Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Инженерно-экономический факультет

Кафедра экономической информатики

Дисциплина: Распределенные системы обработки информации

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему:

**Разработка автоматизированной системы работы кинотеатра**

Выполнила: студентка гр. 074002

Гришанович Валерия Александровна

Проверил: старший преподаватель кафедры ЭИ

Лыщик Артём Петрович

Минск 2023

РЕФЕРАТ

**Гришанович В.А.** Разработка автоматизированной системы работы кинотеатра /Гришанович В.А. – Минск: БГУИР, 2023. – с. 50

Пояснительная записка с. 50, 41 рис., 11 источников, 2 приложения

**Ключевые слова**: информационная система, администратор, модели UML, IDEF0, Java, база данных, информационное моделирование.

*Цель* *курсового проекта*: разработка наиболее эффективных подходов в реализации системы онлайн-продажи билетов в кинотеатр.

*Объект исследования –* онлайн-продажа билетов в кинотеатр.

*Предмет исследования –* влияние внедрения автоматизации на улучшение качества работы кинотеатра.

*Методология проведения работы*: в процессе решения поставленных задач использованы принципы системного подхода, аналитические методы, моделирование системы с помощью стандарта UML 2.0, информационное моделирование базы данных.

*Результаты работы*: выполнена постановка задачи и определены основные методы ее решения; в ходе объектного моделирования системы построен ряд *UML*-диаграмм; разработана информационная модель системы, представленная в виде схемы базы данных; разработано руководство пользователя; выполнено тестирование системы, показавшее ее соответствие функциональным требованиям, поставленным в задании на разработку.

Программный продукт разработан на языке *Java* с применением платформы *JavaFX* и СУБД *PostgreSQL*.

*Область применения результатов*: в результате выполнения данного курсового проекта было разработано программное приложение, позволяющие оптимизировать работу кинотеатра.

Приложение удовлетворяет основным характеристикам, которые были заявлены. Оно удобно в эксплуатации, целостно, конкретизировано.

Для пользователя был разработан удобный и понятный интерфейс.

СОДЕРЖАНИЕ

[Введение 6](#_Toc122220931)

[1 Анализ и моделирование предметной области программного средства 7](#_Toc122220932)

[1.1 Описание предметной области 7](#_Toc122220933)

[1.2 Разработка функциональной модели предметной области 8](#_Toc122220934)

[1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований 1](#_Toc122220935)1

[1.4 Разработка информационной модели предметной области 1](#_Toc122220936)3

[1.5 UML-модели представления программного средства и их описание. 16](#_Toc122220937)

[2 Проектирование и конструирование программного средства](#_Toc122220938) 24

[2.1 Постановка задачи](#_Toc122220939) 24

[2.2 Обоснование выбора компонентов и технологий для реализации программного средства](#_Toc122220940) 24

[2.3 Архитектурные решения 2](#_Toc122220941)6

[2.4 Описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства 2](#_Toc122220942)7

[2.5](#_Toc122220943) Проектирование пользовательского интерфейса……………………….31

[3 Тестирование и проверка работоспособности программного средства](#_Toc122220944) 34

[4 Руководство по развертыванию и использованию программного средства](#_Toc122220945)…………………………………………………………………………..36

4.1 Руководство по установке (развёртыванию) программного средства...36

4.2 Руководство пользователя………………………………………………..36

[Заключение](#_Toc122220946) 47

[Список использованных источников 4](#_Toc122220947)8

[Приложение А (обязательное) Отчет о проверке на заимствования в системе «Антиплагиат» 46](#_Toc122220948)

[Приложение Б (обязательное) Листинг кода алгоритмов, реализующих основную бизнес-логику 47](#_Toc122220949)

[Ведомость документов курсового проекта 55](#_Toc122220950)

**ВВЕДЕНИЕ**

Курсовой проект на тему «Разработка автоматизированной системы работы кинотеатра» предназначен для улучшения и совершенствования работы современного кинотеатра, а также для сокращения времени при обработке данных.

В наше время, когда информационные технологии занимают все большее место в различных аспектах современной жизни, трудно представить себе, что управление таким процессом, как обработка информации о доступных фильмах, сеансах и проданных билетах в кинотеатре осуществляется вручную или на устаревшей технике.

Поэтому, проблема создания подобной автоматизированной системы весьма актуальна, поскольку объем обрабатываемой информации значителен, и вручную справляться с таким количеством данных не только сложно и трудоемко, но и экономически невыгодно.

Таким образом, можно говорить об актуальности разрабатываемого проекта.

Целью курсового проекта является упрощение и оптимизация многих процессов, связанных с организацией работы кинотеатра путем разработки программного продукта.

Разрабатываемый продукт позволит:

* 1. Планировать и управлять расписанием показов фильмов, определять время начала и окончания сеансов, а также устанавливать цены на билеты. Это позволяет кинотеатру эффективно распределять свои ресурсы и ресурсы персонала;
  2. Приобретать билеты на показы фильмов через интернет. Это значительно упрощает процесс покупки билетов для клиентов и позволяет кинотеатру увеличить количество продаж;
  3. Анализировать данные о продажах билетов, посещаемости кинотеатра и других параметрах, чтобы оптимизировать работу кинотеатра и увеличить прибыль. Это помогает кинотеатру принимать обоснованные решения на основе данных и улучшать свою эффективность.

Для достижения поставленной цели необходимо выполнить ряд задач:

* проанализировать предметную область и выполнить проектирование программного средства, используя UML-диаграммы, IDEF0, IDEF1X;
* спроектировать и реализовать клиент-серверное приложение;
* предусмотреть все исключительные ситуации, а также произвести тестирование приложения;
* разработать и описать руководство пользователя.

**1 АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА**

* 1. **Описание предметной области**

Кинотеатры имеют большое значение для общества, так как они предоставляют возможность людям наслаждаться культурными и развлекательными мероприятиями. Кинотеатры не только развивают культуру и искусство, но и создают рабочие места для многих людей, являются местом социальных встреч, где люди могут проводить время вместе с друзьями и семьей. Они играют важную роль в образовании и расширении кругозора, показывая фильмы на различные темы и жанры.

Организация работы кинотеатра — это достаточно сложная комплексная система, которая включает в себя следующие аспекты: выбор и аренду помещения для кинотеатра; приобретение необходимого оборудования; найм персонала; разработку расписания сеансов и выбор фильмов для показа; разработку маркетинговых стратегий для привлечения зрителей, включая рекламу в средствах массовой информации и социальных сетях; организацию продажи билетов и контроль за доступом зрителей в кинозал; обеспечение безопасности и комфорта зрителей, включая обеспечение чистоты помещения и соблюдение правил пожарной безопасности; управление финансами; развитие и внедрение новых технологий и инноваций для улучшения качества показа фильмов и удовлетворения потребностей зрителей и другое.

В настоящее время для улучшения работы современного кинотеатра вводятся автоматизированные системы, которые помогает упростить и оптимизировать многие процессы, связанные с организацией работы кинотеатра. Они позволяют:

* 1. Планировать и управлять расписанием показов фильмов, определять время начала и окончания сеансов, а также устанавливать цены на билеты. Это позволяет кинотеатру эффективно распределять свои ресурсы и ресурсы персонала;
  2. Приобретать билеты на показы фильмов через интернет. Это значительно упрощает процесс покупки билетов для клиентов и позволяет кинотеатру увеличить количество продаж;
  3. Анализировать данные о продажах билетов, посещаемости кинотеатра и других параметрах, чтобы оптимизировать работу кинотеатра и увеличить прибыль. Это помогает кинотеатру принимать обоснованные решения на основе данных и улучшать свою эффективность.

В целом, автоматизированная система является необходимой для оптимизации и улучшения работы кинотеатра, повышения эффективности и удобства для клиентов. Её применение значительно сокращает время и усилия, затрачиваемые на рутинные задачи, и повышает качество обслуживания зрителей. Кроме того, автоматизированная система может предоставлять аналитические данные о посещаемости и доходности, что позволяет управлять бизнесом более эффективно и принимать обоснованные решения, являющиеся важными для успешной работы в современном мире.

**1.2. Разработка функциональной модели предметной области**

Рассмотрим процесс онлайн-продажи билета в кинотеатре для более детального разбора предметной области при помощи модели IDEF0.

Компонентами синтаксиса IDEF0 являются блоки, стрелки, диаграммы и правила.

Блоки представляют собой функции, определяемые как деятельность, процесс, операция, действие или преобразование. Стрелки представляют данные или материальные объекты, связанные с функциями. Правила определяют, как следует применять компоненты. Диаграммы обеспечивают формат графического и словесного описания моделей. Формат образует основу для управления конфигурацией модели.

Поскольку IDEF0 является методологией функционального моделирования, имя блока, описывающее функцию, должно быть глаголом. После присваивания блоку имени, к соответствующим его сторонам присоединяются входные, выходные и управляющие стрелки, а также стрелки механизма, что и определяет наглядность и выразительность изображения блока IDEF0.

Стрелки и их сегменты, как отдельные, так и связанные в «пучок», помечаются существительными или оборотами существительного. Метки сегментов позволяют конкретизировать данные или материальные объекты, передаваемые этими сегментами, с соблюдением синтаксиса ветвлений и слияний.

Каждая сторона функционального блока имеет стандартное значение с точки зрения связи блок-стрелки. В свою очередь, сторона блока, к которой присоединена стрелка, однозначно определяет ее роль. Стрелки, входящие в левую сторону блока – это входы. Входы преобразуются или расходуются функцией, чтобы создать то, что появится на ее выходе. Стрелки, входящие в блок сверху – это управления. Управления определяют условия, необходимые функции, чтобы произвести правильный выход. Стрелки, покидающие блок справа – это выходы, то есть данные или материальные объекты, произведенные функцией. Стрелки, присоединённые к нижней стороне блока, представляют механизмы.

Каждая модель должна иметь контекстную диаграмму верхнего уровня, на которой объект моделирования представлен единственным блоком с граничными стрелками. Стрелки на этой диаграмме отображают связи объекта моделирования с окружающей средой. Поскольку единственный блок представляет весь объект, то его имя является общим для всего проекта. Это же справедливо и для всех стрелок диаграммы, поскольку они представляют полный комплект внешних интерфейсов объекта.

На рисунке 1.1 представлена контекстная диаграмма верхнего уровня. Входными данными являются информация о фильмах и заявка на покупку билета.

В процессе покупки билета входные данные преобразуются в оплаченный билет и возможность распечатать билет.

Управление процессом определяется Законами Республики Беларусь «О защите прав потребителей», «О защите персональных данных» и Постановлением Министерства культуры Республики Беларусь №139 «О кинотеатрах».

Ресурсами, необходимыми для выполнения процесса являются:

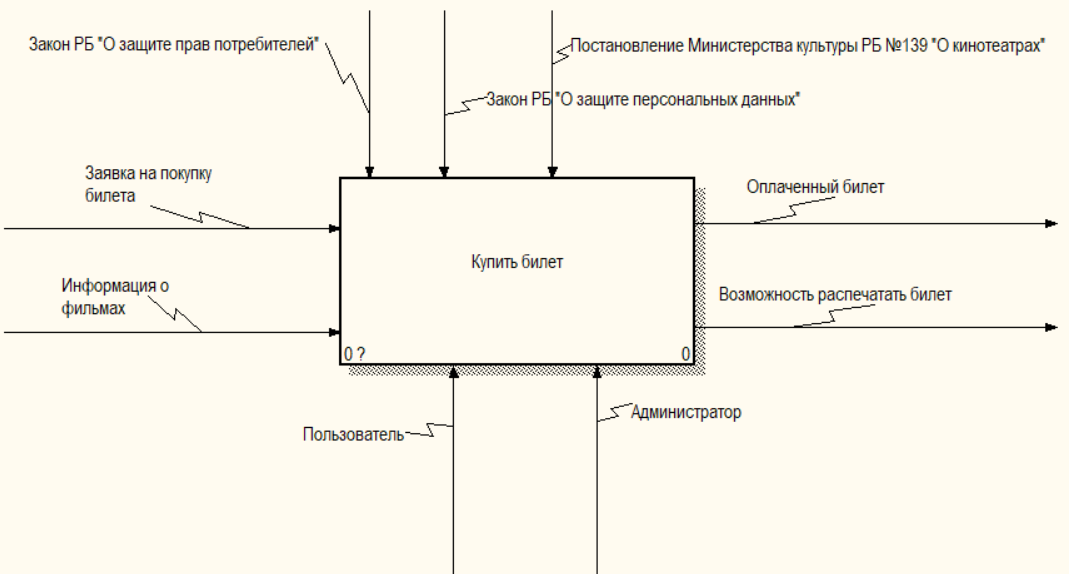
* администратор;
* пользователь.

Рисунок 1.1 — Контекстная диаграмма верхнего уровня

Процесс покупки билета начинается с выбора пользователем конкретного сеанса и проводится в соответствии с Постановлением Министерства культуры Республики Беларусь №139 «О кинотеатрах». Декомпозиция контекстной диаграммы, представленная на рисунке 1.2, описывает разделение основного процесса на подпроцессы. Руководствуясь вышеприведёнными законами и Постановлением Министерства культуры процесс покупки билета можно разделить на следующие этапы:

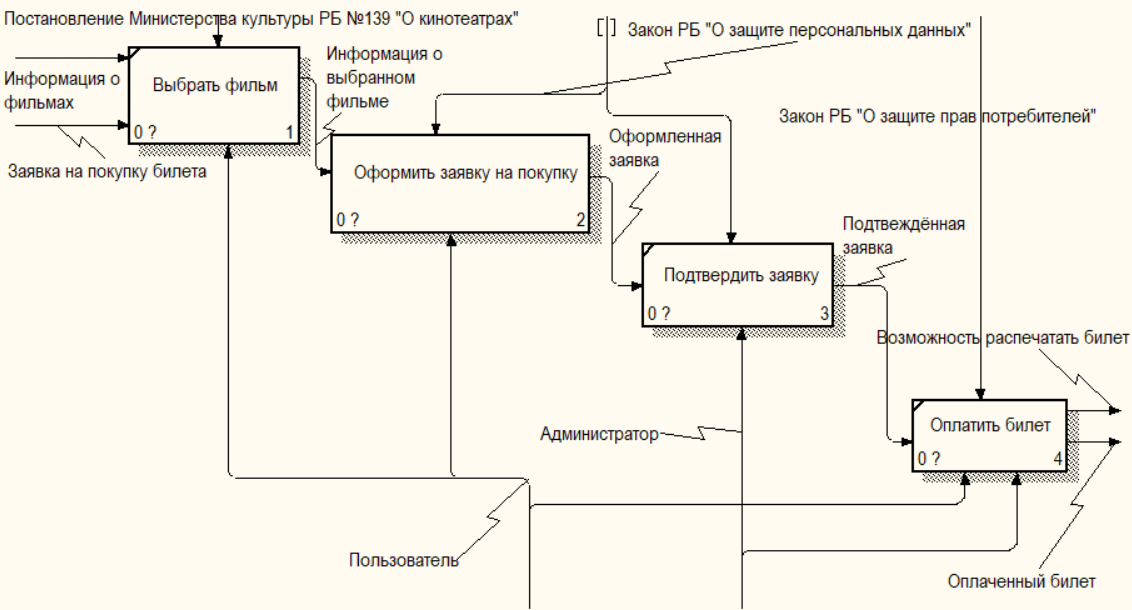
* выбрать фильм;
* оформить заявку на покупку;
* подтвердить заявку;
* оплатить билет.

