|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  федеральное государственное АВТОНОМНОЕ образовательное учреждение высшего образования  «Национальный исследовательский ядерный университет «МИФИ» |
| **Димитровградский инженерно–технологический институт –**  филиал федерального государственного автономного образовательного учреждения высшего  образования «Национальный исследовательский ядерный университет «МИФИ»  **(ДИТИ НИЯУ МИФИ)** |

Факультет информационно-технологический

Кафедра информационных технологий

Специальность 09.05.01 [Применение и эксплуатация автоматизированных систем специального назначения](http://priem.diti-mephi.ru/system/resources/BAhbBlsHOgZmSSImMjAxMi8wNS8yNy8yMl8zNV80OV8xNDNfXy5fLl8ucGRmBjoGRVQ/%21%D0%9F%D1%80%D0%B8%D0%BC%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%B8%20%D1%8D%D0%BA%D1%81%D0%BF%D0%BB%D1%83%D0%B0%D1%82.%20%D0%B0%D0%B2%D1%82%D0%BE%D0%BC.%20%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%20%D0%A1%D0%9D.pdf)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ**

«Разработка интернет-магазина зимних видов спорта»

Студент

Группы АС-51 Е.С. Паньков

(подпись)

Руководитель

ст. преподаватель Н.С. Казынбаев

(подпись)

Заведующий кафедрой

доцент, к.п.н. О.А. Ракова

(подпись)

Димитровград 2018

|  |
| --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  федеральное государственное АВТОНОМНОЕ образовательное учреждение высшего образования  «Национальный исследовательский ядерный университет «МИФИ» |
| **Димитровградский инженерно–технологический институт –**  филиал федерального государственного автономного образовательного учреждения высшего  образования «Национальный исследовательский ядерный университет «МИФИ»  **(ДИТИ НИЯУ МИФИ)** |

**ФАКУЛЬТЕТ** информационно-технологический

**КАФЕДРА** информационных технологий

**СПЕЦИАЛЬНОСТЬ** 09.05.01 **ГРУППА** АС-51

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

1.Фамилия, имя, отчество Паньков Егор Сергеевич

2.Тема работы «Разработка интернет-магазина зимних видов спорта»

3. Срок сдачи студентом готовой работы

4. Руководитель курсовой работы Казынбаев Никита Сергеевич

Дата выдачи задания «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_\_ г.

Руководитель курсовой работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата и подпись студента)

1. Исходные данные к работе:

требования к уровню разработки программного продукта:

\_\_\_*а) возможность взаимодействия с консолью;*

\_\_\_*б) реализация концепций объектно-ориентированного программирования;*

\_\_\_*в) применение платформы .Net Framework;*

1. Содержание работы:
   1. *теоретическая часть:*
      * исследование предметной области;
   2. *экспериментальная часть:*
      * подробное описание разрабатываемого программного продукта;
      * тестирование программного продукта.
2. Основная литература по теме работы:
3. *Введение в .Net-программирование [Электронный ресурс]. Режим доступа: https://vk.cc/85j68J (дата обращения: 12.04.2018)*
4. *Бадд, Тимоти. Объектно-ориентированное программирование в действии [Текст] / Тимоти Бадд – СПб: «Питер», 1997 – 296 c.*
5. *Фримен, Адам. LINQ: язык интегрированных запросов в C# 2010 для профессионалов [Текст] : Пер. с англ. / Адам Фримен, Джозеф C. Раттц – М. : ООО “И.Д. Вильямс”, 2011. – 656 с.*

4. Отчетный материал работы:

1. *пояснительная записка;*

РЕФЕРАТ

ПАНЬКОВ Е.С. «РАЗРАБОТКА ИНТЕРНЕТ-МАГАЗИНА ЗИМНИХ ВИДОВ СПОРТА»: Курсовой проект / ДИТИ НИЯУ МИФИ № КП.09.05.01.049 – Димитровград, 2018. – стр. 31, рис. 14, библ. назв. 12.

Ключевые слова: AsP.Net, MVC, VISUAL studio, c#, JAVAscript, ajax, jquery, html, WEB-приложение, linq, База данных, программирование,

Объект исследования – процесс разработки веб-приложения на платформе ASP.NET MVC.

Цель работы – разработать веб-приложение на платформе ASP.NET MVC по заданному условию в соответствии с вариантом.

Выводы:

Во время разработки приложения были пройдены и изучены все этапы проектирования разработки веб-приложения на платформе ASP.NET MVC «Интернет-магазин зимних видов спорта» от разработки алгоритма и структуры программы до тестирования конечного продукта. Результатом данной работы стал работоспособный программный продукт, готовый к дальнейшему использованию.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc533368439)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 8](#_Toc533368440)

[1.1 Технологии веб-разработки 9](#_Toc533368441)

[1.1.1 HTML 9](#_Toc533368442)

[1.1.2 JavaScript 11](#_Toc533368443)

[1.1.3 Фреймворк jQuery 11](#_Toc533368444)

[1.1.4 Технология AJAX 13](#_Toc533368445)

[1.1.5 Технология ASP.NET 14](#_Toc533368446)

[1.1.6 Фреймворк ASP.NET MVC 15](#_Toc533368447)

[1.2 Модульное тестирование 17](#_Toc533368448)

[1.3 Среда разработки Microsoft Visual Studio 19](#_Toc533368449)

[Вывод к разделу 1 20](#_Toc533368450)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 21](#_Toc533368451)

[2.1 Разработка приложения 21](#_Toc533368452)

[2.1.1. Моделирование предметной области 21](#_Toc533368453)

[2.1.2. Реализация бизнес-логики приложения 23](#_Toc533368454)

[2.1.3 Разработка веб-приложения 26](#_Toc533368455)

[2.2 Тестирование продукта 31](#_Toc533368456)

[ЗАКЛЮЧЕНИЕ 33](#_Toc533368457)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 34](#_Toc533368458)

ВВЕДЕНИЕ

На сегодняшний день всё прочнее и прочнее информационные технологии занимают практически все сферы нашей деятельности. Такая тенденция обусловлена широким спектром задач, которые высокие технологии позволяют решить: от эффективного использования информационных ресурсов общества до оптимизации и автоматизации процессов, которые ранее требовали человеческий ресурс. В частности, возможности современных информационных систем позволяют исключить человеческий фактор в процессе покупательской деятельности обычных людей, т.е. появляется возможность совершать покупки без контакта с продавцом, не выходя из дома. Учитывая повсеместный доступ к сети Интернет и бесконтактной оплате, развитие подобных технологий имеет высокий интерес в обществе, так как повышает комфорт бытовой деятельности. Таким образом, жители, проживающие далеко от мегаполисов, имеют возможность приобретать товары, отсутствующие в их городе, в интернет-магазинах с большим ассортиментом. С учетом выше сказанного, разработка веб-приложения для интернет-магазина является актуальной задачей на сегодняшний день.

В данной работе представлена разработка веб-приложения на платформе ASP.NET MVC для практического закрепления знаний о возможностях данной платформы и развития способности написания самостоятельных программ.

Объектом исследования данной курсовой работы процесс разработки веб-приложения на платформе ASP.NET MVC.

Предмет исследования: усовершенствование навыков работы с современной визуальной средой программирования Microsoft Visual Studio при разработке веб-приложения на платформе ASP.NET MVC.

