

ОТЗЫВ
на дипломный проект
Годовича Евгения Сергеевича

Тема проекта: Веб-приложение интеграции кинематографического контента.

Дипломный проект посвящен разработке тематического сервиса о кинофильмах. Разработанные программные средства предназначены для предоставления пользователям доступа к информации о кинофильмах, актерах и режиссерах, а также сервиса для извлечения, сбора и систематизации информации по различным критериям с сайта <https://www.kinopoisk.ru/>.

Тема является актуальной, так как решение задачи извлечения и систематизации информации с ведущих сайтов (парсинга) в настоящий момент используется тематическими web интеграторами в представлении информации по теме: новости, онлайн-игры, видео и кинематографический контент.

В процессе дипломного проектирования автор спроектировал и создал базу данных, содержащую данные о кинофильмах и людях, принимающих участие в этих фильмах, реализовал программные средства для доступа к ней, реализовал, разработанный ООО «АйТи Штурман» алгоритм получения web контента с сайта <https://www.kinopoisk.ru/>.

В результате дипломной работы разработан сервис интеграции кинематографического контента.

В ходе выполнения дипломного проекта осуществлена постановка задачи, проведен анализ требований, изучены сервисы-аналоги.

Изложение пояснительной записки выполнено грамотно с соблюдением всех требуемых стандартов, достаточно полно описывает технологии, применяемые в проекте, а так же способы применения программного продукта в целом. Графическая часть полностью описывает структуры данных и алгоритмы программных модулей.

Годович Е.С. успешно справился с поставленной задачей, проявив хорошие знания, инициативность и трудолюбие, умение самостоятельно решать поставленные задачи, пользоваться научно-технической литературой. В настоящее

время программные средства полностью отлажены и протестированы на базе тестового кластера «СИНЕМА 2019» ООО «АйТиШтурман».

Считаю, что дипломный проект выполнен полностью, заслуживает оценки «отлично», а Годович Е.С. достоин присвоения квалификации специалиста по направлению 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения» (специализация «Эксплуатация вычислительных машин, комплексов, систем и сетей специального назначения»).

Руководитель дипломного проекта
к.т.н., доцент



Федюнин Р.Н.

РЕЦЕНЗИЯ

На дипломный проект студента факультета вычислительной техники, специальности 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения», по специализации «Эксплуатация вычислительных машин, комплексов, систем и сетей специального назначения», Годовича Евгения Сергеевича, выполненный на тему: «Веб-приложение интеграции кинематографического контента».

В работе было изучено большое количество материала, выполнен достаточно полный анализ существующих решений на рынке, выявлены плюсы и минусы программных комплексов. Определены цели и метод анализа, на базе которого будет разрабатываться система. Детально раскрыт обзор функционально важных частей системы.

Цель работы – разработка программного обеспечения, позволяющего производить импорт, анализ и визуализацию данных с сайтов о кинематографии на едином интернет портале.

Разработка велась на языке C# с применением РНР. В процессе разработки проведен анализ требований к разрабатываемому продукту, анализ существующих современных средств проектирования программных продуктов, анализ аналогичных систем, изучены способы сетевого взаимодействия.

В результате выполнения дипломного проекта было разработано программное обеспечение, осуществляющее импорт и синтаксический анализ информации о фильмах и кинематографии, посредством авторизации пользователя, для дальнейшего просмотра, анализа и кинематографической информации с помощью интернет - приложения. Осуществлено функциональное тестирование разработанного приложения, которое показало корректность его работы.

Существенных недостатков в дипломной работе не выявлено. Можно отметить следующие:

1. Сложность в реализации алгоритма парсинга;
2. Не полностью рассмотрен пункт, связанный с оисанием среды разработки программной системы.

Указанные замечания не влияют на качество работы.

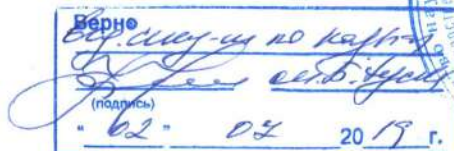
Выпускная квалификационная работа Годовича Е.С. оформлена качественно, технически грамотно. Исследования автора изложены ясно и четко, между разделами существует логическая взаимосвязь. Считаю, что дипломный проект может быть рекомендован к защите и заслуживает оценки «отлично».

Рецензент:

К.т.н., доцент, доцент кафедры "Вычислительные машины и системы" ПензГТУ

А. И. Мартышкин.

02.07.2019 г.



Протокол

проверки на оригинальность в системе «АНТИПЛАГИАТ» (<https://www.antiplagiat.ru>) выпускной квалификационной работы на тему «веб-приложение интеграции кинематографического контента» студента 5 курса Годовича Е.С. направления подготовки/специальности 09.05.01, Применение и эксплуатация АССН.

Руководитель ВКР к.т.н., доцент кафедры «ВТ» Федюнин Р.Н.



Отчет о проверке на заимствования №1



Автор: Федюнин Роман Николаевич frn_penza@mail.ru / ID: 253
Проверяющий: Федюнин Роман Николаевич (frn_penza@mail.ru) / ID: 253
Организация: Пензенский государственный университет
Отчет предоставлен сервисом «Антиплагиат»- <http://pnzgu.antiplagiat.ru>

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 13
Начало загрузки: 05.06.2019 15:19:22
Длительность загрузки: 00:00:36
Имя исходного файла:
pz_godovich_2019-norm
Размер текста: 14689 кБ
Тип документа: Выпускная квалификационная работа
Символов в тексте: 139933
Слов в тексте: 16738
Число предложений: 1745

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Последний готовый отчет (ред.)
Начало проверки: 05.06.2019 15:20:02
Длительность проверки: 00:02:33
Корректировка от 05.06.2019 15:44:51
Комментарии: [Автосохраненная версия]
Модули поиска: Модуль поиска ИПС "Адилет", Модуль выделения библиографических записей, Сводная коллекция ЭБС, Коллекция РГБ, Цитирование, Модуль поиска переводных заимствований, Коллекция eLIBRARY.RU, Коллекция ГАРАНТ, Модуль поиска Интернет, Коллекция Медицина, Модуль поиска перефразирований eLIBRARY.RU, Модуль поиска перефразирований Интернет, Коллекция Патенты, Модуль поиска "ПГУ", Модуль поиска общеупотребительных выражений, Кольцо вузов



ЗАИМСТВОВАНИЯ	ЦИТИРОВАНИЯ	ОРИГИНАЛЬНОСТЬ
29,14%	0,27%	70,59%

« » 2019 г. Федюнин Р.Н.

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная
техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«23» апреля 2019 г.

**ЗАДАНИЕ
НА ДИПЛОМНЫЙ ПРОЕКТ**

1. Студент Годович Е.С. гр. 14ВО2 факультета ВТ

Годович Евгений Сергеевич

2. Тема работы: Разработка программного обеспечения, позволяющего производить импорт, анализ и визуализацию данных с сайтов о кинематографии на едином интернет портале.

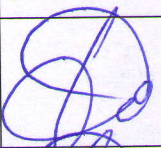
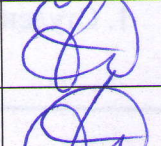
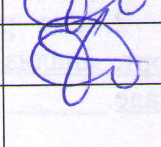
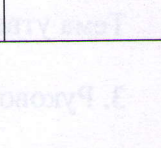
Тема утверждена приказом ПГУ № _____ от "____" _____ 20____ г.

3. Руководитель работы Федюнин Роман Николаевич

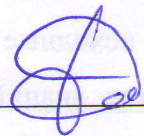
4. Задание на работу (назначение разработки, исходные данные и т.п.) В данной работе поставлена задача разработать тематический портал знаний, содержащего информацию о кинофильмах, о людях, связанных с киноиндустрией. Сайт должен предоставлять пользователям интуитивно понятный интерфейс, чтобы поиск информации был максимально удобным. Для администратора должны быть реализованы возможности добавления информации прямо со страниц сайта. Также необходимо защитить эти страницы от доступа со стороны простых пользователей. Для пользователей должна быть создана специальная страница с расширенным поиском, которая будет позволять им находить информацию по всем возможным критериям, таким как: жанр фильма, актеры, играющие в фильме, режиссеры фильма, страна производитель фильма и т.д.

5. Перечень подлежащих разработке вопросов Разработка базы данных онлайн сервиса,
разработка логической модели базы данных, разработка физической модели базы данных,
разработка программного обеспечения, разработка и анализ пользовательского интерфейса,
разработка программного обеспечения клиентской части, разработка программного обеспечения
серверной части.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Выбор темы и разработка индивидуального плана проведения работ	10	11.02.19-12.02.19	
2	Подбор и изучение материала по теме работы	464	12.02.19-17.04.19	
3	Получение и анализ результатов	300	17.04.19-25.05.19	
4	Оформление отчёта	90	02.06.19	
	Общий объём часов	864		

Дата выдачи задания " 10 " 02 20 19 г.

Руководитель работы  (подпись, дата)

Задание к исполнению принял студент  (подпись, дата)

Работу к защите допустить
Декан факультета  5.02.19
(подпись, дата)

Заведующему кафедрой
"Вычислительная техника"
от студента группы 14В02
Годовича Евгения Сергеевича

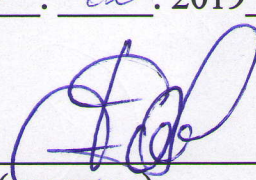
Заявление

Прошу утвердить в качестве руководителя ВКР: Федюнина Р. Н.,
кандидата технических наук, доцента Пензенского государственного
университета.

Руководитель - тема ВКР: Веб-приложение интеграции
кинематографического контента.

Дата: 11. 02. 2019 г.

Руководитель


(подпись)

Студент


(подпись)

Ректору Пензенского
государственного университета
А. Д. Гулякову
студента 5 курса ФВТ очной
формы обучения
Годовича Евгения Сергеевича

ЗАЯВЛЕНИЕ

Я, Годович Евгений Сергеевич, студент 4 курса очной формы обучения направления подготовки 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения», прошу провести проверку с использованием системы «Антиплагиат» бакалаврской работы на тему «Веб-приложение интеграции кинематографического контента», выполненной мной самостоятельно, на содержание элементов плагиата.

Все прямые заимствования из печатных и электронных источников, а также из защищенных ранее выпускных квалификационных работ, кандидатских и докторских диссертаций имеют соответствующие ссылки.

Я ознакомлен с действующим Стандартом университета «Выпускная квалификационная работа обучающихся по образовательным программам высшего образования – программам бакалавриата, программам специалитета и программам магистратуры», согласно которому обнаружение плагиата является основанием для недопуска ВКР к защите и отчисления из университета.

Подпись  «30» 06 2019 г.

ВКР представлена на проверку «30» 06 2019г.

Руководитель ВКР  Р.Н.Федюнин

РЕФЕРАТ

Пояснительная записка содержит «151» лист, «61» рисунок, «19» таблиц, «14» источников, «4» приложения.

СИСТЕМА СИНТАКСИЧЕСКОГО РАЗБОРА, БАЗА ДАННЫХ, КИНЕМАТОГРАФИЧЕСКИЙ КОНТЕНТ.

Объектом исследования является система анализа и интеграции кинематографического контента с сайта «Кинопоиск».

Цель работы – разработка программного обеспечения, позволяющего производить импорт, анализ и визуализацию данных с сайтов о кинематографии на едином интернет портале.

Разработка велась на языке C#, PHP.

В процессе разработки был проведен анализ требований к разрабатываемому продукту, анализ существующих современных средств проектирования программных продуктов, анализ аналогичных систем, изучены способы сетевого взаимодействия.

В результате выполнения дипломного проекта было разработано программное обеспечение, осуществляющее импорт и синтаксический анализ информации о фильмах и кинематографии, посредством авторизации пользователя, для дальнейшего просмотра, анализа и кинематографической информации с помощью интернет - приложения. Осуществлено функциональное тестирование разработанного приложения, которое показало корректность его работы.

В проектной части описаны выполненные функции приложения.

Описана технология разработки и отладки парсинга сайтов.

В экономической части дается технико-экономическое обоснование целесообразности разработки системы.

					ПГУ 1.090501.05.001 ПЗ							
Изм.	Лист	№ докум.	Подпись	Дата	Веб-приложение интеграции кинематографического контента Пояснительная записка			Лит.	Лист	Листов		
Разраб.		Годович Е.С.									3	151
Провер.		Федюнин Р.Н.						Кафедра ВТ Гр.14В02				
Реценз.												
Н. Контр.		Кучин А.В.										
Утверд.												

В разделе безопасности жизнедеятельности описаны психофизиологические факторы, воздействующие на оператора ЭВМ, а также рассмотрено воздействие работы с ПК на зрение человека.

Графическая часть содержит схемы всей системы, данных, структуры базы данных, алгоритма работы программных модулей

					ПГУ 1.090501.05.001 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата					
Разраб.		Годович Е.С.			Веб-приложение интеграции кинематографического контента Пояснительная записка	Лит.	Лист	Листов	
Провер.		Федюнин Р.Н.						3	151
Реценз.						Кафедра ВТ Гр.14ВО2			
Н. Контр.		Кучин А.В.							
Утверд.									

Содержание

Введение.....	5
1 Предпроектный анализ	6
1.1 Постановка задачи на разработку программного обеспечения	6
1.2 Анализ функциональных требований	8
1.3 Обзор методов и средств проектирования	14
2 Разработка программного обеспечения сайта.....	19
2.1 Схема базы данных	19
2.2 Разработка интерфейса пользователя	26
2.3 Разработка программного кода.....	34
3 Тестирование и развертка программного обеспечения	46
3.1 Тестирование программного обеспечения	46
3.2 Развертывание программного обеспечения	56
3.3 Руководство пользователя.....	57
4 Технология разработки синтаксического анализатора сайтов.....	59
Далее требуется проверить работоспособность реализованного кода..	72
5. Техничко-экономическое обоснование разработки программы.....	78
5.1 Введение.....	78
5.2 Расчет затрат на разработку программного обеспечения.....	80
5.3 Расчет цены разработанной программы	81
5.4 Расчет капитальных вложений	82
5.5 Расчет эксплуатационных расходов	82

5.6 Определение расходов на интернет трафик	85
5.7 Заключение	86
6. Безопасность жизнедеятельности. Эргономика рабочего места программиста.	87
6.1 Характеристика условий труда программиста	87
6.2 Эргономические требования к рабочему месту программиста	88
6.3 Освещенность рабочего места программиста.....	93
6.4 Расчет освещения рабочего места программиста.....	95
Заключение	98
Список использованных источников	99
ЛИСТИНГ ПРИЛОЖЕНИЯ.....	101
Приложение А	101
ВЕДОМОСТЬ ДИПЛОМНОГО ПРОЕКТА.....	142
Приложение Б.....	142
ГРАФИЧЕСКАЯ ЧАСТЬ	143
Приложение В	143
ПРЕЗЕНТАЦИЯ	144
Приложение Г	144

Введение

В настоящее время в сети Интернет представлены громадные объемы информации, которые из года в год неконтролируемо растут. Большая часть этой информации становится практически недоступной из-за неэффективной работы поисковых систем, не учитывающих семантику запросов. Проблема усугубляется еще и тем, что в качестве основной задачи поисковых систем в настоящее время рассматривается индексация ресурсов сети при полном отсутствии средств анализа смысла представленной на них информации.

Для решения задачи повышения эффективности поиска в сети Интернет предлагается строить порталы знаний, каждый из которых предоставляет доступ к ресурсам определенной тематики. Основу таких порталов составляют онтологии, содержащие описание структуры и типологии соответствующих сетевых ресурсов.

Каждый портал знаний может иметь иерархическую или сетевую структуру, т.е. состоять из более специализированных порталов, связанных некоторыми отношениями, например, отношением «общее-частное». Такая архитектура делает портал знаний гибким и легко расширяемым. Его основными компонентами являются: онтология, коллекционер онтологической информации о ресурсах, конструктор запросов и формирователь ответов. Портал знаний может поддерживать два вида поиска: локальный и глобальный.

В данной выпускной квалификационной работе создан тематический портал знаний о кинофильмах.

					ПГУ 1.090501.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		7

1 Предпроектный анализ

1.1 Постановка задачи на разработку программного обеспечения

Основной задачей дипломного проекта является создание тематического портала знаний, содержащего информацию о кинофильмах, о людях, связанных с киноиндустрией.

В Интернете есть много сайтов о кинофильмах, но у большинства из них существует ряд недостатков таких как:

- не упорядоченная хранимая информация;
- захламленность рекламой;
- слабый поиск;
- слабый интуитивно понятный интерфейс;
- неуместное использование Wiki технологии;
- обязательная регистрация.

На основе анализа уже существующих сайтов было принято решение о создании собственного сайта.

Сайт должен предоставлять пользователям интуитивно понятный интерфейс, чтобы поиск информации был максимально удобным. Для администратора должны быть реализованы возможности добавления информации прямо со страниц сайта. Также необходимо защитить эти страницы от доступа со стороны простых пользователей. Для пользователей должна быть создана специальная страница с расширенным поиском, которая будет позволять им находить информацию по всем возможным критериям, таким как: жанр фильма, актеры, играющие в фильме, режиссеры фильма, страна производитель фильма и т.д.

Для реализации этого проекта необходимо создать базу данных, которая будет содержать информацию о кинофильмах, актерах и режиссерах. Хранилище данных должно состоять из нескольких взаимосвязанных таблиц [1]. А также сайт, который будет взаимодействовать с этой базой. В соответствии смыслу разрабатываемых таблиц следует задать ограничения для поддержания целостности данных. Механизмы соблюдения корректности работы с хранилищем данных могут быть реализованы как на уровне базы данных и хранилища (триггеры), так и на уровне клиентского приложения [2].

Для получения данных должен формироваться и отсылаться запрос серверу, на котором размещено хранилище данных [3]. Администратору должен быть предоставлен доступ к выполнению всех этих запросов, а также изменения, добавления и удаления строк таблиц базы данных, для чего нужно разработать понятный и удобный в навигации интерфейс.

Краткое описание основных действующих лиц представлено в таблице 1.1.

Таблица 1.1 – Основные действующие лица

Название	Профиль, подготовка и навыки
Администратор	Лицо, постоянно работающее с этим сетевым ресурсом. Опытный пользователь.
Пользователь	Нерегулярный пользователь, имеющий доступ в интернет.

Цели и приоритеты целей основных действующих лиц представлены в таблице 1.2.

Таблица 1.2 – Цели и приоритеты целей основных действующих лиц

Действующее лицо	Цель уровня задачи	Приоритет
Администратор	Авторизация в системе (получение доступа к администрированию сайта)	1
	Добавить / Изменить / Удалить информацию о фильме	1
	Добавить / Изменить / Удалить информацию о актере или режиссере	1
Пользователь	Поиск фильма	1

	Просмотр списка всех фильмов в базе Просмотр списка всех актеров и режиссеров в базе	1 1
	Просмотр конкретной информации о фильме Просмотр конкретной информации об актере или режиссере Оставить отзыв о фильме	1 1 2

Модель предметной области представлена на рисунке 1.1.

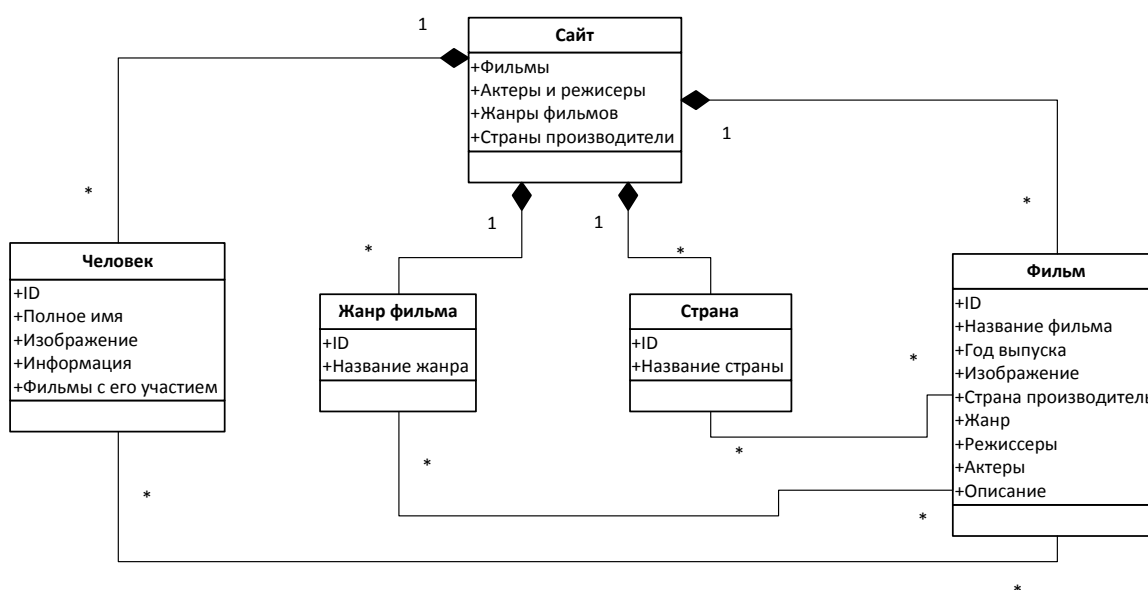


Рисунок 1.1 – Диаграмма классов предметной области

Классы «Человек», «Жанр фильма», «Страна», «Фильм» являются представлениями данных, которые должны храниться в системе. Класс «Сайт» взаимодействует базой и хранит копии данных о фильмах, жанрах фильмов, странах производителей, актерах и режиссерах в виде списков.

1.2 Анализ функциональных требований

Требования можно определить как описание того, что должно быть реализовано. Существуют два основных типа требования:

– функциональные требования – какое поведения должна предлагать система;

– нефункциональные требования – особое свойство или ограничение, накладываемое на систему [4].

В соответствии с требованиями к осуществляемым программой функциям можно отнести:

- поиск фильма;
- авторизация в системе;
- добавление фильма в базу;
- изменение информации о фильме;
- удаление фильма из базы;
- добавление актера или режиссера в базу;
- изменение информации об актере или режиссере;
- удаление актера или режиссера из базы;
- добавление отзыва о фильме.

Исходя из анализа деятельности системы, уже на данном этапе анализа можно дать описание основным вариантам использования разрабатываемого программного обеспечения. Данное описание представлено в таблице 1.3.

Таблица 1.3 – Краткое описание вариантов использования

Действующее лицо	Цель	Краткое описание
Пользователь	Просмотр списка фильмов	Пользователь запрашивает список всех фильмов в БД и получает результат запроса.
	Просмотр списка людей	Пользователь запрашивает список всех актеров и режиссеров в БД и получает результат запроса.
	Поиск фильма	Пользователь вводит критерии поиска, такие как жанр фильма, актеры, режиссеры фильма, а также часть названия фильма и получает список всех возможных фильмов, подходящих под критерии поиска.
	Просмотр информации о фильме	Пользователь запрашивает детальную информацию о конкретном фильме и получает результат запроса.

Продолжение таблицы 1.3 – Краткое описание вариантов использования

Действующее лицо	Цель	Краткое описание
Пользователь	Просмотр информации о человеке Авторизация в системе	Пользователь запрашивает детальную информацию о конкретном актере или режиссере и получает результат запроса. Пользователь вводит логин и пароль для доступа к администрированию сайта. Если данные корректны, то пользователь получает права администратора.
Администратор	Добавление фильма в базу Добавление актера или режиссера в базу Изменение информации о фильме Изменение информации об актере или режиссере Удаление фильма из базы Удаление актера или режиссера из базы	Администратор заполняет информацию о фильме и сохраняет ее в БД. Администратор заполняет информацию об актере или режиссере и сохраняет ее в БД. Администратор изменяет информацию о фильме, хранящуюся в базе, и сохраняет ее. Администратор изменяет информацию об актере или режиссере, хранящуюся в базе, и сохраняет ее. Администратор запрашивает удаление конкретного фильма из базы. Система находит все связи с фильмом и удаляет их. После чего из базы удаляется фильм. Администратор запрашивает удаление конкретного человека из базы. Система находит все связи с этим человеком и удаляет их. После чего из базы удаляет человек.

Диаграмма прецедентов (вариантов использования) – это наглядное представление субъектов и прецедентов вместе с любыми дополнительными определениями и спецификациями. Диаграмма прецедентов представляет

собой не просто некую схему, а является полностью документированной моделью предлагаемого поведения системы [5].

На основании таблицы 3 можно создать диаграмму вариантов использования для разрабатываемого программного обеспечения, включающую основные действия пользователя.

Диаграмма вариантов использования представлена на рисунке 1.2.

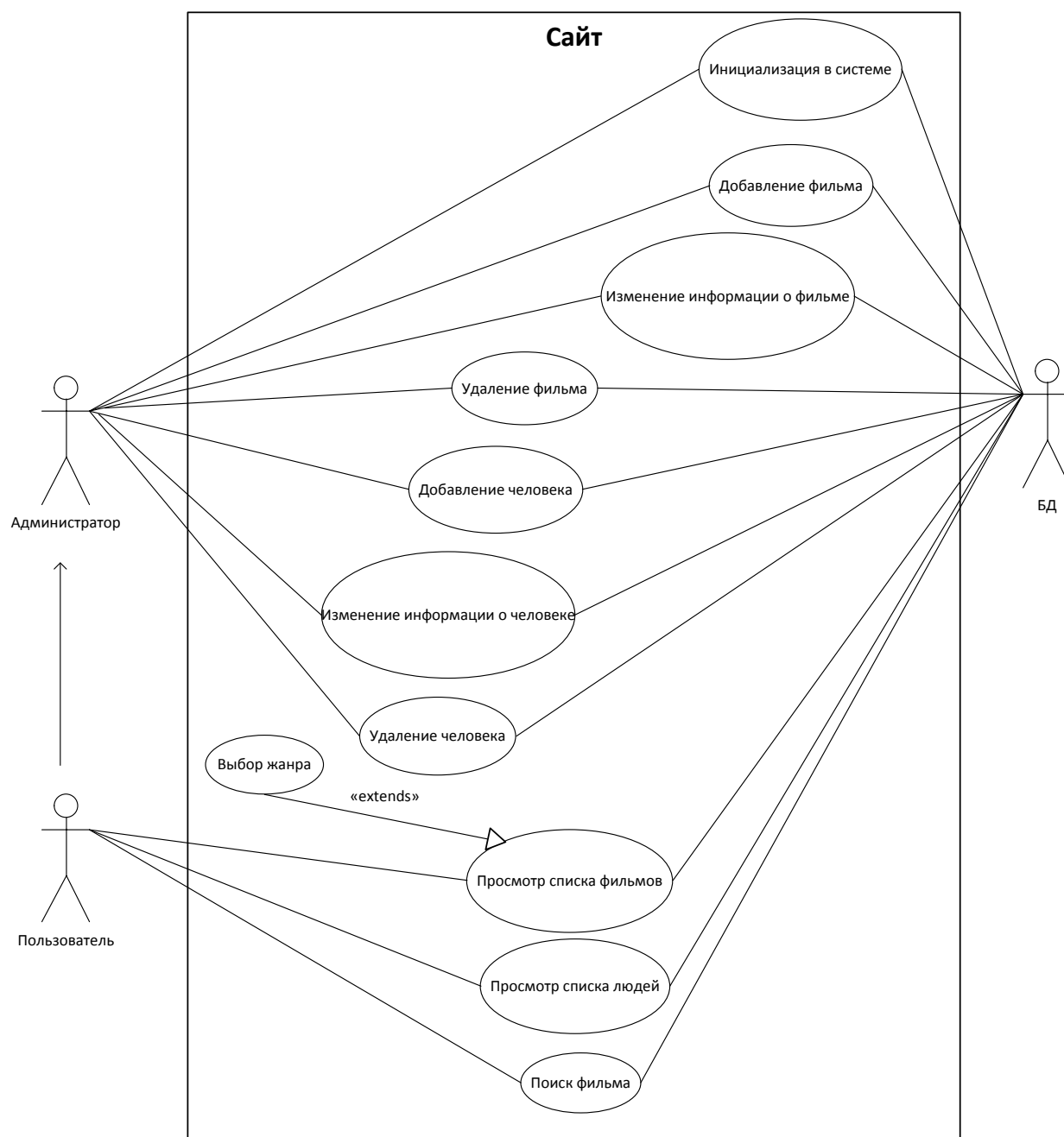


Рисунок 1.2 – Диаграмма вариантов использования

Одним из способов описания вариантов использования является сценарий. Сценарий представляет собой последовательность шагов, описывающих взаимодействие между пользователем и системой. Количество деталей в сценарии зависит от риска в соответствующем варианте использования: чем больше риск, тем больше деталей необходимо указать. Часто случается так, что на фазе исследования детально описывается небольшое количество вариантов использования [6].

Описание вариантов использования:

Вариант использования: Просмотр списка фильмов

Основное действующее лицо: Пользователь.

Область действия: Сайт.

Уровень: Цель пользователя.

Участники и интересы: Пользователь хочет просмотреть список всех фильмов на сайте.

Предусловие: У пользователя есть доступ в интернет. Открыта любая страница сайта.

Минимальные гарантии: Пользователь остался на текущей странице сайта.

Гарантия успеха: Пользователь перешел на страницу со список всех фильмов.

Основной сценарий:

- 1) пользователь нажимает кнопку «Список фильмов»;
- 2) система генерирует результаты из базы;
- 3) пользователь получает результат из базы со списком фильмов.

Вариант использования: Поиск.

Основное действующее лицо: Пользователь.

Область действия: Сайт.

Уровень: Цель пользователя.

Участники и интересы: Пользователь хочет найти информацию о фильме по названию фильма.

Предусловие: У пользователя есть доступ в интернет. Открыта любая страница сайта.

Минимальные гарантии: Пользователь получил список всех возможных фильмов.

Гарантия успеха: Пользователь получил информацию о нужном фильме.

Основной сценарий:

- 1) пользователь вводит название фильма в форму поиска;
- 2) пользователь нажимает кнопку «Найти»;
- 3) пользователь получает результаты по своему запросу.

Обобщенный процесс функционирования системы изображен на диаграмме состояний (рисунок 3).



Рисунок 1.3 – Диаграмма состояний (обобщенное функционирование)

Работу системы можно описать так:

- 1) пользователь входит в систему и выбирает требуемое действие;
- 2) система производит транзакцию и выдает результат пользователю;

3) пользователь возвращается в начало, где он может либо выбрать следующее действие, либо выйти из системы.

Целевым вариантом использования является «Добавление фильма». Диаграмма деятельности, построенная в соответствии с подробным текстовым описанием данного варианта использования, представлена ниже на рисунке 4.

Диаграмма деятельности описывает последовательность подобных действий, позволяя при этом одновременно изображать как условное, так и параллельное поведение. Диаграмма деятельности представляет собой вариант диаграммы состояний, в которой большинство состояний являются состояниями деятельности. Таким образом, большая часть терминологии совпадает с терминологией диаграмм состояний [6].

После отправки данных о фильме на добавление система должна проверить соединение с базой. Если соединение не установлено, то она должна оповестить администратора об ошибке, иначе система смотрит наличие фильма в базе. При присутствии фильма в базе, система удаляет все ассоциации с этим фильмом. Далее система пробует добавить новые данные о фильме в базу. После добавления данных в базу система пробует добавить ассоциации с фильмом в базу.

В результате анализа функциональных требований установлены варианты использования приложения, расписаны сценарии некоторых из них, и один из вариантов использования рассмотрен на диаграмме деятельности.

1.3 Обзор методов и средств проектирования

Язык программирования C#, разработанный в корпорации Microsoft рабочей группой под руководством Андерса Хейльсберга и Скотта Вилтамата, предназначен специально для платформы .NET в качестве языка, позволяющего программистам перейти на технологию .NET.

