# **ВВЕДЕНИЕ**

История существования кинематографа насчитывает более ста лет. За эти годы кинематограф стал одним из наиболее популярных и известных видов искусства, но, тем не менее, кино – на рынке всего лишь продукт, требующий, как и все продукты специализированного и научного подхода по популяризации и продвижению в массы. Как и все продукты, он требует грамотного продвижения, тем более учитывая роль рекламной компании в его кассовых сборах и дальнейшей возможности быть узнаваемым.

Использование информационных технологий в сервисе оценки и подбора кинофильмов является неотъемлемой частью разувающегося информационного пространства. Спектр их применения широк и варьируется от блогов отдельно взятых людей до полноценного сайта со своей системой оценки каждого из фильмов.

Целью данного курсового проекта является создание удобного, полезного и функционального сайта для оценки кинофильмов с понятным пользователю интерфейсом.

Для достижения поставленной цели необходимо решить следующие задачи:

* изучить предметную область сервиса оценки кинофильмов;
* выполнить функциональное моделирование процесса оценки кинофильмов;
* разработать модели представления сервиса оценки кинофильмов на основе UML;
* проанализировать и создать логическую и физическую модель представления данных;
* создать базу данных, хранящую информацию о кинофильмах, о пользователях и объектах, необходимых для выставления оценки кинофильму;
* реализовать возможности регистрации и авторизации;
* разработать программное обеспечение;
* протестировать программный продукт.

После решения поставленных задач, приложение, разработанное в данном курсовом проекте, позволит оптимизировать процесс автоматизации сервиса оценки фильмов представляется возможность удобной организации рейтинга популярности представленных кинофильмов, а также помогает обеспечить фильмы релевантными отзывами.

**1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

Предметная область данной курсовой работы – сервис оценки кинофильмов.

Задачей курсового проекта является создание приложения, которое позволило бы автоматизировать процессы оценки кинофильмов.

При написании программного кода необходимо использовать язык Java. Java — объектно-ориентированный язык программирования, разрабатываемый компанией Sun Microsystems с 1991 года и официально выпущенный 23 мая 1995 года. Изначально новый язык программирования назывался Oak (James Gosling) и разрабатывался для бытовой электроники, но впоследствии был переименован в Java и стал использоваться для написания апплетов, приложений и серверного программного обеспечения [3].

У языка Java есть много преимуществ перед другими языками программирования, что позволяет решать с его помощью практически любые задачи. Java - это объектно-ориентированный язык. Это позволяет создавать модульные программы, исходный код которых может использоваться многократно.

Одним из основных преимуществ языка Java является возможность переноса программ из одной системы в другую. Поскольку программы на Java не зависят от платформы как на уровне исходного кода, так и на двоичном уровне, их можно запускать в различных системах, что особенно важно для программ, предназначенных для World Wide Web [4].

Широкие возможности Java, простота применения, независимость от платформы и встроенные функции защиты делают этот язык программирования одним из лучших для создания приложений для Internet.

Работа приложения основывается на том, что клиентская часть реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него, в то время как серверная часть получает запрос от клиента, выполняет вычисления, после чего формирует ответ и отправляет её клиенту. Выполнение всех описанных сетевых функций приложения без проблем реализуется с помощью Java, так как сетевая работа является одной из сильных сторон данного языка программирования.

В Java имеется изящный и вместе с тем сложный механизм реализации взаимодействия потоков, он позволяет сосредоточиться непосредственно на задаче - без раздумий о том, как бы организовать взаимодействие потоков самому [5].

Многочисленные библиотеки классов в Java значительно упрощают работу программиста и помогают сделать красивый и понятный код. Java предоставляет программисту богатый набор классов объектов для ясного абстрагирования многих системных функций, используемых при работе с окнами, сетью и для ввода-вывода. Ключевая черта этих классов заключается в том, что они обеспечивают создание независимых от используемой платформы абстракций для широкого спектра системных интерфейсов [3].

При выполнении предыдущих курсовых работ мы использовали файлы для хранения данных, помещая в них данные, а затем извлекая для последующей работы. Данный способ накладывает дополнительные ограничения на работу программы, значительно увеличивает количество и сложность кода. Вместе с этим возрастает вероятность совершения ошибки при написании программы.

Для решения вышеназванный проблем придумано отличное средство – база данных. База данных - набор сведений, хранящихся некоторым упорядоченным способом. Базы данных сами заботятся о безопасности информации и её сортировке и позволяют извлекать и размещать информацию при помощи небольшого количества специализированного кода. При использовании базы данных код получается более компактным, понятным и легче тестируется и отлаживается.

В противоположность файловой организации хранения информации, использование баз данных дает неоспоримые преимущества. Например, легко организовывать поиск, сортировать записи по дате и времени, проводить различные отборы записей, так как в БД предусмотрена эффективная организация хранения информации, минимизирующая время доступа и поиска. Можно быстро найти среди многих тысяч записей одну определенную (по заданному идентификатору).

