Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Инженерно-экономический факультет

Кафедра экономической информатики

Дисциплина: Программирование сетевых приложений

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**РАЗРАБОТКА АВТОМАТИЗИРОВАННЫЙ СИСТЕМЫ РАБОТЫ КИНОТЕАТРА**

Студент гр. 772302 Жанкевич В.А.

Руководитель Мухин Ю.С.

ассистент кафедры ЭИ

Минск 2019

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc532855503)

[1 Описание работы кинотеатра 5](#_Toc532855504)

[2 Постановка задачи автоматизации системы работы кинотеатра и обзор методов ее решения 8](#_Toc532855505)

[2.1 Постановка задачи 8](#_Toc532855506)

[2.2 Методы решения поставленной задачи 8](#_Toc532855507)

[3 Описание процесса оформление билета 12](#_Toc532855510)

[4 Информационная модель системы работы кинотеатра 16](#_Toc532855511)

[5 Диаграммы представлений автоматизированной системы работы кинотеатра 18](#_Toc532855512)

[5.1 Диаграмма вариантов использования 18](#_Toc532855513)

[5.2 Диаграмма последовательности 19](#_Toc532855514)

[5.3 Диаграмма классов 20](#_Toc532855515)

[5.6 Диаграмма компонентов 22](#_Toc532855516)

[5.6 Диаграмма развертывания 22](#_Toc532855517)

[6 Описание алгоритмов проектирующиз бизнес-логику системы работы кинотеатра 24](#_Toc532855518)

[7 Руководство пользователя автоматизированной системы работы кинотеатра 27](#_Toc532855520)

[7.1 Руководсвто пользователя 27](#_Toc532855521)

[7.2 Руководство пользователя по развтыванию системы 35](#_Toc532855522)

8. Результаты тестирования разработанной автоматизированной системы работы кинотеатра………………………………………………………………...37

[Заключение 38](#_Toc532855523)

Список использованной литературы……………………………………………39

[Приложение А(обязательное) Функциональное моделирование 40](#_Toc532855526)

[Приложение Б(обязательное) Схема алгоритма 45](#_Toc532855527)

[Приложение В(обязательное) Листинг кода 47](#_Toc532855529)

Приложение Г(обязательное) Листинг генерации скрипта базы данных…....50

**ВВЕДЕНИЕ**

В современной бизнес-среде управление является ценным ресурсом организации наряду со многими другими, такими как финансовые, материальные и человеческие ресурсы. Поэтому совершенствование управленческой деятельности становится одним из основных направлений совершенствования предприятия в целом. Наиболее распространенный способ повысить эффективность трудового процесса - автоматизировать его. Информационные системы решают множество различных задач, таких как: бухгалтерский учет, управление финансовыми потоками, управление складами и закупками, управление производственными процессами, управление маркетингом, документооборот, оперативное управление предприятием и т.д.

В данной курсовой работе будет раскрыта тема автоматизации приложения для кинотеатра. Одной из основных проблем в кинотеатре является скорость оформления билетов. Из-за больших очередей ежедневно множество клиентов кинотеатров отказывается от идеи сходить в кино.

В своей работе я хочу показать, как можно упростить работу для администраторов, улучшив способ продажи билетов.

Данная тема показалась мне актуальной по одной причине. Существует большое количество кинотеатров, в которых отсутствует автоматизация.

Целью данной курсовой работы автоматизировать системы работы кинотеатра. Оптимизация этой области сэкономит время и сотрудникам, к которым могли бы обращаться клиенты за помощью, и самим клиентам, которым не придется долго искать информацию.

Эта система помогает вести расчет выручки за прокат фильма, учет фильмов и сеансов. В любой момент администратор имеет возможность добавить фильм в прокат, удалить или изменить. Клиенту же в свою очередь гораздо проще становиться пополнить баланс, просмотреть фильмы в прокате и купить билет

Для достижения поставленной цели необходимо решить следующие задачи:

* описать предметную область и разработать модель, отражающую бизнес-процесс на предприятии;
* разработать и описать постановку задачи на разработку программного средства;
* выполнить проектирование программного средства, используя UML-диаграммы;
* разработать и описать алгоритм программного средства;
* выполнить программную реализацию программного средства;
* разработать и описать руководство пользователя;
* изучить формулы для расчета прибыли.

**1 ОПИСАНИЕ РАБОТЫ КИНОТЕАТРА**

Кинотеатр – это общественное здание (или часть его), оборудованное для показа фильмов.

Мультиплексный театр – это название одного комплекса или здания, в котором размещается более пяти кинозалов, каждый с соответствующими экранами фильмов. Число немного расплывчато, хотя. Иногда люди говорят, что для того, чтобы стать мультиплексным театром, нужно восемь кинотеатров.

IMAX означает Image MAXimum. Проще говоря, IMAX – это метод съемки с более высоким разрешением изображения и отображения больших изображений на экране кинотеатра IMAX.

Съемочные группы должны использовать специализированные кинокамеры с технологией IMAX для съемок с разрешением и размером IMAX. По сравнению со стандартным оборудованием камеры IMAX имеют возможность снимать в три раза больше теоретического разрешения по горизонтали. Чтобы перевести снятый фильм IMAX на экран кинотеатра, специализированный проектор IMAX становится необходимой частью кинотеатра IMAX.

Заездный кинотеатр или заездной кинотеатр - это форма кинотеатра, состоящая из большого киноэкрана на открытом воздухе, проекционной будки, концессионного стенда и большой парковки для автомобилей. В пределах этого закрытого пространства клиенты могут просматривать фильмы в уединении и комфорте своих автомобилей. В некоторых въездах есть небольшие игровые площадки для детей и несколько столов для пикника или скамейки.

Экран может быть таким же простым, как стена, окрашенная в белый цвет, или это может быть стальная ферменная конструкция со сложной отделкой. Первоначально звук фильма обеспечивался динамиками на экране, а затем отдельными динамиками, подвешенными к окну каждого автомобиля, которые были прикреплены с помощью провода. Эти системы были заменены более экономичным и простым в обслуживании способом трансляции звукового сопровождения при низкой выходной мощности на AM или FM-радио, которое принимается автомобильным радио. Это также позволяет саундтреку воспринимать аудиосистему в стерео-режиме в автомобиле, которая обычно более высокого качества и точности, чем простые динамики, используемые в старых системах.

Современный кинотеатр может многое предложить зрителю: интересный фильм, качественное изображение и хороший звук, комфортные залы с удобными креслами.

В Минске большое количество кинотеатров. У каждого кинотеатра есть свой репертуар, категория, разная вместимость, разные цены на билеты. Так же недавно открылся новый панорамный кинозал с обзором в 270 градусов.

Статистика напоминает нам, что посещение кино за прошедшее столетие стало правом каждого участвовать в культурной жизни нации.

Исторически сложилось так, что кино достигло этой решающей роли в качестве культурного и социального места встречи, будучи доступным для всех - обеспечивая, чтобы самые разные слои населения могли позволить себе регулярно ходить в кино. В этом смысле он принципиально отличается от телевидения или видеоигр, потому что посещение кинотеатра – это культурная практика, требующая социального контракта между зрителями, участниками выставки и режиссерами, когда вы соглашаетесь покинуть дом, сидеть в темноте. Театр и поделиться опытом фильма с людьми, которых вы не знаете.

Это может звучать странно, но смотреть кино и ходить в кино - это разные вещи. Ходить в кино требует уровня приверженности, который принципиально отличается от выбора сидеть и смотреть телевизор или играть с игровой приставкой дома.

В то же время современный дневной кинотеатр сталкивается со своими особыми проблемами. Цифровые технологии могут сделать кинопроизводство более доступным для простых людей, но все же имеет свои недостатки. Мультиплексы могут создавать очереди на бесконечные часы, чтобы смотреть последний блокбастер реже, чем раньше, но они также не всегда хороши.

Всем клиентам необходимо быстрое и точное обслуживание, как и в любой другой сфере услуг. Клиенту может понадобиться выбрать фильм, в какое-то определённое время суток, или в каком-то определённом кинотеатре, или какой-то определённый фильм, посмотреть его оценку. Также клиенту нужно определённое количество мест и расположение, и, наконец, клиента интересует цена билета, хоть в наших кинотеатрах она почти одинаковая.

Зрительные залы заполнены обычно большим количеством мест, даже не думая о том, чтобы дать зрителям оптимальный опыт. Большую часть времени сиденья неудобные и маленькие с недостаточным пространством для ног. Так же у вас есть кинотеатры, где все ряды находятся на одном уровне (без наклона); если он занят, скорее всего, кто-то сядет перед вами и заблокирует ваш взгляд. Конечно, это все потому, что им нужно как можно больше мест в одной аудитории.

Это совершенно глупо, потому что чаще всего большинство мест будет пустым. Да, вечер пятницы и субботы может быть переполнен. Но в остальные дни недели в зал будет пустой. Кинотеатры должны отказаться от идеи заполнения как можно большего количества мест и постараться удовлетворить максимальное впечатление от фильма.

Одну из важных проблем в современных кинотеатрах занимает пиратство фильмов. Пиратство означает несанкционированное копирование защищенного авторским правом контента, который затем продается по значительно более низким ценам на «сером» рынке. Легкость доступа к технологиям означает, что с годами пиратство стало более распространенным.

Пиратство в кино не только ранит большие голливудские киностудии. Потерянные доходы от пиратства влияют на плотников, электриков и других мастеров, которые работают в индустрии развлечений, а также на экономику стран по всему миру, где производятся фильмы. В кинозале всегда за этим следит оператор зала.

Для качественного обслуживания и удобного взаимодействия кинотеатров с клиентом существует справочная система кинотеатров.

Справочная система кинотеатров должна оказывать следующие услуги:

– Выдача справки о сеансе фильма в указанном кинотеатре.

– Выдача справки о сеансах указанного фильма во всех кинотеатрах.

В справочной есть человек, который отвечает за работу справочной службы. Он получает сведения о расписании сеансов, времени их проведения, стоимости билета, и количестве мест у кинотеатров, обрабатывает, сортирует и заносит данные в базу данных.

Также в справочной службе работают менеджеры по работе с клиентами, которые предоставляют нужные данные заказчикам по телефону.

В случае, если заносить все информация в базу данных, то изменять данные, брать нужные выборки и заносить новые данные будет гораздо проще.

Наиболее подходящим вариантом для хранилища всей информации подходят базы данных.

Использование базы данных и автоматизированной системы для работы с базой данных существенно упростит работу, а главное, вся информация, касающаяся сеансов в кинотеатре, будет храниться в одном месте. Если предоставить данную базу для общего пользования, то клиенты, даже не имея специальной подготовки, смогут самостоятельно найти необходимую информацию. Время работы работников справочной службы значительно сократится.

# **ПОСТАНОВКА ЗАДАЧИ АВТОМАТИЗАЦИИ СИСТЕМЫ РАБОТЫ КИНОТЕАТРА И ОБЗОР МЕТОДОВ РЕШЕНИЯ**

* 1. **Постановка задачи**

В соответствии с техническим заданием необходимо разработать программное приложение на объектно-ориентированном языке программирования Java. Так же должна быть разработана архитектура клиент-сервер, где сервер должен обладать свойством многопоточности. Необходимо реализовать механизмы автоматизации и регистрации пользователей, которые выполняют разные роли, т.е. в данном случае «администратор» и «клиент».

Приложение должно быть связано с базой данных, в которую будет сохраняться необходимая нам информация.

Для создания функциональной модели был выбран стандарт IDEF0. IDEF0 – методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность.

Основной задачей было разобраться, как работает персонал кинотеатра и усовершенствовать его работу с помощью автоматизации системы работы кинотеатра.

