Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Южно-Уральский государственный университет (национальный исследовательский университет)»
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования Направление подготовки: 09.03.04 Программная инженерия

РАБОТА ПРОВЕРЕНА
Рецензент,

_____
«____»_____2019г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н., доцент
_____А.А.Замышляева
«____»_____2019 г.

Разработка сайта кинотеатра

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–09.03.04.2019.109.ПЗ ВКР

Руководитель работы, доцент
_____/А.К. Демидов
«____»_____2019 г.

Автор работы
Студент группы ЕТ-414
_____/С.Р. Хусаинов
«____»_____2019 г.

Нормоконтролер, доцент
_____/Т.Ю. Оленчикова
«____»_____2019 г.

Челябинск
2019

АННОТАЦИЯ

Хусаинов С.Р. Разработка сайта кинотеатра. – Челябинск: ЮУрГУ, ЕТ-414, 47 с., 27 ил., 6 табл., библиогр. список – 22 наим., 2 прил.

Данная работа посвящена разработке сайта, позволяющего пользователям покупать билеты на сеансы в кинозале.

В работе выполнен обзор существующих сайтов подобного типа, наиболее востребованных средств разработки сайтов и систем управления базами данных.

Спроектирована и разработана архитектура системы, включающая в себя диаграмму вариантов использования, диаграмму классов и диаграмму активности. Выполнен анализ предметной области и спроектирована база данных для хранения информации о фильмах и купленных билетах. Разработана и отлажена серверная часть приложения.

Сайт реализован на языке программирования C# с использованием СУБД Microsoft SQL Server.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

На сегодняшний день существует множество сайтов, предоставляющих услуги покупки билетов на сеансы в кинотеатре. В условиях глобализации особенно важно, чтобы каждый мог осуществить покупку. Это может стать затруднением для туристов или же студентов по обмену, а также для тех, кто приехал сюда по рабочей визе.

В этой работе анализируется проблема отсутствия поддержки нескольких языков у сайтов кинотеатров.

Целью данной работы является разработка сайта кинотеатра, для покупки и резервирования билетов, с поддержкой многоязычности.

Для достижения поставленной цели нужно решить список задач:

– исследовать современные технологии и проанализировать инструментарий, для проектирования сайтов;

– составление требований к сайту;

– проектирование и реализация базы данных;

– реализовать логику серверной части;

– проектирование архитектуры сайта;

– осуществление отладки и тестирования сайта.

1 ТРЕБОВАНИЯ. ОБЗОР ТЕХНОЛОГИЙ СОЗДАНИЯ САЙТА

1.1 Существующие сайты кинотеатров

Суть проблемы, рассматриваемой в работе, заключается в том, что в эпоху информационных технологий, мало кто будет идти в кинотеатр только для того, чтобы узнать, какие сеансы там проводятся, разумнее было бы узнать на сайте, но как быть, если человек плохо знает местный язык?

Было предложено реализовать систему покупки (бронирования) билетов онлайн с переводом на различные языки.

Принцип бронирования зародился еще в начале второй половы двадцатого века, в Соединенных Штатах Америки. В процессе изучения этой темы, были найдены области применения бронирования:

– гостиничный бизнес;

– рестораны;

– авиаперелеты;

– железнодорожное передвижение;

– туристический бизнес;

– кинотеатры.

В качестве примера, я рассмотрел имеющиеся в открытом доступе сайты кинотеатров:

– http://www.afisha.ru/ – «Афиша», на которой можно как бронировать, так и покупать билеты. Осуществление заказа билета происходит так: сеансы выбираются в расписании, они помечены синим, переходите по ссылке, выбираете интересующие вас места, осуществляете безналичную оплату через электронный кошелек. Сайт дает возможность выбора и просмотра сеансов вплоть до десяти дней вперед, осуществить покупку можно до шести билетов на одну персону. Этот сайт покрывает более тридцати российских городов, в числе которых Челябинск.

– https://kinomax.ru/ – сайт кинотеатра для покупки и бронирования билетов онлайн. Приобретение и бронь билета на сеанс фильма реализуется описанными шагами. Выбор фильма в горизонтальном слайдере. Подбор интересующего сеанса в нужном филиале кинотеатра. Оплата покупки, оплатить с помощью мобильного телефона, банковской карты или электронного кошелька.

Был выявлен круг задач, проанализировав поставленные цели и ознакомившись с современными решениями:

– полный перевод интерфейса и информации в базе данных;

– наличие выбора клиентом интересующего его сеанса на определенный фильм;

– бронирование и покупка билета на сеанс, выбор ряда и места, множественная покупка и резервирование;

– добавление, редактирование и удаление администратором таких сущностей, как сеансы, фильмы, тарифы и залы;

– базовая стратегия для отчетов и пара реализаций;

– поиск по сайту интересующего фильма.

1.2 Требования к программе

Для решения данной проблемы были выдвинуты следующие цели:

– покупка и бронирование с мобильных устройств и персональных компьютеров;

– осуществление поиска фильмов по сайту;

– реализация администраторской части, в том числе и для тестирования.

1.2.1 Функционал

Система обязана внедрять модули для реализации поставленных задач:

– изменение существующих записей в базе данных;

– вывод информации по запросам пользователя;

– удаление записей из базы данных;

– осуществление выборки информации и вывод оной в согласовании с запросом с клиентской стороны;

– ввод новой информации в базу данных.

1.2.2 Безопасность

Основные требования к надежности, следующие:

– необходимость корректного вывода информации с серверной стороны;

– информация, хранящаяся на сервере, должна быть сохранна и конфиденциальна;

– обеспечение контроля блокировки некорректных действий пользователя при взаимодействии с сайтом, защита паролем, анализ входной информации.

Серверное железо на базе последней Windows, с установленным пакетом Visual Studio 2019 и IIS. Для успешного функционирования системы необходимо, чтобы параметры устройства удовлетворяли следующим:

– процессор с поддержкой последней Windows;

– RAM, в размере от пятисот Мб;

– накопитель с не менее чем 100 Мб не занятой памяти.

1.2.3 Совместимость

Компьютер или смартфон, работающий с современными браузерами.

Поддержка последнего протокола https в браузере для настольных пк, либо Android или IOS для мобильных устройств.

1.2.4 Документация

Разрабатываемый сайт обязан иметь справочный материал о работе системы и подсказки для пользователя, и все это помимо документации.

В документацию включить описание модулей, моделей, классов, хранимых процедур, триггера, контроллеров и сервисов.

1.3 Инструментарий разработки

Создание Web-приложений на стороне сервера осуществляется с помощью различных технологий и тех языков программирования, которые способны производить вывод в стандартную консоль.

В таблице 1.1 приведена информация о лицензировании каждого языка, а также необходимые характеристики веб-сервера для его работы.

Таблица 1.1 – Инструментарий для веб-приложений

Название	Сервер	Лицензия
C/C++	всякий	общедоступная
C#	частный	проприетарная
ASP.NET	частный	триал
Python	практически любой	общедоступная
PHP	практически любой	общедоступная
Java	выбор огромен	общедоступная
Nodejs	личный	лицензия от массачусетского университета
Perl	всякий	общедоступная
Ruby	всякий	общедоступная

C#, Python, PHP и Ruby являются ведущими языками создания сайтов. MSSQL, MySQL и Oracle Database неоспоримые лидеры среди средств управления базами данных.

1.3.1 Transact-SQL

Transact-SQL (T-SQL) – созданное компанией Microsoft, для Microsoft SQL Server, процедурное дополнение языка SQL.

SQL был дополнен такими добавочными особенностями, как:

– глобальные переменные, а также локальные;

– операторы управления;

– поддержка аутентификации через подлинность Microsoft Windows;

– дополнительные функции для обработки дат, математики, строк и т. п..

Все приложения, взаимодействующие с экземпляром MS SQL Server, независимо от их пользовательского интерфейса и реализации, отправляют серверу инструкции Transact-SQL, поэтому можно сказать, что язык Transact-SQL является ключом к использованию MS SQL Server.

1.3.2 JSON

В дополнение к языкам для разработки веб-приложений стоит рассмотреть инструмент, позволяющий клиенту читать данные из серверного приложения – JSON.

JSON (от англ. JavaScript Object Notation) формат обмена текстами, на основе JavaScript. JSON удобно читать людям. Этот формат был разработан Дугласом Крокфортом-Домом.

Важно отметить, что JSON, полученный из JavaScript (или, чтобы быть как можно более точным, из одного из языков стандарта ECMA-2621999), однако, классифицируется как независимый инструмент, который не имеет прямой связи с JavaScript. Сегодня, под многие языки программирования разработали множество инструментов, для создания и обработки данных любого типа в формате JSON.

Текст JSON (в зашифрованном виде) может быть представлен одной или двумя структурами:

− как набор пар ключей и их соответствующих значений. Много языков в Интернете: записи, записи, хеш-таблицы, списки с ключевыми или ассоциативными массами. Ключи могут появляться только в виде строк (они могут отличаться), а знак - любой формы;

− рассматриваются, как упорядоченный набор значений. Разные языковые реалии по-разному: вектор, массив, список, последовательность.

Такие структуры данных универсальны, потому что все современные языки программирования в той или иной форме их поддерживают. Необходимо было обмениваться данными между разными языками. [7].

Термы в JSON:

– объект – это неупорядоченный набор пар ключей и значений, заключенных в фигурные скобки "{}". Строка обязательно используется в качестве описания ключа, между ним и значением стоит символ «:». Пары ключей и значений отделяются друг от друга запятыми;

– массив – это упорядоченный набор значений. Он заключен в квадратные скобки "[]". Значения внутри массива разделяются запятыми;

– число;

– литералы логического типа: «истина», «ложь» и «ноль»;

– строка представляет собой упорядоченный набор символов Unicode, количество которых может быть любым, начиная с нуля. Этот набор заключен в двойные кавычки, и включенные в него символы указываются одним из двух способов: либо как escape-последовательности, начинающиеся с обратной косой черты (косая черта) "\" (параметры \ ", \\, \ / поддерживаются \ t, \ n, \ r, \ f и \ b) или в форме шестнадцатеричного кода (как \ uFFFF, так и в кодировке UTF-8).

Допускается использование пробелов между любыми двумя элементами синтаксиса.

Не стоит забывать, что физическая строка в JSON есть равнозначимый аналог эквивалентного типа данных в C-подобных языках программирования.

В JSON число оно представлено только в десятичном формате.

1.3.3 Microsoft SQL Server

Microsoft SQL Server – это система управления реляционными базами данных (RDBMS), разработанная Microsoft. Основным языком запросов является Transact-SQL, созданный совместно Microsoft и Sybase. Transact-SQL – это реализация языка структурированных запросов (SQL) в формате ANSI / ISO с расширениями. Используется для работы с базами данных, размер которых варьируется от личных до крупных корпоративных баз данных; конкурирует с другими СУБД в этом сегменте рынка.

1.3.4 MySQL

MySQL – это бесплатная система управления базами данных (СУБД). MySQL принадлежит приложениям SunMicrosystems и организациям поддержки. Лицензия GNU (General Public License) или использование собственной коммерческой лицензии. MySQL [1].

MySQL наиболее эффективно используется в малых и средних приложениях. Он поставляется в стандартных пакетах веб-разработки для доступных систем, таких как Windows (или WAMP) и Linux (LAMP). В большинстве случаев MySQL выполняет [1].

MySQL способен поддерживать поддержку нескольких похожих баз данных. Эти механизмы определяют, что происходит в определенный момент времени. Это соглашение между силой и скоростью.

1.3.5 Oracle Database

Oracle Database – это система управления объектно-реляционными базами данных, разработанная Oracle.

Oracle – это мощный программный пакет, который позволяет разрабатывать приложения различного уровня сложности. Ядром комплекса является база данных, способная хранить практически неограниченное количество информации благодаря инструментам масштабирования. Такая база данных позволяет эффективно работать с информацией практически о любом количестве пользователей одновременно, а производительность не снижается даже при резком увеличении их количества. Возможности здесь ограничены не самой базой данных, а доступными аппаратными ресурсами.

Еще одним важным преимуществом Oracle является его способность работать практически на любой платформе, поскольку сегодня эта база данных поставляется практически для всех существующих операционных систем. Соответственно, можно работать с продуктами Oracle под Linux, Windows, Sun Solaris или другой ОС, не опасаясь возникновения каких-либо проблем, связанных с несовместимостью. Это преимущество СУБД Oracle по-

зволяет компаниям, которые решили использовать ее продукты, обойтись без изменений в уже существующей сетевой среде.

1.3.6 C#

C# (Си Шарп) – парадигма оного заключается в объектно-ориентированном подходе, иначе никак.

Преимущества C#:

– средний порог практического вхождения для знакомых с C-подобными языками;

– всецело объектно-ориентированный;

– LinQ;

– конечная программа собирается уникально для каждого компьютера;

– события;

– интеграция с другими языками;

– глубокая обработка исключений;

– в ноябре 2014 года была выпущена свободная версия.

Недостатки C#:

– для ведения бизнеса лицензия Visual Studio обойдется в крупную сумму;

– уступает Java в скорости;

– присутствует возможность работать с памятью напрямую, что негативно сказывается на надёжности.

1.3.7 Python

Python является динамически типизированным, интерпретированным языком. Изысканность Python заключается в том, что вместо фигурных скобок используются отступы.

Преимущества Python:

– свободный;

– легкость обучения;

– условия толкают программиста на написание кода, удобного для последующего чтения им и другими разработчиками;

– имеет инструменты для динамической семантики и быстрого прототипирования;

– сформировано большое сообщество;

– чрезвычайно унифицирован, что позволяет легко использовать все виды языковых и библиотечных расширений в процессе разработки проекта;

Недостатки Python:

– множество тестовых проверок продемонстрировали превосходство скорости основной виртуальной машины Java над Python, что заслужило репутацию медленного языка [20];

– синтаксис на любителя.

1.3.8 PHP

PHP – действующий на веб-сервере язык, создан исключительно для сайтов. Эффект его популярности виден до сих пор, хоть и угасает на фоне конкурентов.

Преимущества PHP:

– свободный;

– прост;

– огромное комьюнити;

– разнообразное взаимодействие с базами данных;

– поддержка со стороны сообщества;

– PHP можно использовать в изолированной среде;

– взаимодействие с сессиями;

– простота размещения.

Недостатки PHP:

– только для сайтов;

– скудное взаимодействие с эксепшенами;

– трудности в освоении синтаксиса;

– только значимые типы данных;

– дополнительные действия для обеспечения должного уровня безопасности.

1.3.9 Ruby

Ruby – интерпретируемый язык, похожий своей философией на Перл и Пайтон, объектно-ориентированный и динамический.

Преимущества Ruby:

– интерпретатор свободно распространяется;

– мультиплатформенный;

– встраиваемый в представления;

– мощные средства для работы со строковыми типами и массивами;

– легок в освоении;

– элементарная реализация мультипоточности и параллельности;

– можно совмещать с Си;

– обеспечение безопасности.

Недостатки Ruby:

– не подходит в качестве первого языка;

– малочисленное комьюнити;

– относительно невысокая производительность;

– разработка относительно медленная.

1.4 Обзор архитектур Web-приложений

Разработка веб-приложений является сложным процессом, одной из важнейших задач которого является поиск решений для наиболее эффективного разделения функций программы между его серверной и клиентской частями.

Решение этой проблемы в разных случаях приводит к появлению архитектуры, состоящей из двух, трех или множества составляющих. Это зависит

от того, сколько промежуточных связей должно быть включено между клиентом и сервером [2].

В основу мировой паутины лежит схема «клиент-сервер». Изначально, классическая реализация этой схемы реализуется так:

– клиент формирует и отправляет запрос на сервер сайта, где располагается база данных;

– сервер анализирует запрос и данные, собирает результат и отправляет его обратно клиенту;

– когда результат принят, клиент видит его на используемом устройстве вывода данных, в процессе чего переходит в режим слушания, для последующих операций пользователя.

Этот цикл непрерывно повторяется до тех пор, пока пользователь не завершит работу с сервером.

Если пользователь прекратит посылать запросы серверу, то данная цикличность прекратит свое существование.

Транзакции, исполняемые в HTTP (HyperText Transmition Protocol).

1. Интернет-браузер совершает декодировку первой доли URL (Universal Resource Locator), уже после чего же вводится связь с сервером.

2. Интернет-браузер представляет клиенту оставшуюся долю URL.

