СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc450000324)

[1.АНАЛИЗ ЗАДАНИЯ И ПОСТАНОВКА ЗАДАЧ 6](#_Toc450000325)

[1.1 Постановка задач 6](#_Toc450000326)

[1.2 Описание предметной области 6](#_Toc450000327)

[1.3 Выбор и обоснование средств и методов решения задач 7](#_Toc450000328)

[2.СОЗДНАНИЕ ФУНКЦИОНАЛЬНОЙ МОДЕЛИ ПО 9](#_Toc450000329)

[2.1 Разработка диаграммы вариантов использования 9](#_Toc450000330)

[2.2 Оценка трудозатрат на разработку ПО на основе диаграммы вариантов использования 10](#_Toc450000331)

[2.2.1 Определение весовых показателей действующих лиц 10](#_Toc450000332)

[2.2.2 Определение весовых показателей вариантов использования 10](#_Toc450000333)

[2.2.3 Определение технической сложности проекта 10](#_Toc450000334)

[2.2.4 Определение уровня квалификации разработчиков 11](#_Toc450000335)

[2.2.5 Оценка трудозатрат проекта 12](#_Toc450000336)

[2.3 Создание макета графического интерфейса пользователя 13](#_Toc450000337)

[3. СОЗДАНИЕ ЛОГИЧЕСКОЙ МОДЕЛИ ПО 15](#_Toc450000338)

[3.1 Разработка диаграммы классов 15](#_Toc450000339)

[3.1.1 Описание модели 15](#_Toc450000340)

[3.1.2 Проектирование уровня данных 16](#_Toc450000341)

[3.1.3 Проектирование уровня интерфейса и бизнес-логики 16](#_Toc450000342)

[3.2 Разработка диаграмм последовательностей 16](#_Toc450000343)

[3.3 Разработка диаграмм деятельности 17](#_Toc450000344)

[3.4 Разработка диаграмм состояний 18](#_Toc450000345)

[4 СОЗДАНИЕ ФИЗИЧЕСКОЙ МОДЕЛИ ПО 19](#_Toc450000346)

[4.1 Разработка диаграммы компонентов 19](#_Toc450000347)

[4.2 Разработка диаграммы развертывания 20](#_Toc450000348)

[5 ОПИСАНИЕ РЕАЛИЗАЦИИ И ТЕСТИРОВАНИЯ ПО 21](#_Toc450000349)

[5.1 Разработка и построение интерфейса 21](#_Toc450000350)

[5.2 Результат тестирования ПО 21](#_Toc450000351)

[ЗАКЛЮЧЕНИЕ 23](#_Toc450000352)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 24](#_Toc450000353)

[ПРИЛОЖЕНИЕ А 25](#_Toc450000354)

[ПРИЛОЖЕНИЕ Б 27](#_Toc450000355)

[ПРИЛОЖЕНИЕ В 28](#_Toc450000356)

[ПРИЛОЖЕНИЕ Г 29](#_Toc450000357)

[ПРИЛОЖЕНИЕ Д 31](#_Toc450000358)

[ПРИЛОЖЕНИЕ E 34](#_Toc450000359)

[ПРИЛОЖЕНИЕ Ж 36](#_Toc450000360)

[ПРИЛОЖЕНИЕ З 38](#_Toc450000361)

[ПРИЛОЖЕНИЕ И 39](#_Toc450000362)

## ВВЕДЕНИЕ

Сегодня фильмы являются неотъемлемой частью в жизни каждого человека. Узнавать о новинках и наиболее популярных фильмах, оставлять комментарии и просматривать рейтинги. Все это возможно делать с помощью различных сайтов, на которых можно быстро найти необходимые фильмы, список которых удобно хранить в базе данных. Базы данных занимают одно из первых мест среди различных структур хранения данных. Базы данных помогают систематизировать информацию из определенной предметной области, облегчают доступ к данным, поиск и предоставление необходимых сведений.

Создание автоматизированной системы в виде информационной системы основанной на современных технологиях сбора, обработки, анализа, передачи и хранения информации смогло бы значительно облегчить поиск необходимой информации. Такой подход имеет решающее значение для уменьшения временных затрат на обработку информации и другие показатели эффективности выполнения работы.

Поэтому разработка и проектирование информационной системы составления рейтингов фильмов на сайте с фильмами является актуальной темой сегодняшнего дня.

В данной курсовой работе рассматривалась проблема автоматизированного хранения данных о фильмах на сайте с фильмами, на основании которых пользователи смогут просматривать рейтинги.

## 1.АНАЛИЗ ЗАДАНИЯ И ПОСТАНОВКА ЗАДАЧ

1.1 Постановка задач

Заданием данной курсовой работы является разработка программного обеспечения информационной системы составления рейтингов фильмов на сайте с фильмами, которая должна позволять пользователю выполнять следующие действия:

* возможность авторизации пользователя;
* возможность регистрации пользователя;
* возможность добавления, удаления фильма;
* возможность добавления, удаления рейтинга;
* возможность редактирования описания фильма;
* возможность добавления фильма в закладки;
* возможность просматривать результаты запросов;
* возможность оставлять комментарий.

Система должна быть представлена в двух частях:

* база данных, основанная на системе управления базами данных (СУБД), которая поддерживает серверный режим работы;
* клиентское приложение.

1.2 Описание предметной области

Предметной областью проектируемой системы является система составления рейтингов фильмов на сайте с фильмами. В системе выполняются следующие действия: добавление фильмов; редактирование описания; удаление фильмов; добавление рейтинга; удаление рейтинга.

Для построения информационной системы требуется для начала выделить необходимый набор сущностей, которые описывают эту систему. Данный набор должен удовлетворять всем условиям на проектирование системы.

В ходе анализа технического задания и предметной области был определен следующий минимальный набор сущностей, необходимый для проектирования информационной системы составления рейтингов фильмов на сайте с фильмами.

Администратор – добавляет и удаляет фильмы, редактирует описание, добавляет и удаляет рейтинг.

Пользователь – просматривает фильм, добавляет фильм в закладки, оставляет комментарий, выставляет рейтинг.

1.3 Выбор и обоснование средств и методов решения задач

Для создания моделей анализа и проектирования объектно-ориентированных программных систем используют языки визуального моделирования. В процессе разработки программного обеспечения информационной системы составления рейтингов фильмов на сайте с фильмами будем использовать язык UML – стандартный язык для написания моделей анализа, проектирования и реализации объектно-ориентированных программных систем.

UML является языком широкого профиля, это открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML - моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна генерация кода. [1]

В процессе разработки данной автоматизированной информационной системы будем использовать CASE-средство Enterprise Architect 12, который представляет собой мощный UML-инструмент для профессионалов, занимающихся разработкой, тестированием и внедрением программного обеспечения. Enterprise Architect полностью поддерживает нотацию языка UML 2.1 и позволяет создавать диаграммы всех типов.

Для разработки базы данных будем использовать систему управления реляционными базами данных SQL. SQL формальный непроцедурный язык программирования, применяемый для создания, модификации и управления данными в произвольной реляционной базе данных.

