МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки / специальности)

Бизнес-информатика
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Проект автоматизации билетной кассы Новосибирского государственного академического театра оперы и балета»

Обучающийся Е.С. Васютин
(Инициалы Фамилия) (личная подпись)

Руководитель к.т.н., Н.В. Хрипунов
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

Выпускная квалификационная работа посвящена автоматизации билетной кассы Новосибирского государственного академического театра оперы и балета. Работа состоит из 68 страниц печатного текста, 25 рисунков, 3 таблиц, 27 источников информации и 1 приложения.

Объектом исследования в работе будет Новосибирский государственный академический театр оперы и балета.

Предметом исследования является билетная касса академического театра.

Целью выпускной квалификационной работы является автоматизация билетной кассы Новосибирского государственного академического театра оперы и балета путем разработки информационной системы.

Работы выполнена в трех главах.

Анализ деятельности академического театра оперы и балета, работа билетной кассы и выявление мест, которые можно улучшить описаны в первой главе выпускной работы. Также проведено моделирование бизнес-процессов, которые осуществляются в билетной кассе театра.

Второй этап работы описывает проведенное концептуальное проектирование информационной системы билетной кассы. Для дальнейшей разработки информационной системы была изучена информация, которая поступает для обработки в билетной кассе и что выходит из процесса в качестве исходящей информации.

В третьей главе работы показан разработанный контрольный пример автоматизированной информационной системы билетной кассы. Показаны основные формы, с которыми будут работать кассиры и информация, которая необходима для обслуживания посетителей билетной кассы театра. Для получения данных о эффективности проекта были проведены расчеты.

Заключение показывает какие задачи были выполнены в ходе выполнения выпускной квалификационной работы.

Оглавление

Введение

Выпускная квалификационная работа направлена на изучение деятельности академического театра оперы и балета, а также автоматизации процесса, который нуждается в улучшении обслуживания посетителей в билетной кассе.

Автоматизация позволит сотрудникам билетной кассы повысить качество обслуживания посетителей, сократить время на подбор мест и продажу билетов на представления, которые предлагает театр. Для повышения конкурентоспособности театра и в силу высоких требований клиентов появилась необходимость автоматизации.

Хранение информации о билетах, представлениях, свободных местах будет храниться в одном хранилище данных, что позволит разным кассирам одновременно обращаться к данным и вести параллельную работу с одной базой данных.

Разработанная информационная система для билетной кассы позволит снизит количество ошибок, допускаемых кассирами при продаже билетов, и увеличит скорость обслуживания посетителей театра.

Объектом исследования в работе будет Новосибирский государственный академический театр оперы и балета.

Предметом исследования является билетная касса академического театра.

Целью выпускной квалификационной работы является автоматизация билетной кассы Новосибирского государственного академического театра оперы и балета путем разработки информационной системы.

Задачи выпускной квалификационной работы:

– провести исследование деятельности Новосибирского государственного академического театра оперы и балета;
– идентифицировать процессы билетной кассы, требующие автоматизации;

– провести концептуальное проектирование информационной системы, включающее определение функциональных требований, структуры базы данных и взаимодействия пользователей с системой;
– реализовать информационную систему для билетной кассы Новосибирского государственного академического театра оперы и балета;
– рассчитать экономическую эффективность разработки информационной системы.

В ходе работы применялись различные программы для проектирования и моделирования, для проектирования организационной структуры был использован бесплатный сервис для моделирования draw.io, проектирование диаграмм осуществлялась с помощью программы AllFusion ERwin Data Modeler.

В работе имеется три главы описывающих весь ход работы.

Анализ деятельности академического театра оперы и балета, работа билетной кассы и выявление мест, которые можно улучшить описаны в первой главе выпускной работы. Также проведено моделирование бизнес-процессов, которые осуществляются в билетной кассе театра.

Второй этап работы описывает проведенное концептуальное проектирование информационной системы билетной кассы. Для дальнейшей разработки информационной системы была изучена информация, которая поступает для обработки в билетной кассе и что выходит из процесса в качестве исходящей информации.

В третьей главе работы показан разработанный контрольный пример автоматизированной информационной системы билетной кассы. Показаны основные формы, с которыми будут работать кассиры и информация, которая необходима для обслуживания посетителей билетной кассы театра. Для получения данных о эффективности проекта были проведены расчеты.

Заключение показывает какие задачи были выполнены в ходе выполнения выпускной квалификационной работы.

Глава 1 Функциональное моделирование деятельности театра оперы и балета

1.1 Обзор предметной области

«Открытие Новосибирского театра оперы и балета состоялось 12 мая 1945 года — была представлена опера Глинки «Иван Сусанин». Ранее, в военные годы, в незавершенном театральном здании хранились эвакуированные фонды Третьяковской галереи, музея изобразительных искусств им. Пушкина, дворцов-музеев Павловска и Царского Села, а также бесценные музыкальные инструменты работы Страдивари, Гварнери, Амати из государственной коллекции. С сентября 1941 года в Новосибирске размещался эвакуированный из осажденного города симфонический оркестр Ленинградской филармонии. Оркестр, руководимый Евгением Мравинским, во время эвакуации дал более 500 концертов, которые посетило около 400 тысяч слушателей; состоялось также 240 концертов по радио. 7 ноября 1942 года в присутствии автора была исполнена первая часть Седьмой симфонии Дмитрия Шостаковича, которая была создана и прозвучала в блокадном Ленинграде» [4].

«Миссия Новосибирского театра оперы и балета сегодня — быть открытым пространством и центром притяжения всех творческих сил. Театр должен стать «безбарьерной средой» и точкой роста для всех видов искусств, а его гигантский купол будет защищать любые инициативы. Мощный культурный кластер, где плодотворно работают филармония, консерватория, специальная музыкальная школа, хореографический колледж, нуждается в надежной несущей конструкции — и эту роль берет на себя театр, чтобы укрепить позиции Новосибирска как лидера в евразийском культурном пространстве» [4].

«НОВАТ предоставляет зрителям возможность прикоснуться к истории театра. Предлагает обзорную и несколько тематических экскурсий, которые

откроют двери в удивительный и волнующий театральный мир! Экскурсионные маршруты уникальны и будут интересны как искушенным поклонникам оперного и балетного искусства, так и начинающим театралам» [4].

Организационная структура управления представляет собой линейную иерархию управления персоналом. Структура Новосибирского театра оперы и балета представлена на рисунке 1.

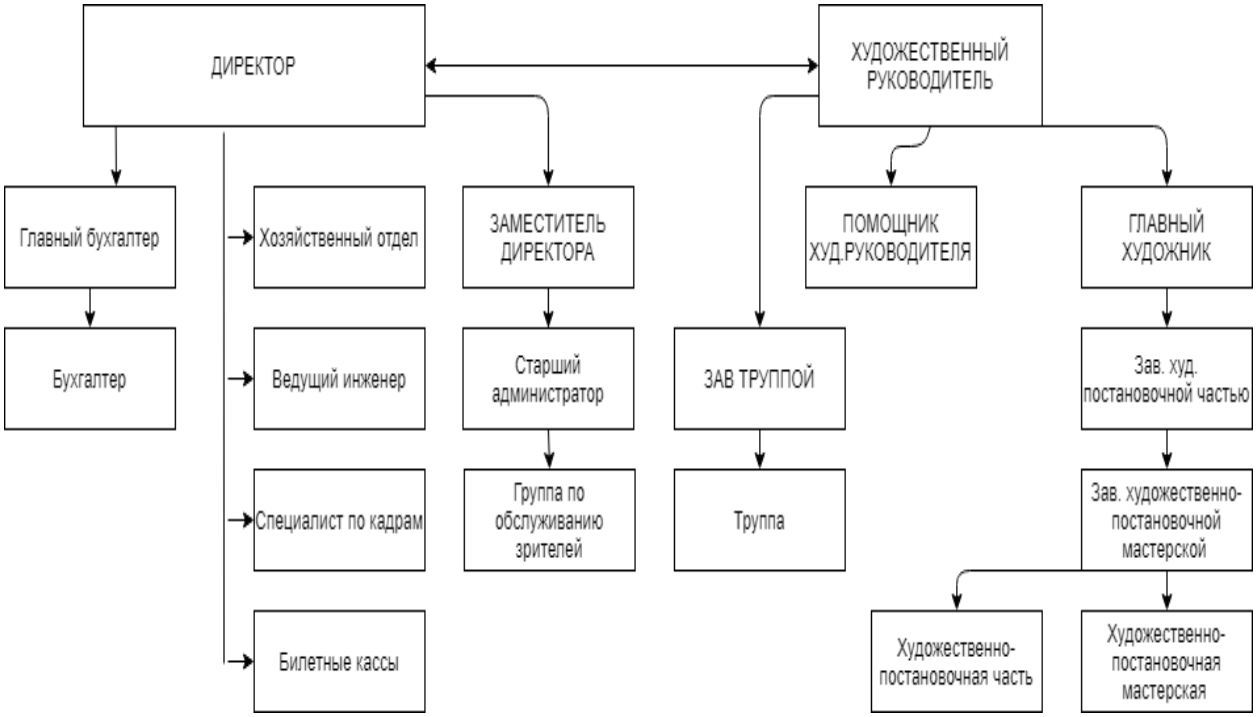

Рисунок 1 – Организационная структура Новосибирского театра оперы и балета

На структуре показаны руководители их помощники и группы, которые занимаются постановочной частью мероприятий, художественной постановкой и работа с труппой.

Также показаны такие специалисты как по кадрами, ведущий инженер, бухгалтера.

Продажа билетов в театр происходит в билетных кассах. Основная цель кассы – обеспечение информации о наличии билетов на различные постановки театра и балета в режиме реального времени.

Для того чтобы информация была актуальной необходимо обновлять ее и вести отметку проданных и забронированных билетов, атак как в театре не одна билетная касса то этот процесс затруднен и может вызвать ошибки при продаже билетов, особенно когда поток покупателей превышает средний.

Как и все современные компании и учреждения Новосибирский театр оперы и балета внедряет новейшие информационные технологии для оптимизации своей деятельности и подразделений, где это становится необходимостью.

Объектом рассмотрения в данной организационной схеме являются билетные кассы.

Субъектом управления в билетных кассах является непосредственно старший кассир и кассиры, которые ведут весь процесс продажи билетов от начала и до конца.

Основным технико-экономическими показателями процесса продажи билетов в театре являются извлечение прибыли от выполняемых функций в кассе.

1.2 Концептуальное проектирование предметной области

Проектирование информационной системы начинается с изучения бизнес-процесса, который в дальнейшем будет автоматизирован [2]. Изначально проводится концептуальное проектирование бизнес-процесса. Модели бизнес-процесса позволят определить информацию, которая поступает, обрабатывается и получается на выходе из процесса.

Данное проектирование можно провести различными инструментами и в различных нотациях, которые позволяют достичь целей проектирования.

Опишем самые распространённые и наиболее применяемое программные продукты для данного моделирования: BPWin 7.0, ARIS, Microsoft Visio [5].

Программный продукт BPWin 7.0 является самой простой и из первых кто был создан для моделирования процессов. В программе осуществляется несколько нотаций, которые позволяют моделировать бизнес-процессы предприятий на разные подпроцессы и даже действия.

ARIS Express современная и очень красочная программа, которая позволяет создавать диаграммы различных бизнес-процессов и не только в цветном формате и экспортировать их как в рисунки так и в другие файлы. С помощью ARIS можно быстро создать диаграмму организационной структуры, бизнес-процессов в нотации EPC, BPMN и других полезных.

Microsoft Visio это программный продукт от Microsoft. Инструменты программы позволяют выполнять практически все виды моделирования, не только бизнес-процессов, но и других необходимых процессов и модулей разрабатываемых продуктов.

Для моделирования выбранного бизнес-процесса наиболее подходящим будет нотация IDEF0, которая позволит рассмотреть процесс как абстрактно, так опуститься более глубже, где можно провести анализ и разделить процесс на более мелки подпроцессы.

Для анализа процесса построим диаграммы в нотации IDEF0, которая наглядно показывает какие процессы выполняется при обслуживании посетителей театра в билетной кассе.

Главная диаграмма покажет, как выглядит общий процесс при обслуживании посетителей в билетной кассе театра, и какая информация при этом задействована.

Рисунок 2 демонстрирует главную диаграмму первого уровня процесса обслуживания посетителей в билетной кассе.

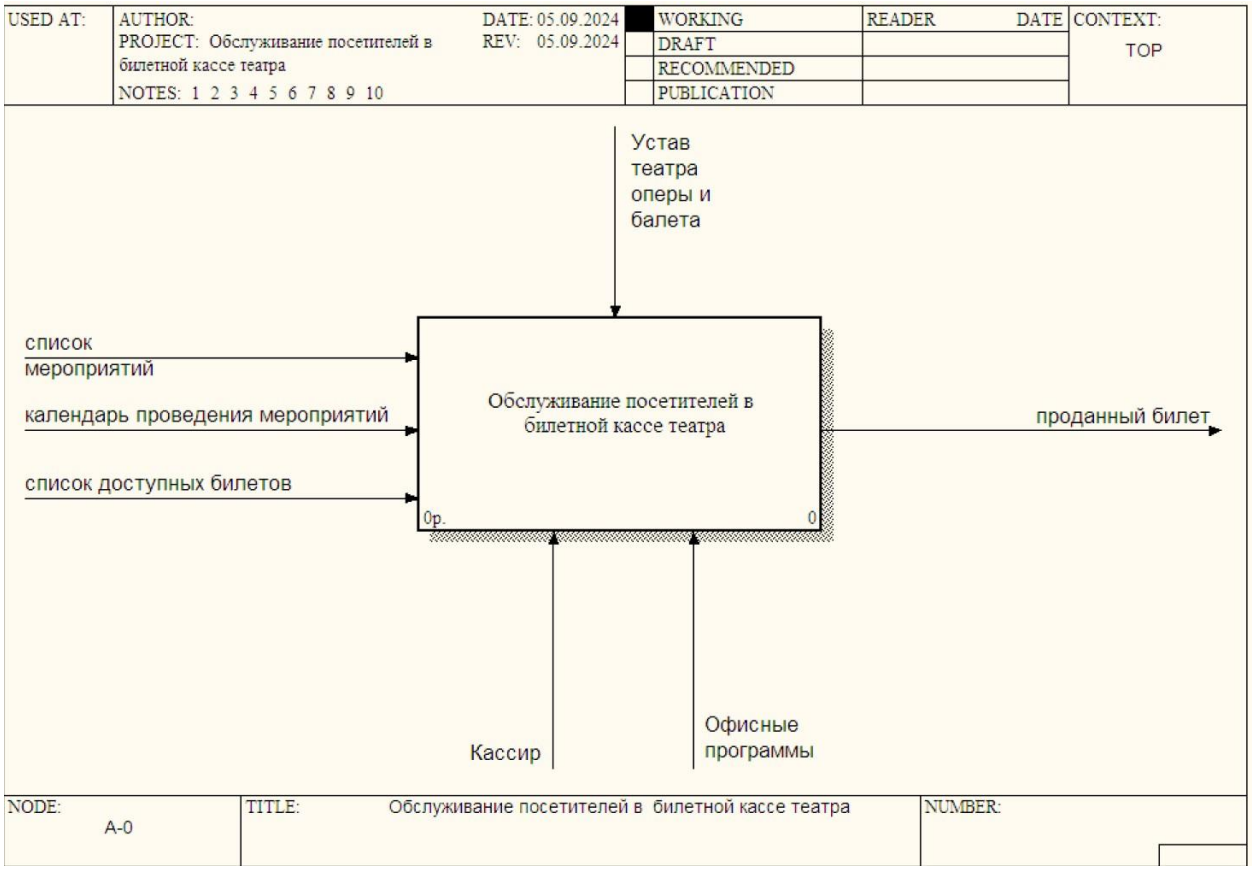

Рисунок 2 – Диаграмма первого уровня обслуживания посетителей в билетной кассе театра

Обслуживание посетителей имеет входящие данные в процесс: список мероприятий, календарь проведения мероприятий, список доступных билетов.

На основе этих данных посетителю предоставляется информация о проведении и наличии билетов на мероприятия.

Выполним декомпозицию процесса, разделив процесс обслуживания посетителей театра в билетной кассе:

– предоставление информации о стоимости мероприятия;

– выбор нужной даты;

– выбор места в зале;

– оформление продажи билета.

На рисунке 3 представлена диаграмма декомпозиции процесса обслуживания посетителей театра в билетной кассе.

