МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

И ПРОГРАММНОЙ ИНЖЕНЕРИИ (КАФЕДРА 43)

|  |  |  |
| --- | --- | --- |
| ОТЧЕТ ЗАЩИЩЕН С ОЦЕНКОЙ: |  |  |

ПРЕПОДАВАТЕЛЬ:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| старший преподаватель | / |  | / |  | / | Е. О. Шумова |
| (должность, учёная степень, звание) |  | (подпись) |  | (дата защиты) |  | (инициалы, фамилия) |

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

«ОПИСАНИЕ КЛАССОВ И ПОРОЖДЕНИЕ ОБЪЕКТОВ»

ПО КУРСУ: «ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ»

|  |  |  |  |
| --- | --- | --- | --- |
| РАБОТУ ВЫПОЛНИЛ (-А) СТУДЕНТ (-КА): | Z1431 |  | М.Д. Быстров |
|  | (номер группы) |  | (инициалы, фамилия) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | / |  | / | 10.01.2024 |
|  |  | (подпись студента) |  | (дата отчета) |

ВВЕДЕНИЕ

**Цель лабораторной работы:**

Научиться на практике применять паттерны проектирования.

**Задание:**

Проанализировать предметную область в соответствии с выбранной темой: «Разработка иерархии классов для обеспечения работы гостиницы».

Спроектировать иерархию классов в соответствии с произведенным анализом.

Создать программу, обрабатывающую данные в соответствии с сущностями предметной области. При разработке приложения должны быть использованы паттерны проектирования.

Привести диаграммы классов, спроектированных в ходе выполнения лабораторной работы.

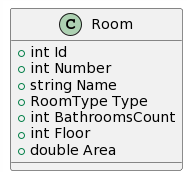
1. Выделенные сущности предметной области

1. Класс «Room»

Класс «Room» описывает номер – сущность предметной области. Каждый экземпляр класса описывает один номер гостиницы.

Описание полей:

* Id – уникальный идентификатор
* Number – числовой код номера, в соответствии с распределением номеров в гостинице
* Name – название номера
* Type – тип (класс) номера
* BathroomsCount – количество ванных комнат в номере
* Floor – номер этажа, на котором располагается номер
* Area – площадь номера (м2)



*Рисунок 1 Класс Room*

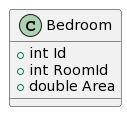
Все поля являются свойствами языка C#, т.е. имеют неявные методы доступа (getter, setter), инкапсулирующие поля. Далее во всех описаниях классов подразумевается, что доступ к данным, содержащимся в полях экземпляров классов, происходит посредством неявных вызовов методов доступа.

2. Класс «Bedroom»

Класс «Bedroom» описывает спальню, принадлежащую номеру.

Описание полей:

* Id – уникальный идентификатор комнаты
* RoomId – идентификатор комнаты, которой принадлежит спальня
* Area – площадь спальни



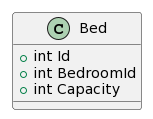
*Рисунок 2 Класс Bedroom*

3. Класс «Bed»

Класс «Bed» описывает кровать, расположенную в спальне номера.

Описание полей:

* Id – уникальный идентификатор кровати
* BedroomId – идентификатор спальни, в которой расположена кровать
* Capacity – вместимость кровати (количество человек)



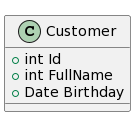
*Рисунок 3 Класс Bed*

4. Класс «Customer»

Класс «Customer» описывает гостя гостиницы (постоялец, заказчик).

Описание полей:

* Id – уникальный идентификатор гостя
* FullName – ФИО
* Birthday – дата рождения



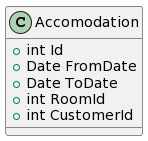
*Рисунок 4 Класс Customer*

5. Класс «Accomodation»

Класс «Accomodation» описывает заселение постояльца в номер.

Описание полей:

* Id – уникальный идентификатор заселения
* FromDate – дата заселения
* ToDate – дата выезда
* RoomId – ИД комнаты
* CustomerId – ИД постояльца

