**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

Механико-математический факультет

[Кафедра веб-технологий и компьютерного моделирования](http://www.bsu.by/ru/main.aspx?guid=4361)

**Курсовая работа**

**«ASP.NET MVC. Веб-проект. Интернет-магазин»**

Выполнил студент:

2 курса 2 группы

Специальность:

математика (информационные технологии)

Гончаров Александр Витальевич

Научный руководитель:

кандидат физико-математических наук, доцент

Кремень Елена Васильевна

Минск, 2019

СОДЕРЖАНИЕ

[Введение 3](#_Toc8606918)

[1. Постановка задачи 4](#_Toc8606919)

[1.1 Описание предметной области 4](#_Toc8606920)

[1.2 Модель сайта 4](#_Toc8606921)

[2. Разработка интернет-магазина 6](#_Toc8606922)

[2.1 Введение в разработку 6](#_Toc8606923)

[2.2 Создание структуры интернет-магазина 6](#_Toc8606924)

[2.3 Разбиение на страницы, стилизация контента и навигация 10](#_Toc8606925)

[2.4 Создание корзины покупок и отправки заказов 11](#_Toc8606926)

[2.5 разработка страниц администрирования 13](#_Toc8606927)

[3. Описание работы сайта 18](#_Toc8606928)

[3.1 Демонстративный пример работы сайта 18](#_Toc8606929)

[3.2 Демонстративный пример работы страниц администратора 25](#_Toc8606930)

[Заключение 28](#_Toc8606931)

[Список использованных источников: 29](#_Toc8606932)

## Введение

Тема данной курсовой работы – «ASP.NET MVC. Веб-проект. Интернет-магазин».

Данный курсовой проект был выполнен с целью практического освоения основных приемов и правил создания веб-сайта.

В настоящее время Интернет представляет собой один из самых активно развивающихся средств информации. По статистике, значительное число населения имеют дома компьютер и имеют доступ к сети Интернет.

В современном обществе больше всего цениться время, люди очень много тратят времени на покупки. Для того что бы экономить время на покупки начали создаваться интернет - магазины. Интернет - магазин - это сайт, на котором при помощи браузера можно заказать необходимые товары, которые курьер доставит домой или в офис. Преимущества такого вида покупки очевидны:

- не надо никуда идти и стоять в очереди;

- не надо думать о доставке товара - это сделают сотрудники интернет-магазина;

- можно сэкономить время;

- просто и удобно оформить заказ.

Настоящая работа посвящена разработке интернет-магазина на технологии ASP.NET MVC 5 с использованием движка визуализации Razor, Entity Framework, Ninject, Bootstrap.

## 1. Постановка задачи

## 1.1 Описание предметной области

Предметной областью называется часть реального мира, представляющая интерес для данного исследования.

Для проектируемого сайта предметной областью является информация о фильмах, имеющихся в продаже на сайте.

В разделе «Магазин» посетитель может узнать информацию о фильмах, имеющихся в продаже (жанр, стоимость, название, описание) и при желании добавить фильм в свою корзину покупок.

Раздел «Корзина» предоставляет информацию о товарах, находящихся в корзине, а также позволит посетителю удалить из своей корзины нежелательный фильм, вернуться к списку фильмов и продолжить добавление их в корзину, либо оформить заказ с существующим списком покупаемых товаров

Раздел «Оформление заказа» предоставляет пользователю возможность ввести для последующей доставки товара.

Раздел «Страница администрации» является недоступным для обычного пользователя, для доступа к нему необходима авторизация с использованием данных администратора

Раздел «Вакансии» и «Контакты» расскажет об актуальных вакансиях предприятия и соответственно всех актуальных контактах: телефоны, электронный адрес, физический адрес, реквизиты.

 Целью разработки интернет-магазина является последующая продажа своих товаров посетителям.

## Модель сайта

Одним из важных моментов является разграничение прав доступа у различных категорий пользователей.

1. Посетитель магазина.
2. Администратор.

Посетитель магазина при входе на сайт обладает базовыми правами. Он может просматривать всю информацию о фильме: название, описание, жанр, стоимость. При желании он может добавить товар в корзину и позже оформить заказ. Также, при необходимости, посетитель может удалить товар из своей корзины.

Администратор может авторизоваться на сайте для того использовать CRUD (create, read, update, delete)-средства для управления каталогом товаров.

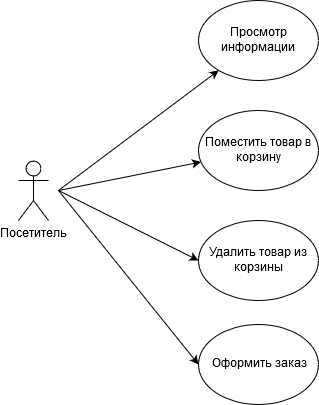


Рисунок 1.2.1 Диаграмма прецедентов категории "Посетитель”

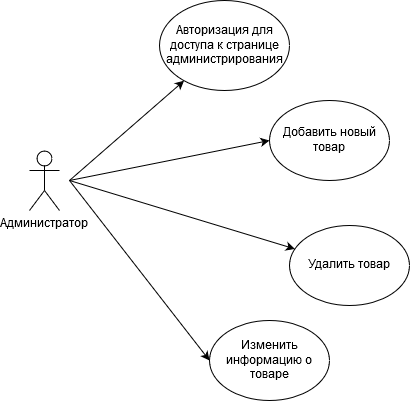


Рисунок 1.2.2 Диаграмма прецедентов категории "Администратор"

## 2. Разработка интернет-магазина

## Введение в разработку

* Выбранный язык программирования: C#;
* Используемые библиотеки: ASP.NET MVC, Ninject, Entity Framework, Bootstrap, jQuery.
* Среда разработки: Visual Studio 2019\2017.
* SQL сервер: Microsoft SQL Server 2012 Express

## 2.2 Создание структуры интернет-магазина

Интернет-магазин будет следовать классическому подходу, который повсеместно используется в интернет-магазинах. Мы создадим онлайновый 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, будет создана административная область, которая включает в себя для управления каталогом товаров. Она будет защищена для предотвращения нежелательных манипуляций со списком товаров.

|  |  |  |
| --- | --- | --- |
| Имя проекта | Шаблон проекта Visual Studio | Назначение |
| Domain | Class Library (Библиотека классов) | Содержит сущности и логику предметной области; настраивается на обеспечение постоянства посредством хранилища, которое создано с помощью инфраструктуры Entity Framework. |
| WebUI | ASP.NET MVC Web Application.  (Веб-приложение ASP.NET MVC 5) | Содержит контроллеры и представления; выступает в качестве пользовательского интерфейса для приложения. |
| UnitTests | Unit Test Project (Проект модульного тестирования) | Содержит модульные тесты для других двух проектов |

Таблица 2.2.1 Три проекта для приложения интернет-магазин

Во избежание проблем на этапе компиляции необходимо настроить зависимости между проектами, а также от определенных сборок Microsoft.

|  |  |  |
| --- | --- | --- |
| Имя проекта | Зависимости внутри решения | Зависимости от сборок |
| Domain | - | System.ComponentModel.DataAnnotations. |
| WebUI | Domain | - |
| UnitTests | Domain,WebUI | System.Web,Microsoft.CSharp |

Таблица 2.2.2 Обязательные зависимости между проектами

Следующий шаг – создание первой сущности предметной области, товара (фильма). Структура сущности:

* FilmId (int)
* Name (string)
* Description (string)
* Genre (string)
* Price (decimal)
* ImageData (byte [])
* ImageMimeType (string)

Далее необходимо создать абстрактное хранилище. Поля и методы интерфейса IFilmRepository:

* IEnumerable<Film> Films {get;}
* Void SaveFilm (Film film)
* Film DeleteFilm(int filmId)

Интерфейс использует другой интерфейс IEnumerable<T>, чтобы позволить вызывающему коду получать последовательность объектов Film, ничего не сообщая о том, как или где хранятся или извлекаются данные. Класс, зависящий от интерфейса IFilmRepository, может получить объекты Film, ничего не зная о том, откуда они поступают или каким образом класс реализации будет их доставлять. В этом и состоит суть шаблона хранилище (repository).