Целью курсовой работы является разработка веб-приложение на платформе ASP.NET MVC по заданному условию в соответствии с вариантом Данная цель обусловила выделение следующих задач:

1. Изучить техническую литературу по заданной теме.
2. Осуществить выбор используемых технологий, базы данных, фреймворков, среды программирования и встроенных библиотек.
3. Спроектировать модель предметной области.
4. Разработать структуру базы данных. Реализовать объекты базы данных: таблицы, представления, хранимые процедуры, индексы, скалярные функции и функции, возвращающие табличные значения, необходимые для корректной работы информационной системы.
5. Реализовать бизнес-логику информационной системы с учетом выбранных методологий абстрагировано от пользовательского интерфейса.
6. Реализовать веб-приложение на платформе ASP.NET MVC, используя разработанную ранее бизнес-логику интернет-магазина.
7. Провести модульное тестирование (unit testing) и интеграционное тестирование.
8. Составить пояснительную записку для описания функциональных возможностей и сопровождения разрабатываемой системы.

Для достижения цели данной работы применены следующие методы научного исследования:

* изучение литературы и других источников по теме;
* обобщение;
* анализ;
* моделирование и проектирование.

Предполагаемая область применения веб-приложения на платформе ASP.NET MVC «Интернет-магазина зимних видов спорта»: реализация коммерческих идей в сети Интернет с целью получения прибыли на рынке сбыта товаров зимних видов спорта.

Структура пояснительной записки включает в себя введение, две главы, заключение, список литературы и приложение.

Во введении обоснована актуальность работы, определены объект, предмет, цель, актуальность, сформулированы задачи и методы исследования, дана предполагаемая область применения.

В первой главе «Теоретическая часть» описываются технологии, языки программирования и фреймворки применяемые при разработке веб-приложений. Также рассматривается интегрированная среда разработки и принципы модульного тестирования приложений.

Во второй главе «Практическая часть» приведены особенности построения и работы алгоритма, основных классов, объектов и сторонних библиотек, обосновывается выбор стратегии тестирования, приведено тестирование поведения приложения в различных ситуациях.

В заключении отражены краткие выводы и результаты поставленных задач.

# 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Компьютерные технологии без преувеличения можно назвать наиболее динамичной областью современных знаний, которые концентрируют в себе самые последние достижения в сфере науки и техники. Появление новых моделей процессоров и комплектующих, версий операционных систем и программного обеспечения происходит на фоне постоянного усложнения не только отдельных физических и программных компонентов, но и лежащих в их основе концепций. Разработка и совершенствование информационных систем приводит к необходимости поддержания единого стиля для различных версий программ при их постоянной доработке и модификации[1]. В настоящее время существует достаточно много различных методик разработки программного обеспечения.

Сейчас, в век развития не только информационных технологий, но и сетевого взаимодействия между многочисленными программами и их пользователями, Интернет предоставляет замечательную возможность для разработчиков использовать единую, относительно стандартизированную среду для своих программ. Браузеры, которые есть на компьютере практически каждого пользователя, позволяют сразу, без установки какого-либо стороннего программного обеспечения, открывать нужные пользователю веб-страницы, пользоваться услугами различных сервисов и получать доступ практически к любой информации. В итоге многие и крупные, и мелкие сервисы предлагают свои услуги именно в качестве веб-приложений. Самыми популярными примерами последних лет являются успешные запуски таких компаний, как Facebook, Google и отечественного Вконтакте, которые позволяют пользователям получать доступ к своим страницам практически с любого устройства, которое имеет подключение к сети Интернет. Практически всегда при создании веб-сервисов используются те или иные фреймворки или библиотеки, которые позволяют значительно ускорить разработку и избавить от шаблонного кода[2].

## 1.1 Технологии веб-разработки

Зачастую, выбор языка и фреймворков в процессе веб-разработки определяется знаниями программистов, которые хотят реализовать какой-либо проект. Также, существует разница между языком программирования и фреймворком. Язык программирования – это некий базовый синтаксис (бывают и со стандартными библиотеками), с помощью которого создаются приложения. Для упрощения создания программ и сайтов существуют различные библиотеки, которые предоставляет фреймворк. Бывают и такие языки и фреймворки, которые представляют собой одно целое (например, JSP и ASP.NET). Остальные языки используются без фреймворка (Perl и PHP). Языки веб-программирования и фреймворки классифицируются на клиентские и серверные (рис. 1.1). Программы, написанные на клиентских языках выполняются на стороне клиента, а серверные – на стороне сервера.

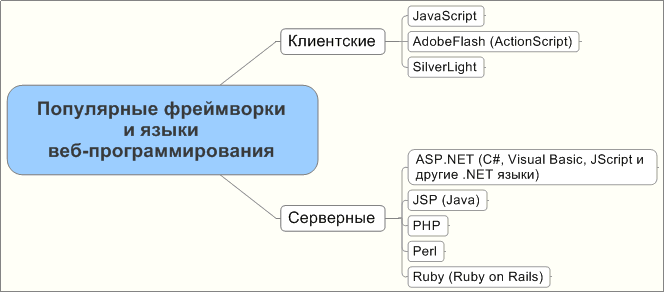


Рисунок 2.1 – Классификация популярных фреймворков и языков

веб-программирования

### 1.1.1 HTML

Язык гипертекстовой разметки, сокращенно HTML, является стандартным языком разметки, используемым для создания веб-страниц[3]. Наряду с CSS и JavaScript, HTML является основанием для создания веб–страниц, а также для создания пользовательских интерфейсов для мобильных и для веб–приложений. Веб–браузеры могут считывать HTML-файлы и визуализировать их в видимые или слышимые веб-страницы. HTML описывает структуру сайта семантически и, до появления стилей каскадных таблиц (CSS), отображались подсказки во внешнем виде документа (веб-страницы), что делает HTML языком разметки, а не языком программирования.

HTML элементы образуют некие блоки HTML-страниц. HTML отображает изображения и другие объекты, которые встроены и могут быть использованы для создания интерактивных форм. Предоставляются средства для создания структурированных документов, а также для форматирования структуры текста, таких как заголовки, параграфы, списки, ссылки, цитаты и другие объекты. HTML элементы разграничены тегами, написанные с использованием угловых скобок (рис. 1.2)



Рисунок 1.2 – Пример гипертекстовой разметки HTML

Теги, такие как <img/> и <input/> вводят контент непосредственно на страницу. Другие, такие как <р>...</p> выделяют и предоставляют информацию о тексте документа и могут включать в себя другие теги в качестве вспомогательных элементов. Браузеры не отображают HTML-теги, но используют их, чтобы интерпретировать содержание страницы.

В HTML можно вставлять скрипты, написанные на таких языках, как JavaScript, которые влияют на поведение веб-страниц. HTML разметка со стилями каскадных таблиц (CSS) может также быть доступна браузеру, чтобы определить внешний вид и расположение текста и других материалов.

