Дипломная работа

Разработка интернет-магазина по продаже компьютерной техники

Содержание

	Введение……	2
1.	Определение целей и задач проектирования ИС……..	7
	1.1 Назначение и цели создания…….	7
	1.2 Способы и средства разработки интернет-магазина……..	10
2.	Реализация информационной системы……..	16
	2.1 Выбор и создание структуры проекта……..	16
	2.2 Реализация информационной системы……	19
3.	Руководство пользователя……...	21
4.	Тестирование автоматизированной системы……	29
	4.1 Тестирование на надежность……	30
	4.2 Тестирование на сопровождаемость……	30
	4.3 Тестирование на практичность…….	31
	4.4 Тестирование на эффективность……..	32
	4.5 Тестирование на мобильность……..	32
	4.6 Тестирование на функциональность……	33
5.	Заключение ……..	34
6.	Список использованных источников…….	35
	Приложение А……..	37

ВВЕДЕНИЕ

Успех любого бизнеса во многом зависит от того, насколько эффективно компания доводит свою информацию до клиентов и партнеров.

Создание интернет-сайта и размещение его в Интернете – один из альтернативных методов позиционирования компании и информирования целевой аудитории.

Именно в Интернете многие ищут подробную, и свежую информацию, на основе которой можно получить представление об интересующей компании, товарах и услугах.

Сейчас трудно представить крупную компанию без интернет-сайта. Наличие у компании хорошего сетевого представительства не только усиливает положительный образ фирмы, но и говорит в пользу надежности и основательности компании, создает определенный имидж и впечатление.

В настоящее время наиболее популярным видом виртуальной торговли является интернет-магазин. Интернет-магазин обычно содержит наглядный и красочный каталог предоставляемых товаров, с их достаточным описанием и указанием цены, что позволяет заинтересовать потенциального покупателя, помочь ему сделать выбор и, в итоге, совершить покупку.

Использование Интернет-магазинов для продажи товаров весьма эффективно как в России, так и за рубежом. Многие пользователи компьютеров уже активно используют поиск информации о необходимых им товарах в сети Интернет. Наличие Интернет-магазина и возможности заказать товар в режиме «онлайн» улучшают имидж фирмы, вызывают интерес у активного сегмента пользователей, а также фирм–партнеров. Таким образом, Интернет-магазины являются хорошим способом продвижения своих товаров на рынке.

Сейчас существует достаточно много фирм, реализующих компьютерную технику с доставкой на дом. Подавляющее их большинство имеет свои сайты в сети Интернет и на многих таких сайтах размещены

Интернет-магазины. Продажи компьютеров и комплектующих с использованием Интернет-магазинов увеличивают объем реализуемой продукции, повышают популярность магазина.

Фирма по продаже компьютерной техники в современных реализациях должна иметь собственный представительский веб-сайт. Если в Интернете присутствие фирмы представлено только группами в социальных сетях, это не позволяет активно расширять продажи реализуемой продукции. Это делает актуальной задачу разработки нового сайта и Интернет-магазина.

Внедряемая информационная система должна обеспечивать следующие функции:

1) учет номенклатуры и запаса компьютерных товаров, реализуемых магазином;
2) формирование специализированной управленческой отчетности.
3) поддержка регистрации посетителей Интернет-магазина;
4) работа с заказом:

- прием заказа на приобретение товаров от зарегистрированного покупателя;
- поддержка обработки заказов покупателей менеджером магазина;

5) предоставление покупателю дисконтной скидки в зависимости от накопленного объема покупок в магазине;

6) формирование для покупателя оповещений о состоянии заказа и о возможности получения дисконтной скидки в зависимости;

7) прием от привилегированного покупателя заявки на товар, отсутствующий в наличии на момент обращения.

С целью выявления возможных способов решения поставленной задачи были проанализированы следующие варианты:

1) заказ в Web-студии индивидуальной разработки Интернет-магазина;

2) использование готовой платформы Интернет-магазина;

3) привлечение фрилансера для разработки и ведения Интернет-магазина;

4) подключение к SaaS – платформе, предоставляющей услуги Интернет-магазина на основе аренды;

5) аренда Интернет-магазина;

6) самостоятельная разработка Интернет-магазина.

Даже при хорошей организации процесса продаж отсутствие эффективных инструментов работы с клиентами не позволит магазину достичь достаточного объемов продаж и прибыли.

Увеличение объемов сбыта товаров в настоящее время затруднено следующими обстоятельствами:

- присутствует высокий уровень конкуренции с другими магазинами;
- наличие у организации единственного магазина обуславливает недостаточный охват потребителей различных районов города;
- имеющийся персонал организации при реализации товаров затрачивает много времени и усилий на ознакомление посетителей магазина с характеристиками продукции при относительно невысоком обороте, получаемом с прямых продаж;
- рутинные операции по учету продаж, ведению реестра постоянных покупателей, формированию плана закупок также требуют от персонала дополнительных времени и трудозатрат;
- сходство ассортимента и уровня цен продукции, реализуемой магазином с ассортиментом и продукцией многих других магазинов «обезличивает» фирму владельца магазина, снижает тем самым ее привлекательность для потенциальных покупателей и партнеров.

Широкое распространение в современном обществе средств вычислительной техники и наличие у потенциальных покупателей доступа к сети Интернет позволяют использовать для преодоления перечисленных выше проблем средства автоматизации взаимодействия с клиентами магазинов.

В настоящее время такая автоматизация в магазине компьютерной техники реализована только на уровне размещения на сайте организации сведений о реализуемой продукции и ее ценам. Подобный сайт-визитка не позволяет активно привлекать клиентов и расширять продажи реализуемой продукции.

Для устранения указанных недостатков, организация намерена расширить свое присутствие в сети Интернет путем создания Интернет-магазина, предоставляющего покупателям наличие следующих преимуществ:

- возможность размещения заказа через Интернет-магазин;
- возможность размещения заявок на приобретение товаров, отсутствующих на момент посещения Интернет-магазина в наличии (для их учета при формировании плана закупок магазина).

Создание такого Интернет-магазина позволит снизить нагрузку на персонал магазина, повысит число привлекаемых посетителей, сократит среднее время обслуживания покупателя.

При успешном внедрении Интернет-магазина в работу фирмы предполагается повышение уровня продаж фирмы и улучшения ее деловой репутации на рынке компьютерной техники. Разработку данного Интернет-магазина можно также рассматривать как самостоятельную разработку типового коммерческого программного решения, реализуемого в дальнейшем владельцам других магазинов сходного профиля.

Теперь сформулируем задачи проекта. Приложение должно иметь графический интерфейс со следующим функционалом, который повсеместно используется в интернет-магазинах:

Онлайн каталог товаров, который пользователи могут просматривать по категориям и страницам, корзину для покупок, куда пользователи могут добавлять и удалять товары, и форму оплаты, где пользователи могут вводить сведения, связанные с доставкой. Кроме того, иметь административную область, которая включает в себя средства создания, чтения, обновления и удаления для управления каталогом товаров, и защитить ее так, чтобы изменения могли вносить только зарегистрированные администраторы.

Для интернет-магазина целесообразно разработать именно динамический сайт, так как пользователю необходимо в реальном времени просматривать существующий товар в данном магазине, согласно каталогу, а для этого технология ASP.NET отлично подходит.

Задачи Проекта:

1) провести подробный анализ бизнес-процессов предметной области;
2) определить требования к внедряемому Интернет-магазину;
3) выбрать способ и средства для реализации Интернет-магазина;
4) спроектировать информационную систему Интернет-магазина.

Цель дипломного проекта: повышение эффективности деятельности фирмы по продаже компьютерной техники путем создания Интернет-магазина. Для этого потребуется разработать веб-приложение интернет-магазина на основе технологии ASP.NET.

1. ОПРЕДЕЛЕНИЕ ЦЕЛЕЙ И ЗАДАЧ ПРОЕКТИРОВАНИЯ ИС

1.1 Назначение и цели создания

Интернет-магазин - это привычный каждому пользователю интернета web-сайт, представляющий информацию о товарах в удобном структурированном виде.

Интернет-магазины создаются с применением систем управления содержимым сайтов, оснащенных необходимыми модулями. Крупные интернет-магазины работают на специально разработанных или адаптированных типовых системах управления.

Система управления содержимым сайта интернет-магазина может быть коробочным товаром, самостоятельно устанавливаемым на хостинг-площадку, может быть частной разработкой web-студии, ей же обслуживаемой, или может быть программным сервисом, предоставляемым с помесячной оплатой.

С технической точки зрения интернет-магазин - это совокупность web-витрины и торговой системы - фронт-системы и бэк-офиса. Web-витрина предоставляет интерфейс к базе данных продаваемых товаров (в виде каталога, прайс-листа), работает с виртуальной торговой тележкой, оформляет заказы и регистрирует покупателя, предоставляет помощь покупателю в онлайновом режиме, передает информацию в торговую систему и обеспечивает безопасность личной информации покупателя. Далее торговая система осуществляет автоматическую обработку поступающих заказов - резервирует товар на складе, контролирует оплату и доставку товара. В общем случае основные функции интернет-магазина - это информационное обслуживание покупателя, обработка заказов, проведение платежей, а также сбор и анализ различной статистической информации. Как было упомянуто выше, программный комплекс управления интернет-магазином позволяет формировать и интерфейс с покупателем, и функциональные возможности интернет-магазина, исходя из потребностей компании.

Для разработки нашего магазина необходимо выбрать ближайшие аналоги разрабатываемого программного средства и оценим, как они решают поставленную проблему. Основными способами реализации Интернет-магазина являются:

- заказ в Web-студии индивидуальной разработки Интернет-магазина;
- использование готовой платформы Интернет-магазина;
- привлечение фрилансера для разработки и ведения Интернет-магазина;
- подключение к SaaS – платформе, предоставляющей услуги Интернет-магазина на основе аренды;
- аренда Интернет-магазина;
- самостоятельная разработка Интернет-магазина.

Рассмотрим критерии для оценки способа реализации Интернет-магазина. В качестве таких критериев можно рассматривать:

- стоимость создания и поддержки работы Интернет-магазина;
- длительность разработки;
- возможность сопровождения после разработки.

Результаты сравнения различных вариантов разработки приведены в таблицах 1. и 2.

Таблица 1- Описание вариантов реализации Интернет-магазинов

Вариант реализации Интернет-магазина	Ориентировочная стоимость	Особенности варианта
Заказ в Web-студии индивидуальной разработки Интернет-магазина	От 30 - 90 т.р.	Достоинства: - юридически оформленные отношения; - наличие ответственных за ведение работ лиц; - чёткие сроки работ; - наличие гарантии на выполненные работы. Недостатки: - стоимость.
Использование готовой платформы Интернет-магазина	От 15 т.р.	Достоинства: малые трудозатраты. Недостаток: необходимость самостоятельной адаптации программного решения
Привлечение фрилансера для разработки и ведения Интернет-магазина	От 10 т.р.	Достоинства: малые трудозатраты при относительно высокой квалификации. Недостаток: потенциальная частая смена персонала по разработке и сопровождению сайта.
Подключение к SaaS – платформе, предоставляющей услуги Интернет-магазина на основе аренды	От 15 т.р. в год	Достоинства: 1. Отсутствие затрат на покупку лицензии программного обеспечения для интернет магазина. 2. Отсутствие затрат на разработку интернет магазина. 3. Экономия времени связанного с разработкой интернет магазина. Недостатки: 1.Отсутствие доверия к поставщикам услуг 2.SaaS-услуги мало модернизируемы
Аренда Интернет-магазина	От 6 т.р. в год	Достоинство: возможность выкупа в собственность Недостатки: трудность смены функционала.
Самостоятельная разработка Интернет-магазина	-	Достоинство: возможность гибко адаптировать возможности магазина. Недостатки: необходимость затрат времени и наличия квалификации по разработке Web-приложений.

По результатам анализа вариантов автоматизации было принято решение

разрабатывать Интернет-магазин собственными силами, так как это позволит при относительно небольших затратах проводить собственные исследования необходимой степени автоматизации магазина, самостоятельно формировать набор функций, реализуемых Интернет-магазином.

1.2 Способы и средства разработки интернет-магазина

При проектировании Web-сайта необходимо принять проектные решения по программному, техническому и технологическому обеспечению задачи.