Основные преимущества SQL:

* несмотря на наличие диалектов и различий в синтаксисе, в большинстве своём тексты SQL-запросов, содержащие DDL и DML, могут быть достаточно легко перенесены из одной СУБД в другую;
* использование языка SQL в программах стандартизировано международными организациями;
* язык SQL легко использовать в приложениях, которым необходимо обращаться к базам данных;
* с помощью SQL можно представить такую структуру данных, что тот или иной пользователь будет видеть различные их представления;
* язык SQL позволяет манипулировать структурой БД, тем самым обеспечивая гибкость с точки зрения приспособленности БД к изменяющимся требованиям предметной области. [2]

Для реализации приложения будем использовать среду разработки Microsoft Visual Studio 2015. Из языков программирования выбран C# – безопасный объектно-ориентированный язык, предназначенный для разработки разнообразных безопасных и мощных приложений, выполняемых в среде .NET Framework. Visual C# обеспечивает расширенный редактор кода, удобные пользовательские конструкторы интерфейса, интегрированный отладчик и многие другие средств, облегчающие разработку приложения. В языке C# удачно сочетаются испытанные средства программирования и предоставляется возможность для эффективного и практичного написания программ.

## 2.СОЗДНАНИЕ ФУНКЦИОНАЛЬНОЙ МОДЕЛИ ПО

2.1 **Разработка диаграммы вариантов использования**

Визуальное моделирование в UML можно представить, как некоторый процесс поуровневого спуска от наиболее общей и абстрактной концептуальной модели исходной системы к логической, а затем и к физической модели соответствующей программной системы. Для достижения этих целей вначале строится модель в форме так называемой диаграммы вариантов использования (use case diagram), которая описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования. Каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актером. Диаграмма вариантов может дополняться пояснительным текстом, который раскрывает смысл или семантику составляющих ее компонентов. Такой пояснительный текст получил название сценария.

Актером в данном случае является пользователь приложения (администратор или пользователь). Стандартным графическим обозначением актера на диаграммах является фигурка «человечка», под которой записывается конкретное имя актера.

Варианты использования для разрабатываемой информационной системы и их сценарии представлены в приложении А (таблицах А.1, А.2). Отдельный вариант использования обозначается на диаграмме эллипсом, внутри которого содержится его краткое название.

Между компонентами диаграммы вариантов использования могут существовать различные отношения, которые описывают взаимодействие экземпляров одних актеров и вариантов использования с экземплярами других актеров, и вариантов.

Между актером и вариантом использования возможен только один вид отношения – ассоциация, отображающая их взаимодействие. Между вариантами использования определены отношение обобщения и две разновидности отношения зависимости – включения и расширения. Отношение включения означает, что базовый элемент явно включает поведение другого элемента. Отношение расширения означает, что базовый элемент неявно включает поведение другого элемента; это отношение применяется для моделирования выбираемого поведения программы. [3]

Итоговая диаграмма вариантов использования для разрабатываемого программного обеспечения информационной системы составления рейтингов фильмов на сайте с фильмами, выполненная в CASE-средстве Enterprise Architect 12, представлена в приложении Б.

## 2.2 **Оценка трудозатрат на разработку ПО на основе диаграммы вариантов использования**

2.2.1 Определение весовых показателей действующих лиц

Все действующие лица системы делятся на три типа: простые, средние и сложные. В данном случае действующие лица (пользователь и администратор) являются сложными, так как они представляют личность, пользующуюся графическим интерфейсом (GUI). Подсчитанное количество действующих лиц каждого типа умножается на соответствующий весовой коэффициент, затем вычисляется общий весовой показатель А. Весовой коэффициент сложного действующего лица равен 3.

Действующих лиц в данном случае два, следовательно, общий весовой показатель А равен 6.

2.2.2 Определение весовых показателей вариантов использования

Все варианты использования делятся на три типа: простые, средние и сложные, в зависимости от количества транзакций в потоках событий (основных и альтернативных). Для простого типа варианта использования (3 или менее транзакций) весовой коэффициент принимается равным 5, для среднего (от 4 до 7 транзакций) – 10, для сложного (более 7 транзакций) – 15. Сложность вариантов использования представлена в приложении А в таблице 2.

Общий весовой показатель равен:

UCP = 11 \* 5 + 3 \* 10 = 85.

В результате получаем показатель UUCP:

UUCP = А + UCP = 6 + 85 = 91.

2.2.3 Определение технической сложности проекта

Техническая сложность проекта (TCF — technical complexity factor) вычисляется с учетом показателей технической сложности. Каждому показателю присваивается значение Ti в диапазоне от 0 до 5 (0 означает отсутствие значимости показателя для данного проекта, 5 — высокую значимость). Значение TCF вычисляется по следующей формуле:

TCF = 0.6 + (0.01 \* (∑ Ti Весi)).

Настройка показателей технической сложности проекта, выполненная в CASE-средстве Enterprise Architect 12, представлена на рисунке 2.1.

Вычислим значение TCF:

TCF = 0.6 + (0.01 \* 47.00) = 1.07.

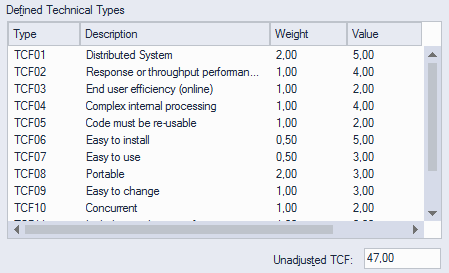


Рисунок 2.1 - Настройка показателей технической сложности проекта

2.2.4 Определение уровня квалификации разработчиков

Уровень квалификации разработчиков (EF — environmental factor) вычисляется с учетом показателей F1 – F8 по формуле:

EF = 1.4 + (-0.03 \* (∑ Fi Весi)).

Каждому показателю присваивается значение в диапазоне от 0 до 5. Для показателей F1 – F4 0 означает отсутствие, 3 – средний уровень, 5 – высокий уровень. Для показателя F5 0 означает отсутствие мотивации, 3 – средний уровень, 5 – высокий уровень мотивации. Для F6 0 означает высокую нестабильность требований, 3 – среднюю, 5 – стабильные требования. Для F7 0 означает отсутствие специалистов с частичной занятостью, 3 – средний уровень, 5 – все специалисты с частичной занятостью. Для показателя F8 0 означает простой язык программирования, 3 – среднюю сложность, 5 – высокую сложность. [4]

Настройка показателей уровня квалификации разработчиков, выполненная в CASE-средстве Enterprise Architect 12, представлена на рисунке 2.2.

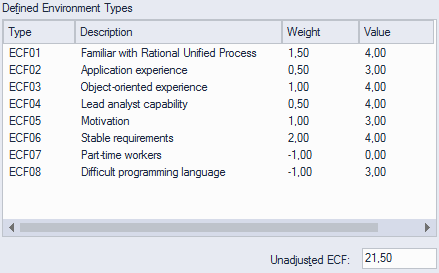


Рисунок 2.2 – Настройка показателей уровня квалификации разработчиков

Вычислим значение EF:

EF = 1.4 + (-0.03 \* 21.50) = 0.755.

В результате получаем окончательное значение UCP (use case points):

UCP = UUCP \* TCF \* EF = 91 \* 1.07 \* 0.755 = 73.