Автоматизация процесса ввода информации о фильме приведет к упрощению работы всего кинотеатра, снижению задействования человеческого ресурса, сокращению временного ресурса а, следовательно, к получению большей прибыли и повышению уровня рентабельности предприятия.

* 1. **Методы решения поставленной задачи.**

Для разработки программного приложения был выбран язык программирования Java. Объектно-ориентированный язык программирования Java в настоящее время является одним из самых популярных языков, для разработки программных приложений.

Одним из плюсов языка Java является то, что он не зависит от операционной системы, тем самым позволяет использовать, например, такую операционную систему как Linux. Методы JDK найдут применение в реализации данной программы.

Пакет java.net обеспечит сетевое приложение, разработанное на архитектуре клиент-сервер по протоколу TCP/IP. Пакет java.sql обеспечивает доступ и обработку данных, сохраненных в источнике данных, в том числе и в реляционной базе данных

Для решения поставленной задачи целесообразно обратиться к среде

программирования Java. Проанализированные, в первой части работы, методы JDK

найдут применение в реализации данной программы.

Первым делом необходимо подключить библиотеки и пакеты. Так например awt -

это исходная платформо-независимая оконная библиотека графического интерфейса, io -

поддерживает базовую систему ввода-вывода Java, включая файловый ввод-вывод, util –

пакет, содержащий большой ассортимент классов и интерфейсов, поддерживающих

широкий диапазон функциональных возможностей, swing – пакет, содержащий классы,

реализующие базовые компоненты библиотеки Swing, такие как кнопки, метки и флажки.

Для решения поставленной задачи целесообразно обратиться к среде

программирования Java. Проанализированные, в первой части работы, методы JDK

найдут применение в реализации данной программы.

Первым делом необходимо подключить библиотеки и пакеты. Так например awt -

это исходная платформо-независимая оконная библиотека графического интерфейса, io -

поддерживает базовую систему ввода-вывода Java, включая файловый ввод-вывод, util –

пакет, содержащий большой ассортимент классов и интерфейсов, поддерживающих

широкий диапазон функциональных возможностей, swing – пакет, содержащий классы,

реализующие базовые компоненты библиотеки Swing, такие как кнопки, метки и флажки.

Для решения поставленной задачи целесообразно обратиться к среде

программирования Java. Проанализированные, в первой части работы, методы JDK

найдут применение в реализации данной программы.

Первым делом необходимо подключить библиотеки и пакеты. Так например awt -

это исходная платформо-независимая оконная библиотека графического интерфейса, io -

поддерживает базовую систему ввода-вывода Java, включая файловый ввод-вывод, util –

пакет, содержащий большой ассортимент классов и интерфейсов, поддерживающих

широкий диапазон функциональных возможностей, swing – пакет, содержащий классы,

реализующие базовые компоненты библиотеки Swing, такие как кнопки, метки и флажки.

Для решения поставленной задачи целесообразно обратиться к среде

программирования Java. Проанализированные, в первой части работы, методы JDK

найдут применение в реализации данной программы.

Первым делом необходимо подключить библиотеки и пакеты. Так например awt -

это исходная платформо-независимая оконная библиотека графического интерфейса, io -

поддерживает базовую систему ввода-вывода Java, включая файловый ввод-вывод, util –

пакет, содержащий большой ассортимент классов и интерфейсов, поддерживающих

широкий диапазон функциональных возможностей, swing – пакет, содержащий классы,

реализующие базовые компоненты библиотеки Swing, такие как кнопки, метки и флажки.

Первым делом необходимо подключить библиотеки и пакеты. Так например, io – поддерживает базовую систему ввода-вывода Java, включая файловый ввод-вывод, swing – пакет, содержащий классы, реализующее базовые компоненты библиотеки Swing, такие как кнопки, метки, флажки.

Современные программы нуждаются в графическом интерфейсе пользователя (GUI). Пользователи отвыкли работать через консоль: они управляют программой и вводят входные данные посредством так называемых элементов управления (в программировании их также называют визуальными компонентами), к которым относятся кнопки, текстовые поля, выпадающие списки и т.д.

Каждый из современных языков программирования предоставляет множество библиотек для работы со стандартным набором элементов управления. Напомним, что под библиотекой в программировании набор готовых классов и интерфейсов, предназначенных для решения определенного круга задач.

В Java есть три библиотеки визуальных компонентов для создания графического интерфейса пользователя. Самая ранняя из них называется AWT. Считается, что при ее проектировании был допущен ряд недочетов, вследствие которых с ней довольно сложно работать. Библиотека Swing разработана на базе AWT и заменяет большинство ее компонентов своими, спроектированными более тщательно и удобно. Третья, самая новая библиотека, называется SWT.

Каждая библиотека предоставляет набор классов для работы с кнопками, списками, окнами, меню и т.д., но эти классы спроектированы по-разному: они имеют различный набор методов с разными параметрами, поэтому «перевести» программу с одной библиотеки на другую (например, с целью увеличения быстродействия) не так-то просто. Это почти как перейти с одного языка программирования на другой: все языки умеют делать одно и то же, но у каждого из них свой синтаксис, своя программная структура и свои многочисленные хитрости.

По этой причине вместо того, чтобы делать обзор всех трех библиотек, мы постараемся получше разобраться в одной из них – библиотеке Swing. Полноценный графический интерфейс может быть разработан с ее помощью.

Каждая GUI-программа запускается в окне и по ходу работы может открывать несколько дополнительных окон.

В библиотеке Swing описан класс JFrame, представляющий собой окно с рамкой и строкой заголовка (с кнопками «Свернуть», «Во весь экран» и «Закрыть»). Оно может изменять размеры и перемещаться по экрану.

Для обеспечения взаимодействия приложения, написанного на языке программирования Java, и реляционной системой управления базами данных MySQL используется драйвер mysql-connector-java. Данный драйвер совместим с промышленным стандартом взаимодействия Java-приложений с СУБД JDBC, который реализован в виде выше упомянутого пакета java.sq

Для разработки программного приложения была выбрана среда разработки IntelliJ Idea Community Edition 2018, в качестве базы данных была взята MySQL, сама база данных создавался в MySQL Workbench

Так же были использованы паттерны проектирования, такие как DAO и Singlteon.

Шаблон объекта доступа к данным (DAO) – это структурный шаблон, который позволяет нам изолировать прикладной / бизнес-уровень от уровня постоянства (обычно это реляционная база данных, но это может быть любой другой механизм сохранения) с использованием абстрактного API.

Функциональность этого API заключается в том, чтобы скрыть от приложения все трудности, связанные с выполнением операций CRUD в базовом механизме хранения. Это позволяет обоим слоям развиваться отдельно, ничего не зная друг о друге.

С помощью шаблона проектирования DAO у нас есть следующие компоненты, от которых зависит наш дизайн:

* интерфейс объектов доступа к данным. Этот интерфейс определяет стандартные операции, которые должны выполняться на объектах модели;
* конкретный класс объекта доступа к данным. Этот класс реализует интерфейс выше. Этот класс отвечает за получение данных из источника данных, который может быть базой данных /xml или любым другим механизмом хранения;
* объект объекта или объекта модели. Этот объект является простым POJO, содержащим методы get/set для хранения данных, полученных с использованием класса DAO.

Так же использовался singleton-паттерн. В разработке программного обеспечения шаблон singleton – это шаблон разработки программного обеспечения, который ограничивает создание экземпляра класса одним «единственным» экземпляром. Это полезно, когда ровно один объект необходим для координации действий в системе.

Рисунки предоставлены в диаграммах классов.

# **ОПИСАНИЕ ПРОЦЕССА ОФОРМЛЕНИЯ БИЛЕТА**

Для получения более полного представления о том, как можно автоматизировать деятельность логистической компании и обеспечить её эффективную работу с помощью программного продукта, была создана функциональная модель.

Для создания функциональной модели был выбран стандарт IDEF0. IDEF0 – методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность.

На контекстной диаграмме верхнего уровня (рисунок 3.1) представлена функциональная модель «Обработать заказ», а также определены потоки входных и выходных данных, механизмы ограничения и управления данными.

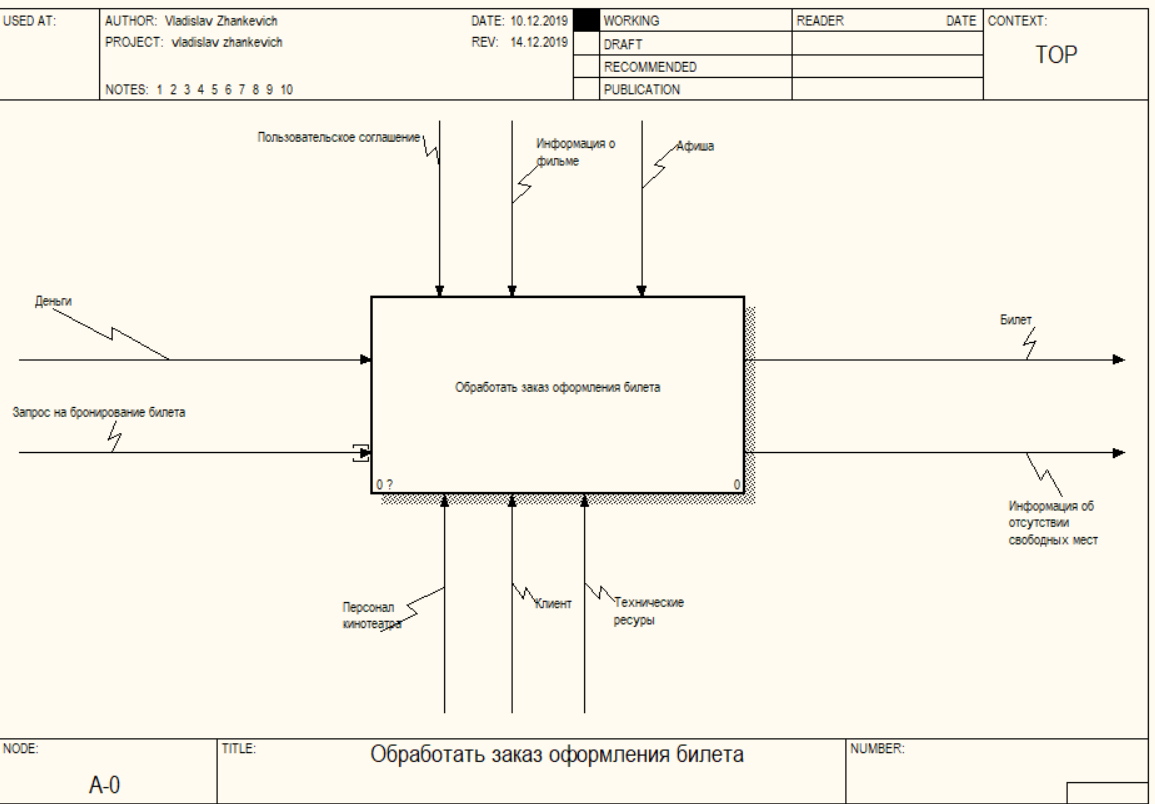


Рисунок 3.1 – Контекстная диаграмма верхнего уровня

Входной поток включает в себя деньги и запрос на бронирование билета. После прохождения всех стадий, на выходе имеем купленный билет или информацию об отсутствии свободных мест в зале.

На рисунке 3.2 отображена декомпозиция контекстной диаграммы 1-го уровня, состоящая из трех блоков: «Изучение информации о фильме», «Изучении афиши», «Оформление билета».

