**14 Технологии создания графического пользовательского интерфейса**

Задание 1. Система бронирования отеля

Листинг задачи:

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Runtime.CompilerServices;

using System.Windows;

using System.Windows.Input;

using WpfApp1.Commands;

using WpfApp1.Models;

using WpfApp1.Views;

namespace WpfApp1.ViewModels

{

public class MainWindowViewModel : INotifyPropertyChanged

{

private ObservableCollection<Room> \_rooms = new ObservableCollection<Room>();

public ObservableCollection<Room> Rooms

{

get => \_rooms;

set => SetField(ref \_rooms, value);

}

private ObservableCollection<Booking> \_bookings = new ObservableCollection<Booking>();

public ObservableCollection<Booking> Bookings

{

get => \_bookings;

set { SetField(ref \_bookings, value); OnPropertyChanged(nameof(BookingCount)); }

}

private Room? \_selectedRoom;

public Room? SelectedRoom

{

get => \_selectedRoom;

set => SetField(ref \_selectedRoom, value);

}

private Booking? \_selectedBooking;

public Booking? SelectedBooking

{

get => \_selectedBooking;

set

{

if (SetField(ref \_selectedBooking, value))

{

(EditBookingCommand as RelayCommand)?.RaiseCanExecuteChanged();

(CancelBookingCommand as RelayCommand)?.RaiseCanExecuteChanged();

StatusMessage = \_selectedBooking != null ? "Бронь выбрана" : "Готово";

}

}

}

private string? \_statusMessage;

public string? StatusMessage

{

get => \_statusMessage;

set => SetField(ref \_statusMessage, value);

}

public int BookingCount => Bookings?.Count ?? 0;

private int \_nextBookingId = 1;

public ICommand BookRoomCommand { get; }

public ICommand EditBookingCommand { get; }

public ICommand CancelBookingCommand { get; }

public ICommand ExitCommand { get; }

public MainWindowViewModel()

{

LoadRoomsData();

LoadBookingsData();

BookRoomCommand = new RelayCommand(ExecuteBookRoomDialog);

EditBookingCommand = new RelayCommand(ExecuteEditBooking, CanExecuteEditOrCancelBooking);

CancelBookingCommand = new RelayCommand(ExecuteCancelBooking, CanExecuteEditOrCancelBooking);

ExitCommand = new RelayCommand(ExecuteExit);

StatusMessage = "Система готова.";

}

private void ExecuteBookRoomDialog(object? parameter)

{

var dialogWindow = new BookingDialogWindow();

var dialogViewModel = new BookingDialogViewModel(dialogWindow, newBooking =>

{

Room? roomToBook = Rooms.FirstOrDefault(r => r.RoomNumber == newBooking.RoomNumber);

if (roomToBook == null)

{

MessageBox.Show($"Номер комнаты '{newBooking.RoomNumber}' не найден.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

throw new InvalidOperationException("Room not found during save callback.");

}

if (roomToBook.Status != RoomStatus.Available)

{

MessageBox.Show($"Номер комнаты '{newBooking.RoomNumber}' занят или недоступен.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

throw new InvalidOperationException("Room not available during save callback.");

}

newBooking.Id = \_nextBookingId++;

Bookings.Add(newBooking);

roomToBook.Status = RoomStatus.Occupied;

SelectedBooking = newBooking;

StatusMessage = $"Добавлена новая бронь: №{newBooking.Id}";

OnPropertyChanged(nameof(BookingCount));

}, isEditing: false);

dialogWindow.DataContext = dialogViewModel;

dialogWindow.Owner = Application.Current.MainWindow;

try

{

var result = dialogWindow.ShowDialog();

StatusMessage = result == true ? "Операция сохранения завершена." : "Добавление брони отменено.";

}

catch (InvalidOperationException ex)

{

StatusMessage = $"Ошибка: {ex.Message}";

}

}

private void ExecuteEditBooking(object? parameter)