3. В соответствии с URL, сервер определяет имя файла и путь.

4. Пересылка указанного файла браузеру, посредством сервера.

5. Соединение прерывается, посредством сервера.

6. Документ, с помощью браузера, выводится на экран клиента.

В сервисе WWW (World Wide Web) для пересылки информации применяется протокол HTTP (см. рисунок 1.5).

Сервер не имеет никакой информации о состоянии браузера, в процессе осуществления этих транзакций, т.е. HTTP можно назвать протоколом «с одним направлением», позволяющим проводить контакт с сервером только через механизм URL, что грозит сложностями при осуществлении клиентской составляющей (см. рисунок 1.4).

При реализации архитектуры прототипа «клиент-сервер», функции распределяются таким списком:

– архитектура, состоящая из двух долей, основывается на распределении всей работы приложения (бизнес-логика и прикладные задачи) между клиентом и сервером;

– клиентское приложение занимается, прежде всего, реализацией удобного для пользователя интерфейса, то есть ввод данных и отображения результатов в интуитивно понятном виде, также, оно управляет триггерами работы приложения;

– серверная система управления базами данных (СУБД) реализует надежность, защищенность и согласованность информации, управляет запросами клиентов и способствует быстрой обработке SQL-запросов.

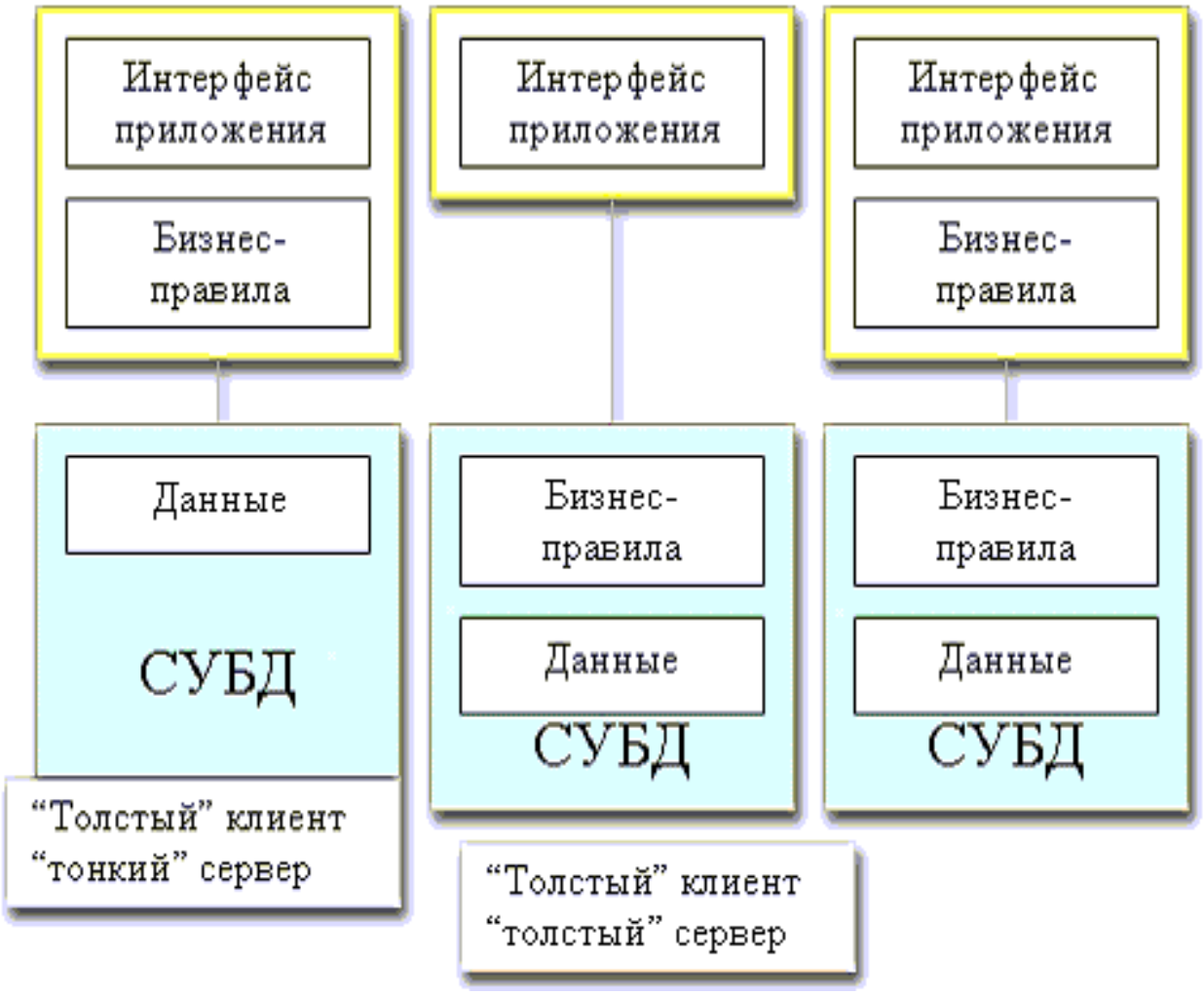


Рисунок 1.1 – Клиент-сервер, разделение функционала

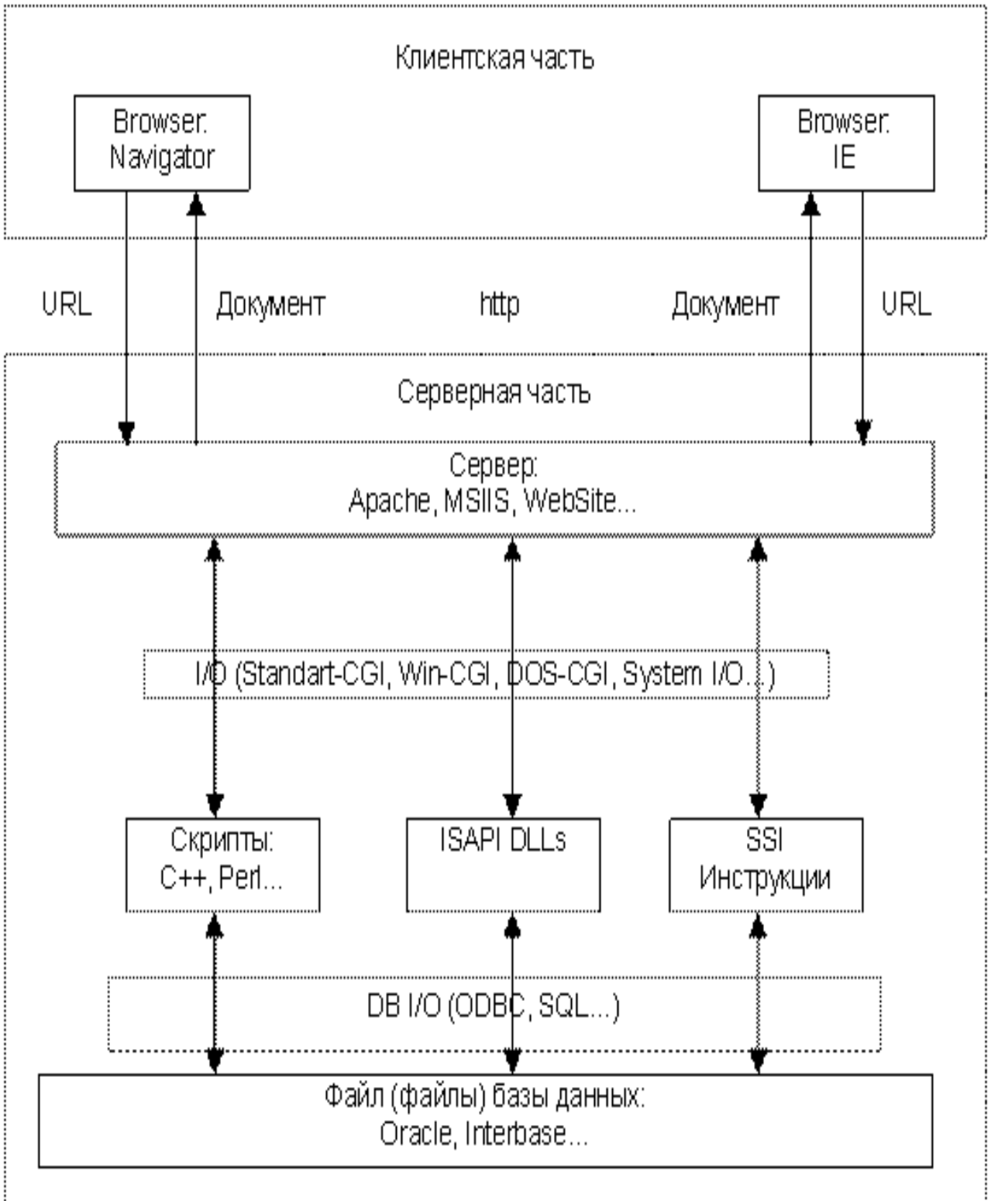


Рисунок 1.2 – Схема клиент-серверное приложение

Схема двухзвенной архитектуры приведена на рисунке 1.6.

При архитектуре, называемой «толстый клиент – тонкий сервер» вся бизнес-логика лежит на стороне клиента, сервер занят исключительно работой с SQL-запросами.

При подобной архитектуре рождаются следующие неудобства:

– степень защищенности данных снижается, это происходит из-за разграничения полномочий;

– разграничение доступа происходит по специальным таблицам, а не по списку действий, из-за чего повышается трудность распределения полномочий;

– информация, передающаяся по сети, является необработанной, в следствие чего появляется избыточная нагруженность;

– неудобства при осуществлении администрирования;

– изменение пакета ПО затруднено, так как он должен меняться единовременно во всей среде.

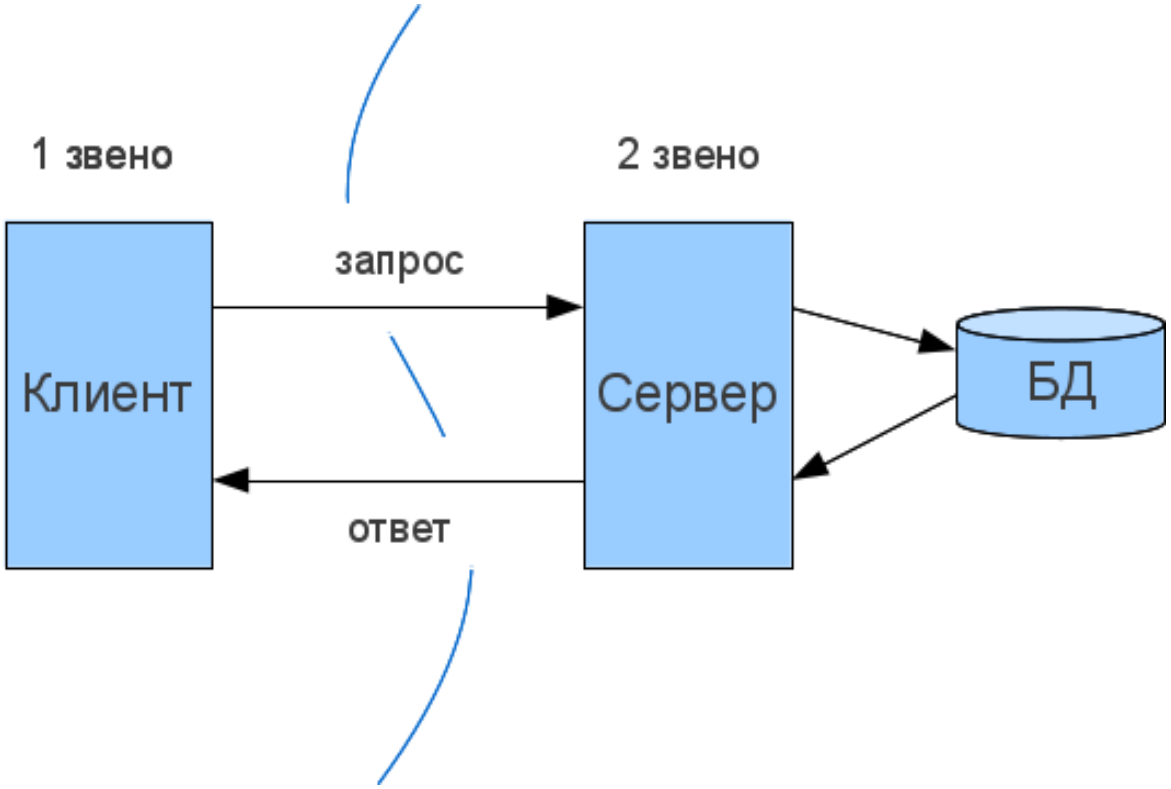


Рисунок 1.3 – Клиент-сервер, двудольная архитектура.

В качестве архитектуры был выбран толстый сервер, при тонком клиенте, так как изначальные требования к проекту оказались масштабными, а большая часть функционала, бизнес-логики, лежит на сервере. Так же не стоит забывать, что поддержка такой архитектуры менее затратна.

Но не все так просто, стоит упомянуть и о том, что:

– перенести подобную программу на иную платформу или же систему – крайне тяжело;

– возникновение ошибки в работе программы, написанной с помощью языков баз данных, может стать причиной выхода из строя всего сервера, хранящего базы данных программы, как правило, недостаточно надежны в работе;

– низкая масштабируемость, так как программа, написанная таким образом, является относительно непроизводительной;

– поскольку языки типа SQL изначально не были созданы для разработки ПО как такового и не имеют надежных средств отладки, в следствие чего происходит усложнение реализации.

Заслуживает внимания тот факт, что эпоха многоядерных вычислений уже наступила, а это означает рассвет многопоточного программирования, что способствует переходу к трехзвенную (см рисунок 1.7) архитектуре.

Клиент-серверная архитектура превращается в трехзвенную (см рисунок 1.8), превращается в таковую из двудольной, благодаря тенденции в технологиях типа «клиент-сервер», связано это с активным ростом использования различных распределенных вычислений. Для осуществления таких вычислений применяется модель сервера приложений. В этой модели сетевое приложение разделяется на составные части, при этом любая из этих долей способна запускаться на отдельном компьютере. Части приложения, которые выделены, взаимодействуют друг с другом, происходит обмен информацией в формате, который оглашен заранее, обычно это JSON.

Третьей долей в такой системе является сервер приложений, который способствует:

– представление данных происходит на стороне клиента;

– прикладной компонент существует на выполняющем функции промежуточного программного обеспечения, иными словами, находится на специально выделенном сервере приложений;

– управление ресурсами осуществляется на сервере базы данных, который и предоставляет интересуемые данные.

Трёхзвенная архитектура обеспечивает гораздо большую масштабируемость, чем двухзвенная, это происходит благодаря горизонтальной масштабируемости приложений, а также мультифлексированию соединений.

Более того, если выделять дополнительные сервера, каждый из которых способен представлять собственные сервисы, то трехзвенная архитектура может быть расширена до многозвенной, помимо этого, изолированность звеньев друг от друга дает ей иметь большую вариативность (см рисунок 1.8).