Также необходимо создать контроллер FilmController, в конструкторе которого нужно объявить зависимость от интерфейса IFilmRepository. Это приведет к внедрение библиотекой Ninject данной зависимости для хранилища фильмов при создании экземпляра класса контроллера. Вдобавок нужно добавить метод действия List (), который будет визуализировать представление, отображающее полный список товаров.

Прежде чем можно будет реализовать настоящее хранилище, необходимо настроить базу данных и заполнить ее данными. Для базы данных будет применятся SQL Server 2012, а доступ к ней будет осуществляться с помощью Entity Framework (EF) – инфраструктуры ORM для платформы .NET. Инфраструктура ORM представляет таблицы, столбцы и строки реляционной базы данных с помощью обычных объектов C#. Cледует упомянуть, что LINQ может работать с различными источниками данных, и один из них - Entity Framework.

Это область, где доступен широкий диапазон инструментальных средств и технологий. Можно взаимодействовать не только с разными реляционными базами данных, но также с хранилищами объектов, хранилищами документов и некоторыми вообще экзотическими альтернативами. Кроме того, существуют другие инфраструктуры ORM для .NET, каждая из которых использует слегка отличающийся подход - подобное разнообразие позволяет выбрать то, что наиболее подходит для конкретных проектов.

В проекте использована инфраструктура Entity Framework по нескольким причинам: она проста и ее легко запустить в работу; она прекрасно интегрирована с LIN. В ранних версиях этой библиотеки имелись серьезные недостатки, но в текущих версиях эти недостатки были исправлены, и на данный момент библиотека эффективна и обладает большими функциональными возможностями.

Для создания базы данных используем инструменты Visual Studio и заполняем созданную базу данных, например, таким образом:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Genre | Price |
| Начало | Кобб — талантливый вор, лучший из лучших в опасном искусстве извлечения: он крадет ценные секреты из глубин подсознания во время сна, когда человеческий разум наиболее уязвим. Редкие способности Кобба сделали его ценным игроком в привычном к предательству мире промышленного шпионажа, но они же превратили его в извечного беглеца и лишили всего, что он когда-либо любил. | Драма | 24,91 |

Таблица 2.2.3 Пример заполнения базы данных

FilmId-столбец идентичности, для которого SQL Server будет генерировать уникальное значение при переходе на следующую строку. Изображение корректно добавлять вручную на сайте.

Далее нужно использовать библиотеку Ninject для создания специального распознавателя зависимостей, который MVC Framework будет применять при создании экземпляров объектов внутри приложения. В проекте WebUI создается соответствующий файл класса с именем NinjectDependencyResolver.cs, в который записывается нужная реализация.

Следующий шаг – создание контекста Entity Framework. Последние версии Entity Framework включают замечательное средство, которое называется сначала код (code-first). Идея состоит в том, что в модели можно определить классы, а затем на их основе сгенерировать базу данных. Это очень удобно для проектов, создаваемых с нуля.

Создается класс контекста, который будет ассоциировать модель с базой данных. Чтобы воспользоваться преимуществом подхода "сначала код", потребуется создать класс, производный от **System.Data.Entity.DbContext**. Этот класс затем автоматически определяет свойство для каждой таблицы базы данных, с которой необходимо работать.

Имя свойства указывает таблицу, а параметр типа результата **DbSet** - тип модели, который инфраструктура Entity Framework должна использовать для представления строк в этой таблице. В рассматриваемом случае именем свойства является Films, а параметром типа - Film, т.е. для представления строк в таблице Films инфраструктура Entity Framework будет применять тип модели Film.

Далее Entity Framework нужно указать, как подключаться к базе данных, и это делается путем добавления в файл Web.config проекта WebUI строки подключения к базе данных с таким же именем, как у класса контекста:

<configuration>

<connectionStrings>

<add name="EFDbContext" connectionString="Data Source=(localdb)\v11.0;Initial Catalog=FilmStore;Integrated Security=True" providerName="System.Data.SqlClient" />

</connectionStrings>

Логика модели и хранилища определяются в проекте Domain, но информация подключения к базе данных помещается в файл Web.config проекта WebUI.

Все, что осталось сделать – создать хранилище EFFilmRepository, реализующий интерфейс IFilmRepository и использует экземпляр EFDbContext для извлечения данных из базы данных посредством Entity Framework. Также в NinjectDependencyResolver добавляется привязка к хранилищу.

Для создания представления интернет-магазина необходимо создать стандартное представление метода List () FilmController, а позже редактировать его. Средствами Visual Studio будет создан не только файл List.cshtml, но также и файлы \_ViewStart.cshtml и Shared/\_Layout.cshtml. В файл \_Layout.cshtml заносится необходимая компоновка, а в файл List.cshtml код, состоящий из HTML-разметки и несколько выражений Razor для отображения полной информации о товарах.

На этом этапе создана структура интернет-магазина и интернет-магазин извлекает сведения о товарах из базы данных. На сайте интернет-магазина добавлено представление товаров, имеющихся в базе данных.

## Разбиение на страницы, стилизация контента и навигация

Следующий этап разработки – создание разбиения контента на стницы, стилизация контента, и реализация навигации для упрощенного доступа пользователей к интересующим их товарам.

На этом этапе были достигнуты следующие результаты:

* Добавлены различные URL для доступа к различными страницам и жанрам фильмов.

|  |  |
| --- | --- |
| URL | Что выводит |
| / | Выводит первую страницу списка товаров всех категорий |
| /Page2 | Выводит указанную страницу (в этом случае страницу 2), отображая товары всех категорий |
| /Драма | Отображает первую страницу элементов указанного жанра (в этом случае фильма жанра "Драма") |
| /Симулятор/Page2 | Отображает заданную страницу (в этом случае страницу 2) элементов указанного жанра (Драма) |

Таблица 2.3.1 Сводка маршрутизации

* Создано и стилизовано меню поиска фильмов по жанрам. Жанры добавляются в меню по при добавлении фильма с этим жанром в базу данных.

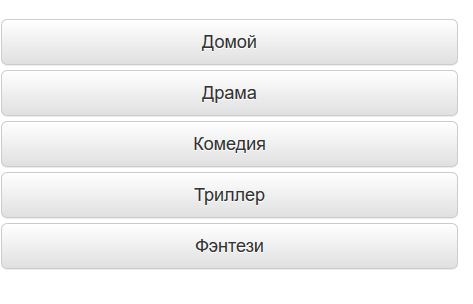


Рисунок 2.3.1 Отображение меню на сайте

## Создание корзины покупок и отправки заказов

Для реализации функционала сайта необходимо создать корзину покупок, в которую пользователь может добавлять понравившиеся ему фильмы, а также дать пользователю возможность оформить заказ.

На данном этапе разработки были достигнуты следующие результаты:

* Стилизовано и оформлено верхнее меню, содержащее название сайта и переход к корзине покупок