					ПГУ 1.090501.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		16

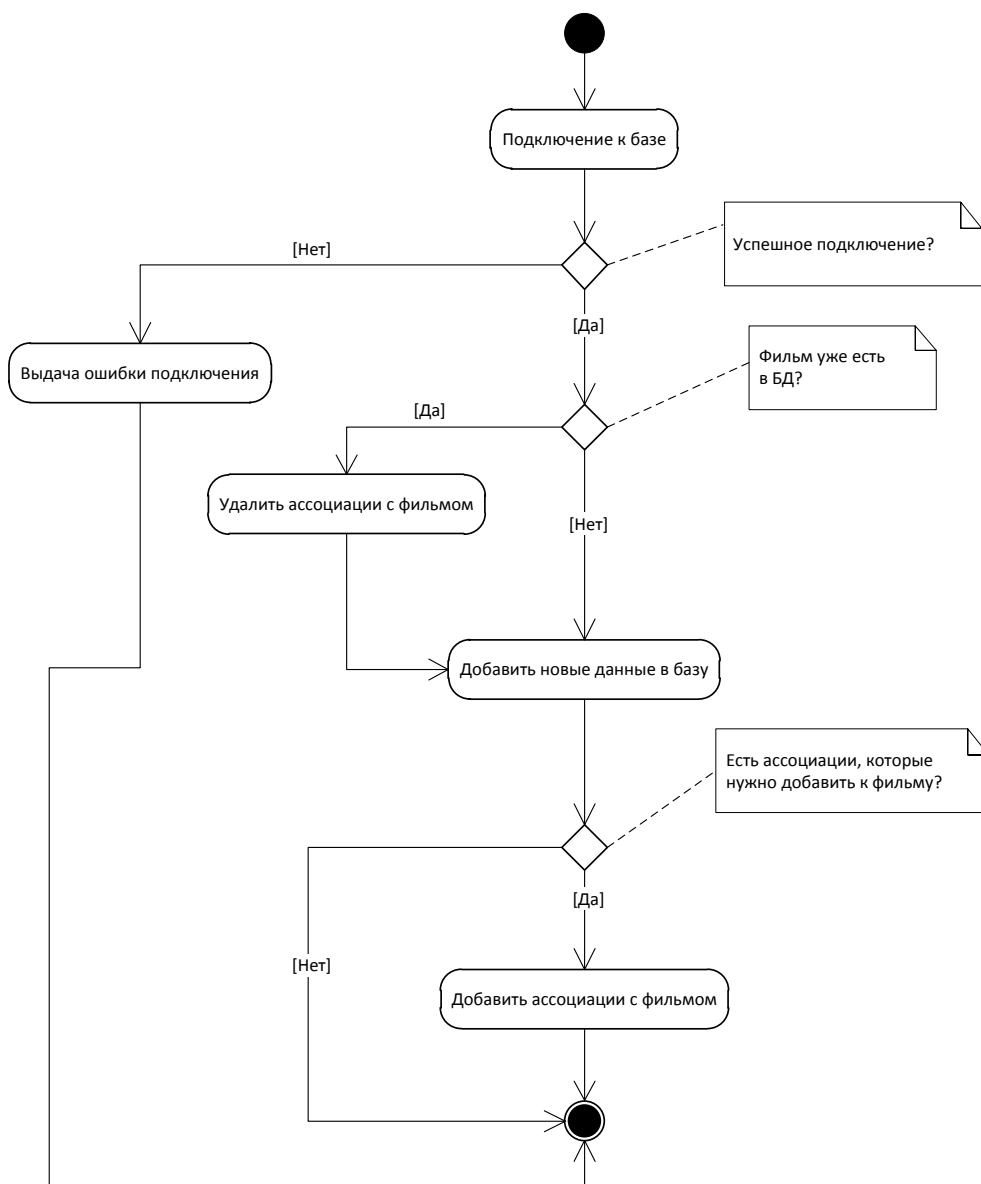


Рисунок 1.4 – Диаграмма деятельности «Добавить фильм»

Своими корнями С# уходит в С, С++ и Java с адаптацией лучших функций каждого языка и добавлением собственных.

С# - это событийно-управляемый, полностью объектно-ориентированный язык визуального программирования, в котором программы создаются с помощью интегрированной среды разработки (IDE). В этой среде программист может спокойно писать, запускать, тестировать и отлаживать программы, написанные на С#, сэкономив массу времени на создание рабочей программы без помощи IDE.

.Net Framework – ядро концепции .NET. Данная структура управляет программными приложениями и запускает их, содержит библиотеку классов

.Net Framework (FCL), обеспечивает защиту и множество других возможностей программирования. .Net Framework существует только для платформы Windows.

Среда Common Language Runtime (CLR, общезыковая среда выполнения) – еще одна центральная часть .Net Framework; она выполняет программы, написанные на C#. Сначала программы компилируются в промежуточный язык Microsoft (MSIL), определяющий команды для CLR. Код преобразуется в MSIL из других языков, и источники переплетаются с помощью CLR. Затем другой компилятор в CLR компилирует MSIL в машинный код с созданием единого программного приложения.

.Net Framework также обеспечивает высокую степень возможности взаимодействия языков. Все программы, написанные на разных языках, компилируются в MSIL; различные части могут быть объединены для создания единой унифицированной программы. MSIL позволяет .Net Framework быть независимой от языка, потому что MSIL не связан ни с каким языком программирования.

Возможность взаимодействия языков предлагает много преимуществ. К примеру, разработчики на C#, Visual Basic .NET, F# и Visual C++ могут работать совместно над одним и тем же проектом без необходимости изучения другого языка программирования; части кода легко кодируются в MSIL и связываются в единую программу. Кроме того, .Net Framework может объединять в пакеты уже существующие компоненты с компонентами .Net для совместной работы, что позволяет многократно использовать код.

Другим преимуществом .Net Framework являются функции исполнения и управления CLR. CLR управляет памятью, защитой и другими функциями, избавляя программиста от ответственности за выполнение этих задач[7].

LINQ – технология Microsoft, предназначенная для поддержки запросов к данным всех типов на уровне языка. Эти типы включают массивы, коллекции в памяти, базы данных, документы XML и многое другое[8].

Используя некоторые новые особенности языка, LINQ позволяет использовать SQL-подобный синтаксис непосредственно в коде программы, написанной, например, на языке C#:

- лямбда-выражения;
- деревья выражений;
- ключевое слово var, инициализация объектов и коллекций и анонимные типы;
- расширяющие методы;
- частичные методы;
- выражения запросов.

RНР.YII – это платформа для разработки веб-приложений от Microsoft, которая сочетает в себе эффективность и аккуратность архитектуры «модель-представление-контроллер» (model-view-controller – MVC), новейшие идеи и приемы гибкой разработки, а также все лучшее из платформы RНР.

Платформа MVC построена в виде серии независимых компонентов – реализующих интерфейс RНР.YII или построенных в виде абстрактного базового класса, - поэтому можно легко заменить систему маршрутизации, механизм визуализации, фабрику контроллеров или прочие компонент платформы другими, с собственной реализацией.

Для каждого компонента MVC Framework разработчики RНР MVC представляют три возможности:

- 1) использование стандартной реализации компонента в том виде, как она есть;
- 2) порождение подкласса от стандартной реализации с целью корректировки существующего поведения;
- 3) полная замена компонент новой реализацией интерфейса или абстрактного базового класса.

В платформе PHP.YII MVC учтена важность генерации ясного и соответствующего стандартам кода разметки. Ее встроенные вспомогательные методы разметки HTML создают соответствующий стандартам вывод. Вместо громадного объема трудно поддающегося управлению HTML-кода, платформа MVC стимулирует создание простых и элегантных, стилизованных с помощью CSS компонентов[9].

JavaScript – это интерпретируемый язык программирования с объектно-ориентированными возможностями. Это нетипизированный язык, т.е. в нем не требуется определять типы переменных.

Обычно JavaScript применяется в веб-браузерах, а расширение его возможностей за счет введения объектов позволяет организовать взаимодействие с пользователем, управлять веб-браузером и изменять содержимое документа, отображаемое в пределах окна веб-браузера. Эта встроенная версия запускает сценарии, внедренные в HTML-код веб-страниц [10].

В данной выпускной квалификационной работе в качестве среды разработки использовалась среда Visual Studio 2016, позволяющая создавать серверные приложения MVC3 на языке программирования C#. Страницы сайта написаны на языке разметки XHTML с использованием синтаксиса Razor. Оформление сайта создано с помощью CSS (формальный язык описания внешнего вида документа, написанного с использованием языка разметки).

2 Разработка программного обеспечения сайта

2.1 Схема базы данных

В соответствии с анализом предметной области была разработана структура БД. Диаграмма базы данных приведена на рисунке 2.1.

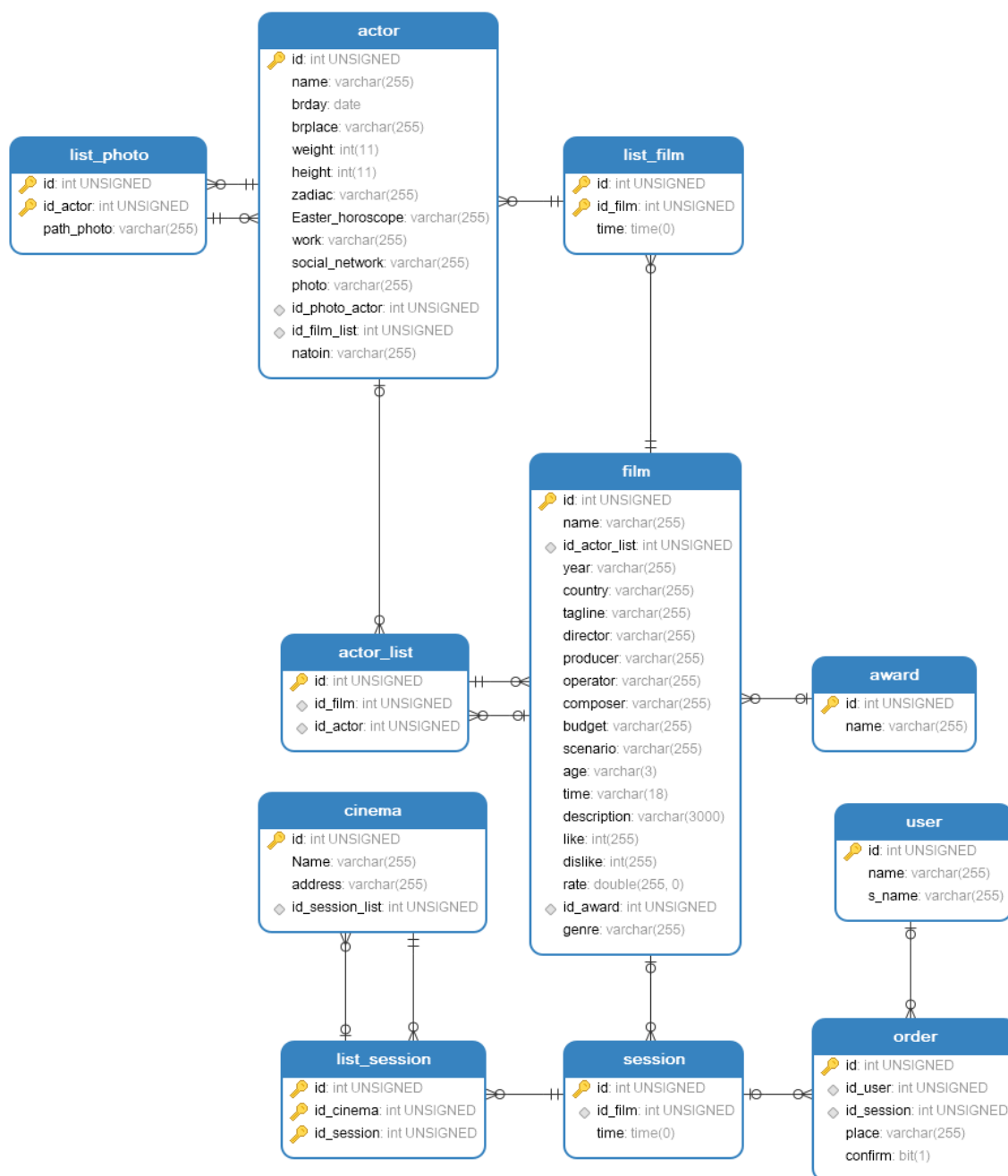


Рисунок 2.1 – Диаграмма базы данных

Для хранения информации о фильме была созданная таблица Films, которая хранит информацию о названии фильма, годе выпуска, описании фильма и постере фильма. Т.к. фильм может иметь множество жанров, стран производителей, а также в нем могут принимать участие множество людей, было решено выделить эти данные в отдельные таблицы: Countries, People, Genres. Для реализации зависимостей многие-ко-многим между таблицами были созданы вспомогательные таблицы: FilmCountry (отношение фильма и страны производителя), FilmGenre (отношение фильма и жанра фильма), FilmPersonAssociation (отношение фильма и людей принимающих участие в фильме).

Таблица 2.1 – Сущность actor, отвечает за хранение данных о актёрах.

№	Наименование поля	Описание поля	Тип данных	Первичный/Внешний ключ
1	Id	Уникальный идентификатор актёра	Int	PK
2	Name	Имя актёра	Varchar	-
3	Brday	Дата рождения	Date	-
4	Brplace	Место рождения	Varchar	-
5	Weight	Вес	Int	-
6	Height	Рост	Int	-
7	Zodiac	Знак зодиака	Varchar	-
8	Easter_horoscope	Знак по восточному гороскопу	Varchar	-
9	Work	Должности	Varchar	-
10	Social_network	Социальные сети	Varchar	-

11	Photo	Путь к фотографии	Varchar	-
12	Id_photo_list	Список путей к фотографиям	Int	FK
13	Id_fim_list	Список фильмов	Int	FK
14	nation	Национальность	varchar	-

Таблица 2.2 –Сущность list_photo, отвечает за хранение данных о путях нахождения фотографий актёров.

№	Наименование поля	Описание поля	Тип данных	Первичный/Внешний ключ
1	Id	Уникальный идентификатор записи	Int	PK
2	Id_actor	Уникальный идентификатор актёра	Int	PK/FK
3	Path_photo	Путь нахождения фотографии актёра	varchar	-

Таблица 2.3 – Сущность list_film, отвечает за хранение данных о списке фильмов.

№	Наименование поля	Описание поля	Тип данных	Первичный/Внешний ключ
1	id	Уникальный идентификатор	int	PK

		записи		
2	Id_film	Уникальный идентификатор фильма	int	PK/FK
3	time	Время	time	-

Таблица 2.4 – Сущность actor_list, отвечает за хранение информации о связи актёров и фильмов.

№	Наименование поля	Описание поля	Тип данных	Первичный/Внешний ключ
1	id	Уникальный идентификатор записи	int	PK
2	Id_film	Уникальный идентификатор фильма	Int	FK
3	Id_actor	Уникальный идентификатор Актёра	int	FK

Таблица 2.5 – Сущность award, отвечает за хранение информации о премиях.

№	Наименование поля	Описание поля	Тип данных	Первичный/Внешний ключ
1	Id	Уникальный идентификатор премии	Int	PK
2	Name	Наименование премии	varchar	-

Таблица 2.6 – Сущность film, отвечает за хранение информации о фильмах.

№	Наименование поля	Описание поля	Тип данных	Первичный/Внешний ключ
1	id	Уникальный идентификатор фильма	Int	РК
2	Name	Наименование фильма	Varchar	-
3	Id_actor_list	Уникальный идентификатор списка актёров	Int	FK
4	Year	Год создания	varchar	-
5	Country	Страна создания	varchar	-
6	Tagline	Слоган	varchar	-
7	Director	Режиссёр	varchar	-
8	Producer	Продюсер	varchar	-
9	Operator	Оператор	varchar	-
10	Composer	Композитор	varchar	-
11	Budget	Бюджет	varchar	-
12	Scenario	Сценарист	varchar	-
13	Age	Ограничение возраста фильма	varchar	-
14	Time	Продолжительность фильма	varchar	-
15	Description	Краткое описание	varchar	-
16	Like	Указатель симпатии пользователей	Int	-

17	Dislike	Указатель антипатии пользователей	Int	-
18	Rate	Рейтинг фильма	double	-
19	Id_award	Уникальный идентификатор премии	Int	FK
20	genre	Жанр	varchar	-

Таблица 2.7 – Сущность cinema, отвечает за хранение информации о кинотеатрах.

№	Наименование поля	Описание поля	Тип данных	Первичный/Внешний ключ
1	id	Уникальный идентификатор кинотеатра	Int	PK
2	name	Наименование кинотеатра	Varchar	-
3	address	Адрес кинотеатра	Varchar	-
4	Id_session_list	Список сеансов	int	FK

Таблица 2.8 – Сущность list_session, отвечает за хранение информации о списках сеансов.

№	Наименование поля	Описание поля	Тип данных	Первичный/Внешний ключ
1	id	Уникальный идентификатор записи	Int	PK
2	Id_cinema	Уникальный	Int	PK/FK

		идентификатор кинотеатра		
3	Id_session	Уникальный идентификатор сеанса	Int	PK/FK

Таблица 2.9 – Сущность session, отвечает за хранение информации о сеансах.

№	Наименование поля	Описание поля	Тип данных	Первичный/Внешний ключ
1	id	Уникальный идентификатор записи	Int	PK
2	Id_film	Уникальный идентификатор фильма	Int	FK
3	time	Время начала сеанса	time	-

Таблица 2.10 – Сущность order, отвечает за хранение информации о заказе билета.

№	Наименование поля	Описание поля	Тип данных	Первичный/Внешний ключ
1	id	Уникальный идентификатор записи	Int	PK
2	Id_user	Уникальный идентификатор пользователя	Int	FK
3	Id_session	Уникальный	int	FK

		идентификатор сеанса		
4	Place	Место пользователя	Varchar	-
5	confirm	Подтверждение покупки	Bit	-

Таблица 2.11 – Сущность user, отвечает за хранение информации о пользователях.

№	Наименование поля	Описание поля	Тип данных	Первичный/Внешний ключ
1	id	Уникальный идентификатор записи	Int	РК
2	Name	Имя пользователя	Varchar	-
3	S_name	Фамилия пользователя	varchar	-

2.2 Разработка интерфейса пользователя

Дизайн пользовательского интерфейса должен быть одобрен и принят его пользователями. Без хорошо проработанного интерфейса даже выдающаяся система не может быть успешной [11].

При открытии сайта появляется домашняя страница, на которой пользователь может выбирать жанр фильмов, выбрать просмотр списка людей и фильмов, а перейти на страницу входа в систему. Прототип домашней страницы сайта представлен на рисунках 2.2 – 2.10.

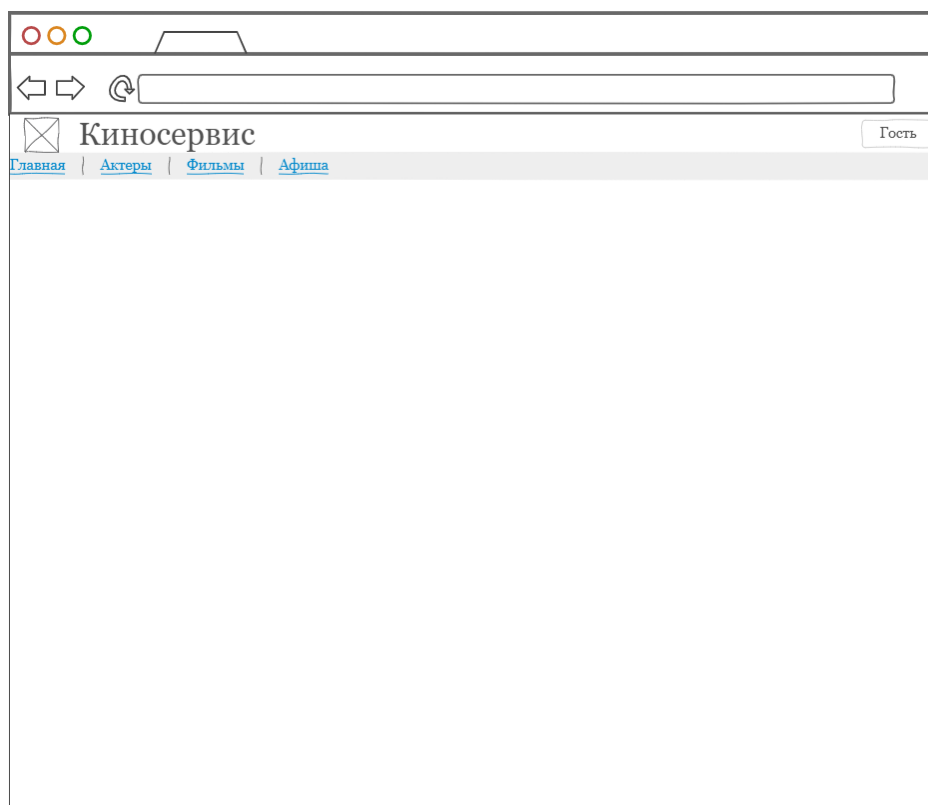


Рисунок 2.2 – Основная страница сайта до парсинга

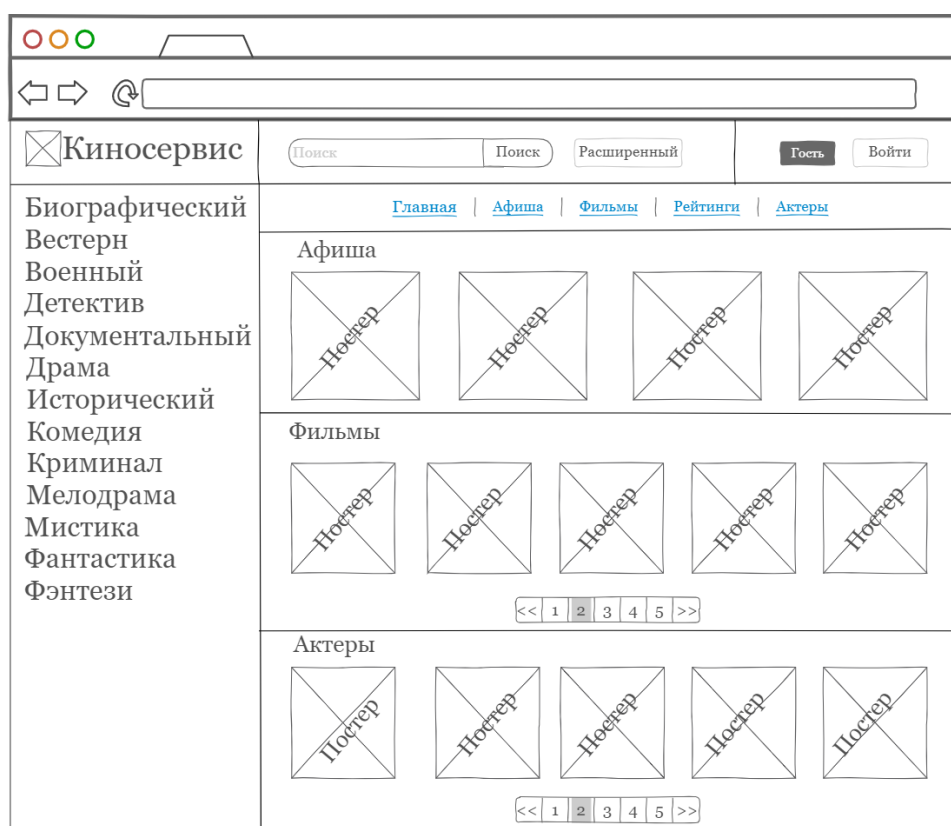


Рисунок 2.3 – Основная страница сайта после парсинга

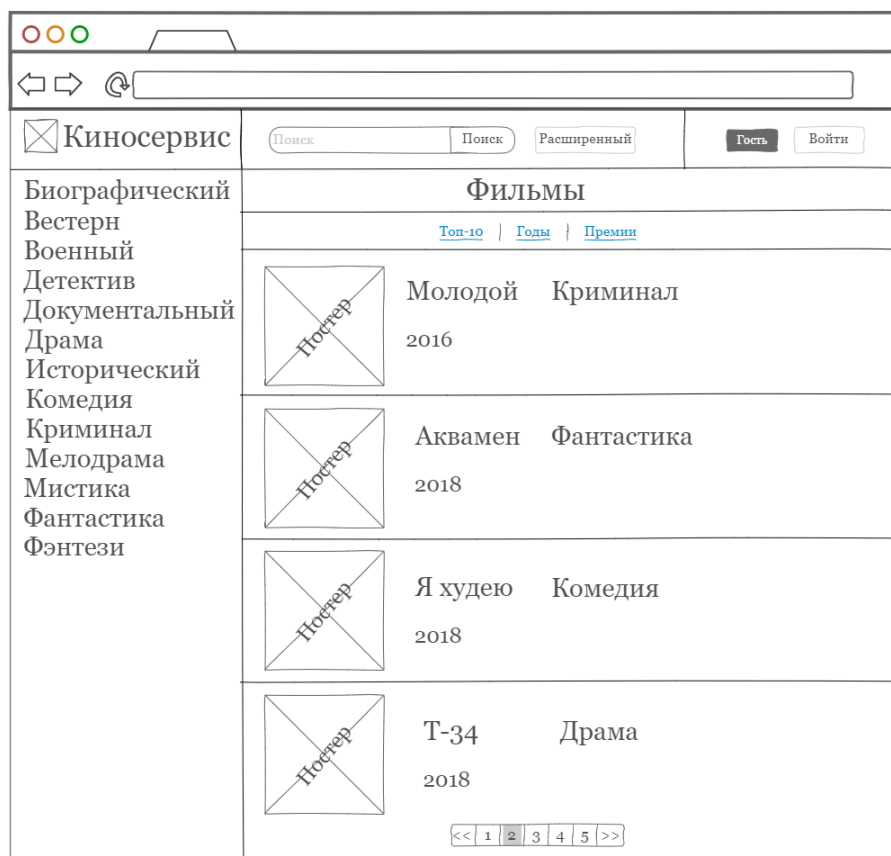


Рисунок 2.4 –Страница со списком фильмов

При нажатии на постере в списке фильмов произойдет переход на страницу с детальной информацией об фильмах, афише и рейтинге.

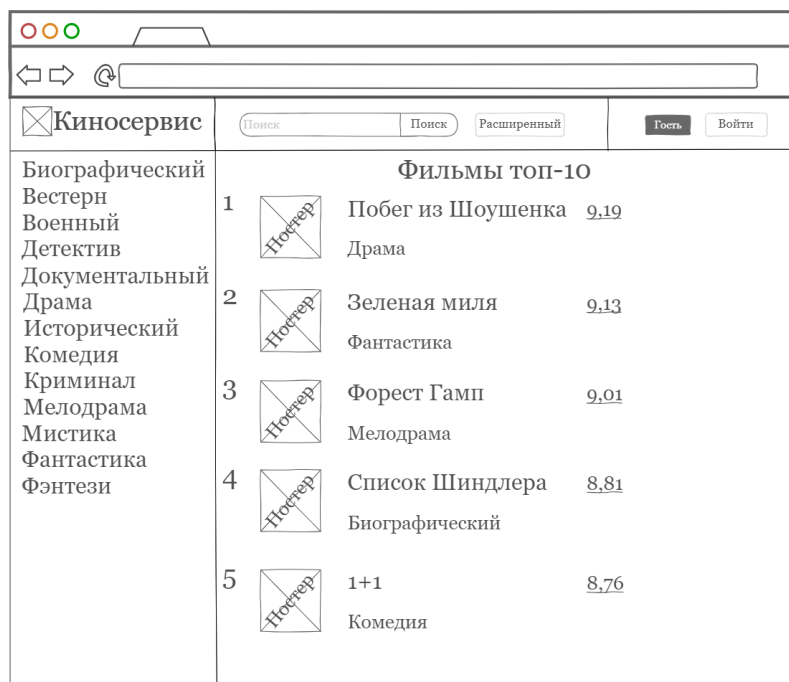


Рисунок 2.5 – Страница с фильмами из топ рейтинга

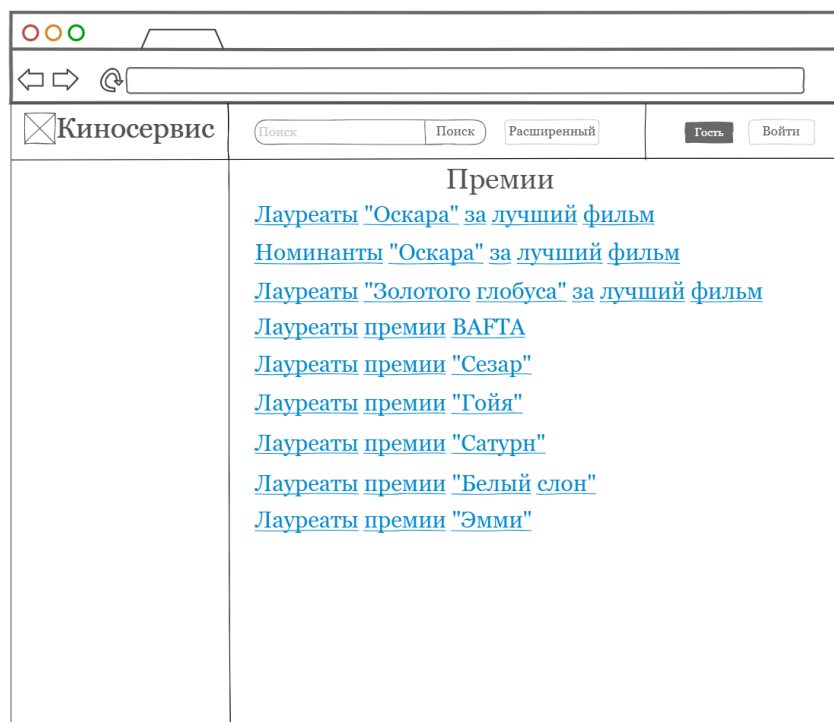


Рисунок 2.6 – Страница со списком премий кинематографа

При нажатии на ссылку пользователь попадает на страницу с афишей конкретных фильмов.

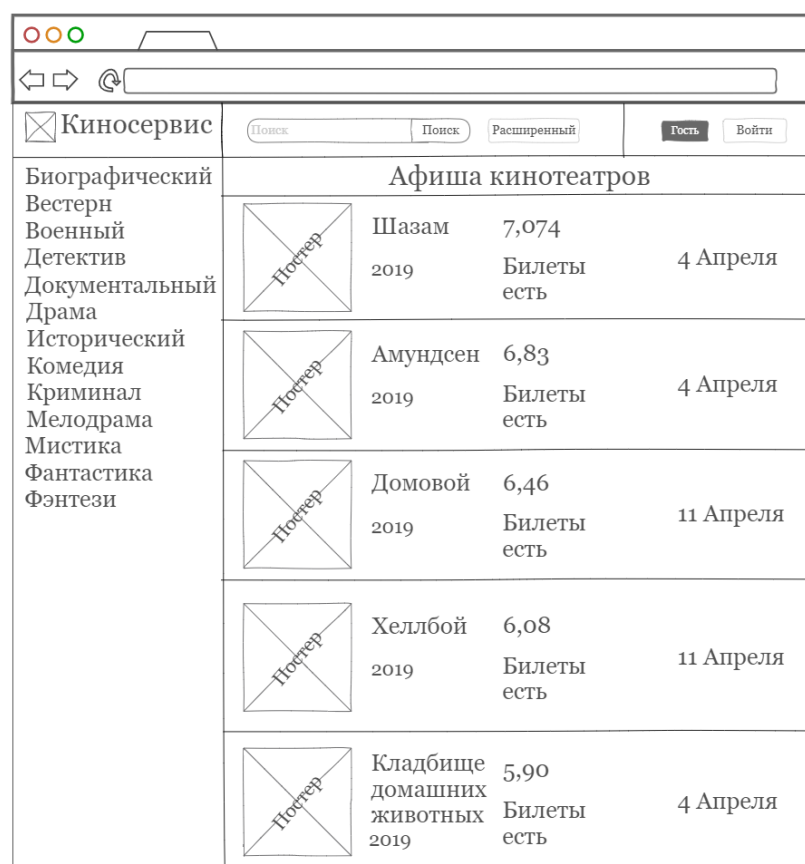


Рисунок 2.7 – Страница киноафиша

Если информация была изменена, то при нажатии на кнопку «Сохранить» новая информация записывается в базу данных. При нажатии на ссылку «Отмена» пользователь возвращается на станицу со списком людей.

Для получения доступа к основным ресурсам сайта пользователь должен авторизоваться.

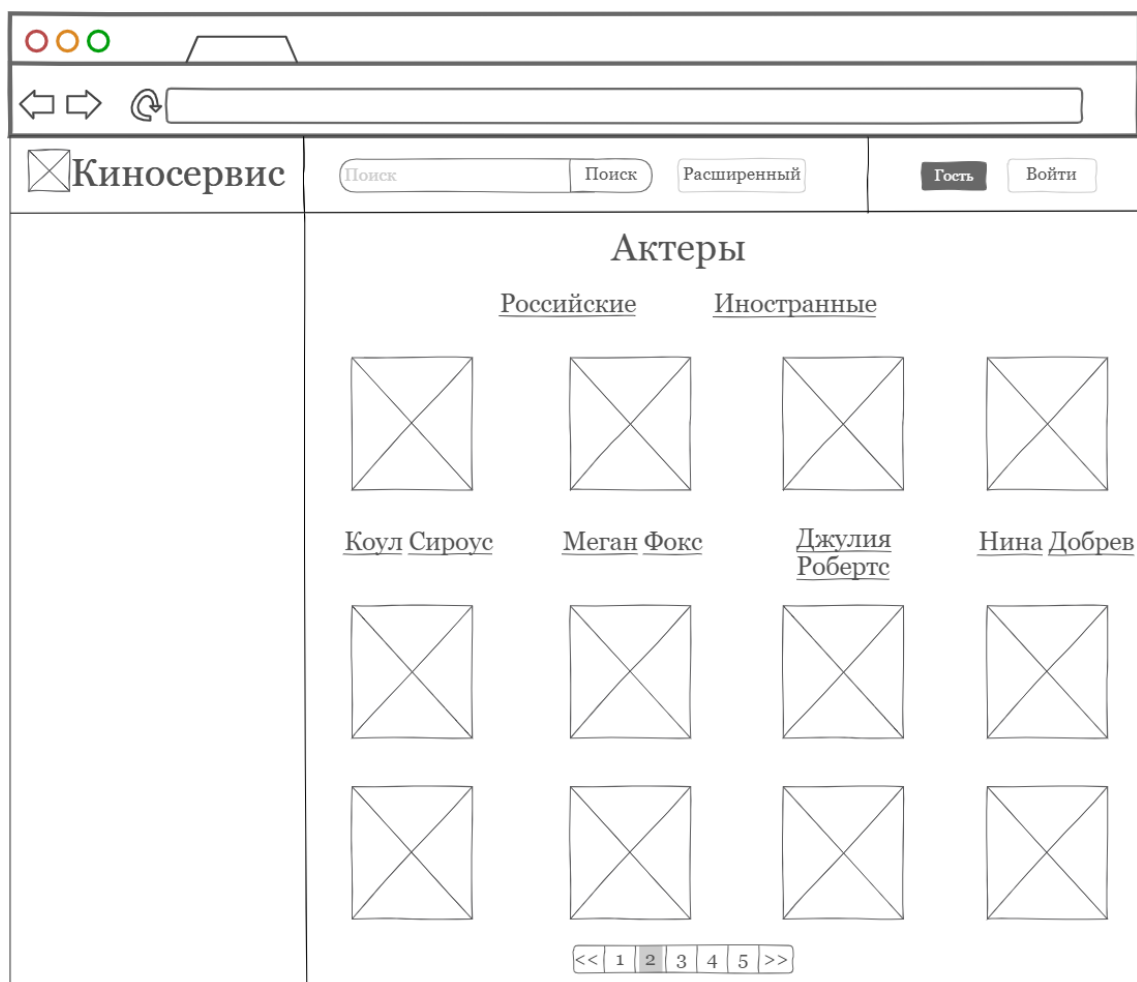


Рисунок 2.8 – Страница актеры

Прототипы остальных страниц представлены на рисунках 2.9-2.12.

