Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Дисциплина «Технология проектирования сложных информационных систем»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.Н. Тонкович |
|  | \_\_\_.\_\_\_\_.2024 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«Проектирование и разработка программного средства управления деятельностью   
кинотеатра-мультиплекса на основе автоматизации ключевых бизнес-процессов»**

БГУИР КП 1-40 05 01-10 003 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 214302  ВАСИЛЕВИЧ Полина Максимовна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку ................ 2024  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2024

**РЕФЕРАТ**

БГУИР КП 1-40 05 01-10 003 ПЗ

**СОДЕРЖАНИЕ**

[Введение 6](#_Toc163903586)

[1 Анализ литературных исследований и программных решений 8](#_Toc163903587)

[1.1 Описание и анализ предметной области 8](#_Toc163903588)

[1.2 Обзор функциональности аналогов программного средства 11](#_Toc163903589)

[2 Моделирование предметной области и разработка требований к программному средству 16](#_Toc163903590)

[2.1 Анализ и формализация бизнес-процессов предметной области 16](#_Toc163903591)

[2.2 Анализ требований к разрабатываемому программному средству и разработка их спецификации 18](#_Toc163903592)

[2.3 Образ предлагаемого решения 42](#_Toc163903593)

[3 Проектирование и разработка программного средства 44](#_Toc163903594)

[3.1 Архитектурные решения 44](#_Toc163903595)

[3.2 Обоснование выбора компонентов и технологий для реализации программного средства 46](#_Toc163903596)

[3.3 Проектирование и разработка пользовательского интерфейса 48](#_Toc163903597)

[3.4 Разработка модели данных 60](#_Toc163903598)

[3.5 Описание статических и динамических аспектов поведения программных объектов 70](#_Toc163903599)

[3.6 Разработка и описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства 75](#_Toc163903600)

[3.7 Разработка программного средства 78](#_Toc163903601)

[3.8 Механизмы обеспечения информационной безопасности 84](#_Toc163903602)

[4 Тестирование и проверка работоспособности программного средства 87](#_Toc163903603)

[Список использованных источников 99](#_Toc163903604)

# ВВЕДЕНИЕ

В настоящее время, с ростом технологий и увеличением количества кинолюбителей, процесс управления контентом кинотеатров становится все более важным и требует удобных и эффективных инструментов. Оптимизация и цифровизация бизнес-процессов кинотеатров приобретает актуальность в свете необходимости уменьшения их издержек и количества ошибок в их работе, а также снижения нагрузки на сотрудников кинотеатра. Таким образом, разработка программного средства автоматизации бизнес-процессов кинотеатра-мультиплекса является важной задачей.

Цель данного курсового проекта заключается в разработке программного средства, которое позволит автоматизировать и упростить процессы управления контентом кинотеатра. Данное программное средство охватывает несколько ключевых аспектов: обеспечение ролей администратора и пользователя, поиск информации о фильмах, добавление информации о кинозалах, создание сеансов на определенное время в определенном зале, составление и изменение расписания.

Для достижения поставленной цели надо решить следующие задачи:

1.Выполнить обзор и анализ предметной области, а также существующих программных решений.

2.Осуществить моделирование предметной области и разработать требования к программному средству.

3 Спроектировать, разработать и протестировать программное средство.

В основу разработки данного программного средства положены принципы эффективной автоматизации процесса работы с контентом кинотеатра, учета требований безопасности и удобства пользователей. Проект будет разработан с применением таких технологий, как *Java*, *Spring* *Boot*, *Kinopoisk API* и *MySQL*, что обеспечит высокую производительность и надежность программного средства.

Таким образом, предлагаемое программное средство позволит улучшить работу кинотеатров, повысив ее эффективность, и предоставит сотрудникам кинотеатров удобные средства для управления их контентом.

Курсовой проект состоит из введения, пяти разделов с краткими выводами по каждому разделу, заключения, списка использованных источников, приложений и ведомости.

В первом разделе приведен обзор современного состояния проблемы управления деятельностью кинотеатра-мультиплекса на основе автоматизации его бизнес-процессов, проанализированы аналоги программного средства.

Во втором разделе представлены анализ и моделирование предметной области с формализацией ее основных бизнес-процессов. На основании выполненного анализа предметной области сформулированы предложения по

совершенствованию рассмотренных бизнес-процессов. Определены бизнес-требования. Разработаны спецификации пользовательских, функциональных и нефункциональных требований к программному средству.

В третьем разделе предложены архитектурные решения, выполнены проектирование и разработка пользовательского интерфейса, показана реализация модели данных, представлена разработка проектных решений на основе моделирования свойств и линий поведения программных объектов, разработаны алгоритмические и программные реализации модуле программного средства.

В четвертом разделе представлены доказательства работоспособности программного средства на основе тестовых примеров.

В пятом разделе разработано руководство по развертыванию и использованию программного средства.

В приложениях представлены отчет о проверке на заимствования в системе «Антиплагиат» и текст программы.

Курсовой проект выполнен самостоятельно, проверен в системе «Антиплагиат». Процент оригинальности составляет …..%. Цитирования обозначены ссылками на публикации, указанными в «Списке использованных источников».

Скриншот приведен в приложении А на рисунке А.1.

# АНАЛИЗ ЛИТЕРАТУРНЫХ ИССЛЕДОВАНИЙ И ПРОГРАММНЫХ РЕШЕНИЙ

## Описание и анализ предметной области

Кино – одно из самых молодых и в то же время одно из самых массовых искусств. Из всех видов искусства оно занимает уникальное место. Кино передает наши чувства, привычки, обычаи, погружает человека в мир его иллюзий. По кинематографу можно рассмотреть историю целых поколений, десятилетий. Фильмы позволяют нам узнать о событиях, которые происходили очень давно. Кино способно охватывать почти все сферы общественного сознания. Оно заставляет человека осмысливать жизнь как нечто большее, чем то, как он воспринимал ее до этого [1].

Объем кинорынка с 2000 года неуклонно растет. Миллионные инвестиции в реконструкцию кинотеатров и строительство мультиплексов стали выгодным бизнесом. Многозальные кинотеатры вытеснили однозальные, оказавшись более рентабельными. Доходность коммерческих кинотеатров в Москве сегодня оценивается в 25% годовых (это значит, что окупаемость составляет 3 – 4 года, при средней загрузке зала от 30%). В лучших московских кинотеатрах заполняемость достигает 70 – 80%. Сейчас мультиплексы, ставшие для зрителя не только местом просмотра фильмов, но и местом общения, способны удовлетворить любые запросы потребителей.

Кинотеатры с множеством залов, каждый из которых рассчитан на небольшое количество зрительских мест, впервые возникли в США. Их назвали «мультиплексами». Выяснилось, что зрители наиболее комфортно чувствовали себя в небольших залах на 80 – 120 человек. В результате появился многозальный кинотеатр, состоящий из нескольких небольших (на 60–150 мест) кинозалов с повышенным уровнем комфорта. Преимущество формата многозального кинотеатра – гибкая репертуарная политика с одновременным показом разных фильмов или демонстрацией одного фильма в разных залах с интервалом в 20 – 30 минут. Формат быстро завоевал популярность в Западной Европе и в России.

Преобладающей тенденцией на европейском кинорынке последних лет стало активное строительство кинотеатров с общим количеством экранов 10 и более. Во Франции почти 70% всех кинотеатров являются мультиплексами, в Великобритании – более 65%, Испании и Дании – около 30%, Германии – почти 25%, Италии – 20%. По данным зарубежных аналитиков, мультиплексы обеспечивают около 80 – 90% общемировых кассовых сборов.

Число мультиплексов в стране и в мире постепенно увеличивается. По предварительным оценкам *Screen Digest*, в течение года только в Европе появится более 1000 новых киноэкранов. Ожидается, что через 2 года более половины всех экранов Европы будут находиться именно в мультиплексах. В настоящее время этот показатель составляет 46% [2].

Таким образом, можно сделать вывод, что киноиндустрия занимает важную роль в жизни современного общества, и что ее развитие актуально на сегодняшний день.

Для описания и анализа предметной области необходимо знать ее ключевые понятия.

Кинотеатр-мультиплекс – кинотеатр с несколькими, как правило, тремя

или больше кинозалами [3]. Его деятельность заключается в публичной демонстрации фильмов. Ключевыми сущностями кинотеатра являются:

1 Фильм – аудиовизуальное произведение, созданное в художественной, хроникально-документальной, научно-популярной, учебной, анимационной, телевизионной или иной форме, предназначенное для восприятия с помощью соответствующих технических устройств. Они имеют название, продолжительность, популярность, возрастное ограничение и формат (*2D*, *3D*, *5D*).

2 Кинозал – помещение, имеющее необходимое оборудование для публичной демонстрации фильмов. Кинозалы характеризуются вместительностью (количеством мест) и техническими возможностями (поддержка формата *2D*, *3D* и *5D*).

3 Сеанс – процесс показа фильма, осуществляемый в определенный промежуток времени в определенном месте. Для сеанса нужны фильмом для показа, дата и время проведения и зал для показа.

Управление деятельностью кинотеатра-мультиплекса – совокупность действий, направленных на достижение целей предприятия, таких как получение прибыли и снижение затрат [4]. Для управления деятельностью кинотеатров используются системы *Theatre Management System* (*TMS*) – программы управления и осуществления различных операций для кинотеатров. Они позволяет максимально повысить производительность, поскольку представляет собой надежный инструмент для централизованного управления и автоматизации кинокомплексов с несколькими залами. Кроме основных функций, таких как централизованный мониторинг статуса воспроизведения фильмов во всех залах, составление расписания и централизованное управление контентом экрана, *TMS* выполняет такие удобные функции, как синхронизация расписания с системой продажи билетов (*POS*) [5].

TMS состоит из следующих частей:

1 Система управления контентом (*CMS*) - это программное обеспечение для создания, редактирования, организации и публикации контента кинотеатра (фильмов).

2 Система управления данными и ключами (*DKMS*). Она используется для безопасного хранения, управления и резервного копирования криптографических ключей.

3 Система управления показом и расписанием (*SSMS*). Она обеспечивает просмотр, создание, сохранение и редактирование списков воспроизведения контента в конкретном кинозале мультиплекса [6].

В рамках данного курсового проекта рассмотрим бизнес-процессы систем *CMS* и *SSMS* и проблем, которые могут возникнуть при их работе.

Бизнес-процесс – это совокупность взаимосвязанных мероприятий или

работ, направленных на создание определённого продукта или услуги для потребителей [7]. Их эффективное выполнение необходимо для успешного управления деятельностью кинотеатра. В рамках данного курсового проекта необходимо рассмотреть такие бизнес-процессы, как:

– работа с фильмами;

– работа с кинозалами;

– управление расписанием.

Автоматизация бизнес-процессов – это перевод типовых бизнес-задач и стандартных операций под контроль программно-аппаратного комплекса. В результате высвобождаются ресурсы, что позволяет увеличить производительность труда и эффективность стратегического управления. Автоматизация бизнес-процессов нужна для:

1 Увеличения прибыли. Перед настройкой автоматизации нужно проанализировать текущие бизнес-процессы, чтобы найти их слабые места и подобрать инструменты для их усиления. Анализ, оплата сервисов и инструментов требуют финансовых вложений. Однако эти затраты позволяют выстроить работу так, чтобы выполнять больше действий за меньшее количество времени. В результате этого повышается эффективность и растет прибыль.

2 Оптимизации работы сотрудников. У сотрудников появляется больше времени на стратегические и креативные задачи, когда они не отвлекаются на рутинные дела. В результате бизнесу легче развиваться и внедрять инновации. А рост прибыли компании позволяет повышать зарплаты и укреплять лояльность ценных работников.

3 Сокращения ошибок. Специалист настраивает алгоритм один раз. Все дальнейшие операции происходят автоматически, а значит — исключают человеческий фактор.

4 Обеспечения прозрачности и прослеживаемости. Появляется возможность отслеживания данных в режиме реального времени, без потери их актуальности [8].

Современным кинотеатрам-мультиплексам необходим удобный и полезный инструмент эффективного управления контентом. Без такого средства бизнес-процессы кинотеатра будут требовать значительное количество времени, будут подвержены большому количеству ошибок и оказывать большую нагрузку на сотрудников. К примеру, в процессе приобретения фильма сотрудник сам ищет детальную информацию о нем, что занимает много времени. Далее на основе полученной информации он добавляет фильм в расписание. Этот процесс занимает много времени и подвержен большому количеству ошибок, что может привести к неверному или неудобно составленному расписанию. Эта же проблема может возникнуть и при изменении расписания вручную.

Описанные выше проблемы можно решить с помощью автоматизации рассматриваемых бизнес-процессов: поиск информации о фильме будет автоматизирован, поэтому будет происходить гораздо быстрее. Вся информация о фильмах и расписаниях будет хранится в базе, что обеспечивает ее сохранность данных и удобство работы с ними. Использование удобных графических инструментов при работе с расписанием позволит ускорить процесс его создания и изменения и избежать ошибок.

Это показывает потребность в автоматизации ключевых бизнес- процессов кинотеатра для повышения эффективности его работы.

Таким образом, для рассматриваемой предметной области характерны такие проблемы, как увеличение издержек кинотеатров при возникновении ошибок в их работе и большое количество неэффективного рабочего времени сотрудников кинотеатра. По этим причинам необходима автоматизация деятельности кинотеатра. Она играет важную роль в обеспечении его успешной работы, позволяя уменьшить ручную работу сотрудников и избежать большого количества ошибок.

## Обзор функциональности аналогов программного средства

Проблема автоматизации деятельности кинотеатров достаточно актуальна на сегодняшний день, поэтому существует большое количество программных средств, решающих ее. Суть обзора аналогов разрабатываемого программного средства заключается в ознакомлении с их функциональностью. В ходе курсового проекта было рассмотрено три аналога разрабатываемого программного средства: «*UCS*-Премьера», *«*Подкова *X-Frame»* и *TMS*. Данные аналоги были выбраны в результате данных, приведенных агентством *LiveBusiness* [9], порталом об ИТ технологиях для предпринимателей и руководителей компаний, и компанией *Tadviser* – порталом выбора технологий и поставщиков [10].

В таблице 1 представлено описание аналогов разрабатываемого программного средства.

Таблица 1 – Аналоги программного средства

|  |  |  |  |
| --- | --- | --- | --- |
| **Название аналога** | **Компания-разработчик** | ***URL* официального сайта компании-разработчика** | ***URL* источников для обзора** |
| 1 «*UCS*-Премьера» | *UCS* | *https://ucs-spb.ru* | *https://www.tadviser.ru/index.php/Продукт:UCS-Премьера* |
| 2 *«*Подкова *X-Frame»* | «Подкова» | *https://www.pidkova.biz/* | *https://www.livebusiness.ru/tool/2081/* |
| 3 *TMS* | *Dev.House* | *https://dev.house/* | *https://www.tadviser.ru/index.php/Продукт:TMS\_(система\_управления\_кинотеатрами)* |

Ниже в таблице 2 описана функциональность аналогов программного средства автоматизации бизнес-процессов кинотеатра.

Таблица 2 – Функциональность аналогов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Аналог**  **(компания-разра-**  **ботчик)** | **Назначение аналога** | **Функциональность аналога** | **Стоимость** | ***URL*** |
| 1 «*UCS*-Премьера» (*UCS*) | Автомати-зация работы кинотеатра | 1.Ведение списка фильмов.  2 Указание названия и длительности.  3 Возможно указание компании-  прокатчика и | От 12000 российских рублей в месяц | *https://ucs-spb.ru/*  *programms/*  *ucs-premera/* |

Продолжение таблицы 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Аналог**  **(компания-разра-**  **ботчик)** | **Назначение аналога** | **Функциональность аналога** | **Стоимость** | ***URL*** |
| 1 «*UCS*-Премьера» (*UCS*) |  | задание периода проката.  4.Ведение списка сеансов.  5.Создание расписания сеансов.  6 Предупреждение о возможных  нестыковках в графике.  7.Отображение законченных и  незаконченных сеансов в режиме данного времени.  8 Учет рекламных  роликов.  9 Определение цен на билеты.  10.Построение отчетов о прошедших сеансах, выручке, посещаемости и аналитических отчетов.  11.Создание собственных отчетов  пользователем. |  |  |
| 2 *«*Подко-ва *X-Frame»*  (Подкова) | Автомати-зация и управление деятель-ностью кинотеатра | 1 Обслуживание несколько залов просмотров  2 Составление плана зала с возможностью поделить зал на секторы, с последующей | От 200 долларов США в год | https://www.  pidkova  .biz/xframe.  php |

Окончание таблицы 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Аналог**  **(компания-разра-**  **ботчик)** | **Назначение аналога** | **Функциональность аналога** | **Стоимость** | ***URL*** |
| 2 *«*Подко-ва *X-Frame»*  (Подкова) |  | установкой специальных цен на каждый сектор.  3 Составление репертуара с учетом  продолжительности ленты.  4 Составление гибкого расписания сеансов, со спецификой времени сеанса.  5 Аналитический анализ. |  |  |
| 3 *TMS* (*Dev.*  *House)* | Управление контентом кинотеатра | 1 Автоматизация управления ключами безопасности.  2 Автоматизация работы с плейлистами показа на уровне одного зала и всего мультиплекса в целом.  3. Получение статистики и отчетов на уровне одного зала и всего мультиплекса в целом.  4 Немедленное дистанционное информирование об ошибках и сбоях в работе. | На заказ | *https://www.*  *tadviser*  *.ru/index.*  *php/*  *Продукт:*  *TMS*  *\_(система\_*  *Управления*  *\_кинотеа*  *трами)* |

«*UCS*-Премьера», программное обеспечение для сетей и мегаплексов, разработанное компанией *UCS*, – одна из самых популярных билетных систем в России. По статистике РБК на 95% мест в залах московских кинотеатров билеты продаются через программное обеспечение *UCS*-Премьера.Данное программное средство покрывает весь необходимый функционал, но стоимость программного средства слишком высока и не будет выгодна для компании в долгосрочном использовании. Разработать собственное программное средство будет дешевле, а также покроет весь необходимый функционал. Также отсутствует разделение прав доступа, что может привести к нарушению защищенности данных кинотеатра.

*X-Frame*, простое и гибкое решение для автоматизации кинотеатров, театров, различных залов просмотра. Простой и удобный интерфейс программы позволит легко и быстро ее освоить и в тоже время доступная функциональность превращает *X-Frame* в мощный инструмент контроля и анализа.В данном аналоге отсутствует возможность поиска информации о фильмах, а также отсутствует разделение прав доступа, что может привести к нарушению защищенности данных кинотеатра.

*TMS*, полностью совместима и отвечает стандартам и требованиям *The Society of Motion Picture and Television Engineers (SMPTE), Digital Cinema Initiatives(DCI)* и *National Association of Theatre Owners (NATO)*, а также настоящим и будущим оборудованием и технологиями для цифрового кинопоказа.Данное программное средство реализует управление ключами безопасности, что позволит защитить данные кинотеатра. Однако в этом аналоге отсутствует возможность работы с кинозалами и фильмами. Другими словами, данное программное средство не покрывает необходимый персонал.

С учетом приведенных исследований предполагаемая функциональность разрабатываемого программного средства представлена в таблице 3.

Таблица 3 – Предполагаемая функциональность программного средства с учётом ролей пользователей

|  |  |  |
| --- | --- | --- |
| **Функции** | **Сотрудник** | **Администратор** |
| Работа с фильмами | Да | Да |
| Работа с кинозалами | Да | Да |
| Работа с расписанием | Да | Да |
| Управление учетными записями сотрудников | Нет | Да |

В результате анализа предметной области были выявлены проблемы кинотеатров при работе с фильмами, кинозалами и расписанием, а именно: большое количество ручной работы, большое количество затрачиваемого времени и высокий риск возникновения ошибок при работе с расписанием. Разработка программного средства позволит решить данные проблемы: автоматизировать процесс приобретения фильмов, уменьшить нагрузку на сотрудников при составлении расписания, организовать единое хранилище данных, а также снизить количество ошибок при работе с контентом кинотеатра.

# Моделирование предметной области и разработка требований к программному средству

## Анализ и формализация бизнес-процессов предметной области

**Модель «*AS-IS*».** Основным бизнес-процессом предметной области является управление деятельностью кинотеатра-мультиплекса. Оно заключается в приобретении фильмов и использовании их для показа в кинозалах. Рассмотрим следующие шаги, реализующие управление деятельностью кинотеатра:

1 Приобретение фильма. Сотрудник самостоятельно ищет информацию о добавляемом фильме.

2 Добавление фильма. Сотрудник заносит найденный фильм в список имеющихся фильмов.

3 Выбор фильма. Сотрудник выбирает фильм для показа из имеющихся фильмов.

4 Выбор кинозала. Сотрудник выбирает кинозал для показа выбранного фильма.

5 Выбор времени. Сотрудник выбирает время для показа выбранного фильма.

6 Добавление сеанса. Сотрудник добавляет сформированный сеанс в расписание.

7 Обновление расписания. Сотрудник вручную обновляет расписание кинотеатра в соответствии с изменениями.

Модель «*AS-IS*»в нотации *BPMN* представлена на рисунке 1.

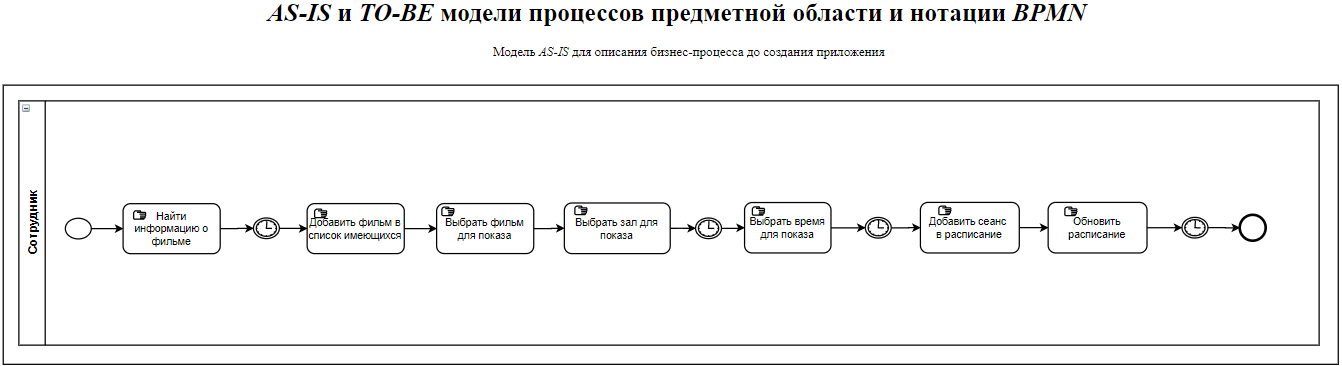


Рисунок 1 – Модель *AS-IS* целевых бизнес-процессов предметной области

В ходе обследования объекта автоматизации были выявлены следующие

слабые стороны:

– затраты времени на поиск информации о фильме;

– затраты времени на выбор зала и времени показа;

–/высокий риск возникновения ошибок при выборе времени и зала для показа (например, несоответствие разрешений фильма и зала, или показ нескольких фильмов в одном зале одновременно).

– затраты времени на обновление расписания вручную;

– незащищенность хранимой информации о фильмах и сеансах;

**Модель «*TO BE*».** Найденные в модели «*AS-IS*» недостатки необходимо устранить при создании модели «*ТО-ВЕ*», отражающей представление о новых технологиях работы организации в условиях внедренного программного средства. На основе модели «*AS-IS*» формулируются предложения по улучшению выявленных проблем. Модель «*TО-ВЕ*» в нотации *BPMN* представлена на рисунке 2.

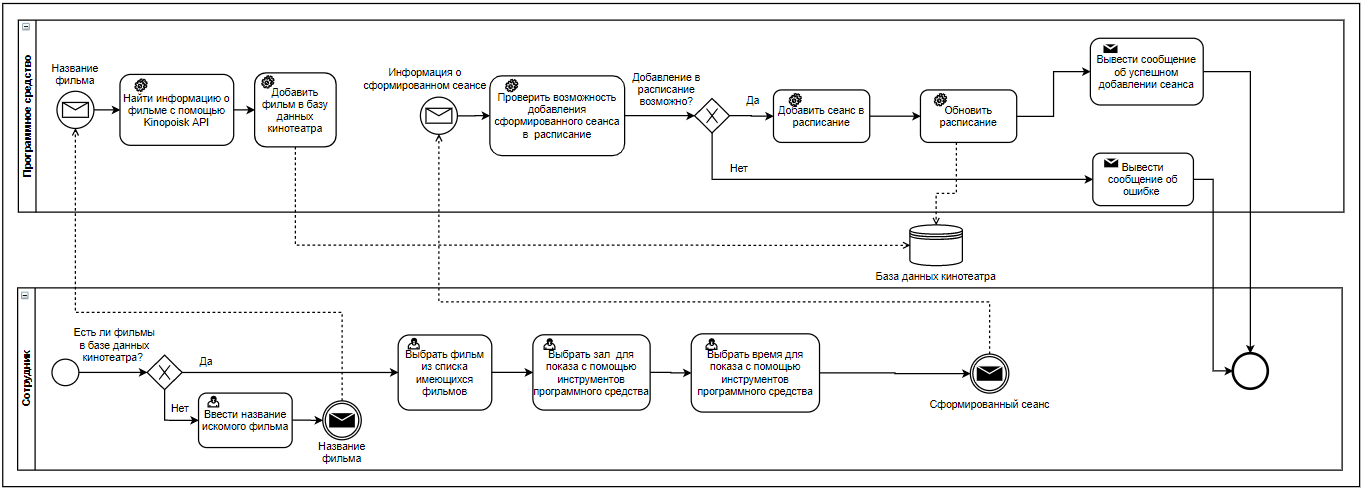


Рисунок 2 – Модель *TO-BE* целевых бизнес-процессов предметной области

После устранения выявленных проблем процесс должен выглядеть следующим образом:

1 Поиск информации о фильме. Программное средство ищет информацию о фильме по введенному названию с помощью *Kinopoisk API.*

2 Добавление фильма. Программное средство добавляет найденный фильм в базу данных кинотеатра.

3 Выбор фильма. Сотрудник выбирает фильм для показа из списка имеющихся фильмов.

4 Выбор кинозала. Сотрудник выбирает кинозал для показа выбранного фильма с помощью графических инструментов, предоставленных программным средством.