2.2.5 Оценка трудозатрат проекта

Для разрабатываемой системы примем 10 человеко-часов на одну UCP. Таким образом, общее количество человеко-часов на весь проект равно 10 \* 73 = 730. Если принять стоимость часа работы равной 40 денежным единицам, то стоимость проекта будет равна 730\*40 = 29200 денежным единицам.

Расчет трудозатрат проекта в CASE-средстве Enterprise Architect 12 представлен на рисунке 2.3.

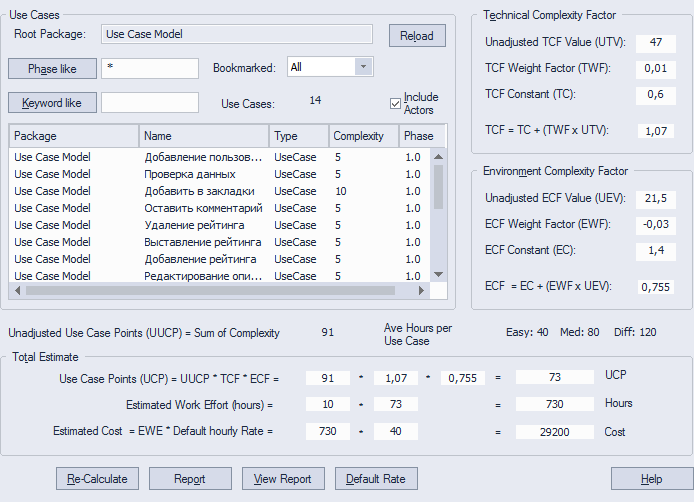


Рисунок 2.3 – Расчет трудозатрат проекта в CASE-средстве Enterprise Architect 12

2.3 Создание макета графического интерфейса пользователя

В процессе создания программного продукта одним из основных элементов, подлежащих разработке, является графический пользовательский интерфейс (GUI).

Исходя из основных сценариев вариантов использования создадим макеты экранных форм проектируемого программного обеспечения (приложение В).

Формы авторизация и регистрация будут содержать элементы button, каждый из которых отвечает следующим определенным задачам: авторизация, регистрация. Также будут представлены несколько кнопок для выполнения определенных действий: несколько RadioButton позволяют выбирать рейтинг для упорядочивания всех фильмов.

К рабочим формам приложения будут относиться:

* форма авторизации;
* форма регистрации;
* форма добавления фильма;
* форма просмотра фильма.

В форме любого приложения есть компоненты, обеспечивающие взаимодействие программы с пользователем. В разрабатываемом приложении имеются следующие компоненты:

* EditBox – для ввода произвольного текста;
* Button – кнопка;
* RadioButton – радио-кнопка;
* TextBox – для ввода короткого текста;
* Label – для предназначен для отображения текстовой информации;
* List – для вывода информации.

Основной упор при проектировании интерфейса приложения был сделан на простоту и понятность для обычного пользователя.

## 3. СОЗДАНИЕ ЛОГИЧЕСКОЙ МОДЕЛИ ПО

3.1 Разработка диаграммы классов

Диаграммой классов в терминологии UML называется диаграмма, на которой показан набор классов (и некоторых других сущностей, не имеющих явного отношения к проектированию БД), а также связей между этими классами. Кроме того, диаграмма классов может включать комментарии и ограничения.

Классом называется именованное описание совокупности объектов с общими атрибутами, операциями, связями и семантикой.

Диаграммы классов создаются при логическом моделировании ПС и служат для следующих целей:

* Для моделирования данных. Анализ предметной области позволяет выявить основные характерные для нее сущности и связи между ними. Это удобно моделируется с помощью диаграмм классов. Эти диаграммы являются основой для построения концептуальной схемы базы данных.
* Для представления архитектуры ПС. Можно выделить архитектурно значимые классы и показать их на диаграммах, описывающих архитектуру ПС.
* Для моделирования навигации экранов. На таких диаграммах показываются пограничные классы и их логическая взаимосвязь. Информационные поля моделируются как атрибуты классов, а управляющие кнопки – как операции и отношения.
* Для моделирования логики программных компонент (будет описано в последующих статьях).
* Для моделирования логики обработки данных. [5]

3.1.1 Описание модели

В процессе проектирования программного обеспечения будем придерживаться трехуровневой модели архитектуры разрабатываемого приложения. Первый уровень - уровень данных, по сути, является сервером, хранящим все данные приложения. Уровень данных содержит таблицы базы данных, файлы XML и другие средства хранения. Бизнес-уровень – второй уровень, работает как мост между уровнем данных и уровнем интерфейса. Уровень интерфейса – уровень, на котором пользователи взаимодействуют с приложением. Уровень интерфейса содержит общий код интерфейса пользователя, и конструкторов, используемых для представления информации пользователю.

Диаграмма взаимодействия уровней, выполненная в CASE-средстве Enterprise Architect 12, представлена в приложении Г (рисунок Г.1).

3.1.2 Проектирование уровня данных

Уровень данных в нашем случае представляет собой собственно базу данных.

Необходимый набор сущностей выявим на основании краткого анализа предметной области, проведенного в пункте 1.2. Для каждой сущности определим набор атрибутов. Атрибуты сущности – это свойства сущности, которые описывают характеристики сущности [13]. Они предназначены для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности [5].

В результате построена диаграмма классов уровня данных, выполненная в CASE-средстве Enterprise Architect 12, которая представлена в приложении Г (рисунок Г.2).

3.1.3 Проектирование уровня интерфейса и бизнес-логики

На диаграмме классов уровня интерфейса опишем классы экранных форм, эскизы которых были построены в пункте 2.3. К ним относятся такие классы, как класс авторизации, главное окно, добавления фильма, удаления фильма, редактирования фильма. Между ними всеми есть взаимосвязь. Через класс авторизации, вызываются все остальные классы – авторизации, главного окна, добавления, удаления, редактирования фильма.

На уровне бизнес-логики размещается описание классов, реализующих функции приложения – это класс администратора.

Как было описано в предыдущем пункте, уровень данных будет основополагающим уровнем, далее идет связь этого уровня с уровнем интерфейса через уровень бизнес-логики.

Диаграмма классов уровня интерфейса и бизнес-логики, выполненная в CASE-средстве Enterprise Architect 12, представлена в приложении Г (рисунок Г.3, Г.4).

3.2 Разработка диаграмм последовательностей

Диаграмма последовательности – это диаграмма, чаще всего, описывающая один сценарий приложения. На диаграмме последовательности, каждый участник представлен вместе со своей линией жизни (lifeline), это вертикальная линия под объектом, вертикально упорядочивающая сообщения на странице. Важно: все сообщения на диаграмме следует читать сверху вниз. Каждая линия жизни имеет полосу активности (прямоугольники), которая показывает интервал активности каждого участника при взаимодействии.