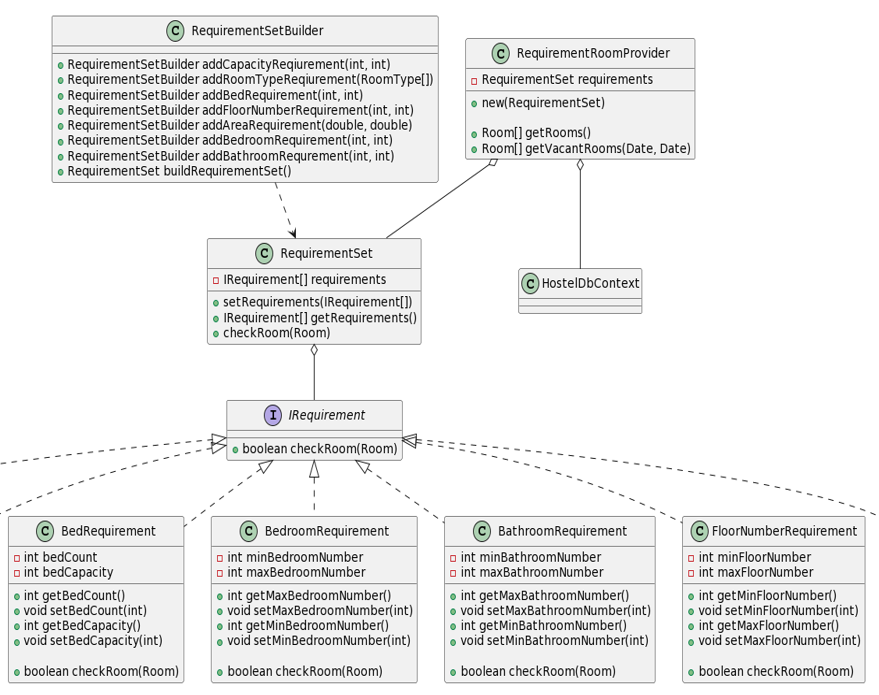


*Рисунок 5 Класс Accomodation*

1. Диаграммы классов

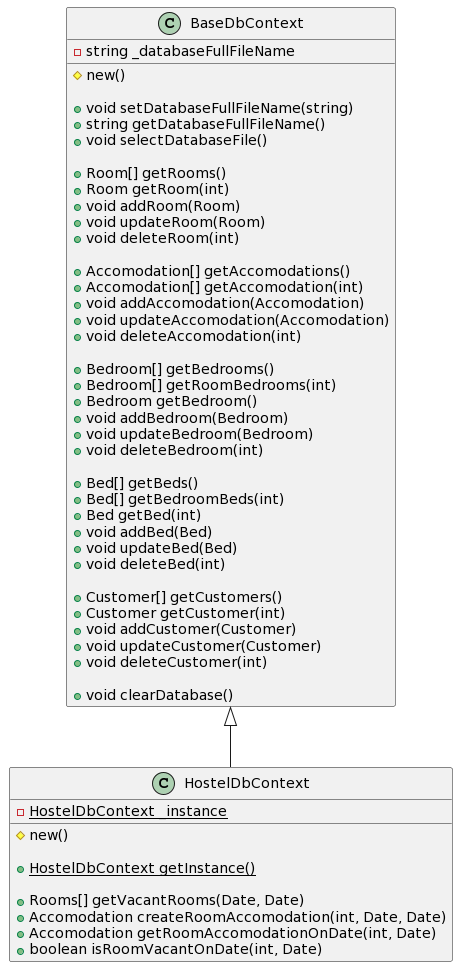
На рисунке 6 представлена диаграмма классов для паттерна «Строитель» (“Builder”), который применен в разработанном приложении.

Паттерн «Строитель» позволяет упростить создание комплексных объектов, прямое создание которых требует большого количества вызовов функций и объявления локальных переменных, либо вызова конструкторов с большим количеством аргументов. При использовании паттерна невозможно получить объект, чье состояние ввиду ошибки не полностью приведено в соответствие с требуемым, поскольку результат предоставляется только в самом конце работы со «Строителем».