5 Выбор времени. Сотрудник выбирает время для показа выбранного фильма с помощью графических инструментов, предоставленных программным средством.

6 Проверка сформированного сеанса. Программное средство проверяет корректность выбранных зала и времени для показа выбранного фильма.

7 Добавление сеанса. Программное средство добавляет сформированный сеанс в расписание.

8 Обновление расписания. Программное средство сохраняет измененное расписание в базе данных кинотеатра.

Таким образом, предлагаемая автоматизация и оптимизация бизнес-процессов позволит снизить издержки и количество ошибок в работе кинотеатра-мультиплекса, улучшить качество, скорость и удобство управления его контентом.

## Анализ требований к разрабатываемому программному средству и разработка их спецификации

Требования к программному обеспечению рассматриваются на трех уровнях: бизнес-требования, пользовательские и функциональные требования. В каждой из перечисленных групп требований имеются свои нефункциональные требования.

Бизнес-требования – это высокоуровневая бизнес-цель организации или

заказчиков системы. Они выражают цель, ради которой разрабатывается продукт. Результатом выявления бизнес-требований является

**2.2.1** Документ о концепции и границах (*vision and scope document*) собирает бизнес-требования в единый документ, который подготавливает основу для последующей разработки продукта.

**Исходные данные.** В среднем сотрудник кинотеатра тратит 30 минут на поиск информации о фильме, которая имеет высокий риск оказаться неактуальной, и 20 минут на добавление его в расписание. На редактирование расписания уходит порядка 15 минут. При этом высока вероятность возникновения ошибок и нестыковок в расписании (например, что в одно время в одном зале должны показываться несколько фильмов одновременно). Также имеются трудности с отслеживанием состояния имеющихся фильмов, кинозалов и расписания.

**Возможности бизнеса.** Программное средство автоматизации бизнес-процессов кинотеатра позволяет автоматизировать поиск информации о фильме. При необходимости создать сеанс или редактировать расписание сотрудник может изменять его с помощью удобных графических инструментов, предоставляемых программой. Это позволит ускорить процесс изменения и избежать ошибок. Программа также предоставляет средства учета и контроля состояния фильмов, кинозалов и расписания кинотеатра, что позволяет более эффективно планировать его деятельность.

**Бизнес-цели.** Ниже описаны бизнес-цели, которые должно достичь разрабатываемое программное средство.

*ВО*-1 Уменьшить количество ошибок в работе кинотеатра на 40% в течение месяца после внедрения программного средства.

*ВО*-2 Снизить количество неэффективного рабочего времени сотрудников на 65 минут в день в течение месяца после первого выпуска программы.

*BO*-3 Снизить нагрузку на сотрудников кинотеатра на 30% в течение месяца после первого выпуска программы.

**Видение решения.** Для сотрудников кинотеатра данное программное средство – это веб-приложение или сайт, которое служит для создания оптимизированного расписания и его изменения, а также для просмотра информации о фильмах и сеансах. В отличие от имеющихся систем автоматизации бизнес-процессов кинотеатра, сотрудникам не придется самим искать информацию о фильме, что сэкономит им время.

**Бизнес-правила.** Бизнес-правила для разрабатываемого программного средства представлены в таблице 4.

Таблица 4 – Бизнес-правила

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Иден-тифи-катор** | **Определение правила** | **Тип правила** | **Статическое или динамическое** | **Источник** |
| *BR*-1 | Фильмы можно показывать только в залах с таким же разрешением | Ограничение | Статическое | Технические параметры фильма и зала |
| *BR*-2 | Нельзя добавлять фильм в расписание в дни после истечения срока проката данного фильма | Ограничение | Динамическое | Законы дистрибью-  торства |
| *BR*-3 | Фильм автоматически удаляется из базы данных кинотеатра после истечения срока его проката | Факт | Динамическое | Законы дистрибью-  торства |

Продолжение таблицы 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Иден-тифи-катор** | **Определение правила** | **Тип правила** | **Статическое или динамическое** | **Источник** |
| *BR*-4 | Продолжитель-ность сеанса должна быть больше продолжитель-ности фильма (учитываются рекламные ролики и уборка зала) | Ограничение | Статическое | Политика кинотеатра |
| *BR*-5 | Только администратор может создавать учетные записи сотрудников и добавлять их в систему | Ограничение | Статическое | Политика безопас-ности кинотеатра |

**Профили заинтересованных лиц.** Профили заинтересованных лиц представлены в таблице 4.

Таблица 4 – Профили заинтересованных лиц

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Заинтере-сованное лицо** | **Основная ценность** | **Отношение** | **Основные интересы** | **Ограни-чения** |
| Управляю-щий кино-театром | Увелечение эффектив-ности работы сотрудников;  уменьшение издержек работы кинотеатра | Поддерживают последующие выпуски в зави-симости от результатов предыдущих выпусков | Экономия  расходов  должна пре-  высить затра-  ты на разра-  ботку и ис-  пользование | Не определены |

Продолжение таблицы 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Заинтере-сованное лицо** | **Основная ценность** | **Отношение** | **Основные интересы** | **Ограни-чения** |
| Сотрудники  кинотеатра | Снижение нагрузки, экономия времени | Поддерживают, но беспокоятся об уменьшении заработной платы в связи с автоматизацией части выполняемой ими работы | Простота ис-пользования, сохранение рабочих мест | Необходи-мость обучения сотрудников работе с программ-ным средст-вом; |

**2.2.2**.Пользовательские требованияописывают задачи, которые пользователь может выполнять с помощью разрабатываемой системы.

**Диаграмма вариантов использования.** Для описания пользовательских требований к разрабатываемому программному средству была построена диаграмма вариантов использования. Она представлена ниже на рисунке 3.

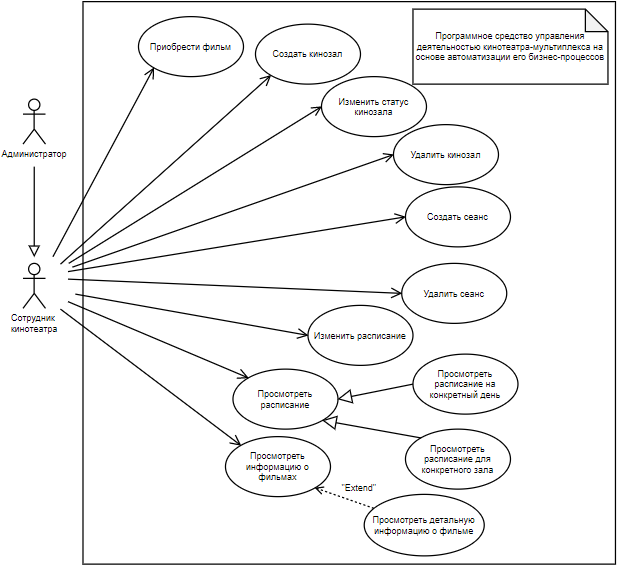


Рисунок 3 – Диаграмма вариантов использования

Данная диаграмма содержит варианты использования, которые представляют перечень действий, которые пользователь может выполнять при работе с программным средством.

**Описание действующих лиц.** Пользователями программного средства являются сотрудники кинотеатра, обладающие навыками использования компьютера и ожидающие удобные инструменты для составления и изменения расписания сеансов, а также систему для учета имеющегося контента.

**Описание сценарием вариантов использования.** Сценарии, описывающие каждый из вариантов использования, представлены ниже в таблице 5.

Таблица 5 – Описание варианта использования сценарием

|  |  |
| --- | --- |
| ***UC*-1 Приобрести фильм** | |
| Действующее лицо | Сотрудник кинотеатра |
| Краткое описание варианта использования | Данный вариант использования описывает возможность поиска информации о фильме и добавления ее в базу данных кинотеатра |
| Входные условия | Сотрудник авторизован в системе |
| Основной поток действий при исполнении варианта использования | 1.Сотрудник вводит название добавляемого фильма.  2.По введенному названию система ищет информацию по фильму с помощью *API* и выводит ее на экран.  3.Система сохраняет фильм в базе данных кинотеатра. |
| Альтернативный поток событий при исполнении варианта использования | При введении некорректного названия фильма система выводит сообщение об ошибке. |
| Выходные условия | 1.Если вариант использования выполнен успешно, то информация о фильме сохраняется в базе данных.  2 В противном случае информация о фильме не заносится в базу данных. |
| ***UC*-2 Создать кинозал** | |
| Действующее лицо | Сотрудник кинотеатра |
| Краткое описание варианта использования | Данный вариант использования описывает возможность добавления информации о кинозале в базу данных кинотеатра. |
| Входные условия | Сотрудник авторизован в системе |
| Основной поток действий при исполнении варианта использования | 1.Сотрудник вводит номер кинозала, его разрешение и количество мест в нем.  2.Система сохраняет введенную информацию в базу данных кинотеатра. |
| Альтернативный поток событий при исполнении варианта использования | 1.При введении некорректных данных система выводит сообщение об ошибке. |
| Выходные условия | 1.Если вариант использования выполнен успешно, то информация о кинозале сохраняется в базе данных. |

Продолжение таблицы 5

|  |  |
| --- | --- |
| ***UC*-2 Создать кинозал** | |
| Выходные условия | 2 В противном случае информация о кинозале не заносится в базу данных. |
| ***UC*-3 Изменить статус кинозала** | |
| Действующее лицо | Сотрудник кинотеатра |
| Краткое описание варианта использования | Данный вариант использования описывает возможность изменения статуса кинозала с «Активен» н «Неактивен» и наоборот |
| Входные условия | 1 Сотрудник авторизован в системе  2 Наличие кинозала. |
| Основной поток действий при исполнении варианта использования | 1.Сотрудник выбирает кинозал, статус которого он хочет изменить.  2 Сотрудник нажимает на картинку замка, соответствующую кинозалу.  3 Статус кинозала изменяется на противоположный, изменения сохраняются в базу данных. |
| Альтернативный поток событий при исполнении варианта использования | 1.При отсутствии кинозалов для изменения выводится соответствующее сообщение. |
| Выходные условия | 1.Если вариант использования выполнен успешно, то статус кинозала изменяется на противоположный.  2.В противном случае сеанс не создается. |
| Действующее лицо | Сотрудник кинотеатра |
| Краткое описание варианта использования | Данный вариант использования описывает возможность удаления зала из базы данных |
| Входные условия | 1 Сотрудник авторизован в системе  2 Наличие кинозала для удаления. |
| Основной поток действий при исполнении варианта использования | 1 Сотрудник выбирает кинозал, который надо удалить.  2 Сотрудник нажимает на картинку корзины, соответствующую кинозалу. |

Продолжение таблицы 5

|  |  |
| --- | --- |
| ***UC*-4 Удалить кинозал** | |
| Основной поток действий при исполнении варианта использования | 3 Кинозал удаляется из базы данных. |
| Альтернативный поток событий при исполнении варианта использования | 1.При отсутствии кинозалов для удаления выводится соответствующее сообщение. |
| Выходные условия | 1.Если вариант использования выполнен успешно, то кинозал удаляется из базы данных.  2.В противном случае кинозал не удаляется. |

|  |  |
| --- | --- |
| ***UC*-5 Создать сеанс** | |
| Действующее лицо | Сотрудник кинотеатра |
| Краткое описание варианта использования | Данный вариант использования описывает возможность создания сеанса для определенного фильма и добавление созданного сеанса в расписание |
| Входные условия | 1 Сотрудник авторизован в системе  2 Наличие фильма.  3 Наличие свободного зала. |
| Входные условия | 4 Наличие свободного времени. |
| Основной поток действий при исполнении варианта использования | 1.Сотрудник выбирает фильм для показа  2.Сотрудник выбирает зал для показа.  3 Сотрудник выбирает время показа.  4.Созданный сеанс добавляется в расписание. |
| Альтернативный поток событий при исполнении варианта использования | 1.При отсутствии фильмов, свободного кинозала или свободного времени, выводится соответствующее сообщение.  2.При выборе некорректного времени показа или кинозала выводится соответствующее сообщение. |
| Выходные условия | 1.Если вариант использования выполнен успешно, то созданный сеанс добавляется в расписание. |

Продолжение таблицы 5

|  |
| --- |
| ***UC*-5 Создать сеанс** |

|  |  |
| --- | --- |
| Выходные условия | 2.В противном случае сеанс не создается. |

|  |  |
| --- | --- |
| ***UC*-6 Удалить сеанс** | |
| Действующее лицо | Сотрудник кинотеатра |
| Краткое описание варианта использования | Данный вариант использования описывает возможность удаления сеанса из расписания |
| Входные условия | 1 Сотрудник авторизован в системе.  2 Наличие сеансов в расписании. |
| Основной поток действий при исполнении варианта использования | 1.Сотрудник выбирает сеанс для удаления.  2.Выбранный сеанс убирается из расписания. |
| Альтернативный поток событий при исполнении варианта использования | Если в расписании нет сеансов, выводится соответствующее сообщение. |
| Выходные условия | Если вариант использования выполнен успешно, то выбранный сеанс удаляется из расписания. |
| ***UC*-7 Изменить расписание** | |
| Действующее лицо | Сотрудник кинотеатра |
| Краткое описание варианта использования | Данный вариант использования описывает возможность изменения времени и/или зала для сеанса в расписании |
| Входные условия | 1 Сотрудник авторизован в системе.  2 Наличие сеансов в расписании. |
| Основной поток действий при исполнении варианта использования | 1.Сотрудник выбирает сеанс для изменения.  2 Он выбирает новое время и/или зал для сеанса.  3 Информация о сеансе в расписании обновляется. |
| Альтернативный поток событий при исполнении варианта использования | 1.Если в расписании нет сеансов, выводится соответствующее сообщение. |

Продолжение таблицы 5

|  |  |
| --- | --- |
| ***UC*-7 Изменить расписание** | |
| Альтернативный поток событий при исполнении варианта использования | 2.Если нет свободного времени или зала, выводится соответствующее сообщение и никаких изменений не происходит. |
| Выходные условия | 1.Если вариант использования выполнен успешно, то расписание обновляется.  2.В противном случае расписание не изменяется. |
| ***UC*-8 Просмотреть расписание** | |
| Действующее лицо | Сотрудник кинотеатра |
| Краткое описание варианта использования | Данный вариант использования описывает возможность просмотра расписания за все дни для всех залов |
| Входные условия | 1 Сотрудник авторизован в системе.  2 Наличие сеансов в расписании. |
| Основной поток действий при исполнении варианта использования | Выводится расписание сеансов для всех залов на все дни. |
| Альтернативный поток событий при исполнении варианта использования | Если в расписании нет сеансов, выводится соответствующее сообщение. |
| Выходные условия | Если вариант использования выполнен успешно, то расписание отображается на экране. |
| ***UC*-9 Просмотреть расписание на конкретный день** | |
| Действующее лицо | Сотрудник кинотеатра |
| Краткое описание варианта использования | Данный вариант использования описывает возможность просмотра расписания за определенный день для всех залов |
| Входные условия | 1 Сотрудник авторизован в системе.  2 Наличие сеансов в выбранный день в расписании. |
| Основной поток действий при исполнении варианта использования | 1.Сотрудник выбирает день, для которого он хочет просмотреть расписание.  2.Выводится расписание сеансов для всех залов на этот день. |

Продолжение таблицы 5

|  |  |
| --- | --- |
| ***UC*-9 Просмотреть расписание на конкретный день** | |
| Альтернативный поток событий при исполнении варианта использования | 1.Если в расписании нет сеансов, выводится соответствующее сообщение.  2.Если в расписании нет сеансов на выбранный день, выводится соответствующее сообщение. |
| Выходные условия | Если вариант использования выполнен успешно, то отображается расписание на выбранный день. |
| ***UC*-10 Просмотреть расписание для конкретного зала** | |
| Действующее лицо | Сотрудник кинотеатра |
| Краткое описание варианта использования | Данный вариант использования описывает возможность просмотра расписания за определенный день для всех залов |
| Входные условия | 1 Сотрудник авторизован в системе.  2 Наличие сеансов для выбранного зала в расписании. |
| Основной поток действий при исполнении варианта использования | 1.Сотрудник выбирает зал, для которого он хочет просмотреть расписание.  2.Выводится расписание сеансов для этого зала на все дни. |
| Альтернативный поток событий при исполнении варианта использования | 1.Если в расписании нет сеансов, выводится соответствующее сообщение.  2.Если в расписании нет сеансов для этого зала, выводится соответствующее сообщение. |
| Выходные условия | Если вариант использования выполнен успешно, то отображается расписание для выбранного зала. |
| ***UC*-11 Просмотреть информацию о фильмах** | |
| Действующее лицо | Сотрудник кинотеатра |
| Краткое описание варианта использования | Данный вариант использования описывает возможность просмотра детальной информации о фильме |

Окончание таблицы 5

|  |  |
| --- | --- |
| ***UC*-11 Просмотреть информацию о фильмах** | |
| Входные условия | 1 Сотрудник авторизован в системе.  2 Наличие фильмов для показа. |
| Основной поток действий при исполнении варианта использования | 1.Сотрудник заходит в меню работы с фильмами  2.На экране отображаются постер, название, рейтинг, разрешение и возрастное ограничение для всех имеющихся фильмов. |
| Альтернативный поток событий при исполнении варианта использования | Если фильмов для показа нет в наличии, выводится соответствующее сообщение. |
| Выходные условия | Если вариант использования выполнен успешно, то отображается список фильмов. |
| ***UC*-12 Просмотреть детальную информацию о фильме** | |
| Входные условия | 1 Сотрудник авторизован в системе.  2 Наличие фильмов для показа. |
| Основной поток действий при исполнении варианта использования | 1.Сотрудник выбирает фильм, информацию о котором он хочет просмотреть.  2.На экране отображаются название, рейтинг, разрешение, возрастное ограничение и краткое описание выбранного фильма, а также постер фильма. |
| Альтернативный поток событий при исполнении варианта использования | Если фильмов для показа нет в наличии, выводится соответствующее сообщение. |
| Выходные условия | Если вариант использования выполнен успешно, то отображается информация о выбранном фильме. |

Описание вариантов использования дает ясное представление функциональности и идентифицирует реальные потребности пользователей. Оно также служит основой для документации требований и визуализации взаимодействия пользователя с системой.

**2.2.3**.Спецификация требований к программному средству описывает все функциональные и нефункциональные требования к нему. Спецификация требований нужна для более ясного представления о проекте.

**2.2.3.1**.Общий взгляд на программное средство позволяет оценить основные характеристики программного продукта.

**Функциональное назначение.** Разрабатываемое программное средство предназначено для автоматизация ключевых бизнес-процессов, таких как приобретение фильмов, составление и изменение расписания сеансов. Его основным функциональным назначением является приобретение фильмов, составление и изменение расписания и просмотр информации о фильмах и сеансах.

**Функциональная архитектура решения.** Разрабатываемое программное средство представляет собой инструмент для составления и изменения расписания, а также для работы с фильмами и кинозалами. В системе кинотеатра-мультиплекса данное программное средство может рассматриваться как компонент более крупной системы информационной технологии кинотеатра. Оно взаимодействует с другими системами кинотеатра, такими как система продажи билетов.

**Контекстная диаграмма.** Ниже на рисунке 4 представлена контекстная диаграмма разрабатываемого программного средства.

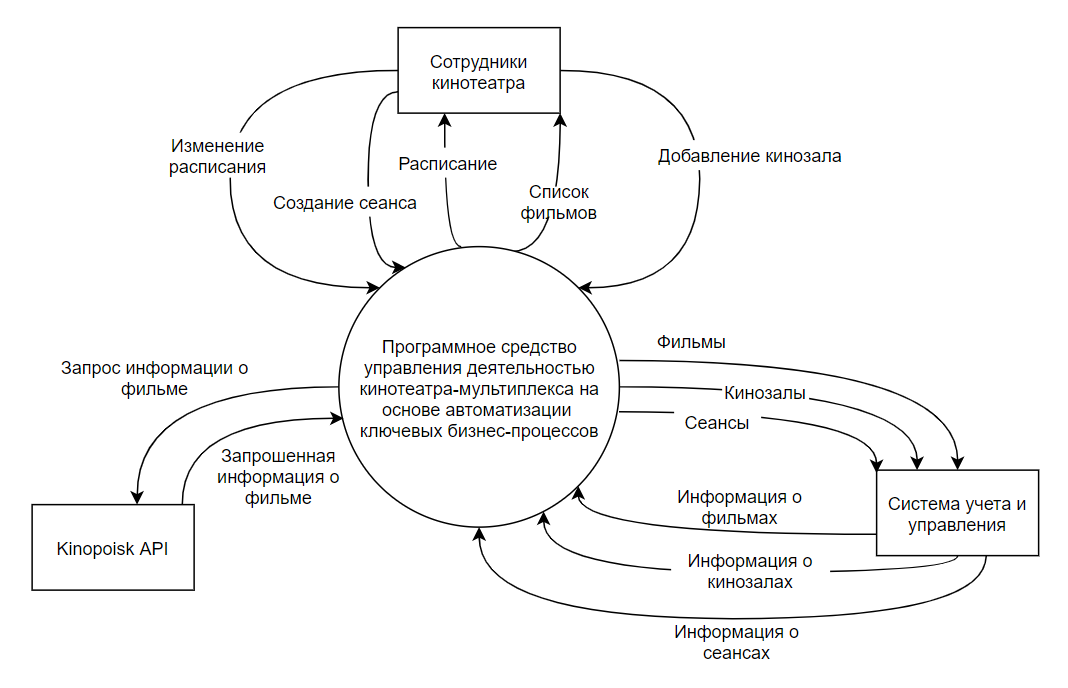


Рисунок 4 – Контекстная диаграмма программного средства

**2.2.3.2**.Функциональные требования– положение о фрагменте требуемой функциональности или поведения системы при определенных условиях. Их описание представлено ниже в таблице 6.

Таблица 6 – Функциональные требования к программному средству

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***ID*** | **Название функции** | **Роль** | **Описание** | **Функциональные требования** |
| *UС*1 | Приоб-рести фильм | Возмож-ность поиска информа-ции о фильме и добавления ее в базу данных кинотеатра | Сотрудник вводит название добавляемого фильма. По введенному названию система ищет информацию по фильму с помощью *API* и выводит ее на экран. Система сохраняет фильм в базе данных кинотеатра | *FR*1-1 Система должна иметь доступ к базе данных.  *FR*1-2 Система должна иметь интеграцию с *API*.  *FR*1-3 При успешном нахождении информации система должна вывести ее на экран  *FR*1-4 При возникновении ошибки система должна вывести соответствующее сообщение |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *UC*2 | Создать кинозал | Возмож-ность добавления информа-ции о кинозале в базу данных кинотеатра. | Сотрудник вводит номер кинозала, его разрешение и количество мест в нем. Система сохраняет введенную информацию в базу данных кинотеатра. | *FR*2-1 Система должна иметь доступ к базе данных.  *FR*2-2 При успешном добавлении информации о кинозале в базу данных система должна вывести соответствующее сообщение  *FR*2-3 При возникновении ошибки система |

Продолжение таблицы 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***ID*** | **Название функции** | **Роль** | **Описание** | **Функциональные требования** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *UC*2 |  |  |  | должна вывести соответствующее сообщение |
| *UC*3 | Изменить статус кинозала | Возмож-ность изменения статуса кинозала на «Недосту-пен» и наоборот | Сотрудник выбирает зал, статус которого он хочет изменить, и нажимает на значок замка. Статус зала меняется на противоположный | *FR3*-1 Пользователь должен видеть список доступных залов.  *FR*3-2 Система должна иметь доступ к базе данных.  *FR*3-3 При успешном выполнении операции статус зала меняется.  *FR3*-4 При возникновении ошибки система должна вывести соответствующее сообщение |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *UC*4 | Удалить кинозал | Возмож-ность удаления кинозала | Сотрудник выбирает зал, который он хочет удалить, и нажимает на значок корзины. Зал удаляется из базы данных | *FR*3-1 Пользователь должен видеть список доступных залов.  *FR*3-2 Система должна иметь доступ к базе данных.  *FR*3-3 При успешном выполнении операции зал удаляется из базы данных.  *FR*3-4 При возникновении ошибки система |

Продолжение таблицы 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***ID*** | **Название функции** | **Роль** | **Описание** | **Функциональные требования** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *UC*4 |  |  |  | должна вывести соответствующее сообщение |
| *UC*-5 | Создать сеанс | Возмож-ность создания сеанса для определен-ного фильма и добавление созданного сеанса в расписание | Сотрудник выбирает зал и время показа фильма. Созданный сеанс добавляется в расписание | *FR*5-1 Пользователь должен видеть список доступных фильмов и залов.  *FR*5-2 Система должна иметь доступ к базе данных.  *FR*-3 При успешном создании сеанса происходит его добавление в расписание |
| *UC*6 | Удалить сеанс | Возмож-ность удаления сеанса из расписания | Сотрудник выбирает сеанс для удаления.  Выбранный сеанс убирается из расписания | *FR*6-1 Система должна выводить подтверждение на удаление.  *FR*6-2 Система должна иметь доступ к базе данных.  *FR*6-3 Система должна запросить подтверждение на удаление.  *FR*6-4 При успешном удалении сеанса из расписания система должна вывести соответствующее сообщение.  *FR*6-5 При возникновении ошибки система |

Продолжение таблицы 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***ID*** | **Название функции** | **Роль** | **Описание** | **Функциональные требования** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *UC*6 |  |  |  | должна вывести соответствующее сообщение. |
| *UC*7 | Изменить распи-сание | Возмож-ность изменения времени и/или зала для сеанса в расписании | Сотрудник выбирает сеанс для изменения,  назначает ему новое время и/или зал, после чего информация о сеансе в расписании обновляется | *FR*7-1 Пользователь должен видеть список доступных залов и доступное время.  *FR*7-2 Система должна иметь доступ к базе данных.  *FR*7-3 Система должна выводить подтверждение на изменение.  *FR*7-4 При успешном изменении расписания система должна вывести на экран соответствующее сообщение.  *FR*7-5 При возникновении ошибки система должна вывести соответствующее сообщение |
| *UC*8 | Просмот-реть распи-сание | Возмож-ность просмотра расписания за все дни для всех залов | Выводится расписание сеансов для всех залов на все дни | *FR*8-1 Должна быть возможность сортировки и фильтрации сеансов по определенным параметрам  *FR*8-2 Система должна иметь |

Продолжение таблицы 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***ID*** | **Название функции** | **Роль** | **Описание** | **Функциональные требования** |
| *UC*8 |  |  |  | доступ к базе данных. |
| *UC*9 | Просмот-реть распи-сание на конкрет-ный день | Возмож-ность прос-мотра рас-писания за определен-ный день для всех залов | Выводится расписание сеансов для всех залов на выбранный сотрудником день день | *FR*9-1 Должно выводится расписание только за конкретный день  *FR*9-2 Система должна иметь доступ к базе данных.  *FR*9-3 При отсутствии расписания на выбранный день должно выводиться соответствующее сообщение |
| *UC*10 | Просмот-реть распи-сание для конкрет-ного зала | Возмож-ность прос-мотра рас-писания для определен-ного зала за все дни | Выводится расписание сеансов на все дни для выбранного сотрудником зала | *FR*10-1 Должно выводится рас-писание только для конкретного зала  *FR*10-2 Система должна иметь доступ к базе данных  *FR*10-3 При отсутствии расписания для выбранного зала должно выводиться соответствующее сообщение |
| *UC*11 | Просмот-реть информа-цию о фильмах | Возмож-ность прос-мотра детальной информа- | Выводится название, рейтинг, разрешение, возрастное ограничение и краткое | *FR*11-1 Пользователь должен видеть список |

Окончание таблицы 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***ID*** | **Название функции** | **Роль** | **Описание** | **Функциональные требования** |
|  |  | ции о фильмах | описание выбранного фильма | имеющихся фильмов. |
| *UC*11 | Просмот-реть информа-цию о фильмах | Возмож-ность прос-мотра информа-ции о фильмах | Выводится название, рейтинг, разрешение и возрастное ограничение для всех фильмов | *FR*11-1 Пользователь должен видеть список имеющихся фильмов.  *FR*11-2  Система должна иметь доступ к базе данных |
| *UC*12 | Просмот-реть деталь-ную информа-цию о фильме | Возмож-ность прос-мотра детальной информации о фильме | Выводится постер, название, рейтинг, разрешение, возрастное ограничение и краткое описание выбранного фильма | *FR*12-1 Должна выводиться информация только для выбранного фильма.  *FR*12-2 Система должна иметь доступ к базе данных. |

**2.2.3.3**.Внешние интерфейсы – это интерфейсы взаимодействия с внешними системами или операционной средой. Они определяют, как программное средство будет взаимодействовать с пользователем и другими программными средствами.