Диаграммы последовательностей можно разделить на области, называемые комбинированными фрагментами. У каждого комбинированного фрагмента есть оператор, один или более операндов и нуль или более сторожевых условий. [4]

В результате проектирования, были созданы следующие диаграммы последовательностей:

* Диаграмма последовательностей процесса авторизации, выполненная в CASE-средстве Enterprise Architect 12, представлена в приложении Д (рисунок Д.1);
* Диаграмма последовательностей процесса изменения данных, выполненная в CASE-средстве Enterprise Architect 12, представлена в приложении Д (рисунок Д.2);
* Диаграмма последовательностей процесса просмотра фильма, выставления рейтинга и написания комментария, выполненная в CASE-средстве Enterprise Architect 12, представлена в приложении Д (рисунок Д.3);

3.3 Разработка диаграмм деятельности

Для моделирования процесса выполнения операций в языке UML используются так называемые диаграммы деятельности. Диаграмма деятельности показывает последовательность действий, необходимых достижения цели. Действия (action) это элементарные шаги, которые не предполагают дальнейшую декомпозицию. Деятельность может содержать входящие и/или исходящие дуги деятельности, показывающие потоки управления и потоки данных. Если поток соединяет две деятельности, он является потоком управления. Если поток заканчивается объектом, он является потоком данных. Деятельность выполняется, только тогда, когда готовы все его «входы», после выполнения, деятельность передает управление и(или) данные на свои «выходы». Саму диаграмму деятельности принято располагать таким образом, чтобы действия следовали слева направо или сверху вниз. Чтобы указать, где именно находится процесс, используется абстрактная точка «маркер» (или «токен»). Визуально на диаграмме маркер не показывается, данное понятие вводится только для удобства описания динамического процесса. Деятельность (activity) представляет собой некоторую совокупность отдельных вычислений, выполняемых автоматом. При этом отдельные элементарные вычисления могут приводить к некоторому результату или действию (action). [7]

В результате проектирования, были созданы следующие диаграммы деятельности:

* Диаграмма деятельности работы с данными, выполненная в CASE-средстве Enterprise Architect 12, представлена в приложении Е (рисунок Е.1).
* Диаграмма деятельности процесса просмотра фильма, рейтингов, отзывов, выставления рейтинга и написания отзыва. Результат построения этой диаграммы в CASE-средстве Enterprise Architect 12, представлен в приложении E (рисунок E.2).

3.4 Разработка диаграмм состояний

Диаграммы состояний определяют все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате наступления некоторых событий.

На диаграмме состояний имеются два специальных состояния — начальное и конечное. Когда объект находится в каком-то конкретном состоянии, могут выполняться различные процессы. Процессы, происходящие, когда объект находится в определенном состоянии, называются действиями. [8]

Диаграммы состояний разрабатываемого программного обеспечения, выполненные в CASE-средстве Enterprise Architect 12, представлены в приложении. А именно:

* Диаграмма состояний процесса изменения данных, выполненная в CASE-сред-стве Enterprise Architect 12, представлена в приложении Ж (рисунок Ж.1).
* Диаграмма состояний просмотра данных. Результат построения этой диаграммы в CASE-средстве Enterprise Architect 12, представлен в приложении Ж (рисунок Ж.2).

## 4 СОЗДАНИЕ ФИЗИЧЕСКОЙ МОДЕЛИ ПО

Название «физическая модель» в терминологии языка UML определяют способы размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне. Физическая модель — это модель, создаваемая путем замены объектов моделирующими устройствами, которые имитируют определённые характеристики либо свойства этих объектов. При этом моделирующее устройство имеет ту же качественную природу, что и моделируемый объект. В UML физическая модель отражает компонентный состав проектируемой системы с точки зрения ее реализации на некоторой технической базе и вычислительных платформах конкретных производителей.

В языке UML для физического представления моделей систем используются так называемые диаграммы реализации, которые включают в себя две отдельные канонические диаграммы: диаграмму компонентов и диаграмму развертывания. [8]

4.1 Разработка диаграммы компонентов

Диаграммы компонентов моделируют физический уровень системы. Она показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п. Компоненты связываются через зависимости, когда соединяется требуемый интерфейс одного компонента с имеющимся интерфейсом другого компонента. Таким образом иллюстрируются отношения клиент-источник между двумя компонентами. Различают два способа связи интерфейса и компонента. Если компонент реализует некоторый интерфейс, то такой интерфейс называют экспортируемым или поддерживаемым, поскольку этот компонент предоставляет его в качестве сервиса другим компонентам. Если же компонент использует некоторый интерфейс, который реализуется другим компонентом, то такой интерфейс для первого компонента называется импортируемым. Особенность импортируемого интерфейса состоит в том, что на диаграмме компонентов это отношение изображается с помощью зависимости.

Информационная система составления рейтингов фильмов на сайте с фильмами состоит из исполняемого файла приложения и базы данных. Для компиляции исполняемого файла требуются файлы исходного кода, для которых в свою очередь требуются библиотеки языка C#. Диаграмма компонентов разрабатываемого программного обеспечения, выполненная в CASE-средстве Enterprise Architect 12, представлена в приложении З (рисунок З.1).

4.2 Разработка диаграммы развертывания

Диаграммы развертывания используются для моделирования статического вида системы с точки зрения развертывания. В основном под этим понимается моделирование топологии аппаратных средств, на которых выполняется система. По существу, диаграммы развёртывания - это просто диаграммы классов, сосредоточенные на системных узлах.

Диаграммы развертывания важны не только для визуализации, специфицирования и документирования встроенных, клиент-серверных и распределенных систем, но и для управления исполняемыми системами с использованием прямого и обратного проектирования.

Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения (runtime). При этом представляются только компоненты экземпляры программы, являющиеся исполнимыми файлами или динамическими библиотеками. Т.е. компоненты, которые не используются на этапе исполнения, на диаграмме развертывания не показываются. Так, компоненты с исходными текстами программ могут присутствовать только на диаграмме компонентов. На диаграмме развертывания они не указываются.

Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними. Узлы представляются как прямоугольные параллелепипеды с артефактами, расположенными в них, изображенными в виде прямоугольников. Узлы могут иметь подузлы, которые представляются как вложенные прямоугольные параллелепипеды. Один узел диаграммы развертывания может концептуально представлять множество физических узлов, таких как кластер серверов баз данных. Существует два типа узлов:

* узел устройства;
* узел среды выполнения.

В отличие от диаграмм логического представления, диаграмма развертывания является единой для системы в целом, поскольку должна всецело отражать особенности ее реализации. [8]

Диаграмма развертывания разрабатываемого программного обеспечения, выполненная в CASE-средстве Enterprise Architect 12, представлена в приложении И (рисунок И.1).

## 5 ОПИСАНИЕ РЕАЛИЗАЦИИ И ТЕСТИРОВАНИЯ ПО

5.1 Разработка и построение интерфейса

Формы авторизация и регистрация будут содержать элементы button, каждый из которых отвечает следующим определенным задачам: авторизация, регистрация. Также будут представлены несколько кнопок для выполнения определенных действий: несколько RadioButton позволяют выбирать рейтинг для упорядочивания всех фильмов.

Опишем работу приложения с базой данных. Для подключения к базе будем использовать SqlClient. Строка подключения представлена в листинге 1.

Листинг 1.

<add name="Film\_BD.Properties.Settings. FilmConnectionString"

connectionString="Data Source=(local);Initial Catalog=Film;Integrated Security=True"

providerName="System.Data.SqlClient" />

Для авторизации будет использоваться хранимая процедура, созданная на сервере. Использовать таблицы, представления и хранимые процедуры из сервера будем с помощью объекта *DataSet*.