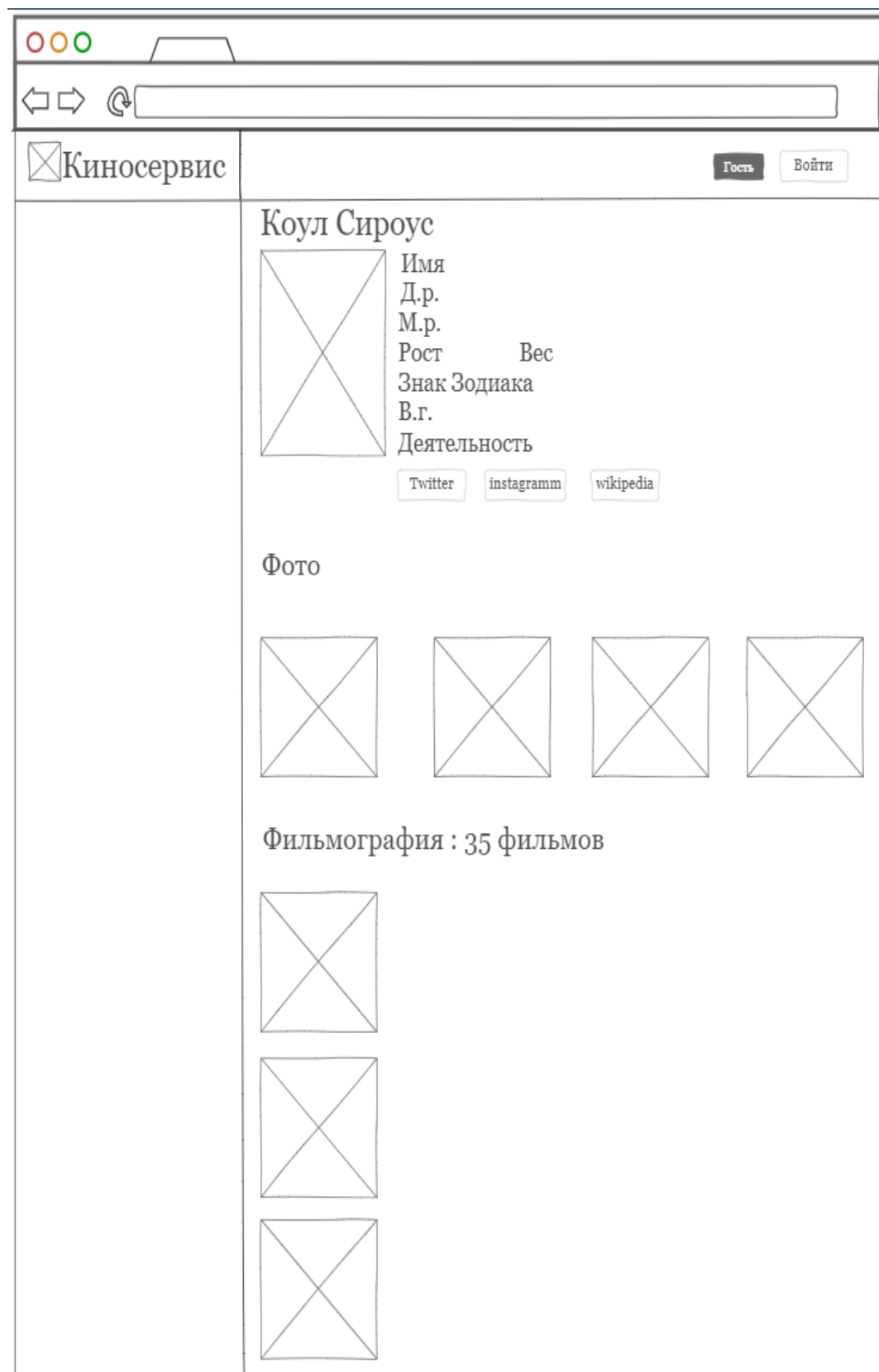


Рисунок 2.9 – Страница со списком фильмов для пользователя

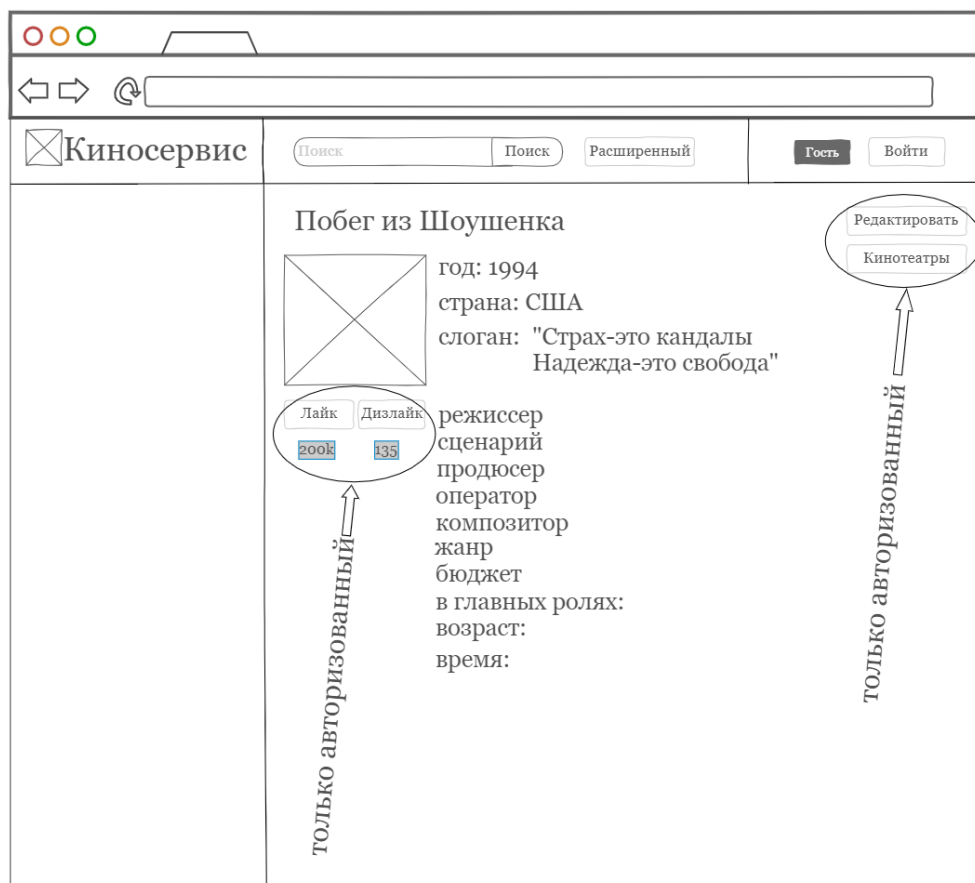


Рисунок 2.11 – Страница детальной информации о фильме для пользователя

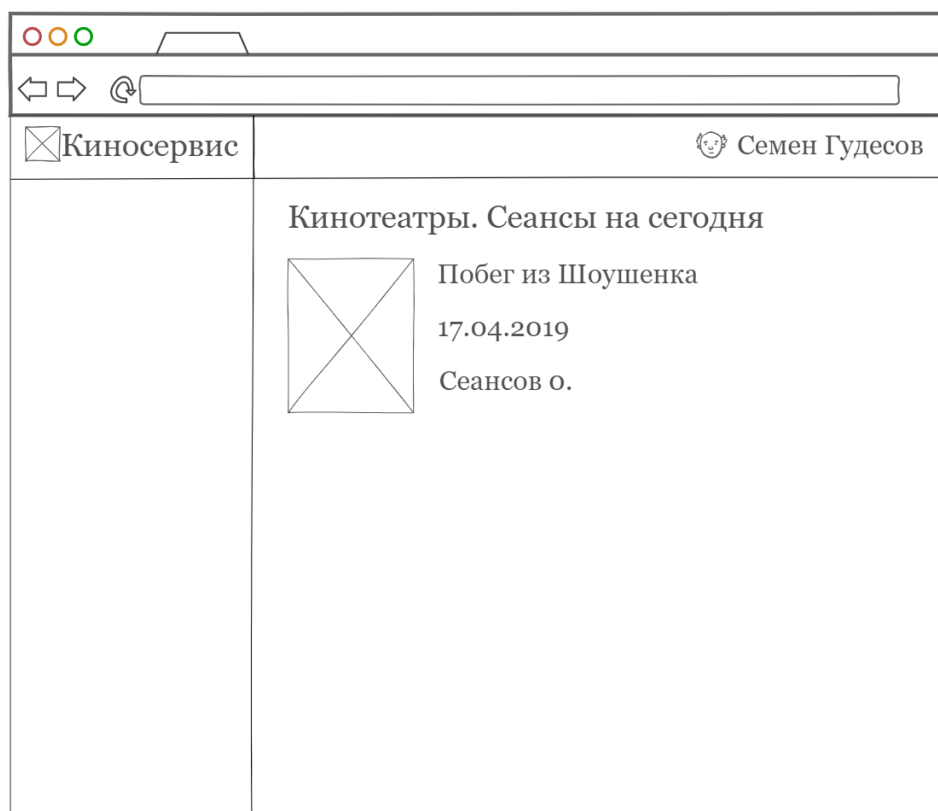


Рисунок 2.12 – Страница детальной информации о фильме для администратора

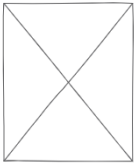
<div> ⏮ ⏭ 🔄 <input type="text"/> </div>	
☒ Киносервис	<div>  Семен Гудесов </div>
	Кинотеатры. Сеансы на сегодня <div>  <div> Хеллбой 17.04.2019 </div> </div>
	Высшая лига <div> <input type="text" value="14:15"/> <input type="text" value="17:55"/> <input type="text" value="20:05"/> <input type="text" value="00:05"/> </div> ул. Московская, 37, ТРЦ "Высшая лига"
	Роликс <div> <input type="text" value="14:15"/> <input type="text" value="18:00"/> <input type="text" value="22:00"/> </div> ул. Суворова
	Современник <div> <input type="text" value="15:30"/> <input type="text" value="19:50"/> <input type="text" value="22:30"/> </div>
	Берлин Синема <div> <input type="text" value="19:45"/> <input type="text" value="17:55"/> <input type="text" value="23:45"/> <input type="text" value="00:05"/> </div> проспект Строителей, 152Б, ТРЦ "Берлин"

Рисунок 2.13 – Верхняя часть страницы для изменения информации о фильме

2.3 Разработка программного кода

Сайт был разработан по технологии PHP. MVC3 и на языке программирования C# с использованием языка JavaScript. Эта технология была выбрана, потому что она обладает рядом преимуществ:

- архитектура MVC;
- расширяемость;
- жесткий контроль над HTML и HTTP;
- тестируемость;
- мощная система маршрутизации;
- современный API-интерфейс;
- открытый исходный код.

Диаграмма классов иллюстрирует спецификации программных классов и интерфейсов в приложении.

Расширенный поиск

Название фильма :

Начиная с (года) :

До (года) :

Страны производители :

☐ США
 ☐ Россия
 ☐ Бельгия

Жанры фильмов :

☐ биографический	☐ боевик	☐ вестерн
☐ военный	☐ детектив	☐ документальный
☐ драма	☐ исторический	☐ комедия
☐ криминал	☐ мелодрама	☐ мистика
☐ научный	☐ приключения	☐ спорт
☐ триллер	☐ ужасы	☐ фантастика
☐ фэнтези		

Люди принимавшие участие в фильме :

☐ Б
 ☐ Д
 ☐ К
 ☐ М
 ☐ Р
 ☐ Т

☐ Брэд Питт

Жадный поиск :

☐

Рисунок 2.14 – Нижняя часть страницы для изменения информации о фильме

Страны производители :

☐ США ☐ Россия ☐ Бельгия

Жанры фильмов :

☐ биографический	☐ боевик	☐ вестерн
☐ военный	☐ детектив	☐ документальный
☐ драма	☐ исторический	☐ комедия
☐ криминал	☐ мелодрама	☐ мистика
☐ научный	☐ приключения	☐ спорт
☐ триллер	☐ ужасы	☐ фантастика
☐ фэнтези		

Люди принимавшие участие в фильме :

Б Д К М Р Т

☐ Брэд Питт

Рисунок 2.15 – Страница расширенного поиска

Обычно на такую диаграмму выносятся следующая информация:

- классы, ассоциации и атрибуты;
- интерфейсы со своими операциями и константами;
- методы;
- информация о типах атрибутов;
- способы навигации;
- зависимости.

В отличие от диаграммы классов из модели предметной области, диаграммы классов проектирования отображают определения программных сущностей, а не понятия предметной области [12].

Диаграмма классов разрабатываемой системы представлена на рисунке 2.16.

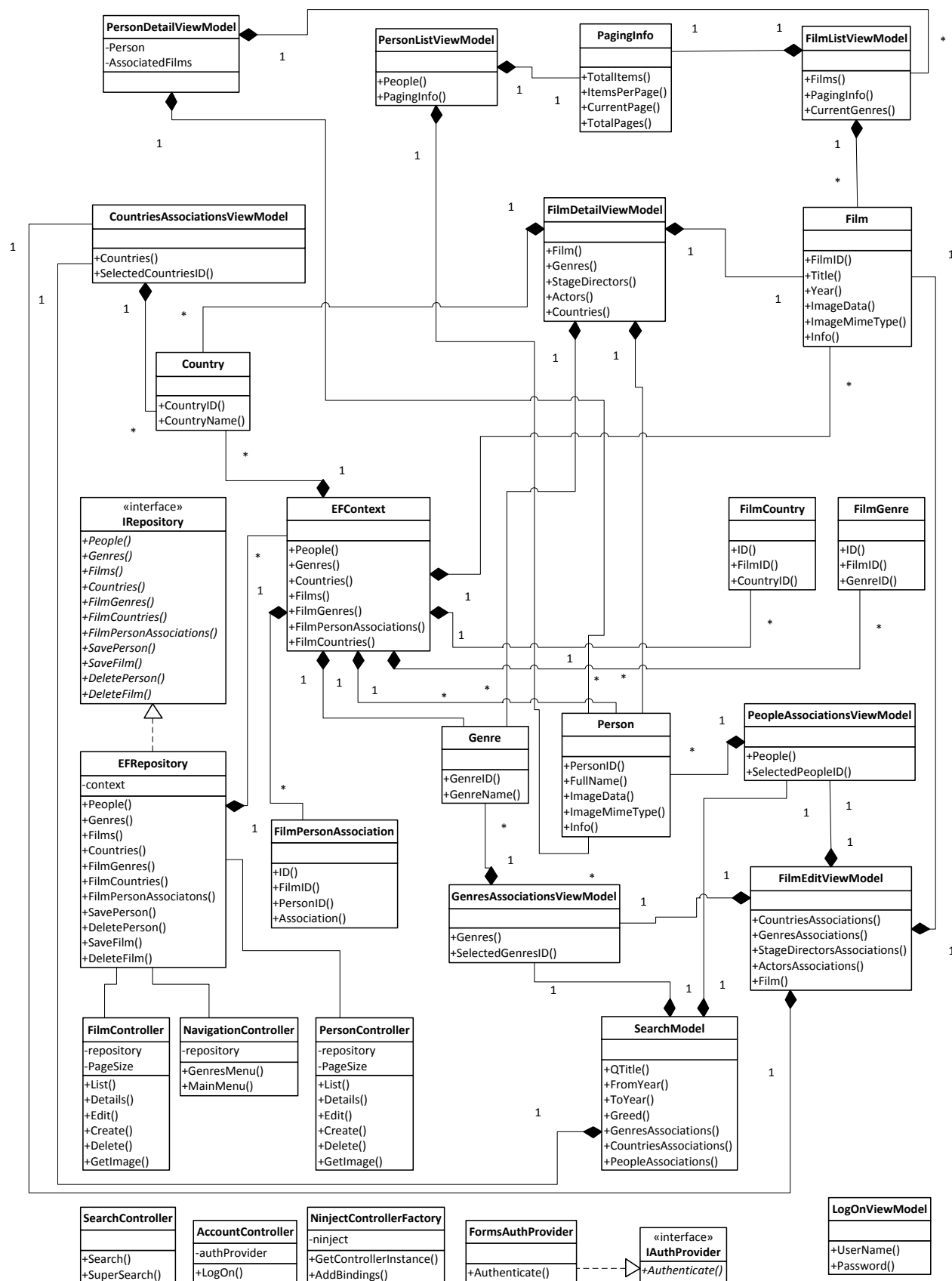


Рисунок 2.16 – Диаграмма классов

Таблица 2.12 – Описание классов

Название	Описание
EFContext – класс для взаимодействия сайта с базой данных	
Свойства класса:	
People	Свойство связано с таблицей People в базе данных. Данное свойство доступно только для чтения.
Genres	Свойство связано с таблицей Genres в базе данных. Данное свойство доступно только для чтения.
Countries	Свойство связано с таблицей Countries в базе данных. Данное свойство доступно только для чтения.
Films	Свойство связано с таблицей Films в базе данных. Данное свойство доступно только для чтения.
FilmGenres	Свойство связано с таблицей FilmGenres в базе данных. Данное свойство доступно только для чтения.
FilmCountries	Свойство связано с таблицей FilmCountries в базе данных. Данное свойство доступно только для чтения.
FilmPersonAssociations	Свойство связано с таблицей FilmPersonAssociations в базе данных. Данное свойство доступно только для чтения.
EFRepository – класс хранилище данных, предоставляет пользователю доступ к данным и операции над этими данными.	
Поля класса:	
Context	Поле, связанное с определенной базой данных.
Свойства класса:	
People	Возвращает список актеров и режиссеров, хранящихся в базе данных. Данное свойство доступно только для чтения.
Genres	Возвращает список всех жанров, хранящихся в базе данных. Данное свойство доступно только для чтения.
Countries	Возвращает список всех стран производителей, хранящихся в базе данных. Данное свойство доступно только для чтения.
Films	Возвращает список всех фильмов, хранящихся в базе данных. Данное свойство доступно только для чтения.
FilmGenres	Возвращает список ассоциаций фильмов и жанров. Данное свойство доступно только для

	чтения.
Название	Описание
FilmCountries	Возвращает список ассоциаций фильмов и стран производителей. Данное свойство доступно только для чтения.
FilmPersonAssociations	Возвращает список ассоциаций фильмов и людей, принимающих участие в этом фильме. Данное свойство доступно только для чтения.
Методы класса:	
SavePerson	Метод для сохранения данных о человеке в базе.
SaveFilm	Метод для сохранения данных о фильме в базе.
DeletePerson	Метод для удаления данных о человеке из базы.
DeleteFilm	Метод для удаления данных о фильме из базы.
FilmController – класс контроллер, обрабатывающие запросы пользователя для получения данных о фильмах.	
Поля класса:	
repository	Поле для доступа к хранилищу данных.
PageSize	Поле значение, которого показывает, сколько элементов должно быть в списке на одной странице.
Методы класса:	
List	Метод, перенаправляющий пользователя на страницу со список фильмов. Данные для заполнения страницы также предоставляются этим методом.
Details	Метод, перенаправляющий пользователя на страницу с детальной информацией о фильме. Данные для заполнения страницы также предоставляются этим методом.
Edit	Метод, перенаправляющий пользователя на страницу для изменения информации о фильме. Данные для заполнения страницы также предоставляются этим методом. Данный метод доступен только администратору.
Create	Метод, перенаправляющий пользователя на страницу для заполнения данных о новом фильме. Данные для заполнения страницы также предоставляются этим методом. Данный метод доступен только администратору.
Delete	Метод для удаления фильма из базы. Данный метод доступен только администратору.
GetImage	Метод, возвращающий постер к фильму.
NavigationController – класс контроллер отвечающий за меню сайта	

Продолжение таблицы 2.12

Название	Описание
Поля класса:	
Repository	Поле для доступа к хранилищу данных.
Методы класса:	
GenresMenu	Метод, отвечающий за создание меню по жанрам.
MainMenu	Метод, отвечающий за создание главного меню сайта.
PersonController – класс контроллер, обрабатывающие запросы пользователя для получения данных об актерах и режиссерах.	
Поля класса:	
Repository	Поле для доступа к хранилищу данных.
PageSize	Поле значение, которого показывает, сколько элементов должно быть в списке на одной странице.
Методы класса:	
List	Метод, перенаправляющий пользователя на страницу со список людей. Данные для заполнения страницы также предоставляются этим методом.
Details	Метод, перенаправляющий пользователя на страницу с детальной информацией о человеке. Данные для заполнения страницы также предоставляются этим методом.
Edit	Метод, перенаправляющий пользователя на страницу для изменения информации о человеке. Данные для заполнения страницы также предоставляются этим методом. Данный метод доступен только администратору.
Create	Метод, перенаправляющий пользователя на страницу для заполнения данных о новом актере или режиссере. Данные для заполнения страницы также предоставляются этим методом. Данный метод доступен только администратору.
Delete	Метод для удаления человека из базы. Данный метод доступен только администратору.
SearchController – класс контроллер для управления поиском.	
Методы класса:	
Search	Метод, отвечающий за создание элемента поиска на сайте.
SuperSearch	Метод, отвечающий за перенаправление пользователя на страницу с расширенным

	поиском.
Название	Описание
AccountController – класс контроллер отвечающий за авторизацию пользователя в системе.	
Поля класса:	
authProvider	Поле, указывающее каким способом происходит авторизация на сайте.
Методы класса:	
LogOn	Метод, отвечающий за проверку данных пользователя для доступа к системе.
FormAuthProvider – класс для авторизации с помощью форм.	
Методы класса:	
Authenticate	Метод для аутентификации в системе.
LogOnViewModel – класс модель данных для авторизации.	
Свойства класса:	
UserName	Имя пользователя в системе.
Password	Пароль пользователя в системе.
Genre – класс, описывающий жанр фильмов.	
Свойства класса:	
GenreID	ID жанра.
GenreName	Название жанра.
Country – класс, описывающий страны производителей фильмов.	
Свойства класса:	
CountryID	ID страны.
CountryName	Название страны.
Person – класс, описывающий людей, принимающих участие в фильмах.	
Свойства класса:	
PersonID	ID человека.
FullName	Полное имя человека.
ImageData	Изображение человека.
ImageMimeType	Расширение изображения.
Info	Дополнительная информация о человеке.
Film – класс, содержащий информацию о фильме.	
Свойства класса:	
FilmID	ID фильма.
Title	Название фильма.
Year	Год выпуска фильма.
ImageData	Постер к фильму.
ImageMimeType	Расширение изображения.
Info	Описание сюжета фильма.
FilmPersonAssociation – класс, описывающий ассоциации людей с фильмами.	
Свойства класса:	
Название	Описание

ID	ID ассоциации.
FilmID	ID фильма.
PersonID	ID человека.
Association	Кем является человек для фильма.
FilmCountry – класс, описывающий ассоциации стран производителей с фильмами.	
ID	ID ассоциации.
FilmID	ID фильма.
CountryID	ID страны.
FilmGenre – класс, описывающий ассоциации фильмов с жанрами.	
ID	ID ассоциации.
FilmID	ID фильма.
GenreID	ID жанра.
PagingInfo – класс, описывающий информацию о странице.	
Свойства класса:	
TotalItems	Всего элементов для вывода.
ItemsPerPage	Количество элементов для вывода на одной странице.
CurrentPage	Текущая страница.
TotalPages	Всего страниц.
PersonListViewModel – класс модель данных для вывода списка актеров и режиссеров.	
Свойства класса:	
People	Список людей для вывода.
PagingInfo	Информация о страницах.
FilmListViewModel – класс модель данных для вывода списка актеров и режиссеров.	
Свойства класса:	
Films	Список фильмов для вывода.
PagingInfo	Информация о страницах.
PersonDetailViewModel – класс модель данных для вывода конкретной информации о актере или режиссере.	
Свойства класса:	
Person	Информация о человеке.
AssociatedFilms	Фильмы, в которых он принимал участие.
FilmDetailViewModel – класс модель данных для вывода конкретной информации о фильме.	
Свойства класса:	
Film	Информация о фильме.
Genres	Жанры фильма.
Название	Описание
Countries	Страны производители фильма.
Actors	Актеры в фильме.

StageDirectors	Режиссеры фильма.
CountriesAssociationViewModel – класс модель данных для создания ассоциаций со странами.	
Свойства класса:	
Countries	Все страны для выбора.
SelectedCountriesID	ID всех, выбранных, стран.
PeopleAssociationViewModel – класс модель данных для создания ассоциаций со странами.	
Свойства класса:	
People	Все люди для выбора.
SelectedPeopleID	ID всех, выбранных, людей.
GenresAssociationViewModel – класс модель данных для создания ассоциаций с жанрами.	
Свойства класса:	
Genres	Все жанры для выбора.
SelectedGenresID	ID всех, выбранных, жанров.
FilmEditViewModel – класс модель данных для изменения информации о фильме.	
Свойства класса:	
Film	Ссылка на информацию о фильме.
GenresAssociations	Ассоциации фильма с жанрами.
ActorsAssociations	Ассоциации фильма с актерами.
StageDirectorsAssociations	Ассоциации фильма с режиссерами.
CountriesAssociations	Ассоциации фильма со странами производителями.
SearchModel – класс модель данных для поиска фильма.	
Свойства класса:	
QTitle	Название фильма для поиска.
FromYear	Фильмы, начиная с какого года выпуска искать.
ToYear	Фильмы, до какого года искать.
Greed	Если равно true, то поиск идет по конкретному критерию. Если – false то ищутся все варианты, у которых совпадают хотя бы по одному критерию.
GenresAssociations	Ассоциации фильма с жанрами.
ActorsAssociations	Ассоциации фильма с актерами.
StageDirectorsAssociations	Ассоциации фильма с режиссерами.
NinjectControllerFactory – класс фабрика для создания контроллеров.	
Название	Описание
Поля класса:	
ninject	Ссылка на ядро системы для внесения зависимости.

Методы класса:	
GetControllerInstance	Метод, возвращающий ссылки на контроллеры по имени контроллера.
AddBindings	Метод, внедряющий зависимости интерфейса и его реализации в системе.

Один из вариантов показать взаимодействие объектов – воспользоваться диаграммой последовательности. На ней обмен объектами сообщениями представлен во времени более точно и наглядно. Актер обычно располагается в левом верхнем углу диаграммы [13].

Диаграмма последовательности для сохранения человека в базе представлена на рисунке 2.17.

Сначала данные о человеке поступают в хранилище. Если в хранилище уже есть человек с данным ID, то его данные изменяются и сохраняются, иначе в базу заносится информация о новом человеке.

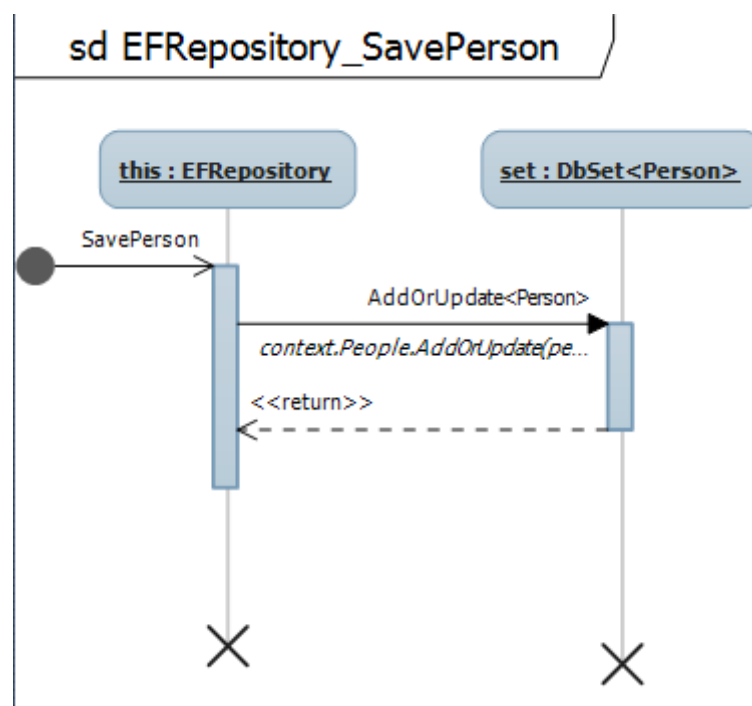


Рисунок 2.17 – Диаграмма последовательности

Текст программы приведен в приложении А.

Диаграммы компонентов следует применять, когда система разделяется на компоненты и надо показать их взаимоотношение посредством интерфейсов или схему компонентов в низкоуровневой структуре систем

Диаграмма компонентов данного приложения представлена на рисунке 2.18.

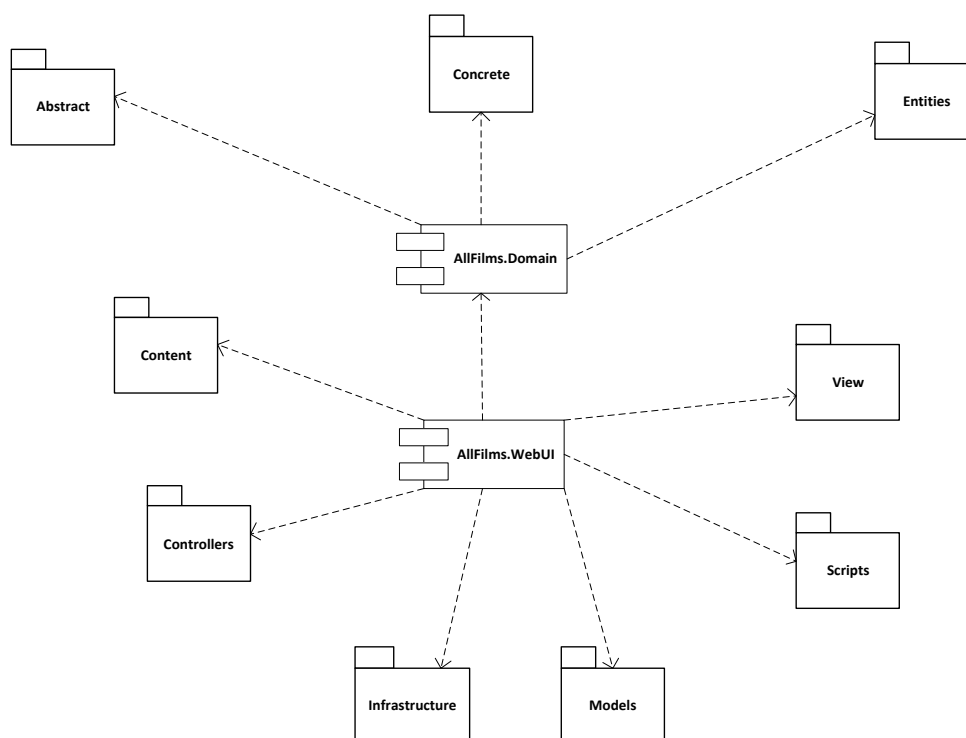


Рисунок 2.18 –Диаграмма компонентов

В папке Abstract хранятся все интерфейсы, для доступа к данным. В папке Concrete хранятся реализации этих интерфейсов. В папке Entities хранятся классы, описывающие модели данных в таблицах базы данных. В папке Content, хранятся файлы каскадных страниц стилей для дизайна сайта. В папке Scripts хранятся файлы скриптов, для валидации вводимых данных на сайте. В папке Controller хранятся файлы с классами контроллерами. В папке View хранятся файлы самих страничек сайта или частей страниц сайта. В папке Models хранятся файлы классов моделей данных, передающихся от контроллеров к представлениям. В папке Infrastructure хранятся файлы классов для авторизации в системе.

3 Тестирование и развертка программного обеспечения

3.1 Тестирование программного обеспечения

Качество программного продукта характеризуется набором его свойств, определяющих это с точки зрения заинтересованных сторон. Каждый из участников может иметь различное представление о продукте, и о том, насколько высоко качество продукта. Таким образом, постановка задачи обеспечения качества продукта выливается в задачу определения заинтересованных лиц, их критериев качества и затем – нахождения оптимального решения, удовлетворяющего этим критериям. Тестирование является одним из наиболее устоявшихся способов обеспечения качества разработки программного обеспечения [14].