### 1.1.2 JavaScript

JavaScript является, динамичным, не типизированным и интерпретированным языком программирования высокого уровня[4]. Он был стандартизован в спецификации языка ECMAScript. Наряду с HTML и CSS, JavaScript является одним из трех основных технологий Всемирной паутины (World Wide Web), с помощью которого создаётся контент. Большинство веб-сайтов используют JavaScript, и он поддерживается всеми современными веб-браузерами без плагинов. JavaScript является прототипом в основе которого лежат функции первого класса, что делает его объектно-ориентированным языком с императивными и функциональными стилями программирования. JavaScript имеет API для работы с текстами, массивами, датами и регулярными выражениями, но не включает в себя функции ввода/вывода, такие как создание сетей, системы хранения данных, или графических объектов, а опирается на них по факту принимающей среды, в которой он встроен.

Хотя есть сильные внешние сходства между JavaScript и Java, в том числе название языка, синтаксиса и соответствующих стандартных библиотек, но все же это два различных языка, которые сильно различаются по своей конструкции. JavaScript был создан под влиянием таких языков программирования, как Self и Scheme.

JavaScript также используется в средах, которые не используют веб-интерфейс (например, в PDF-документах или сайт-специфических браузерах и настольных виджетов). Новые и быстрые JavaScript виртуальные машины (VM) и платформы, построенные на них также увеличили популярность JavaScript для серверных веб-приложений. На стороне клиента, JavaScript традиционно реализован как интерпретируемый язык, но более поздние браузеры используют его в момент компиляции. JavaScript также используется в разработке игр, создании настольных и мобильных приложений, а также сетевого программирования на стороне сервера (например, Node.js).

### 1.1.3 Фреймворк jQuery

JavaScript считается довольно неудобным языком для написания веб-приложений. Отсутствие унифицированной поддержки различными браузерами, трудности отладки и пугающий синтаксис способны лишить новичков веры в возможность успешного освоения этого языка.

Справедливости ради следует отметить, что элементы, создающие видимые трудности, в то же время придают мощь этому языку, хотя вряд ли данный аргумент сделает JavaScript более привлекательным для начинающих веб-разработчиков и побудит их добавить этот язык в свой рабочий арсенал.

Процесс изучения JavaScript всегда был болезненным для разработчиков, и по мере того, как всеобщее чувство недовольства нарастало, некоторые из них взялись создавать библиотеки JavaScript, которые часто называют JavaScript-фреймворками.

Библиотеки этого типа призваны упростить использование языка JavaScript и сделать его более доступным как для новичков, так и для разработчиков со стажем за счет предоставления простых в применении функций, облегчающих решение повседневных задач. Эффективность библиотек становится особенно заметной при работе с асинхронным JavaScript и XML (AJAX). Поскольку решение соответствующих задач с применением непосредственно JavaScript оказывается более сложным.

Библиотеки JavaScript предоставляют упрощенный синтаксис для решения типовых задач, что позволяет сократить сроки разработки и ускорить процесс обучения новичков. Кроме того, они частично упрощают написание сценариев JavaScript, не зависящих от типа браузера, беря на себя всю работу по проверке совместимости с помощью собственных встроенных средств, что обеспечивает огромную экономию времени на этапе написания программного кода[5].

Возможности jQuery:

* Обращаться к любому элементу DOM (объектной модели документа) и не только обращаться, но и манипулировать ими.
* Работать с событиями.
* Легко осуществлять различные визуальные эффекты.
* Работать с AJAX (очень полезная технология, позволяющая общаться с сервером без перезагрузки страницы, но пока мы ее трогать не будем).
* Имеет огромное количество JavaScript плагинов, предназначенных для создания элементов пользовательских интерфейсов [6].

### 1.1.4 Технология AJAX

Ajax (от англ. Asynchronous Javascript and XML – «асинхронный JavaScript и XML») – подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером. В результате, при обновлении данных веб-страница не перезагружается полностью, и веб-приложения становятся быстрее и удобнее.

Ajax поднимает на новый уровень гибкость и интерактивность веб-приложений. Ушли или по крайней мере отодвинуты на второй план те времена, когда формы веб-страниц давали результат только после того, как был сделан запрос на сервер. Не стало и статичных HTML-страниц, которые можно прочитать, но которые не доступны действиям читателя. Сегодня разработчики веб-страниц могут предоставить непосредственно на странице почти всю функциональность, для получения которой ранее было необходимо обращаться к серверу. С помощью Ajax компоненты страницы могут быть свернуты или развернуты по необходимости, заполнены по требованию, и обратная связь с пользователем может варьироваться от выделения цветом до системы стратегически расположенных сообщений. Каждый элемент страницы может быть использован для запроса к веб службам, что позволяет создавать более "отзывчивые" и более удобные для пользователя страницы. Результат более строгое и упорядоченное приложение. Ajax-приложения могут использовать библиотеки и веб-сервисы таких компаний, как Google, Amazon, Microsoft, Yahoo!, а также такие независимые библиотеки, как Prototype, Dojo, MochiKit и jQuery. Для работы с Ajax вебсервисы часто могут использоваться "как есть" либо лишь слегка видоизмененными. Уникальность Ajax в том, что это передовая концепция, опирающаяся на довольно зрелую технологию. Даже самые "изощренные" эффекты могут быть созданы с комфортом и уверенностью; разработчики могут быть спокойны, зная, что даже самые необычные эффекты будут работать в большинстве, если не во всех, современных браузерах. Ajax-разработка имеет многообещающие аспекты, но в действительности к этой технологии можно отнести принцип "80/20": 80 процентов людей, заинтересованных в ней, нуждаются только в 20 процентах всех ее возможностей. Большинство дискуссий, посвященных Ajax, исследуют границы применения данной технологии[7].

### 1.1.5 Технология ASP.NET

ASP.NET является технологией создания веб-приложений и веб–сервисов[8]. ASP.NET разработана компанией Microsoft, чтобы позволить программистам создавать динамические веб-сайты, веб–приложения и веб–сервисы.

Впервые эта технология была выпущена в январе 2002 года с версией 1.0 .NET Framework, и является разработкой компании Microsoft технологии Active Server Pages технологии (ASP). ASP.NET построена на Common Language Runtime (CLR), что позволяет программистам писать код ASP.NET с помощью любого поддерживаемого языка .NET.

Преимущества ASP.NET перед ASP:

* компилируемый код выполняется быстрее, большинство ошибок отлавливается ещё на стадии разработки;
* значительно улучшенная обработка ошибок во время выполнения запущенной готовой программы, с использованием блоков try-catch;
* пользовательские элементы управления (controls) позволяют выделять часто используемые шаблоны, такие как меню сайта;
* использование метафор, уже применяющихся в Windows–приложениях, например, таких как элементы управления и события;
* расширяемый набор элементов управления и библиотек классов позволяет быстрее разрабатывать приложения;
* ASP.NET опирается на многоязыковые возможности .NET, что позволяет писать код страниц на VB.NET, Delphi.NET, Visual C#, J# и т. д.;
* возможность кэширования всей страницы или её части для увеличения производительности;
* возможность кэширования данных, используемых на странице;
* возможность разделения визуальной части и бизнес–логики по разным файлам («code behind»);
* расширяемая модель обработки запросов;
* расширенная событийная модель;
* расширяемая модель серверных элементов управления;
* наличие master-страниц для задания шаблонов оформления страниц;
* поддержка CRUD–операций при работе с таблицами через GridView;
* встроенная поддержка AJAX;
* ASP.NET имеет преимущество в скорости по сравнению с другими технологиями, основанными на скриптах.