Рисунок 2.4.1 Отображение верхнего меню на сайте

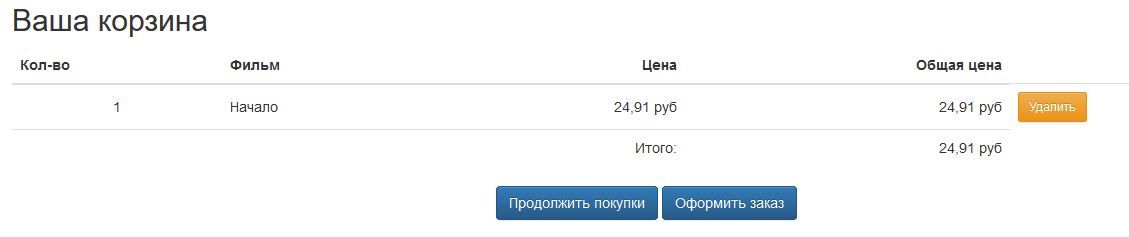
* Создана корзина покупок

Рисунок 2.4.2 Корзина покупок

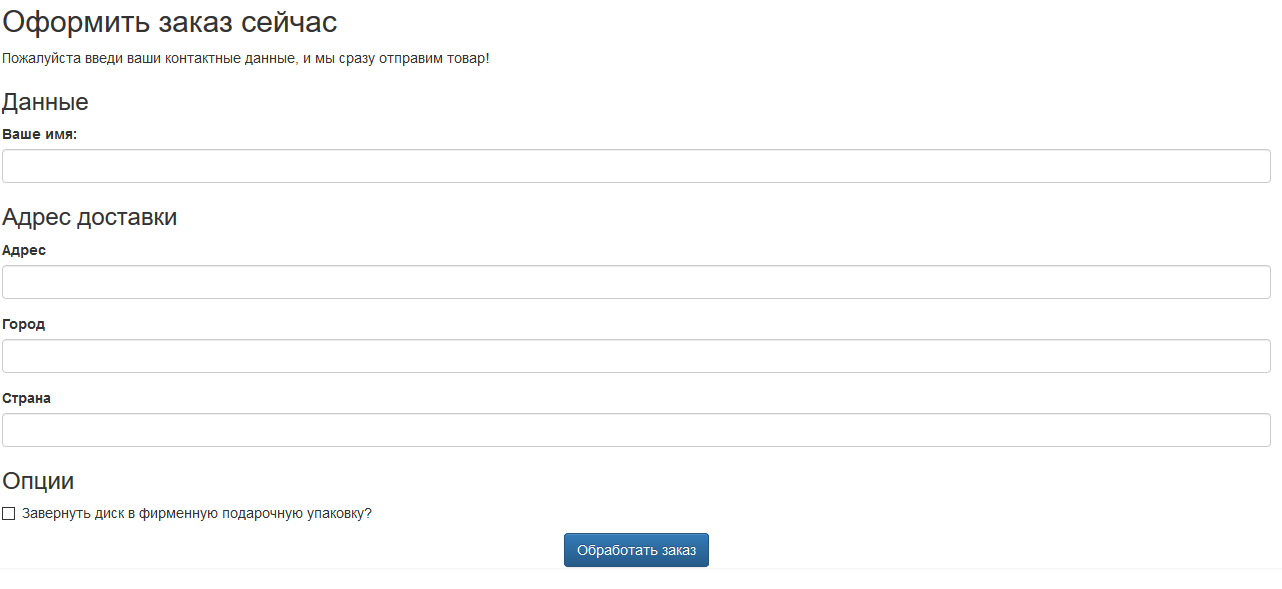
* Создана страница оформления заказа

Рисунок 2.4.3 Оформление заказа

* Доставка сообщения о заказе на почту администратору (в данном случае в файл на компьютер).

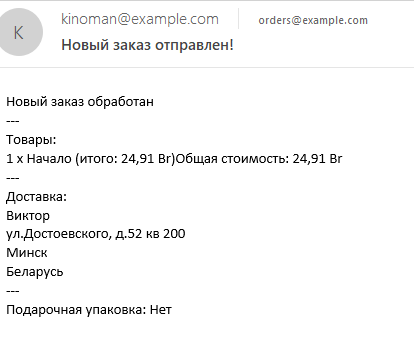


Рисунок 2.4.4 Сообщение о заказе

## 2.5 разработка страниц администрирования

Необходимо предоставить возможность администрации манипулировать списком товаров, а также предотвратить неавторизованный доступ к страницам администрирования.

На данном этапе разработки были достигнуты следующие результаты:

* Создана страница авторизации для доступа к ресурсам администрации.

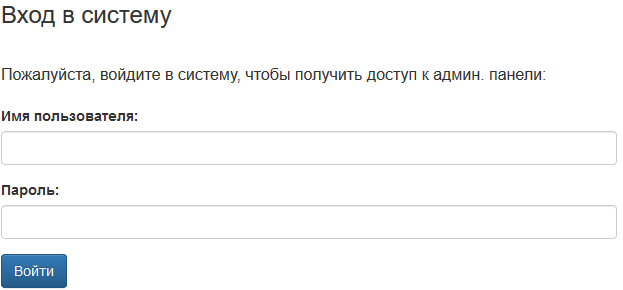


Рисунок 2.5.1 Страница авторизации администратора

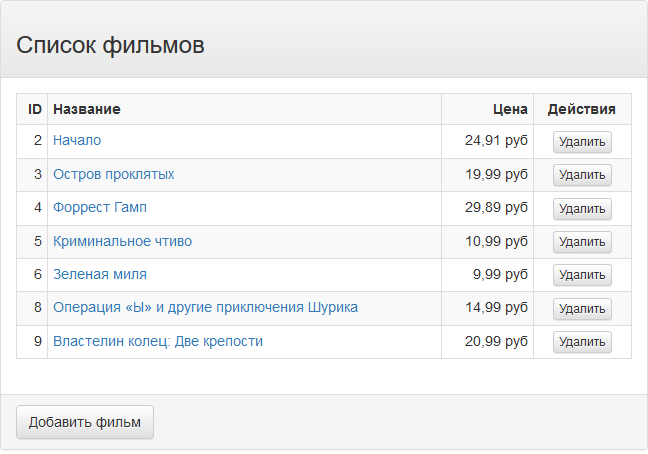
* Разработана страница администрации, на которых администратор может использовать CRUD-средства.

Рисунок 2.5.2 Страница администрирования списка фильмов

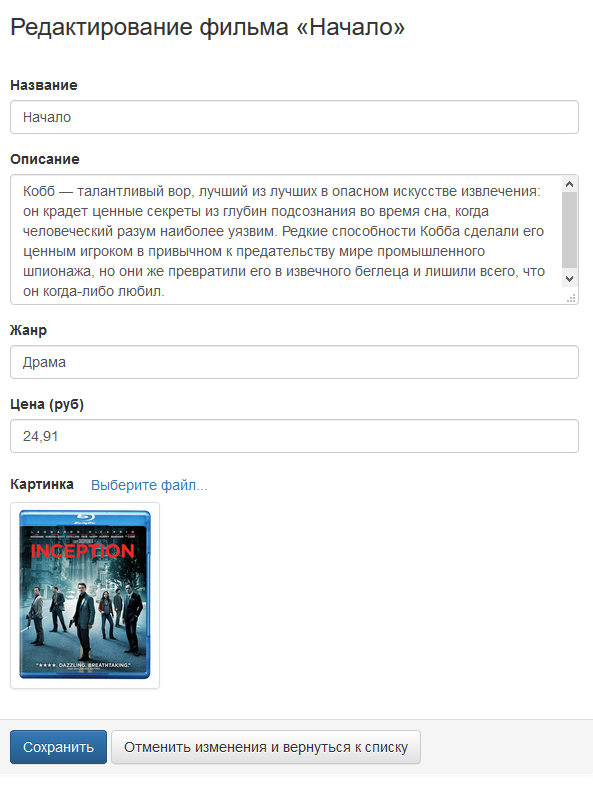


Рисунок 2.5.3 Страница редактирования информации о фильме

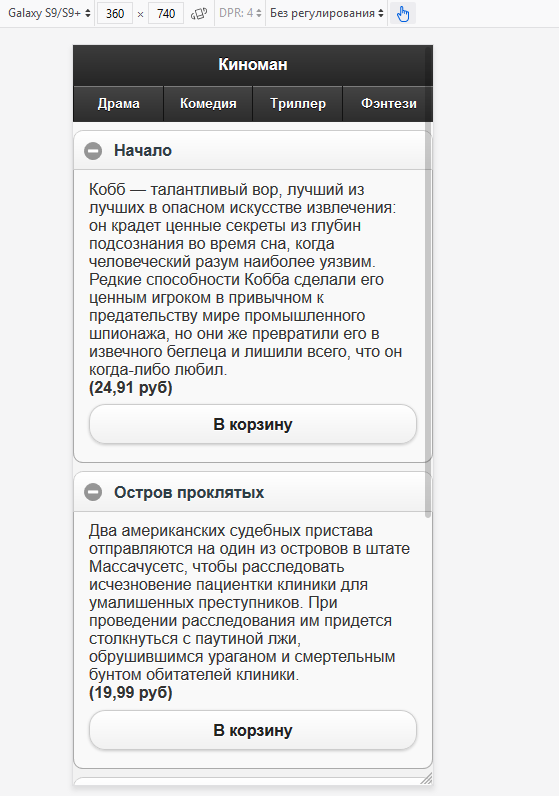
* На этом этапе также была проведена оптимизация дизайна сайта под мобильные устройства.