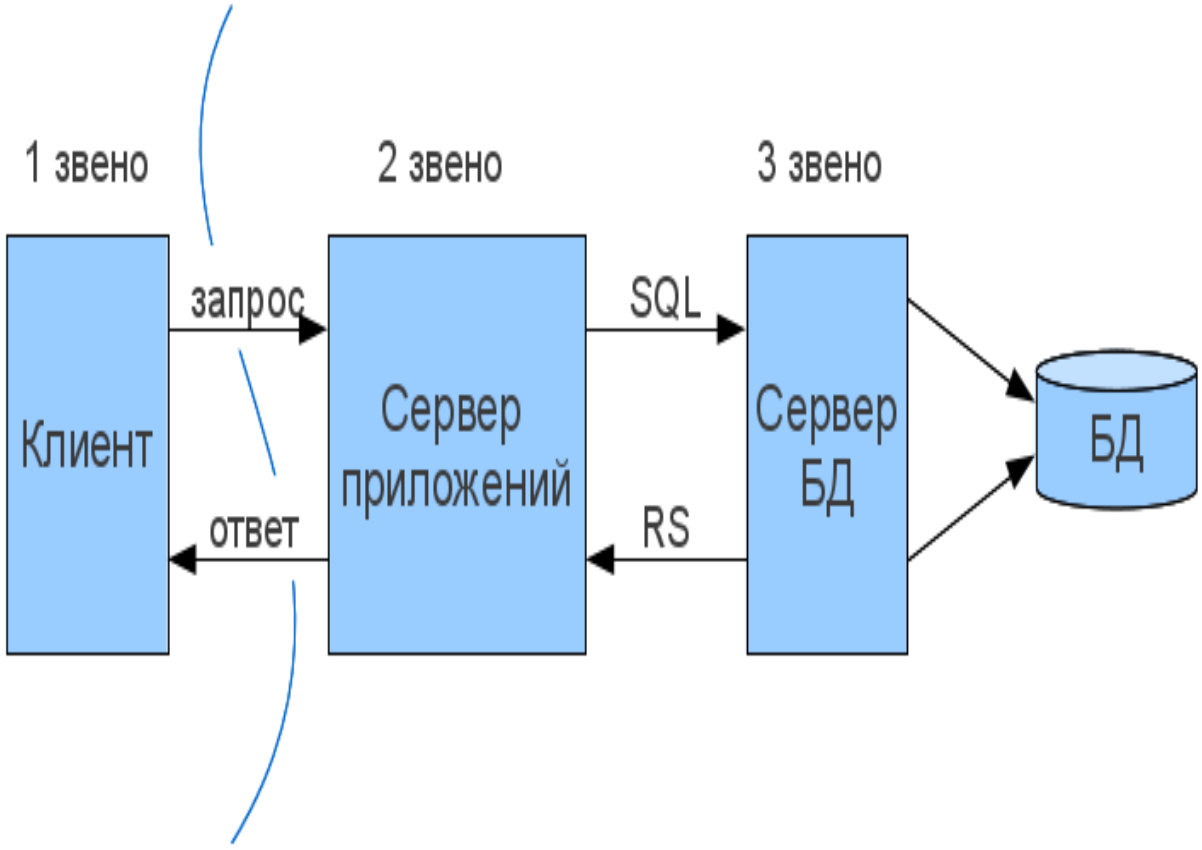


Рисунок 1.4 – Трехзвенная архитектура приложения

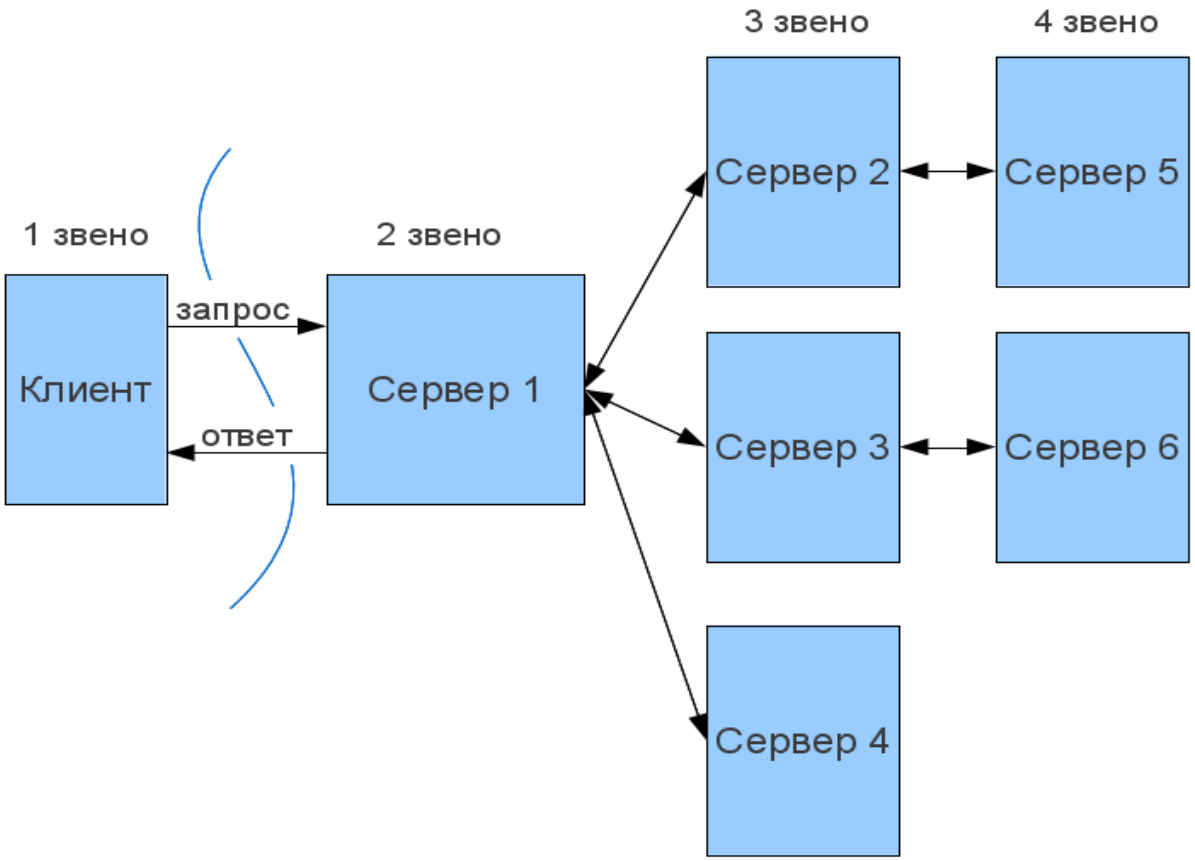


Рисунок 1.5 – Пример многозвенной архитектуры приложения

Для разработки приложения была выбрана двухзвенная архитектура «толстый сервер – тонкий сервер».

1.5 Выводы по разделу

В этом разделе анализируются требования к сайту, в том числе:

– требования к надежности;

– требования к функциональным характеристикам;

– требования к составу и параметрам технического оборудования;

– требования к информационной и программной совместимости;

– требования к документации программного обеспечения.

Был проведен обзор существующих веб-приложений, и эти приложения были сопоставлены с разрабатываемой системой.

Он также рассмотрел распределение функций в веб-приложении, клиент-серверную организацию Интернета, протокол HTTP и основные транзакции в этом протоколе, а также типы клиент-серверных архитектур.

Был проведен сравнительный обзор наиболее широко используемых и популярных языков веб-программирования – PHP, Ruby, Python и C #, в котором изложены преимущества и недостатки этих языков, на основе которых был выбран C #.

Кроме того, был использован обзор самых популярных систем управления базами данных – MySQL, Oracle Database и MSSQL, на основе которых был выбран MSSQL.

JSON, формат обмена текстовыми данными на основе JavaScript, также был пересмотрен: даны его функциональность и синтаксис.

Наконец, на основе приведенной выше информации был сделан выбор инструментов разработки.

MVC 5 (Model-View-Controller) был выбран в качестве архитектуры клиент-сервер, используемой из-за упрощенной реализации и надежной работы СУБД.

В качестве инструмента разработки для веб-приложения был выбран язык программирования C # из-за его практичности, гибкости и распространенности.

MSSQL была выбрана в качестве СУБД, поскольку она надежна и поддерживается Microsoft.

Текстовый формат JSON был выбран для обмена данными между клиентом и сервером, поскольку он прост и поддерживается большинством языков программирования.

2 РАЗРАБОТКА АРХИТЕКТУРЫ СИСТЕМЫ

2.1 Диаграмма вариантов использования

Для общепринятого построения диаграмм архитектуры приложения использовался язык графического описания UML (Unified Model Language). Этот язык очень удобен для стандартизированного представления сущностей, таких как, актор, обобщение, класс [6].

На рисунке 2.1 приведена диаграмма вариантов использования (или диаграмма прецедентов) системы в целом, раскрывающая, какие взаимодействия и типы пользователей предусмотрены в системе.

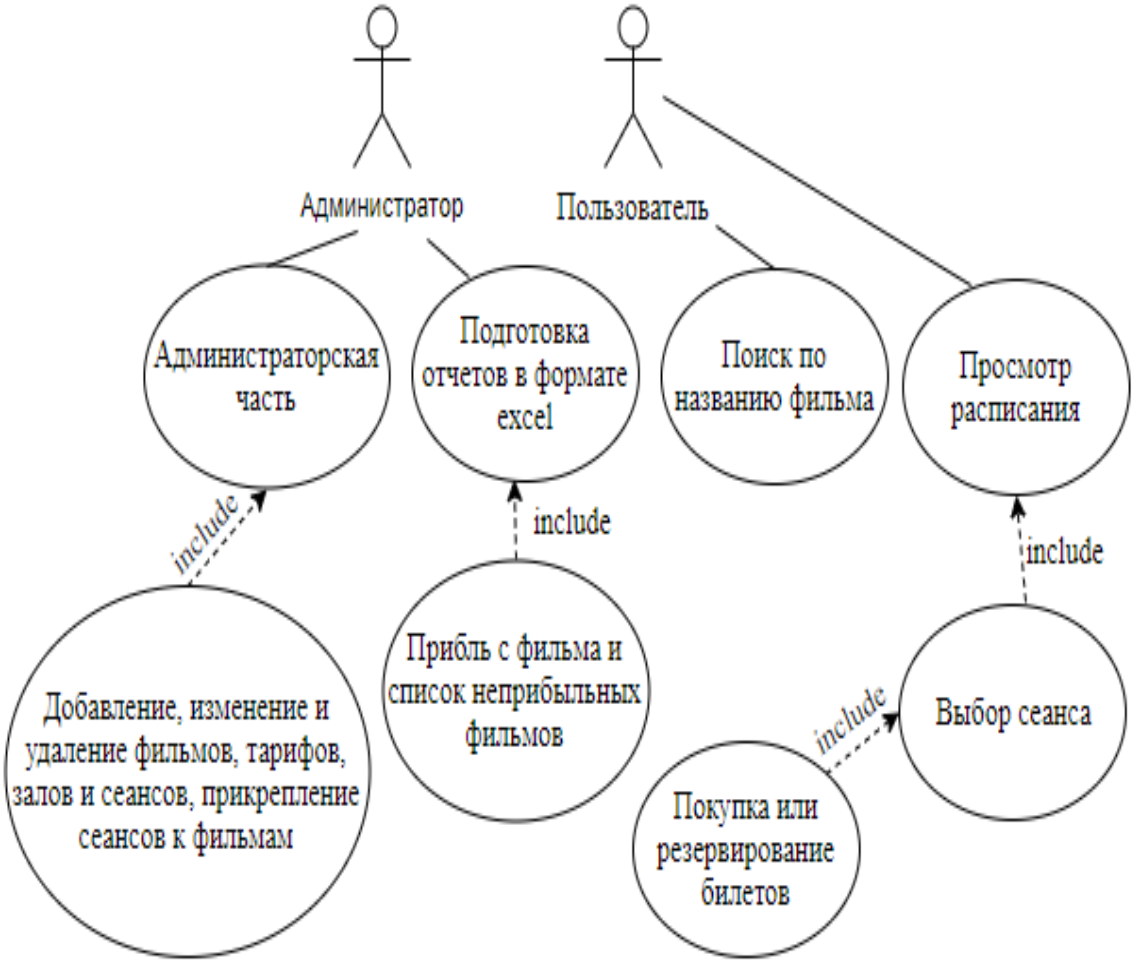


Рисунок 2.1 – Диаграмма вариантов использования системы в целом

2.2 Диаграммы классов

При разработке архитектуры приложения использовался шаблон проектирования Model-View-Controller (MVC). Данный паттерн проектирования призван упростить разработку дизайнерам и программистам, отделив логику от отображения:

– модель (Model) – элемент, содержащий всю бизнес-логику, которая при необходимости получает данные из репозитория (в данном случае базы данных);

– представление (View) – реализует отображение данных, обращаясь к контроллеру за обновлениями;

– контроллер (Controller) – контроллер обеспечивает «связь» между пользователем и системой. Отслеживает и отправляет данные от пользователя в систему и наоборот. Использует модель и представление для реализации необходимых действий.

На рисунке 2.2 показана общая диаграмма классов системы, показывающая взаимодействие классов друг с другом. Ниже приведены диаграммы классов, которые показывают элементы «Controller» (см. рисунок 2.3) и «Model» (см. рисунок 2.4). Паттерн стратегия для отчетов (см. рисунок 2.5).

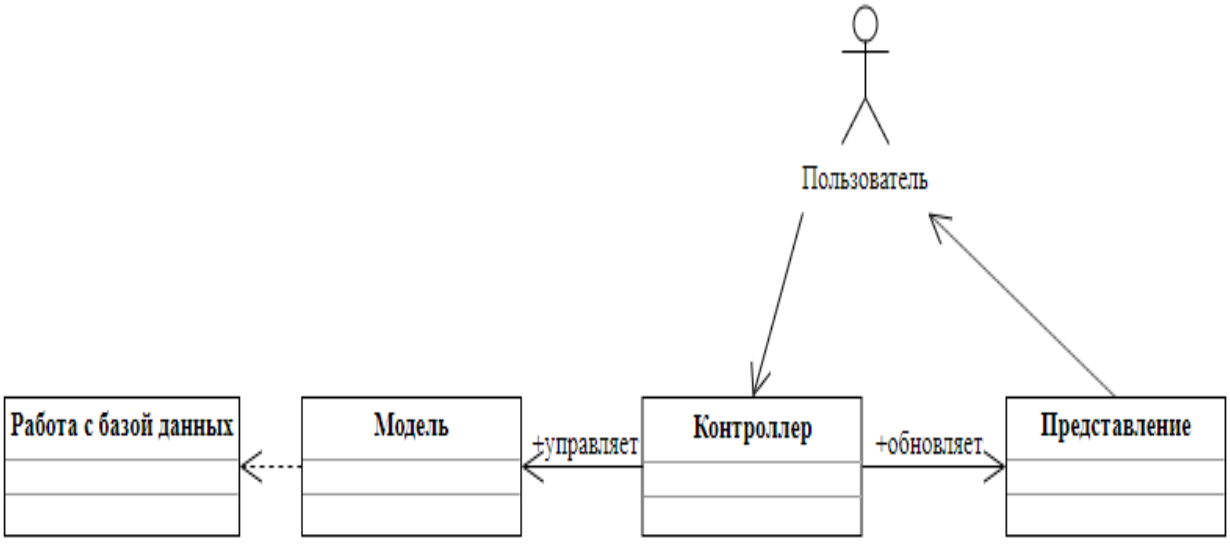


Рисунок 2.2 – Общая диаграмма классов системы

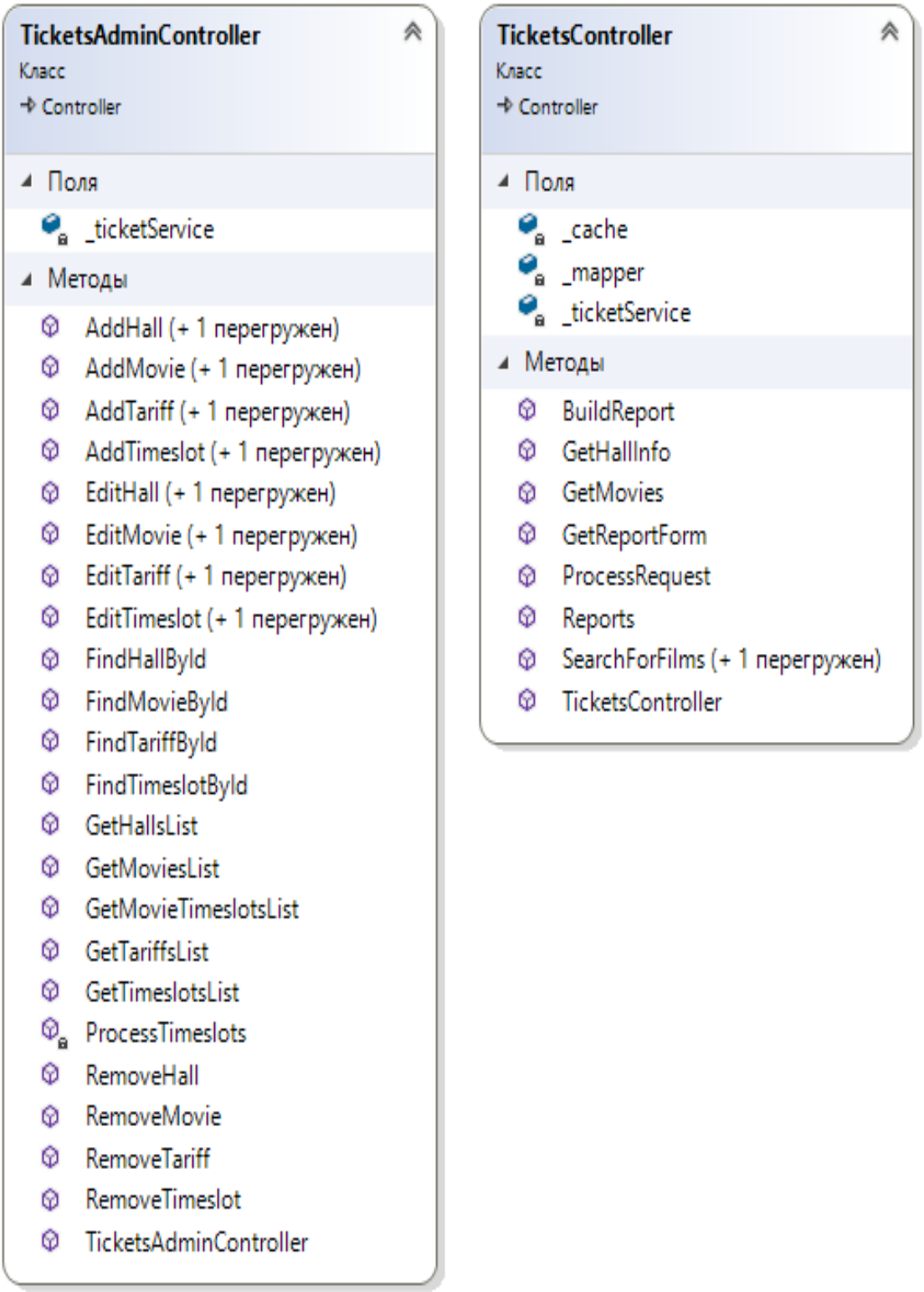


Рисунок 2.3 – Диаграмма классов элемента «Контроллер»