Рисунок 1.2 — Декомпозиция контекстной диаграммы верхнего уровня

Произведём декомпозицию блока «Оформление заявки на покупку» (рисунок 1.3), который делится на следующие подпроцессы:

– выбрать день показа;

– выбрать сеанс;

– выбрать место.

Входная информация для первого блока — информация о выбранном фильме. Пользователь выбирает день показа выбранного им фильма, затем сеанс и место в зале. Затем пользователь вносит информацию в систему и оформляет заявку.

Результат предшествующих операций — оформленная заявка.

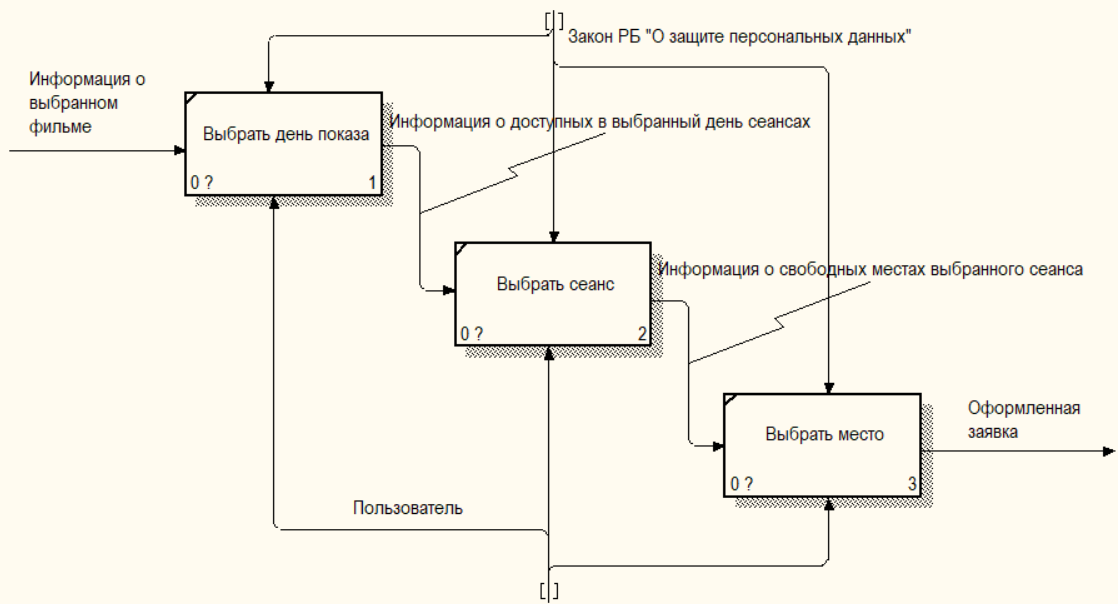


Рисунок 1.3 — Декомпозиция блока «Оформить заявку на покупку»

Прочитав и изучив все данные, описанные выше, можно понять, как происходит процесс онлайн-продажи билета в кинотеатре, как организован процесс и что обязательно должно в нём присутствовать, что способствует пониманию предметной области и поможет в дальнейшей разработке логики программы.

**1.3. Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований**

При выполнении данного курсового проекта можно выделить 2 основные задачи: изучение работы кинотеатра и разработка программы для онлайн-продажи билетов.

В ходе решения второй задачи можно выделить следующие пункты:

* разработать приложение на объектно-ориентированном языке;
* приложение должно быть выполнено в архитектуре клиент-сервер;
* бизнес-логика системы должна быть реализована только на серверной части приложения;
* построить функциональную модель системы в IDEF0;
* смоделировать информационную систему с помощью стандарта UML;
* данные должны храниться в базе данных, связь с базой данных осуществляется на серверной части;
* обеспечить систематизацию данных, хранящихся в базе;
* обеспечить ограниченный доступ к базе данных, содержащей данные клиентов.

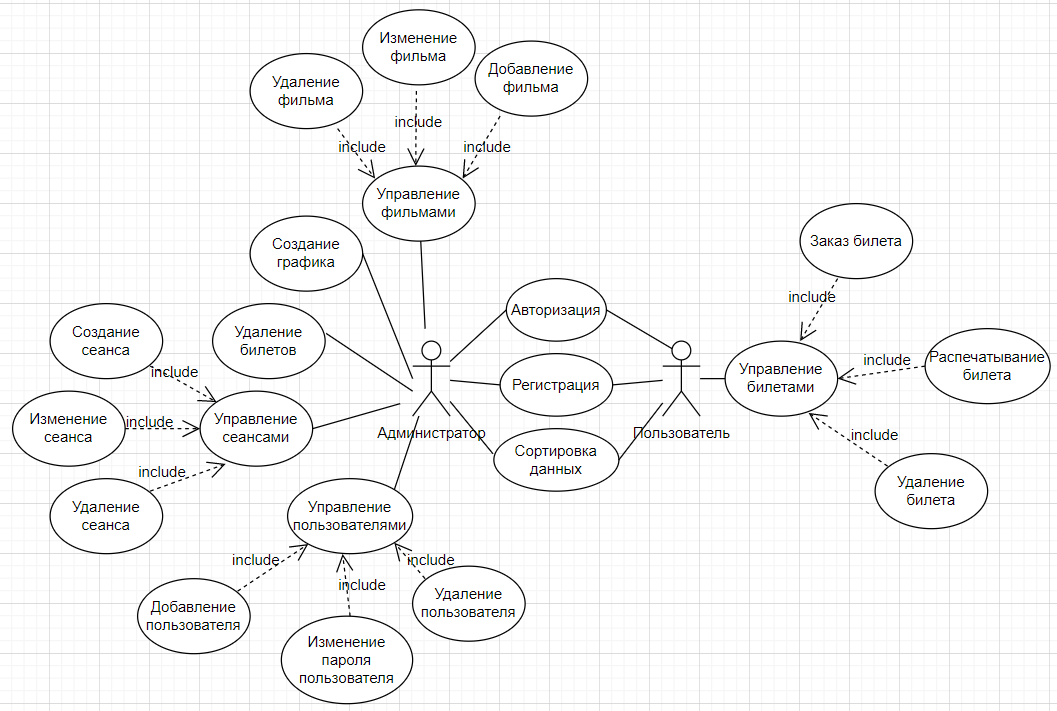


Рисунок 1.4 — Диаграмма вариантов использования

В программе также должны быть реализованы основные функции и действия:

* регистрация (при первом использовании приложения пользователю необходимо зарегистрироваться, чтобы ему был доступен весь остальной функционал). При попытке зарегистрировать пользователя с уже существующим логином будет выброшена исключительная ситуация.
* авторизация (будет выброшена исключительная ситуация, если такого пользователя не существует). При авторизации администратора необходимо ввести логин: admin и пароль: admin, после чего Вам откроется весь функционал системы. Если же вы авторизуетесь как обычный пользователь данные для ввода будут следующие: логин: user1 и пароль: user1. И соответственно Ваш функционал будет гораздо меньше;
* добавление фильма (данный метод доступен только администратору). Необходимо заполнить соответствующую форму, чтобы добавить новый фильм в кинотеатр;
* удаление фильма (данный метод доступен только администратору). Необходимо указать фильм, информацию о котором Вы хотите удалить;
* изменение фильма (данный метод доступен только администратору). Необходимо указать фильм, информацию о котором Вы хотите изменить и заполнить форму с новыми данными;
* создание сеанса (данный метод доступен только администратору). Необходимо заполнить соответствующую форму, чтобы добавить новый сеанс указанного фильма;
* изменение сеанса (данный метод доступен только администратору). Необходимо указать сеанс, информацию о котором Вы хотите изменить и заполнить форму с новыми данными;
* удаление сеанса (данный метод доступен только администратору). Необходимо указать сеанс, информацию о котором Вы хотите удалить;
* заказ билета (доступен только пользователю). Необходимо выбрать сеанс, на который Вы хотите приобрести билет. После купленный Вами билет появится во вкладке Билеты;
* удаление билета (данный метод доступен всем). Необходимо указать билет, который Вы хотите удалить;
* печать билет (данный метод доступен только пользователю). Необходимо указать билет, который Вы хотите распечатать;
* сортировка (данный метод доступен всем). По нажатию на колонку таблицы все данные будут отсортированы либо в алфавитном порядке, либо по убыванию или возрастанию указанного параметра;
* формирование статистики стран-производителей (доступен только пользователю). Данный метод предоставляет возможность получить информацию об основных производителях спортивного питания в мире.

**1.4. Разработка информационной модели предметной области**

Информационная модель представляет объекты в виде таблиц. Она содержит массу информации, которая описывает свойства объекта, его связи и отношения с другими объектами моделирования.

Чтобы построить информационную модель необходимо проанализировать объект моделирования как сложную систему, а также определить цели моделирования.

В данном курсовом проекте для хранения, обработки и использования информации используется одна из наиболее популярных систем управления базами данных PostgreSQL.

Для представления структуры хранимых данных программного средства выбрана информационная методология IDEF1X.

IDEF1X является методом для разработки реляционных баз данных и использует условный синтаксис, специально разработанный для удобного построения концептуальной схемы. Использование метода IDEF1X наиболее целесообразно для построения логической структуры базы данных после того, как все информационные ресурсы исследованы и решение о внедрении реляционной базы данных, как части корпоративной информационной системы, было принято. Однако не стоит забывать, что средства моделирования IDEF1X специально разработаны для построения реляционных информационных систем, и если существует необходимость проектирования другой системы, скажем объектно-ориентированной, то лучше избрать другие методы моделированиях.

Основными компонентами в методике IDEF1X являются сущности, отношения и атрибуты.

Сущность определяют как множество объектов, обладающих общими свойствами. Конкретные элементы этого множества называют экземплярами сущности.

Отношения между сущностями в IDEF1X являются бинарными отношениями. Выделяют идентифицирующие отношения – связи типа родитель-потомок, в которых потомок (зависимая сущность) однозначно определяется своей связью с родителем, и не идентифицирующие отношения, означающие, что у связанного этим отношением экземпляра одной сущности может быть, а может и не быть соответствующего экземпляра второй сущности.

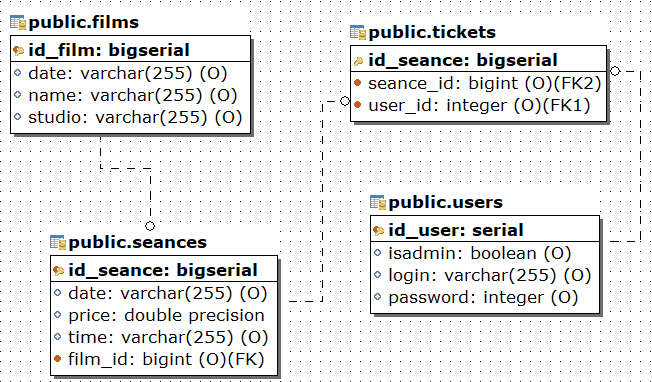
Идентифицирующее отношение изображают на IDEF1X-диаграмме сплошной линией между прямоугольниками связанных сущностей, не идентифицирующее отношение показывают пунктирной линией. На дочернем конце линии должно быть утолщение (жирная точка).

Различают также специфические и неспецифические отношения. Специфические отношения — это связи "один ко многим", а неспецифические — связи типа "многие ко многим".

Свойства сущностей называют атрибутами. Различают ключевые и неключевые атрибуты. Значение ключевого атрибута (ключа) однозначно идентифицирует экземпляр сущности. Ключевые атрибуты могут быть составными.

Внешний ключ — это атрибут, входящий в ключ родителя и наследуемый потомком. На IDEF1X-диаграммах ключи записывают в верхней части прямоугольника сущности, причем внешние ключи помечают меткой FK (Foreign Key), неключевые атрибуты помещают в нижнюю часть прямоугольников. В идентифицирующих отношениях все ключи родителя входят и в ключи потомка, в неидентифицирующих ключи родителя относятся к неключевым атрибутам потомка.

Информационная модель программного средства представлена на рисунке 1.5.

Рисунок 1.5 — Информационная модель

Информационная модель программного средства состоит из следующих сущностей: public.films, public.seances, public.tickets, public.users.

Сущность «public.films» предназначена для хранения данных о фильмах, которые есть в системе. Данная сущность содержит поля: «id\_film», «date», «name», «studio». Поле «id\_film» является первичным ключом сущности.

Сущность «public.seances» предназначена для хранения информации о доступных сеансах. Данная сущность содержит поля: «date», «price», «time». Поле «id\_seance» является первичным ключом сущности и имеет внешний ключ на «film\_id» из таблицы «public.films».

Сущность «public.tickets» предназначена для хранения информации о билетах. Данная сущность содержит поля: «id\_seance», «user\_id», «seance\_id». Поля «seance\_id» и «user\_id» являются внешними ключами сущности, а поле «id\_seance» — первичным ключом.

Сущность «public.users» предназначена для хранения данных о человеке, который пользуется системой. . Данная сущность содержит поля: «id\_user», «isadmin», «login», «password». Поле «id\_user» является первичным ключом сущности.

**1.5. UML-модели представления программного средства и их описание**

Унифицированный язык моделирования (UML) является стандартным инструментом для создания «чертежей» информационных систем (ИС). С помощью UML можно визуализировать, специфицировать, конструировать и документировать элементы этих систем.

UML пригоден для моделирования любых систем: от информационных систем масштаба предприятия до распределенных Web-приложений и даже встроенных систем реального времени. Это очень выразительный язык, позволяющий рассмотреть систему со всех точек зрения, имеющих отношение к ее разработке и последующему развертыванию. Несмотря на обилие выразительных возможностей, этот язык прост для понимания и использования. Изучение UML удобнее всего начинать с его концептуальной модели, которая включает в себя три основных элемента: базовые строительные блоки, правила, определяющие, как эти блоки могут сочетаться между собой, и некоторые общие механизмы языка.