Также было написано множество других процедур и функций. Самые важные из них:

* + хранимая процедура, позволяющая добавить пользователя в БД;
  + хранимая процедура, позволяющая определить рейтинг, который выставляется определенному фильму;
  + функции для добавления, удаления и редактирования данных.

Данная программа имеет в своем распоряжении множество информационных сообщений, которые направлены на более удобное использование. Также учтены возможные случаи некорректной работы программы, поэтому некоторое количество нештатных ситуаций сопровождается оповещениями.

Успешным завершением проверки служит вход пользователя в систему.

5.2 Результат тестирования ПО

Продуманная схема, доступное и простое выполнение приложения дают возможность тестировать каждый метод во время разработки. Реализация программы позволила обеспечить составление рейтингов фильмов для наиболее удобного поиска популярных для просмотра фильмов.

В процессе испытания СУБД были проверены основные моменты работы системы, к ним относятся:

* проверка авторизации;
* проверка на корректность добавления/ удаления / редактирования.

Проверка авторизации проводилась путём произведения попытки входа в систему с указанием логина и пароля доступа для работы с БД.

Успешным завершением проверки служит вход пользователя в систему.

Если не выбран фильм или рейтинг для удаления и нажать кнопку «Удалить», то будет выведено соответствующее сообщение об ошибке.

Если выставление рейтинга и добавление отзыва прошло успешно, то выводиться сообщение об успешном внесении данных, иначе – сообщение об ошибке.

Результаты разработки и построения интерфейса форм протестированы и представлены в таблице 5.1.

Таблица 5.1 – Результаты тестирования программы

|  |  |
| --- | --- |
| **Проверенная функция** | **Результат** |
| Авторизация пользователя | Успешно |
| Добавление данных | Успешно |
| Редактирование данных | Успешно |
| Удаление данных | Успешно |
| Регистрация | Успешно |

В результате тестирования были выявлены и исправлены ошибки и недочеты в работе ПО. Интерфейс является простым и понятным для работы пользователя с ПО.

## ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы были освоены технологии разработки программного продукта, а также спроектирована система «Составление рейтингов фильмов на сайте с фильмами». Это приложение представляет собой систему составления рейтингов, а также приложение, позволяющее пользователю взаимодействовать с базой данных.

В ходе выполнения были изучены основные принципы проектирования, а также созданы функциональная, логическая, физическая модели ПО и следующие диаграммы:

* диаграммы вариантов использования;
* диаграммы классов;
* диаграммы последовательностей;
* диаграммы компонентов;
* диаграммы развертывания.

Было использовано CASE-средство Enterprise Architect 12, в котором содержаться все инструменты для проектирования и отлично адаптировано под пользователя.

Разработанное приложение удовлетворяет всем требованиям, предназначенным для комфортной работы и позволяет без проблем хранить и извлекать требуемые данные.

В ходе выполнения курсовой работы были выявлены критерии и атрибуты, позволяющие упростить работу с информацией для работников в сфере кино, телевидения и других областей деятельности, позволяя систематизировать всю необходимую информацию.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Официальный сайт справочной информации "Wikipedia" [Электронный ресурс]. Режим доступа https://ru.wikipedia.org/wiki/UML. Дата доступа – 01.04.2016 г.

2. Электронный форум "ИсходникиRU", тема раздела "МЕТОДИКИ ОЦЕНКИ ТРУДОЕМКОСТИ РАЗРАБОТКИ ПО" [Электронный ресурс]. Режим доступа http://forum.sources.ru/index.php?showtopic=8820. Дата доступа – 01.04.2016 г.

3. Электронный ресурс "INFORMICUS", статься "Диаграммы классов UML. Логическое моделирование" [Электронный ресурс]. Режим доступа http://www.informicus.ru/default. aspx?SECTION=6&id=73& subdivisionid=3. Дата доступа – 25.02.2016 г.

4. Электронный ресурс "Интуит", статься "Нотация и семантика языка UML: Информация" [Электронный ресурс]. Режим доступа http://www.intuit.ru/studies/courses/1007/229/lecture/5960. Дата доступа – 25. 02.2016 г.

5. Электронная библиотека "IT-GOST.RU", статься "Теория и практика UML. Диаграмма деятельности" [Электронный ресурс]. Режим доступа http://it-gost.ru/articles/view\_articles/96. Дата доступа – 16.03.2016 г.

6. Электронная библиотека "IT-GOST.RU", статься "Теория и практика UML. Диаграмма деятельности" [Электронный ресурс]. Режим доступа [http://it-gost.ru/articles/view\_articles/97. Дата доступа – 01.04.2016](http://it-gost.ru/articles/view_articles/97.%20Дата%20доступа%20–%2001.04.2016) г.

7. Электронный ресурс "Интуит", статься "Нотация и семантика языка UML: Информация" [Электронный ресурс]. Режим доступа [http://www.intuit.ru/studies/courses/32/32/lecture/1024. Дата доступа – 01.04.2016](http://www.intuit.ru/studies/courses/32/32/lecture/1024.%20Дата%20доступа%20–%2001.04.2016) г.

8. Официальный сайт справочной информации "Wikipedia" [Электронный ресурс]. Режим доступа https://ru.wikipedia.org/wiki/Диаграмма\_развертывания. Дата доступа – 01.04.2016 г.

## ПРИЛОЖЕНИЕ А

**Варианты использования для разрабатываемой информационной системы**

Таблица А.1 – Варианты использования и их сценарии

| **№** | **Вариант использования** | **Базовый сценарий:** |
| --- | --- | --- |
| 1. | Авторизация | 1. Ввести логин, пароль (при заполнении имени сервера)  2. Нажать на кнопку “Вход”  3. Дождаться ответа  4. Перейти на форму администратора или пользователя программы |
| 2. | Редактирование БД фильмов | 1. Выбрать нужную таблицу  2. Дождаться результата отображения |
| 3. | Добавление фильма | 1. Выбрать таблицу в соответствующем окне  2. Заполнить данные  3. Нажать кнопку «Добавить» |
| 4. | Редактирование описания | 1. Выбрать таблицу в соответствующем окне  2. Заполнить данные  3. Нажать кнопку «Редактировать» |
| 5. | Удаление фильма | 1. Выбрать таблицу в соответствующем окне  2. Выбрать удаляемый фильм  3. Нажать на кнопку «Удалить» |
| 6. | Добавление рейтинга | 1. Выбрать таблицу в соответствующем окне  2. Ввести рейтинг  3. Нажать кнопку «Добавить» |
| 7. | Удаление рейтинга | 1. Выбрать таблицу в соответствующем окне  2. Выбрать удаляемый рейтинг  3. Нажать на кнопку «Удалить» |
| 8. | Регистрация | 1. Ввести логин, пароль, имя, фамилию  2. Нажать на кнопку “Зарегистрироваться”  3. Дождаться ответа  4. Перейти на форму пользователя программы |
| 9. | Добавление пользователя в БД | 1. Зарегистрироваться  2. Дождаться результата |
| 10. | Просмотр фильма | 1. Перейти на форму пользователя  2. Дождаться результата |
| 11. | Добавление в закладки | 1. Перейти на форму пользователя  2. Выбрать фильм  3. Нажать кнопку «Добавить в закладки»  4. Дождаться результата |
| 12. | Выставление рейтинга | 1. Перейти на форму пользователя  2. Выставить рейтинг |
| 13. | Оставить комментарий | 1. Перейти на форму пользователя  2. Оставить комментарий |
| 14. | Проверка данных | 1. Ввести данных  2. Дождаться ответа |