{

if (SelectedBooking == null) return;

var dialogWindow = new BookingDialogWindow();

var originalBooking = SelectedBooking;

var bookingToEditCopy = new Booking

{

Id = originalBooking.Id,

ClientName = originalBooking.ClientName,

RoomNumber = originalBooking.RoomNumber,

StartDate = originalBooking.StartDate,

EndDate = originalBooking.EndDate

};

var dialogViewModel = new BookingDialogViewModel(dialogWindow, bookingToEditCopy, updatedBooking =>

{

originalBooking.ClientName = updatedBooking.ClientName;

originalBooking.StartDate = updatedBooking.StartDate;

originalBooking.EndDate = updatedBooking.EndDate;

StatusMessage = $"Бронь №{originalBooking.Id} изменена.";

}, isEditing: true);

dialogWindow.DataContext = dialogViewModel;

dialogWindow.Owner = Application.Current.MainWindow;

var result = dialogWindow.ShowDialog();

StatusMessage = result == true ? $"Бронь №{originalBooking.Id} сохранена." : $"Редактирование брони №{originalBooking.Id} отменено.";

}

private void ExecuteCancelBooking(object? parameter)

{

if (SelectedBooking == null) return;

var result = MessageBox.Show($"Вы уверены, что хотите отменить бронь?\n\n{SelectedBooking.BookingSummary}",

"Подтверждение отмены", MessageBoxButton.YesNo, MessageBoxImage.Warning);

if (result == MessageBoxResult.Yes)

{

int cancelledId = SelectedBooking.Id;

string roomNumberOfCancelledBooking = SelectedBooking.RoomNumber;

Bookings.Remove(SelectedBooking);

Room? roomToUpdate = Rooms.FirstOrDefault(r => r.RoomNumber == roomNumberOfCancelledBooking);

if (roomToUpdate != null)

{

if (roomToUpdate.Status == RoomStatus.Occupied)

{

roomToUpdate.Status = RoomStatus.Available;

}

}

else

{

StatusMessage = $"Внимание: Номер {roomNumberOfCancelledBooking} отмененной брони не найден!";

}

OnPropertyChanged(nameof(BookingCount));

StatusMessage = $"Бронь №{cancelledId} отменена.";

}

else

{

StatusMessage = "Отмена брони прервана.";

}

}

private void ExecuteExit(object? parameter)

{

Application.Current.Shutdown();

}

private bool CanExecuteEditOrCancelBooking(object? parameter)

{

return SelectedBooking != null;

}

private void LoadRoomsData()

{

Rooms.Clear();

Rooms.Add(new Room { RoomNumber = "101", RoomType = "Стандарт", Price = 2500m, Status = RoomStatus.Available });

Rooms.Add(new Room { RoomNumber = "102", RoomType = "Стандарт", Price = 2500m, Status = RoomStatus.Available });

Rooms.Add(new Room { RoomNumber = "201", RoomType = "Комфорт", Price = 3500m, Status = RoomStatus.Available });

Rooms.Add(new Room { RoomNumber = "202", RoomType = "Комфорт", Price = 3500m, Status = RoomStatus.Available });

Rooms.Add(new Room { RoomNumber = "301", RoomType = "Люкс", Price = 5000m, Status = RoomStatus.Maintenance });

Rooms.Add(new Room { RoomNumber = "302", RoomType = "Люкс", Price = 5200m, Status = RoomStatus.Available });

}

private void LoadBookingsData()

{

Bookings.Clear();

var booking1 = new Booking { Id = 1, ClientName = "Иванов И.И.", RoomNumber = "101", StartDate = DateTime.Today.AddDays(1), EndDate = DateTime.Today.AddDays(3) };

var booking2 = new Booking { Id = 2, ClientName = "Петрова А.С.", RoomNumber = "202", StartDate = DateTime.Today.AddDays(5), EndDate = DateTime.Today.AddDays(8) };