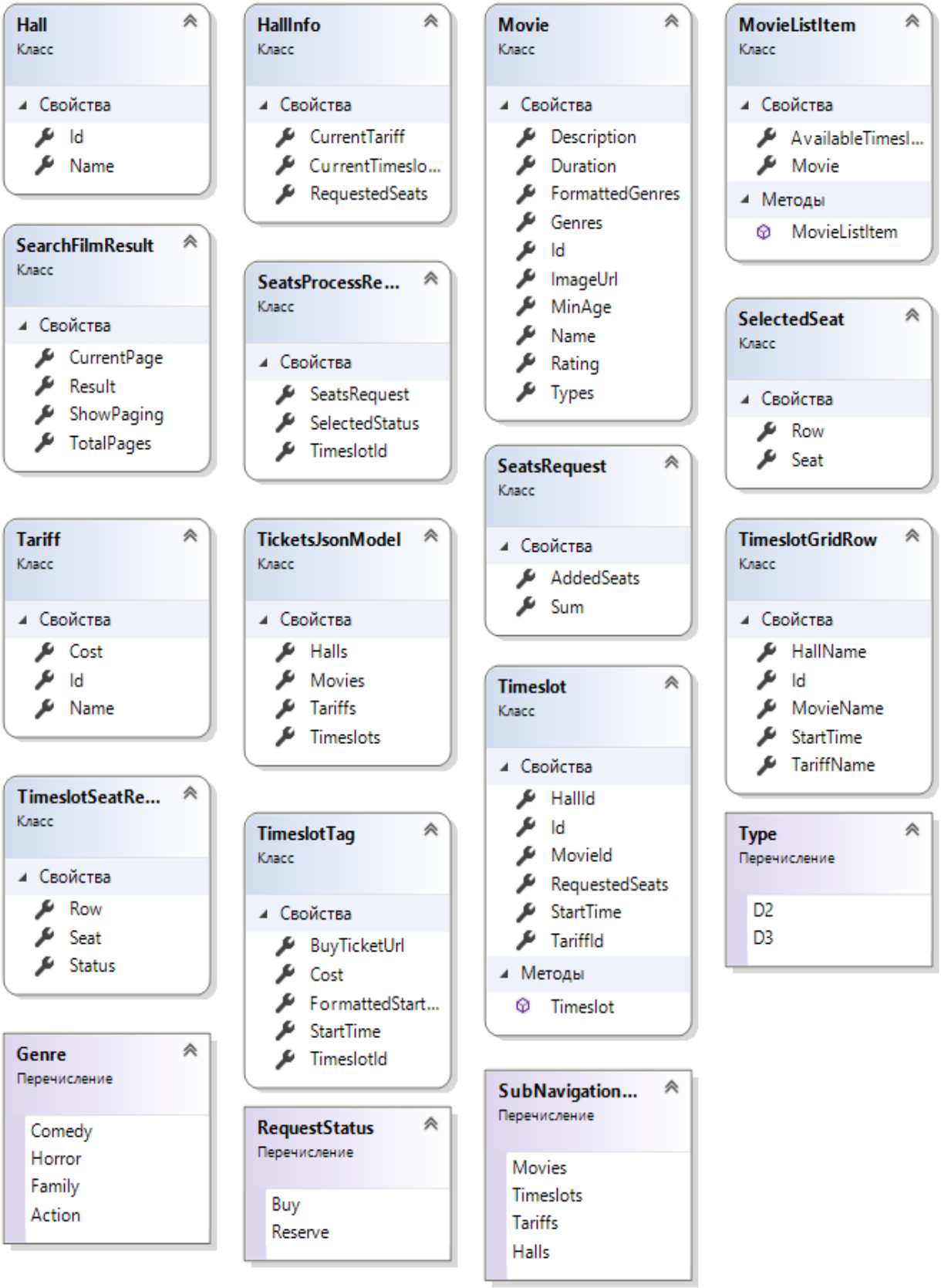


Рисунок 2.4 – Диаграмма классов элемента «Модель» билеты

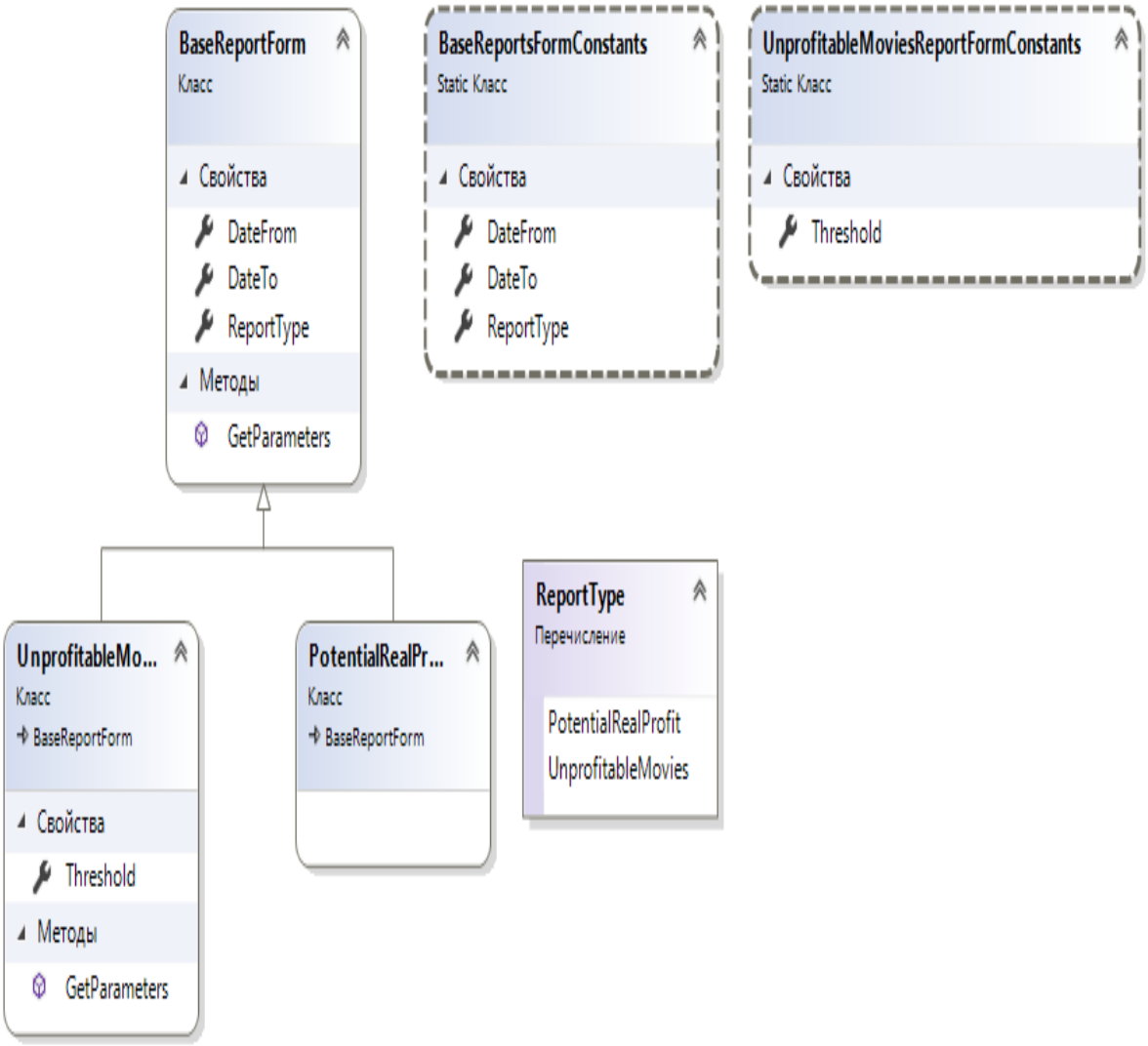


Рисунок 2.5 – Диаграмма классов элемента «Модель» отчеты

2.3 Диаграмма активности

На рисунке 2.6 показана общая диаграмма активности системы, показывающая поведение исполняемой программы в виде последовательного выполнения подчиненных элементов – вложенных типов действий и отдельных действий.

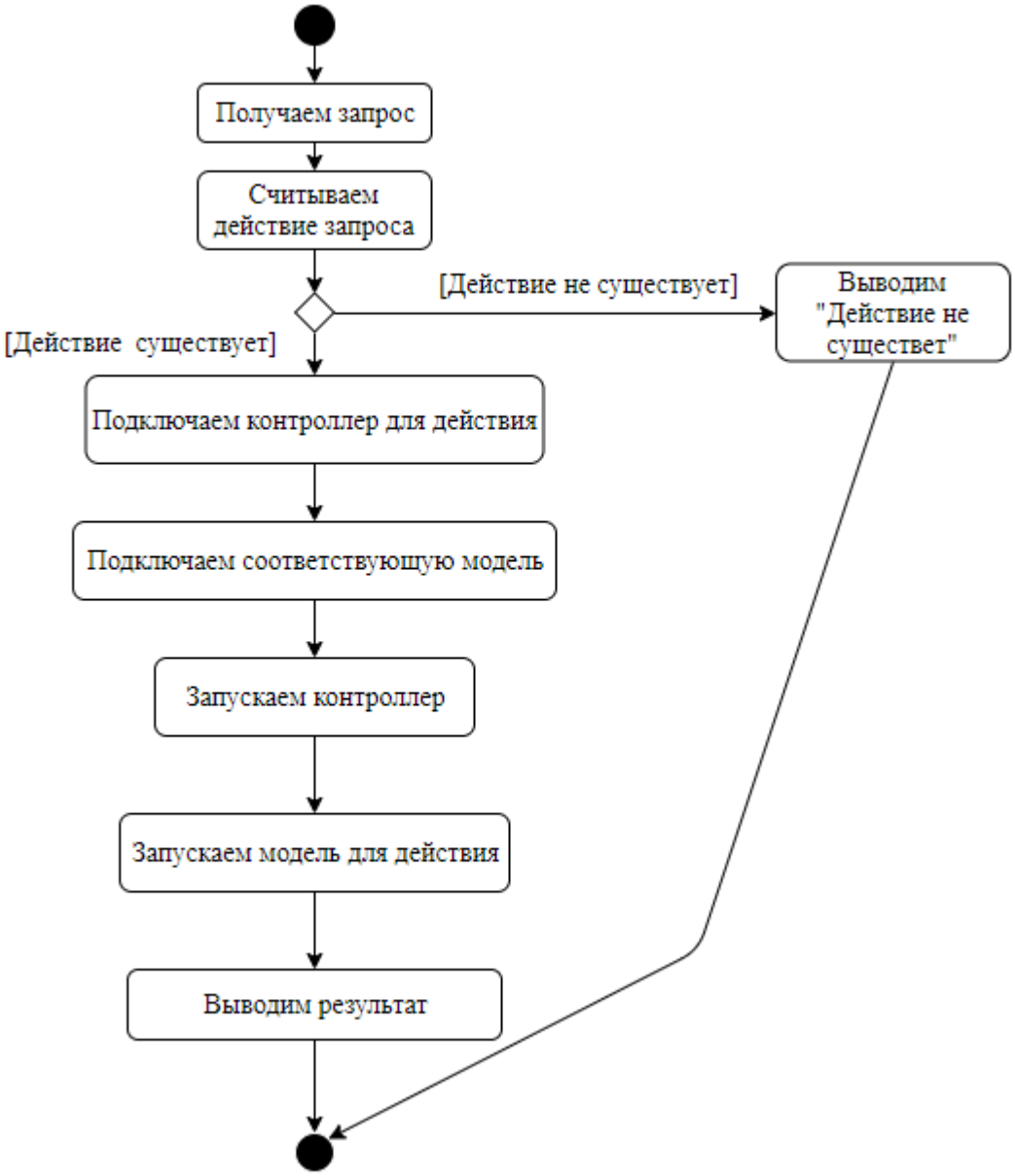


Рисунок 2.6 – Диаграмма активности системы

2.4 Выводы по разделу

В этом разделе была разработана архитектура системы, диаграмма вариантов использования, диаграммы классов и диаграммы деятельности.

Диаграмма вариантов использования включает в себя все возможные варианты использования системы. Диаграмма классов описывает структурные компоненты системы. Наконец, диаграмма действий описывает поведение исполняемой части основной части системы.

Диаграммы были описаны с использованием языка UML. При разработке системы использовался шаблон Model-View-Controller (MVC).

3 РАЗРАБОТКА БАЗЫ ДАННЫХ

3.1 Концептуальное проектирование

Разработка базы данных начинается с концептуального дизайна. Для этого необходимо выбрать все объекты и соединения в системе.

Пользователь может купить билеты на сеансы. Добавить, редактировать и удалять фильмы, временные интервалы, залы и цены. Запрос статистических отчетов Excel.

Бронирование билетов делится на покупку и бронирование.

На основе этих данных было идентифицировано 5 сущностей, 4 взаимосвязи «один ко многим» и построена диаграмма ER (см. рисунок 3.1).

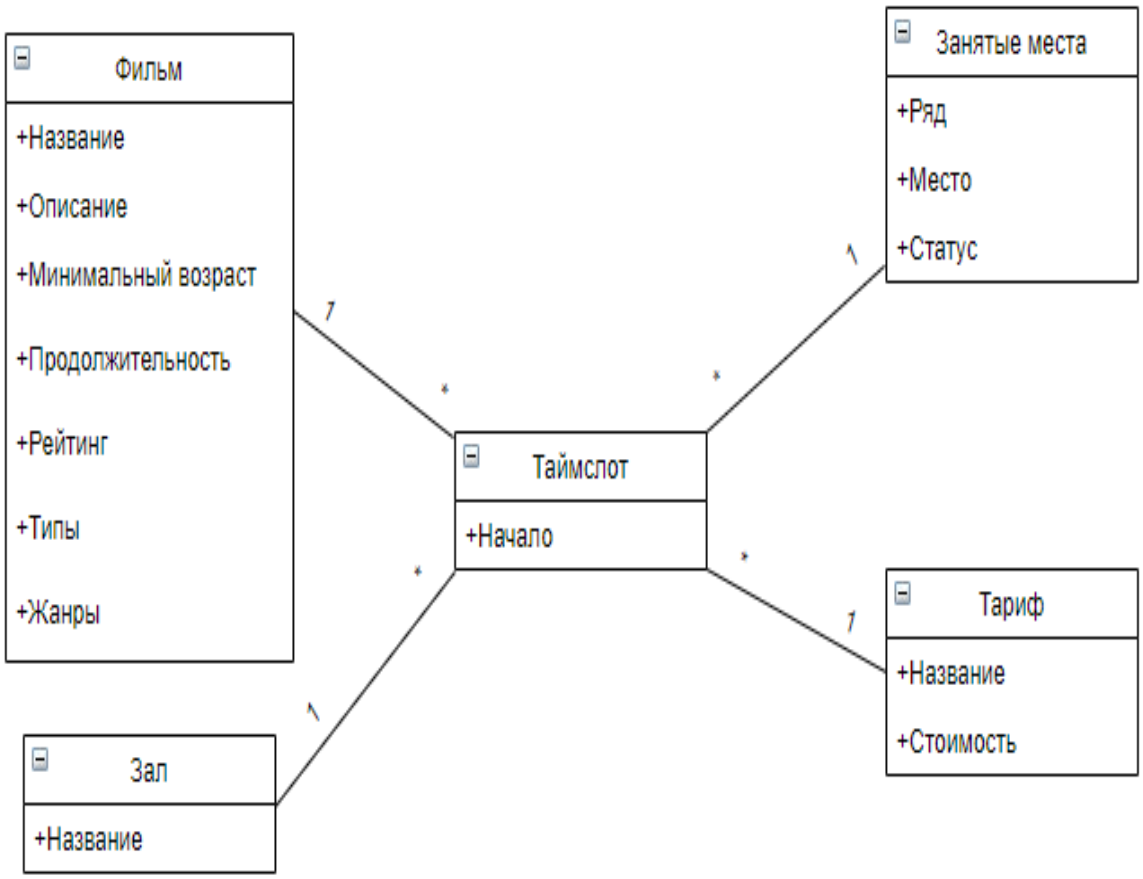


Рисунок 3.1 – ER-диаграмма базы данных

Все таблицы в этой базе данных находятся в третьей нормальной форме. Кроме того, каждая таблица имеет только один первичный ключ, что оз-

начает, что разработанная база данных находится в обычной форме Бойса-Кодда.

