МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники (ФИТР)

Кафедра программного обеспечения информационных систем и технологий

**КУРСОВАЯ РАБОТА**

по дисциплине: ”Языки программирования”

на тему: ***”Многостраничный сайт E-Cosmetics”***

Выполнили**:** ст. гр. 10701219 Колосов А.А.,Кузьмич И.В.

Приняла**:** ст**.** преподаватель Лапанович И.О.

Минск 2020

Белорусский национальный технический университет

Кафедра программного обеспечения информационных систем и технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту (работе)**

**по дисциплине Языки программирования**

Тема: ***Многостраничный сайт “E-Cosmetics”***

**Исполнители**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Колосов А.А., Кузьмич И.В. (фамилия, инициалы)

(подпись)

**Студенты второго курса группы 10701219**

**Руководитель** : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. преподаватель Лапанович И.О.(фамилия, инициалы)

(подпись)

Минск 2020

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 3](#_Toc58771962)

[ВВЕДЕНИЕ 4](#_Toc58771963)

[1 ТЕОРЕТИЧЕСКИЙ ВОПРОС 6](#_Toc58771964)

[1.1 История платформы .NET 6](#_Toc58771965)

[1.2 Технологии использованные при создании программы 6](#_Toc58771966)

[1.2.1 Объектно-ориентированное программирование 6](#_Toc58771967)

[1.2.2 ASP.NET Core 7](#_Toc58771968)

[1.2.3 Aсинхронное программирование 8](#_Toc58771969)

[1.2.4. Классы 8](#_Toc58771970)

[1.2.5. Model-View-Controller 9](#_Toc58771971)

[1.3 История создания веб программирования 9](#_Toc58771972)

[2 Многостраничный сайт «E-Cosmetics» 11](#_Toc58771973)

[2.1 Описание задания 11](#_Toc58771974)

[2.2 Описание функционала сайта 11](#_Toc58771975)

[2.3 Блок схемы алгоритмов 11](#_Toc58771976)

[2.4 Руководство пользователя 14](#_Toc58771977)

[2.5 Тестирование программы 15](#_Toc58771978)

[2.6 Вывод 24](#_Toc58771979)

[ЗАКЛЮЧЕНИЕ 25](#_Toc58771980)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРА 26](#_Toc58771981)

[ПРИЛОЖЕНИЕ 1 – Листинг исходных кодов программы 27](#_Toc58771982)

# ВВЕДЕНИЕ

Первая версия языка разметки гипертекста (HyperText Markup Language, HTML) была разработана в 1991 году сотрудниками CERN. За его основу взят SGML (Standard Generalized Markup Language), стандартный язык структурной разметки. Разметка была логической, то есть не несла никакой информации о внешнем виде документа, а лишь указывала границы и соподчинение его составных частей. Позднее была добавлена возможность выделения текста. HTML 1.0 не был стандартизирован, так как на данный момент существовал только один браузер, Mosaic.

С появлением спецификации HTML 3.2 (версия 3.0 не стала стандартом) начался подъем в web-дизайне. Были реализованы таблицы, разметки математических формул, обтекание текста вокруг объектов. Самым важным обновлением в данной версии HTML была поддержка Каскадных таблиц стиля (Cascading Style Sheets, CSS), позволявших производить графическое описание документа отдельно. Таким образом разрешались противоречия между идеологиями структурной разметки и визуального представления. Текущая версия HTML 4.01 была стандартизирована 24 декабря 1999 года. Она отличается законченностью и полнотой, а также поддерживает усовершенствованную версию CSS.

По мере роста количества и изменения качества документов в Интернет, росли предъявляемые к ним требования. Простота HTML превратилась в его главный недостаток. Ограниченность количества тегов и полное безразличие к структуре документа побудили разработчиков в лице консорциума W3C к созданию такого языка разметки, который был бы не столь сложен, как SGML, и не настолько примитивен, как HTML. В результате появился язык XML, сочетающий в себе простоту HTML, логику разметки SGML и удовлетворяющий требованиям Интернета. Он используется для создания многих web-приложений, в том числе и для мобильных устройств.

Web-программирование – это раздел программирования, ориентированный на разработку динамических web-приложений. Языки web-программирования делятся на серверные (команды которых обрабатываются web-сервером) и клиентские (команды обрабатываются браузером). Для связи с сервером используется интерфейс CGI (англ. Common Gateway Interface — «общий интерфейс шлюза»).

В 1994 году датский программист Расмус Лердорф создал набор сценариев на Perl/CGI для вывода и учёта посетителей его онлайн-резюме, обрабатывающий шаблоны HTML-документов. Лердорф назвал его PHP (англ. Personal Home Page – «личная домашняя страница»). Вскоре функциональности и быстроты Perl, интерпретатора скриптов, перестало хватать, и, спустя год, Лердорф разработал с использованием языка C новый интерпретатор шаблонов PHP/FI (англ. Personal Home Page / Forms Interpreter — «Личная Домашняя Страница / Интерпретатор форм»), который мог становиться частью web-сервера. PHP/FI был размещен в Сети для всеобщего использования, и началось его повсеместное распространение. К концу 1997 года РНР использовался более чем на пятидесяти тысячах сайтов.

Так как исходный код интерпретатора был открыт, то энтузиасты стали заниматься его доработкой, и летом 1998 года появился РНРЗ – разработка Зива Сураски и Энди Гутманса. РНРЗ был создан практически «с нуля», так как его авторы сочли код предыдущих версий недостаточно эффективным. Кроме того, РНРЗ стал весьма легко расширяемым продуктом. В 2000-м году вышла разработанная компанией Zend Technologies четвертая версия интерпретатора РНР, дополненная множеством новых функций. В настоящее время именно она является наиболее распространенной — РНР используется более чем на 20% сайтов Сети. В это же время, группа разработчиков компании Netscape под руководством Брендона Айха создала язык LiveScript, который впоследствии был переименован в JavaScript и стандартизован Европейской компьютерной ассоциацией.

В 1996 году компания Microsoft выпустила технологию ASP (англ. Active Server Pages — «активные серверные страницы»), позволяющую подключать программные модули во время процесса формирования web-страницы. Относительная популярность ASP основана на простоте используемых при этом языков Visual Basic Script и JScript (аналога языка JavaScript).

Технология ASP получила своё развитие в виде ASP.NET в 2003 году. Разработчики могут писать код для ASP.NET, используя практически любые языки программирования, входящие в комплект программной платформы .NET (C#, Visual Basic.NET, и JScript .NET). В 2005 году была разработана технология AJAX (от англ. Asynchronous JavaScript and XML — «асинхронный JavaScript и XML»). Это новый подход к построению интерактивных пользовательских интерфейсов web-приложений, заключающийся в «фоновом» обмене данными браузера с сервером.

Программирование – это искусство выражать решения задач так, чтобы компьютер мог их решить. Работу программиста можно сравнить с работой переводчика иностранных языков: программист, используя свои знания какого-либо языка программирования, говорит компьютеру что тот должен сделать.

Зачем нужны программы и программисты? Современная цивилизация основана на компьютерных программах. Не зная, как работают эти программы, вы будете вынуждены верить в ”волшебство”, и многие полезные, интересные сферы деятельности будут для вас закрыты. Компьютерные программы применяются в очень широком спектре различных сфер – от программ для персональных компьютеров с графическим интерфейсом, программ для инженерных вычислений и встроенных систем управления до приложений, предназначенных для манипулирования текстами. Программирование представляет собой полезное интеллектуальное упражнение, оттачивающее мыслительные способности. Кроме того, программирование - довольно увлекательное занятие. С помощью программирования также можно разбогатеть и изменить мир к лучшему.

# ТЕОРЕТИЧЕСКИЙ ВОПРОС

## История платформы .NET

В начале 2000-х компания Microsoft объявила о появлении новой платформы .NET. Данная платформа существенно отличается от ранних платформ, так как позволяет разрабатывать программное обеспечение на разных языках программирования и обеспечивает их взаимодействие.Кроме этого, разработчики могут создавать программы на базе Web – их можно использовать не только на обычных компьютерах, соединенных в сеть, но также и на мобильных телефонах.

.NET – это новый взгляд на всемирную паутину в плане разработки, использования средств программирования и проектирования. Одна из особенностей .NET – это то, что платформа не завязана на каком-либо конкретном языке программирования: разработчики могут писать на множестве совместимых языков. Это C#, APL, J#Perl, Fortran Visual C++ .NET и некоторые другие – программисты создают свои приложения на тех языках, какие понимают лучше всего (или какие лучше подходят под решение текущих задач).

Одна из составляющих инициативы .NET – это технология активных серверных страниц ASP.NET (аббревиатура Active Server Page); с её помощью можно относительно быстро программировать приложения на базе Web, взаимодействующие с базами данных и задействующие мощности объектно-ориентированного программирования. Разработчики применяют ASP.NET в связи с тем, что получающиеся приложения обладают лучшей производительностью и защитой.

В 2002 году корпорация Майкрософт выпустила .NET Framework, платформу разработки для создания приложений Windows. Сегодня доступна версия 4.8 платформы .NET Framework, и она по-прежнему поддерживается Майкрософт.

В 2014 году корпорация Майкрософт начала создавать кроссплатформенный преемник платформы .NET Framework с открытым исходным кодом. Эта новая реализация .NET называлась .NET Core, пока не была выпущена версия 3.1. Следующая версия после .NET Core 3.1 — это .NET 5.0, Номер версии 4 был пропущен во избежание путаницы между этой реализацией .NET и .NET Framework 4.8. Слово "Core" было удалено из названия, чтобы указать, что теперь это основная реализация .NET.

## Технологии использованные при создании программы

1. **Объектно-ориентированное программирование**

Объе́ктно-ориенти́рованное программи́рование (ООП)  — [методология программирования](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), основанная на представлении программы в виде совокупности [объектов](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), каждый из которых является экземпляром определённого [класса](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), а классы образуют иерархию наследования.

Идеологически ООП — подход к программированию как к моделированию информационных объектов, решающий на новом уровне основную задачу [структурного программирования](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5): структурирование информации с точки зрения управляемости, что существенно улучшает управляемость самим процессом моделирования, что, в свою очередь, особенно важно при реализации крупных проектов.

Управляемость для иерархических систем предполагает минимизацию избыточности данных (аналогичную [нормализации](https://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0)) и их целостность, поэтому созданное удобно управляемым — будет и удобно пониматься. Таким образом, через тактическую задачу управляемости решается стратегическая задача — транслировать понимание задачи программистом в наиболее удобную для дальнейшего использования форму.

Основные [принципы](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF) структурирования в случае ООП связаны с различными аспектами базового понимания предметной задачи, которое требуется для оптимального управления соответствующей моделью:

* [абстракция](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) для выделения в моделируемом предмете важного для решения конкретной задачи по предмету, в конечном счёте — контекстное понимание предмета, формализуемое в виде класса;
* [Инкапсуляция](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) для быстрой и безопасной организации собственно иерархической управляемости: чтобы было достаточно простой команды «что делать», без одновременного уточнения как именно делать, так как это уже другой уровень управления;
* [Наследование](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) для быстрой и безопасной организации родственных понятий: чтобы было достаточно на каждом иерархическом шаге учитывать только изменения, не дублируя всё остальное, учтённое на предыдущих шагах;
* [полиморфизм](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) для определения точки, в которой единое управление лучше распараллелить или наоборот — собрать воедино.
* Повторное использование кода — [методология проектирования](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) компьютерных и других систем, заключающаяся в том, что система ([компьютерная программа](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), программный модуль) частично либо полностью должна составляться из частей, написанных ранее компонентов.

1. **ASP.NET Core**

ASP.NET Сorе MVC - это инфраструктура для разработки веб-приложений производства Microsoft. которая сочетает в себе эффективность и аккуратность архитектуры "модель-представление-контроллер" (model-view-controller - MVC).

В 2015 году Microsoft заявила о новом направлении для ASP.NEТ и МVС Framework, что в итоге привело к появлению инфраструктуры ASP.NEТ Core МVС.

Платформа ASP.NEТ Core построена на основе .NET Core, которая представляет собой межплатформенную версию .NЕТ Framework без интерфейсов программирования приложений (application programming interface - API), специфичных для Windows.

Господствующей операционной системой по-прежнему является Windows, но веб-приложения все чаще размещаются в небольших и простых контейнерах на облачных платформах. За счет принятия межплатформенного подхода компания Microsoft расширила область охвата .NET, сделав возможным развертывание приложений ASP.NET Core на более широком наборе сред размещения, а в качестве бонуса предоставила разработчикам возможность создавать веб-приложения ASP.NET Core на машинах Linux и macOS.