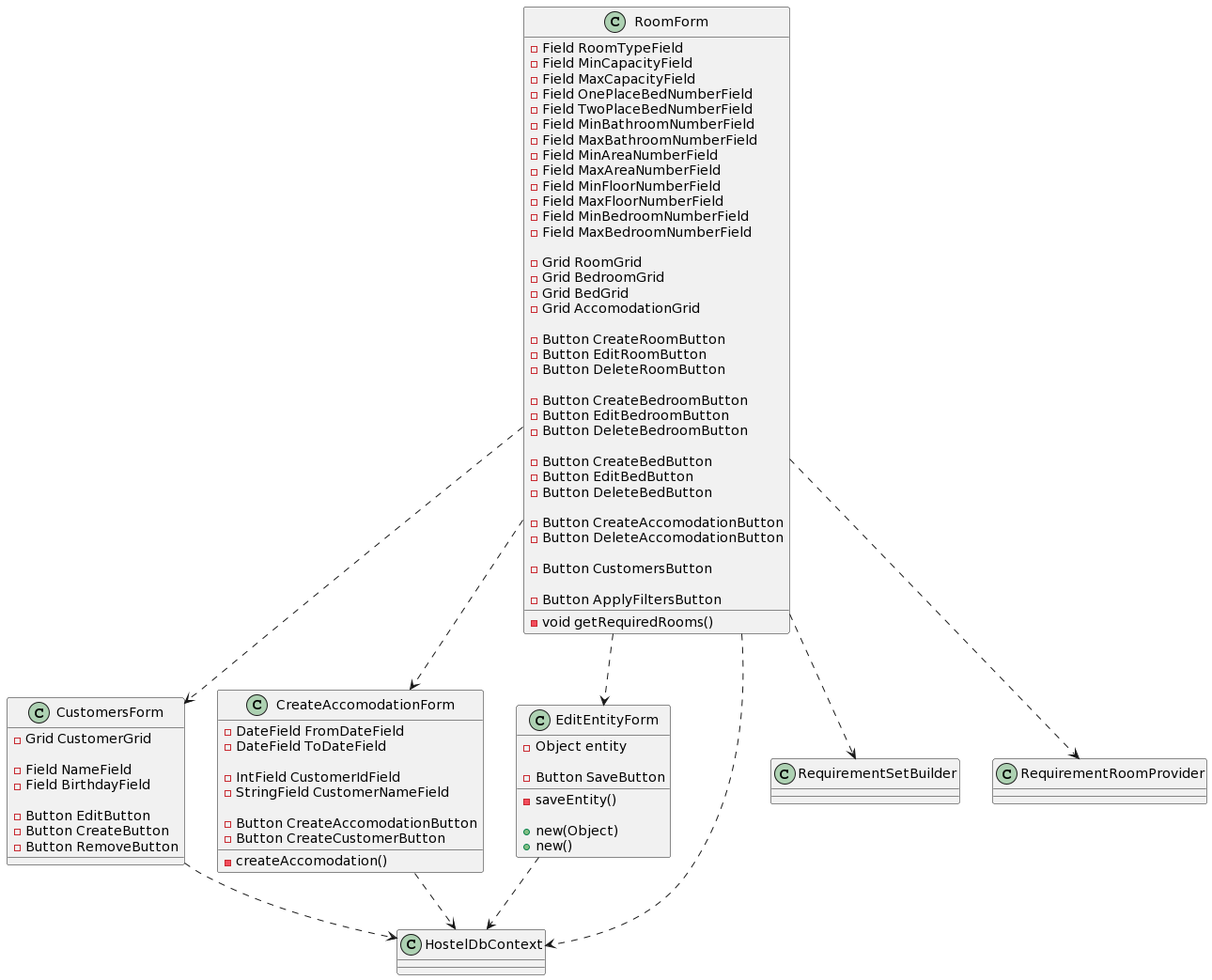
*Рисунок 6 Диаграмма классов для паттерна «Строитель»*

На рисунке 7 представлена диаграмма классов для паттерна «Singleton». В разрабатываемом приложении паттерн реализован в классе-наследнике класса, предоставляющего интерфейс для работы с данными.



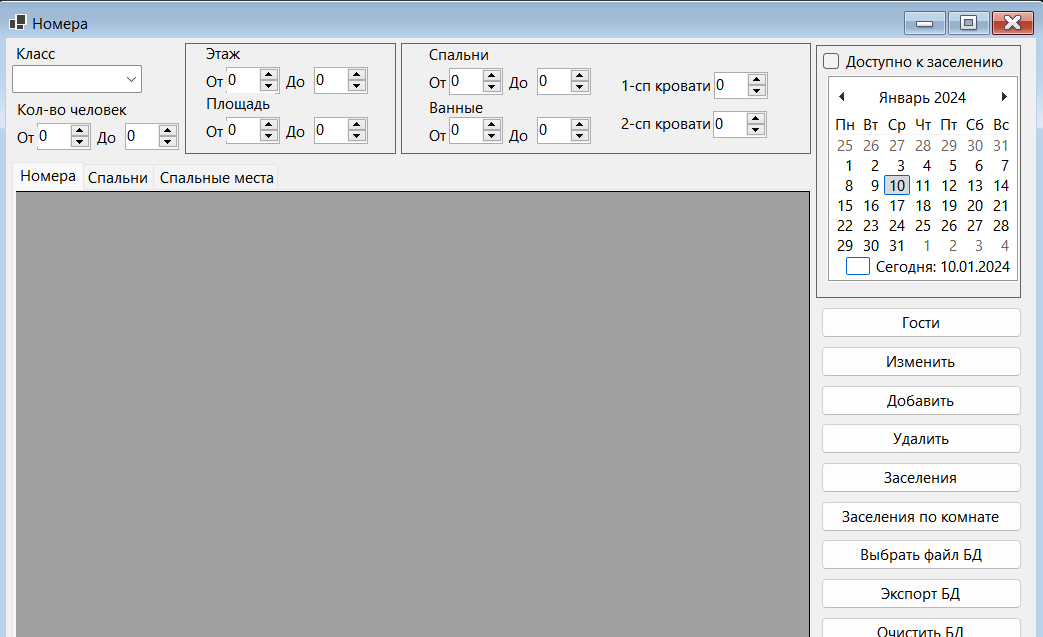
*Рисунок 7 Диаграмма классов паттерна «Singleton»*

На рисунке 8 показана главная диаграмма классов приложения. Используется графический пользовательский интерфейс.



*Рисунок 8 Главная диаграмма классов приложения*

1. Скриншоты всех разработанных программных форм в режиме дизайнера с указанием программных имен всех виджетов
2. Программная форма «RoomForm» - главное окно приложения.



