УДК 004.6

Поварницын Е.Н., студент

4 курс, факультет «Информационные системы и технологии»

Северный Арктический Федеральный Университет

Россия, г. Архангельск

Povarnitsyn E. N., student

4rd year, faculty of Information systems and technologies»

Northern Arctic Federal University

Russia, Arkhangelsk

Использование паттернов при решении задач профессиональной деятельности

Using patterns in solving professional tasks

Аннотация:

Статья посвящается использованию паттернов MVC и прототип. В ней детально рассматриваются разработка сайта.

Ключевые слова: MVC, прототип, C#, шаблон, паттерн, сайт, веб-ресурс.

Annotation:

This article focuses on the use of MVC and prototype patterns. It discusses in detail the development of the site.

Keyword:

MVC, prototype, C#, template, pattern, site, web resource.

1 ПАТТЕРНЫ

При реализации работы следует использовать паттерны. Для данного проекта будут использованы такие паттерны как «Прототип» и «MVC». Для начала разберём что такое паттерны, перед тем как углубляется в эту тему.

Шаблон проектирования или паттерн в разработке программного обеспечения — повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

Обычно шаблон не является законченным образцом, который может быть прямо преобразован в код; это лишь пример решения задачи, который можно использовать в различных ситуациях. Объектно-ориентированные шаблоны показывают отношения и взаимодействия между классами или объектами, без определения того, какие конечные классы или объекты приложения будут использоваться.

«Низкоуровневые» шаблоны, учитывающие специфику конкретного языка программирования, называются идиомами. Это хорошие решения проектирования, характерные для конкретного языка или программной платформы, и потому не универсальные.

На наивысшем уровне существуют архитектурные шаблоны, они охватывают собой архитектуру всей программной системы.

Алгоритмы по своей сути также являются шаблонами, но не проектирования, а вычисления, так как решают вычислительные задачи.

Теперь разберём паттерны «Прототип» и «MVC», с которыми и будет работать веб-ресурс.

Разберём сначала прототип, так как он менее сложный, чем «MVC». Данный паттерн чем-то напоминает «фабрику», он также служит для создания объектов, однако с немного другим подходом. Представьте, что у вас есть пустой пакет (к примеру, из-под сока), а вам нужен полный с апельсиновым соком. Вы «говорите» пакету «Хочу пакет апельсинового

сока», он в свою очередь создает свою копию и заполняет ее соком, который вы попросили. Немного «сказочный пример», но в программировании часто так и бывает. В данном случае пустой пакет и является «прототипом», и в зависимости от того, что вам требуется, он создает на своей основе требуемые вами объекты (пакеты сока).

Клонирование не обязательно должно производиться на самом «пакете», это может быть и какой-то другой «объект», главное лишь что данный «прототип» позволяет получать его экземпляры.

MVC, иначе Model-View-Controller, переводиться как «Модель-Представление-Контроллер» - это схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо

Модель предоставляет данные и методы работы с ними: запросы в базу данных, проверка на корректность. Модель не зависит от представления (не знает как данные визуализировать) и контроллера (не имеет точек взаимодействия с пользователем), просто предоставляя доступ к данным и управлению ими. Модель строится таким образом, чтобы отвечать на запросы, изменяя своё состояние, при этом может быть встроено уведомление «наблюдателей». Также модель, за счёт независимости от визуального представления, может иметь несколько различных представлений для одной «модели».

Представление отвечает за получение необходимых данных из модели и отправляет их пользователю. Представление не обрабатывает введённые данные пользователя.

Контроллер обеспечивает «связь» между пользователем и системой. Контролирует и направляет данные от пользователя к системе и наоборот. Использует модель и представление для реализации необходимого действия.

Поскольку MVC не имеет строгой реализации, то реализован он может быть по-разному. Нет общепринятого определения, где должна располагаться бизнес-логика. Она может находиться как в контроллере, так и в модели. В последнем случае, модель будет содержать все бизнес-объекты со всеми данными и функциями.

Некоторые фреймворки жестко задают, где должна располагаться бизнес-логика, другие не имеют таких правил.

Также не указано, где должна находиться проверка введённых пользователем данных. Простая валидация может встречаться даже в представлении, но чаще они встречаются в контроллере или модели.

Интернационализация и форматирование данных также не имеет четких указаний по расположению.

2 АНАЛИЗ ПРЕДМЕТНОЙ ОБАЛСТИ

Для реализации проекта была выбрана тема база фильмов. Поэтому следует разобрать предметную область кинофильмов.

Существует всего несколько основных видов кино. Их можно поделить на три основных вида: документальное кино, художественное и сериалы. Документальное кино - самый первый вид кинематографа. Документальные фильмы изображают исключительно реальные события и реальных личностей. Если фильм показывает реконструкцию каких-либо событий, он будет художественно-документальным.

Один из подвидов документального кино - образовательные фильмы. Они используются в образовательных целях и очень широкое применение нашли в современной системе образования. Очень часто образовательные фильмы показывают в школах, так как известно, что материал, преподнесенный в виде фильма, усваивается лучше из-за своей наглядности.

Также есть место быть и сериалом. Сериал – фильм или телефильм, состоящий из некоторого - обычно большого - числа серий и имеющий несколько сюжетных линий («потенциальная бесконечность» в отличие от многосерийных телефильмов вообще у которых присутствует «финальная точка»), особая циклическая форма телевизионного искусства. Наиболее известным форматом сериалов являются мыльные оперы, начавшие своё существование в 1930-х годах на американском радио.

Отдельно от документального стоит художественное кино. Главное отличие таких фильмов от документальных - в художественных играют актеры, а происходящие события могут быть полностью вымышленными.

На сегодняшний день существует множество различных жанров кино. Мы рассмотрим несколько основных, наиболее распространенных жанров.

Комедия. Жанр комедийного кино был популярен всегда, ведь все любят от души посмеяться. В таких фильмах обыгрываются различные

комические ситуации, часто совершенно неожиданные и непредсказуемые, что добавляет им эффектности.

Боевик. В данном жанре кино обыгрываются различные сцены насилия, драк, погонь, перестрелок и так далее. Довольно часто этот жанр совмещают с фантастикой или фэнтези.

Фантастика и фэнтези. В фильмах данного жанра изображаются полностью вымышленные события и герои. Стоит помнить, что в фантастических фильмах главный акцент делается на различные технологии и технологический прогресс. Фантазийные фильмы чаще напоминают сказку. В мире фэнтези привычную науку заменяют различные виды магии и колдовства.

Детективы. Данный жанр показывает определенные преступления и пути их раскрытия. Часто такие фильмы имеют сложный и запутанный сюжет, который заставляет зрителя проникнуться процессом поиска преступника.

Мелодрама. В данном жанре раскрывается духовная часть жизни героев, а основное внимание уделено их переживанием и эмоциям.

Триллер. Данный жанр кино не имеет четких границ и элементы триллера могут присутствовать в любом жанре. Суть жанра в том, чтобы вызвать у зрителя определенные переживания, волнение, страх.

Ужасы. Название данного жанра говорит само за себя. В данных фильмах показывают события и героев, которые должны откровенно пугать зрителя.

