**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

Курсовая работа

**по дисциплине «Объектно-ориентированное программирование»**

Тема: «СОЗДАНИЕ ПРОГРАММНОГО КОМПЛЕКСА СРЕДСТВАМИ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ»

|  |  |  |
| --- | --- | --- |
| Студент гр. 3311 |  | Баймухамедов Р. Р. |
| Преподаватель |  | Павловский М. Г. |

Санкт-Петербург

2024

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**Введение**

ПК администрирования кинотеатра предназначен для автоматизации процессов управления данными о фильмах, репертуаре, сеансах и проданных билетах. Это приложение создано для курсового проекта по дисциплине "Объектно-ориентированное программирование".

**Основание для разработки**

Основанием для разработки ПК "Управление репертуаром кинотеатра" является курсовой проект, требующий реализации объектно-ориентированного подхода

**Назначение разработки**

ПК предназначен для автоматизации следующих задач:

* Добавление, изменение, удаление данных о фильмах, сеансах и проданных билетах.
* Просмотр репертуара на месяц.
* Получение информации о количестве проданных билетов

**Требования к программе**

***Перечень функций***

ПК должен предоставлять:

* Добавление, удаление, редактирование данных.
* Выдачу справочной информации по запросам пользователя.

***Организация данных***

Данные должны быть связаны через таблицу "многие ко многим". Основные таблицы:

* Фильмы
* Репертуар
* Сеансы
* Билеты

**Надежность**

Программа должна стабильно работать в рамках ОС Windows, предотвращать сбои и проверять корректность входных данных.

**Условия эксплуатации**

Программа ориентирована на однопользовательский режим работы с монопольным доступом к базе данных.

**Требования к информационной и программной совместимости**

ПК должен быть выполнен на языке программирования Java и работать под управлением ОС Windows. Интерфейс программы должен быть интуитивно понятным и удобным для восприятия пользователем. Выходные данные должны быть представлены в табличной форме, обеспечивая быстрый доступ к информации о репертуаре, сеансах и проданных билетах.

***Обязательные требования при реализации кода:***

* Закрытые и открытые члены классов.
* Использование наследования.
* Реализация конструкторов с параметрами и конструктора копирования.
* Абстрактные базовые классы.
* Виртуальные функции.
* Обработка исключительных ситуаций.
* Динамическое создание объектов.

*Рекомендуется включить:*

* Дружественные функции.
* Переопределение функций или операторов.
* Шаблонные классы.

**Требования к программной документации**

Программная документация должна соответствовать требованиям стандартов ЕСПД и включать:

1. Описание процесса проектирования ПК.
2. Руководство пользователя для работы с приложением.
3. Исходные тексты программы с комментариями.

**Стадии и этапы разработки**

1. Разработка технического задания
2. Описание вариантов использования ПК
3. Создание прототипа интерфейса пользователя
4. Разработка объектной модели ПК
5. Построение диаграмм программных классов
6. Описание поведения ПК
7. Построение диаграмм действий

**Порядок контроля и приёмки**

В процессе приема работы устанавливается соответствие ПК и прилагаемой документации требованиям, обозначенным в техническом задании

**ПРОЕКТИРОВАНИЕ ПК**

**Описание вариантов использования ПК**

Развернутое описание функциональных требований осуществляется на этапе проектирования комплекса. Для того чтобы детализировать требования, необходимо выделить процессы, происходящие в заданной предметной области. Описание таких процессов на UML выполняется в виде прецедентов (use case). Прецеденты являются сценарием или вариантом использования ПК при взаимодействии с внешней средой. Они являются продолжением описаний требований и функциональных спецификаций, указанных в техническом задании. Прецедент изображается в виде эллипса, в котором содержится имя прецедента. Название прецедента обязательно включает в себя глагол, выражающий суть выполняемой функции. С помощью прецедентов описывается функционирование ПК с точки зрения внешнего пользователя, который называется в UML актором (actor). Актор представляет собой любую внешнюю по отношению к моделируемой системе сущность (человек, программная система, устройство), которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач. Актор на диаграмме изображается пиктограммой в виде человечка, под которым указано его имя. Совокупность функций, реализуемых ПК, изображается в виде диаграммы (use case diagram). Для построения диаграммы необходимо определить акторы, прецеденты (функции) и взаимоотношение между акторами и прецедентами, и между прецедентами, если один прецедент расширяет или использует другой. В языке UML для вариантов использования и действующих лиц поддерживается несколько типов связей. Это связи коммуникации (communication), использования (uses) и расширения (extends).

Связь коммуникации — это связь между прецедентом и актором. На языке UML связь коммуникации изображают в виде стрелки. Направление стрелки показывает, кто инициирует коммуникацию. При задании коммуникации необходимо указать данные, которые вводит или получает пользователь. Кроме данных на концах стрелки можно указать кратности отношения, которые характеризуют количество взаимодействующих между собой акторов и прецедентов. На диаграммах прецедентов наиболее распространенными являются две формы записи кратности 1 и 1 .. \*. Первая форма записи означает, что один актор (прецедент) участвует во взаимодействии, а вторая форма записи, что один или несколько акторов (прецедентов) участвуют во взаимодействии.

Связь использования предполагает, что один прецедент всегда применяет функцио­нальные возможности другого. С помощью таких связей структурируют прецеденты, показывая тем самым, какой прецедент является составной частью другого прецедента. Такой включаемый прецедент является абстрактным прецедентом в том смысле, что он не может исполняться независимо от других прецедентов, а лишь в их составе. Связь использования изображается с помощью стрелок и слова «uses» (использование). Направление стрелки указывает, какой прецедент используется для реализации функциональности другого прецедента.