Таблица 2.13 – План тестирования

Название теста	Действие	Ожидаемый результат	Полученный результат
Получение списка актеров и режиссеров	Пользователь нажимает на ссылку «Актеры и режиссеры»	Пользователь переходит на страницу со списком актеров и режиссеров	Соответствует ожидаемым результатам (рисунок 22)
Получение списка фильмов	Пользователь нажимает на ссылку «Фильмы»	Пользователь переходит на страницу со списком фильмов	Соответствует ожидаемым результатам (рисунок 23)
Ввод неправильных данных для входа в систему	Пользователь вводит неправильный логин или пароль в форму входа	Отказ доступа и вывод сообщения об ошибке	Соответствует ожидаемым результатам (рисунок 24)
Ввод правильных данных для входа в систему	Пользователь вводит правильные логин и пароль в форму входа	Получение доступа к возможностям администратора	Соответствует ожидаемым результатам
Получение детальной информации о	Пользователь нажимает на ссылку с именем	Получение детальной информации о	Соответствует ожидаемым результатам

человеке	актера	человеке	(рисунок 25)
Название теста	Действие	Ожидаемый результат	Полученный результат
Получение детальной информации о фильме	Пользователь нажимает на ссылку с название фильма	Получение детальной информации о фильме	Соответствует ожидаемым результатам (рисунок 26)
Добавление нового человека в базу	Администратор нажимает на ссылку «добавить человека», вводит информацию и нажимает сохранить	Информация о человеке добавлена в базу	Соответствует ожидаемым результатам (рисунки 27-28)
Добавление нового фильма в базу	Администратор нажимает на ссылку «добавить фильм», вводит информацию и нажимает сохранить	Информация о фильме добавлена в базу	Соответствует ожидаемым результатам
Удаление человека из базы	Администратор выбирает человека из списка и нажимает на кнопку «Delete»	Человек удален из базы	Соответствует ожидаемым результатам (рисунок 29)
Удаление фильма из базы	Администратор выбирает фильм из списка и нажимает на кнопку «Delete»	Фильм удален из базы	Соответствует ожидаемым результатам
Поиск фильма по названию	Пользователь вводит название фильма (в тесте «мстит») в форму поиска и нажимает на кнопку «Найти»	Пользователь перенаправляется на страницу со списком всех возможных фильмов	Соответствует ожидаемым результатам (рисунок 30)

Добавление незаполненной информации о человеке в базу	Администратор нажимает на ссылку «добавить человека» и нажимает сохранить без ввода информации	Администратор получает ошибку о том, что данные не заполнены полностью	Соответствует ожидаемым результатам (рисунок 31)
Добавление незаполненной информации о фильме в базу	Администратор нажимает на ссылку «добавить фильм» и нажимает сохранить без ввода информации	Администратор получает ошибку о том, что данные не заполнены полностью	Соответствует ожидаемым результатам
Получение списка фильмов определенного жанра	Пользователь нажимает на ссылку с названием жанра (в тесте «боевик»)	Пользователь переходит на страницу со списком всех фильмов данного жанра	Соответствует ожидаемым результатам (рисунок 32)
Расширенный поиск	Пользователь нажимает на ссылку поиск, выбирает критерии поиска и нажимает на кнопку «Найти» (в тесте Страна = США, Жанр = Боевик и Триллер)	Пользователь переходит на страницу со списком всех фильмов, удовлетворяющих критерию поиска	Соответствует ожидаемым результатам (рисунок 33)



Рисунок 2.19 – Список актеров и режиссеров (Пользователь)



Рисунок 2.20 – Список актеров и режиссеров (Гость)

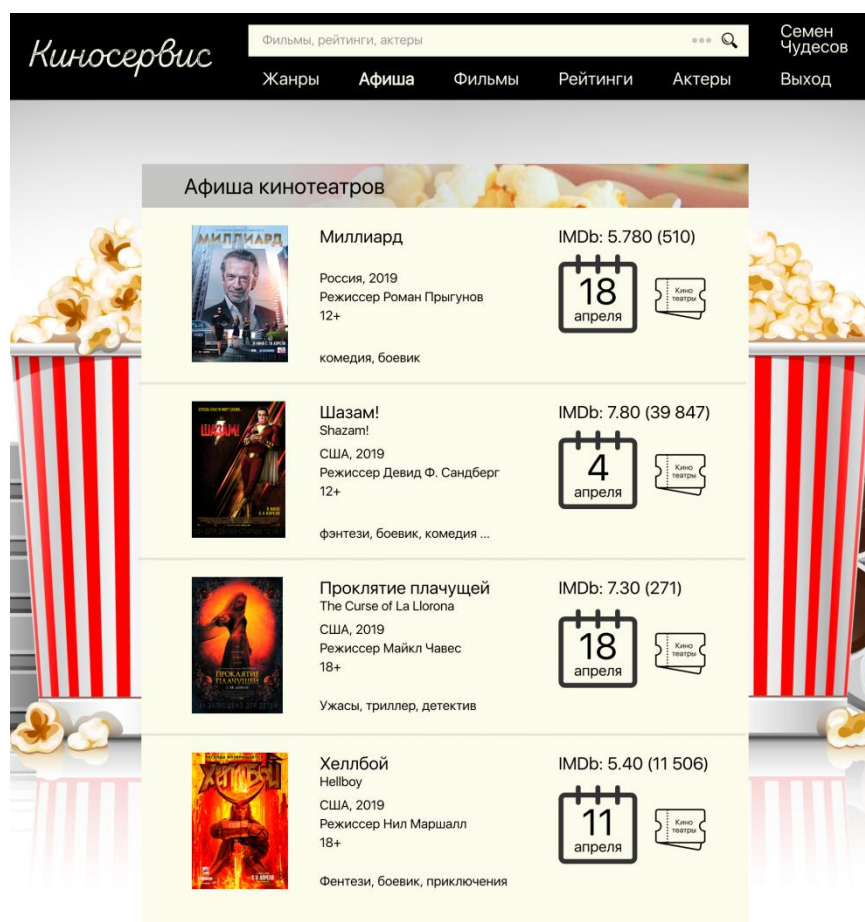


Рисунок 2.21 – Афиша кинотеатров (Пользователь)

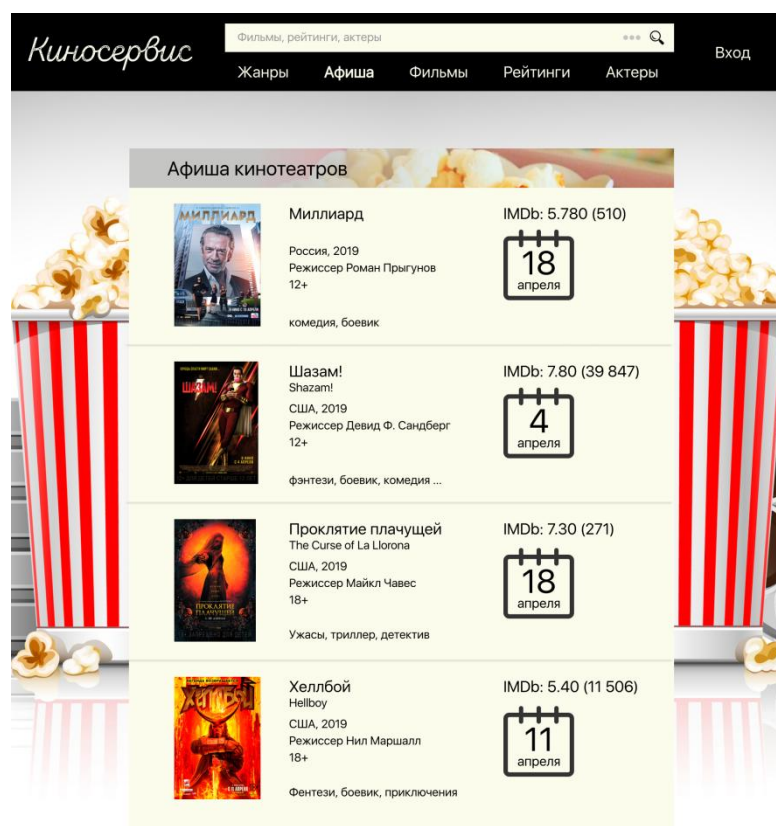


Рисунок 2.22 – Афиша кинотеатров (Гость)

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.05.001 ПЗ

Лист

52

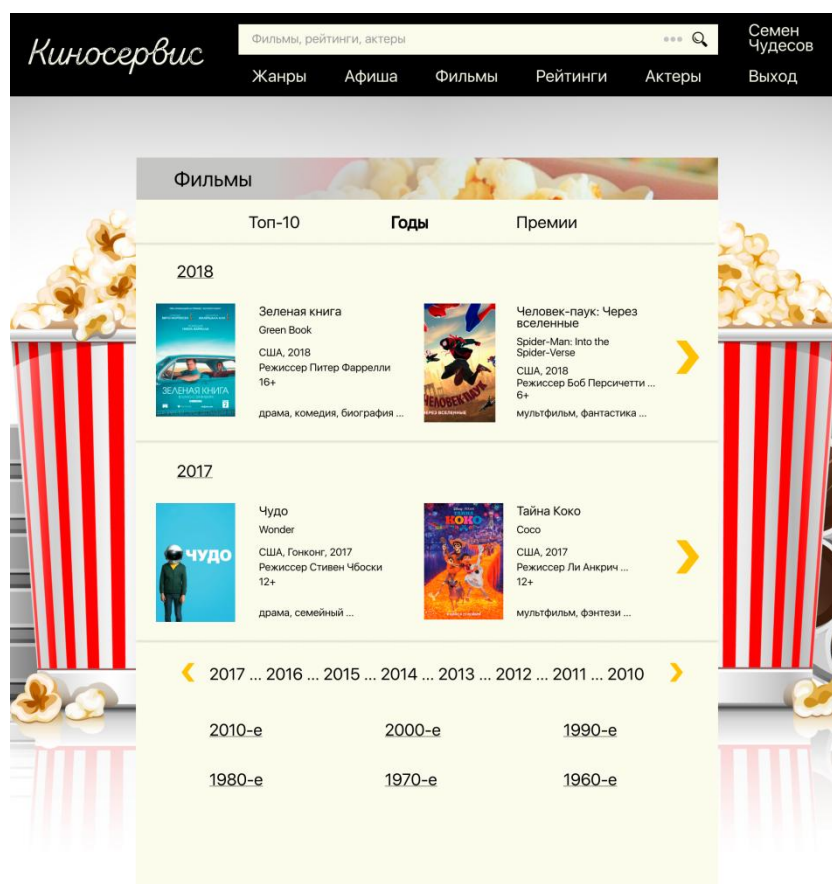


Рисунок 2.23 – Список фильмов по годам (Пользователь)

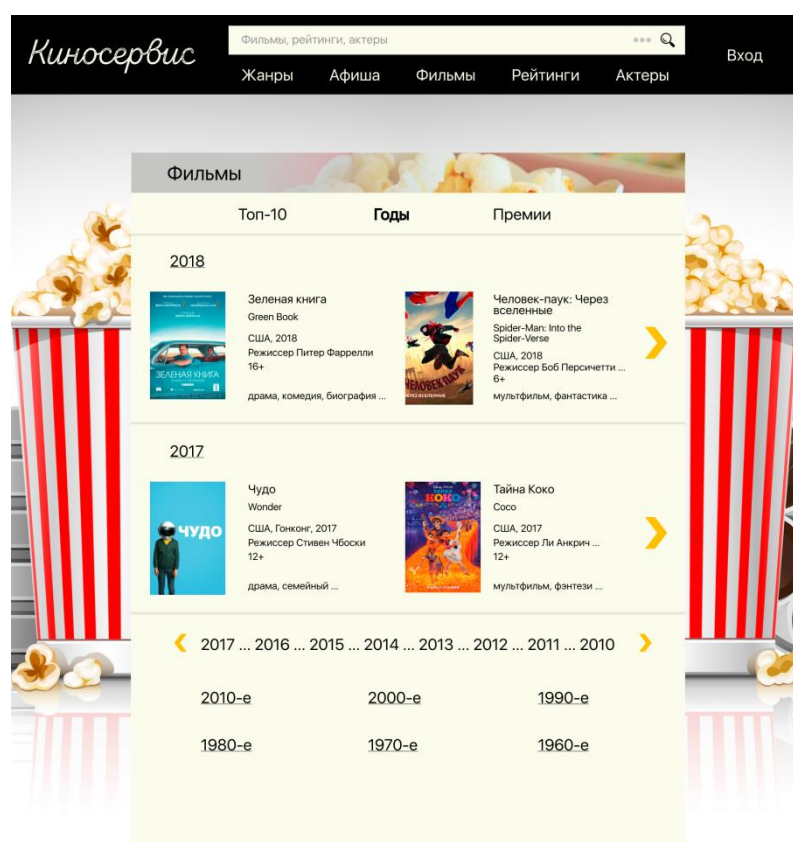


Рисунок 2.24 – Список фильмов по годам (Гость)

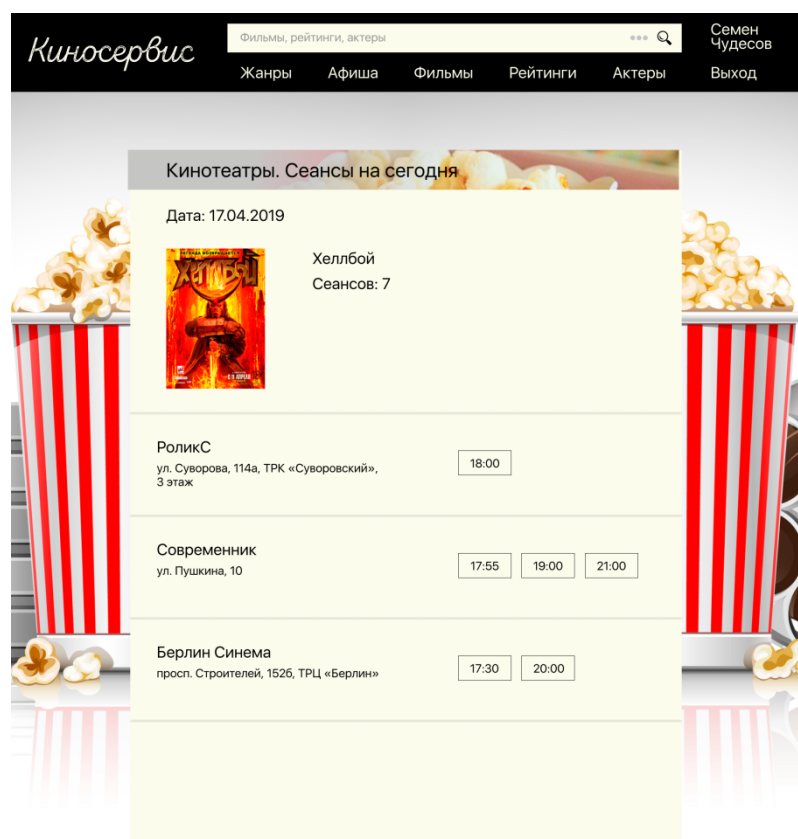


Рисунок 2.25 – Сеансы кинофильмов . Вариант 1



Рисунок 2.26 – Сеансы кинофильмов. Вариант 2

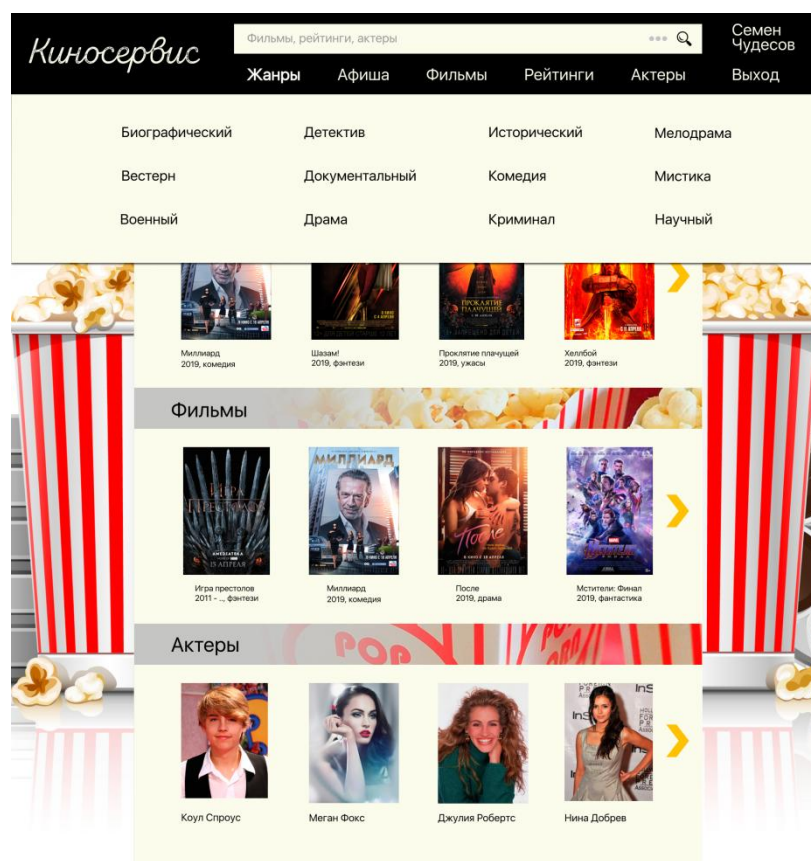


Рисунок 2.27 – Основная жанры

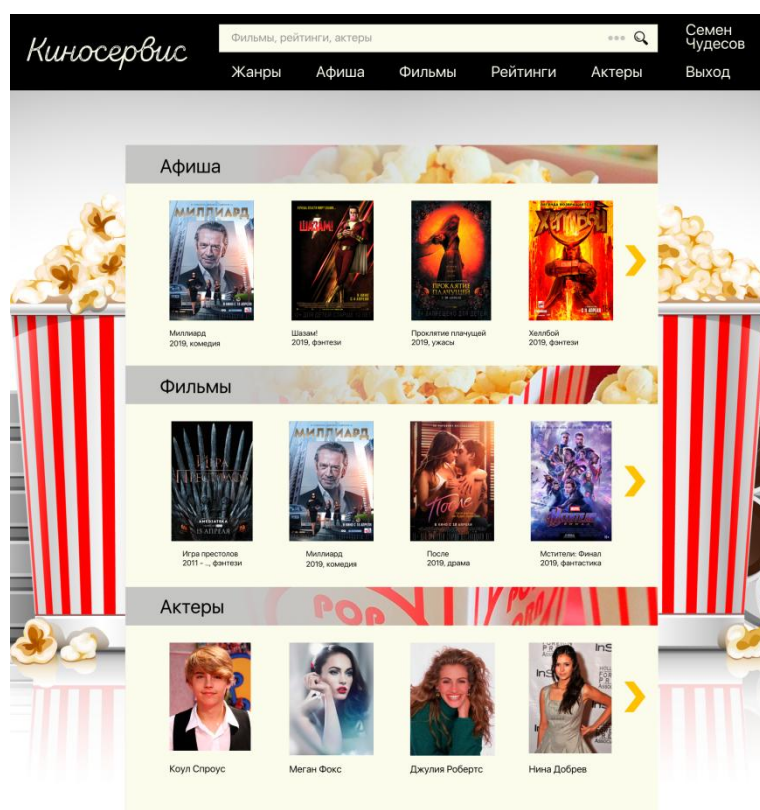


Рисунок 2.28 – Основная пользователи

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.05.001 ПЗ

Лист

55

Киносервис Фильмы, рейтинги, актеры *** 🔍 Семен Чудесов

Жанры Афиша Фильмы Рейтинги Актеры Выход

Расширенный поиск

Искать фильм:

Название фильма целиком, или частично

Страна Жанр Год

Искать фильм по создателям:

Актер

Режиссер

Добавить

Искать фильм по создателям:

Имя, фамилия

Год Пол

Место рождения

Фильм Амплуа Что искать

Поиск

Рисунок 2.29 – Расширенный поиск



Рисунок 2.30 – Рейтинг фильмов

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.05.001 ПЗ

Лист

56



Рисунок 2.31 – Страница актера (Пользователь)



Рисунок 2.32 – Страница актера (Гость)

Тестирование программного обеспечения представляет собой процесс выявления наличия дефектов в программных системах. Тестирование означает попытки обнаружения дефектов и вносит существенный вклад в защиту пользователей от отказов программного обеспечения [15].

На этапе тестирования были выявлены ошибки в работе программного обеспечения, которых в процессе дальнейшей работы были устранены. На основании проведенных тестов можно сказать, что сайт работает корректно.

3.2 Развертывание программного обеспечения

Диаграммы развертывания представляют физическое расположение системы, показывая на каком физическом оборудовании, запускается та или иная составляющая программного обеспечения [6].

Приложение использует клиент-серверную технологию. На сервере располагается сайт и база данных. За организацию канала на стороне пользователя отвечает браузер, функционирование канала на стороне сервера выполняет MS SQL Server и IIS Server. Диаграмма развертывания представлена на рисунке 2.33.

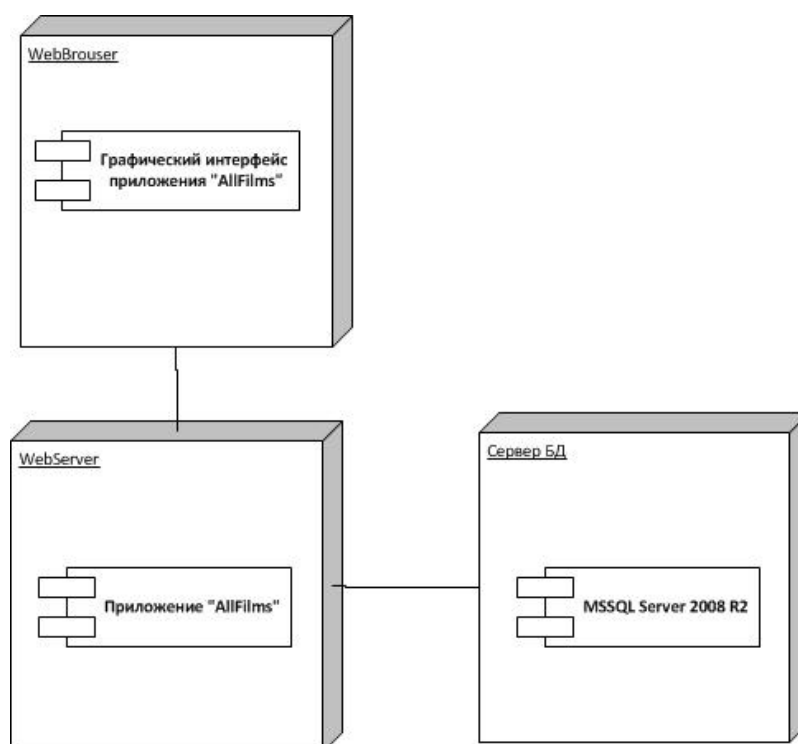


Рисунок 2.33 – Диаграмма развертывания

3.3 Руководство пользователя

Документ «Руководство пользователя» относится к пакету эксплуатационной документации. Основная цель руководства пользователя заключается в обеспечении пользователя необходимой информацией для самостоятельной работы с программой или автоматизированной системой [16].

Назначение системы. Основной целью сайта является предоставление пользователю информации о кинофильмах, а также удобный и расширенный поиск этой информации.

Условие применение системы. Требования к системе:

- доступ в интернет;
- наличие современного браузера;
- включенная поддержка JavaScript в браузере.

Подготовка к работе системы. Для запуска сайта необходимо скопировать все файлы на сервер, подключить базу данных и подключить домен для сайта. После этого сайта будет доступен пользователям через браузер.

Описание операций. Для доступа к сайту необходимо запустить браузер и ввести адрес сайта. Появится главная страница с выбором действий:

- фильмы – получение списка всех фильмов;
- актеры и режиссеры – получение списка всех актеров и режиссеров;
- поиск – переход на страницу расширенного поиска;
- выбор жанра – получение списка фильмов определенного жанра;
- вход – авторизация в системе для администратора.

При входе в систему у пользователя появляются дополнительные возможности:

- добавить человека – добавление информации о человеке в базу;
- изменить информацию о человеке – изменение информации о человеке с сохранением в базу;
- удалить человека – удаление информации о человеке из базы;
- добавить фильм – добавление информации о фильме в базу;
- изменить информацию о фильме – изменение информации о фильме с сохранением в базу;
- удалить фильм – удаление информации о фильме из базы.

Аварийные ситуации. При отказе доступа к страницам сайта проверьте свое интернет соединение. При сбое в работе попробуйте перезапустить браузер.

4 Технология разработки синтаксического анализатора сайтов

Для реализации парсера сайта требуется реализовать инструмент работы с CURL, и для начала необходимо разобраться, что эта аббревиатура обозначает.

CURL – это программа командной строки, позволяющая общаться с серверами используя для этого различные протоколы, в случае реализации данного приложения HTTP и HTTPS. Для работы с CURL в PHP есть библиотека libcurl, функции которой были использованы для отправки запросов и получения ответов от сервера.

Далее требуется перейти на сайт донор, в данном случае это ресурс - <https://www.kinopoisk.ru>.

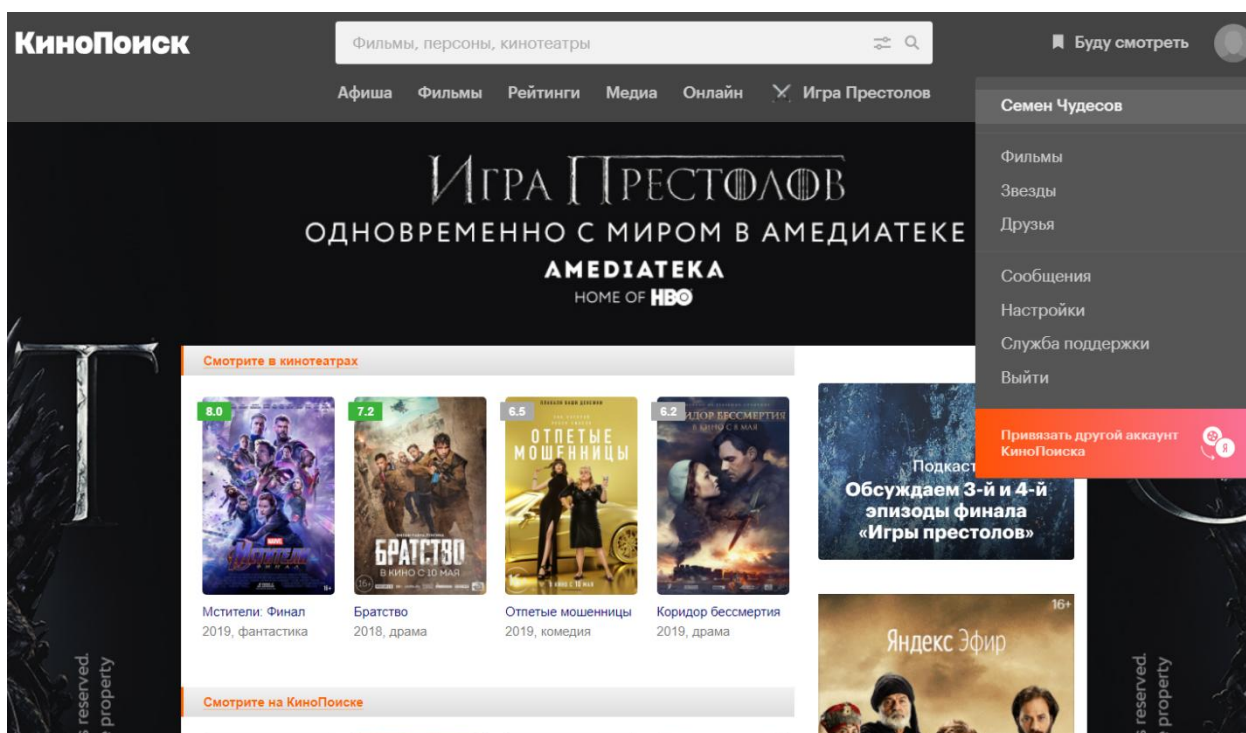


Рисунок 4.1 – Главная страница сайта «Кинопоиск»

Для того, чтобы спарсить категории этого сайта, необходимо перейти на страницы каталогов: Афиша, Фильмы, Рейтинги (рисунок 3.2).

Подкатегории находятся внутри отдельного элемента списка в таком же нумерованном списке.

Структура несложная, с отсутствием встроенных макросов и вложений третьего уровня, осталось только её получить. Конкретные кинокартины возьмем из раздела «Фильмы»:

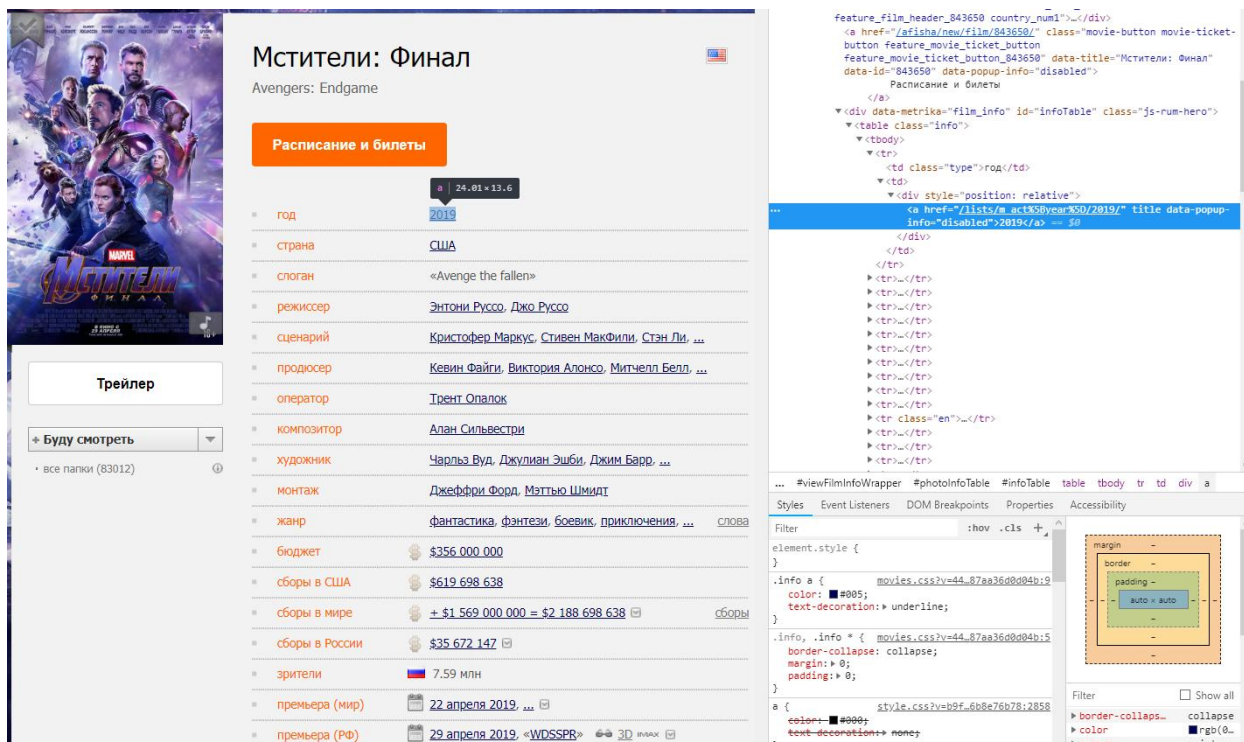


Рисунок 4.4 – Главная страница сайта «Кинопоиск»

На странице получается 5 фильмов, у каждого требуется загрузить: картинку, название, ссылку на информацию о фильме, характеристики и т.д..

Сам процесс и скрипт парсинга сайта состоит из двух частей:

1. Нужно получить HTML код страницы, которой необходим сайту производителю;
2. Разбор полученного кода с сохранением данных и дальнейшей обработки их (в данном случае будет использоваться phpQuery, установленный через метод composer).

Для решения первого пункта был написан класс с одним статическим методом, который будет оберткой над CURL. Так код можно будет

использовать в дальнейшем и, если необходимо, модифицировать его. Первое, с чем нужно определиться - как будет называться класс и метод и какие будут у него обязательные параметры:

```
class Parser{

    public static function getPage($params = []){

        if($params){

            if(!empty($params["url"])){

                $url = $params["url"];

                // Остальной код пишем тут

            }

        }

        return false;

    }

}
```

Основной метод, который будет использован в парсере –getPage() и у него всего один обязательный параметр URL страницы, которой будет разбираться.

Представленный выше метод будет использовать следующие функции:

- \$useragent –возможность устанавливать заголовок User-Agent, так сервис сможет делать обращения к серверу похожими на обращения из браузера;
- \$timeout – будет отвечать за время выполнения запроса на сервер;
- \$connecttimeout – так же важно указывать время ожидания соединения;

- \$head – если требуется проверить только заголовки, которые отдаёт сервер на запрос сервиса этот параметр просто будет необходим;
- \$cookie_file – тут всё просто: файл, в который будут записывать куки донора контента и при обращении передаваться;
- \$cookie_session – иногда может быть необходимо, запрещать передачу сессионных кук;
- \$proxy_ip – параметр IP прокси-сервера, сегодня спарсим несколько страниц, но если необходимо несколько тысяч, то без прокси это сделать будет невозможно;
- \$proxy_port – соответственно порт прокси-сервера;
- \$proxy_type – тип прокси CURLPROXY_HTTP, CURLPROXY_SOCKS4, CURLPROXY_SOCKS5, CURLPROXY_SOCKS4A или CURLPROXY_SOCKS5_HOSTNAME;
- \$headers – выше был указан параметр, отвечающий за заголовок User-Agent, но иногда нужно передать помимо его и другие, для это потребуется массив заголовков;
- \$post – метод для отправки POST запроса.