Рисунок 2.5.4 Вид сайта на мобильном устройстве

По завершению этого этапа разработки сайт готов к развертке: В корзину безошибочно добавляются товары, заказа оформляется с верными данными, также администратор может легко добавлять, изменять и удалять фильмы. Для доступа к администраторской панели необходима авторизация, благодаря чему неавторизованный доступ к данным о товарах невозможен.

## 3. Описание работы сайта

## 3.1 Демонстративный пример работы сайта

Так как сайт не развернут в интернете, необходимо запустить локальный сервер. После запуска локального сервера и ввода веб-адреса сайта на локальном сервере, открывается главная страница сайта:

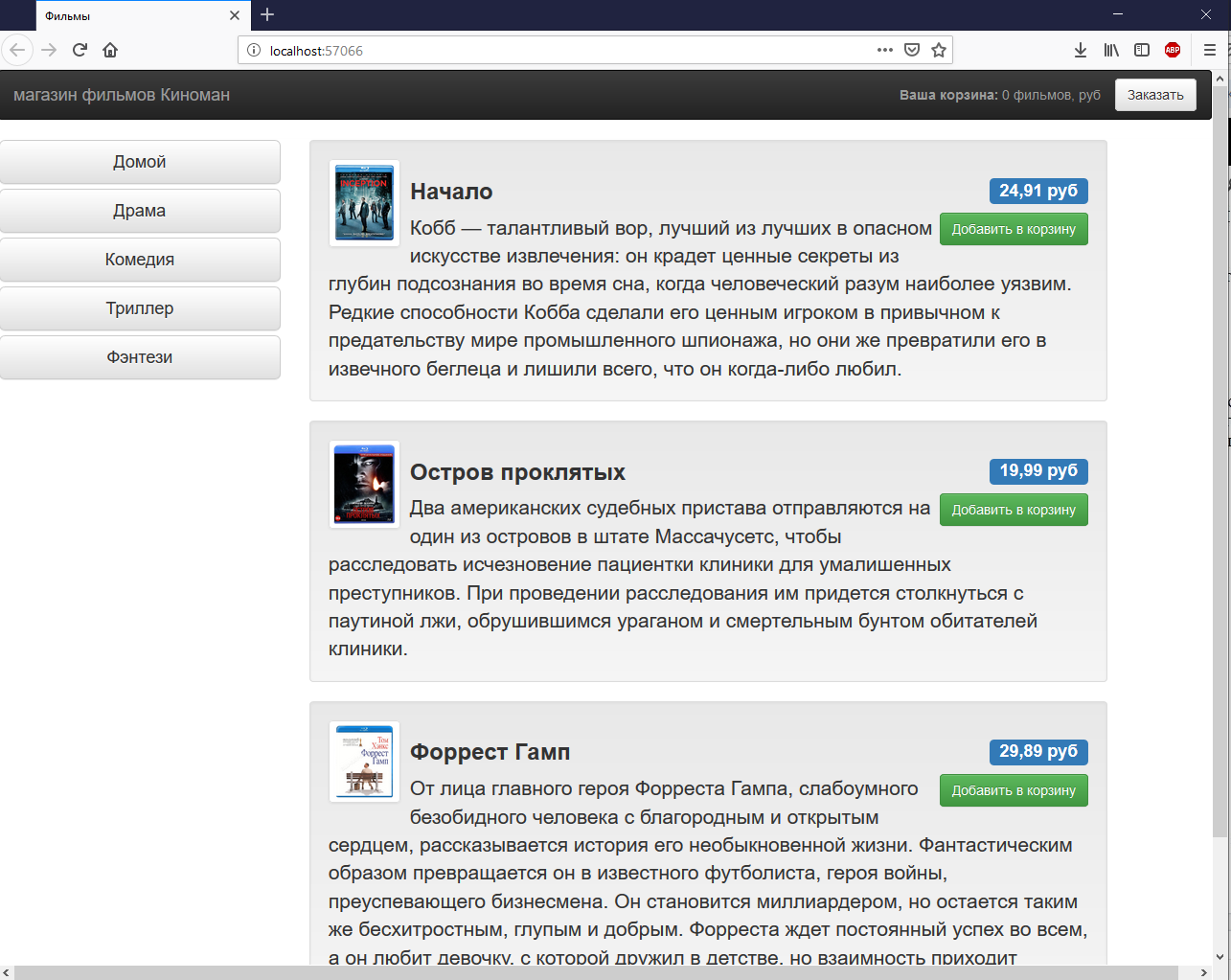


Рисунок 3.1.1 Вид главной страницы сайта

На главной странице сайта размещен список товаров без какой-либо фильтрации по жанрам. При нажатии на любой из жанров со списка слева, мы попадем на страницу только с теми фильмами, чей жанр совпадает с указанным.

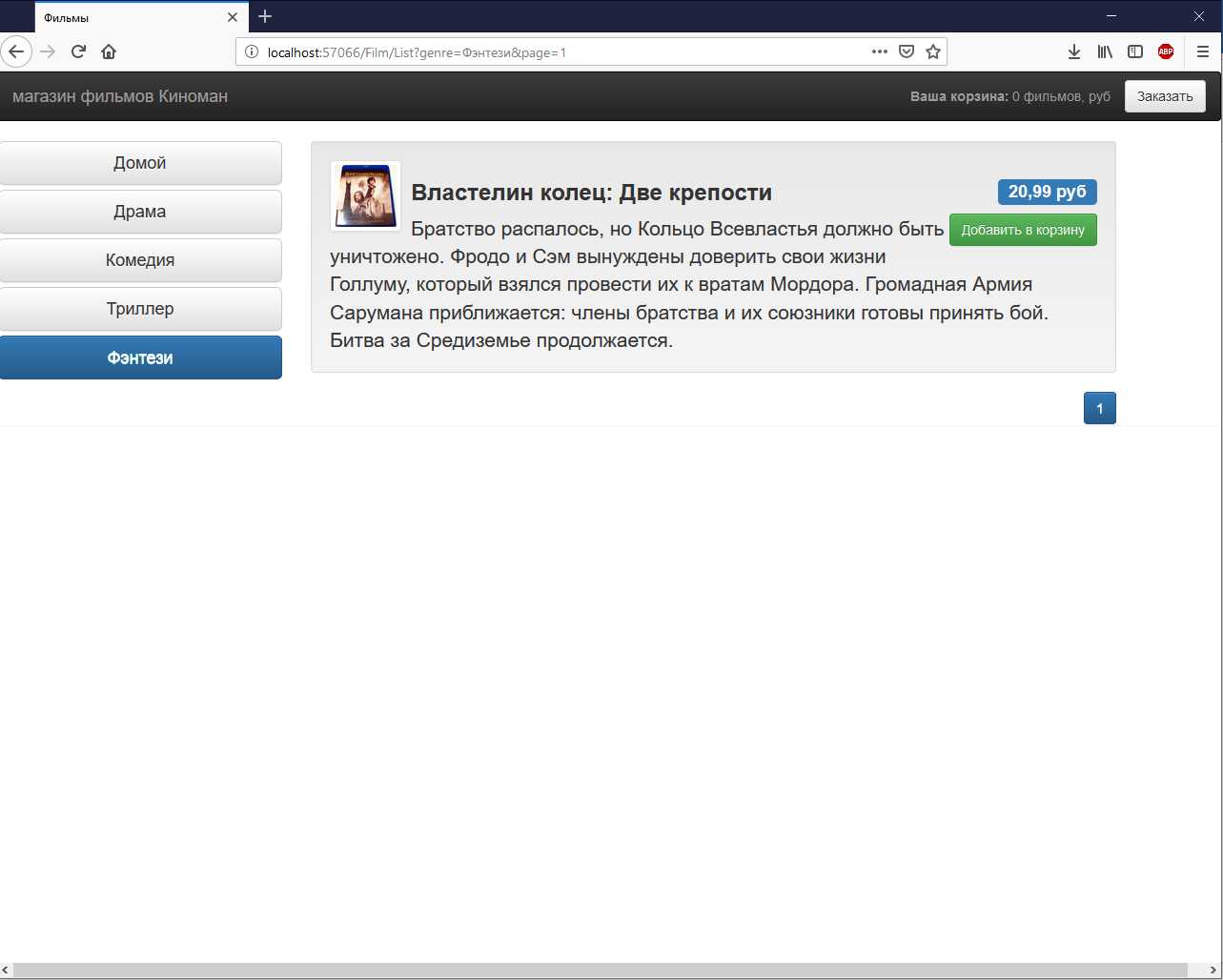


Рисунок 3.1.2 Вид страницы с фильмами жанра Фэнтези

Допустим, пользователь хочет купить фильм Властелин колец: Две крепости. Добавим его в корзину. Нас отправит на страницу с текущей корзиной, где уже находится только что нами выбранный фильм.