Связь расширения задается в том случае, если необходимо показать родственные отношения между двумя прецедентами. Один из них является базовым, а другой его расширением. Базовый прецедент не зависит от расширяющих прецедентов и способен функционировать без них. С другой стороны, расширяющие прецеденты без базового прецедента функционировать не могут. Связи расширения изображают в виде стрелки со словом «extends» (расширение), которая имеет направление от базового прецедента к расширяемому.

Прецеденты необходимо ранжировать, чтобы в начальных циклах разработки реализовать наиболее приоритетные из них. Разбиение функциональности системы на отдельные прецеденты служит примерно той же цели, что и разбиение сложного алгоритма на подпрограммы. Основная стратегия должна заключаться в том, чтобы сначала сконцентрировать внимание на тех прецедентах, которые в значительной мере определяют базовую архитектуру ПК.

*Диаграмма прецедентов представлена на рис. 1*

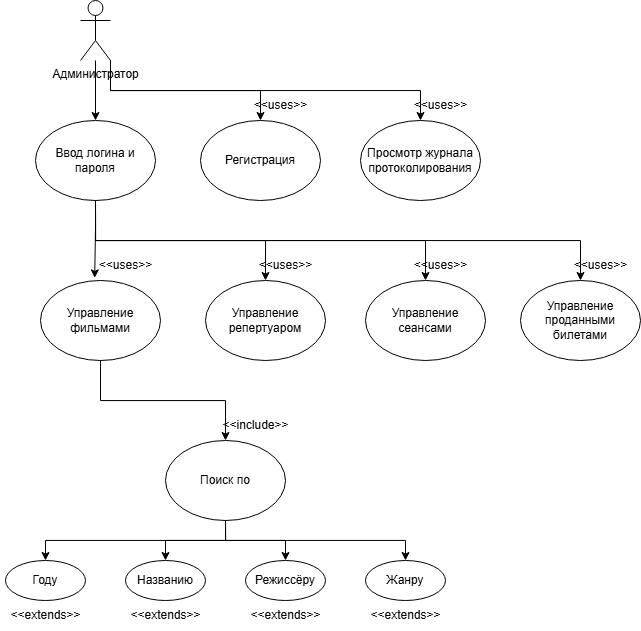


Рисунок 1

*Под прецедентом управление разделом имеется в виду обобщенный прецедент, включающий в себя добавление, удаление, изменение данных раздела*

**Создание прототипа интерфейса пользователя**

Описание прецедента выражает общую сущность процесса без детализации его реализации. Проектные решения, связанные с интерфейсом пользователя, при этом опускаются. Для разработки пользовательского интерфейса необходимо описать процесс в терминах реальных проектных решений, на основе конкретных технологий ввода-вывода информации. Когда речь идет об интерфейсе пользователя, прецеденты разбиваются на экранные формы, которые определяют содержимое диалоговых окон и описывают способы взаимодействия с конкретными устройствами. Для каждой экранной формы указываются поля ввода и перечень элементов управления, действия пользователя (нажать кнопку, выбрать пункт меню, ввести данные, нажать правую/левую кнопку мыши) и отклики системы (отобразить данные, вывести подсказку, переместить курсор). Такое описание интерфейса представляется в виде таблицы экранных форм. Данная таблица будет отображена ниже, после определения архитектуры и макета экранной

**Архитектура программы**

На следующей диаграмме *(рис. 2.1)* отображена архитектура разделов и данных программы, а ниже приведена информация о разделе, его описание и функционал

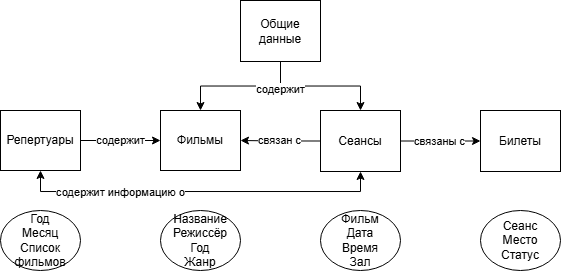


Рисунок 2.1

В данной модели данных сущности "Фильмы" и "Сеансы" связаны между собой через промежуточную сущность "Репертуары". Это позволяет реализовать связь "многое ко многим".

Сущность "Фильмы" содержит информацию о различных фильмах, таких как название фильма, год выпуска, режиссер и жанр. Каждый фильм может быть включен в несколько репертуаров, что означает связь "один ко многим" между "Фильмами" и "Репертуары".

Сущность "Репертуары" служит промежуточной таблицей, которая связывает фильмы с сеансами. Она содержит информацию о том, какие фильмы включены в каждый репертуар и в какое время они будут показаны. Один репертуар может включать несколько фильмов, и один фильм может быть включен в несколько репертуаров.

Сущность "Сеансы" содержит информацию о конкретных показах фильмов, таких как дата и время сеанса, количество мест и количество проданных билетов. Каждый сеанс связан с одним репертуаром, что означает связь "один ко многим" между "Репертуары" и "Сеансы".