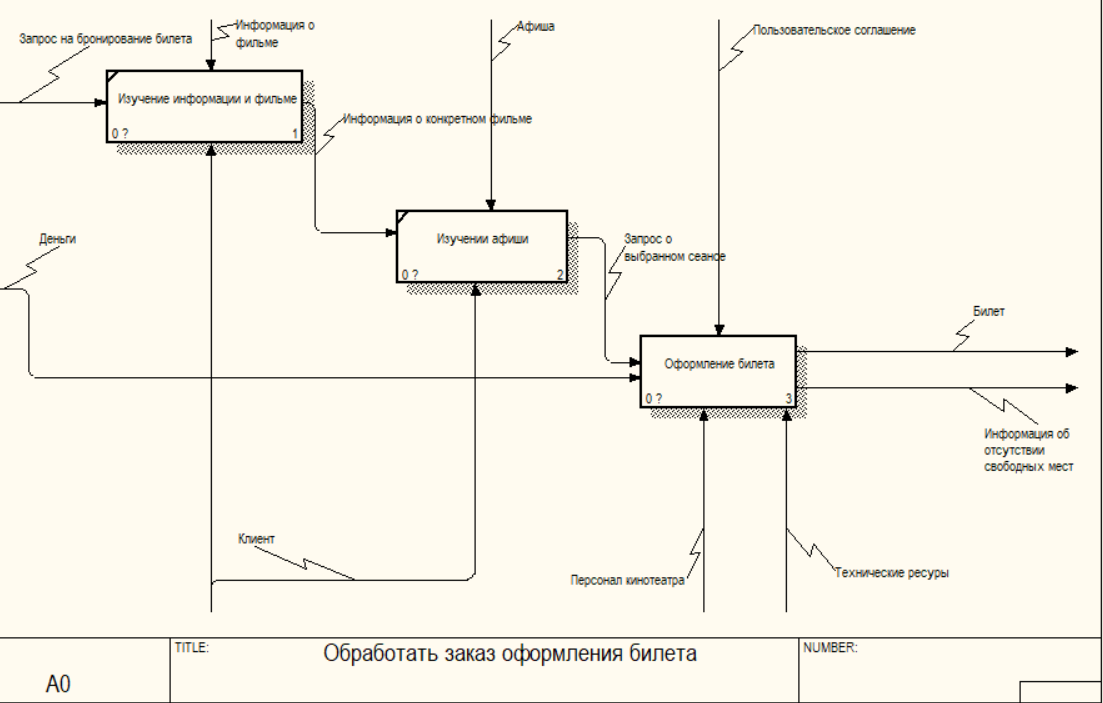


Рисунок 3.2 – Декомпозиция контекстной диаграммы верхнего уровня

Процесс обработки заказа состоит из трех основных работ:

* Изучение информации о фильме – процесс выполняется клиентом. Сюда входит изучение полной информации о фильме.
* Изучение афиши – процесс также выполняется клиентом. Вначале клиент смотрит, какой кинотеатр располагается к нему ближе. Затем на основании данных полученных о фильме раннее, он выбирает удобное для себя время.
* Оформление билета – процесс выполняется администратором. После получения данных и клиенте, менеджер либо оформляет ему билет, либо сообщает об отсутствии свободных мест.
* Стрелки, описывающие связь внутри диаграммы декомпозиций:
* Запрос – устное выражение клиента о желании посмотреть фильм. Исходя из данных о фильмах, хранящихся в базе данных, клиент выбирает подходящие для него.
* Деньги – в случаи если на данный сеанс остались свободные места, менеджер возьмет у клиента деньги.
* Информация о фильме – информация, пришедшая от поставщика фильмов.
* Информация о конкретном фильме – после того как клиент выбрал для себя ряд преимущественных фильмов, он переход с этими мыслями к поиску интересующего его сеанса.
* Афиша – распоряжение от директоров кинотеатров говорящие о том в какое время, и какое кино проводить.
* Запрос о выбранном сеансе – после того как клиент решил для себя «чего он хочет», он высказывает свои пожелания менеджеру.

Декомпозиция блока «Оформление билета» предоставлена на рисунке 3.3. Она представлена четырьмя компонентами: «Выбор кинотеатра», «Выбор зала», «Выбор сеанса» и «Формирование билета»

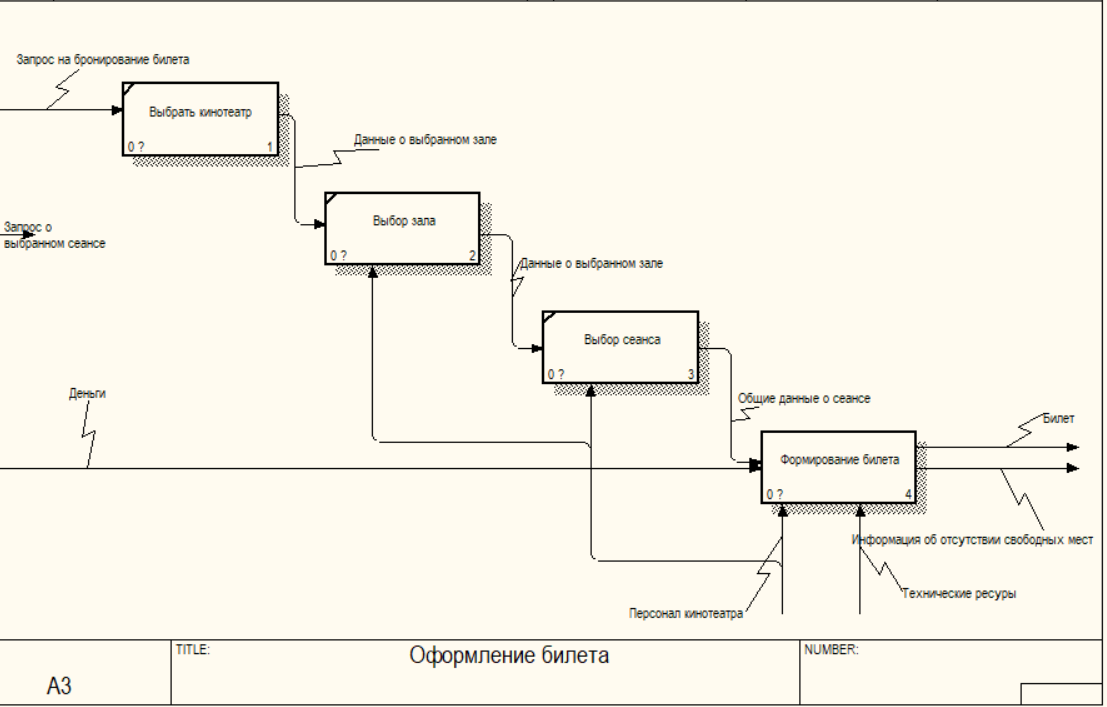


Рисунок 3.3 – Декомпозиция блока «Оформление билета»

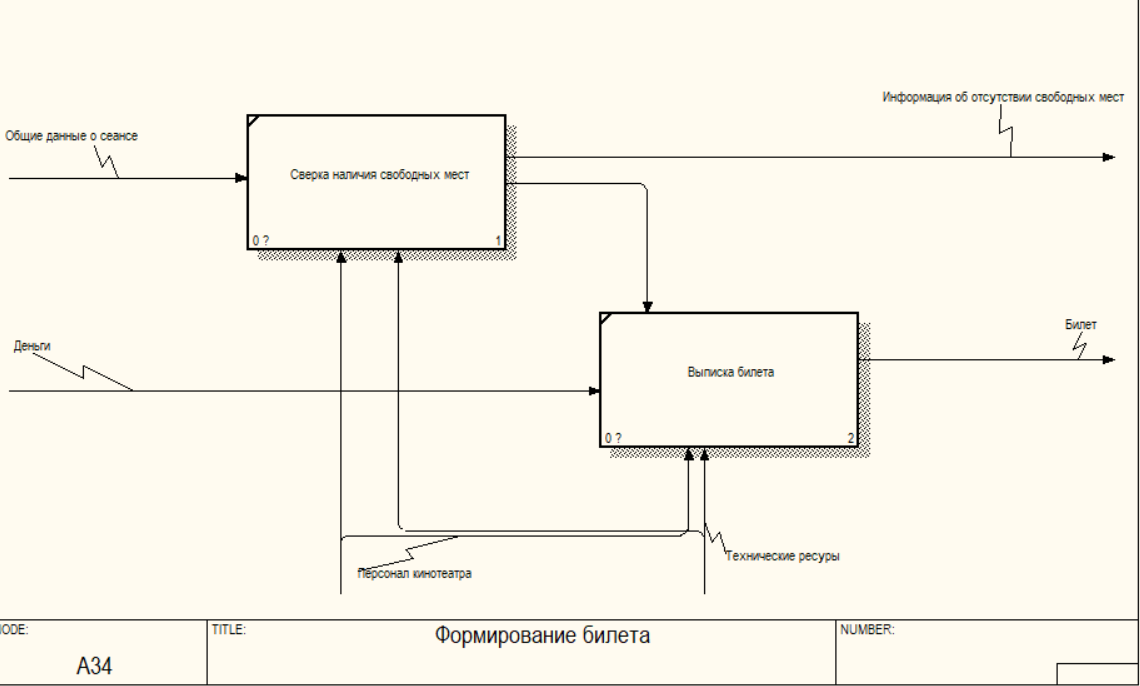


Рисунок 3.4 – Декомпозиция блока «Формирования билета»

Декомпозиция компонента «Формирования билета» состоит из двух блоков (рисунок 3.4). Она представлена двумя блоками «Сверка наличия свободных мест» и «Выписка билета».

# **4 ИНФОРМАЦИОННАЯ МОДЕЛЬ АВТОМАТИЗАЦИИ СИСТЕМЫ РАБОТЫ КИНОТЕАТРА**

В соответствии с техническим заданием была выбрана информационная модель системы MySQL. Вся необходимая информация для заполнения таблиц в созданном программном приложении считывается из созданной в MySQL Workbench базы данных. Таблицы и их связи представлены на рисунке 4.1.

В процессе информационного моделирования были выделены следующие сущности:

– Фильм;

– Сеанс;

– Корзина;

– Билет;

– Пользователь;

– Клиент.

Сущности «Фильм», «Сеанс» созданы, чтобы администратор мог реализовать добавления фильма и выход фильма в прокат. В своем личном кабинете, администратор может добавить информацию такую как «Названия фильма», «Жанр», «Главные роли» и «Режиссер».

Сущности «Билет» и «Корзина» нужны для того, чтобы администратор мог видеть заказы пользователей.

Таблица «user» хранит в себе информацию о зарегистрированных пользователях. В ней находятся поля:

– id – уникальный идентификатор пользователя;

– login – логин пользователя;

– password – пароль пользователя;

– role – роль, может быть администратором или пользователем.

Таблица «film» хранит информацию о фильме. В данной таблице находятся поля:

– id – уникальный идентификатор фильма;

– name – названия фильма;

– genre – жанр фильма;

– producers – имя и фамилия продюсера фильма;

– main\_roles – имена, кто исполняет главные роли;

Данная таблица связана с «session» связью 1:M.

В таблице «session» хранится информация о конкретном сеансе фильма. Поля, которые используются в данной таблице:

– id – уникальный идентификатор сеанса;

– date\_time – дата проведения сеанса;

–price – стоимость билета на сеанс;

Таблица «basket» хранит в себе информацию о заказах клиента. Она связана с таблицами «session» и «user» и имеет уникальный номер пользователя и уникальный номер сеанса.

Таблица «ticket» связана с таблицами «session» и «user» связью 1:М и имеет уникальный номер пользователя и уникальный номер сеанса.

Таблице «client» хранит в себе информацию о количестве денег на балансе и уникальный номер пользователя. Связана с таблицей «user» связью 1:М.