UML является одной из составляющих процесса разработки ИС. Хотя UML не зависит от моделируемой реальности, лучше всего применять его, когда процесс моделирования основан на рассмотрении прецедентов использования, является итеративным и пошаговым, а сама система имеет четко выраженную архитектуру.

UML – это язык для визуализации, специфицирования, конструирования и документирования элементов программных систем. Язык состоит из словаря и правил, позволяющих комбинировать входящие в него слова и получать осмысленные конструкции. В языке моделирования словарь и правила ориентированы на концептуальное и физическое представление системы. Язык моделирования, подобный UML, является стандартным средством для составления «чертежей» ИС.

Словарь UML включает три вида строительных блоков:

* Диаграммы;
* Сущности;
* Связи.

Диаграмма последовательностей относится к диаграммам взаимодействия UML, описывающим поведенческие моменты системы, взаимодействие объектов в течение определённого отрезка времени.

Диаграмма последовательности отражает поток событий, происходящих в рамках варианта использования. На этой диаграмме изображаются только те объекты, которые непосредственно участвуют во взаимодействии так как ключевым моментом является именно динамика взаимодействия объектов во времени и не используются возможные статистические ассоциации с другими.

Все действующие лица показаны в верхней части диаграммы. Стрелки соответствуют сообщениям, передаваемым между действующим лицом и объектом или между объектами для выполнения требуемых функций.

Как было сказано выше, взаимодействие между актёрами отображается при помощи специальных стрелок, передающих управление от отправителя (от кого идёт стрелка) к получателю (тот, к кому направлена стрелка). Стрелки демонстрируют ход сценария и те события, которые происходят во время анализируемого прецедента. Всего существует 5 видов стрелок:

* синхронное сообщение – актёр-отправитель передаёт ход управления актёру-получателю, которому необходимо провести в прецеденте некоторое действие. Пока проводимое актёром-получателем действие не будет завершено (соответственно, не будет получено ответное сообщение), актёр-отправитель теряет возможность производить какие-либо действия. Графически изображается как сплошная линия со стрелкой в виде закрашенного треугольника, после которой идёт прямоугольник, отражающий деятельность объекта, в конце которого находится ответное сообщение;
* ответное сообщение – данное сообщение является ответом на синхронное сообщение. Обычно, содержит какое-либо возвращаемое изначальному актёру-отправителю значение, также возвращающее ему управление (возможность действовать). Графически изображается пунктирной линией с открытой стрелкой;
* асинхронное сообщение – актёр-отправитель передаёт ход управления актёру-получателю, которому необходимо провести в прецеденте некоторое действие. Основное отличие от синхронного сообщения состоит в том, что актёр-отправитель не теряет возможности совершать другие действия. Графически изображается сплошной линией с открытой стрелкой.
* потерянное сообщение – сообщение без адресата (есть отправитель, нет получателя);
* найденное сообщение – сообщение без отправителя.

Диаграмма последовательности процесса онлайн-покупки билета предоставлена на рисунке 1.6.

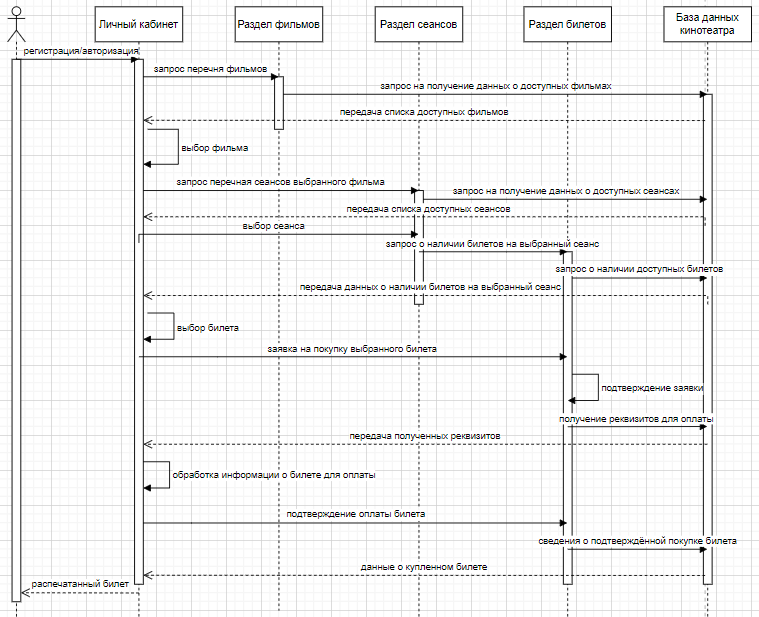
Рисунок 1.6 – Диаграмма последовательности процесса онлайн-покупки билета

Диаграмма последовательности процесса онлайн-покупки билета состоит из следующих объектов: пользователь, личный кабинет, раздел фильмов, раздел сеансов, раздел билетов, база данных кинотеатра.

Пользователь регистрируется/авторизируется в личном кабинете, после чего отправляет запрос на получение перечня доступных фильмов базе данных кинотеатра через раздел фильмов. Полученный от базы данных список раздел фильмов отправляет в личный кабинет, в котором происходит выбор фильма и отправка запроса о доступных сеансах. Далее после получения списка доступных сеансов через раздел билетов идёт запрос о наличии билетов на выбранный сеанс. Затем оформляется заявка на покупку выбранного билета, из базы данных поступают нужные для оплаты реквизиты и производится оплата. Пользователь из личного кабинета подтверждает оплату и получает данные о купленном билете из базы данных кинотеатра, а также возможность распечатать купленный билет.

Диаграмма состояний – это [диаграмма](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D0%B9_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%BE%D0%B2)) из теории автоматов со стандартизированными условными обозначениями, которая может определять множество систем от компьютерных программ до [бизнес-процессов](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81).

Используются следующие условные обозначения:

* круг, обозначающий начальное состояние;
* окружность с маленьким кругом внутри, обозначающая конечное состояние;
* скруглённый прямоугольник, обозначающий состояние. Верхушка прямоугольника содержит название состояния. В середине может быть горизонтальная линия, под которой записываются активности, происходящие в данном состоянии;
* стрелка, обозначающая переход. Название события, вызывающего переход, отмечается рядом со стрелкой;
* толстая горизонтальная линия с либо множеством входящих линий и одной выходящей, либо одной входящей линией и множеством выходящих. Это обозначает объединение и разветвление соответственно.

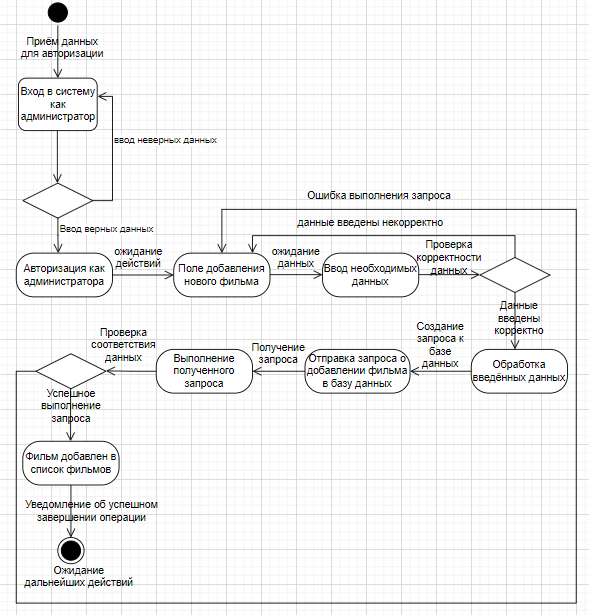
Диаграмма состояний процесса «Добавление фильма» представлена на рисунке 1.7.

Рисунок 1.7 — Диаграмма состояний процесса «Добавление фильма»

Диаграмма компонентов – это диаграмма классов, которая фокусируется на компонентах системы, которые часто используются для моделирования представления статической реализации системы.

Грамотное использование диаграммы компонентов позволяет:

– представить физическую структуру компонентов;

– обратить внимание на компоненты системы и как они связаны;

– акцентировать внимание на поведение системы в части, касающейся интерфейса.

С помощью данной диаграммы можно определить архитектуру разрабатываемой системы, а также просмотреть зависимости между программными компонентами.

Диаграмма развертывания ‒ это UML-диаграмма, которая отражает архитектуру системы, включая такие узлы, как программные или аппаратные среды исполнения, а также промежуточное программное обеспечение, соединяющее их. Она показывает топологию системы и распределение компонентов системы по ее узлам, а также соединения-маршруты передачи информации между аппаратными узлами. Данная диаграмма содержит графические изображения процессоров, устройств, процессов и связей между ними. Она является единой для системы в целом. Предназначена для визуализации элементов и компонентов программы, существующих на этапе ее выполнения, определяет распределение компонентов системы по ее физическим узлам, показывает физические связи между всеми узлами реализации системы, выявляет узкие места системы и реконфигурирует ее топологию для достижения требуемой производительности. Разработка данных диаграмм необходима для визуализации программной структуры.

Диаграмма развёртывания представлена на рисунке 1.8.

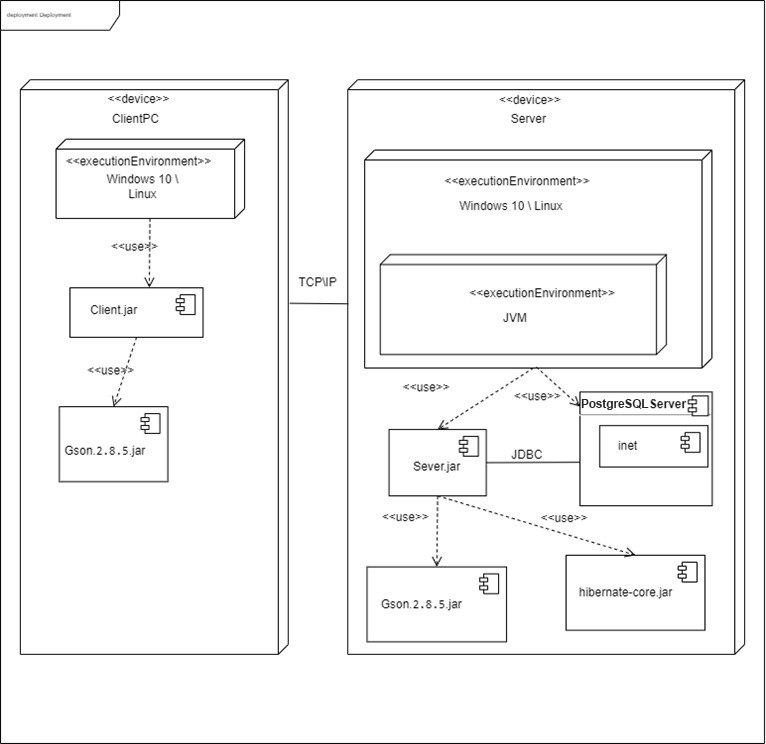


Рисунок 1.8 — Диаграмма развёртывания и компонентов

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

Целью создания диаграммы классов является графическое представление статической структуры декларативных элементов системы (классов, [типов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и т. п.). Она содержит в себе также некоторые элементы поведения (например – операции), однако их динамика должна быть отражена на диаграммах других видов ([диаграммах коммуникации](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BC%D1%83%D0%BD%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8), диаграммах состояний).  При представлении сущностей реального мира разработчику требуется отразить их текущее состояние, их поведение и их взаимные отношения.

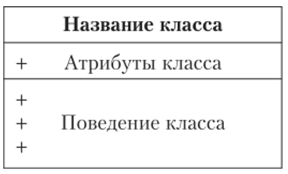


Рисунок 1.9 – Изображение класса в UML

Из рисунка 1.9 видно, что класс изображается в UML в виде прямоугольника, состоящего из трех разделов. Верхний раздел содержит название класса, средний – атрибуты класса, а нижний – варианты поведения класса.