Таким образом, через сущность "Репертуары" реализуется связь "многое ко многим" между "Фильмами" и "Сеансы". Это позволяет одному фильму участвовать в нескольких сеансах, и одному сеансу показывать один из нескольких фильмов, включенных в репертуар. Диаграмму описанной связи можно увидеть на рис. 2.2



Рисунок 2.2

***Фильмы (Управление базой фильмов)***

1. Просмотр списка фильмов:
   * Отображение списка всех фильмов в базе данных. Для каждого фильма указываются:
     + Название.
     + Режиссёр.
     + Год выпуска.
     + Жанр.
2. Добавление нового фильма:
   * Ввод информации о фильме:
     + Название, режиссёр, год, жанр.
   * После сохранения запись становится доступной для добавления сеансов в разделе "Репертуар".
3. Удаление фильма:
   * При удалении фильма автоматически удаляются связанные записи в "Репертуаре" и "Сеансах".
   * Удаление возможно только если у фильма нет проданных билетов. В противном случае система выводит предупреждение.
4. Редактирование информации о фильме:
   * Возможность изменить название, режиссёра, год или жанр фильма.
   * Изменения немедленно отражаются в связанных разделах ("Репертуар" и "Сеансы").

***Репертуар (Управление расписанием кинотеатра)***

1. Просмотр репертуара на месяц:
   * Отображение списка фильмов и сеансов на текущий месяц с указанием:
     + Названия фильма.
     + Даты и времени сеанса.
     + Зала.
2. Добавление сеансов фильма:
   * Пользователь выбирает фильм из списка.
   * Вводит дату, время, зал.
   * Запись синхронизируется с разделом "Сеансы".
3. Удаление сеанса:
   * Удаляется выбранный сеанс из расписания.
   * Сеанс и все связанные данные удаляются из раздела "Сеансы".
4. Редактирование информации о сеансе:
   * Пользователь может изменить время, дату или зал для выбранного сеанса.
   * Все изменения автоматически обновляются в "Сеансах".

***Сеансы (Управление билетами и деталями сеансов)***

1. Просмотр информации о сеансах конкретного фильма:
   * После выбора фильма отображаются все его сеансы с указанием:
     + Даты, времени, зала.
     + Проданных и доступных билетов.
2. Продажа билетов:
   * Пользователь может увидеть количество проданных билетов и оставшиеся места.
3. Информация о зале:
   * Отображение количества мест в зале и текущего статуса мест (свободные/занятые).
4. Добавление новых сеансов:
   * Пользователь может добавить новый сеанс (с указанием фильма, даты, времени, зала).
   * Новый сеанс автоматически добавляется в "Репертуар".
5. Удаление и изменение текущих сеансов:
   * Удаление сеанса из "Сеансов" автоматически удаляет его из "Репертуара".
   * Изменение данных о сеансе (дата, время, зал) синхронизируется с "Репертуаром".

***Билеты (Управление продажами и учётом билетов)***

1. Просмотр проданных билетов на сеанс:
   * Отображение списка проданных билетов с указанием:
     + Номера места.
     + Статуса (продан/свободен).
   * Общая сумма продаж на выбранный сеанс.
2. Продажа билетов:
   * Выбор сеанса и номера места.
   * Ввод данных покупателя.
   * После подтверждения продажа сохраняется, место становится занятым.
3. Просмотр статистики продаж:
   * Отображение общего количества проданных билетов за месяц.
4. Информация о доступных местах:
   * Список свободных мест для выбранного сеанса.
   * Возможность быстро выбрать свободное место для продажи.

***Логика взаимодействия разделов “Сеансы” и “Репертуар”***

*Основная идея*

* Раздел "Репертуар" оперирует более высоким уровнем абстракции, работая с расписанием на месяц. Он предназначен для управления общим списком фильмов и их сеансов.
* Раздел "Сеансы" работает с деталями каждого отдельного сеанса, предоставляя функционал для управления билетами и информации о зале.

*Связь и функциональные зависимости*

1. Добавление и удаление сеансов:
   * Когда администратор добавляет сеанс через "Репертуар", создаётся соответствующая запись в "Сеансах", включающая дату, время и зал.
   * Удаление сеанса в "Репертуаре" автоматически удаляет связанные записи в "Сеансах" и все связанные билеты.
2. Редактирование информации о сеансе:
   * Изменение информации о сеансе в "Репертуаре" синхронизируется с соответствующей записью в "Сеансах". Например, изменение времени или зала в "Репертуаре" обновляет эти данные в "Сеансах".
3. Просмотр информации:
   * В "Репертуаре" доступен общий просмотр сеансов по месяцам, без детальной информации о зале и билетах.
   * Для более подробной информации (проданные билеты, зал) пользователь переходит в "Сеансы", где данные фильтруются по выбранному сеансу.

***Пример логики взаимодействия между "Сеансами" и "Билетами":***

* При создании нового сеанса система автоматически генерирует записи о всех доступных местах для зала.
* При продаже билета запись о месте обновляется, и его статус меняется на "продан".
* Раздел "Сеансы" предоставляет быстрый доступ к данным о проданных и свободных билетах.