**Пользовательские интерфейсы.** Ниже представлены требования к пользовательским интерфейсам разрабатываемого программного средства.

*UI*-1 Все экраны программного средства должны иметь такие элементы, как кнопки, поля ввода и т.д., которые пользователь может использовать для взаимодействия с системой.

*UI*-2 Все экраны программного средства должны иметь оптимизированное расположение элементов интерфейса для удобства использования и минимизации действий пользователя.

*UI*-3 Нажатие на логотип программного средства на любой странице должно переносить пользователя в главное меню.

**Интерфейсы ПО.** Ниже представлены требования к интерфейсам программного обеспечения разрабатываемого программного средства.

*SI*-1 Система учета контента кинотеатра-мультиплекса

*SI*-1.1 Программное средство должно опрашивать систему учета кинотеатра для определения состояния фильмов, кинозалов и расписания.

*SI*-1.1 При истечении срока проката определенного фильма, система должна убрать данный фильм из базы данных кинотеатра.

*SI*-2 Сервис поиска информации о фильмах

*SI*-2.1 Программное средство должна сообщаться с сервисом поиска информации о фильмах через программный интерфейс, передавая в запросе на поиск информации название требуемого фильма.

**Коммуникационные интерфейсы.** Ниже представлены требования к коммуникационным интерфейсам разрабатываемого программного средства.

*CI*-1 При поиске информации о фильме система должна отправить запрос веб-сервису *Kinopoisk API*.

**Требования к юзабилити.** Ниже представлены требования к юзабилити разрабатываемого программного средства.

*USE*-1 Программное средство должно отправлять пользователю сообщения об успешно выполненных действиях или об ошибках.

*USE*-2 Программное средство должно запрашивать подтверждение у пользователя при совершении необратимых действий, таких как удаление данных.

*USE*-3 Интерфейс программного средства должен быть понятен и удобен в использовании людям с разным уровнем навыков работы с компьютером.

**Требования к производительности.** Ниже представлены требования к производительности разрабатываемого программного средства.

*PER*-1 Все веб-страницы, генерируемые системой, должны полностью загружаться не более чем за 6 секунд после запроса.

*PER*-2 Система должна выводить пользователю сообщение о подтверждении не более чем через 4 секунды после того, как пользователь отослал информацию системе.

*PER*-3 Запись информации в базу данных должна производиться не более 2 секунд.

**2.2.3.6**.Требования к данным определяют список данных, которые используются для описания элементов в разрабатываемой системе.

**Логическая модель данных.** Ниже на рисунке 5 представлена логическая модель данных разрабатываемого программного средства.

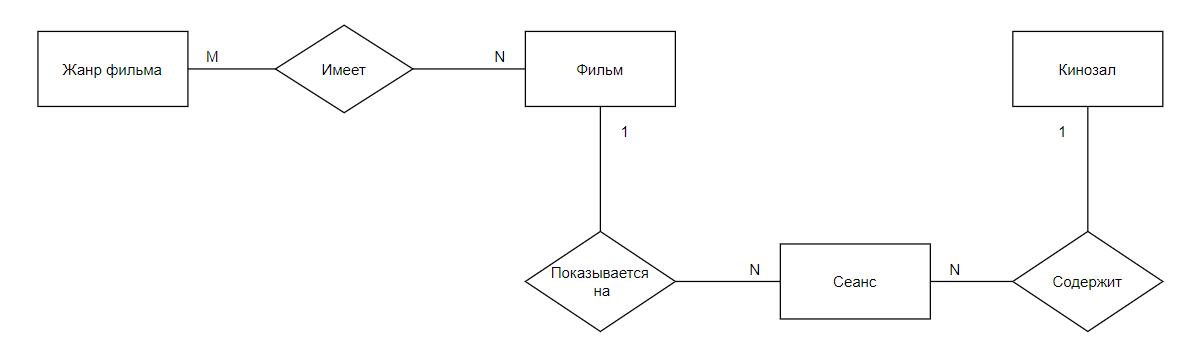


Рисунок 5 – Логическая модель данных

**Словарь данных.** Словарь данных для данного программного средства представлен в таблице 7.

Таблица 7 – Словарь данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Элемент данных** | **Описание** | **Структура или тип данных** | **Длина** | **Значения** |
| Фильм | Информация о фильме для показа | Номер фильма  Название фильма  Рейтинг  фильма  Жанр фильма  Возрастное ограничение фильма  Формат фильма  Краткое описание фильма |  |  |
| Номер фильма | Уникальный номер, идентифицирующий фильм | Тип переменной – целое число (*int*) | 7 символов |  |
| Название фильма | Полное название фильма | Тип переменной – строка (*String*) | До 100 символов |  |
| Рейтинг  фильма | Рейтинг фильма от 1 до 10 | Тип переменной – целое число (*int*) | 1-2 символа |  |
| Жанр фильма | Группа фильмов с общими тематикой и стилистикой | Тип переменной – строка (*String*) | До 100 символов |  |

Продолжение таблицы 7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Элемент данных** | **Описание** | **Структура или тип данных** | **Длина** | **Значения** |
| Возраст-ное ограниче-ние фильма | Возраст, с которого разрешается смотреть фильм | Тип переменной – целое число (*int*) | 1-2 символа | 0, 6, 12, 18 |
| Формат фильма | Техническая характеристика фильма, к примеру, 2*D*, 3*D* и т.д. | Тип переменной – строка (*String*) | 2 символа | 2*D*, 3*D,* 5*D*, 7*D* |
| Краткое описание фильма | Краткое описание сюжета фильма | Тип переменной – строка (*String*) | До 1000 символов |  |
| Кинозал | Помещение для показа фильма | Номер зала  Разрешение зала  Количество мест в зале  Статус зала |  |  |
| Номер зала | Уникальный номер, идентифицирующий зал | Тип переменной – целое число (*int*) | 7 символов |  |
| Разреше-ние зала | Определяет, фильмы какого формата в нем можно показывать | Тип переменной – строка (*String*) | 2 символа | 2*D*, 3*D,* 5*D*, 7*D* |
| Коли-чество мест в зале | Общее количество мест в зале | Тип переменной – целое число (*int*) | 4 символа |  |
| Статус зала | Статус активности зала | Тип переменной – логическая (*true/false*) |  | *true, false* |
| Сеанс | Показ фильма в определенное время в определенном кинозале | Номер сеанса  Зал для показа  Фильм для показа  Дата показа  Время начала показа |  |  |

Окончание таблицы 7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Элемент данных** | **Описание** | **Структура или тип данных** | **Длина** | **Значения** |
| Сеанс |  | Время конца показа  Статус сеанса |  |  |
| Номер сеанса | Уникальный номер, иденти-фицирующий сеанс | Тип переменной – целое число (*int*) | 7 символов |  |
| Зал для показа | Номер зала, в ко-тором будет про-водится сеанс | Тип переменной – целое число (*int*) | 7 символов |  |
| Фильм для показа | Фильм, который будет показываться | Тип переменной – целое число (*int*) | 7 символов |  |
| Дата  показа | Дата, когда проходит сеанс | Дата, дд.мм.гггг | 10 символов |  |
| Время начала показа | Время, когда начнется сеанс | Время, HH:MM, где H – часы, М - минуты | 5 символов |  |
| Время конца показа | Время, когда закончится сеанс | Время, HH:MM, где H – часы, М - минуты | 5 символов |  |
| Статус сеанса | Статус показа сеанса | Тип переменной – целое число (*int*) | 1 символ | 0, 1, 2 |

**2.2.3.7** Отчеты нужны для отображения результатов управления деятельностью кинотеатров-мультиплексов. Все отчеты в данном программном средстве будут сформированы в *pdf* формате.

**2.2.3.8**.Системные атрибуты программного обеспечения определяют системные свойства программной системы, которые не относятся непосредственно к ее функционалу.

**Надежность.**Ниже представлены требования к надежности разрабатываемого программного средства.

*ROB*-1 Если соединение между пользователем и системой разрывается во время работы с расписанием, программное средство должна позволять пользователю восстановить изменения и продолжить работу.

**Доступность.**Ниже представлены требования к доступности разрабатываемого программного средства.

*AVL*-1 Программное средство должно быть доступно 98% в течении рабочего дня сотрудников кинотеатра, за исключением времени планового обслуживания.

**Безопасность.** Ниже представлены требования к безопасности разрабатываемого программного средства.

*SEC*-1 Для работы с программным средством все пользователи обязаны авторизоваться в системе.

*SEC*-2 Только администратор имеет право добавлять сотрудников в систему и удалять из нее.

*SEC*-3 Пароли пользователей должны храниться в зашифрованном виде.

**Удобство сопровождения/эксплуатации.**Ниже представлены требования к удобству сопровождения/эксплуатации разрабатываемого программного средства.

1 Сайт должен быть сделан легким в изучении, простым в использовании, интуитивно понятным.

2 Кнопки при наведении должны подсвечиваться другим цветом.

3 Для эффективного взаимодействия между системой и пользователем должны быть окна обратной связи.

**2.2.4** Верификация программного средства – это процесс подтверждения того, что разработанное программное средство соответствует заявленным требованиям и спецификациям заказчика. Для разрабатываемого программного средства процесс верификации может включать следующие виды тестирования:

1 Тестирование каждой функции программного средства для убеждения в том, что они работают согласно спецификации. Это включает тестирование интерфейсов, работы с данными и т.д.

2 Тестирование системы на защищенность данных, аутентификация пользователей и т.д.

3 Проверка работы системы на различных платформах.

4 Оценка удобства использования, проверка соответствия дизайна, проверка на соответствие ожиданиям пользователей.

5 Проверка корректности хранения и обработки данных, а также выполнения запросов к базам данных.

## Образ предлагаемого решения

В ходе проведенного обследования были выявлены следующие бизнес-

проблемы:

– большие затраты времени на поиск необходимой информации о фильмах;

– возможность добавления неактуальной информации о фильмах;

– большие затраты времени при добавлении фильма в расписание;

– возможные ошибки и нестыковки при составлении и изменении расписания;

– сложность контроля и учета имеющихся фильмов и расписания;

– ненадежность хранения данных о контенте кинотеатра.

Указанные бизнес-проблемы предлагается решить посредством создания программного средства, позволяющего:

– автоматизировать поиск актуальной информации о фильме;

– использовать удобные графические инструменты для выбора места и времени показа фильма;

– использовать графические инструменты для визуализации расписания кинотеатра и удобного внесения в него изменений;

– использовать инструменты для визуализации и контроля фильмов;

– хранить информацию о контенте кинотеатра в базе данных;

– обеспечить защиту от несанкционированного доступа к данным кинотеатра.

На рисунке 3 представлена функциональная архитектура предлагаемого решения.

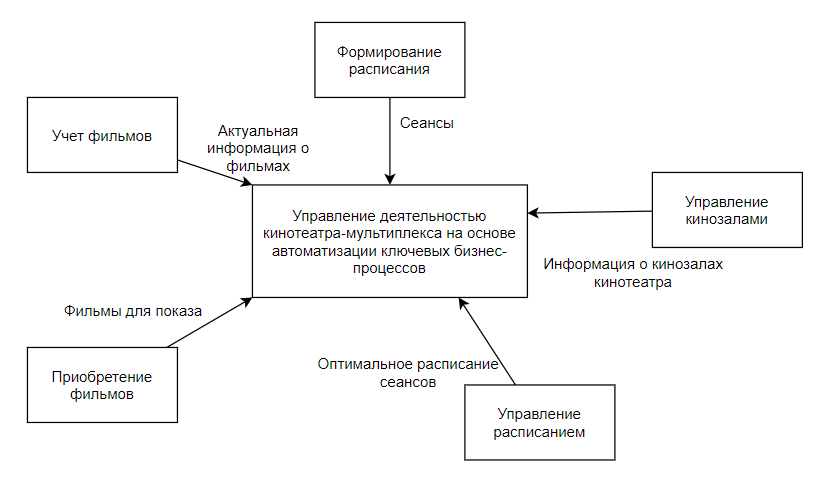


Рисунок 5 – Функциональная архитектура решения

Назначение программного средства:

– создание механизма автоматизированного поиска о фильмах;

– облегчение работы сотрудников при составлении и изменении расписания;

– облегчение контроля и учета состояния контента кинотеатра.

Цели создания программного средства:

– повышение эффективности и оптимизация работы кинотеатра:

– снижение издержек кинотеатра;

– снижение количества ручного труда сотрудников кинотеатра.

Необходимо предусмотреть интеграцию с системами поиска информации о фильмах, таких как *Kinopoisk API*. Должно быть обеспечено разграничение доступа и отсутствие возможности несанкционированного доступа в систему. Все пользователи программного средства должны иметь свои учетные записи и роли.

# ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

## Архитектурные решения

Архитектура программного обеспечения– это схема высокого уровня, определяющая структуру, дизайн и поведение программной системы. Она включает в себя организацию компонентов, их взаимодействие и ограничения системы.

Для описания архитектурных решений программного средства была использована нотация *С4 model*. Ее суть заключается в том, что архитектура ПО представляется в виде четырех уровней декомпозиции, где каждый последующий уровень более детально раскрывает предыдущий. Первым (самым верхним) уровнем является контекстный. Он представляет собой отправную точку, которая показывает, как проектируемая система вписывается в окружающий мир. Контекстный уровень представления архитектуры ПО представлен ниже на рисунке 6.

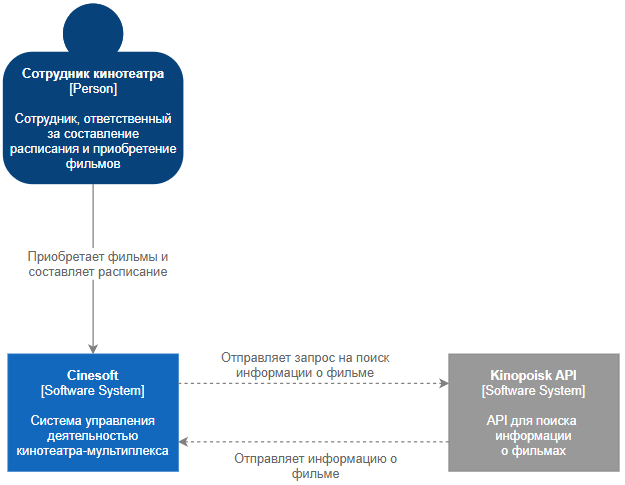


Рисунок 6 – Контекстный уровень представления архитектуры ПО

Следующим уровнем представления архитектуры ПО является контейнерный. В нем составные части архитектуры, определенные на контекстном уровне, декомпозируются для предоставления информации о технических блоках высокого уровня (здесь представляются части архитектуры, внутри которых будет исполняться программный код ПО). Контейнерный уровень представления архитектуры ПО представлен ниже на рисунке 7.

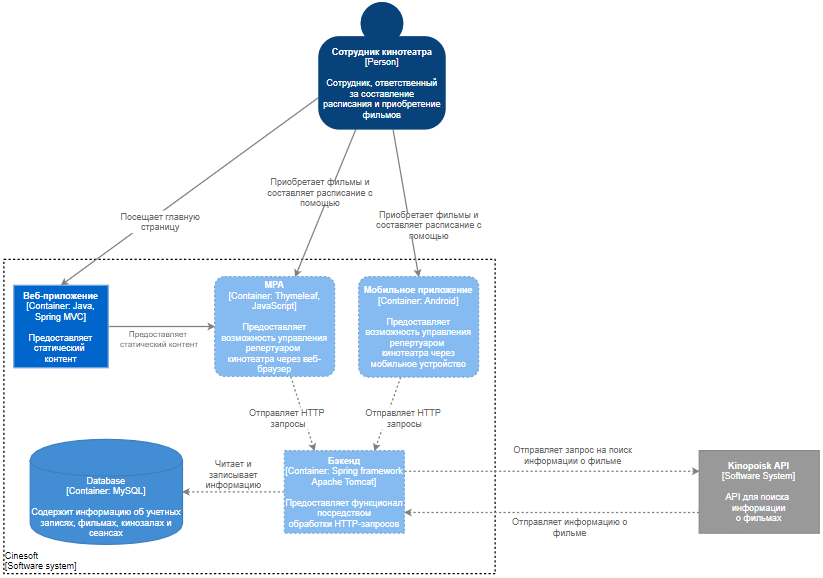


Рисунок 7 – Контейнерный уровень представления архитектуры ПО

Следующим уровнем представления архитектуры ПО является компонентный. В нем представляются внутренние блоки контейнеров, определенных на контейнерном уровне. Компонентный уровень представления архитектуры ПО представлен ниже на рисунке 8.

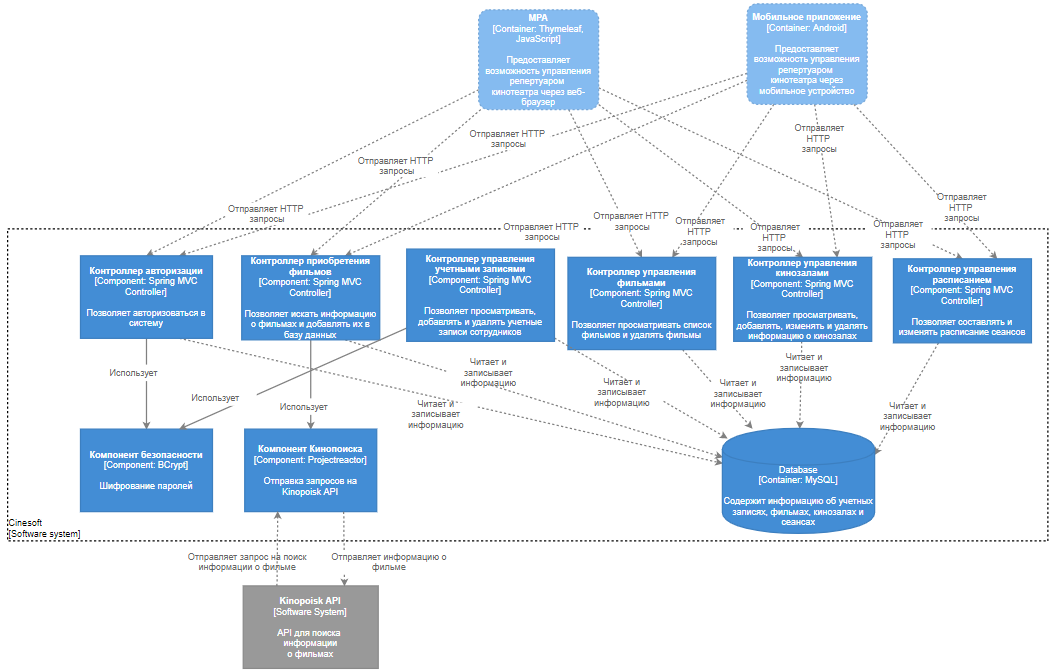


Рисунок 8 – Компонентный уровень представления архитектуры ПО

Самым нижним уровнем представления архитектуры ПО является кодовый. В нем представляется внутренняя организация компонентов, определенных на компонентном уровне. Кодовый уровень представления архитектуры ПО представлен ниже на рисунке 9.

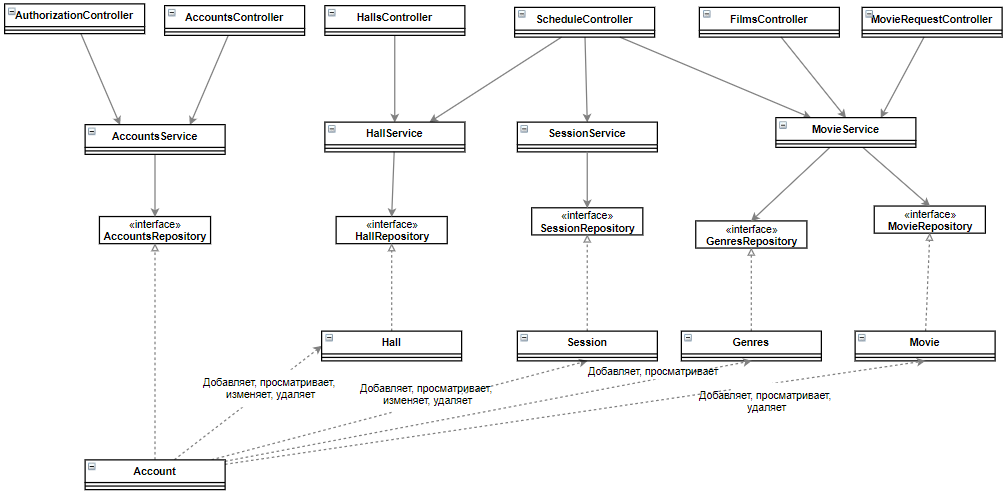


Рисунок 9 – Кодовый уровень представления архитектуры ПО

Программное средство было спроектировано в соответствии с описанной архитектурой.

## Обоснование выбора компонентов и технологий для реализации серверной и клиентской частей программного средства

Выбор компонентов и технологий для реализации программного средства является одним из важных этапов в разработке, т.к. позволяет сократить время и сроки разработки. Необходимо учитывать особенности функциональности программного средства, а также инфраструктуры, в которой разработанное программное средство будет осуществлять работу. Для выбора языка программирования был проведен сравнительный анализ языков *Java* и *Python*. Результат их сравнения представлен ниже в таблице 8.

Таблица 8 – Сравнительный анализ языков программирования

|  |  |  |
| --- | --- | --- |
| **Характеристика** | **Язык программирования** | |
| *Java* | *Python* |
| Простота изучения | Прост в изучении | Прост в изучении |
| Читабельность кода | Более детальный и понятный | Более краткий и лаконичный |
| Типизация | Строгая | Динамическая |

Окончание таблицы 8

|  |  |  |
| --- | --- | --- |
| Производительность | Высокая | Средняя |
| Особенности синтаксиса | Использование фигурных скобок и точек с запятой | Использование табуляции |
| Масштабируемость | Очень высокая | Высокая |
| Экосистема | Больше миллиона различных библиотек и фреймворков | Больше тысячи различных библиотек и фреймворков |
| Применение | Разработка мобильных, веб, энтерпрайз и настольных приложений | Веб-разработка, нейронные сети, *DevOps* инженерия |

На основе результатов сравнительного анализа, представленных в таблице 8, для разработки был выбран язык программирования *Java.*

В качестве среды разработки была выбрана среда *IntelliJ IDEA*. Выбор обусловлен возможностью среды работать с фреймворками, ее удобство и надежность. *IntelliJ IDEA* обладает следующими преимуществами:

1..*IntelliJ IDEA* предоставляет интеллектуальную систему анализа кода, которая помогает выявлять ошибки, предостерегать от потенциальных проблем и улучшать качество кода. Это особенно важно для проекта, который требует надежности.

2..*IntelliJ IDEA* предлагает такие инструменты как редактор кода, отладчик, систему контроля версий, инструменты для работы с базами данных и многие другие возможности.

3..*IntelliJ IDEA* поддерживает *Windows*, *macOS* и *Linux*, что обеспечивает универсальность и возможность работы на различных операционных системах.

Для выбора фреймворка в качестве технологии реализации разрабатываемой системы был проведен сравнительный анализ фреймворков *Spring Framework* и *Apache Struts*. Результат их сравнения представлен ниже в таблице 9.

Таблица 9 – Сравнительный анализ фреймворков

|  |  |  |
| --- | --- | --- |
| **Характеристика** | **Язык программирования** | |
| *Spring* | *Apache Struts* |
| Гибкость | Очень высокая | Ниже, чем у *Spring* |
| Интеграция с базой данных | Есть | Нету |
| Архитектура | Многослойная | *MVC* |

Окончание таблицы 9

|  |  |  |
| --- | --- | --- |
| **Характеристика** | **Язык программирования** | |
| *Spring* | *Apache Struts* |
| Вес | Легковесный из-за низкой связности модулей | Тяжеловесный из-за высокой связности модулей |
| Популярность | Один из самых популярных фреймворков *Java* | Менее популярен, чем *Spring* |

На основе результатов сравнительного анализа, представленных в таблице 9, для разработки был выбран фреймворк *Spring.*

В качестве системы управления базами данных была выбрана *MySQL*. Выбор обусловлен ее производительностью, безопасностью и удобством использования. *MySQL* обладает следующими преимуществами:

1..*MySQL* позволяет установить высокий уровень безопасности благодаря системе доступа и управления учетными записями.