Выбор средства проектирования информационной системы основывается, прежде всего, на технологии проектирования, выбранной для данной системы и подходящей для описываемой предметной области.

Методологии, технологии и инструментальные средства проектирования (CASE-средства) составляют основу проекта любой ИС. Наибольшее распространение получили следующие два подхода к проектированию информационных систем: структурный и объектно-ориентированный подходы.

Сущность структурного подхода к проектированию ИС заключается в ее декомпозиции на автоматизируемые функции: система разбивается на функциональные подсистемы, которые в свою очередь делятся на подфункции, подразделяемые на задачи и так далее. При структурном подходе модели процессов и модели данных создаются отдельно. Данный подход применяется к непосредственному проектированию информационных систем.

Сущность объектно-ориентированного подхода к проектированию информационных систем заключается в объектной декомпозиции. При этом статическая структура системы описывается в терминах объектов и связей между ними, а поведение системы описывается в терминах обмена

сообщениями между объектами. При объектно-ориентированном подходе модели процессов и модели данных создаются параллельно. Данный подход применяется к разработке сложных корпоративных систем с использованием объектно-ориентированного метода программирования.

Для проектирования системы использованы следующие CASE-средства: AllFusion Process Modeler и Rational Rose. У данных средств больше возможностей для осуществления проектирования системы.

AllFusion Process Modeler – средство функционального моделирования, реализующее методологии IDEF0 (SADT), IDEF3 и DFD.

Rational Rose – семейство объектно-ориентированных CASE-средств, предназначенных для автоматизации процессов анализа и проектирования ПО, а также для генерации кодов на различных языках и выпуска проектной документации.

Диаграмму деятельности процесса «Авторизация» можно увидеть на рисунке 1.

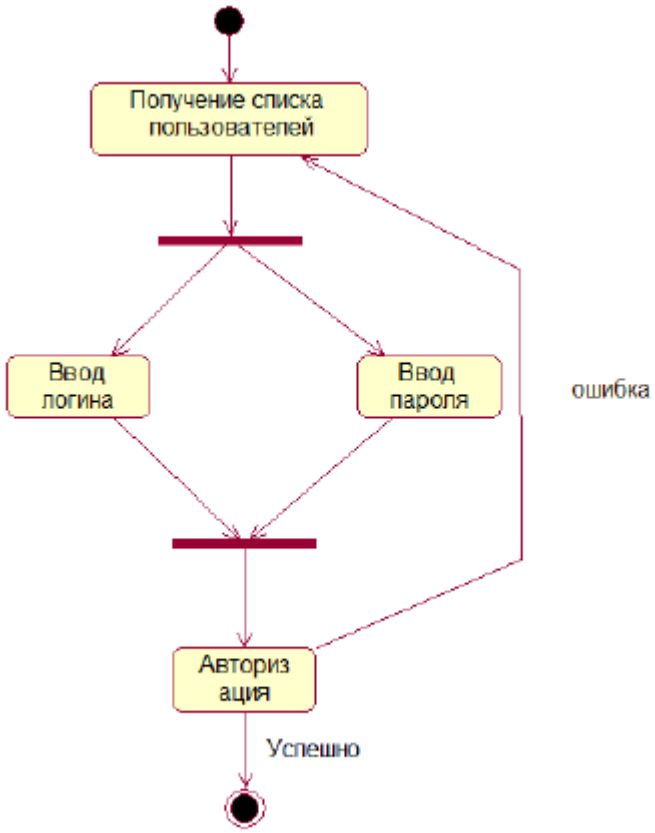

Рисунок 1 – Диаграмма деятельности процесса «Авторизация»

Web-сайт разрабатывается с применением следующих программных средств:

1) язык программирования C#;
2) Microsoft Visual Studio 2017;
3) СУБД MS SQL.

C# является основным языком разработки программ на платформе .NET. Данный язык является прямым наследником двух самых удачных языков программирования – C и C++. Так же имеет тесные «родственные» связи с языком: Java.

C# разработан для удовлетворения таких потребностей, как межъязыковая возможность взаимодействия, необходимая в построении крупных, распределенных программных систем, а также полная интеграция с платформой Windows (так как данная платформа является самой распространенной в мире) и многих других.

Несмотря на то, что C# является самодостаточным языком программирования, как говорится в книге Шилдта [2], у него имеется особая взаимосвязь со средой выполнения asp.net mvc framework.

Фреймворки – это программная платформа, призванные упростить и ускорить решение типовых задач, характерных для конкретного языка программирования.

Платформа ASP.NET MVC базируется на взаимодействии трех компонентов: контроллера, модели и представления. Контроллер принимает запросы, обрабатывает пользовательский ввод, взаимодействует

с моделью и представлением и возвращает пользователю результат обработки запроса.

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных.

Для работы с базами данных в используемой мной среде разработке Visual Studio имеются встроенные решения. В данном дипломном проекте осуществляется работа с СУБД MS SQL Server.

2. ОБОСНОВАНИЕ ИСПОЛЬЗОВАНИЯ ПЛАТФОРМЫ ASP.NET MVC

Для решения задачи написания web-приложения интернет-магазина на языке C# была выбрана платформа ASP.NET MVC.

Инфраструктура MVC Framework построена в виде набора независимых компонентов, которые удовлетворяют интерфейсу .NET или созданы на основе абстрактного базового класса. Компоненты, подобные системе маршрутизации, механизму визуализации и фабрике контроллеров, можно легко заменять другими компонентами с собственной реализацией. Преимуществом является жесткий контроль над HTML и HTTP.

Инфраструктура ASP.NET MVC генерирует ясный и соответствующий стандартам код разметки. Ее встроенные вспомогательные методы HTML

производят соответствующий стандартам вывод, но существует также гораздо более значимое философское изменение по сравнению с Web Forms. Вместо генерации громадного объема трудно поддающейся управлению HTML разметки инфраструктура MVC Framework стимулирует создание простых и элегантных элементов, оформленных стилями CSS.

Конечно, если действительно требуется использовать некоторые готовые виджеты для таких сложных элементов пользовательского интерфейса, как окна выбора даты или каскадные меню, применяемый в ASP.NET MVC подход "никаких специальных требований" к разметке, позволяет легко использовать наилучшие библиотеки для построения пользовательских интерфейсов, подобные jQuery или Bootstrap CSS. К примеру, библиотека jQuery настолько эффективно поддерживается, что поставляется в качестве встроенной части стандартного шаблона проекта ASP.NET MVC в Visual Studio наряду с другими популярными библиотеками, такими как Bootstrap, Knockout и Modernizr.

Сгенерированные ASP.NET MVC страницы не содержат никаких данных View State, поэтому они меньше типовых страниц ASP.NET Web Forms по размеру. Несмотря на современные быстрые соединения, такая экономия трафика по-прежнему повышает комфорт конечного пользователя и помогает сократить затраты, связанные с запуском популярных веб-приложений.

Инфраструктура ASP.NET MVC работает в тесном сотрудничестве с HTTP. При этом имеется контроль над запросами, передаваемыми между браузером и сервером, что позволяет очень точно настраивать пользовательский интерфейс по своему усмотрению. Технология AJAX проста, и ей не нужны какие-то автоматические обратные отправки запросов для взаимодействия с состоянием клиентской стороны.

Основной целью концепции MVC (Рисунок 2) является отделение бизнес-логики (модели) от ее визуализации (представления, вида). Удобство ее заключается в том, что ни одно изменения внешнего вида никаким образом

не затрагивает логики моделей, а в свою очередь к любой модели может быть подключено сколько угодно представлений. Данная концепция разделяет приложение на три основные части:

- Модель – основной структурный элемент, отвечает за описание хранящихся данных и знаний, является хранителем реакций на запросы;
- Представление – визуализатор, отвечает за отображение данных хранящихся в моделях в должном виде;
- Контроллер – связующее звено между пользователем и системой, отвечает за вводимые пользователем данные и за реакцию на них, основанную на модели и представлении.



Рисунок 2 – шаблон MVC

2. РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ

2.1 Выбор и создание структуры проекта

Первым этапом реализации поставленных задач было создание решения для Visual Studio 2017, состоящего из трех проектов. Для рассматриваемого примера приложения требуются три проекта (рисунок 3).

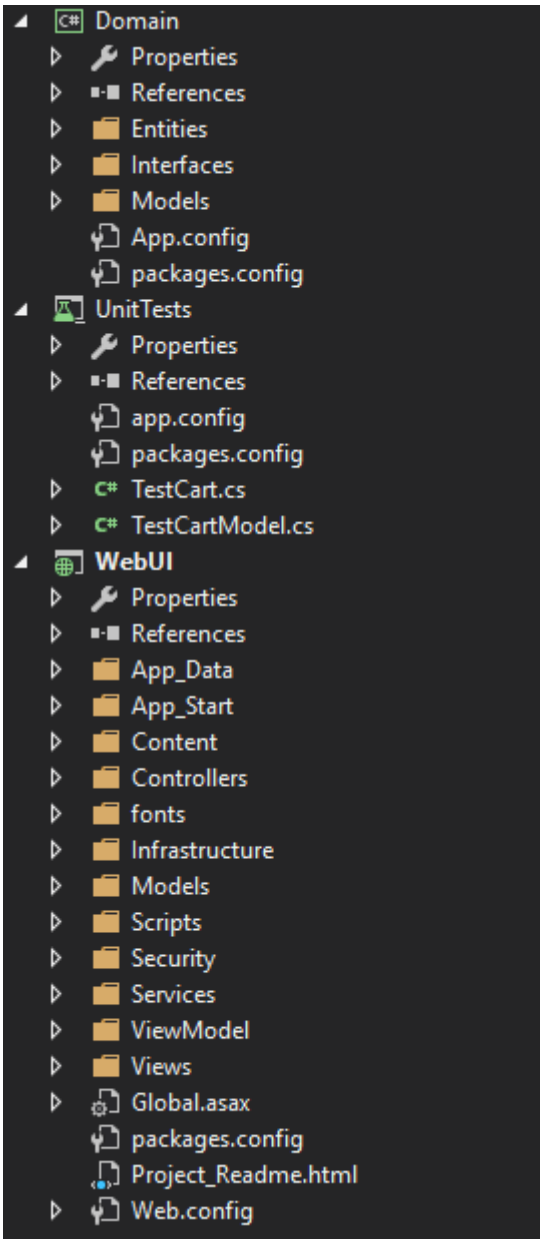

Рисунок 3 – Структура решения Visual Studio

Разберем подробнее представленные (рисунок 9) проекты:

- Domain содержит сущности и логику предметной области;

настраивается на обеспечение постоянства посредством хранилища, которое создано с помощью инфраструктуры Entity Framework;

- WebUI содержит контроллеры и представления; выступает в качестве пользовательского интерфейса;
- UnitTests содержит модульные тесты для других двух проектов.

Также, используя менеджер NuGet, были подключены следующие пакеты:

- "Antlr" version="3.4.1.9004" - это инструмент языка, который обеспечивает основу для построения распознавателей, переводчиков, компиляторов и переводчиков из грамматических описаний, содержащих действия на различных целевых языках.
- "bootstrap" version="3.0.0" для более быстрой и легкой веб-разработки.
- "EntityFramework" version="6.1.3" рекомендуемая Microsoft технология доступа к данным для новых приложений.
- "jQuery" version="1.10.2" - это быстрая и лаконичная библиотека JavaScript, которая упрощает перемещение HTML-документов, обработку событий, анимацию и взаимодействие Ajax для быстрой веб-разработки.
- "Microsoft.AspNet.Identity.Core" version="2.2.1" этот пакет содержит русские вспомогательные сборки для библиотек ASP.NET Identity Core.
- "Microsoft.AspNet.Identity.EntityFramework" version="2.2.1" этот пакет содержит русские вспомогательные сборки для библиотек ASP.NET Identity Entity Framework.
- "Microsoft.AspNet.Identity.Owin" version="2.2.1"
- "Microsoft.AspNet.Mvc" version="5.2.3" этот пакет содержит сборки среды выполнения для ASP.NET MVC. ASP.NET MVC дает мощный, основанный на шаблонах способ создания динамических веб-сайтов, который позволяет чистое разделение проблем, и что дает полный контроль над разметкой.
- "Microsoft.AspNet.Razor" version="3.2.3" этот пакет содержит сборки среды выполнения для ASP.NET веб-страницы. ASP.NET веб-страницы и новый синтаксис Razor обеспечивают быстрый, лаконичный, чистый и легкий способ объединения серверного кода с HTML для создания динамического веб-контента.