### 1.1.6 Фреймворк ASP.NET MVC

ASP.NET MVC – это фреймворк для веб-разработки, основанный на платформе Microsoft .NET, который предоставляет разработчикам возможность создавать хорошо структурированные веб-приложения. Представленная как альтернатива Web Forms платформа ASP.NET MVC приобрела значительную популярность с момента первой публичной демонстрации ее предварительной версии в 2007 году, и на сегодняшний момент большое количество крупных веб-приложений создано посредством использования данной технологии.

ASP.NET MVC создан как альтернатива использованию технологии Web Forms при создании веб-приложений на платформе .NET. Эта платформа была впервые представлена компанией Microsoft в ноябре 2007 года, и с тех пор вышло уже 4 крупных релиза этой платформы. Третий крупный релиз, ASP.NET MVC 3 был выпущен в конце января 2011 года и стал первой версией платформы ASP.NET MVC, использующей .NET 4. ASP.NET MVC 4 работает как с .NET 4, так и с платформой .NET 4.5, о выпуске которой еще не было объявлено на момент написания этой книги. Аббревиатура MVC расшифровывается как Model-View-Controller (Модель-Представление-Контроллер) и представляет собой архитектурный паттерн, очень популярный в области веб-разработки. Будучи альтернативой технологии Web Forms, ASP.NET MVC использует другой подход к вопросу структурирования веб-приложений. Это означает, что вам не придется иметь дело с ASPX-страницами и элементами управления, обратными запросами или ViewState, а также жизненными циклами сложных событий. Вместо этого вы будете определять контроллеры, действия и представления. Тем не менее, платформа ASP.NET, лежащая в основе ASP.NET MVC, остается прежней, поэтому в ASP.NET MVC все еще применяются такие вещи, как HTTP-обработчики и HTTP-модули, и вы можете использовать в одном и том же приложении одновременно и MVC, и Web Forms страницы. Эти платформы, ASP.NET Web Forms и ASP.NET MVC, расположены рядом друг с другом над ядром платформы ASP.NET. Компоненты паттерна MVCвзаимодействуют между собой (рис. 1.3).

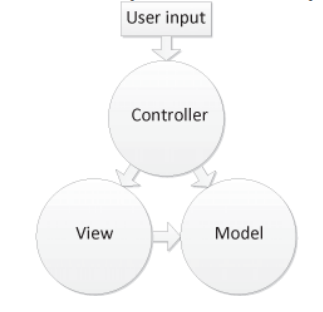


Рисунок 1.3 – ­ Компоненты MVC паттерна

На рисунке 1.3 показано как контроллер получает пользовательские входные данные, обращается к подходящей модели, а затем передает ее в представление. И контроллер, и представление зависят от модели, а модель не зависит ни от представления, ни от контроллера. ASP.NET MVC не является первой реализацией MVC паттерна на платформе .NET. Платформа MonoRail с открытым исходным кодом, создание которой первоначально было навеяно Ruby on Rails, принесла в 2005 году в .NET веб-разработку парадигму MVC, и ее сильное влияние можно наблюдать в ASP.NET MVC и по сегодняшний день.

В то время как ASP.NET Web Forms пытается полностью скрыть не сохраняющую состояние сущность разметки, ASP.NET MVC не пытается ее скрыть. За счет того, что в нем используется MVC паттерн, а также за счет возможности преобразования отдельного HTTP запроса в вызов конкретного метода ASP.NET MVC, позволяет получить навыки разработки, которая более знакома разработчикам, обладающим теоретическими познаниями в создании веб-приложений. Модель также радикально упрощена – уход от сложных событий жизненного цикла страницы, используемых в Web Forms, а также минимальное количество абстрактных конструкций HTTP.

В то время как ASP.NET Web Forms устойчиво связывает пользовательский интерфейс с его выделенным кодом, ASP.NET MVC поддерживает конструкцию, в которой пользовательский интерфейс (представление) сохраняет свою изолированность от кода, управляющего им (контроллера). При хорошей реализации это означает, что разработчикам легче передвигаться по приложениям, а также такая возможность упрощает процесс сохранения приложения – то, что вы внесете изменения в контроллер, еще не означает, что вам придется изменять пользовательский интерфейс.

С помощью отделения логики приложения от его пользовательского интерфейса ASP.NET MVC упрощает тестирование отдельных компонентов приложения. Классы контроллеров могут быть протестированы без тестирования реального пользовательского интерфейса. В отличие от Web Forms MVC контроллеры не имеют прямой зависимости от имеющего позорную славу, нетестируемого класса HttpContext, а вместо этого полагаются на абстракцию, что упрощает процесс написания автоматизированных тестов[9].

Таким образом, применение паттерна MVC в ASP.NET привносит концепции близости к протоколу, разделения и тестируемости.

## 1.2 Модульное тестирование

Модульное тестирование (Unit testing) – тестирование каждой атомарной функциональности приложения отдельно, в искусственно созданной среде. Именно потребность в создании искусственной рабочей среды для определенного модуля, требует от тестировщика знаний в автоматизации тестирования программного обеспечения, некоторых навыков программирования. Данная среда для некоторого юнита создается с помощью драйверов и заглушек. Каждая сложная программная система состоит из отдельных частей - модулей, выполняющих ту или иную функцию в составе системы. Для того, чтобы удостовериться в корректной работе всей системы, необходимо вначале протестировать каждый модуль системы по-отдельности. В случае возникновения проблем при тестировании системы в целом это позволяет проще выявить модули, вызвавшие проблему, и устранить соответствующие дефекты в них. Такое тестирование модулей по отдельности получило называние модульного тестирования (unit testing).

Основная цель модульного тестирования - удостовериться в соответствии требованиям каждого отдельного модуля системы перед тем, как будет произведена его интеграция в состав системы.

Первая задача - классическая задача тестирования, включающая в себя не только разработку тестового окружения и тестовых примеров, но и выполнение тестов, протоколирование результатов выполнения, составление отчетов о проблемах.

Вторая задача больше свойственна "легким" методологиям типа XP, где применяется принцип тестирования перед разработкой (Test-driven development), при котором основным источником требований для программного модуля является тест, написанный до реализации самого модуля.

Третья задача связана с поддержкой процесса изменения системы. Достаточно часто в ходе разработки требуется проводить рефакторинг модулей или их групп - оптимизацию или полную переделку программного кода с целью повышения его сопровождаемости, скорости работы или надежности. Модульные тесты при этом являются мощным инструментом для проверки того, что новый вариант программного кода выполняет те же функции, что и старый.

Последняя, четвертая, задача сопряжена с обратной связью, которую получают разработчики от тестировщиков в виде отчетов о проблемах. Подробные отчеты о проблемах, составленные на этапе модульного тестирования, позволяют локализовать и устранить многие дефекты в программной системе на ранних стадиях ее разработки или разработки ее новой функциональности.

В силу того, что модули, подвергаемые тестированию, обычно невелики по размеру, модульное тестирование считается наиболее простым (хотя и достаточно трудоемким) этапом тестирования системы. Однако, несмотря на внешнюю простоту, с модульным тестированием сопряжены две проблемы.

Первая из них связана с тем, что не существует единых принципов определения того, что в точности является отдельным модулем.