ASP.NEТ Core - совершенно новая инфраструктура. Она проще, с нею легче работать, и она свободна от наследия, происходящего из Web Forms. Будучи основанной на .NЕТ Core, она поддерживает разработку веб-приложений для ряда платформ и контейнеров. Инфраструктура ASP.NET Core MVC предлагает функциональность первоначальной инфраструктуры ASP.NEТ МVС Framework, построенной поверх новой платформы ASP.NET Core. Она интегрирует функциональные средства, которые ранее предоставлялись Web API, поддерживает более естественный способ генерирования сложного содержимого и делает основные задачи разработки, такие как модульное тестирование.

более простыми и предсказуемыми.

1. **Aсинхронное программирование**

Асинхронность позволяет вынести отдельные задачи из основного потока в специальные асинхронные методы или блоки кода. Особенно это актуально в графических программах, где продолжительные задачи могу блокировать интерфейс пользователя. И чтобы этого не произошло, нужно задействовать асинхронность. Также асинхронность несет выгоды в веб-приложениях при обработке запросов от пользователей, при обращении к базам данных или сетевым ресурсам. При больших запросах к базе данных асинхронный метод просто уснет на время, пока не получит данные от БД, а основной поток сможет продолжить свою работу. В синхронном же приложении, если бы код получения данных находился в основном потоке, этот поток просто бы блокировался на время получения данных.

Ключевыми для работы с асинхронными вызовами в C# являются два ключевых слова: async и await, цель которых - упростить написание асинхронного кода. Они используются вместе для создания асинхронного метода.

1. **Классы**

Класс — тип, описывающий устройство объектов. Объект — это экземпляр класса. Класс можно сравнить с чертежом, по которому создаются объекты. В python всё является объектами - и строки, и списки, и словари, и всё остальное. Но возможности ООП в python этим не ограничены. Программист может написать свой тип данных (класс), определить в нём свои методы. Это не является обязательным - мы можем пользоваться только встроенными объектами. Однако ООП полезно при долгосрочной разработке программы несколькими людьми, так как упрощает понимание кода.

1. **Model-View-Controller**

**Model-View-Controller** (**MVC**,«Модель-Представление-Контроллер»),—схема разделения данных приложения, [пользовательского интерфейса](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

* ***Модель*** (*Model*) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
* ***Представление*** (*View*) отвечает за отображение данных модели пользователю, реагируя на изменения модели.
* ***Контроллер*** (*Controller*) интерпретирует действия пользователя, оповещая модель о необходимости изменений.



Рисунок 1 – Схема MVC архитектуры

## История создания веб программирования

**Веб-программирование** — раздел [программирования](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), ориентированный на разработку [веб-приложений](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) (программ, обеспечивающих функционирование [динамических сайтов](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%81%D0%B0%D0%B9%D1%82) [Всемирной паутины](https://ru.wikipedia.org/wiki/%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D0%B0)).

Языки веб-программирования — это [языки](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), которые в основном предназначены для работы с веб-технологиями. Языки веб-программирования можно условно разделить на две пересекающиеся группы: [клиентские](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) и [серверные](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)).

* Как следует из названия, программы на клиентских языках обрабатываются на стороне [пользователя](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C), как правило, их выполняет [браузер](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80). Это и создает главную проблему клиентских языков — результат выполнения программы ([скрипта](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)) зависит от браузера пользователя. То есть, если пользователь запретил выполнять клиентские программы, то они исполняться не будут, как бы ни желал этого [программист](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82). Кроме того, может произойти такое, что в разных браузерах или в разных версиях одного и того же браузера один и тот же скрипт будет выполняться по-разному. С другой стороны, если программист возлагает надежды на серверные программы, то он может упростить их работу и снизить нагрузку на [сервер](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)) за счет программ, исполняемых на стороне клиента, поскольку они не всегда требуют перезагрузку (генерацию) [страницы](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0).
* Когда пользователь дает запрос на какую-либо страницу (переходит на неё по ссылке или вводит адрес в адресной строке своего браузера), то вызванная страница сначала обрабатывается на сервере, то есть выполняются все программы, связанные со страницей, и только потом возвращается к посетителю по сети в виде файла. Этот файл может иметь расширения [HTML](https://ru.wikipedia.org/wiki/HTML), [PHP](https://ru.wikipedia.org/wiki/PHP), [ASP](https://ru.wikipedia.org/wiki/Active_Server_Pages), [ASPX](https://ru.wikipedia.org/wiki/Asp.net), [Perl](https://ru.wikipedia.org/wiki/Perl), [SSI](https://ru.wikipedia.org/wiki/SSI_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [XML](https://ru.wikipedia.org/wiki/XML), [DHTML](https://ru.wikipedia.org/wiki/DHTML), [XHTML](https://ru.wikipedia.org/wiki/XHTML).

# МНОГОСТРАНИЧНЫЙ САЙТ «E-COSMETICS»

## Описание задания

Создать многостраничный сайт с использованием современных технологий

(C# 9.0, .NET 5, ASP.NET Core, Entity Framework Core, JS, CSS, HTML). Создать удобный и понятный интерфейс для взаимодействия с пользователем.

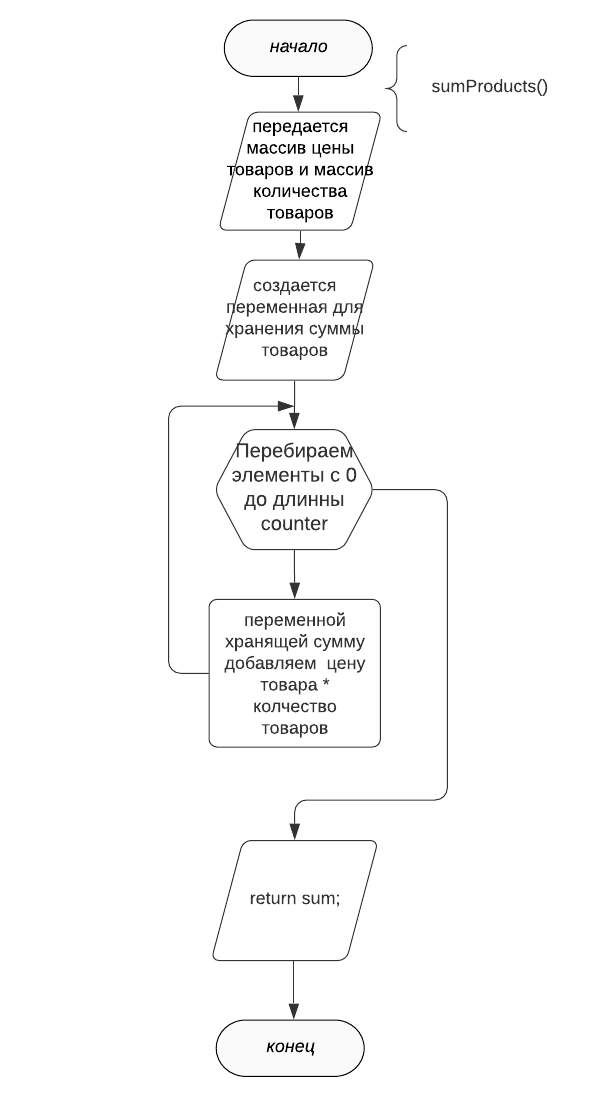
## Описание функционала сайта

На сайте реализован весь функционал свойственный интернет-магазину. На нём реализована система авторизации (регистрация, вход на сайт), а также разделения по ролям (пользователь, админ, разработчик). Пользователь может просматривать доступные товары, осуществлять поиск товаров по определённым критериям, добавлять товары в корзину и производить оформление заказов. Администратор, в дополнении к возможностям пользователя , может изменять и удалять товары, просматривать список заказов, изменять и удалять заказы, производить поиск заказов. Разработчик получает доступ к API сайта. API позволяет получать данные о товарах, которые есть на сайте. Для сайта реализована база данных, взаимодействие с которой осуществляется посредством фреймворка Entity Framework Core. База данных реализована на основе MS-SQL Server.

## 

## Блок схемы алгоритмов

Блок схемы алгоритмов представлены ниже :



*Рисунок 2 - Схема функции sumProducts*

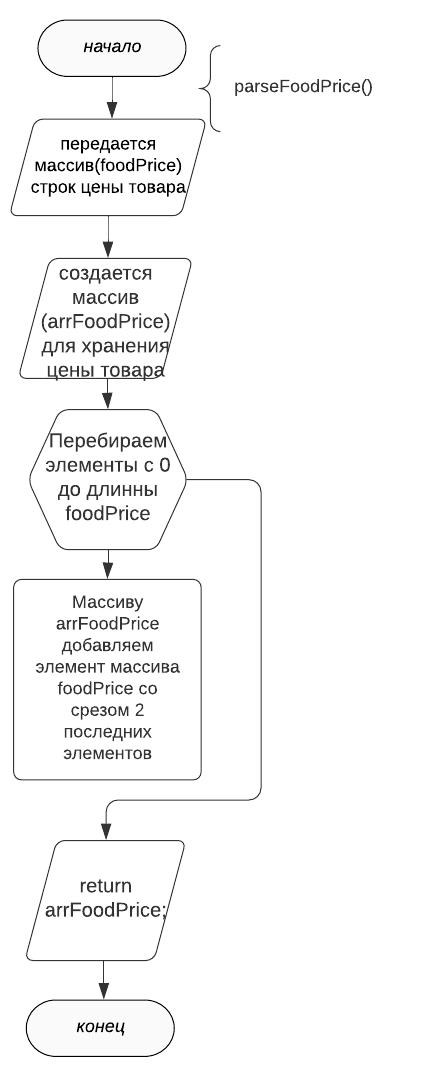


Рисунок 3 - Схема функции parseFoodPrice

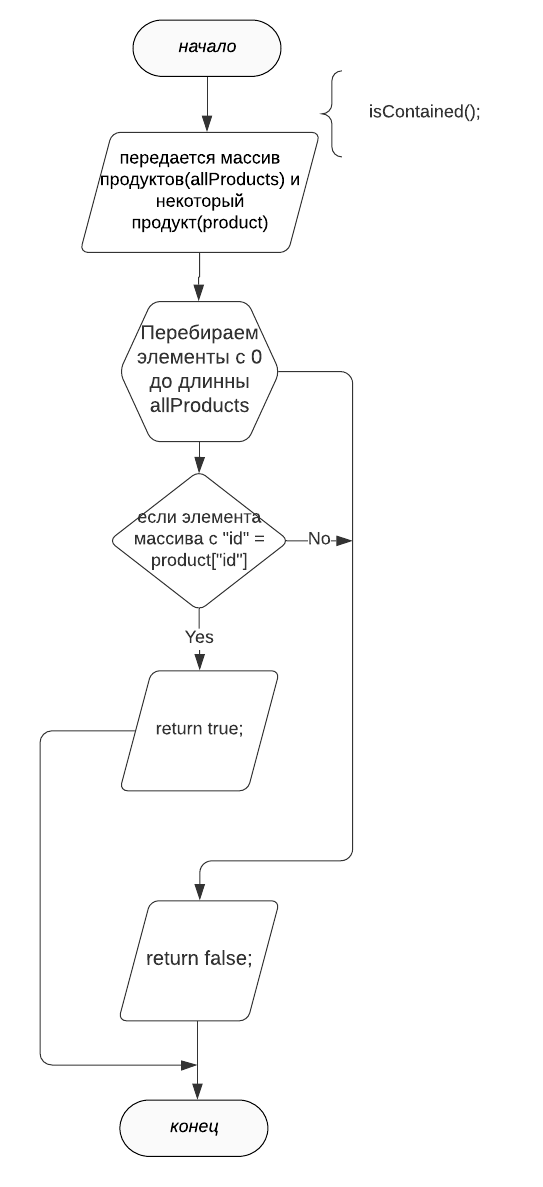


Рисунок 4 - Схема функции isContained

## 2.4 Руководство пользователя

При заходе на сайт пользователя встречает главная страница сайта. На главной странице есть навигационная панель и строка поиска. Эта панель содержит кнопки, при нажатии на которые происходят определённые действия. Названия кнопок отображают то, что делают эти кнопки. Для просмотра списка товаров пользователь должен нажать на кнопку “Каталог” при нажатии этой кнопки он попадает на страницу со всеми доступными товарами. На странице “Каталог” появляется возможность произвести поиск товара по определённым критериям (артикул, название, цена и т.п.). После того, как пользователь нашел нужный товар, он может кликнуть на блок предварительного просмотра, что переведёт его на страницу с полным описанием товара. На этой странице пользователь сможет добавить товар в корзину. Для перехода в корзину на навигационной панели есть кнопка, при нажатии которой открывается корзина со всем списком товаров, которые выбрал пользователь. В корзине он может изменить кол-во выбранных товаров или удалить ненужные. В корзине есть кнопка “Оформить заказ”, при нажатии на которую пользователь переходит на страницу оформления заказа. На этой странице пользователь вводит данные для оформления (имя, фамилия, адрес, номер телефона и т.п.), также на этой странице выводится информация о товарах. После отправки пользователь переходит на окно, которое сообщает, что заказ оформлен, и дублирует информацию из предыдущей странице.