Конечно, обрабатываемых значений много и не всё будут использоваться для поставленной задачи, но разобрать их стоит, так как при парсинге больше одной страницы многое выше описанное пригодится для визуализации. И так добавим разбор параметров скрипт:

```
$useragent      = !empty($params["useragent"]) ? $params["useragent"] : "Mozilla/5.0
(Windows NT 6.3; W...) Gecko/20100101 Firefox/57.0";

$timeout        = !empty($params["timeout"]) ? $params["timeout"] : 5;
$connecttimeout = !empty($params["connecttimeout"]) ? $params["connecttimeout"] : 5;
$head           = !empty($params["head"]) ? $params["head"] : false;

$cookie_file    = !empty($params["cookie"]["file"]) ? $params["cookie"]["file"] : false;
$cookie_session = !empty($params["cookie"]["session"]) ? $params["cookie"]["session"]
: false;

$proxy_ip       = !empty($params["proxy"]["ip"]) ? $params["proxy"]["ip"] : false;
```

```

$proxy_port = !empty($params["proxy"]["port"]) ? $params["proxy"]["port"] : false;
$proxy_type = !empty($params["proxy"]["type"]) ? $params["proxy"]["type"] : false;

$headers = !empty($params["headers"]) ? $params["headers"] : false;

$post = !empty($params["post"]) ? $params["post"] : false;

```

Как видите, у всех параметров есть значения по умолчанию. Двигаемся дальше и следующей строчкой необходимо реализовать кусок кода, который будет очищать файл с куками при запросе:

```

if($cookie_file){

    file_put_contents(__DIR__."/". $cookie_file, "");
}

```

Так код обезопасит себя от ситуации, когда по какой-либо причине не создался файл.

Для работы с CURL необходимо вначале инициализировать сеанс, а по завершению работы его закрыть, также при работе важно учесть возможные ошибки, которые наверняка появятся, а при успешном получении ответа вернуть результат:

```

$ch = curl_init();

curl_setopt($ch, CURLOPT_URL, $url);

// Далее продолжаем кодить тут

curl_setopt($ch, CURLINFO_HEADER_OUT, true);

$content = curl_exec($ch);
$info      = curl_getinfo($ch);

$error = false;

```

```

if($content === false){

    $data = false;

    $error["message"] = curl_error($ch);
    $error["code"]      = self::$error_codes[
        curl_errno($ch)
    ];
}else{

    $data["content"] = $content;
    $data["info"]     = $info;
}

curl_close($ch);

return [
    "data"    => $data,
    "error" => $error
];

```

Ниже описана реализация статического свойства `$error_codes`, к которому обращается система. Фактически данное свойство работает с массивом и с расшифровкой кодов функции `curl_errno()`.

```

private static $error_codes = [
    "CURLE_UNSUPPORTED_PROTOCOL",
    "CURLE_FAILED_INIT",

    // Тут более 60 элементов, Приложении А

    "CURLE_FTP_BAD_FILE_LIST",
    "CURLE_CHUNK_FAILED"
];

```

После того, как произошла инициализация соединения через функцию `curl_setopt()`, нужно установить несколько параметров для текущего сеанса:

`CURLOPT_URL` – первый и обязательный - это адрес, на который мы обращаемся;

`CURLINFO_HEADER_OUT` – массив с информацией о текущем соединении.

Используя функцию `curl_exec()`, осуществляется непосредственно запрос при помощи `CURL`, а результат сохраняет переменную `$content`, по умолчанию после успешной отработки результат отобразиться на экране, а в `$content` будет установлен в статус `true`. Отследить попутную информацию при запросе поможет функция `curl_getinfo()`. Также важно, если произойдет ошибка - результат общения будет `false`, поэтому, ниже по коду используется строгое равенство с учетом типов. Осталось рассмотреть ещё две функции это `curl_error()` – вернёт сообщение об ошибке, и `curl_errno()` – код ошибки. Результатом работы метода `getPage()` будет массив, а чтобы его увидеть воспользуемся, а для теста сделаем запрос на сервис [httpbin](http://httpbin.org/ip) для получения своего IP.

Кстати очень удобный сервис, позволяющий отладить обращения к серверу. Так как, например, для того что бы узнать свой IP или заголовки отправляемые через `CURL`, нам бы пришлось реализовать функцию.

```
$html = Parser::getPage([  
    "url" => "http://httpbin.org/ip"  
]);
```

Если вывести на экран, то должна появиться следующая картина:

```
{ "origin": "37.214.14.173" }
```

```
Array
(
    [data] => Array
        (
            [content] => 1
            [info] => Array
                (
                    [url] => http://httpbin.org/ip
                    [content_type] => application/json
                    [http_code] => 200
                    [header_size] => 302
                    [request_size] => 52
                    [filetime] => -1
                    [ssl_verify_result] => 0
                    [redirect_count] => 0
                    [total_time] => 0.562
                    [namelookup_time] => 0.125
                    [connect_time] => 0.343
                    [pretransfer_time] => 0.343
                    [size_upload] => 0
                    [size_download] => 32
                    [speed_download] => 56
                    [speed_upload] => 0
                    [download_content_length] => 32
                    [upload_content_length] => -1
                    [starttransfer_time] => 0.562
                    [redirect_time] => 0
                    [redirect_url] =>
                    [primary_ip] => 50.16.227.190
                    [certinfo] => Array
                        (
                        )

                    [primary_port] => 80
                    [local_ip] => 192.168.1.2
                    [local_port] => 31295
                    [request_header] => GET /ip HTTP/1.1

                )
            )
        )
    [error] =>
)
```

Рисунок 4.5 Результат работы метода getPage()

Если произойдет ошибка, то результат будет выглядеть так:

Изм.	Лист	№ докум.	Подпись	Дата

```

Array
(
    [data] =>
    [error] => Array
        (
            [message] => Could not resolve host: ht11tpbin.org
            [code] => CURLE_COULDNT_CONNECT
        )
)

```

Рисунок 4.6 Обработка ошибки методом getPage()

При успешном запросе получаем заполненную ячейку массива data с контентом и информацией о запросе, при ошибке заполняется ячейка error. Из первого скриншота можно заметить первую неприятность, контент сохранился не в переменную, а отрисовался на странице. Чтобы решить это, нужно добавить ещё один параметр сеанса CURLOPT_RETURNTRANSFER.

```
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
```

Обращаясь к страницам, можем обнаружить, что они осуществляют редирект на другие, чтобы получить конечный результат добавляем:

```
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);
```

Теперь можно увидеть более приятную картину:

```

Array
(
    [data] => Array
        (
            [content] => {
                "origin": "37.214.14.173"
            }

            [info] => Array
                (
                    [url] => http://httpbin.org/ip
                    [content_type] => application/json
                    [http_code] => 200
                    [header_size] => 302
                    [request_size] => 52
                    [filetime] => -1
                    [ssl_verify_result] => 0
                    [redirect_count] => 0
                    [total_time] => 0.641
                    [namelookup_time] => 0.266
                    [connect_time] => 0.453
                    [pretransfer_time] => 0.453
                    [size_upload] => 0
                    [size_download] => 32
                    [speed_download] => 49
                    [speed_upload] => 0
                    [download_content_length] => 32
                    [upload_content_length] => -1
                    [starttransfer_time] => 0.641
                    [redirect_time] => 0
                    [redirect_url] =>
                    [primary_ip] => 54.243.65.67
                    [certinfo] => Array
                        (
                        )

                    [primary_port] => 80
                    [local_ip] => 192.168.1.2
                    [local_port] => 31345
                    [request_header] => GET /ip HTTP/1.1

                )

            [error] =>

        )

    )

```

Рисунок 4.7 Заполнение массива методом getPage()

Описав переменные \$useragent, \$timeout и \$connecttimeout, добавляем их в скрипт:

```
curl_setopt($ch, CURLOPT_USERAGENT, $useragent);
```

```
curl_setopt($ch, CURLOPT_TIMEOUT, $timeout);
curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, $connecttimeout);
```

Для того, чтобы получить заголовки ответа, потребуется добавить следующий код:

```
if($head){

    curl_setopt($ch, CURLOPT_HEADER, true);
    curl_setopt($ch, CURLOPT_NOBODY, true);
}
```

Ниже произведем отключение вывода тела документа и включение вывода шапки в результате:

```
Array
(
    [data] => Array
        (
            [content] => HTTP/1.1 200 OK
            Connection: keep-alive
            Server: meinheld/0.6.1
            Date: Thu, 01 Feb 2018 19:02:26 GMT
            Content-Type: application/json
            Access-Control-Allow-Origin: *
            Access-Control-Allow-Credentials: true
            X-Powered-By: Flask
            X-Processed-Time: 0.000501871109009
            Content-Length: 32
            Via: 1.1 vegur
        )

    [info] => Array
        (
            [url] => http://httpbin.org/ip
        )
    )
```

Рисунок 4.8 Выход заголовка страницы методом getPage()

Для работы со ссылками с SSL сертификатом, добавляем:

```
if(strpos($url, "https") !== false){

    curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, true);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, true);
}
```

Ниже описан код для сохранения cookies в парсере сайта донора. Одной из основных причин не запоминания кодом cookies может быть

указание относительного пути, поэтому стоит это учесть и написать следующие строки:

```
if($cookie_file){

    curl_setopt($ch, CURLOPT_COOKIEJAR, __DIR__."/". $cookie_file);
    curl_setopt($ch, CURLOPT_COOKIEFILE, __DIR__."/". $cookie_file);

    if($cookie_session){

        curl_setopt($ch, CURLOPT_COOKIESESSION, true);

    }

}
```

Предлагаю проверить, а для этого система попытается вытянуть куки с сайта «Кинопоиск»:

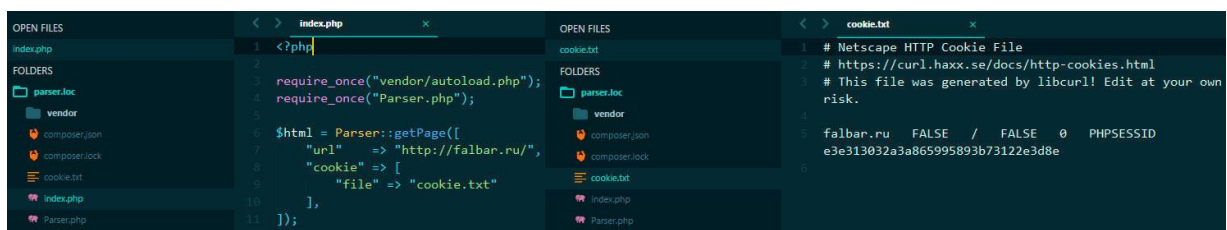


Рисунок 4.9 Сохранение cookies

Всё получилось, двигаемся дальше и осталось добавить в параметры сеанса: прокси, заголовки и возможность отправки запросов POST:

```
if($proxy_ip && $proxy_port && $proxy_type){

    curl_setopt($ch, CURLOPT_PROXY, $proxy_ip." ".$proxy_port);
    curl_setopt($ch, CURLOPT_PROXYTYPE, $proxy_type);

}

if($headers){

    curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);

}
```

```

if($post){

    curl_setopt($ch, CURLOPT_POSTFIELDS, $post);
}

```

Это часть параметров, с которыми можно работать, все остальные находятся в официальной документации PHP.

Далее требуется проверить работоспособность реализованного кода.

Теперь, при помощи класса Parser, можем сделать запрос и получить страницу с контентом.

Давайте и поступим:

```

$html = Parser::getPage([
    "url" => "https://www.svyaznoy.ru/catalog"
]);

```

Следующим шагом разбираем пришедший ответ и сохраняем название и ссылку категории в результирующий массив:

```

if(!empty($html["data"])){

    $content = $html["data"]["content"];

    $pq = new phpQuery($content);

    $categories = $pq->find(".b-category-menu__link");

    $tmp = [];

    foreach($categories as $key => $category){

        $category = $pq($category);

        $tmp[$key] = [
            "text" => trim($category->text()),
            "url" => trim($category->attr("href"))
        ];
    }
}

```

```

        $submenu = $category->next(".b-category-submenu")->find(".b-category-
submenu__link");

        foreach($submenu as $submen){

            $submen = pq($submen);

            $tmp[$key]["submenu"][] = [
                "text" => trim($submen->text()),
                "url" => trim($submen->attr("href"))
            ];
        }
    }

    phpQuery::unloadDocuments();
}

```

В запросе метода phpQuery::unloadDocuments(); пробегаемся по DOM дереву и вытягиваем нужные нам данные, при этом требуется убрать лишние пробелы. А теперь выведем категории фильмом на экран:

```

<ul>
    <?php foreach($tmp as $value): ?>
        <li>
            <a href="https://www.svyaznoy.ru<?php echo($value["url"]); ?>"
target="_blank">
                <?php echo($value["text"]); ?>
            </a>
            <ul>
                <? if(!empty($value["submenu"])): ?>
                <?php foreach($value["submenu"] as $val): ?>
                    <li>
                        <a href="https://www.svyaznoy.ru<?php
echo($val["url"]); ?>" target="_blank">
                            <?php echo($val["text"]); ?>
                        </a>
                    </li>
                </li>
            </ul>
        </li>
    </ul>

```

<?php endforeach; ?>

<? endif; ?>

<?php endforeach; ?>

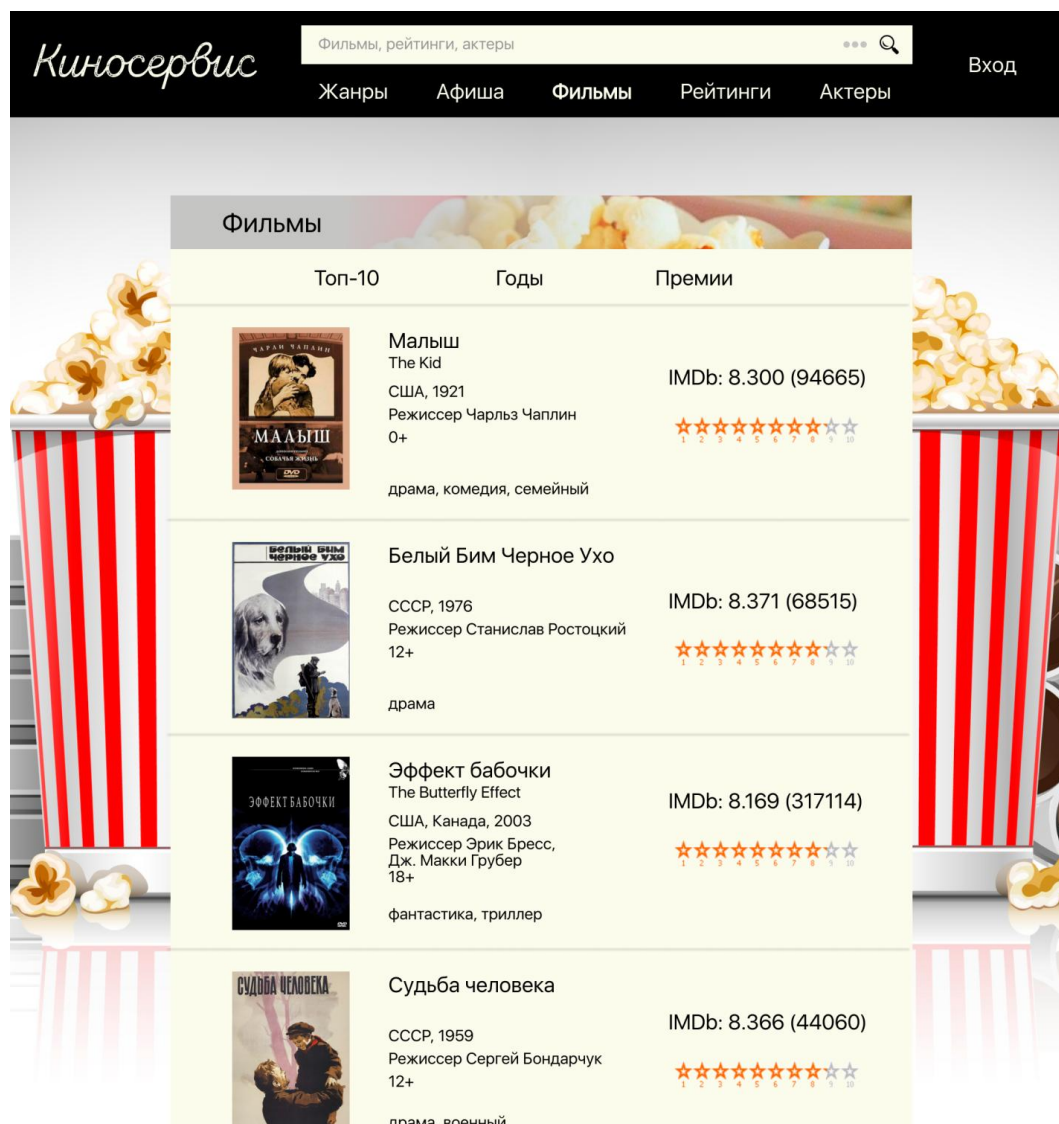


Рисунок 4.10 Вывод результата парсинга в браузер

В результате получили все ссылки на фильмы и категории им сопутствующие. Для получения результатов о конкретном фильме используется тот же принцип:

```
$html = Parser::getPage([
```

```
    "url"      => "https://www.kinopoisk.ru/catalog/film/224",
```

```
    "timeout" => 10
```

```
]);
```

Получаем страницу, тут я увеличил время соединения, так как 5 секунд не хватило, и разбираем её, парся необходимый контент:

```
if(!empty($html["data"])){
```

```
    $content = $html["data"]["content"];
```

```
    phpQuery::newDocument($content);
```

```
    $products = pq(".b-listing__generated-container")->find(".b-product-block .b-product-block__content");
```

```
    $tmp = [];
```

```
    foreach($products as $key => $product){
```

```
        $product = pq($product);
```

```
        $tmp[] = [
            "name"      =>  trim($product->find(".b-product-block__name")->text()),
            "image"     =>  trim($product->find(".b-product-block__image img")->attr("data-original")),
            "price"      =>  trim($product->find(".b-product-block__misc .b-product-block__visible-price")->text()),
            "url"        =>  trim($product->find(".b-product-block__info .b-product-block__main-link")->attr("href"))
        ];
```

```
        $chars = $product->find(".b-product-block__info .b-product-block__tech-
chars li");
```

```
        foreach($chars as $char){
```

```
            $tmp[$key]["chars"][] = pq($char)->text();
```

```

    }
}

```

```

    phpQuery::unloadDocuments();
}

```

Теперь проверим, что у получилось, и выведем на экран:

```

<div class="film">
    <?php foreach($tmp as $value): ?>
        <a href="https://www.kinopoisk.ru<?php echo($value["url"]); ?>" target="_blank"
class="tovar">
            " alt="<?php
echo($value["name"]); ?>" />
            <span class="name">
                <?php echo($value["name"]); ?>
            </span>
            <span class="describe">
                <?php echo($value["describe "]); ?>
            </span>
            <ul class="chars">
                <? if(!empty($value["chars"])): ?>
                <?php foreach($value["chars"] as $val): ?>
                <li>
                    <?php echo($val); ?>
                </li>
                <?php endforeach; ?>
                <? endif; ?>
            </ul>
        </a>
    <?php endforeach; ?>
</div>

```

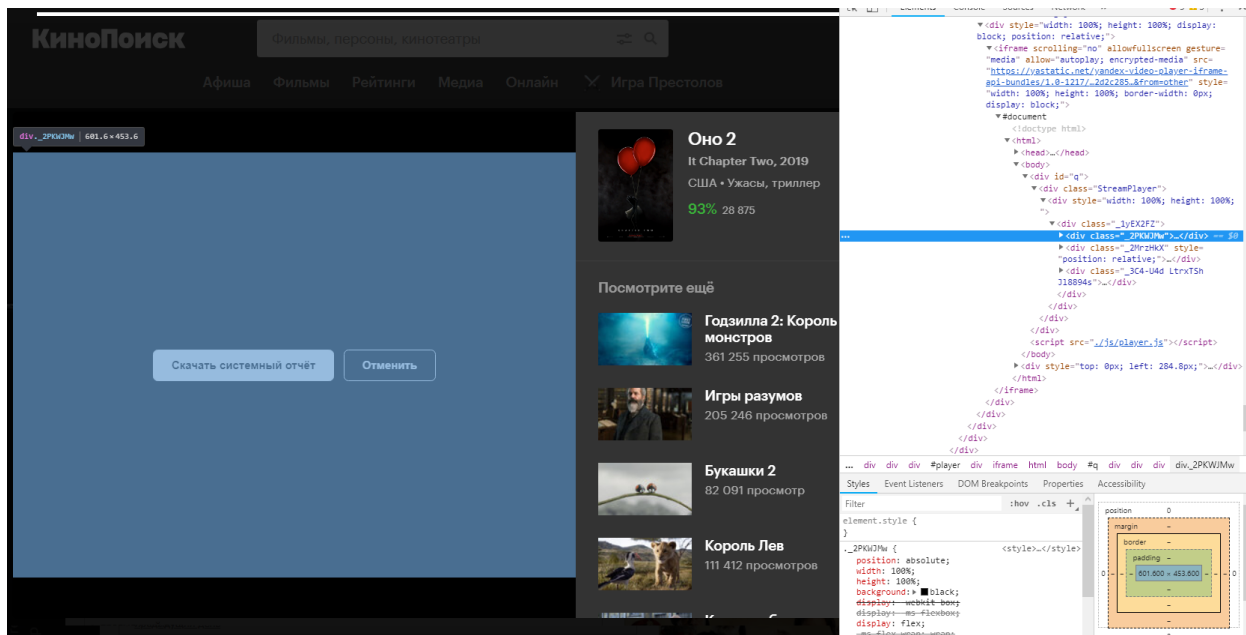


Рисунок 4.11 Вывод результата парсинга конкретного фильма в браузер

В результате в данной главе описан процесс реализации парсера контента РНР, при помощи скрипта можно легко парсить страницы сайта.

Выводы по технологии:

1. если хотите парсить более одной страницы, то не стоит забывать, что сам процесс парсинга ресурса затратная операция, поэтому в идеале лучше, чтобы скрипт был вынесен на отдельный сервер, где и будет запускаться по крону.
2. к каждому донору стоит подходить индивидуально, так как, во-первых: у них разный HTML код и он, с течением времени, может меняться, во-вторых: могут быть различные защиты от парсинга и проверки, поэтому для подбора необходимого набора заголовков и параметров может потребоваться отладочный прокси (я пользуюсь Fiddler).
3. используйте для парсинга прокси и чем больше, тем лучше, так как, когда на сервер донора полетят тысячи запросов, то неизбежно IP, с которого осуществляется обращение будет заблокирован, поэтому стоит прогонять свои запросы через прокси-сервера.

5. Технико-экономическое обоснование разработки программы

5.1 Введение

Целью технико-экономического обоснования разработки является количественное и качественное доказательство экономической целесообразности усовершенствования программы, а также определение организационно-экономических условий ее эффективного функционирования.

Эффективность программного модуля определяется его качеством и эффективностью процесса разработки и сопровождения. Качество программного изделия определяется тремя составляющими:

- с точки зрения специалиста-пользователя данного программного продукта;
- с позиции использования ресурсов и их оценки;
- по выполнению требований на программное изделие.

Программное изделие должно быть разработано так, чтобы оно выполняло свои функции без лишних затрат ресурсов (оперативной памяти ПК, машинного времени, пропускной способности каналов передачи данных и др. – на стадии функционирования; времени разработки и денежных ресурсов – на стадии использования программного изделия).

Содержание технико-экономического обоснования разработки программы заключается в следующем:

- расчет технико-экономических показателей и выбор базы сравнения;

- определение трудоемкости и стоимости программного обеспечения (ПО);
- расчет цены ПО;
- расчет капитальных и эксплуатационных затрат на разработку;
- определение показателей финансово-экономической эффективности.

Приведенные в данном разделе дипломной работы результирующие таблицы, перечни данных, позволяют сопоставить результаты разработки и затраты на нее, чтобы сделать вывод об эффективности проекта. Исходные данные для расчета экономических показателей приведены в таблице 5.1.

Таблица 5.1. Исходные данные для расчета экономических показателей

Обозначение	Наименование показателя	Единицы измерения	Значение показателя		
$C_{ЭВМ}$	Стоимость ЭВМ	тыс. руб.	20		
D_M	Среднее количество дней в месяце	дни	22		
ρ_n	Норматив рентабельности	—	0,24		
ω_d	Коэффициент, учитывающий дополнительную заработную плату разработчика программы	—	0,20		
ω_c	Коэффициент, учитывающий начисления органам социального страхования	—	0,268		
ω_n	Коэффициент, учитывающий накладные расходы организации	—	1,35		
q_I	Количество I-задач, решаемых потребителем	зад. год	60		
$t_{M.B.I}$	Время решения I-ой задачи разработанной программой	маш. час	1		
n_p	Количество организаций, которые приобретут данную программу	шт.	1г. 8	2г. 10	3г. 15
$Z_{ЭЛ}$	Тариф за 1 кВт/час	руб.	3,44		
T_C	Срок службы разработанной программы	год	3		
$НДС$	Налог на добавленную стоимость	%	20		
T_P	Количество рабочих дней в году	дн	264		
N_{CM}	Количество смен работы ЭВМ	—	1		
t_{CM}	Продолжительность смены	ч	8		
α	Простои ЭВМ	%	5		
P	Мощность, потребляемая ЭВМ	кВт	0,4		
N_{CP}	Среднее количество ремонтов в год	—	2		
S_d	Стоимость деталей, заменяемых при ремонте	руб.	1300		

5.2 Расчет затрат на разработку программного обеспечения

Суммарные затраты на разработку программы рассчитываются по следующей формуле:

$$S_{\text{РП}} = S_{\text{ЗП}} + S_{\text{НАК}}, \quad (5.1)$$

где:

$S_{\text{ЗП}}$ – затраты по заработной плате инженера-программиста;

$S_{\text{НАК}}$ – накладные расходы.

Затраты по заработной плате инженера-программиста рассчитываются по формуле:

$$S_{\text{ЗП}} = \text{ОЗП} \cdot (1 + \omega_c) \cdot (1 + \omega_d) \cdot t_{\text{pi}}, \quad (5.2)$$

где:

ОЗП – основная заработная плата инженера-программиста за месяц (10000 руб.);

t_{pi} – время, необходимое для разработки программы программистом i -го разряда (чел.-мес.);

ω_d – коэффициент, учитывающий дополнительную заработную плату разработчика программы, в долях к сумме основной заработной платы;

ω_c – коэффициент, учитывающий начисления органам социального страхования на заработную плату разработчика программы, в долях к сумме основной заработной плате разработчика.

Программа разрабатывалась 30 дней, если учесть, что в одном месяце 22 рабочих дня, то:

$$t_{\text{p12}} = \frac{30}{22} = 1,36 \text{ (чел. –мес.)} \quad (5.3)$$

Таким образом, затраты по заработной плате инженера-программиста составят:

$$S_{\text{ЗП}} = 10000 \cdot (1 + 0,268) \cdot (1 + 0,20) \cdot 1,36 = 20693,76 \text{ (руб.)}$$

Накладные затраты рассчитываются с учетом ω_n – коэффициента, определяющего уровень накладных расходов организации по формуле:

$$S_{\text{НАК}} = \text{ОЗП} \cdot \omega_n \cdot t_{\text{p12}}, \quad (5.4)$$

$$S_{\text{НАК}} = 10000 \cdot 1,35 \cdot 1,36 = 18360 \text{ (руб.)}$$

Таким образом, суммарные затраты на разработку программного обеспечения составляют:

$$S_{\text{РП}} = 20693,76 + 18360 = 39053,76 \text{ (руб.)}$$

5.3 Расчет цены разработанной программы

Оптовая цена разработанной программы определяется по следующей формуле:

$$Z_{\text{П}} = S_{\text{РП}} + \Pi, \quad (5.5)$$

где:

$Z_{\text{П}}$ – оптовая цена (цена разработчика) (руб.);

$S_{\text{РП}}$ – суммарные затраты на разработку программы (руб.);

Π – прибыль, рассчитанная по формуле:

$$\Pi = \rho_n \cdot S_{\text{РП}}, \quad (5.6)$$

где:

ρ_n – норматив рентабельности, учитывающий прибыль организации, разрабатывающей данную программу в долях ко всем затратам данной организации на разработку программы.

Итак,

$$Z_{\text{П}} = S_{\text{РП}} \cdot (1 + \rho_n), \quad (5.7)$$

$$Z_{\text{П}} = 39053,76 \cdot (1 + 0,24) = 48426,7 \text{ (руб.)}$$

Розничная цена программы рассчитывается с учетом налога на добавленную стоимость ($\text{НДС} = 20\%$) по формуле:

$$Z_{\text{Пр}} = Z_{\text{П}} + \text{НДС} = Z_{\text{П}} + \frac{Z_{\text{П}} \cdot 20}{100} = Z_{\text{П}} \cdot (1 + 0,20), \quad (5.8)$$

$$Z_{\text{Пр}} = 48426,7 \cdot (1 + 0,20) = 58112,04 \text{ (руб.)}$$

Выручка от продаж при условии $n_{\text{п}}$ – количество организаций, желающих приобрести программу, составит:

$$B = Z_{\text{Пр}} \cdot n_{\text{п}}, \quad (5.9)$$

$$B_1 = 58112,04 \cdot 8 = 464896,32 \text{ (руб.)}$$

$$B_2 = 58112,04 \cdot 10 = 581120,40 \text{ (руб.)}$$

$$B_3 = 58112,04 \cdot 15 = 871680,60 \text{ (руб.)}$$

5.4 Расчет капитальных вложений

Капиталовложения, связанные с работой ЭВМ рассчитываются по формуле:

$$K_{\text{ЭВМ}} = C_{\text{ЭВМ}} + S_3, \quad (5.10)$$

где:

$C_{\text{ЭВМ}}$ – стоимость ЭВМ (руб.);

S_3 – стоимость запасных частей (руб.);

Итак, произведем расчет коэффициентов входящих в формулу расчета величины капиталовложений:

$$S_3 = 0,15 \cdot C_{\text{ЭВМ}}; \quad (5.11)$$

$$S_3 = 0,15 \cdot 20000 = 3000 \text{ (руб.)};$$

Капиталовложения в ЭВМ составляют:

$$K_{\text{ЭВМ}} = 20000 + 3000 = 23000 \text{ (руб.)}$$

5.5 Расчет эксплуатационных расходов

Эксплуатационные расходы на ЭВМ рассчитываются по формуле:

$$E = (T_{\text{М.В.}} \cdot e_{\text{ч}}) + \frac{Z_{\text{п}}}{T_{\text{с}}}, \quad (5.12)$$

где:

$T_{\text{М.В.}}$ – машинное время для решения задач с помощью разработанной программы, (маш.час/год);

$e_{\text{ч}}$ – эксплуатационные расходы, приходящиеся на 1 час работы ЭВМ;

$Z_{\text{П}}$ – цена, по которой продается программа (руб.);

$T_{\text{С}}$ – срок службы программы (г).

Полезный фонд времени работы ЭВМ рассчитывается по формуле:

$$T_{\text{ПОЛ}} = T_{\text{ОБЩ}} \cdot t_{\text{СМ}} \cdot N_{\text{СМ}} \cdot \left(1 - \frac{\alpha}{100}\right), \quad (5.13)$$

где:

$T_{\text{ОБЩ}}$ – общий фонд времени работы ЭВМ (дни); $T_{\text{ОБЩ}} = T_{\text{Р}}$;

$N_{\text{СМ}}$ – количество смен работы ЭВМ;

$t_{\text{СМ}}$ – время одного рабочего дня (час);

α – простои ЭВМ (в % от общего фонда времени работы ЭВМ).

Полезный фонд времени работы ЭВМ получим:

$$T_{\text{ПОЛ}} = 264 \cdot 8 \cdot 1 \cdot \left(1 - \frac{5}{100}\right) = 2006,4 \text{ (Маш. час /год)}.$$

Машинное время для решения задач с помощью данной программы рассчитывается по формуле:

$$T_{\text{М.В.}} = q_{\text{I}} \cdot t_{\text{М.В. I}}, \quad (5.14)$$

где:

q_{I} – количество I-задач, решаемых потребителем в год(шт.);

$t_{\text{М.В. I}}$ – время решения I-ой задачи, разработанной программой (Маш. час).

$$T_{\text{М.В.}} = 60 \cdot 1 = 60 \text{ (Маш. час / год)}.$$

Эксплуатационные расходы, приходящиеся на 1 час работы ЭВМ, оцениваются по формуле:

$$e_{\text{ч}} = \frac{(A_{\text{О}} + S_{\text{ЗП}} + S_{\text{ЭЛ}} + R_{\text{РМ}})}{T_{\text{ПОЛ}}}, \quad (5.15)$$

где:

$A_{\text{О}}$ – амортизационные отчисления (руб.);

$S_{\text{ЗП}}$ – затраты по заработной плате инженера в год (руб./год);

$S_{\text{ЭЛ}}$ – стоимость потребляемой энергии (руб.);

$R_{\text{РМ}}$ – затраты на ремонт ЭВМ (руб.);

$T_{\text{ПОЛ}}$ – полезный годовое фонд работы ЭВМ, (маш.час/год).

Амортизационные отчисления рассчитываются с учетом нормы амортизации ($a_{\text{н}} = 12,5 \%$);

$$A_{\text{О}} = a_{\text{н}} \cdot K_{\text{ЭВМ}} = 0,125 \cdot 23000 = 2875,0 \text{ (руб.)}. \quad (5.16)$$

Затраты по заработной плате инженера за год рассчитывается по формуле:

$$S_{\text{ЗП}} = (1 + \omega_{\text{с}}) \cdot (1 + \omega_{\text{д}}) \cdot \text{ОЗП} \cdot 12, \quad (5.17)$$

где:

$\omega_{\text{с}}$ – коэффициент, учитывающий начисления органам социального страхования на заработную плату разработчика программы, в долях к сумме основной заработной плате разработчика.