2..*MySQL* обеспечивает хорошую производительность, особенно при правильной настройке и оптимизации. Это важно для обеспечения быстрого доступа к данным и отклика приложения, управляющего кинотеатром.

3..*MySQL* поддерживает множество языков программирования, включая *Java*, что облегчит интеграцию с приложением, разработанным с использованием *IntelliJ IDEA*.

4..*MySQL* предоставляет множество инструментов для администрирования и управления базой данных, что упрощает ее обслуживание.

При использовании выбранных компонентов и технологий будет создано программное средство, которое позволит автоматизировать бизнес-процессы кинотеатра-мультиплекса.

## Проектирование и разработка пользовательского интерфейса

**Система дизайна пользовательского интерфейса программного средства.** В процессе разработки программного средства особое внимание уделено системе дизайна для обеспечения единого стиля, функциональности и привлекательности элементов интерфейса. Разработанная система дизайна, на которой представлены основные элементы пользовательского интерфейса, изображена на рисунке 10.

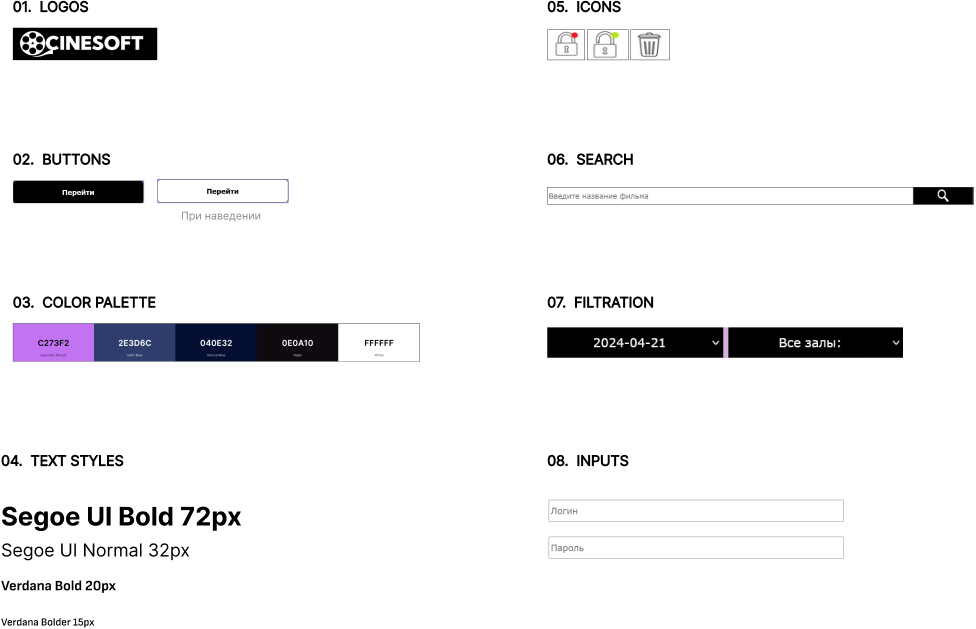


Рисунок 10 – Система дизайна пользовательского интерфейса программного средства

В основе системы дизайна лежит логотип, который стал ключевым элементом в создании узнаваемого облика программного продукта. Его стиль и форма служат основой для остальных элементов интерфейса. В качестве цвета *White* и *Black* подчеркивает минимализм и простоту интерфейса.

Кнопки и иконки управления разработаны с учетом минимализма и удобства использования. Их дизайн обеспечивает интуитивно понятное взаимодействие с пользователем.

Один из ключевых аспектов – стиль текста. Основным шрифтом для пользовательского интерфейса является шрифт *Segoe UI*, который обеспечивает хорошую читаемость.

Таким образом, система дизайна пользовательского интерфейса программного средства разработана с учетом удобства, функциональности и эстетики, что создает основу для приятного и эффективного использования программного средства.

**План расположения элементов на странице программного средства.** При подготовке к разработке пользовательского интерфейса программного средства важным этапом является создание эффективной схемы расположения элементов на странице.

При создании схемы расположения элементов особое внимание уделено обеспечению интуитивной навигации и легкого доступа к основным функциям приложения.

Схема расположения элементов главного окна приложения представлена на рисунке 10.

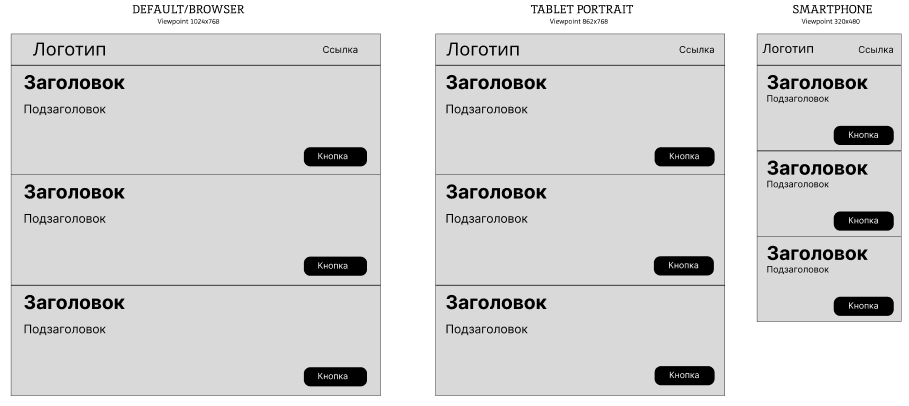


Рисунок 11 – Схема расположения элементов главной страницы

Схема расположения элементов окна приобретения фильмов представлена на рисунке 12.

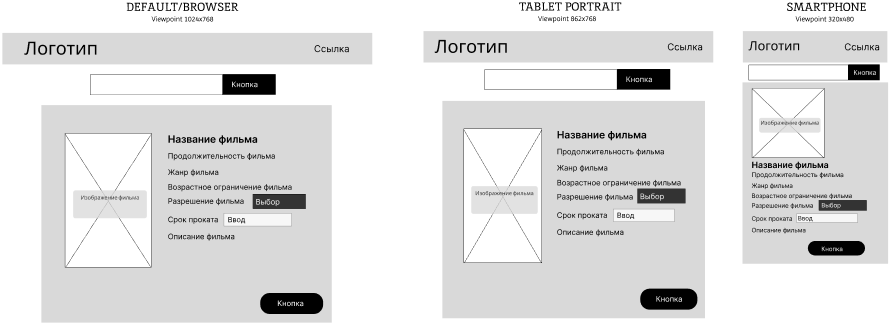


Рисунок 12 – Схема расположения элементов окна приобретения фильмов

Схема расположения элементов главного окна добавления сеансов представлена на рисунке 13.

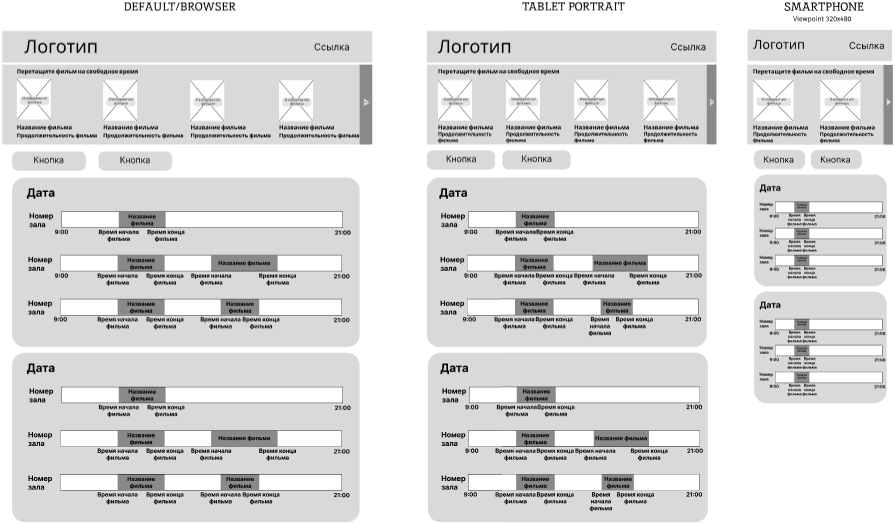


Рисунок 13 – Схема расположения элементов окна добавления сеансов

Схема расположения элементов окна работы с расписанием представлена на рисунке 14.

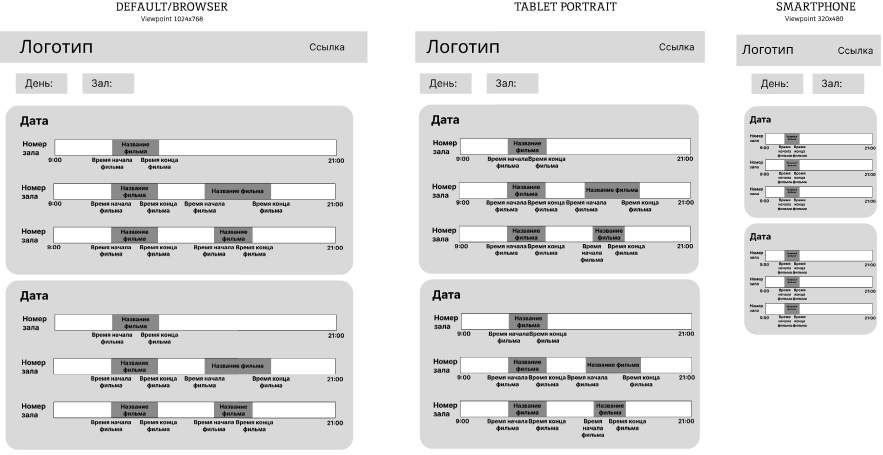


Рисунок 14 – Схема расположения элементов окна работы с расписанием

Схема расположения элементов окна работы с фильмами представлена на рисунке 15.

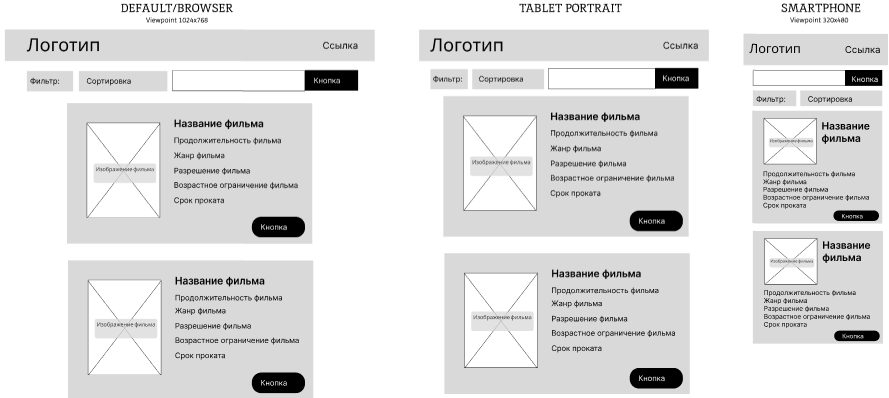


Рисунок 15 – Схема расположения элементов окна работы с фильмами

Дизайн окон приложения был спроектирован в соответствии с построенными схемами.

**Логика действий пользователя в программном средстве.** Для визуального представления логики действий пользователя программного средства построена диаграмма *User-flow*. Данная диаграмма представляет основные шаги, которые выполняют пользовали при использовании программного средства.

На рисунке 16 отображена *User-flow* диаграмма логики действий администратора.

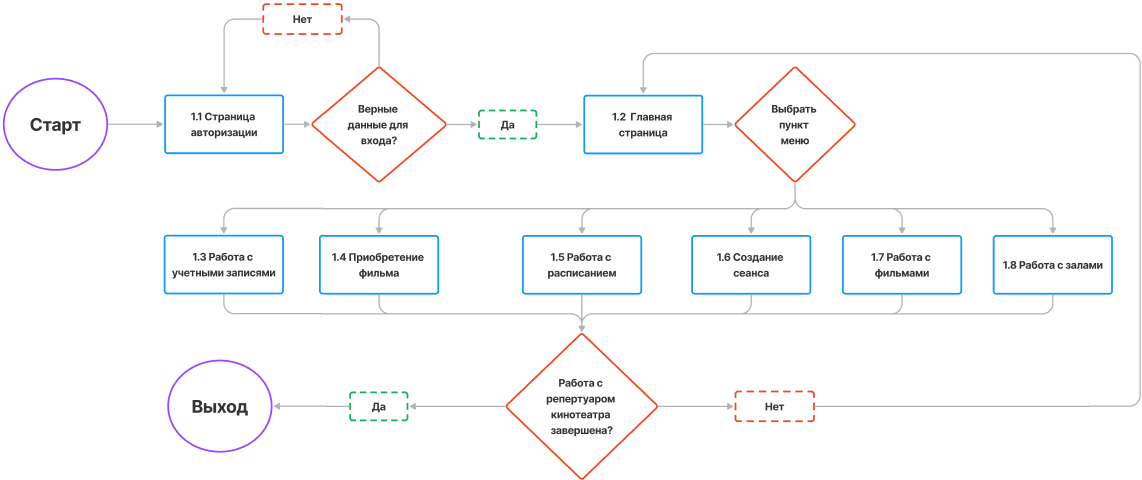


Рисунок 16 – *User-flow* диаграмма логики действий администратора

После входа в аккаунт администратор может перейти на страницы «Работа с учетными записями», «Приобретение фильма», «Работа с расписанием», «Создание сеанса», «Работа с фильмами», «Работа с залами», на которых может использовать функционал программного средства.

На рисунке 17 отображена *User-flow* диаграмма логики действий сотрудника.

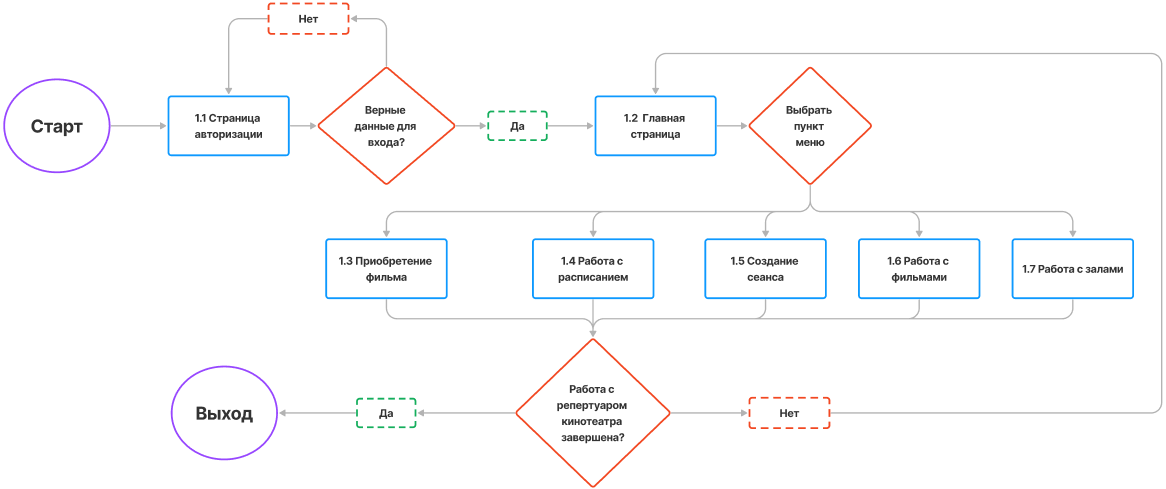


Рисунок 17 – *User-flow* диаграмма логики действий сотрудника

После входа в аккаунт сотрудник может перейти на страницы «Приобретение фильма», «Работа с расписанием», «Создание сеанса», «Работа с фильмами», «Работа с залами», на которых может использовать функционал программного средства.

**Детальное представление экранов пользовательского интерфейса.** На рисунке 18 изображено детальное представление страницы авторизации пользователя. Для авторизации пользователь должен вести логин и пароль.

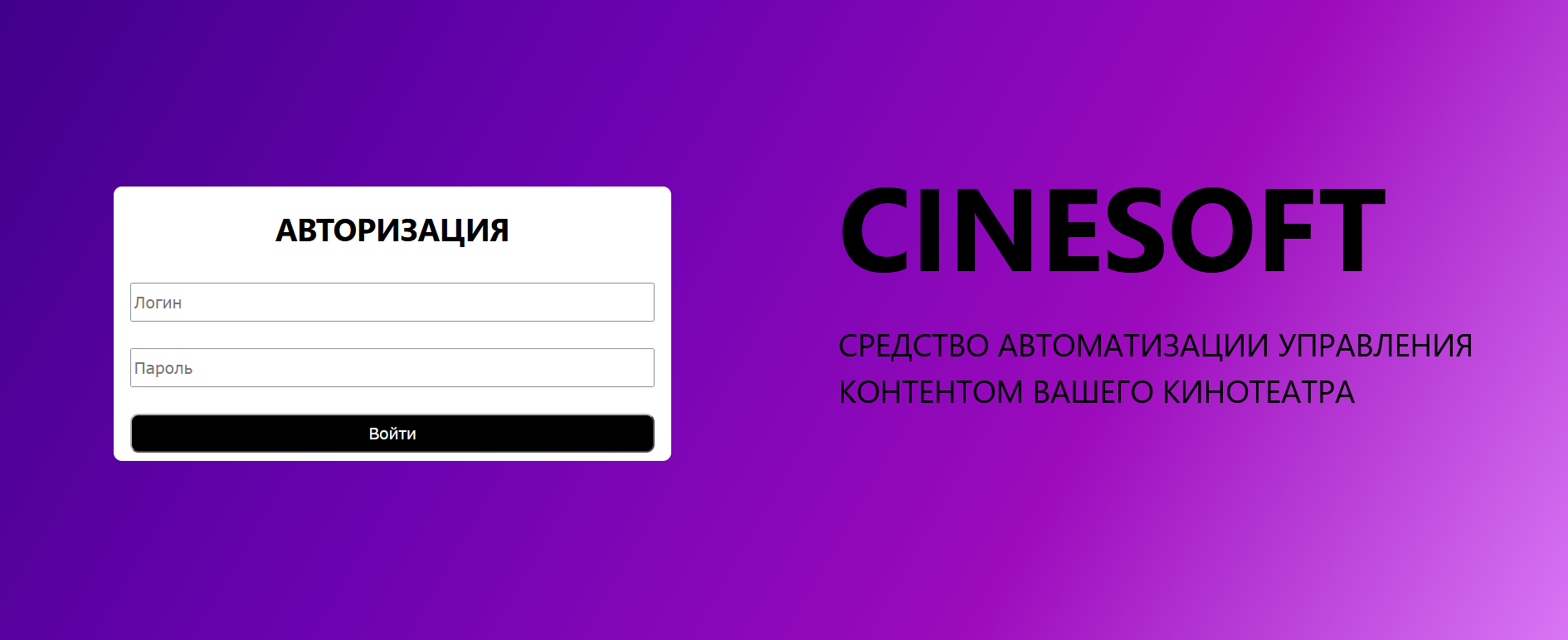


Рисунок 17 – Детальное представление экрана авторизации

Если пользователь вводит некорректные данные, на странице отображается сообщение об ошибке. Пример данного сообщения представлен на рисунке 18.

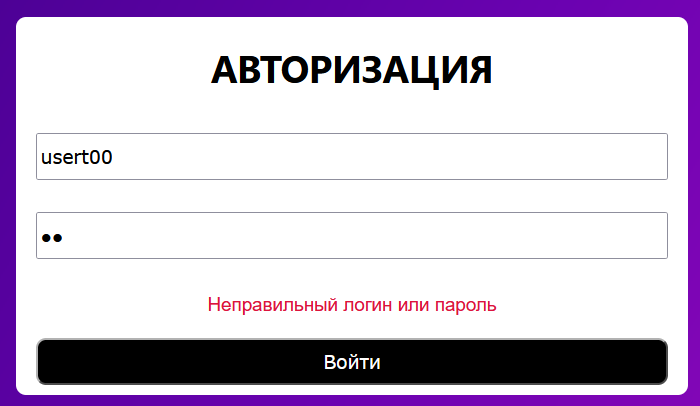


Рисунок 18 – Сообщение об ошибке

После успешной авторизации пользователь попадает в главное меню. На рисунке 19 изображено детальное главного меню программного средства.

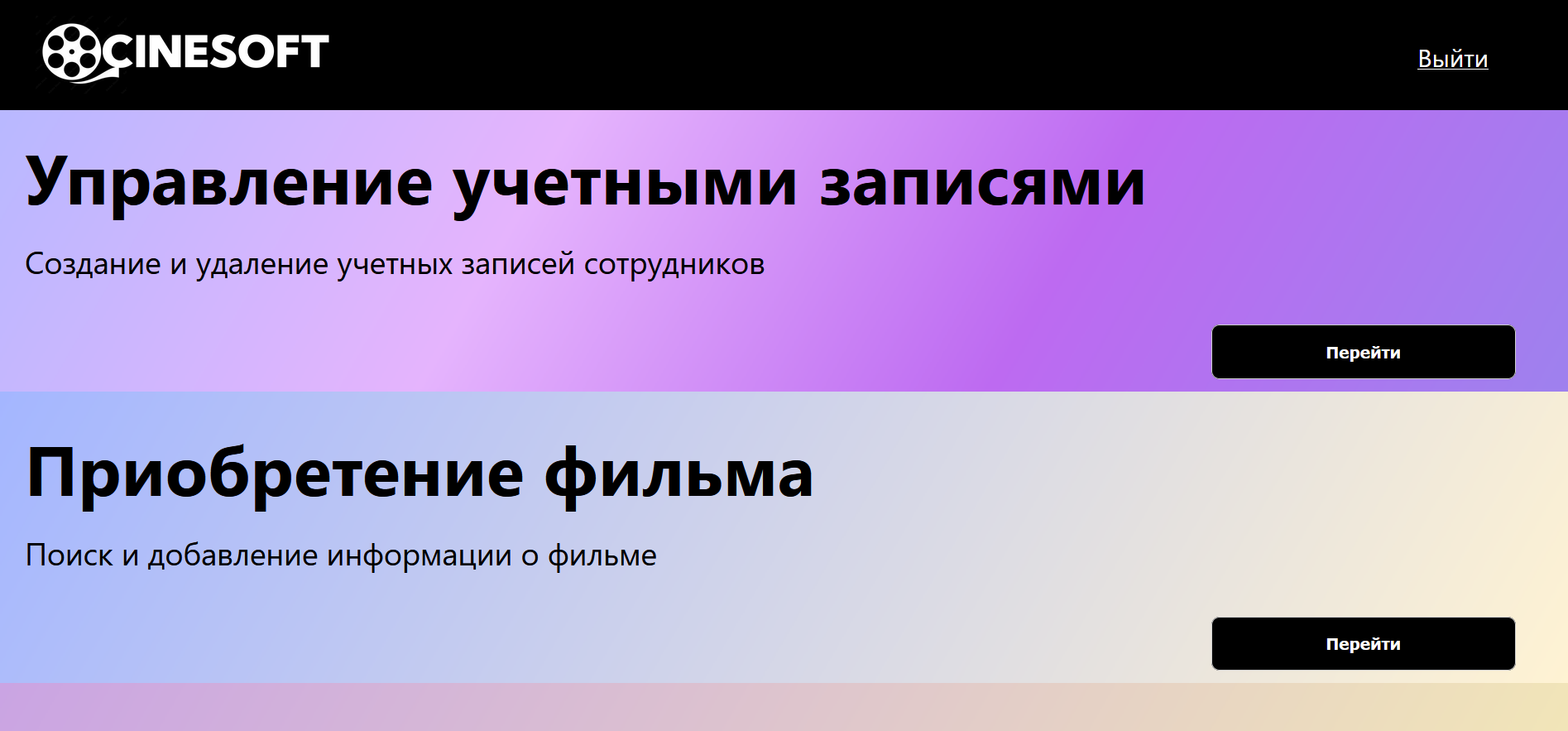


Рисунок 19 – Детальное представление главного меню

Ниже на рисунке 20 представлено детальное представление меню работы с учетными записями.

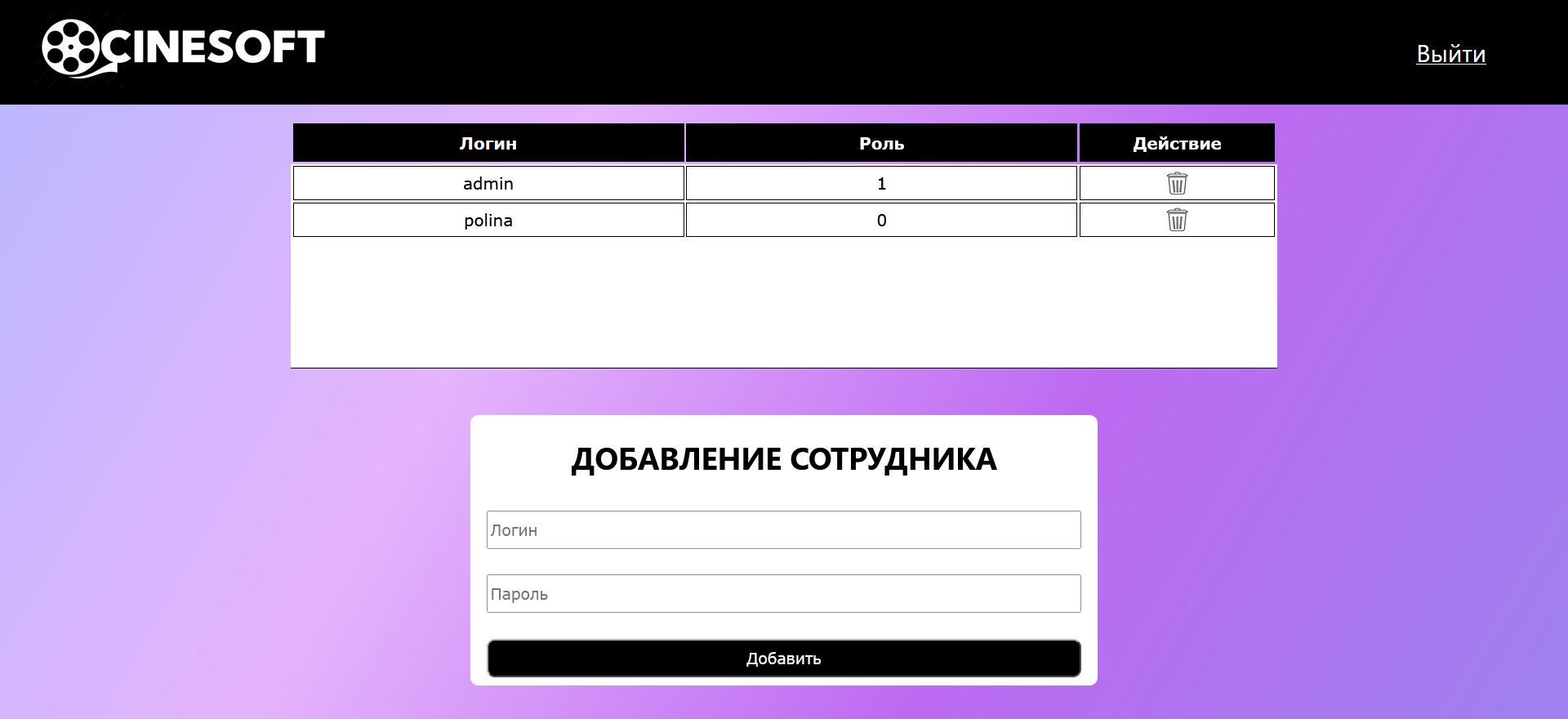


Рисунок 20 – Детальное представление меню работы с учетными записями

При успешном добавлении или удалении сотрудника выводится соответствующее сообщение. Его пример представлен на рисунке 21.