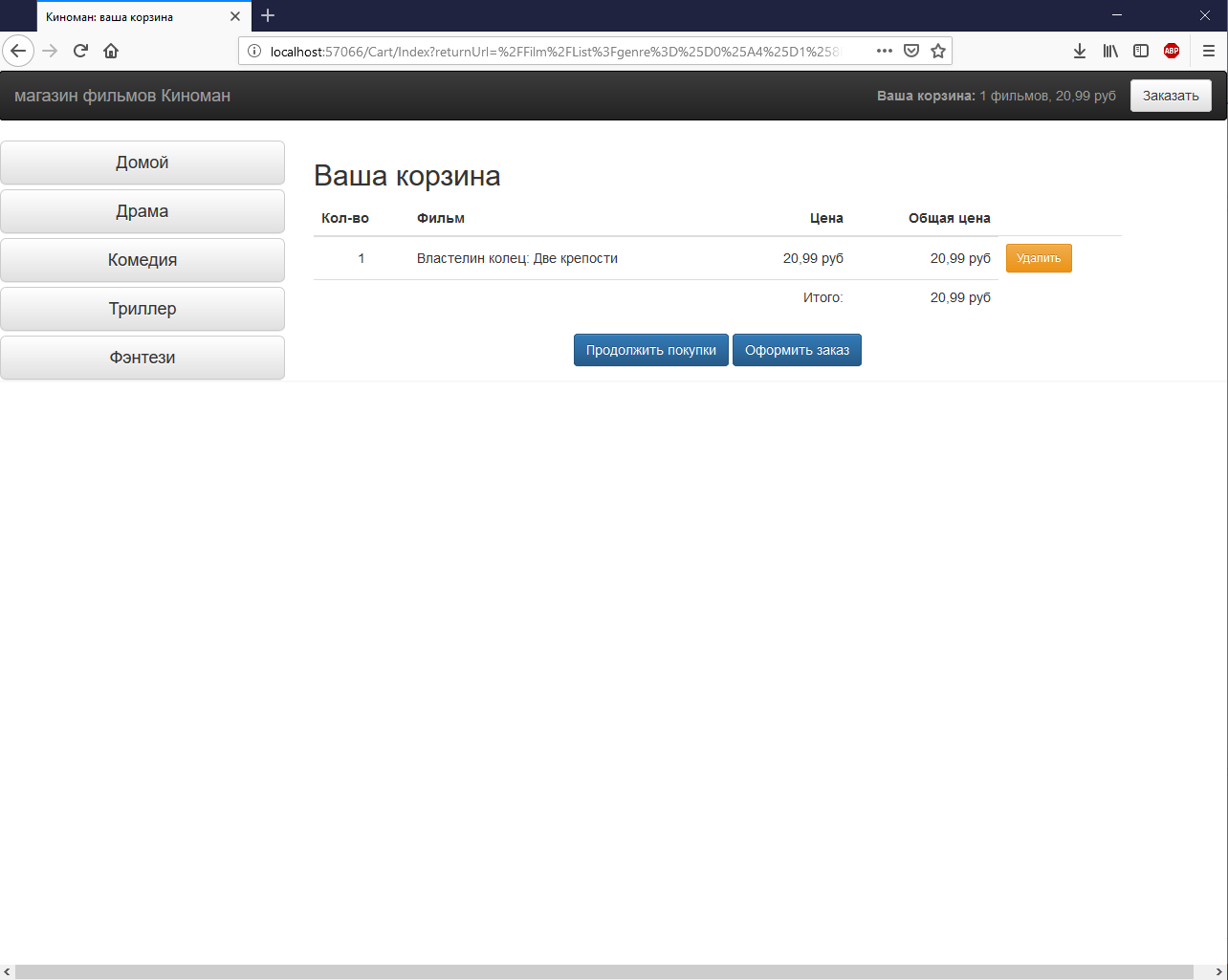


Рисунок 3.1.3. Корзина после выбора 1 товара

При желании пользователь хочет выбрать что-нибудь еще. Нажмем кнопку продолжить покупки и нас переправит обратно на страницу с фильмами жанра фэнтези, но можно заметить, что в верхнем правом углу виден статус корзины, и на данный момент там находится один фильм

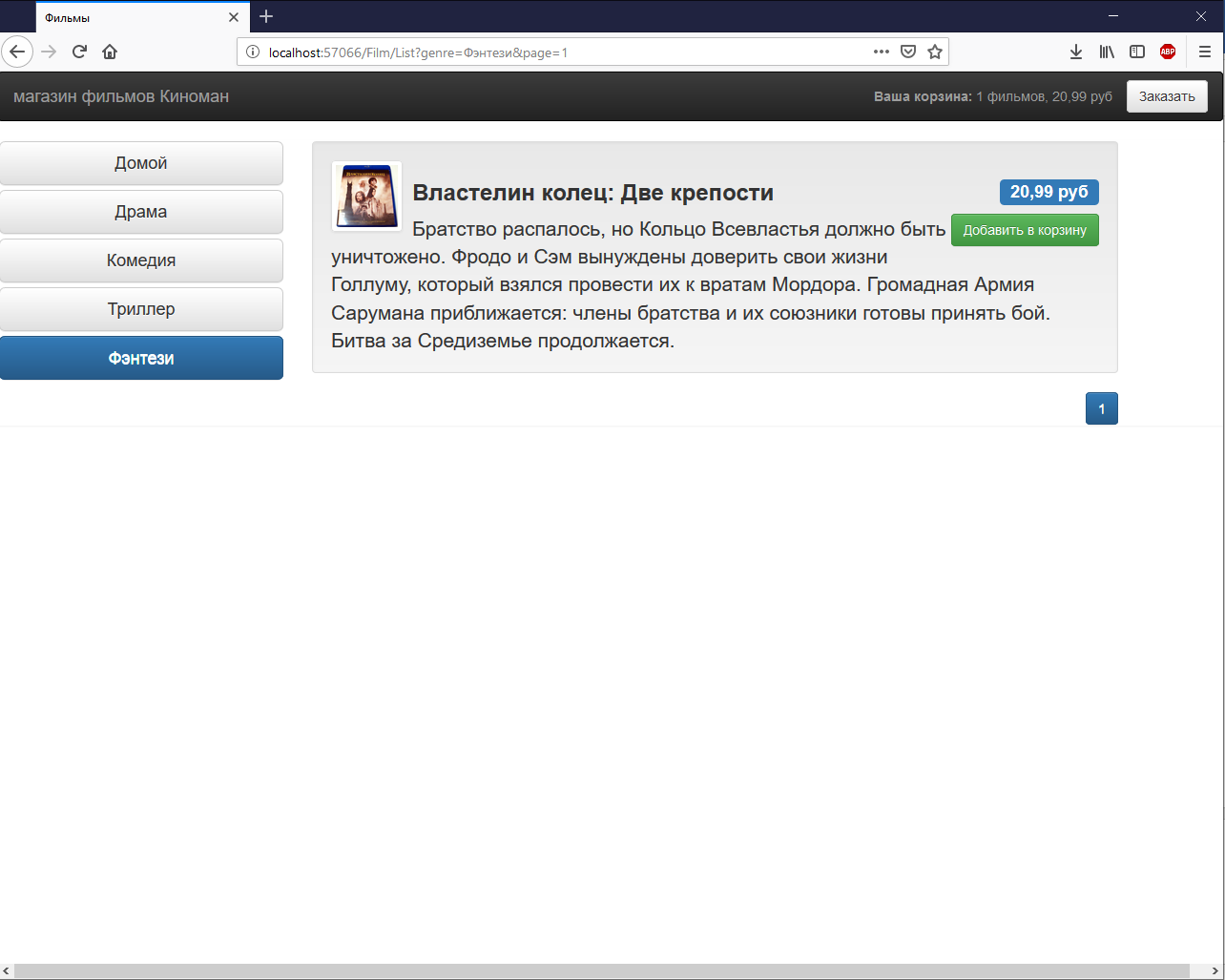


Рисунок 3.1.4 Вид страницы с фильмами жанра фэнтези и обновленным статусом корзины

Допустим пользователь захотел также купить что-нибудь жанра комедии. Перейдем на страницу с комедийными фильмами.

