

**АВТОНОМНАЯ НЕКОММЕРЧЕСКАЯ ОРГАНИЗАЦИЯ
ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ
ОРГАНИЗАЦИЯ МОСКОВСКИЙ МЕЖДУНАРОДНЫЙ
КОЛЛЕДЖ ЦИФРОВЫХ ТЕХНОЛОГИЙ
«АКАДЕМИЯ ТОП»**

**Тема: Разработка проекта «Система онлайн - бронирования
билетов в кинотеатр»**

Преподаватель:
Рослова О.А

Участники проекта:
Белов Е.В
Вакула Я.Г
Сухинин А.В

Санкт-Петербург
2025

Оглавление

Введение:	3
Цель проекта:	4
Задачи проекта:	4
Проектирование:	5
ER Диаграмма:	5
Описание сущностей:	6
1. Пользователь (User) Лицо, создающее бронирования.	6
2. Бронирование (Booking) Хранит дату заказа, сумму и статус оплаты.	6
3. Билет (Ticket) Отвечает за место и цену билета.	6
4. Сеанс (Screening) Определяет конкретный показ фильма.	7
5. Зал (Cinema Hall) Организует размещение зрителей.	7
6. Фильм (Movie) Предоставляет информацию для расписания.	7
7. Роль (Roles) Определяет роли пользователей, например администратор, пользователь и т.п.	8
8. Тип зала (HallType) Определяет тип зала и повышает стоимость.	8
Нормализация базы данных "Кинотеатр" до 3NF	9
Исходная ненормализованная таблица	9
Первая нормальная форма (1NF)	10
Вторая нормальная форма (2NF)	11
Третья нормальная форма (3NF)	12
Описание таблиц	14
1. User	14
2. Movie	14
3. Hall_Type	14
4. Cinema_Hall	14
5. Screening	15
6. Booking	15
7. Ticket	16
8. Роли	16
Индексы	16
Диаграмма	17
Дополнительные функции:	17
Интерфейс программы	18
Интерфейс:	18
1. Вход	18
2. Фильмы/Добавление фильмов	19
3. Сеансы	20
4. Добавление Пользователей	21
5. Создание бронирования	22
Тестирование системы	23
Тест-кейс 1: Вход администратора в систему	23
Тест-кейс 2: Вход обычного пользователя без регистрации	23
Тест-кейс 3: Добавление нового фильма администратором	24
Вывод:	25
Приложение с кодом:	26

Введение:

В настоящее время наблюдается активное развитие цифровых технологий в сфере услуг и развлечений. Особенно это касается киноиндустрии, где традиционные способы бронирования билетов через кассы постепенно

уступают место онлайн системам. Как студент, изучающий информационные технологии, я вижу практическую значимость в разработке подобных систем, поскольку они не только решают конкретные бизнес задачи, но и позволяют применить полученные в университете знания на практике.

Современные потребители привыкли к возможности быстрого и удобного получения услуг через интернет. Это касается бронирования билетов в кино – пользователи хотят иметь возможность выбрать фильм, время сеанса и места в зале без необходимости личного посещения кинотеатра. Такие системы особенно востребованы в крупных городах, где ценность времени значительно высока.

С технической точки зрения, разработка системы бронирования билетов представляет интерес позиции проектирования архитектуры базы данных, реализации бизнес логики и создания удобного пользовательского интерфейса. Это комплексная задача, требующая понимания принципов работы с базами данных, организации многопользовательского доступа и обеспечения целостности информации.

Для нас этот проект является возможностью продемонстрировать умение работать с современными технологиями.NET и SQL Server, а также показать способность к полноценной разработке программного обеспечения – от проектирования базы данных до реализации функциональности и тестирования. Полученный опыт будет полезен в дальнейшей профессиональной деятельности в области разработки программного обеспечения.

Цель проекта:

Разработать консольное приложение для онлайн бронирования билетов в кинотеатр, позволяющее пользователям просматривать расписание сеансов, бронировать билеты, управлять бронированиями и просматривать информацию о фильмах и залах.

Задачи проекта:

1. Разработать нормализованную базу данных (3 НФ) с использованием MS SQL Server.
2. Какие объекты будут(3шт)
3. Реализовать консольный интерфейс с меню и очисткой экрана.
 - a. С использованием языка программирования C#
 - b. IDE visual studio
4. Обеспечить CRUD-операции для основных сущностей с использованием технологий ADO.net DAPPER
5. Реализовать несколько видов выборки с фильтрацией и сортировкой.
6. Протестировать
7. Отчет о работе
8. Презентация

Проектирование:

В данной системе предусмотрено 8 концептуальных объектов (сущностей) – бронирование, пользователь, сеанс, зал, билет, фильм, тип зала, тип пользователя.

ER Диаграмма:

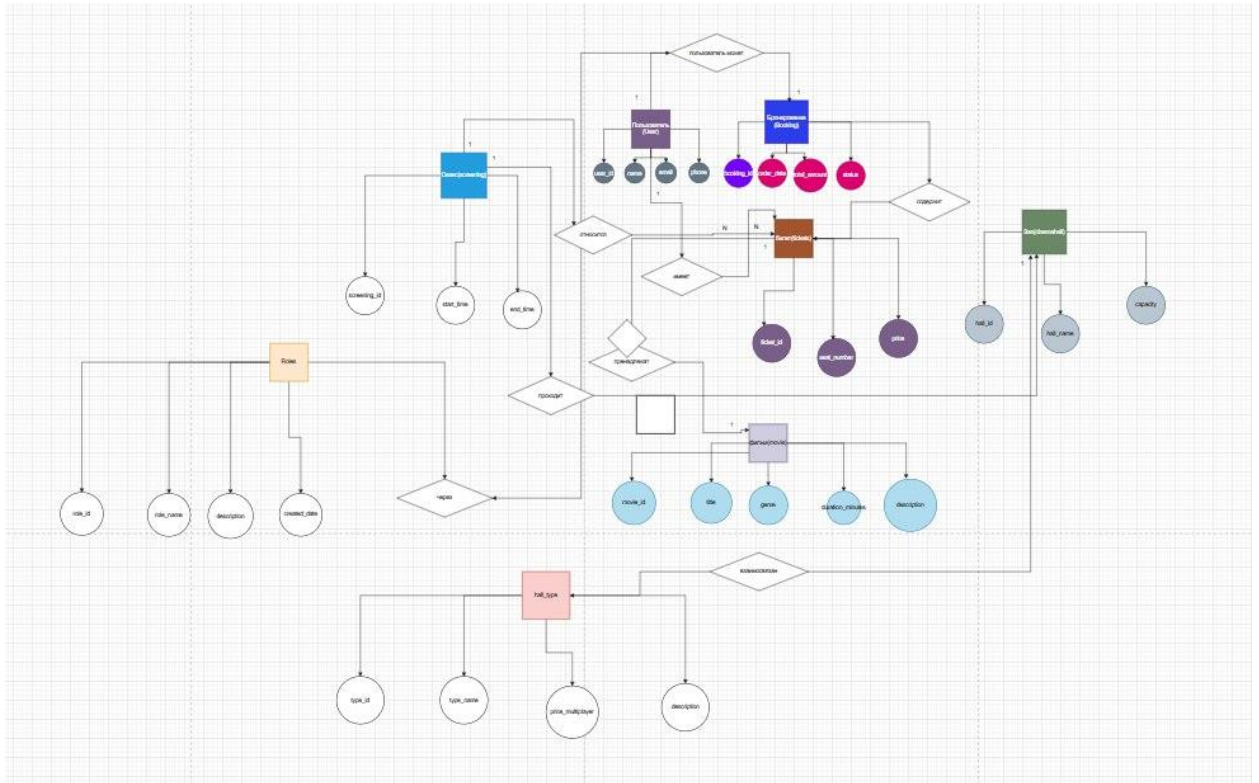


Рис.1 – ER Диаграмма

Описание сущностей:

1. Пользователь (User) Лицо, создающее бронирования.

Атрибуты:

- user_id
- name
- email
- phone
- role_id

Имеет несколько бронирований (1 ко многим). email и phone используются для контактов и уведомлений.

Через бронирования связан с билетами.

2. Бронирование (Booking) Хранит дату заказа, сумму и статус оплаты.

Атрибуты:

- booking_id
- user_id
- screening_id
- booking_date
- total_price
- status

Связано с одним пользователем (1 к 1 у пользователя, 1 ко многим у бронирования).

Содержит много билетов (1 ко многим).

Центральная сущность для покупки билетов.

3. Билет (Ticket) Отвечает за место и цену билета.

Атрибуты:

- ticket_id
- booking_id
- seat_number
- price.

Принадлежит одному бронированию (N к 1).

Относится к одному сеансу (N к 1).

Формирует состав заказа.

4. Сеанс (Screening) Определяет конкретный показ фильма.

Атрибуты:

- screening_id
- movie_id
- hall_id
- start_time
- end_time

Присваивается одному фильму (N к 1).

Проходит в одном зале (N к 1).

Связан с билетами (1 ко многим), определяет время и место просмотра.

5. Зал (Cinema Hall) Организует размещение зрителей.

Атрибуты:

- hall_id
- hall_name
- hall_type

Связан с несколькими сеансами (1 ко многим).

Ограничивает количество и нумерацию мест.

6. Фильм (Movie) Предоставляет информацию для расписания.

Атрибуты:

- movie_id
- title
- description
- duration
- genre
- release_date
- rating

Имеет несколько сеансов (1 ко многим).

Через сеансы связан с билетами и залами.

Связи между сущностями Пользователь —>Бронирование —>Билет —> Сеанс —> Зал
Сеанс относится к Фильму

Зал содержит места для Сеанса и участвует в бронировании через билеты.

7. Роль (Roles) Определяет роли пользователей, например администратор, пользователь и т.п.

Атрибуты:

- role_id
- role_name
- description
- created_date

8. Тип зала (HallType) Определяет тип зала и повышает стоимость

Атрибуты:

- hall_type_id
- type_name
- capacity
- price_per_seat

Нормализация базы данных "Кинотеатр" до 3NF

Исходная ненормализованная таблица

Таблица: CinemaData (до нормализации)

SessionID	MovieTitle	Genre	Duration	HallName	Capacity	DateTime	CustomerName	SeatsBooked	TotalPrice	CustomerPhone	TicketIDs
1	Аватар	Sci Fi	180	Зал 1	100	20240120 14:00	Иван Иванов	3	1050.00	+79161234567	101, 102, 103
1	Аватар	Sci Fi	180	Зал 1	100	20240120 14:00	Петр Петров	2	700.00	+79169876543	104, 105
2	Оппенгеймер	Drama	180	Зал 2	80	20240120 16:30	Мария Сидорова	4	1200.00	+79165554433	201, 202, 203, 204

Проблемы:

- Повторяющиеся данные о фильмах и залах
- Частичные зависимости
- Транзитивные зависимости
- Аномалии обновления, удаления, вставки

Первая нормальная форма (1NF)

Table: User

user_id	name	email	phone
1	Иван Петров	ivan@gmail.com	+79151112233
2	Мария Сидорова	Maria@mail.ru	+79152223344

Table: Booking

booking_id	order_date	total_amount	status
101	20240115	1500.00	confirmed
102	20240116	800.00	pending

Table: Ticket

ticket_id	seat_number	price
1001	A1	500.00
1002	A2	500.00
1003	A3	500.00

Table: Screening

screening_id	start_time	end_time
501	20240120 18:00	20240120 20:30
502	20240120 19:00	20240120 21:00

Table: Movie

movie_id	title	genre	duration_minutes	description
10	Интерстеллар	фантастика	150	Описание фильма...
11	Остров проклятых	триллер	138	Описание фильма...

Table: Cinema_Hall

hall_id	hall_name	capacity	hall_type	type_multiplier
1	Зал 1	100	стандарт	1.00
2	Зал 2	50	VIP	1.50

Вторая нормальная форма (2NF)

Table: User

user_id	name	email	phone
1	Иван Петров	ivan@gmail.com	+79151112233

Table: Booking

booking_id	order_date	total_amount	status
101	20240115	1500.00	confirmed

Table: Ticket

ticket_id	seat_number	price
1001	A1	500.00

Table: Screening

screening_id	start_time	end_time
501	20240120 18:00	20240120 20:30

Table: Movie

movie_id	title	genre	duration_minutes	description
10	Интерстеллар	фантастика	150	Описание фильма...

Table: Hall_Type

type_id	type_name	price_multiplier	description
1	стандарт	1.00	Стандартный кинозал
2	VIP	1.50	Зал повышенной комфортности

Table: Cinema_Hall

hall_id	hall_name	capacity
1	Зал 1	100
2	Зал 2	50

Третья нормальная форма (3NF)

Table: User

user_id	name	email	phone
1	Иван Петров	ivan@gmail.com	+79151112233

Table: Role

role_id	role_name	description	created_date
1	client	Обычный пользователь	20240101
2	admin	Администратор системы	20240101

Table: User_Role

user_id	role_id	assigned_date
1	1	20240115

Table: Booking

booking_id	order_date	total_amount	status	user_id
101	20240115	1500.00	confirmed	1

Table: Ticket

ticket_id	seat_number	price	booking_id	screening_id
1001	A1	500.00	101	501

Table: Screening

screening_id	start_time	movie_id	hall_id	base_price
501	20240120 18:00	10	1	500.00

Table: Movie

movie_id	title	genre	duration_minutes	description
10	Интерстеллар	фантастика	150	Описание фильма...