Базы данных лишены большого недостатка файлов: с ними нет проблем с совместным доступом к данным. Сценарий, который изменяет файл в процессе своей работы, могут запустить одновременно два человека, и, если не принять мер по блокировке файла, то могут возникнуть проблемы. С базами данных таких проблем не существует, потому что предусмотрено решение проблем совместного доступа на низком уровне с максимальной эффективностью.

Система управления базами данных - это совокупность языковых и программных средств, которая осуществляет доступ к данным, позволяет их создавать, менять и удалять, обеспечивает безопасность данных и т.д. В общем СУБД - это система, позволяющая создавать базы данных и манипулировать сведениями из них. А осуществляет этот доступ к данным СУБД посредством специального языка - SQL. SQL (Stuctured Query Language ) - язык структурированных запросов, основной задачей которого является предоставление простого способа считывания и записи информации в базу данных. SQL включает много разных типов операторов, разработанных для взаимодействия с базами данных.

Для реализации хранения, обработки и дальнейшего использования информации в данном приложении используется СУБД MySQL.

MySQL – это реляционная система управления базами данных. То есть данные в ее базах хранятся в виде логически связанных между собой таблиц, доступ к которым осуществляется с помощью языка запросов SQL. MySQL – свободно распространяемая система, т.е. платить за ее применение не нужно. Кроме того, это достаточно быстрая, надежная и, главное, простая в использовании СУБД, вполне подходящая для не слишком глобальных проектов [6].

MySQL Workbench представляет собой единый визуальный инструмент для разработчиков баз данных, разработчиков и администраторов баз данных. MySQL Workbench обеспечивает моделирование данных, разработку SQL и комплексные средства администрирования для настройки сервера, управления пользователями, резервного копирования [7].

 Таким образом, базы данных имеют достаточное количество явно неоспоримых преимуществ, используемых для решения задач данного курсового проекта.

**2 ОПИСАНИЕ ПРОЦЕССА ВЫСТАВЛЕНИЯ ОЦЕНКИ КИНОФИЛЬМУ**

Чтобы получить полное представление о том, как можно автоматизировать процесс оценки кинофильмов, создана функциональная модель.

Ключевой процесс в данной теме – оценивание кинофильма, ради чего и будет писаться программное обеспечение. Для создания же функциональной модели был выбран стандарт IDEF0. IDEF0 — методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность[3].

На контекстной диаграмме верхнего уровня (рисунок 2.1) представлена функциональная модель «Выставление оценки кинофильму», а также определены потоки входных и выходных данных, механизмы ограничения и управления данными.

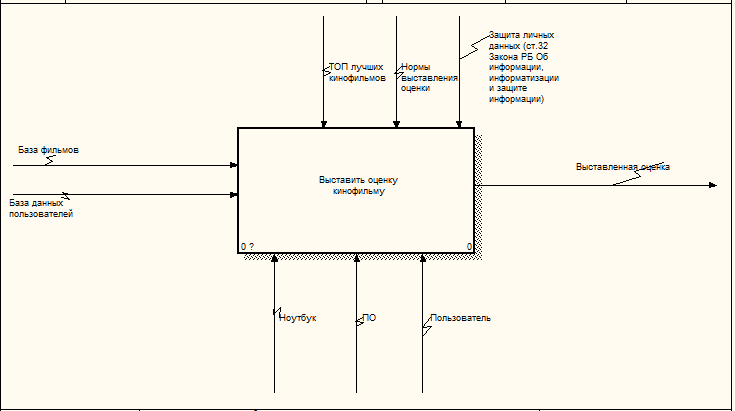


Рисунок 2.1 – Контекстная диаграмма верхнего уровня «Выставить оценку кинофильму»

Входной поток включает в себя базу данных фильмов, а так же базу данных пользователей. После соответствующей обработки в выводном потоке имеем выставленную оценку. В роли управляющих механизмов выступают ТОП лучших кинофильмов, нормы выставления оценок и закон РБ «Об информации, информатизации и защите информации», ст.32. Механизмами являются ноутбук, пользователь и программное обеспечение.

На рисунке 2.2 отображена декомпозиция контекстной диаграммы, состоящая из четырёх блоков: «Найти кинофильм», «Посмотреть кинофильм», «Зайти в аккаунт на сервисе» и «Выставить оценку».

Первый компонент данной декомпозиции («Найти кинофильм») подразумевает нахождение кинофильма с помощью ТОПа лучших кинофильмов.

Декомпозиция данного блока отображена на рисунке 2.3. Она представлена тремя компонентами: «Выбрать жанр», «Выбрать режиссера», «Определиться с фильмом».