## 2.5 Тестирование программы

Результаты проверки основного функционала сайта E-Cosmetics представлены на скриншотах:

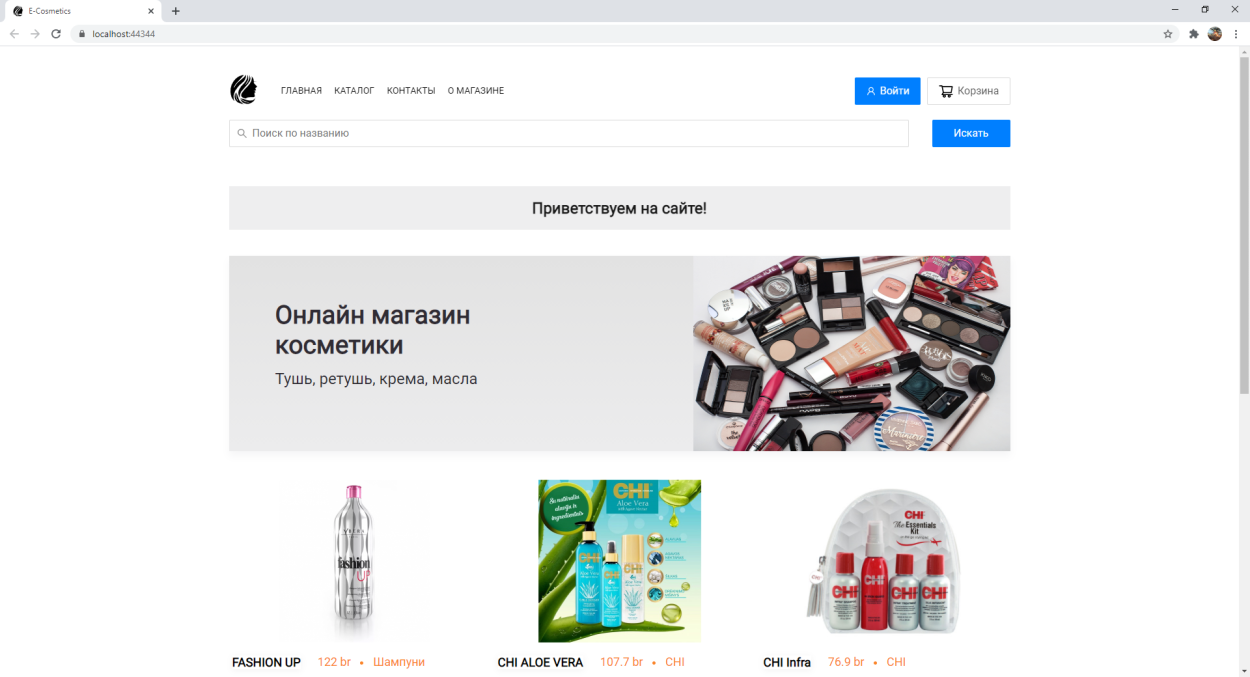


Рисунок 5 – Скриншот главной страницы

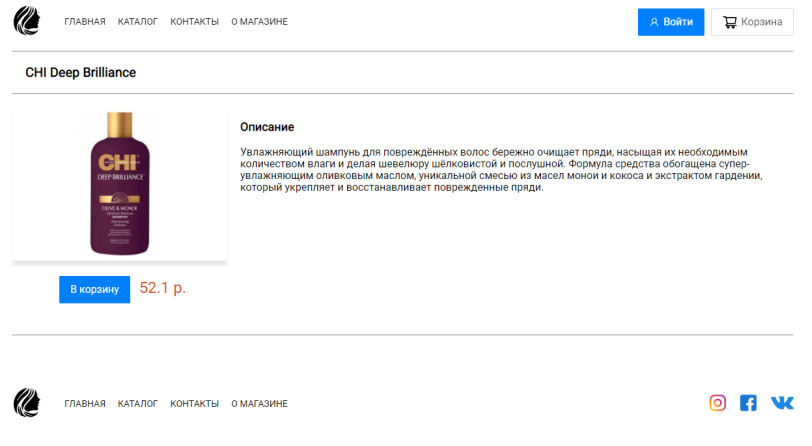


Рисунок 6 - Скриншот страницы товара

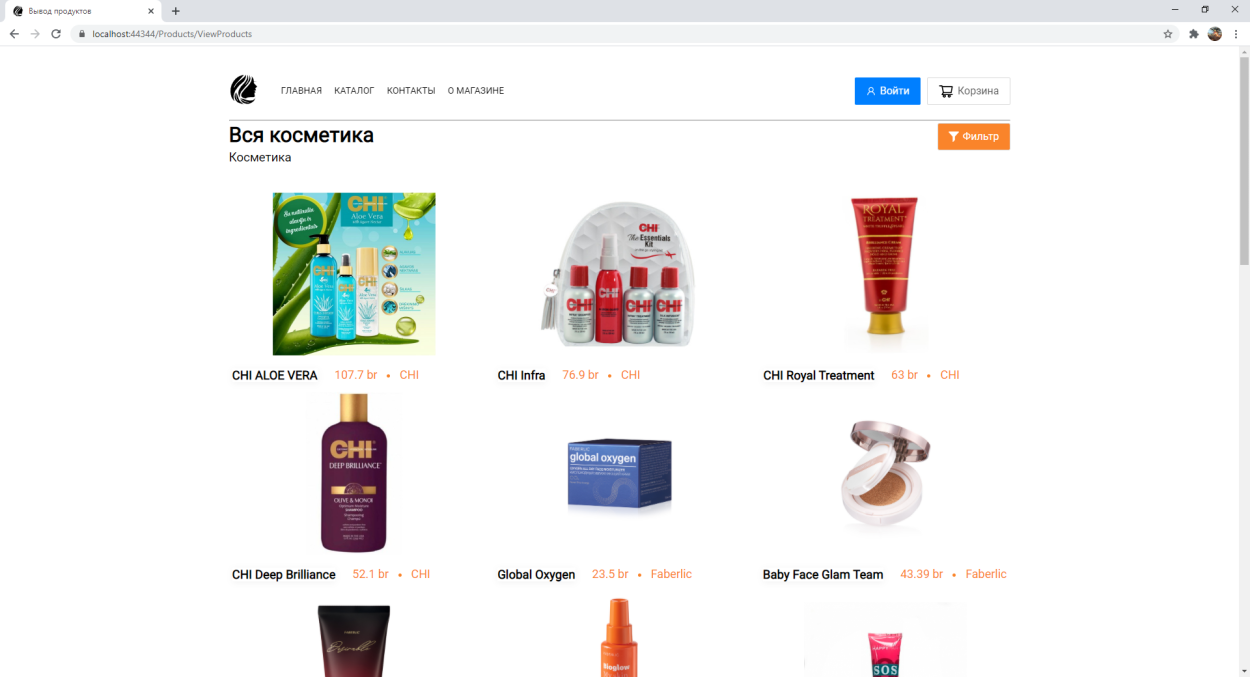


Рисунок 7 - Скриншот страницы каталога

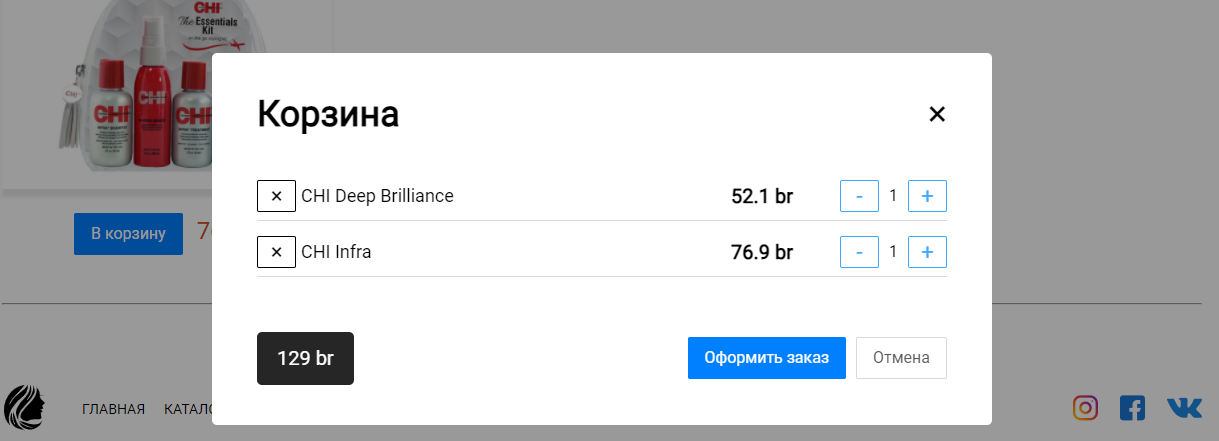


Рисунок 8 – Скриншот корзины (можно открыть на любой странице)