Table: Hall_Type

type_id	type_name	price_multiplier	description
1	стандарт	1.00	Стандартный кинозал
2	VIP	1.50	Зал повышенной комфортности

Table: Cinema_Hall

hall_id	hall_name	capacity	type_id
1	Зал 1	100	1
2	Зал 2	50	2

Устраненные проблемы:

1. Повторяющиеся данные информация о фильмах, залах, клиентах хранится однократно
2. Аномалии обновления изменение жанра в одном месте
3. Аномалии удаления удаление брони не затрагивает данные о клиентах
4. Аномалии вставки можно добавлять фильмы без сеансов

Описание таблиц

1. User

Назначение: Хранение информации о зарегистрированных пользователях системы.

user_id	INT	IDENTITY(1,1)	PRIMARY KEY
name	NVARCHAR(100)	NOT NULL	
email	NVARCHAR(255)	UNIQUE	NOT NULL
phone	NVARCHAR(20)		

2. Movie

Назначение: Каталог фильмов, доступных для показа в кинотеатре.

movie_id	INT	IDENTITY(1,1)	PRIMARY KEY
title	NVARCHAR(255)	NOT NULL	
genre	NVARCHAR(100)		
duration_minutes	INT		
description	NTEXT		

3. Hall_Type

Назначение: Справочник типов кинозалов с коэффициентами цены.

type_id	INT	IDENTITY(1,1)	PRIMARY KEY
type_name	NVARCHAR(50)	NOT NULL	
price_multiplier	DECIMAL(3,2)	NOT NULL	DEFAULT 1.00
description	NTEXT		

4. Cinema_Hall

Назначение: Информация о физических кинозалах кинотеатра.

hall_id	INT	IDENTITY(1,1)	PRIMARY KEY
hall_name	NVARCHAR(100)	NOT NULL	
capacity	INT	NOT NULL	
type_id	INT		
FOREIGN KEY	(type_id)	REFERENCES	Hall_Type(type_id)

5. Screening

Назначение: Расписание сеансов.

screening_id	INT	IDENTITY(1,1)	PRIMARY KEY		
start_time	DATETIME	NOT NULL			
end_time	DATETIME	NOT NULL			
movie_id	INT				
hall_id	INT				
FOREIGN KEY	(movie_id)	REFERENCES	Movie(movie_id)	ON	DELETE CASCADE
FOREIGN KEY	(hall_id)	REFERENCES	Cinema_Hall(hall_id)	ON	

6. Booking

Назначение: Информация о бронированиях пользователей.

booking_id	INT	IDENTITY(1,1)	PRIMARY KEY	
order_date	DATETIME	NOT NULL		
total_amount	DECIMAL(10,2)	NOT NULL		
status	NVARCHAR(20)	CHECK(status IN ('pending', 'confirmed', 'cancelled'))	DEFAULT 'pending'	
user_id	INT			
FOREIGN KEY	(user_id)	REFERENCES	User	ON DELETE CASCADE

7. Ticket

Назначение: Детальная информация о билетах в рамках бронирования

ticket_id	INT	IDENTITY(1,1)	PRIMARY KEY
seat_number	NVARCHAR(10)	NOT NULL	
price	DECIMAL(10,2)	NOT NULL	
base_price	DECIMAL(10,2)	NOT NULL	
final_price	DECIMAL(10,2)	NOT NULL	
booking_id	INT		
screening_id	INT		
FOREIGN KEY	(booking_id)	REFERENCES	
Booking(booking_id)	ON DELETE CASCADE		
FOREIGN KEY	(screening_id)	REFERENCES	
Screening(screening_id)	ON DELETE CASCADE		

8. Роли

role_id	INT	IDENTITY(1,1)	PRIMARY KEY
role_name	NVARCHAR(50)	NOT NUL	UNIQUE
description	NVARCHAR(200)		
created_date	DATETIME2	DEFAULT	GETDATE()

Индексы

INDEX	idx_user_email	ON	User
INDEX	idx_screening_time	ON	Screening(start_time)
INDEX	idx_booking_date	ON	Booking(order_date)
INDEX	idx_ticket_seat	ON	Ticket(seat_number, screening_id)