**Структура базы данных**

*Таблица раздела “Фильмы”*

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| movie\_id | INT (PK) | Уникальный идентификатор фильма |
| movie\_name | TEXT | Название фильма |
| director | TEXT | Имя режиссёра |
| movie\_year | INT | Год выпуска фильма |
| genre | TEXT | Жанр фильма |

*Таблица раздела “Репертуар”*

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| rep\_id | INT (PK) | Уникальный идентификатор записи |
| movie\_id | INT (FK) | Внешний ключ, указывает на фильм |
| date | DATE | Дата показа фильма |
| time | TIME | Время показа фильма |
| host | INT | Номер зала |

*Таблица раздела “Сеансы”*

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| session\_id | INT (PK) | Уникальный идентификатор сеанса |
| rep\_id | INT (FK) | Внешний ключ, указывает на запись в репертуаре |
| numb\_seats | INT | Общее количество мест в зале |
| numb\_sold\_tickets | INT | Количество проданных билетов |

*Таблица раздела “Билеты”*

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| ticket\_id | INT (PK) | Уникальный идентификатор билета |
| session\_id | INT (FK) | Внешний ключ, указывает на сеанс |
| numb\_of\_seat | INT | Номер места в зале |
| status | TEXT | Статус билета (свободен/продан) |
| date\_bought | DATETIME | Дата и время покупки билета |

Первичный ключ (Primary Key, PK) — это уникальный идентификатор для каждой записи в таблице базы данных. Он обеспечивает уникальность каждой записи и не может содержать значения NULL.

Внешний ключ (Foreign Key, FK) — это столбец или набор столбцов в одной таблице, который ссылается на первичный ключ в другой таблице. Он используется для установления и поддержания связей между таблицами. Внешний ключ обеспечивает ссылочную целостность, то есть значения в столбце внешнего ключа должны соответствовать значениям первичного ключа в другой таблице.

Представление базы данных в виде диаграммы изображено на рис. 3.1

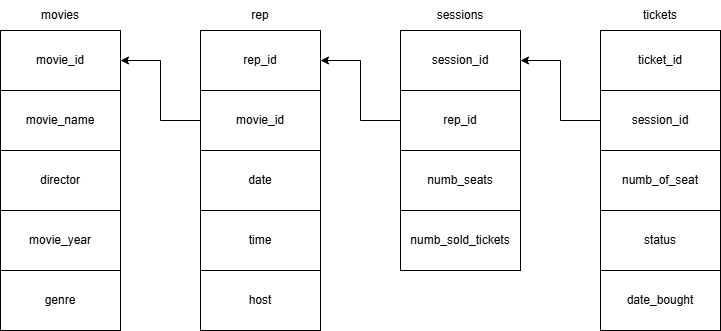


Рисунок 3.1

**Реализация базы данных**

В ходе разработки приложения для администратора кинотеатра было принято решение использовать SQLite в качестве системы управления базами данных. Данное решение обусловлено несколькими ключевыми факторами, которые соответствуют требованиям проекта и общей концепции программной реализации.

SQLite представляет собой компактную, высокоэффективную и кроссплатформенную реляционную базу данных, которая идеально подходит для встроенных приложений, таких как наше. Она предоставляет все необходимые возможности реляционной модели данных и соответствует стандарту SQL, что делает её отличным выбором для создания и управления структурированными данными в приложении. Более того SQLite входит в тройку самых популярных реляционных баз данных, что представлено на рис. 3.2 ([Источник Stack Overflow Developer Survey 2021](https://insights.stackoverflow.com/survey/2021))

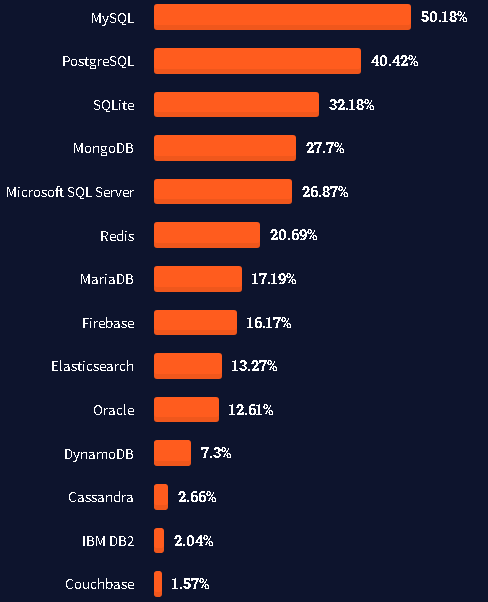


Рисунок 3.2

***Преимущества выбора SQLite***

* 1. Лёгкость и удобство использования

SQLite — это библиотека, которая интегрируется непосредственно в приложение без необходимости установки отдельного серверного ПО. Это упрощает процесс настройки и минимизирует затраты на администрирование. Для запуска достаточно лишь одного файла базы данных, что делает SQLite особенно удобной для небольших и средних проектов.

* 1. Эффективность и производительность

SQLite разработана таким образом, чтобы быть максимально быстрой и эффективной в обработке запросов. В рамках приложения администратора кинотеатра это особенно важно, так как функционал предусматривает работу с репертуаром, сеансами, билетами и фильмами.

* 1. Отсутствие зависимостей

В отличие от других СУБД, SQLite не требует наличия серверной части. Это позволяет исключить потенциальные проблемы, связанные с совместимостью программного обеспечения, а также снизить требования к аппаратному обеспечению.

* 1. Поддержка SQL-стандарта

SQLite поддерживает стандарт SQL, что упрощает написание запросов для работы с таблицами, связи между данными и выполнение сложных операций.

***Интеграция и инструменты взаимодействия***

SQLite была интегрирована в приложение посредством JDBC (Java Database Connectivity). Это позволяет напрямую взаимодействовать с базой данных, реализуя функционал просмотра, добавления, редактирования и удаления данных через интерфейс приложения.