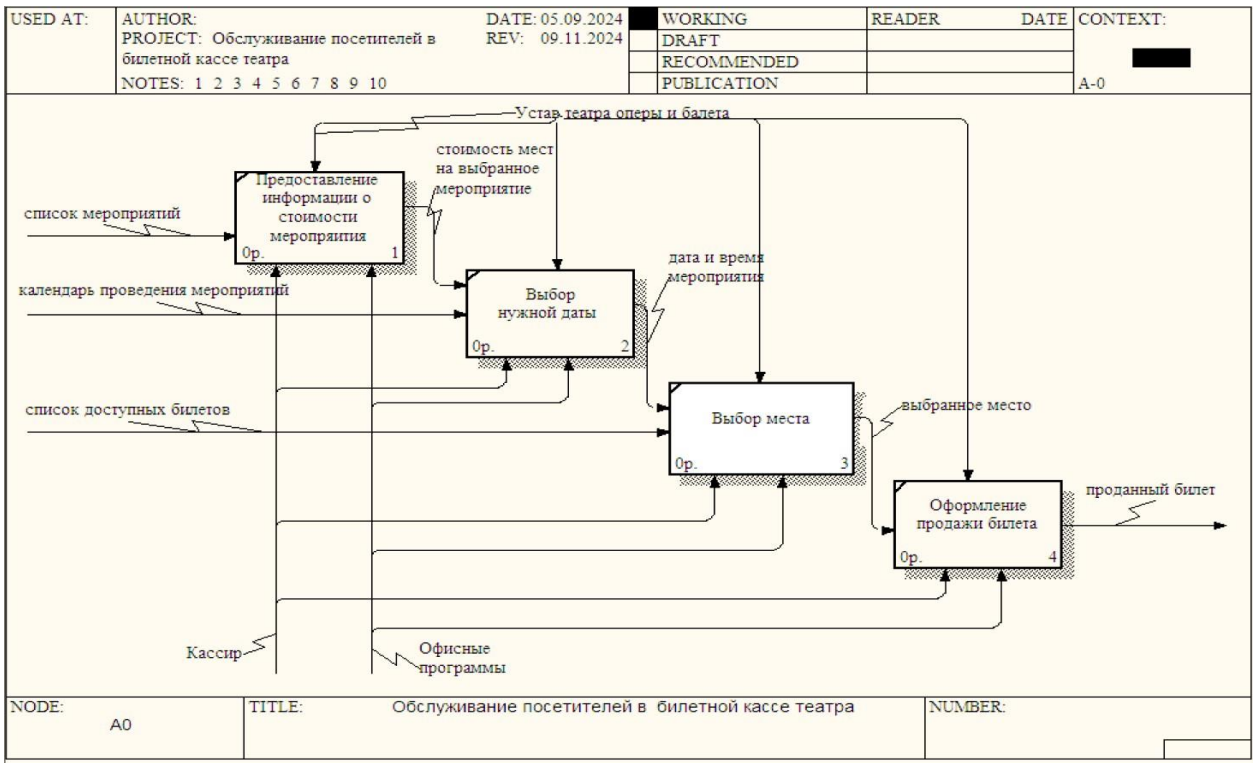

Рисунок 3 – Диаграмма декомпозиции обслуживания посетителей в билетной кассе театра

Все процессы в билетной кассе выполняются с помощью офисных программ. Кассир фиксирует все проданные билеты в офисной программе, что не гарантирует допущения ошибок, особенно в случае, если билет захотят вернуть.

Обслуживание посетителя в билетной кассе начинается с информирования о стоимости мероприятия, на которое посетитель хочет пойти. Далее происходит выбор даты мероприятия и выбор места, что сказывает на стоимости билета. После выбора всех параметров посетитель может купить билет, а кассир оформить продажу.

Для ускорения завершения процесса продажи билетов в билетной кассе и снижения ошибок при продаже билетов необходимо оптимизировать процесс.

На рисунке 4 показана главная диаграмма первого уровня обслуживания посетителей в билетной кассе театра.

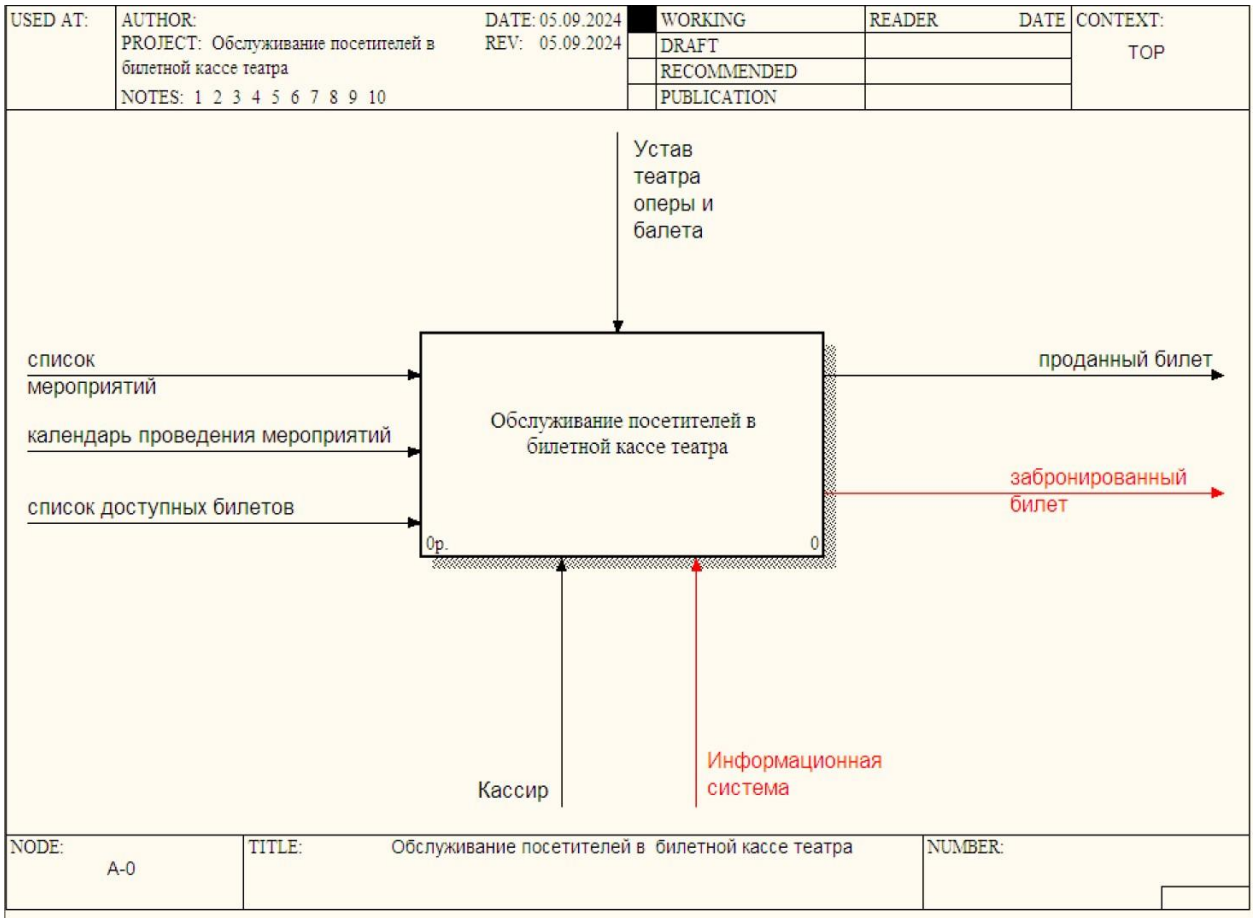

Рисунок 4 – Контекстная диаграмма первого уровня обслуживания посетителей в билетной кассе театра

На рисунке 4 видно, что в процессе появилась информационная система, в которой и будет осуществляться процесс бронирования и продажи билетов в кассе театра.

Покажем диаграмму декомпозиции этого процесса после автоматизации, рисунок 5.

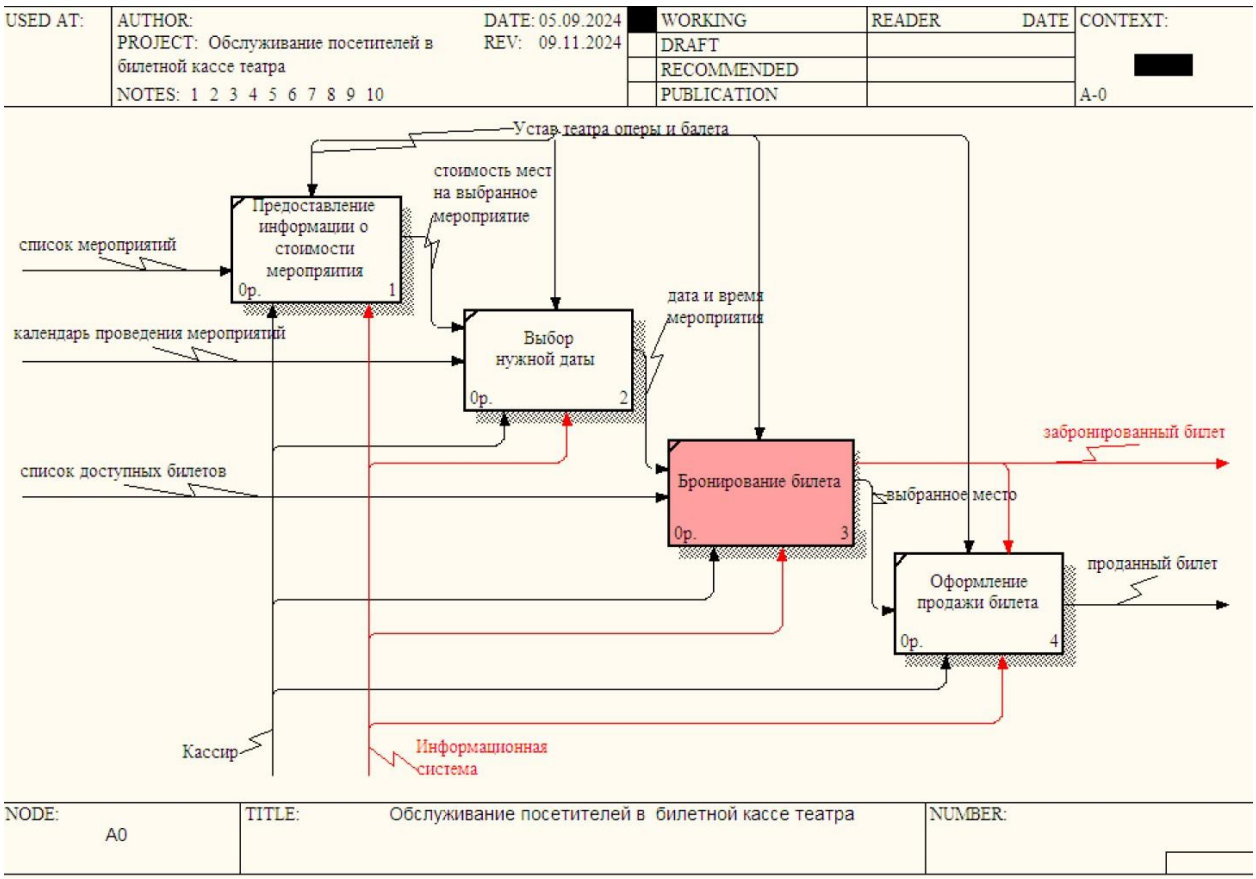

Рисунок 5 – Диаграмма декомпозиции обслуживания посетителей в билетной кассе театра после автоматизации

На диаграмме декомпозиции показан процесс обслуживания посетителей в билетной кассе театра с учетом автоматизации. После того как посетитель обратился в кассу театра кассир информирует его о проводимых мероприятиях и датах проведения, также помогает выбрать дату, на основе данной информации посетитель может купить билет или забронировать его , после чего кассир сможет отслеживать в режиме реального времени сколько доступных билетов осталось в кассе на выбранное мероприятие.

Такой процесс позволит снизить время обслуживания посетителя в билетной кассе театра и что увеличит лояльность гостей.

Также появится возможность у посетителя получить билет в электронном виде, где будут прописаны все данные о мероприятии и места, которые были выкуплены.

Пользователи системы получат возможность контролировать процесс продажи билетов и оповещать посетителя при запросе на покупку или бронирование билета.

Информация система билетной кассы должна выполнять ряд функций:

– выбор спектакля;

– выбор даты проведения спектакля;

– выбор зала;

– выбор места;

– оплата билетов.

Нефункциональные требования:

– быстрое взаимодействие модулей системы;

– наличие навигации в меню пользователя;

– отклик системы в течении 1 сек.

При выполнении таких функций процессы будут выполняться значительно быстрее в билетной кассе театра.

1.3 Анализ существующих разработок

Сравним три информационные системы, которые выполняют функции билетной кассы. Рассмотрим российский рынок программных продуктов.

AppEvent.

Возможности программного продукта:

– статистика по количеству проданных билетов;

– возможность отметки посещения мероприятия;

– учет базы клиентов;

– создание мероприятий с возможностью создания расписания;

– предусмотрены напоминания клиентам;

– автоматическая отправка билетов на мероприятия;

– интеграции с сервисами Яндекс Афиша и Kassir.ru;

– уведомления о мероприятии.

InTickets.

Основные характеристики:

– есть программа лояльности для зрителей;

– интеграция системы с клиентской CRM;

– возможность вести клиентскую базу;

– интеграция с 1С;

– возможность продажи виртуальных билетов;

– программа лояльности.

Radario.

Функции системы:

– данные о покупателях;

– статистика запущенных с использованием платформы кампаний;

– позволяет сегментировать покупателей по многочисленным параметрам;

– встроенный конструктор рассылок со всеми необходимыми шаблонами.

Сравнительный анализ трех информационных систем показал, что целесообразно будет приложить усилия для собственной разработки, которая будет отвечать требованиям заказчика и содержать только необходимый функционал.

В билетной кассе театра не требуется многофункциональная система с мобильным приложением для предоставления информации о доступных местах и продажи билетов достаточно будет автоматизировать процесс путем внедрения информационной системы.

1.4 Постановка задачи на разработку системы билетной кассы театра

Информационная система создается для автоматизации билетной кассы в театре. Система будет основным программным продуктов билетной кассы с помощью, которой будет осуществляться продаж билетов в театр.

Хранение информации в одной базе данных позволит кассирам кассы быстро и эффективно находить информацию о доступности билетов и стоимости.

Целью разработки будет упрощение выполняемых процессов и возможность обслуживания посетителей театра в билетной кассе более эффективно [3].

Система позволит работать одновременно нескольким кассирам. Пользователь, который будет вести продажу билетов получит возможность видеть все спектакли, которые открыты для продажи, даты выбора спектакля, посмотреть свободные места и стоимость билетов, которые будут видны при выборе места, и осуществлять продажи.

Система будет выполнять такие процессы как:

– учет проданных билетов;

– учет доступных билетов;

– продажу билетов;

– выбор спектакля и даты.

Меню для пользователя будет содержать навигацию и фон, который будет помогать сотрудникам кассы комфортно работать.

Доступ к программе будут иметь сотрудники билетной кассы и администраторы системы.

Реализация автоматизированной информационной системы должна быть произведена с помощью языка высокого уровня.

Меню пользователя должно выполнятся с помощью простых элементов применение, которых не потребует дополнительных знаний пользователя.

Архитектура приложения должна позволять масштабировать приложения на основе потребностей Новосибирского театра в том числе и билетной кассы.

Вывод по первой главе.

В первой главе были изучены деятельность Новосибирского театра оперы и балета и бизнес-процессы, которые осуществляются в билетной кассе при обслуживании посетителей театра. Бизнес-процессы билетной кассы включают подбор билета на выбранный спектакль, подбор даты посещения театра и предоставление информации по ценам на основе выбранного места.

На основе полученных данных провели моделирование процессов билетной кассы в нотации IDEF0. На разных уровнях было показано как происходит процесс обслуживания в билетной кассе посетителей и осуществляется продажа билетов. Диаграммы были проанализированы и выделены процессы, которые необходимо улучшить с помощью автоматизации.

Глава 2 Логическое проектирование автоматизированной информационной системы билетной кассы

2.1 Логическая модель информационной системы билетной кассы

Моделирование системы на логическом уровне позволит правильно спроектировать систему, и программисты реализуют в системе только те функции, которые необходимы кассиру в билетной кассе.

Логическое моделирование осуществляется с помощью специализированных программ, которые также представлены на рынке. Рассмотрим некоторые из них. Чтобы представлять функционал и возможности, которыми можно будет пользоваться при моделировании.

На рынке представлены такие программы как StarUML, Rational Rose, Visio, которые заявлены как самые распространенные программные продукты для логического моделирования.

Все программы, которые рассматривались имеют схожие инструменты и разные интерфейсы, которые значительно влияют на использование программ.

С помощью программ можно провести логическое моделирование информационной системы, где будут показаны модули, их взаимодействие между собой и пользователем системы.

Диаграмма вариантов использования покажет, как будет происходить взаимодействие между кассиром билетной кассы и все функции системы, которые помогут автоматизировать процесс в билетной кассе. Кассир будет непосредственным участником процесса обслуживания посетителей в билетной кассе театра, а администратор системы будет осуществлять добавление информации о спектаклях и билетах в систему.

На рисунке 6 показана диаграмма вариантов использования с двумя акторами.