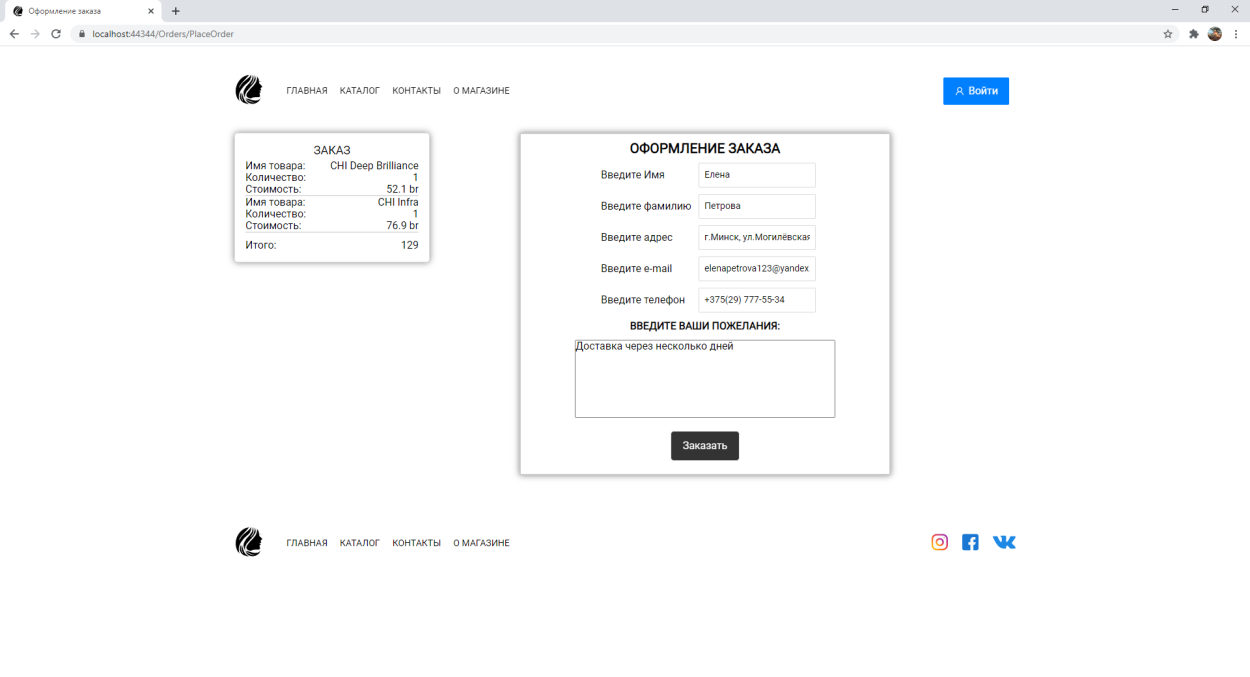


Рисунок 9 – Скриншот страницы оформление заказа

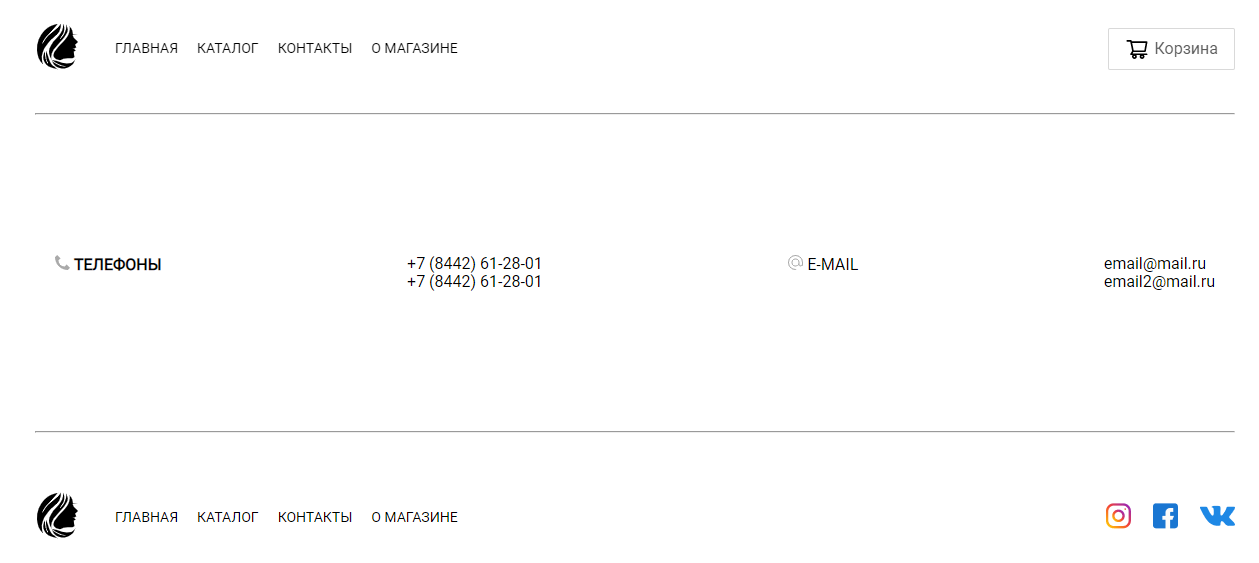


Рисунок 10 - Скриншот страницы контактов

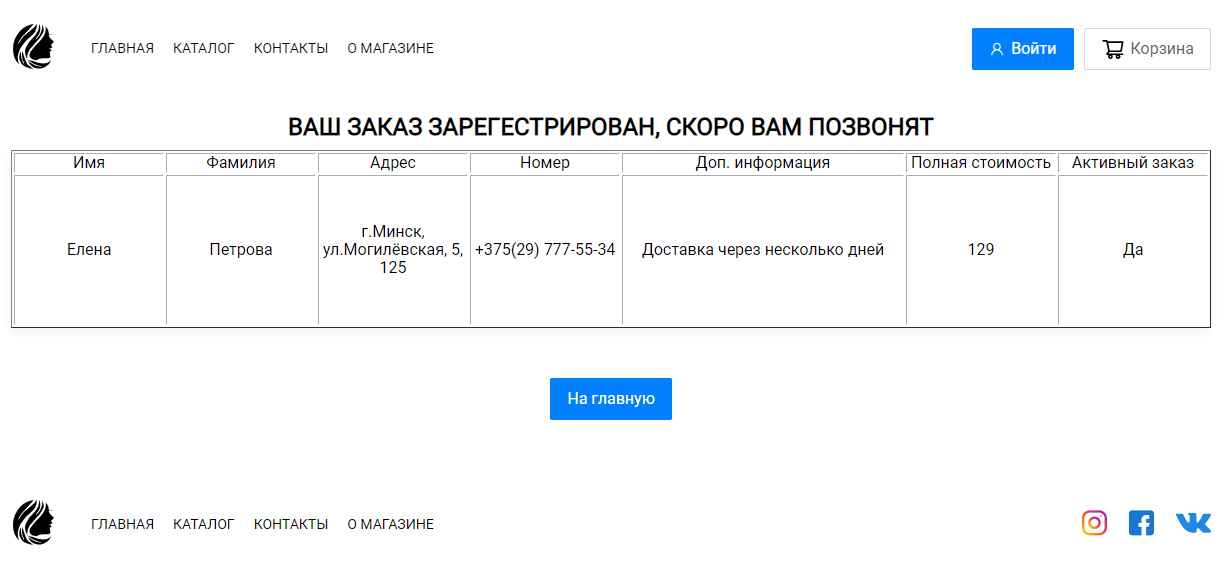


Рисунок 11 – Скриншот страницы «Заказ оформлен успешно»