3.2 Даталогическое проектирование

На рисунке 3.2 показана схема реляционной базы данных для разрабатываемой системы.

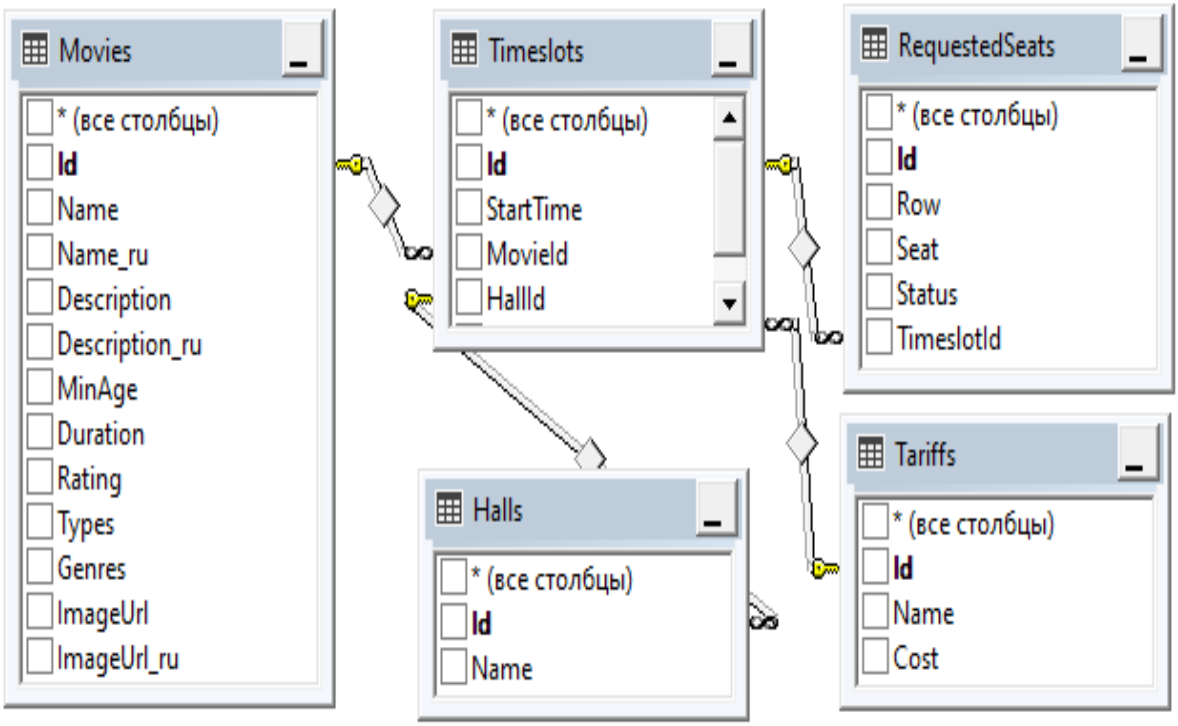


Рисунок 3.2 – Схема реляционной базы данных

База данных предназначена для хранения всей долговременной информации системы: фильмы, залы, таймслоты, купленные или зарезервированные билеты и тарифы.

Эта база данных является реляционной и реализована с использованием системы управления базами данных MS SQL Server 2019.

Описание назначения и свойств полей базы данных приведено в таблицах 3.1-3.5.

Таблица 3.1 – RequestedSeats (зарезервированные места)

	Тип данных	Значение по умолчанию	Обязатель-ность	Первичный ключ	Внешний ключ	Ограничения
Id	int	-	+	+	-	-
Row	int	-	+	-	-	-
Seat	int	-	+	-	-	-
Status	int	-	+	-	-	-
TimeslotId	int	-	+	-	+	-

В данной таблице хранятся данные о ряде (Row), месте (Seat) и тайм-слоте (TimeslotId).

Таблица 3.2 – Movies (фильмы кинотеатра)

	Тип данных	Значение по умолчанию	Обязатель-ность	Первичный ключ	Внешний ключ	Ограничения
Id	int	-	+	+	-	-
Name	nvarchar(100)	-	+	-	-	-
Name_ru	nvarchar(100)	null	-	-	-	-
Description	nvarchar(max)	null	-	-	-	-
Description_ru	nvarchar(max)	null	-	-	-	-
MinAge	int	null	-	-	-	-
Duration	int	null	-	-	-	-
Rating	real	null	-	-	-	-
Types	nvarchar(20)	null	-	-	-	-
Genres	nvarchar(20)	null	-	-	-	-
ImageUrl	nvarchar(400)	null	-	-	-	-
ImageUrl_ru	nvarchar(400)	null	-	-	-	-

В данной таблице хранятся все данные по фильмам – название фильма (Name) и аналог для русского языка (Name_ru), описание (Description) и аналог для русского языка (Description_ru), минимальный возраст (MinAge), продолжительность (Duration), рейтинг (Rating), типы (Types), жан-

ры (Genres), ссылка на изображение (ImageUrl) и аналог для русского языка (ImageUrl_ru).

Таблица 3.3 – RequestedSeats (зарезервированные места)

	Тип данных	Значение по умолчанию	Обязательность	Первичный ключ	Внешний ключ	Ограничения
Id	int	-	+	+	-	-
Row	int	-	+	-	-	-
Seat	int	-	+	-	-	-
Status	int	-	+	-	-	-
TimeslotId	int	-	+	-	+	-

В данной таблице хранятся данные о ряде (Row), месте (Seat) и таймслоте (TimeslotId).

Таблица 3.4 – Tariffs (тарифы)

	Тип данных	Значение по умолчанию	Обязательность	Первичный ключ	Внешний ключ	Ограничения
Id	int	-	+	+	-	-
Name	double	-	+	-	-	-
Cost	money	-	+	-	-	-

В данной таблице хранится название тарифа (Name) и его стоимость (Cost).

Таблица 3.5 – Halls (залы кинотеатра)

	Тип данных	Значение по умолчанию	Обязательность	Первичный ключ	Внешний ключ	Ограничения
Id	int	-	+	+	-	-
Name	nvarchar(100)	-	+	-	-	-

В данной таблице хранятся данные залов – название (Name).

3.3 Выводы по разделу

В этом разделе была разработана база данных для системы.

Прежде всего, был осуществлен концептуальный дизайн, в ходе которого сущности и отношения были впервые идентифицированы в системе, и поэтому на их основе была построена диаграмма ER. Таким образом, было доказано, что разрабатываемая база данных находится в нормальной форме Бойса-Кодда.

После этого был осуществлен переход к дизайну данных – была построена схема реляционной базы данных, дано описание всех таблиц и атрибутов, содержащихся в ней.

4 РЕАЛИЗАЦИЯ СИСТЕМЫ

4.1 Разработка алгоритмов

4.1.1 Общий алгоритм системы

На рисунке 4.1 приведена схема общего алгоритма системы.

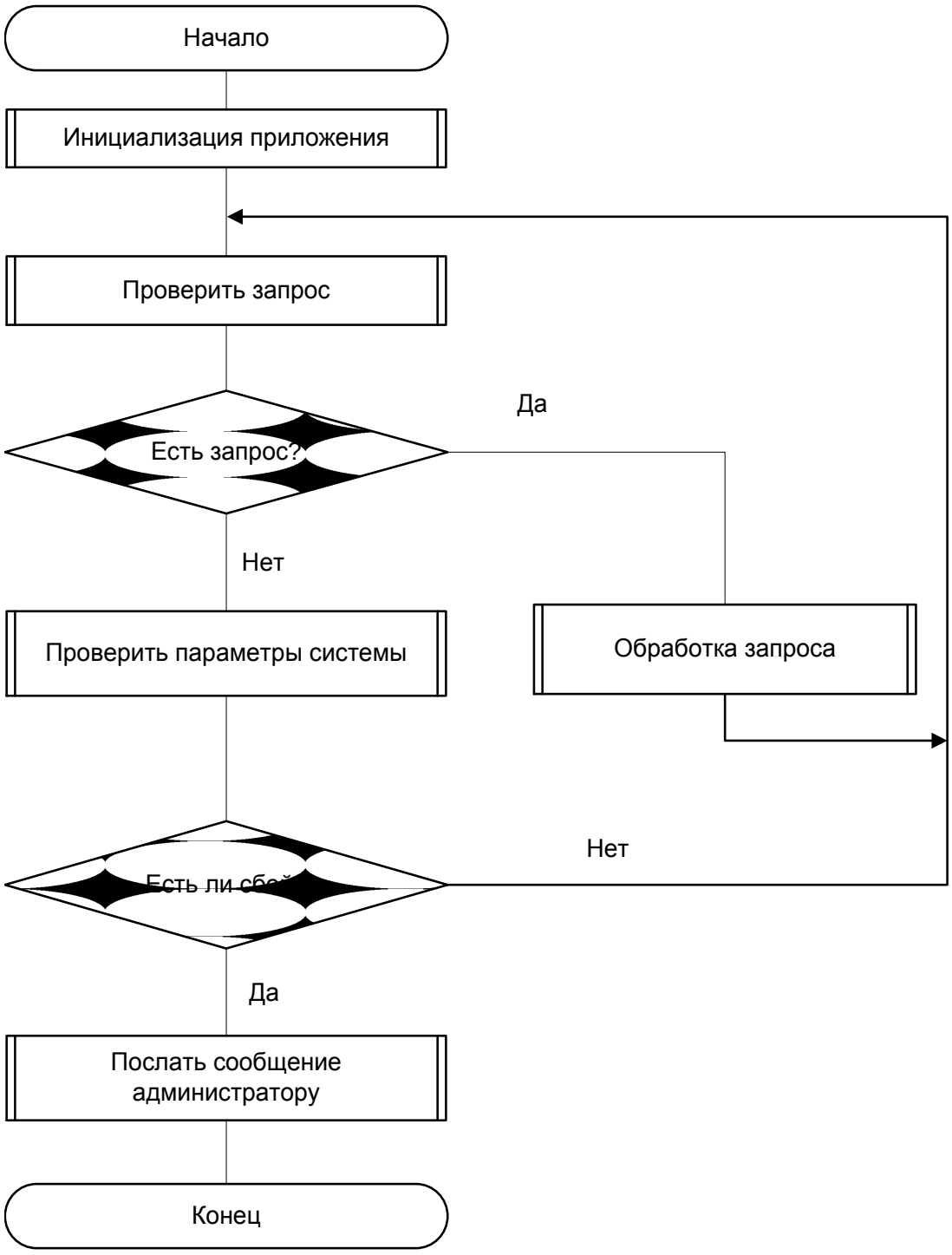


Рисунок 4.1 – Общий алгоритм системы

4.1.2 Алгоритм обработки запроса

На рисунке 4.2 приведена схема вспомогательного алгоритма, описывающего элемент «Обработка запроса».

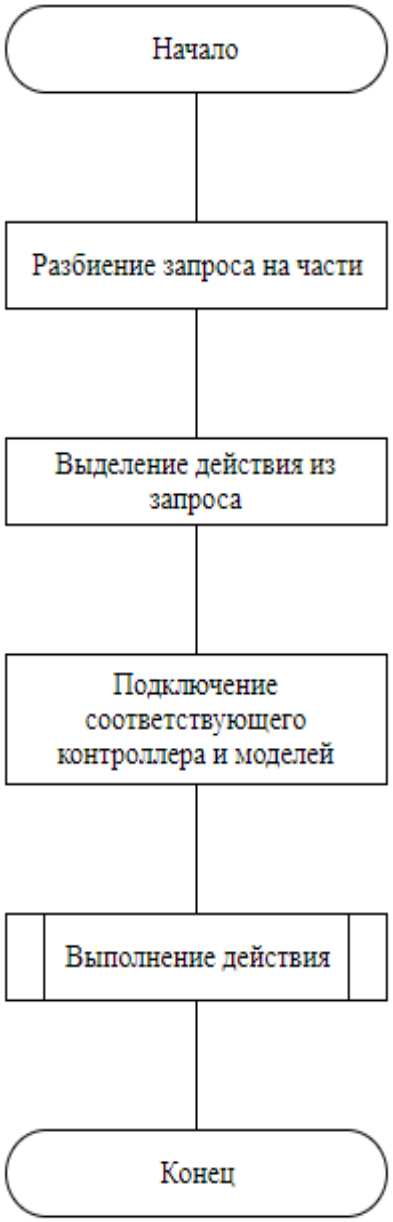


Рисунок 4.2 – Вспомогательный алгоритм «Обработка запроса»

4.1.3 Алгоритм выполнения действия

На рисунке 4.3 приведена схема вспомогательного алгоритма, описывающего элемент «Выполнение действия».

Рисунок 4.3 – Вспомогательный алгоритм «Выполнение действия»

4.2 Отладка и тестирование системы

4.2.1 Отладочный модуль

Для проверки производительности системы была разработана административная часть, которая позволяет вводить любые данные, предоставленные системой, и отправлять их на сервер, который, в свою очередь, отправляет выходные данные ответа в соответствующей форме.

Краткий обзор скриншотов:

– на рисунках с 4.4 по 4.7 продемонстрирована работа с фильмами;

– с 4.8 по 4.10 – осуществление покупки билета и результат;

– с 4.11 по 4.13 – работа с отчетами.