*Рисунок 9 – Главное окно в режиме дизайнера*

Использованные виджеты:

DataGridView RoomGrid;

ComboBox RoomTypeField;

NumericUpDown MinCapacityField;

NumericUpDown MaxCapacityField;

NumericUpDown OnePlaceBedNumberField;

NumericUpDown TwoPlaceBedNumberField;

NumericUpDown MaxFloorNumberField;

NumericUpDown MinFloorNumberField;

NumericUpDown MaxAreaField;

NumericUpDown MinAreaField;

NumericUpDown MaxBathroomNumberField;

NumericUpDown MinBathroomNumberField;

NumericUpDown MaxBedroomNumberField;

NumericUpDown MinBedroomNumberField;

CheckBox IsOnlyVacantField;

MonthCalendar VacantCalendar;

TabControl TabControl;

TabPage RoomTab;

TabPage BedroomTab;

TabPage BedTab;

DataGridView BedroomGrid;

DataGridView BedGrid;

Button CustomersButton;

StatusStrip statusStrip1;

ToolStripStatusLabel CurrentCustomerLabel;

Button EditButton;

Button AddButton;

Button AccomodationButton;

Button RoomAccomodationButton;

Button SelectDbButton;

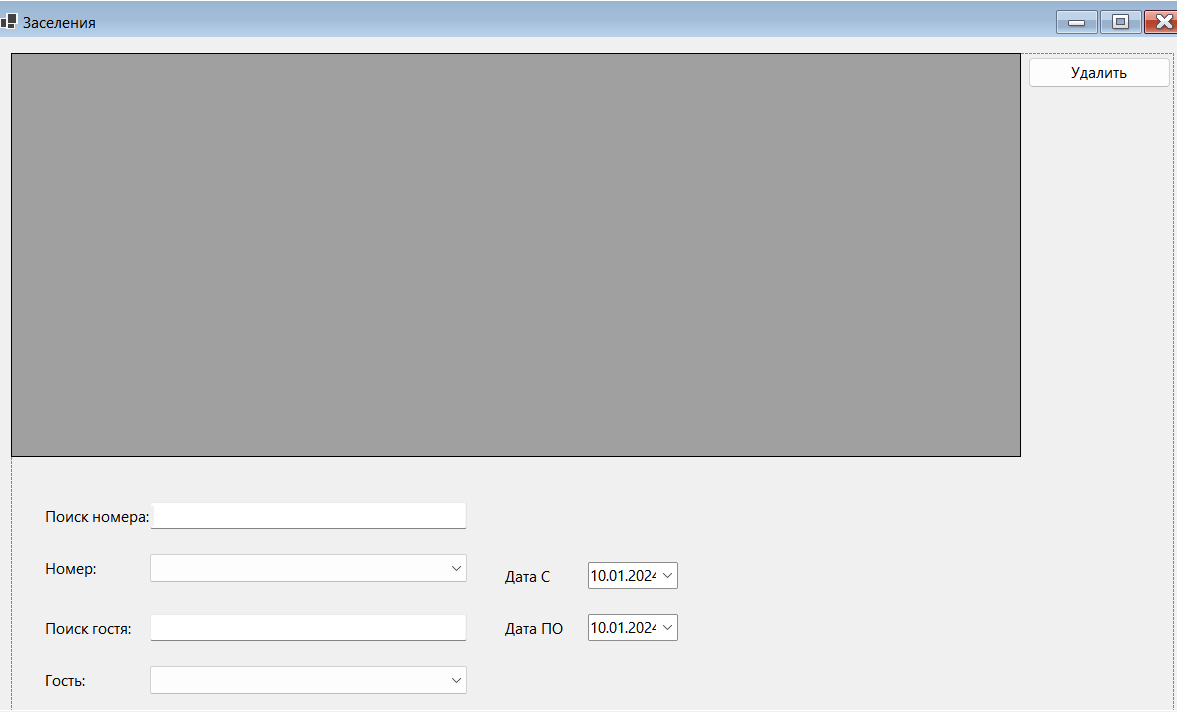
Button SaveDbButton;

Button ClearDbButton;

Button TestBdButton;

Button DeleteButton;

1. Программная форма «AccomodationForm»



*Рисунок 10 – Форма заселений*

Использованные виджеты:

DataGridView AccomodationGrid;

TextBox SearchNameField;

ComboBox CustomerComboBox;

DateTimePicker ToDatePicker;

DateTimePicker FromDatePicker;

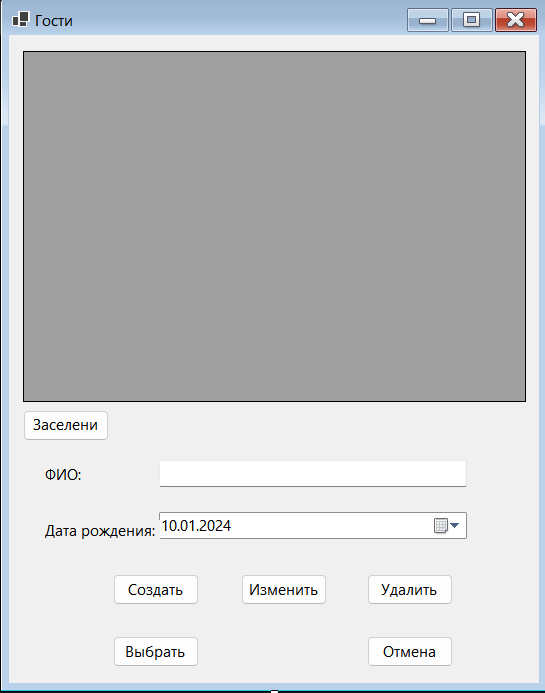
Button CreateAccomodationButton;