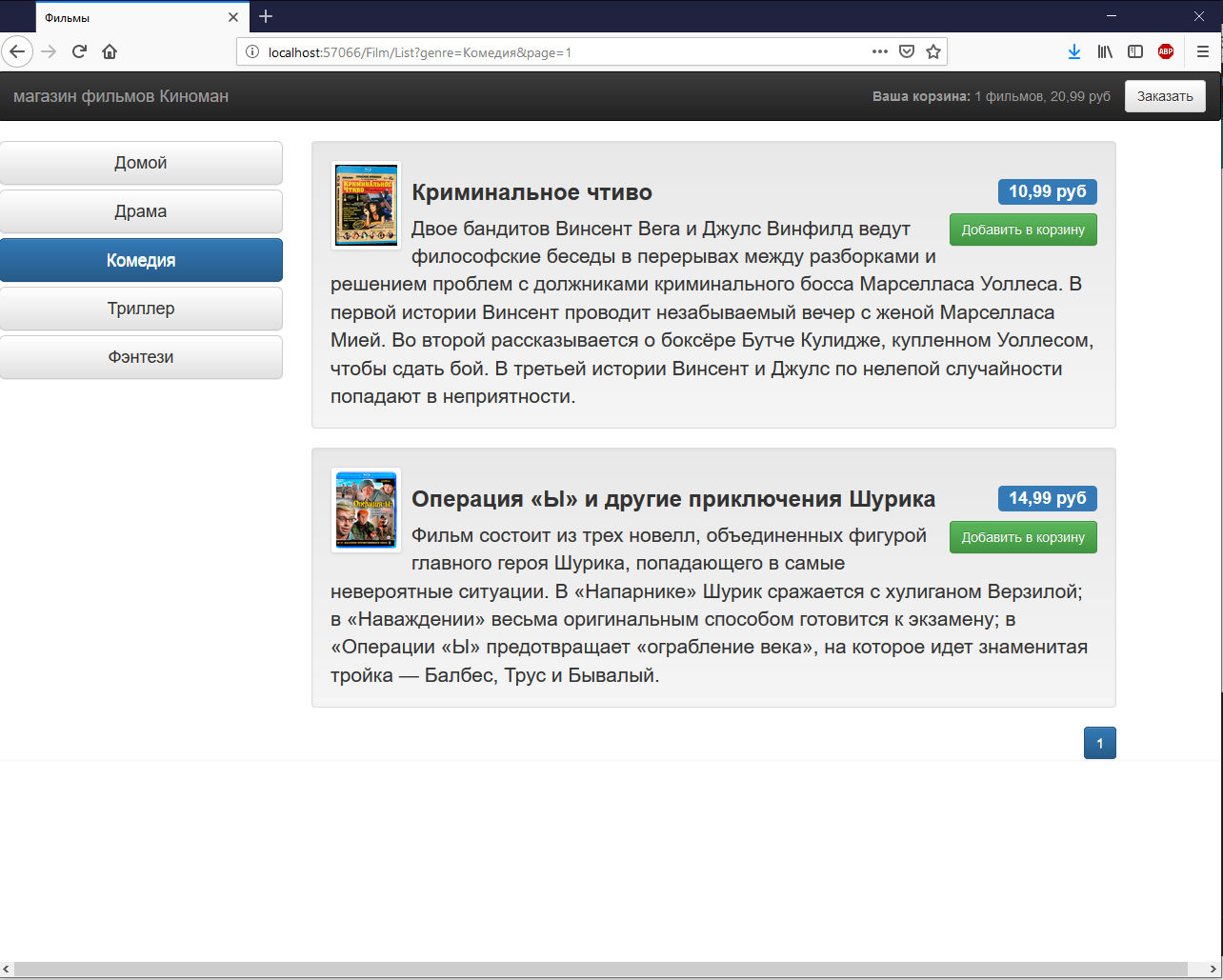


Рисунок 3.1.5 Вид страницы с фильмами жанра комедия

Добавим фильм Операция “Ы” и другие приключения Шурика в корзину. Нас снова отправит на страницу с нашей корзиной и обновленной общею стоимостью товаров, находящихся в ней.

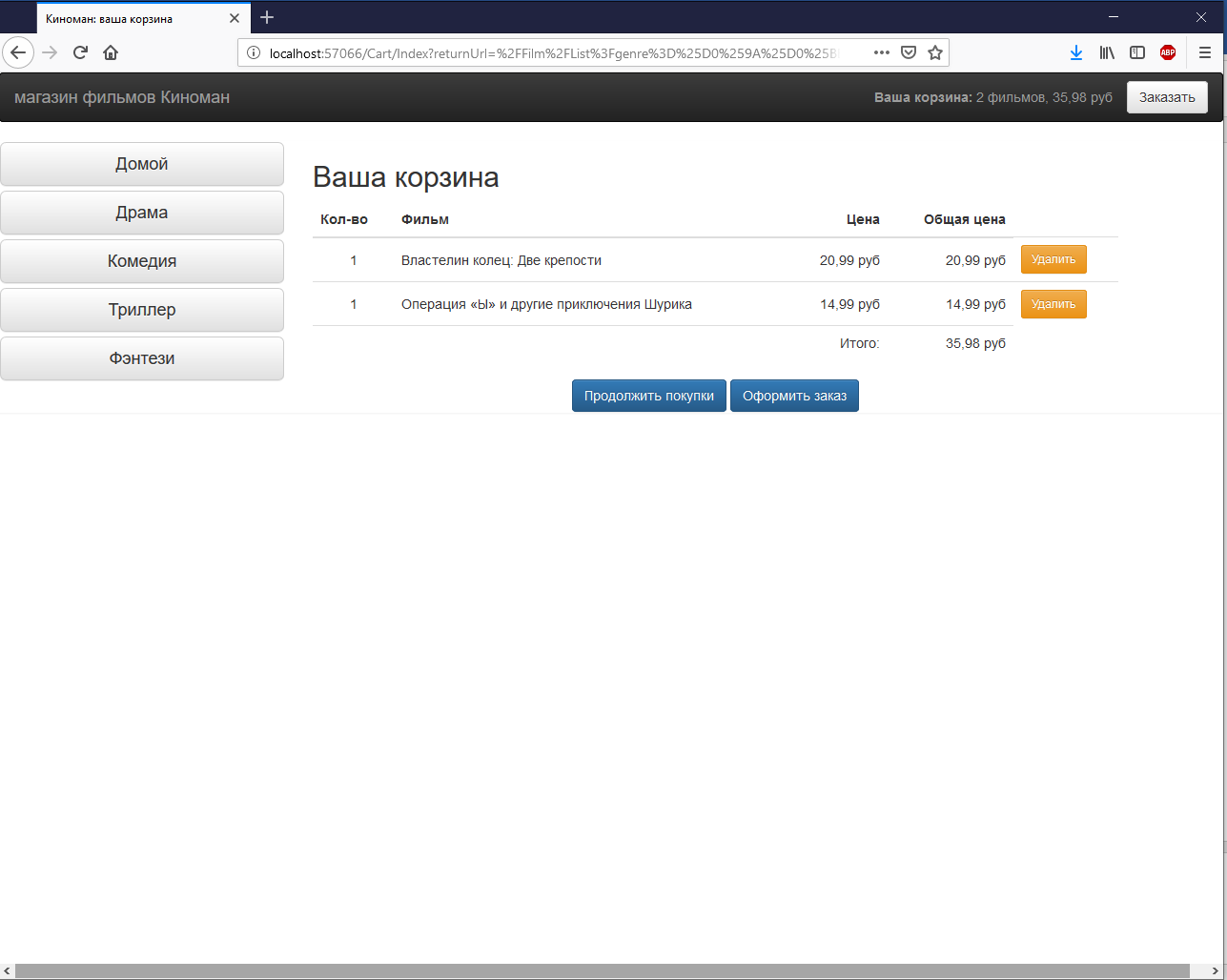


Рисунок 3.1.6 Корзина после добавления двух фильмов в нее.