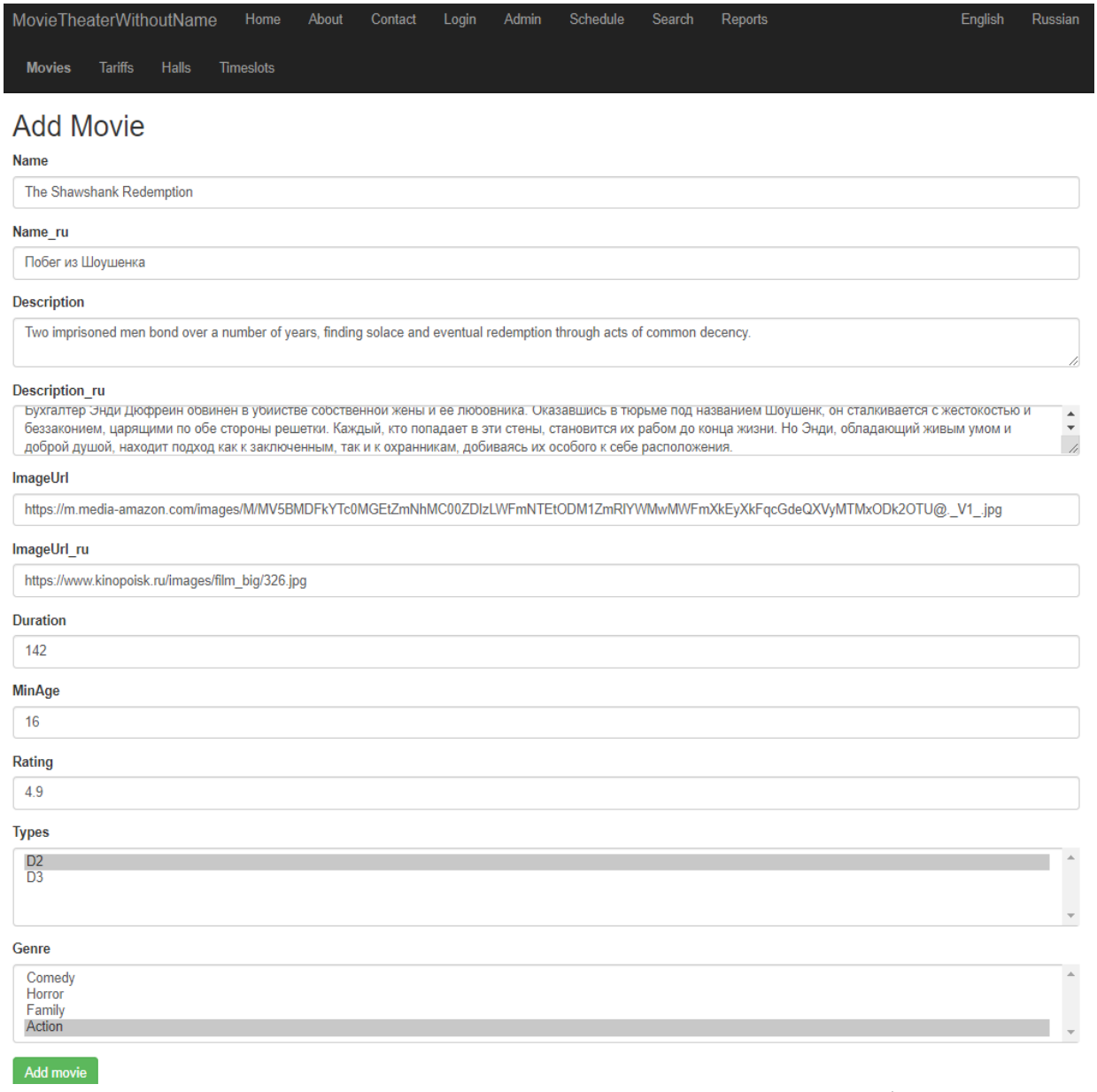


Рисунок 4.4 – Администраторская часть: добавление фильма

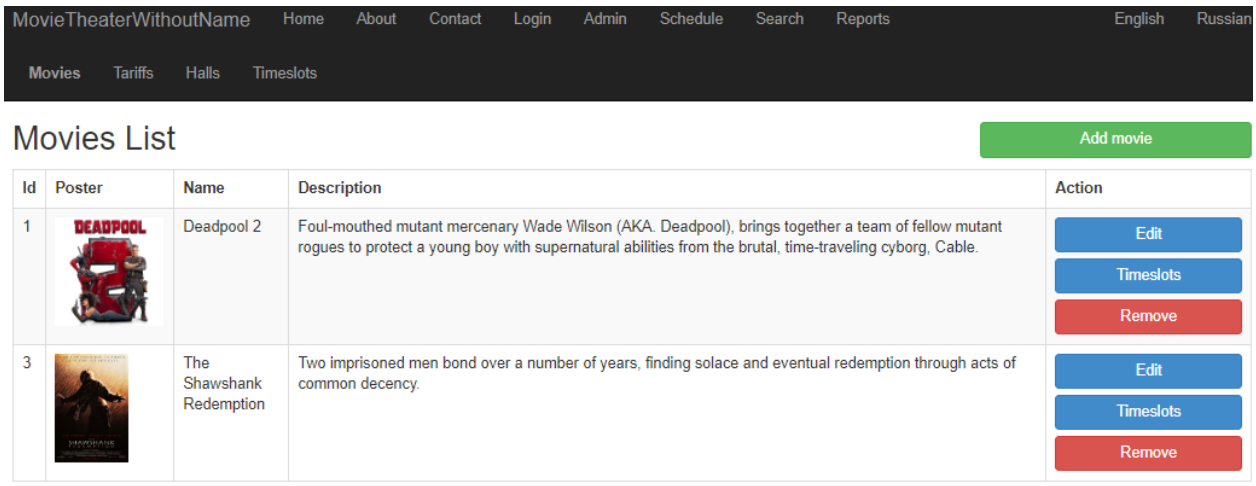


Рисунок 4.5 – Администраторская часть: результат добавления

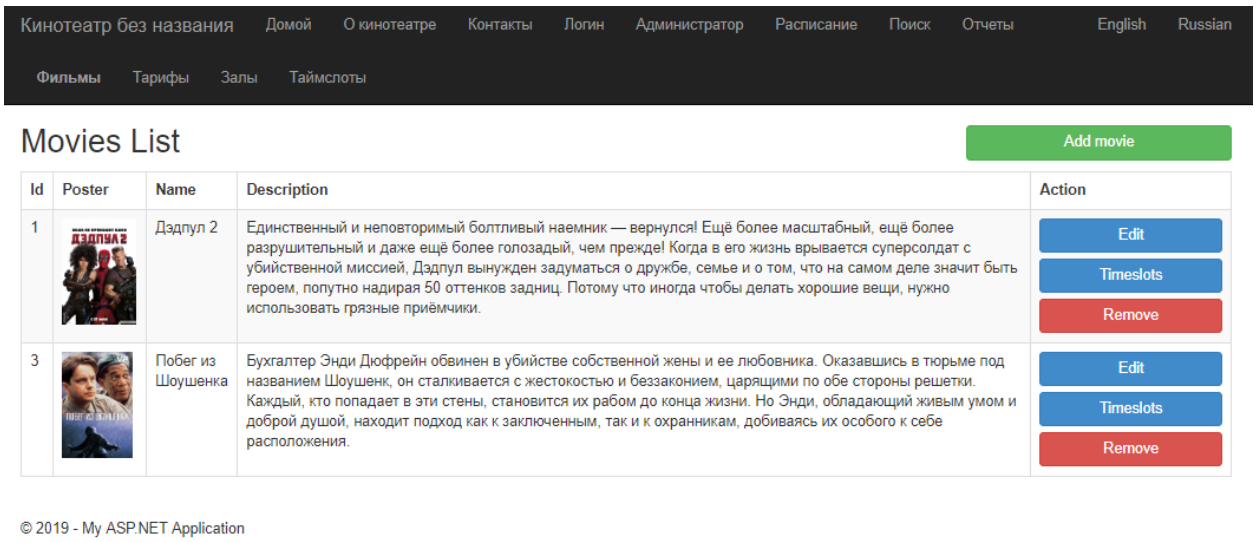


Рисунок 4.6 – Администраторская часть: результат добавления на русском языке

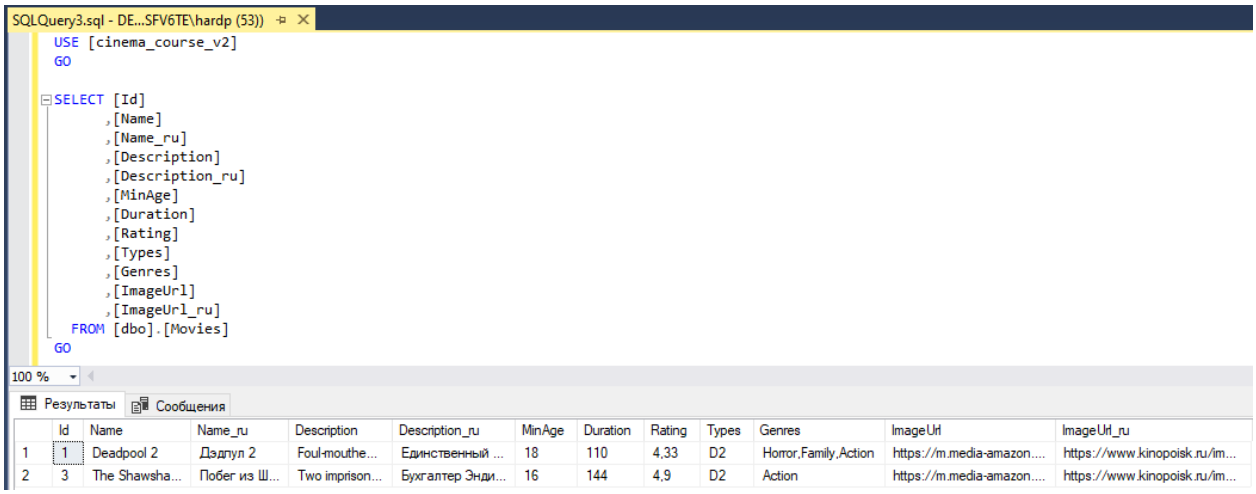


Рисунок 4.7 – Management Studio: список фильмов

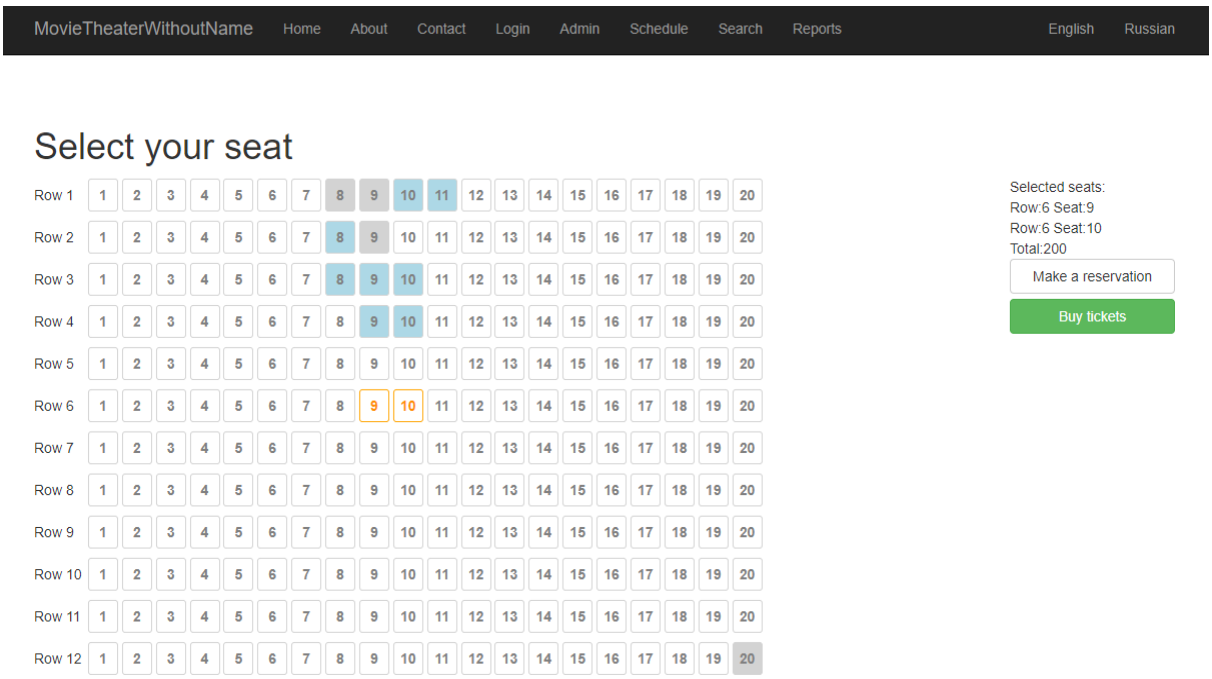


Рисунок 4.8 – Пользовательская часть: процесс покупки

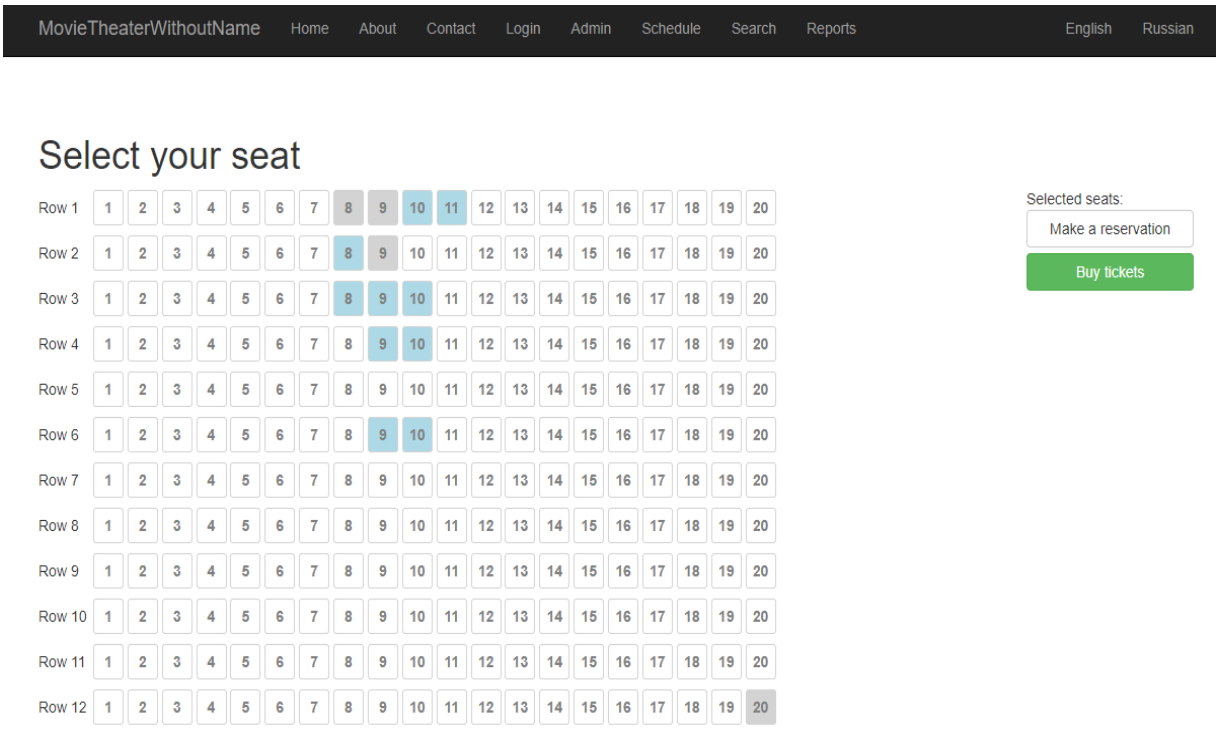
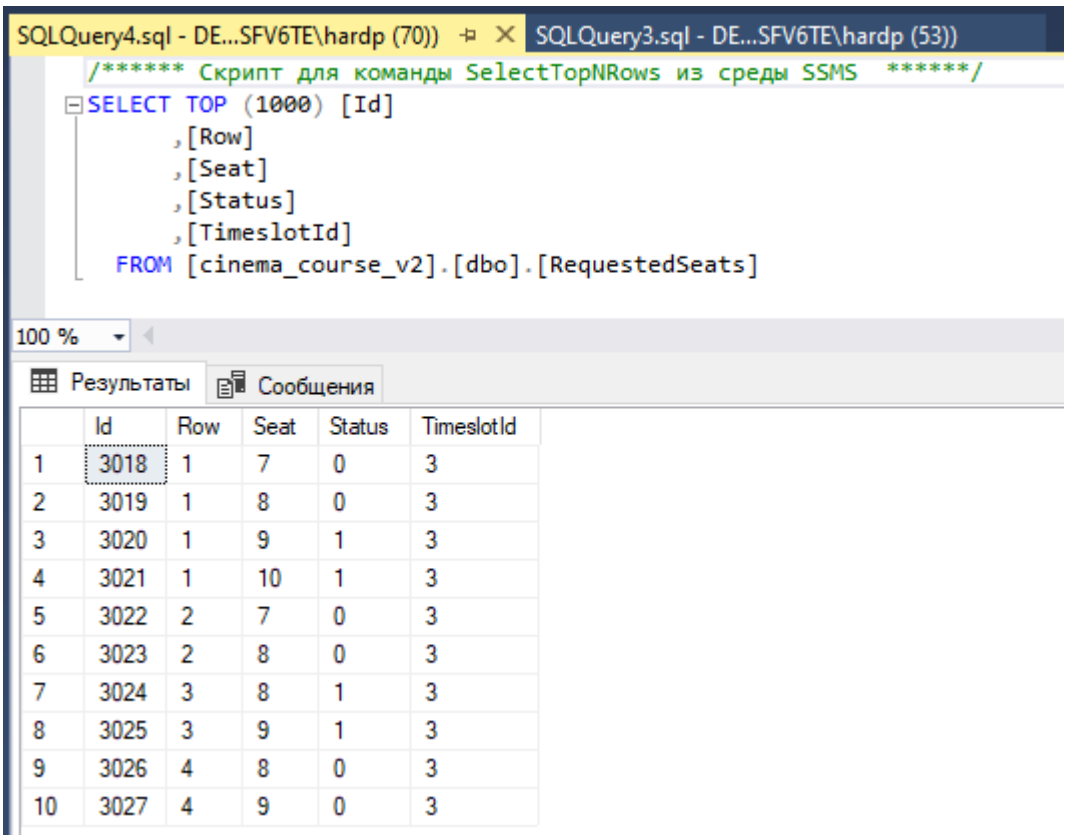