В качестве версии SQLite было выбрано SQLite version 3.47.2. Также для работы с SQLite было установлен инструмент для удобного взаимодействия с БД — SQLite Browser. Интерфейс можно увидеть на рис. 3.3

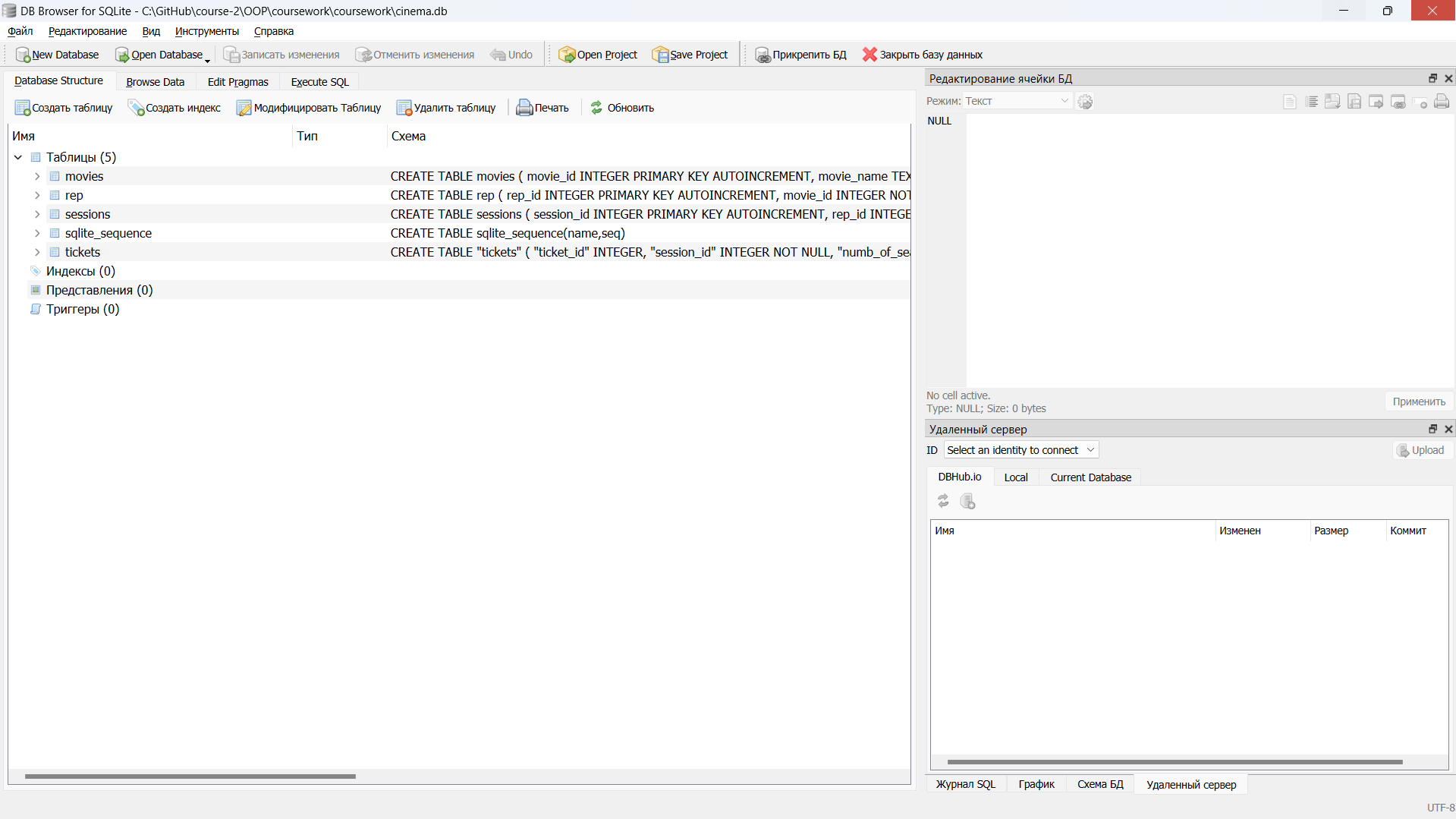


Рисунок 3.3

**Журнал протоколирования (Логи)**

В рамках текущего проекта терминология "логи" и "журнал протоколирования" будет использоваться как синонимы, подразумевая одно и то же — запись различных событий и действий, происходящих в приложении.

В качестве инструмента для логирования был выбран встроенный инструментарий Java, а не Log4j или Logback. Это решение было принято по нескольким причинам. Во-первых, встроенный инструментарий Java предоставляет все необходимые функции для логирования, такие как различные уровни логирования, форматирование сообщений и настройка вывода в файлы или консоль. Это позволяет удовлетворить потребности проекта без необходимости внедрения дополнительных библиотек.

Во-вторых, использование встроенного инструментария Java упрощает процесс разработки и поддержки приложения. Встроенный инструментарий не требует дополнительных зависимостей и конфигураций, что делает его более удобным для использования в небольших и средних проектах. Это особенно важно для однопользовательского приложения, где требования к логированию не столь высоки, как в многопользовательских или распределенных системах.

Кроме того, в данном проекте было принято решение сохранять логи в файл, а не в базу данных. Это связано с тем, что хранение логов в файле обеспечивает простоту и надежность, а также упрощает процесс доступа к логам для анализа и отладки. Файловая система позволяет легко архивировать и удалять старые логи, не нагружая при этом базу данных и не требуя дополнительных ресурсов для управления логами.

**Учётные записи и права администратора**

В данном проекте предусмотрено использование только одной учётной записи с административными правами. Эта запись позволяет управлять всеми аспектами приложения, включая работу с фильмами, репертуаром, сеансами и билетами. Для входа в систему администратор использует логин и пароль, которые по умолчанию заданы как *admin*.