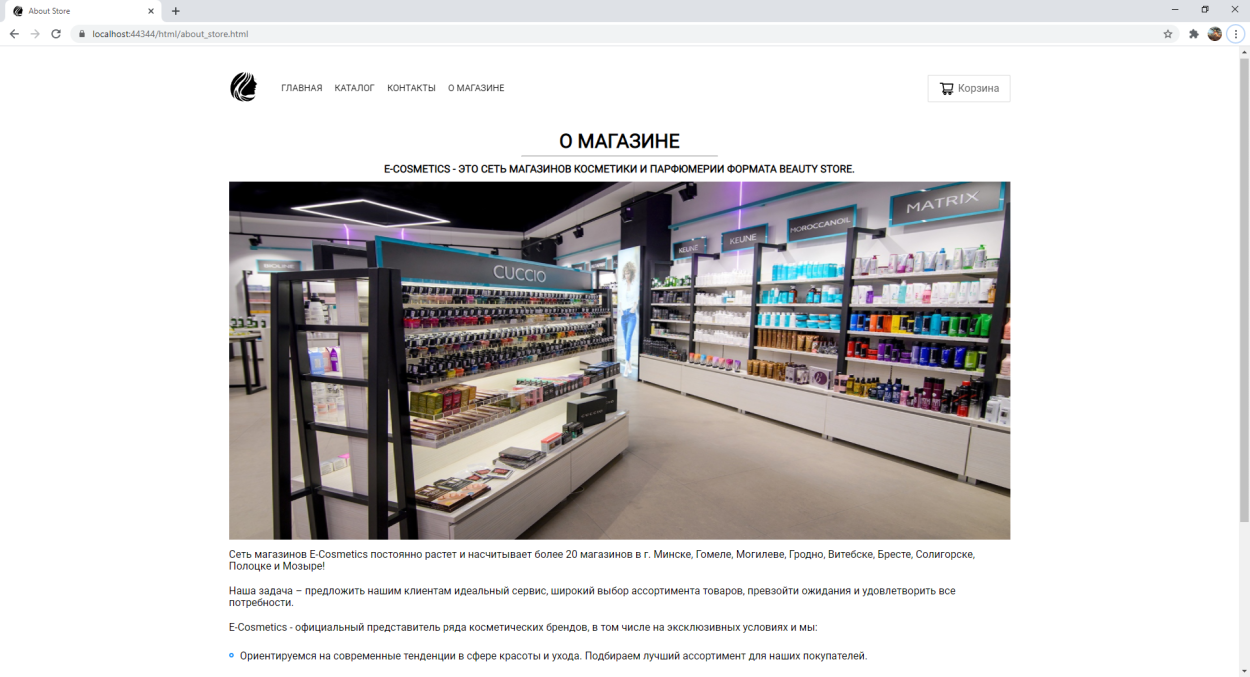


Рисунок 12 –Скриншот страницы О магазине

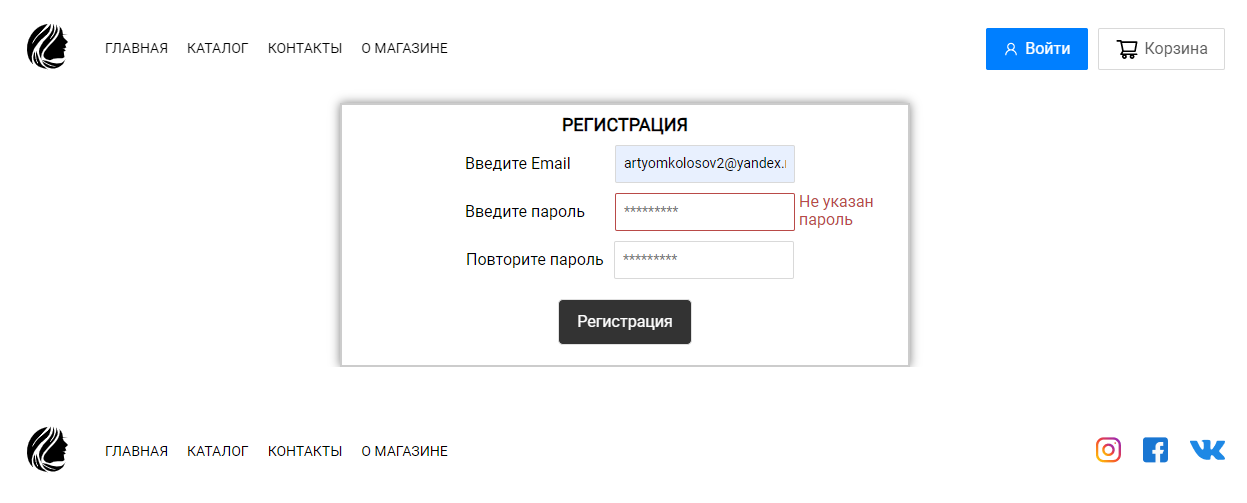


Рисунок 13 – Скриншот страницы регистрация

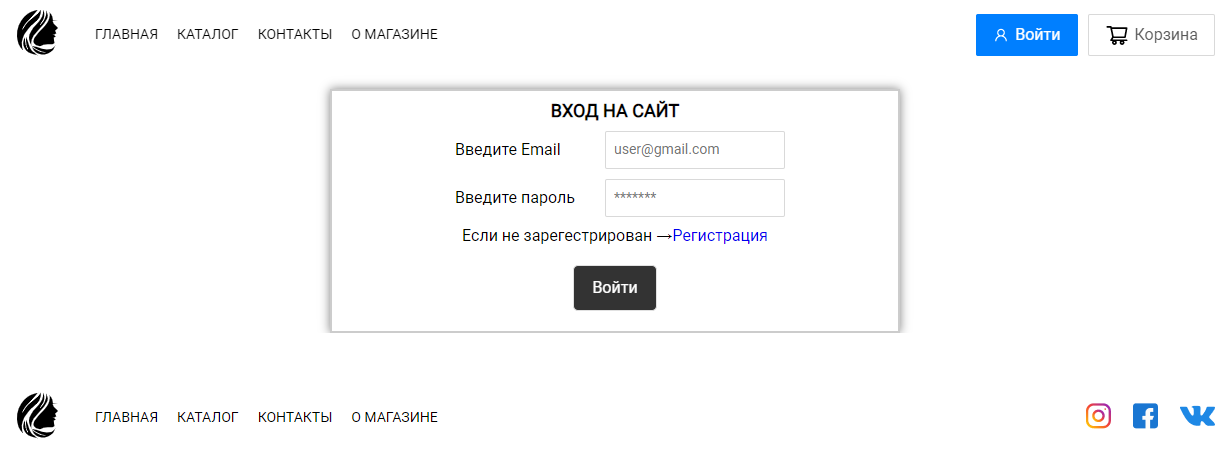


Рисунок 14 - Скриншот страницы входа на сайт

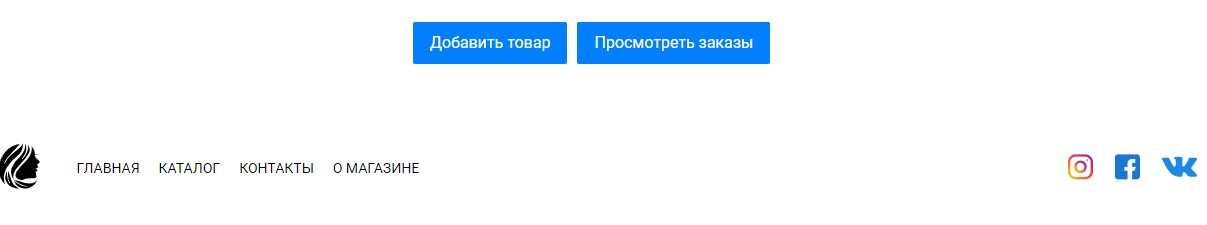


Рисунок 15 - Скриншот главной страницы для админа

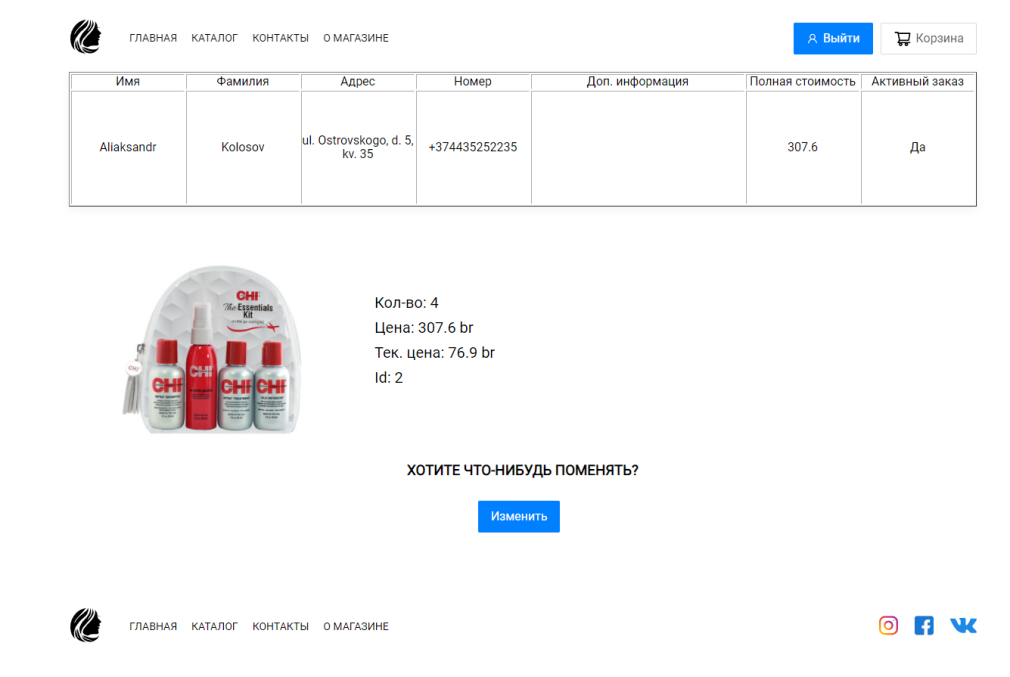


Рисунок 16 - Скриншот страницы просмотра заказа для админа

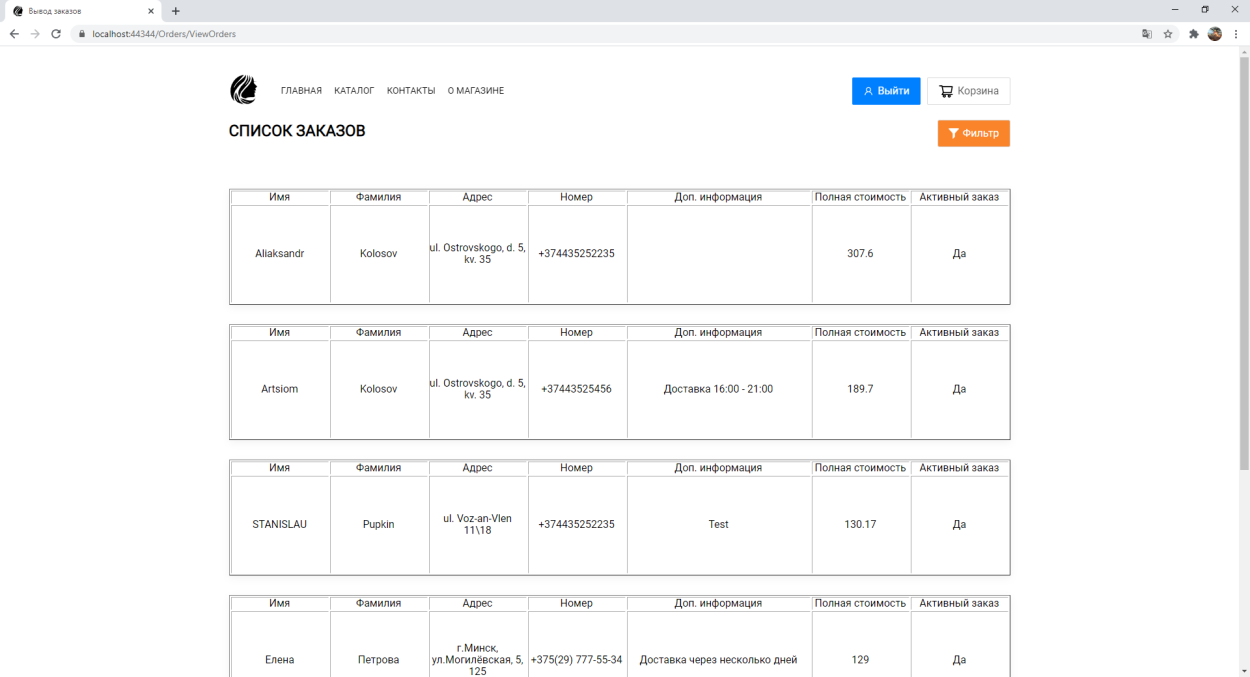


Рисунок 17 – Скриншот страницы списка заказов

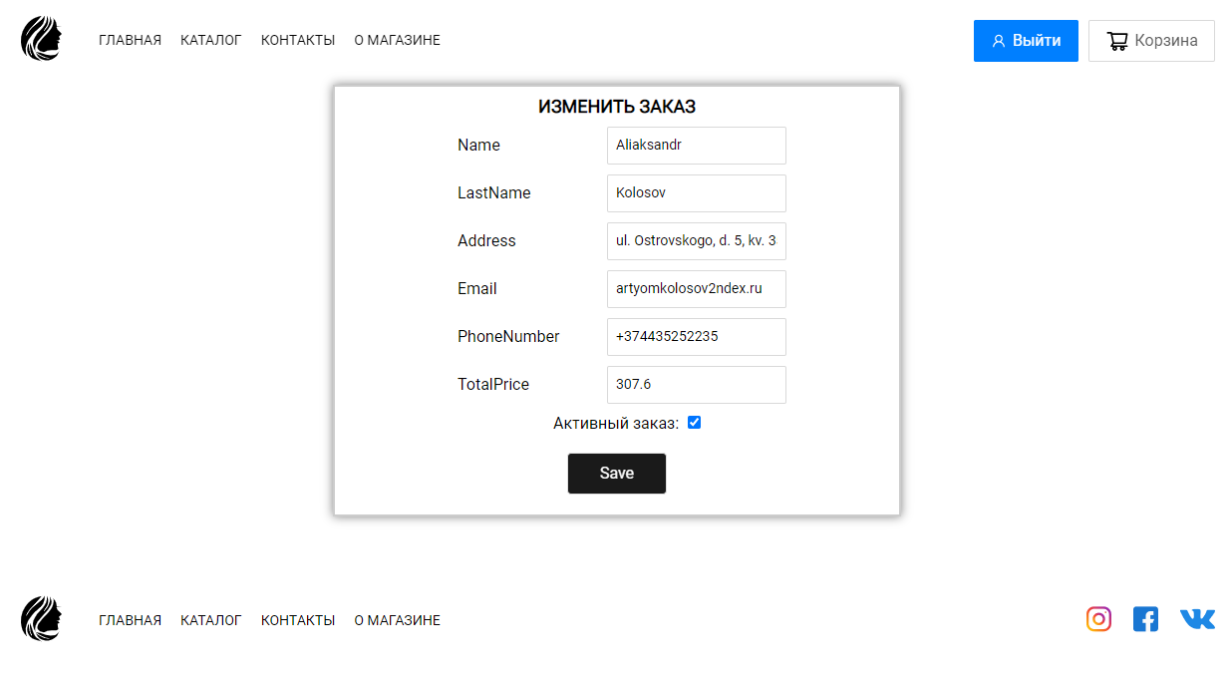


Рисунок 18 – Скриншот страницы изменения заказа

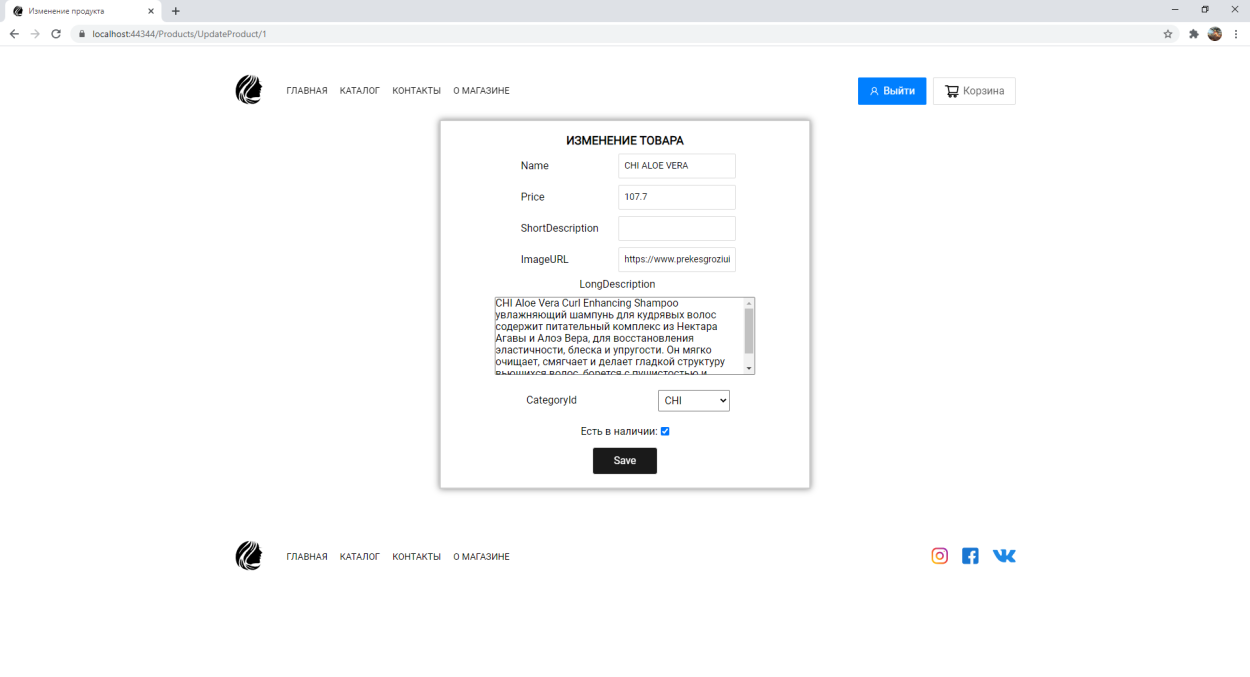


Рисунок 19 - Скриншот страницы «изменения товара» заполненного

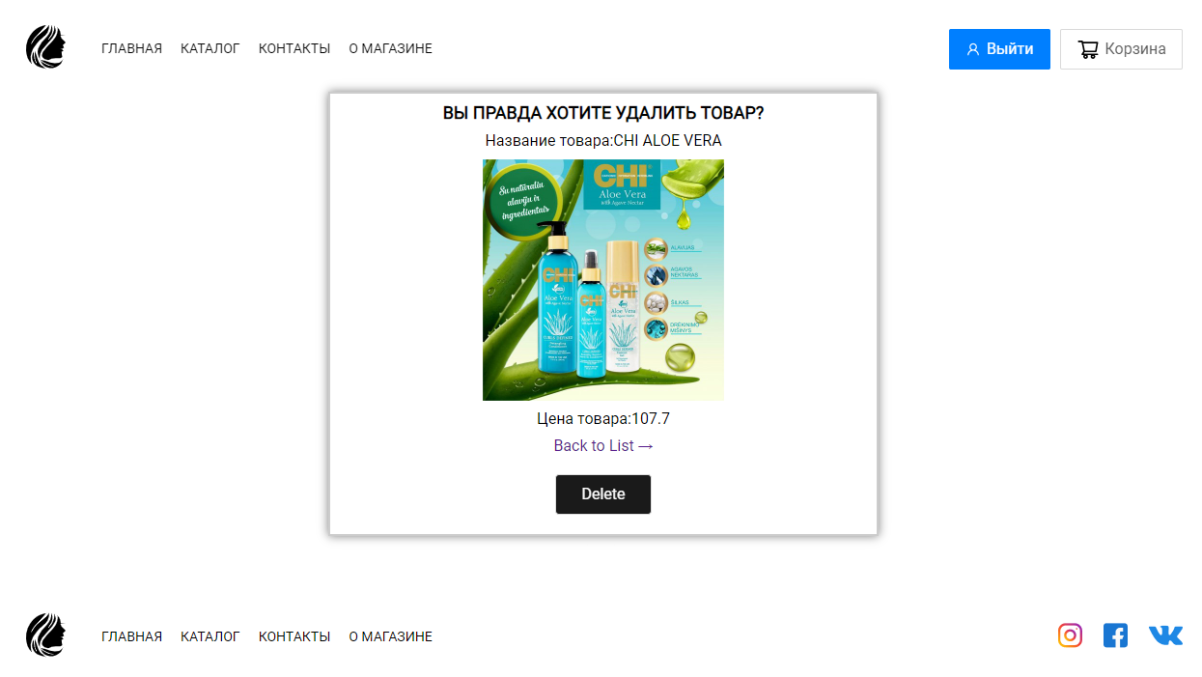


Рисунок 20 - Скриншот страницы «Удаления товара» (Для админа)