Диаграмма

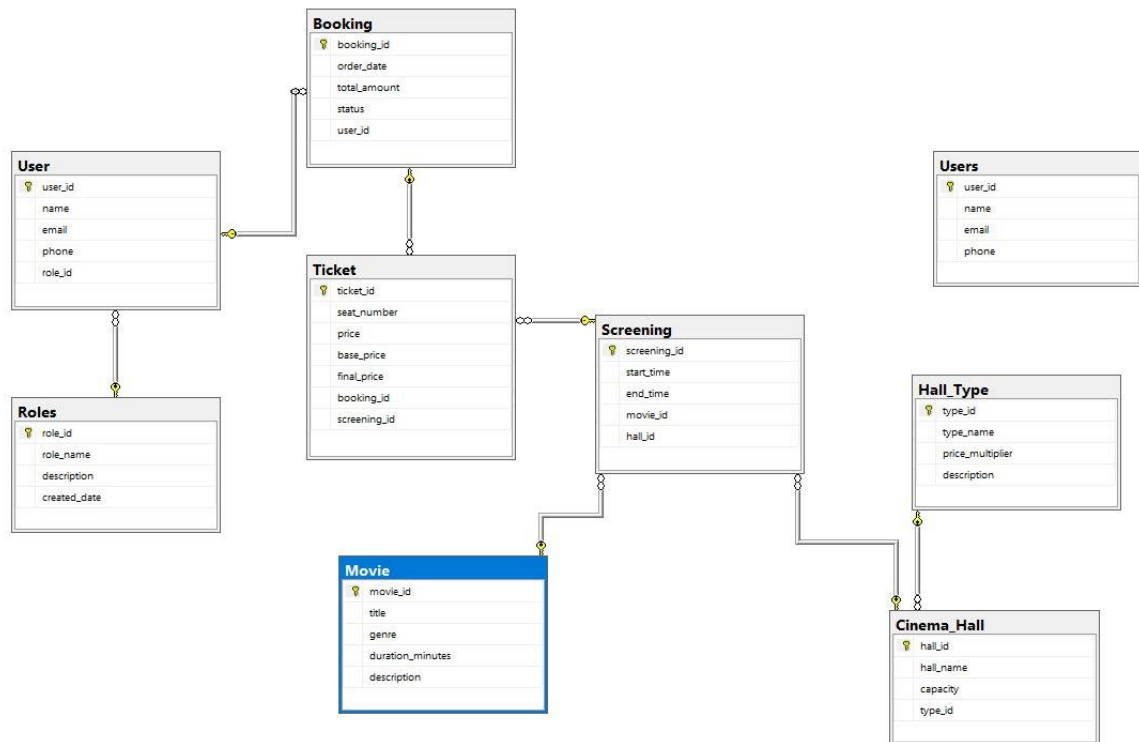


Рис. 2 - Диаграмма

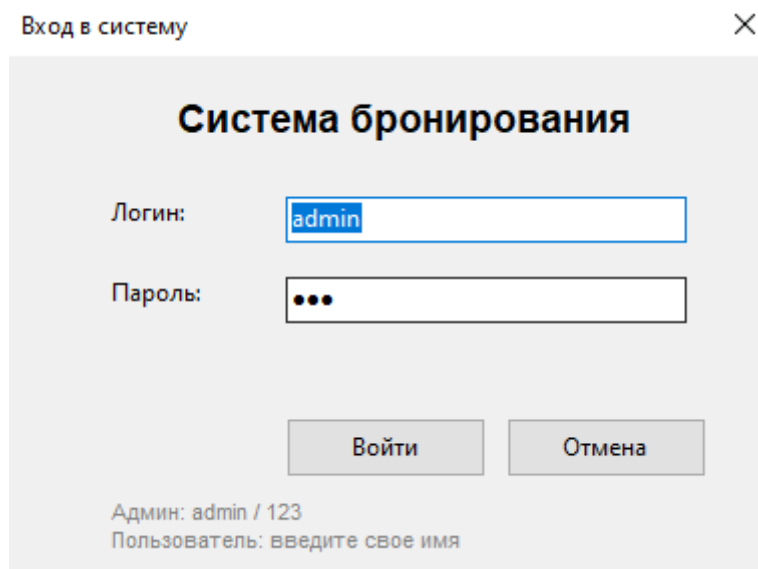
Дополнительные функции:

1. Валидация ввода данных
2. Проверка доступности мест
3. Автоматический расчет стоимости
4. Подтверждение бронирований
5. История бронирований

Интерфейс программы

Интерфейс:

1. Вход



Вход в систему

Система бронирования

Логин:

Пароль:

Админ: admin / 123
Пользователь: введите свое имя

Рис.3 – Вход в систему

Упрощенный вход для пользователей: Администратор: admin / 123 (полный доступ)
Обычный пользователь: любое имя (например: Иван, Мария) - без регистрации

Для администратора: запустите приложение. Введите admin / 123 Получите полный доступ ко всем функциям.

Для обычного пользователя: запустите приложение. Введите любое имя (например: Петр) Получите доступ только к фильмам и бронированиям

Ключевые особенности:

- Простота: Пользователи могут войти без регистрации
- Безопасность: Административные функции полностью скрыты
- Удобство: Интерфейс адаптируется под роль пользователя
- Гибкость: Автоматическое создание временных пользователей

2. Фильмы/Добавление фильмов

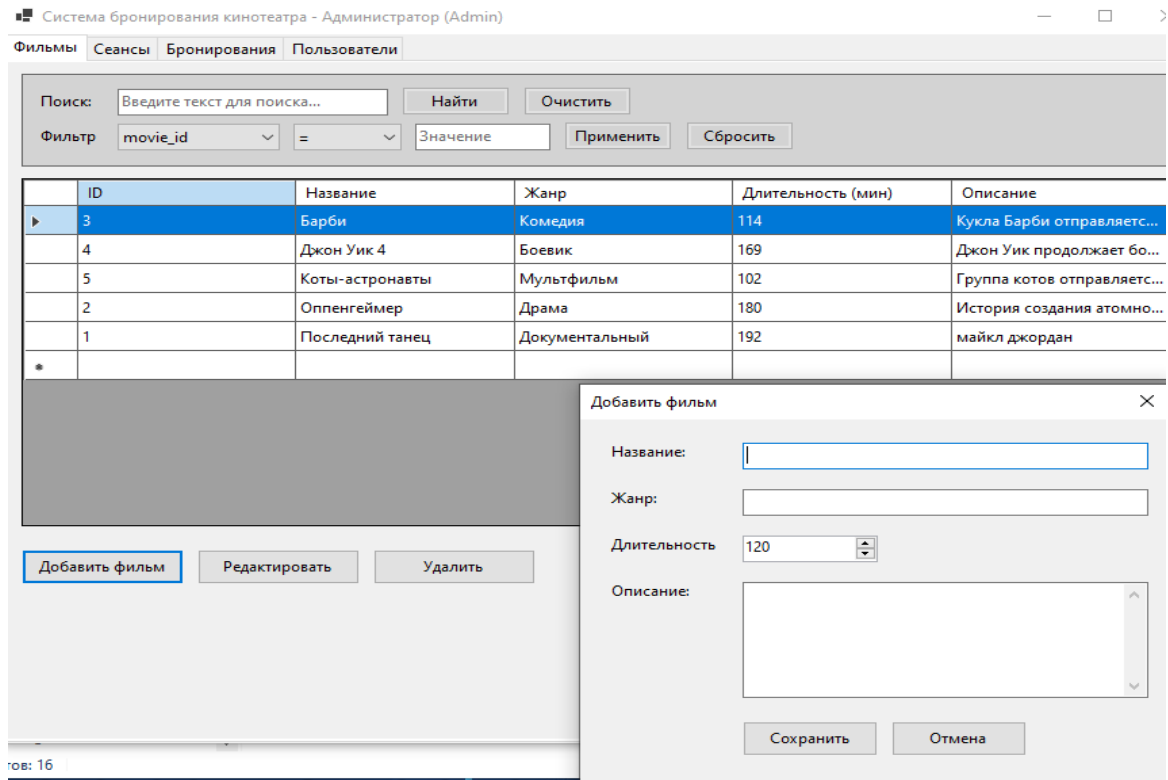


Рис.4 – Добавление фильмов

Просматривать и управлять фильмами:

- Видеть список всех фильмов с ID, названием, жанром, длительностью и описанием
- Добавлять новые фильмы
- Редактировать существующие фильмы
- Удалять фильмы

Работать с данными:

- Искать фильмы через поисковую строку
- Фильтровать фильмы по различным параметрам (например, по movie_id)
- Переходить к управлению сеансами, бронированиями и пользователями через верхнее меню

Интерфейс показывает:

- Таблицу с 6 фильмами разных жанров
- Инструменты для поиска и фильтрации
- Кнопки управления (Добавить/Редактировать/Удалить)