Популярность жанров кино можно посмотреть ниже в соответствии с рисунком 1.

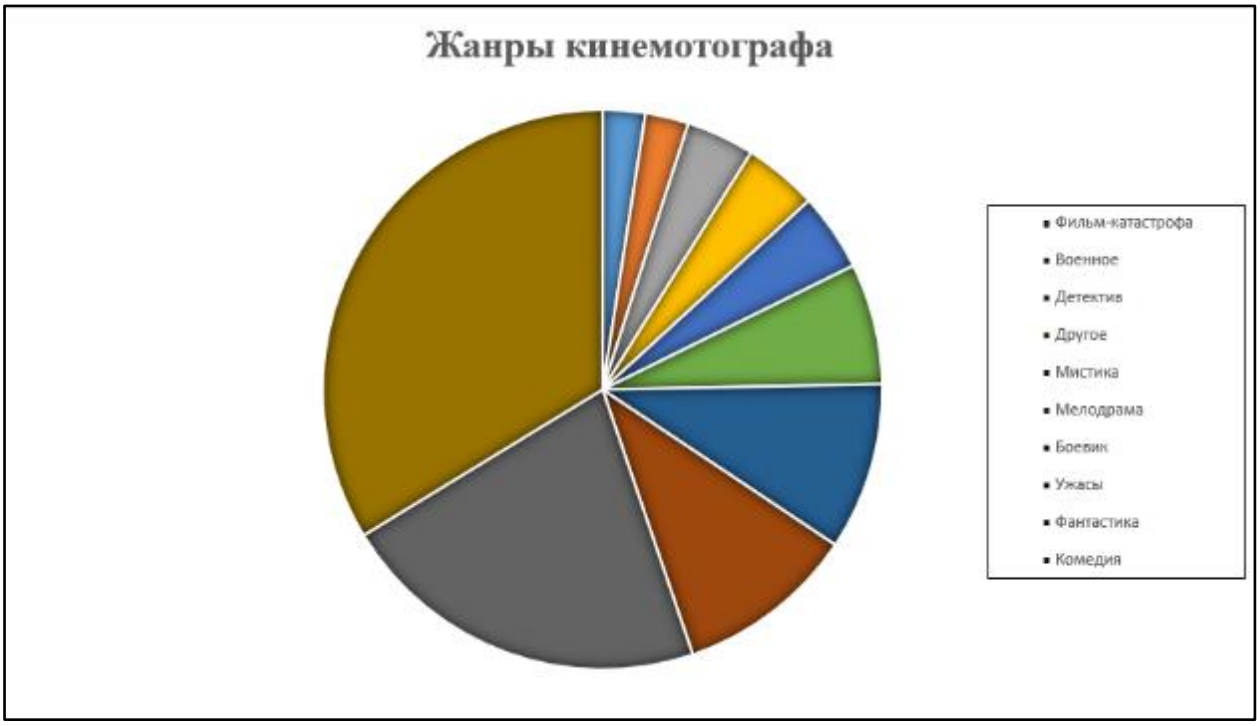


Рисунок 1 – Процентное соотношение популярности жанров кино

Кино - широко распространенное сокращенное название киноискусства, кинематографии или кинотеатра, часть сложных слов, указывающая на связь с кинематографией (например «киноискусство»).

Блок схема с тем, что куда будет идти будет выглядеть в соответствии с рисунком 1.1.

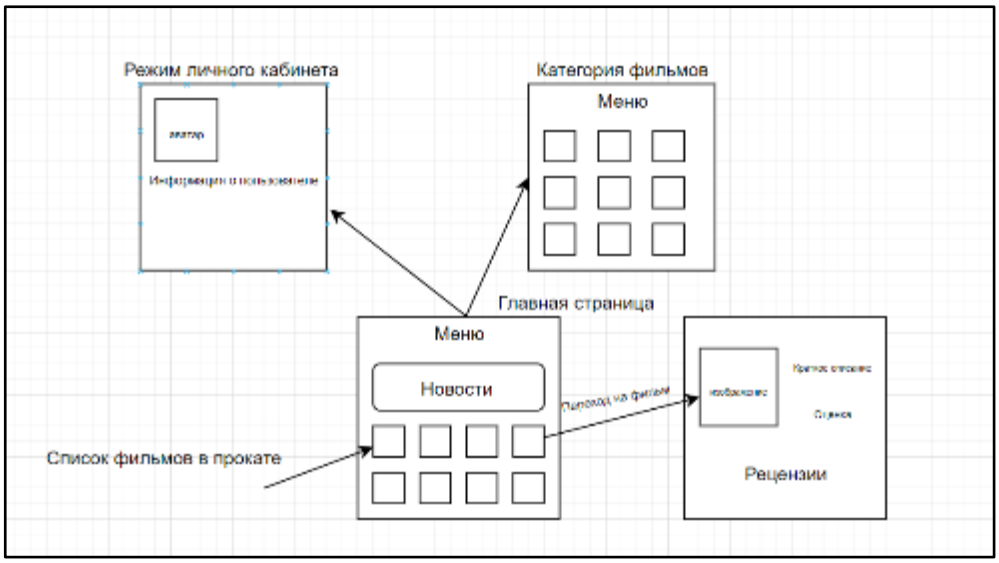


Рисунок 1.1 – Блок схема

3 ПРОЕКТНАЯ ЧАСТЬ

Существует много способов реализации веб-ресурса. Наиболее популярные способы разработки — это такие платформы как Django, Flask, WordPress, Spring.

Django – свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Проект поддерживается организацией Django Software Foundation. Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других. Один из основных принципов фреймворка — DRY.

Flask – фреймворк для создания веб-приложений на языке программирования Python, использующий набор инструментов Werkzeug, а также шаблонизатор Jinja2. Относится к категории так называемых микрофреймворков – минималистичных каркасов веб-приложений, сознательно предоставляющих лишь самые базовые возможности.

WordPress — система управления содержимым сайта с открытым исходным кодом; написана на PHP; сервер базы данных — MySQL; выпущена под лицензией GNU GPL версии 2. Сфера применения – от блогов до достаточно сложных новостных ресурсов. Встроенная система «тем» и «плагинов» вместе с удачной архитектурой позволяет конструировать проекты широкой функциональной сложности.

Spring Framework – универсальный фреймворк с открытым исходным кодом для Java-платформы. Также существует форк для платформы .NET Framework, названный Spring.NET. Несмотря на то, что Spring не обеспечивал какую-либо конкретную модель программирования, он стал широко распространённым в Java-сообществе главным образом как альтернатива и замена модели Enterprise JavaBeans. Spring предоставляет большую свободу Java-разработчикам в проектировании; кроме того, он

предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений корпоративного масштаба.

Но данные платформы не подходят для реализации, ибо некоторые из них предназначены для других разработок, к примеру WordPress стоит использовать, если программист создаёт web магазин.

Проектирование системы будет выполнено с помощью ASP NET Core 3.0 в программе Microsoft Visual Studio. Будут задействованы такие языки программирования как C# и JavaScript. Так же был использован язык разметки HTML и таблица стилей CSS.

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.