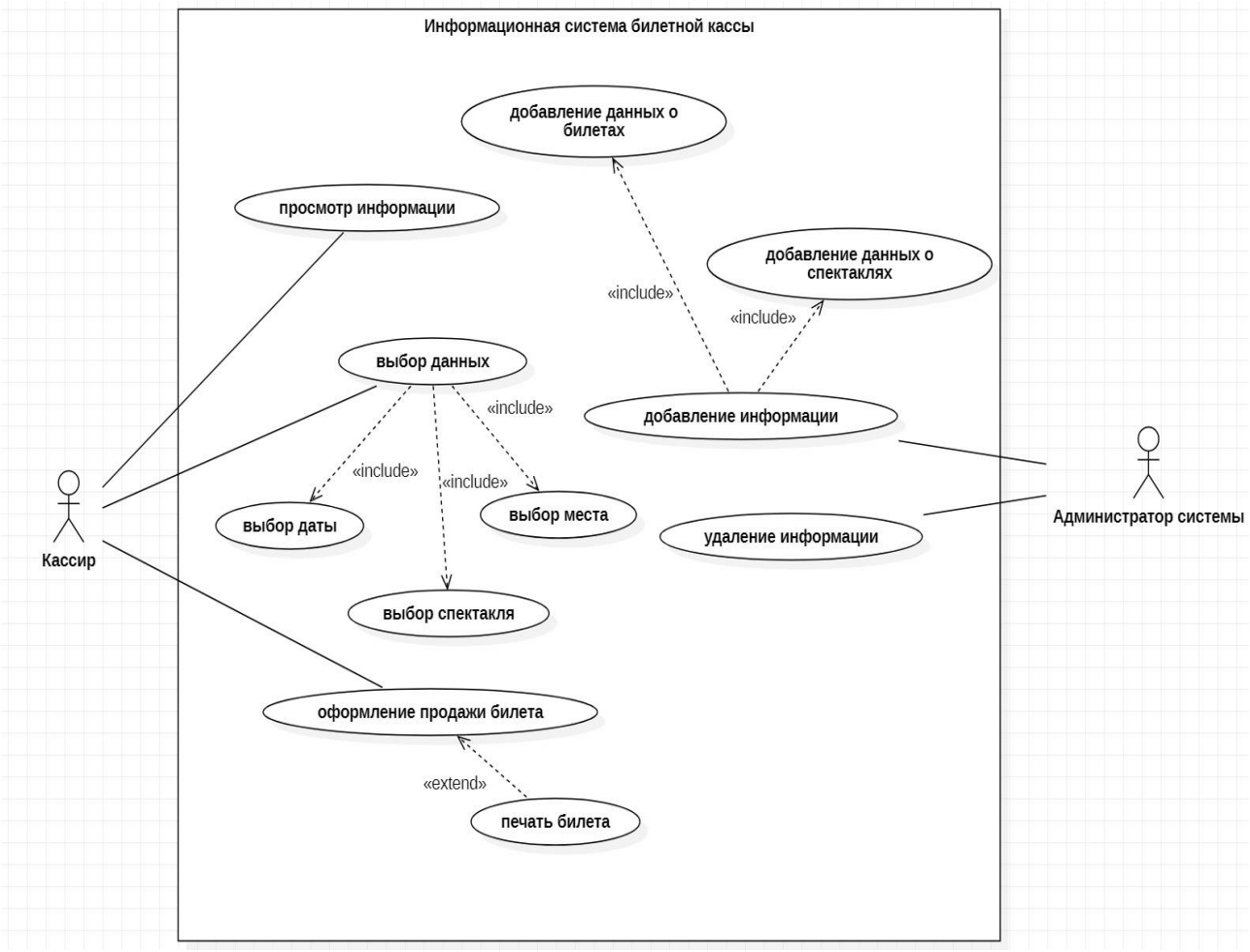

Рисунок 6 – Диаграмма вариантов использования

Диаграмма вариантов использования демонстрирует выполнение функций двух акторов информационной системы.

Кассир билетной кассы театра выполняет такие функции:

– просмотр информации;

– выбор данных;

– оформление продажи билета;

– печать билета;

– выбор даты;

– выбор места;

– выбор спектакля.

Администратор системы выполняет функции:

– удаление информации;

– добавление информации;

– добавление данных о спектаклях;

– добавление данных о билетах.

Спроектируем диаграмму деятельности [27], где показана последовательность работы кассира в информационной системе билетной кассы. Диаграмма показана на рисунке 7.

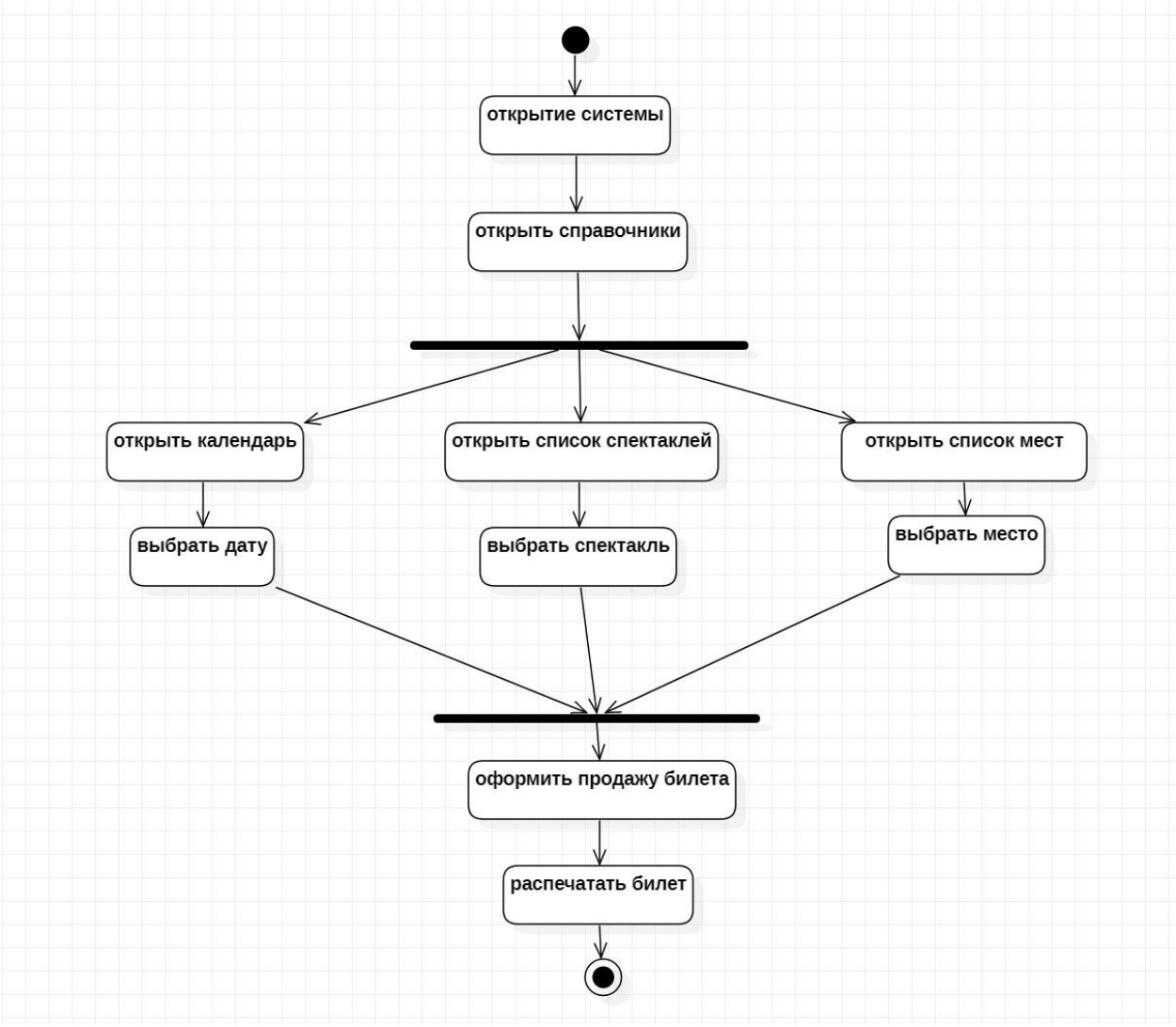

Рисунок 7 – Диаграмма деятельности кассира в информационной системе

Теперь рассмотрим диаграмму классов обработки данных [24] и методов системы при обслуживании посетителей в билетной кассе театра.

Диаграмма классов показана на рисунке 8.

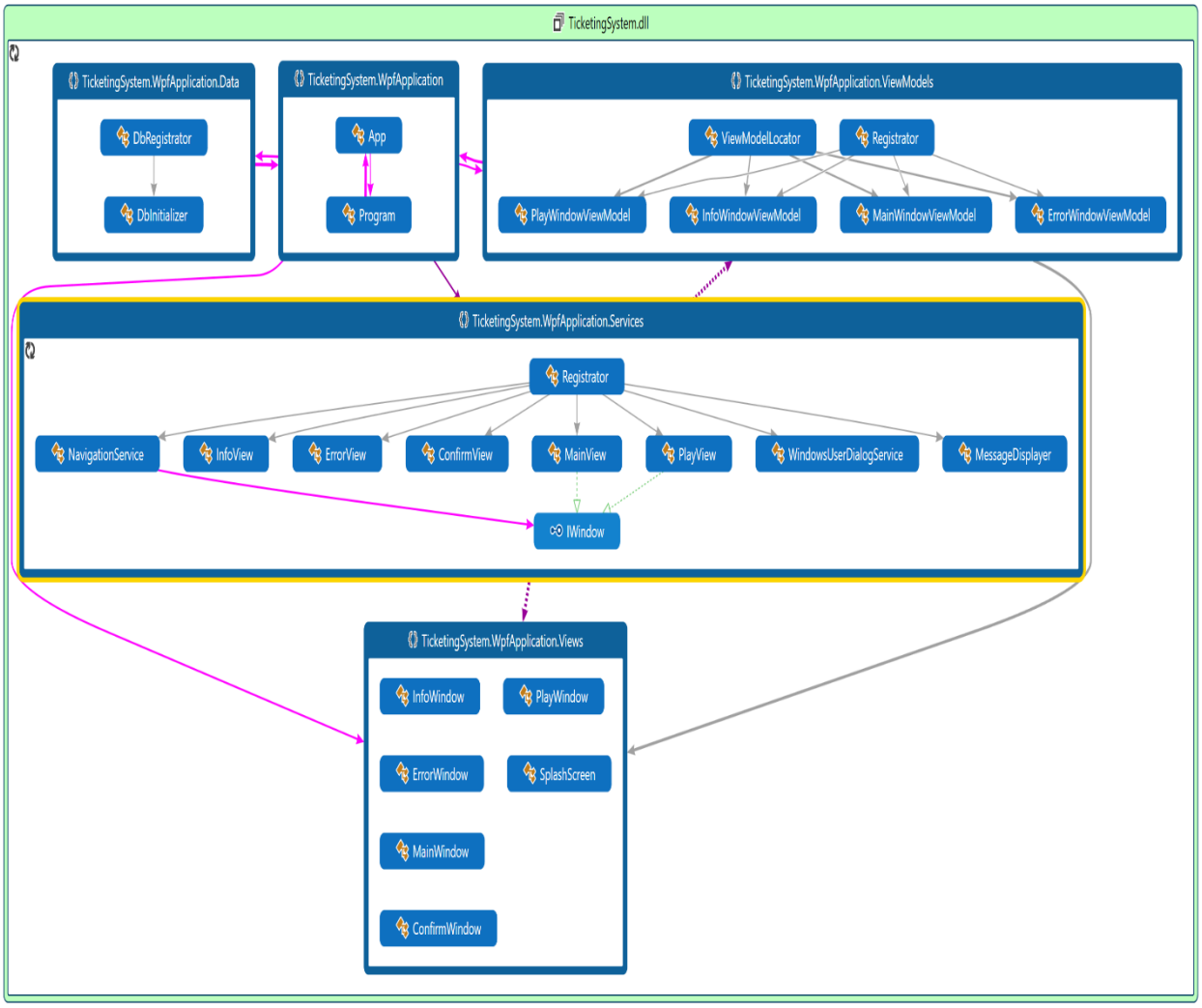

Рисунок 8 – Диаграмма классов информационной системы

При разработке информационной системы потребуются классы для обработки данных билетов и их продаж. Для этого будут предусмотрены такие классы как:

– TicketingSystem.WPFApplication.Data;

– TicketingSystem.WPFApplication.

Для отображения представления информации предусмотрены классы такие как:

– TicketingSystem.WPFApplication.ViewModels;

– TicketingSystem.WPFApplication.Services;

– TicketingSystem.WPFApplication.Views.

С помощью данных классов будет обрабатываться вся информация, которая касается билетов в билетной кассе и с помощью методов данные будут сгенерированы и выводиться пользователю на консоль.

2.2 Проектирование базы данных системы билетной кассы театра

Хранение информации для билетной кассы театра подразумевает определенные данные, которые нужно выделить [25].

Объектами хранения данных будет информация о спектаклях, залах, местах и билетах.

База данных должна быть реляционной, чтобы обеспечивать получение информации более удобным способом и иметь связи между таблицами.

Таблица Залы должна содержать данные о всех залах, где будут проходить спектакли. Таблица должна информировать о всех подробностях.

Таблица Места будет содержать информацию о местах в залах, данные о бронировании и категории места.

Таблица Спектакли предоставит информацию о всех спектаклях и залах, где будут проводиться.

Таблица Билеты предоставит информацию о всех билетах и их стоимости.

Для создания логической модели необходимо определить сущности, которые будут хранить данные информационной системы билетной кассы.

Для владения информацией о наличии спектаклей и билетов в кассе нужно хранить эти данные в базе данных.

Основные таблицы, которые потребуются для хранения информации:

– Залы (Halls),

– Места (Seats),

– Билеты (Ticket),

– Спектакли (Plays).

На рисунке 9 показана логическая модель данных.

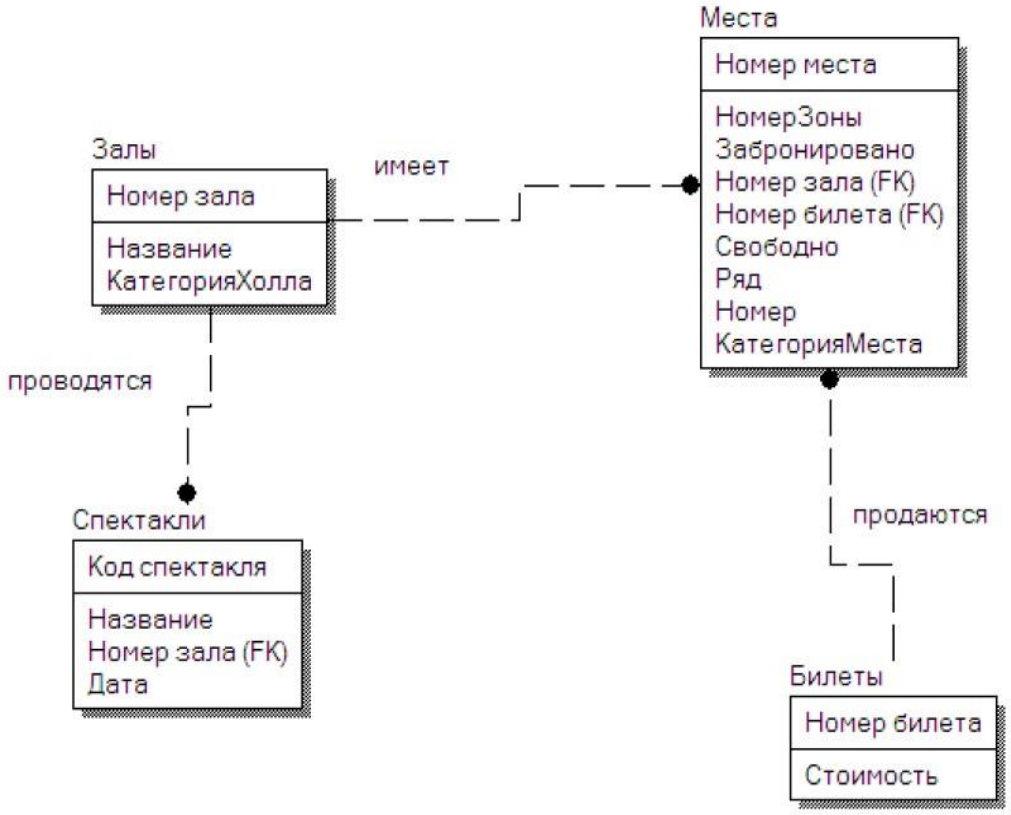

Рисунок 9 — Логическая модель данных

Информационная система билетной кассы для предоставления данных посетителям театра оперы и балета должна иметь хранилище, де будут храниться вся информация для билетной кассы театра оперы и балета.

Опишем связи, которые будут присутствовать между таблицами в базе данных, что обеспечит получении данных по средствам запросов.

В одном зале могут проводить несколько спектаклей в разное время, однако один спектакль не может ставиться сразу в разных залах.

Как правило в одном зале имеется множество мест в разных зонах, и много мест располагается в одном зале.

Билет оформляется на места в зале, на разные места не может быть продан один билет.

Представим реквизиты сущностей в таблице 1.

Таблица 1 — Описание таблиц и типов данных

Имя сущности	Описание	Тип
Билеты	Номер отдела	Числовой
	Стоимость	Денежный
Залы	Номер зала	Числовой
	Название	Текст
	Категория зала	Числовой
Спектакль	Код спектакля	Числовой
	Название	255 символов
	Дата	Дата
	Номер зала	Числовой
Места	Номер места	Числовой
	Номер зоны	Числовой
	Забронировано	Логический
	Номер зала	Числовой
	Номер билета	Числовой
	Свободно	Логический
	Ряд	Числовой
	Номер	Числовой
	Категория места	Числовой

В таблице 1 описаны все сущности, которые будут содержаться в базе данных, также видно, что все таблицы будут связаны между собой ключами, которые будут способствовать передачи данных.