Поскольку проект разрабатывается для одиночного пользователя с административными функциями, реализация других уровней доступа или дополнительных ролей (например, кассиров, менеджеров) не требуется. Такой подход упрощает архитектуру приложения, минимизируя затраты времени на разработку сложных систем разграничения прав.

Если в будущем появится необходимость в поддержке нескольких пользователей или разделении их ролей, это можно будет доработать. Однако на текущем этапе реализация ограничивается только одним пользователем-администратором, что вполне соответствует поставленным требованиям.

**Макет экранной формы**

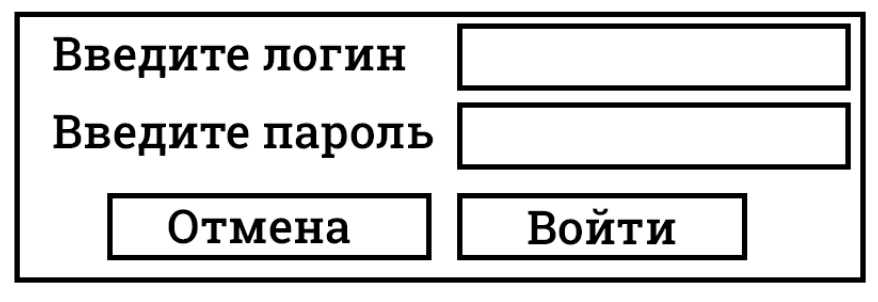


Рисунок 4.1.1 (Экранная форма входа)

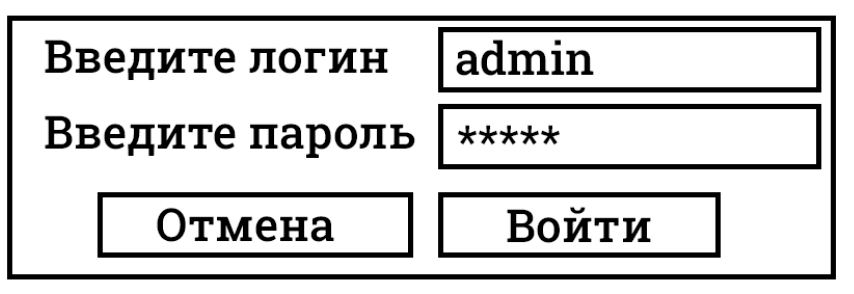


Рисунок 4.1.2 (Заполненная экранная форма входа)



Рисунок 4.1.3 (Экранная форма входа неправильного логина/пароля)



Рисунок 4.2 (Экранная форма приложения)

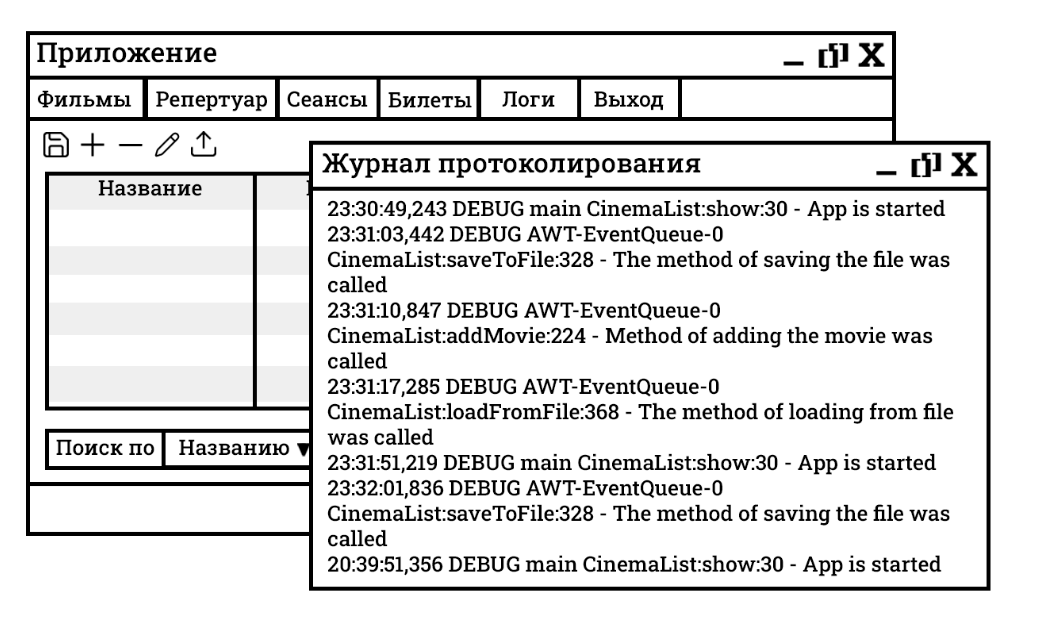


Рисунок 4.3 (Экранная форма приложения при нажатии вкладки “Логи”)

++++++++++еще допилить ввод там даты времени зала

В таблицах ниже представлено описание экранных форм и их взаимодействие с пользователем

*Таблица 4.1 (Панель вкладок)*

| Элементы управления | Действия пользователя | Отклик системы |
| --- | --- | --- |
| Меню с вкладками: | | |
| - "Фильмы" | Выбор вкладки "Фильмы" | Открывается экранная форма "Фильмы". |
| - "Репертуар" | Выбор вкладки "Репертуар" | Открывается экранная форма "Репертуар". |
| - "Сеансы" | Выбор вкладки "Сеансы" | Открывается экранная форма "Сеансы". |
| - "Билеты" | Выбор вкладки "Продажа билетов" | Открывается экранная форма "Продажа билетов". |
| - "Логи" | Выбор вкладки "Логи" | Открывается log.txt (журнал протоколирования приложения) поверх окна приложения |
| - "Выход" | Нажатие кнопки "Выход" | Приложение закрывается. |