ASP.NET Core – свободно-распространяемый кроссплатформенный фреймворк для создания веб-приложений с открытым исходным кодом. Данная платформа разрабатывается компанией Майкрософт совместно с сообществом и имеет большую производительность по сравнению с ASP.NET. Имеет модульную структуру и совместима с такими операционными системами как Windows, Linux и macOS.

Microsoft SQL Server – система управления реляционными базами данных, разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку

запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

C# – объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота, как язык разработки приложений для платформы Microsoft .NET Framework.

JavaScript – мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили.

HTML – «язык гипертекстовой разметки» – стандартизированный язык разметки документов во Всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

CSS – каскадные таблицы стилей – формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

Я выбрал данные средства разработки так как ASP NET Core платформа, которая всегда развивается и будет поддерживаться долгое время, так как её создатели Microsoft, поэтому делая веб-ресурс здесь, можно не переживать что в будущем нужно будет переносить всё на другую платформу. Разрабатывать я всё буду на Microsoft Visual Studio так как это так же разработка Microsoft и не будет никаких проблем с привязкой файлов. Язык C# используется в ASP Net Core и других альтернатив на данной платформе нет. JS язык, который будет использован в Front end разработке, так как с его помощью можно сделать многие вещи, которые на CSS невозможны. Ну и HTML, и CSS это визуальная основа сайта.

Составим UML диаграмму для лучшего представления того, как будет выглядеть логика сайта. UML диаграмма будет выглядеть в соответствии с рисунком 2.

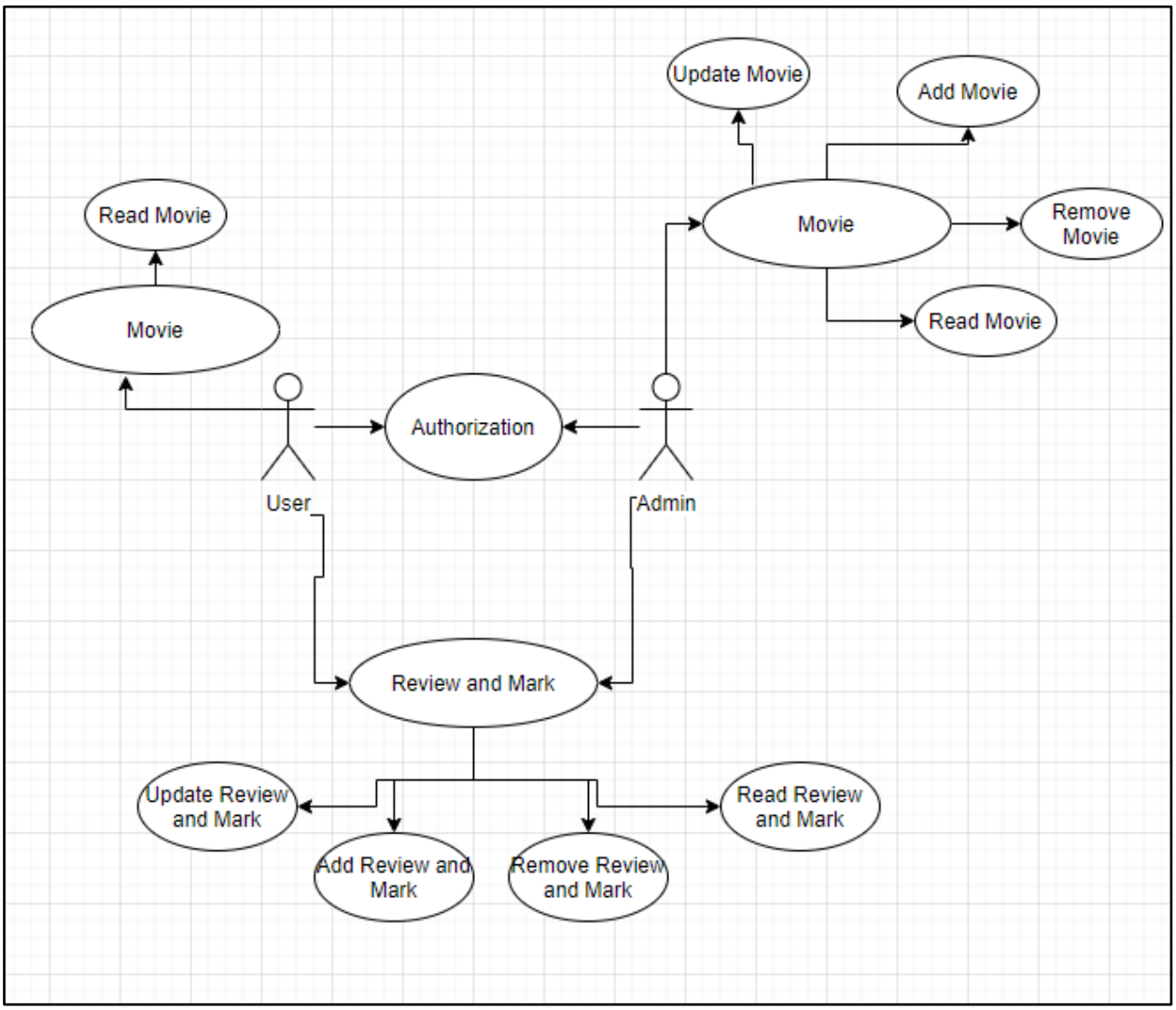


Рисунок 2 – UML диаграмма

На данной диаграмме изображено что пользователь и администратор будут сначала авторизовываться в системе, после чего, если авторизация прошла под ролью обычного пользователя, то можно будет смотреть всю информацию о фильмах, которые находятся на веб-ресурсе. Так же пользователь сможет оставлять отзывы, а так же читать их, к тому же он сможет отредактировать уже имеющий отзыв, и удалить его, по его усмотрению.

Администратор так же имеет тот функционал, который имеет пользователь, но к тому же он способен удалить, обновить или добавить новый фильм, для его отображения пользователю.

Далее необходимо создать концептуальную модель системы. Данная модель будет выглядеть в соответствии с рисунком 3.