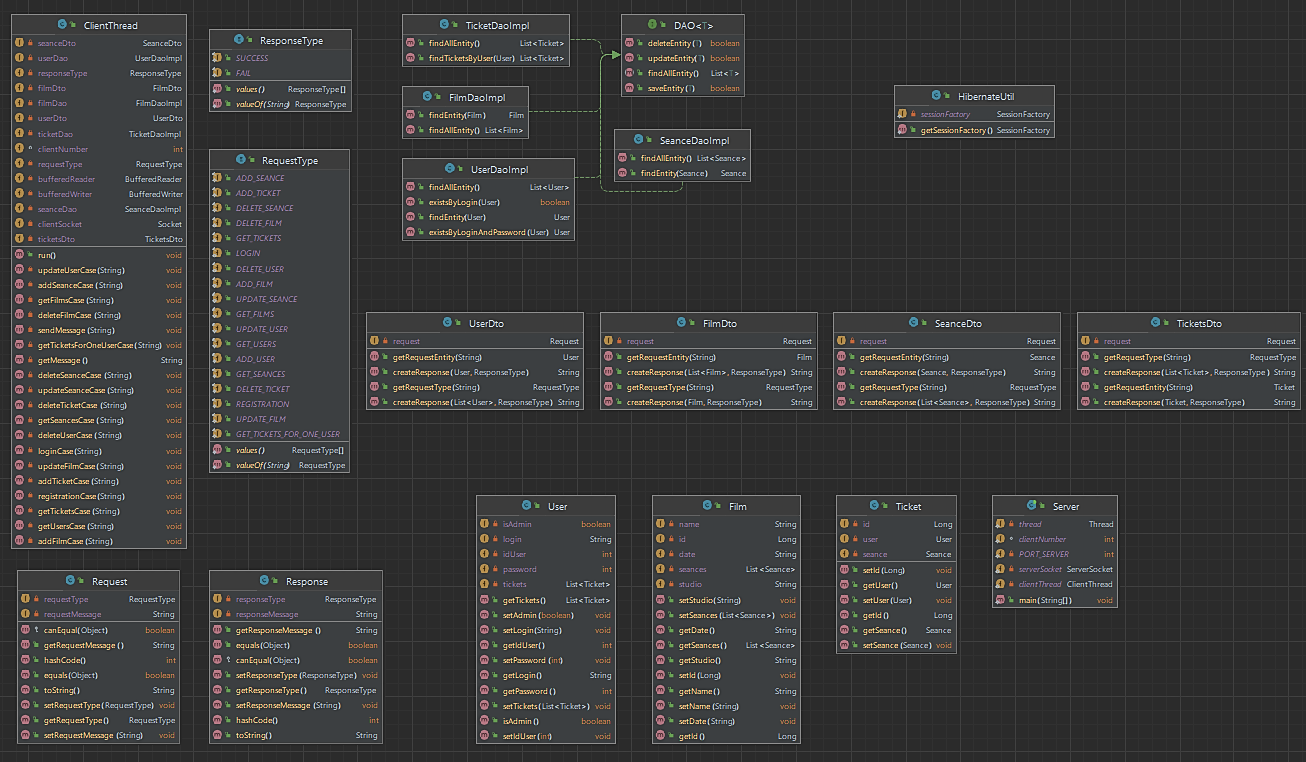
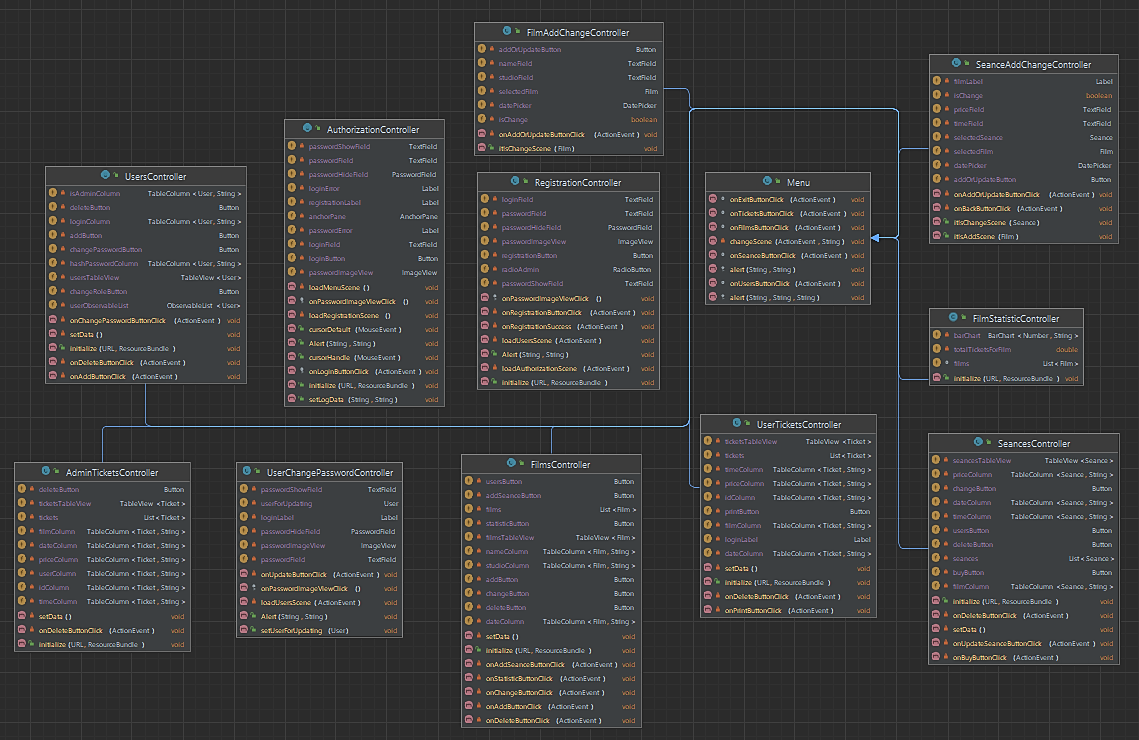
Диаграмма классов серверного программного средства представлена на рисунке 1.10.

Рисунок 1.10 — Диаграмма классов серверного программного средства

На рисунке 1.11 представлена диаграмма основных классов-контроллеров приложения.

Рисунок 1.11 – Диаграмма основных классов-контроллеров

В данном главе рассмотрен унифицированный язык UML и его применение и описано построение моделей представления системы. Разработанные диаграммы упрощают дальнейшую разработку приложения и самого кода программы. В данных диаграммах визуально отображены модели представления системы логистической компании, что помогает программисту при разработке приложения.

**2 Проектирование и конструирование ПРОГРАММНОГО СРЕДСТВА**

* 1. **Постановка задачи**

Задача данной курсовой работы состоит в разработке web-сервиса на языке Java для онлайн-продажи билетов в кинотеатр. Разработанный web-сервис направлен на ускорение оформления пользователя, а также управление процессом продажи билетов.

**2.2. Обоснование выбора компонентов и технологий для реализации программного средства**

Java — это популярный язык программирования высокого уровня, который был выпущен компанией Sun Microsystems в 1995 году. Он разработан для независимости от платформы, что означает, что код, написанный на Java, может работать на любом устройстве или операционной системе, на которых установлена виртуальная машина Java (JVM). Java — это мощный и универсальный язык программирования, который широко принят в программной индустрии. Он продолжает развиваться, с добавлением новых функций и улучшений в каждом новом выпуске.

Одной из ключевых особенностей Java является ее сильный упор на концепции объектно-ориентированного программирования (ООП). В Java все рассматривается как объект, и программы написаны с использованием классов и объектов.

Некоторые из ключевых особенностей Java включают автоматическое управление памятью (сбор мусора), обработку исключений и возможности многопоточности. Java также имеет обширную библиотеку предварительно созданных классов и API, которые упрощают выполнение общих задач, таких как сетевое взаимодействие, ввод-вывод файлов и доступ к базам данных.

IntelliJ IDEA — это интегрированная среда разработки (IDE) для языков программирования Java, Kotlin и других. Она была разработана компанией JetBrains и выпущена в 2001 году.

IntelliJ IDEA предоставляет широкий набор инструментов для разработки приложений на языках Java и Kotlin. Она включает в себя поддержку автодополнения, статический анализ кода, интеграцию с системами контроля версий, инструменты для рефакторинга кода, отладчик и многие другие функции.

Клиент реализует пользовательский интерфейс. Для этого используется *JavaFX*. *JavaFX* представляет инструментарий для кроссплатформенных графических приложений на базе Java. Бизнес-логика курсового проекта выполняется на серверной части. Клиент отправляет запросы на сервер, где с помощьюHibernate будет добавляться, обновляться, удаляться и выбираться вся необходимая информация из БД. Hibernate – это ORM фреймворк для Java с открытым исходным кодом. Эта технология является крайне мощной и имеет высокие показатели производительности.

Hibernate создаёт связь между таблицами в базе данных (далее – БД) и Java-классами и наоборот. Это избавляет от написания запросов путем создания объектов, данный процесс показан на рисунке 2.1.



Рисунок 2.1 – Взаимодействие Hibernate c приложением Java

Hibernate это фреймворк, предназначенный для задач объектно-реляционного отображения. Данная технология позволяет работать с базой данных не напрямую, а с помощью представления таблиц базы данных как классов Java.

Hibernate представляет собой свободное программное обеспечение с открытым исходным кодом. Используется для сокращения повторяющегося кода, позволяет явно не использовать JDBC, позволяет сделать реализацию кода независимой.

Работа с данными происходит через транзакции, что обеспечивает корректное взаимодействие, так как в случае ошибки происходит полный откат транзакции до первоначального состояния без внесения каких-либо изменений.

При разработке данной автоматизированной системы использована СУБД PostgreSQL, так как она является оптимальным решением для построения сложных систем с большим числом запросов.

**2.3. Архитектурные решения**

Автоматизированная система контроля экспорта товаров будет построена в соответствии с архитектурным паттерном MVC.

MVC – это архитектурный паттерн, который разделяет данные, представление и обработку действий пользователя на компоненты.

Концепция MVC представлена на рисунке 2.2.

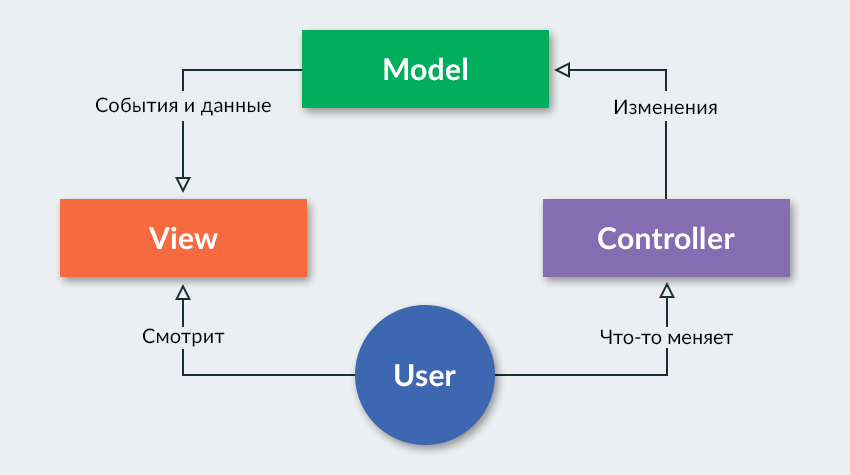


Рисунок 2.2 – Концепция MVC

Модель (Model) представляет собой сущность некой предметной области, которая включает в себя данные и методы работы с этими данными. Модель реагирует на запросы из контроллера, возвращая данные или изменяя своё состояние. При этом модель не содержит в себе информации о способах визуализации данных или форматах их представления, а также не взаимодействует с пользователем напрямую.

Представление (View) отвечает за визуализацию данных. Одни и те же данные могут представляться различными способами и в различных форматах. Например, коллекцию объектов при помощи разных представлений можно представить на уровне пользовательского интерфейса как в табличном виде, так и списком; на уровне API можно экспортировать в различные форматы данных.

Контроллер (Controller) обеспечивает связь между пользователем (User) и системой, использует модель и представление для реализации необходимой реакции на действия пользователя. Как правило, на уровне контроллера осуществляется фильтрация полученных данных и авторизация.

За счет такого разделения повышается возможность повторного использования программного кода: например, добавить представление данных какого-либо существующего маршрута в иных форматах становится очень просто и не требует изменений слоя бизнес-логики исходного маршрута. Также упрощается и сопровождение программного кода: внесение изменений во внешний вид не отражается на бизнес-логике, а изменения бизнес-логики не затрагивают визуализацию.

## **2.4. Описание алгоритмов, реализующих ключевую бизнес-логику**

**разрабатываемого программного средства**

Схема алгоритма работы приложения при входе с ролью администратора представлена в Приложении Б.

Из неё видно, что сначала производится авторизация, затем, если она прошла успешно, администратор может работать с фильмами, сеансами, билетами и пользователями.

При работе с фильмами, сеансами и пользователями администратору доступно добавление, изменение и удаление выбранных фильмов или пользователей, а также создание, изменение и удаление сеансов. В итоге все изменения производятся в базе данных.

При работе с билетами администратору доступно только удаление выбранных билетов.

Одним из самых главных алгоритмов программы является установка соединения с сервером, который представлен на рисунке 2.3.

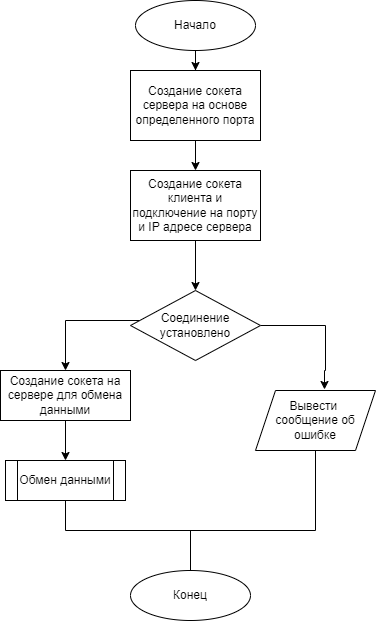
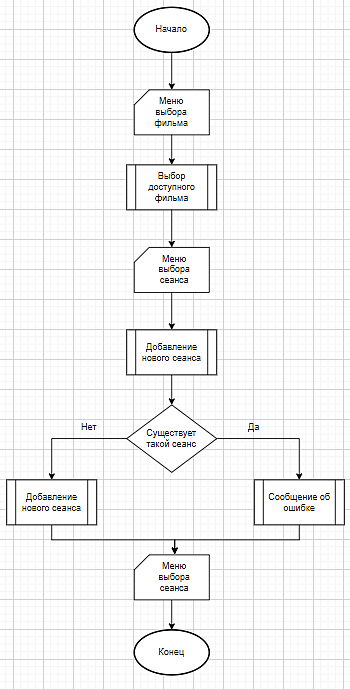
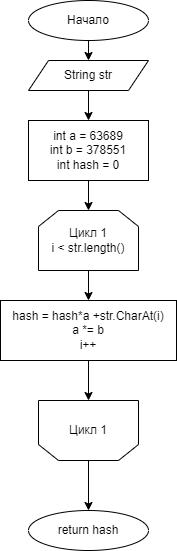


Рисунок 2.3 – Алгоритм установки соединения

Алгоритм добавления нового сеанса представлен на рисунке 2.4. Происходит создание сеанса для выбранного фильма путём указания стоимости билета, даты и времени показа. Если такого сеанса ещё нет в системе, он добавляется.

Рисунок 2.4 – Алгоритм добавления ****нового сеанса

Для шифрования паролей был использован алгоритм хэширования Роберта Седжвика, являющийся хэш-функцией (рисунок 2.5). Хэширование представляет собой преобразование любого объема информации в уникальный набор символов, который присущ только этому массиву входящей информации. Этот набор символов и будет называться хэшем.

Рисунок 2.5 – Алгоритм хэширования Роберта Седжвика

В данной подглаве были описаны алгоритмы работы программы, для лучшего ее понимая.

**2.5. Проектирование пользовательского интерфейса**

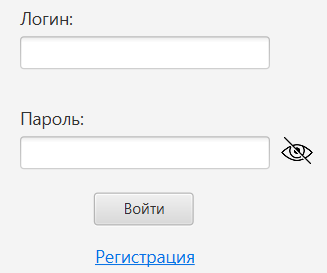
Для начала работы пользователю необходимо войти в систему. Для этого на экране входа в систему требуется ввести логин и пароль, а затем нажать кнопку «Войти» (рисунок 2.3). Если же введенные логин и пароль оказываются недействительными, то выбрасывается исключительная ситуация.