- "Microsoft.Net.Compilers" version="1.0.0" этот пакет может быть использован для компиляции кода для любой платформы.
- "Microsoft.Owin" version="3.0.1" предоставляет набор вспомогательных типов и абстракций для упрощения создания компонентов OWIN.
- "Modernizr" version="2.6.2" - это небольшая и простая библиотека JavaScript, которая помогает использовать преимущества новых веб-технологий (CSS3, HTML5), сохраняя уровень контроля над старыми браузерами, которые еще не поддерживают эти новые технологии.
- "Newtonsoft.Json" version="6.0.4"
- "Ninject" version="3.2.2.0"
- "Ninject.MVC5" version="3.2.1.0"
- "Owin" version="1.0" позволяет отвязывать веб-приложения от веб-серверов. Он определяет стандартный способ использования связующего ПО при обработке запросов и соответствующих ответов.

После этого необходимо настроить зависимости между проектами с помощью ссылок в окне Solution Explorer.

- WebUI должен зависеть от Domain
- UnitTests должен зависеть от Domain и WebUI

Были созданы следующие директории в проектах:

- App_Data – в эту папку помещаются закрытые данные, такие как XML-файлы или базы данных. В данном случае в нашем проекте там находится база данных магазина RcShop – RcBD.mdf
  - App_Start – эта папка содержит ряд основных настроек конфигурации для проекта, в том числе определение маршрутов и фильтров, а также пакетов содержимого
  - Content – сюда помещается статическое содержимое, такое как CSS-файлы и изображения
  - Controllers – сюда помещаются классы контроллеров
  - Models – сюда помещаются классы моделей представлений и моделей

предметной области

- Scripts – эта папка предназначена для хранения библиотек JavaScript, используемых в приложении

В сформированной структуре директорий велась вся дальнейшая работа по написанию кода веб-приложения. Реализация этих методов также представлена в приложении А.

2.2 Реализация информационной системы

Для интернет-магазина компьютерной техники потребуется создать базу данных в которой будет храниться информация о каталоге. В базе данных нам нужна только одна таблица, которая будет использоваться для хранения подробной информации о товаре.

Таблица оформленных заказов:

```
CREATE TABLE [dbo].[Orders] (
    [Id]                        INT          IDENTITY (1, 1) NOT NULL,
    [DeliveryAddress_AdressLine] NVARCHAR (MAX) NULL,
    [DeliveryAddress_City]      NVARCHAR (MAX) NULL,
    [DeliveryAddress_ZipCode]   INT          NOT NULL,
    CONSTRAINT [PK_dbo.Orders] PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

Таблица связи товара и заказов:

```
CREATE TABLE [dbo].[OrderItems] (
    [Id]       INT IDENTITY (1, 1) NOT NULL,
    [ProductId] INT NOT NULL,
    [OrderId]   INT NOT NULL,
    [Quantity]  INT NOT NULL,
    CONSTRAINT [PK_dbo.OrderItems] PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_dbo.OrderItems_dbo.Orders_OrderId] FOREIGN KEY
([OrderId]) REFERENCES [dbo].[Orders] ([Id]) ON DELETE CASCADE,
    CONSTRAINT [FK_dbo.OrderItems_dbo.Products_ProductId] FOREIGN KEY
([ProductId]) REFERENCES [dbo].[Products] ([Id]) ON DELETE CASCADE
);
```

Таблица товаров:

```
CREATE TABLE [dbo].[Products] (
    [Id]        INT            IDENTITY (1, 1) NOT NULL,
    [Name]      NVARCHAR (MAX)  NOT NULL,
    [UnitPrice]  DECIMAL (18, 2) NOT NULL,
    [PictureRef] NVARCHAR (MAX)  NULL,
    CONSTRAINT [PK_dbo.Products] PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для работы с полным функционалом веб-приложения пользователю необходимо иметь:

- устройство на ОС Windows/Linux/OSX/Android;
- интернет-браузер последней версии;
- доступ к сети Интернет.

После этого откроется главная страница сайта разработанного интернет-магазина (рисунок 4). В шапке сайта находятся его основные разделы, список товаров представлен в основной части страницы. Пролистать страницы с доступным товаром можно переключением соответствующих кнопок внизу страницы (рисунок 5). При выборе товара для покупки, чтобы поместить его в корзину, необходимо нажать кнопку «В корзину».

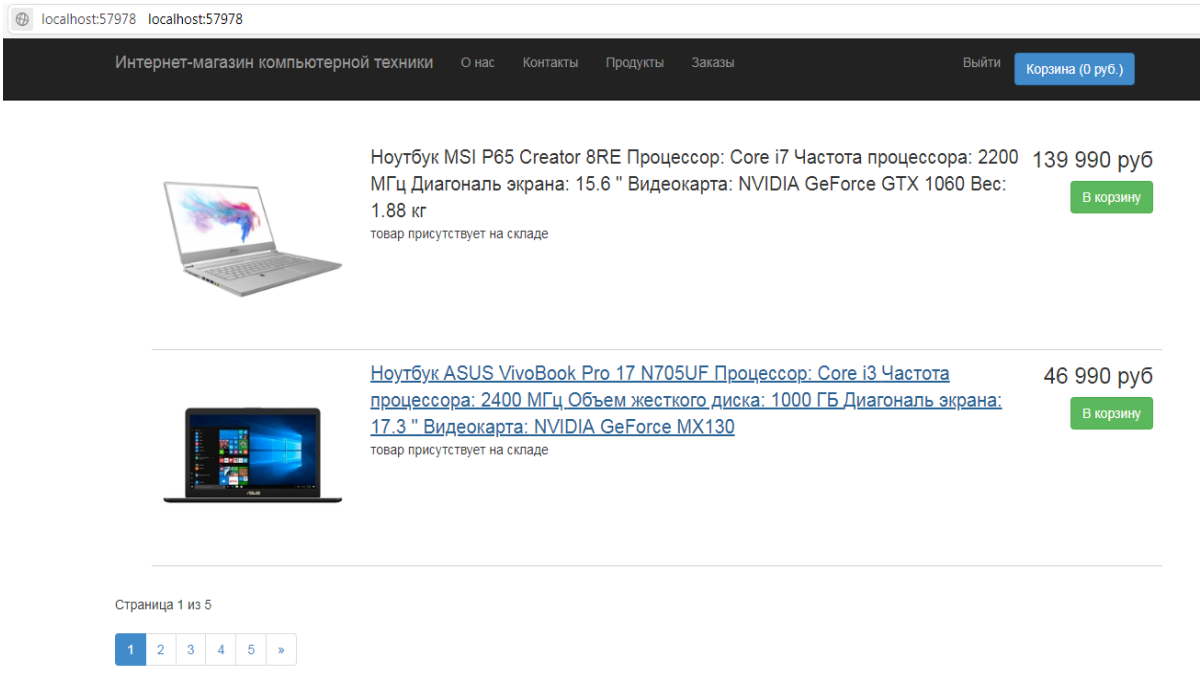

Рисунок 4 – Главная страница сайта

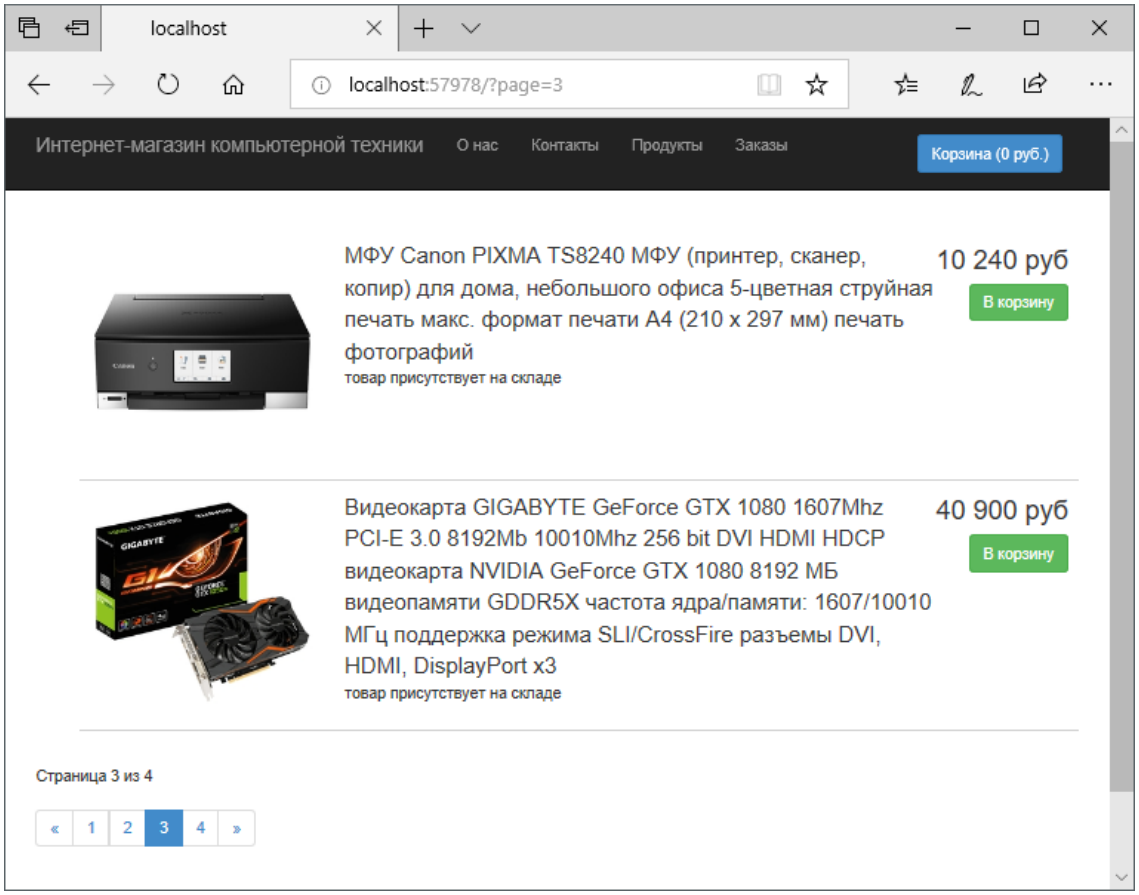

Рисунок 5 – Переход по страницам списка товаров

После выполнения необходимых действий (завершения покупок), пользователь может оформить заказ. Для этого необходимо перейти в раздел сайта «Корзина» нажатием на соответствующую кнопку в шапке сайта.

После этого откроется страница раздела (рисунок 12), где потребуется ввести данные в поля:

- Адрес дома;
- Город (РФ);
- Почтовый индекс.

После этого нажать «Заказать», проверить введенную информацию, после чего подтвердить действие (рисунок 13).