$\omega_{\text{д}}$ – коэффициент, учитывающий дополнительную заработную плату разработчика программы, в долях к сумме основной заработной платы;

ОЗП – основная заработная плата инженера за месяц 8-го разряда (10000 руб.);

Рассчитываем годовые затраты по заработной плате и социальным отчислениям для инженера:

$$S_{\text{ЗП}} = (1 + 0,268) \cdot (1 + 0,20) \cdot 10000 \cdot 12 = 182592 \text{ (руб./год)}$$

Стоимость потребляемой энергии оценивается по формуле:

$$S_{\text{ЭЛ}} = P \cdot T_{\text{ПОЛ}} \cdot Z_{\text{ЭЛ}}, \quad (5.18)$$

где:

P – мощность, потребляемая ЭВМ (кВт);

$T_{\text{ПОЛ}}$ – полезный годовой фонд работы ЭВМ (маш.час/год);

$Z_{\text{ЭЛ}}$ – тариф за 1 кВт/час (руб. /кВт).

Итак, произведем расчет стоимости потребляемой энергии:

$$S_{\text{ЭЛ}} = 0,4 \cdot 2006,4 \cdot 3,44 = 2760,81 \text{ (руб.)}.$$

Затраты на ремонт ЭВМ вычисляются по формуле:

$$R_{\text{РМ}} = N_{\text{СР}} \cdot S_{\text{Д}}, \quad (5.19)$$

где:

$N_{\text{СР}}$ – среднее количество ремонтов в год;

$S_{\text{Д}}$ – стоимость деталей заменяемых при одном ремонте, в среднем.

$$R_{\text{РМ}} = 2 \cdot 1300 = 2600 \text{ (руб.)}.$$

Произведем вычисление эксплуатационных расходов, приходящихся на 1 час работы ЭВМ:

$$e_{\text{ч}} = \frac{(2875 + 182592 + 2760,81 + 2600)}{2006,4} = 95,12 \text{ (руб./час)}.$$

Далее вычислим эксплуатационные расходы на ЭВМ:

$$E = (60 \cdot 95,12) + \frac{48426,7}{3} = 21849,43 \text{ (руб.)}.$$

5.6 Определение расходов на интернет трафик

Для облегчения работы программиста и повышения его производительности в проекте предусмотрено использование глобальной сети интернет.

В настоящее время на пензенском рынке интернет очень много провайдеров предлагающие свои услуги. Цены предлагаемые основными провайдерами сведем в таблицу 4.6.1, для анализа и экономически целесообразного выбора поставщика услуги по интернет трафику.

					ПГУ 1.090501.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		87

Таблица 5.2 Таблица интернет трафиков и провайдеров

Название провайдера	Предлагаемый тариф	Стоимость тарифа (руб.)	Скорость доступа (Мбит/с)
Ростелеком	Пакет-Для впечатлений	699	100
ПАО «МТС»	ФИТ Интернет	450	100
Дом.РУ	Скорость света	550	100

Т.к. выше перечисленные провайдеры предлагают без лимитные интернет трафики с одинаковой скоростью доступа в сеть интернет, то экономически целесообразно выбрать трафик с самой низкой стоимостью. Выбираем ПАО "МТС", тариф ФИТ Интернет, с абонентской платой в месяц 450 рублей.

5.7 Заключение

Обобщенные технико-экономические показатели разработки программы сведены в таблицу 5.7.1.

Таблица 5.3 Итоговая таблица

Показатель	Значение
Капитальные вложения (руб.)	23000
Эксплуатационные расходы (руб.)	21849,43
Затраты на разработку ПО (руб.)	39053,76
Затраты на интернет трафик (руб.)	450
Затраты на заработную плату программисту (руб.)	20693,76

6. Безопасность жизнедеятельности. Эргономика рабочего места программиста.

6.1 Характеристика условий труда программиста

Технический прогресс глобально поменял условия производственной деятельности сотрудников умственного труда. Их труд стал более интенсивнее, требующим больших затрат умственной, эмоциональной и физической энергии. Это потребовало системного решения проблем эргономики, гигиены и организации труда, регламентации режимов труда и отдыха.

В настоящее время компьютеры широко применяется во всех областях деятельности человека. При работе с компьютером, человек подвергается воздействию ряда вредных электромагнитных полей (ВЧ, УВЧ и СВЧ), шума и вибрации, статического электричества и др.

Работа с компьютером характеризуется значительным умственным напряжением и нервно-эмоциональной нагрузкой операторов, высокой напряженностью зрительной работы и большой нагрузкой на мышцы рук при работе с клавиатурой. Большое значение имеет рациональная конструкция и расположение элементов рабочего места, что важно для поддержания оптимальной рабочей позы программиста.

В процессе работы с компьютером необходимо соблюдать правильный режим труда и отдыха. В противном случае у персонала отмечаются болезненное напряжение зрительного аппарата с появлением жалоб на неудовлетворенность работой, головные боли, раздражительность, нарушение сна, усталость и болезненные ощущения в глазах, в пояснице, в области шеи и руках.

					ПГУ 1.090501.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		89

Одним из критериев, по которому определяется «пригодность» рабочего места для постоянного пребывания сотрудников, является эргономические требования к рабочему месту и освещенность рабочего места программиста.

6.2 Эргономические требования к рабочему месту программиста

Проектирование рабочих мест, снабженных мониторами, относится к числу важных проблем эргономического проектирования в области вычислительной техники.

Рабочее место и взаимное расположение всех его элементов должно соответствовать антропометрическим, физическим и психологическим требованиям. Большое значение имеет также характер работы. В частности, при организации рабочего места программиста должны быть соблюдены следующие основные условия: оптимальное размещение оборудования, входящего в состав рабочего места и достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения.

Эргономическими аспектами проектирования рабочих мест программиста, являются:

1. высота рабочей поверхности,
2. размеры пространства для ног,
3. требования к расположению документов на рабочем месте
(наличие и размеры подставки для документов, возможность различного размещения документов, расстояние от глаз пользователя до экрана, документа, клавиатуры и т.д.),
4. характеристики рабочего кресла,
5. требования к поверхности рабочего стола,
6. регулируемость элементов рабочего места.

Главными элементами рабочего места программиста являются стол и кресло.

Основным рабочим положением является положение сидя.

Рабочая поза сидя вызывает минимальное утомление программиста.

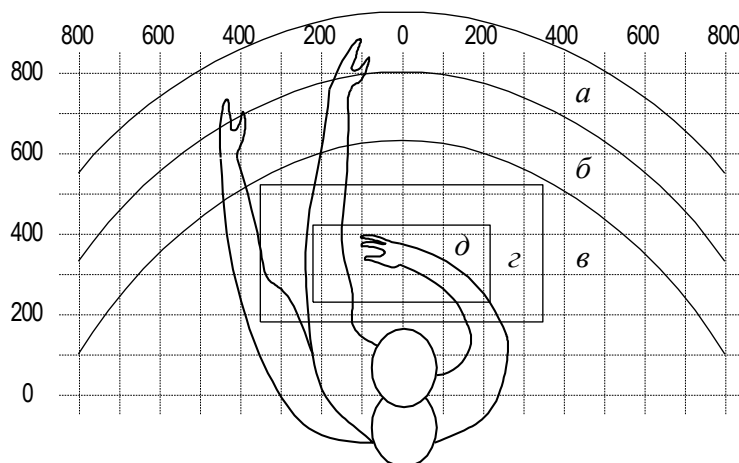
Рациональная планировка рабочего места предусматривает четкий порядок и постоянство размещения предметов, средств труда и документации.

То, что требуется для выполнения работ чаще, расположено в зоне легкой досягаемости рабочего пространства.

Моторное поле - пространство рабочего места, в котором могут осуществляться двигательные действия человека.

Максимальная зона досягаемости рук - это часть моторного поля рабочего места, ограниченного дугами, описываемыми максимально вытянутыми руками при движении их в плечевом суставе.

Оптимальная зона - часть моторного поля рабочего места, ограниченного дугами, описываемыми предплечьями при движении в локтевых суставах с опорой в точке локтя и с относительно неподвижным плечом.



- а - зона максимальной досягаемости;
- б - зона досягаемости пальцев при вытянутой руке;
- в - зона легкой досягаемости ладони;
- г - оптимальное пространство для грубой ручной работы;
- д - оптимальное пространство для тонкой ручной работы

Рисунок 6.1 Зоны досягаемости рук в горизонтальной

Оптимальное размещение предметов труда и документации в зонах досягаемости:

Монитор размещается в зоне **а** (в центре);

Системный блок размещается в предусмотренной нише стола;

Клавиатура - в зоне г/д;

«**Мышь**» - в зоне **в** справа;

Сканер в зоне, **а/б** (слева);

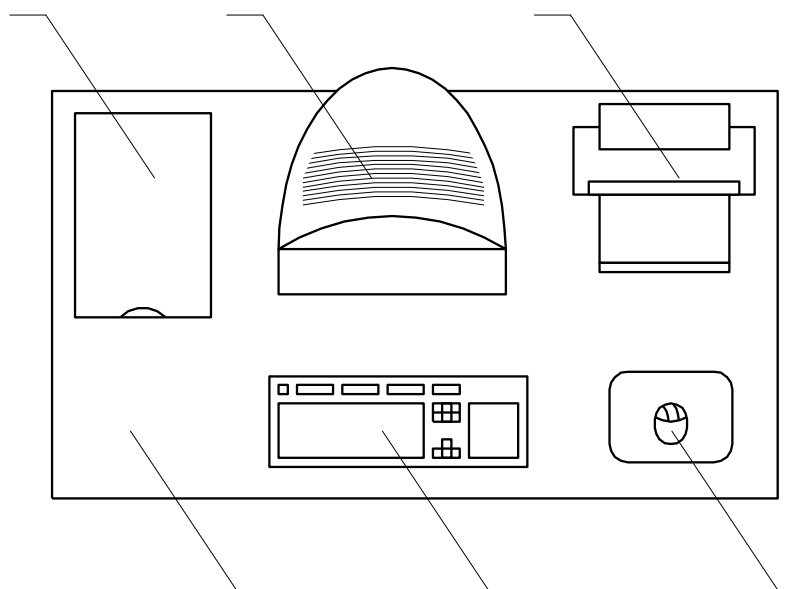
Принтер находится в зоне **а** (справа);

Документация необходимая при работе - в зоне легкой досягаемости ладони;

В выдвижных ящиках стола - литература, неиспользуемая постоянно.

На рис. 6.2 показан пример размещения основных и периферийных составляющих ПК на рабочем столе программиста.

1 – сканер, 2 – монитор, 3 – принтер, 4 – поверхность рабочего стола,



5 – клавиатура, 6 – «мышь».

Рисунок 6.2 Размещение основных и периферийных составляющих ПК.

Для комфортной работы стол должен удовлетворять следующим условиям:

1. высота стола должна быть выбрана с учетом возможности сидеть свободно, в удобной позе, при необходимости опираясь на подлокотники;
2. нижняя часть стола должна быть сконструирована так, чтобы программист мог удобно сидеть, не был вынужден поджимать ноги;
3. поверхность стола должна обладать свойствами, исключающими появление бликов в поле зрения программиста;
4. конструкция стола должна предусматривать наличие выдвижных ящиков (не менее 3 для хранения документации, канцелярских принадлежностей).
5. высота рабочей поверхности рекомендуется в пределах 680-760 мм. Высота поверхности, на которую устанавливается клавиатура, должна быть около 650 мм.

Большое значение придается характеристикам рабочего кресла. Так, рекомендуемая высота сиденья над уровнем пола находится в пределах 420-550 мм. Поверхность сиденья мягкая, передний край закругленный, а угол наклона спинки - регулируемый.

Необходимо предусматривать при проектировании возможность различного размещения документов: сбоку от монитора, между монитором и клавиатурой и т.п. Кроме того, в случаях, когда монитор имеет низкое качество изображения, например заметны мелькания, расстояние от глаз до экрана делают больше (около 700 мм), чем расстояние от глаза до документа (300-450 мм). Вообще при высоком качестве изображения на видеотерминале расстояние от глаз пользователя до экрана, документа и клавиатуры может быть равным.

Положение экрана определяется:

1. расстоянием считывания (0,6...0,7 м);

2. углом считывания, направлением взгляда на 20° ниже горизонтали к центру экрана, причем экран перпендикулярен этому направлению. Должна также предусматриваться возможность регулирования экрана:
3. по высоте $+3$ см;
4. по наклону от -10° до $+20^\circ$ относительно вертикали;
5. в левом и правом направлениях.

Большое значение также придается правильной рабочей позе пользователя. При неудобной рабочей позе могут появиться боли в мышцах, суставах и сухожилиях. Требования к рабочей позе пользователя монитора следующие:

1. голова не должна быть наклонена более чем на 20° ,
2. плечи должны быть расслаблены,
3. локти - под углом $80^\circ \dots 100^\circ$,
4. предплечья и кисти рук - в горизонтальном положении.

Причина неправильной позы пользователей обусловлена следующими факторами: нет хорошей подставки для документов, клавиатура находится слишком высоко, а документы - низко, некуда положить руки и кисти, недостаточно пространство для ног.

В целях преодоления указанных недостатков даются общие рекомендации: лучше передвижная клавиатура; должны быть предусмотрены специальные приспособления для регулирования высоты стола, клавиатуры и монитора, а также подставка для рук.

Существенное значение для производительной и качественной работы на компьютере имеют размеры знаков, плотность их размещения, контраст и соотношение яркостей символов и фона экрана. Если расстояние от глаз оператора до экрана дисплея составляет $60 \dots 80$ см, то высота знака должна быть не менее 3 мм, оптимальное соотношение ширины и высоты знака составляет 3:4, а расстояние между знаками – $15 \dots 20\%$ их высоты.

Соотношение яркости фона экрана и символов - от 1:2 до 1:15.

Во время пользования компьютером медики советуют устанавливать монитор на расстоянии 50-60 см от глаз. Специалисты также считают, что верхняя часть видеодисплея должна быть на уровне глаз или чуть ниже. Когда человек смотрит прямо перед собой, его глаза открываются шире, чем когда он смотрит вниз. За счет этого площадь обзора значительно увеличивается, вызывая обезвоживание глаз. К тому же если экран установлен высоко, а глаза широко открыты, нарушается функция моргания. Это значит, что глаза не закрываются полностью, не омываются слезной жидкостью, не получают достаточного увлажнения, что приводит к их быстрой утомляемости.

Создание благоприятных условий труда и правильное эстетическое оформление рабочих мест на производстве имеет большое значение, как для облегчения труда, так и для повышения его привлекательности, положительно влияющей на производительность труда.

6.3 Освещенность рабочего места программиста

Правильно спроектированное и выполненное производственное освещение улучшает условия зрительной работы, снижает утомляемость, способствует повышению производительности труда, благотворно влияет на производственную среду, оказывая положительное психологическое воздействие на работающего, повышает безопасность труда и снижает травматизм.

Недостаточность освещения приводит к напряжению зрения, ослабляет внимание, приводит к наступлению преждевременной утомленности. Чрезмерно яркое освещение вызывает ослепление, раздражение в глазах. Неправильное направление света на рабочем месте может создавать резкие тени, блики, дезориентировать работающего. Все эти причины могут привести к несчастному случаю или профзаболеваниям, поэтому столь важен правильный расчет освещенности.

Существует три вида освещения - естественное, искусственное и совмещенное (естественное и искусственное вместе).

Естественное освещение - освещение помещений дневным светом, проникающим через световые проемы в наружных ограждающих конструкциях помещений. Естественное освещение характеризуется тем, что меняется в широких пределах в зависимости от времени дня, времени года, характера области и ряда других факторов.

Искусственное освещение применяется при работе в темное время суток и днем, когда не удастся обеспечить нормированные значения коэффициента естественного освещения (пасмурная погода, короткий световой день). Освещение, при котором недостаточное по нормам естественное освещение дополняется искусственным, называется совмещенным освещением.

Искусственное освещение подразделяется на рабочее, аварийное, эвакуационное, охранное. Рабочее освещение, в свою очередь, может быть общим или комбинированным. Общее - освещение, при котором светильники размещаются в верхней зоне помещения равномерно или применительно к расположению оборудования. Комбинированное - освещение, при котором к общему добавляется местное освещение.

Согласно СНиП 23-05-2010 в помещениях вычислительных центров необходимо применить систему комбинированного освещения.

При выполнении работ требующих высокой зрительной точности (наименьший размер объекта различения 0,3...0,5 мм), величина коэффициента естественного освещения (КЕО) должна быть не ниже 1,5%, а при зрительной работе средней точности (наименьший размер объекта различения 0,5...1,0 мм) КЕО должен быть не ниже 1,0%.

В качестве источников искусственного освещения обычно используются люминесцентные лампы типа ЛБ или ДРЛ, которые попарно объединяются в светильники, которые должны располагаться над рабочими поверхностями равномерно.

					ПГУ 1.090501.05.001 ПЗ	Лист
						96
Изм.	Лист	№ докум.	Подпись	Дата		

Требования к освещенности в помещениях, где установлены компьютеры, следующие: при выполнении зрительных работ высокой точности общая освещенность должна составлять 300 лк, а комбинированная - 750 лк; аналогичные требования при выполнении работ средней точности - 200 и 300 лк соответственно.

Кроме того, все поле зрения должно быть освещено достаточно равномерно – это основное гигиеническое требование. Иными словами, степень освещения помещения и яркость экрана компьютера должны быть примерно одинаковыми, т.к. яркий свет в районе периферийного зрения значительно увеличивает напряженность глаз и, как следствие, приводит к их быстрой утомляемости.

6.4 Расчет освещения рабочего места программиста

Расчет освещенности рабочего места сводится к выбору системы освещения, определению необходимого числа светильников, их типа и размещения. Исходя из этого, рассчитаем параметры искусственного освещения.

Обычно искусственное освещение выполняется посредством электрических источников света двух видов: ламп накаливания и люминесцентных ламп. Будем использовать люминесцентные лампы, которые по сравнению с лампами накаливания имеют ряд существенных преимуществ:

1. по спектральному составу света они близки к дневному, естественному свету;
2. обладают более высоким КПД (в 1,5-2 раза выше, чем КПД ламп накаливания);
3. обладают повышенной светоотдачей (в 3-4 раза выше, чем у ламп накаливания);
4. более длительный срок службы.

Расчет освещения производится для комнаты площадью 15 м^2 , ширина которой 5 м, высота - 3 м. Воспользуемся методом светового потока.

Для определения количества светильников определим световой поток, падающий на поверхность по формуле:

$$F = \frac{E \cdot K \cdot S \cdot Z}{n}, \text{ где}$$

F - рассчитываемый световой поток, Лм;

E - нормированная минимальная освещенность, Лк (определяется по таблице). Работу программиста, в соответствии с этой таблицей, можно отнести к разряду точных работ, следовательно, минимальная освещенность будет $E = 300 \text{ Лк}$;

S - площадь освещаемого помещения (в нашем случае $S = 15 \text{ м}^2$);

Z - отношение средней освещенности к минимальной (обычно принимается равным 1,1...1,2, пусть $Z = 1,1$);

K - коэффициент запаса, учитывающий уменьшение светового потока лампы в результате загрязнения светильников в процессе эксплуатации (его значение зависит от типа помещения и характера проводимых в нем работ и в нашем случае $K = 1,5$);

n - коэффициент использования, (выражается отношением светового потока, падающего на расчетную поверхность, к суммарному потоку всех ламп и исчисляется в долях единицы; зависит от характеристик светильника, размеров помещения, окраски стен и потолка, характеризуемых коэффициентами отражения от стен (P_C) и потолка (P_{Π}), значение коэффициентов P_C и P_{Π} были указаны выше: $P_C = 40\%$, $P_{\Pi} = 60\%$. Значение n определим по таблице коэффициентов использования различных светильников. Для этого вычислим индекс помещения по формуле:

$$I = \frac{S}{h \cdot (A + B)}, \text{ где}$$

S - площадь помещения, **S** = 15 м²;

h - расчетная высота подвеса, **h** = 2.92 м;

A - ширина помещения, **A** = 3 м;

B - длина помещения, **B** = 5 м.

Подставив значения получаем:

$$I = \frac{15}{2,92 \cdot (3+5)} = 0,64$$

Зная индекс помещения **I**, по таблице находим **n** = 0,22

Подставим все значения в формулу для определения светового потока

F:

$$F = \frac{300 \cdot 1,5 \cdot 15 \cdot 1,1}{0,22} = 33750 \text{ Лм}$$

Для освещения выбираем люминесцентные лампы типа ЛБ40-1, световой поток которых **F** = 4320 Лк.

Рассчитаем необходимое количество ламп по формуле:

$$N = \frac{F}{F_{\text{л}}}$$

N - определяемое число ламп;

F - световой поток, **F** = 33750 Лм;

F_л - световой поток лампы, **F_л** = 4320 Лм.

$$N = \frac{33750}{4320} = 8 \text{ шт.}$$

При выборе осветительных приборов используем светильники типа ОД. Каждый светильник комплектуется двумя лампами.

Заключение

В процессе выполнения данной выпускной квалификационной работы проведены анализ требований, проектирование и разработка и тестирование программного обеспечения для тематического портала знаний о кинофильмах.

В процессе анализа требований, были выделены основные действующие лица. Проведен анализ приоритетов вариантов использования. Разработаны модель предметной области, проектная модель системы, представленные диаграммами классов. Были выбраны отдельные классы и алгоритмы, и описано их поведение диаграммами состояний и деятельности. В результате выполненной выпускной квалификационной работы реализован сайт, являющийся порталом знаний о кинофильмах.

Проект реализован на языках C#, JavaScript в среде разработки Visual Studio 2016. На основе анализа, полученных результатов, можно сделать вывод, что все функции, разработанной программы, работают корректно, на всех наборах тестовых данных. Следовательно, цель выпускной квалификационной работы достигнута.

Сайт может быть запущен в любой операционной системе с наличием современного браузера и доступом в интернет.

Список использованных источников

- 1 Интеллектуальные модели анализа информации: курс лекций 2017 BaseGroup Labs.
- 2 Елманова Н. Введение в OLAP-технологии Microsoft. / Н. Елманова, Федоров А. – М.: Диалог-МИФИ, 2017.
- 3 П. Блаттнер - Special Edition Using MVC. – М.: Вильямс, 2016.
- 4 Арлоу Д. Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование / Д. Арлоу, И. Нейштадт – СПб: Символ-Плюс, 2017.
- 5 Мацаяшек А. Анализ требований и проектирование системы. Разработка информационных систем с использованием UML / А. Мацаяшек – М.: Издательский дом «Вильямс», 2015.
- 6 Фаулер М. UML. Основы / М. Фаулер, К. Скотт – СПб: Символ-Плюс, 2014.
- 7 Дейтел Х. – С# / Х. Дейтел, Дейтел П., Листфид Дж., Нието Т., Йегер Ш., Златкина М. - СПб: БХВ-Петербург, 2016.
- 8 Фримен А. – LINQ: язык интегрированных запросов в С# 2016 для профессионалов / А. Фримен, Раттц-мл Д. – М.: Вильямс, 2018.
- 9 Фримен А. – ASP.NET MVC3 Framework с примерами на С# для профессионалов / Сандерсон С. - М.: Вильямс, 2017.
- 10 Флэнаган Д. – JavaScript Подробное руководство – СПб: Символ-Плюс, 2018.
- 11 Мандел Т. Разработка пользовательского интерфейса / Т. Мандел – М.: ДМК Пресс, 2018.
- 12 Гома Х. UML. Проектирование систем реального времени параллельных и распределенных приложений / Х. Гома – М.: ДМК Пресс, 2015.

14 Котляров В. П. Основы тестирования программного обеспечения: Учебное пособие / В. П. Котляров, Т. В. Коликова – М.: Бином. Лаборатория знаний, 2016.

15 Макгрегор Дж. Тестирование объектно-ориентированного программного обеспечения / Дж. Макгрегор, Д. Сайкс. – К.: ООО «ТИД «ДС», 2015.

					ПГУ 1.090501.05.001 ПЗ	Лист
						102
Изм.	Лист	№ докум.	Подпись	Дата		

ЛИСТИНГ ПРИЛОЖЕНИЯ

Приложение А (обязательное)

IRepository.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using AllFilms.Domain.Entities;

namespace AllFilms.Domain.Abstract
{
    public interface IRepository
    {
        IQueryable<Person> People { get; }
        IQueryable<Genre> Genres { get; }
        IQueryable<Film> Films { get; }
        IQueryable<Country> Countries { get; }
        IQueryable<FilmGenre> FilmGenres { get; }
        IQueryable<FilmPersonAssociation> FilmPersonAssociations { get; }
        IQueryable<FilmCountry> FilmCountries { get; }
        void SavePerson(Person person);
        void SaveFilm(Film film, Int32[] AssociatedCountriesID, Int32[]
AssociatedGenresID, Int32[] AssociatedStageDirectorsID, Int32[] AssociatedActorsID);
        void DeleteFilm(Film film);
        void DeletePerson(Person person);
    }
}
```

EFDDbContext.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data.Entity;
using AllFilms.Domain.Entities;

namespace AllFilms.Domain.Concrete
{
    public class EFDDbContext : DbContext
    {
        public DbSet<Person> People { get; set; }
        public DbSet<Genre> Genres { get; set; }
        public DbSet<Country> Countries { get; set; }
        public DbSet<Film> Films { get; set; }
        public DbSet<FilmGenre> FilmGenres { get; set; }
        public DbSet<FilmPersonAssociation> FilmPersonAssociations { get; set; }
        public DbSet<FilmCountry> FilmCountries { get; set; }
    }
}
```

EFRepository.cs

```
using System;
```

					ПГУ 1.090501.05.001 ПЗ	Лист
						103
Изм.	Лист	№ докум.	Подпись	Дата		

```

using System.Data.Entity;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using AllFilms.Domain.Abstract;
using AllFilms.Domain.Entities;
using System.Data.Entity.Migrations;

namespace AllFilms.Domain.Concrete
{
    public class EFRepository : IRepository
    {
        private EFDbContext context = new EFDbContext();
        public IQueryable<Person> People
        {
            get
            {
                return context.People;
            }
        }
        public IQueryable<Genre> Genres
        {
            get
            {
                return context.Genres;
            }
        }
        public IQueryable<Film> Films
        {
            get
            {
                return context.Films;
            }
        }
        public IQueryable<Country> Countries
        {
            get
            {
                return context.Countries;
            }
        }
        public IQueryable<FilmCountry> FilmCountries
        {
            get
            {
                return context.FilmCountries;
            }
        }
        public IQueryable<FilmGenre> FilmGenres
        {
            get
            {
                return context.FilmGenres;
            }
        }
        public IQueryable<FilmPersonAssociation> FilmPersonAssociations
        {
            get
            {
                return context.FilmPersonAssociations;
            }
        }
        public void SavePerson(Person person)
        {

```

```

        context.People.AddOrUpdate(person);
        context.SaveChanges();
    }
    public void DeletePerson(Person person)
    {
        DeletePersonAssociations(person);
        context.SaveChanges();
        context.People.Remove(person);
        context.SaveChanges();
    }
    public void SaveFilm(Film film, Int32[] AssociatedCountriesID, Int32[]
AssociatedGenresID, Int32[] AssociatedStageDirectorsID, Int32[] AssociatedActorsID)
    {
        if (film.FilmID != 0)
        {
            DeleteFilmAssociations(film);
        }
        context.Films.AddOrUpdate(film);
        context.SaveChanges();
        if (AssociatedCountriesID != null)
        {
            foreach (Int32 CountryID in AssociatedCountriesID)
            {
                context.FilmCountries.Add(new FilmCountry { FilmID = film.FilmID,
CountryID = CountryID });
            }
        }
        if (AssociatedGenresID != null)
        {
            foreach (Int32 GenreID in AssociatedGenresID)
            {
                context.FilmGenres.Add(new FilmGenre { FilmID = film.FilmID, GenreID
= GenreID });
            }
        }
        if (AssociatedStageDirectorsID != null)
        {
            foreach (Int32 PersonID in AssociatedStageDirectorsID)
            {
                context.FilmPersonAssociations.Add(new FilmPersonAssociation { FilmID
= film.FilmID, PersonID = PersonID, Association = "StageDirector" });
            }
        }
        if (AssociatedActorsID != null)
        {
            foreach (Int32 PersonID in AssociatedActorsID)
            {
                context.FilmPersonAssociations.Add(new FilmPersonAssociation { FilmID
= film.FilmID, PersonID = PersonID, Association = "Actor" });
            }
        }
        context.SaveChanges();
    }
    private void DeleteFilmAssociations(Film film)
    {
        foreach (FilmCountry fc in context.FilmCountries.Where(x => x.FilmID ==
film.FilmID))
        {
            context.FilmCountries.Remove(fc);
        }
        foreach (FilmGenre fg in context.FilmGenres.Where(x => x.FilmID ==
film.FilmID))
        {
            context.FilmGenres.Remove(fg);
        }
    }

```

```

        }
        foreach (FilmPersonAssociation fp in context.FilmPersonAssociations.Where(x
=> x.FilmID == film.FilmID))
        {
            context.FilmPersonAssociations.Remove(fp);
        }
    }
    private void DeletePersonAssociations(Person person)
    {
        foreach (FilmPersonAssociation fp in context.FilmPersonAssociations.Where(x
=> x.PersonID == person.PersonID))
        {
            context.FilmPersonAssociations.Remove(fp);
        }
    }
    public void DeleteFilm(Film film)
    {
        DeleteFilmAssociations(film);
        context.SaveChanges();
        context.Films.Remove(film);
        context.SaveChanges();
    }
}
}

```

Country.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ComponentModel.DataAnnotations;

namespace AllFilms.Domain.Entities
{
    [Table("Countries")]
    public class Country
    {
        public Int32 CountryID { get; set; }
        public String CountryName { get; set; }
    }
}

```

Film.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ComponentModel.DataAnnotations;
using System.Web.Mvc;
using System.ComponentModel;

namespace AllFilms.Domain.Entities
{
    [Table("Films")]
    public class Film
    {
        [HiddenInput(DisplayValue = false)]
        public Int32 FilmID { get; set; }

        [DisplayName("Название фильма")]
        [Required(ErrorMessage = "Введите название фильма")]
        public String Title { get; set; }
    }
}

```

```

        [DisplayName("Год выпуска")]
        [Required(ErrorMessage = "Введите год выпуска фильма")]
        public Int32? Year { get; set; }

        [DisplayName("Постер")]
        [HiddenInput(DisplayValue = false)]
        public Byte[] ImageData { get; set; }

        [HiddenInput(DisplayValue = false)]
        public String ImageMimeType { get; set; }

        [DisplayName("Описание")]
        [DataType(DataType.MultilineText)]
        public String Info { get; set; }
    }
}

```

FilmCountry.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ComponentModel.DataAnnotations;

namespace AllFilms.Domain.Entities
{
    [Table("FilmCountry")]
    public class FilmCountry
    {
        public Int32 ID { get; set; }
        public Int32 FilmID { get; set; }
        public Int32 CountryID { get; set; }
    }
}

```

FilmGenre.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ComponentModel.DataAnnotations;

namespace AllFilms.Domain.Entities
{
    [Table("FilmGenre")]
    public class FilmGenre
    {
        public Int32 ID { get; set; }
        public Int32 FilmID { get; set; }
        public Int32 GenreID { get; set; }
    }
}

```

FilmPersonAssociation.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ComponentModel.DataAnnotations;

```

```

namespace AllFilms.Domain.Entities
{
    [Table("FilmPersonAssociation")]
    public class FilmPersonAssociation
    {
        public Int32 ID { get; set; }
        public Int32 FilmID { get; set; }
        public Int32 PersonID { get; set; }
        public String Association { get; set; }
    }
}

```

Genre.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ComponentModel.DataAnnotations;

namespace AllFilms.Domain.Entities
{
    [Table("Genres")]
    public class Genre
    {
        public Int32 GenreID { get; set; }
        public String GenreName { get; set; }
    }
}

```

Person.cs

```

using System;
using System.Web.Mvc;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel;

namespace AllFilms.Domain.Entities
{
    [Table("People")]
    public class Person
    {
        [HiddenInput(DisplayValue = false)]
        public Int32 PersonId { get; set; }

        [DisplayName("Полное имя")]
        [Required(ErrorMessage = "Введите имя")]
        public String Fullname { get; set; }

        [DisplayName("Изображение")]
        [HiddenInput(DisplayValue = false)]
        public Byte[] ImageData { get; set; }

        [HiddenInput(DisplayValue = false)]
        public String ImageMimeType { get; set; }

        [DisplayName("Информация")]
        [DataType(DataType.MultilineText)]
        public String Info { get; set; }
    }
}

```

Изм.	Лист	№ докум.	Подпись	Дата

ResetCss.css

```
html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video {
    margin: 0;
    padding: 0;
    border: 0;
    font-size: 100%;
    font: inherit;
    vertical-align: baseline;
}
/* HTML5 display-role reset for older browsers */
article, aside, details, figcaption, figure,
footer, header, hgroup, menu, nav, section {
    display: block;
}
body {
    line-height: 1;
}
ol, ul {
    list-style: none;
}
blockquote, q {
    quotes: none;
}
blockquote:before, blockquote:after,
q:before, q:after {
    content: '';
    content: none;
}
table {
    border-collapse: collapse;
    border-spacing: 0;
}
```