Рисунок 2.3 – Экран входа в систему

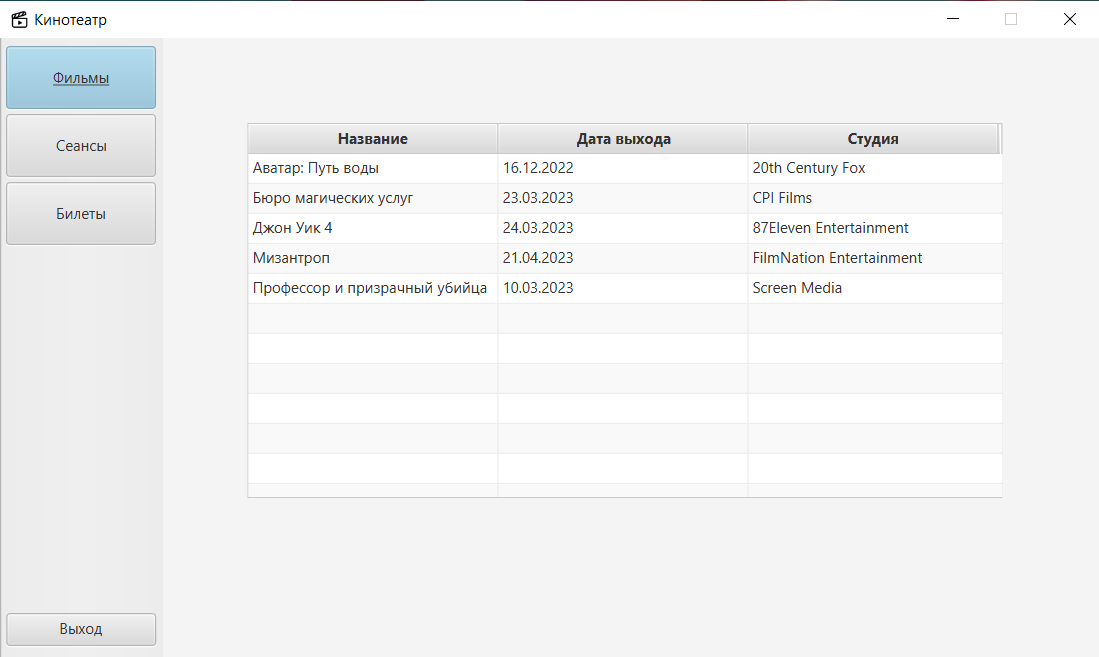
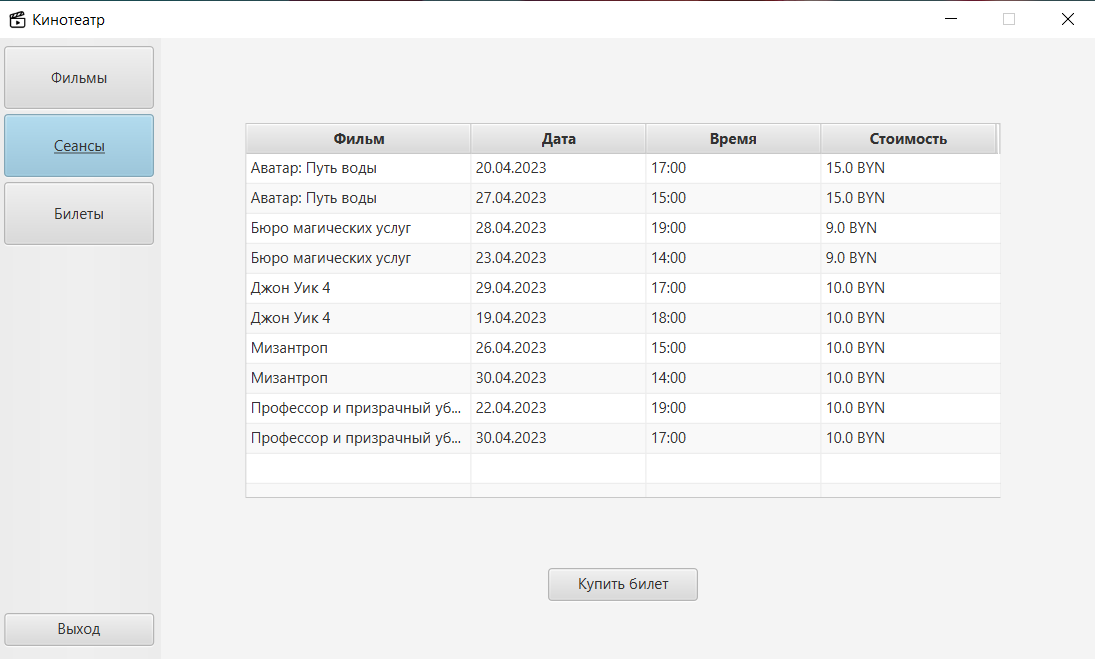
После выполнения процедуры авторизации пользователь попадает на экран просмотра списка доступных фильмов (рисунок 2.4).

Рисунок 2.4 – Экран просмотра списка фильмов

Нажав на пункт меню «Сеансы», пользователь попадает на экран просмотра списка сеансов (рисунок 2.5).

Рисунок 2.5 – Экран просмотра списка сеансов

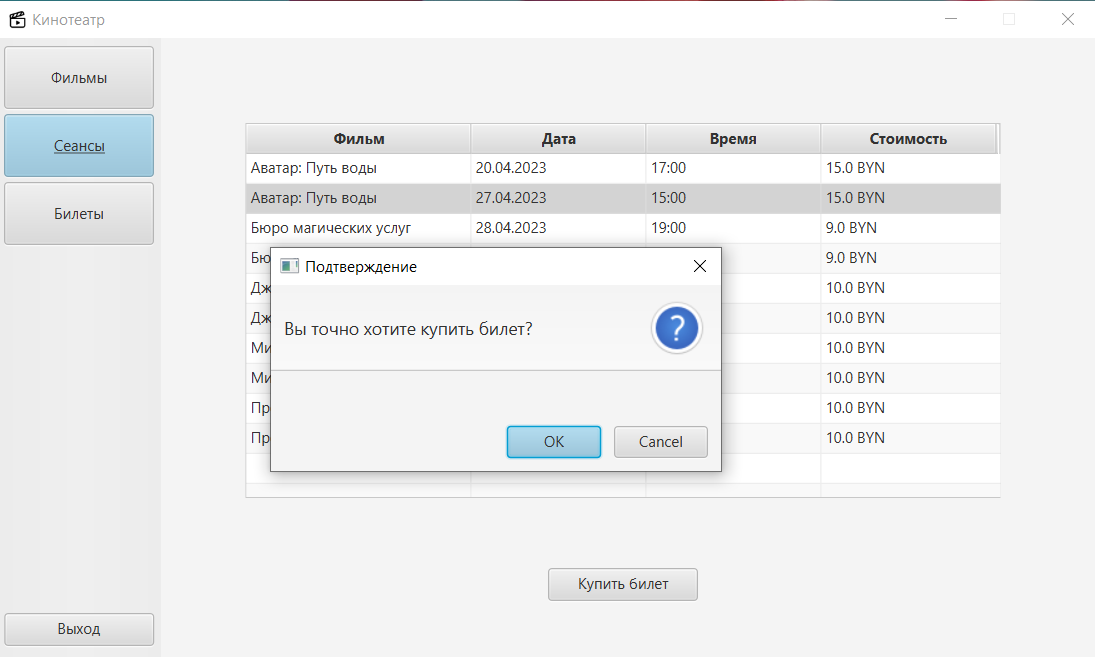
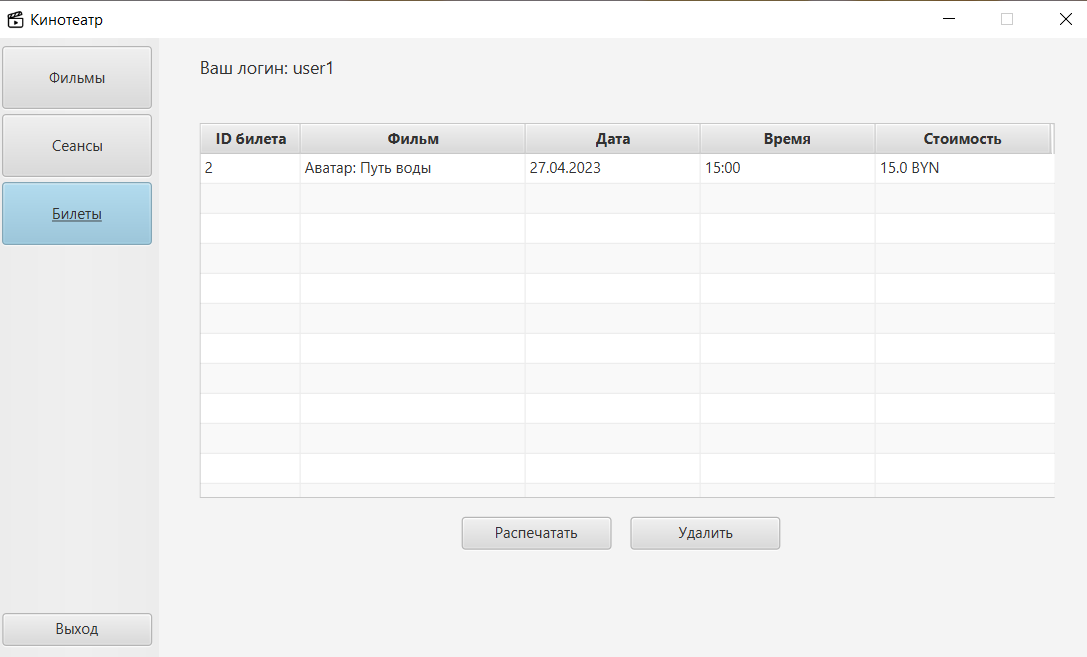
Выбрав из списка сеансов нужный, пользователь может приобрести билет на выбранный сеанс (рисунок 2.6).

Рисунок 2.6 – Экран покупки билета

После удачной покупки билета пользователь, нажав на пункт меню «Билеты», перенаправляется на экран со списком билетов (рисунок 2.7).

Рисунок 2.7 – Экран просмотра списка билетов

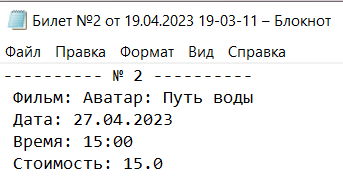
На экране просмотра списка билетов отображается список купленных билетов с возможностью удалить их, а также распечатать конкретный билет в формате txt (рисунок 2.8).

Рисунок 2.8 – Открытый TXT файл купленного билета

Завершение текущей сессии происходит с помощью нажатия кнопки «Выход», после чего пользователь будет снова перенаправлен на страницу входа.

**3 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ ПРОГРАММЫ**

Для ведения учета, регистрации и онлайн-продажи билетов в кинотеатр система должна успешно работать и быть защищенной от ошибок. Любая ошибка данной системы будет причиной потери денег компании. В случае ошибок ввода или некорректной работы с системой должны выводиться сообщения с предупреждениями или исключительные ситуации.

Данная система не должна предоставлять доступ неподтверждённым пользователям. Если введенный логин и пароль есть в базе данных, то вход выполнен успешно, в противном случае вылетит исключительная ситуация.

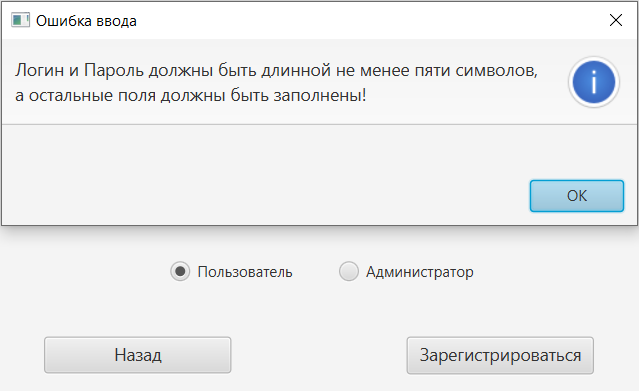
При регистрации все поля являются обязательными для заполнения. В случае некорректного ввода информации или её отсутствия в одном из полей невозможно обратиться к серверу, а значит и пройти регистрацию. Также при регистрации учитывается минимальный размер логина и пароля, в случае некорректного ввода система выдаст предупреждение (рисунок 3.1).

Рисунок 3.1 – Некорректный ввод информации для регистрации

При добавлении новых фильмов или сеансов недопустим ввод некорректных данных, чтобы система не выдавала ошибок при дальнейшем её использовании. При вводе информации все поля должны быть заполнены в соответствии с допустимым форматом (рисунок 3.2 и рисунок 3.3).

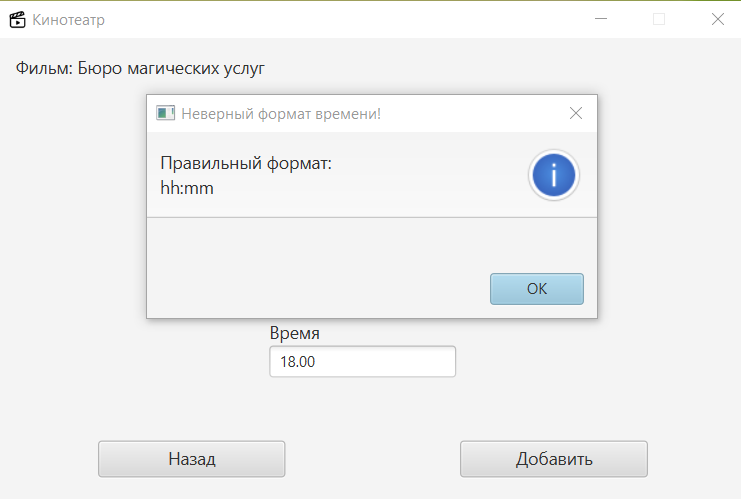
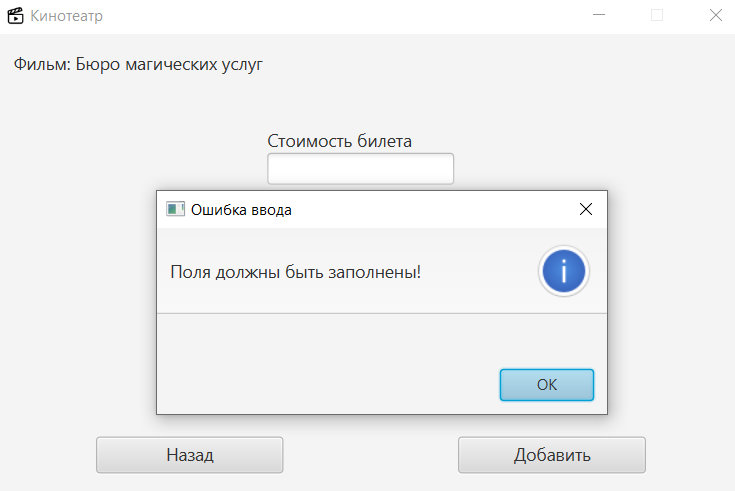
Рисунок 3.2 – Некорректный ввод информации о времени сеанса

Рисунок 3.3 – Предупреждение в случае незаполненного поля

Результаты тестирования показывают, что программное средство удовлетворяет функциональным требованиям и работает корректно.