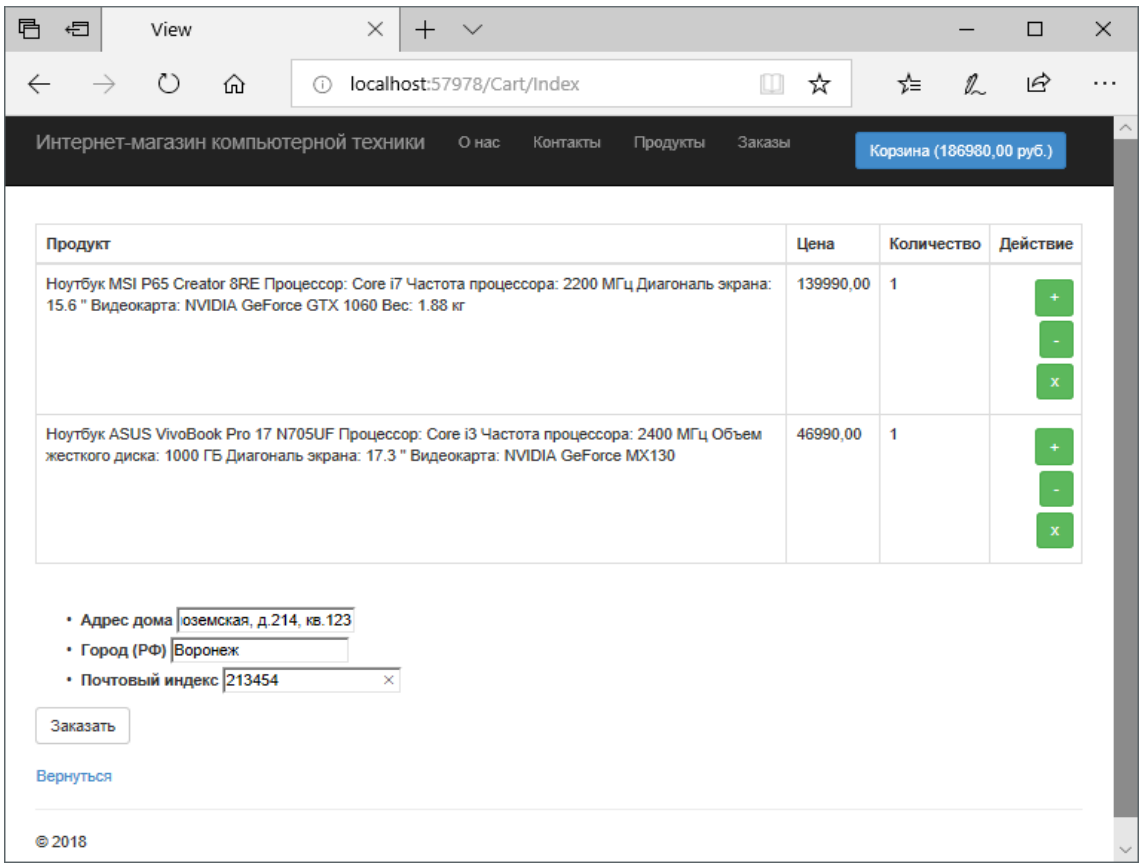

Рисунок 6 – Корзина

Для того что бы посмотреть товар в корзине, необходимо нажать на «Корзину», после чего откроется список

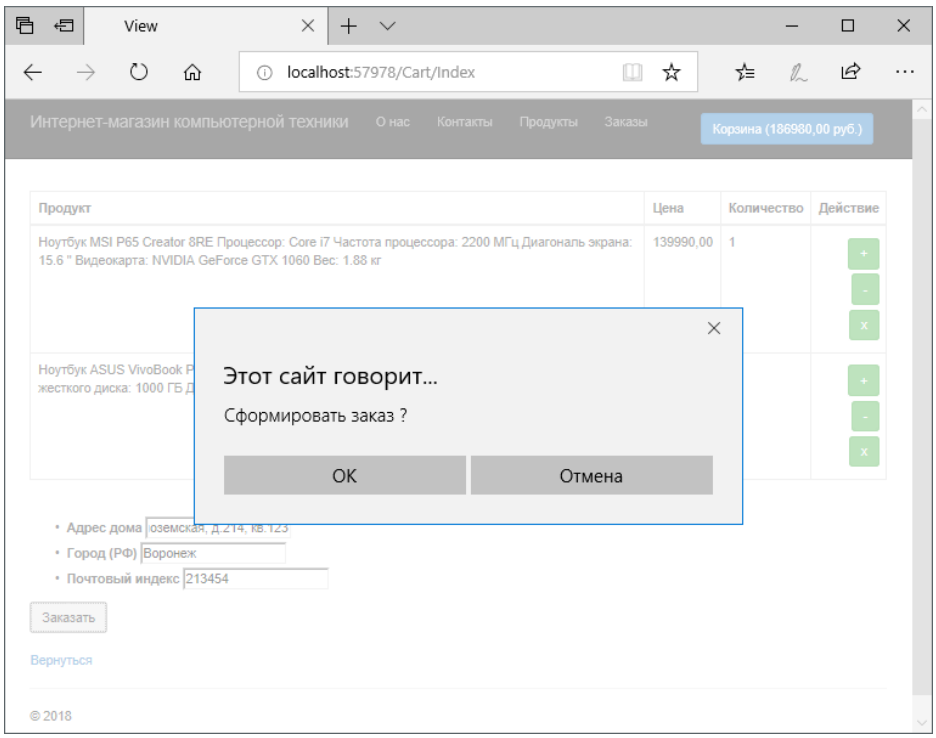

Рисунок 7 – Оформление заказа из корзины

После подтверждения (рисунок 7), заказ формируется и заносится в БД, таблица оформленных заказов может просматриваться администратором. Пользователь же получает извещение о том, что заказ успешно оформлен (рисунок 8).

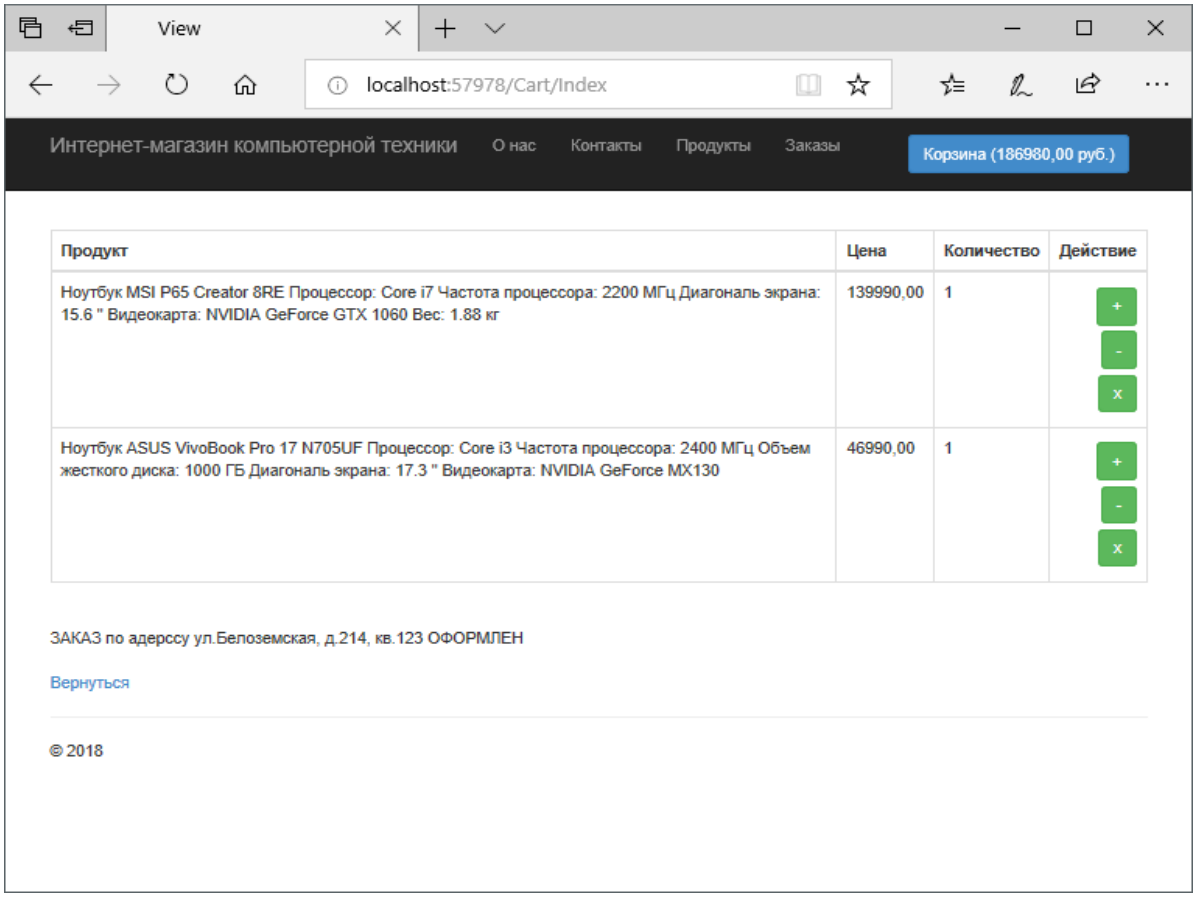

Рисунок 8 – Извещение о успешном оформлении заказа

Для просмотра заказов, управления списком характеристик продуктов, необходимо зайти на сайт под учетной записью администратора (рисунок 9).

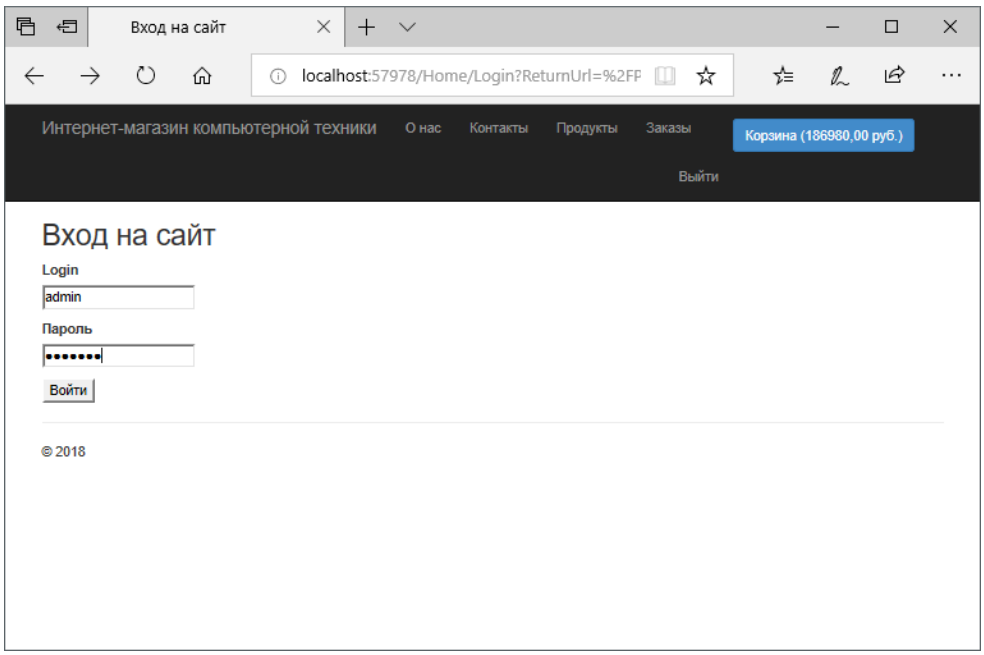

Рисунок 9 – Вход под администраторской учетной записью

Для этого необходимо перейти в раздел «Продукты» и ввести логин и пароль учетной записи администратора. После этого откроется список продуктов в БД, над которым администратору можно проводить следующие операции (рисунок 10):

- Добавить новый товар;
- Редактировать данные о товаре (рисунок 10);
- Вывести детализацию характеристик товара (рисунок 11);
- Удалить товар из БД;
- Вывести детализацию оформленных клиентами заказов (рисунок 13).