Таблица А.2 – Сложность вариантов использования

| **Вариант использования** | **Тип** | **Весовой коэффициент** |
| --- | --- | --- |
| Авторизация | средний | 10 |
| Редактирование БД фильмов | простой | 5 |
| Добавление фильма | простой | 5 |
| Редактирование описания | простой | 5 |
| Удаление фильма | простой | 5 |
| Добавление рейтинга | простой | 5 |
| Удаление рейтинга | простой | 5 |
| Регистрация | средний | 10 |
| Добавление пользователя в БД | простой | 5 |
| Просмотр фильма | простой | 5 |
| Добавление в закладки | средний | 10 |
| Выставление рейтинга | простой | 5 |
| Оставить комментарий | простой | 5 |
| Проверка данных | простой | 5 |

## ПРИЛОЖЕНИЕ Б

**Диаграмма вариантов использования разрабатываемого программного обеспечения**

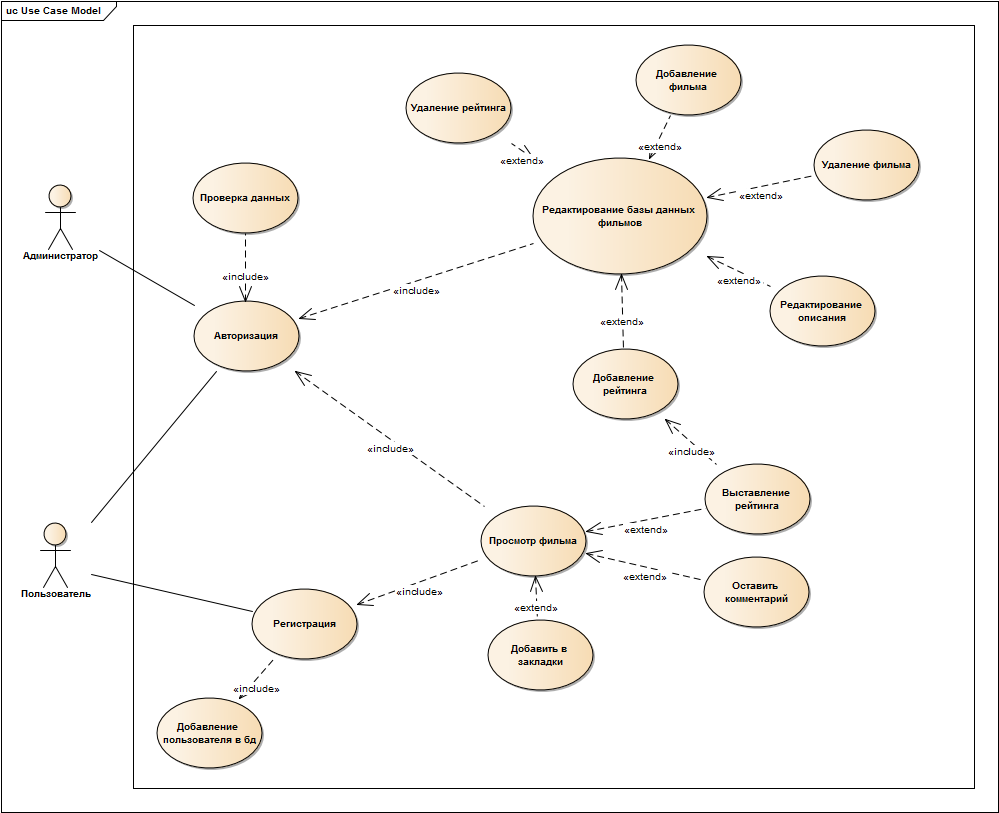
****

Рисунок Б.1 – Диаграмма вариантов использования

## ПРИЛОЖЕНИЕ В

**Макеты оконного интерфейса**

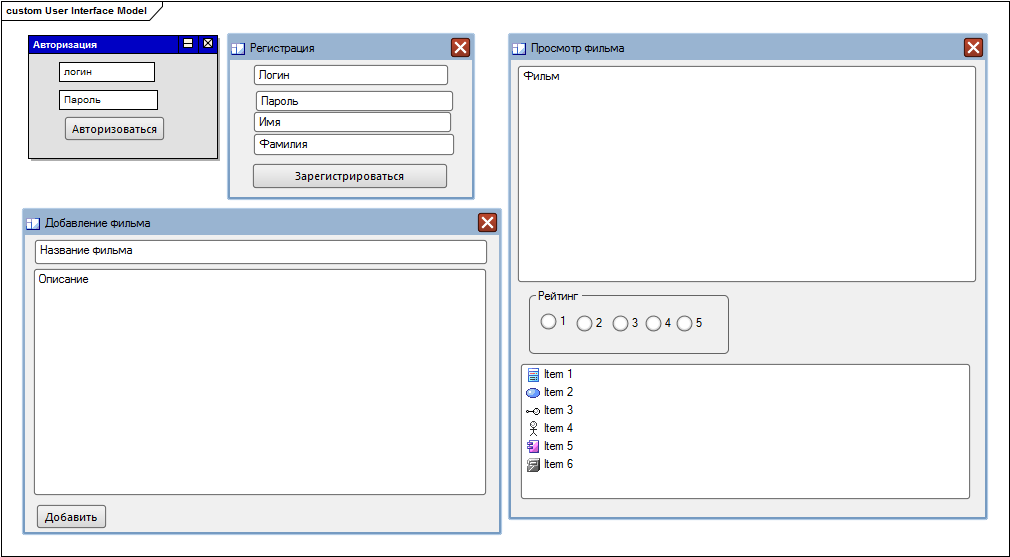