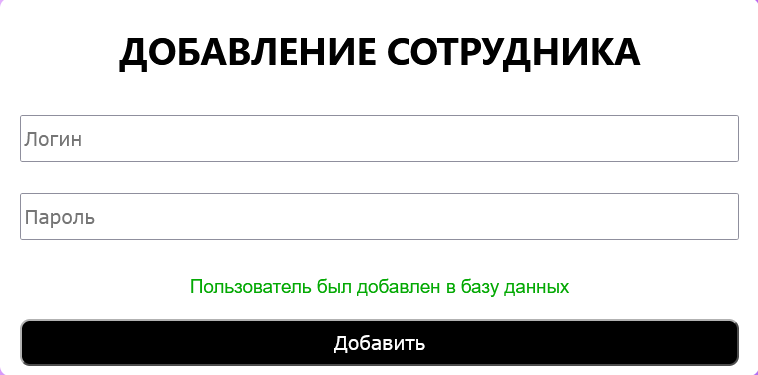


Рисунок 21 – Сообщение об успешном добавлении

Ниже на рисунке 22 представлено детальное представление меню приобретения фильма.

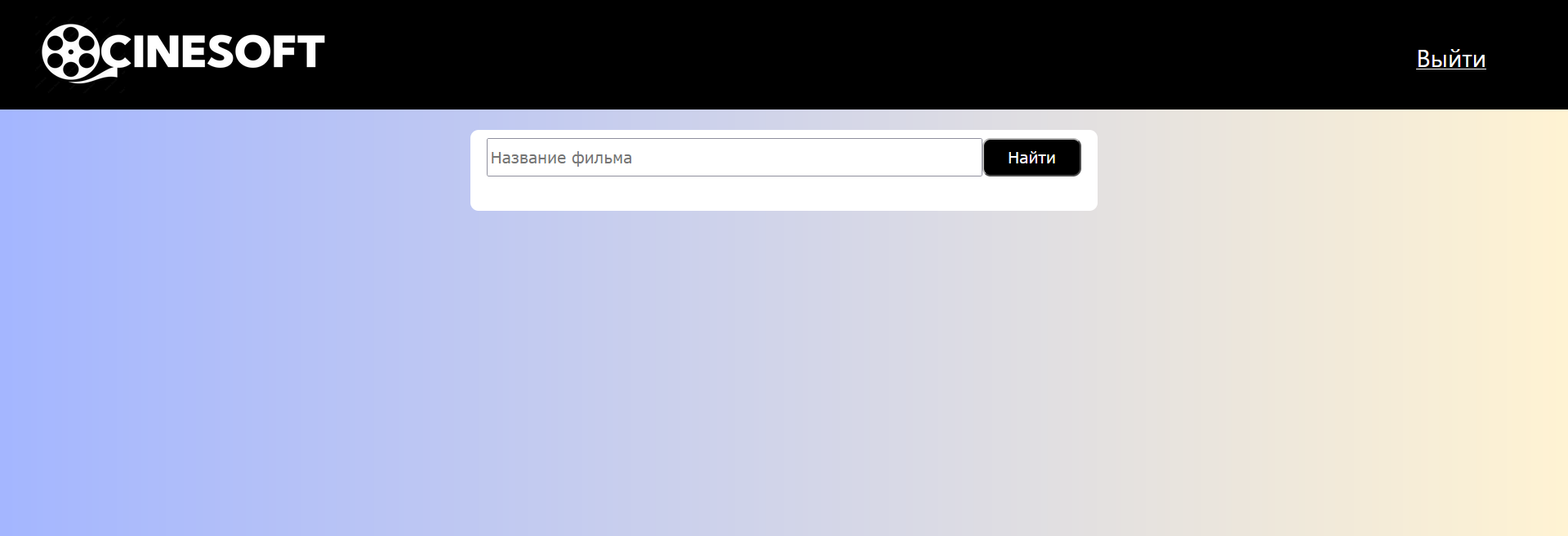


Рисунок 22 – Детальное представление меню приобретения фильма

При успешном нахождении информации о фильме она выводится на экран. Пример найденной информации представлен на рисунке 23.

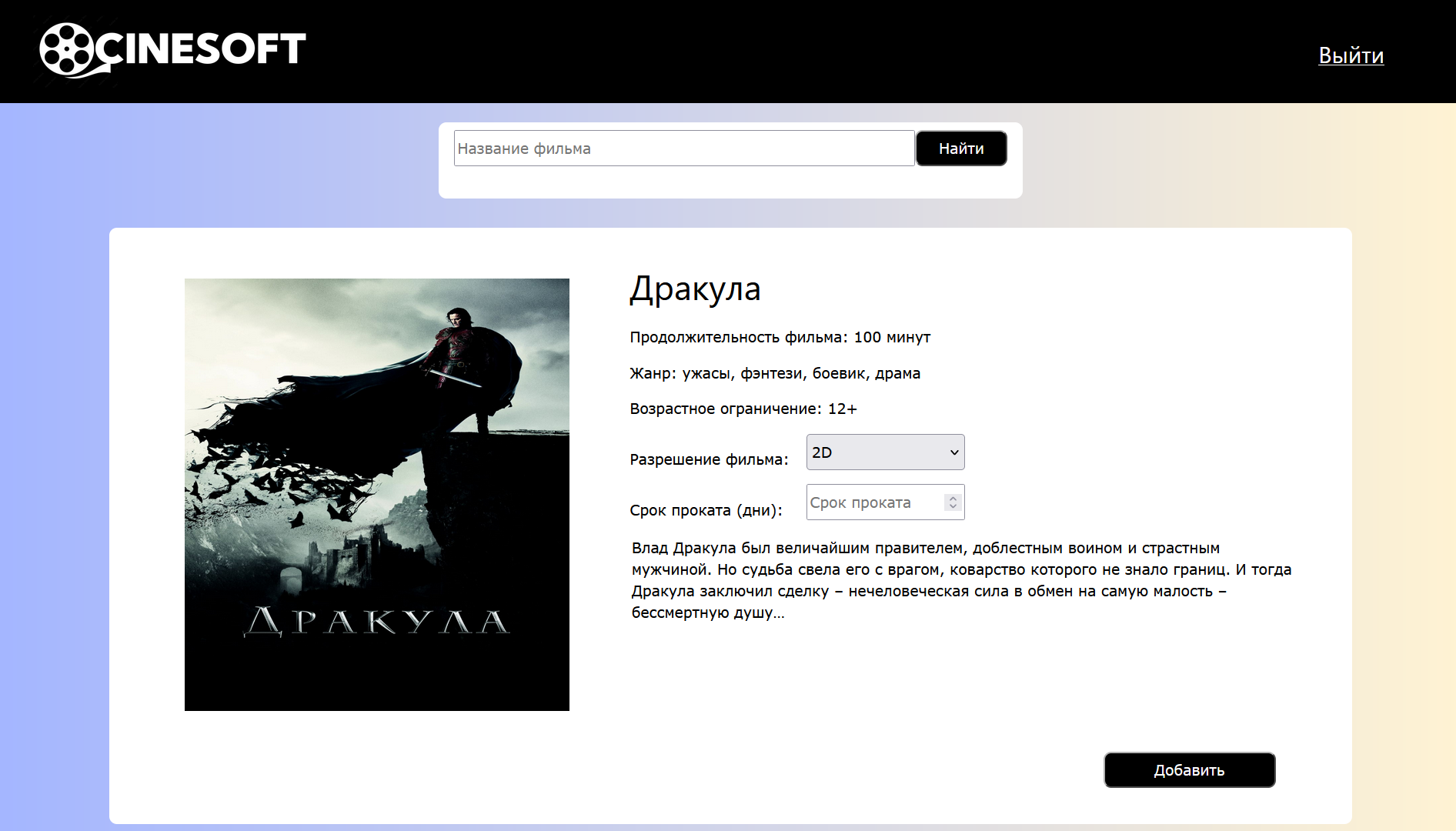


Рисунок 23 – Отображение найденной информации

При ошибке поиска информации выводится соответствующее сообщение. Его пример представлен на рисунке 24.

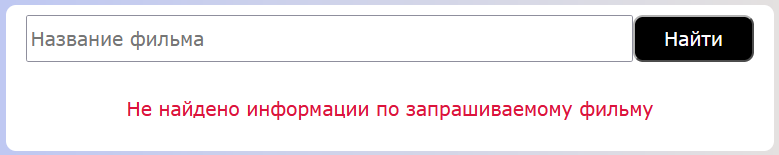


Рисунок 24 – Сообщение об ошибке

При успешном добавлении фильма в базу данных выводится соответствующее сообщение. Оно представлено на рисунке 25.

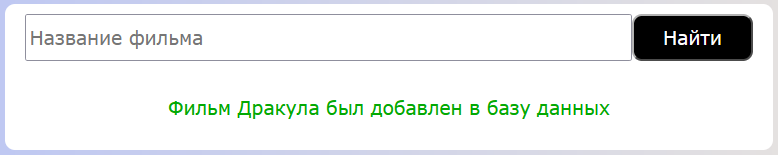


Рисунок 25 – Сообщение об успешном добавлении

Ниже на рисунке 26 представлено детальное представление меню работы с фильмами.

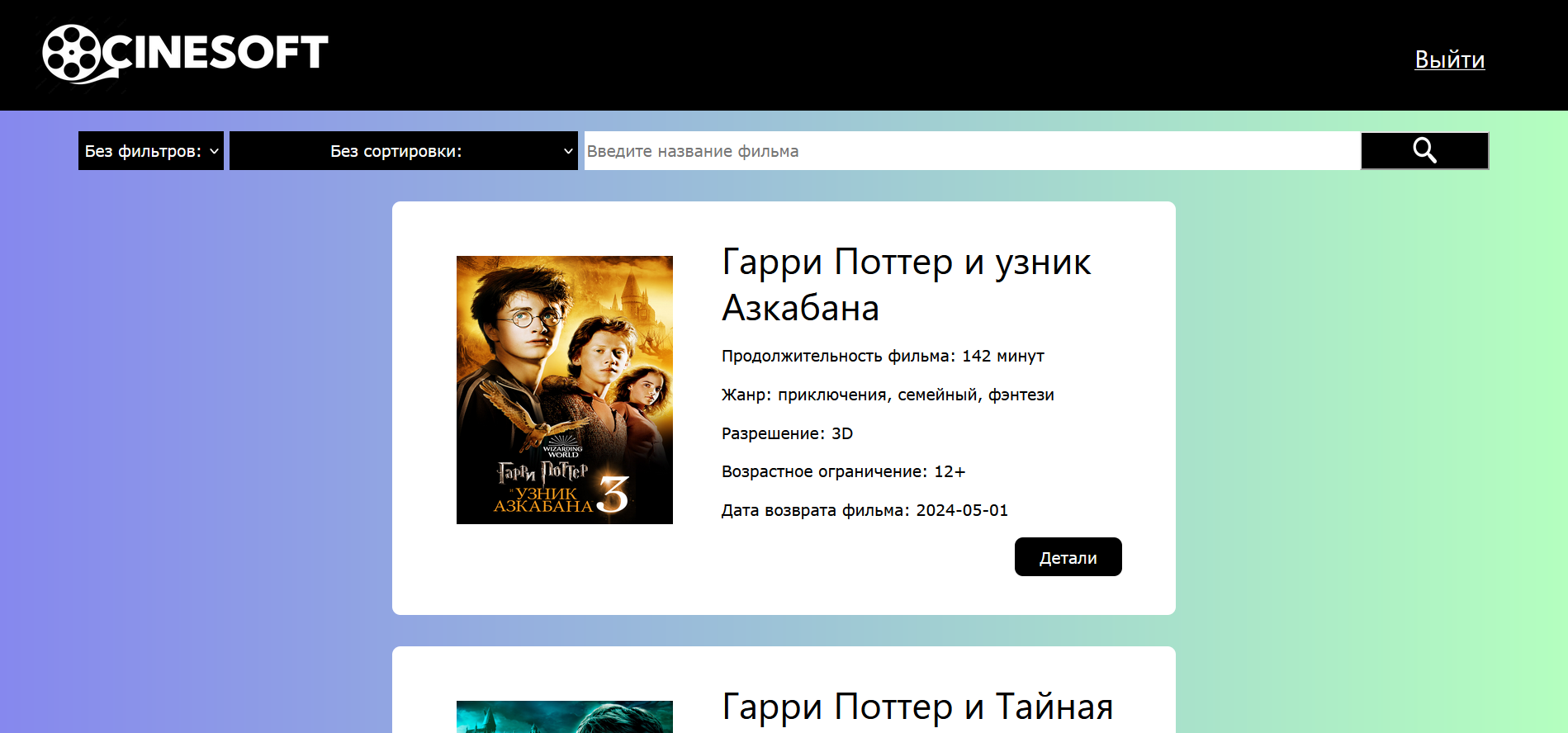


Рисунок 26 – Детальное представление меню работы с фильмами

При введении в поисковую строку названия фильма или его части происходит поиск и вывод на экран этого фильма. Также можно одновременно проводить сортировку и фильтрацию фильмов по определенным признакам. Пример поиска, сортировки и фильтрации представлен на рисунке 27.

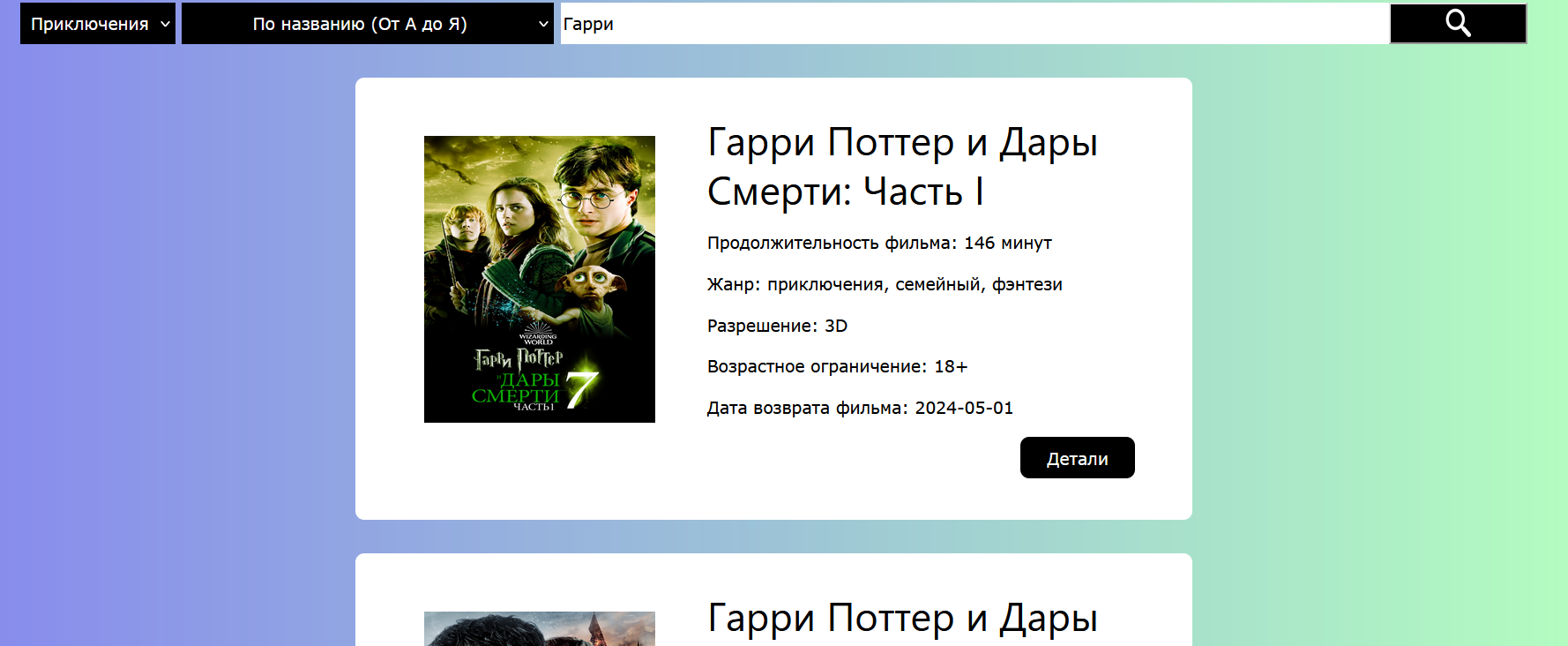


Рисунок 27 – Поиск, сортировка и фильтрация фильмов

При нажатии кнопки «Детали» происходит переход на страницу отображения детальной информации о соответствующем фильме. На рисунке 28 представлено детальное представление меню информации о фильме.

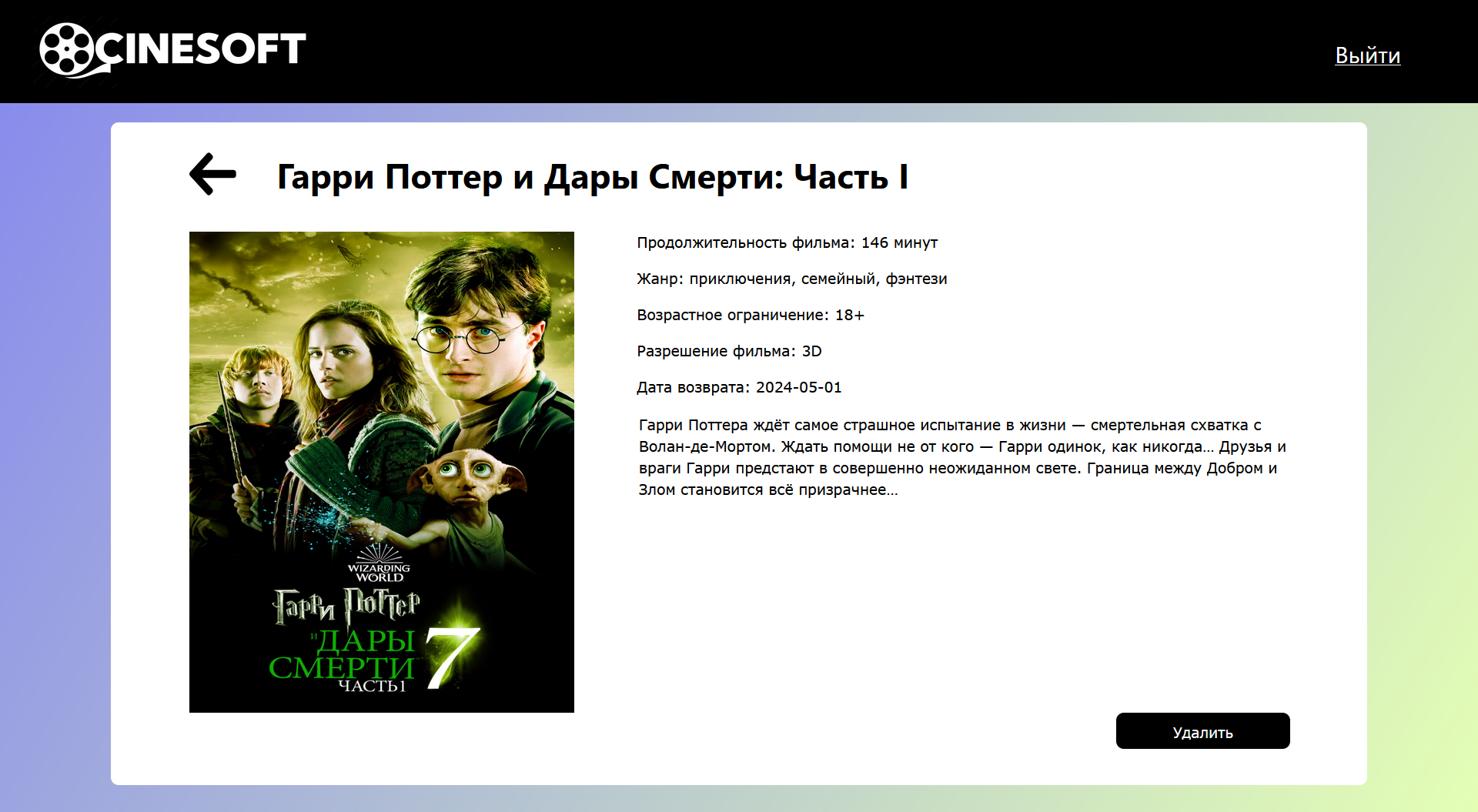


Рисунок 28 – Детальное представление меню информации о фильме

При нажатии на кнопку «Удалить» выводится сообщение о подтверждении удаления. Оно представлено на рисунке 29.

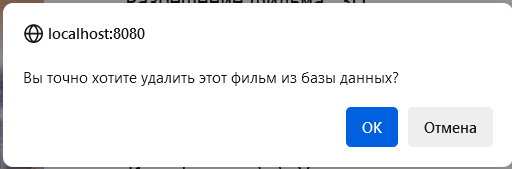


Рисунок 29 – Сообщение о подтверждении удаления фильма

При подтверждении удаления фильма происходит его удаление, и пользователь переходит на страницу работы с фильмами. Там ему выводится сообщение об успешном удалении фильма. Оно представлено на рисунке 30.

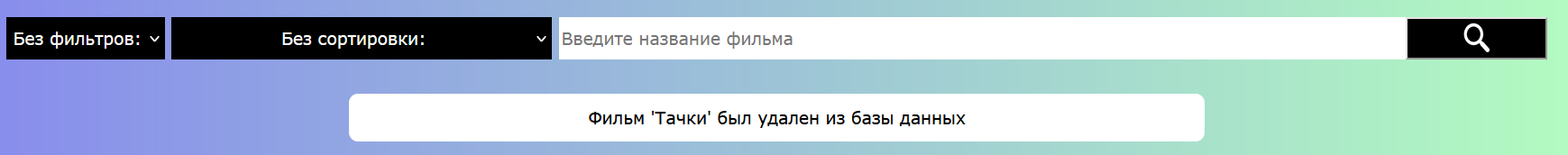


Рисунок 30 – Сообщение об успешном удалении

Ниже на рисунке 31 представлено детальное представление меню работы с залами.

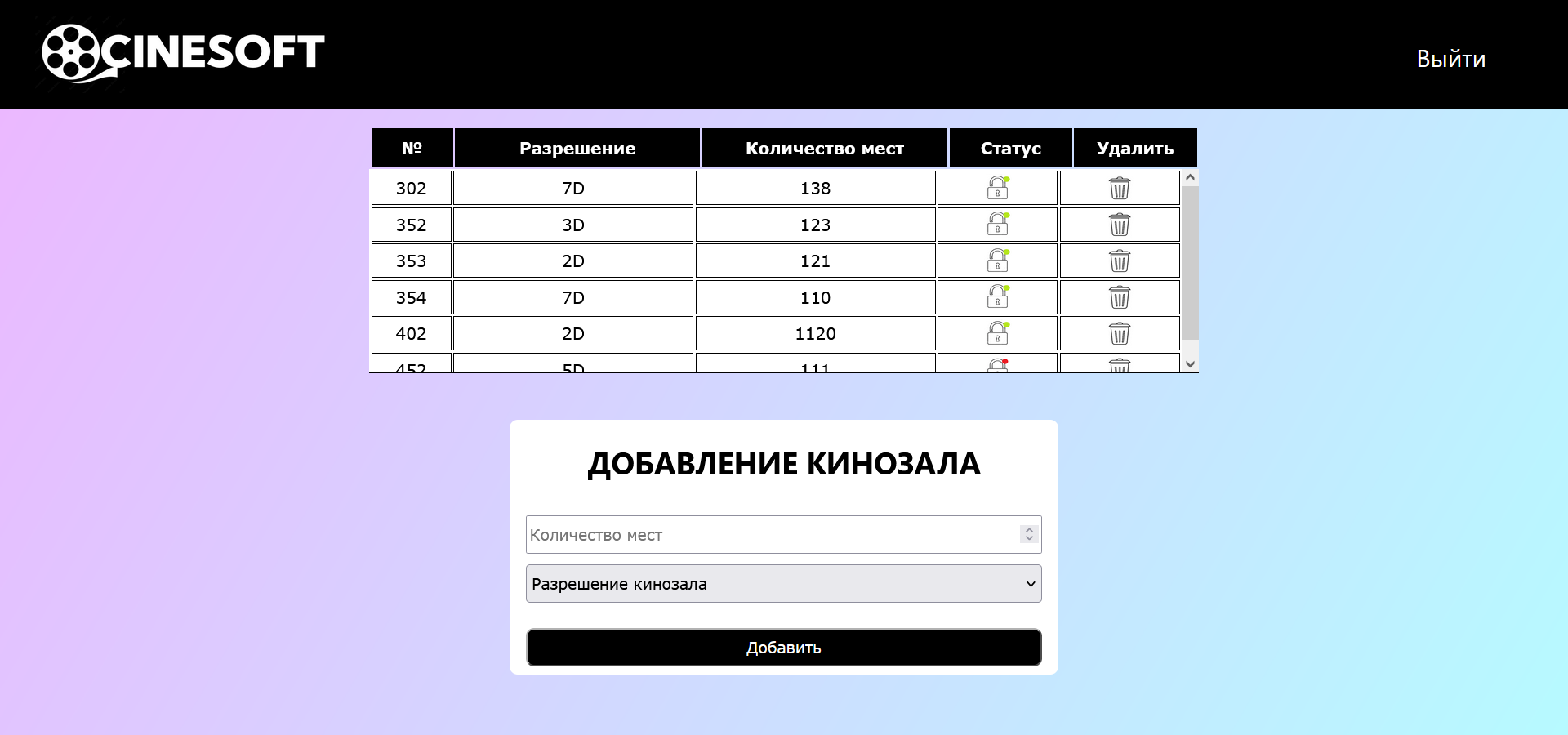


Рисунок 31 – Детальное представление меню работы с залами

При нажатии на значок корзины для определенного зала выводится сообщение о подтверждении удаления. Оно представлено на рисунке 32.