*Таблица 4.2 (Панель инструментов)*

| Элементы управления | Действия пользователя | Отклик системы |
| --- | --- | --- |
| Кнопки: | | |
| - "Добавить данные" | Нажатие кнопки "Добавить данные" | Открывается окно для ввода информации о новых данных. |
| - "Изменить данные" | Выбор данных из списка и нажатие кнопки "Изменить данные" | Открывается окно для редактирования выбранных данных |
| - "Удалить данные" | Выбор данных из списка и нажатие кнопки "Удалить данные" | Выбранные данные удаляются из базы данных. |
| - "Сохранить данные" | Нажатие кнопки "Сохранить данные" | Сохранение текущих данных в базу данных. |
| - "Загрузить данные" | Нажатие кнопки "Загрузить данные" | Загрузка данных из базы данных в приложение. |

*Таблица 4.3 (Панель поиска)*

| Элементы управления | Действия пользователя | Отклик системы |
| --- | --- | --- |
| Поле выбора: | Выбор поля для поиска (например, название фильма) | Определяется критерий для выполнения поиска. |
| Поле ввода: | Ввод значения для поиска | Значение используется для выполнения поиска. |
| Кнопка "Найти" | Нажатие кнопки "Найти" | Отображаются результаты поиска на основе заданных условий. |

*Таблица 4.4 (Панель входа)*

|  |  |  |
| --- | --- | --- |
| Элементы управления | Действия пользователя | Отклик системы |
| Поле ввода "Логин" | Ввод логина | Используется для проверки учетных данных. |
| Поле ввода "Пароль" | Ввод пароля | Используется для проверки учетных данных. |
| Кнопка "Войти" | Нажатие кнопки "Войти" | При правильном вводе учетных данных открывается приложение. |
|  |  | При неправильном вводе отображается сообщение об ошибке. |
| Кнопка "Отмена" | Нажатие кнопки "Отмена" | Приложение закрывается. |

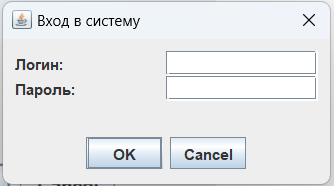
***Взаимодействие с данными при помощи панели инструментов***

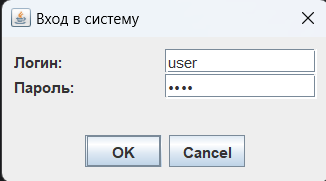
На всех вкладках управление данными работает одинаково: добавление, изменение, удаление, сохранение и загрузка. Единственное различие заключается в том, какая экранная форма и структура данных используются для каждой вкладки. Например, вкладка "Фильмы" предназначена для работы с информацией о фильмах, тогда как вкладка "Репертуар" отображает расписание и связанную информацию о фильмах.

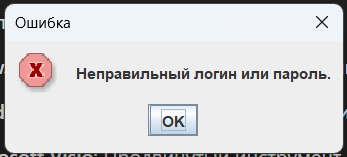
**Реализация экранной формы**

В качестве графической библиотеки для разработки пользовательского интерфейса приложения была выбрана Swing. Это решение было обусловлено тем, что Swing предоставляет богатый набор компонентов и возможностей для создания нативных приложений на языке Java. Swing позволяет легко управлять графическими элементами, такими как кнопки, панели, меню и окна, обеспечивая при этом высокую гибкость и производительность. Кроме того, для использования в интерфейсе были взяты иконки с сайта lucide.dev, который предоставляет широкий выбор качественных и современных иконок, что позволило улучшить визуальную привлекательность и удобство пользовательского интерфейса.

В ходе проектирования пользовательского интерфейса было принято решение о внесении нескольких изменений в структуру и расположение элементов. В частности, был изменен порядок расположения панели инструментов и панели вкладок. Это было сделано для улучшения удобства использования и навигации по приложению. Кроме того, функции "Логи" и "Выход" были перенесены в панель инструментов, что позволило пользователям быстрее получать доступ к этим важным функциям. Также был изменен порядок элементов в самой панели инструментов, чтобы сделать интерфейс более интуитивно понятным и логичным. Эти изменения направлены на повышение удобства и эффективности работы пользователей с приложением.







Fdgdfg

Dfgdfg

**Валидация данных**

В данном проекте особое внимание уделяется валидации данных, что является критически важным аспектом для обеспечения надежности и точности всей системы. Валидация данных включает в себя проверку корректности дат, чисел и временных меток, что позволяет избежать ошибок и несоответствий, которые могут привести к неправильным выводам и решениям. Проверка дат гарантирует, что все временные метки соответствуют реальным событиям и хронологическому порядку, что особенно важно для анализа временных рядов и исторических данных. Валидация чисел обеспечивает точность вычислений и предотвращает ошибки, связанные с неправильным вводом или интерпретацией числовых значений. Проверка времени позволяет синхронизировать различные процессы и события, что особенно важно в системах, где точность времени играет ключевую роль, таких как финансовые транзакции или управление реальным временем. В целом, валидация данных является фундаментальным элементом, который обеспечивает целостность и достоверность информации, что, в свою очередь, способствует принятию обоснованных решений и повышению общей эффективности системы.