****

Рисунок В.1 – Макет интерфейса приложения

## ПРИЛОЖЕНИЕ Г

**Диаграмма классов разрабатываемого программного обеспечения**

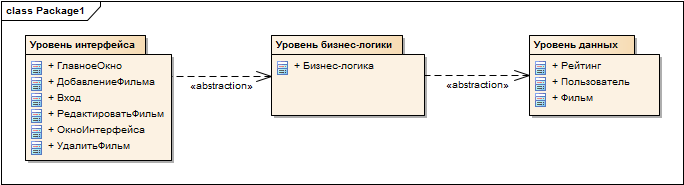


Рисунок Г.1 – Диаграмма взаимодействия уровней

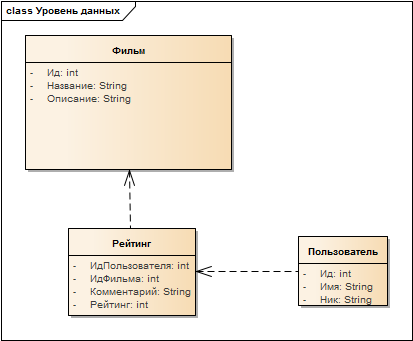


Рисунок Г.2 – Диаграмма классов уровня данных

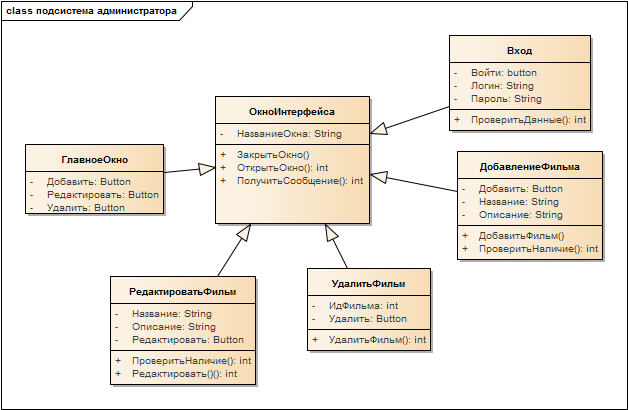


Рисунок Г.3 – Диаграмма классов уровня интерфейса

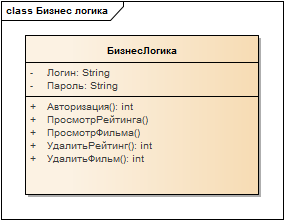


Рисунок Г.4 – Диаграмма классов уровня бизнес-логики

## ПРИЛОЖЕНИЕ Д

**Диаграммы последовательностей разрабатываемого программного обеспечения**

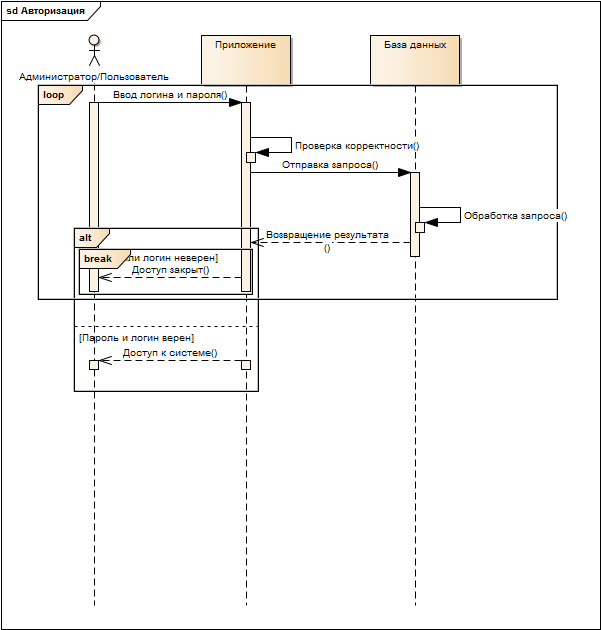


Рисунок Д.1 – Диаграмма последовательностей авторизации

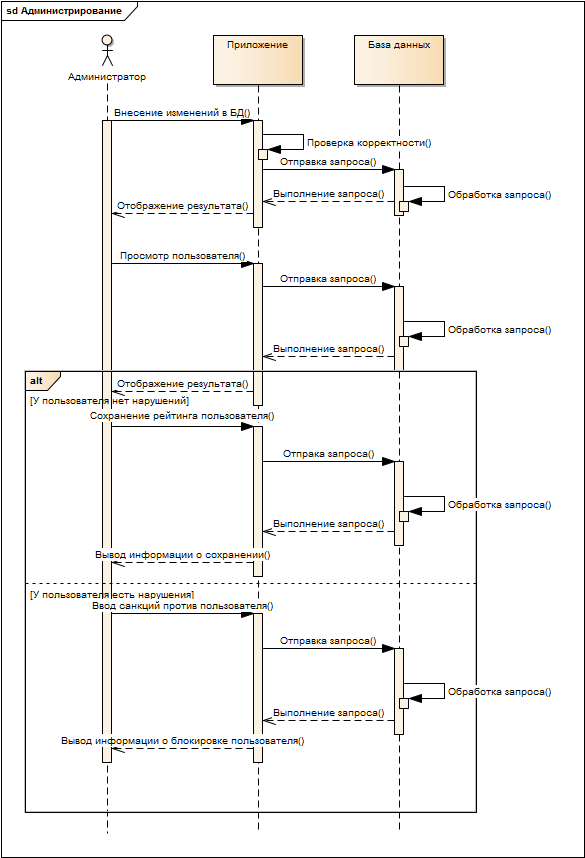


Рисунок Д.2 – Диаграмма последовательностей процесса изменения данных

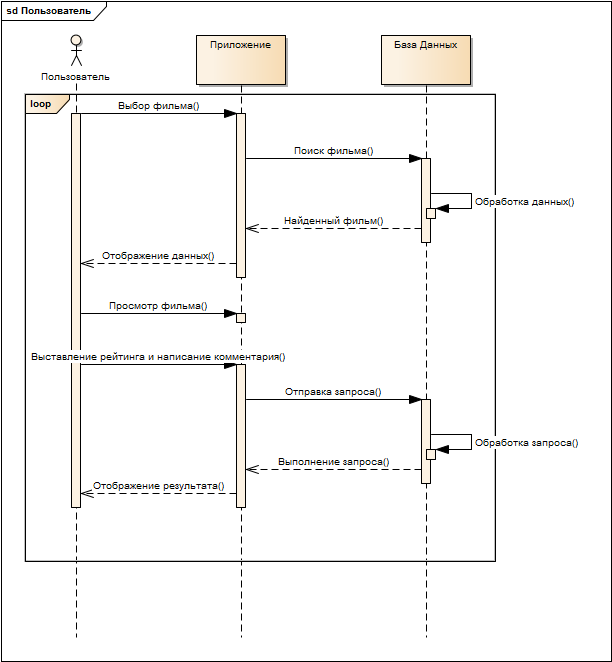


Рисунок Д.3 – Диаграмма последовательностей процесса просмотра фильма, выставления рейтинга и написания комментария