3.Сеансы

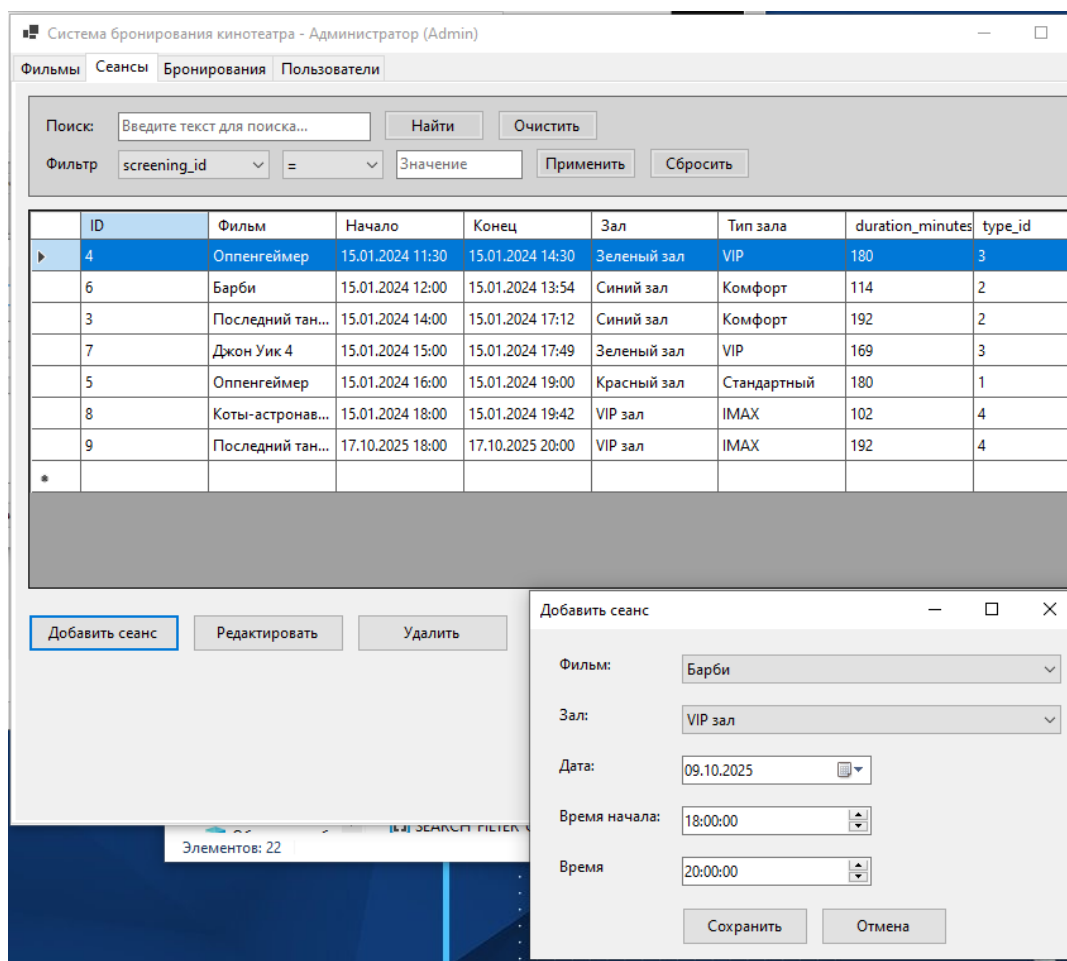


Рис.5 – Сеансы, добавление сеансов

Основные функции:

- Просматривать список всех бронирований в таблице
- Видеть развернутую информацию по первому бронированию
- Создавать новые бронирования для клиентов
- Отменять существующие брони

Создание нового бронирования:

- Выбор пользователя из системы
- Выбор конкретного сеанса (фильма)
- Указание количества билетов
- Автоматический расчет общей стоимости

4. Добавление Пользователей

Система бронирования кинотеатра - Администратор (Admin)

Фильмы Сеансы Бронирования Пользователи

Поиск:

Фильтр:

	ID	Имя	Email	Телефон
▶	7	Администратор	admin@cinema.com	+7 (999) 123-45-67
	3	Алексей Козлов	alex.kozlov@yandex.ru	+79219876543
	5	Дмитрий Соколов	dmitry.sokolov@gmail.com	+79165556677
	4	Елена Волкова	elena.volkova@mail.ru	+7 (123) 123-12-41
	8	Иван Иванов	fafa@asd.cpm	+7 (111) 111-11-11
	9	Иван Иванов	fafa123@asd.cpm	+7 (111) 111-11-11
	1	Иван Петров	ivan.petrov@mail.ru	+79161234567
	2	Мария Сидорова	maria.sidorova@gmail.com	+79037654321
	6	петр	dad1@wqewe.dasda	+7 (895) 177-82-45
*				

Добавить пользователя

Имя:

Email:

Телефон:

Элементов: 22

Рис.6 – Добавление пользователей

Просмотр пользователей:

- Видит список всех зарегистрированных пользователей
- В таблице отображается ID, имя, email и телефон

Управление пользователями:

- Добавление новых пользователей
- Редактирование данных существующих пользователей
- Удаление пользователей

Особенности интерфейса:

Форма добавления пользователя:

- Поля для ввода имени, email и телефона
- Маска для телефона

5.Создание бронирования

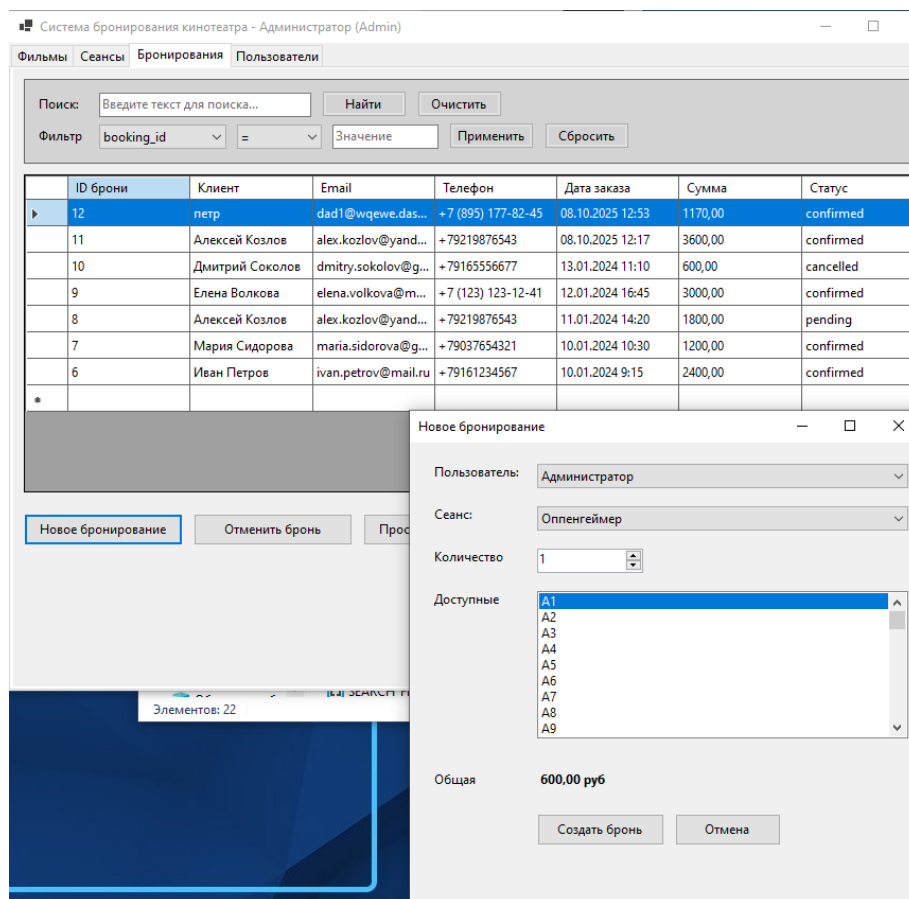


Рис.7 – Создание бронирования

Просмотр бронирований:

- Таблица со всеми бронированиями
- Отображение ID, клиента, контактов, даты, суммы и статуса
- Разные статусы: confirmed, pending, cancelled

Управление бронированиями:

- Создание новых бронирований
- Отмена существующих броней
- Просмотр деталей бронирований

Поиск и фильтрация:

- Поиск по любым данным в таблице
- Фильтрация по ID бронирования

Создание нового бронирования:

Выбор мест:

- Показывает доступные места в зале (A1-A9)
- Автоматический расчет общей суммы (600 руб)

Тестирование системы

Тест-кейс 1: Вход администратора в систему

Идентификатор	ТС_01
Описание	Функция предназначена для авторизации администратора в системе с полным доступом к функционалу
Вводные данные	Имя пользователя Пароль
Обработка	Ввести имя пользователя Ввести пароль Нажать кнопку входа
Выходные данные	<ul style="list-style-type: none">• Пользователь получает полный доступ• Отображается интерфейс администратора с верхним меню
Сообщения об ошибках	"Введено неправильное имя пользователя или пароль"
Требования к функционалу	Может использовать только пользователь с правами администратора

Тест-кейс 2: Вход обычного пользователя без регистрации

Идентификатор	ТС_02
Описание	Функция предназначена для упрощенного входа обычного пользователя без процедуры регистрации
Вводные данные	Имя пользователя – любое произвольное имя Пароль – не требуется
Обработка	Ввести любое имя пользователя Нажать кнопку входа
Выходные данные	<ul style="list-style-type: none">• Пользователь получает доступ только к функциям просмотра фильмов и бронирования билетов• Административные функции скрыты от пользователя• Система автоматически создает временного пользователя
Сообщения об ошибках	Сообщения об ошибке не ожидается
Требования к функционалу	Может использовать любой пользователь без предварительной регистрации

Тест-кейс 3: Добавление нового фильма администратором

Идентификатор	ТС_03
Описание	Функция предназначена для добавления нового фильма в каталог кинотеатра администратором системы
Вводные данные	Название фильма – обязательное поле Жанр – опциональное поле Длительность – опциональное поле Описание – опциональное поле
Обработка	Авторизоваться как администратор Перейти в раздел "Фильмы" Нажать кнопку "Добавить фильм" Заполнить обязательные и опциональные поля данными Подтвердить добавление
Выходные данные	<ul style="list-style-type: none">• Новый фильм появляется в таблице фильмов с присвоенным movie_id• Фильм становится доступен для создания сеансов• Данные сохраняются в таблице Movie базы данных
Сообщения об ошибках	<ul style="list-style-type: none">• "Название фильма не может быть пустым"• "Ошибка при сохранении данных"
Требования к функционалу	Может использовать только пользователь с ролью "Администратор"

Вывод:

В ходе работы над проектом мы отработали навыки пользования SQL Server Management и научились работать с ADO.NET, в частности, с Dapper.

Достигнута главная цель проекта разработана полнофункциональная система бронирования реализованы все основные операции CRUD

Выполненные задачи:

1. Создана нормализованная база данных соответствие 3NF обеспечено для всех таблиц
2. Реализован удобный пользовательский интерфейс интуитивная навигация и валидация ввода
3. Обеспечена целостность данных внешние ключи, проверочные ограничения, уникальные индексы
4. Проведено комплексное тестирование успешная проверка всех сценариев использования

Покрытие основных бизнес-процессов:

- Анализ клиентской базы и покупательской активности
- Управление жизненным циклом бронирований (статусы: confirmed, pending, cancelled)
- Мониторинг финансовых показателей (общие расходы, средний чек)
- Персонализация работы с клиентами через пользовательские функции
- Отслеживание временных тенденций (анализ за последний месяц)

Разработанная система успешно решает поставленные задачи и представляет собой готовое к использованию решение для автоматизации процессов бронирования в кинотеатре.

Проект демонстрирует высокий уровень владения современными технологиями разработки и может служить прочной основой для дальнейшего профессионального роста в области создания программного обеспечения.