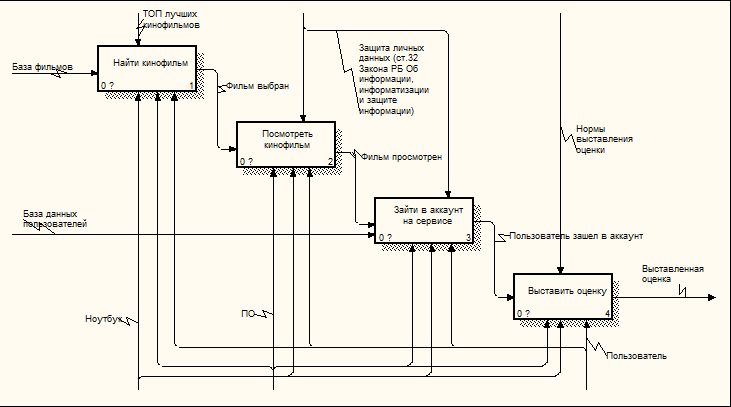


Рисунок 2.2 – Декомпозиция контекстной диаграммы верхнего уровня «Выставить оценку кинофильму»

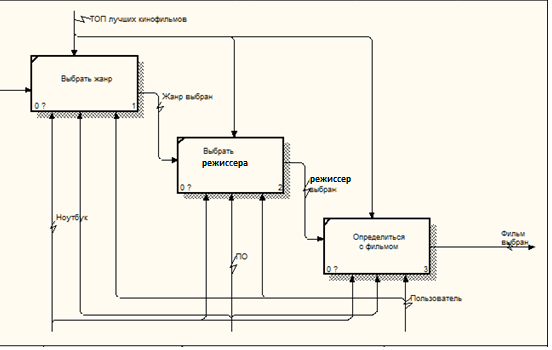


Рисунок 2.3 – Декомпозиция блока «Найти кинофильм»

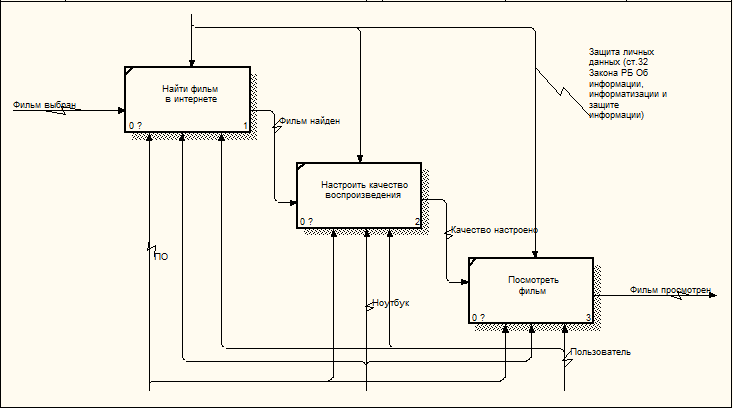


Рисунок 2.4 – Декомпозиция блока «Посмотреть кинофильм»

Второй компонент декомпозиции контекстной диаграммы («Посмотреть кинофильм») включает в себя поиск и просмотр выбранного фильма.

Декомпозиция этого блока отображена на рисунке 2.4. Она представлена тремя компонентами: «Найти фильм в интернете», «Настроить качество воспроизведения» и «Посмотреть фильм».

Третий компонент декомпозиции контекстной диаграммы («Зайти в аккаунт на сервисе») подразумевает собой вход в уже зарегистрированный аккаунт на сервисе. Декомпозиция данного блока отображена на рисунке 2.5. Она представлена тремя компонентами: «Зайти на сайт сервиса», «Ввести логин и пароль от аккаунта» и «Войти в свой аккаунт».

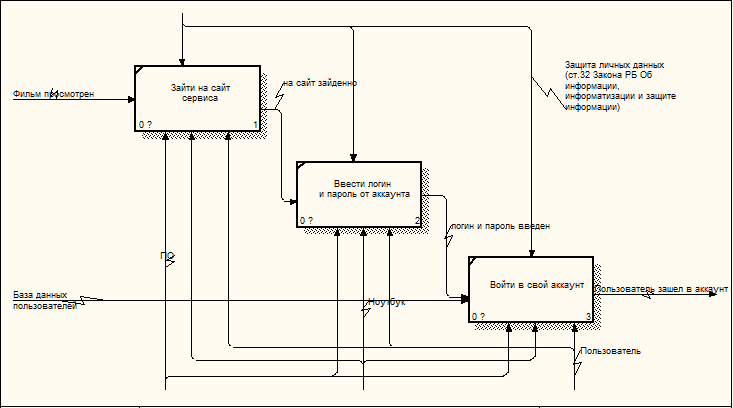


Рисунок 2.5 – Декомпозиция блока «Зайти в аккаунт на сервисе»

Четвертый компонент декомпозиции контекстной диаграммы («Выставить оценку») подразумевает собой выставление оценки на сайте сервиса оценки кинофильмов. Декомпозиция данного блока отображена на рисунке 2.6. Она представлена тремя компонентами: «Найти просмотренный фильм», «Выставить заслуженную оценку» и «Убедиться в верно выставленной оценке».