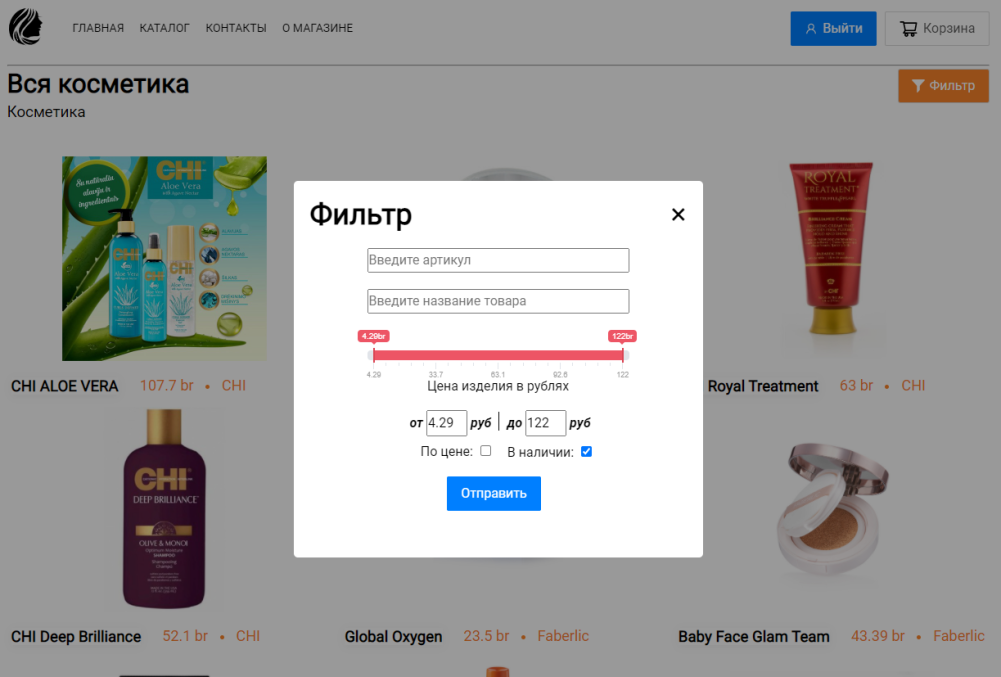


Рисунок 21 - Скриншот модального окна фильтра товаров

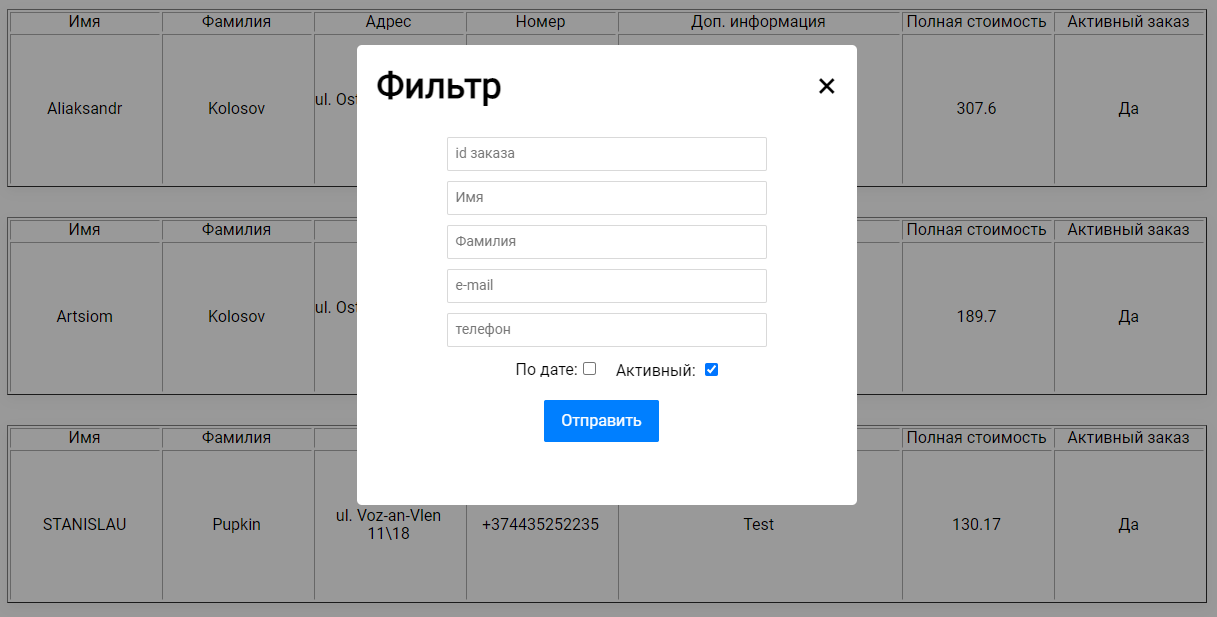


Рисунок 22 - Скриншот модального окна фильтра заказов

## 2.6 Вывод

После тестирования программы можно сделать вывод, что сайт работает корректно, имеет понятный и дружелюбный интерфейс. Все баги устранены. Благодаря курсовому проекту был закреплен объектно-ориентированный подход программирования, сервер разработан с применение архитектуры MVC, изучена и закреплены на практике такие технологии как: Jquery, C# 9.0, .NET 5, ASP.NET Core, Entity Framework Core, JS, CSS, HTML. Закреплены навыки работы с разными типами данных. Следовательно, тема была усвоена успешно.

# ЗАКЛЮЧЕНИЕ

После тестирования программы можно сделать вывод, что сайт работает корректно, имеет понятный и дружелюбный интерфейс. Все баги устранены. Благодаря курсовому проекту был закреплен объектно-ориентированный подход программирования, сервер разработан с применение архитектуры MVC, изучены и закреплены на практике такие технологии, как: Jquery, C# 9.0, .NET 5, ASP.NET Core, Entity Framework Core, JS, CSS, HTML. Закреплены навыки работы с разными типами данных. Следовательно, тема была усвоена успешно.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРА

1. Информация по разработке сайта - https://wikipedia.org/
2. Дополнительные сведения по языку – <https://habr.com/>
3. Дополнительные сведения по языку – <https://metanit.com/>
4. Васильев А.Н. – «Программирование в примерах и задачах»
5. Адам Фримен ASP.NET Core MVC 2 с примерами на С# для профессионалов 7-е изд.
6. Рихтер Дж. CLR via C# «Программирование на платформе .NET Framework 4.5 на языке С#, 4-e изд.»
7. Прайс Марк Дж. П68 C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов. 3-е изд.

# ПРИЛОЖЕНИЕ 1 – Листинг исходных кодов программы

Файл cart.js

﻿function delElement(arr, index)

{

let newArr = [];

for (let i in arr) {

if (i == index) {

continue;

}

newArr.push(arr[i]);

}

return newArr;

}

function addProduct(product) {

const products = document.getElementById('products');

const block = document.createElement('div');

block.innerHTML = `

<div class="food-row">

<button class="btn-counter-close" id="btn\_delete\_food\_row">&times;</button>

<span class="food-name">${product.name}</span>

<strong class="food-price">${product.cost} br</strong>

<div class="food-counter">

<button class="btn-counter" id="btn-counter">-</button>

<span class="counter" min="0" max="10000">${product.number}</span>

<button class="btn-counter" id="btn-counter">+</button>

</div>

</div>

`;

products.appendChild(block);

}

function createProducts(product) {

for (let i in product) {

addProduct(product[i]);

}

}

function sumProducts(arrFoodPrice, counter) {

let sum = 0;

for (let i = 0; i < counter.length; i++) {

sum += arrFoodPrice[i] \* counter[i].innerHTML;

}

return sum;

}

function parseFoodPrice(foodPrice) {

let arrFoodPrice = [];

for (let i = 0; i < foodPrice.length; i++) {

arrFoodPrice.push(Number(foodPrice[i].innerHTML.slice(0, -2)));

}

return arrFoodPrice;

}

function getPricetag() {

const foodPrice = document.querySelectorAll('.food-price');

const counter = document.querySelectorAll('.counter');

let arrFoodPrice = parseFoodPrice(foodPrice);

return sumProducts(arrFoodPrice, counter);

}

export function uploadCart(getCookie) {

const products = getCookie("products") ? JSON.parse(getCookie("products")) : [];

createProducts(products);

const pricetag = getCookie("pricetag") ? getCookie("pricetag") : 0;

document.querySelector('.modal-pricetag').innerHTML = `${pricetag} br`;

}

export function cartHandler(getCookie, setCookie) {

$().ready(function () {

$('.btn-counter-close').on('click', function () {

let parent = $(this).parent();

let index = Math.round($('.food-row').find('button').parent().index(parent) / 2);

const foodRow = document.querySelectorAll(".food-row");

foodRow[index].remove();

let allProducts = getCookie("products") ? JSON.parse(getCookie("products")) : [];

allProducts = delElement(allProducts, index);

const pricetag = Math.round(getPricetag() \* 100) / 100;

const modalPricetag = document.querySelector('.modal-pricetag');

modalPricetag.innerHTML = `${pricetag} br`;

setCookie("products", JSON.stringify(allProducts));

setCookie("pricetag", pricetag);

});

$('.btn-counter').on('click', function () {

const newProducts = getCookie("products") ? JSON.parse(getCookie("products")) : [];

const counter = document.querySelectorAll('.counter');

let parent = $(this).parent();

let index = $('.food-counter').find('button').parent().index(parent);

if (this.textContent === '+') {

counter[index].innerHTML++;

}

else {

counter[index].innerHTML--;

if (counter[index].innerHTML < 1) {

counter[index].innerHTML = 1;

}

}

const pricetag = Math.round(getPricetag() \* 100) / 100;

const modalPricetag = document.querySelector('.modal-pricetag');

modalPricetag.innerHTML = `${pricetag} br`;

newProducts[index].number = Number(counter[index].innerHTML);

setCookie("products", JSON.stringify(newProducts));

setCookie("pricetag", pricetag);

});

});

}

function isContained(allProducts, product) {

for (let i = 0; i < allProducts.length; i++) {

if (allProducts[i]["id"] === product["id"]) {

return true;

}

}

return false;

}

export function addToCart(getCookie, setCookie)

{

const productName = document.querySelector('.product-info\_\_header');

const cost = document.querySelector('.offer-footer\_\_cost');

let counter = document.querySelectorAll('.counter');

const id = Number(document.getElementById('id\_product').innerHTML);

const product = new Object();

product["name"] = productName.innerHTML;

product["cost"] = Number(cost.innerHTML.slice(0, -2));

product["id"] = id;

let allProducts = getCookie("products") ? JSON.parse(getCookie("products")) : [];

const index = allProducts.length - 1;

// The product is contained in cart

if (isContained(allProducts, product)) {

counter[index].innerHTML = Number(counter[index].innerHTML) + 1;

product["number"] = Number(counter[index].innerHTML);

allProducts[index]["number"] = product["number"];

} else {

product["number"] = 1;

addProduct(product);// add to basket

allProducts.push(product); // push to array objects

location.reload();

}

let pricetag = Math.round(getPricetag() \* 100) / 100;

document.querySelector('.modal-pricetag').innerHTML = `${pricetag} br`;

setCookie("products", JSON.stringify(allProducts));

setCookie("pricetag", pricetag);

console.log(allProducts);

}

Файл cropText.js

﻿export function cropTextArray(textArray, size, endCharacter) {

textArray.forEach(el => {

let text = el.innerHTML;

if (el.innerHTML.length > size) {

text = text.substr(0, size);

el.innerHTML = text + endCharacter;

}

});

};

export function cropText(text, size, endCharacter) {

let data = text.innerHTML;

if (data.length > size) {

data = data.substr(0, size);

text.innerHTML = data + endCharacter;

}

};

Файл getDataFromCookie.js

﻿export function setCookie(name, value, exp\_y, exp\_m, exp\_d, path = "/", domain = "", secure = true) {

let cookieString = `${name}=${escape(value)}`;

if (exp\_y) {

const expires = new Date(exp\_y, exp\_m, exp\_d);

cookieString += `; expires=${expires.toGMTString()}`;

}

if (path) {

cookieString += `; path=${escape(path)}`;

}

if (domain) {

cookieString += `; domain=${escape(domain)}`;

}

if (secure) {

cookieString += "; secure";

}

document.cookie = cookieString;

}

export function deleteCookie(cookieName) {

let cookieDate = new Date(); // Текущая дата и время

cookieDate.setTime(cookieDate.getTime() - 1);

document.cookie = `${cookieName}=; expires=${cookieDate.toGMTString()}`;

}

export function getCookie(cookieName) {

let results = document.cookie.match(`(^|;) ?${cookieName}=([^;]\*)(;|$)`);

return results ? unescape(results[2]) : null;