Приложение с кодом:

```
using System;           // Базовые типы и классы .NET Framework
using System.Data;      // Классы для работы с данными (DataTable, DataSet)
using System.Drawing;   // Классы для работы с графикой и цветами
using System.Linq;      // LINQ для работы с коллекциями
using System.Windows.Forms; // Классы для создания Windows Forms приложений
using CinemaBookingApp.Data; // Наши классы для работы с базой данных

namespace CinemaBookingApp.Forms
{
    /// <summary>
    /// Форма для создания нового бронирования билетов в кинотеатре
    /// Позволяет выбрать пользователя, сеанс, места и количество билетов
    /// Поддерживает разные режимы для администраторов и обычных пользователей
    /// </summary>
    public partial class BookingForm : Form
    {
        // =====
        // ПОЛЯ ДЛЯ РАБОТЫ С БАЗОЙ ДАННЫХ
        // =====

        /// <summary>
        /// Менеджер базы данных для выполнения операций с бронированиями
        /// </summary>
        private DataBaseManager dbManager;

        /// <summary>
        /// Строка подключения к базе данных SQL Server
        /// </summary>
        private string connectionString = "Server=192.168.9.203\\SQLEXPRESS;Database=Проект Вакула, Белов, Сухинин;User Id=student1;Password=123456;TrustServerCertificate=true;";

        // =====
        // ИНФОРМАЦИЯ О ТЕКУЩЕМ ПОЛЬЗОВАТЕЛЕ
        // =====
    }
}
```

```

/// <summary>
/// Имя текущего пользователя для ограничения доступа
/// </summary>
private string? currentUser_name;

/// <summary>
/// Флаг, указывающий является ли текущий пользователь администратором
/// </summary>
private bool isCurrentUserAdmin;

// =====
// ЭЛЕМЕНТЫ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА
// =====

/// <summary>
/// Выпадающий список для выбора пользователя (только для администраторов)
/// </summary>
private ComboBox comboUsers = null!;

/// <summary>
/// Выпадающий список для выбора сеанса
/// </summary>
private ComboBox comboScreenings = null!;

/// <summary>
/// Список доступных мест в зале
/// </summary>
private ListBox lstSeats = null!;

/// <summary>
/// Числовое поле для выбора количества билетов
/// </summary>
private NumericUpDown numTickets = null!;

/// <summary>
/// Метка для отображения общей стоимости бронирования

```

```

/// </summary>
private Label lblTotalPrice = null!;

/// <summary>
/// Кнопка "Создать бронирование" для сохранения заказа
/// </summary>
private Button btnCreateBooking = null!;

/// <summary>
/// Кнопка "Отмена" для закрытия формы без сохранения
/// </summary>
private Button btnCancel = null!;

// =====
// ПОЛЯ ДЛЯ РАСЧЕТА СТОИМОСТИ
// =====

/// <summary>
/// Базовая цена одного билета в рублях
/// </summary>
private decimal basePrice = 300m;

/// <summary>
/// ID текущего выбранного сеанса
/// </summary>
private int currentScreeningId = -1;

/// <summary>
/// Конструктор формы бронирования
/// Инициализирует форму с учетом роли и имени пользователя
/// </summary>
/// <param name="currentUserName">Имя текущего пользователя</param>
/// <param name="isCurrentUserAdmin">Флаг администратора</param>
public BookingForm(string? currentUserName = null, bool isCurrentUserAdmin = false)
{
    // Сохраняем информацию о текущем пользователе
    this.currentUserName = currentUserName;    // Устанавливаем имя пользователя

```

```

this.isCurrentUserAdmin = isCurrentUserAdmin; // Устанавливаем флаг администратора

// Инициализируем форму
InitializeComponent(); // Создаем элементы интерфейса
dbManager = new DataBaseManager(connectionString); // Инициализируем менеджер БД
LoadComboBoxData(); // Загружаем данные для выпадающих списков
}

private void InitializeComponent()
{
    // Устанавливаем заголовок в зависимости от роли пользователя
    if (isCurrentUserAdmin)
    {
        this.Text = "Новое бронирование";
    }
    else
    {
        this.Text = $"Новое бронирование - {currentUser.Name}";
    }

    this.Size = new Size(500, 500);
    this.StartPosition = FormStartPosition.CenterParent;
    this.FormBorderStyle = FormBorderStyle.FixedDialog;

    // Пользователь
    var lblUser = new Label();
    if (isCurrentUserAdmin)
    {
        lblUser.Text = "Пользователь:";
    }
    else
    {
        lblUser.Text = "Пользователь (ваше имя):";
    }
    lblUser.Location = new Point(20, 20);
    lblUser.Size = new Size(100, 20);
}

```

```

comboUsers = new ComboBox();
comboUsers.Location = new Point(120, 20);
comboUsers.Size = new Size(350, 20);
comboUsers.DropDownStyle = ComboBoxStyle.DropDownList;

// Сеанс
var lblScreening = new Label();
lblScreening.Text = "Сеанс:";
lblScreening.Location = new Point(20, 60);
lblScreening.Size = new Size(100, 20);

comboScreenings = new ComboBox();
comboScreenings.Location = new Point(120, 60);
comboScreenings.Size = new Size(350, 20);
comboScreenings.DropDownStyle = ComboBoxStyle.DropDownList;
comboScreenings.SelectedIndexChanged += ComboScreenings_SelectedIndexChanged;

// Количество билетов
var lblTickets = new Label();
lblTickets.Text = "Количество билетов:";
lblTickets.Location = new Point(20, 100);
lblTickets.Size = new Size(100, 20);

numTickets = new NumericUpDown();
numTickets.Location = new Point(120, 100);
numTickets.Size = new Size(100, 20);
numTickets.Minimum = 1;
numTickets.Maximum = 10;
numTickets.Value = 1;
numTickets.ValueChanged += NumTickets_ValueChanged;

// Доступные места
var lblSeats = new Label();
lblSeats.Text = "Доступные места:";
lblSeats.Location = new Point(20, 140);
lblSeats.Size = new Size(100, 20);

```

```

lstSeats = new ListBox();
lstSeats.Location = new Point(120, 140);
lstSeats.Size = new Size(350, 150);
lstSeats.SelectionMode = SelectionMode.MultiSimple;

// Общая стоимость
var lblTotal = new Label();
lblTotal.Text = "Общая стоимость:";
lblTotal.Location = new Point(20, 310);
lblTotal.Size = new Size(100, 20);

lblTotalPrice = new Label();
lblTotalPrice.Text = "0 руб";
lblTotalPrice.Location = new Point(120, 310);
lblTotalPrice.Size = new Size(200, 20);
lblTotalPrice.Font = new Font(lblTotalPrice.Font, FontStyle.Bold);

// Кнопки
btnCreateBooking = new Button();
btnCreateBooking.Text = "Создать бронь";
btnCreateBooking.Location = new Point(120, 350);
btnCreateBooking.Size = new Size(120, 30);
btnCreateBooking.Click += BtnCreateBooking_Click;

btnCancel = new Button();
btnCancel.Text = "Отмена";
btnCancel.Location = new Point(250, 350);
btnCancel.Size = new Size(100, 30);
btnCancel.Click += BtnCancel_Click;

this.Controls.AddRange(new Control[] {
    lblUser, comboUsers,
    lblScreening, comboScreenings,
    lblTickets, numTickets,
    lblSeats, lstSeats,
    lblTotal, lblTotalPrice,
    btnCreateBooking, btnCancel
});