## ПРИЛОЖЕНИЕ E

**Диаграммы деятельности разрабатываемого программного обеспечения**

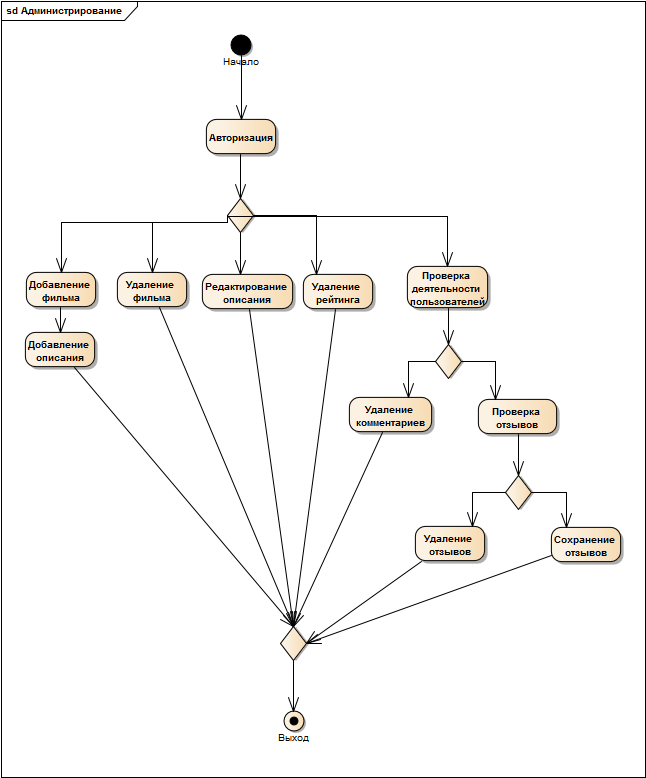


Рисунок E.1 – Диаграмма деятельности работы с данными

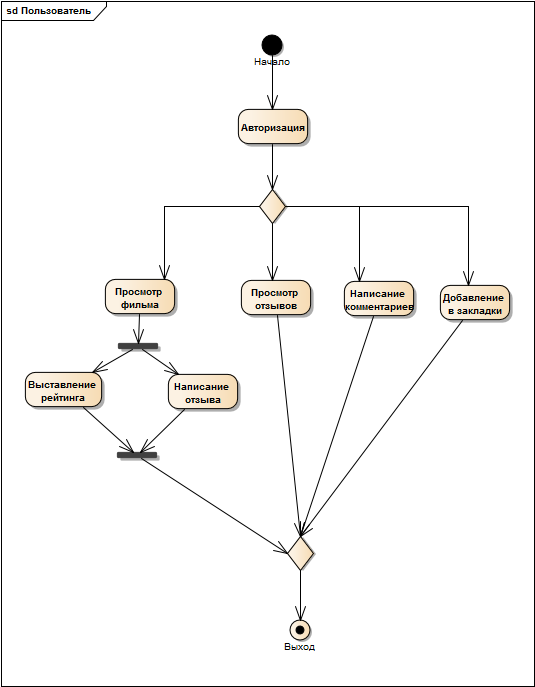


Рисунок Е.2 – Диаграмма деятельности процесса просмотра фильма, рейтингов, отзывов, выставления рейтинга и написания отзыва

## ПРИЛОЖЕНИЕ Ж

**Диаграммы состояний разрабатываемого программного обеспечения**

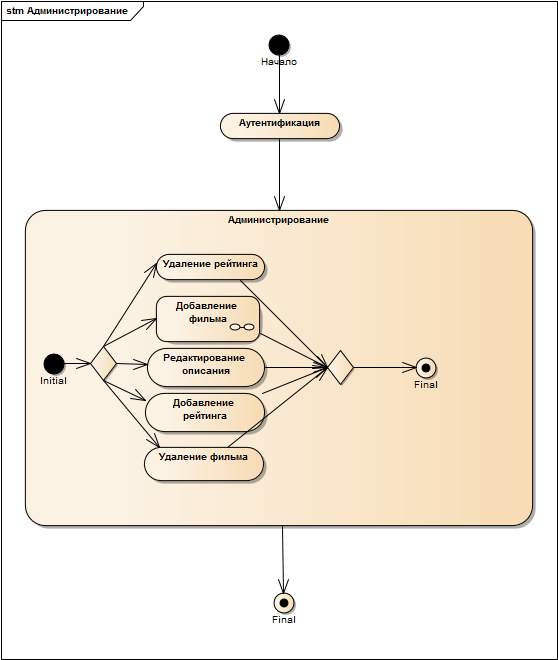


Рисунок Ж.1 – Диаграмма состояний изменения данных

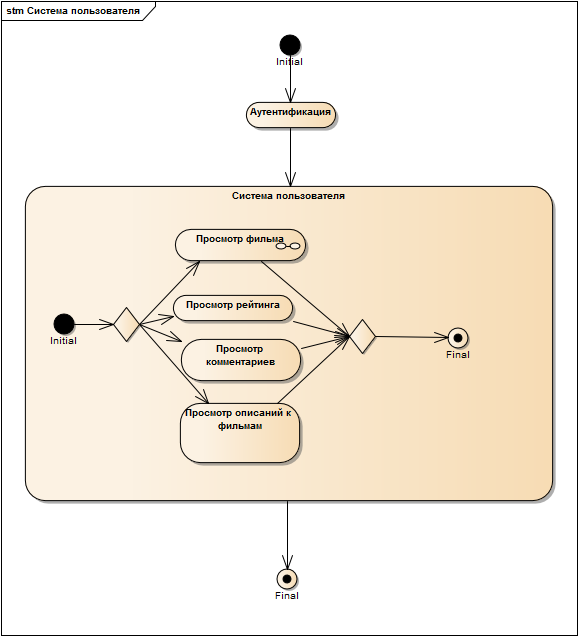


Рисунок Ж.2 – Диаграмма состояний получения данных

## ПРИЛОЖЕНИЕ З

**Диаграмма компонентов разрабатываемого программного обеспечения**

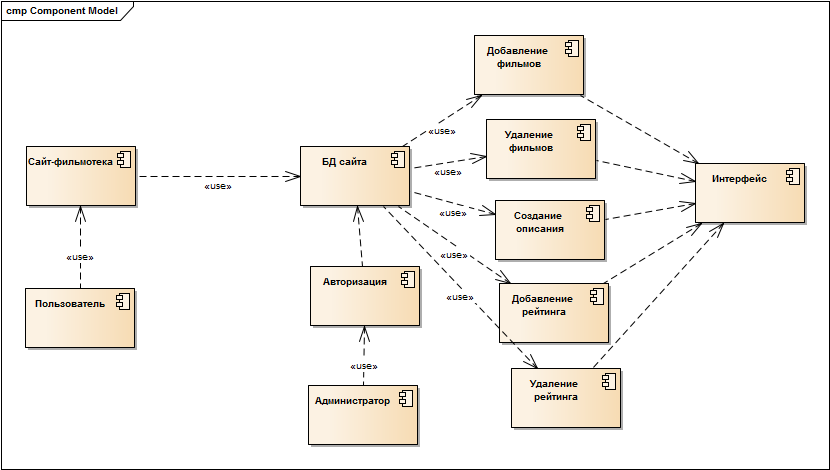
****

Рисунок З.1 – Диаграмма компонентов разрабатываемого программного обеспечения

## ПРИЛОЖЕНИЕ И

**Диаграмма развертывания разрабатываемого программного обеспечения**

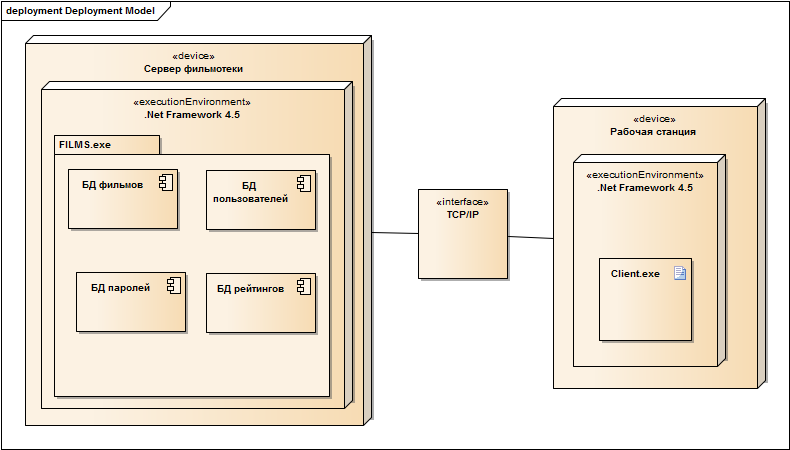


Рисунок И.1 – Диаграмма развертывания разрабатываемого программного обеспечения