Вторая заключается в различиях в трактовке самого понятия модульного тестирования - понимается ли под ним обособленное тестирование модуля, работа которого поддерживается только тестовым окружением, или речь идет о проверке корректности работы модуля в составе уже разработанной системы. В последнее время термин "модульное тестирование" чаще используется во втором смысле, хотя в этом случае речь скорее идет об интеграционном тестировании[10].

## 1.3 Среда разработки Microsoft Visual Studio

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, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

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. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения[11].

# Вывод к разделу 1

На основании рассмотренного теоретического материала в разделе проанализированы возможные для использования фреймворки и языки веб–программирования, были отражены их сильные и слабые стороны. В результате был выбран современный и мощный для разработки веб-сайта фреймворк ASP.NET MVC с применением JavaScript-фреймворка jQuery и технологии Ajax. Для тестирования бизнес-логики приложения применены модульные тесты на основе открытой среды юнит-тестирования NUnit для .Net приложений. Разработка веб-приложения производится в интегрированной среде разработки Microsoft Visual studio 2017, т.к. она поддерживает все вышеперечисленные технологии.

# 2 ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1 Разработка приложения

### 2.1.1. Моделирование предметной области

Одной из основных частей информационной системы интернет-магазина является база данных. Для разработки базы данных базы данных был выбран язык Transact-SQL и соответствующий сервер MS SQL, среда разработки – Microsoft SQL Server Management Studio 2012.

В ходе проектирования базы данных интернет-магазина были созданы схемы [store] и [lk] а также 13 таблиц с советующими первичными и внешними ключами (рис 2.1).

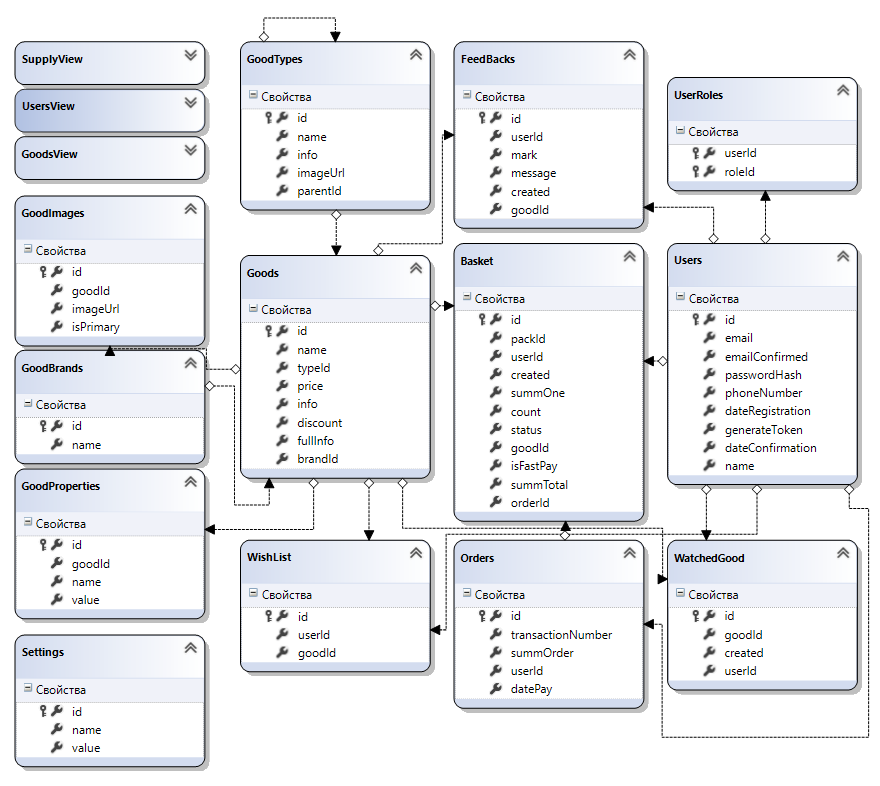


Рисунок 2.1 – Диаграмма базы данных интернет-магазина

Рассмотрим назначение таблиц базы данных интернет-магазина:

1. Таблица [store].[Users] содержит список зарегистрированных пользователей с логином входа, хешированным паролем и контактной информацией
2. Таблица [store].[UserRoles] содержит описание типов пользователей, таких как администратор, покупатель и т.д.
3. Таблица [store].[Settings] содержит глобальные настройки информационной системы, такие как настройки SMTP сервера и шаблоны писем Email, отправляемых при регистрации пользователей и восстановлении пароля.
4. Таблица [store].[GoodBrands] содержит список брендов, реализуемых товаров зимних видов спорта.
5. Таблица [store].[GoodTypes] содержит информацию о группе, к которой относится товар. Группы могут входить в родительскую группу, если обладают общей спецификой.
6. Таблица [store].[Goods] содержит информацию о товарах (название, тип товара, бренд, цена, процентная ставка скидки и краткая информация и полная информация).
7. Таблица [store].[GoodProperties] содержит информацию о характеристиках товара в формате «название характеристики»-«значение характеристики». Один товар может иметь несколько характеристик одновременно.
8. Таблица [lk].[GoodImages] содержит url-адреса изображений товаров интернет-магазина. Кроме того одно из изображений каждого товара имеет флаг isPrimary, которое определяет главное изображение товара в каталоге.
9. Таблица [lk].[Feedback] содержит отзывы пользователей интернет магазина об товарах из каталога с их оценками и комментариями.
10. Таблица [lk].[WishList] содержит список товаров интернет-магазина, которые пользователи добавили в избранное.
11. Таблица [lk].[WatchedGoods] содержит историю просмотров товаров пользователями для возможности анализа популярности товаров и отображения наиболее популярных товаров на главной странице.
12. Таблица [lk].[Basket] хранит данные о товарах, которые зарегистрированные пользователи добавили в корзину: количество единиц товара, сумма позиции корзины, статус элемента корзины (добавлен, удален или оплачен).
13. Таблица [lk].[Orders] содержит информацию о заказах. (дата заказа, сумма заказа, идентификатор покупателя и номер транзакции).

Для реализации функциональных возможностей базы данных были реализованы следующие процедуры и функции:

* процедура добавления элемента в корзину;
* процедура обновления списка просмотренных товаров;
* процедура обновления пароля пользователя;
* процедура генерации токена для смены пароля или регистрации;
* функция, возвращающая полную информацию о товаре;
* функция, возвращающая список наиболее популярных товаров;
* функция, проверяющая валидность комбинации логин-пароль;
* процедура оплаты товара;
* процедура регистрации пользователя;
* процедура подтверждения email пользователя.

Для взаимодействия с базой данных применен API-интерфейс LINQ to SQL, входящая в состав технологии ADO.NET. За счет автоматически сгенерированных классов из сущностей базы данных появляется возможность прямого обращения к хранимым данным в форме объектов классов C#, что сокращает время разработки взаимодействия с БД, т.к. не требует реализацию адаптеров, мапперов и других программных интерфейсов доступа к базам данных самостоятельно.

Таким образом, спроектированная модель предметной области полностью охватывает все объекты, участвующие в процессе работы интернет-магазина.

### 2.1.2. Реализация бизнес-логики приложения

Спроектирована и реализована бизнес-логика приложения, призванная решать ряд задач, необходимых для полноценной работы интернет-магазина. Структурно модель предметной области делится 4 модуля:

1. «AccontAction» – регистрационно-аутентификационная логика интернет-магазина. Позволяет регистрировать пользователя, проверять валидность комбинации логин-пароль, изменять пароль, отправлять письма на электронную почту пользователя для ее подтверждения или изменения пароля, подтверждать регистрации пользователя и получать контактную информацию о пользователе. В качестве примера ниже приведена реализация метода аутентификации пользователя в системе (рис 2.2).