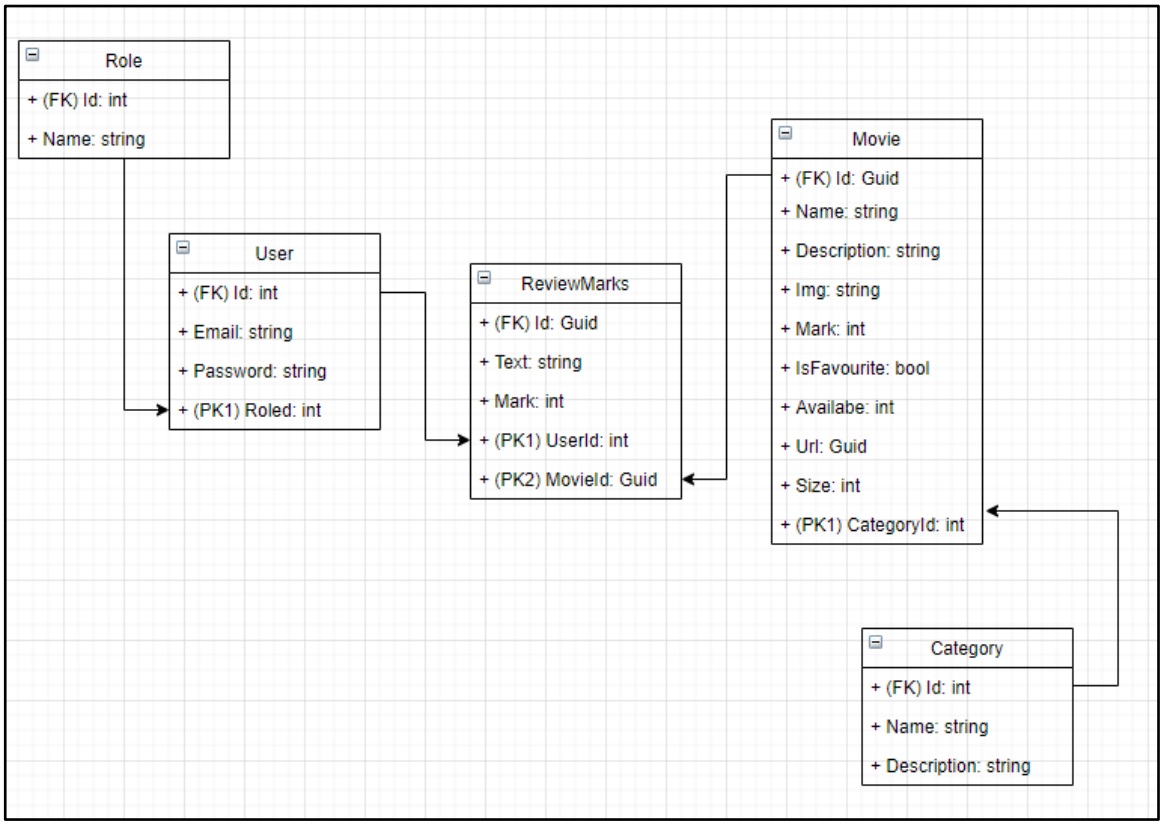


Рисунок 3 – Концептуальная модель базы данных

Так как на нашем сайте будет взаимодействие с пользователем. То в первую очередь добавляем таблицу User (Пользователь). К тому же если будут только пользователи без администраторов и модераторов, то будет анархия, люди будут писать всё что захотят, не будет цензуры и добавления новых фильмов. Поэтому добавляем таблицу Role (Роль пользователя). Пропишем данные таблицы в папке Model в нашем проекте для будущей её миграции. Таблица User будет выглядеть в соответствии с рисунком 5.

```
public class User
{
    public int Id { get; set; }
    public string Email { get; set; }
    public string Password { get; set; }

    public int? RoleId { get; set; }
    public Role Role { get; set; }
}
```

Рисунок 4 – Таблица User

В данной таблице будет содержаться уникальный индификатор пользователя, его почта и пароль, которые он будет вводить при авторизации или регистрации, если аккаунт ещё не создан, также вторичный ключ на ссылку роль, которая сейчас будет создана.

Создаём таблицу роль, в соответствии с рисунком 5.

```
public class Role
{
    public int Id { get; set; }
    public string Name { get; set; }
    public List<User> Users { get; set; }
    public Role()
    {
        Users = new List<User>();
    }
}
```

Рисунок 5 – Таблица Role

В данной таблице будет отображаться уникальный индифиактор роли и её название. По умолчанию при регистрации пользователя ему будет присвоена роль обычного пользователя

Так как веб-ресурс будет связан с кинолентами, добавим таблицу Category (Категория). В данной категории будут располагаться все доступные на сайте категории фильмов, а также сериалы и документальные фильмы. Выглядеть таблица будет в соответствии с рисунком 6.

```
public class Category
{
    public int Id { get; set; }

    public string CategoryName { set; get; }

    public string Description { set; get; }

    public List<Movie> Movies { set; get; }
}
```

Рисунок 6 – Таблица Category

В данной таблице будет отображаться уникальный индифиактор категории, её название и описание категории. Все категории должны быть заранее прописаны в базе данных до начала будущей проверки работоспособности веб-ресурса.

После чего стоит создать саму таблицу с фильмами, назовём её Movie (Фильмы). В данной таблице будут расположены все доступные на сайте фильмы и сериалы. Это является по сути главным звеном на веб-ресурса. С данной таблицей будут прямо или косвенно связаны все остальные таблицы. Выглядеть она будет в соответствии с рисунком 7.

```
public class Movie
{

    public Guid Id { set; get; }
    public string Name { set; get; }
    public string Description { set; get; }
    public string Img { set; get; }
    public float Mark { set; get; }
    public bool IsFavourite { set; get; }
    public int Available { set; get; }
    public int CategoryID { set; get; }
    public string url { set; get; }
    public string size { set; get; }

    public virtual Category Category { set; get; }
}
```

Рисунок 7 – Таблица Movie

В данной таблице будет отображаться уникальный индифиактор фильма, его название, описание, изображение с постером фильма, оценка фильма, отображается ли она на главной странице, наличие его в кино, вторичный ключ категории фильма, его ссылка и время просмотра.

Последняя таблица будет связана с рецензиями и обзорами фильмов, название ReviewMark (Рецензия и оценка). В данной таблице будет считать средняя оценка каждого фильма, а также записываться рецензия, если пользователь её оставил. Таблица будет выглядеть в соответствии с рисунком 8.

```
public class ReviewMark
{
    public Guid Id { set; get; }
    public string Text { set; get; }
    public string Mark { set; get; }
    public int UserId { set; get; }
    public Guid MovieId { set; get; }
    public virtual Movie Movie { set; get; }
    public virtual User User { set; get; }
}
```

Рисунок 8 – Таблица ReviewMark

В данной таблице будет отображаться уникальный индифиактор рецензии, текст, написанный пользователем, так же его оценка, вторичный ключ пользователя, написавшего рецензию, вторичный ключ фильма на который была написана рецензия.

4 РАЗРАБОТА ВЕБ РЕСУРСА

После того как база готово к эксплуатации приступим к созданию интерфейса веб-ресурса.

На всех страницах будет отображено главное меню, на котором расположено название сайта, категории фильмов, вход на сайт, если пользователь уже вошёл, то личный кабинет и выход, а также поиск.

На главной старице будет отображаться лучшие картины среди всех фильмов с рейтингом не менее 4. Выглядеть заглавная страница будет в соответствии с рисунком 9.

Рисунок 9 – Главная страница

Если пользователь захочет посмотреть категории фильмов он может в любое время нажать на понравившуюся ему категорию, и ему откроется страница с этим жанром, в соответчики с рисунком 10.