Допустим, что теперь пользователь удовлетворен двумя выбранными фильмами и хочет сделать заказ. Нажмем кнопку оформить заказ и переправимся на страницу оформления заказа.

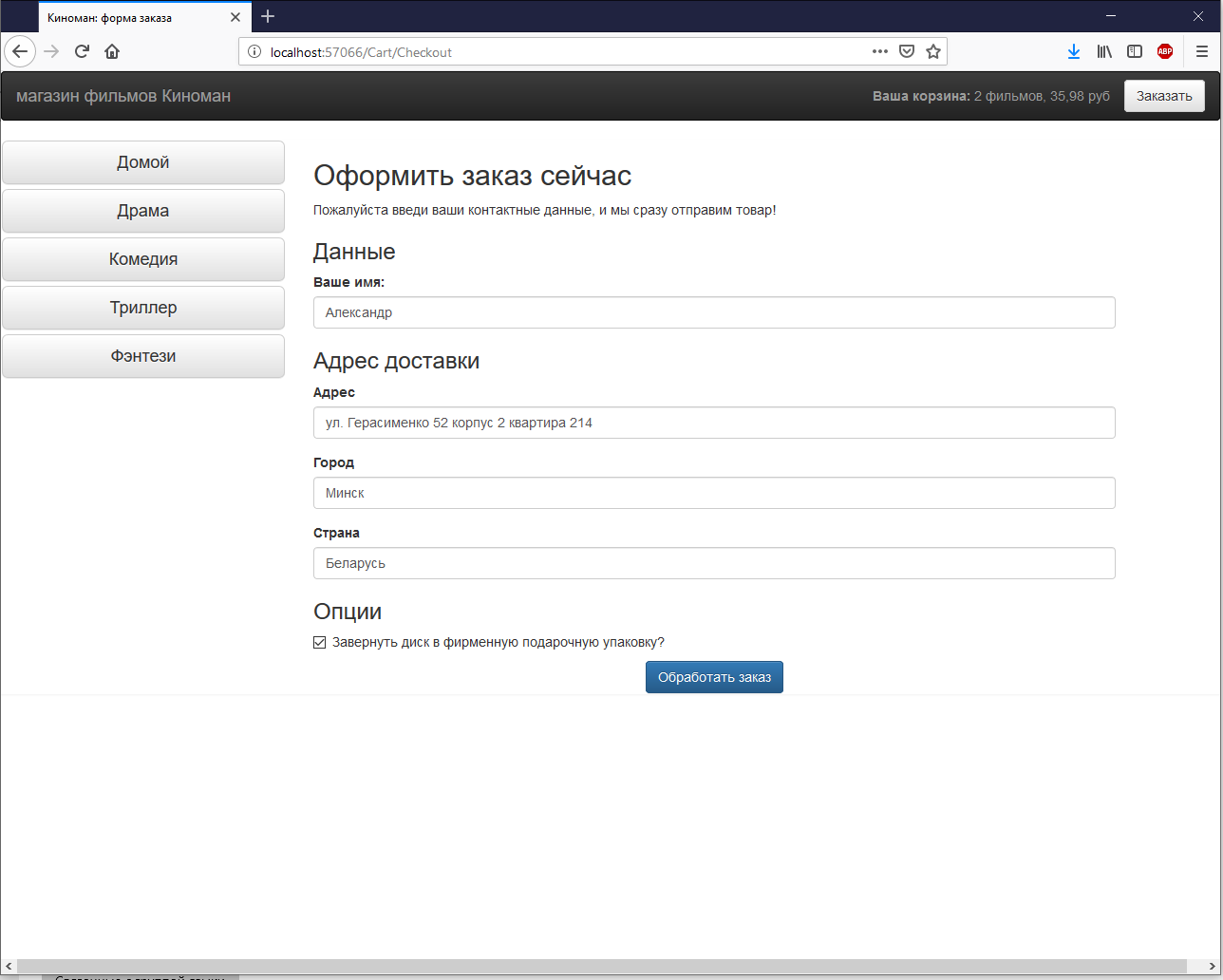


Рисунок 3.1.7 Страница оформления заказа

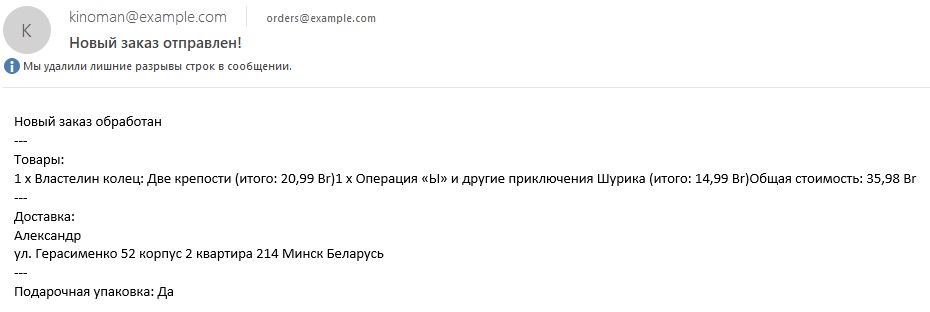
Пользователь вводит свои даные и его переправляет на страницу с оповещением о успешном заказе и сообщении о том, что заказ скоро будет отправлен в службу доставку. Владельцу магазина же на почту (в нашем случае, в файл) приходит сообщение о заказе:

Рисунок 3.1.8 Оповещение о заказе

После вышеописанных действиях можно сказать что весь функционал, связанный с обычными посетителями магазинами, работает правильно и исправно.

## 3.2 Демонстративный пример работы страниц администратора

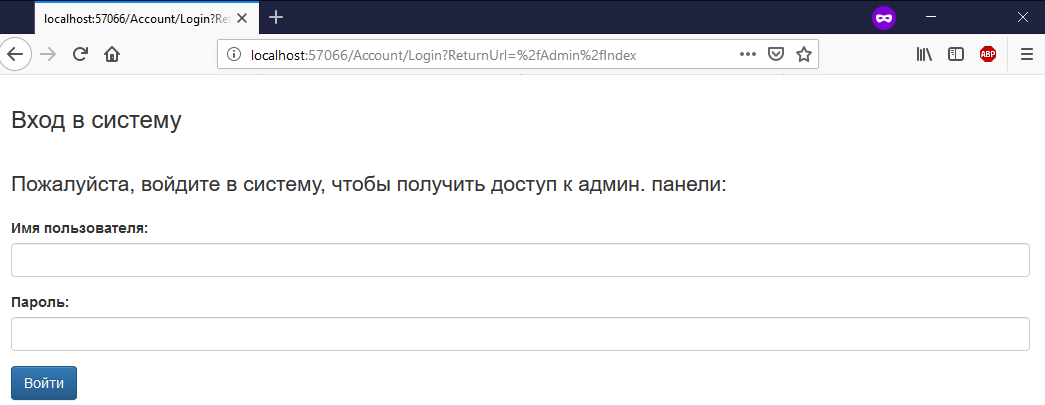
Для доступа к администраторской части сайта необходимо в строке браузера после имени сайта вписать /Admin/Index. Пользователя переправит на страницу авторизации. Если же пользователь уже был авторизован с этого браузера, повторная авторизация не понадобится.