}

Файл getDataFromDB.js

﻿export async function setMinMaxPrice() {

const response = await fetch("/api/ProductsApi/MinMaxPrice", {

method: "GET",

headers: { "Accept": "application/json" }

});

if (response.ok === true) {

const user = await response.json();

const form = document.forms["form\_search"];

form.elements["priceMin"].value = user.min;

form.elements["priceMax"].value = user.max;

}

}

export async function getMinPrice() {

const response = await fetch("/api/ProductsApi/MinPrice", {

method: "GET",

headers: { "Accept": "application/json" }

});

let data = await response.json();

return data.min;

}

export async function getMaxPrice() {

const response = await fetch("/api/ProductsApi/MaxPrice", {

method: "GET",

headers: { "Accept": "application/json" }

});

let data = await response.json();

return data.max;

}

Файл makingOrder.js

﻿function addCostToOrder(cost) {

const products = document.getElementById('total-cost');

const block = document.createElement('div');

block.innerHTML = `

<div class="total-cost">

<span>Итого:</span><span>${cost}</span>

</div>

`;

products.appendChild(block);

}

function addProductToOrder(product) {

const products = document.getElementById('cart\_info\_body');

const block = document.createElement('div');

block.innerHTML = `

<div class="cart-info\_\_row">

<span>Имя товара:</span><span>${product.name}</span>

</div>

<div class="cart-info\_\_row">

<span>Количество:</span><span>${product.number}</span>

</div>

<div class="cart-info\_\_row">

<span>Стоимость:</span><span>${product.cost} br</span>

</div>

<hr class="hr"/>

`;

products.appendChild(block);

}

function makingOrderProducts(products) {

for (let i = 0; i < products.length; i++) {

addProductToOrder(products[i]);

}

}

function makingOrderTotalCost(pricetag) {

addCostToOrder(pricetag);

}

export default async function makingOrder(getCookie) {

const products = JSON.parse(getCookie("products"));

const pricetag = getCookie("pricetag");

await makingOrderProducts(products);

await makingOrderTotalCost(pricetag);

}

Файл modalCart.js

﻿export function modalCart() {

function eventModalCart() {

modalCart.classList.toggle("on-click");

}

const cardBtn = document.querySelector("#btn-basket");

const modalCart = document.querySelector(".modal-cart");

const close = document.querySelector(".btn-close");

cardBtn.addEventListener("click", eventModalCart);

close.addEventListener("click", eventModalCart);

}

Файл modalFilter.js

﻿export function modalFilter(filterBtn) {

function eventModalFilter() {

modalSearch.classList.toggle("on-click");

}

const modalSearch = document.querySelector(".modal-search");

const closeFilter = document.querySelector(".btn-close-filter");

filterBtn.addEventListener("click", eventModalFilter);

closeFilter.addEventListener("click", eventModalFilter);

}

Файл slider.js

﻿export function buildSlider(minRange, maxRange, $slider) {

let $inputFrom = $("#priceMin");

let $inputTo = $("#priceMax");

let instance;

let from = 0;

let to = 0;

$slider.ionRangeSlider({

skin: "flat",

type: "double",

step: 0.1,

grid: true,

postfix: "br",

min: minRange,

max: maxRange,

from: localStorage.getItem("SliderFrom") ? JSON.parse(localStorage.getItem("SliderFrom")) : minRange,

to: localStorage.getItem("SliderTo") ? JSON.parse(localStorage.getItem("SliderTo")) : minRange + 10,

onStart: updateScroll,

onChange: updateScroll,

onFinish: updateScroll

});

instance = $slider.data("ionRangeSlider");

function updateScroll(data) {

from = data.from;

to = data.to;

$inputFrom.prop("value", from);

$inputTo.prop("value", to);

localStorage.setItem("SliderFrom", JSON.stringify(from));

localStorage.setItem("SliderTo", JSON.stringify(to));

}

$inputFrom.on("input", function () {

let value = $(this).prop("value");

/\* validate \*/

// down range

if (value < minRange.length) {

value = minRange;

}

// up range

else if (value > maxRange) {

if (value.length === maxRange.length) {

value = to - 1;

}

else {

value = String(to).slice(0, to.length) - 1;

}

}

// if zero

else if (value[0] == 0, value.length > 1) {

value = value.slice(0, 3);

}

instance.update({

from: value

});

$(this).prop("value", value);

localStorage.setItem("SliderFrom", JSON.stringify(value));

});

$inputTo.on("input", function () {

let value = $(this).prop("value");

// validate

if (value > maxRange) {

if (value.length === maxRange.length) {

value = maxRange;

}

else {

value = value.slice(0, maxRange.length);

}

}

instance.update({

to: value

});

$(this).prop("value", value);

localStorage.setItem("SliderTo", JSON.stringify(value));

});

}

Файл main.js

import { modalCart } from "./modules/modalCart.js";

import { modalFilter } from "./modules/modalFilter.js";

import { getMinPrice, getMaxPrice, setMinMaxPrice } from "./modules/getDataFromDB.js";

import { setCookie, deleteCookie, getCookie } from "./modules/getDataFromCookie.js";

import { addToCart, uploadCart, cartHandler } from "./modules/cart.js";

import { buildSlider } from "./modules/slider.js";

import makingOrder from "./modules/makingOrder.js";

import { cropText, cropTextArray } from "./modules/cropText.js";

async function main() {

modalCart();

const filterAdminBtn = document.querySelector('#btn\_filter\_admin');

if (filterAdminBtn) {

modalFilter(filterAdminBtn);

}

// slider for filter

const filterBtn = document.querySelector('#btn\_filter');

if (filterBtn) {

modalFilter(filterBtn);

setMinMaxPrice();

const maxPrice = await getMaxPrice();

const minPrice = await getMinPrice();

const $slider = $(".js-range-slider");

buildSlider(minPrice, maxPrice, $slider);

}

//!slider for filter

// basket

uploadCart(getCookie);

cartHandler(getCookie, setCookie);

const btnAddProduct = document.getElementById('btn\_add\_product');

if (btnAddProduct)

{

btnAddProduct.addEventListener('click', function () {

addToCart(getCookie, setCookie);

});

}

//!basket

// making order

const order = document.getElementById('order');

if (order) {

document.getElementById("btn-basket").remove();

await makingOrder(getCookie);

}

// !making order

// crop text

const textSmall = document.querySelectorAll("#card-info\_\_name");

const endCharacter = '...';

if (textSmall) {

cropTextArray(textSmall, 20, endCharacter);

}

// !crop text

//длинна анимации

new WOW().init();

}

main();

Файл AccountController.cs

﻿using System.Collections.Generic;

using System.Security.Claims;

using System.Threading.Tasks;

using E\_Shop\_Cosmetic.Data.Interfaces;

using E\_Shop\_Cosmetic.Data.Models;

using E\_Shop\_Cosmetic.ViewModels;

using Microsoft.AspNetCore.Authentication;

using Microsoft.AspNetCore.Authentication.Cookies;

using Microsoft.AspNetCore.Mvc;

using System.Linq;

using Microsoft.Extensions.Logging;

using E\_Shop\_Cosmetic.Data.Specifications;

namespace E\_Shop\_Cosmetic.Controllers

{

public class AccountController : Controller

{

private readonly ILogger \_logger;

private readonly IUserRepository \_userRepository;

private readonly IRoleRepository \_roleRepository;

public AccountController(IUserRepository userRepository, IRoleRepository roleRepository, ILogger<AccountController> logger)

{

\_logger = logger;

\_userRepository = userRepository;

\_roleRepository = roleRepository;

}

[HttpGet]

public IActionResult Login()

{

ViewBag.Title = "Логин";

\_logger.LogInformation("Http Get Account\\Login called");

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Login(LoginViewModel model)

{

\_logger.LogInformation("Http Post Account\\Login called");

if (ModelState.IsValid)

{

var users = await \_userRepository.GetUsersAsync(new UserSpecification().WithoutTracking());

var user = users.FirstOrDefault(u => u.Email == model.Email && u.Password == model.Password);

if (user != null)

{

await Authenticate(user);

return RedirectToAction("Index", "Home");

}

ModelState.AddModelError("", "Некорректные логин и(или) пароль");

}

return View(model);

}

[HttpGet]

public IActionResult Register()

{

ViewBag.Title = "Регистрация";

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Register(RegisterViewModel model)

{

if (ModelState.IsValid)

{

var users = await \_userRepository.GetUsersAsync();

var user = users.FirstOrDefault(u => u.Email == model.Email);

if (user == null)

{

user = new User { Email = model.Email, Password = model.Password, UserRoleId = 1, Role = await \_roleRepository.GetRoleByIdAsync(1) };

await \_userRepository.AddUserAsync(user);

await Authenticate(user);

return RedirectToAction("Index", "Home");

}

else

{

ModelState.AddModelError("", "Некорректные логин и(или) пароль");

}

}

return View(model);

}

private async Task Authenticate(User user)

{

var roles = await \_roleRepository.GetRolesAsync();

var roleName = roles.FirstOrDefault(i => i.Id == user.UserRoleId);

var claims = new List<Claim>

{

new Claim(ClaimsIdentity.DefaultNameClaimType, user.Email),

new Claim(ClaimsIdentity.DefaultRoleClaimType, roleName?.Name ?? "user")

};

var id = new ClaimsIdentity(claims, "ApplicationCookie", ClaimsIdentity.DefaultNameClaimType, ClaimsIdentity.DefaultRoleClaimType);

await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(id));

}

public async Task<IActionResult> Logout()

{

await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);

return RedirectToAction("Index", "Home");

}

public IActionResult AccessDenied()

{

return RedirectToAction("Index", "Home");

}

}

}

Файл HomeController.cs

﻿using E\_Shop\_Cosmetic.Data.Interfaces;

using E\_Shop\_Cosmetic.Data.Specifications;

using E\_Shop\_Cosmetic.ViewModels;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using System.Text;

using System.Threading.Tasks;

namespace E\_Shop\_Cosmetic.Controllers

{

public class HomeController : Controller

{

private readonly ILogger \_logger;

private readonly IProductsRepository \_productsRepository;

public HomeController(ILogger<HomeController> logger, IProductsRepository productsRepository)

{

\_logger = logger;

\_productsRepository = productsRepository;

}

public async Task<IActionResult> Index()

{

ViewBag.Title = "E-Cosmetics";

StringBuilder messageBuilder = new StringBuilder($"Приветствуем на сайте");

if (User.Identity.IsAuthenticated)

{

messageBuilder.Append($", {User.Identity.Name}!");

}

else

{

messageBuilder.Append('!');

}

var obj = new HomeViewModel

{

Message = messageBuilder.ToString(),

Products = await \_productsRepository.GetProductsAsync

(

new ProductSpecification().

IncludeCategory().

SortByPrice().

WithoutTracking().

AddPagination(9)

),

};

\_logger.LogInformation("Home/Index is executed");

return View(obj);

}

}

}

Файл OrdersController.cs

﻿using E\_Shop\_Cosmetic.Data.Interfaces;

using E\_Shop\_Cosmetic.Data.Models;

using E\_Shop\_Cosmetic.Data.Specifications;

using E\_Shop\_Cosmetic.ViewModels;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Linq;

using System.Threading.Tasks;

namespace E\_Shop\_Cosmetic.Controllers

{

public class OrdersController : Controller

{

private readonly ICookieService \_cartService;

private readonly IOrderRepository \_orderRepository;

public OrdersController(ICookieService cartService, IOrderRepository orderRepository)

{

\_cartService = cartService;

\_orderRepository = orderRepository;

}

[HttpGet]

public async Task<IActionResult> Search(SearchOrderParams searchParams)

{

var searchSpecification = new OrderSpecification();

var isPrimaryKeyUsed = false;

if (searchParams.OrderId is not null)

{

searchSpecification.WhereId(searchParams.OrderId.Value);

isPrimaryKeyUsed = true;

}

if (!isPrimaryKeyUsed)

{

if (searchParams.Name is not null)

{

searchSpecification.WhereName(searchParams.Name);

}

if (searchParams.LastName is not null)

{

searchSpecification.WhereLastName(searchParams.LastName);

}

if (searchParams.Email is not null)

{

searchSpecification.WhereEmail(searchParams.Email);

}

if (searchParams.PhoneNumber is not null)

{

searchSpecification.WherePhone(searchParams.PhoneNumber);

}

}

if (searchParams.IsSortByDateRequired)

{

searchSpecification.SortByDate();

}

searchSpecification.WhereActive(searchParams.IsOrderActive).WithoutTracking();

var viewModel = new SearchOrderViewModel()

{

Orders = await \_orderRepository.GetOrdersAsync(searchSpecification),

SearchParams = searchParams

};

ViewBag.Title = "Поиск по товарам";

return View(viewModel);