Рассмотрим правильность связей между таблицами.

После создания моделей для базы данных можно приступать к физическому проектированию базы данных и реализации информационной системы [7].

2.3 Выбор архитектуры информационной системы

Все данные и документы, поступающие в билетную кассу, должны храниться в одном хранилище данных для общего доступа всем кассирам театра. Объем хранилища должен позволять хранение информации при накоплении ее в случае большого потока и информации или расширения функционала кассы театра в будущем.

Вся информация, которая будет храниться в базе данных будет получена пользователем по средствам запросов, которые также будут созданы к информационной системе.

Информация система должна быть написана на любом языке высокого уровня, который будет хорошо поддерживать инструменты Windows и интегрироваться с другими система.

Архитектура приложения должна быть гибкой и доступной для вносимых изменений при расширении системы [8].

Рассмотрим несколько распространенных архитектур приложения, чтобы определить какая из них подойдет для информационной системы билетной кассы театра.

Монолитная архитектура характеризуется единым монолитным кодом, который развертывается единым приложением. Архитектура имеет достоинства такие как простота разработки и тестирование [23].

Микросервисная архитектура подразумевает что приложение состоит из нескольких автономных сервисов, каждый из которых отвечает за определенную бизнес-функциональность.

Преимущества данной архитектуры заключаются в высокой гибкости и масштабируемости. Недостатки тоже есть, возможные проблемы

согласованности данных между сервисами, а также дополнительные затраты на управление инфраструктурой.

Клиент-серверная архитектура имеет клиента и сервер.

Функциональность клиент-серверной архитектуры позволяет обрабатывать запросы от клиентов, управлять ресурсами и выполнять операции, необходимые для клиентов [10].

На рисунке 10 показана клиент-серверная архитектура.

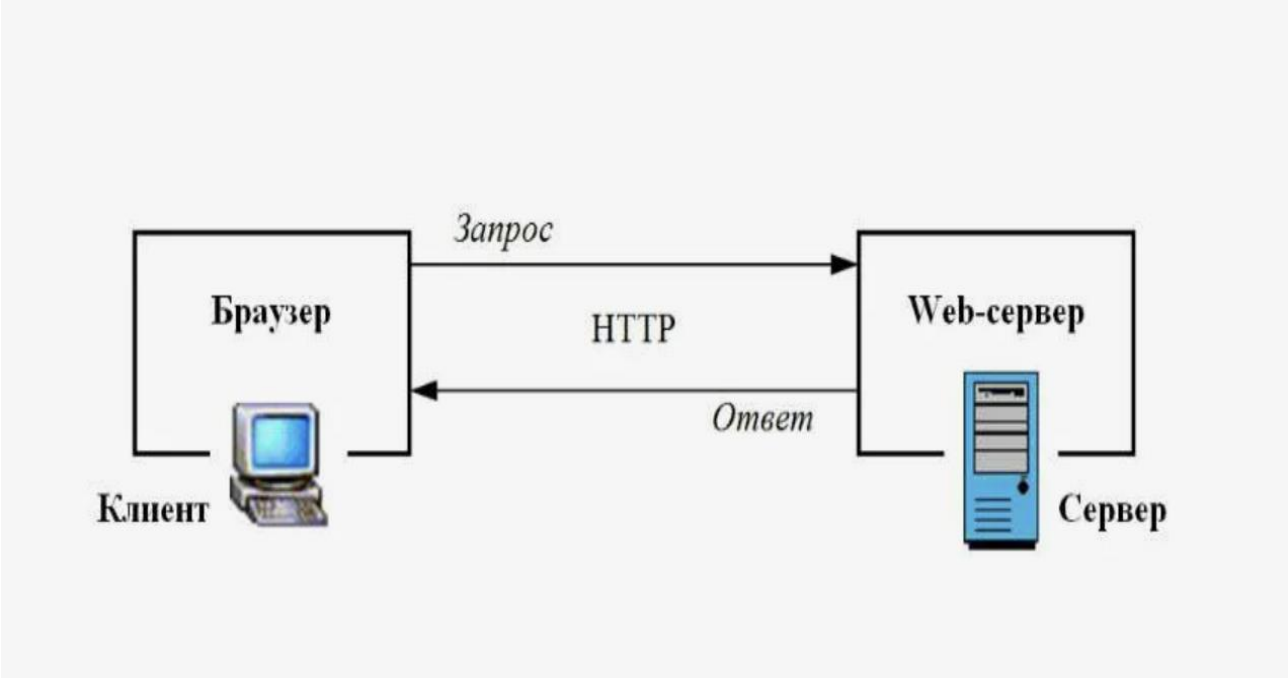

Рисунок 10 — Клиент-серверная архитектура

Клиент-серверная архитектура самая подходящая для реализации информационной системы билетной кассы Новосибирского театра оперы и балета.

Все запросы от клиентов и пользователей системы будут поступать на сервер театра, там все данные будут обработаны и возвращаться информация будет на интерфейс пользователя в виде представления данных [26].

Также такая архитектура позволяет разделить модули на несколько уровней, с помощью которых будет осуществляться реализация все методов и классов в системе.

Выводы по второй главе.

Во второй главе проведено проектирование информационной системы билетной кассы. Проектирование было проведено на основе ранее проведенного концептуального проектирования, где были определены слабые места процессов, осуществляемых в билетной кассе театра.

Проектирование информационной системы включает также рассмотрение и выбор архитектуры, на основе которой и будет разработана система. Для лучшего понимания как устроена информационная система билетной кассы были смоделированы ряд диаграмм, такие как диаграмма вариантов использования, диаграмма деятельности. Все эти диаграммы показывают, как устроена система изнутри и взаимодействие с пользователем на уровне логики. Для хранения всех данных системы было проведено проектирование основных объектов хранилища данных.

Глава 3 Физическое проектирование автоматизированной информационной системы билетной кассы

3.1 Выбор инструментов разработки

Рассмотрим C++, C# и Java, наиболее распространенные.

«C# (произносится «си-шарп») — это современный язык программирования, разработанный компанией Microsoft. Он широко используется для разработки разнообразных приложений, в том числе для создания Windows-приложений, веб-приложений, мобильных приложений под платформу Xamarin, игр на платформе Unity и многих других приложений» [12].

«К 2000 году у Microsoft были готовы промышленные версии новых технологий и решений для обмена сообщениями и данными, а также для создания Internet-приложений. Была выпущена и новая платформа для разработки под новые решения — .NET. В ней объединились сразу несколько языков программирования, что было в новинку для того времени» [12].

Ещё одним новшеством платформы .NET была технология активных серверных страниц ASP.NET (Active Server Page). С её помощью можно было относительно быстро разработать веб-приложения, взаимодействующие с базами данных. Специально для ASP.NET был создан язык программирования C#. Да и сама ASP.NET была полностью написана на нём» [12].

«C++ — чрезвычайно мощный язык, содержащий средства создания эффективных программ практически любого назначения, от низкоуровневых утилит и драйверов до сложных программных комплексов самого различного назначения. Высокая совместимость с языком C, позволяющая использовать весь существующий C-код (код C может быть с минимальными переделками скомпилирован компилятором C++; библиотеки, написанные на C, обычно могут быть вызваны из C++ непосредственно без каких-либо дополнительных

затрат, в том числе и на уровне функций обратного вызова, позволяя библиотекам, написанным на C, вызывать код, написанный на C++)» [13].

«Поддерживаются различные стили и технологии программирования, включая традиционное директивное программирование, ООП, обобщенное программирование, мета программирование (шаблоны, макросы). Имеется возможность работы на низком уровне с памятью, адресами, портами. Возможность создания обобщённых контейнеров и алгоритмов для разных типов данных, их специализация и вычисления на этапе компиляции, используя шаблоны. Кроссплатформенность. Доступны компиляторы для большого количества платформ, на языке C++ разрабатывают программы для самых различных платформ и систем» [13].

Приведем сравнительную таблицу языков программирования

Таблица 2 – Результаты сравнения языков программирования

Основные отличия	C++	C#	Java
1	2	3	4
Понятный синтаксис	-	+	+
Интеграция с базой данных	+	+	+
Скорость обработки данных	+	+	-
Функциональность и гибкость языка	-	+	-
Интеграция с фрэймворками	-	+	+

Для создания системы выберем инструмент разработки, система будет небольшой и создаваться в учебных целях, поэтому создадим пользовательский интерфейс и выберем СУБД, где будет храниться информация по заявкам и клиентам.

Сравниваться по функциональности будут СУБД такие как:

«SQL Server является надежной базой данных для любых целей, может продолжать расширяться по мере наполнения информацией, без заметного уменьшения быстродействия операций с записями в многопользовательском режиме. Пользователи могут быть добавлены путем модернизации оборудования. В последнем тесте поддерживалось до 4600 пользователей базы данных» [15].

«Обеспечивается максимальная безопасность. Ваши данные защищены от несанкционированного доступа за счет интеграции сетевой безопасности с сервером безопасности. Поскольку безопасность на уровне пользователя, пользователи могут иметь ограниченный доступ к записи данных, тем самым защищая их от модификации или поиска, указав доступ на уровне пользовательских привилегией. Кроме того, с данными, хранящимися на отдельном сервере, сервер работает как шлюз, который ограничивает несанкционированный доступ» [15].

«SQL Server обрабатывает запросы от пользователей и только отправляет пользователю результаты запроса. Таким образом, минимальная информация передается по сети. Это улучшает время отклика и устраняет узкие места в сети. Это также позволяет использовать SQL Server в качестве идеальной базы данных для интернет» [15].

Access можно быстро изучить, любой пользователь найдет много информации как в ней работать, поэтому она доступна для пользователей с низкой квалификацией.

«PostgreSQL — это мощная объектно-реляционная система баз данных с открытым исходным кодом, более 35 лет активно разрабатываемая, которая заслужила прочную репутацию за надежность, функциональность и производительность» [6].

«В официальной документации можно найти огромное количество информации, описывающей, как установить и использовать PostgreSQL. Сообщество с открытым исходным кодом предоставляет множество полезных мест, где можно ознакомиться с PostgreSQL, узнать, как это работает, и найти

возможности для карьерного роста. Узнайте больше о том, как взаимодействовать с сообществом» [6].

Ниже представлены несколько функциональных возможностей СУБД:

– вложенные запросы;

– представления;

– ссылочная целостность – внешние ключи.

Однако есть и минусы, как и любой другой системы управления данными. Главным минусом является плохая генерация с языками программирования.

СУБД SQL Server разработана Microsoft и имеет хорошую интеграцию с разными языками высокого уровня. Объем хранения данных, может расширяться по мере наполнения информацией, без заметного уменьшения быстродействия операций с записями в многопользовательском режиме [16].

«Код взаимодействия с базой данных может быть очень громоздким, однако его можно сократить, воспользовавшись Entity Framework. Entity Framework — это решение для работы с базами данных, которое используется в программировании на языках семейства .NET. Оно позволяет взаимодействовать с СУБД с помощью сущностей (entity), а не таблиц. Также код с использованием EF пишется гораздо быстрее. Подключить Entity Framework можно к любому проекту — от Xamarin до ASP.NET» [1].

Пользователь получает ответы от SQL Server путем отправки запроса на сервер и получения данных в ответ. Как правило такие СУБД выбирают для обработки данных в сети интернет [19].

На основе рассмотренных выше систем управления данными можно сделать вывод и выбрать оптимальный вариант для приложения. СУБД Microsoft SQL Server подойдет больше всех, она имеет все доступные средства для разработки приложения. Представим базы данных в сравнительной таблице 3.

Таблица 3 – Результаты сравнения баз данных

Основные отличия	Access	MS SQL	PostgreSQL
1	2	3	4
Доступность	+	+	+
Хранение большого объема данных	-	+	+
Скорость обработки операций с большими данными	-	+	-
Защита от несанкционированного доступа	-	+	+
Простота использования	+	-	-

Сравнительный анализ СУБД показал, что с применением языка программирования C# подходящей будет СУБД Microsoft SQL Server 2022 [17].

3.2 Разработка системы

База данных разрабатывается в системе управления базой данных. После рассмотрения нескольких СУБД была выбрана Microsoft SQL Server, создадим в ней таблицы. Таблицы для системы были определены выше. На рисунке 11 показана модель данных, созданная в СУБД SQL Server 2022 [22].

– Залы (Halls),

– Места (Seats),

– Билеты (Ticket),

– Спектакли (Plays).

Таблицы содержат название атрибута, тип данных и длину допустимую при вводе данных в таблицу [11].

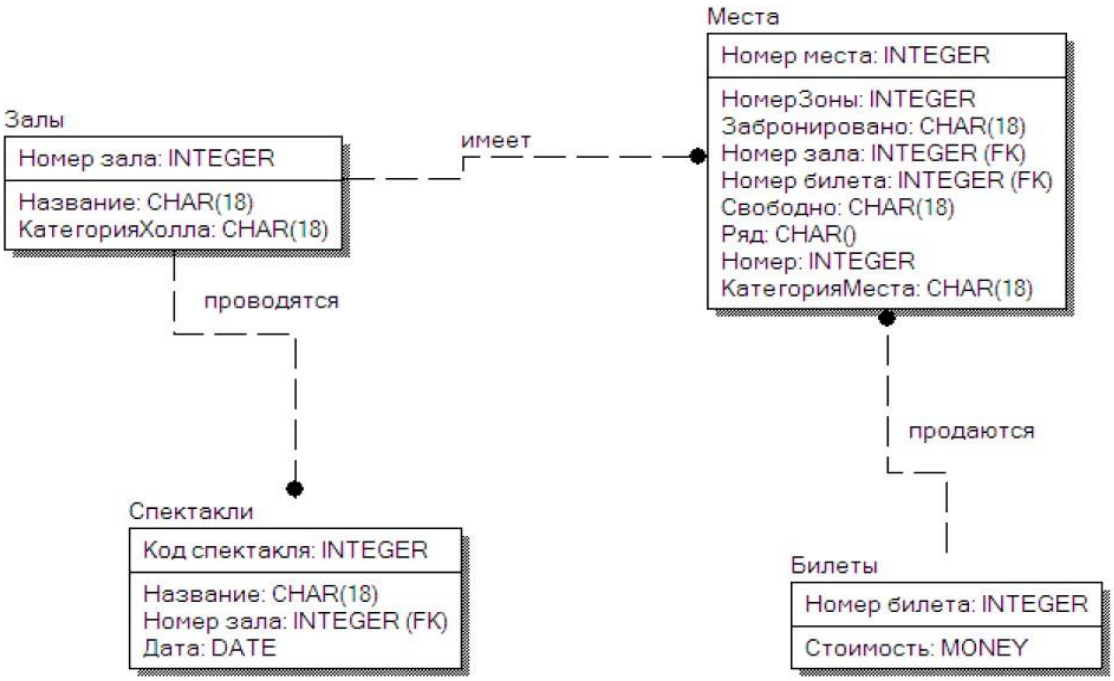

Рисунок 11 – Физическая модель данных

Модель показывает, как будет выглядеть схема данных в СУБД [20]. Из рисунка 11 видно, что все таблицы соединены связями и имеют внешние ключи в таблицах. Все таблицы почти связаны с таблицей «Места», которая содержит данные о билетах, залах.

Автоматизированная информационная система реализована в современной среде Visual Studio 2022 [20], которая позволяет подсвечивать код при выявлении ошибок или рекомендаций от разработчика. Система разрабатывается на языке программирования C# [18] который прекрасно интегрируется и имеет много преимуществ в Visual Studio 2022 [21].

Загрузка система осуществляется с помощью с exe файла, как происходит загрузка показано на рисунке 12.

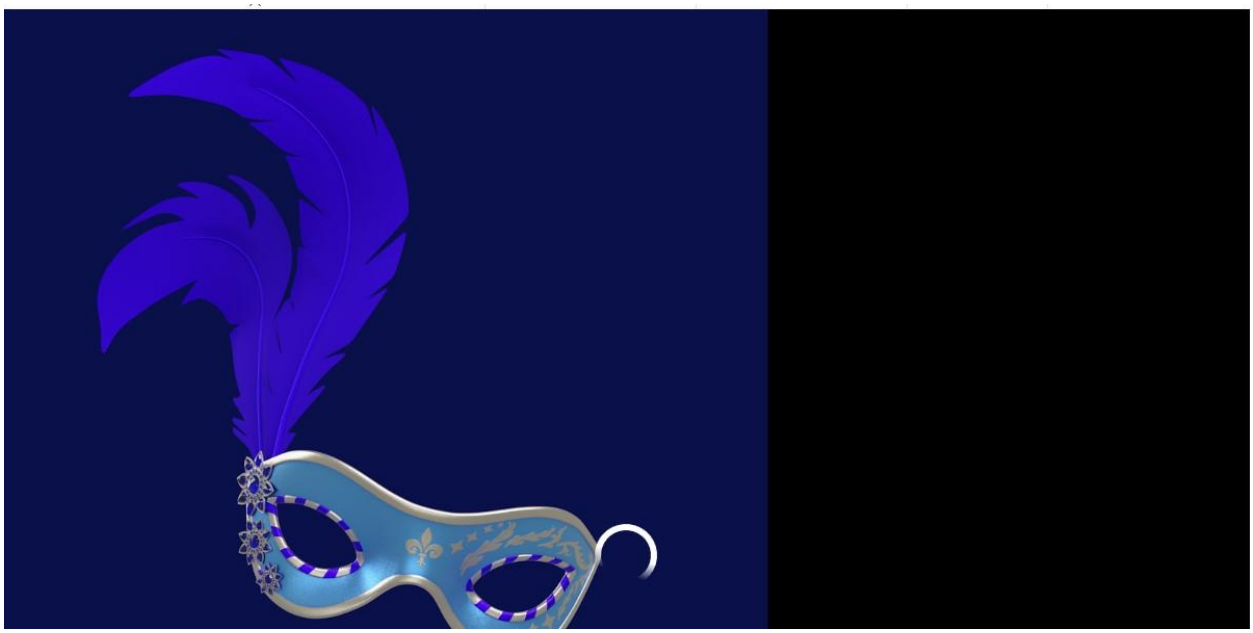

Рисунок 12 – Загрузка информационной системы билетной кассы театра

После загрузки информационной системы кассир будет видеть список спектаклей, которые стоят на постановке в театре на данный момент и на них открыта продажа билетов, рисунок 13.