```

```

});

UpdateTotalPrice();
}

private void LoadComboBoxData()
{
    try
    {
        // Загрузка пользователей
        if (isCurrentUserAdmin)
        {
            // Администратор видит всех пользователей
            var users = dbManager.GetUsers();
            if (users.Rows.Count == 0)
            {
                MessageBox.Show("В базе данных нет пользователей. Сначала добавьте пользователей.");
                return;
            }

            comboUsers.DisplayMember = "name";
            comboUsers.ValueMember = "user_id";
            comboUsers.DataSource = users;
        }
        else
        {
            // Обычный пользователь видит только себя
            if (!string.IsNullOrEmpty(currentUserName))
            {
                var userId = dbManager.GetUserIdByName(currentUserName);
                if (userId.HasValue)
                {
                    // Создаем DataTable с одним пользователем
                    var userTable = new DataTable();
                    userTable.Columns.Add("user_id", typeof(int));
                    userTable.Columns.Add("name", typeof(string));
                }
            }
        }
    }
}

```



```

        var row = userTable.NewRow();
        row["user_id"] = userId.Value;
        row["name"] = currentUser.Name;
        userTable.Rows.Add(row);

        comboUsers.DisplayMember = "name";
        comboUsers.ValueMember = "user_id";
        comboUsers.DataSource = userTable;

        // Делаем ComboBox недоступным для изменения
        comboUsers.Enabled = false;
    }
    else
    {
        MessageBox.Show("Пользователь не найден в базе данных.", "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        this.Close();
        return;
    }
}
else
{
    MessageBox.Show("Не удалось определить текущего пользователя.", "Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    this.Close();
    return;
}
}

// Загрузка сеансов - показываем все сеансы, не только будущие
var screenings = dbManager.GetScreenings();
if (screenings.Rows.Count == 0)
{
    MessageBox.Show("В базе данных нет сеансов. Сначала добавьте сеансы.");
    return;
}

```

```

        // Показываем все сеансы без фильтрации по дате
        comboScreenings.DisplayMember = "title";
        comboScreenings.ValueMember = "screening_id";
        comboScreenings.DataSource = screenings;
    }

    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка загрузки данных: {ex.Message}\n\nПроверьте подключение к базе
данных и наличие данных в таблицах.");
    }
}

private void ComboScreenings_SelectedIndexChanged(object? sender, EventArgs e)
{
    if (comboScreenings.SelectedValue != null)
    {
        currentScreeningId = Convert.ToInt32(comboScreenings.SelectedValue);
        LoadAvailableSeats();
        UpdateTotalPrice();
    }
}

private void NumTickets_ValueChanged(object? sender, EventArgs e)
{
    UpdateTotalPrice();
}

private void LoadAvailableSeats()
{
    if (currentScreeningId == -1) return;

    try
    {
        var screening = dbManager.GetScreeningById(currentScreeningId);
        if (screening == null)
        {
            MessageBox.Show("Не удалось загрузить информацию о сеансе.");
            return;
        }
    }
}

```

```

    }

    var hallCapacity = Convert.ToInt32(screening["capacity"]);
    var occupiedSeats = dbManager.GetOccupiedSeats(currentScreeningId);

    lstSeats.Items.Clear();

    // Генерируем доступные места (ряды A-Z, места 1-20)
    for (char row = 'A'; row <= 'Z'; row++)
    {
        for (int seatNum = 1; seatNum <= 20; seatNum++)
        {
            var seat = $"{row}{seatNum}";
            if (!occupiedSeats.Contains(seat))
            {
                lstSeats.Items.Add(seat);
            }
        }
    }

    if (lstSeats.Items.Count == 0)
    {
        MessageBox.Show("Нет доступных мест для данного сеанса.");
        return;
    }

    // Автоматически выбираем первые N мест
    var ticketsCount = (int)numTickets.Value;
    for (int i = 0; i < Math.Min(ticketsCount, lstSeats.Items.Count); i++)
    {
        lstSeats.SetSelected(i, true);
    }
}

catch (Exception ex)
{
    MessageBox.Show($"Ошибка загрузки мест: {ex.Message}");
}

```

```

    }

    private void UpdateTotalPrice()
    {
        if (currentScreeningId == -1)
        {
            lblTotalPrice.Text = "0 руб";
            return;
        }

        try
        {
            var screening = dbManager.GetScreeningById(currentScreeningId);
            if (screening != null)
            {
                var hallTypeId = Convert.ToInt32(screening["type_id"]);
                var ticketPrice = dbManager.CalculateTicketPrice(basePrice, hallTypeId);
                var total = ticketPrice * (int)numTickets.Value;
                lblTotalPrice.Text = $"{total:F2} руб";
            }
            else
            {
                lblTotalPrice.Text = "0 руб";
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка расчета стоимости: {ex.Message}");
            lblTotalPrice.Text = "Ошибка";
        }
    }

    private void BtnCreateBooking_Click(object? sender, EventArgs e)
    {
        if (comboUsers.SelectedValue == null || comboScreenings.SelectedValue == null)
        {
            MessageBox.Show("Выберите пользователя и сеанс");
        }
    }

```

```

        return;
    }

    var selectedSeats = lstSeats.SelectedItems;
    if (selectedSeats.Count != numTickets.Value)
    {
        MessageBox.Show($"Выберите ровно {numTickets.Value} мест(a)");
        return;
    }

    if (selectedSeats.Count == 0)
    {
        MessageBox.Show("Выберите места для бронирования");
        return;
    }

    try
    {
        var userId = Convert.ToInt32(comboUsers.SelectedValue);
        var screeningId = Convert.ToInt32(comboScreenings.SelectedValue);

        var screening = dbManager.GetScreeningById(screeningId);
        if (screening == null)
        {
            MessageBox.Show("Не удалось загрузить информацию о сеансе");
            return;
        }

        var hallTypeId = Convert.ToInt32(screening["type_id"]);
        var ticketPrice = dbManager.CalculateTicketPrice(basePrice, hallTypeId);
        var totalAmount = ticketPrice * (int)numTickets.Value;

        // Создаем бронирование
        var bookingId = dbManager.CreateBooking(userId, totalAmount, "confirmed");

        // Создаем билеты для каждого выбранного места
        foreach (var seat in selectedSeats)

```

```

    {
        if (seat != null)
        {
            dbManager.AddTicket(
                seat.ToString(),
                basePrice,
                ticketPrice,
                bookingId,
                screeningId
            );
        }
    }

    MessageBox.Show($"Бронирование успешно создано!\nID бронирования: {bookingId}\nОбщая
сумма: {totalAmount:F2} руб");

    this.DialogResult = DialogResult.OK;
    this.Close();
}

catch (Exception ex)
{
    MessageBox.Show($"Ошибка создания бронирования: {ex.Message}\n\nПроверьте подключение к
базе данных и корректность данных.");
}

}

private void BtnCancel_Click(object? sender, EventArgs e)
{
    this.DialogResult = DialogResult.Cancel;
    this.Close();
}
}
}

```