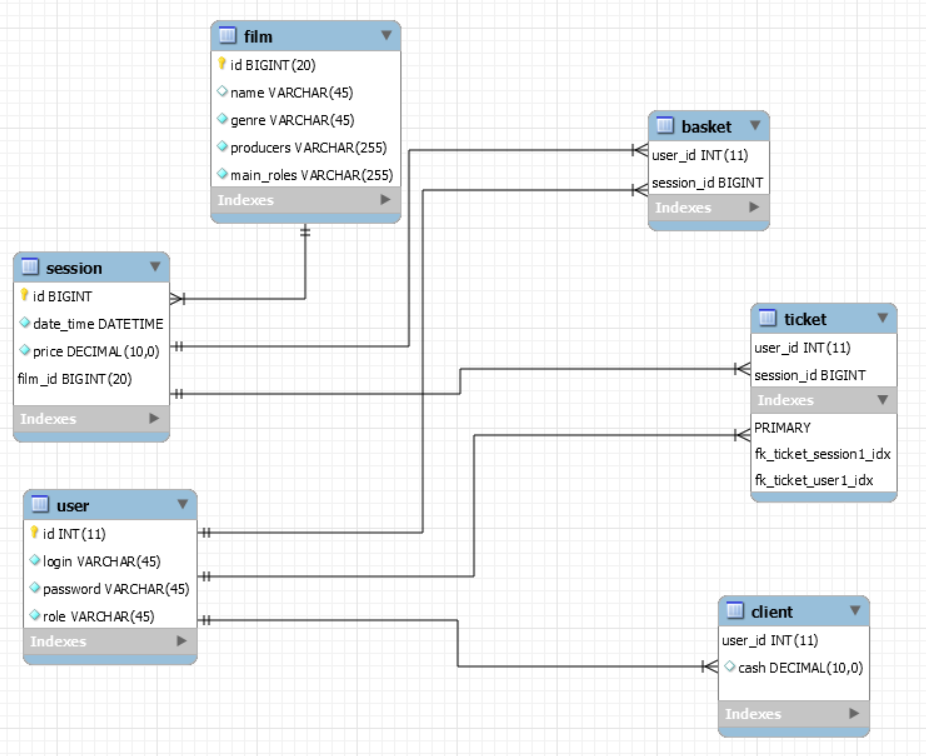


Рисунок 4.1 – Информационная модель системы

Приведем эту модель к третьей нормальной форме. Моя база данных должна удовлетворять требованиям второй нормальной формы и не содержать транзитивных зависимостей (функциональных зависимостей между не ключевыми полями). В результате приведения, получаем, что не ключевой атрибут сущности функционально зависит от всего первичного ключа и ни от чего другого.

# **ДИАГРАММЫ ПРЕДСТАВЛЕНИЙ АВТОМАТИЗАЦИИ СИСТЕМЫ РАБОТЫ КИНОТЕАТРА**

## 5.1 Диаграмма вариантов использования

Диаграмма использования (use case diagram) ‒ это наиболее общее представление функционального назначения системы. В разработке системы работы кинотеатра были задействованы такие лица, как администратор и пользователь.

На рисунке 5.1 представлена диаграмма вариантов использования. Она показывает функционал двух актеров: администратора и пользователя. Администратор имеет функционал шире, чем пользователь, так как должен управлять всей информацией о фильмах и сеансах. Он может добавлять фильмы в афишу, указывать стоимость билетов, работать с удалением информации.

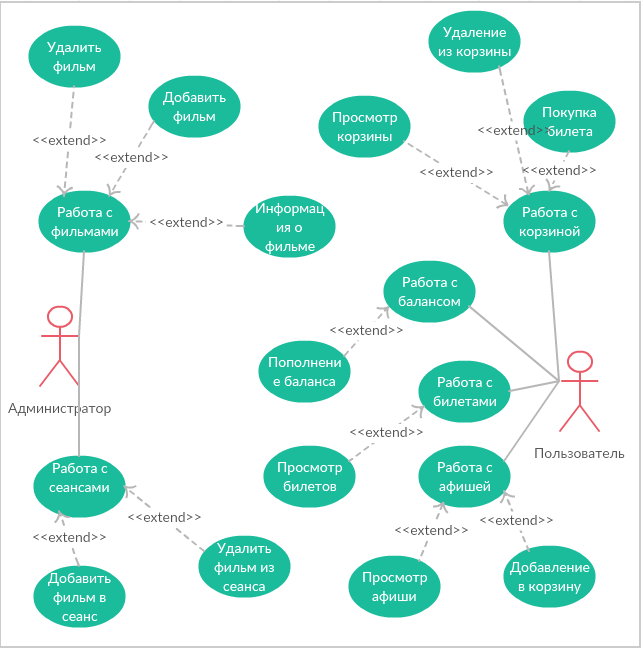


Рисунок 5.1 – Диаграмма вариантов использования

**5.2 Диаграмма развертывания**

На рисунке 5.2 изображена диаграмма развертывания, которая представляет фрагмент физического предтсавления системы автоматизации работы кинотеатра.

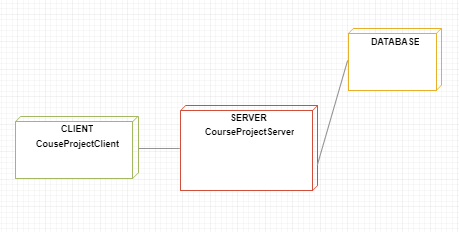


Рисунок 5.2 – Диаграмма развертывания

**5.3 Диаграмма компонентов**

На рисунке 5.3 представлена диаграмма компонентов. Она показывает независимые модули ПО, скрывающие свою реализацию и взаимодействующие друг с другом через интерфейсы. Так модуль ClientScreen реализует пользовательский интерфейс и связь с сервером. Модуль Server отвечает за бизнес-логику сервера и за взаимодействие с базой данных.



Рисунок 5.3 – Диаграмма компонентов

## 5.4 Диаграмма последовательности

На диаграмме последовательности(рисунок 5.4) представлены 4 основные объекта: «Пользователь», «Controller», «Server», «Database». Диаграмма показывает хронологическую последовательность регистрации пользователя, но не включает в себя информацию о взаимосвязях объектов. После ввода корректных данных, информация отправляется на сервер. На основании полученных данных сервер формирует запрос в базу данных. Если в базе данных уже существует пользователь с полученным логином, клиенту вернется соответствующее сообщение. В противном случае, пользователь успешно пройдет регистрацию.

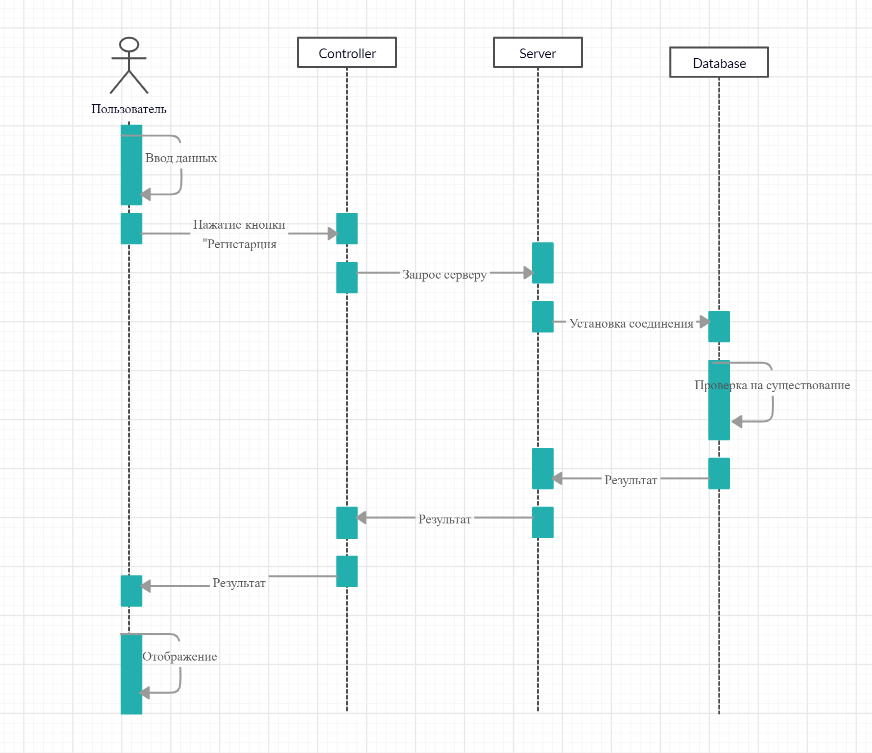


Рисунок 5.4 – Диаграмма последовательностей

## 5.5 Диаграмма классов

На рисунках 5.5 и 5.6, 5.7 представлены диаграммы классов серверной, клиентской части. Они используется для общего концептуального моделирования структуры приложения и для детального моделирования, перевода моделей в программный код. Диаграммы классов также можно использовать для моделирования данных. Классы в диаграмме классов представляют как основные элементы, взаимодействия в приложении, так и классы, подлежащие программированию.

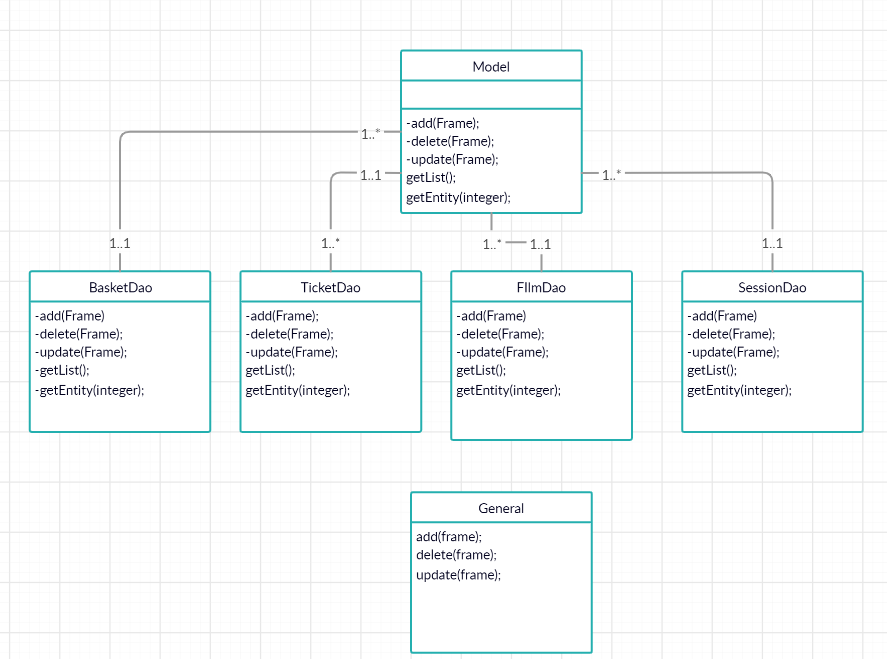


Рисунок 5.5 – Диаграмма классов

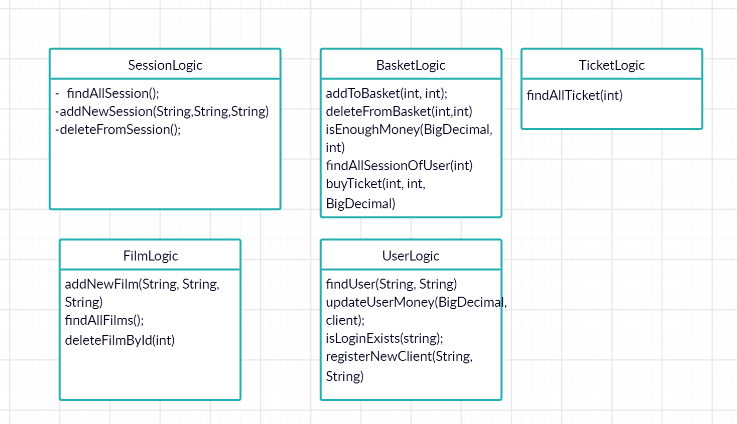


Рисунок 5.6 – Диаграмма классов

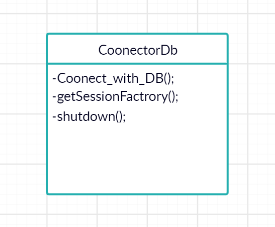


Рисунок 5.7 – Диаграмма классов

**5.6 Диаграмма состояний**

Диаграмма состояний при входе пользователя в систему представлена на рисунке 5.9.