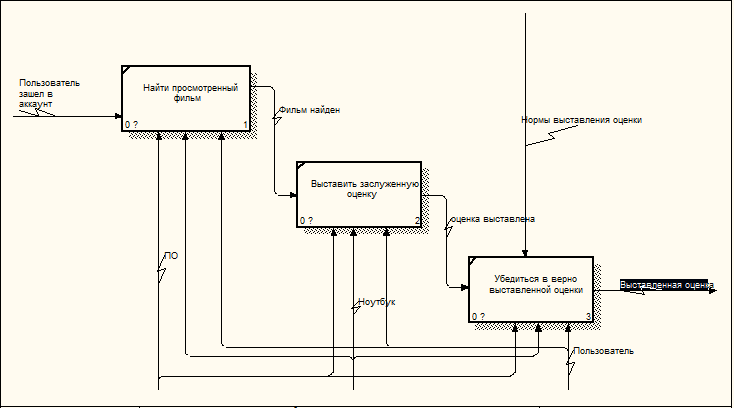


Рисунок 2.5 – Декомпозиция блока «Зайти в аккаунт на сервисе»

Данный продукт открывает некоторые перспективы для упрощения поиска фильма. Пользователи смогут без труда получить информацию о самых интересных фильмах по мнению других пользователей, которые уже посмотрели выбранный фильм.

**3 СПЕЦИФИКАЦИЯ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ СИСТЕМЫ**

Между элементами диаграммы прецедентов могут быть разные отношения, которые описывают взаимодействие экземпляров одного из участников и вариантов использования с примерами других участников и вариантов. Один актер может взаимодействовать с несколькими видами использования. В этом случае этот актер ссылается на несколько сервисов этой системы. В свою очередь, один случай использования может взаимодействовать с несколькими участниками, обеспечивая для всех их собственное обслуживание.

В то же время два варианта использования, определенные в одной и той же моделируемой системе, могут также взаимодействовать друг с другом, но характер этого взаимодействия будет отличаться от взаимодействия с участниками. Однако в обоих случаях методы взаимодействия элементов модели принимают обмен сигналами или сообщениями, которые инициируют реализацию функционального поведения моделируемой системы.

На рисунке 3.1 представлена разработанная диаграмма вариантов использования (Use Case Diagram). На данной диаграмме изображаются отношения между Actor и вариантами использования проектируемой системы. Основные субъекты выбранной диаграммы – «Пользователь» и «Администратор».

Функции приложения, доступные пользователю обеим ролям представлены на рисунке 3.1.

Разработанная для приложения диаграмма вариантов использования задействует два типа отношений между вариантами использования: расширение и обобщение. Отношение «extend» определяет отношение базового варианта использования к другому варианту использования, функциональное поведение которого не всегда используется базовым случаем, но только при выполнении дополнительных условий. Отношение обобщения между вариантами использования применяется, когда необходимо отметить, что случаи использования детьми имеют все особенности поведения родительских вариантов. В этом случае случаи использования дочерних вариантов использования задействованы во всех родительских вариантах. В свою очередь, варианты ребенка могут быть наделены новыми поведением, которые отсутствуют в родительских вариантах использования, а также уточняют или модифицируют свойства поведения, унаследованные от них.

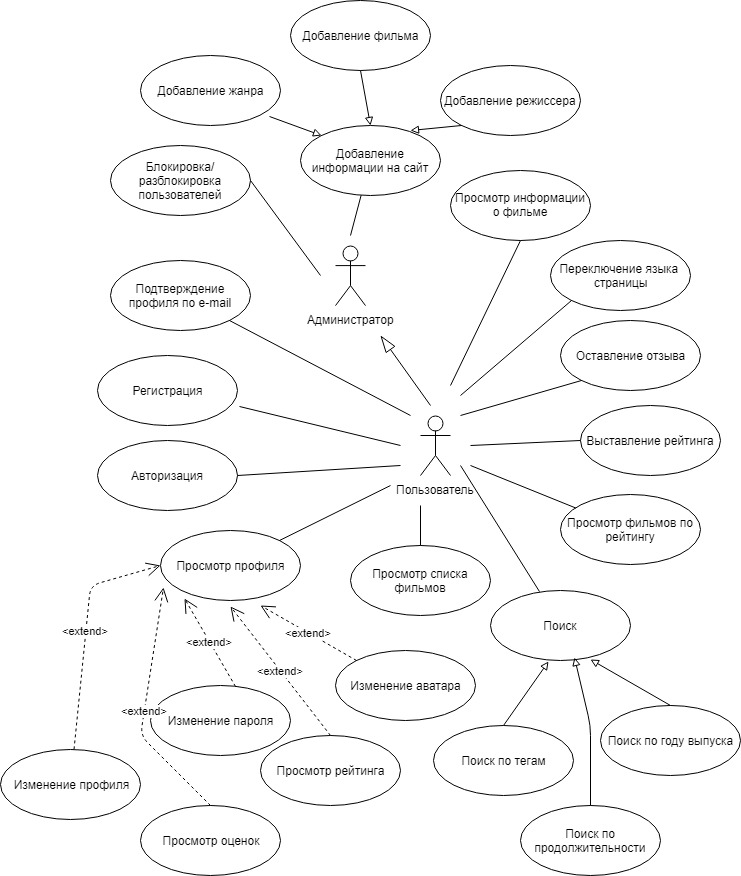


Рисунок 3.1 – Диаграмма вариантов использования

**4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЁ ОПИСАНИЕ**

Для разработки системы оценки кинофильмов потребовалось выделить и создать шесть сущностей: фильм, жанр, рейтинг, отзыв, режиссер и пользователь.