Рисунок 3.2.1 Страница авторизации

После корректного ввода данных администратора переправит на страницу администрирования.

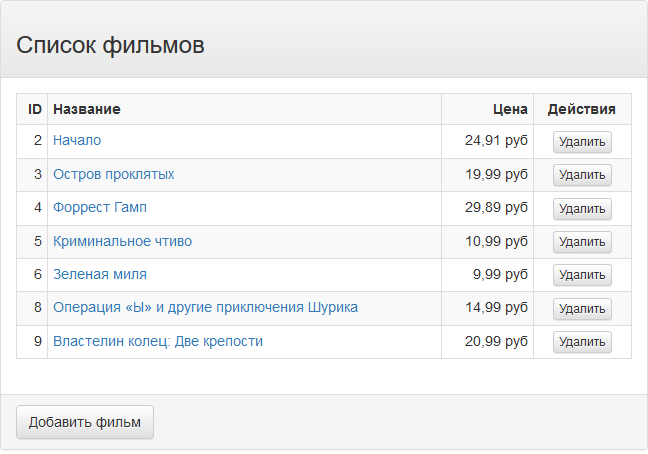


Рисунок 3.2.2 Страница администрирования

Предположим, что администратор захотел изменить стоимость фильма “Начало”. Нажмем на название фильма.

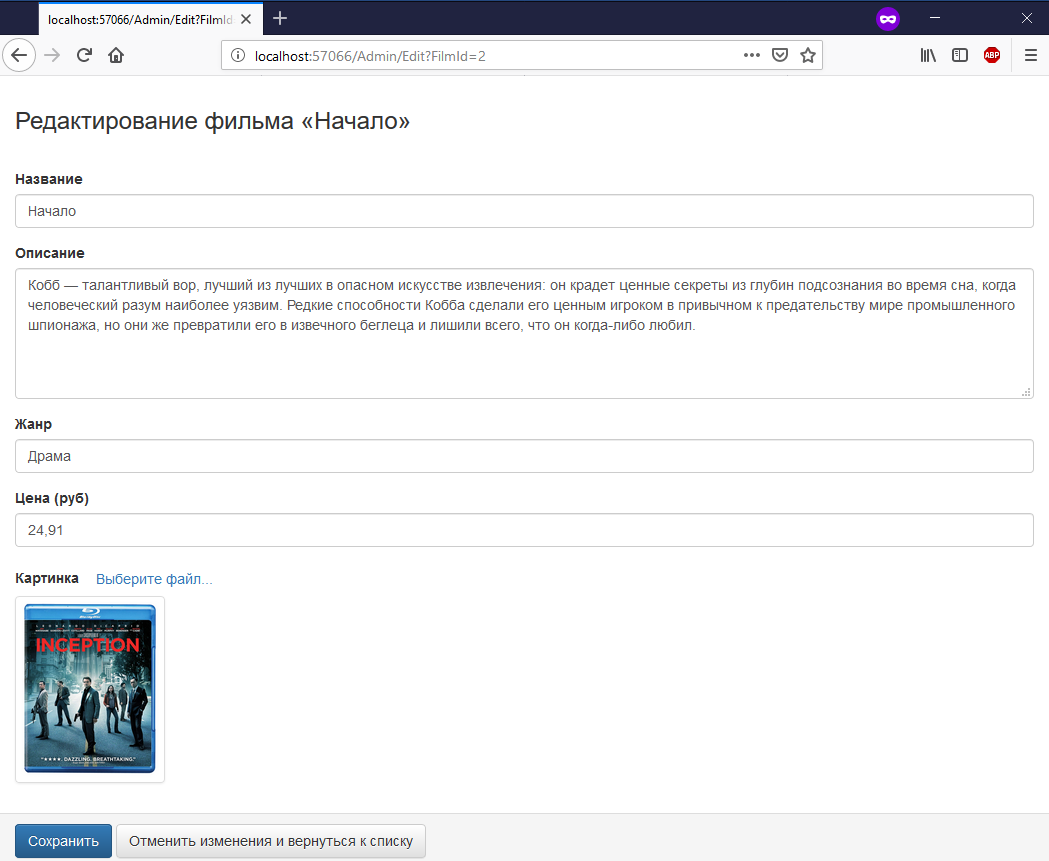
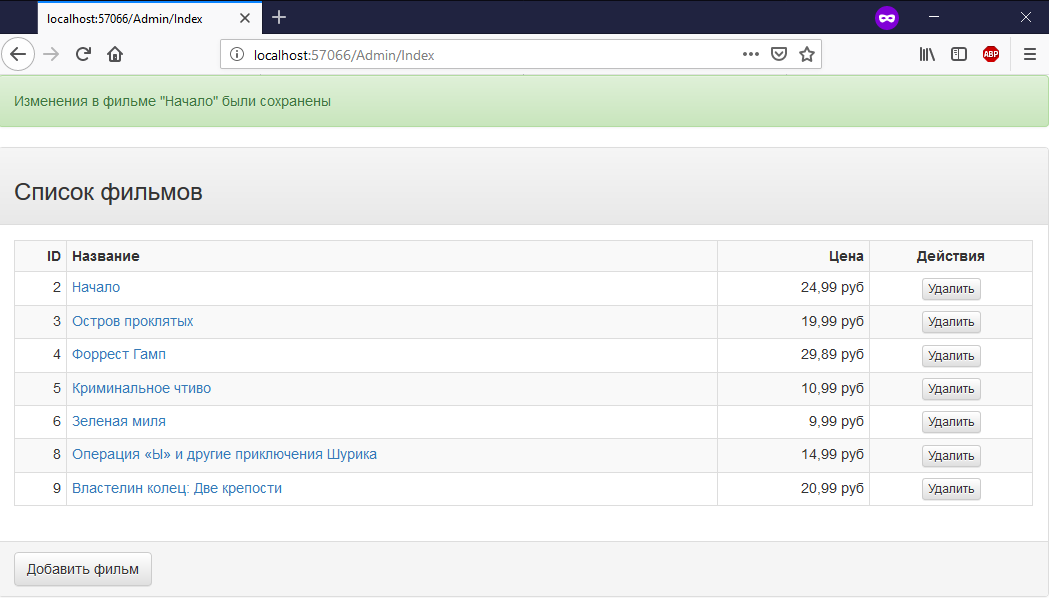


Рисунок 3.2.3 Страница изменения информации о фильме “Начало”

Введем новую цену - 24,99 рублей и нажмем сохранить. Нас вернет на страницу администрирования. Легко заметить, что цена фильма изменилась.

Рисунок 3.2.3 Страница администрирования после редактирования информации о фильме.

Функции добавить работает аналогично, отображается страница редактирования еще несуществующего фильма. Функция удалить удаляет фильм из базы данных. Также стоит заметить, что только со страницы администрирования подразумевается корректное добавления изображения к фильму, остальные поля можно добавлять при обращении к серверу SQL-запросами, как изначально и были заданы значения полей фильмов при разработке проекта.

## Заключение

В ходе выполнения курсовой работы был получен полнофункциональный интернет-магазин, полностью готовый к применению. Данный интернет-магазин создан с целью предоставления возможности посетителям приобресит Blu-ray изданий различных филмьов.

При разработке интернет были проанализированы современные технологии ASP-NET MVC, позволяющие разработчику пользоваться множеством различных функций.

В качестве дальнейшего совершенствования интернет-магазина представляется возможным возможность регистрации пользователей для дальнейшего осуществления скидочных акций и, возможно, розыгрышей подарков для самых частых покупателей. Также приоритетной целью является улучшение интерфейса сайта.

Тестирование сайта производилось в браузерах: Google Chrome, Mozilla Firefox.

## Список использованных источников:

1. “Pro ASP.NET MVC 5” автор – Адам Фримэн
2. “Руководство по ASP.NET MVC 5” [Электронный ресурс] – режим доступа: <https://metanit.com/sharp/mvc5/>
3. “Полное руководство по языку программирования C#” [Электронный ресурс] -режим доступа: https://metanit.com/sharp/tutorial/