Рисунок 4.9 – Пользовательская часть: покупка произведена



	Id	Row	Seat	Status	TimeslotId
1	3018	1	7	0	3
2	3019	1	8	0	3
3	3020	1	9	1	3
4	3021	1	10	1	3
5	3022	2	7	0	3
6	3023	2	8	0	3
7	3024	3	8	1	3
8	3025	3	9	1	3
9	3026	4	8	0	3
10	3027	4	9	0	3

Рисунок 4.10 – Management Studio: покупка произведена

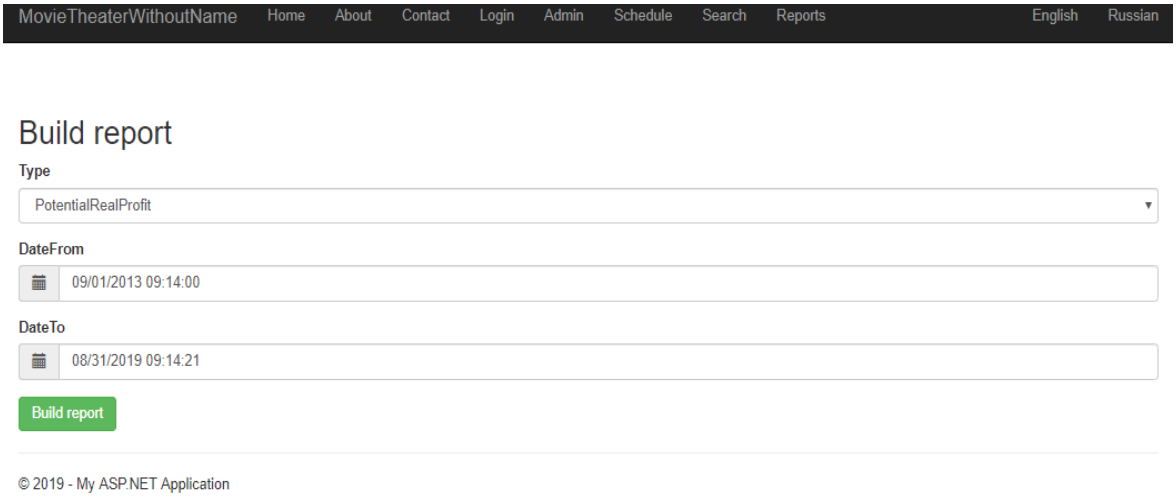


Рисунок 4.11 – Отчеты: Выбор нужного отчета и сроков

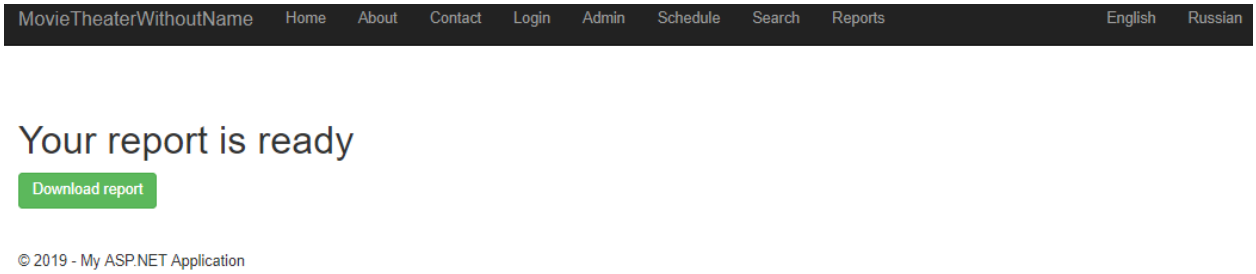


Рисунок 4.12 – Отчеты: скачивание

	A	B	C
1	Movie Name	Guaranteed Profit	Potential Profit
2	Deadpool 2	600	400

Рисунок 4.13 – Отчеты: результат

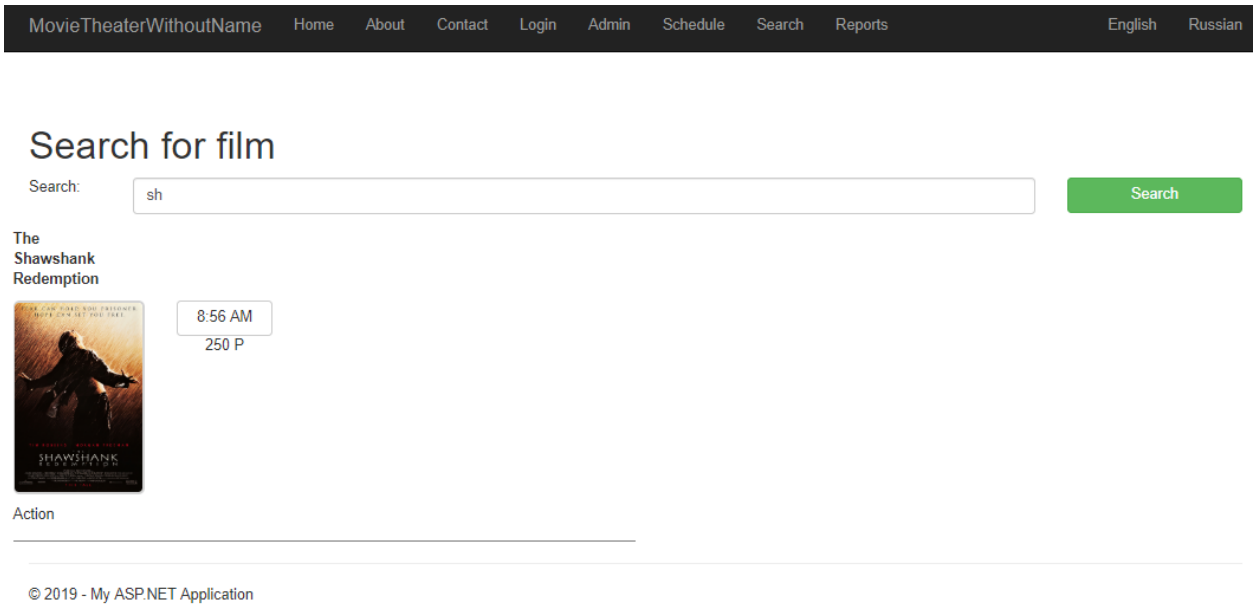


Рисунок 4.14 – Поиск: результат по содержанию

4.2.2 Тестирование системы

Тестирование программного обеспечения всегда актуально, так как именно оно позволяет быть уверенным в том, что продукт будет стабильно и корректно работать, отвечать на запросы пользователя, выдавать нужную ему информацию и обрабатывать исключительные ситуации.

Нельзя не упомянуть о стратегиях тестирования:

– восходящее – от меньшего к большему, однотипное тестирование, высокая вероятность найти ошибки;

– нисходящее – от большего к меньшему, наличие заглушек.

Также стоит сказать о стратегиях ящика:

– белый ящик – исходный код доступен;

– черный ящик – нет доступа к исходному коду;

– серый ящик – доступ к исходному коду есть, но он не используется.

4.3 Выводы по разделу

В этом разделе было дано описание реализации системы, включая разработку алгоритмов, отладку и тестирование сайта.

Алгоритмы включают общий системный алгоритм, а также вспомогательные алгоритмы для обработки запроса и выполнения действия.

Было проведено покрытие функциональных запросов.

Тестирование системы проводилось с использованием нисходящего тестирования и стратегии «серого ящика».

ЗАКЛЮЧЕНИЕ

В этой работе рассмотрел существующие технологии разработки и наиболее популярные языки программирования для веб-приложений, а также существующие приложения такого типа. Анализ требований к сайту.

Разработана архитектура системы и базы данных. В результате была разработана серверная часть приложения, позволяющая пользователям приобретать билеты на сеансы.

Для тестирования системы был выполнен ряд функциональных запросов, охватывающих систему, что позволило проверить стабильность системы.

Серверная часть приложения отвечает за обработку запросов, поступающих от клиентской части (разработанной совместно), которая выполняет функции пользовательского интерфейса.

Система позволяет размещать и изменять в базе данных информацию о таких субъектах, как:

– сеансы, состоящие из даты и времени проведения показа фильма;

– фильмы, состоящие из описания, возрастного ограничения, продолжительности, рейтинга и названия;

– залы, имеющие свое название;

– зарезервированные и купленные места на определенный сеанс;

– тарифы, указывающие стоимость определенного сеанса.

– таким образом, все поставленные задачи были успешно выполнены.

В дальнейшем планируется:

– разделение администраторской и пользовательской части;

– ввод платежной системы для полноценных покупок и расширение моделей и базы данных, для выдачи клиенту идентификационного номера;

– модерация чата, общение только для администраторов;

– планируется добавить третий язык для системы – китайский;

– изменение количества мест и рядов в зале;

− осуществление платежей с помощью Yandex.API.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Ульман, Л. MySQL [Электронный ресурс] / Л. Ульман. – Москва : ДМК Пресс, 2008. – 352 с. – Режим доступа: https://e.lanbook.com/book/1241 (дата обращения: 18.06.2019).

2 Хоган, Б. HTML5 и CSS3. Веб-разработка по стандартам нового поколения / Б. Хоган. – СПб.: Питер, 2014. – 320 с.

3 Одиночкина, С.В. Web-программирование PHP [Электронный ресурс] / С.В. Одиночкина. – Санкт-Петербург : НИУ ИТМО, 2012. – 79 с. – Режим доступа: https://e.lanbook.com/book/43562 (дата обращения: 18.06.2019).

4 Буч, Г. Язык UML. Руководство пользователя [Электронный ресурс] : руководство / Г. Буч, Д. Рамбо, И. Якобсон. – Москва : ДМК Пресс, 2008. – 496 с. – Режим доступа: https://e.lanbook.com/book/1246 (дата обращения: 10.06.2019).

5 Тарасов, С.В. СУБД для программиста. Базы данных изнутри [Электронный ресурс] / С.В. Тарасов. – Москва : СОЛОН-Пресс, 2015. – 320 с. – Режим доступа: https://e.lanbook.com/book/64959 (дата обращения: 10.06.2019).

6 Иванов, Д. Моделирование на UML [Электронный ресурс] / Д. Иванов, Ф. Новиков. – Санкт-Петербург : НИУ ИТМО, 2010. – 200 с. – Режим доступа: https://e.lanbook.com/book/40879 (дата обращения: 10.06.2019).

7 Прохоренок, Н.А. HTML, JavaScript, PHP, MySQL. Джентльменский набор веб-мастера /Н.П. Прохоренок – СПб. : БХВ-Петербург , 2009. – 840 с.

8 Ревунков, Г.И. Проектирование баз данных [Электронный ресурс] : учебное пособие / Г.И. Ревунков. – Москва : МГТУ им. Н.Э. Баумана, 2009. — 20 с. – Режим доступа: https://e.lanbook.com/book/52390 (дата обращения: 10.06.2019).

9 Фултон, Х. Программирование на языке Ruby [Электронный ресурс] : справочник / Х. Фултон. – Москва : ДМК Пресс, 2009. – 688 с. – Режим доступа: https://e.lanbook.com/book/1250 (дата обращения: 10.06.2019).

10 Хартл, М. Ruby on Rails для начинающих [Электронный ресурс] / М. Хартл. – Москва : ДМК Пресс, 2017. – 572 с. – Режим доступа: https://e.lanbook.com/book/90110 (дата обращения: 10.06.2019).

11 Саммерфилд, М. Python на практике [Электронный ресурс] : учебное пособие / М. Саммерфилд ; пер. с англ. А.А. Слинкин. – Москва : ДМК Пресс, 2014. – 338 с. – Режим доступа: https://e.lanbook.com/book/66480 (дата обращения: 10.06.2019).

12 Новиков, Ф.А. Учебно-методическое пособие по дисциплине «Анализ и проектирование на UML» [Электронный ресурс] : учебно-методическое пособие / Ф.А. Новиков. – Санкт-Петербург : НИУ ИТМО, 2007. – 286 с. – Режим доступа: https://e.lanbook.com/book/43540 (дата обращения: 10.06.2019).

13 Фримен, А. ASP.NET MVC 5 с примерами на C# 5.0. / Пер. с англ. под ред. Артеменко Ю. Н. [Электронный ресурс]. – Вильямс, 2015. – 736 с.

14 Астахова, И.Ф. СУБД: язык SQL в примерах и задачах. [Электронный ресурс] / И.Ф. Астахова, В.М. Мельников, А.П. Толстобров, В.В. Фертиков. — М. : Физматлит, 2009. — 168 с. — Режим доступа: http://e.lanbook.com/book/2101 (дата обращения: 12.06.2019).

15 Майерс, Г. Д. Искусство тестирования программ / Пер. с англ. под ред. Б. А. Позина / Г. Д. Майерс. – М. : Финансы и статистика, 1982. – 176 с.

16 Котеров, Д.В. PHP 5 : наиболее полн. рук. / Д. В. Котеров, А. Ф. Костарев. – Электрон. дан. – СПб. : БХВ-Петербург , 2010. – 1078 с.

17 Пейтон, К. PHP 5 и MySQL 5 / К. Пейтон, А. Меллер ; пер. с нем. под ред. С. М. Молявко. – Электрон. дан. – М. : Бином-Пресс , 2007. – 366 с.

18 Гольцман, В. MySQL 5.0. / В. Гольцман. – СПб. и др. Питер, 2010. – 253 с.

19 Кузнецов, С.Д. Базы данных учебник для вузов по направлению "Приклад. математика и информатика". / С. Д. Кузнецов. – М. : Академия, 2012. – 490 с.

20 PHP, Ruby и Python – краткая характеристика трёх языков программирования [Электронный ресурс] / Режим доступа: http://www.internet-technologies.ru/articles/article_1991.html, свободный (дата обращения: 12.06.2019).

21 ГОСТ 34.602-89. Техническое задание на создание автоматизированной системы. – М. : Стандартинформ, 2006. – 17 с.

22 ГОСТ 19.402-78. Описание программы. – М. : Стандартинформ, 2010. – 2 с.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ 1

ОПИСАНИЕ ПРОГРАММЫ
Сайт кинотеатра

2019

ОГЛАВЛЕНИЕ

П1.1 Общие сведения

Продукт «Сайт кинотеатра» представляет собой сайт для людей разных языковых групп, для покупки билетов на сеансы онлайн.

П1.2 Функциональное назначение

Данный сайт использует многоязычную систему для большего охвата аудитории.

Сайт обладает следующими возможностями:

– изменение языка;

– покупка билетов на интересующий сеанс;

– поиск по названию;

– отчеты по фильмам;

– администраторская часть.

П1.3 Описание логической структуры

Серверная часть состоит из модулей, обеспечивающих работу с базой данных. Клиентская часть визуализирует страницы на стороне пользователя.