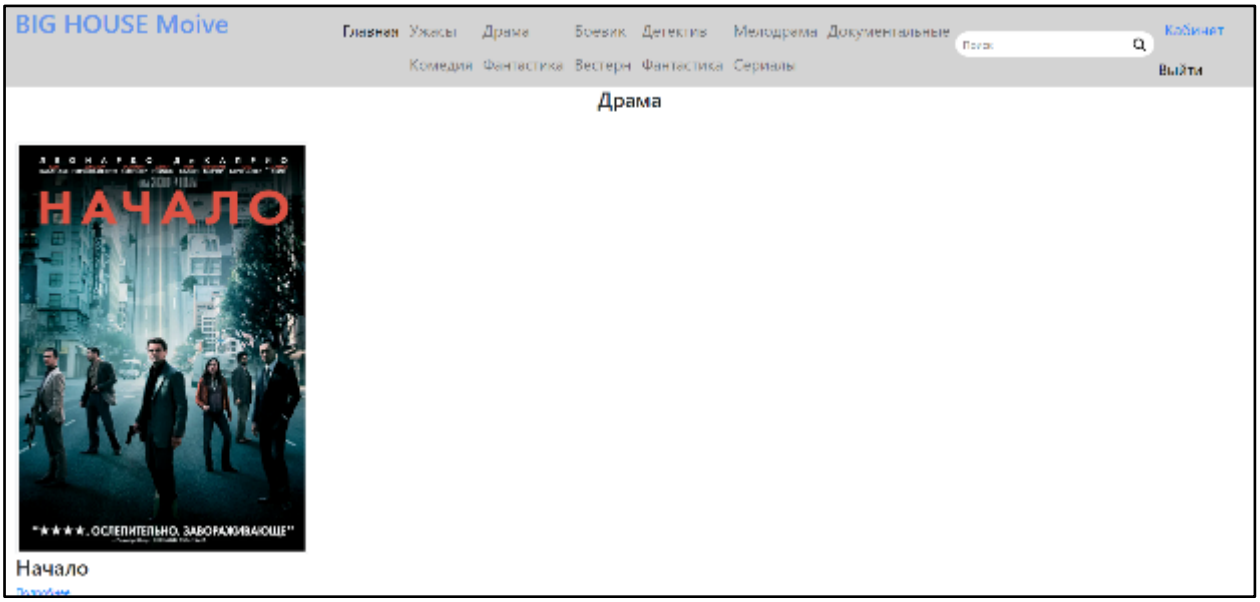


Рисунок 10 – Отображение категории жанра

Если пользователь заглянет в подробное описание фильма, то он увидит само расширенное описание, так же постер в большем размере и после этого будет отображаться суммарная оценка и рецензии, написанные на данный фильм, если пользователь ещё не писал рецензии ему предложат написать и поставить оценку фильму. Страница выглядит в соответствии с рисунком 11.

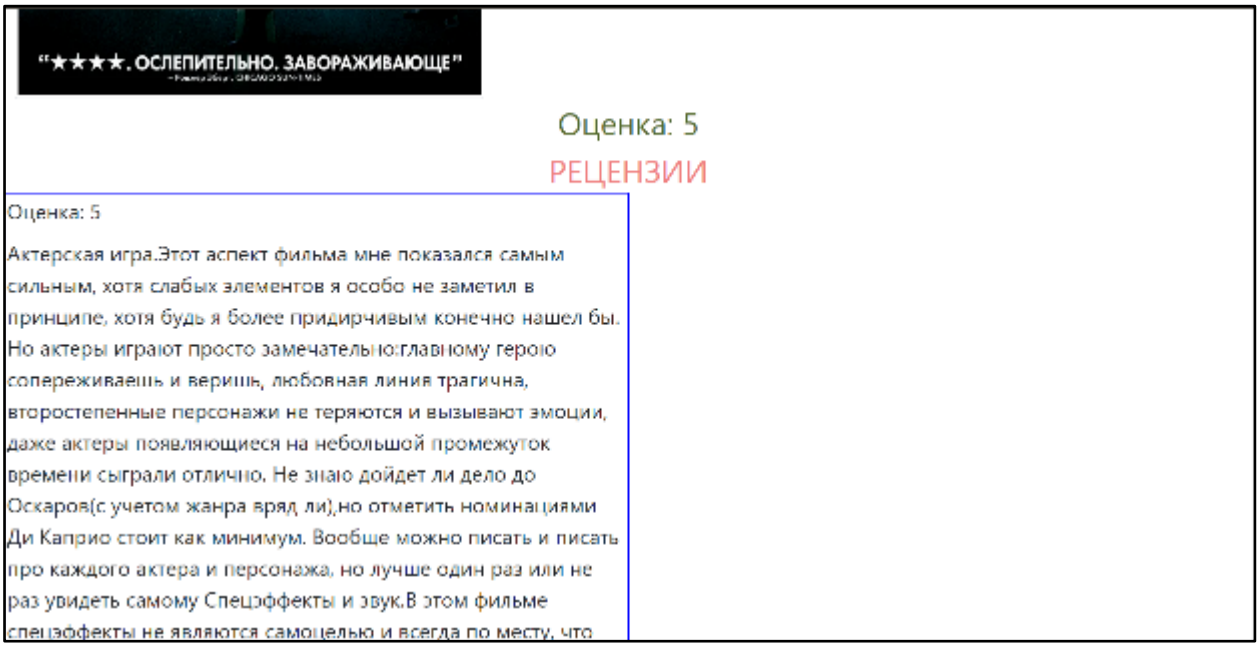


Рисунок 11 – Отображение рецензий на фильм

Если пользователь захочет посмотреть свои прошлые оценки и рецензии он сможет посмотреть их в своём личном кабинете. В нём будут отображаться все фильмы, которые были за рецензированы пользователем. Страница будет выглядеть в соответствии с рисунком 12.

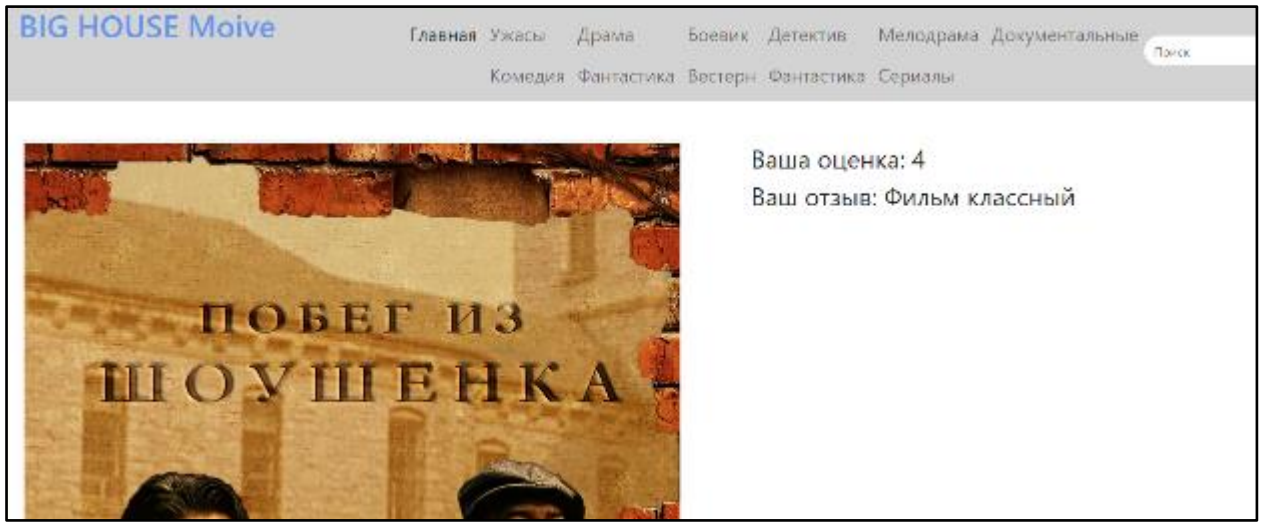


Рисунок 12 – Отображение информации о рецензиях пользователя

Сейчас рассмотрим авторизацию на сайте. В ней возможно ввести свою почту и пароль или же если почты не имеется, то зарегистрироваться. Страница выглядит в соответствии с рисунком 13.