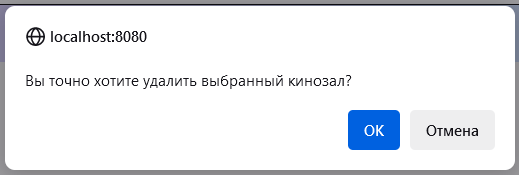


Рисунок 32 – Сообщение о подтверждении удаления кинозала

При подтверждении удаления зал удаляется из базы данных и выводится соответствующее сообщение. Оно представлено ниже на рисунке 33.

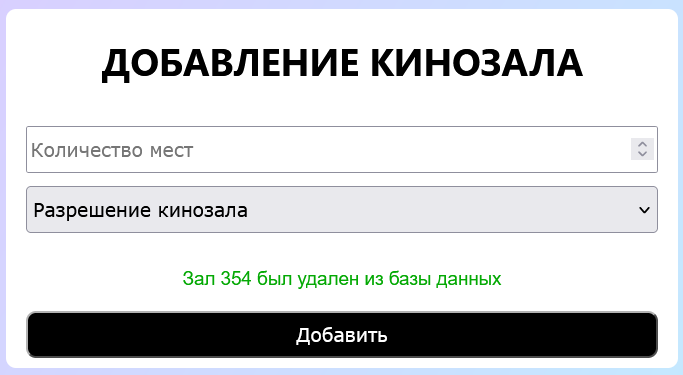


Рисунок 33 – Сообщение об успешном удалении

Ниже на рисунке 34 представлено детальное представление меню добавления сеанса.

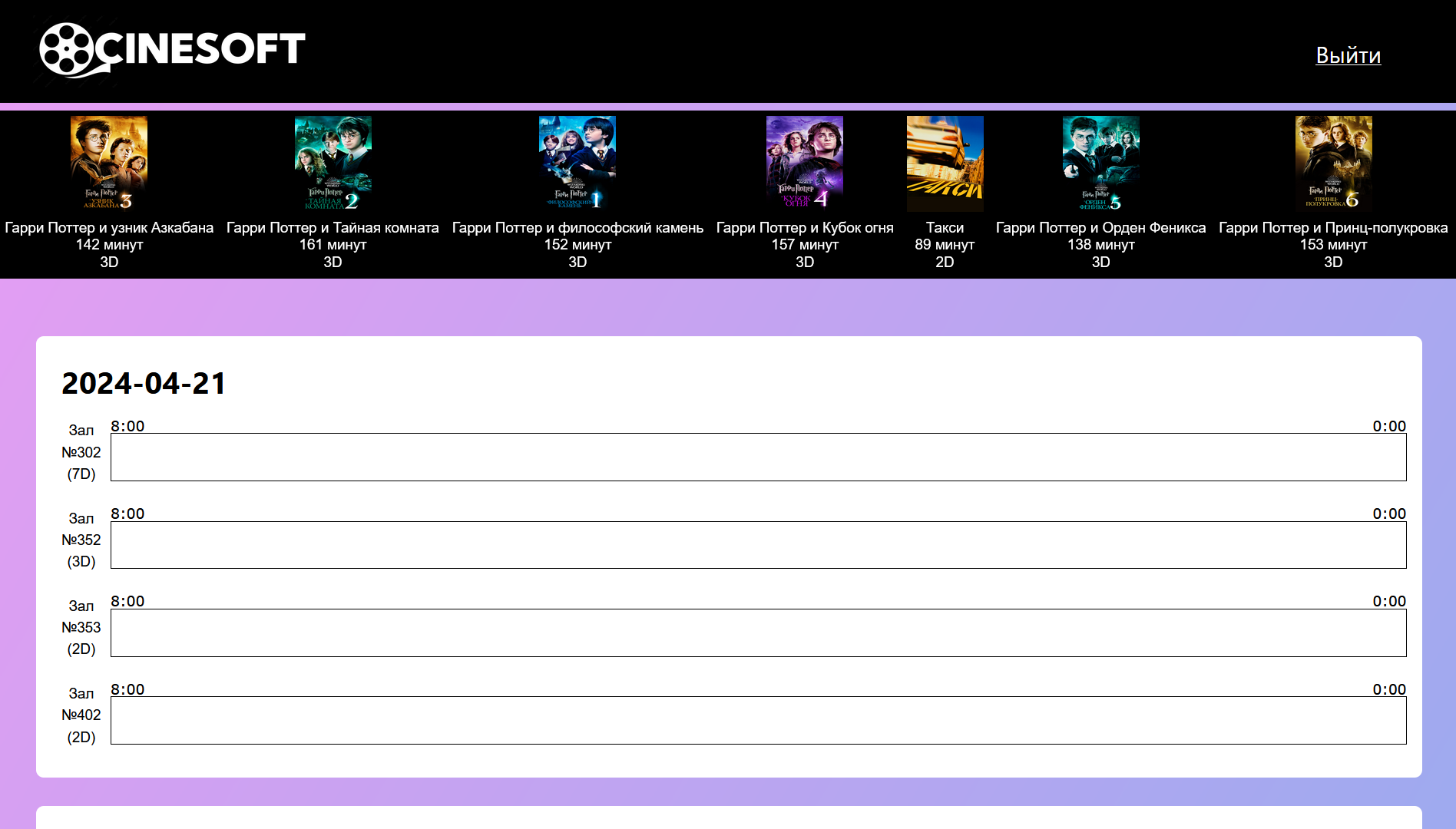


Рисунок 34 – Детальное представление меню добавления сеанса

Ниже на рисунке 35 представлено детальное представление меню работы с расписанием.



Рисунок 35 – Детальное представление меню работы с расписанием

Дизайны всех страниц были созданы простыми и интуитивно понятными для пользователей.

## Разработка модели данных

Для проектирования и разработки программного средства будут использованы данные, представленные в таблицах 8 – 10.

Таблица 8 – Данные о фильмах

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Номер фильма** | **Название** | **Жанр** | **Формат** | **Продолжительность, минуты** | **Описание** | **Возрастное ограничение** |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| 1 | «Тита-ник» | Драма, история | *2D* | 194 | В первом и последнем плавании шикарного «Титаника» встречаются двое. Пассажир нижней палубы Джек выиграл билет в карты, а богатая наследница Роза отправляется в Америку. Чувства молодых людей только успевают расцвести, и даже не классовые различия создадут испытания влюблённым, а айсберг. | 12+ |
| Номер фильма | Название | Жанр | Формат | Продолжительность, минуты | Описание | Возрастное ограничение |
| 2 | «Аватар» | Фантас-тика, боевик, драма, приклю-чения | *2D* | 162 | Бывший морпех Джейк Салли прикован к инвалидному креслу. Несмотря на немощное тело, Джейк в душе по-прежнему остается воином. Он получает задание совершить путешествие в несколько световых лет к | 12+ |

Окончание таблицы 8

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|  |  |  |  |  | базе землян на планете Пандора, где корпорации добывают редкий минерал, имеющий огромное значение для выхода Земли из энергетического кризиса. |  |
| 3 | «Губка Боб в *3D*» | Мульт-  фильм, фэнтези, коме-дия, приклю-  чения | *3D* | 80 | Всемирный любимчик со дна океана Губка Боб Квадратные Штаны должен спасти свой глубоководный мир, но для этого ему и его друзьям придется выбраться на сушу и оказаться в нашем. | 6+ |

Таблица 9 – Данные о кинозалах

|  |  |  |  |
| --- | --- | --- | --- |
| Номер зала | Разрешение | Количество мест | Статус |
| 1 | *2D* | 522 | Активен |
| 2 | *3D* | 456 | Активен |
| 3 | *5D* | 204 | Активен |
| 4 | *7D* | 101 | Неактивен |
| 5 | *2D* | 404 | Активен |

Таблица 10 – Данные о сеансах

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Номер сеанса | Дата начала | Время начала | Время конца | Номер фильма | Номер зала | Статус |
| 1 | 15.03 | 12:20 | 15:30 | 1 | 1 | Показан |
| 2 | 16.03 | 11:00 | 12:40 | 3 | 2 | В  процессе |
| 3 | 16.03 | 19:00 | 20:45 | 2 | 5 | Не начался |
| 4 | 18.03 | 15:30 | 19:00 | 1 | 1 | Не начался |

С помощью данных таблиц далее будет построена *ER*-модель. Основываясь на таблицы 12 – 14, можно выделить основные сущности предметной области: «Фильм», «Кинозал» и «Сеанс». Данные сущности представлены в виде *ER*-модели на рисунке 36.

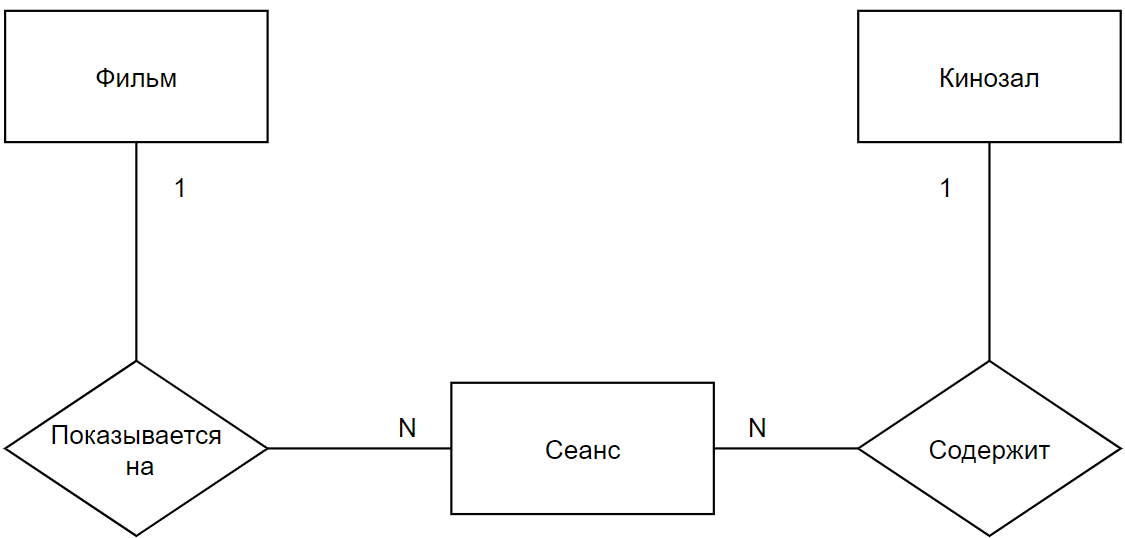


Рисунок 36 – *ER*-модель

На основе данной модели далее будет произведено логическое проектирование.

Рассмотрим сущность «Фильм». Ее ключевым атрибутом является поле «*ID*\_Фильма». Остальные атрибуты являются простыми, а именно «Название», «Жанр», «Формат», «Продолжительность», «Возрастное ограничение» и «Описание». Отображение сущности «Фильм» в нотации Чена представлено на рисунке 37.

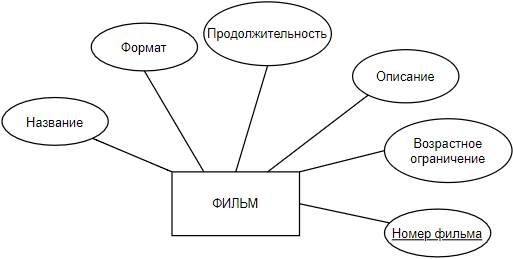


Рисунок 37 – Сущность «Фильм»

Рассмотрим сущность «Кинозал». Ее ключевым атрибутом является поле «Номер зала». Остальные атрибуты являются простыми, а именно «Разрешение», «Количество мест» и «Статус». Отображение сущности «Кинозал» в нотации Чена представлено на рисунке 38.



Рисунок 38 – Сущность «Кинозал»

Рассмотрим сущность «Сеанс». Ее ключевым атрибутом является поле «Номер сеанса». Остальные атрибуты являются простыми, а именно «Дата начала», «Время начала», «Время конца», «Номер фильма», «Номер зала» и «Статус». Отображение сущности «Сеанс» в нотации Чена представлено на рисунке 39.



Рисунок 39 – Сущность «Сеанс»

Ниже на рисунке 40 представлена логическая схема базы данных вместе со связями между указанными ранее сущностями.

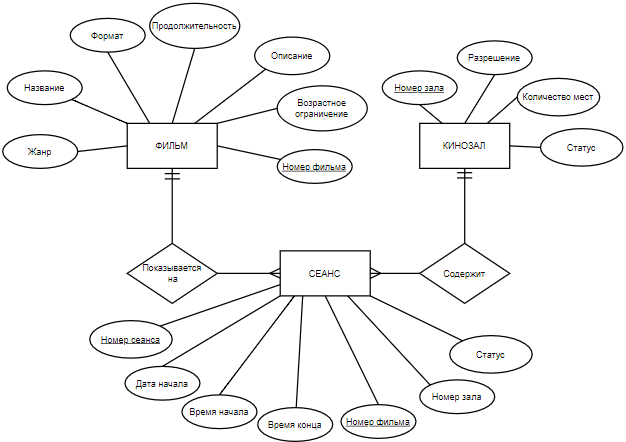


Рисунок 40 – Логическая схема базы данных

Для приведения схемы в третью нормальную форму была создана сущность «Жанр фильма». Она имеет атрибуты «Название жанра» и «ID\_Фильма». Данная сущность нужна для того, чтобы избежать неатомарности атрибута «Жанр» сущности «Фильм». Исправленная логическая модель данных представлена на рисунке 41.

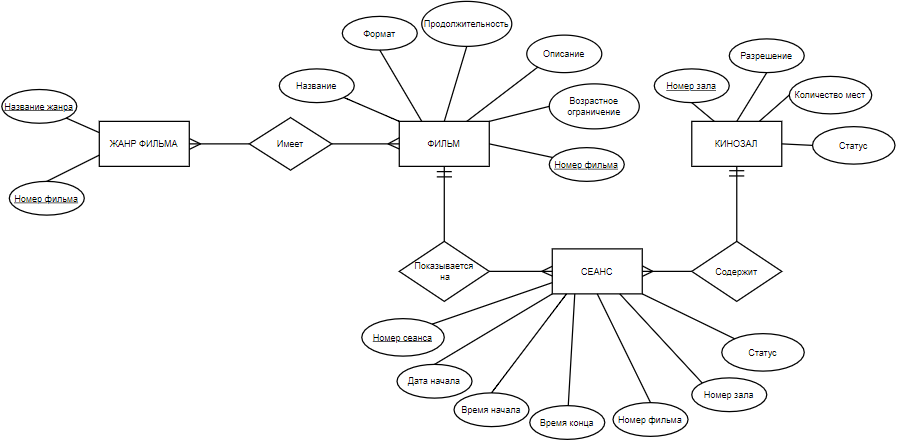


Рисунок 41 – Исправленная логическая схема базы данных

Данная схема соответствует третьей нормальной форме по следующим причинам:

1 Схема находится в первой нормальной форме, т. к. все атрибуты ее сущностей атомарны.

2 Схема находится во второй нормальной форме, т. к. она находится в первой нормальной форме и каждый неключевой атрибут находится в полной функциональной зависимости от ключа.

3..Схема находится в третьей нормальной форме, т. к. данное она находится во второй нормальной форме и каждый неключевой атрибут находится в нетранзитивной зависимости от первичного ключа.

На этапе физического проектирования необходимо выбрать конкретную систему управления базой данных и преобразовать логическую схему базы данных в физическую. Для примера воспользуемся *MySQL*, а в качестве визуального средства проектирования используем *MySQL Workbench*. В итоге преобразования получим физическую схему базы данных, представленную на рисунке 42.

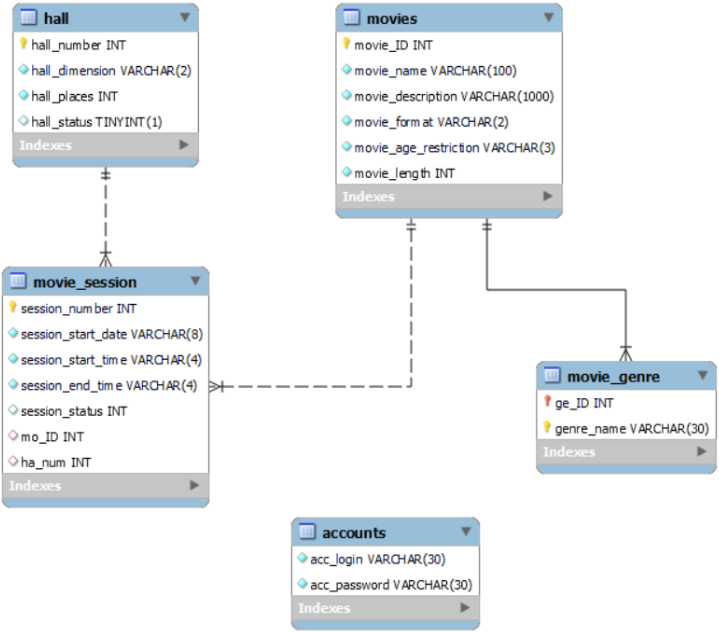


Рисунок 42 – Физическая схема базы данных

Текстовое описание сущностей с их полями представлено в таблице 11.

Таблица 11 – Описание сущностей базы данных

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование поля** | **Назначение атрибута** | **Тип данных** | **Примечание** |
| *movies (*фильмы*):* | | | |
| *movie\_ID* | Номер фильма | Целое число типа  *INT*, принимает  значения из диапазона [1;  9999999] | Первичный ключ |
| *movie\_name* | Название фильма | Строковое значение типа *VARCHAR*, содержащее до 100 символов |  |
| *movie\_description* | Описание фильма | Строковое значение типа *VARCHAR*, содержащее до 1000 символов |  |
| *movie\_format* | Формат фильма | Строковое значение типа *VARCHAR*, принимающее только значения «*2D*», «*3D*», «*5D*» и «*7D*» |  |
| *movie\_age\_restriction* | Возрастное ограничение фильма | Строковое значение типа *VARCHAR*, принимающее только значения «0+», «6+», «12+» и «18+» |  |
| *movie\_length* | Продолжитель-ность фильма | Целое число типа *INT*, принимает значения из диапазона [30;  240] |  |

Продолжение таблицы 11

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Наименование поля** | | **Назначение атрибута** | **Тип данных** | **Примечание** |
| *movie\_genre* (жанр фильма) | | | | |
| *ge\_id* | Номер фильма | | Целое число типа  *INT*, принимает  значения из диапазона [1;  9999999] | Первичный  ключ; внешний ключ, связывает с таблицей *movies*,  кардиналь-ность  связи − *M:N* |
| *genre\_name* | Название жанра | | Строковое значение типа *VARCHAR*, содержащее до 30 символов | Первичный ключ |
| *hall* (кинозал) | | | | |
| *hall\_number* | Номер зала | | Целое число типа  *INT*, принимает  значения из диапазона [1;  9999999] | Первичный ключ |
| *hall\_dimension* | Разрешение зала | | Строковое значение типа *VARCHAR*, принимающее только значения «*2D*», «*3D*», «*5D*» и «*7D*» |  |
| *hall\_places* | Количество мест в зале | | Целое число типа *INT*, принимает значения из диапазона [100;1500] |  |

Продолжение таблицы 11

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Наименование поля** | | **Назначение атрибута** | **Тип данных** | **Примечание** |
| *hall* (кинозал) | | | | |
| *hall\_status* | Статус кинозала | | Значение типа *TINYINT*(1), принимающее только значения 0 и 1 (используется для обозначения переменных типа *BOOL*) |  |
| *movie\_session* (сеанс) | | | | |
| *session\_number* | Номер сеанса | | Целое число типа  *INT*, принимает  значения из диапазона [1;  9999999] | Первичный ключ |
| *sesion\_start\_date* | Дата начала сеанса | | Строковое значение типа *VARCHAR*, содержащее до 8 символов |  |
| *sesion\_start\_time* | Время начала сеанса | | Строковое значение типа *VARCHAR*, содержащее до 4 символов |  |
| *sesion\_end\_time* | Время конца сеанса | | Строковое значение типа *VARCHAR*, содержащее до 4 символов |  |
| *sesion\_status* | Статус показа сеанса | | Целое число типа  *INT*, принимает только  значения 0, 1 и 2 |  |

Окончание таблицы 11

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Наименование поля** | | **Назначение атрибута** | **Тип данных** | **Примечание** |
| *movie\_session* (сеанс) | | | | |
| *mo\_id* | Номер фильма | | Целое число типа  *INT*, принимает  значения из диапазона [1;  9999999] | Внешний ключ,  связывает с  таблицей *movies*,  кардиналь-ность  связи − *M:*1 |
| *ha\_num* | Номер зала | | Целое число типа  *INT*, принимает  значения из диапазона [1;  9999999] | Внешний ключ,  связывает с  таблицей *hall*,  кардиналь-ность  связи − *M:*1 |

Все строчные данные в сущностях ограничены по размеру исходя из их назначения.

## Описание статических и динамических аспектов поведения программных объектов

**Описание статических аспектов программных объектов.** Для описания статических аспектов программных объектов были построены диаграммы классов, компонентов и размещения.

Диаграмма классов программного средства представлена ниже на рисунке 43.

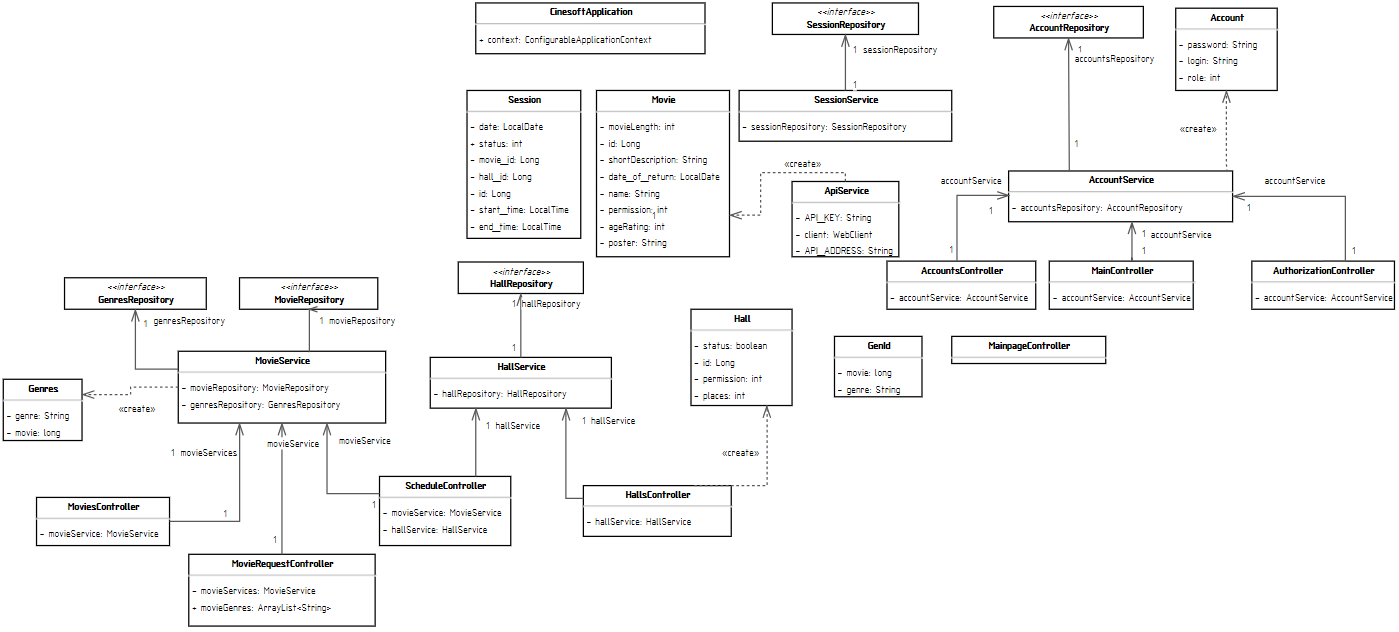


Рисунок 43 – Диаграмма классов

На диаграмме классов показаны классы, интерфейсы и кооперации разработанного программного средства, а также связи между ними.

На рисунке 44 представлена диаграмма компонентов программного средства.

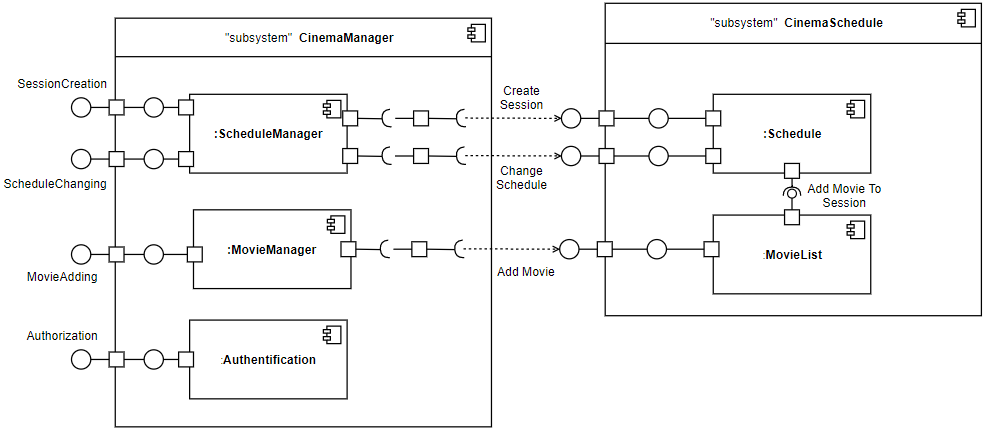


Рисунок 44 – Диаграмма компонентов

На диаграмме компонентов представлены инкапсулированные классы и их интерфейсы, порты и внутренние структуры, состоящие из вложенных компонентов и коннекторов.