Рисунок 5.8 – Диаграмма состояния

На ней отображено взаимодействие между клиентом и сервером в ходе использования программы. После успешного ввода пароля и логина, появляется окно меню. Необходимо будет выбрать действие, после чего система сформулирует запрос серверу. Далее произойдет выбор данных из базы данных и отправка результатов. И в самом конце будут выведены нужные данные.

1. **ОПИСАНИЕ АЛГОРИТМОВ РЕАЛИЗУЮЩИХ БИЗНЕС ЛОГИКУ ПРОЕКТИРУЕМОЙ СИСТЕМЫ РАБОТЫ КИНОТЕАТРА**

В соответствии в техническим заданием, необходимо было реализовать многопоточность сервера, на рисунке 6.1 изображена блок-схема которая отображает работу сервера и предполагает его безостановочную работу.

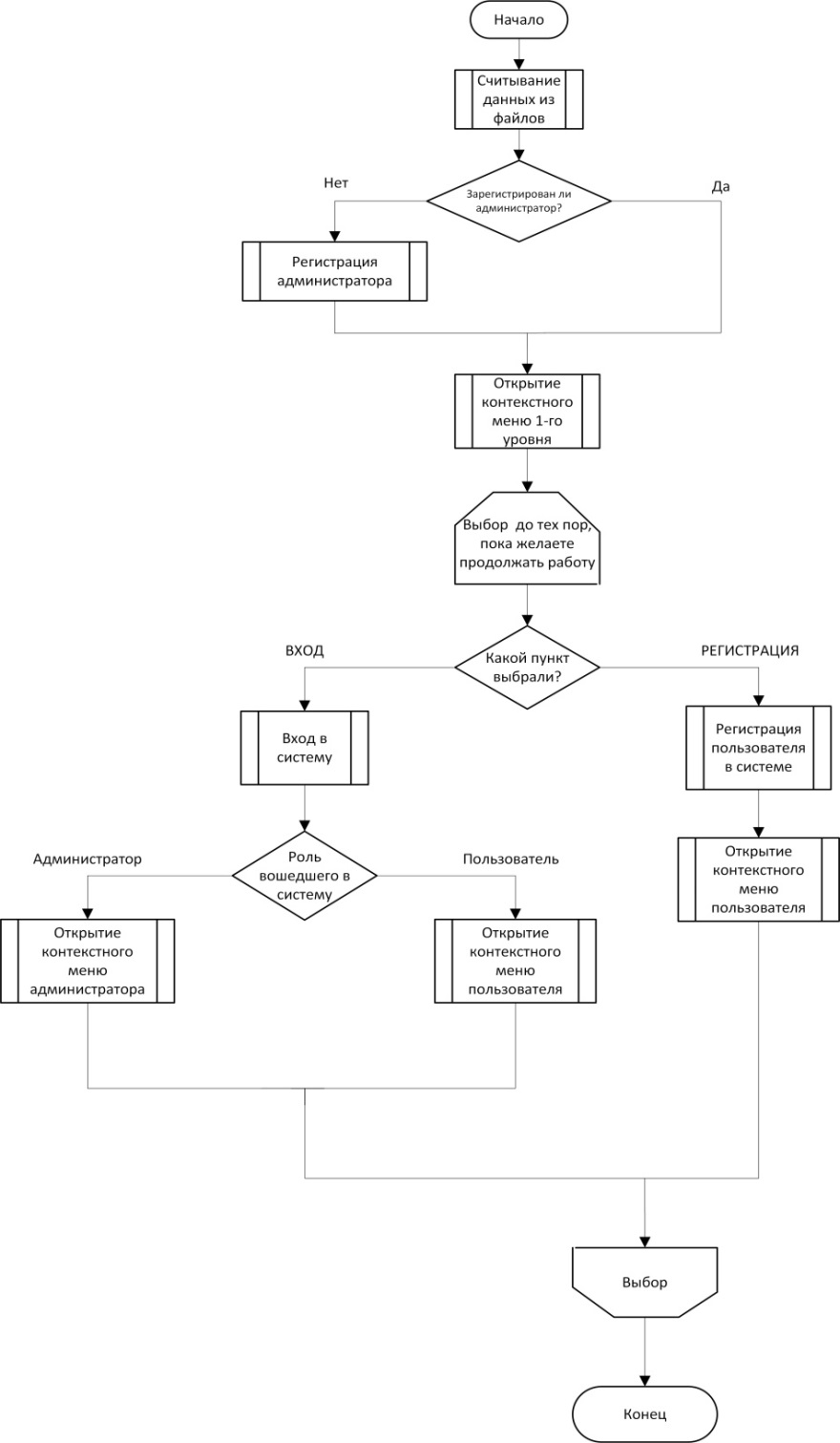
****

Рисунок 6.1 – Блок схема работы приложения

Важным аспектом разработанного программного приложения является взаимосвязь с базой данных. Для хранения и управления данными была выбрана база данных MySQL. Работа взаимодействия сервера с базой данных представлена на рисунке 6.2.



Рисунок 6.2 – Блок-схема взаимодействия сервера с базой данных

Блок-схема бизнес-логики проектируемой системы предоставлена на рисунке 6.3. Одним из примеров это расчет чистой прибыли за прокат фильма. Администратору необходимо выбрать фильм, ввести его ID. Если данные не проходят валидацию, то работа приложения прерывается, администратора информирует определенное сообщение. Если данные введены корректно, то отправляется запрос на сервер, на котором высчитывается чистая прибыль от проката фильма таким путем, что от прибыли отнимаются затраты на энергию и другие ресурсы. Далее сервер отправляет сообщение с результатом и успешным статусом выполнения. Информация о прибыли выводится на экран.

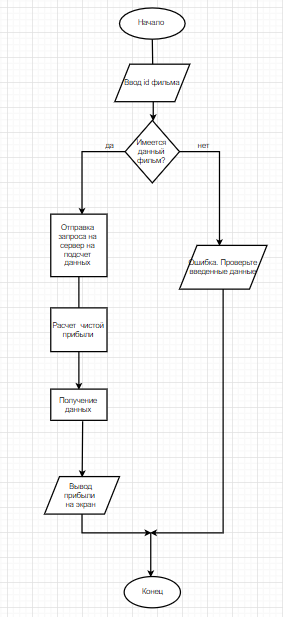


Рисунок 6.3 – Блок-схема расчета чистой прибыли за прокат фильм

1. **руководство пользователя АВТОМАТИЗИРОВАННоЙ СИСТЕМЫ РАБОТЫ КИНОТЕАТРА**

**7.1 Результаты тестирования разработанной системы**

При запуске появляется главное меню предлагающие авторизацию как пользователя или администратора (рисунок 5.1).

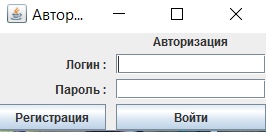


Рисунок 7.1 – Меню Авторизации

При авторизации как администратор, мы попадаем в меню (рисунок 7.2). Администратор имеет возможность добавлять, удалять, изменять фильмы и добавлять их в прокат, рассчитывать среднюю продолжительность фильмов, статистику стоимости билетов и среднее количество фильмов в прокате за последний год.

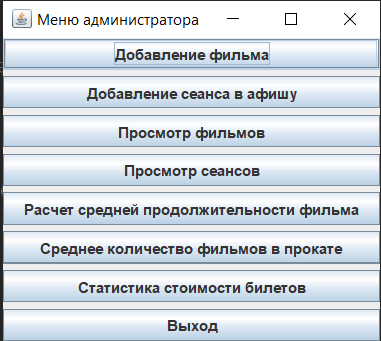


Рисунок 7.2 – Меню Администратора

При нажатии кнопки «Добавление фильма» попадаем в меню добавления фильма (рисунок 7.3). Для добавления фильма необходимо ввести название фильма, жанр, режиссера, главные роли.

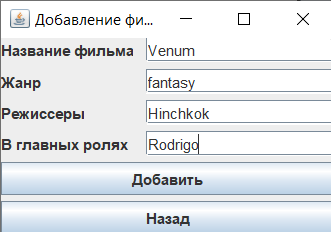


Рисунок 7.3 – Меню Администратора

После добавления фильма администратор может посмотреть добавленные фильмы и в противном случае удалить фильм (рисунок 7.4). Для удаления фильма необходимо ввести соответствующий id. Так же можно рассчитать прибыль определенного фильма (рисунок 7.5).

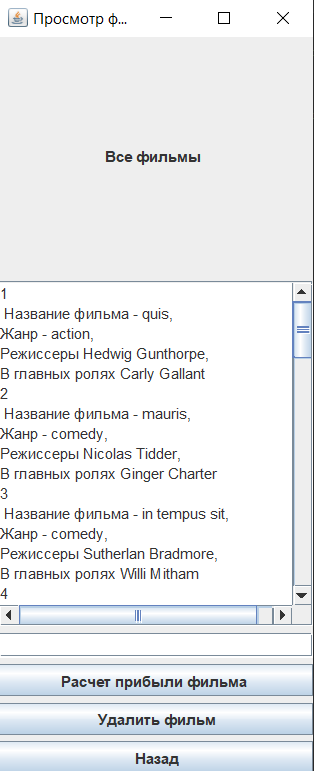


Рисунок 7.4. – Меню добавления фильма

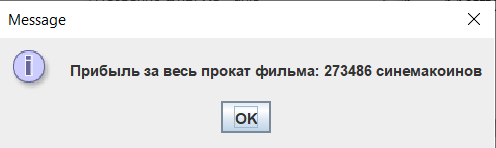


Рисунок 7.5. – Расчет прибыли фильма за месяц

В меню администратора присутствует возможность фильма в афишу (рисунок 7.6). Здесь понадобится ввести название фильма, дату и время, и стоимость билета.

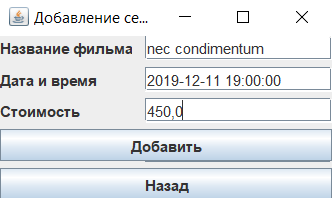


Рисунок 7.5. – Добавления фильма в афишу

Далее имеем возможность просмотреть все сеансы (рисунок 7.7.). В этом пункте так же можем посчитать прибыль за прошедший сеанс определенного фильма(рисунок 7.6.).

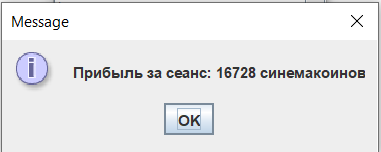


Рисунок 7.6 – Расчет прибыли за сеанс

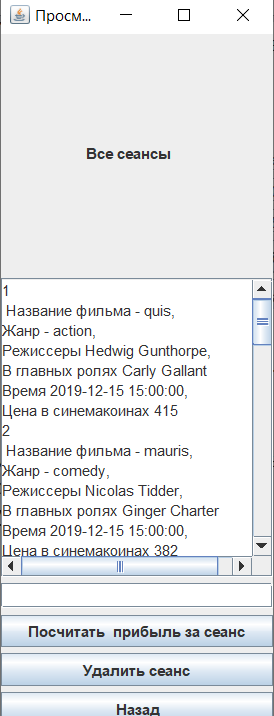


Рисунок 7.7. – Просмотр сеансов

В этом окне можно так же удалить сеанс, для этого нужно выбрать пункт меню «Удалить сеанс»(рисунок 7.8.). Для сеанса необходимо ввести id сеанса.