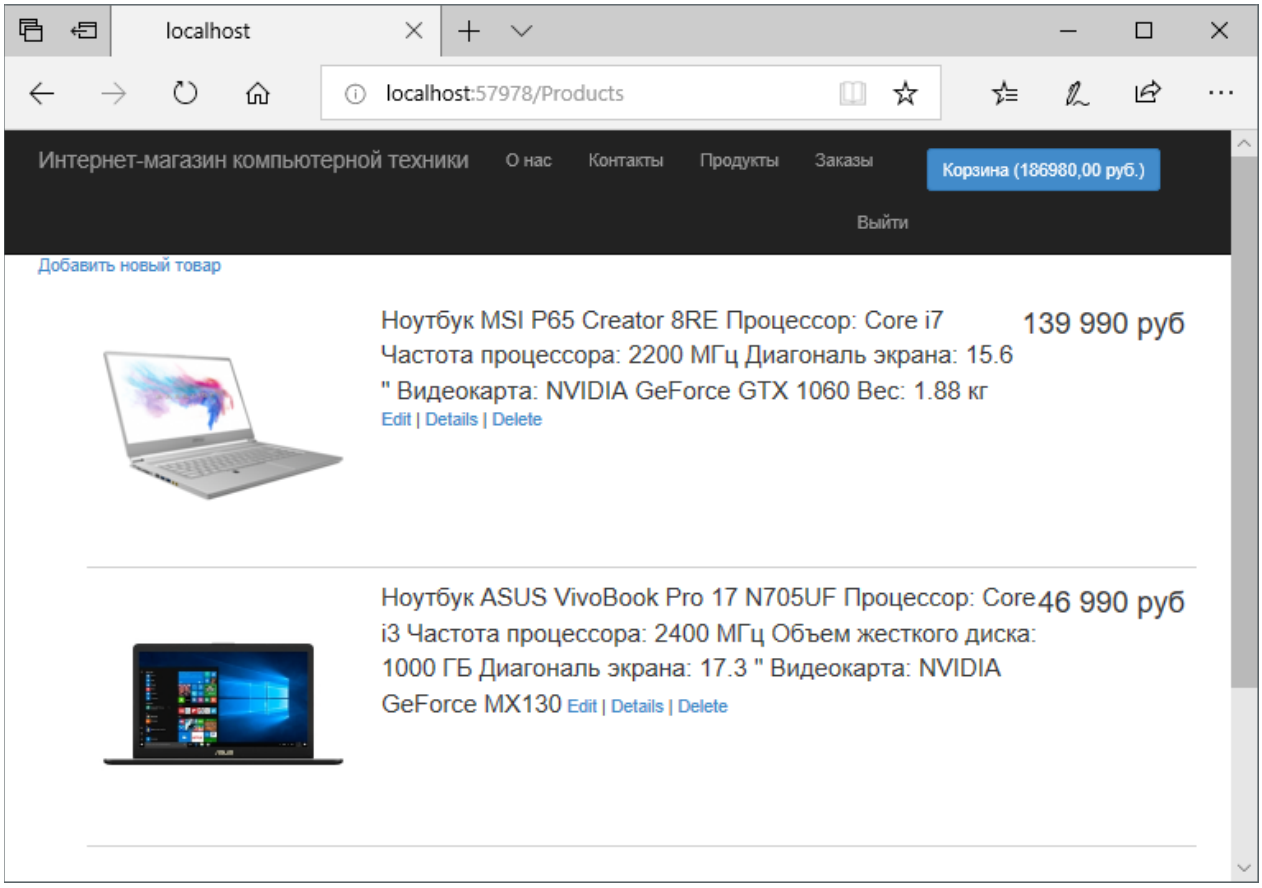

Рисунок 10 – Отображение доступных функций редактированию списка товаров из-под учетной записи администратора

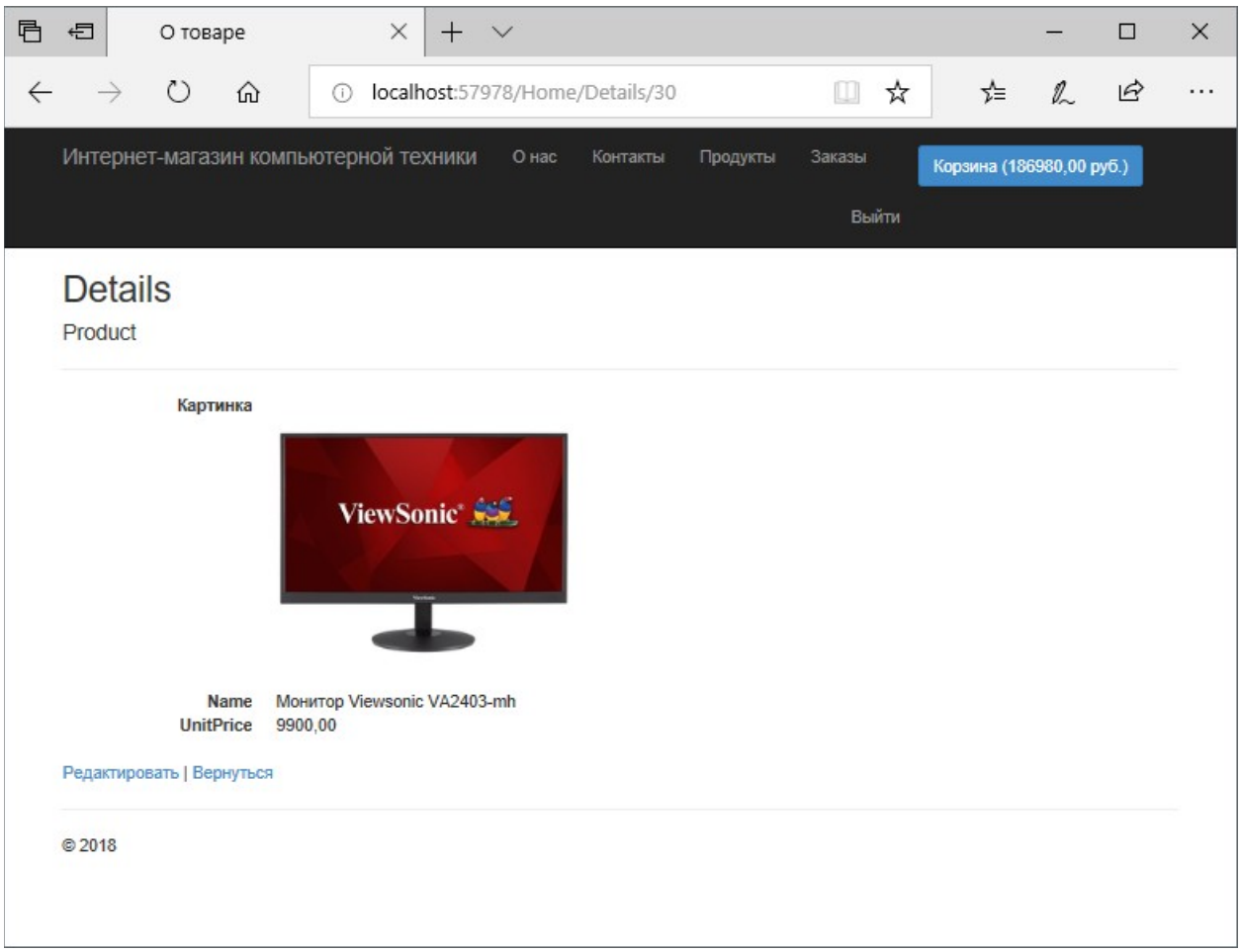

Рисунок 11 – Детализация информации по товару

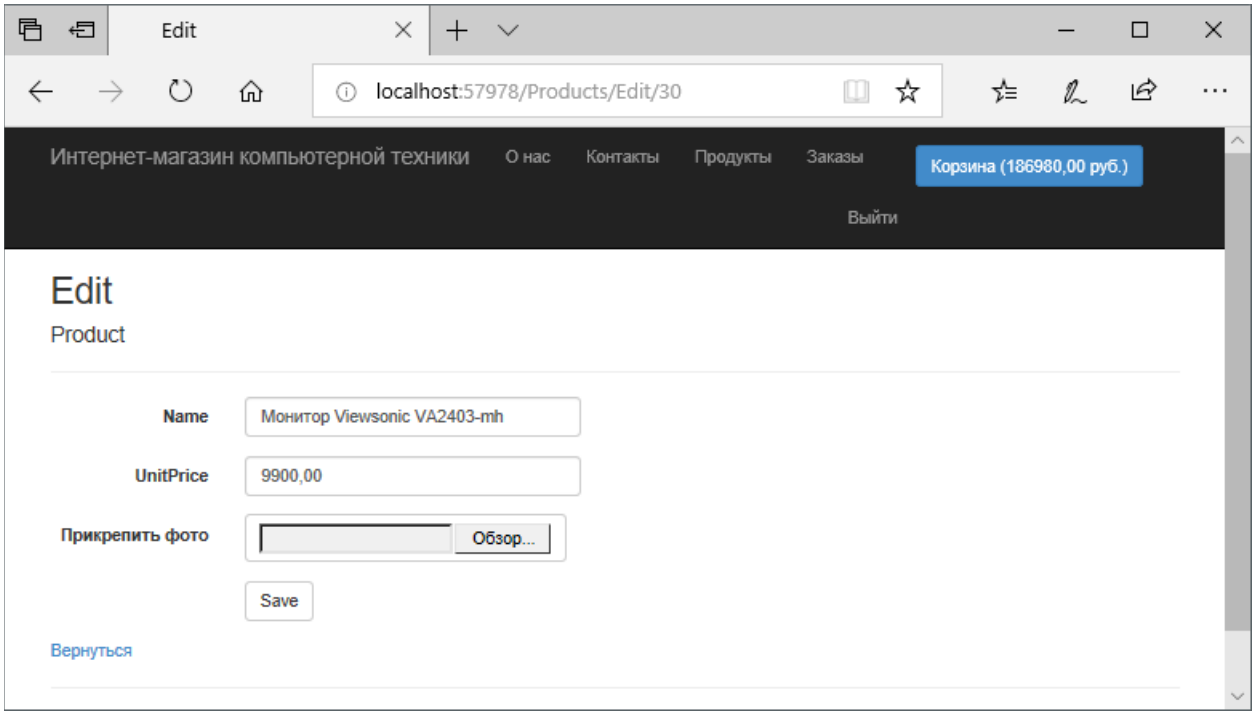

Рисунок 12 – Изменение характеристик, цены, фотографии товара

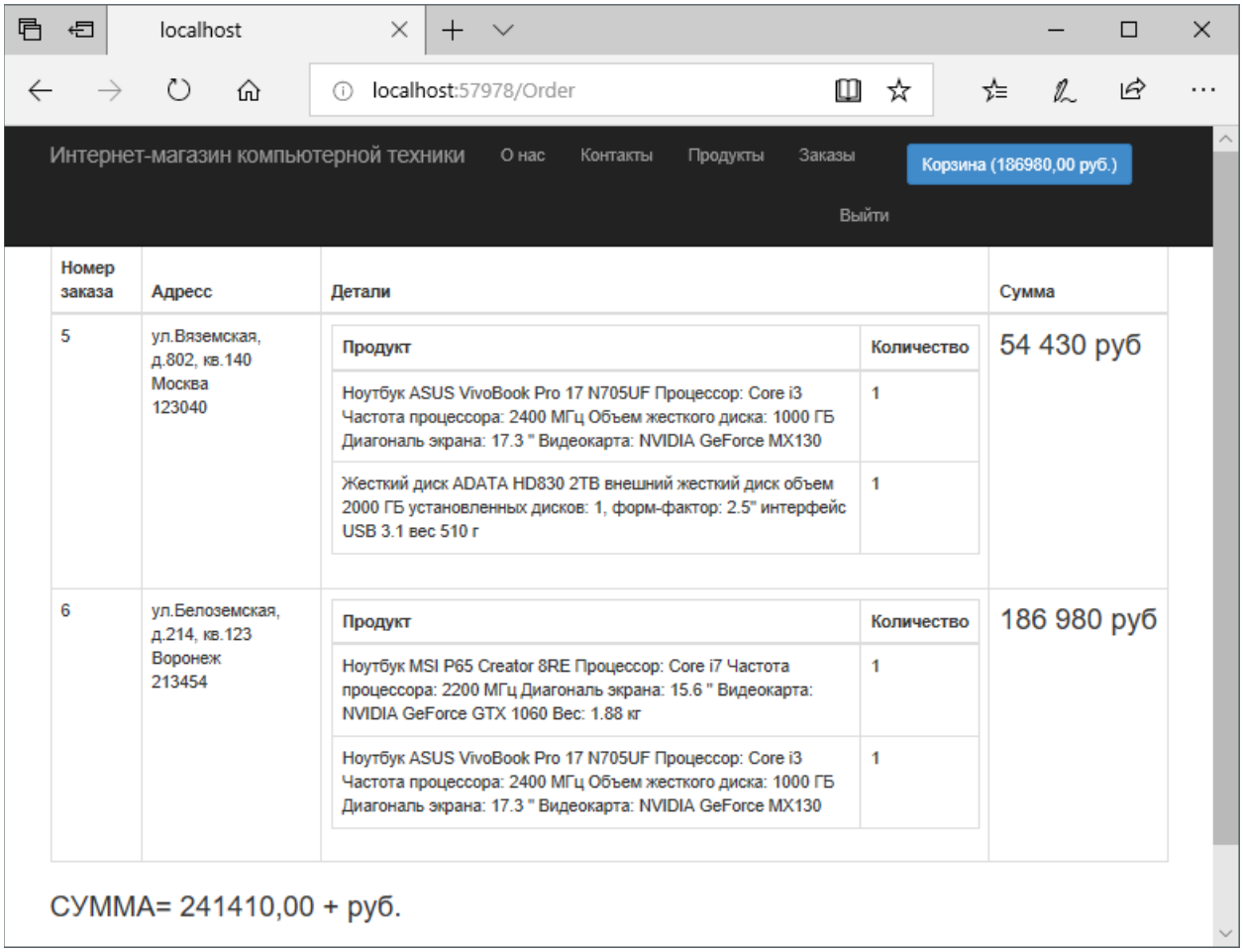

Рисунок 13 – Детализация всех оформленных заказов

4. Тестирование автоматизированной системы

Тестирование программного обеспечения — это процесс исследования разрабатываемого программного продукта, целями которого являются:

- Верификация разрабатываемого программного обеспечения на предмет соответствия требованиям;
- Выявление ситуаций, в которых поведение программы становится некорректным или не соответствующим спецификации.