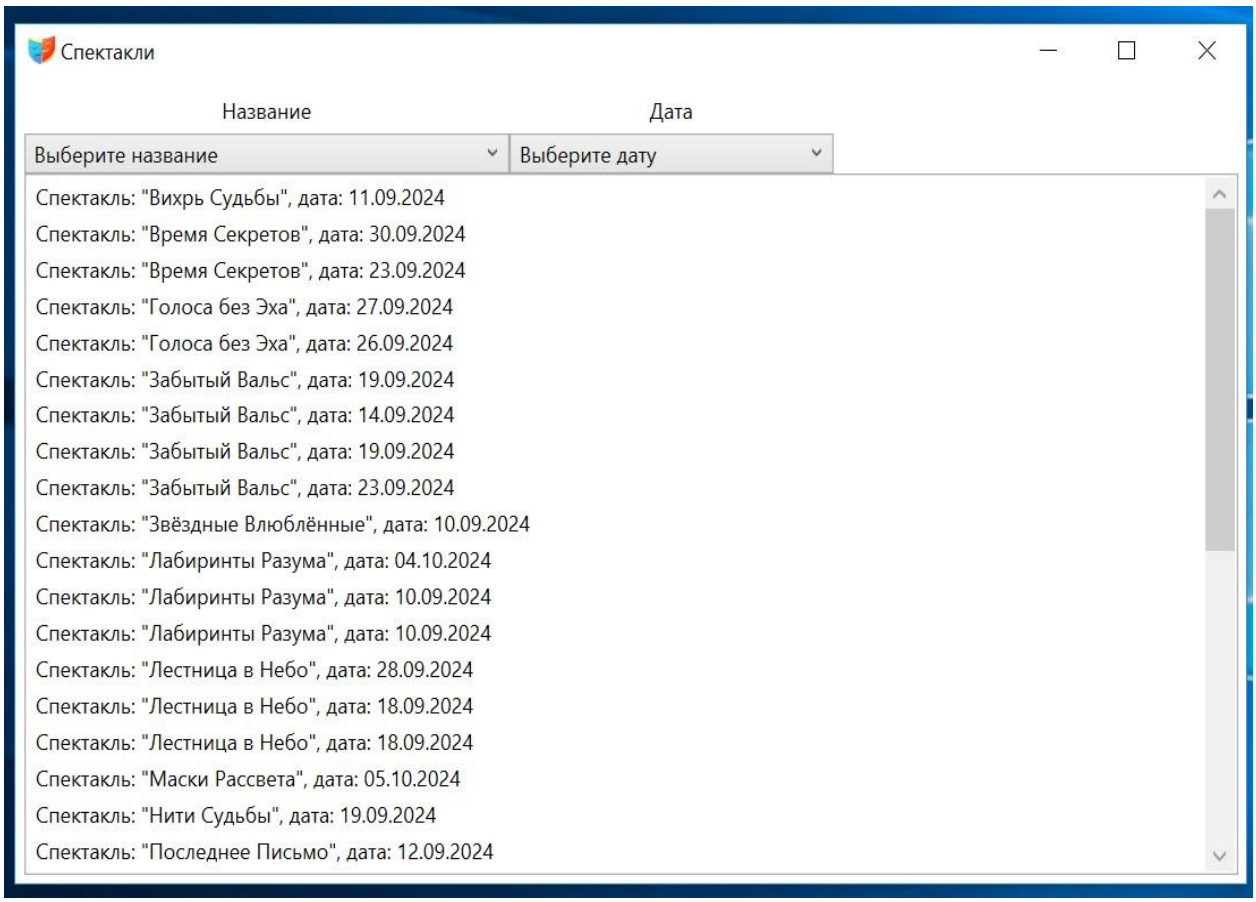

Рисунок 13 – Список спектаклей

Кассир при обслуживании посетителя в кассе может выбрать из списка спектакль, который нужен и дату, рисунок 14.

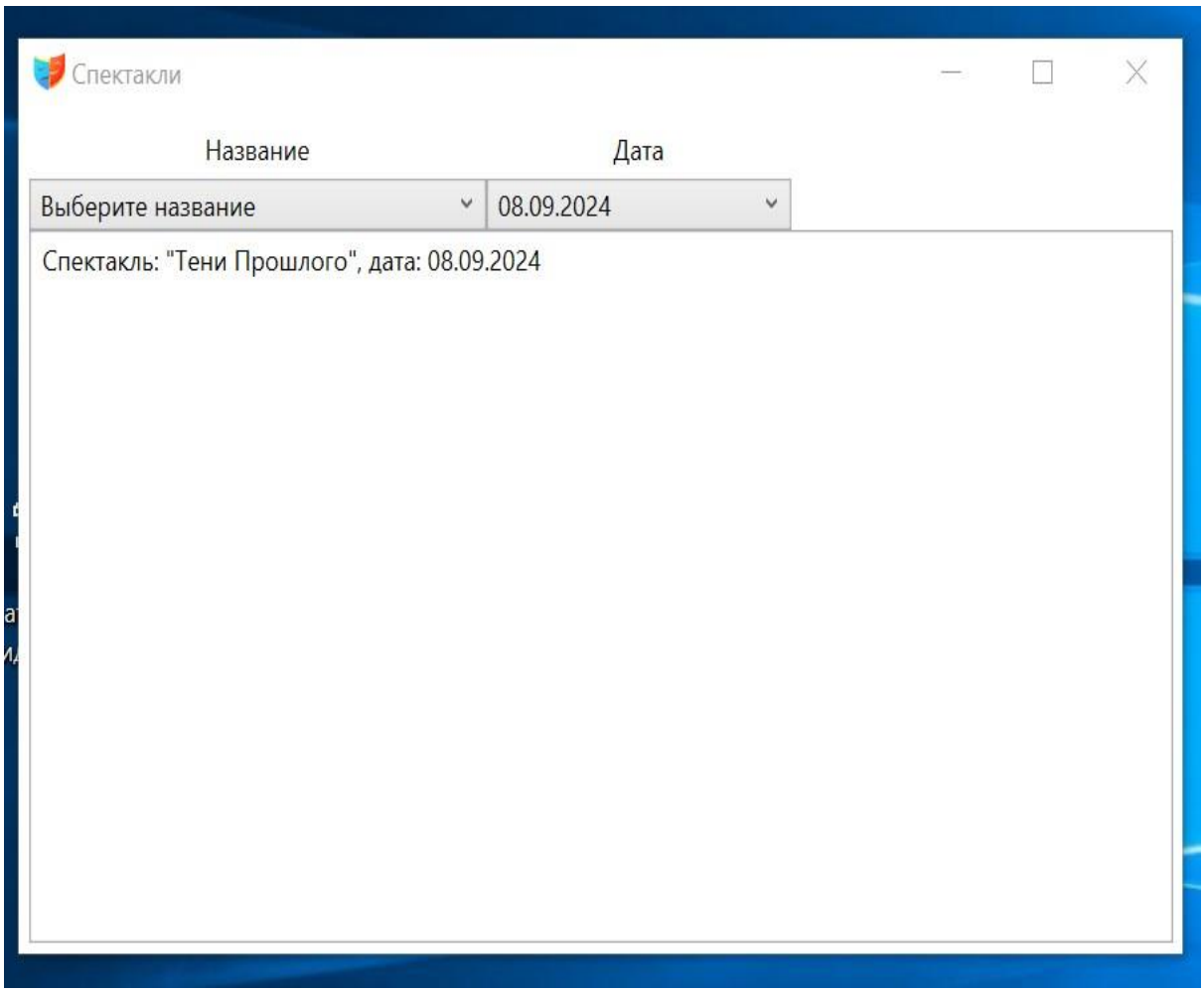

Рисунок 14 – Форма выбора спектакля и даты

При выборе спектакля по запросу посетителя театра открывается список зала и места, которые можно приобрести, рисунок 15.

Рисунок 15 – Выбор мест на спектакль

Как видно места имеют разный цвет, каждый цвет имеет свое значение:

– светло-зеленый цвет – место в первом ряду;

– зеленый – стандартное место;

– темно-зеленый цвет – место с ограниченным обзором

– желтый цвет – место в центре зала;

– черный цвет – зарезервировано;

– серый цвет – место не доступно для покупки.

При выборе места и наведения на него курсором подсвечивается информация о данном месте, рисунок 16.

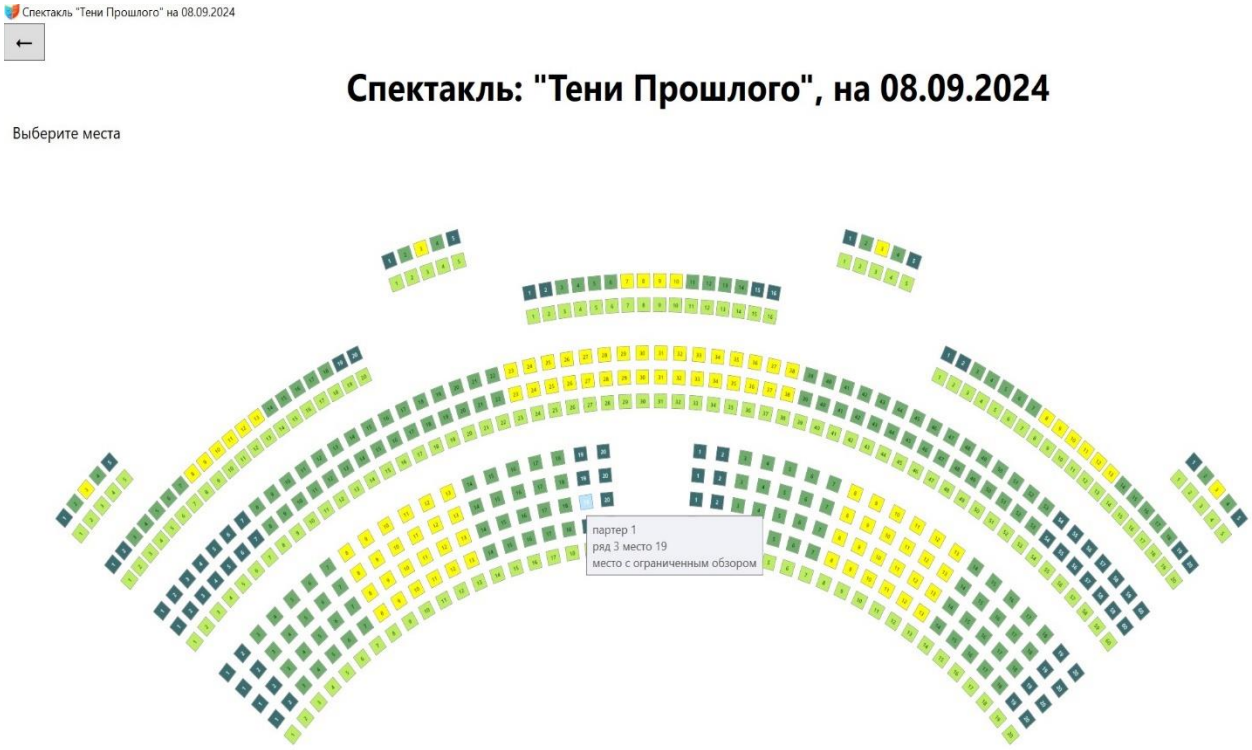

Рисунок 16 – Описание выбранного места

На картинке видно, что выбранное место находится в партере номер 1, на третьем ряду и место номер 19, которое является местом ограниченного обзора. Если посетитель выбирает данное место, и кассир подтверждает открывается информация о стоимости билета, также появляется возможность оплатить билет, рисунок 17.

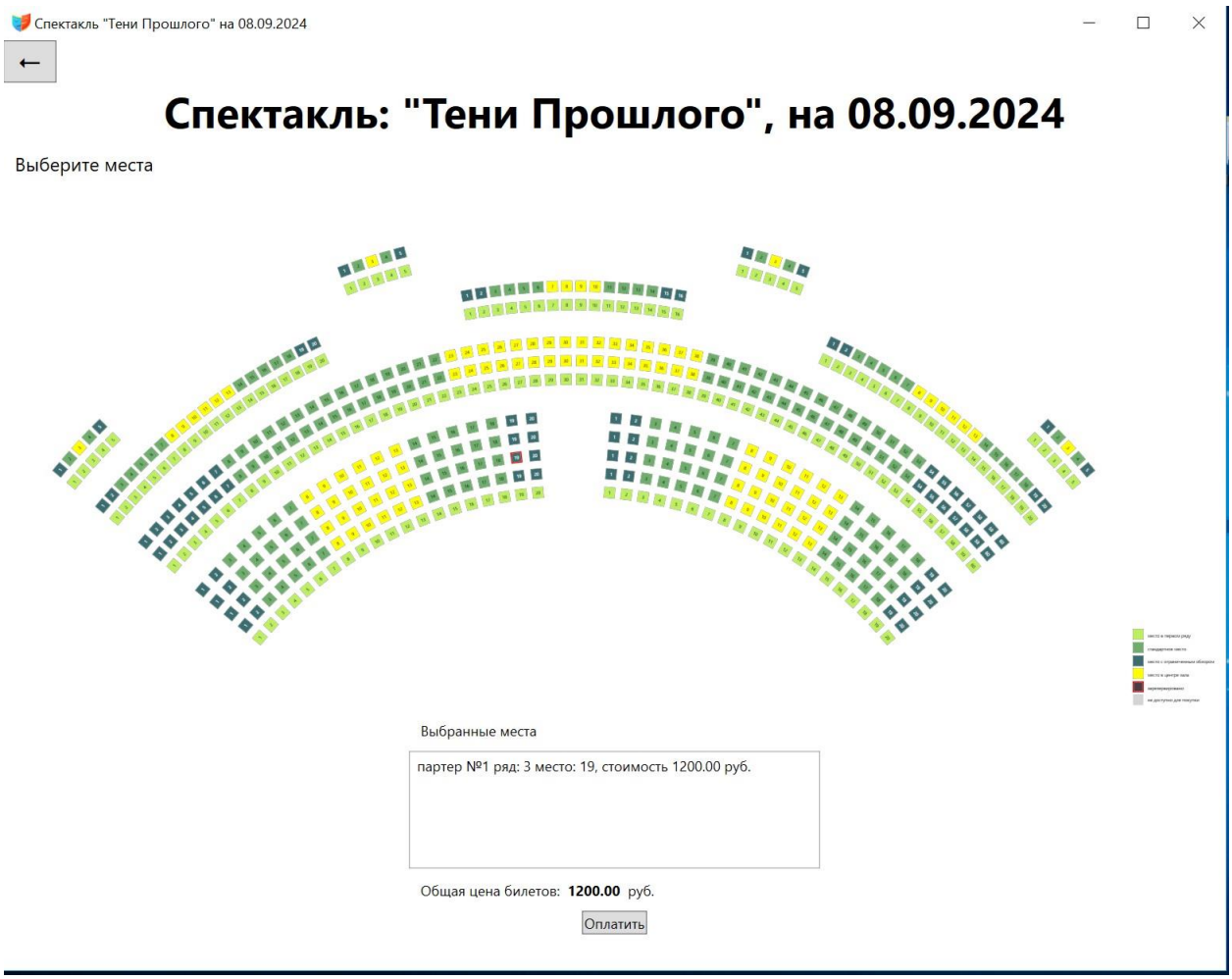

Рисунок 17 – Информация о стоимости билета на выбранное место в зале

Также информационная система позволяет выбрать сразу несколько мест и оплатить одним платежом, вся информация также отображается в системе и видна кассиру, рисунок 18.