Site.css

```
body
{
    margin: 0;
    padding: 0;
    font-family: Georgia, Times New Roman, Serif;
    color: Black;
    background-color: White;
    font-size: 14px;
}

header,
footer,
nav,
section {
```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.05.001 ПЗ

Лист

109

```

        display: block;
    }

    /* Styles for basic forms
    -----*/

    fieldset
    {
        border:1px solid #ddd;
        padding:0 1.4em 1.4em 1.4em;
        margin:0 0 1.5em 0;
    }

    legend
    {
        font-size:1.2em;
        font-weight: bold;
    }

    textarea
    {
        min-height: 75px;
    }

    .editor-label
    {
        margin: 1em 0 0.3em 0;
    }

    .editor-field
    {
        margin:0.5em 0 0 0;
    }

    /* Styles for validation helpers
    -----*/

    .field-validation-error
    {
        color: #ff0000;
    }

    .field-validation-valid
    {
        display: none;
    }

    .input-validation-error
    {
        border: 1px solid #ff0000;
        background-color: #ffebee;
    }

    .validation-summary-errors
    {
        font-weight: bold;
        color: #ff0000;
    }

    .validation-summary-valid
    {
        display: none;
    }

```

```

a
{
    text-decoration: none;
    color: rgb(78, 114, 154);
}

a:hover
{
    text-decoration: underline;
}

.clear
{
    clear: both;
}

/*Стилизация разметки*/
#page
{
    margin: 0px auto;
    width: 800px;
}

#header
{
    background-color: rgb(78, 114, 154);
    height: 40px;
    border: 2px solid rgb(60, 90, 110);
}

#columnLeft
{
    float: left;
    width: 200px;
    min-height: 200px;
    padding-top: 10px;
}

#columnRight
{
    float: right;
    width: 600px;
    min-height: 580px;
}

#footer
{
    clear: both;
    height: 40px;
    text-align: center;
    background-position: center;
    background-repeat: no-repeat;
    background-image: url("../Images/footer.png");
    width: 240px;
    margin: 0pt auto;
}

/*Стилизация меню*/
#mainmenu
{
    float: right;
    margin-right: 10px;
    height: 40px;
}

```

```

        color: White;
    }

    #mainmenu a
    {
        font-family: "Comic Sans MS","Trebuchet MS","Arial","Helvetica",sans-serif;
        display: block;
        height: 30px;
        padding-top: 10px;
        padding: 10px 5px 0px;
        float: left;
        color: White;
        text-decoration: none;
        margin: 0;
    }

    #mainmenu a:hover
    {
        background-color: rgb(60, 90, 110);
    }

    /*Стилизация заголовков*/
    h1,h2,h3,h4,h5,h6
    {
        text-align: center;
        font-family: "Comic Sans MS","Trebuchet MS","Arial","Helvetica",sans-serif;
        color: Gray;
    }

    h1
    {
        font-size: 40px;
    }

    h2
    {
        font-size: 35px;
    }

    h3
    {
        font-size: 20px;
    }

    /*Стилизация блока поиска*/
    #searchblock
    {
        clear: both;
        width: 370px;
        height: 20px;
        margin-right: 30px;
        float: right;
        background-color: rgb(78, 114, 154);
        border: 2px solid rgb(60, 90, 110);
        border-top: none;
        padding: 5px;
        margin-bottom: 20px;
    }

    .searchtextbox
    {
        padding: 0;
    }

```

```

        width: 325px;
        height: 20px;
    }

    .searchbutton
    {
        width: 20px;
        height: 20px;
        float: right;
        margin-right: 10px;
        padding: 2px;
        background: transparent url("../Images/searchicon.png") no-repeat center top;
        border: none;
    }

    #Logo
    {
        height: 40px;
    }

    .center
    {
        text-align: center;
    }

    DIV.pager
    {
        text-align: right;
        border-top: 2px solid Silver;
        padding-top: 5px;
        margin-top: 5px;
    }

    DIV.pager a
    {
        font-size: 15px;
        padding: 3px;
        color: #666;
    }

    DIV.pager a:hover
    {
        background-color: Silver;
    }

    DIV.pager a.selected
    {
        background-color: #353535;
        color: White;
    }

    .adminlinks
    {
        float: right;
    }

    .block
    {
        margin: 10px 0px;
        padding: 15px;
        border: 1px solid rgb(78, 114, 154);
    }

    .multi-line

```

```

{
    width: 100%;
}

.Message
{
    padding: 10px;
    margin: 3px 0;
    background-color: rgb(78, 114, 154);
    color: White;
    font-size: 18px;
}

.editor-field { margin-bottom: .8em; }
.editor-label { font-weight: bold; }
.editor-label:after { content: ":" }
.multi-line { height: 5em; font-family: Segoe UI, Verdana; }

/*Person*/
.content
{
    text-align: justify;
}
.content img
{
    margin-right: 5px;
    float: left;
}

.content h2
{
    margin-bottom: 5px;
}

div#genremenu
{
    width: 195px;
}

div#genremenu a
{
    font: bold 1.1em "ArrialNarrow", Arial;
    display: block;
    text-decoration: none;
    padding: 3px;
    color: Black;
    border-bottom: 1px dotted silver;
}

div#genremenu a.selected { background-color: #666; color: White; }
div#genremenu a:hover { background-color: #CCC; }
div#genremenu a.selected:hover { background-color: #666; }

table.Grid td, table.Grid th
{
    text-align: left;
    padding: 2px;
}

table.Grid
{
    border-collapse: collapse;
    width: 100%;
}

```

```

}

table.Grid td
{
    border: 1px solid #ccc;
    padding: 3px;
}

table.Grid tr
{
    background-color: #eee;
}

table.Grid tr:nth-child(odd)
{
    background-color: #ddd;
}

td.admintd
{
    text-align: right;
    width: 70px;
}

td.admintd a,
td.admintd input
{
    margin: 0 3px;
}

span.links a
{
    margin-right: 5px;
    float: left;
}

#List
{
    padding: 5px;
}

#LogOn
{
    margin: 0px auto;
    width: 300px;
    text-align: center;
}

strong
{
    font-weight: bold;
}

SiteCss3.css

#header, #searchblock
{
    border-bottom-left-radius: 7px;
    border-bottom-right-radius: 7px;
    box-shadow: 0px 0px 10px rgb(78, 114, 154);
}

#footer

```

```

{
    box-shadow: 0px 0px 10px rgb(78, 114, 154);
}

.block
{
    border-radius: 15px;
}

```

AccountController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using AllFilms.WebUI.Infrastructure.Abstract;
using AllFilms.WebUI.Models;

namespace AllFilms.WebUI.Controllers
{
    public class AccountController : Controller
    {
        IAuthProvider authProvider;

        public AccountController(IAuthProvider auth)
        {
            authProvider = auth;
        }

        public ActionResult LogOn()
        {
            return View();
        }

        [HttpPost]
        public ActionResult LogOn(LogOnViewModel model, String returnUrl)
        {
            if (ModelState.IsValid)
            {
                if (authProvider.Authenticate(model.UserName, model.Password))
                {
                    return Redirect(returnUrl ?? Url.Action("List", "Person"));
                }
                else
                {
                    ModelState.AddModelError("", "Неверное имя пользователя или пароль");
                    return View();
                }
            }
            else
            {
                return View();
            }
        }
    }
}

```

FilmController.cs

```

using System;
using System.Collections.Generic;

```

					ПГУ 1.090501.05.001 ПЗ	Лист
						116
Изм.	Лист	№ докум.	Подпись	Дата		

```

using System.Linq;
using System.Web;
using System.Web.Mvc;
using AllFilms.Domain.Abstract;
using AllFilms.WebUI.Models;
using AllFilms.Domain.Entities;

namespace AllFilms.WebUI.Controllers
{
    public class FilmController : Controller
    {
        IRepository repository;
        Int32 PageSize = 50;

        public FilmController(IRepository repo)
        {
            repository = repo;
        }

        public ActionResult List(Int32 page = 1, Boolean greed = false, Int32[] genres =
null,
        Int32[] countries = null, Int32[] people = null,
        Int32? fromyear = null, Int32? toyear = null, String qtitle = null)
        {
            SearchModel sm = (SearchModel)TempData["SearchModel"];
            if (sm != null)
            {
                qtitle = sm.QTitle;
                if (sm.GenresAssociations != null)
                {
                    genres = sm.GenresAssociations.SelectedGenresID;
                }
                if (sm.CountriesAssociations != null)
                {
                    countries = sm.CountriesAssociations.SelectedCountriesID;
                }
                if (sm.PeopleAssociations != null)
                {
                    people = sm.PeopleAssociations.SelectedPeopleID;
                }
                fromyear = sm.FromYear;
                toyear = sm.ToYear;
                greed = sm.Greed;
            }
            IEnumerable<Film> SelectedFilms = repository.Films;
            SelectedFilms = SelectedFilms.Where(f => qtitle == null ||
f.Title.IndexOf(qtitle, StringComparison.CurrentCultureIgnoreCase) != -1);
            SelectedFilms = SelectedFilms.Where(f => fromyear == null || f.Year >=
fromyear).Where(f => toyear == null || f.Year <= toyear);
            if (greed)
            {
                if (genres != null)
                {
                    foreach (Int32 GenreID in genres)
                    {
                        SelectedFilms = (from films in SelectedFilms
join filmgenres in repository.FilmGenres on
films.FilmID equals filmgenres.FilmID
where filmgenres.GenreID == GenreID
select films).ToArray();
                    }
                }
                if (countries != null)
                {

```

					ПГУ 1.090501.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		117

```

        foreach (Int32 CountryID in countries)
        {
            SelectedFilms = (from films in SelectedFilms
                              join filmcountries in repository.FilmCountries on
films.FilmID equals filmcountries.FilmID
                              where filmcountries.CountryID == CountryID
                              select films).ToArray();
        }
    }
    if (people != null)
    {
        foreach (Int32 PersonID in people)
        {
            SelectedFilms = (from films in SelectedFilms
                              join filmpeople in
repository.FilmPersonAssociations on films.FilmID equals filmpeople.FilmID
                              where filmpeople.PersonID == PersonID
                              select films).ToArray();
        }
    }
}
else
{
    if (genres != null)
    {
        SelectedFilms = (from films in SelectedFilms
                          join filmgenres in repository.FilmGenres on
films.FilmID equals filmgenres.FilmID
                          where genres.Contains(filmgenres.GenreID)
                          select films).ToArray();
    }
    if (countries != null)
    {
        SelectedFilms = (from films in SelectedFilms
                          join filmcountries in repository.FilmCountries on
films.FilmID equals filmcountries.FilmID
                          where countries.Contains(filmcountries.CountryID)
                          select films).ToArray();
    }
    if (people != null)
    {
        SelectedFilms = (from films in SelectedFilms
                          join filmpeople in repository.FilmPersonAssociations
on films.FilmID equals filmpeople.FilmID
                          where people.Contains(filmpeople.PersonID)
                          select films).ToArray();
    }
}
SelectedFilms = SelectedFilms.Distinct();
SelectedFilms = SelectedFilms.OrderBy(f => f.Title).Skip((page - 1) *
PageSize).Take(PageSize);
List<FilmDetailViewModel> filmlist = new List<FilmDetailViewModel>();
foreach (var film in SelectedFilms)
{
    IEnumerable<Genre> fg = from g in repository.Genres
                            join filmgenres in repository.FilmGenres on
g.GenreID equals filmgenres.GenreID
                            where filmgenres.FilmID == film.FilmID
                            select g;
    FilmDetailViewModel details = new FilmDetailViewModel()
    {
        Film = film,
        Genres = fg.ToList()
    };
}

```

```

        filmlist.Add(details);
    }
    FilmListViewModel viewModel = new FilmListViewModel()
    {
        Films = filmlist,
        PagingInfo = new PagingInfo()
        {
            CurrentPage = page,
            ItemsPerPage = PageSize,
            TotalItems = SelectedFilms.Count()
        }
    };
    ViewBag.Genres = genres;
    ViewBag.Countries = countries;
    ViewBag.People = people;
    ViewBag.FromYear = fromyear;
    ViewBag.ToYear = toyear;
    ViewBag.QTitle = qtitle;
    ViewBag.Greed = greed;
    return View(viewModel);
}

public ViewResult Details(Int32 id)
{
    Film film = repository.Films.FirstOrDefault(f => f.FilmID == id);
    IEnumerable<Genre> fg = from genres in repository.Genres
                           join filmgenres in repository.FilmGenres on
genres.GenreID equals filmgenres.GenreID
                           where filmgenres.FilmID == film.FilmID
                           select genres;
    IEnumerable<Person> fa = from actors in repository.People
                           join filmactors in repository.FilmPersonAssociations
on actors.PersonID equals filmactors.PersonID
                           where filmactors.FilmID == film.FilmID &&
filmactors.Association == "Actor"
                           select actors;
    IEnumerable<Person> fsd = from stagedirectors in repository.People
                             join filmactors in
repository.FilmPersonAssociations on stagedirectors.PersonID equals filmactors.PersonID
                             where filmactors.FilmID == film.FilmID &&
filmactors.Association == "StageDirector"
                             select stagedirectors;
    IEnumerable<Country> fc = from countries in repository.Countries
                             join filmcountries in repository.FilmCountries on
countries.CountryID equals filmcountries.CountryID
                             where filmcountries.FilmID == film.FilmID
                             select countries;
    FilmDetailViewModel details = new FilmDetailViewModel()
    {
        Film = film,
        Genres = fg,
        StageDirectors = fsd,
        Actors = fa,
        Countries = fc
    };
    return View(details);
}

[Authorize]
[HttpGet]
public ViewResult Edit(Int32 id)
{
    Film film = repository.Films.FirstOrDefault(f => f.FilmID == id);
    Int32[] selectedgenres = null;

```

					ПГУ 1.090501.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		119

```

        Int32[] selectedactors = null;
        Int32[] selectedstagedirectors = null;
        Int32[] selectedcountries = null;
        if (film != null)
        {
            selectedgenres = (from g in repository.Genres
                               join filmgenres in repository.FilmGenres on g.GenreID
                               equals filmgenres.GenreID
                               where filmgenres.FilmID == film.FilmID
                               select g.GenreID).ToArray();

            selectedactors = (from a in repository.People
                               join filmactors in repository.FilmPersonAssociations on
                               a.PersonID equals filmactors.PersonID
                               where filmactors.FilmID == film.FilmID &&
                               filmactors.Association == "Actor"
                               select a.PersonID).ToArray();

            selectedstagedirectors = (from sd in repository.People
                                       join filmstagedirectors in
                                       repository.FilmPersonAssociations on sd.PersonID equals filmstagedirectors.PersonID
                                       where filmstagedirectors.FilmID == film.FilmID
                                       && filmstagedirectors.Association == "StageDirector"
                                       select sd.PersonID).ToArray();

            selectedcountries = (from countries in repository.Countries
                                  join filmcountries in repository.FilmCountries on
                                  countries.CountryID equals filmcountries.CountryID
                                  where filmcountries.FilmID == film.FilmID
                                  select countries.CountryID).ToArray();
        }
        FilmEditViewModel details = new FilmEditViewModel()
        {
            Film = film,
            GenresAssociations = new GenresAssociationsViewModel()
            {
                Genres = repository.Genres,
                SelectedGenresID = selectedgenres
            },
            StageDirectorsAssociations = new PeopleAssociationsViewModel()
            {
                People = repository.People,
                SelectedPeopleID = selectedstagedirectors
            },
            ActorsAssociations = new PeopleAssociationsViewModel()
            {
                People = repository.People,
                SelectedPeopleID = selectedactors
            },
            CountriesAssociations = new CountriesAssociationsViewModel()
            {
                Countries = repository.Countries,
                SelectedCountriesID = selectedcountries
            }
        };
        return View(details);
    }

    [Authorize]
    [HttpPost]
    public ActionResult Edit(FilmEditViewModel details, HttpPostedFileBase image)
    {
        if (ModelState.IsValid)
        {
            if (image != null && image.ContentLength > 0)

```

					ПГУ 1.090501.05.001 ПЗ	Лист
						120
Изм.	Лист	№ докум.	Подпись	Дата		

```

        {
            details.Film.ImageMimeType = image.ContentType;
            details.Film.ImageData = new Byte[image.ContentLength];
            image.InputStream.Read(details.Film.ImageData, 0,
image.ContentLength);
        }
        Int32[] SelectedGenresID = null;
        Int32[] SelectedStageDirectorsID = null;
        Int32[] SelectedActorsID = null;
        Int32[] SelectedCountriesID = null;
        if (details.GenresAssociations != null)
        {
            SelectedGenresID = details.GenresAssociations.SelectedGenresID;
        }
        if (details.StageDirectorsAssociations != null)
        {
            SelectedStageDirectorsID =
details.StageDirectorsAssociations.SelectedPeopleID;
        }
        if (details.ActorsAssociations != null)
        {
            SelectedActorsID = details.ActorsAssociations.SelectedPeopleID;
        }
        if (details.CountriesAssociations != null)
        {
            SelectedCountriesID =
details.CountriesAssociations.SelectedCountriesID;
        }
        repository.SaveFilm(details.Film, SelectedCountriesID, SelectedGenresID,
SelectedStageDirectorsID, SelectedActorsID);
        TempData["Message"] = String.Format("{0} сохранен", details.Film.Title);
        return RedirectToAction("List");
    }
    else
    {
        return View(details);
    }
}

public FileContentResult GetImage(Int32 filmId)
{
    Film film = repository.Films.FirstOrDefault(f => f.FilmID == filmId);
    if (film != null)
    {
        return File(film.ImageData, film.ImageMimeType);
    }
    else
    {
        return null;
    }
}

[Authorize]
public ViewResult Create()
{
    FilmEditViewModel details = new FilmEditViewModel()
    {
        Film = new Film(),
        GenresAssociations = new GenresAssociationsViewModel()
        {
            Genres = repository.Genres,
            SelectedGenresID = new Int32[1]
        },
        ActorsAssociations = new PeopleAssociationsViewModel()
    }
}

```

```

        {
            People = repository.People,
            SelectedPeopleID = new Int32[1]
        },
        StageDirectorsAssociations = new PeopleAssociationsViewModel()
        {
            People = repository.People,
            SelectedPeopleID = new Int32[1]
        },
        CountriesAssociations = new CountriesAssociationsViewModel()
        {
            Countries = repository.Countries,
            SelectedCountriesID = new Int32[1]
        }
    };
    return View("Edit", details);
}

[HttpPost]
public ActionResult Delete(Int32 id)
{
    Film film = repository.Films.FirstOrDefault(f => f.FilmID == id);
    if (film != null)
    {
        repository.DeleteFilm(film);
        TempData["message"] = String.Format("{0} was deleted", film.Title);
    }
    return RedirectToAction("List");
}
}
}

```

NavigationController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using AllFilms.Domain.Abstract;

namespace AllFilms.WebUI.Controllers
{
    public class NavigationController : Controller
    {
        IRepository repository;

        public NavigationController(IRepository repo)
        {
            repository = repo;
        }

        public PartialViewResult GenresMenu()
        {
            return PartialView(repository.Genres.OrderBy(g => g.GenreName));
        }

        public PartialViewResult MainMenu()
        {
            return PartialView();
        }
    }
}

```

PersonController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using AllFilms.Domain.Abstract;
using AllFilms.WebUI.Models;
using AllFilms.Domain.Entities;
using System.Text;

namespace AllFilms.WebUI.Controllers
{
    public class PersonController : Controller
    {
        IRepository repository;
        Int32 PageSize = 50;

        public PersonController(IRepository repo)
        {
            repository = repo;
        }

        public ViewResult List(Int32 page = 1)
        {
            PersonListViewModel viewModel = new PersonListViewModel
            {
                People = repository.People
                    .OrderBy(p => p.Fullname)
                    .Skip((page - 1) * PageSize)
                    .Take(PageSize),
                PagingInfo = new PagingInfo
                {
                    CurrentPage = page,
                    ItemsPerPage = PageSize,
                    TotalItems = repository.People.Count()
                }
            };
            return View(viewModel);
        }

        public ViewResult Details(Int32 id)
        {
            Person person = repository.People.FirstOrDefault(p => p.PersonId == id);
            IEnumerable<Film> af = from film in repository.Films
                                join filmassociation in
repository.FilmPersonAssociations on film.FilmID equals filmassociation.FilmID
                                where filmassociation.PersonID == person.PersonID
                                select film;
            PersonDetailViewModel details = new PersonDetailViewModel()
            {
                Person = person,
                AssociatedFilms = af
            };
            return View(details);
        }

        [Authorize]
        public ViewResult Edit(Int32 id)
        {
            Person person = repository.People.FirstOrDefault(p => p.PersonId == id);
            return View(person);
        }
    }
}
```

					ПГУ 1.090501.05.001 ПЗ	Лист
						123
Изм.	Лист	№ докум.	Подпись	Дата		

```

[Authorize]
public ActionResult Create()
{
    return View("Edit", new Person());
}

[Authorize]
[HttpPost]
public ActionResult Edit(Person person, HttpPostedFileBase image)
{
    if (ModelState.IsValid)
    {
        if (image != null && image.ContentLength > 0)
        {
            person.ImageMimeType = image.ContentType;
            person.ImageData = new Byte[image.ContentLength];
            image.InputStream.Read(person.ImageData, 0, image.ContentLength);
        }
        repository.SavePerson(person);
        TempData["Message"] = String.Format("{0} сохранен", person.Fullname);
        return RedirectToAction("List");
    }
    else
    {
        return View(person);
    }
}

public FileContentResult GetImage(Int32 personId)
{
    Person person = repository.People.FirstOrDefault(p => p.PersonId ==
personId);
    if (person != null)
    {
        return File(person.ImageData, person.ImageMimeType);
    }
    else
    {
        return null;
    }
}

[Authorize]
public ActionResult Delete(Int32 id)
{
    Person person = repository.People.FirstOrDefault(p => p.PersonId == id);
    if (person != null)
    {
        repository.DeletePerson(person);
        TempData["message"] = String.Format("{0} was deleted", person.Fullname);
    }
    return RedirectToAction("List");
}
}
}

```

SearchController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

```

					ПГУ 1.090501.05.001 ПЗ	Лист
						124
Изм.	Лист	№ докум.	Подпись	Дата		

```

using AllFilms.Domain.Abstract;
using AllFilms.Domain.Entities;
using AllFilms.WebUI.Models;

namespace AllFilms.WebUI.Controllers
{
    public class SearchController : Controller
    {
        IRepository repository;

        public SearchController(IRepository repo)
        {
            repository = repo;
        }

        public PartialViewResult Search()
        {
            return PartialView();
        }

        [HttpGet]
        public ViewResult SuperSearch()
        {
            SearchModel sm = new SearchModel()
            {
                GenresAssociations = new GenresAssociationsViewModel()
                {
                    Genres = repository.Genres,
                    SelectedGenresID = new Int32[1]
                },
                CountriesAssociations = new CountriesAssociationsViewModel()
                {
                    Countries = repository.Countries,
                    SelectedCountriesID = new Int32[1]
                },
                PeopleAssociations = new PeopleAssociationsViewModel()
                {
                    People = repository.People,
                    SelectedPeopleID = new Int32[1]
                },
                Greed = false
            };
            return View(sm);
        }

        [HttpPost]
        public ActionResult SuperSearch(SearchModel sm)
        {
            TempData["SearchModel"] = sm;
            return RedirectToAction("List", "Film");
        }
    }
}

```

PagingHelpers.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using AllFilms.WebUI.Models;
using System.Text;

```

					ПГУ 1.090501.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		125

```

namespace AllFilms.WebUI.HtmlHelpers
{
    public static class PagingHelpers
    {
        public static MvcHtmlString PageLinks(this HtmlHelper html,
            PagingInfo pagingInfo, Func<Int32, String> pageUrl)
        {
            StringBuilder result = new StringBuilder();
            for (Int32 i = 1; i <= pagingInfo.TotalPages; i++)
            {
                TagBuilder tag = new TagBuilder("a");
                tag.MergeAttribute("href", pageUrl(i));
                tag.InnerHtml = i.ToString();
                if (i == pagingInfo.CurrentPage)
                {
                    tag.AddCssClass("selected");
                }
                result.Append(tag.ToString());
            }
            return MvcHtmlString.Create(result.ToString());
        }
    }
}

```

IAuthProvider.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace AllFilms.WebUI.Infrastructure.Abstract
{
    public interface IAuthProvider
    {
        Boolean Authenticate(String username, String password);
    }
}

```

FormsAuthProvider.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using AllFilms.WebUI.Infrastructure.Abstract;
using System.Web.Security;

namespace AllFilms.WebUI.Infrastructure.Concrete
{
    public class FormsAuthProvider : IAuthProvider
    {
        public Boolean Authenticate(String username, String password)
        {
            Boolean result = FormsAuthentication.Authenticate(username, password);
            if (result)
            {
                FormsAuthentication.SetAuthCookie(username, false);
            }
            return result;
        }
    }
}

```

					ПГУ 1.090501.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		126

NinjectControllersFactory.cs

```
using System;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Web.Mvc;
using System.Web.Routing;
using Ninject;
using AllFilms.Domain.Abstract;
using AllFilms.Domain.Entities;
using AllFilms.Domain.Concrete;
using Moq;
using AllFilms.WebUI.Infrastructure.Abstract;
using AllFilms.WebUI.Infrastructure.Concrete;

namespace AllFilms.WebUI.Infrastructure
{
    public class NinjectControllerFactory : DefaultControllerFactory
    {
        private IKernel ninject;

        public NinjectControllerFactory()
        {
            ninject = new StandardKernel();
            AddBindings();
        }

        protected override IController GetControllerInstance(RequestContext
requestContext, Type controllerType)
        {
            return controllerType == null
                ? null
                : (IController)ninject.Get(controllerType);
        }

        private void AddBindings()
        {
            ninject.Bind<IRepository>().To<EFRepository>();
            ninject.Bind<IAuthProvider>().To<FormsAuthProvider>();
        }
    }
}
```

CountriesAssociationsViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using AllFilms.Domain.Entities;

namespace AllFilms.WebUI.Models
{
    public class CountriesAssociationsViewModel
    {
        public IEnumerable<Country> Countries { get; set; }
        public Int32[] SelectedCountriesID { get; set; }
    }
}
```

Изм.	Лист	№ докум.	Подпись	Дата

FilmDetailViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using AllFilms.Domain.Entities;

namespace AllFilms.WebUI.Models
{
    public class FilmDetailViewModel
    {
        public Film Film { get; set; }
        public IEnumerable<Genre> Genres { get; set; }
        public IEnumerable<Person> StageDirectors { get; set; }
        public IEnumerable<Person> Actors { get; set; }
        public IEnumerable<Country> Countries { get; set; }
    }
}
```

FilmEditViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using AllFilms.Domain.Entities;
using System.ComponentModel;

namespace AllFilms.WebUI.Models
{
    public class FilmEditViewModel
    {
        public Film Film { get; set; }
        public CountriesAssociationsViewModel CountriesAssociations { get; set; }
        public GenresAssociationsViewModel GenresAssociations { get; set; }
        public PeopleAssociationsViewModel StageDirectorsAssociations { get; set; }
        public PeopleAssociationsViewModel ActorsAssociations { get; set; }
    }
}
```

FilmListViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using AllFilms.Domain.Entities;

namespace AllFilms.WebUI.Models
{
    public class FilmListViewModel
    {
        public IEnumerable<FilmDetailViewModel> Films { get; set; }
        public PagingInfo PagingInfo { get; set; }
    }
}
```

GenresAssociationsViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using AllFilms.Domain.Entities;
using System.ComponentModel.DataAnnotations;

namespace AllFilms.WebUI.Models
{
    public class GenresAssociationsViewModel
    {
        public IEnumerable<Genre> Genres { get; set; }
        public Int32[] SelectedGenresID { get; set; }
    }
}
```

LogOnViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel;

namespace AllFilms.WebUI.Models
{
    public class LogOnViewModel
    {
        [Required]
        [DisplayName("Имя пользователя")]
        public String UserName { get; set; }

        [Required]
        [DisplayName("Пароль")]
        [DataType(DataType.Password)]
        public String Password { get; set; }
    }
}
```

PagingInfo.cs

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

namespace AllFilms.WebUI.Models
{
    public class PagingInfo
    {
        public Int32 TotalItems { get; set; }
        public Int32 ItemsPerPage { get; set; }
        public Int32 CurrentPage { get; set; }

        public Int32 TotalPages
        {
            get
            {
            }
        }
    }
}
```

					ПГУ 1.090501.05.001 ПЗ	Лист
						129
Изм.	Лист	№ докум.	Подпись	Дата		

```

        return (Int32)Math.Ceiling((Decimal)TotalItems / ItemsPerPage);
    }
}
}

```

PeopleAssociationsViewModel.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using AllFilms.Domain.Entities;

namespace AllFilms.WebUI.Models
{
    public class PeopleAssociationsViewModel
    {
        public IEnumerable<Person> People { get; set; }
        public Int32[] SelectedPeopleID { get; set; }
    }
}

```

PersonDetailViewModel.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using AllFilms.Domain.Entities;

namespace AllFilms.WebUI.Models
{
    public class PersonDetailViewModel
    {
        public Person Person { get; set; }
        public IEnumerable<Film> AssociatedFilms { get; set; }
    }
}

```

PersonListViewModel.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using AllFilms.Domain.Entities;

namespace AllFilms.WebUI.Models
{
    public class PersonListViewModel
    {
        public IEnumerable<Person> People { get; set; }
        public PagingInfo PagingInfo { get; set; }
    }
}

```

SearchModel.cs

					ПГУ 1.090501.05.001 ПЗ	Лист
						130
Изм.	Лист	№ докум.	Подпись	Дата		

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel;

namespace AllFilms.WebUI.Models
{
    public class SearchModel
    {
        [DisplayName("Название фильма")]
        public String QTitle { get; set; }
        [DisplayName("Начиная с (года)")]
        public Int32? FromYear { get; set; }
        [DisplayName("До (года)")]
        public Int32? ToYear { get; set; }
        [DisplayName("Жадный поиск")]
        public Boolean Greed { get; set; }
        public GenresAssociationsViewModel GenresAssociations { get; set; }
        public CountriesAssociationsViewModel CountriesAssociations { get; set; }
        public PeopleAssociationsViewModel PeopleAssociations { get; set; }
    }
}

```

Account/LogOn.cshtml

```

@model AllFilms.WebUI.Models.LogOnViewModel

@{
    ViewBag.Title = "Вход";
}

<div id="LogOn">
    <h2>Вход</h2>
    @using (Html.BeginForm())
    {
        @Html.ValidationSummary(true)
        @Html.EditorForModel()
        <p style="margin: 3px;"><input type="submit" value="Вход" /></p>
    }
</div>

```

Film/Details.cshtml

```

@model AllFilms.WebUI.Models.FilmDetailViewModel

@{
    ViewBag.Title = "Details";
}

<div class="content">
    <h2>@Model.Film.Title</h2>
    @if (Model.Film.ImageData == null)
    {
        
    }
    else
    {
        
    }
}

```

					ПГУ 1.090501.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		131

```

}
<div style="line-height: 20px;">
  <div>
    <span style="font-weight: bold">Год выпуска:</span> @Model.Film.Year
  </div>
  @if (Model.Countries.Count() > 0)
  {
    <div style="display:table">
      <span style="font-weight: bold; float: left;">Страна:</span>
      @Html.Partial("CountriesLinks", Model.Countries)
    </div>
  }
  @if (Model.Genres.Count() > 0)
  {
    <div style="display:table">
      <span style="font-weight: bold; float: left;">Жанр:</span>
      @Html.Partial("GenresLinks", Model.Genres)
    </div>
  }
  @if (Model.StageDirectors.Count() > 0)
  {
    <div>
      <span style="font-weight: bold">Режисеры:</span>
      @Html.Partial("PeopleLinks", Model.StageDirectors)
    </div>
  }
  @if (Model.Actors.Count() > 0)
  {
    <div>
      <span style="font-weight: bold">Актеры:</span>
      @Html.Partial("PeopleLinks", Model.Actors)
    </div>
  }
  <div>
    <span style="font-weight: bold">Информация:</span> @Model.Film.Info
  </div>
</div>
</div>
@if (Request.IsAuthenticated)
{
  <div class="adminlinks clear">
    <table class="Grid">
      <tr>
        <td class="admintd">@Html.ActionLink("Изменить", "Edit", new { id =
Model.Film.FilmID })</td>
        <td class="admintd">
          @using (Html.BeginForm("Delete", "Film"))
          {
            @Html.Hidden("id", Model.Film.FilmID)
            <input type="submit" value="Удалить" />
          }
        </td>
      </tr>
    </table>
  </div>
}

```

Film/Edit.cshtml

@model AllFilms.WebUI.Models.FilmEditViewModel

```

@{
    ViewBag.Title = "Edit";
}