**4 РУКОВОДСТВО ПО РАЗВЁРТЫВАНИЮ И ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА**

## **4.1. Руководство по установке (развёртыванию) программного средства**

Для возможности корректного использования системы требуется наличие на компьютере 32х или 64ч разрядной операционной системы Windows 7 и выше. Данный программный продукт использует PostgreSQL для работы с базой данных. Также необходимо наличие на компьютере для клиента и сервера JDK версии 15 и выше.

Для запуска приложения необходимо последовательно запустить из консоли server.jar, а затем client.jar. При первом запуске приложения будет создана база данных, если до этого не была создана, то она сгенерируется автоматически с помощью SQL-скрипта.

В результате описанных выше действий пользователь может корректно запустить данное программное приложение на любом удовлетворяющем требованиям устройстве.

## **4.2. Руководство пользователя**

В первую очередь необходимо запустить серверную часть приложения, затем в ней происходит соединение с клиентами. Если клиент впервые запускает приложение ему следует зарегистрироваться (рисунок 4.1).

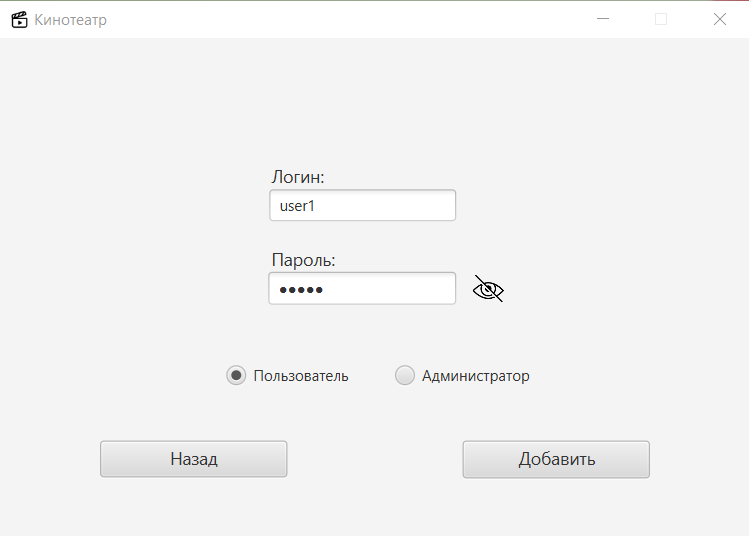


Рисунок 4.1 – Регистрация пользователя

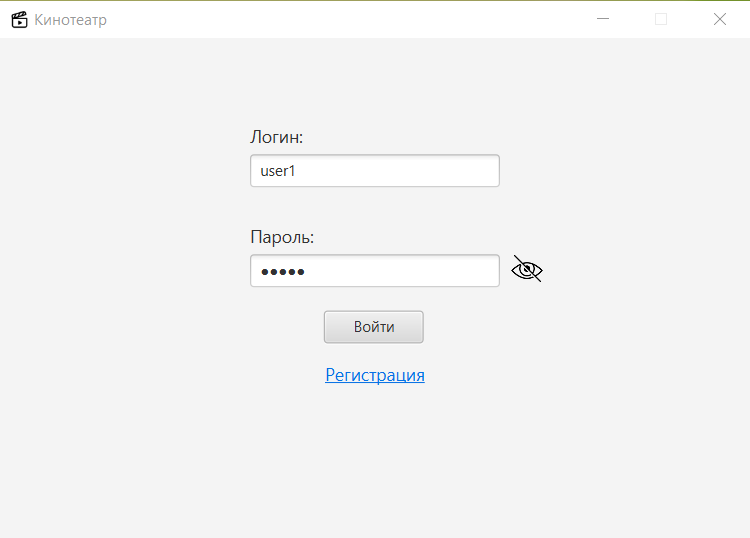
 Если регистрация прошла успешно, то клиента выкинет на окно с авторизацией, где уже будут введены логин и пароль, которые он вводил при регистрации (рисунок 4.2).

Рисунок 4.2 – Авторизация пользователя

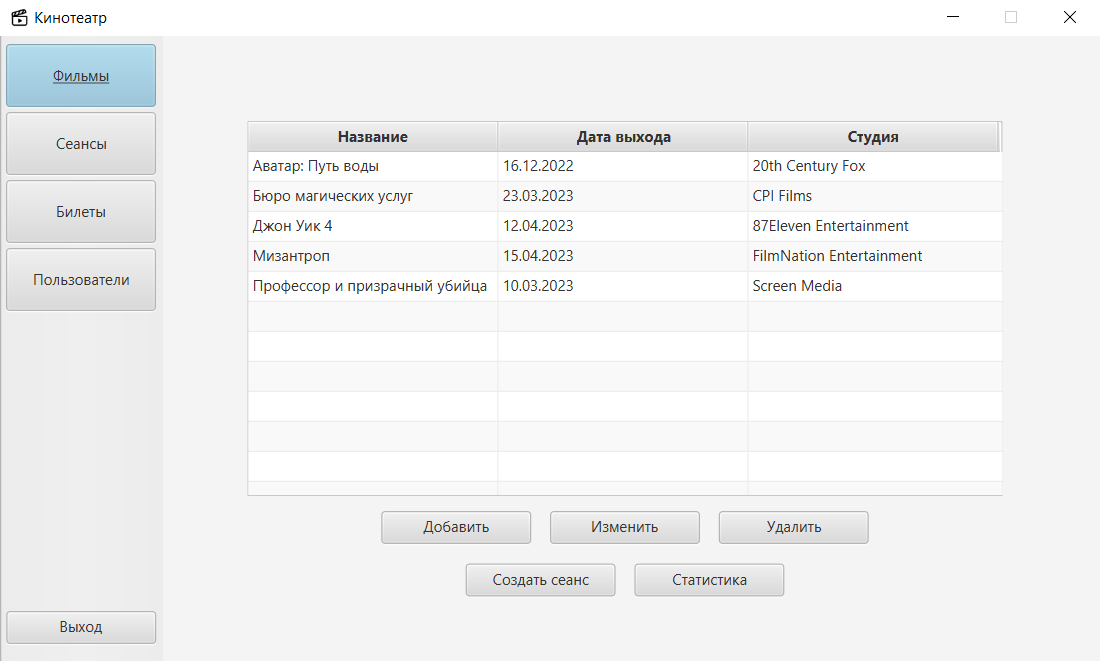
 Сначала рассмотрим возможности Администратора. Если авторизация прошла успешно, то на экране появится меню для управления фильмами (рисунок 4.3).

Рисунок 4.3 – Меню администратора для управления фильмами

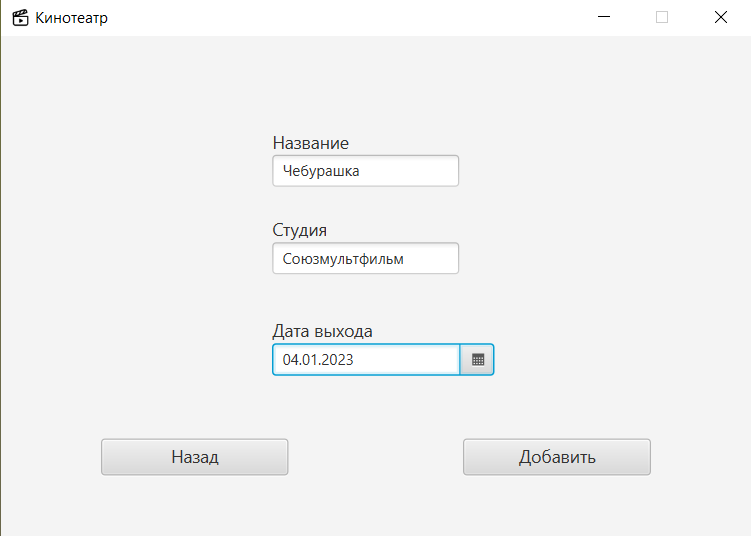
На рисунке 4.3 показано, что администратору доступны CRUD операции над фильмами, создание сеансов и просмотра статистики. Также в этом меню видны студии-создатели фильмов и дата их выхода в прокат. Процесс добавления нового фильма показан на рисунке 4.4.

Рисунок 4.4 – Добавление нового фильма

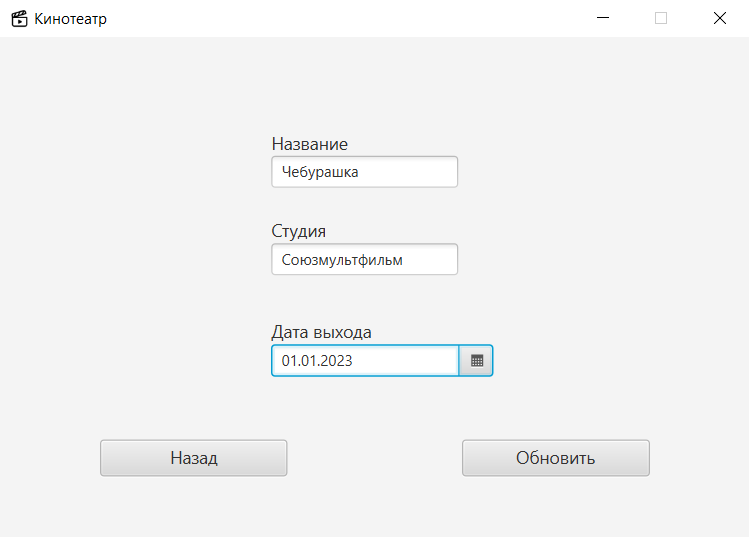
На рисунке 4.5 показано как выглядит изменение данных о выбранном фильме.

Рисунок 4.5 – Изменение данных о фильме

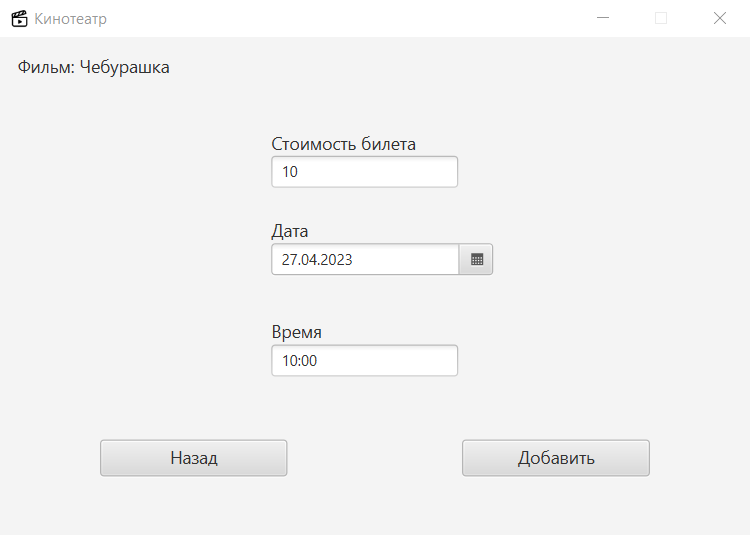
На рисунке 4.6 показано как выглядит создание сеанса выбранного фильма. Для добавления сеанса нужно указать цену билета на сеанс, а также дату и время показа.

Рисунок 4.6 – Создание нового сеанса

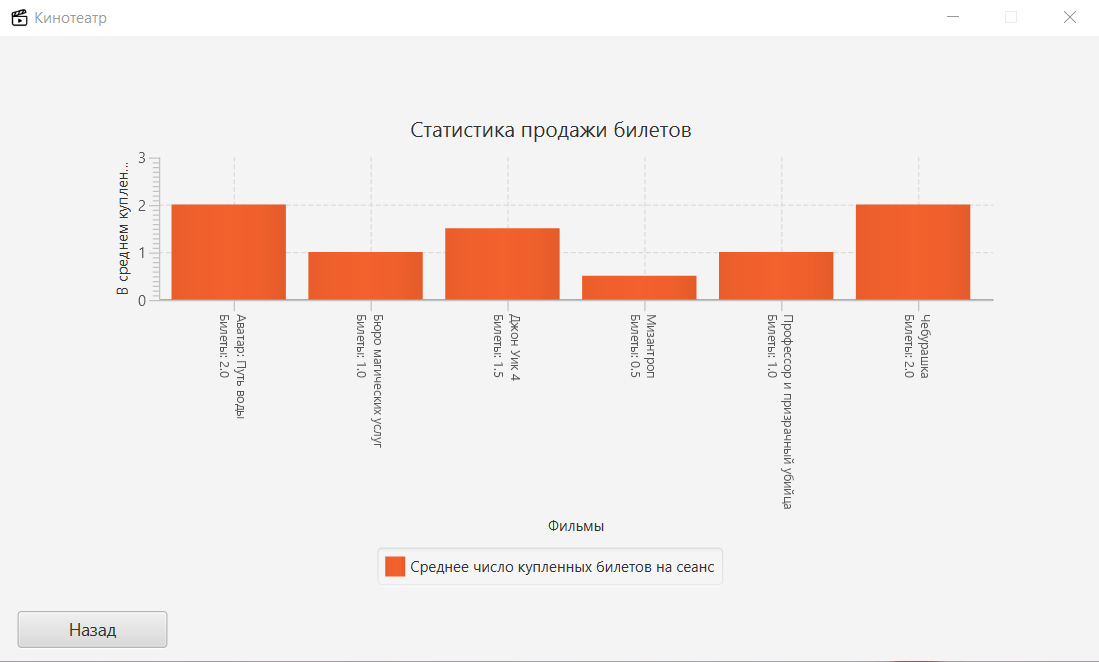
На рисунке 4.7 показано, что администратору доступен просмотр статистики — среднее число купленных билетов на сеанс.