Bookings.Add(booking1);

Bookings.Add(booking2);

\_nextBookingId = 3;

UpdateRoomStatusBasedOnBooking(booking1.RoomNumber, true);

UpdateRoomStatusBasedOnBooking(booking2.RoomNumber, true);

OnPropertyChanged(nameof(BookingCount));

}

private void UpdateRoomStatusBasedOnBooking(string roomNumber, bool isOccupied)

{

Room? room = Rooms.FirstOrDefault(r => r.RoomNumber == roomNumber);

if (room != null && room.Status != RoomStatus.Maintenance)

{

room.Status = isOccupied ? RoomStatus.Occupied : RoomStatus.Available;

}

}

public event PropertyChangedEventHandler? PropertyChanged;

protected virtual void OnPropertyChanged([CallerMemberName] string? propertyName = null)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

protected bool SetField<T>(ref T field, T value, [CallerMemberName] string? propertyName = null)

{

if (EqualityComparer<T>.Default.Equals(field, value)) return false;

field = value;

OnPropertyChanged(propertyName);

return true;

}

}

}

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 |

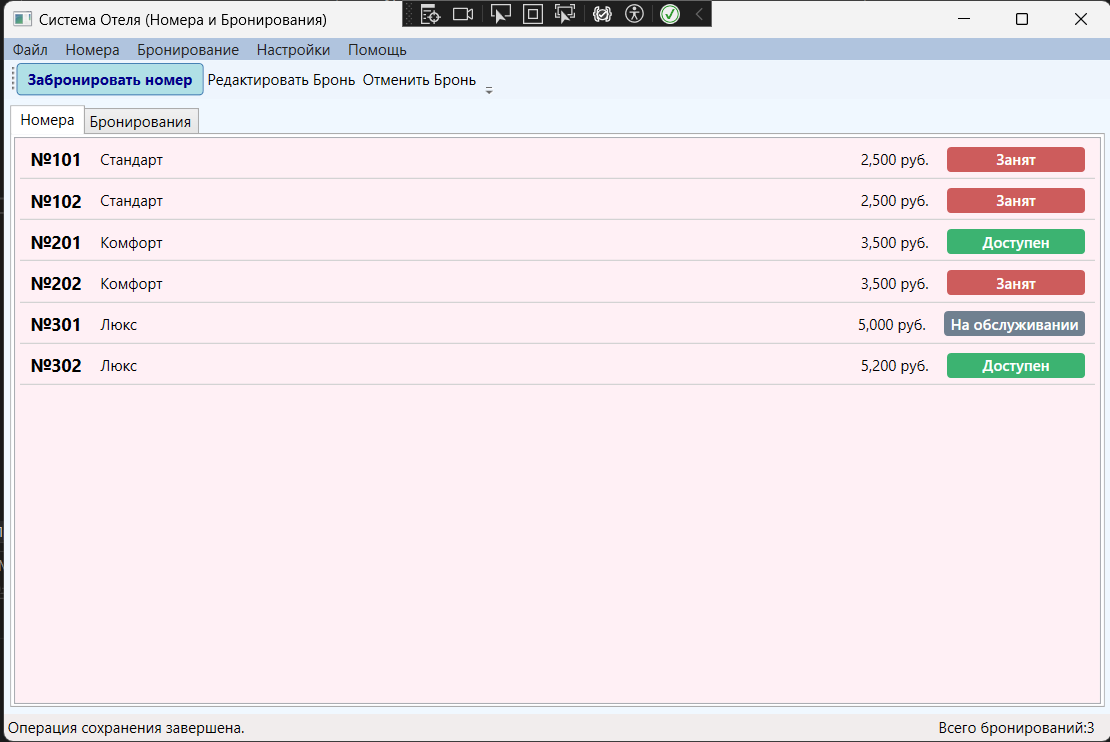
Анализ результатов:

Рисунок 1.1 – Результат работы программы