ComboBox RoomComboBox;

TextBox SearchRoomField;

Button DeleteButton;

1. Программная форма «CustomersForm»



*Рисунок 11 – Форма с данными гостей гостиницы*

Использованные виджеты:

DataGridView CustomersGrid;

DateTimePicker BirthdayPicker;

TextBox NameField;

Button CreateButton;

Button EditButton;

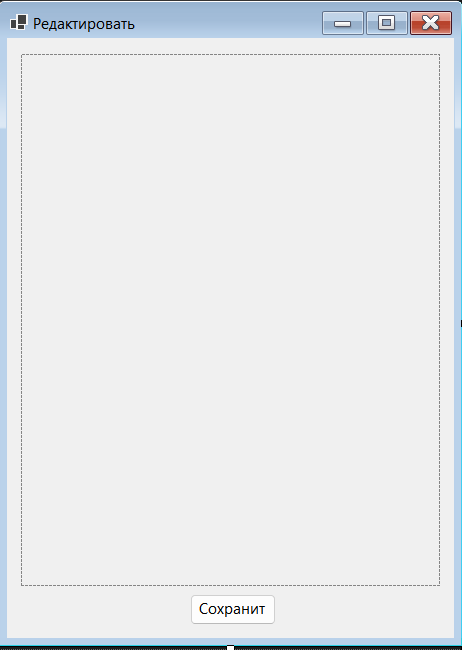
Button RemoveButton;

Button CancelSelectionButton;

Button SelectButton;

Button AccomodationButton;

1. Программная форма «EditEntityForm»



*Рисунок 12 Окно редактирования сущности.*

Использованные виджеты:

FlowLayoutPanel MainContentLayoutPanel;

Button SaveButton;

Форма редактирования создает элементы для редактирования данных динамически при запуске, в зависимости от свойств объекта, переданного на редактирование.

1. Текст (листинг) программы с комментариями

1. ./AccomodationForm.cs

﻿**using** HostelApp.Entities;

**using** HostelApp.Persistence;

**using** System.Data;

**using** System.Diagnostics.Eventing.Reader;

**namespace** HostelApp

{

**public** **partial** **class** AccomodationForm : Form

{

**private** **readonly** Customer? \_defaultCustomer;

**private** **readonly** Customer? \_filterCustomer;

**private** **readonly** Room? \_defaultRoom;

**private** **readonly** Room? \_filterRoom;

**public** AccomodationForm(

Customer? defaultCustomer = **null**,

Customer? filterCustomer = **null**,

Room? defaultRoom = **null**,

Room? filterRoom = **null**)

{

InitializeComponent();

\_defaultCustomer = defaultCustomer;

\_filterCustomer = filterCustomer;

\_defaultRoom = defaultRoom;

\_filterRoom = filterRoom;

**if** (\_filterRoom != **null**)

{

SearchRoomField.Enabled = **false**;

RoomComboBox.Enabled = **false**;

\_defaultRoom = \_filterRoom;

}

**if** (\_filterCustomer != **null**)

{

SearchNameField.Enabled = **false**;

CustomerComboBox.Enabled = **false**;

\_defaultCustomer = \_filterCustomer;

}

SetCustomersComboBoxSource();

SetRoomsComboBoxSource();

ExecuteAccomodationQuery();

}

**private** **void** ExecuteAccomodationQuery()

{

**var** accomodations = HostelDbContext

.GetInstance()

.GetAccomodationsAsync()

.GetAwaiter()

.GetResult()

.**Where**(acc => (acc.CustomerId == \_filterCustomer?.Id

|| \_filterCustomer == **null**)

&& (acc.RoomId == \_filterRoom?.Id

|| \_filterRoom == **null**))

.ToList();

AccomodationGrid.DataSource = accomodations;

}

**private** **void** SearchNameField\_TextChanged(**object** sender, EventArgs e)

{

SetCustomersComboBoxSource();

}

**private** **void** SetCustomersComboBoxSource()

{

**var** searchText = SearchNameField.Text.ToLower();

**var** customers = HostelDbContext

.GetInstance()

.GetCustomersAsync()

.GetAwaiter()

.GetResult()

.**Where**(c => c.FullName.ToLower().Contains(searchText)

|| c.Id.ToString().ToLower().Contains(searchText)

|| **string**.IsNullOrWhiteSpace(searchText))

.ToList();

CustomerComboBox.Items.Clear();

CustomerComboBox.Items.AddRange(customers.ToArray());

**if** (**string**.IsNullOrWhiteSpace(searchText)

&& \_defaultCustomer != **null**

&& CustomerComboBox.Items.Contains(\_defaultCustomer))