Рисунок 4.7 – Просмотр статистики среднего числа купленных билетов

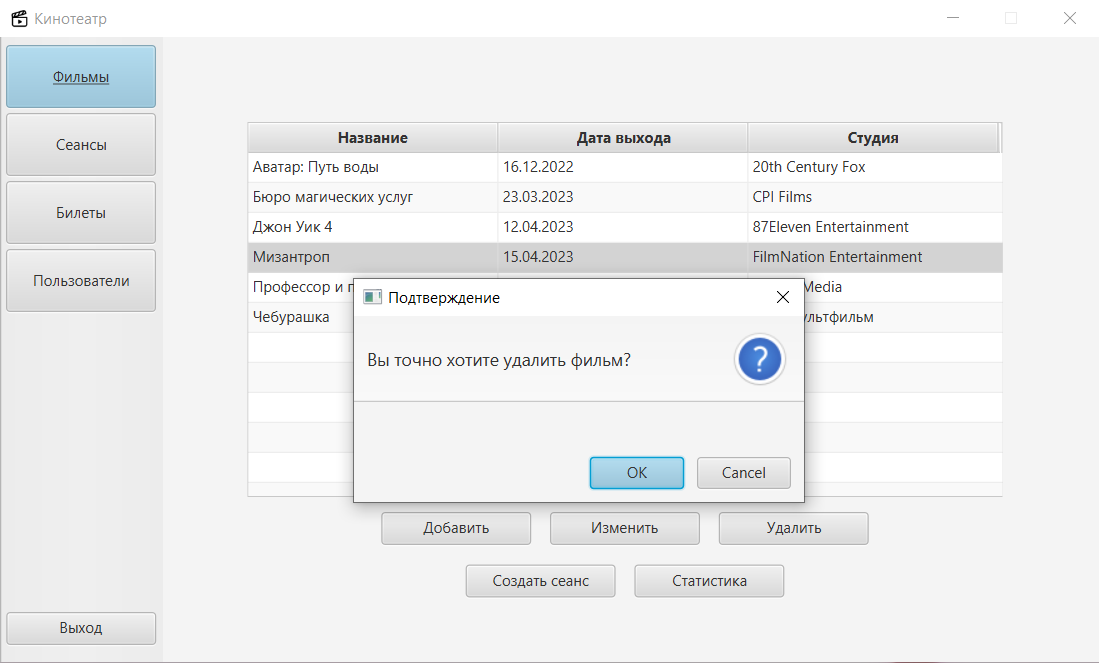
При удалении фильма выведется окно с подтверждением удаления (рисунок 4.8) так как при удалении фильма удалятся все связанные с ним сеансы и билеты.

Рисунок 4.8 – Удаление фильма

В меню администратора для управления сеансами также доступны CRUD операции (рисунок 4.9).

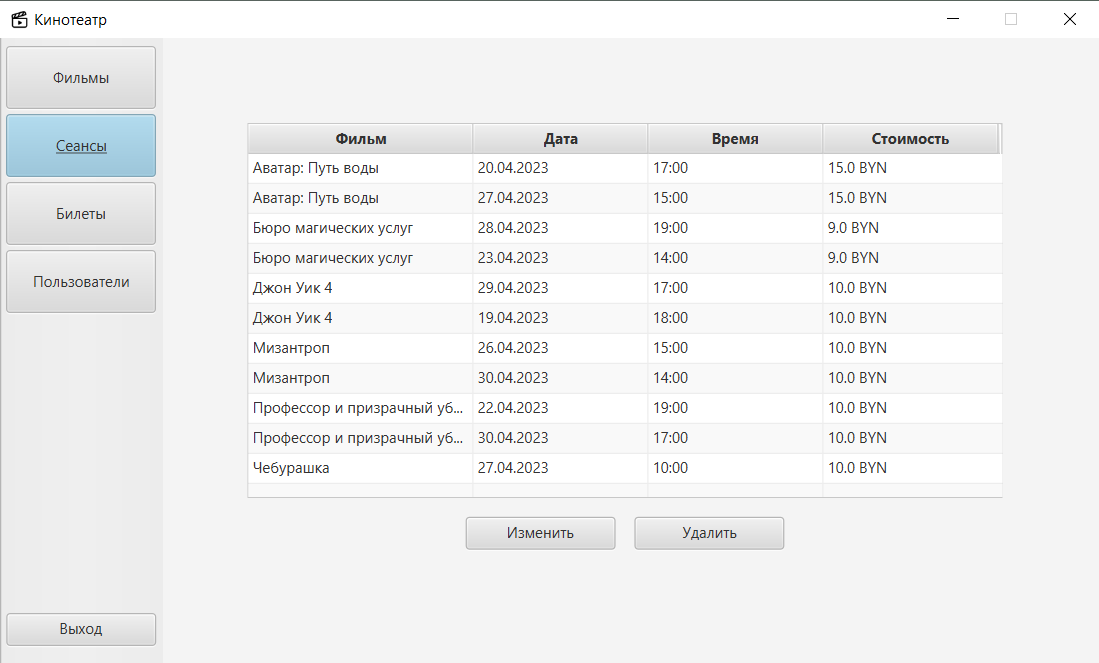


Рисунок 4.9 – Меню администратора для управления сеансами

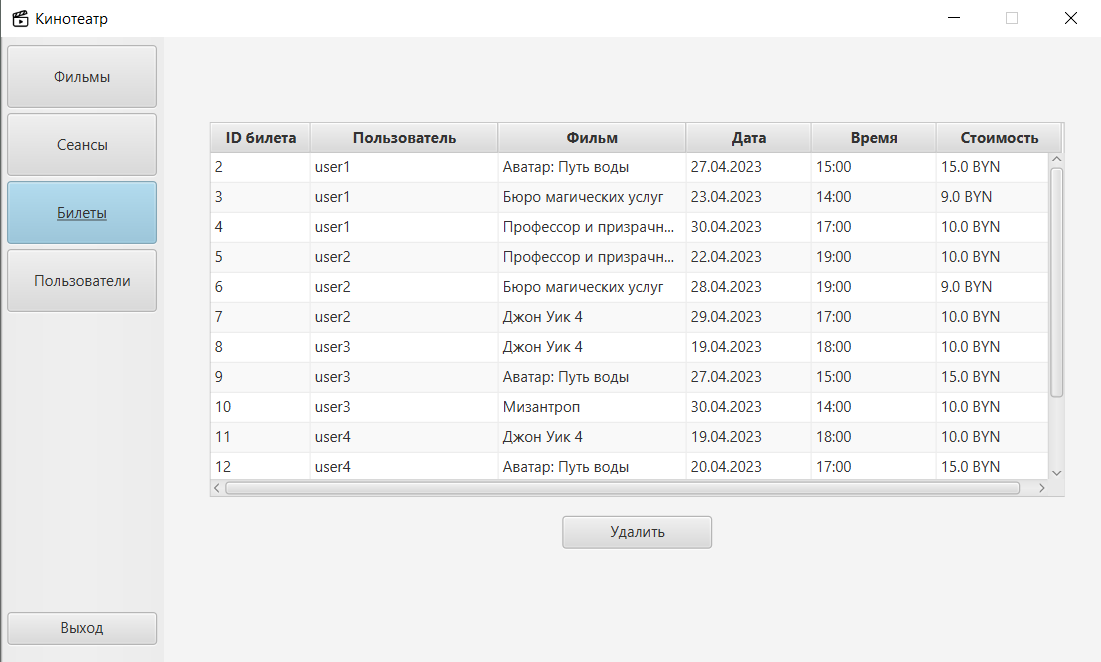
На рисунке 4.10 представлено меню управления билетами.

Рисунок 4.10 – Меню администратора для управления билетами

На рисунке 4.11 представлено меню управления пользователями.

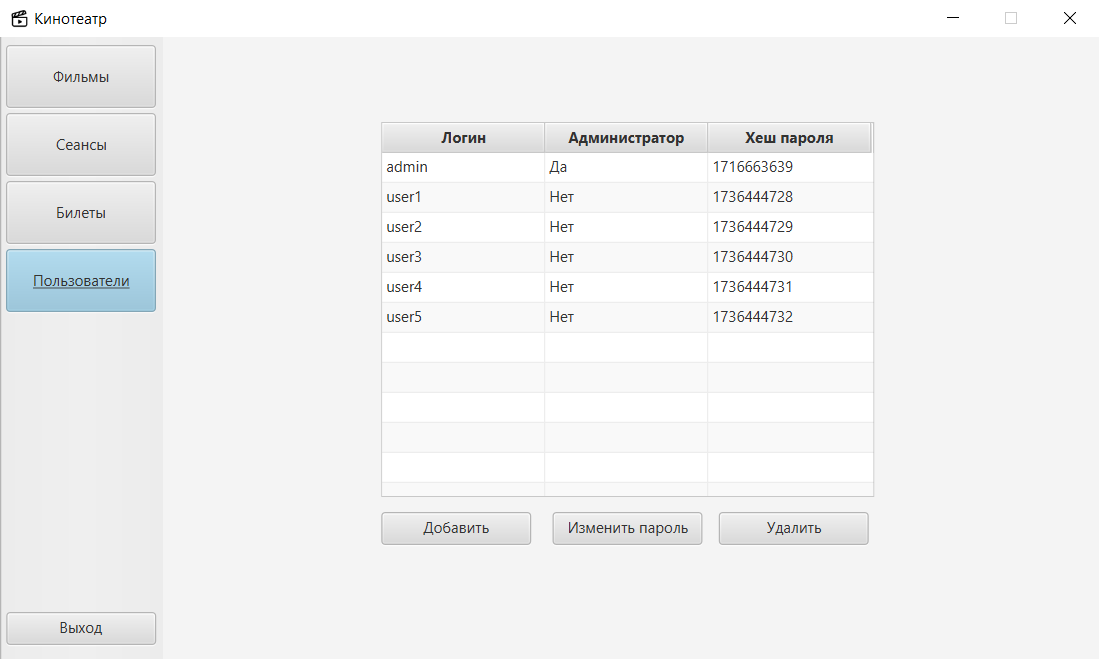


Рисунок 4.11 – Меню администратора для управления пользователями

В меню администратора для управления пользователями доступны CRUD операции.

При изменении пользователя нельзя изменить логин так как он является уникальным (рисунок 4.12).

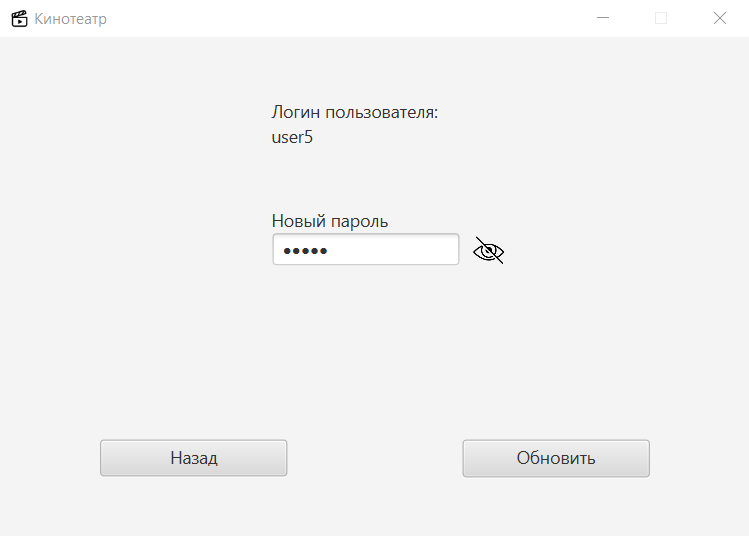
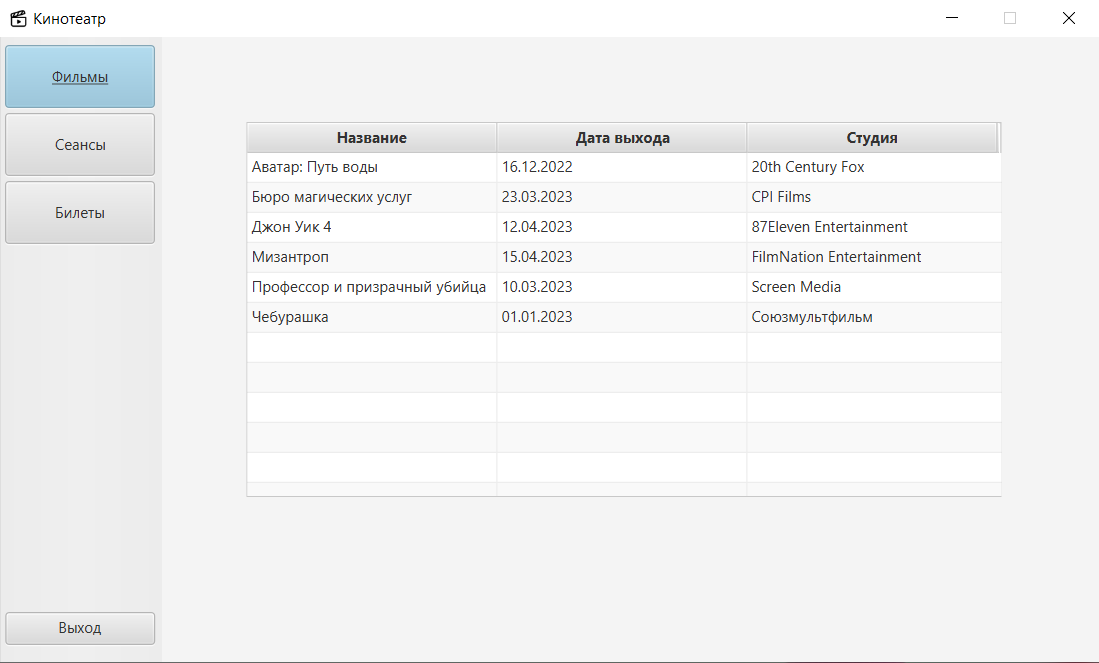


Рисунок 4.12 – Изменение пользователя

Далее рассмотрим вход в качестве пользователя.

Как видно на рисунке 4.13 для пользователя недоступны CRUD операции с фильмами.