**РУКОВОДСТВО ОПЕРАТОРА**

**Назначение программы**

Программный комплекс (ПК) "Учет и администрирование репертуара кинотеатра" предназначен для автоматизации управления сведениями о фильмах, сеансах и билетах в рамках деятельности администратора кинотеатра. Программа входит в состав автоматизированной системы учета и администрирования информации, обеспечивая учет репертуара и продаж.

В рамках ПК "Учет и администрирование репертуара кинотеатра" администратор может:

* добавлять, изменять и удалять информацию о фильмах, доступных в репертуаре;
* добавлять, изменять и удалять сведения о сеансах и их расписании;
* управлять информацией о проданных билетах;
* формировать справочную информацию о текущем репертуаре и статистике продаж.

**Условия выполнения программы**

Программа предназначена для работы под операционной системой Windows, с использованием SQLite или PostgreSQL в качестве базы данных. Для корректного функционирования требуется:

1. Персональная электронно-вычислительная машина (ПЭВМ) с операционной системой Windows;
2. Минимальные характеристики системы:
   * процессор Intel Core i3 или аналогичный;
   * объем оперативной памяти — не менее 4 ГБ;
   * объем свободного дискового пространства — не менее 500 МБ;
   * стандартная клавиатура и манипулятор типа "мышь".

**Описание задачи**

Программный комплекс должен обеспечивать хранение сведений о фильмах, расписании сеансов и проданных билетах. Администратор кинотеатра может управлять этими данными и запрашивать следующую информацию:

* доступность фильма для показа;
* расписание сеансов для выбранного фильма;
* количество проданных билетов и оставшихся мест;
* статистику посещаемости по сеансам.

Обязательными требованиями при разработке кода ПК являются использование следующих конструкций языка Java:

* закрытые и открытые члены классов;
* наследование;
* конструкторы с параметрами;
* абстрактные базовые классы;
* виртуальные функции;
* обработка исключительных ситуаций;
* динамическое создание объектов.

Для реализации поставленной задачи была разработана общая модель программного комплекса с выявлением основных объектов и их взаимосвязей. В ходе проектирования были созданы программные классы для работы с информацией о фильмах и сеансах, а также их связи через базу данных

**Входные и выходные данные**

Входные данные:

* Информация о фильмах, сеансах и продажах вводится администратором через графический интерфейс программы. Ввод данных осуществляется в интерактивном режиме.

Выходные данные:

* Таблицы, отображающие список фильмов с характеристиками, расписание сеансов и статистику проданных билетов.

**ИСХОДНЫЕ ТЕКСТЫ ПК**

**ЗАКЛЮЧЕНИЕ**

**Репозиторий**

Исходный код курсовой работы размещен в публичном репозитории на GitHub и доступен по следующе - . В репозитории содержится полный исходный код проекта, а также документация, которая может быть полезна для дальнейшего изучения и эксплуатации программного комплекса.

Сгенерированная документация Javadoc, содержащая подробное описание классов, методов и их взаимодействий, находится в папке doc внутри репозитория. Она предоставляет структурированное представление о внутреннем устройстве приложения и может быть использована как справочное руководство для разработчиков.

Видеоотчет о работе программного комплекса, включающий демонстрацию функционала и объяснение ключевых решений, доступен по следующей ссылке -

Эти материалы дополняют проект и обеспечивают его прозрачность и удобство изучения.

**Результат логирования**

Лог-файл, содержащий около 1,5 тысячи строк, оказался крайне компактным, занимая менее 100 КБ. Такой небольшой объем памяти при хранении значительного количества информации стал возможным благодаря рациональному подходу к структуре и формату записей. Логирование сыграло ключевую роль в процессе отладки и разработки проекта. Оно позволило оперативно выявлять и анализировать ошибки, отслеживать последовательность выполнения операций и поведение системы в различных сценариях. Благодаря этому удалось не только ускорить процесс устранения багов, но и улучшить качество реализации функционала, обеспечив стабильность и надежность итогового решения.

Итоги

В результате выполнения курсовой работы был разработан программный комплекс (ПК) «Учет и администрирование репертуара кинотеатра», предназначенный для управления сведениями о фильмах, расписании сеансов и проданных билетах. Программа успешно автоматизирует процессы, связанные с учетом репертуара и продаж, упрощает администрирование и повышает оперативность получения данных.

В ходе выполнения курсового проекта была проделана следующая работа:

1. Проведен анализ требований и разработана спецификация функциональности программного комплекса.
2. Построена модель предметной области, выявлены основные сущности и их взаимосвязи.
3. Созданы описание вариантов использования ПК, включая сценарии добавления, изменения и удаления информации о фильмах, сеансах и билетах.
4. Разработан прототип графического интерфейса пользователя (GUI), ориентированный на удобство и интуитивность использования.
5. Построена объектная модель ПК и диаграмма классов, описывающая структуру программных компонентов и их взаимодействие.
6. Составлено описание поведения ПК и диаграммы действий, иллюстрирующие основные рабочие процессы программы.
7. Реализован программный код с учетом обязательных требований, включая использование наследования, абстрактных классов, обработки исключений и динамического создания объектов.
8. Проведено тестирование программного комплекса, выявлены и исправлены ошибки.
9. Разработано руководство оператора, описывающее порядок установки, настройки программного комплекса.

Курсовой проект полностью удовлетворяет поставленным требованиям и обеспечивает автоматизацию ключевых задач администратора кинотеатра. Программный комплекс разработан с использованием современных подходов к объектно-ориентированному программированию на языке Java, что гарантирует его надежность, расширяемость и удобство в эксплуатации.