**12. Многопользовательское приложение**

Задание 1. Система бронирования отеля

Листинг задачи:

using HotelBookingApp.Models;

namespace HotelBookingApp.Services

{

public class BookingService

{

private readonly List<BookingModel> \_activeBookings = new List<BookingModel>();

private int \_nextBookingId = 1;

public async Task<BookingModel> BookRoomAsync(RoomModel room, string guestName, DateTime checkIn, DateTime checkOut)

{

if (room == null)

{

throw new ArgumentNullException(nameof(room), "Room cannot be null.");

}

if (room.Status != RoomStatus.Available)

{

throw new InvalidOperationException($"Room {room.RoomNumber} is not available for booking.");

}

if (string.IsNullOrWhiteSpace(guestName))

{

throw new ArgumentException("Guest name cannot be empty.", nameof(guestName));

}

if (checkOut.Date <= checkIn.Date)

{

throw new ArgumentException("Check-out date must be after check-in date.");

}

await Task.Delay(2000);

var newBooking = new BookingModel

{

BookingId = \_nextBookingId++,

RoomId = room.Id,

BookedRoom = room,

GuestName = guestName,

CheckInDate = checkIn.Date,

CheckOutDate = checkOut.Date

};

\_activeBookings.Add(newBooking);

room.Status = RoomStatus.Occupied;

Console.WriteLine($"Booking confirmed: Id={newBooking.BookingId}, Room={room.RoomNumber}, Guest={guestName}, From={newBooking.CheckInDate:d} To={newBooking.CheckOutDate:d}");

return newBooking;

}

public async Task<bool> CancelBookingAsync(BookingModel bookingToCancel)

{

if (bookingToCancel == null) return false;

var booking = \_activeBookings.FirstOrDefault(b => b.BookingId == bookingToCancel.BookingId);

if (booking != null)

{

await Task.Delay(1000);

if (booking.BookedRoom != null)

{

booking.BookedRoom.Status = RoomStatus.Available;

}

\_activeBookings.Remove(booking);

Console.WriteLine($"Booking cancelled: Id={booking.BookingId}");

return true;

}

Console.WriteLine($"Booking cancellation failed: Booking Id={bookingToCancel.BookingId} not found.");

return false;

}

public BookingModel? GetBookingForRoom(RoomModel room)

{

if (room == null) return null;

return \_activeBookings.FirstOrDefault(b => b.RoomId == room.Id);

}

public async Task<bool> EditBookingAsync(BookingModel updatedBooking)

{

if (updatedBooking == null) return false;

if (string.IsNullOrWhiteSpace(updatedBooking.GuestName))

{

throw new ArgumentException("Guest name cannot be empty for update.", nameof(updatedBooking.GuestName));

}

if (updatedBooking.CheckOutDate.Date <= updatedBooking.CheckInDate.Date)

{

throw new ArgumentException("Check-out date must be after check-in date for update.");

}

await Task.Delay(1500);

var existing = \_activeBookings.FirstOrDefault(b => b.BookingId == updatedBooking.BookingId);

if (existing != null)

{

existing.GuestName = updatedBooking.GuestName;

existing.CheckInDate = updatedBooking.CheckInDate.Date;

existing.CheckOutDate = updatedBooking.CheckOutDate.Date;

Console.WriteLine($"Booking updated: Id={existing.BookingId}, Guest={existing.GuestName}, From={existing.CheckInDate:d} To={existing.CheckOutDate:d}");

return true;

}

Console.WriteLine($"Booking update failed: Booking Id={updatedBooking.BookingId} not found.");

return false;

}

public IEnumerable<BookingModel> GetAllActiveBookings()

{

return \_activeBookings.ToList();

}

}

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Данные о номерах, ФИО, даты бронирования | Подтверждение о бронировании, Сообщение об ошибке, Обновление списка |

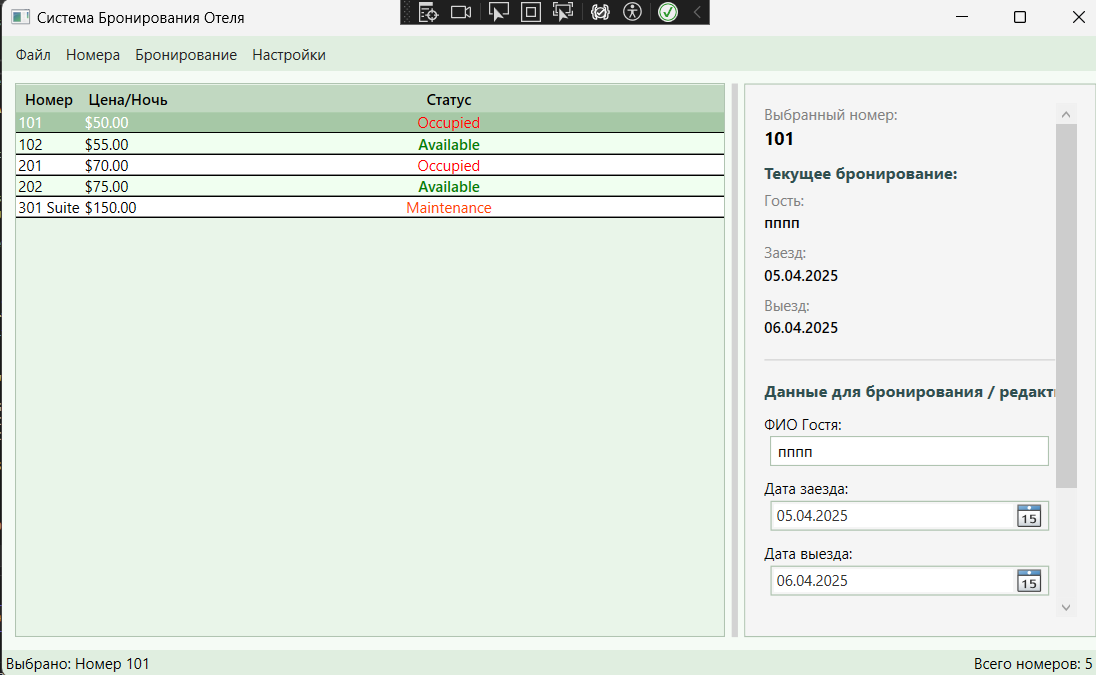
Анализ результатов:

Рисунок 1.1 – Результат работы программы