На рисунке 45 представлена диаграмма размещения программного средства.

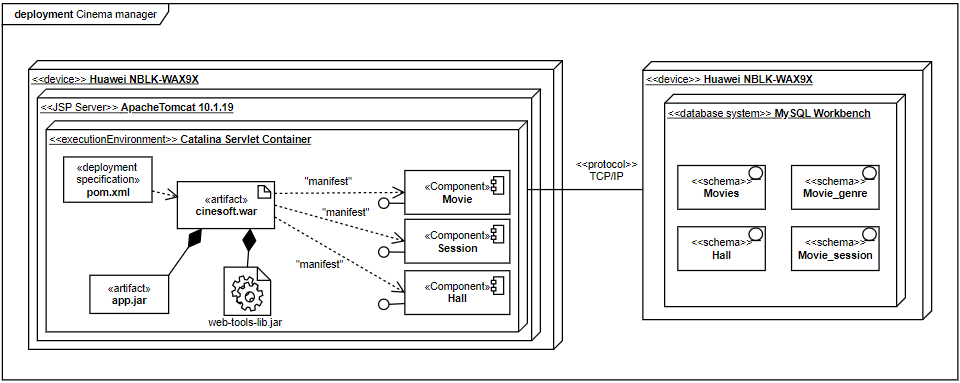


Рисунок 45 – Диаграмма размещения

На диаграмме размещения показана конфигурация узлов-процессоров, а также размещаемые на них компоненты. Она дает статическое представление размещения архитектуры.

**Описание динамических аспектов поведения программных объектов.**

Для моделирования поведения объектов программной системы широко используются следующие диаграммы:

1 Диаграмма деятельности – диаграмма, отображающая последовательности действий, исполняемых объектами программной системы.

2 Диаграмма последовательности – диаграмма, использующаяся для моделирования упорядоченного во времени взаимодействия объектов системы;

3 Диаграмма состояний – диаграмма, отображающая состояния объектов программной системы, а также переходы между этими состояниями.

Для описания поведения системы в рамках варианта использования «Создать сеанс» были построены диаграммы деятельности и последовательности. Для описания состояний объекта «Сеанс» была построена диаграмма состояний. Диаграмма деятельности представлена на рисунке 46.

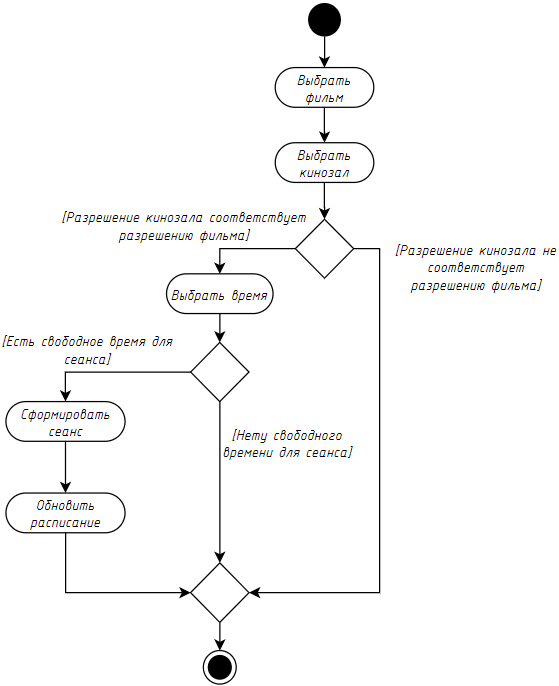


Рисунок 46 – Диаграмма деятельности для варианта использования «Создать сеанс»

На диаграмме деятельности описаны следующие шаги:

1 Выбрать фильм для показа.

2.Выбрать зал для показа фильма. При соответствии разрешения кинозала и разрешения фильма, зал закрепляется за сеансом. В противном случае возникает ошибка.

3 Выбрать время показа. При наличии свободного времени для показа сеанса, сеанс назначается на это время. В противном случае возникает ошибка.

4.Сформировать сеанс. Сформированный сеанс добавляется в расписание.

5 Обновить расписание. Добавленный сеанс сохраняется в базе данных.

Диаграмма последовательности представлена на рисунке 47.

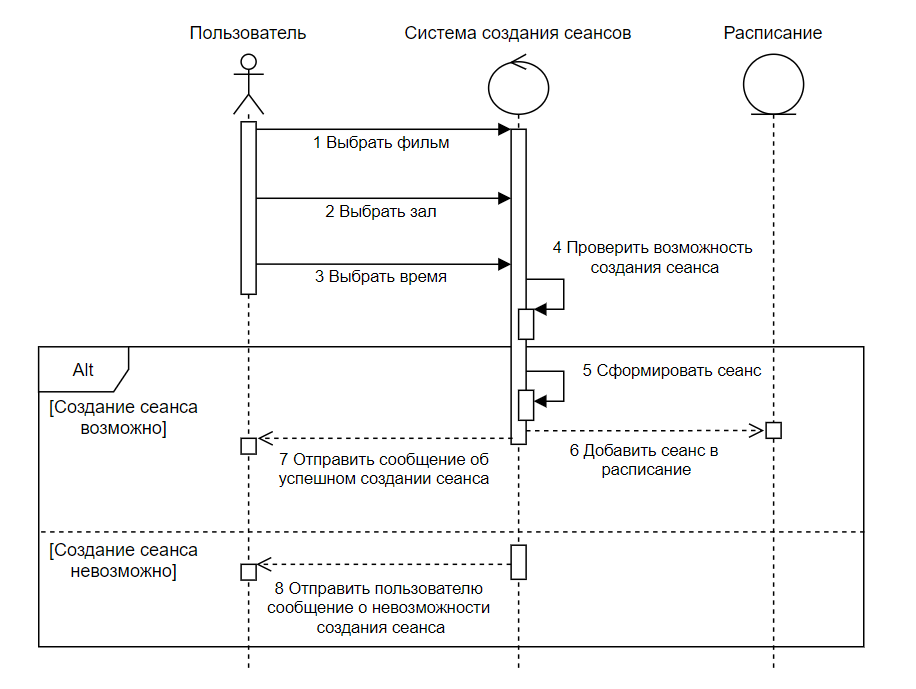


Рисунок 47 – Диаграмма последовательности для варианта использования «Создать сеанс»

В рамках варианта использования «Создать сеанс» взаимодействуют три объекта: пользователь, система создания сеансов и расписание. Для создания сеанса пользователю необходимо выбрать фильм для показа, зал и время показа. После этого система создания сеансов проверяет возможность создания сеанса на основе выбранных пользователем данных. Если создание сеанса возможно, то система формирует сеанс и добавляет его в расписание, после чего отправляет пользователю сообщение об успешном создании сеанса. В противном случае система выводит пользователю сообщение о невозможности создания сеанса.

Диаграмма состояний для объекта «Сеанс» представлена на рисунке 48.

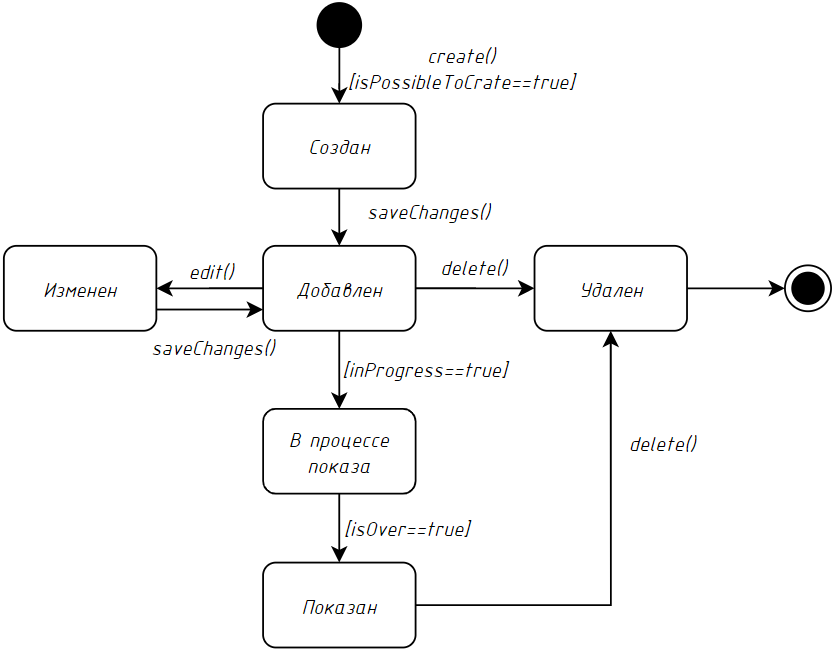


Рисунок 48 – Диаграмма состояний для объекта «Сеанс»

Как видно из диаграммы, сеанс может находиться в шести состояниях: «Создан», «Добавлен», «Изменен», «В процессе показа», «Показан» и «Удален». Изначально при успешном создании сеансу присваивается статус «Создан». При сохранении изменений в расписании созданный сеанс добавляется в базу данных кинотеатра, и его статус изменяется на «Добавлен».

При изменении времени или кинозала сеансу присваивается статус «Изменен», а после сохранения – статус «Добавлен». Если текущее время соответствует времени показа сеанса, то его статус становится «В процессе показа». Если же время конца сеанса прошло, то сеансу присваивается статус «Показан». Пользователь также имеет возможность удалить сеанс из расписания, и тогда статус сеанса становится «Удален».

## Разработка и описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства

Ниже представлено текстовое описание алгоритма, реализующего получение информации о фильме и добавление его в базу данных кинотеатра.

1 Пользователь авторизуется в системе. Вводит логин и пароль.

2 Осуществляется проверка логина и пароля. При успешной авторизации пользователю отображается главное меню программы. В противном случае выдается сообщение об ошибке, и работа завершается.

3 Отображается меню получения фильма.

4.Пользователь вводит название запрашиваемого фильма и отправляет запрос.

5.Система проверяет корректность введенных данных. Если данные корректны, то система производит поиск информации о фильме и выводит ее на экран. В противном случае пользователю выведется сообщение об ошибке.

6 При нажатии на кнопку добавления информация о фильме заносится в базу данных кинотеатра. В противном случае информация не сохраняется.

Для реализации алгоритма получения информации о фильме был создан отдельный метод класса *movieServices*. Пользователь заносит название фильма в поле ввода. По нажатию кнопки «Найти» введенные данные передаются на сервер. При получении сообщения от пользователя вызывается метод *getMovie()* класса *movieServices*. Код данного метода представлен ниже.

@PostMapping("/film\_request/get\_film")

public String getFilm(@RequestParam String title, Model model, HttpSession session) {

if (!movieGenres.isEmpty()) movieGenres.clear();

if (title.isEmpty()) model.addAttribute("error", "Введите название фильма");

else {

Movie movie = movieServices.getMovie(title);

if (movie.getId() == null) model.addAttribute("error", "Не найдено информации по запрашиваемому фильму");

else if (movie.getMovieLength() == 0)

model.addAttribute("error", "Данный фильм не предназначен для просмотра в кинотеатре");

else if (movieServices.findMovieById(movie.getId()))

model.addAttribute("error", "Данный фильм уже есть в базе данных кинотеатра");

else {

if (!movie.getName().equals(title))

model.addAttribute("warning", "Возможно, вы имели ввиду этот фильм");

String genres = "";

StringBuilder stringBuilder = new StringBuilder(genres);

for (String s : movieGenres) {stringBuilder.append(s + ", ");}

genres = stringBuilder.toString();

genres = genres.substring(0, genres.lastIndexOf(','));

model.addAttribute("movie", movie);

session.setAttribute("movie", movie);

model.addAttribute("genres", genres);

session.setAttribute("genres", movieGenres);

}

}

return "film\_request";

}

На рисунке 49 представлена схема алгоритма поиска информации о фильме.

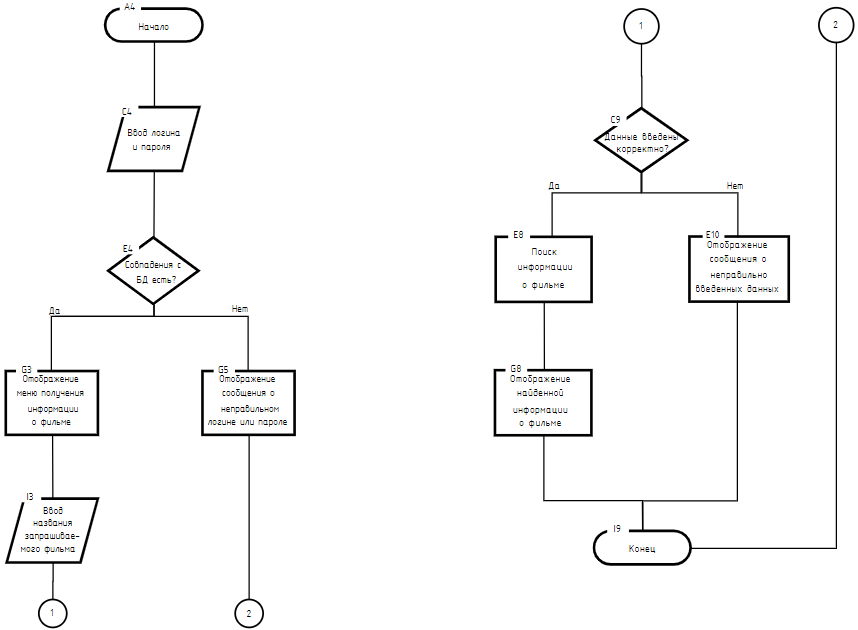


Рисунок 49 – Схема алгоритма поиска информации о фильме

Ниже представлено текстовое описание алгоритма, реализующего ввод информации о кинозале и добавление его в базу данных кинотеатра.

1 Пользователь авторизуется в системе. Вводит логин и пароль.

2 Осуществляется проверка логина и пароля. При успешной авторизации пользователю отображается главное меню программы. В противном случае выдается сообщение об ошибке, и работа завершается.

3 Отображается меню работы с кинозалами.

4.Пользователь выбирает разрешение фильма и вводит количество мест в нем.

5.Система проверяет корректность введенных данных. Если данные корректны, то система добавляет введенную информацию в базу данных и обновляет список кинозалов. В противном случае пользователю выведется сообщение об ошибке.

Для реализации алгоритма получения информации о фильме был создан отдельный метод класса *hallService*. Пользователь выбирает разрешение кинозала и вводит количество мест в нем. По нажатию кнопки «Добавить» введенные данные передаются на сервер. При получении сообщения от пользователя вызывается метод *addHallToDB()* класса *hallService*. Код данного метода представлен ниже.

@PostMapping("/halls/add")

public String addHall(@RequestParam String perm, @RequestParam int places, Model model, HttpSession session) {

System.out.println("perm: "+perm);int permission = Integer.parseInt(perm);

System.out.println("places: "+places);

Hall hall = new Hall(permission,true,places);

model.addAttribute("success","Созданный зал был добавлен в базу данных");

hallService.addHallToDB(hall);

return get(model, session);

}

Ниже на рисунке 50 представлена схема алгоритма добавления информации о кинозале.

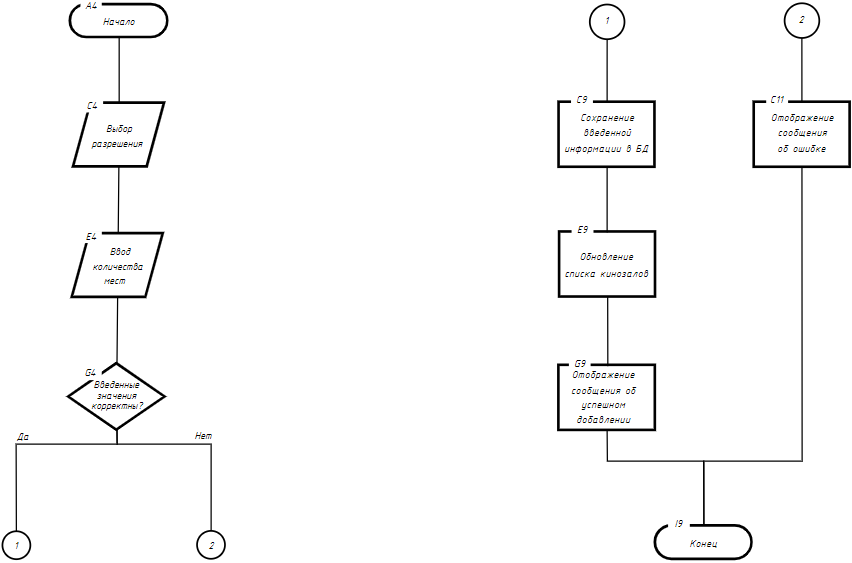


Рисунок 50 – Схема алгоритма добавления информации о кинозале

С помощью разработанных схем алгоритмов далее будут разработаны чертежи схем алгоритмов.

## Разработка программного средства

Функциональность серверной части программного средства реализована с использованием принципов *SOLID, DRY, KISS*, шаблона *DI*.

**Перечень функциональных возможностей, реализованных в программном средстве**. В таблице 12 представлен перечень функциональных возможностей программного средства.

Таблица 12 – Перечень функциональных возможностей

|  |  |  |  |
| --- | --- | --- | --- |
| **Функциональная возможность (*FN*)** | | **Роль пользователя** | |
| Иденти-  фикатор | Название | Админист-ратор | Сотрудник |
|  | **Управлять информацией о сотрудниках** |  |  |
| *FN*-01 | Добавление нового сотрудника | Да | Нет |
| *FN*-02 | Просмотр списка сотрудников | Да | Нет |
| *FN*-03 | Удаление сотрудника | Да | Нет |

Окончание таблицы 12

|  |  |  |  |
| --- | --- | --- | --- |
| **Функциональная возможность (*FN*)** | | **Роль пользователя** | |
| Иденти-  фикатор | Название | Админист-ратор | Сотрудник |
|  | **Управлять информацией о фильмах** |  |  |
| *FN*-04 | Поиск информации о фильме | Да | Да |
| *FN*-05 | Добавление фильма в базу данных | Да | Да |
| *FN*-06 | Просмотр списка фильмов | Да | Да |
| *FN*-07 | Просмотр информации об отдельном фильме | Да | Да |
| *FN*-08 | Удаление фильма | Да | Да |
|  | **Управлять информацией о кинозалах** |  |  |
| *FN*-09 | Просмотр списка кинозалов | Да | Да |
| *FN*-10 | Добавление информации о кинозале в базу данных | Да | Да |
| *FN*-11 | Изменение статуса кинозала | Да | Да |
| *FN*-12 | Удаление кинозала из базы данных | Да | Да |
|  | **Управлять расписанием** |  |  |
| *FN*-13 | Просмотр расписания сеансов | Да | Да |
| *FN*-14 | Добавление сеанса в расписание | Да | Да |
| *FN*-15 | Удаление изменение времени и/или зала показа сеанса | Да | Да |
| *FN*-16 | Удаление сеанса из расписания | Да | Да |

**Документация к разработанному *API*.** В таблице 13 дано описание запросов и ответов, возникающих при реализации вариантов использования, с представлением параметров и обработкой исключительных ситуаций, которые могут возникнуть в процессе выполнения запроса.

При написании программного кода были использованы принципы *DRY*, *KISS*, *SOLID*, а также использован шаблон *DI*.

***DRY.*** Данный принцип гласит избегать повторений одного и того же кода. Рассмотрим метод *deleteAllExpired()*. В данном методе происходит удаление фильмов с истекшим сроком проката*.* Так как это действие повторяется несколько раз, оно было вынесено в цикл. Пример реализации данного принципа на языке *Java* представлен ниже.

public void deleteAllExpired(LocalDate localDate) {

Iterable<Movie> movies = movieRepository.findAll();

for (Movie movie : movies) if (!movie.getDate\_of\_return().isAfter(localDate)) movieRepository.delete(movie);

}

***KISS***. Смысл этого принципа заключается в том, что стоит делать максимально простую и понятную архитектуру, применять шаблоны проектирования. Пример реализации данного принципа представлен методом *addAccountToDB()*. Этот метод короток, понятен и эффективен, в нем нет сложной логики. По этим причинам он соответствует принципу *KISS.* Программный код, реализующий данный метод, представлен ниже.

public boolean addAccountToDB(String login, String password, int role) {

try {

Account acc = new Account(login,hashedPassword,role);

accountsRepository.save(acc);

return true;

}

catch (Exception e){

return false;

}

}

**Принципы *SOLID*.** В упрощенном варианте означают, что когда при написании кода используется несколько принципов вместе, то это значительно облегчает дальнейшую поддержку и развитие программы. Пример реализации принципов *SOLID*на языке *Java* представлен ниже.

***Single responsibility principle –*** принцип единой ответственности. Его суть заключается в том, что на каждый класс должна быть возложена одна единственная обязанность. Примером реализации данного принципа являются классы *AccountService* и *HallService.* Каждый из них реализует свою обязанность: класс *AccountService* ответственен только за работу с учетными записями, а класс *HallService –* за работу с кинозалами. Ниже представлен код классов *AccountService* и *HallService*.

public class AccountService {

private final AccountRepository accountsRepository;

@Autowired

public AccountService(AccountRepository accountsRepository) {…}

public Iterable<Account> getAllAccounts() {…}

public boolean addAccountToDB(String login, String password, int role) {…}

public int checkPassword(String password){…}

public boolean findAccByLogin(String login) {…}

public boolean doesAdminExist() {…}

public Account getAcc(String login){…}

public boolean checkPassword(String password,Account acc){…}

public void deleteAccountByLogin(String login) {…}

}

@Service

public class HallService {

private final HallRepository hallRepository;

@Autowired

public HallService(HallRepository hallRepository) {…}

public Iterable<Hall> getAllHalls(){…}

public void changeStatus(long id){…}

public void delete(long id){…}

public Hall getHall(Long id){…}

public boolean addHallToDB(Hall hall){…}

public ArrayList<Hall> findNotBlockedHalls(){…}

}

***Open/closed principle*** – принцип открытости/закрытости. Он заключается в том, что программные сущности (классы, модули, функции и т. п.) должны быть открыты для расширения, но закрыты для изменения. Пример реализации данного принципа представлен методом *filter(),* созданным для фильтрации фильмов по разным параметрам, включая жанры. Этот метод открыт для расширения, поэтому он может выполнять фильтрацию для любого жанра. При появлении нового жанра метод не придется изменять. Реализация метода *filter()* на языке *Java* представлена ниже.

public Iterable<Movie> filter(Iterable<Movie> allMovies, String filt) {

ArrayList<Movie> movies = (ArrayList<Movie>) StreamSupport.stream(allMovies.spliterator(), false)

.collect(Collectors.toList());

Stream<Movie> movs = movies.stream();

switch (filt) {

case "-": return movies;

case "2", "3", "5", "7":

allMovies = movs.filter(m -> m.getPermission() == Integer.parseInt(filt)).collect(Collectors.toList());

return allMovies;

case "0", "6", "12", "18":

allMovies = movs.filter(m -> m.getAgeRating() == Integer.parseInt(filt)).collect(Collectors.toList());

return allMovies;

default:

System.out.println(filt);

ArrayList<Movie> deleted = new ArrayList<>();

for (Movie m : movies) {

if (!(findArrGenres(m.getId())).contains(filt)) {

deleted.add(m);

}

}

movies.removeAll(deleted);

}

return movies;

}

***Liskov substitution principle***– принцип подстановки Лисков. Его суть заключается в том, функции, которые используют базовый тип, должны иметь возможность использовать подтипы базового типа, не зная об этом. Данный принцип реализован в каждом контроллере программного средства. Реализован он таким образом, что вместо любого интерфейса сервиса может быть подставлен класс, реализующий данный интерфейс, и при этом программное средство будет работать корректно. Ниже представлена реализация данного принципа на примере метода *delete(),* в котором использован метод *deleteById()* интерфейса *HallRepository* и реализующего его класса *HallService*.

@Autowired

private HallService hallService;

@GetMapping("/halls/delete/{id}")

public String delete(@PathVariable(value = "id") long id, Model model, HttpSession session) {

System.out.println("hall id:" + id);

hallService.deleteById(id);

model.addAttribute("deleted",id);

return get(model, session);

}

@Autowired

private HallRepository hallRepository;

@GetMapping("/halls/delete/{id}")

public String delete(@PathVariable(value = "id") long id, Model model, HttpSession session) {

System.out.println("hall id:" + id);

hallRepository.deleteById(id);

model.addAttribute("deleted",id);

return get(model, session);

}

Из кода видно, что нету разницы между методами *deleteById()* интерфейса *HallRepository* и класса *HallService.* Это показывает успешную реализацию принципаподстановки Лисков.

***Interface segregation principle*** – принцип разделения интерфейсов. Он заключается в создании нескольких специализированных интерфейсов вместо одного универсального. Ниже продемонстрированы интерфейсы для работы с учетными записями, фильмами, кинозалами и сеансами. Для соответствия принципу они были разделены между собой, и у каждого из них своя зона ответственности.

public interface AccountRepository extends CrudRepository <Account,String> {

boolean existsByRole(int role);

}

public interface MovieRepository extends CrudRepository<Movie,Long> {

}

public interface HallRepository extends CrudRepository<Hall,Long> {

Iterable<Hall> findHallByStatus(boolean stat);

}

public interface SessionRepository extends CrudRepository<Session,Long> {

public void deleteAllByStatus(int status);

}

***Dependency inversion principle*** – принцип инверсии зависимостей. Он декларирует, что модули верхних уровней не должны зависеть от модулей нижних уровней, а оба типа модулей должны зависеть от абстракций; сами абстракции не должны зависеть от деталей, а вот детали должны зависеть от абстракций. Ниже представлен код метода *changeStatus()* для изменения статуса кинозала. Данный метод является модулем верхнего уровня, использующий для выполнения задачи одноименный метод класса *HallService* и не зависящий от него.

@GetMapping("/halls/status/{id}")

public String changeStatus(@PathVariable(value = "id") long id, Model model, HttpSession session) {

System.out.println("hall id:" + id);

hallService.changeStatus(id);

model.addAttribute("changed",id);

return get(model, session);

}

Внедрение зависимости (*DI*) – шаблон проектирования, суть которого заключается в следующем: зависимый объект (или функция) получает другой объект (или функцию), от которого он зависит. Примером реализации данного шаблона является метод *changeStatus()*. В нем используется переменная типа *Hall*, от которой зависит ход выполнения данного метода. Реализация данного шаблона на языке *Java* представлена ниже.

public void changeStatus(long id){

Hall hall = getHall(id);

if (hall.isStatus()) hall.setStatus(false);

else hall.setStatus(true);

hallRepository.save(hall);

}

## Механизмы обеспечения информационной безопасности

Обеспечение информационной безопасности– это комплекс мероприятий, направленных на предотвращение несанкционированного доступа к внутренней инфраструктуре, незаконного завладения конфиденциальной информацией и внесения изменений в базы данных. Учитывая важность информации в современном мире, защите от утечек конфиденциальной информации в адрес конкурентов необходимо уделять повышенное внимание. Возможный ущерб может быть намного большим, чем стоимость всех материальных активов предприятия.