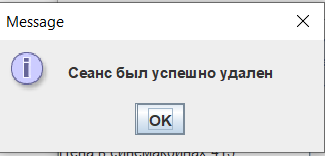


Рисунок 7.8 – Удаление сеанса

Выбрав в меню пункт «Расчет средней продолжительности фильма», будет производен расчет, после нажатия на кнопку (рисунок 7.9).

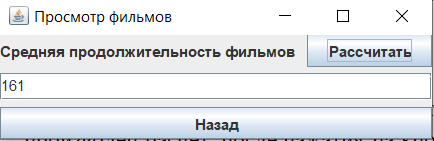


Рисунок 7.9 – Расчет средней продолжительности фильмов

При выборе пункта меню «Количество жанров в прокате», сможем увидеть график, который показывает какое количество жанров имеется в прокате.(Рисунок 7.10).



Рисунок 7.10 – График количество жанров в прокате

Рассмотрим возможности пользователя. После регистрации открывается меню пользователя (рисунок 7.11).Пользователь может посмотреть афишу, добавить фильмы в корзину и удалить их оттуда, купить билет, пополнить баланс.

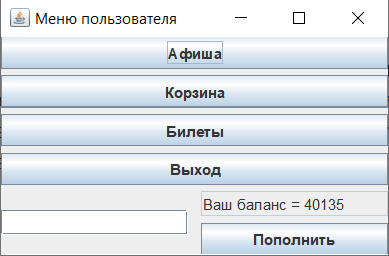


Рисунок 7.11 – Меню пользователя

При просмотре афиши перед нами будет список фильмов в прокате. Тут можно добавить фильм в корзину по определенному номеру (рисунок 7.12).

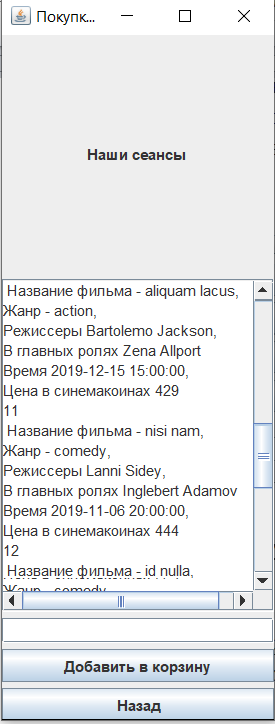


Рисунок 7.12 – Афиша

В пункте «Корзина» мы увидим список фильмов, добавленных в корзину. Здесь можно отменить заказ или купить билет, если хватает денег. После покупки билета, фильм будет удален из корзины и снимутся деньги (рисунок 7.13).

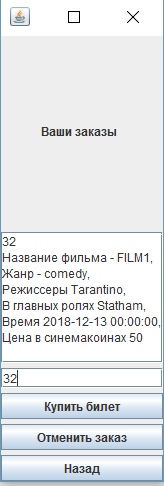


Рисунок 7.13 – Корзина

После покупки билета пользователь может посмотреть список купленных билетов(рисунок 7.14).

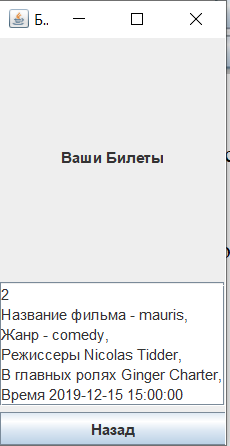


Рисунок 7.14 – Билеты

Если пользователю не хватает средств на покупку билета, он имеет возможность пополнить свой баланс при нажатии кнопки «Пополнить» (рисунок 7.15).

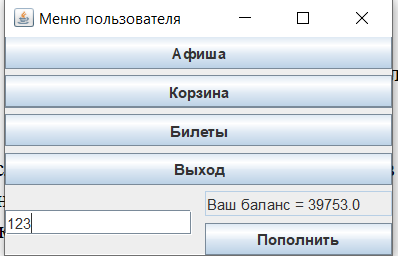


Рисунок 7.15 – Пополнение баланса

В этой главе рассмотрели функционал работы приложения. Он будет понятен даже новичку.

# **Руководство пользователя по развертыванию системы автоматизации работы кинотеатра**

Для развертывания системы необходимо установить программу MySQL Workbench.

Скрипт создания базы данных предоставлен ниже. Для его запуска надо зайти в File – Open SQL Script. И далее нажать на значок молнии и база данных сгенерируется.

Следующим пунктом будет развертывание приложения. Клиент-серверное приложение представлено двумя JAR-файлами. Основная их задача – хранить файлы с классами. Первым должен запускаться файл Server, после него файл Client. Многопоточность сервера обеспечивает возможность запускать несколько клиентов одновременно. Если настройка приложения была выполнена правильно, то пользователь может приступать к работе приложения.

-- MySQL Script generated by MySQL Workbench

-- Tue Dec 10 08:31:43 2019

-- Model: New Model Version: 1.0

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema vlados

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema vlados

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `vlados` DEFAULT CHARACTER SET utf8 ;

USE `vlados` ;

-- -----------------------------------------------------

-- Table `vlados`.`film`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `vlados`.`film` (

`id` BIGINT(20) NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NULL,

`genre` VARCHAR(45) NOT NULL,

`producers` VARCHAR(255) NOT NULL,

`main\_roles` VARCHAR(255) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `vlados`.`session`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `vlados`.`session` (

`id` BIGINT NOT NULL AUTO\_INCREMENT,

`date\_time` DATETIME NOT NULL,

`price` DECIMAL(10,0) NOT NULL,

`film\_id` BIGINT(20) NOT NULL,

PRIMARY KEY (`id`, `film\_id`),

INDEX `fk\_session\_film\_idx` (`film\_id` ASC) VISIBLE,

CONSTRAINT `fk\_session\_film`

FOREIGN KEY (`film\_id`)

REFERENCES `vlados`.`film` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `vlados`.`user`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `vlados`.`user` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`login` VARCHAR(45) NOT NULL,

`password` VARCHAR(45) NOT NULL,

`role` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `vlados`.`basket`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `vlados`.`basket` (

`user\_id` INT(11) NOT NULL AUTO\_INCREMENT,

`session\_id` BIGINT NOT NULL,

PRIMARY KEY (`user\_id`, `session\_id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `vlados`.`client`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `vlados`.`client` (

`user\_id` INT(11) NOT NULL AUTO\_INCREMENT,

`cash` DECIMAL(10,0) NULL,

PRIMARY KEY (`user\_id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `vlados`.`ticket`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `vlados`.`ticket` (

`user\_id` INT(11) NOT NULL AUTO\_INCREMENT,

`session\_id` BIGINT NOT NULL,

PRIMARY KEY (`user\_id`, `session\_id`))

ENGINE = InnoDB

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

1. **РЕЗУЛЬТАТЫ ТеСТИРОВАНИЯ РАЗРАБОТАННОЙ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ РАБОТЫ КИНОТЕАТРА**

При разработке программы были разработаны исключительные ситуации, которые помогают избежать некорректный ввод данных. В этой главе рассмотрим их работу.

Если при входе введете незарегистрированный логин, то появится сообщение о том, что такой пользователь не существует (рисунок 8.1).

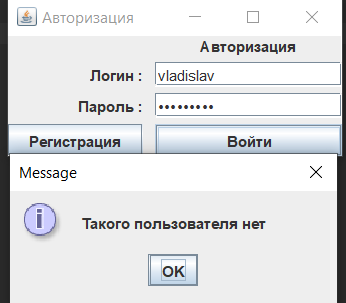


Рисунок 8.1 – Ошибка авторизации

При попытке добавить сеанс, если ввести названия фильма не имеющегося в каталоге, то появится сообщение об ошибке (рисунок 8.2).

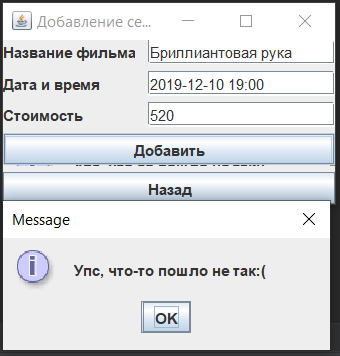


Рисунок 8.2 – Ошибка добавления сеанса

Таким образом все ошибки были исправлены, а недостатки устранены.

# **ЗАКЛЮЧЕНИЕ**

Актуальность выбранной мной темы растет с каждым днем. Посещение кинотеатра уже с прошлого века является одним из лучших развлечений во всем мире. Ввиду этого созданный программный продукт будет так же актуален еще долгое время. После детального исследования предметной области и нахождения оптимальных и наилучших путей началась работа по созданию.

В ходе работы, как итог, было разработано полноценное клиент-серверное приложение. Легко в использовании за счет использования GUI. Благодаря использованию современных методов программирования, данное приложение становится легко расширяемым.

Автоматизированная система работы кинотеатра позволяет вести полный контроль над продажей билетов на сеансы в онлайн-режиме, а также добавлять, редактировать и удалять информацию о сеансах, что позволит сократить потери рабочего времени на ведение документации.

Приложение способно помогать в данной области, так как имеются проверки на правильность и корректность, в случаи срабатывания, которые помогают справится пользователю с проблемой.

Таким образом по завершении написания данной работы можно утверждать, что все поставленные цели и задачи были выполнены.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Пахолкова А. Ю. Анализ новых информационных технологий, используемых в кинематографе [Текст].

[2] Т.В. Алесинская. «Основы Java», ТРТУ, 2005 г.

[3] [Электронный ресурс]. – <https://ru.wikipedia.org>.

[4] [Электронный ресурс]. – <https://tagilcity.ru>.

[5] [Электронный ресурс]. – <https://www.ibm.com>.

[6] [Электронный ресурс]. – <https://www.codeguru.com.ua/article/a-204.html>.

[7] [Электронный ресурс]. – <https://www.journaldev.com>.