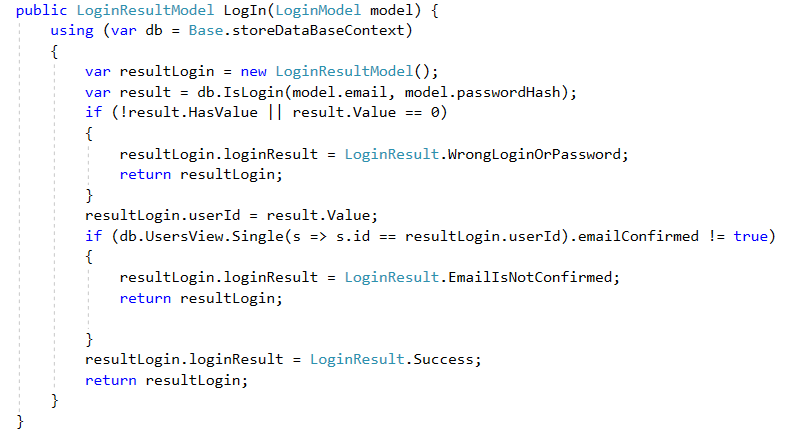


Рисунок 2.2 – Метод аутентификации пользователя в системе

Здесь на вход подается логин и хеш пароля с применением алгоритма хеширования SHA256, в теле метода производится подключение к базе данных для сопоставления переданных параметров с имеющимися в таблице зарегистрированных пользователей. Результатом метода является объект, содержащий в себе результат проверки переданных параметров (неправильная пара логин-пароль, неподтвержденный пользователь, успешная аутентификация) и идентификатор пользователь в случае успешно пройденной проверки.

2. «StoreAction» – включает в себя общую логику интернет-магазина без привязки к конкретному пользователю. Позволяет получить следующую информацию:

* список всех групп товаров;
* список брендов, представленных в интернет-магазине;
* список товаров, входящих в указанную группу;
* полная информации об указанном товаре (технические характеристики, описание товара, наименование бренда, стоимость, процент скидки, отзывы пользователей с оценками и средняя оценка товара);
* полный список товаров, представленных в интернет-магазине;
* список наиболее популярных товаров, который рассчитывается на основе собранной статистике об просмотрах товаров пользователями;
* список товаров, которые в данных момент просматривают пользователи интернет-магазина;
* список отзывов об указанном товаре.

3. «CustomerAction». Данный модуль решает функциональные задачи, связанные с конкретным пользователем интернет магазина:

* добавление товара в корзину с указанием количества единиц в позиции;
* получение актуального списка элементов корзины пользователя;
* удаление указанного элемента корзины;
* полная очистка корзины пользователя;
* изменение количества единиц товара для одной указанной позиции корзины пользователя;
* создание оплаты для собранной корзины пользователем;
* получение списка совершенных заказов пользователя;
* создание отзыва о товаре с оценкой и комментарием;
* добавление товара в избранное;
* получение списка избранных товаров пользователя;
* удаление элемента(ов) списка избранных товаров пользователя.

1. «OdmenAction» – модуль бизнес-логики, решающий задачи по администрированию интернет магазина. Данный модуль содержит в себе реализацию следующих процессов:

* создание, удаление и редактирование группы/подгруппы товаров;
* создание, удаление и редактирование брендов;
* создание, удаление и редактирование товара (название, описание, стоимость, процент скидки);
* создание, удаление и редактирование характеристик товара;
* привязка, удаление привязки, назначение главным изображений товаров;
* изменение статуса указанного заказа пользователя.

Таким образом, реализованная бизнеса-логика для спроектированных ранее объектов предметной области полностью решают функциональные задачи для создания, наполнения и поддержки интернет-магазина администрацией, а также для комфортного его использования рядовыми пользователями. Учитывая, что реализованная часть информационной системы абстрагирована от пользовательского интерфейса, это позволит в дальнейшем без проблем расширить ее функциональное применение, например, в виде мобильного приложения, создав WEB API на основе реализованной библиотеки. Другой вариант расширения информационной системы может стать создание настольного приложения в виде рабочего места кассира для решения задач розничной торговли на основе реализованной бизнес-логики.

### 2.1.3 Разработка веб-приложения

В процессе разработки веб-приложения интернет-магазина был создан пустой проект ASP.NET MVC 3 и подключена ранее разработанной, скомпилированной и скомпонованной библиотека бизнес-логики интернет-магазина. Учитывая необходимость использования паттерна Model-View-Controller, были реализованы следующие контроллеры с соответствующими им представлениями:

1. «AccountController». Генерирует html-верстку для страниц регистрации, авторизации, восстановления и смены пароля. Рассмотрим пример реализации представления восстановления пароля пользователя «ForgotPassword.cshtml» (рис 2.3).



Рисунок 2.3 – Представление «ForgotPassword.cshtml»

Здесь, как и во всех других реализованных представлениях, используется синтаксис разметки MVC-приложения Razor-код, а также различные предопределенные html-хелперы такие как «LaberlFor», «TextBoxFor», «ValidationMessageFor», которые генерирует html-разметку в процессе генерации страницы на основе полученных данных из модели. В данном случае модель для представления «ForgotPassword» описывает класс «ForgotPasswordViewModel» (рис 2.4).

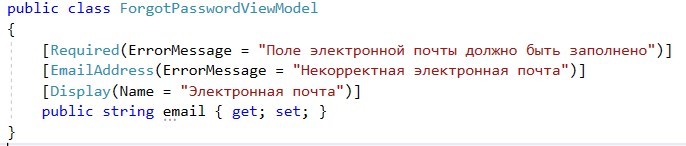


Рисунок 2.4 – Модель данных для представления «ForgotPassword»

В данном случае модель данных состоит из одного параметра email (электронная почта), которую пользователю необходимо ввести, чтобы получить письмо с описанием порядка восстановления доступа к своему аккаунту. Кроме того, этот параметр содержит 3 атрибута, которые участвуют в отрисовке веб-страницы пользователю:

а) атрибут «Required» устанавливает обязательность поля ввода (в данном случае email) перед отправкой на сервер. Параметр «ErrorMessage» определяет текст ошибки, если не выполнить это требование.

б) атрибут «EmailAddress» устанавливает, что вводимые данные должны соответствовать формату электронной почты, т.е. содержать символ «@», доменное имя и тд. Параметр «ErrorMessage» определяет текст ошибки, если не выполнить это требование.

в) атрибут «Display» с параметром «Name» определяют какое название параметра будет генерироваться при использовании html-хелпера «LabelFor» для данного элемента модели.

Важным является html-хелпер «AntyForgeryToken», который применен в представлении «ForgotPassword.cshtml» (рис 2.3). Он позволяет сгенерировать уникальный ключ внутри скрытого поля, а также второй уникальный ключ в cookie-файл при GET запросе на сервер с целью предотвращения проведения CSRF-атаки («фишинга»). Защита заключается в проверке полученных от пользователя ключей, которые ранее сгенерировал сервер. Если ключи не соответствуют заложенному сервером алгоритму или вовсе отсутствуют, то это говорит о том, что совершается CSRF-атака и данные были скомпрометированы. Такие запросы отвергаются сервером.