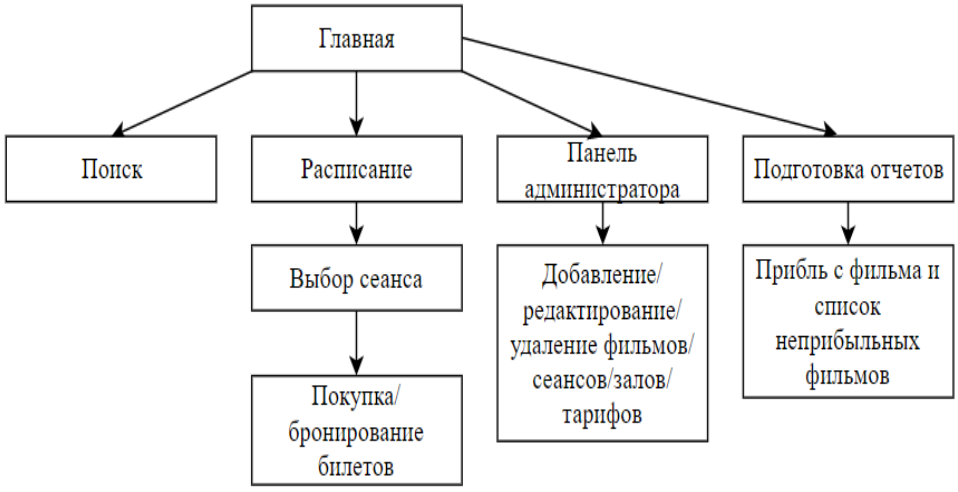


Рисунок П1.1 – Модульная структура сайта

П1.4 Используемые технические средства

Для обеспечения функционирования среды требуются следующие технические средства:

– IBM PC-совместимый компьютер;

– цветной монитор с диагональю не менее 15”;

– клавиатура;

– мышь.

Необходимые программно-аппаратные ресурсы для запуска локального сервера представлены в таблице 1.1.

Таблица П1.1 – Программно-аппаратная поддержка

Составляющие	Требования
Операционная система	Microsoft Windows 7 и выше
Оперативная память, Гб	2
Пространство на жестком диске, Мб	10
Рабочая частота ЦПУ, МГц	1000

П1.5 Входные и выходные данные

Входными данными для приложения является текстовая информация, вводимая пользователем в предназначенные для этого поля и информация о выборе предлагаемых опций, элементов меню, нажатии кнопок при управлении программой при помощи мыши.

На выходе программа формирует html файлы, представляющих собой готовую текущую страничку веб-сайта.

ПРИЛОЖЕНИЕ 2

ТЕКСТ ПРОГРАММЫ

2019

ОГЛАВЛЕНИЕ

П2.1 RouteConfig.cs Логика смены языка

```
namespace Cinema
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{language}/{controller}/{action}/{id}",
                defaults: new {
                    controller = "Home",
                    action = "Index",
                    id = UrlParameter.Optional,
                    language = "en-Us"
                }
            ).RouteHandler = new CustomRouteHandler();
        }
    }

    public class CustomRouteHandler : MvcRouteHandler
    {
        protected override IHttpHandler GetHttpHandler(RequestContext requestContext)
        {
            var lang = requestContext.RouteData.Values["language"].ToString();
            if (lang == "ru-Ru" || lang == "en-Us")
            {
                    Thread.CurrentThread.CurrentCulture =
                    CultureInfo.CreateSpecificCulture(lang);
                    Thread.CurrentThread.CurrentUICulture = new CultureInfo(lang);
            }
            return base.GetHttpHandler(requestContext);
        }
    }
}
```

П2.2 _Layout.cshtml Представление смены языка

```
<ul class="nav navbar-nav navbar-right">
      <li>
```

```
            @Html.ActionLink("English",
            ViewContext.RouteData.Values["action"].ToString(),
            ViewContext.RouteData.Values["controller"].ToString(),
            new { language = "en-Us" },
            null)
      </li>
      <li>
            @Html.ActionLink("Russian",
            ViewContext.RouteData.Values["action"].ToString(),
            ViewContext.RouteData.Values["controller"].ToString(),
            new { language = "ru-Ru" },
            null)
      </li>
</ul>
```

П2.3 schedule-film.js Ajax-скрипт покупки или бронирования билетов `$(document).ready(`

```
    function() {
        var seatTemplateContainer = document.querySelector(".js-selected-seat-
template");
        if (seatTemplateContainer === null || seatTemplateContainer === undefined) {
            return;
        }
        var source = seatTemplateContainer.innerHTML;
        var template = Handlebars.compile(source);
        var currentCost = $('.js-seat-container')[0].dataset.currentCost;
        var currentTimeslotId = $('.js-seat-container')[0].dataset.currentTimeslotId;
        var selectedSeats = {
            addedSeats: [],
            sum: 0
        };

        $(".js-seat-container").on('click',
            '.js-seat-selector',
            function(e) {
                var targetElem = e.currentTarget;
                var dataSet = targetElem.dataset;
                var newSeat = {
                    row: dataSet.seatRow,
                    seat: dataSet.seatCol,
                    elem: targetElem
                };
                var existingSeatIndex = -1;
                for (var i = 0; i < selectedSeats.addedSeats.length; i++) {
                    var currentSeat = selectedSeats.addedSeats[i];
                   if (currentSeat.row === newSeat.row && currentSeat.seat ===
                   newSeat.seat) {
                        existingSeatIndex = i;
                        break;
                    }
                }
                if (existingSeatIndex !== -1) {
                    selectedSeats.addedSeats.splice(existingSeatIndex, 1);
                } else {
                    selectedSeats.addedSeats.push(newSeat);
                }
                selectedSeats.sum = currentCost * selectedSeats.addedSeats.length;
```

```
                var resultHtml = template(selectedSeats);
                $(".js-seat-result-container").html(resultHtml);
            });

        $(".js-seat-container").on('click',
            '.js-reserve-seats',
            function(e) {
                sendSeatsToServer('reserve');
            });

        $(".js-seat-container").on('click',
            '.js-buy-seats',
            function(e) {
                sendSeatsToServer('buy');
            });

        function sendSeatsToServer(status) {
            var resultModel = {
                seatsRequest: selectedSeats,
                selectedStatus: status,
                timeslotId: currentTimeslotId
            };

            $.ajax({
                url: "processRequest",
                type: 'POST',
                dataType: 'json',
                contentType: 'application/json;charset=utf-8',
                data: JSON.stringify(resultModel)
            }).done(function() {
                for (var i = 0; i < selectedSeats.addedSeats.length; i++) {
                    var currentSeat = selectedSeats.addedSeats[i].elem;
                    if (status === 'reserve') {
                        currentSeat.parentNode.classList.add('is-reserved');
                    } else {
                        currentSeat.parentNode.classList.add('is-sold');
                    }
                    currentSeat.checked = false;
                    currentSeat.disabled = true;
                }
                selectedSeats.addedSeats = [];
                var resultHtml = template(selectedSeats);
                $(".js-seat-result-container").html(resultHtml);
            }).fail(function() {
                    alert("Order processing failed. Please, contact system adminis-
                    trator.");
            });
        }
    });
```

П2.4 TicketsController.cs Контроллер покупки

```
public string ProcessRequest(SeatsProcessRequest request)
{
      var requestProcessingResult =
      _ticketService.AddRequestedSeatsToTimeslot(request);
      return JsonConvert.SerializeObject(new
      {
            requestResult = requestProcessingResult
      });
}
```

П2.5 SqlTicketService.cs Собрать модель и отправить запрос БД

```
public bool AddRequestedSeatsToTimeslot(SeatsProcessRequest request)
{
	var requestTable = new DataTable("TimeslotSeatRequest");
	requestTable.Columns.Add("Row", typeof(int));
	requestTable.Columns.Add("Seat", typeof(int));
	requestTable.Columns.Add("Status", typeof(int));
	foreach (var seatRequest in request.SeatsRequest.AddedSeats)
	{
		requestTable.Rows.Add(seatRequest.Row, seatRequest.Seat, re-
		quest.SelectedStatus);
	}
	SqlParameter requestTableparameter = new SqlParameter
	{
		ParameterName = "@seatsRequest",
		SqlDbType = SqlDbType.Structured,
		Value = requestTable
	};
	var parameters = new[]
	{
		requestTableparameter,
		new SqlParameter("@TimeslotId", request.TimeslotId)
	};
	return DatabaseUtil.ExecuteNonQuery("AddRequestedSeatsToTimeslot", parameters)
	!= 0;
}
```

П2.6 AddRequestedSeatsToTimeslot Процедура запроса

```
USE [cinema_course_v2]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[AddRequestedSeat]
	@Row int,
	@Seat int,
	@Status int,
	@TimeslotId int
AS
BEGIN
	INSERT INTO RequestedSeats
	([Row],
```

```
      Seat,
      [Status],
      TimeslotId)
      VALUES(
      @Row,
      @Seat,
      @Status,
      @TimeslotId)
END
```

П2.7 RequestedSeatsDuplicatesCheck Триггер проверки покупки

```
USE [cinema_course_v2]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [dbo].[RequestedSeatsDuplicatesCheck] ON [dbo].[RequestedSeats]
INSTEAD OF INSERT
AS
if exists ( select * from RequestedSeats rs
    inner join inserted i on
             i.Seat = rs.Seat and
             i.[Row] = rs.[Row] and
             i.[TimeslotId] = rs.TimeslotId)
begin
    rollback
    RAISERROR ('Request for sold seat detected', 16, 1);
end
ELSE
BEGIN
      INSERT INTO RequestedSeats ([Row],[Seat],[Status],TimeslotId)
      SELECT i.[Row],i.Seat,i.[Status],i.[TimeslotId]
      FROM inserted i
END
```

П2.8 ReportBaseStrategy.cs Базовая стратегия отчетов

```
namespace Cinema.Reports
{
    public abstract class ReportBaseStrategy<T> : IReportBuilder
    {
        protected SqlDatabaseUtil DatabaseUtil { get; set; }

        public Dictionary<string, object> Parameters { get; set; }

        protected ReportBaseStrategy(IMapper mapper)
        {
            DatabaseUtil = new SqlDatabaseUtil(mapper);
            Parameters = new Dictionary<string, object>();
        }

        public string BuildReport()
        {
            var model = GetDataModel();

            CreateReportsDirectoryIfNotExists();

            var filename = Path.Combine(Constants.ReportsDirectory,
string.Concat(InternalGetDownloadFileName(), DateTime.Now.ToString("_yyyyMMdd-
hhmmss"), GetTargetExtension()));

            InternalBuildReport(filename, model);

            return GetFileLinkUrl(filename);
        }

        protected string TemplateFileName => InternalGetTemplateFileName();

        protected abstract string InternalGetTemplateFileName();
        protected abstract string InternalGetDownloadFileName();

        protected abstract T GetDataModel();

        protected string GetFileLinkUrl(string filePath)
        {
            var approot = HostingEnvironment.ApplicationPhysicalPath.TrimEnd('\\');
            return filePath.Replace(approot, string.Empty).Replace('\\', '/');
        }
```

```
        protected void InternalBuildReport(string filename, T model)
        {
            var templatePath =
HostingEnvironment.MapPath(Path.Combine(Constants.ExcelTemplatesDirectory,
TemplateFileName));
            if (string.IsNullOrEmpty(templatePath))
            {
                throw new ApplicationException($"Unable to map path
\"{templatePath}\".");
            }

            using (var templateFileStream = new FileStream(templatePath,
FileMode.Open, FileAccess.Read))
            {
                var workbook = WorkbookFactory.Create(templateFileStream);

                ProcessWorkBook(workbook, model);

                SaveWorkbook(workbook, filename);
            }
        }

        protected abstract void ProcessWorkBook(IWorkbook workbook, T model);

        private void SaveWorkbook(IWorkbook workbook, string filename)
        {
            var targetFilePath = HostingEnvironment.MapPath(filename);
            if (string.IsNullOrEmpty(targetFilePath))
            {
                throw new ApplicationException($"Unable to map path \"{filename}\"");
            }

            if (File.Exists(targetFilePath))
            {
                File.Delete(targetFilePath);
            }

            using (var outputFileStream = new FileStream(targetFilePath,
FileMode.CreateNew))
            {
                workbook.Write(outputFileStream);
                outputFileStream.Close();
            }
        }

        private object GetTargetExtension()
        {
            return Path.GetExtension(TemplateFileName);
        }

        private static void CreateReportsDirectoryIfNotExists()
        {
            try
            {
                var reportsPath =
HostingEnvironment.MapPath(Constants.ReportsDirectory);
                if (string.IsNullOrEmpty(reportsPath))
                {
                    throw new ApplicationException($"Unable to map path
'{reportsPath}'");
                }
```

```
                if (!Directory.Exists(reportsPath))
                {
                    Directory.CreateDirectory(reportsPath);
                }
            }
            catch (Exception e)
            {
                throw new ApplicationException($"Unable to create Reports directo-
ry.", e);
            }
        }
    }
}
```

П2.9 search-film.js Ajax-скрипт поиска фильма по названию

```
$(document).ready(
    function() {
        var termInput = $(".js-search-film-term")[0];
        if (termInput === null || termInput === undefined) {
            return;
        }
        var searchResultTemplateContainer = $(".js-search-result-item")[0].innerHTML;
        var resultTemplate = Handlebars.compile(searchResultTemplateContainer);

        var pagingTemplateContainer = $(".js-search-page-item")[0].innerHTML;
        var pagingTemplate = Handlebars.compile(pagingTemplateContainer);
        var currentTerm = termInput.value;

        Handlebars.registerHelper('for',
            function(from, to, incr, block) {
                var accum = '';
                for (var i = from; i <= to; i += incr)
                    accum += block.fn(i);
                return accum;
            });


        $(".js-search-page-container").on('click',
            '.js-page-selector',
            function(e) {
                var targetElem = e.currentTarget;
                var dataSet = targetElem.dataset;
                searchFilm(dataSet.selectedPage);
            });

        $(".js-search-film-submit").on('click',
            function() {
                currentTerm = termInput.value;
                searchFilm(1);
            });

        function searchFilm(selectedPage) {
            var reqestModel = {
                term: currentTerm,
                currentPage: selectedPage
            };
            $.ajax({
```

```
                url: "SearchForFilms",
                type: 'POST',
                dataType: 'json',
                contentType: 'application/json;charset=utf-8',
                data: JSON.stringify(reqestModel)
            }).done(function(result) {
                var resultHtml = resultTemplate(result);
                $(".js-search-result-container").html(resultHtml);
                if (result.ShowPaging) {
                    var resultPaging = pagingTemplate(result);
                    $(".js-search-page-container").html(resultPaging);
                } else {
                    $(".js-search-page-container").html('');
                }
            }).fail(function() {
                alert("Search request processing failed. Please, contact system ad-
ministrator.");
            });});});
```

П2.10 TicketsController.cs Контроллер поиска филмьа по названию

```
public string SearchForFilms(string term, int currentPage = 1, int pageSize = 1)
{
	var cacheKey =
	string.Format("Tickets_SearchForFilms_Term:{0}_CurrentPage:{1}_PageSize:{2}",
	term, currentPage, pageSize);
	var cacheResult = _cache.CacheResult(() =>
	{
		var allResults = _ticketService.SearchMoviesByTerm(term);
		var totalPages = Math.Ceiling(allResults.Length / (double)pageSize);
		var currentPageResults = allResults.Skip((currentPage - 1) *
		pageSize).Take(pageSize).ToArray();
		var model = new SearchFilmResult
		{
			Result = currentPageResults,
			TotalPages = (int)totalPages,
			CurrentPage = currentPage,
			ShowPaging = totalPages > 1
		};
		var resultModel = JsonConvert.SerializeObject(model);
		return resultModel;
	},
	cacheKey);
	return cacheResult;
}
```

П2.11 SqlTicketService.cs Собрать модель поиска и отправить запрос БД

```
public MovieListItem[] SearchMoviesByTerm(string term)
{
	var parameters = new[]
	{
		new SqlParameter("@Term", term)
	};
	return DatabaseUtil.Execute("SearchMoviesByTerm", parameters,
	_movieListItemMappingFunc).ToArray();
}
```

П2.12 SearchMoviesByTerm Процедура поиска в базе данных

```
USE [cinema_course_v2]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [dbo].[SearchMoviesByTerm]
	@Term nvarchar(100)
AS
BEGIN

	Declare @SqlTerm nvarchar(100) = '%'+@Term+'%'
	Declare @SelectedMovieIds TABLE(Id int)

	INSERT INTO
		@SelectedMovieIds
	SELECT
		Id
	FROM
		Movies
	WHERE
		[Description] like @SqlTerm OR
		[Name] like @SqlTerm

	/*Select 1: Movies*/
	Select
		*
	FROM
		Movies
	Where
		Id = ANY (SELECT Id from @SelectedMovieIds)

	/*Select 2: Timeslot Tags*/

	Select
		ts.Id [TimeslotId],
		ts.StartTime [StartTime],
		tf.Cost [Cost],
		ts.MovieId
	FROM
```

```
            Timeslots ts
            JOIN Tariffs tf ON tf.Id=ts.TariffId
      Where
            ts.MovieId = ANY (SELECT Id from @SelectedMovieIds)

END
```