Рисунок 4.13 – Просмотр фильмов пользователем

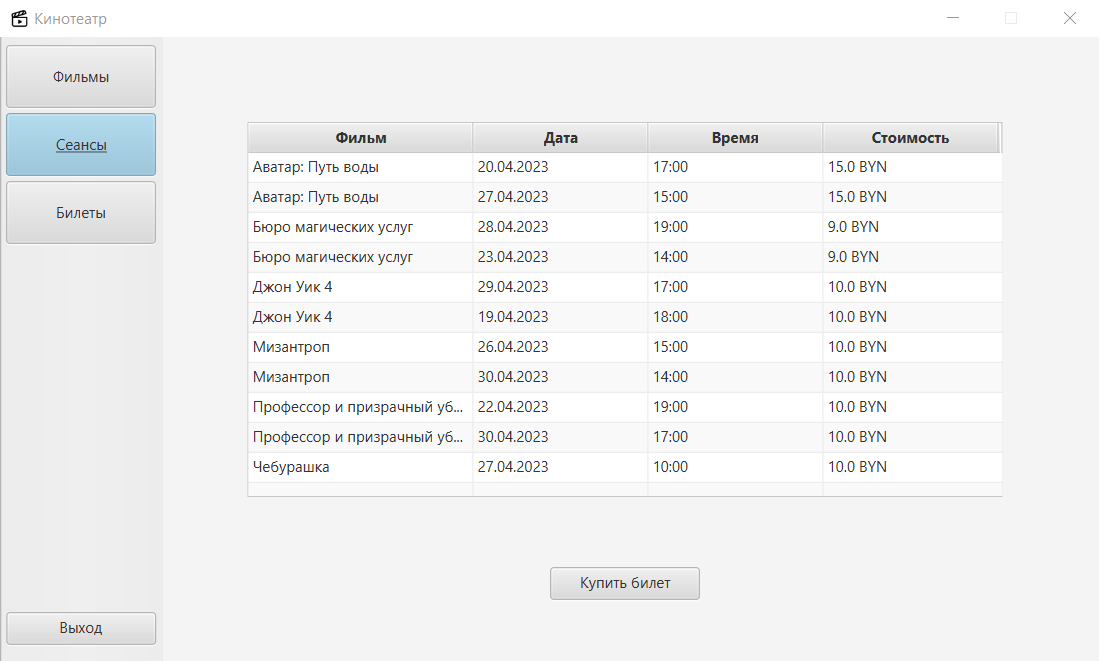
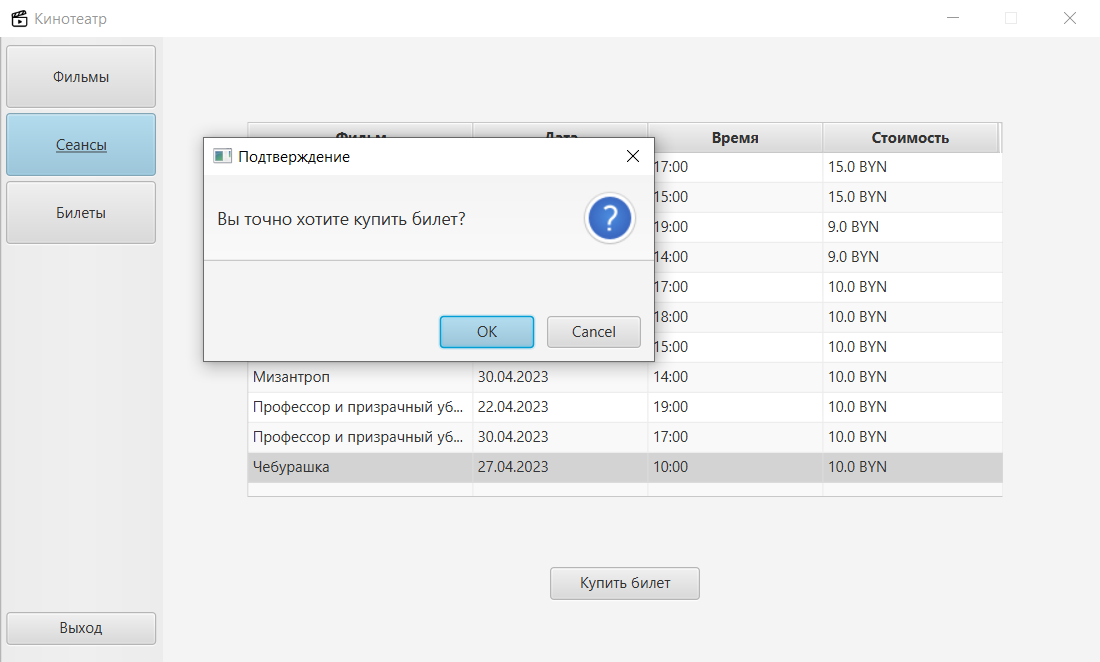
Также пользователь не может создавать фильмам сеансы и управлять ими (рисунок 4.14).

Рисунок 4.14 – Меню пользователя для выбора сеанса

Однако пользователь может приобрести билет на сеанс (рисунок 4.15).

Рисунок 4.15 – Окно покупки билета пользователем на выбранный сеанс

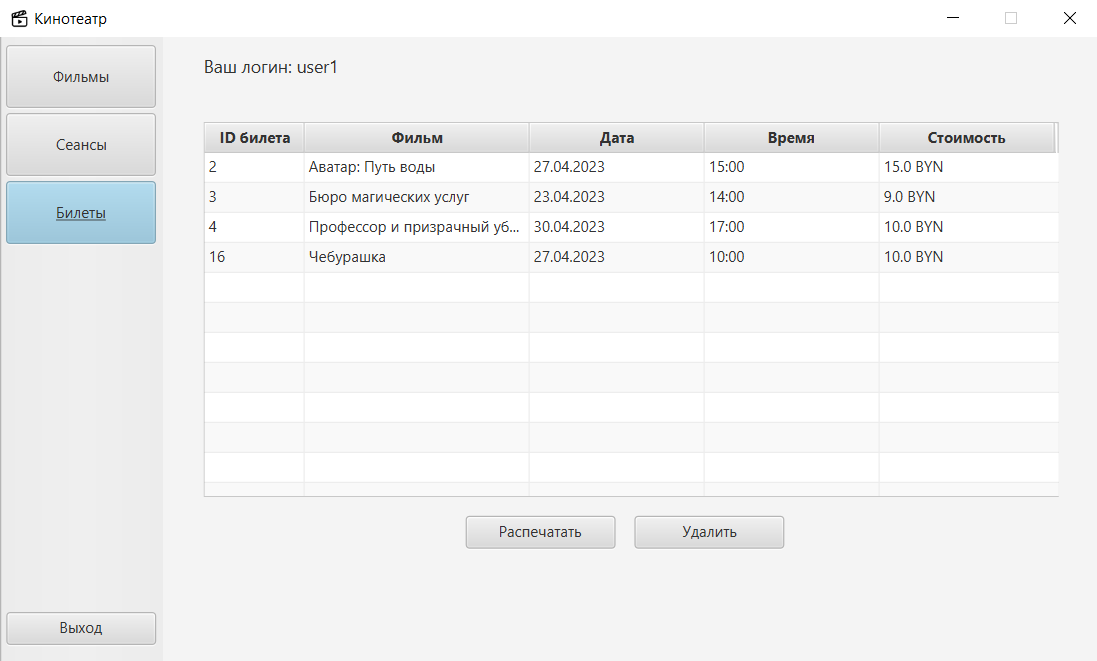
В меню пользователя для управления билетами он может просматривать купленные им билеты, удалить или распечатать нужные (рисунок 4.16).

Рисунок 4.16 – Меню пользователя для управления билетами

В данной главе были рассмотрены основные возможности данного приложения и подробно описано пользование всеми функциями. Описаны возможности пользователя и администратора. Таким образом, выполнены проектирование и разработка программного средства автоматизированной системы расчёта стипендий.

**ЗАКЛЮЧЕНИЕ**

В результате выполнения данного курсового проекта было разработано программное приложение, которое позволит улучшить и усовершенствовать работу современного кинотеатра.

Была изучена работа кинотеатров по онлайн-продаже билетов и также была достигнута основная цель курсового проекта, а именно разработка наиболее эффективных подходов в реализации системы онлайн-продажи билетов в кинотеатр.

Основные функции программного средства реализованы в соответствии поставленными перед разработкой требованиями. Для реализации поставленных целей был использован объектно-ориентированный язык Java, а также PostgreSql для хранения данных.

Использование данной программы сделает работу автоматизированной, повысит ее производительность, поможет проводить привычные процедуры с меньшими временными затратами. В дальнейшем данная программа может масштабироваться.

Выполнено проектирование и разработка программного средства планирования; в ходе объектного моделирования системы построен ряд *UML*-диаграмм; разработана информационная модель системы, представленная в виде схемы базы данных; описаны основные алгоритмы работы программного средства; разработано руководство пользователя; выполнено тестирование системы, показавшее ее соответствие функциональным требованиям, поставленным в задании на разработку.

Также было выполнено успешное тестирование системы, что подтверждает, что система может быть применена в любом кинотеатре.

Подводя итог, можно сказать, что все заявленные цели и задачи были успешно достигнуты.

**СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ**

1. Киммел, П. Основы визуального анализа и проектирования. / П. Киммел. – Москва: Нолидж, 2015. – 237 с.
2. Мельников И.Н. Интернет-технологии в маркетинге, 2018. – 230 с.
3. Эккель Б. Философия Java, 2019. – 180 с.
4. Роберт Мартин. Чистый код, создание, анализ и рефакторинг, 2009.-210 с.
5. Мартин Фаулер. Шаблоны корпоративных программных приложений, 2017. – 340 с.
6. Леоненков, А. В. Самоучитель UML / А. В. Леоненков. – СПб.: BHV, 2014. – 304 с.
7. Фаулер, М. Архитектура корпоративных программных приложений / М. Фаулер. – М.: Вильямс, 2016 – 544 с.
8. UML. Классика CS. 2-у изд./Пер. с англ.; Под общей редакцией проф. С. Орлова - СПб.: Питер, 2006. - 736 с.: ил.
9. Роб П., Коронел К. Системы баз данных: проектирование, реализация и управление – М: БХВ-Петербург, 2004. – 1040 с.
10. Харрингтон Джен Л. Проектирование реляционных баз данных – М: Лори, 2016. – 514 с.
11. КАК ОРГАНИЗОВАТЬ ОНЛАЙН-ПРОДАЖИ В КИНОТЕАТРЕ [Электронный ресурс]. – Электронные данные. – Режим доступа: https://cinemaplex.ru/2017/10/23/kak-organizovat-onlajn-prodazhi-v-kinoteatre.html

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

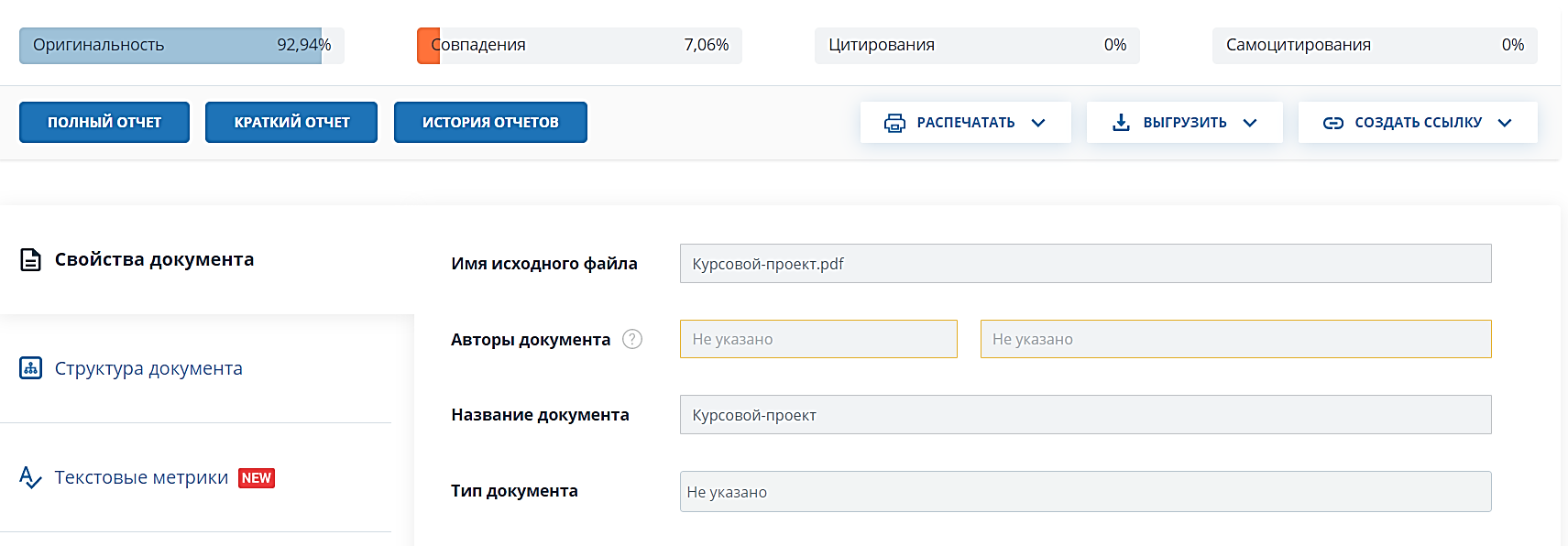
**Отчет о проверке на заимствования в системе «Антиплагиат»**

Рисунок А.1 – Отчет о проверке на заимствования в системе «Антиплагиат»

**ПРИЛОЖЕНИЕ Б**

**(обязательное)**

**Листинг кода алгоритмов, реализующих основную бизнес-логику**

Файл **Films.java**

@Getter  
@Setter  
@Entity  
@Table(name = "films")  
public class Film {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 @Column(name = "id\_film")  
 private Long id;  
  
 private String name;  
  
 private String date;  
  
 private String studio;  
  
  
 @OneToMany(mappedBy = "film", cascade = CascadeType.*ALL*, fetch = FetchType.*EAGER*)  
 private List<Seance> seances;  
}

Файл **Seances.java**

@Getter  
@Setter  
@Entity  
@Table(name = "seances")  
public class Seance {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 @Column(name = "id\_seance")  
 private Long id;  
  
 private String time;  
  
 private double price;  
  
 private String date;  
  
 @ManyToOne(fetch = FetchType.*EAGER*)  
 @JoinColumn(name = "film\_id")  
 private Film film;  
  
  
  
 @OneToMany(mappedBy = "seance", cascade = CascadeType.*ALL*, fetch = FetchType.*EAGER*)  
 private List<Ticket> tickets;  
  
}

**Продолжение приложения Б**

Файл **Tickets.java**

@Getter  
@Setter  
@Entity  
@Table(name = "tickets")  
public class Ticket {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 @Column(name = "id\_ticket")  
 private Long id;  
  
 @ManyToOne(fetch = FetchType.*EAGER*)  
 @JoinColumn(name = "seance\_id")  
 private Seance seance;  
  
 @ManyToOne(fetch = FetchType.*EAGER*)  
 @JoinColumn(name = "user\_id")  
 private User user;  
}

Файл **Users.java**

@Getter  
@Setter  
@Entity  
@Table(name = "users")  
public class User {  
  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 @Column(name = "id\_user")  
 private int idUser;  
  
 @Column(name = "login")  
 private String login;  
  
 @Column(name = "password")  
 private int password;  
  
 @Column(name = "isAdmin")  
 private boolean isAdmin;  
  
  
 @OneToMany(mappedBy = "user", cascade = CascadeType.*ALL*, fetch = FetchType.*EAGER*)  
 private List<Ticket> tickets;  
}