[8] [Электронный ресурс]. – https://www. <http://trade.bobrodobro.ru>

[9] [Электронный ресурс]. - <https://www.theconversation.com>

[10] [Электронный ресурс]. - <https://refactoring.guru/>

[11] [Электронный ресурс]. - <http://www.sunsetdriveincinema.com/>

[12] [Электронный ресурс]. - <https://economictimes.indiatimes.com/>

[13] [Электронный ресурс]. - <https://missgeeky.com/>

[14] [Электронный ресурс]. - <https://www.denofgeek.com/>

# **ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Функциональное моделирование**

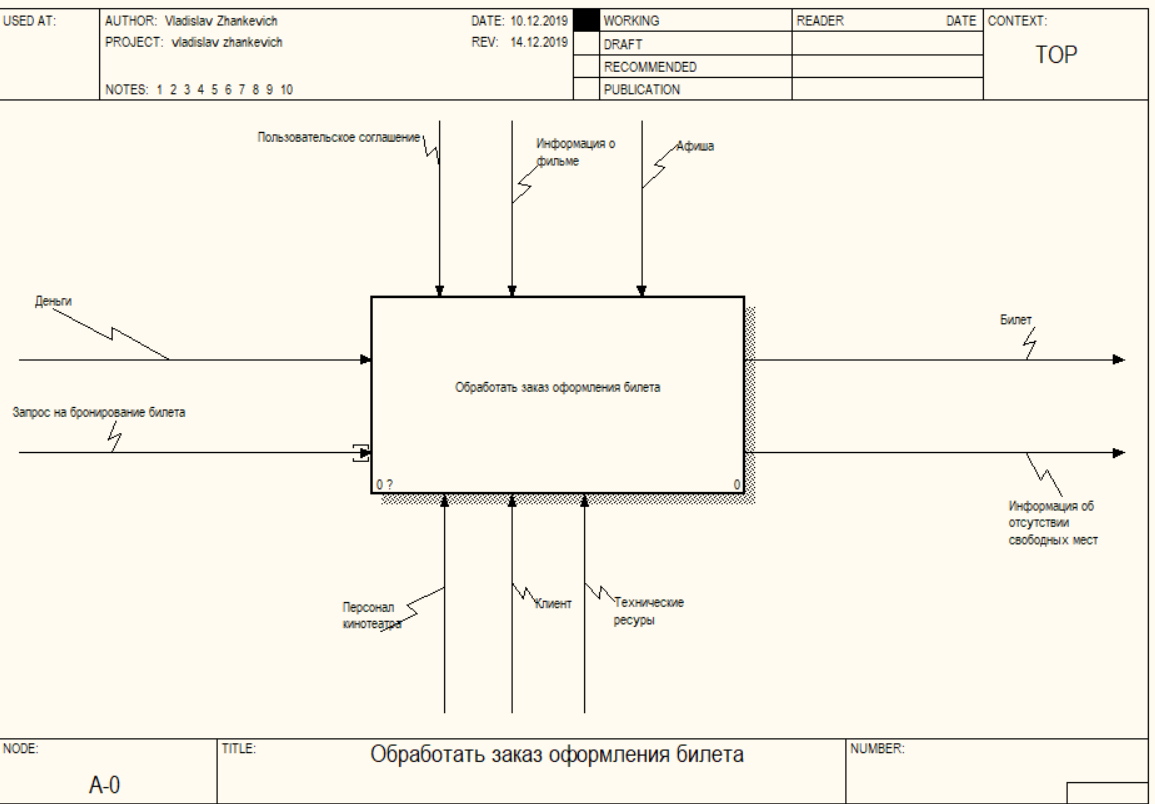


Рисунок А.1 – Диаграмма верхнего уровня

Продолжение приложения А

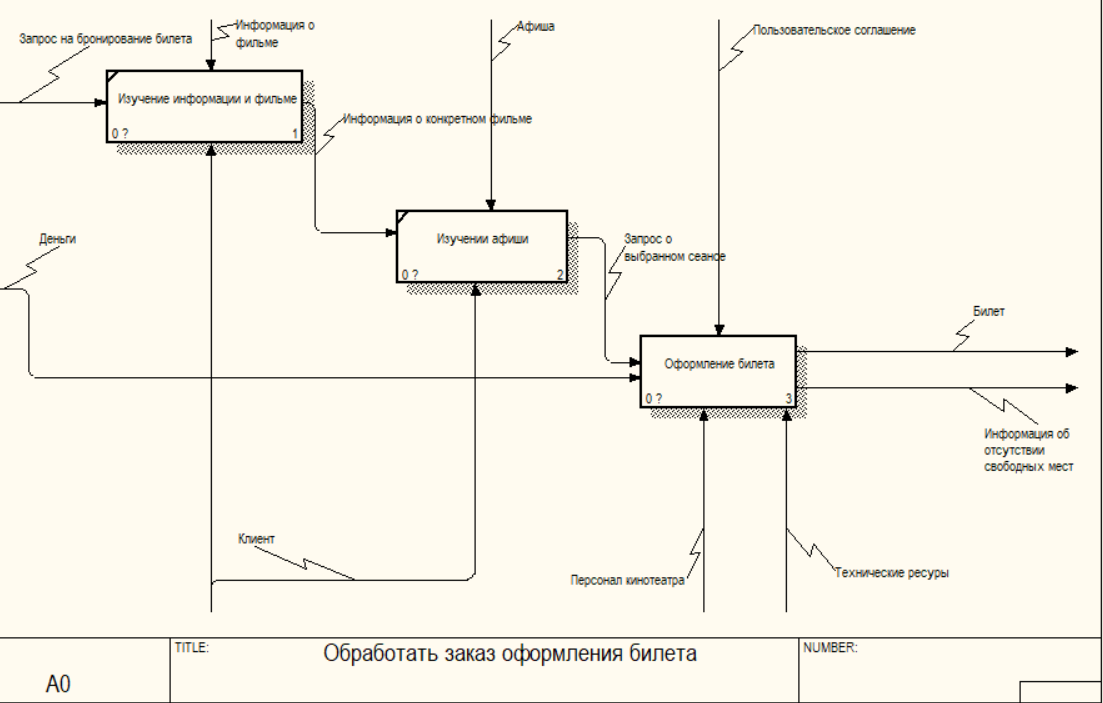


Рисунок А.2 – Декомпозиция контекстной диаграммы верхнего уровня

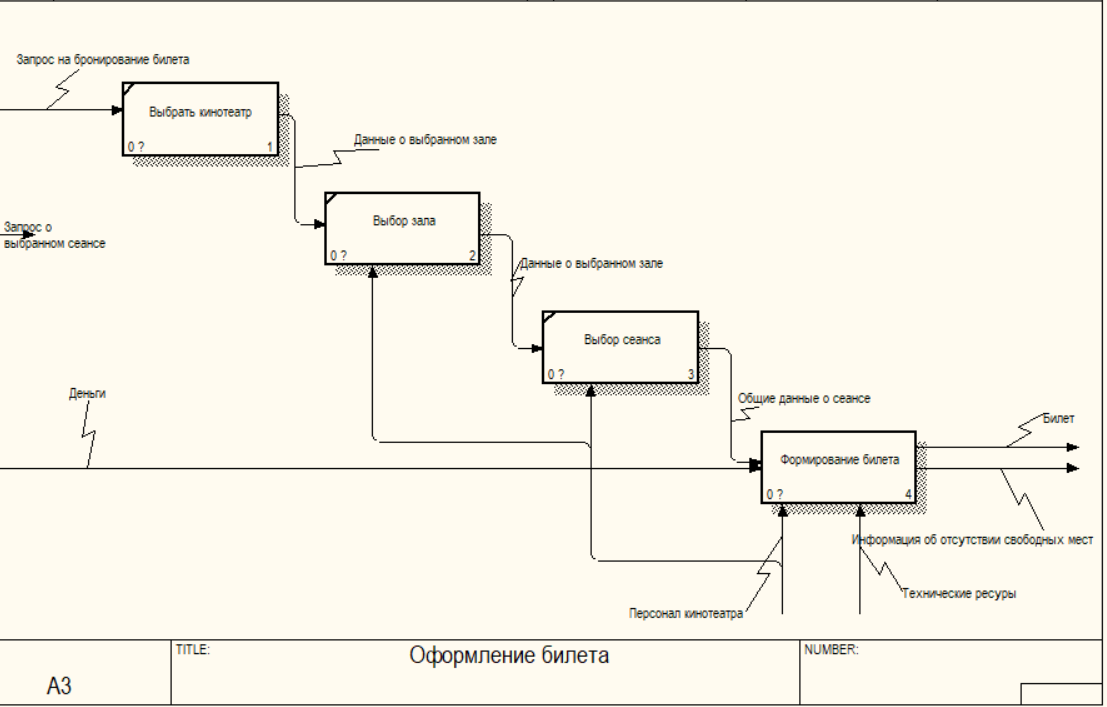


Рисунок А.3 – Декомпозиция блока «Оформление билета»

Продолжение приложения А

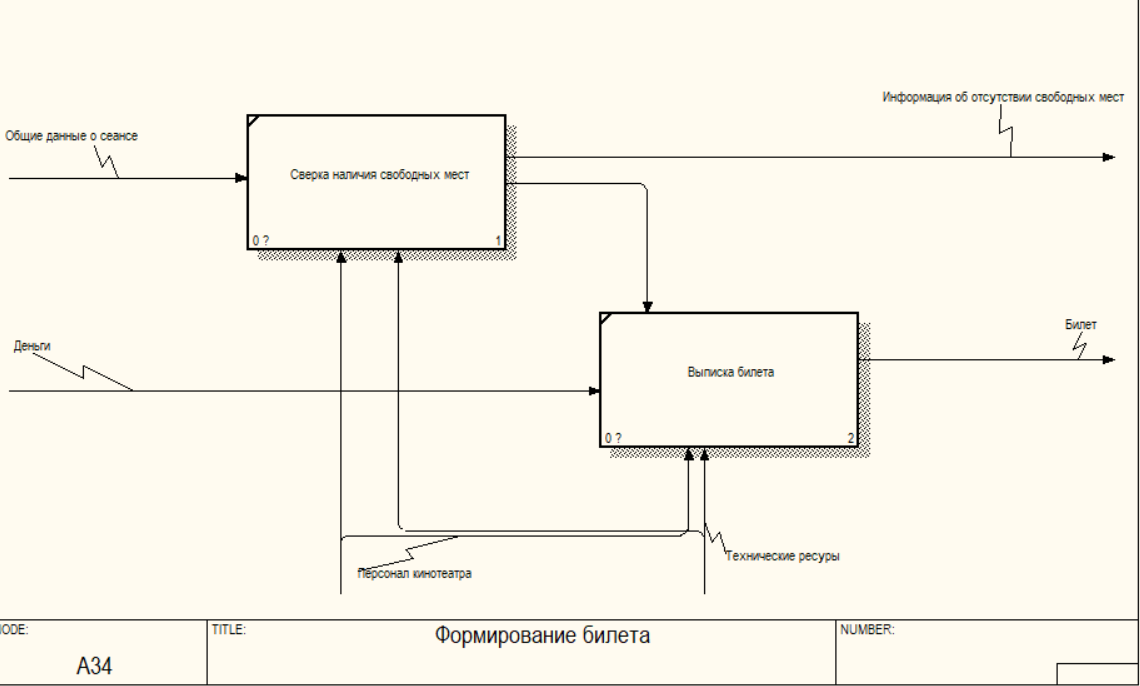
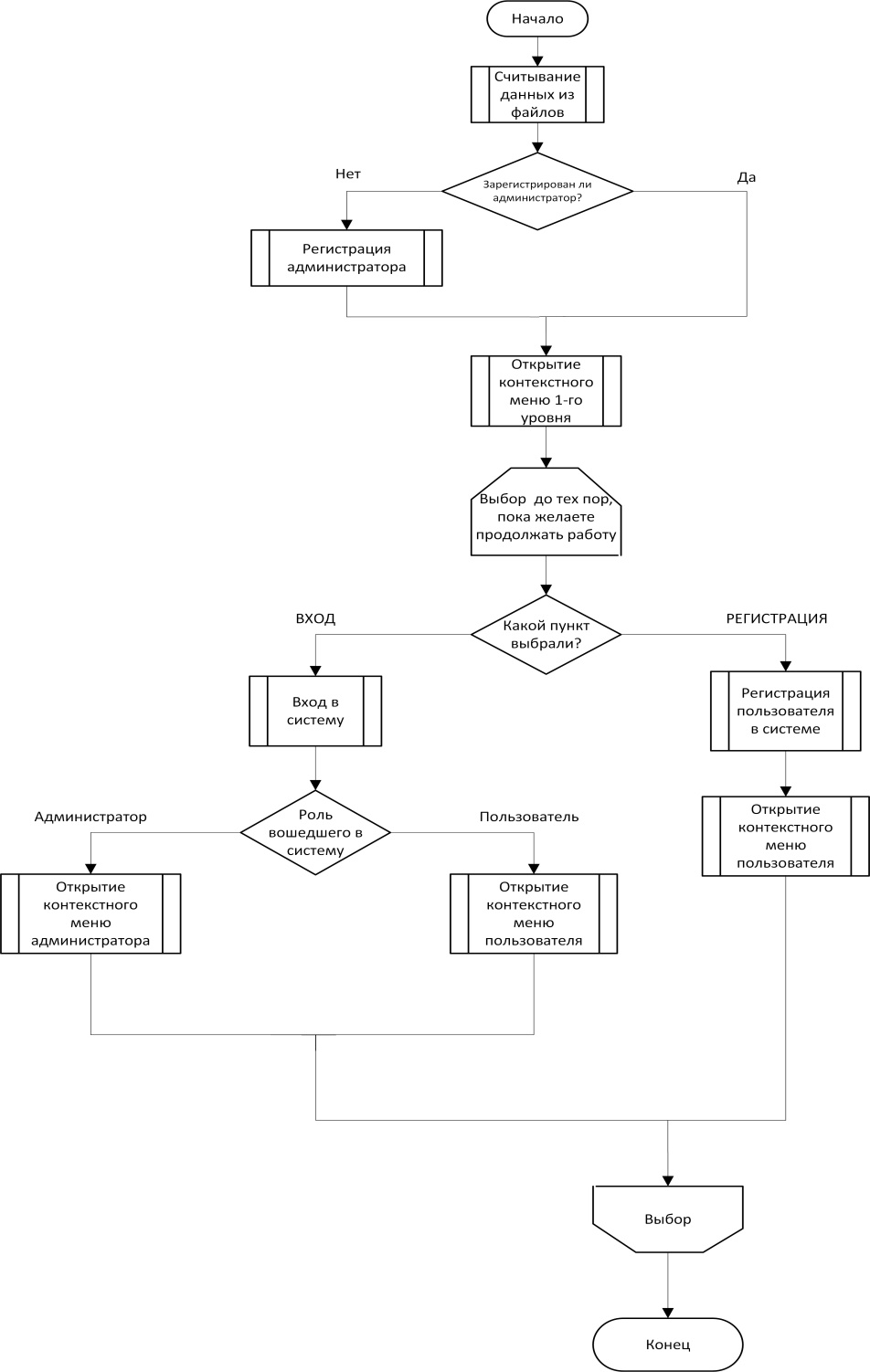