{

CustomerComboBox.SelectedItem = \_defaultCustomer;

}

}

**private** **void** SetRoomsComboBoxSource()

{

**var** searchText = SearchRoomField.Text.ToLower();

**var** rooms = HostelDbContext

.GetInstance()

.GetRoomsAsync()

.GetAwaiter()

.GetResult()

.**Where**(c => c.Name.ToLower().Contains(searchText)

|| c.Number.ToString().ToLower().Contains(searchText)

|| c.RoomType.ToString().ToLower().Contains(searchText)

|| c.Id.ToString().ToLower().Contains(searchText)

|| **string**.IsNullOrWhiteSpace(searchText))

.ToList();

RoomComboBox.Items.Clear();

RoomComboBox.Items.AddRange(rooms.ToArray());

**if** (**string**.IsNullOrWhiteSpace(searchText)

&& \_defaultRoom != **null**

&& RoomComboBox.Items.Contains(\_defaultRoom))

{

RoomComboBox.SelectedItem = \_defaultRoom;

}

}

**private** **void** RoomSearchField\_TextChanged(**object** sender, EventArgs e)

{

SetRoomsComboBoxSource();

}

**private** **void** CreateAccomodationButton\_Click(**object** sender, EventArgs e)

{

**try**

{

CreateAccomodation();

}

**catch** (Exception ex)

{

MessageBox.Show(**this**, ex.Message, "Ошибка");

}

}

**private** **void** CreateAccomodation()

{

**var** room = RoomComboBox.SelectedItem **as** Room

?? **throw** new NullReferenceException("Номер не выбран");

**var** customer = CustomerComboBox.SelectedItem **as** Customer

?? **throw** new NullReferenceException("Гость не выбран");

**var** fromDate = FromDatePicker.**Value**;

**var** toDate = ToDatePicker.**Value**;

**using** **var** session = HostelDbContext.GetInstance().BeginSession();

**var** acc = HostelDbContext

.GetInstance()

.CreateRoomAccomodationAsync(

room.Id,

fromDate,

toDate,

customer.Id)

.GetAwaiter()

.GetResult();

ExecuteAccomodationQuery();

**foreach** (DataGridViewRow item **in** AccomodationGrid.Rows)

{

**if** (item.DataBoundItem is Accomodation gridAcc

&& gridAcc.Id == acc.Id)

{

AccomodationGrid.ClearSelection();

AccomodationGrid.CurrentCell = item.Cells[0];

item.Cells[0].Selected = **true**;

**break**;

}

}

}

**private** **void** DeleteButton\_Click(**object** sender, EventArgs e)

{

**try**

{

DeleteAccomodation();

}

**catch** (Exception ex)

{

MessageBox.Show(**this**, ex.Message, "Ошибка");

}

}

**private** **void** DeleteAccomodation()

{

**var** result = MessageBox.Show(

**this**,

"Удалить заселение?",

"Внимание",

MessageBoxButtons.OKCancel);

**if** (result != DialogResult.OK)

{

**return**;

}

**var** acc = AccomodationGrid.CurrentRow.DataBoundItem **as** Accomodation

?? **throw** new NullReferenceException("Заселение не выбрано");

**using** **var** session = HostelDbContext.GetInstance().BeginSession();

HostelDbContext.GetInstance().DeleteAccomodationAsync(acc.Id).GetAwaiter().GetResult();

ExecuteAccomodationQuery();

}

}

}

2. ./AccomodationForm.Designer.cs

﻿**namespace** HostelApp

{

**partial** **class** AccomodationForm

{

*/// <summary>*

*/// Required designer variable.*

*/// </summary>*

**private** System.ComponentModel.IContainer components = **null**;

*/// <summary>*

*/// Clean up any resources being used.*

*/// </summary>*

*/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>*

**protected** **override** **void** Dispose(**bool** disposing)

{

**if** (disposing && (components != **null**))

{

components.Dispose();

}

**base**.Dispose(disposing);

}

#region Windows Form Designer generated code

*/// <summary>*

*/// Required method for Designer support - do not modify*

*/// the contents of this method with the code editor.*

*/// </summary>*

**private** **void** InitializeComponent()

{

panel1 = new Panel();

DeleteButton = new Button();

label5 = new Label();

label6 = new Label();

RoomComboBox = new ComboBox();

SearchRoomField = new TextBox();

CreateAccomodationButton = new Button();

label4 = new Label();

label3 = new Label();

ToDatePicker = new DateTimePicker();

FromDatePicker = new DateTimePicker();

label2 = new Label();

label1 = new Label();

CustomerComboBox = new ComboBox();

SearchNameField = new TextBox();

AccomodationGrid = new DataGridView();

panel1.SuspendLayout();

((System.ComponentModel.ISupportInitialize)AccomodationGrid).BeginInit();

SuspendLayout();