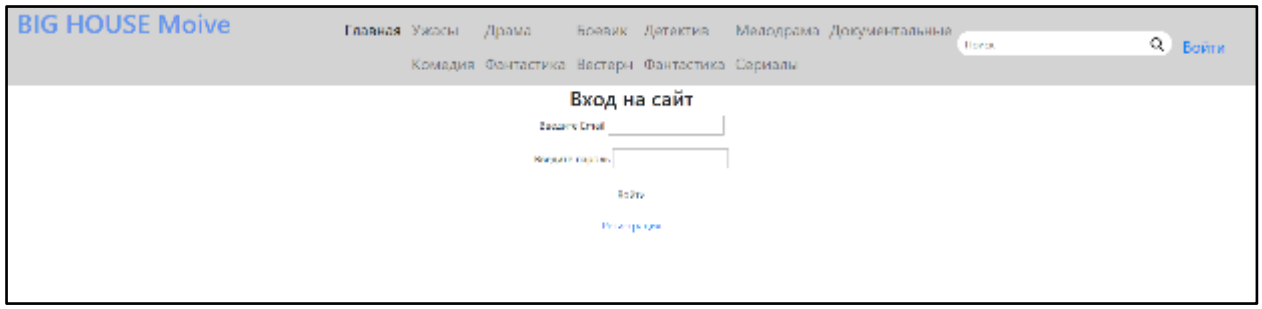


Рисунок 13 – Авторизация на веб-ресурса

Окно регистрации выглядит в соответствии с рисунком 14.

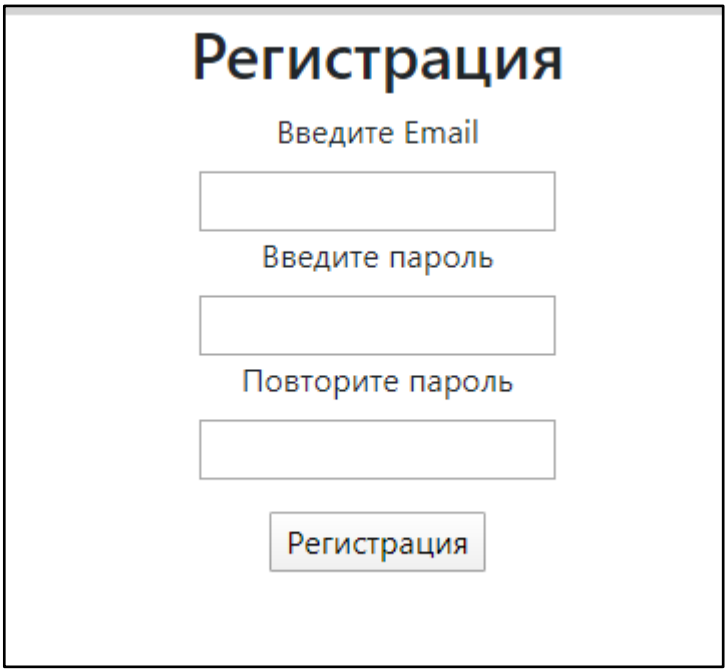


Рисунок 14 – Регистрация

Так же если пользователь имеет роль администратор, то он может добавлять фильмы на сайт. Нужно только ввести название фильма, его описание, выбрать изображение, показывать ли фильм на главной странице, идёт ли фильм в прокате в данный момент, какой жанр фильма. Страница будет выглядеть в соответствии с рисунком 15.

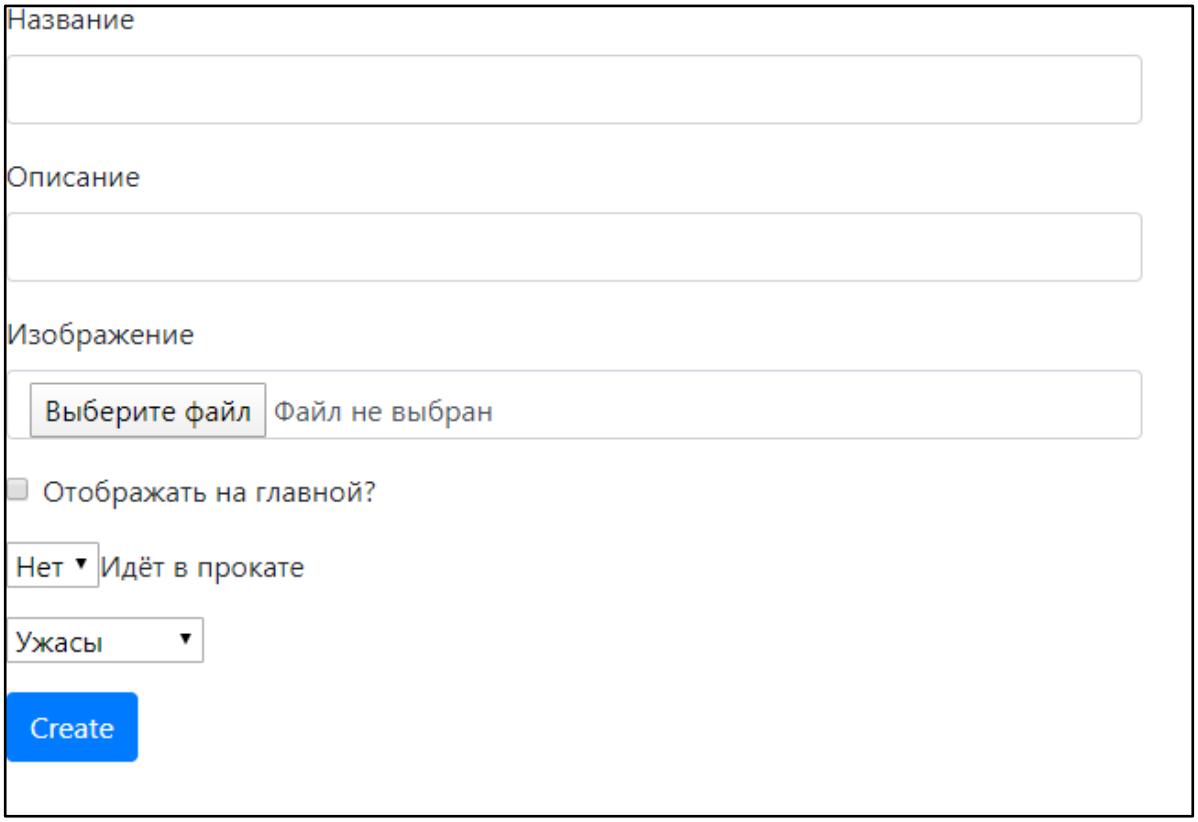


Рисунок 15 – Добавление фильма

Далее будет разобрана сама логика, которая происходит в ресурсе.