При разработке программного средства были организованы процессы обеспечения информационной безопасности.

**Аутентификация и авторизация пользователей.** Аутентификация и авторизация обеспечивают безопасность пользовательских данных и защищают от несанкционированного доступа к конфиденциальной информации. В разработанном программном средстве данные процессы происходят посредством ввода логина и пароля.

Ниже представлен алгоритм аутентификации и авторизации пользователей.

1 Пользователь вводит логин и пароль.

2 Система проверяет наличие пользователя с введенным логином в базе данных. Если такого пользователя нет, то система выводит соответствующее сообщение.

3 При наличии пользователя с введенным логином в базе система проверяет корректность введенного пароля. Если пароль некорректен, то система выводит соответствующее сообщение.

4 В случае корректно введенного пароля система проверяет роль пользователя и создает ему соответствующую пользовательскую сессию.

Для реализации аутентификации и авторизации был создан метод *authorize*. Его описание представлено ниже.

@PostMapping("/authorization/authorize")

public String authorize(@RequestParam String login, @RequestParam String password, Model model, RedirectAttributes redirectAttributes, HttpSession session) {

if (login == "") model.addAttribute("emptyLog", "Введите логин");

else {

if (password == "") {model.addAttribute("emptyPass", "Введите пароль");

model.addAttribute("getLogin", login);}

else {

if (!accountService.findAccByLogin(login)) {model.addAttribute("error", "Неправильный логин или пароль");

model.addAttribute("getLogin", login);

model.addAttribute("getPassword", password);}

else {

Account acc = accountService.getAcc(login);

if (!accountService.checkPassword(password, acc)){

model.addAttribute("error", "Неправильный логин или пароль");

model.addAttribute("getLogin", login);

model.addAttribute("getPassword", password);}

else {

if (acc.getRole() == 1) {

session.setAttribute("user", "admin");

redirectAttributes.addFlashAttribute("user", "admin");

} else {

session.setAttribute("user", "user");

redirectAttributes.addFlashAttribute("user", "user");

}

return "redirect:/mainpage";

}

}

}

}

return "authorization";

}

**Защита учетных записей пользователей от несанкционированного доступа.** Защита от несанкционированного доступа нужна для защиты конфиденциальных данных. В разработанном программном средстве защита от несанкционированного доступа происходит посредством создания и проверки пользовательских сессий.

Ниже представлен алгоритм, с помощью которого реализован данный процесс.

1 Пользователь авторизуется в системе.

2 Система проверяет роль пользователя и создает ему соответствующую пользовательскую сессию.

3 Пользователь переходит в главное меню.

4 Программное средство отображает только тот контент, к которому имеет доступ пользователь с ролью, указанной в сессии.

Пример реализации защиты от несанкционированного входа в меню работы с учетными записями представлен в методе *get*, чей код представленниже.

@GetMapping("/accounts")

public String get(Model model, HttpSession session) {

System.out.println("accs get");

String us = (String) session.getAttribute("user");

if (us != null) {

if (us.equals("user")) return "mainpage";

if (us.equals("admin")) {

Iterable<Account> accounts = accountService.getAllAccounts();

if (StreamSupport.stream(accounts.spliterator(), false).count() == 0)

model.addAttribute("noAccs", "Список учетных записей пуст");

else model.addAttribute("accounts", accounts);

return "accounts";

} }

return "redirect:/authorization";

}

**Обеспечение безопасности хранимых данных.** Обеспечение безопасности хранимых данных нужно для защиты от их потери и несанкционированного доступа к ним. В разработанном программном средстве обеспечение безопасности хранимых данных происходит с помощью хэширования.

Ниже представлен алгоритм, с помощью которого реализован данный процесс.

1 Администратор вводит логин и пароль для добавляемого сотрудника.

2 Паролю создается его уникальное хэш-значение.

3 Хэшированный пароль заносится в базу данных.

Реализация механизма защиты хранимых данных представлена в методе *addAccountToDB*, код которого приведен ниже.

public boolean addAccountToDB(String login, String password, int role) {

try {

String salt = BCrypt.gensalt();

String hashedPassword = BCrypt.hashpw(password, salt);

Account acc = new Account(login,hashedPassword,role);

accountsRepository.save(acc); return true;

}

catch (Exception e){

return false;}

}

Таким образом, обеспечение информационной безопасности поможет защитить информацию кинотеатра от несанкционированного доступа, потери и утечки.

# ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

Тестирование и проверка работоспособности программного средства играют ключевую роль в разработке качественного и надежного программного обеспечения. Важность тестирования и проверки работоспособности заключается в следующих аспектах, представленных ниже.

1 Обнаружение ошибок. Тестирование помогает выявить ошибки, дефекты и несоответствия в функциональности программного средства. Раннее обнаружение и исправление ошибок позволяет снизить затраты на исправление проблем в дальнейшем.

2 Гарантия качества. Тестирование помогает убедиться, что программное средство соответствует требованиям заказчика, работает корректно и выполняет свои функции без сбоев. Это обеспечивает высокое качество продукта и удовлетворение конечных пользователей.

3 Повышение надежности. Тестирование помогает улучшить надежность программного средства, обнаруживая потенциальные проблемы, которые могут привести к сбоям или некорректной работе приложения. Надежное программное обеспечение повышает доверие пользователей и уменьшает риски для бизнеса.

Для проверки работоспособности программного средства было проведено тестирование на основе тест-кейсов.

Тестирование на основе тест-кейсов – формализованный подход, в котором тестирование производится на основе заранее подготовленных тест-кейсов, наборов тест-кейсов и иной документации. Тест-кейс – набор входных данных, условий выполнения и ожидаемых результатов, разработанный с целью проверки того или иного свойства или поведения программного средства.

Ниже в таблице 13 представлены разработанные тест-кейсы для проверки уровня базовых пользовательских требований к программному средству.

Таблица 13 – Тест-кейсы для проверки уровня базовых пользовательских   
 требований

|  |  |  |  |
| --- | --- | --- | --- |
| **Иденти-фикатор тест-кейса** | **Заглавие**  **тест-кейса** | **Шаги тест-кейса** | **Ожидаемый результат** |
| **1** | **2** | **3** | **4** |
| *UC-1* | Проверка функции приобре-тения фильма | 1 Перейти в меню приобретения фильма.  2 Ввести название фильма.  3 Нажать на кнопку «Найти».  4 Выбрать разрешение фильма.  5 Ввести срок проката.  6 Нажать на кнопку «Добавить» | 1 Отображение поля ввода названия фильма.  2 Отображение введенного  названия.  3 Отображение постера, названия, продолжительности, жанров, возрастного ограничения и описания найденного фильма, а также полей выбора разрешения и срока проката.  4 Отображение выбранного разрешения.  5 Отображение введенного срока проката.  6 Добавление фильма в базу данных кинотеатра, вывод сообщения об успешном добавлении. |
| *UC-2* | Проверка функции создания кинозала | 1 Перейти в меню работы с кинозалами.  2 Выбрать разрешение кинозала.  3 Ввести количество мест в кинозале.  4 Нажать на кнопку «Добавить». | 1 Отображение таблицы кинозалов и поля ввода информации о кинозале.  2 Отображение выбранного разрешения.  3 Отображение введенного количества мест.  4 Добавление зала в базу данных кинотеатра, вывод сообщения об успешном добавлении. |
| *UC-3* | Проверка функции изме-  нения статуса кинозала | 1 Перейти в меню работы с кинозалами.  2 Нажать на значок замка в строчке зала в таблице. | 1 Отображение таблицы кинозалов.  2 Изменение статуса кинозала на противоположный. |

Продолжение таблицы 13

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **2** | **3** | **4** |
| *UC-4* | Проверка функции удаления кинозала | 1 Перейти в меню работы с кинозалами.  2 Нажать на значок мусорной корзины в строчке определенного зала в таблице. | 1 Отображение таблицы кинозалов.  2 Удаление кинозала из базы данных, удаление записи об удаленном кинозале из таблицы. |
| *UC-5* | Проверка функции создания сеанса | 1 Перейти в меню создания сеанса.  2 Перетянуть фильм на дорожку определенного зала в определенную дату.  3 Нажать на кнопку «Добавить». | 1 Отображение списка фильмов и дорожек кинозалов на разные даты.  2 Отображение окна с выбранными фильмом, датой, залом, временем начала и конца показа.  3 Добавление созданного сеанса в базу данных, вывод сообщения об успешном добавлении. |
| *UC-6* | Проверка функции удаления сеанса | 1 Перейти в меню расписания.  2 Перетянуть сеанс за пределы дорожек залов.  3 Нажать на кнопку «Удалить». | 1 Отображение расписания сеансов и списков значений для фильтрации «Зал:» и «День:».  2 Отображение окна для подтверждения удаления.  3 Удаление сеанса из базы данных, вывод сообщения об успешном удалении. |
| *UC-7* | Проверка функции измене-ния расписа-ния | 1 Перейти в меню расписания.  2 Перетянуть сеанс на другую дорожку кинозала или на другое место текущей.  3 Нажать на кнопку «Сохранить». | 1 Отображение расписания сеансов и списков значений для фильтрации «Зал:» и «День:».  2 Отображение окна для подтверждения изменения, в котором указываются новые зал, день и время показа.  3 Обновление информации о сеансе в базе данных, вывод сообщения об успешном изменении. |

Продолжение таблицы 13

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | | **2** | **3** | **4** |
| *UC-8* | | Проверка функции просмот-ра расписа-ния | 1 Перейти в меню расписания. | 1 Отображение расписания сеансов и списков значений для фильтрации «Зал:» и «День:». |
| *UC-9* | | Проверка функции просмот-ра расписа-ния на выбран-ный зал | 1 Перейти в меню расписания.  2 Выбрать определенный кинозал из списка «Кинозал:». | 1 Отображение расписания сеансов и списков значений для фильтрации «Зал:» и «День:».  2 Отображение на каждый день расписания только для выбранного зала. |
| *UC-10* | | Проверка функции просмот-ра расписа-ния на выбран-ный день | 1 Перейти в меню расписания.  2 Выбрать определенный день из списка «День:». | 1 Отображение расписания сеансов и списков значений для фильтрации «Зал:» и «День:».  2 Отображение расписания только на выбранный день. |
| *UC-11* | Проверка функции просмот-ра списка фильмов | | 1 Перейти в меню работы с фильмами. | 1 Отображение списка всех фильмов из базы данных. |
| *UC-12* | Проверка функции просмот-ра информа-ции о фильме | | 1 Перейти в меню работы с фильмами.  2 Нажать на кнопку «Детали» для выбранного фильма. | 1 Отображение списка всех фильмов из базы данных.  2 Отображение постера, названия, продолжительности, жанра, разрешения, возрастного ограничения, срока проката и краткого описания выбранного фильма. |

Ниже в таблице 14 представлены результаты автоматизированного тестирования функциональных возможностей программного средства.

Таблица 14 - Результаты автоматизированного тестирования

функциональности ПС

|  |  |  |  |
| --- | --- | --- | --- |
| **Тестируе-мый тип** | **Дата проведения теста** | **Результаты тестирования** | **Примечания** |
| **1** | **2** | **3** | **4** |
| Функция добавления фильма в базу данных | 19.03.2024 | При нажатии на кнопку «Добавить» фильм добавляется в базу данных и выводится соответствующее сообщение | Информация о добавленном фильме убирается с экрана |
| Функция добавления кинозала в базу данных | 19.03.2024 | При нажатии на кнопку «Добавить» кинозал добавляется в базу данных и выводится соответствующее сообщение | В таблице кинозалов появляется строка созданного зала |
| Функция изменения статуса кинозала | 19.03.2024 | При нажатии на картинку замка статус кинозала изменяется на противоположный, происходит обновление информации о нем в базе данных | Меняется картинка замка в соответствующей строчке таблицы |
| Функция удаления кинозала | 19.03.2024 | При нажатии на картинку корзины появляется окно подтверждения удаления. При подтверждении кинозал удаляется из базы данных | Строка удаленного кинозала убирается из таблицы |
| Функция создания сеанса | 20.03.2024 | При перетаскивании фильма на дорожку определенного кинозала появляется окно с номером фильма, номером зала, датой, временем начала и конца показа и кнопкой «Добавить». При нажатии на эту кнопку происходит добавление | На дорожке выбранного для созданного сеанса кинозала появляется прямоугольник, отображающий созданный сеанс |

Продолжение таблицы 14

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **2** | **3** | **4** |
| Функция создания сеанса |  | созданного сеанса в базу данных |  |
| Функция удаления сеанса | 20.03.2024 | При перетаскивании прямоугольника, отображающего сеанс, за пределы дорожек фильма, выводится сообщение о подтверждении удаления. При подтверждении происходит удаление сеанса из базы данных и вывод соответствующего сообщения | С дорожки зала, соответствующего удаленному сеансу, убирается прямоугольник, отображающий удаленный сеанс |
| Функция изменения расписания | 20.03.2024 | При перетаскивании прямоугольника, отображающего сеанс, на другую дорожку зала или на другое место текущей, выводится сообщение о подтверждении изменения. При подтверждении происходит изменение информации о сеансе в базе данных и вывод соответствующего сообщения | С дорожки зала, соответствующей сеансу до изменения, убирается прямоугольник, отображающий сеанс до изменения. На дорожке выбранного для измененного сеанса кинозала появляется прямоугольник, отображающий измененный сеанс |
| Функция отображения расписания | 20.03.2024 | При переходе в меню расписания происходит считывание информации о сеансах из базы данных и отображение их на экране на соответствующих дорожках кинозалов в соответствующие дни |  |

Продолжение таблицы 14

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **2** | **3** | **4** |
| Функция фильтрации расписания по дням | 20.03.2024 | При нажатии на список «День:» и выборе определенной даты происходит фильтрация списка залов, и на экран выводится расписание только на выбранную дату | Надпись на кнопке «День:» меняется на выбранную дату |
| Функция фильтрации расписания по залам | 20.03.2024 | При нажатии на список «Зал:» и выборе определенного зала происходит фильтрация списка залов, и на экран выводится расписание только для выбранного зала | Надпись на кнопке «Зал:» меняется на номер выбранного зала |
| Функция отображения списка фильмов | 22.03.2024 | При переходе в меню работы с фильмами происходит считывание информации обо всех фильмах из базы данных и отображение на экране постера, названия, продолжительности, жанров и даты возврата для каждого из них |  |
| Функция отображения информации о фильме | 22.03.2024 | При нажатии на кнопку «Детали» выбранного фильма происходит переход на страницу с информацией об определенном фильме и отображение на экране постера, названия, продолжительности, жанров, даты возврата и краткого описания для этого фильма |  |

Для всех реализованных возможностей программного средства были разработаны *UNIT*-тесты.

Ниже представлен класс, реализующий функционал для работы с фильмами (добавления фильма и получения списка всех фильмов), и *UNIT*-тест для проверки его работоспособности.

@Service

public class MovieService {

private final MovieRepository movieRepository;

private final GenresRepository genresRepository;

@Autowired

public MovieService(MovieRepository movieRepository, GenresRepository genresRepository) {

this.movieRepository = movieRepository;

this.genresRepository = genresRepository;

}

public Iterable<Movie> getAllMovies() {

Iterable<Movie> movies = movieRepository.findAll();

return movies;

}

public boolean addMovieToDB(Movie movie) {

try {

movieRepository.save(movie);

return true;

} catch (Exception e) {

return false;

}

}

}

class MovieServiceTest {

MovieService movieService;

MovieRepository movieRepository;

GenresRepository genresRepository;

@Before

public void setUp(){

movieService = new MovieService(movieRepository,genresRepository);

}

@Test

void addMovieToDB() {

Movie movie = new Movie();

assertEquals(true,movieService.addMovieToDB(movie));

}

@Test(expected = Exception.class)

void addEmptyMovieToDB() {

Movie movie = null;

movieService.addMovieToDB(movie);

}

@Test

void getAllMovies() {

Movie mov1 = new Movie();movieService.addMovieToDB(mov1);

Movie mov2 = new Movie();movieService.addMovieToDB(mov2);

ArrayList<Movie> arr1 = new ArrayList<>();

arr1.add(mov1);arr1.add(mov2);

ArrayList<Movie> arr2 = (ArrayList<Movie>) movieService.getAllMovies();

assertEquals(arr1,arr2);

}

}

Ниже представлен класс, реализующий функционал для работы с кинозалом (создание кинозала, изменение статуса кинозала и удаление кинозала), а также *UNIT*-тесты для проверки его работоспособности.

public class HallService {

private final HallRepository hallRepository;

@Autowired

public HallService(HallRepository hallRepository) {

this.hallRepository = hallRepository;

}

public Iterable<Hall> getAllHalls(){

Iterable<Hall> halls =hallRepository.findAll();

return halls;

}

public void changeStatus(long id){

Hall hall = getHall(id);

if (hall.isStatus()) hall.setStatus(false);

else hall.setStatus(true);

hallRepository.save(hall);

}

public boolean deleteById(long id){

try {

hallRepository.deleteById(id);return true;

}

catch (Exception e){

return false;

}

}

public boolean addHallToDB(Hall hall){

try {

hallRepository.save(hall);return true;

}

catch (Exception e){

return false;

}

}

}

class HallServiceTest {

HallService hallService;

HallRepository hallRepository;

@Before

public void before(){

hallService = new HallService(hallRepository);

}

@Test

void getAllHalls() {

Hall hall1 = new Hall();

Hall hall2 = new Hall();

hallService.addHallToDB(hall1);

hallService.addHallToDB(hall2);

ArrayList<Hall> arr1 = new ArrayList<>();arr1.add(hall1);arr1.add(hall2);

ArrayList<Hall> arr2 = (ArrayList<Hall>) hallService.getAllHalls();

assertEquals(arr1,arr2);

}

@Test

void addHallToDB() {

Hall hall = new Hall();

assertEquals(true,hallService.addHallToDB(hall));

}

@Test(expected = Exception.class)

void addEmptyHallToDB() {

Hall hall = null;

hallService.addHallToDB(hall);

}

@Test

void changeStatus() {

Hall hall = new Hall(3,true,1000);

hallService.addHallToDB(hall);

hallService.changeStatus(hall.getId());

assertEquals(false,hall.isStatus());

}

@Test(expected = Exception.class)

void changeStatusOfNothing() {

Hall hall = new Hall();

hallService.deleteById(hall.getId());

}

@Test

void deleteById() {

Hall hall = new Hall();

hallService.addHallToDB(hall);

assertEquals(true,hallService.deleteById(hall.getId()));

}

@Test(expected = Exception.class)

void changeStatusOfNothing() {

Hall hall = new Hall();

hallService.changeStatus(hall.getId());

}

}

Ниже представлен класс, реализующий функционал для работы с расписанием (создание сеанса, изменение расписания и удаление сеанса, а также фильтрация расписания по выбранным дням и залам), а также *UNIT*-тесты для проверки его работоспособности.

@Service

public class SessionService {

private final SessionRepository sessionRepository;

@Autowired

public SessionService(SessionRepository sessionRepository) {

this.sessionRepository = sessionRepository;

}

public Iterable<Session> getAllSessions(){

Iterable<Session> sessions =sessionRepository.findAll();

return sessions;

}

public boolean createSession(Session session){

try {

sessionRepository.save(session);return true;

}

catch (Exception e){

return false;

}

}

public boolean updateSession(Session session){

try {

sessionRepository.save(session);return true;

}

catch (Exception e){

return false;

}

}

public boolean deleteSession (Long id){

try {

sessionRepository.deleteById(id);return true;

}

catch (Exception e){

return false;

}

}

}

class SessionServiceTest {

SessionService sessionService;

SessionRepository sessionRepository;

@Before

public void setUp(){

sessionService=new SessionService(sessionRepository);

}

@Test

void getAllSessions() {

Session ses1 = new Session();

Session ses2 = new Session();

sessionService.createSession(ses1);

sessionService.createSession(ses2);

ArrayList<Session> arr1 = new ArrayList<>();arr1.add(ses1);arr1.add(ses2);

ArrayList<Session> arr2 = (ArrayList<Session>) sessionService.getAllSessions();

assertEquals(arr1,arr2);

}

@Test

void createSession() {

Session session = new Session();

assertEquals(true,sessionService.createSession(session));

}

@Test(expected = Exception.class)

void createNullSession() {

Session session = null;

sessionService.createSession(session);

}

@Test

void updateSession() {

Session session = new Session(2222L,301L, LocalDate.now(), LocalTime.now(),LocalTime.now().plusHours(2));

sessionService.createSession(session);

session.setDate(LocalDate.now().plusDays(2));

assertEquals(true,sessionService.updateSession(session));

}

@Test(expected = Exception.class)

void updateNotSavedSession() {

Session session = new Session(2222L,301L, LocalDate.now(), LocalTime.now(),LocalTime.now().plusHours(2));

session.setDate(LocalDate.now().plusDays(2));

sessionService.updateSession(session);

}

@Test

void deleteSession() {

Session session = new Session();

sessionService.createSession(session);

assertEquals(true,sessionService.deleteSession(session.getId()));

}

@Test(expected = Exception.class)

void deleteNotSavedSession() {

Session session = new Session();

sessionService.deleteSession(session.getId());

}

@Test

void getScheduleByDay() {

Session ses1 = new Session(2222L,301L, LocalDate.of(22,4,2024), LocalTime.now(),LocalTime.now().plusHours(2));

Session ses1 = new Session(2222L,301L, LocalDate.of(23,4,2024), LocalTime.now(),LocalTime.now().plusHours(2));

sessionService.createSession(ses1);

sessionService.createSession(ses2);

ArrayList<Session> arr1 = new ArrayList<>();

arr1.add(ses1);

ArrayList<Session> arr2 = (ArrayList<Session>)

sessionService.getSessionsByDate(LocalDate.of(22,4,2024));

assertEquals(arr1,arr2);

}

@Test

void getScheduleByHall() {

Session ses1 = new Session(2222L,302L, LocalDate.of(22,4,2024), LocalTime.now(),LocalTime.now().plusHours(2));

Session ses1 = new Session(2222L,301L, LocalDate.of(22,4,2024), LocalTime.now(),LocalTime.now().plusHours(2));

sessionService.createSession(ses1);

sessionService.createSession(ses2);

ArrayList<Session> arr1 = new ArrayList<>();

arr1.add(ses2);

ArrayList<Session> arr2 = (ArrayList<Session>)

sessionService.getSessionsByHall(301L);

assertEquals(arr1,arr2);

}

}

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Роль кинематографа в жизни современного человека [Электронный ресурс]. – Режим доступа: https://pnu.edu.ru/media/filer\_public/d0/42/d04266bd-8c4f-4aa4-8604-e809d6c09  
   094/23.pdf – Дата доступа: 08.03.2024.
2. Мультиплекс [Электронный ресурс]. – Режим доступа: https://pnu.edu.ru/media/filer\_public/d0/42/d04266bd-8c4f-4aa4-8604-e809d6c09094/23.pdf – Дата доступа: 08.03.2024.
3. Мультиплекс (кинотеатр) [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/multiplex(cinema)– Дата доступа: 08.03.2024.
4. Управление деятельностью предприятия [Электронный ресурс]. – Режим доступа: https://studbooks.net/741958/turizm/upravlenie\_deyatelnostyu \_predpriyatiya\_obschie\_ponyatiya – Дата доступа: 08.03.2024.
5. Theater Management System [Электронный ресурс]. – Режим доступа: https://pro.sony/s3/cms-static-content/file/01/1237493048701.pdf – Дата доступа: 08.03.2024.
6. TMS (Система управления кинотеатрами) [Электронный ресурс]. – Режим доступа: https://www.tadviser.ru/index.php/Продукт:TMS\_  
   (система\_управления\_кинотеатрами) – Дата доступа: 08.03.2024.
7. Бизнес-процесс [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/business-process– Дата доступа: 09.03.2024.
8. Автоматизация бизнес-процессов [Электронный ресурс]. – Режим доступа: https://www.unisender.com/ru/glossary/chto-takoe-avtomatizacija-business/#anchor-1 – Дата доступа: 09.03.2024.
9. LiveBusiness [Электронный ресурс]. – Режим доступа: https://www.livebusiness.ru/tool/2081/ – Дата доступа: 08.03.2024.
10. TAdviser [Электронный ресурс]. – Режим доступа: https://www.tadviser.ru/index.php/Продукт:TMS\_(система\_управления\_  
    кинотеатрами) – Дата доступа: 08.03.2024.
11. Intellij IDEA [Электронный ресурс]. – Режим доступа: https://www.jetbrains.com/idea/ – Дата доступа: 11.09.2024.
12. Spring boot [Электронный ресурс]. – Режим доступа: https://spring.io/projects/spring-boot – Дата доступа: 11.09.2024.
13. MySQL [Электронный ресурс]. – Режим доступа: https://www.nic.ru/help/chto-takoe-mysql\_8510.html – Дата доступа: 11.09.2024.
14. Интерфейс [Электронный ресурс]. – Режим доступа: https://blog.skillfactory.ru/glossary/interface/ – Дата доступа: 12.12.2023.
15. GUI [Электронный ресурс]. – Режим доступа: https://blog.skillfactory.ru/glossary/gui/ – Дата доступа: 12.12.2023.
16. Принципы проектирования интерфейсов [Электронный ресурс]. – Режим доступа: https://skillbox.ru/media/design/proektirovanie-interfeysa-8-printsipov/ – Дата доступа: 02.12.2023.
17. CSS [Электронный ресурс]. – Режим доступа: https://www.w3schools.com/w3css/ – Дата доступа: 06.10.2023.
18. Кинопоиск API [Электронный ресурс]. – Режим доступа: https://api.kinopoisk.dev/documentation – Дата доступа: 06.10.2023.
19. Разработка требований к программному обеспечению / К. Вигерс, Дж. Битти. – М. : Издательство «Русская редакция»; СПб. : БХВ-Петербург, 2014. – 736 с.