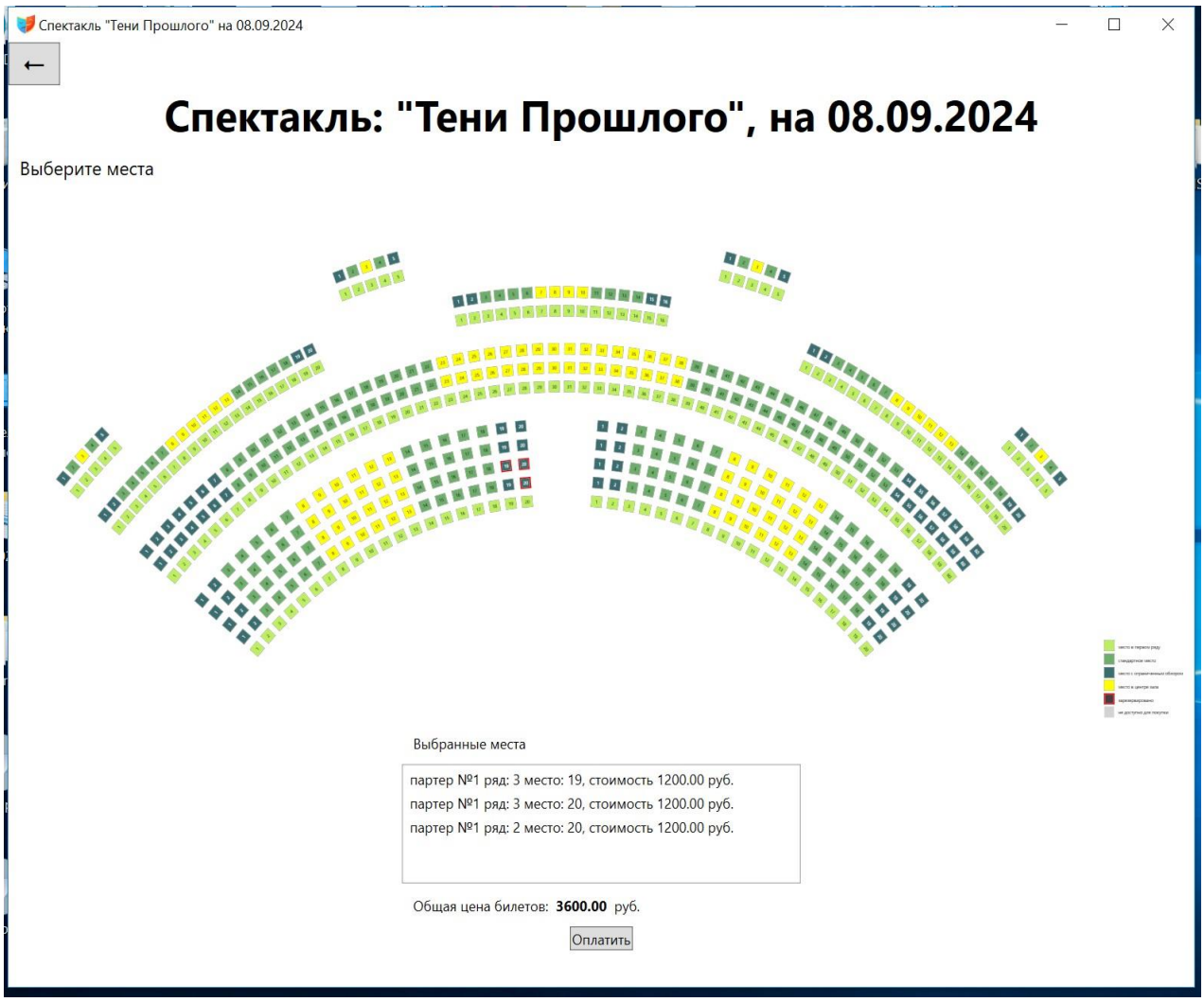

Рисунок 18 – Информация о стоимости нескольких билетов

Если посетитель вдруг передумал покупать билеты или появились другие обстоятельства, то кассир может вернуться назад как показано на рисунке 19 тогда все выбранные места обновятся как не выбранные.

Рисунок 19 – Кнопка для возврата в меню

Если посетитель все же решается купить билеты, тогда кассир может нажать кнопку «Оплатить», и система может запросить подтверждение того действия, которое было выбрано кассиром билетной кассы, рисунок 20.

В форме показаны все выбранные места на выбранный спектакль, также указана стоимость каждого выбранного билета и показана общая стоимость за все билеты который выбрал посетитель театра для покупки.

Чтобы билет был куплен и выбранное место в выбранном ряду было указано в системе как занято, необходимо кассиру подтвердить оплату после того, как система выведет окно для подтверждения оплаченной суммы.

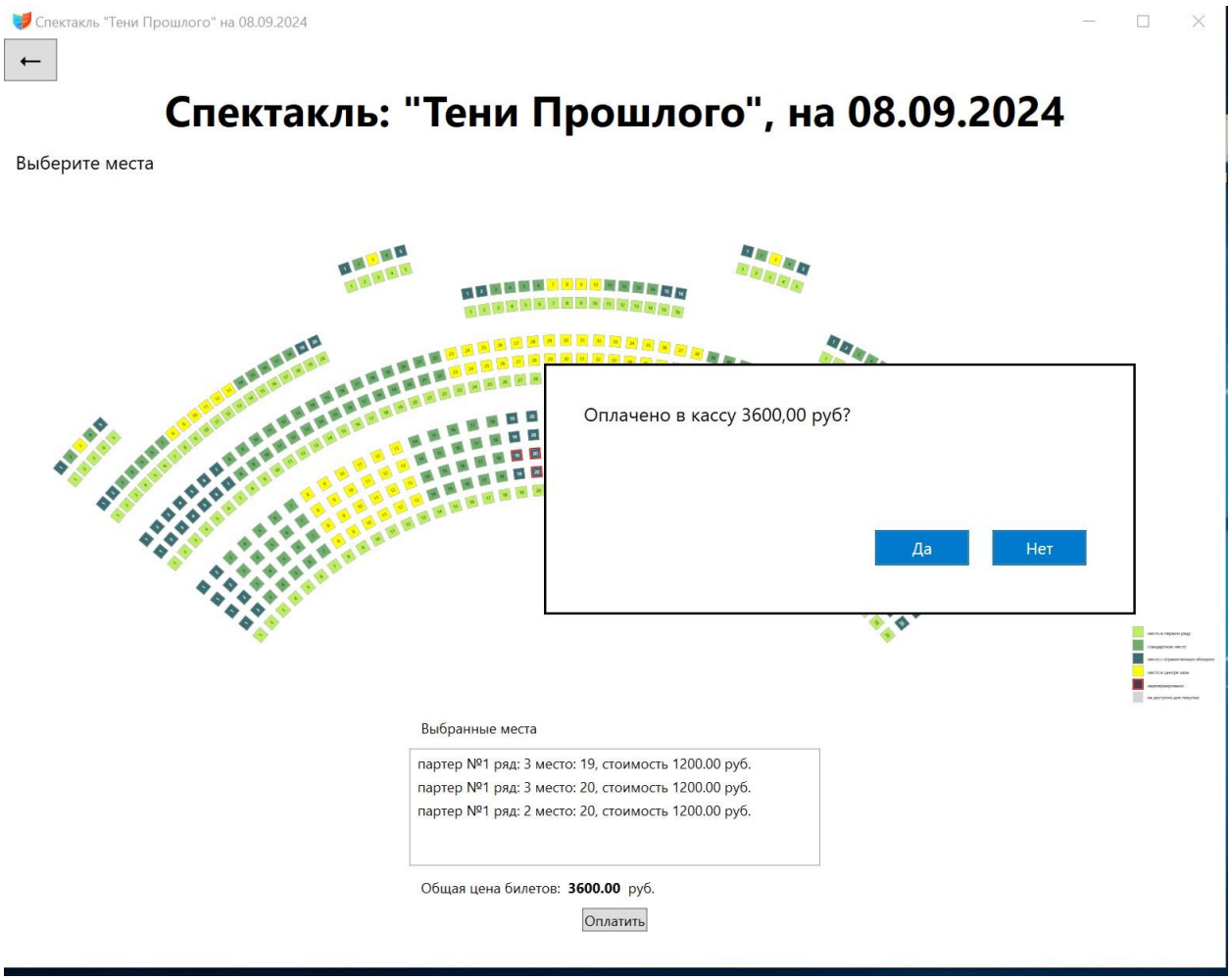

Рисунок 20 – Запрос на подтверждение оплаты билетов

После того как подтверждается кассиром оплата билетов, билеты сохраняются в электронном виде, где их можно открыть и посмотреть, рисунок 21.

Программа показывает путь куда сохраняются купленные билеты в электронном виде. Билет можно отправить посетителю на любой мессенджер в электронном виде, либо распечатать на бумажный носитель билет из электронного файла для предъявления его при посещении театра.

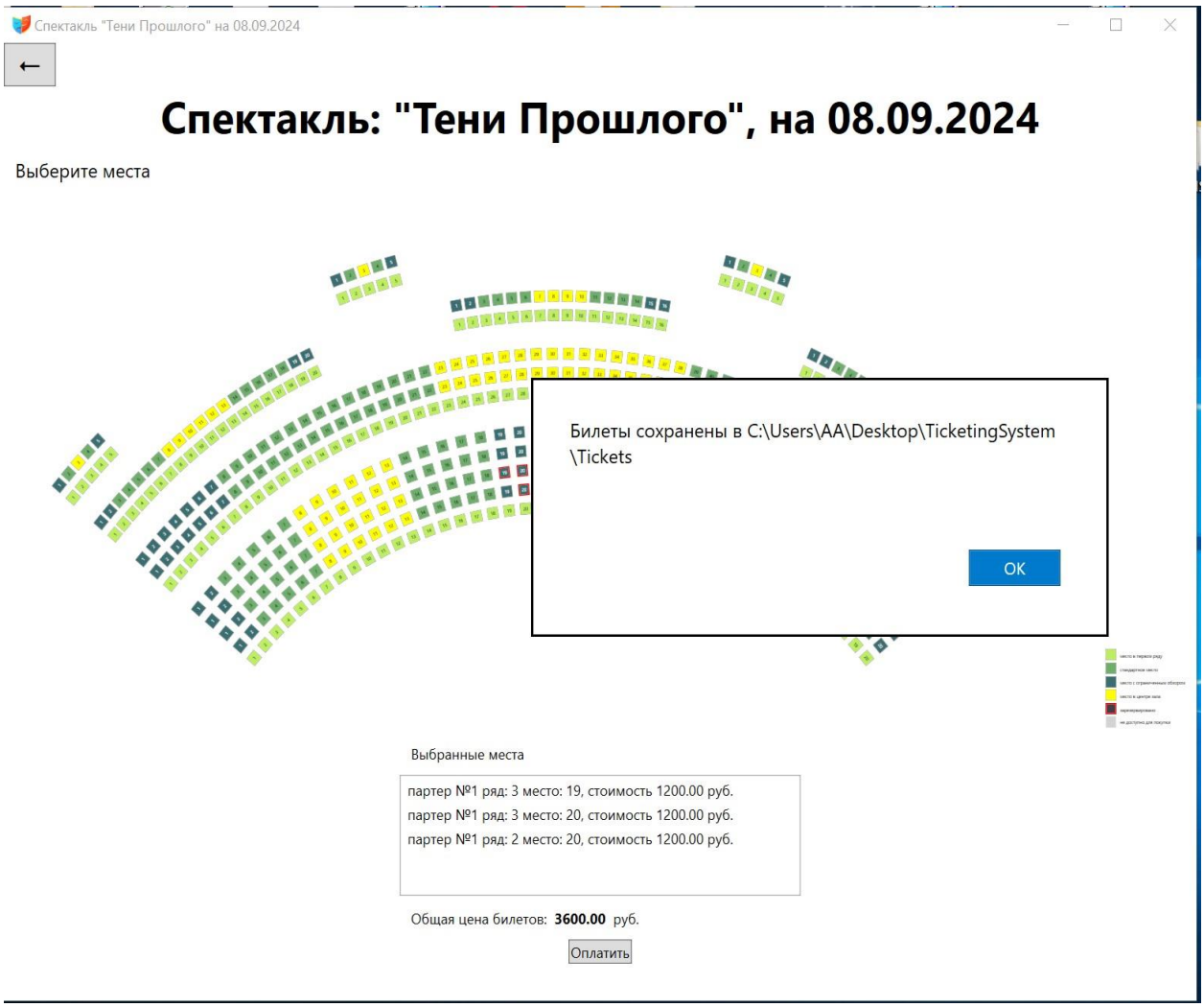

Рисунок 21 – Сохранение билета

Если оплата проходила за три билета сразу система все равно формирует на каждое место отдельный билет, где прописаны все данные о спектакле, ряд, место, стоимость билета и дата проведения спектакля, рисунок 22.

Сохраненный билет представляет собой красочную картинку, которая символизирует название спектакля. Посредине билета написано название спектакля или другой постановки и дату проведения.

На билете также выделен отрывной корешок для контролера, где продублирована вся информация о билете.

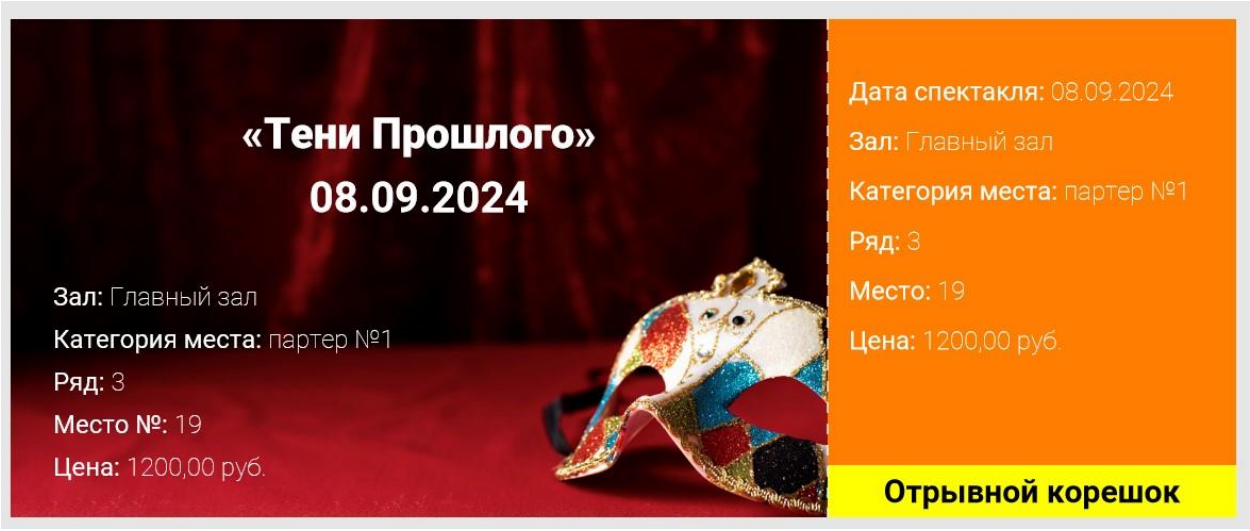

Рисунок 22 – Сформированный билет

После продажи и сохранения билета в системе данные места отображаются серым цветом как недоступные для продажи, рисунок 23.

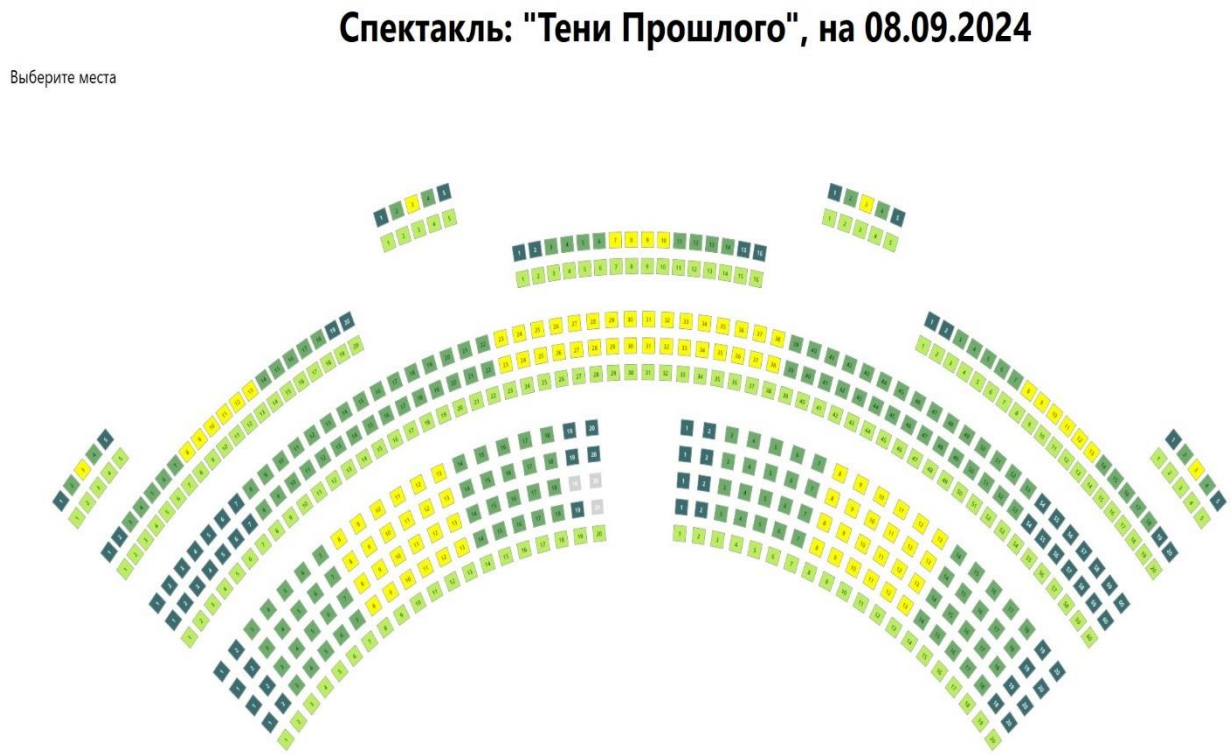

Рисунок 23 – Отображение купленных мест

3.3 Тестирование информационной системы

Проведем тестирование системы на проверку корректности работы и вывода информации об ошибках пользователю [9].