Начнём с авторизации, в данном методе пользователь вводит свою почту и пароль, после чего данные передаются на сервер, где сверяются с базой данных, если такие данные есть, то пользователь спокойно входит в свой аккаунт, если какие-то данные не верны, он получит сообщение о неверной почте или пароле. Код будет выглядеть в соответствии с рисунком 16.

```
[HttpPost]
[ValidateAntiForgeryToken]
Ссылок: 0
public async Task<IActionResult> Login(LoginModel model)
{
    if (ModelState.IsValid)
    {
        User user = await _context.Users
            .Include(u => u.Role)
            .FirstOrDefaultAsync(u => u.Email == model.Email && u.Password == model.Password);
        if (user != null)
        {
            await Authenticate(user); // аутентификация

            return RedirectToAction("Index", "Home");
        }
        ModelState.AddModelError("", "Некорректные логин и(или) пароль");
    }
    return View(model);
}
```

Рисунок 16 – Код авторизации

Далее разберём код, при котором фильмы отображаются на главной странице. Веб-ресурс запрашивает на сервер разрешение, если всё в порядке, то с базы данных возвратятся объекты в виде фильмов. Выглядеть код будет в соответствии с рисунком 17.

```
        var merchObj = new MovieListViewModel
        {
            allMovies = movies,
            currCategory = currCategory
        };
        ViewBag.Title = "MovieSoap";

        return View(merchObj);
    }
```

Рисунок 17 – Вывод фильмов

Далее разберём метод поиска фильмов по его названию. Так же пользователь будет вводить в поисковую строку название фильма, если слово, словосочетание или буква присутствует в названии фильма, то данный фильм отобразится в поиске, в этом случае на сервер будет передаваться и сортироваться то, что вводит пользователь. Код поиска будет выглядеть в соответствии с рисунком 18.

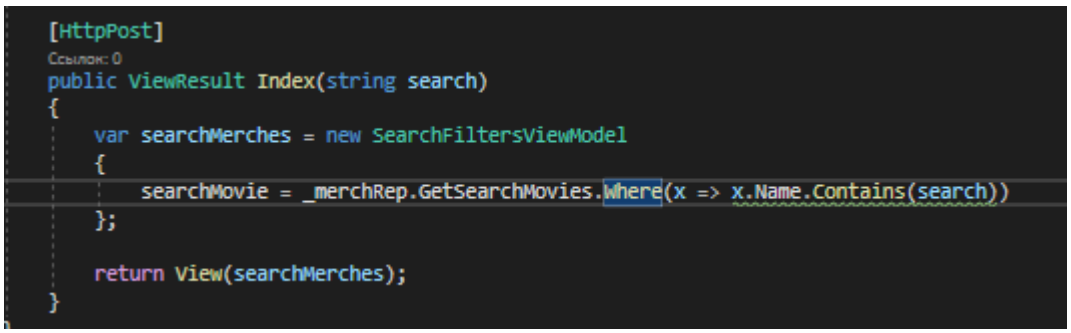
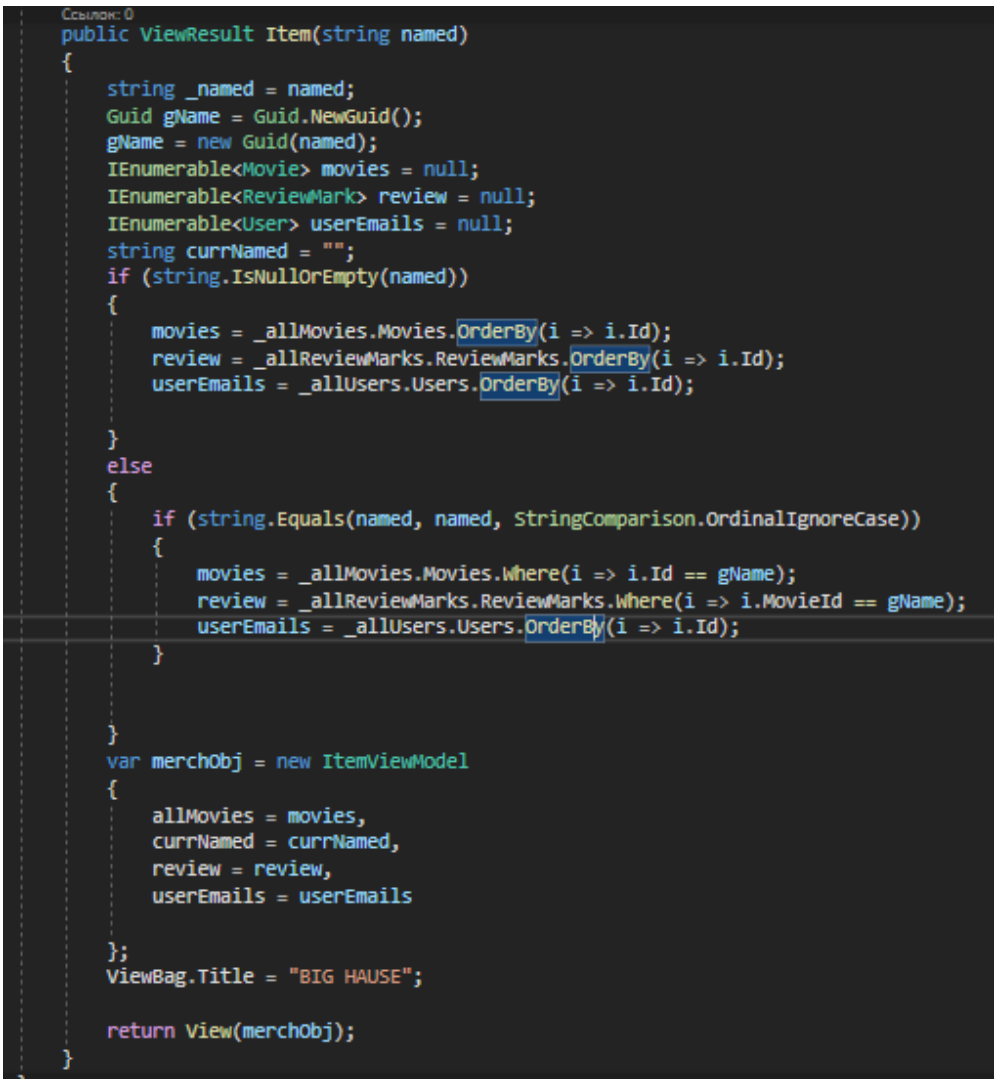