2. «CustomerController». Содержит в себе методы для отрисовки страниц корзины пользователя и списка избранных товаров (представления «Basket.cshtml» и «WishList.cshtml»). Кроме того, данный контроллер содержит в себе методы асинхронного вызова без перезагрузки страницы, результатом которых являются различные объекты данных. Рассмотрим реализацию такого метода, в котором применяются такие элементы веб-разработки как Ajax, jQuery и JSON, на примере метода удаления элемента корзины пользователя. Клиентская часть состоит из функции JavaScript с применением фреймворка jQuery (рис. 2.5).

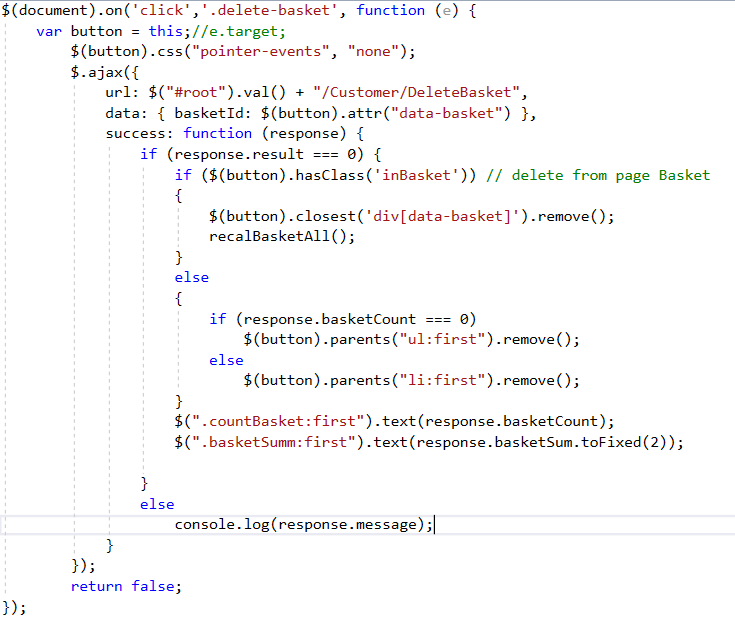


Рисунок 2.5 – Клиентская функция удаления элемента корзины

Данный метод устанавливает слушателя на элемент с классом «delete-basket» после загрузки страницы на стороне клиента, который в случае клика по этому элементу выполняет следующие действия:

1. блокирует нажатую кнопку до конца успешного выполнения запроса;
2. получает значение data-атрибута «data-basket» нажатой кнопки, в которой содержится идентификатор элемента корзины;
3. выполняет асинхронный Ajax-метод с полученным ранее параметром в метод «DeleteBasket» контроллера «Customer».
4. в случае успешного выполнения запроса, определяемого значением поля «result» объекта, полученного в ответе от сервера, удаляется строка с элементом корзины и производится пересчет ее общей суммы. Затем обновляется значение количества и общей суммы корзины в шапке сайта. Если запрос выполнен неудачно, то удаление не производится, а сообщение от сервера записывает в лог.

В свою очередь северная часть обрабатывает удаление элемента корзины в методе «DeleteBasket» (рис. 2.6).

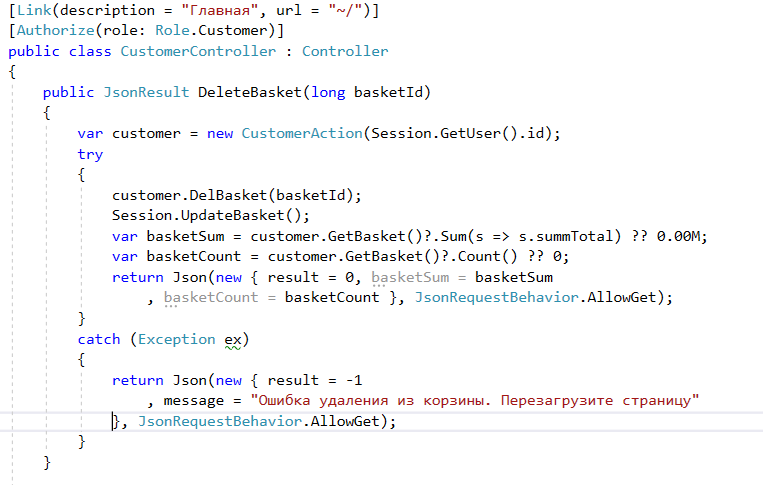


Рисунок 2.6 – Action-метод удаления элемента корзины

В контроллере «CustomerController» важным является наличие атрибута «Authorize», который распространяет своё действие на все методы внутри контроллера. Данный атрибут определяет права на использование методов в зависимости от роли пользователя. Значение параметра «Role.Customer» говорит о том, что для использования метода удаления элемента корзины пользователь должен быть авторизован. Такая техника позволит обезопасить от попытки получения доступа к информации чужого пользователя, т.к. в данном случае гарантируется, что производится взаимодействие с данными, которые привязаны к текущему пользователю. Аналогичным образом произведено ограничение прав администрирования интернет-магазина только для конкретных пользователей, обладающих соответствующими правами.

Метод «DeleteBasket» создает объект класса «CustomerAction», который отвечает за бизнес-логику покупателя, описанную в главе 2.1.2, передавая в конструктор идентификатор пользователя, который хранится в сессии с момента авторизации. Затем производится удаление элемента корзины и пересчет общей суммы и количества оставшихся элементов. Возвращаемым значением Action-метода является объект JSON с результатами выполнения операции.

1. «HomeController». Реализовывает отрисовку главной страницы веб-приложения, страницы контактной информации интернет-магазина и частичного представления ранее просмотренных товаров.
2. «StoreController». Реализовывает отрисовку страниц каталога товаров, страницы информации о товаре, логику фильтрации и поиска товаров.
3. «OdmenController». Реализовывает веб-страницы блока администрирования: создания, удаления и редактирования элементов интернет-магазина.
4. «ErrorController». Реализует отрисовку страниц ошибок таких как HTTP404 и HTTP500.

Таким образом, было разработано и реализовано веб-приложение на платформе ASP.NET MVC «Интернет-магазин зимних видов спорта» с использованием современных технологий таких как jQuery, LINQ to SQL, AJAX. Данный программный продукт отвечает всем функциональным требованиям современных интернет-магазинов.

## 2.2 Тестирование продукта

Тестирование программного обеспечения – процесс выявления ошибок в программном обеспечении (ПО). Существующие на сегодняшний день методы тестирования ПО не позволяют однозначно и полностью устранить все дефекты и ошибки и установить корректность функционирования анализируемой программы особенно в закрытых частных программах. Поэтому все существующие методы тестирования действуют в рамках формального процесса проверки исследуемого или разрабатываемого ПО. Существует множество подходов к решению задачи тестирования и верификации ПО, но эффективное тестирование сложных программных продуктов – это процесс в высшей степени творческий, не сводящийся к следованию строгим и чётким процедурам или созданию таковых. Тестирование ПО – попытка определить, выполняет ли программа то, что от неё ожидают. Как правило, никакое тестирование не может дать абсолютной гарантии работоспособности программы в будущем.