Кассир билетной кассы театра, работающий в системе, выбирает данные, которые уже введены в систему администратором системы. Ожидания от системы, что при выборе всех данных о спектакле или дате которые необходимо для продажи билета посетителю, система должна показать корректные данные по выбранной дате также отфильтровать только те спектакли, которые будут показаны, рисунок 24.

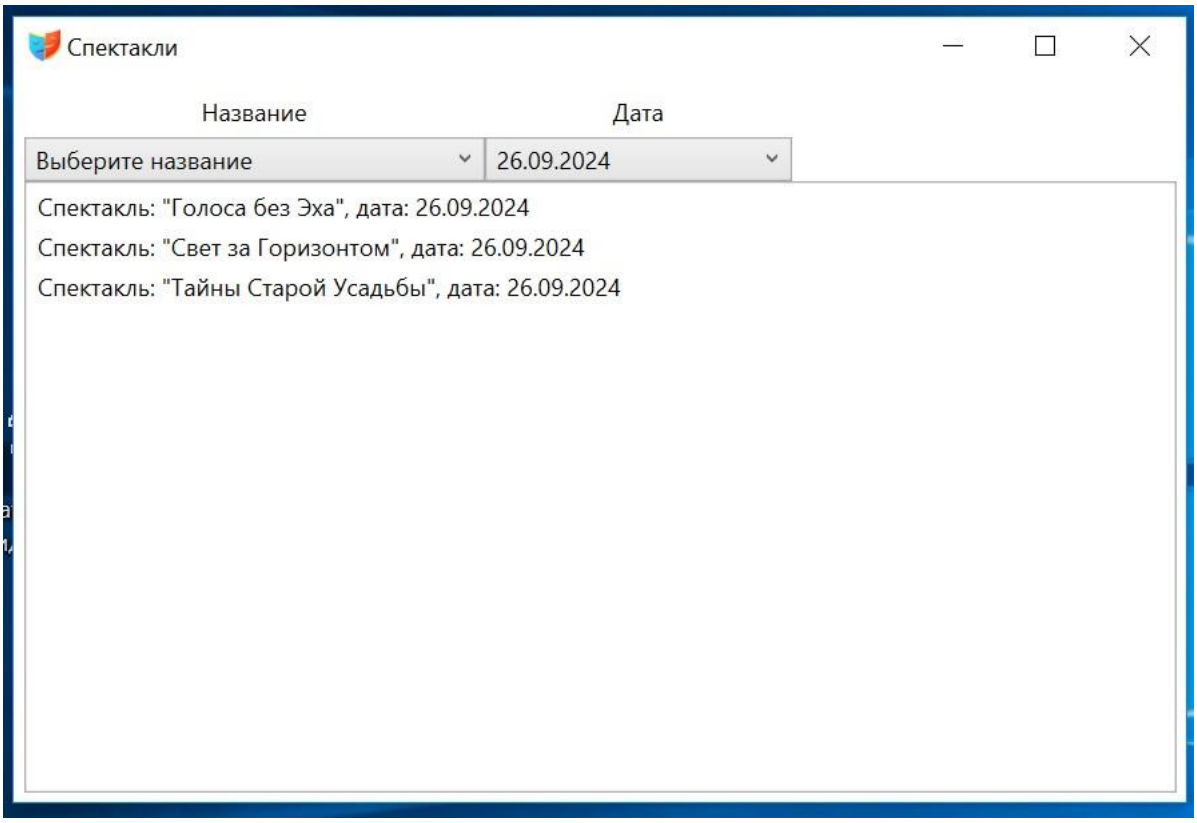

Рисунок 24 – Фильтрация спектаклей по датам

Если кассир не выбирает название спектакля и даты, то система не позволит выбирать места и оплачивать билет, рисунок 25.

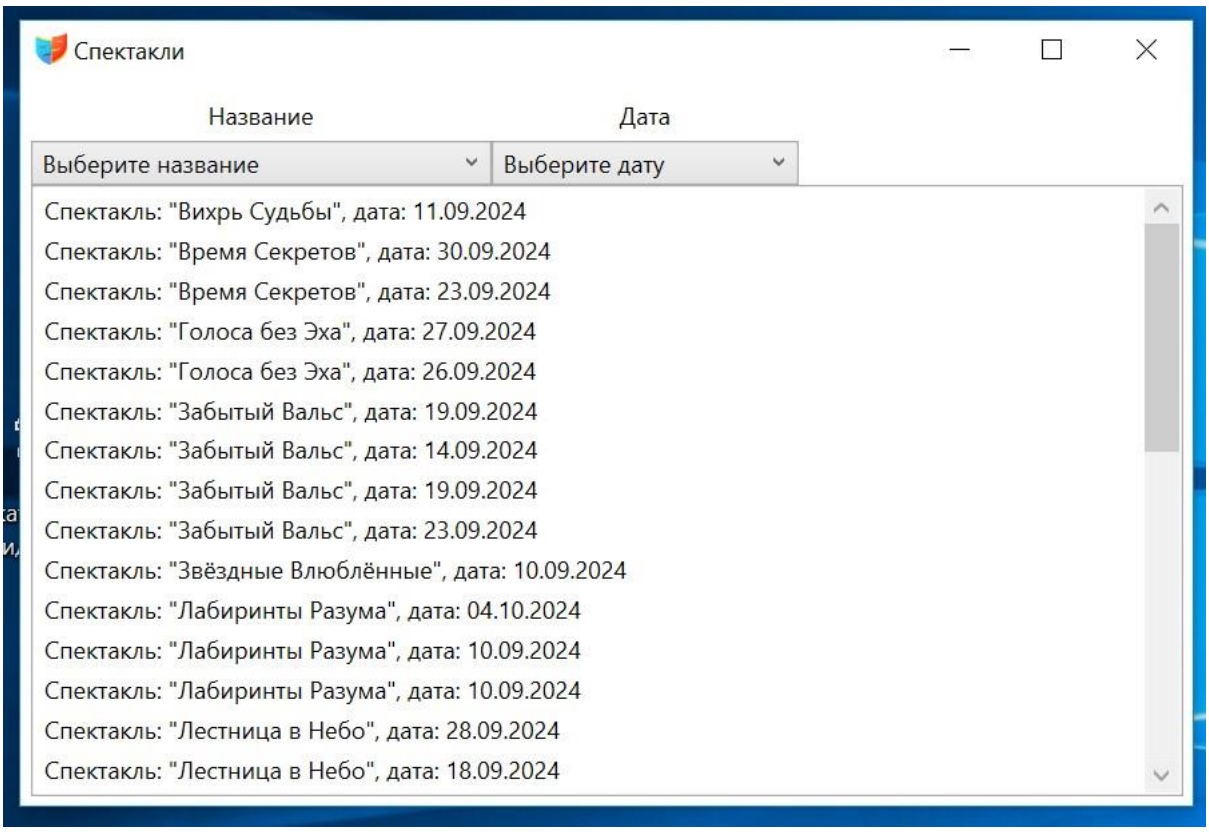

Рисунок 25 – Отсутствие информации о местах

Так была проведена автоматизация билетной кассы в Новосибирском театре оперы и балета которая позволит кассирам быстро и эффективно обслуживать посетителей театра.

Выводы по третьей главе.

В заключительной главе работы была проведена физическая реализация информационной системы билетной кассы Новосибирского театра оперы и балета. Разработана физическая модель базы данных в СУБД MS SQL Server 2022 и основные формы информационной системы с помощью, которой кассиры будут информировать посетителей и продавать билеты в театре. На основе проведенного тестирования можно сделать вывод, что информационная система отвечает всем поставленным требованиям и корректно выполняет функции билетной кассы театра.

Заключение

В ходе выполнения выпускной квалификационной работы описана автоматизация билетной кассы Новосибирского театра оперы и балета.

На первом этапе работы была рассмотрена организационная структура, в которой выделено подразделение, которое будет предметом рассмотрения в работе. Проведено изучение основных процессов билетной кассы театра, которые включают предоставление информации посетителям театра о наличии билетов на проводимые спектакли, доступности мест и стоимости выбранного билета. После чего были выявлены места, которые не автоматизированы и нуждаются в автоматизации с помощью внедрения информационных технологий.

На основе выявленных узких мест в процессах билетной кассы были изучены аналогичные программные разработки, представленные на российском рынке отечественными разработчиками. Анализ показал, что целесообразно будет разработать собственную автоматизированную информационную систему для билетной кассы.

Для реализации информационной системы были выбраны инструменты разработки C# и СУБД Microsoft SQL 2022, среда разработки Visual Studio 2022, как наиболее подходящие и эффективные для реализации системы.

Было проведено тестирование информационной системы. По результатам тестирования было установлено, что система выполняет все заложенные в нее функции.

Разработанная информационная система повысит эффективность работы билетных касс театра, а также позволит сотрудникам касс получать достоверную информацию о спектаклях, проводимых в театре и информировать посетителей актуальной информацией и о наличии билетов.

По результатам выполненной работы можно считать, что задачи выпускной квалификационной работы выполнены в полном объеме и поставленная цель достигнута.

Список используемой литературы и используемых источников

1. Как использовать Entity Framework. [Электронный ресурс]. – URL: https://skillbox.ru/media/code/entity_framework/ (дата обращения: 04.09.2024)

2. Концептуальное логическое и физическое моделирование данных. [Электронный ресурс]. – URL: https://frameworx.ru/SID/datamodelling.html (дата обращения: 04.09.2024).

3. Моделирование систем с использованием информационных технологий: учебн. пособие / В. Г. Лисиенко, Н. Г. Дружинина, О. Г. Трофимова, С. П. Трофимов. – Екатеринбург: УГТУ-УПИ, 2009. – 440 с.

4. Новосибирский театра оперы и балета. [Электронный ресурс]. – URL: https://novat.ru/theatre/theatre/history/ (дата обращения: 04.09.2024)

5. Онлайн курс обучения программированию: методологии разработки. [Электронный ресурс]. – URL: https://javarush.ru/groups/posts/647-metodologii-razrabotki-po (дата обращения: 04.09.2024)

6. Отличия, достоинства и недостатки базы данных PostgreSQL: что такое PostgreSQL. [Электронный ресурс]. – URL: https://www.postgresql.org/ (дата обращения: 04.09.2024)

7. Попова-Коварцева, Д. А. Основы проектирования баз данных: учеб. пособие / Д.А. Попова-Коварцева, Е.В. Сопченко. – Самара: Изд-во Самарского университета, 2019. – 112 с.

8. C GUI на C#: Кроссплатформенное приложение [Электронный ресурс]. - URL: https://skillbox.ru/media/code/ne_windows_(дата обращения: 04.09.2024)

9. Тамре Л. Введение в тестирование программного обеспечения /пер. с англ. М.: Вильямс, 2018. 368 с.

10. Этапы и методологии проектирования баз данных. [Электронный ресурс]. – URL: https://studfile.net/preview/2674691/page:4/ (дата обращения: 04.09.2024).

11. Access: База данных. [Электронный ресурс]. – URL: https://www.microsoft.com/ru-ru/microsoft-365/access(дата обращения: 04.09.2024)

12. GeekBrains – Язык программирования C#. [Электронный ресурс]. - URL: https://geekbrains.ru/posts/yazyk-programmirovaniya-c-sharp-istoriya-specifika-mesto-na-rynke (дата обращения: 04.09.2024)

13. Helpiks.org: Достоинства и недостатки языка [Электронный ресурс]. - URL: https://helpiks.org/6-21879.html (дата обращения: 04.09.2024)

14. Java-энциклопедия языков программирования: Java. [Электронный ресурс]. – URL: http://progopedia.ru/language/java/ (дата обращения: 04.09.2024)

15. SoftClipper: что такое SQL Server. [Электронный ресурс]. – URL: https://softclipper.net/foxpro-i-sql/sravnenie-baz-dannykh-microsoft-sql-server-i-microsoft-visual-foxpro.html (дата обращения: 04.09.2024)

16. SQL Server: руководство по SQL Server. [Электронный ресурс]. – URL: https://docs.microsoft.com/ru-ru/sql/sql-server/tutorials-for-sql-server-2016?view=sql-server-ver15 (дата обращения: 04.09.2024)

17. SQL Server: программное обеспечение. [Электронный ресурс]. – URL: https://www.microsoft.com/ru-ru/sql-server/sql-server-downloads (дата обращения: 04.09.2024)

18. C#. [Электронный ресурс]. – URL: https://dotnet.microsoft.com/en-us/languages/csharp (дата обращения: 04.09.2024)

19. The C Sharp (C#) Beginner’s Guide [Электронный ресурс]. – URL: https://medium.com/c-sharp-language/the-c-beginners-guide-6a14af03ed85 (дата обращения: 04.09.2024)

20. Visual Studio 2022: программное обеспечение. [Электронный ресурс]. – URL: https://visualstudio.microsoft.com/ru/vs/ (дата обращения: 04.09.2024)

21. Visual Studio 2022. [Электронный ресурс]. – URL: https://www.techspot.com/downloads/7493-visual-studio-2022.html (дата обращения: 04.09.2024)

22. Visual Studio tutorials | C#. [Электронный ресурс]. – URL: https://learn.microsoft.com/en-us/visualstudio/get-started/csharp/?view=vs-2022l (дата обращения: 04.09.2024)

23. Choy, K. L. Development of an intelligent customer-supplier relationship management system: the application of case-based reasoning / K. L. Choy, Kenny K H Fan, Victor Lo // Industrial Management & Data Systems. – 2003. – Vol. 103, No. 4. – P. 263-274.

24. Law, Y. F. D. An Integrated Case-Based Reasoning Approach for Intelligent Help Desk Fault Management / Y. F. D. Law, B. F. Sew, S. E. J. Kwan // Expert Systems with Applications. – 1997. – Vol. 13, No. 4. – P. 265-274. – EDN AJGJSH.

25. Twidale, M. B. Browsing is a collaborative process / M. B. Twidale, D. M. Nichols, C. D. Paice // Information Processing & Management. – 1997. – Vol. 33, No. 6. – P. 761-783. – EDN AITGCJ.

26. Mark, G. Conventions and Commitments in Distributed CSCW Groups / G. Mark // Computer Supported Cooperative Work. – 2002. – Vol. 11, No. 3-4. – P. 349-387. – EDN BBXQUN.

27. Simazu, H. ExpertGuide: A Conversational Case-Based Reasoning Tool for Developing Mentors in Knowledge Spaces / H. Simazu, A. Shibata, K. Nihei // Applied Intelligence. – 2001. – Vol. 14, No. 1. – P. 33-48. – EDN AMODOB.