Информационная модель в графическом виде представлена ниже на рисунке 4.1.

Сущность «Пользователь» предназначена для хранения информации о пользователях системы. Данные из этой таблицы используются при авторизации в приложении. Сущность хранит в себе следующие атрибуты:

1. iduser – хранит уникальный идентификационный номер пользователя приложения;
2. login – хранит имя пользователя, которое указывается при входе в приложение;
3. password – хранит пароль, используемы для авторизации в системе;
4. email – хранит адрес почты пользователя;
5. real\_name – хранит настоящее имя пользователя, отображающееся на сайте;
6. date\_of\_registration – дата регистрации на сайте;
7. date\_of\_birth – дата рождения пользователя;
8. status – статус пользователя, например «заблокирован»;
9. avatar – хранит ссылку на аватар пользователя;
10. level\_points – количество очков уровня пользователя;
11. role – хранит роль пользователя. По умолчанию роль соответствует «пользователю».

Сущность «Рейтинг» предназначена для хранения информации о рейтинге фильма. Сущность хранит в себе следующие атрибуты:

1. idrating– хранит уникальный идентификационный номер записи;
2. iduser – хранит уникальный идентификационный номер пользователя, которому принадлежит оценка;
3. idfilm – хранит уникальный идентификационный номер фильма, которому предназначена оценка;
4. is\_seen – хранит информацию о том, просмотрен ли фильм;
5. rating\_amount – хранит количественное выражение оценки;

Сущность «Фильм» предназначена для хранения информации о фильме. Сущность хранит в себе следующие атрибуты:

1. idfilm– хранит уникальный идентификационный номер записи;
2. name – хранит название фильма;
3. release\_year – хранит год выпуска фильма;
4. duration – хранит длительность в минутах;
5. poster – хранит ссылку на постер;
6. iddirector – хранит уникальный идентификационный номер режиссера данного фильма;
7. description – хранит описание фильма;

Сущность «Отзыв» предназначена для хранения информации об отзыве. Сущность хранит в себе следующие атрибуты:

1. idreview– хранит уникальный идентификационный номер записи;
2. iduser - хранит уникальный идентификационный номер пользователя, которому принадлежит отзыв;
3. idfilm - хранит уникальный идентификационный номер фильма;
4. text – хранит текст отзыва;
5. date – хранит дату, когда был написан отзыв;