*//*

*// panel1*

*//*

panel1.Anchor = AnchorStyles.Top | AnchorStyles.Bottom | AnchorStyles.Left | AnchorStyles.Right;

panel1.Controls.**Add**(DeleteButton);

panel1.Controls.**Add**(label5);

panel1.Controls.**Add**(label6);

panel1.Controls.**Add**(RoomComboBox);

panel1.Controls.**Add**(SearchRoomField);

panel1.Controls.**Add**(CreateAccomodationButton);

panel1.Controls.**Add**(label4);

panel1.Controls.**Add**(label3);

panel1.Controls.**Add**(ToDatePicker);

panel1.Controls.**Add**(FromDatePicker);

panel1.Controls.**Add**(label2);

panel1.Controls.**Add**(label1);

panel1.Controls.**Add**(CustomerComboBox);

panel1.Controls.**Add**(SearchNameField);

panel1.Controls.**Add**(AccomodationGrid);

panel1.Location = new Point(12, 12);

panel1.Name = "panel1";

panel1.Size = new Size(1018, 530);

panel1.TabIndex = 0;

*//*

*// DeleteButton*

*//*

DeleteButton.Anchor = AnchorStyles.Top | AnchorStyles.Right;

DeleteButton.Location = new Point(890, 3);

DeleteButton.Name = "DeleteButton";

DeleteButton.Size = new Size(125, 23);

DeleteButton.TabIndex = 14;

DeleteButton.Text = "Удалить";

DeleteButton.UseVisualStyleBackColor = **true**;

DeleteButton.Click += DeleteButton\_Click;

*//*

*// label5*

*//*

label5.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

label5.AutoSize = **true**;

label5.Location = new Point(26, 379);

label5.Name = "label5";

label5.Size = new Size(48, 15);

label5.TabIndex = 13;

label5.Text = "Номер:";

*//*

*// label6*

*//*

label6.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

label6.AutoSize = **true**;

label6.Location = new Point(26, 340);

label6.Name = "label6";

label6.Size = new Size(90, 15);

label6.TabIndex = 12;

label6.Text = "Поиск номера:";

*//*

*// RoomComboBox*

*//*

RoomComboBox.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

RoomComboBox.DropDownStyle = ComboBoxStyle.DropDownList;

RoomComboBox.FormattingEnabled = **true**;

RoomComboBox.Location = new Point(122, 376);

RoomComboBox.Name = "RoomComboBox";

RoomComboBox.Size = new Size(278, 23);

RoomComboBox.TabIndex = 11;

*//*

*// SearchRoomField*

*//*

SearchRoomField.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

SearchRoomField.Location = new Point(122, 337);

SearchRoomField.Name = "SearchRoomField";

SearchRoomField.Size = new Size(278, 23);

SearchRoomField.TabIndex = 10;

SearchRoomField.TextChanged += RoomSearchField\_TextChanged;

*//*

*// CreateAccomodationButton*

*//*

CreateAccomodationButton.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

CreateAccomodationButton.Location = new Point(350, 501);

CreateAccomodationButton.Name = "CreateAccomodationButton";

CreateAccomodationButton.Size = new Size(75, 23);

CreateAccomodationButton.TabIndex = 9;

CreateAccomodationButton.Text = "Заселить";

CreateAccomodationButton.UseVisualStyleBackColor = **true**;

CreateAccomodationButton.Click += CreateAccomodationButton\_Click;

*//*

*// label4*

*//*

label4.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

label4.AutoSize = **true**;

label4.Location = new Point(429, 424);

label4.Name = "label4";

label4.Size = new Size(53, 15);

label4.TabIndex = 8;

label4.Text = "Дата ПО";

*//*

*// label3*

*//*

label3.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

label3.AutoSize = **true**;

label3.Location = new Point(429, 385);

label3.Name = "label3";

label3.Size = new Size(43, 15);

label3.TabIndex = 7;

label3.Text = "Дата С";

*//*

*// ToDatePicker*

*//*

ToDatePicker.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

ToDatePicker.Format = DateTimePickerFormat.**Short**;

ToDatePicker.Location = new Point(505, 421);

ToDatePicker.Name = "ToDatePicker";

ToDatePicker.Size = new Size(79, 23);

ToDatePicker.TabIndex = 6;

*//*

*// FromDatePicker*

*//*

FromDatePicker.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

FromDatePicker.Format = DateTimePickerFormat.**Short**;

FromDatePicker.Location = new Point(505, 382);

FromDatePicker.Name = "FromDatePicker";

FromDatePicker.Size = new Size(79, 23);

FromDatePicker.TabIndex = 5;

*//*

*// label2*

*//*

label2.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

label2.AutoSize = **true**;

label2.Location = new Point(26, 463);

label2.Name = "label2";

label2.Size = new Size(40, 15);

label2.TabIndex = 4;

label2.Text = "Гость:";

*//*

*// label1*

*//*

label1.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

label1.AutoSize = **true**;

label1.Location = new Point(26, 424);

label1.Name = "label1";

label1.Size = new Size(77, 15);

label1.TabIndex = 3;

label1.Text = "Поиск гостя:";