Рисунок 18 – Код поиска фильмов

Далее разберём метод, который позволяет увидеть страницу с описанием, оценкой и рецензией на фильм. В данном методе используется 4 таблицы в базе данных. В данном методе будет передаваться адресная строка, что является индивидуальным индификатором фильма, с помощью его идёт сортировка рецензий, пользователей и названия фильма. Код данного метода будет выглядеть в соответствии с рисунком 19.



```
Ссылок: 0
public ViewResult Item(string named)
{
    string _named = named;
    Guid gName = Guid.NewGuid();
    gName = new Guid(named);
    IEnumerable<Movie> movies = null;
    IEnumerable<ReviewMark> review = null;
    IEnumerable<User> userEmails = null;
    string currNamed = "";
    if (string.IsNullOrEmpty(named))
    {
        movies = _allMovies.Movies.OrderBy(i => i.Id);
        review = _allReviewMarks.ReviewMarks.OrderBy(i => i.Id);
        userEmails = _allUsers.Users.OrderBy(i => i.Id);

    }
    else
    {
        if (string.Equals(named, named, StringComparison.OrdinalIgnoreCase))
        {
            movies = _allMovies.Movies.Where(i => i.Id == gName);
            review = _allReviewMarks.ReviewMarks.Where(i => i.MovieId == gName);
            userEmails = _allUsers.Users.OrderBy(i => i.Id);
        }


    }
    var merchObj = new ItemViewModel
    {
        allMovies = movies,
        currNamed = currNamed,
        review = review,
        userEmails = userEmails

    };
    ViewBag.Title = "BIG HAUSE";

    return View(merchObj);
}
```

Рисунок 19 – Отображение рецензий и описание фильма

Личный кабинет имеет почти такой же метод, как и прошлый, но здесь уже идёт сортировка по уникальному индификатору пользователя, ибо именно он оставляет рецензии. Код будет выглядеть в соответствии с рисунком 20.

```
public ViewResult Index()
{

    IEnumerable<Movie> movies = null;
    IEnumerable<ReviewMark> review = null;
    IEnumerable<User> userEmails = null;

    movies = _allMovies.Movies.OrderBy(i => i.Id);
    review = _allReviewMarks.ReviewMarks.OrderBy(i => i.Id);
    userEmails = _allUsers.Users.OrderBy(i => i.Id);


    var merchObj = new ItemViewModel
    {
        allMovies = movies,
        review = review,
        userEmails = userEmails

    };
    ViewBag.Title = "BIG HAUSE";

    return View(merchObj);
}
```

Рисунок 20 – Код отображения уже оставленных рецензий пользователя

Далее разберём последний Controller в данной проекте. Это добавление новый фильмов. Пользователь заполняет некую форму, которая отправляет данные в метод. Там сперва изображение сохраняется на сервере, для того чтобы в будущем отображать обложку фильма и чуть меняет путь самого файла, далее все данные сохраняются в базу данных. Код выглядит в соответствии с рисунком 21.

```
Ссылок: 0
public async Task<IActionResult> CreateAction(string name, string description, IFormFile img,
                    bool isFavourite, int available, int categoryID)
{
    string path = "/img/" + img.FileName;
    using (var fileStream = new FileStream(_appEnvironment.WebRootPath + path, FileMode.Create))
    {
        await img.CopyToAsync(fileStream);
    }


    Movie movie = new Movie { Name = name, Description = description, Img = path,
                    IsFavourite = isFavourite, Available = available, CategoryID = categoryID, Mark = 0 };
    _context.Movies.Add(movie);

    await _context.SaveChangesAsync();

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

Рисунок 21 – Код добавление нового фильма

5 ТЕСТИРОВАНИЕ

В тестирование будет разобрано введение данных фильмов, допустим администратор захочет ввести в базу фильм «Зелёная книга», для этого нужно ему перейти в раздел добавления фильмов, там он должен ввести название фильма, его описание, вставить изображение, указать будет ли он отображаться на главной странице, идёт ли он в прокате и к какому жанру он относиться. Приблизительно это должно выглядеть в соответствии с рисунком 22.

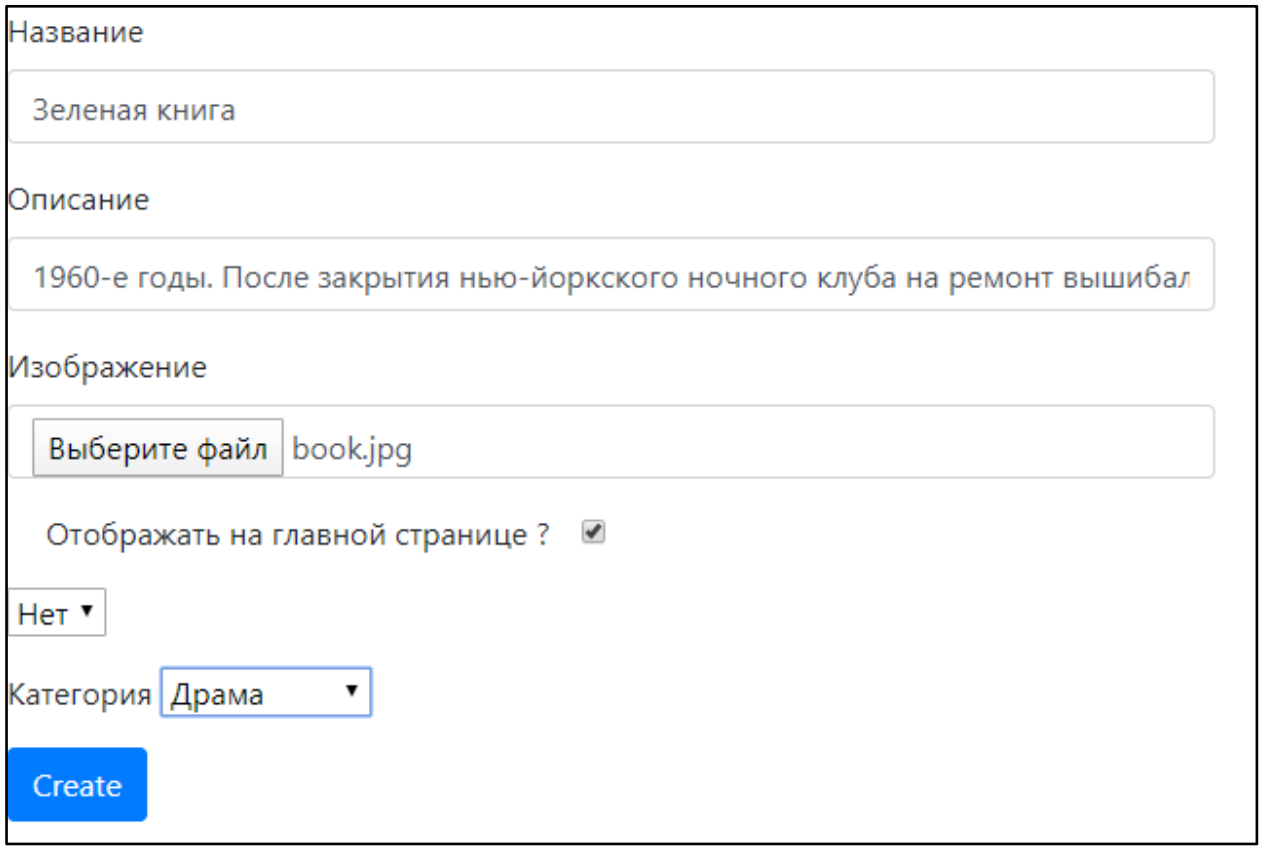


Рисунок 22 – Создание фильма в базе

Далее если всё хорошо создано можно увидеть данный фильм на главной странице, в соответствии с рисунком 23.

Рисунок 23 – Отображение созданного фильма в главном меню