действия

Уведомление, показанное на рис. 38 имеются на всех действиях, связанных с удалением чего-либо.

Выше перечислены лишь основные виды уведомлений, в самом проекте же их несколько больше.

# 4 Тестирование

Для тестирования данного веб-приложения необходимо запустить юнит-тест, находящийся в решении данного проекта (рисунок 39). Как можно увидеть на рисунке 40, он пройден успешно и сервис работает вполне корректно.

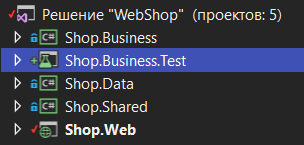


Рисунок 39 – Сборка юнит-теста в обозревателе решений

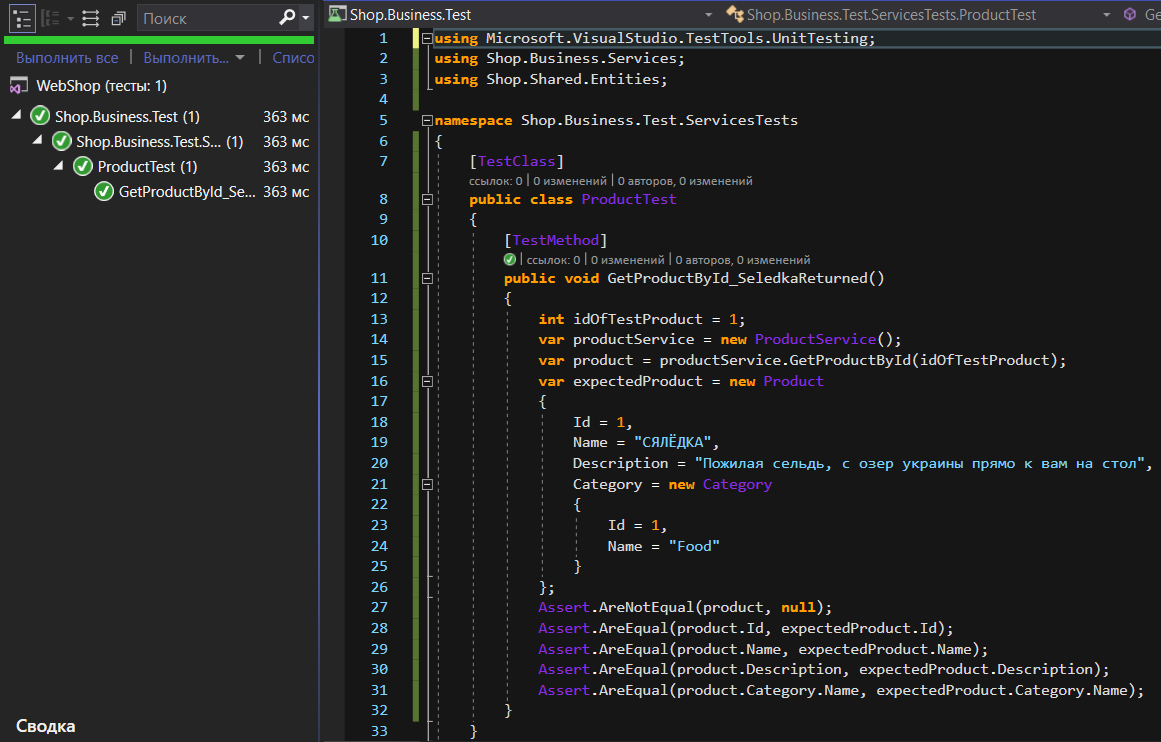


Рисунок 40 – Юнит-тест сервиса

# 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.

К достоинствам можно 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 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, на данный момент которых используется лишь половина), 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 А

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**

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**

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);

}

}

}