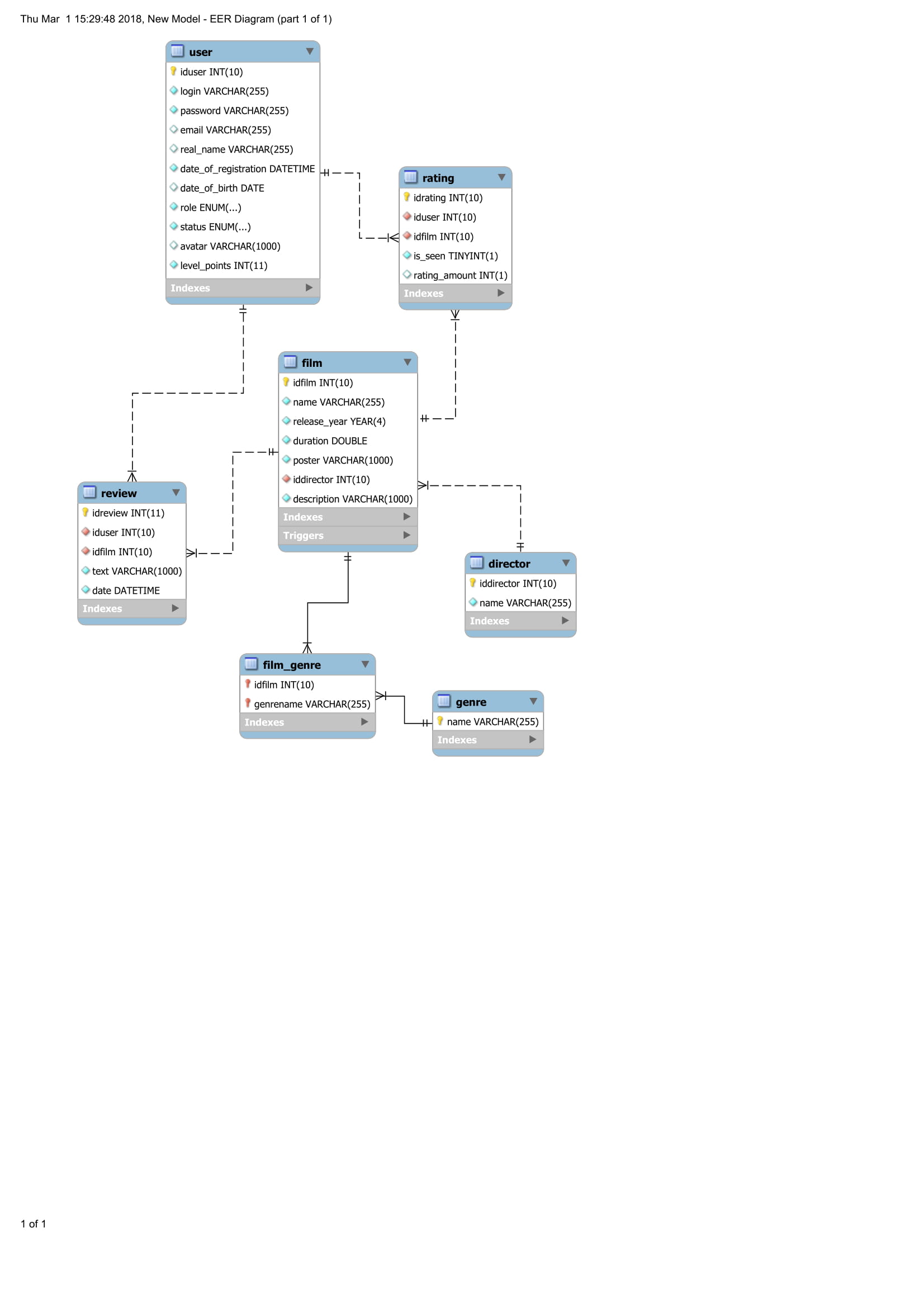


Рисунок 4.1 – Информационная модель системы

Сущность «Жанр» предназначена для хранения информации о жанре. Сущность хранит в себе следующие атрибуты:

1. name– хранит название жанра;

Сущность «Режиссер» предназначена для хранения информации о режиссере. Сущность хранит в себе следующие атрибуты:

1. iddirector– хранит уникальный идентификационный номер записи;
2. name – хранит имя режиссера;

**5 ОБОСНОВАНИЕ ВЫБОРА КОМПОНЕНТОВ И ТЕХНОЛОГИЙ**

Для более подробного изучения и рассмотрения системы программы необходимо рассмотреть такие модели как:

* диаграмма развертывания;
* диаграмма последовательностей;
* диаграмма компонентов.

Система оценки кинофильмов реализована на операционной системе Windows 10 . На диаграмме развертывания (рисунок 5.1), показана конфигурация обрабатывающих узлов, на которых выполняется система, и компонентов, размещенных в этих узлах.

****

Рисунок 5.1 – Диаграмма развёртывания системы оценки кинофильмов

Диаграмма последовательности  — [диаграмма](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определённого объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актёров (действующих лиц) в рамках какого-либо определённого прецедента (отправка запросов и получение ответов)[5]. Представление диаграммы показано на рисунке 5.2.

Рисунок 5.2 – Диаграмма последовательностей входа в систему оценки кинофильмов

Диаграмма компонентов – диаграмма физического уровня, которая служит для представления программных компонентов и зависимостей между ними. Диаграмма компонентов разрабатывается для следующих целей: визуализация общей структуры исходного кода программной системы и спецификация исполнимого варианта программной системы[6] (рисунок 5.3).

****

Рисунок 5.3 – Диаграмма компонентов системы оценки кинофильмов

Диаграмма классов используется для визуального изображения отношений между классами и интерфейсами в программе. В приложении А представлена диаграмма классов приложения.

**6 МОДЕЛИ ПРЕДСТАВЛЕНИЯ СИСТЕМЫ И ИХ ОПИСАНИЕ**

**6.1 Диаграмма состояний**

Диаграмма состояний (state diagram) определяют все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате влияния некоторых событий. Диаграммы состояний строятся для единственного класса и описывают поведение единственного объекта[4].

Диаграмма состояний представляет собой граф состояний в которых может находиться объект и связей между ними (рисунок 6.1).

**6.2 Алгоритм удаления данных из БД**

Алгоритм удаления данных о пользователях из БД состоит из нескольких шагов.

1. Формирование SQL-запроса в БД на удаление по выбранному id.
2. Выполнение запроса.
3. Выборка данных из БД.
4. Удаление данных.

Данный алгоритм представлен на рисунке 6.2.



Рисунок 6.1 – Диаграмма состояний при регистрации нового пользователя

**7 ОПИСАНИЕ ПРИМЕНЕНИЯ ПАТТЕРНОВ ПРОЕКТИРОВАНИЯ**

Model-view-controller (MVC) - это программный архитектурный шаблон для реализации пользовательских интерфейсов на компьютерах. Он делит данное приложение на три взаимосвязанные части. Это делается для разделения внутренних представлений информации от способов представления и принятия пользователем информации. Модель проектирования MVC отделяет эти основные компоненты, что позволяет эффективно использовать повторное использование кода и параллельную разработку.

Традиционно используемая для графических пользовательских интерфейсов (GUI) для настольных компьютеров, эта архитектура стала популярной при разработке веб-приложений и даже мобильных, настольных и других клиентов.

Компонентами являются: модель и представление.