}

[HttpGet]

public async Task<IActionResult> PlaceOrder()

{

ViewBag.Title = "Оформление заказа";

if (await \_cartService.IsAnyProductInCartAsync())

{

return View();

}

return NoContent();

}

[HttpPost]

public async Task<IActionResult> PlaceOrder(OrderViewModel orderViewModel)

{

var ordersDetails = await \_cartService.GetOrderDetailsAsync();

if (ordersDetails.Any())

{

var totalPrice = Math.Round(ordersDetails.Sum(detail => detail.TotalPrice), 2);

var newOrder = new Order()

{

Address = orderViewModel.Address,

IsOrderActive = true,

Information = orderViewModel.Information ?? "None",

Name = orderViewModel.Name,

LastName = orderViewModel.LastName,

PhoneNumber = orderViewModel.PhoneNumber,

TotalPrice = totalPrice,

OrderDetails = ordersDetails,

OrderDate = DateTime.Now,

Email = orderViewModel.Email

};

await \_orderRepository.AddOrderAsync(newOrder);

await \_cartService.ClearCartAsync();

return RedirectToAction("OrderSuccessful", newOrder);

}

else

{

return NoContent();

}

}

public IActionResult OrderSuccessful(Order order)

{

return View(order);

}

[Authorize(Roles = "admin")]

[HttpGet]

public async Task<IActionResult> UpdateOrder(int id)

{

ViewBag.Title = "Изменение заказа";

var order = await \_orderRepository.GetOrderByIdAsync(id);

if (order is not null)

{

return View(order);

}

return NoContent();

}

[Authorize(Roles = "admin")]

[HttpPost]

public async Task<IActionResult> UpdateOrder(int id, Order order)

{

if (id != order.Id)

{

return BadRequest();

}

await \_orderRepository.UpdateOrderAsync(order);

return RedirectToAction("ViewOrders", "Orders");

}

[Authorize(Roles = "admin")]

[HttpGet]

public async Task<IActionResult> Order(int id)

{

ViewBag.Title = "Заказ";

var order = await \_orderRepository.GetOrderByIdWithDetailsAsync(id);

if (order is not null)

{

return View(new ViewOrderViewModel { Order = order });

}

return NoContent();

}

[Authorize(Roles = "admin")]

[HttpGet]

public async Task<IActionResult> ViewOrders()

{

ViewBag.Title = "Вывод заказов";

var orders = await \_orderRepository.GetOrdersAsync

(

new OrderSpecification().

IncludeDetails().

SortByTotalPrice().

WithoutTracking()

);

var viewModel = new SearchOrderViewModel()

{

Orders = orders,

SearchParams = new SearchOrderParams(),

};

return View(viewModel);

}

}

}

Файл ProductsController.cs

﻿using E\_Shop\_Cosmetic.Data.Interfaces;

using E\_Shop\_Cosmetic.Data.Models;

using E\_Shop\_Cosmetic.Data.Specifications;

using E\_Shop\_Cosmetic.ViewModels;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.Extensions.Logging;

using System;

using System.Linq;

using System.Threading.Tasks;

namespace E\_Shop\_Cosmetic.Controllers

{

public class ProductsController : Controller

{

private readonly IProductsRepository \_cosmeticProductsRepository;

private readonly ICategoriesRepository \_categoriesRepository;

private readonly ILogger \_logger;

public ProductsController(IProductsRepository products, ICategoriesRepository category, ILogger<ProductsController> logger)

{

\_cosmeticProductsRepository = products;

\_categoriesRepository = category;

\_logger = logger;

}

[HttpGet]

public async Task<IActionResult> Product(int id)

{

ViewBag.Title = "Продукт";

\_logger.LogInformation("Products\\Product is executed");

var product = await \_cosmeticProductsRepository.GetProductByIdAsync(id);

if (product is not null)

{

return View(product);

}

return NoContent();

}

public async Task<IActionResult> ViewProducts()

{

ViewBag.Title = "Товары";

var viewModel = new ProductsViewModel

{

Products = await \_cosmeticProductsRepository.GetProductsAsync(

new ProductSpecification().

IncludeCategory().

WithoutTracking()),

ProductCategory = "Косметика",

SearchParams = new SearchProductsParams()

};

\_logger.LogInformation("Products\\ViewProducts is executed");

ViewBag.Title = "Вывод продуктов";

return View(viewModel);

}

[HttpGet]

public async Task<IActionResult> Search(SearchProductsParams searchParams)

{

ViewBag.Title = "Поиск";

var searchSpecification = new ProductSpecification().

IncludeCategory();

if (searchParams.StartPrice is not null && searchParams.EndPrice is not null)

{

searchSpecification.WhereInPriceRange(searchParams.StartPrice.Value, searchParams.EndPrice.Value);

}

var isPrimeKeyUsed = false;

if (searchParams.SearchProductId is not null)

{

searchSpecification.WhereId(searchParams.SearchProductId.Value);

isPrimeKeyUsed = true;

}

if (!isPrimeKeyUsed) {

if (searchParams.Name is not null)

{

searchSpecification.WhereName(searchParams.Name);

}

if (searchParams.IsSortByPriceRequired)

{

searchSpecification.SortByPrice();

}

}

searchSpecification.WhereAvailable(searchParams.IsAvailable).WithoutTracking();

ViewBag.Title = "Искомый товар";

ProductsViewModel viewModel = new ProductsViewModel

{

Products = await \_cosmeticProductsRepository.GetProductsAsync(searchSpecification),

ProductCategory = "Косметика",

SearchParams = new SearchProductsParams()

};

\_logger.LogInformation("Products\\Search is executed");

if (!viewModel.Products.Any())

{

\_logger.LogWarning("Search unsuccesful!");

}

return View(viewModel);

}

[Authorize(Roles = "admin")]

[HttpGet]

public async Task<IActionResult> AddProduct()

{

ViewBag.Title = "Добавление продукта";

ViewBag.Categories = new SelectList(await \_categoriesRepository.GetCategoriesAsync(), "Id", "CategoryName");

return View();

}

[Authorize(Roles = "admin")]

[HttpPost]

public async Task<IActionResult> AddProduct(Product newProduct)

{

newProduct.Price = Math.Round(newProduct.Price, 2);

await \_cosmeticProductsRepository.AddProductAsync(newProduct);

return RedirectToAction("ViewProducts", "Products");

}

[Authorize(Roles = "admin")]

[HttpGet]

public async Task<IActionResult> UpdateProduct(int id)

{

ViewBag.Title = "Изменение продукта";

var searchResult = await \_cosmeticProductsRepository.GetProductByIdAsync(id);

if (searchResult is null)

{

return NoContent();

}

ViewBag.Categories = new SelectList(await \_categoriesRepository.GetCategoriesAsync(), "Id", "CategoryName");

return View(searchResult);

}

[Authorize(Roles = "admin")]

[HttpPost]

public async Task<IActionResult> UpdateProduct(int id, Product product)

{

if (id != product.Id)

{

return BadRequest();

}

product.Price = Math.Round(product.Price, 2);

await \_cosmeticProductsRepository.UpdateProductAsync(product);

return RedirectToAction("ViewProducts", "Products");

}

[Authorize(Roles = "admin")]

[HttpGet]

public async Task<IActionResult> DeleteProduct(int id)

{

ViewBag.Title = "Удаление продукта";

var searchResult = await \_cosmeticProductsRepository.GetProductByIdAsync(id);

if (searchResult is null)

{

return NoContent();

}

return View(searchResult);

}

[Authorize(Roles = "admin")]

[ValidateAntiForgeryToken]

[HttpPost]

public async Task<IActionResult> DeleteProduct(int id, IFormCollection collection)

{

var product = await \_cosmeticProductsRepository.GetProductByIdAsync(id);

await \_cosmeticProductsRepository.DeleteProductAsync(product);

return RedirectToAction("ViewProducts", "Products");

}

}

}

Файл CategoriesApiController.cs

﻿using System.Collections.Generic;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using E\_Shop\_Cosmetic.Data.Models;

using E\_Shop\_Cosmetic.Data.Interfaces;

namespace E\_Shop\_Cosmetic.Controllers.Api

{

[Route("api/[controller]")]

[ApiController]

public class CategoriesApiController : ControllerBase

{

private readonly ICategoriesRepository \_categoriesRepository;

public CategoriesApiController(ICategoriesRepository categoriesRepository)

{

\_categoriesRepository = categoriesRepository;

}

// GET: api/CategoriesApi

[HttpGet]

public async Task<ActionResult<IEnumerable<Category>>> GetCategories()

{

return new ActionResult<IEnumerable<Category>>(await \_categoriesRepository.GetCategoriesAsync());

}

// GET: api/CategoriesApi/5

[HttpGet("{id}")]

public async Task<ActionResult<Category>> GetCategory(int id)

{

var category = await \_categoriesRepository.GetCategoryByIdAsync(id);

if (category == null)

{

return NotFound();

}

return category;

}

// PUT: api/CategoriesApi/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutCategory(int id, Category category)

{

if (id != category.Id)

{

return BadRequest();

}

await \_categoriesRepository.UpdateCategoryAsync(category);

return NoContent();

}

// POST: api/CategoriesApi

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<Category>> PostCategory(Category category)

{

await \_categoriesRepository.AddCategoryAsync(category);

return CreatedAtAction("GetCategory", new { id = category.Id }, category);

}

// DELETE: api/CategoriesApi/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteCategory(int id)

{

var category = await \_categoriesRepository.GetCategoryByIdAsync(id);

if (category == null)

{

return NotFound();

}

return NoContent();

}

}

}

Файл ProductsApiController.cs

﻿using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using E\_Shop\_Cosmetic.Data;

using E\_Shop\_Cosmetic.Data.Models;

using E\_Shop\_Cosmetic.Data.Interfaces;

using Microsoft.AspNetCore.Authorization;

namespace E\_Shop\_Cosmetic.Controllers.Api

{

[Route("api/[controller]")]

[ApiController]

public class ProductsApiController : ControllerBase

{

private readonly IProductsRepository \_productsRepository;

public ProductsApiController(IProductsRepository productsRepository)

{

\_productsRepository = productsRepository;

}

// GET: api/ProductsApi

[HttpGet]

public async Task<ActionResult<IEnumerable<Product>>> GetProducts()

{

var products = await \_productsRepository.GetProductsAsync();

return new ActionResult<IEnumerable<Product>>(products);

}

// GET: api/ProductsApi/MinPrice

[HttpGet("MinPrice")]

public async Task<JsonResult> GetMinPrice()

{

var products = await \_productsRepository.GetProductsAsync();

return new JsonResult(new {Min = products.Min(p => p.Price) });

}

// GET: api/ProductsApi/MaxPrice

[HttpGet("MaxPrice")]

public async Task<JsonResult> GetMaxPrice()

{

var products = await \_productsRepository.GetProductsAsync();

return new JsonResult(new { Max = products.Max(p => p.Price) });

}

// GET: api/ProductsApi/MaxPrice

[HttpGet("MinMaxPrice")]

public async Task<JsonResult> GetMinMaxPrice()

{

var products = await \_productsRepository.GetProductsAsync();

return new JsonResult(new { Max = products.Max(p => p.Price), Min = products.Min(p => p.Price) });

}

// GET: api/ProductsApi/5

[HttpGet("{id}")]

public async Task<ActionResult<Product>> GetProduct(int id)

{

var product = await \_productsRepository.GetProductByIdAsync(id);

if (product == null)

{

return NotFound();

}

return product;

}

// PUT: api/ProductsApi/5

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details, see https://go.microsoft.com/fwlink/?linkid=2123754.

[HttpPut("{id}")]

[Authorize(Roles = "admin")]

public async Task<IActionResult> PutProduct(int id, Product product)

{

if (id != product.Id)

{

return BadRequest();

}

await \_productsRepository.UpdateProductAsync(product);

return NoContent();

}

// POST: api/ProductsApi

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details, see https://go.microsoft.com/fwlink/?linkid=2123754.

[HttpPost]

[Authorize(Roles = "admin")]

public async Task<ActionResult<Product>> PostProduct(Product product)

{

await \_productsRepository.AddProductAsync(product);

return CreatedAtAction("GetProduct", new { id = product.Id }, product);

}

// DELETE: api/ProductsApi/5

[HttpDelete("{id}")]

[Authorize(Roles = "admin")]

public async Task<ActionResult<Product>> DeleteProduct(int id)

{

var product = await \_productsRepository.GetProductByIdAsync(id);

if (product == null)

{

return NotFound();

}

await \_productsRepository.DeleteProductAsync(product);

return product;

}

private async Task<bool> ProductExists(int id)

{

var products = await \_productsRepository.GetProductsAsync();

return products.Any(e => e.Id == id);

}

}

}