```

					ПГУ 1.090501.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		132

```

<h2>Edit @Model.Film.Title</h2>
<div class="block">
@using (Html.BeginForm("Edit", "Film", FormMethod.Post, new { enctype = "multipart/form-
data" }))
{
    @Html.EditorFor(x => x.Film.FilmID)
    @Html.EditorFor(x => x.Film.ImageData)
    @Html.EditorFor(x => x.Film.ImageMimeType)
    @Html.ValidationSummary()
    <div class="editor-label">
        @Html.LabelFor(x => x.Film.Title)
    </div>
    @Html.EditorFor(x => x.Film.Title)
    <div class="editor-label">
        @Html.LabelFor(x => x.Film.Year)
    </div>
    <div class="editor-field">
        @Html.EditorFor(x => x.Film.Year)
    </div>
    <div class="editor-label">
        @Html.LabelFor(x => x.Film.ImageData)
    </div>
    <div class="editor-field">
        @if (Model.Film.ImageData == null)
        {
            
        }
        else
        {
            
        }
        <div class="editor-label">Загрузить новое изображение</div>
        <input type="file" name="Image" />
    </div>
    <div class="editor-label">
        Страны производители
    </div>
    <div class="editor-field">
        @Html.EditorFor(x => x.CountriesAssociations)
    </div>
    <div class="editor-label">
        Жанры
    </div>
    <div class="editor-field">
        @Html.EditorFor(x => x.GenresAssociations)
    </div>
    <div class="editor-label">
        Режиссеры
    </div>
    <div class="editor-field">
        @Html.EditorFor(x => x.StageDirectorsAssociations)
    </div>
    <div class="editor-label">
        Актеры
    </div>
    <div class="editor-field">
        @Html.EditorFor(x => x.ActorsAssociations)
    </div>
    <div class="editor-label">
        @Html.LabelFor(x => x.Film.Info)
    </div>
    <div class="editor-field">

```

					ПГУ 1.090501.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		133

```

        @Html.EditorFor(x => x.Film.Info)
    </div>
    <div class="admintd" style="margin: 10px 0 0 0;">
        <input type="submit" value="Сохранить" />
        @Html.ActionLink("Отмена", "List")
    </div>
}
</div>

```

Film/List.cshtml

```

@model AllFilms.WebUI.Models.FilmListViewModel
@using AllFilms.WebUI.HtmlHelpers
@{
    ViewBag.Title = "Список фильмов";
    Boolean IsAuthenticated = Request.IsAuthenticated;
}

<h2>Список фильмов</h2>
@if (Model.Films.Count() == 0)
{
    <div style="text-align:center; margin-top: 10px; font-size: larger;">Список
пуст</div>
}
else
{
    <div id="List">
        @if (IsAuthenticated)
        {
            <div style="margin: 5px 0px 15px 0px;">
                @Html.ActionLink("Добавить фильм", "Create")
            </div>
        }
        <table class="Grid">
            <tr>
                <td><strong>Название</strong></td>
                <td><strong>Год выпуска</strong></td>
                <td><strong>Жанр</strong></td>
                @if (IsAuthenticated)
                {
                    <td></td>
                    <td></td>
                }
            </tr>
            @foreach (var f in Model.Films)
            {
                <tr>
                    <td>@Html.ActionLink(f.Film.Title, "Details", new { id = f.Film.FilmID
})</td>
                    <td>@f.Film.Year</td>
                    <td>@Html.Partial("GenresLinks", f.Genres)</td>
                    @if (IsAuthenticated)
                    {
                        <td class="admintd">@Html.ActionLink("Изменить", "Edit", new { id =
f.Film.FilmID })</td>
                        <td class="admintd">
                            @using (Html.BeginForm("Delete", "Film"))
                            {
                                @Html.Hidden("id", f.Film.FilmID)
                                <input type="submit" value="Delete" />
                            }
                        </td>
                    }
                </tr>
            }
        </table>
    </div>
}

```

```

    }
    </table>
</div>
<div class="pager">
@Html.PagedListLinks(Model.PagingInfo, x =>
{
    String result = Url.Action("List", new { page = x, fromyear = ViewBag.FromYear,
toyear = ViewBag.ToYear, qtitle = ViewBag.QTitle, greed = ViewBag.Greed });
    if (ViewBag.Genres != null)
    {
        foreach (Int32 genreID in ViewBag.Genres)
        {
            result += "&genres=" + genreID;
        }
    }
    if (ViewBag.Countries != null)
    {
        foreach (Int32 countryID in ViewBag.Countries)
        {
            result += "&countries=" + countryID;
        }
    }
    if (ViewBag.People != null)
    {
        foreach (Int32 personID in ViewBag.People)
        {
            result += "&people=" + personID;
        }
    }
    return result;
})
</div>
}

```

Navigation/GenresMenu.cshtml

```

@model IEnumerable<AllFilms.Domain.Entities.Genre>

@{
    Layout = null;
}

@foreach (var g in Model)
{
    @Html.RouteLink(g.GenreName,
        new
        {
            controller = "Film",
            action = "List",
            genres = g.GenreID
        }
    )
}

```

Navigation/MainMenu.cshtml

```

@Html.ActionLink("Фильмы", "List", "Film")
@Html.ActionLink("Актеры и режиссеры", "List", "Person")
@Html.ActionLink("Поиск", "SuperSearch", "Search")
@if (!Request.IsAuthenticated)
{

```

```

    @Html.ActionLink("Вход", "LogOn", "Account")
}

```

Person/Details.cshtml

```
@model AllFilms.WebUI.Models.PersonDetailViewModel
```

```

@{
    ViewBag.Title = Model.Person.Fullname;
}
<div class="content">
    <h2>@Model.Person.Fullname</h2>
    @if (Model.Person.ImageData == null)
    {
        
    }
    else
    {
        
    }
    <div style="line-height: 20px;">
        @if (Model.Person.Info != null)
        {
            <div>
                <span style="font-weight: bold">Информация:</span> @Model.Person.Info
            </div>
        }
        @if (Model.AssociatedFilms.Count() > 0)
        {
            <div>
                <span style="font-weight: bold">Принимал участие в:</span>
            </div>
            <div>
                <ul>
                    @foreach (var film in Model.AssociatedFilms)
                    {
                        <li>@Html.ActionLink(film.Title + " (" + film.Year + ")", "Details",
"Film", new { id = film.FilmID }, null)</li>
                    }
                </ul>
            </div>
        }
    </div>
</div>
@if (Request.IsAuthenticated)
{
    <div class="adminlinks clear">
        <table class="Grid">
            <tr>
                <td class="admintd">@Html.ActionLink("Изменить", "Edit", new { id =
Model.Person.PersonId })</td>
                <td class="admintd">
                    @using (Html.BeginForm("Delete", "Person"))
                    {
                        @Html.Hidden("id", Model.Person.PersonId)
                        <input type="submit" value="Удалить" />
                    }
                </td>
            </tr>
        </table>
    </div>
}

```

Person/Edit.cshtml

```
@model AllFilms.Domain.Entities.Person

@{
    ViewBag.Title = "Edit";
}

<h2>Edit @Model.Fullname</h2>
<div class="block">
    @using (Html.BeginForm("Edit", "Person", FormMethod.Post, new { enctype =
    "multipart/form-data" }))
    {
        @Html.EditorFor(x => x.PersonId)
        @Html.EditorFor(x => x.ImageData)
        @Html.EditorFor(x => x.ImageMimeType)

        @Html.ValidationSummary()
        <div class="editor-label">
            @Html.LabelFor(x => x.Fullname)
        </div>
        @Html.EditorFor(x => x.Fullname)
        <div class="editor-label">
            @Html.LabelFor(x => x.ImageData)
        </div>
        <div class="editor-field">
            @if (Model.ImageData == null)
            {
                
            }
            else
            {
                
            }
            <div class="editor-label">Загрузить новое изображение</div>
            <input type="file" name="Image" />
        </div>
        <div class="editor-label">
            @Html.LabelFor(x => x.Info)
        </div>
        @Html.EditorFor(x => x.Info)
        <div class="admind" style="margin: 10px 0 0 0;">
            <input type="submit" value="Сохранить" />
            @Html.ActionLink("Отмена", "List")
        </div>
    }
</div>
```

Person/List.cshtml

```
@model AllFilms.WebUI.Models.PersonListViewModel
@using AllFilms.WebUI.HtmlHelpers

@{
    ViewBag.Title = "People";
    Boolean IsAuthenticated = Request.IsAuthenticated;
}

<h2>Список людей</h2>
<div id="List">
    @if (IsAuthenticated)
    {
```

					ПГУ 1.090501.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		137

```

        <div style="margin: 5px 0px 15px 0px;">
            @Html.ActionLink("Добавить человека", "Create")
        </div>
    }
    <table class="Grid">
        @foreach (var p in Model.People)
        {
            <tr>
                <td>@Html.ActionLink(p.Fullname, "Details", new { id = p.PersonId })</td>
                @if (IsAuthenticated)
                {
                    <td class="admintd">@Html.ActionLink("Изменить", "Edit", new { id =
p.PersonId })</td>
                    <td class="admintd">
                        @using(Html.BeginForm("Delete", "Person"))
                        {
                            @Html.Hidden("id", p.PersonId)
                            <input type="submit" value="Delete" />
                        }
                    </td>
                }
            </tr>
        }
    </table>
</div>

<div class="pager">
    @Html.PagedListLinks(Model.PagingInfo, x => Url.Action("List", new { page = x }))
</div>

```

Search.cshtml

```

@{
    Layout = null;
}

<div id="searchblock">
    @using (Html.BeginForm("List", "Film", FormMethod.Post, new { enctype =
"multipart/form-data" }))
    {
        @Html.TextBox("qtitle", "", new { @class = "searchtextbox" });
        <button type="submit" class="searchbutton" />
    }
</div>

```

SuperSearch.cshtml

```

@model AllFilms.WebUI.Models.SearchModel

@{
    ViewBag.Title = "Расширенный поиск";
}

<h2>Расширенный поиск</h2>
<div class="block">
    @using (Html.BeginForm("SuperSearch", "Search"))
    {
        <div class="editor-label">
            @Html.LabelFor(x => x.QTitle)
        </div>
        <div class="editor-field">
            @Html.EditorFor(x => x.QTitle)
        </div>
    }

```

```

</div>
<div class="editor-label">
    @Html.LabelFor(x => x.FromYear)
</div>
<div class="editor-field">
    @Html.EditorFor(x => x.FromYear)
</div>
<div class="editor-label">
    @Html.LabelFor(x => x.ToYear)
</div>
<div class="editor-field">
    @Html.EditorFor(x => x.ToYear)
</div>
<div class="editor-label">
    Страны производители
</div>
<div class="editor-field">
    @Html.EditorFor(x => x.CountriesAssociations)
</div>
<div class="editor-label">
    Жанры фильмов
</div>
<div class="editor-field">
    @Html.EditorFor(x => x.GenresAssociations)
</div>
<div class="editor-label">
    Люди принимавшие участие в фильме
</div>
<div class="editor-field">
    @Html.EditorFor(x => x.PeopleAssociations)
</div>
<div class="editor-label">
    @Html.LabelFor(x => x.Greed)
</div>
<div class="editor-field">
    @Html.EditorFor(x => x.Greed)
</div>
<div class="admintd" style="margin: 10px 0 0 0;">
    <input type="submit" value="Найти" />
</div>
}
</div>

```

CountriesAssociationsViewModel.cshtml

```

@model AllFilms.WebUI.Models.CountriesAssociationsViewModel
@using AllFilms.Domain.Entities;
@{
    Int32 Column = 3;
    Country[] Countries = Model.Countries.ToArray();
    String CheckboxName = ViewData.ModelMetadata.GetDisplayName() +
    ".SelectedCountriesID";
}
<table class="Grid">
@for (Int32 i = 0; i < Countries.Length; i++)
{
    Country country = Countries[i];
    if (i % Column == 0)
    {
        @: <tr>
    }
    <td>
        <input type="checkbox" name="@CheckboxName" value="@country.CountryID"
        @(Model.SelectedCountriesID.Contains(country.CountryID) ? "checked=\"checked\"" : "") />
    }
}

```

					ПГУ 1.090501.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		139

```

        @country.CountryName
    </td>
    if (i % Column == Column - 1)
    {
        @: </tr>
    }
}
</table>

```

GenresAssociationsViewModel.cshtml

```

@model AllFilms.WebUI.Models.GenresAssociationsViewModel
@using AllFilms.Domain.Entities;
@{
    Int32 Column = 3;
    Genre[] Genres = Model.Genres.ToArray();
    String CheckboxName = ViewData.ModelMetadata.GetDisplayName() + ".SelectedGenresID";
}

<table class="Grid">
@for (Int32 i = 0; i < Genres.Length; i++)
{
    Genre genre = Genres[i];
    if (i % Column == 0)
    {
        @: <tr>
    }
    <td>
        <input type="checkbox" name="@CheckboxName" value="@genre.GenreID"
@((Model.SelectedGenresID.Contains(genre.GenreID) ? "checked=\"checked\"" : "") />
        @genre.GenreName
    </td>
    if (i % Column == Column - 1)
    {
        @: </tr>
    }
}
</table>

```

PeopleAssociationsViewModel.cshtml

```

@model AllFilms.WebUI.Models.PeopleAssociationsViewModel
@using AllFilms.Domain.Entities;
@{
    Int32 Column = 3;
    Person[] People = Model.People.ToArray();
    String CheckboxName = ViewData.ModelMetadata.GetDisplayName() + ".SelectedPeopleID";
    String[] Alph = {
        "А", "Б", "В", "Г", "Д", "Е", "Ж", "З", "И", "К",
        "Л", "М", "Н", "О", "П", "Р", "С", "Т", "У", "Ф",
        "Х", "Ц", "Ч", "Ш", "Щ", "Э", "Ю", "Я"
    };
}

<div id="tabs-@ViewData.ModelMetadata.GetDisplayName()">
    <ul>
        @for (Int32 i = 0; i < Alph.Length; i++)
        {
            Person[] PeopleStartNameWith = People.Where(p => p.Fullname.IndexOf(Alph[i]) ==
0).ToArray();
            if (PeopleStartNameWith != null && PeopleStartNameWith.Length > 0)

```

```

        {
            String tabs = "#tabs" + ViewData.ModelMetadata.GetDisplayName() + i;
            @:<li><a href="@tabs"><span>@Alph[i]</span></a></li>
        }
    }
</ul>
@for (Int32 i = 0; i < Alph.Length; i++)
{
    Person[] PeopleStartNameWith = People.Where(p => p.Fullname.IndexOf(Alph[i]) ==
0).ToArray();
    if (PeopleStartNameWith != null && PeopleStartNameWith.Length > 0)
    {
        String tabs = "tabs" + ViewData.ModelMetadata.GetDisplayName() + i;
        <text>
        <div id="@tabs">
            <table class="Grid">
                @for(Int32 j = 0; j < PeopleStartNameWith.Length; j++)
                {
                    Person person = PeopleStartNameWith[j];
                    if(j % Column == 0)
                    {
                        @: <tr>
                    }
                    <td>
                        <input type="checkbox" name="@CheckboxName"
value="@person.PersonId" @(Model.SelectedPeopleID.Contains(person.PersonId) ?
"checked=\"checked\" : \"\") />
                        @person.Fullname
                    </td>
                    if (j % Column == Column - 1)
                    {
                        @: </tr>
                    }
                }
            </table>
        </div>
        </text>
    }
}
</div>

```

CountriesLink.cshtml

```

@model IEnumerable<AllFilms.Domain.Entities.Country>
<span class="links">
@foreach (var c in Model)
{
    @Html.ActionLink(c.CountryName, "List", "Film", new { countries = c.CountryID },
null)
}
</span>

```

GenresLink.cshtml

```

@model IEnumerable<AllFilms.Domain.Entities.Genre>
<span class="links">
@foreach (var g in Model)
{
    @Html.ActionLink(g.GenreName, "List", "Film", new { genres = g.GenreID }, null)
}
</span>

```

					ПГУ 1.090501.05.001 ПЗ	Лист
						141
Изм.	Лист	№ докум.	Подпись	Дата		

PeopleLink.cshtml

```
@model IEnumerable<AllFilms.Domain.Entities.Person>
<ul>
@foreach (var person in Model)
{
    <li>@Html.ActionLink(person.Fullname, "Details", "Person", new { id = person.PersonId
}, null)</li>
}
</ul>
```

_Layout.cshtml

```
<!DOCTYPE html>
<html>
<head>
    <title>@ViewBag.Title</title>
    <link href="@Url.Content("~/Content/ResetCss.css")" rel="stylesheet" type="text/css"
/>
    <link href="@Url.Content("~/Content/Site.css")" rel="stylesheet" type="text/css" />
    <link href="@Url.Content("~/Content/SiteCss3.css")" rel="stylesheet" type="text/css"
/>
    <link href="@Url.Content("~/Content/themes/base/jquery.ui.all.css")" rel="stylesheet"
type="text/css" />
    <script src="@Url.Content("~/Scripts/jquery-1.5.1.js")"
type="text/javascript"></script>
    <script src="@Url.Content("~/Scripts/jquery-ui-1.8.11.js")"
type="text/javascript"></script>
    <script src="@Url.Content("~/Scripts/jquery.validate.min.js")"
type="text/javascript"></script>
    <script src="@Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js")"
type="text/javascript"></script>
    <script type="text/javascript">
        $(document).ready(function () {
            $('[id^="tabs-"]').tabs();
        });
    </script>
</head>

<body>
    <div id="page">
        <div id="header">
            <a href="@Url.Content("~/")">
                
            </a>
            <div id="mainmenu">
                @{Html.RenderAction("MainMenu", "Navigation");}
            </div>
        </div>
        <div id="columnLeft">
            <div id="genremenu">
                @{Html.RenderAction("GenresMenu", "Navigation");}
            </div>
        </div>
        <div id="columnRight">
            @{Html.RenderAction("Search", "Search");}
            <div class="clear">
                @if (TempData["message"] != null)
                {
                    <div class="Message">@TempData["message"]</div>
                }
            </div>
        </div>
    </div>
```

					ПГУ 1.090501.05.001 ПЗ	Лист
						142
Изм.	Лист	№ докум.	Подпись	Дата		

```

        }
        @RenderBody()
    </div>
</div>
<div id="footer">
</div>
</div>
</body>
</html>

```

Global.asax.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;
using AllFilms.WebUI.Infrastructure;

namespace AllFilms.WebUI
{
    // Примечание: Инструкции по включению классического режима IIS6 или IIS7
    // см. по ссылке http://go.microsoft.com/?LinkId=9394801

    public class MvcApplication : System.Web.HttpApplication
    {
        public static void RegisterGlobalFilters(GlobalFilterCollection filters)
        {
            filters.Add(new HandleErrorAttribute());
        }

        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                "Default", // Имя маршрута
                "{controller}/{action}/{id}", // URL-адрес с параметрами
                new { controller = "Person", action = "List", id = UrlParameter.Optional } // Параметры по умолчанию
            );
        }

        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();

            RegisterGlobalFilters(GlobalFilters.Filters);
            RegisterRoutes(RouteTable.Routes);

            ControllerBuilder.Current.SetControllerFactory(new
            NinjectControllerFactory());
        }
    }
}

```

ВЕДОМОСТЬ ДИПЛОМНОГО ПРОЕКТА

Приложение Б (обязательное)

№ строки	Формат	Обозначение	Наименование	Кол-во листов	Примечание
1					
2			<u>Документация текстовая</u>		
3					
4	A4		Пояснительная записка	143	
5					
6			<u>Документация графическая</u>		
7					
8	A1		Классы приложения для доступа к базе данных. Диаграмма классов	1	
9					
10					
11	A1		Взаимодействие пользователя с веб-интерфейсом. Диаграмма вариантов использования	1	
12					
13					
14	A1		Бизнес-процесс добавления фильма пользователем.	1	
15					
16			Диаграмма деятельности.		
17					
18	A1		Схема базы данных веб-приложения интеграции кинематографического контента	1	
19					
20					
21					
22	A1		Диаграмма размещения	1	
23					
24	A1		Архитектура сервиса. Диаграмма компонентов.	1	
25					
26					
27	A1		Диаграмма классов предметной области	1	
28					
29					
30					

ГРАФИЧЕСКАЯ ЧАСТЬ

Приложение В (обязательное)

					ПГУ 1.090501.05.001 ПЗ	Лист
						145
Изм.	Лист	№ докум.	Подпись	Дата		

ПРЕЗЕНТАЦИЯ

Приложение Г

(обязательное)



Рисунок 7.1 Титульный лист

Анализ предметной области

ЦЕЛЬ:

Цель работы – разработка программного обеспечения, позволяющего производить импорт, анализ и визуализацию данных с сайтов о кинематографии на едином интернет портале.

ЗАДАЧИ:

- 1) Разработка Базы Данных киносервиса
- 2) Разработка логической и физической моделей Базы Данных
- 3) Разработка Программного Обеспечения
- 4) Разработка и анализ пользовательского веб-интерфейса
- 5) Разработка программного обеспечения клиентской и серверной частей

Рисунок 7.2 Анализ предметной области

Технические характеристики

В соответствии с требованиями к осуществляемым программой функциям можно отнести:

- поиск фильма;
- авторизация в системе;
- добавление фильма в базу;
- изменение информации о фильме;
- удаление фильма из базы;
- добавление актера или режиссера в базу;
- изменение информации об актере или режиссере;
- удаление актера или режиссера из базы;
- добавление отзыва о фильме.

Рисунок 7.3 Технические характеристики

Диаграмма классов предметной области

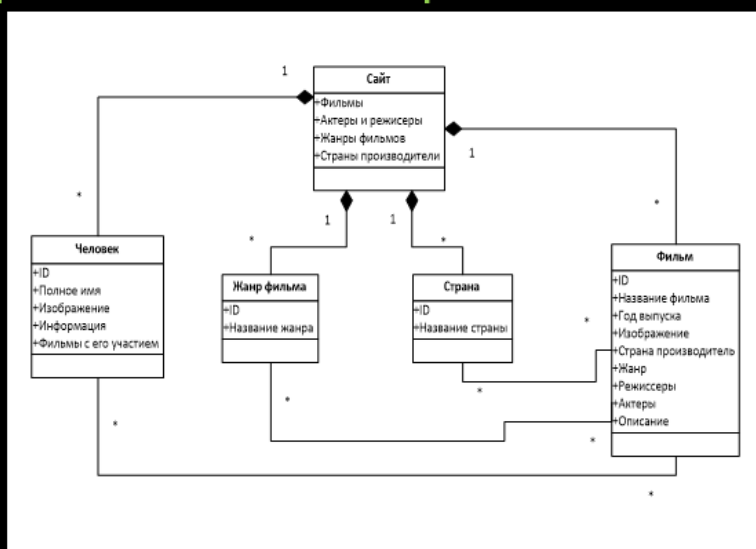


Рисунок 7.4 Диаграмма классов предметной области

Диаграмма базы данных

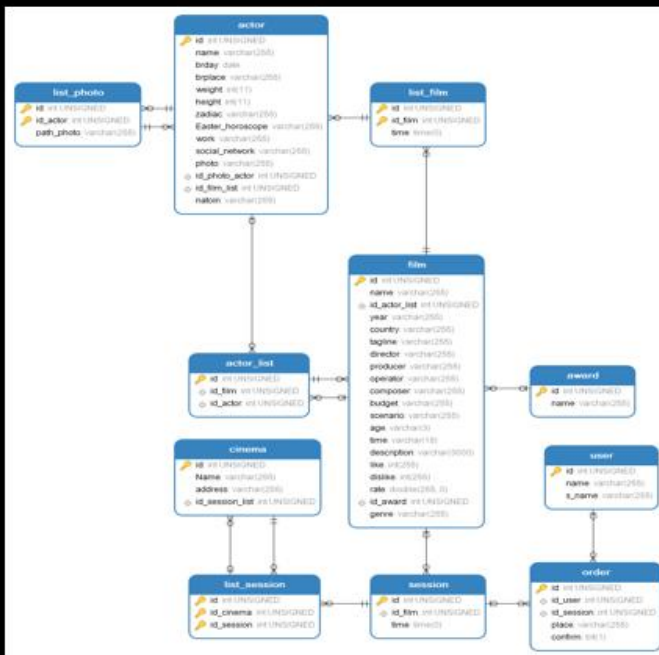


Рисунок 7.7 Диаграмма базы данных

Диаграмма классов

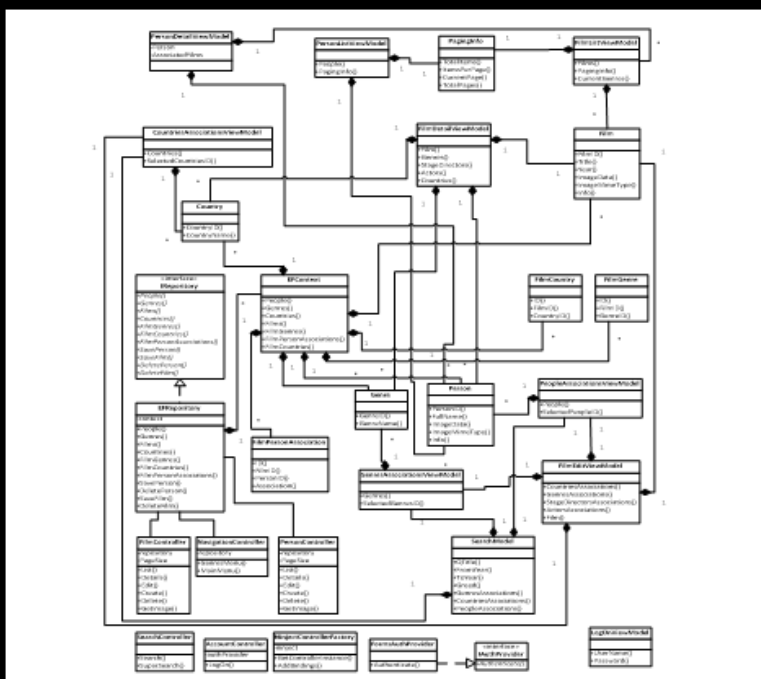


Рисунок 7.8 Диаграмма классов

Диаграмма компонентов

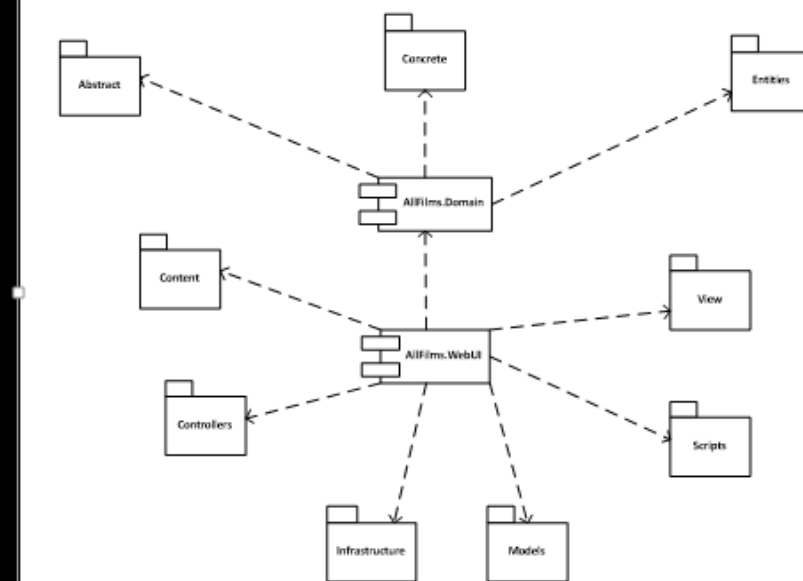


Рисунок 7.9 Диаграмма компонентов

Диаграмма развёртывания

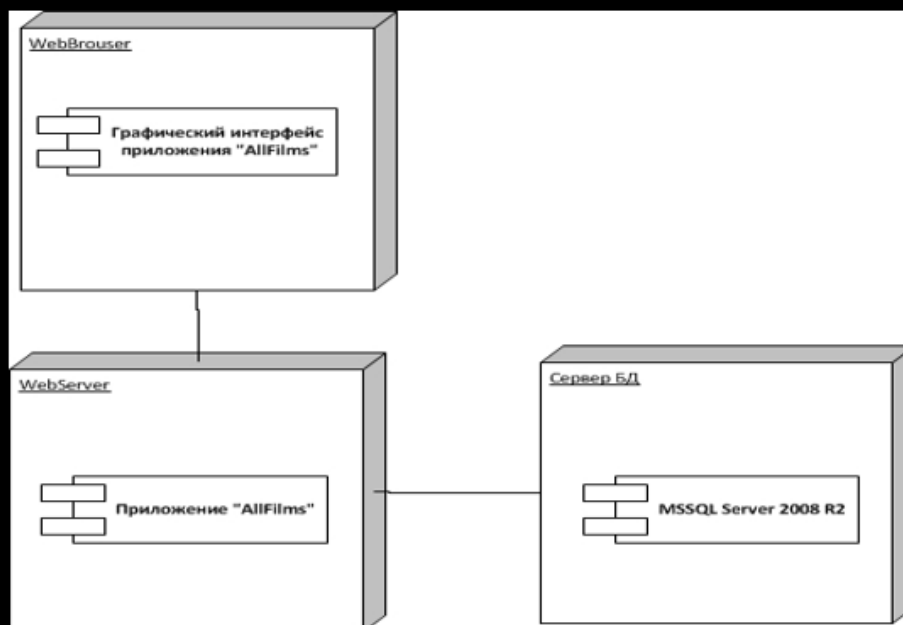


Рисунок 7.10 Диаграмма развёртывания

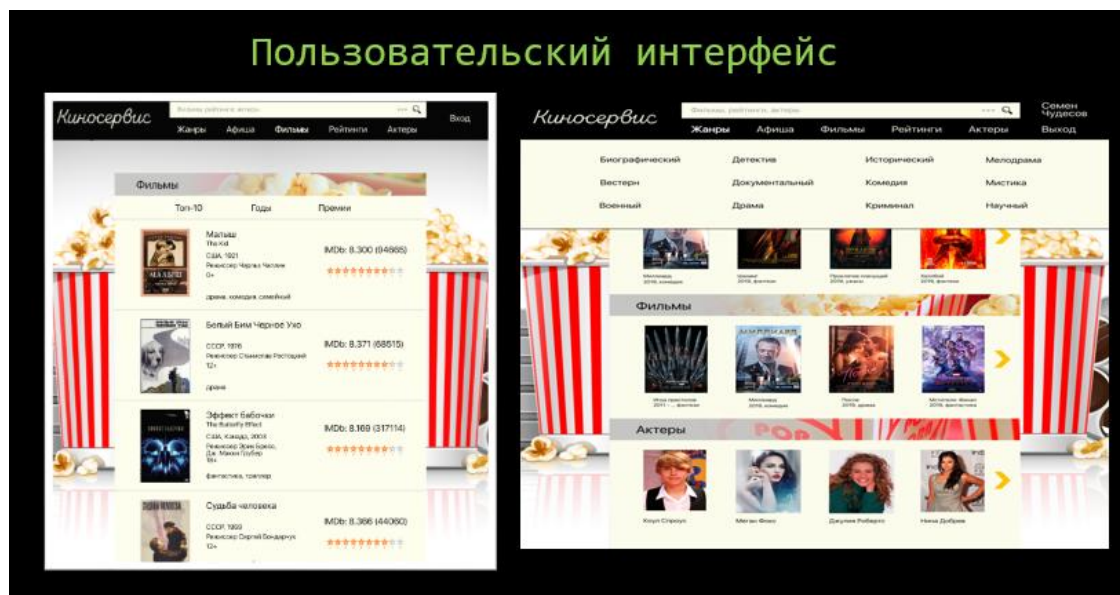


Рисунок 7.11 Пользовательский интерфейс

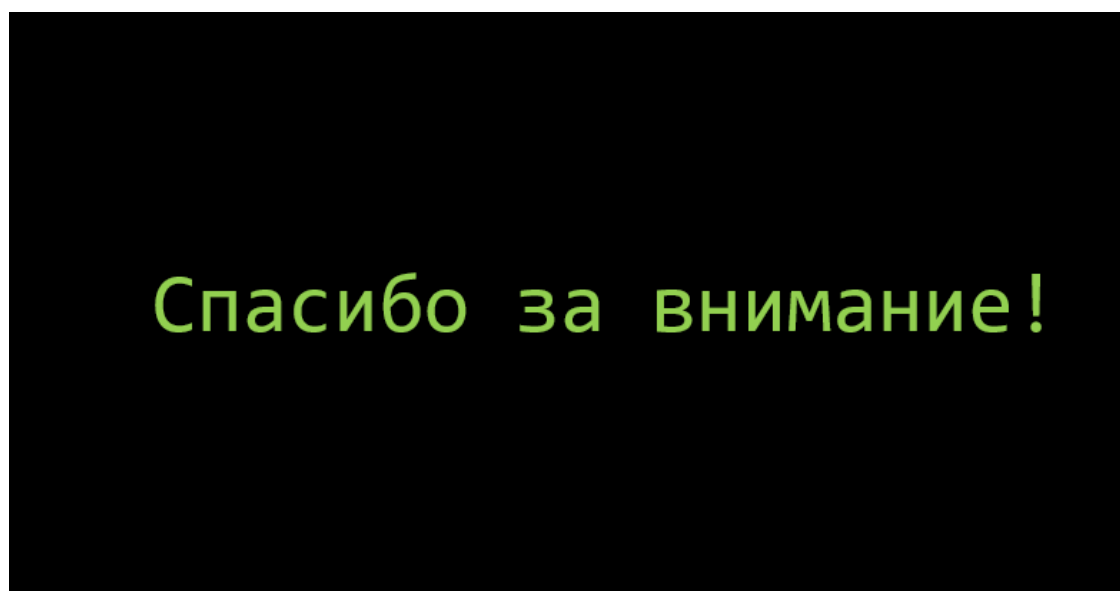


Рисунок 7.12 Окончание презентации

Изм.	Лист	№ докум.	Подпись	Дата