Модель является центральной составляющей шаблона. Он выражает поведение приложения в терминах проблемной области, независимо от пользовательского интерфейса. Он напрямую управляет данными, логикой и правилами приложения.

Представление может представлять собой любое выходное представление информации, например, диаграмму или диаграмму. Возможны множественные представления одной и той же информации, такие как гистограмма для управления и табличное представление для бухгалтеров.

* Третья часть или раздел, контроллер, принимает вход и преобразует его в команды для модели или представления.

В дополнение к делению приложения на три вида компонентов дизайн модели-представления-контроллера определяет взаимодействия между ними.

Модель хранит данные, которые извлекаются в соответствии с командами контроллера и отображаются в представлении.

Представление генерирует новый результат для пользователя на основе изменений в модели.

Контроллер может отправлять команды модели для обновления состояния модели (например, редактирования документа). Он также может отправлять команды в связанный с ним вид, чтобы изменить представление представления модели (например, прокрутка документа, перемещение документа).



Рисунок 5.1 – Паттерн MVC

Плюсы шаблона MVC:

1) Более быстрый процесс разработки. MVC поддерживает быстрое и параллельное развитие. Если модель MVC используется для разработки какого-либо конкретного приложения, то возможно, что один программист может работать над представлением, а другой может работать на контроллере для создания бизнес-логики приложения. Таким образом, приложение, разработанное с использованием модели MVC, может быть выполнено в три раза быстрее, чем приложения, которые разрабатываются с использованием другого шаблона разработки.

2) Возможность предоставления нескольких видов. В модели MVC вы можете создать несколько представлений для модели. Сегодня растет спрос на новые способы доступа к вашему приложению, и для этого развитие MVC, безусловно, является отличным решением. Кроме того, в этом методе дублирование кода очень ограничено, поскольку оно отделяет данные и бизнес-логику от дисплея.

3) Модификация не влияет на всю модель. Для многих приложений пользовательский интерфейс имеет тенденцию меняться чаще, чем даже бизнес-правила компании. Очевидно, что вы часто вносите изменения в свое приложение, например, меняете цвета, шрифты, макеты экрана и добавляете новую поддержку устройств для мобильных телефонов или планшетов. Более того, добавление нового типа просмотров очень просто в шаблоне MVC, потому что часть модели не зависит от части просмотров. Поэтому любые изменения в Модели не будут влиять на всю архитектуру.

Недостатки шаблона MVC:

1) Сложность заключается в разработке приложений с использованием этого шаблона.

2) Не подходит для небольших приложений, которые оказывают неблагоприятное влияние на производительность и дизайн приложения.

3) Изолированный процесс разработки авторами пользовательского интерфейса, авторами бизнес-логики и авторами-контроллерами может привести к задержке в разработке их соответствующих модулей.

Также при создании приложения был использован паттерн «Фабричный метод».

В программировании на основе классов шаблон фабричного метода представляет собой шаблон создания, который использует фабричные методы для решения проблемы создания объектов без указания точного класса объекта, который будет создан. Это делается путем создания объектов путем вызова фабричного метода - либо указанного в интерфейсе, либо реализованного дочерними классами, либо реализованного в базовом классе, либо необязательно переопределенного производными классами, а не путем вызова конструктора.

Цель: Определяет интерфейс для создания объекта, но оставляет решение подкласса, для которого класс должен создавать экземпляр. Метод factory позволяет классу делегировать создание подклассов.

Используется, когда:

1. класс не известен заранее, объекты того, какие подклассы ему нужно создавать;
2. класс разработан таким образом, что создаваемые объекты определяются подклассами;
3. класс делегирует свои обязанности одному из нескольких подклассов, и планируется локализовать знания о том, какой класс берет на себя эти обязанности;

Плюсы:

1. позволяет скрыть реализацию прикладного шва (основные интерфейсы, составляющие ваше приложение)
2. позволяет легко протестировать шов приложения (то есть обмануть / заглушить) определенные части приложения, чтобы вы могли создавать и тестировать другие части.
3. позволяет с легкостью изменять дизайн вашего приложения.

Минусы:

1. делает код более трудным для чтения, поскольку весь ваш код находится за абстракцией, которая, в свою очередь, может скрывать абстракции.
2. может быть классифицирован как анти-шаблон, когда он неправильно используется.

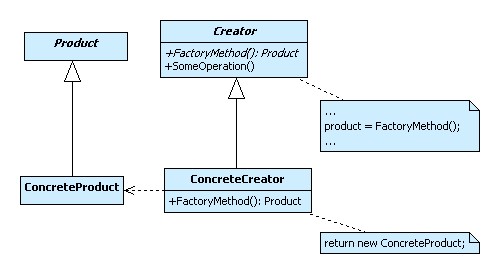


Рисунок 5.2 – Паттерн Фабричный метод

Паттерн Command преобразовывает запрос на выполнение действия в отдельный объект-команду. Такая инкапсуляция позволяет передавать эти действия другим функциям и объектам в качестве параметра, приказывая им выполнить запрошенную операцию. Команда – это объект, поэтому над ней допустимы любые операции, что и над объектом.