Рисунок А.4 – Декомпозиция контекстной «Формирование билета»

**Приложение Б**

**(обязательное)**

**Схема алгоритма**

****

# Рисунок Б.1 – Обобщенная блок-схема приложения

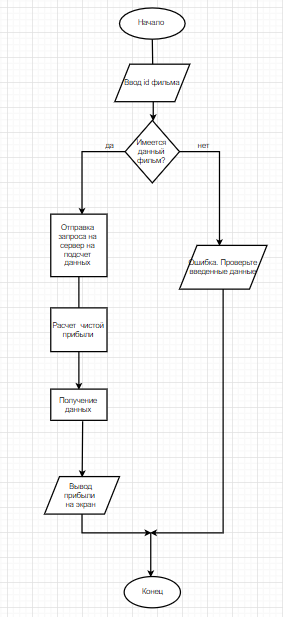
Продолжение приложения Б



# 

# Рисунок Б.2 – Блок-схема связи сервера с базой данных

# Продолжение приложения Б



# Рисунок Б.3 – Блок-схема бизнес функции

**ПРИЛОЖЕНИЕ В**

**(обязательное)**

**Листинг кода**

package by.bsuir.cinema.controller;

import by.bsuir.cinema.entity.user.User;

import by.bsuir.cinema.exception.ProjectException;

import by.bsuir.cinema.jframe.admin.AdminMenuFrame;

import by.bsuir.cinema.jframe.client.SignUpFrame;

import by.bsuir.cinema.logic.UserLogic;

import by.bsuir.cinema.jframe.client.UserMenuFrame;

import by.bsuir.cinema.util.Encryption;

import javax.swing.\*;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.OutputStream;

import java.net.InetAddress;

import java.net.Socket;

public class Starter {

private static final int X = 650;

private static final int Y = 300;

private static final int WIDTH = 1000;

private static final int HEIGHT = 300;

public static User user;

public JPanel authorizationPanel;

private JButton signIn;

private JLabel authtorlabel;

private JLabel login;

private JLabel password;

private JTextField loginText;

private JPasswordField passwordField;

private JButton signUpButton;

private static Socket connection;

private static OutputStream output;

private static DataOutputStream dataOutputStream;

public static void main(String[] args) throws IOException {

connection = new Socket(InetAddress.getByName("127.0.0.1"), 3351);

output = connection.getOutputStream();

dataOutputStream = new DataOutputStream(output);

JFrame authorizationFrame = new JFrame("Авторизация");

authorizationFrame.setBounds(X, Y, WIDTH, HEIGHT);

authorizationFrame.setContentPane(new Starter(authorizationFrame, dataOutputStream).authorizationPanel);

authorizationFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

authorizationFrame.pack();

authorizationFrame.setVisible(true);

}

public Starter(JFrame authorizationFrame, DataOutputStream output) {

signIn.addActionListener(e -> {

try {

user = UserLogic.findUser(loginText.getText(), Encryption.encryptPassword(passwordField.getText()));

} catch (ProjectException e1) {

e1.printStackTrace();

}

if (user != null) {

switch (user.getType()) {

case ADMIN:

authorizationFrame.dispose();

openAdminMenu(dataOutputStream);

break;

case CLIENT:

authorizationFrame.dispose();

openUserMenu(dataOutputStream);

break;

default:

JOptionPane.showMessageDialog(null,

"Ошибка");

break;

}

} else {

JOptionPane.showMessageDialog(null,

"Такого пользователя нет");

}

});

signUpButton.addActionListener(e -> {

authorizationFrame.dispose();

openSignUpMenu(dataOutputStream);

});

}

private void openAdminMenu(DataOutputStream dataOutputStream) {

JFrame frame = new JFrame("Меню администратора");

frame.setBounds(650, 300, 15000, 300);

frame.setContentPane(new AdminMenuFrame(frame, dataOutputStream).adminMenu);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.pack();

frame.setVisible(true);

new Thread(new AdminMenuFrame(dataOutputStream)).start();

}

private void openUserMenu(DataOutputStream dataOutputStream) {

JFrame frame = new JFrame("Меню пользователя");

frame.setBounds(650, 300, 15000, 300);

frame.setContentPane(new UserMenuFrame(frame, dataOutputStream).userMenu);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.pack();

frame.setVisible(true);

new Thread(new UserMenuFrame(dataOutputStream)).start();

}

private void openSignUpMenu(DataOutputStream dataOutputStream) {

JFrame frame = new JFrame("Меню регистрации");

frame.setBounds(650, 300, 15000, 300);

frame.setContentPane(new SignUpFrame(frame, dataOutputStream).signUpPanel);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.pack();

frame.setVisible(true);

}

}

public class FilmDao extends AbstractDao {

    private static final String INSERT\_FILM = "insert into Film(name, genre, producers, main\_roles) values(?, ?, ?, ?)";

    private static final String FIND\_ID\_BY\_NAME = "select id from Film where name = ?";

    private static final String FIND\_ALL = "select \* from Film";

    private static final String DELETE\_FILM\_BY\_ID = "delete from Film where id = ?";

    private static final String find\_comedyclub = "select \* from Film where genre = ?";

    private static final String FIND\_COMEDY = "SELECT COUNT(\*) FROM Film WHERE `genre`='comedy'";

    private static final String FIND\_FANTASY = "SELECT COUNT(\*) FROM Film WHERE `genre`='fantasy'";

    private static final String FIND\_THRILLER = "SELECT COUNT(\*) FROM Film WHERE `genre`='thriller'";

    public boolean insertFilm(String name, String filmGenre, String producers, String mainRoles) throws ProjectException {

        PreparedStatement preparedStatement = null;

        boolean flag;

        try{

            preparedStatement = connection.prepareStatement(INSERT\_FILM);

            preparedStatement.setString(1, name);

            preparedStatement.setString(2, filmGenre);

            preparedStatement.setString(3, producers);

            preparedStatement.setString(4, mainRoles);

            flag = preparedStatement.executeUpdate() != 0;

        } catch (SQLException e) {

            throw new ProjectException("SQLException, ", e);

        } finally {

            if (connection != null){

                close(preparedStatement);

            }

        }

        return flag;

    }

public int find\_comedy() throws ProjectException {

    PreparedStatement statement = null;

    int count\_comedy = 0;

    String str = "SELECT COUNT(\*) FROM Film WHERE `genre`='comedy'";

    try {

        statement = connection.prepareStatement(str);

        ResultSet rs = statement.executeQuery();

        //ResultSet rs = statement.executeQuery(FIND\_COMEDY);

        rs.next();

        count\_comedy = rs.getInt(1);

    } catch (SQLException e) {

        throw new ProjectException("SQLException, ", e);

    } finally {

        if (connection != null) {

            close(statement);

        }

    }

    return count\_comedy;

}

    public int find\_fantasy() throws ProjectException {

        PreparedStatement statement = null;

        int count\_fantasy= 0;

        String str = "SELECT COUNT(\*) FROM Film WHERE `genre`='fantasy'";

        try {

            statement = connection.prepareStatement(str);

            ResultSet rs = statement.executeQuery();

            //ResultSet rs = statement.executeQuery(FIND\_COMEDY);

            rs.next();

            count\_fantasy = rs.getInt(1);

        } catch (SQLException e) {

            throw new ProjectException("SQLException, ", e);

        } finally {

            if (connection != null) {

                close(statement);

            }

        }

        return count\_fantasy;

    }

**ПРИЛОЖЕНИЕ Г**

**(обязательное)**

**Листинг скрипта генерации базы данных**

-- MySQL Script generated by MySQL Workbench

-- Tue Dec 10 08:31:43 2019

-- Model: New Model Version: 1.0

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema vlados

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema vlados

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `vlados` DEFAULT CHARACTER SET utf8 ;

USE `vlados` ;

-- -----------------------------------------------------

-- Table `vlados`.`film`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `vlados`.`film` (

`id` BIGINT(20) NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NULL,

`genre` VARCHAR(45) NOT NULL,

`producers` VARCHAR(255) NOT NULL,

`main\_roles` VARCHAR(255) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `vlados`.`session`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `vlados`.`session` (

`id` BIGINT NOT NULL AUTO\_INCREMENT,

`date\_time` DATETIME NOT NULL,

`price` DECIMAL(10,0) NOT NULL,

`film\_id` BIGINT(20) NOT NULL,

PRIMARY KEY (`id`, `film\_id`),

INDEX `fk\_session\_film\_idx` (`film\_id` ASC) VISIBLE,

CONSTRAINT `fk\_session\_film`

FOREIGN KEY (`film\_id`)

REFERENCES `vlados`.`film` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `vlados`.`user`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `vlados`.`user` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`login` VARCHAR(45) NOT NULL,

`password` VARCHAR(45) NOT NULL,

`role` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `vlados`.`basket`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `vlados`.`basket` (

`user\_id` INT(11) NOT NULL AUTO\_INCREMENT,

`session\_id` BIGINT NOT NULL,

PRIMARY KEY (`user\_id`, `session\_id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `vlados`.`client`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `vlados`.`client` (

`user\_id` INT(11) NOT NULL AUTO\_INCREMENT,

`cash` DECIMAL(10,0) NULL,

PRIMARY KEY (`user\_id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `vlados`.`ticket`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `vlados`.`ticket` (

`user\_id` INT(11) NOT NULL AUTO\_INCREMENT,

`session\_id` BIGINT NOT NULL,

PRIMARY KEY (`user\_id`, `session\_id`))

ENGINE = InnoDB

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;