На сегодняшний день тестирование является неотъемлемой частью процесса разработки программного обеспечения. Тестирование производится не только в течение всего цикла разработки, но и также в течение практически всего периода эксплуатации разработанного программного продукта. Такая тесная связь разработки и тестирования, при правильном следовании всем правилам рабочего процесса, рано или поздно становится гарантом качества разрабатываемого программного обеспечения.

Качество определяется как совокупная характеристика исследуемого программного обеспечения с учётом следующих составляющих:

- Надёжность;
- Сопровождаемость;
- Практичность;
- Эффективность;
- Мобильность;
- Функциональность.

Состав и содержание документации, сопутствующей процессу тестирования, определяется стандартом IEEE 829-1998.

В следующих пунктах данного раздела результаты тестирования разрабатываемого продукта на предмет соответствия выше приведенным характеристикам качества.

4.1 Тестирование на надежность

Тестирование на надежность – это такой вид тестирования программного обеспечения, целью которого служит проверка работоспособности приложения при длительном тестировании с ожидаемым уровнем нагрузки

К данному виду тестирования следует отнести следующие параметры:

- Зрелость – обратная величина к частоте отказов;
- Устойчивость к отказам – способность системы не реагировать на незначительные для системы проблемы.

После тестирования данного проекта на надежность, можно сказать следующее: Надежность с точки зрения параметра «Зрелость» при каждом запуске проекта растет, так как выявленные непредвиденные ситуации, приводившие систему к отказу, ликвидируются, и частота отказов стремится к нулю. Параметр «Устойчивость к отказам» в данном проекте имеет высокий показатель благодаря надежности используемых программных методов, однако в случае обрыва связи с сетью приложение не может выполнять свои функции.

4.2 Тестирование на сопровождаемость

Данный вид тестирования является единственным аспектом качества программного обеспечения, все параметры которого практически не поддаются тестированию, и для них применяются совершенно другие методы контроля качества, такие как: статический анализ кода, code review и т. д.

Этот вид тестирования подразумевает наличие следующих параметров:

- Анализируемость программного кода;
- Изменяемость – удобство внесения изменений в программный код;

- Риск возникновения неожиданных эффектов после того, как мы эти изменения внесли;
- Контролируемость – удобство тестирования программы

Рассмотрев аспект сопровождаемости данного проекта, можно сказать следующее:

Анализируемость программного кода хорошая. Основные архитектурные блоки были выделены в независимые сущности и реализованы в соответствующих классах. Важные особенности реализации имеют комментарии. Структура проекта небольшая и логичная. Исходя из вышеперечисленных особенностей можно смело заявить о том, что изменяемость в данном проекте легкая. Риск возникновения неожиданных эффектов не велик, но, тем не менее, имеет место быть, но за счет легкой анализируемости, так же легко может быть устранен. Удобство тестирования программы очень высокая, потому что данный проект разрабатывается с созданием юнит-тестов в проекте UnitTests, покрывающих наиболее сложные моменты кода.

4.3 Тестирование на практичность

Тестирование на практичность состоит из следующих параметров:

- Интуитивность программного интерфейса;
- Удобство обучения – наличие руководства пользователя;
- Управляемость программой – обязательное реагирование программы на действия пользователя;
- Привлекательность – чисто эстетический параметр

В результате тестирования на практичность было выявлено следующее: Данный продукт имеет интуитивно понятный интерфейс, руководство пользователя имеется, ознакомиться с которым можно в пункте 3.3. Большинство пользователей в состоянии освоить данную программу и без документации. Случаев игнорирования действий пользователя выявлено не было.

4.4 Тестирование на эффективность

Данный аспект качества характеризуется следующими параметрами:

- Время отклика;
- Использование дисковых ресурсов;
- Использование ресурсов процессора;
- Использование оперативной памяти;
- Использование сетевых ресурсов.

После тестирования проекта на эффективность были получены выше приведенные параметры: Время отклика очень маленькое и задержки не заметны для пользователя. Данная программный продукт, даже при использовании кэширования браузером, занимает на диске гораздо меньше места, чем большинство его аналогов. Так же использование ресурсов процессора и оперативной памяти данного проекта зависит в большей мере от браузера клиента, так как содержит мало сценариев, исполняемых на странице – а потому, приблизительно совпадает с аналогичным показателем продукта-аналога. Данный проект требует для работы возможность соединения с сетью (сервером локальной сети), поэтому использование сетевых ресурсов имеет место.

4.5 Тестирование на мобильность

Мобильность предполагает наличие следующих параметров:

- Возможность программы работать в различных окружениях;
- Простота установки программного обеспечения;
- Дружелюбие по отношению к другим работающим программам;
- Наличие у пользователя возможности перейти с другого аналогичного программного обеспечения на использование этой программы.

Данное программное средство было успешно запущено и протестировано на браузерах на следующих платформах: ПК с ОС Windows,

телефон с ОС Android. Работа программного обеспечения не вызывала проблем и конфликтов с другими программами.

4.6 Тестирование на функциональность

В данный аспект качества программного обеспечения включаются следующие параметры:

- Пригодность к использованию;
- Правильность выполнения функции, для которых разрабатывалась данная программа;
- Способность взаимодействия с другими программами, операционной системой или версиями данной программы;
- Безопасность – это обязанность программы делать то, что она обязана, обязанность не делать ничего другого.

Тестирование разработанного программного средства на функциональность по параметру пригодности к использованию и правильности выполнения функций полностью удовлетворяет требованиям. Способность взаимодействия с другими программами не заложена в функционал. Тестирование по параметру безопасность удовлетворяет требованию, программа выполняет строго заявленный функционал.

5. ЗАКЛЮЧЕНИЕ

В процессе работы над дипломным проектом:

- исследована предметная область;
- сформулированы цель и задачи проектирования;
- рассмотрены и проанализированы существующие разработки по изучаемой тематике;
- информационная система спроектирована с использованием таких CASE-средств как Allfusion Process Modeler и Rational Rose. В результате были построены функциональные диаграммы, диаграмма вариантов использования, диаграмма классов, диаграммы последовательности;
- разработан Web-сайт компьютерного магазина. Проектирование структуры информационных ресурсов выполнено в соответствии с техническим заданием.

Было проведено детальное исследование предметной области, подходящие для создания подобного рода сайтов. Наиболее подходящим вариантом была выбрана платформа ASP.NET в связке Фреймворком MVC.

В результате анализа определены основные функции и задачи, требующие решения. При разработке сайта были разработаны модули оформления заказа и редактирование товара администратором, этого вполне достаточно для полноценного функционирования интернет-магазина.

Проведен экономический анализ проекта и определены требования по эргономике рабочего места.

Проведенный анализ показал, что разработанный сайт компьютерного магазина характеризуется положительным экономическим эффектом и имеет положительный социальный эффект.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. HTML [Электронный ресурс]. - Режим доступа:

https://ru.wikipedia.org/wiki/HTML (дата обращения: 10.11.2018)

2. CSS [Электронный ресурс]. - Режим доступа: https://ru.wikipedia.org/wiki/CSS (дата обращения: 10.11.2018)
3. Основы создания Web сайтов на C# в Visual Studio [Электронный ресурс]. - Режим доступа: http://wladm.narod.ru/C_Sharp/webbegin.html (дата обращения: 10.11.2018)
4. Начало работы с ASP.NET MVC 5 [Электронный ресурс]. - Режим доступа: http://metanit.com/sharp/mvc5/1.2.php (дата обращения: 10.11.2018)
5. Введение в ASP.NET [Электронный ресурс]. - Режим доступа: http://brainoteka.com/couse/3051/vvedenie-v-asp-nett (дата обращения: 10.11.2018)
6. Основы ASP.NET [Электронный ресурс]. - Режим доступа: http://professorweb.ru/my/ASP_NET/base/level1/base_aspnet_index.php (дата обращения: 10.11.2018)
7. Рейсиг Д. 4. Инструменты отладки и тестирования // JavaScript. Профессиональные приёмы программирования = Pro JavaScript™ Techniques / Перевод Н. Вильчинский. — СПб.: Питер, 2008. — С. 76. — (Библиотека программиста). — 2500 экз.
8. Гусева Т.И., Башин Ю.Б., Проектирование баз данных в примерах и задачах, М., 2008 г.
9. Гвоздарева Т.В. Проектирование информационных систем /Т.В. Гвоздева, Б.А. Баллод. – Ростов-на-Дону: Феникс, 2009. – 512 с.
10. Виноградова С.Н. Организация и технология торговли/ Виноградова С.Н., Мн.: Высшая школа, 2009. - 460 с.
11. НАДЕЖНОСТЬ В ТЕХНИКЕ, ГОСТ 27.002—89. Основные понятия. Термины и определения. [электронный ресурс]. Режим доступа: https://ru.wikisource.org/wiki/ГОСТ_27.002—89 (дата обращения: 10.11.2018)

12. Гуров В.В. Интернет для бизнеса/ Гуров В.В., М., 2008. – 224 с.

13. Успенский И.В. Интернет как инструмент маркетинга/ Успенский И.В., - СПБ.: БХВ-Санкт-Петербург, 2009. – 256 с.

14. Холмогоров В. Интернет-маркетинг/ Холмогоров В., - Краткий курс. - СПБ.: Питер, 2008 – 272 с.

15. Сравнение CMS Интернет-магазинов. Режим доступа: http://jmworld.ru/ssttss/2012-08-02-13-25-5/2193--cms-.html (дата обращения: 10.10.2018)

16. В. Дубейковский. Эффективное моделирование с CA ERwin Process Modeler (BPwin; AllFusion Process Modeler)/ В.Дубейковский, - М:Диалог МИФИ, 2009 г., - 384 с.

Приложение А.

Листинг файла «cart.cs»

```
using System.Collections.Generic;
using System.Linq;

namespace Domain.Entities
{
        public class Cart
        {
                private readonly List<OrderItem> _lines = new List<OrderItem>();

                public IEnumerable<OrderItem> Lines => _lines;

            public void AddLine(Product product, int quantity)
                {
                        var existingLine = _lines
                                .FirstOrDefault(line => line.Product.Id == product.Id);

                        if (existingLine == null)
                        {
                                _lines.Add(new OrderItem
                                {
                                        Product = product,
                                        Quantity = quantity
                                });
                        }
                        else
                        {
                                existingLine.Quantity += quantity;
                        }
                }


      public void RemoveLine(Product product, int quantity)
      {
        var existingLine = _lines.FirstOrDefault(line => line.Product.Id == product.Id);
        if (existingLine != null)
        {
```

```
            existingLine.Quantity -= quantity;
            existingLine.Quantity = (existingLine.Quantity > 0) ? existingLine.Quantity :
0;
        }
    }


    public void ClearLine(Product product)
    {
        var existingLine = _lines.FirstOrDefault(line => line.Product.Id == product.Id);
        if (existingLine != null)
        {
            _lines.Remove(existingLine);
        }
    }


    public decimal GetTotalAmount()
            {
                return _lines.Sum(line => line.Product.UnitPrice * line.Quantity);
            }
      }
}
```

Листинг файла «OrderItem.cs»

```
using System.ComponentModel.DataAnnotations;

namespace Domain.Entities
{
        public class OrderItem
        {
      [Key]
      public int Id { get; set; }

      public int ProductId { get; set; }
      public virtual Product Product { get; set; }


      public int OrderId { get; set; }
      public virtual Order Order { get; set; }

      [Required]
      public int Quantity { get; set; }
   }
}
```