Приложение А

Листинг программы

```
using TicketingSystem.Application.Interfaces;
using TicketingSystem.Core.Entities;

namespace TicketingSystem.Application
{
   public class NavigateBasedOnPlayUseCase : INavigateBasedOnPlayUseCase
   {
      private readonly INavigationService _navigationService;
      private readonly IPlayContext _playContext;

      public NavigateBasedOnPlayUseCase(
         INavigationService navigationService,
         IPlayContext playContext)
      {
         _navigationService = navigationService ?? throw new ArgumentNullException(nameof(navigationService));
         _playContext = playContext;
      }

      public async Task NavigateAsync(Play play, CancellationToken cancel = default)
      {
         try
         {
            ArgumentNullException.ThrowIfNull(play);
```

```
_playContext.CurrentPlay = play;

            await Task.Run(() =>
            {
                _navigationService.ShowView<IPlayView>();
                _navigationService.CloseView<IMainView>();
            });
        }
        catch (Exception e)
        {
            _navigationService.ShowErrorView($"Что-то пошло не так...{e.Message}");
        }
    }
  }
}

namespace TicketingSystem.Application
{
  public class NavigateBasedOnUserRoleUseCase : INavigateBasedOnUserRoleUseCase
  {
    private readonly IPasswordVerificationUseCase _passwordVerificationUseCase;
    private readonly INavigationService _navigationService;

    public NavigateBasedOnUserRoleUseCase(
      IPasswordVerificationUseCase passwordVerificationUseCase,
```

```
 INavigationService navigationService
        )
    {
        _passwordVerificationUseCase = passwordVerificationUseCase ?? throw new ArgumentNullException(nameof(passwordVerificationUseCase));
        _navigationService = navigationService ?? throw new ArgumentNullException(nameof(navigationService));
    }

    public async Task NavigateAsync(string username, string password, CancellationToken cancel = default)
    {
      try
      {
        await Task.Delay(500);
      }
      catch (AuthenticationException e)
      {
        _navigationService.ShowErrorView(e.Message);
      }
      catch (Exception e)
      {
        _navigationService.ShowErrorView($"Что-то пошло не так...{e.Message}");
      }
    }
  }
}
```

```
namespace TicketingSystem.Application
{
  public class NavigateToMainViewFromPlayView : INavigateToMainViewFromPlayView
  {
    private readonly INavigationService _navigationService;

    public NavigateToMainViewFromPlayView(
      INavigationService navigationService)
    {
      _navigationService = navigationService ?? throw new ArgumentNullException(nameof(navigationService));
    }

    public async Task NavigateAsync(CancellationToken cancel = default)
    {
      try
      {
        await Task.Run(() =>
        {
          _navigationService.ShowView<IMainView>();
          _navigationService.CloseView<IPlayView>();
        });
      }
      catch (Exception e)
      {
        _navigationService.ShowErrorView($"Что-то пошло не так...{e.Message}");
      }
```

```
        }
    }
}

    namespace TicketingSystem.Application
{
    public class PasswordService
    {
        private static object _locker = new object();
        public bool VerifyPassword(string password, string passwordHash)
        {
            string calculatedPasswordHash = CalculateHash(password);
            if (calculatedPasswordHash == passwordHash)
                return true;
            return false;
        }
        private static string CalculateHash(string text)
        {
            lock (_locker)
            {
                byte[] hash = Encoding.ASCII.GetBytes(text);
                byte[] hashEncoded = MD5.HashData(hash);
                string result = "";
                foreach (var oneByte in hashEncoded)
                {
                    result += oneByte.ToString("x2");
                }
                return result;
```

```
        }
      }
   }
}

namespace TicketingSystem.WpfApplication
{
   public static class Program
   {
      [STAThread]
      public static void Main()
      {
         try
         {
            var app = new App();
            app.InitializeComponent();
            app.Run();
         }
         catch (Exception e)
         {
            try
            {
               using var sw = new StreamWriter($"./startupErrors.txt", false, System.Text.Encoding.UTF8);
               sw.WriteLine(e.Message);
            }
            catch (Exception ex)
            {
               MessageBox.Show(ex.Message);
```

```
            }
            MessageBox.Show(e.Message);
        }
    }

    public static IHostBuilder CreateHostBuilder(string[] arg)
    {
        if (App.CurrentDirectory is null)
        {
            throw new InvalidOperationException("App.CurrentDirectory is null");
        }
        var hostBuilder = Host.CreateDefaultBuilder(arg);
        hostBuilder.UseContentRoot(App.CurrentDirectory);
        hostBuilder.ConfigureAppConfiguration((host, cfg) =>
        {
            if (App.IsDesignMode)
            {
                cfg.SetBasePath(App.CurrentDirectory);
            }
            else
            {
                cfg.SetBasePath(Environment.CurrentDirectory);
            }
            cfg.AddJsonFile("appsettings.json", optional: true, reloadOnChange:
true);
        });
hostBuilder.ConfigureServices(App.ConfigureServices);
        return hostBuilder;
    }
```

```
        }
}

namespace TicketingSystem.Infrastructure.Services
{
   public class PdfPrinter : IPrinter
   {
      private const string ArialFontPath = "Fonts/arial.ttf";
      private const string RobotoBlackFontPath = "Fonts/robotoBlack.ttf";
      private const string RobotoBoldFontPath = "Fonts/robotoBold.ttf";
      private const string RobotoLightFontPath = "Fonts/robotoLight.ttf";
      private const string RobotoMediumFontPath = "Fonts/robotoMedium.ttf";
      private const string RobotoRegularFontPath = "Fonts/robotoRegular.ttf";
      private const string RobotoThinFontPath = "Fonts/robotoThin.ttf";
      //private readonly Color _tearOffBackgroundColor = new DeviceRgb(0x47, 0xC4, 0xFF);
      private readonly Color _tearOffBackgroundColor = new DeviceRgb(0xFF, 0x7D, 0x00);
      private readonly Color _tearOffFontColor = ColorConstants.WHITE;

      public string Print(Seat seat, Play play)
      {
         var sourcePath = System.IO.Path.Combine(Environment.CurrentDirectory, "Tickets");
         var path = System.IO.Path.Combine(sourcePath, $"{play.Title}", $"{play.Date:dd-MM-yyyy}");
         var fileName = $"Ticket-{seat.SeatCategory}{seat.ZoneNumber}-Row{seat.Row}-Number{seat.Number}.pdf";
         var fullName = System.IO.Path.Combine(path, fileName);
```

```
        if (!Directory.Exists(path))
        {
            Directory.CreateDirectory(path);
        }
        using (PdfWriter writer = new PdfWriter(fullName))
        {
            PdfDocument pdf = new PdfDocument(writer);
            PageSize pageSize = new PageSize(UnitConverter.InchesToPoints(3.25f),
UnitConverter.InchesToPoints(8.00f)).Rotate();
            pdf.SetDefaultPageSize(pageSize);
            Document document = new Document(pdf);

            PdfFont arialFont = PdfFontFactory.CreateFont(ArialFontPath,
PdfEncodings.IDENTITY_H);
            PdfFont robotoBlackFont =
PdfFontFactory.CreateFont(RobotoBlackFontPath, PdfEncodings.IDENTITY_H);
            PdfFont robotoBoldFont =
PdfFontFactory.CreateFont(RobotoBoldFontPath, PdfEncodings.IDENTITY_H);
            PdfFont robotoLightFont =
PdfFontFactory.CreateFont(RobotoLightFontPath, PdfEncodings.IDENTITY_H);
            PdfFont robotoMediumFont =
PdfFontFactory.CreateFont(RobotoMediumFontPath,
PdfEncodings.IDENTITY_H);
            PdfFont robotoRegularFont =
PdfFontFactory.CreateFont(RobotoRegularFontPath,
PdfEncodings.IDENTITY_H);
            PdfFont robotoThinFont =
PdfFontFactory.CreateFont(RobotoThinFontPath, PdfEncodings.IDENTITY_H);
            PdfPage page = pdf.AddNewPage();
```

```
PdfCanvas canvas = new PdfCanvas(page);

float margin = 20; // Отступы
float mainSectionWidth = pageSize.GetWidth() * 2 / 3; // Основная секция занимает две трети ширины
float tearOffWidth = pageSize.GetWidth() - mainSectionWidth; // Ширина отрывного корешка

//// Путь к изображению (должен быть локальным файлом)
//string imagePath = "background.jpg";
//ImageData imageData = ImageDataFactory.Create(imagePath);

// Получаем сборку, в которой находится класс
Assembly assembly = Assembly.GetExecutingAssembly();

// Полное имя файла ресурса
//string resourceName = "TicketingSystem.Images.background.jpg";
string resourceName = "TicketingSystem.Infrastructure.Services.Images.background.png";

//// Открываем поток ресурса
//using Stream? stream = assembly.GetManifestResourceStream(resourceName);
//if (stream == null)
//    throw new FileNotFoundException("Не удалось найти встроенный ресурс.", resourceName);

//// Создаем ImageData из потока
//ImageData imageData = ImageDataFactory.Create(stream);
```

```
            // Загружаем изображение из потока ресурсов
            using Stream? stream = assembly.GetManifestResourceStream(resourceName);
            if (stream == null)
                throw new FileNotFoundException("Не удалось найти ресурс: " + resourceName);

            using MemoryStream memoryStream = new MemoryStream();
            stream.CopyTo(memoryStream); // Копируем данные в MemoryStream
            byte[] imageDataBytes = memoryStream.ToArray(); // Преобразуем данные потока в массив байтов

            // Создаем ImageData из массива байтов
            ImageData imageData = ImageDataFactory.Create(imageDataBytes);

            //var a = UnitConverter.PointsToInches(mainSectionWidth);

            PageSize imagePageSize = new PageSize(UnitConverter.InchesToPoints(3.25f), mainSectionWidth).Rotate();

            // Размещение фонового изображения на всей странице
            canvas.AddImageFittedIntoRectangle(imageData, imagePageSize, true);


            // Фон для отрывного корешка
            canvas.SaveState()
                .SetFillColor(_tearOffBackgroundColor)
                .Rectangle(mainSectionWidth, 0, tearOffWidth, pageSize.GetHeight())
                .Fill()
```

```
            .RestoreState();

        // Фон для отрывного корешка
        PdfCanvas tearOffCanvas = new PdfCanvas(page);
        tearOffCanvas.SaveState()
            .SetFillColor(_tearOffBackgroundColor)
            .Rectangle(mainSectionWidth, 0, tearOffWidth, pageSize.GetHeight())
            .Fill()
            .RestoreState();

        float xPositionTearOffInfo = mainSectionWidth + 10;

        // Добавление информации в отрывной корешок
        AddTearOffInfo(page, pdf, robotoRegularFont, robotoThinFont, "Дата спектакля", play.Date.ToString("dd.MM.yyyy"), tearOffWidth, xPositionTearOffInfo, pageSize.GetHeight() * 0.7f);
        AddTearOffInfo(page, pdf, robotoRegularFont, robotoThinFont, "Зал", play.Hall.Name, tearOffWidth, xPositionTearOffInfo, pageSize.GetHeight() * 0.6f);
        AddTearOffInfo(page, pdf, robotoRegularFont, robotoThinFont, "Категория места", $"{seat.SeatCategory.ToRussian()} №{seat.ZoneNumber}", tearOffWidth, xPositionTearOffInfo, pageSize.GetHeight() * 0.5f);
        AddTearOffInfo(page, pdf, robotoRegularFont, robotoThinFont, "Ряд", seat.Row.ToString(), tearOffWidth, xPositionTearOffInfo, pageSize.GetHeight() * 0.4f);
        AddTearOffInfo(page, pdf, robotoRegularFont, robotoThinFont, "Место", seat.Number.ToString(), tearOffWidth, xPositionTearOffInfo, pageSize.GetHeight() * 0.3f);
```

```
        AddTearOffInfo(page, pdf, robotoRegularFont, robotoThinFont, "Цена", $"{seat.Ticket.Price} руб.", tearOffWidth, xPositionTearOffInfo, pageSize.GetHeight() * 0.2f);

        // Отрывной корешок
        Paragraph tearOff = new Paragraph("Отрывной корешок")
            .SetTextAlignment(TextAlignment.CENTER)
            .SetFontSize(15)
            .SetFont(robotoBoldFont)
            .SetBackgroundColor(ColorConstants.YELLOW)
            .SetWidth(tearOffWidth)
            .SetFixedPosition(mainSectionWidth, 0, tearOffWidth); // Позиционирование отрывного корешка
        document.Add(tearOff);

        // Рисуем вертикальную пунктирную линию белого цвета для разграничения
        float lineXPosition = mainSectionWidth; // Позиция по горизонтали, где основная часть встречается с отрывным корешком
        canvas.SaveState();
        canvas.SetLineWidth(0.7f); // Устанавливаем толщину линии
        canvas.SetStrokeColor(ColorConstants.WHITE); // Устанавливаем белый цвет линии
        canvas.SetLineDash(5f, 1f); // Устанавливаем пунктирный стиль: 5 точек чередуются с 1 пробелами
        canvas.MoveTo(lineXPosition, 0); // Начало линии в верхней части страницы
        canvas.LineTo(lineXPosition, pageSize.GetHeight()); // Конец линии в нижней части страницы
```

```
        canvas.Stroke();
            canvas.RestoreState();
            // Заголовок спектакля и дата
            Paragraph titleHeader = new Paragraph($"«{play.Title}»")
                .SetTextAlignment(TextAlignment.CENTER)
                .SetFontSize(20)
                .SetFont(robotoBlackFont)
                .SetFontColor(ColorConstants.WHITE)
                .SetWidth(mainSectionWidth - 2 * margin)
                //.SetFixedPosition(margin, pageSize.GetHeight() - 100,
mainSectionWidth - 2 * margin);
                //.SetFixedPosition(margin, pageSize.GetHeight() - 70,
mainSectionWidth - 2 * margin);
                .SetFixedPosition(margin, pageSize.GetHeight() - 70,
mainSectionWidth - 2 * margin);
            document.Add(titleHeader);

            // Заголовок спектакля и дата
            Paragraph dateHeader = new Paragraph($"{play.Date:dd.MM.yyyy}")
                .SetTextAlignment(TextAlignment.CENTER)
                .SetFontSize(20)
                .SetFont(robotoBoldFont)
                .SetFontColor(ColorConstants.WHITE)
                .SetWidth(mainSectionWidth - 2 * margin)
                //.SetFixedPosition(margin, pageSize.GetHeight() - 100,
mainSectionWidth - 2 * margin);
                //.SetFixedPosition(margin, pageSize.GetHeight() - 70,
mainSectionWidth - 2 * margin);
```

```
        .SetFixedPosition(margin, pageSize.GetHeight() - 100, mainSectionWidth - 2 * margin);
            document.Add(dateHeader);

            // Остальные элементы
            float startY = pageSize.GetHeight() - 120;
            AddSeatInfo(document, "Зал", play.Hall.Name, robotoRegularFont, robotoThinFont, mainSectionWidth, margin, startY - 20);
            AddSeatInfo(document, "Категория места", $"{seat.SeatCategory.ToRussian()} №{seat.ZoneNumber}", robotoRegularFont, robotoThinFont, mainSectionWidth, margin, startY - 40);
            AddSeatInfo(document, "Ряд", seat.Row.ToString(), robotoRegularFont, robotoThinFont, mainSectionWidth, margin, startY - 60);
            AddSeatInfo(document, "Место №", seat.Number.ToString(), robotoRegularFont, robotoThinFont, mainSectionWidth, margin, startY - 80);
            AddSeatInfo(document, "Цена", $"{seat.Ticket.Price} руб.", robotoRegularFont, robotoThinFont, mainSectionWidth, margin, startY - 100);

            document.Close();
        }
        return sourcePath;
    }
    private void AddTearOffInfo(PdfPage page, PdfDocument pdf, PdfFont labelFont, PdfFont valueFont, string label, string value, float width, float xPosition, float yPosition)
    {
        //Rectangle rect = new Rectangle(xPosition, yPosition, width, 50);
        //Canvas infoCanvas = new Canvas(new PdfCanvas(page), rect, false);
        //infoCanvas.Add(new Paragraph($"{label}: {value}")
```

```
        //    .SetFont(font)
        //    .SetFontColor(_tearOffFontColor)
        //    .SetFontSize(12)
        //    //.SetTextAlignment(TextAlignment.CENTER)
        //    .SetTextAlignment(TextAlignment.LEFT)
        //    .SetBackgroundColor(_tearOffBackgroundColor))
        //    .Close();

        Rectangle rect = new Rectangle(xPosition, yPosition, width, 50);
        Canvas infoCanvas = new Canvas(new PdfCanvas(page), rect, false);
        infoCanvas.Add(new Paragraph()
            .Add(new Text(label + ": ").SetFont(labelFont))
            .Add(new Text(value).SetFont(valueFont))
            .SetFontColor(_tearOffFontColor)
            .SetFontSize(12)
            //.SetTextAlignment(TextAlignment.CENTER)
            .SetTextAlignment(TextAlignment.LEFT)
            .SetBackgroundColor(_tearOffBackgroundColor))
            .Close();
    }
    /// <summary>
    /// Добавляет информацию о месте в документ.
    /// </summary>
    /// <param name="document">Документ, в который добавляется информация.</param>
    /// <param name="label">Метка поля (например, "Ряд").</param>
    /// <param name="value">Значение поля (например, "5").</param>
    /// <param name="font">Шрифт, используемый для текста.</param>
```

```
        /// <param name="sectionWidth">Ширина секции, где расположена информация.</param>
        /// <param name="marginLeft">Отступ слева для текста.</param>
        /// <param name="positionY">Позиция Y для текста.</param>
        private void AddSeatInfo(Document document, string label, string value, PdfFont labelFont, PdfFont valueFont, float sectionWidth, float marginLeft, float positionY)
        {
            // Создаем параграф с жирным текстом для метки и обычным текстом для значения
            Paragraph paragraph = new Paragraph()
                .Add(new Text(label + ": ").SetFont(labelFont))
                .Add(new Text(value).SetFont(valueFont))
                .SetFontSize(12)
                .SetFontColor(ColorConstants.WHITE)
                .SetWidth(sectionWidth - 2 * marginLeft)
                .SetFixedPosition(marginLeft, positionY, sectionWidth - 2 * marginLeft);

            document.Add(paragraph);
        }
    }
}

namespace TicketingSystem.Infrastructure.Services
{
    public static class UnitConverter
    {
        private const float PointsPerInch = 72f;
        private const float MillimetersPerInch = 25.4f;
```

```
  private const float CentimetersPerMillimeter = 10f;


public static float MillimetersToPoints(float millimeters)
{
   return (millimeters / MillimetersPerInch) * PointsPerInch;
}
public static float CentimetersToPoints(float centimeters)
{
   return MillimetersToPoints(centimeters * CentimetersPerMillimeter);
}
public static float InchesToPoints(float inches)
{
   return inches * PointsPerInch;
}
public static float PointsToMillimeters(float points)
{
   return (points / PointsPerInch) * MillimetersPerInch;
}
public static float PointsToCentimeters(float points)
{
   return PointsToMillimeters(points) / CentimetersPerMillimeter;
}
public static float PointsToInches(float points)
{
   return points / PointsPerInch;
}}}
```