### 2.2.1 Модульные тесты

Для осуществления модульного (юнит) тестирования был создан проект «DomainObject.Tests», загружена открытая среда юнит-тестирования «NUnit» и подключена библиотека бизнес-логики интернет магазина для покрытия ее открытых методов тестами. Для этого в созданном проекте создается класс и методы с атрибутами [Test], внутри которых выполняется подготовка входящих параметров (заполнение начальными данными), вызов метода бизннес-логики и обработка результата выполнения. Тест считается успешно пройденным, если возвращаемое значение соответствует ожидаемому результату. Ниже приведен пример модульного тестирования метода регистрации пользователя (рис. 2.7)

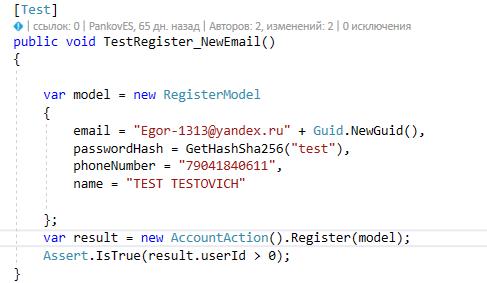


Рисунок 2.6 – Юнит-тест метода регистрации пользователя

В данном случае в методе «TestRegister\_NewEmail» создается модель с уникальным значением электронной почты, хешем пароля и контактной информации, после чего производится вызов метода регистрации пользователя с передачей созданной модели. Тест считается успешно пройденным, если в результате выполнения метода будет сгенерировано значение идентификатора пользователя.

В результате было создано и отработано 28 юнит-тестов по всем четырем модулям бизнес-логики интернет-магазина. Результаты модульного тестирования отражены в обозревателе тестов (рис. 2.7)

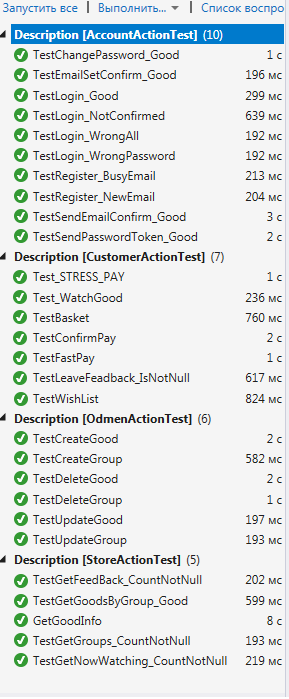


Рисунок 2.7 – Результат прохождения модульного тестирования

Таким образом, информационная система успешно прошла все этапы модульного тестирования без сбоев.

Рассмотрим реакцию программы на различные исключительные ситуации.

Тест №1. Запустить приложение.

Ожидаемый результат – появление консоли с приветствием и текстовым меню.

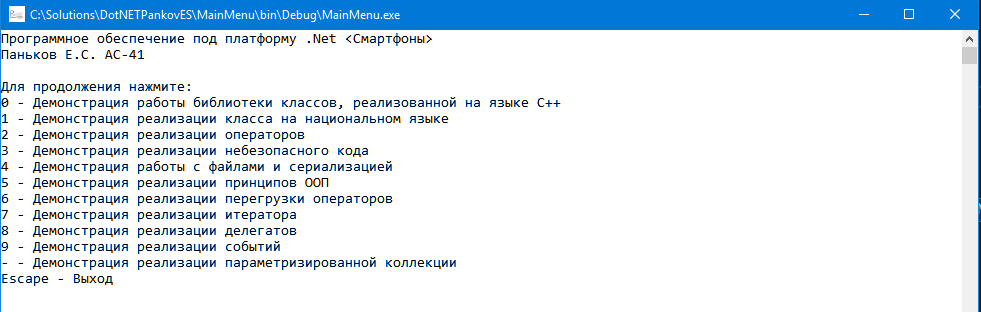


Рисунок 2.2 – Тест №1

По итогам теста выявлено, что результат соответствует ожидаемому (рис 2.2).

ЗАКЛЮЧЕНИЕ

По итогам выполнения курсовой работы было спроектировано и реализовано программное приложение «Программное обеспечение под платформу .Net «Смартфоны» в соответствии с техническим заданием. Таким образом, было разработано и реализовано веб-приложение на платформе ASP.NET MVC «Интернет-магазин зимних видов спорта», отвечающее всем функциональным требованиям современных интернет-магазинов, которое готово к внедрению в продуктивной среде для реализации товаров сферы спортивного инвентаря зимних видов.

Для этого была построена архитектура программы, разработаны алгоритмы для реализации данного программного продукта с использованием современной технологии программирования и методологий объектно-ориентированного программирования. Тестирование, разрабатываемого приложения показало работоспособность, целостность и структурированность, корректную работу в исключительных ситуациях. Интерфейс программы интуитивно прост и понятен, для ее использования не нужно специального обучения пользователя.

В ходе выполнения курсовой работы изучены нормативные документы, регламентирующие состав, содержание и форму технической документации на разрабатываемый программный продукт. В соответствии с которыми составлена и оформлена пояснительная записка для описания функциональных возможностей и сопровождения разрабатываемой системы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Сазонов, В.М. Технологии и языки программирования [Электронный ресурс] : Сборник лекций / В.М. Сазонов. ­– Виртуальная школа компьютерных технологий, 2006. – 23 с. Режим доступа: www.v-school.narod.ru/INI/TEXT/INI-8.doc (дата обращения: 1.11.2018).
2. Титков А.В. Создание веб-приложений [Электронный ресурс] : учебное пособие / А. В. Титков, С. А. Черепанов. – Томск : Эль Контент, 2014. – 72 с. Режим доступа: https://vk.com/doc203672322\_437187050 (дата обращения: 15.11.2018).
3. Гончаров, А. Самоучитель HTML [Текст] / А. Гончаров. – М.: Питер, 2012. – 260 с.
4. JavaScript [Электронный ресурс]. Режим доступа: http://javascript.ru (дата обращения 28.11.2018).
5. Ленгсторф, Джейсон. PHP jQuery для профессионалов [Текст] : Пер. с англ. / Джейсон Ленгсторф – М.: ООО «И.Д. Вильямс», 2011. – 352 с.
6. JQuery [Электронный ресурс]. Режим доступа: https://ru.bmstu.wiki/JQuery (дата обращения 28.11.2018).
7. Пауэрс, Шелли. Добавляем Ajax [Текст] : Пер. с англ. / Шелли Пауэрс – СПб.: БХВ-Петербург, 2009. – 448 с.
8. 104. ASP.NET [Электронный ресурс]. Режим доступа: http://asp.net/ (дата обращения 9.12.2018).
9. Палермо, Джеффри. ASP.NET MVC 4 в действии [Текст] : Пер. с англ. / Джеффри Палермо и др. – M.: Manning, 2012. – 408 с.
10. Лекция 11: Модульное тестирование [Электронный ресурс]. Режим доступа: https://www.intuit.ru/studies/courses/1040/209/lecture/5409 (дата обращения 9.12.2018).
11. Интегрированная среда разработки Visual Studio [Электронный ресурс]. Режим доступа: https://msdn.microsoft.com/ru-ru/library/dn762121.aspx (дата обращения: 1.11.2018).