Листинг файла «Products.cs»

```
using System.ComponentModel.DataAnnotations;

namespace Domain.Entities
{
        public class Product
        {
                [Key]
                public int Id { get; set; }

      [Required(ErrorMessage = "Введите название продукта")]
                public string Name { get; set; }

      public decimal UnitPrice { get; set; }

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

Листинг файла «CartModel.cs»

```
using System;
using Domain.Entities;
using Domain.Interfaces;


namespace Domain.Models
{
   public class CartModel
        {
      readonly ICartService _cartService;
      readonly IUnitOfWork _unitOfWork;

                public CartModel(ICartService cartService, IUnitOfWork unitOfWork)
                {
                        _cartService = cartService;
                        _unitOfWork = unitOfWork;
                }

                public  void Add(int productId, int quantity)
                {
                        var cart = _cartService.Get();
                        var product =  _unitOfWork.Products.Get(productId);

                        cart.AddLine(product, quantity);
                        _cartService.Update(cart);
                }

                public void Remove(int productId, int quantity)
                {
          var cart = _cartService.Get();
          var product = _unitOfWork.Products.Get(productId);

          cart.RemoveLine(product, quantity);
          _cartService.Update(cart);
      }
```

```
        public void Clear(int productId)
        {
            var cart = _cartService.Get();
            var product = _unitOfWork.Products.Get(productId);

            cart.ClearLine(product);
            _cartService.Update(cart);
        }


            public Order CreateOrder()
            {
                        // TODO
                        throw new NotImplementedException();
            }
        }
}
```

Листинг файла «CartController.cs»

```
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using Domain.Entities;
using Domain.Interfaces;
using Domain.Models;

namespace WebUI.Controllers
{
    public class CartController : Controller
    {
                private readonly ICartService _cartService;
                private readonly IUnitOfWork _unitOfWork;

                public CartController(ICartService cartService, IUnitOfWork
unitOfWork)
                {
                        _cartService = cartService;
                        _unitOfWork = unitOfWork;
                }

        // GET: Cart
        public ActionResult Index()
        {
            return View(_cartService.Get());
        }


        [HttpPost]
        public ActionResult Add(int id)
        {
            var sale = new CartModel(_cartService, _unitOfWork);
            sale.Add(id, 1);

            return Json(new
            {
                Total = _cartService.Get().GetTotalAmount()
```

```
            });
        }

        [HttpPost]
        public ActionResult QuantityAdd(int id)
        {
            var sale = new CartModel(_cartService, _unitOfWork);
            sale.Add(id, 1);

            OrderItem orderItemProduct = _cartService.Get().Lines.FirstOrDefault(line
=> line.Product.Id == id);
            if (orderItemProduct != null)
            {
                return Json(new
                {
                    Quantity = orderItemProduct.Quantity,
                    Total = _cartService.Get().GetTotalAmount()
                });
            }
            return null;
        }


        [HttpPost]
        public ActionResult QuantityRemove(int id)
        {
            var sale = new CartModel(_cartService, _unitOfWork);
            sale.Remove(id, 1);

            OrderItem orderItemProduct = _cartService.Get().Lines.FirstOrDefault(line
=> line.Product.Id == id);
            if (orderItemProduct != null)
            {
                return Json(new
                {
                    Quantity = orderItemProduct.Quantity,
                    Total = _cartService.Get().GetTotalAmount(),
                    IdProd= id
```

```
            });
        }
        return null;
    }


    [HttpPost]
    public ActionResult ClearLine(int id)
    {
        var sale = new CartModel(_cartService, _unitOfWork);
        sale.Clear(id);

        var cart = _cartService.Get();
        return PartialView("_CartTablePartial", cart);
    }

    [HttpPost]
    public JsonResult CreateOrder(Address address)
    {
        var cart = _cartService.Get();
        var order= new Order {DeliveryAddress = address, Items =
(ICollection<OrderItem>) cart.Lines };
        _unitOfWork.Orders.Insert(order);
        _unitOfWork.SaveAsync();
        return Json(new
        {
            Message= "ЗАКАЗ по адерссу " + address.AdressLine + " ОФОРМЛЕН"
        });
    }




    public PartialViewResult Summary()
    {
        return PartialView(_cartService.Get());
    }
```

```
    }
}
```

Листинг файла «HomeController.cs»

```
using System;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Security.Claims;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
using Domain.Interfaces;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin.Security;
using PagedList;
using WebUI.Security;

namespace WebUI.Controllers
{
        public class HomeController : Controller
        {
      private readonly IUnitOfWork _unitOfWork;

            public HomeController(IUnitOfWork unitOfWork)
            {
               _unitOfWork = unitOfWork;
            }


            // GET: Products
      public async Task<ActionResult> Index(int? page)
      {
         var qProduct = await _unitOfWork.Products.Get().OrderBy(p =>
p.Id).ToListAsync();
         var pageNumber = page ?? 1;
         var onePageOfProducts = qProduct.ToPagedList(pageNumber, 2);

         return View(onePageOfProducts);
      }
```

```
// GET: Products/Details/5
public async Task<ActionResult> Details(int? id)
{
   if (id == null)
   {
      return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
   }

   var product = await _unitOfWork.Products.GetAsync(id.Value);
   if (product == null)
   {
      return HttpNotFound();
   }
   return View(product);
}



public ActionResult About()
        {
                ViewBag.Message = "Your application description page.";

                return View();
        }

        public ActionResult Contact()
        {
                ViewBag.Message = "Your contact page.";

                return View();
        }


[HttpGet]
    public ActionResult Login()
    {
```

```
        //Создание и добавление нового пользователя (можно закоментить,т.к. Админ уже добавлен)----------------------------------------------------------
        const string adminLogin = "admin";
        const string admineRole = AdminAttribute.AdminRoleName;
        const string password = "1234567";

        var userManager = HttpContext.GetOwinContext().GetUserManager<ShopUserManager>();
        var roleManager = HttpContext.GetOwinContext().GetUserManager<RoleManager<ShopRole>>();

        if (userManager.FindByName(adminLogin) == null)
        {
            var user = new ShopUser { UserName = adminLogin };
            var result = userManager.Create(user, password);
            if (result.Succeeded)
            {
                if (!roleManager.RoleExists(admineRole))
                {
                    roleManager.Create(new ShopRole(admineRole));
                }

                userManager.AddToRole(userManager.FindByName(adminLogin).Id, admineRole);
            }
        }
        //-----------------------------------------------------------------------------------------------
        return View();
          }


    //
    // POST: /Account/Login
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
    {
```

```
            if (ModelState.IsValid)
            {
                var userManager =
HttpContext.GetOwinContext().GetUserManager<ShopUserManager>();
                var authenticationManager =
HttpContext.GetOwinContext().Authentication;

                var user = await userManager.FindAsync(model.Login, model.Password);
                if (user == null)
                {
                    ModelState.AddModelError("", "Неверный логин или пароль.");
                }
                else
                {
                    var claim = await userManager.CreateIdentityAsync(user,
DefaultAuthenticationTypes.ApplicationCookie);
                    authenticationManager.SignOut();
                    authenticationManager.SignIn(new AuthenticationProperties
                    {
                        IsPersistent = true
                    },
                    claim);

                    if (string.IsNullOrEmpty(returnUrl))
                        return RedirectToAction("Index", "Home");

                    return Redirect(returnUrl);
                }
            }
            ViewBag.returnUrl = returnUrl;
            return View(model);
        }


        public ActionResult Logout()
        {
            var authenticationManager = HttpContext.GetOwinContext().Authentication;
```

```
            authenticationManager.SignOut();

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

Листинг файла «OrderController.cs»

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Domain.Entities;
using Domain.Interfaces;
using WebGrease.Css.Extensions;
using WebUI.ViewModel;


namespace WebUI.Controllers
{
    public class OrderController : Controller
    {
        private readonly IUnitOfWork _unitOfWork;

        public OrderController(IUnitOfWork unitOfWork)
        {
            _unitOfWork = unitOfWork;
        }



        // GET: Order
        public ActionResult Index()
        {
            var orders = _unitOfWork.Orders.Get().Include(item => item.Items);

            var ordersVm = new List<OrderViewModel>();
            foreach (var order in orders)
            {
                var item = new OrderViewModel {Number = order.Id, DeliveryAddress =
order.DeliveryAddress};
                foreach (OrderItem orderItem in order.Items)
```

```
            {
                item.OrderItems.Add(new OrderViewModel.Line {Product =
orderItem.Product, Quantity = orderItem.Quantity});
            }

            item.TotalPrice = item.OrderItems.Sum(i => (i.Quantity *
i.Product.UnitPrice));
            ordersVm.Add(item);
        }

        ViewBag.SumPriceOrder = ordersVm.Sum(i => i.TotalPrice);

        return View(ordersVm);
    }
  }
}
```

Листинг файла «ProductsController.cs»

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using System.Net;
using System.Web;
using System.Web.Mvc;
using Domain;
using Domain.Entities;
using Domain.Interfaces;
using PagedList;
using WebUI.Models;
using WebUI.Security;

namespace WebUI.Controllers
{
    [Admin]
    public class ProductsController : Controller
    {
        private readonly IUnitOfWork _unitOfWork;

        public ProductsController(IUnitOfWork unitOfWork)
        {
            _unitOfWork = unitOfWork;
        }

        // GET: Products
        public async Task<ActionResult> Index(int? page)
        {
            var qProduct = await
_unitOfWork.Products.Get().OrderBy(p=>p.Id).ToListAsync();
            var pageNumber = page ?? 1;
            var onePageOfProducts = qProduct.ToPagedList(pageNumber, 2);
```

```
            return View(onePageOfProducts);
        }

        // GET: Products/Details/5
        public async Task<ActionResult> Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }

            var product = await _unitOfWork.Products.GetAsync(id.Value);
            if (product == null)
            {
                return HttpNotFound();
            }
            return View(product);
        }

        // GET: Products/Create
        public ActionResult Create()
        {
            return View();
        }

        // POST: Products/Create
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> Create([Bind(Include = "Id,Name,UnitPrice")]
Product product, HttpPostedFileBase[] fileUpload)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    //Добавить файл на диск
                    var path = AppDomain.CurrentDomain.BaseDirectory +
"UploadedFiles/";
```

```
            if (!Directory.Exists(path))
            {
                Directory.CreateDirectory(path);
            }
            var file = fileUpload.FirstOrDefault();
            var filename = Path.GetFileName(file?.FileName);
            var filePath = filename != null ? Path.Combine(path, filename) : null;
            if (filePath != null)
            {
                file.SaveAs(filePath);
                product.PictureRef = "/UploadedFiles/" + filename;
            }


            //Добавить в БД
            _unitOfWork.Products.Insert(product);
            await _unitOfWork.SaveAsync();

        }
        catch (IOException)
        {
            return View(product);
        }
        catch (DataException)
        {
            //Удаление файла с диска
            if (!string.IsNullOrEmpty(product.PictureRef))
            {
                var filePath = AppDomain.CurrentDomain.BaseDirectory +
product.PictureRef;
                await DeleteFileAsync(filePath);
            }
        }

        return RedirectToAction("Index");
    }

    return View(product);
```

```
        }



        // GET: Products/Edit/5
        public async Task<ActionResult> Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            var product = await _unitOfWork.Products.GetAsync(id.Value);
            if (product == null)
            {
                return HttpNotFound();
            }
            return View(product);
        }

        // POST: Products/Edit/5
        // To protect from overposting attacks, please enable the specific properties
you want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> Edit([Bind(Include = "Id,Name,UnitPrice")]
Product product, HttpPostedFileBase[] fileUpload)
        {
            if (ModelState.IsValid)
            {
                Product oldProduct = null;//_unitOfWork.Products.Get(product.Id);

                try
                {
                    //Добавить файл на диск
                    var path = AppDomain.CurrentDomain.BaseDirectory +
"UploadedFiles/";
                    if (!Directory.Exists(path))
```

```
                {
                    Directory.CreateDirectory(path);
                }

                var file = fileUpload.FirstOrDefault();
                var filename = Path.GetFileName(file?.FileName);
                var filePath = filename != null ? Path.Combine(path, filename) : null;
                if (filePath != null)
                {
                    file.SaveAs(filePath);
                    product.PictureRef = "/UploadedFiles/" + filename;
                }

                //Обновить в БД
                _unitOfWork.Products.Update(product);
                await _unitOfWork.SaveAsync();

            }
            catch (IOException)
            {
                return View();
            }
            catch (DataException)
            {
                //Удаление файла с диска
                if (!string.IsNullOrEmpty(product.PictureRef))
                {
                    var filePath = AppDomain.CurrentDomain.BaseDirectory +
product.PictureRef;
                    await DeleteFileAsync(filePath);
                }
            }

            //Удаление старого  файла с диска
            if (!string.IsNullOrEmpty(oldProduct?.PictureRef))
            {
                var filePath = AppDomain.CurrentDomain.BaseDirectory +
oldProduct.PictureRef;
```

```
            await DeleteFileAsync(filePath);
        }

        return RedirectToAction("Index");
    }
    return View(product);
}

// GET: Products/Delete/5
public async Task<ActionResult> Delete(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Product product = await _unitOfWork.Products.GetAsync(id.Value);
    if (product == null)
    {
        return HttpNotFound();
    }
    return View(product);
}

// POST: Products/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<ActionResult> DeleteConfirmed(int id)
{
    Product product = await _unitOfWork.Products.GetAsync(id);
    try
    {
        //Удалние из БД
        _unitOfWork.Products.Remove(product);
        await _unitOfWork.SaveAsync();

        //Удаление файла с диска
        if (!string.IsNullOrEmpty(product.PictureRef))
        {
```

```
                var filePath = AppDomain.CurrentDomain.BaseDirectory +
product.PictureRef;// "D:\\Git\\Shop\\src\\WebUI\\UploadedFiles/saw.png"
                await DeleteFileAsync(filePath);
            }
        }
        catch (DataException)
        {
            return View();
        }
        catch (IOException)
        {
            //Добавить в БД
            _unitOfWork.Products.Insert(product);
            await _unitOfWork.SaveAsync();
        }

        return RedirectToAction("Index");
    }



    #region Methods

    public async Task DeleteFileAsync(string filePath)
    {
        if (System.IO.File.Exists(filePath))
        {
            await Task.Factory.StartNew(() => { System.IO.File.Delete(filePath); });
        }
    }

    #endregion



    protected override void Dispose(bool disposing)
```

```
        {
            if (disposing)
            {
                _unitOfWork.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

Листинг файла «_CartTablePartial.cshtml»