Интерфейс командного объекта определяется абстрактным базовым классом Command и в самом простом случае имеет единственный метод execute(). Производные классы определяют получателя запроса (указатель на объект-получатель) и необходимую для выполнения операцию (метод этого объекта). Метод execute() подклассов Command просто вызывает нужную операцию получателя. UML-диаграмма классов паттерна Command можно увидеть на рисунке 6.2.

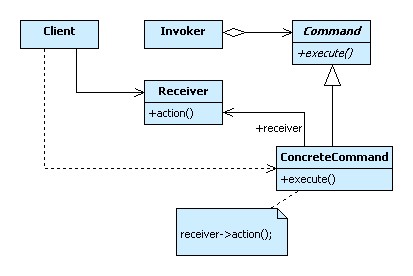


Рисунок 6.2 –Паттерн Команда

**8 РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ СИСТЕМЫ**

Для того, чтобы развернуть онлайн-сервис оценки кинофильмов необходимо выполнить нижеперечисленные шаги.

Во-первых необходимо осуществить подключение к базе данных. В данной системе используется СУБД MySQL и для правильности взаимодействия сервиса с базой данных необходимо настроить некоторые параметры.

Для начала необходимо скачать и установить MySQL. Для создания базы данных для разрабатываемой системы необходимо выполнить скрипт, находящийся в приложении под названием db\_script.sql . После этого необходимо проверить настройки для доступа к базе в файле project.properties, находящемся в movierating/src/main/resources. В случае несовпадения, исправить на корректные.

Во-вторых необходимо загрузить сервер на хостинг. Для этого необходимо скачать Apache Maven и Apache Tomcat и запустить проект уже с помощью Tomcat.

Из всего вышеперечисленного следует, что для корректной работы и обеспечения полного функционирования системы необходимы такие инструменты, как СУБД MySQL, Apache Maven и Apache Tomcat.

**9 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ И ОЦЕНКА ВЫПОЛНЕНИЯ ЗАДАЧ**

**ЗАКЛЮЧЕНИЕ**

В результате проделанной работы было разработано приложение, которое помогает решить задачу управления и учета бюджета семьи. Приложение имеет простой, комфортный и понятный интерфейс, который облегчает процесс взаимодействия с приложением. Программа выполняет поставленные задачи и цели. Также в приложении имеется база данных с авторизационными данными аккаунтов пользователей приложения, а также с информацией о бюджете семьи.

В перспективе данное приложение имеет возможность масштабирования и расширения. За счет модульности и использования паттернов проектирования повышается удобство проведения этих процессов. При модификации возможно добавление дополнительных методов и алгоритмов ведения семейного бюджета.

В итоге хотелось бы отметить, что программа полностью выполняет поставленные задачи, что является ее главным достоинством.

В процессе выполнения курсовой работы была изучена предметная область и рассмотрены теоретические аспекты в области планирования семейного бюджета. Следовательно, задачи курсового проекта можно считать выполненными.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Саати Т. Принятие решений. Метод анализа иерархий. М.: Радио и связь, 1993. - 320с.
2. Эккель Б. Философия Java. Библиотека программиста. – СПб.: Питер, 2001. – 880 с.: ил.
3. Java [Электронный ресурс]. – Электронные данные. –  http://progopedia.ru/language/java/
4. Дейт, К. Введение в системы баз данных/ К. Дейт: пер. с англ. – 6—е изд. — СПб.: Издательский дом «Вильямс», 2000.— 848 с.
5. Преимущества Java [Электронный ресурс]. – Электронные данные. – https://www.ibm.com/support/knowledgecenter/ru/ssw\_aix\_61/com. ibm.aix.perfo rmance/advantages\_java.htm
6. Wiki-учебник по веб-технологиям: MySQL [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://www.webmasterwiki.ru/mysql#h177-4>
7. Основы работы с MySQL Workbench: быстрый старт, управление схемой данных [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://mithrandir.ru/professional/soft-and-hardware/mysql-workbench-basics.html>

**ПРИЛОЖЕНИЕ А**



Рисунок А.1 – Диаграмма классов пакета command

**Продолжение приложения А**



Рисунок А.2 – Диаграмма классов пакета config

**Продолжение приложения А**



Рисунок А.3 – Диаграмма классов пакета controller



Рисунок А.4 – Диаграмма классов пакета dao

**Продолжение приложения А**



Рисунок А.5 – Диаграмма классов пакета connection

**Продолжение приложения А**



Рисунок А.6 – Диаграмма классов пакета entity

**Продолжение приложения А**



Рисунок А.7 – Диаграмма классов пакета entity

**Продолжение приложения А**



Рисунок А.8 – Диаграмма классов пакета exception



Рисунок А.9 – Диаграмма классов пакета filter

**Продолжение приложения А**



Рисунок А.10 – Диаграмма классов пакета listener



Рисунок А.11 – Диаграмма классов пакета util

**Продолжение приложения А**



Рисунок А.12 – Диаграмма классов пакета service