```
@using System.Linq
@model Domain.Entities.Cart


<script type="text/javascript">

    function quantityAdd(productId) {
        $.ajax({
            url: "/Cart/QuantityAdd/" + productId,
            type: "POST"
        }).done(function (data) {

            $('#quantity').text(data.Quantity);
            $('#shopCartBtn').text('В корзине (' + data.Total + ') Руб.');
        });
    };

    function quantityRemove(productId) {
        $.ajax({
            url: "/Cart/QuantityRemove/" + productId,
            type: "POST"
        }).done(function (data) {
            $('#quantity').text(data.Quantity);
            $('#shopCartBtn').text('В корзине (' + data.Total + ') Руб.');
        });
    };

    function clearLine(productId) {
        $.ajax({
            url: "/Cart/ClearOrderItem/" + productId,
            type: "POST",
            success: function (respone) {
                alert(respone);
                //$("#tableCart".text(respone));
            }
        });
    };
```

```
</script>



<tbody>
    @foreach (var line in @Model.Lines)
    {
        <tr>
            <td >
                @Html.DisplayFor(item => line.Product.Name)
            </td>
            <td>

                @Html.DisplayFor(item => line.Product.UnitPrice)
            </td>
            <td id= "quantity">  @* id="quantity" + @line.Product.Id   *@
                @Html.DisplayFor(item => line.Quantity)
            </td>

            <td>
                <div class="item-priceblock-buybutton">
                    <input type="button" class="btn btn-success" value="+"
onclick="quantityAdd(@line.Product.Id)" />
                </div>
                <div class="item-priceblock-buybutton">
                    <input type="button" class="btn btn-success" value="-"
onclick="quantityRemove(@line.Product.Id)" />
                </div>
                <div class="item-priceblock-buybutton">
                    <input type="button" class="btn btn-success" value="x"
onclick="clearLine(@line.Product.Id)" />
                </div>
            </td>
        </tr>
    }
</tbody>
```

Листинг файла «Index.cshtml»

```
@using System.Linq
@model Domain.Entities.Cart

@{
    ViewBag.Title = "View";
}


<script type="text/javascript">
    function OnSuccessCreate(data) {
        var results = $('#OrderCompliteMessage'); // получаем нужный элемент
        results.empty(); //очищаем элемент
        results.append(data.Message); // добавляем данные в список
    }
</script>



@if (Model == null || !Model.Lines.Any())
{
    <div class="alert alert-info">Корзина пуста</div>
}
else
{
    <table class="table table-bordered">
        <thead>
            <tr>
                <th>
                    Продукт
                </th>
                <th>
                    Цена
                </th>
                <th>
                    Количество
                </th>
                <th>
```

```
                Действие
            </th>
        </tr>
    <thead>

    <tbody>
        @Html.Partial("_CartTablePartial", Model)
    </tbody>
</table>
}

<br>
@*//TODO: форму с заполненем адреса вставлять через @Ajax.ActionLink()*@
<div id="OrderCompliteMessage">
    @using (Ajax.BeginForm("CreateOrder", new AjaxOptions { Confirm =
"Сформировать заказ ?", OnSuccess = "OnSuccessCreate" }))
    {
        <ul>
            <li>
                @Html.Label("Адрес дома")
                @Html.Editor("AdressLine")
            </li>

            <li>
                @Html.Label("Город (РФ)")
                @Html.Editor("City")
            </li>

            <li>
                @Html.Label("Почтовый индекс")
                @Html.Editor("ZipCode")
            </li>
        </ul>

        <input type="submit" value="Заказать" class="btn btn-default" />
    }
</div>
```

```
<br>

<div>
    @Html.ActionLink("Вернуться", "Index")
</div>
```

Листинг файла «Details.cshtml»

```
@model Domain.Entities.Product

@{
    ViewBag.Title = "О товаре";
}

<h2>Details</h2>

<div>
    <h4>Product</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.Label("Картинка")
        </dt>

        <dd>
            <img src="@Model.PictureRef" alt="Изображение" title="Картинка" />
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Name)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Name)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.UnitPrice)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.UnitPrice)
        </dd>

    </dl>
```

```
</div>
<p>
    @Html.ActionLink("Редактировать", "Edit", new { id = Model.Id }) |
    @Html.ActionLink("Вернуться", "Index")
</p>
```

Листинг файла «Login.cshtml»

```
@model WebUI.Security.LoginViewModel

@{
    ViewBag.Title = "Вход на сайт";
}

<h2>Вход на сайт</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div>
        @Html.ValidationSummary()
        <input type="hidden" name="returnUrl" value="@ViewBag.returnUrl" />
        <p>
            <label>Login</label><br />
            @Html.EditorFor(x => x.Login)
        </p>
        <p>
            <label>Пароль</label><br />
            @Html.PasswordFor(x => x.Password)
        </p>
        <p><button type="submit">Войти</button></p>
    </div>
}
```

Листинг файла «Product/Create.cshtml»

```
@model Domain.Entities.Product

@{
    ViewBag.Title = "Create";
}

<h2>Create</h2>


@using (Html.BeginForm("Create", "Products", FormMethod.Post, new { enctype =
"multipart/form-data" }))
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Product</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.Name, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Name, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Name, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.UnitPrice, htmlAttributes: new {@class =
"control-label col-md-2"})
            <div class="col-md-10">
                @Html.EditorFor(model => model.UnitPrice, new {htmlAttributes = new
{@class = "form-control"}})
                @Html.ValidationMessageFor(model => model.UnitPrice, "", new {@class =
"text-danger"})
```

```
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.PictureRef, "Прикрепить фото",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.TextBox("fileUpload", "fileUpload", htmlAttributes: new { type =
"file", multiple = "multiple", @class = "btn btn-default" })
                @*@Html.TextBoxFor(m => m.PictureRef, htmlAttributes: new { type =
"file", multiple = "multiple", @class = "btn btn-default" })*@
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Листинг файла «Product/Delete.cshtml»

```
@model Domain.Entities.Product

@{
    ViewBag.Title = "Удаление";
}

<h2>Delete</h2>

<h3>Вы уверены, что хотите удалить?</h3>
<div>
    <h4>Товар</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Name)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Name)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.UnitPrice)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.UnitPrice)
        </dd>

    </dl>

    @using (Html.BeginForm()) {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-default" /> |
            @Html.ActionLink("Вернуться", "Index")
```

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

Листинг файла «Product/Details.cshtml»

```
@model Domain.Entities.Product

@{
    ViewBag.Title = "Details";
}

<h2>Details</h2>

<div>
    <h4>Product</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.Label("Картинка")
        </dt>

        <dd>
         <img src="@Model.PictureRef" alt="Изображение" title="Картинка" />
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Name)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Name)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.UnitPrice)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.UnitPrice)
        </dd>

    </dl>
```

```
</div>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.Id }) |
    @Html.ActionLink("Вернуться", "Index")
</p>
```

Листинг файла «Product/Edit.cshtml»

```
@model Domain.Entities.Product

@{
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>


@using (Html.BeginForm(Html.BeginForm("Edit", "Products", FormMethod.Post,
new { enctype = "multipart/form-data" })))
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Product</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.Id)

        <div class="form-group">
            @Html.LabelFor(model => model.Name, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Name, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Name, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.UnitPrice, htmlAttributes: new {@class =
"control-label col-md-2"})
            <div class="col-md-10">
                @Html.EditorFor(model => model.UnitPrice, new {htmlAttributes = new
{@class = "form-control"}})
```

```
            @Html.ValidationMessageFor(model => model.UnitPrice, "", new {@class =
"text-danger"})
        </div>
    </div>


    <div class="form-group">
        @Html.LabelFor(model => model.PictureRef, "Прикрепить фото",
htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.TextBox("fileUpload", "fileUpload", htmlAttributes: new { type =
"file", multiple = "multiple", @class = "btn btn-default" })
            @*@Html.TextBoxFor(m=>m.PictureRef, htmlAttributes: new { type =
"file", multiple = "multiple", @class = "btn btn-default" })*@
        </div>
    </div>

    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Save" class="btn btn-default"/>
        </div>
    </div>
  </div>
}

<div>
  @Html.ActionLink("Вернуться", "Index")
</div>

@section Scripts {
  @Scripts.Render("~/bundles/jqueryval")
}
```

Листинг файла «Product/Index.cshtml»

```
@using PagedList.Mvc;
@using PagedList;

@model IPagedList<Domain.Entities.Product>

<link href="~/Content/PagedList.css" rel="stylesheet" type="text/css" />


@*<script type="text/javascript">
   function addToCart(productId)  {

      $.ajax({
         url: "/Cart/Add/" + productId,
         type: "POST"
      }).done(function (data) {
         //alert(data.Total);
         $('#shopCartBtn').text('В корзину (' + data.Total + ') Руб.');
      });
   };
</script>*@


<p>
   @Html.ActionLink("Добавить новый товар", "Create")
</p>



<ul>

   @foreach (var item in Model)
   {
      <li class="item">
         <div class="item-imageblock">
            <img class="item-imageblock-img" src="@item.PictureRef"
alt="Изображение" title="Картинка" />
         </div>
```

```
            <div class="item-descblock">
                @Html.ActionLink(item.Name, "Details", new { id = item.Id }, new { @class
= "item-descblock-title" })

                @Html.ActionLink("Edit", "Edit", new { id = item.Id }) |
                @Html.ActionLink("Details", "Details", new { id = item.Id }) |
                @Html.ActionLink("Delete", "Delete", new { id = item.Id })


                <div>

                </div>
            </div>

            <div class="item-priceblock">

                <div class="item-priceblock-price">
                    @Html.DisplayFor(modelItem => item.UnitPrice, "money")
                </div>

                @*<div class="item-priceblock-buybutton">
                    <input type="button" class="btn btn-success" value="В корзину"
onclick="addToCart(@item.Id)" />
                </div>*@

            </div>

        </li>
    }
</ul>
<br />
Страница @(Model.PageCount < Model.PageNumber ? 0 : Model.PageNumber) из
@Model.PageCount
@Html.PagedListPager(Model, page => Url.Action("Index", new { page }))
```