*//*

*// CustomerComboBox*

*//*

CustomerComboBox.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

CustomerComboBox.DropDownStyle = ComboBoxStyle.DropDownList;

CustomerComboBox.FormattingEnabled = **true**;

CustomerComboBox.Location = new Point(122, 460);

CustomerComboBox.Name = "CustomerComboBox";

CustomerComboBox.Size = new Size(278, 23);

CustomerComboBox.TabIndex = 2;

*//*

*// SearchNameField*

*//*

SearchNameField.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

SearchNameField.Location = new Point(122, 421);

SearchNameField.Name = "SearchNameField";

SearchNameField.Size = new Size(278, 23);

SearchNameField.TabIndex = 1;

SearchNameField.TextChanged += SearchNameField\_TextChanged;

*//*

*// AccomodationGrid*

*//*

AccomodationGrid.Anchor = AnchorStyles.Top | AnchorStyles.Bottom | AnchorStyles.Left | AnchorStyles.Right;

AccomodationGrid.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

AccomodationGrid.ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.AutoSize;

AccomodationGrid.Location = new Point(0, 0);

AccomodationGrid.Name = "AccomodationGrid";

AccomodationGrid.RowTemplate.Height = 25;

AccomodationGrid.Size = new Size(884, 303);

AccomodationGrid.TabIndex = 0;

*//*

*// AccomodationForm*

*//*

AutoScaleDimensions = new SizeF(7F, 15F);

AutoScaleMode = AutoScaleMode.Font;

ClientSize = new Size(1042, 554);

Controls.**Add**(panel1);

Name = "AccomodationForm";

Text = "Заселения";

panel1.ResumeLayout(**false**);

panel1.PerformLayout();

((System.ComponentModel.ISupportInitialize)AccomodationGrid).EndInit();

ResumeLayout(**false**);

}

#endregion

**private** Panel panel1;

**private** DataGridView AccomodationGrid;

**private** TextBox SearchNameField;

**private** ComboBox CustomerComboBox;

**private** Label label2;

**private** Label label1;

**private** Label label4;

**private** Label label3;

**private** DateTimePicker ToDatePicker;

**private** DateTimePicker FromDatePicker;

**private** Button CreateAccomodationButton;

**private** Label label5;

**private** Label label6;

**private** ComboBox RoomComboBox;

**private** TextBox SearchRoomField;

**private** Button DeleteButton;

}

}

3. ./CustomersForm.cs

﻿**using** HostelApp.Entities;

**using** HostelApp.Exceptions;

**using** HostelApp.Persistence;

**namespace** HostelApp

{

**public** **partial** **class** CustomersForm : Form

{

**private** Customer? currentCustomer;

**private** **async** Task ExecuteCustomersQuery()

{

**var** dataSource = **await** HostelDbContext.GetInstance().GetCustomersAsync();

CustomersGrid.DataSource = dataSource;

**if** (currentCustomer != **null**)

{

**foreach** (**var** item **in** CustomersGrid.Rows)

{

**if** (item is DataGridViewRow row

&& row.DataBoundItem is Customer customer

&& customer.Id == currentCustomer.Id)

{

CustomersGrid.ClearSelection();

row.Selected = **true**;

CustomersGrid.CurrentCell = row.Cells[0];

**break**;

}

}

}

}

**public** CustomersForm()

{

InitializeComponent();

ExecuteCustomersQuery().GetAwaiter().GetResult();

}

**private** **void** CustomersGrid\_SelectionChanged(**object** sender, EventArgs e)

{

**if** (CustomersGrid.CurrentRow != **null**

&& CustomersGrid.CurrentRow.DataBoundItem is Customer customer)

{

NameField.Text = customer.FullName;

BirthdayPicker.**Value** = customer.BirthDate;

}

}

**private** **void** CancelButton\_Click(**object** sender, EventArgs e)

{

Close();

}

**private** **void** CreateButton\_Click(**object** sender, EventArgs e)

{

**try**

{

**var** customer = ReadCustomerFromForm();

HostelDbContext.GetInstance()

.AddCustomerAsync(customer)

.GetAwaiter()

.GetResult();

**if** (customer.Id > 0)

{

currentCustomer = customer;

}

ExecuteCustomersQuery().GetAwaiter().GetResult();

}

**catch** (PersistenceException ex)

{

MessageBox.Show(**this**, ex.Message, "Ошибка");

}

}

**private** Customer ReadCustomerFromForm()

{

**var** customer = new Customer()

{

FullName = NameField.Text,

BirthDate = BirthdayPicker.**Value**

};

**if** (**string**.IsNullOrWhiteSpace(customer.FullName))

{

**throw** new PersistenceException("Имя не введено");

}

**if** (customer.BirthDate < DateTime.UtcNow.AddYears(-150)

|| customer.BirthDate > DateTime.UtcNow)

{

**throw** new PersistenceException("Неправильная дата рождения");

}

**return** customer;

}

**private** **void** EditButton\_Click(**object** sender, EventArgs e)

{

**try**

{

**var** selectedCustomer = GetCurrentCustomer()

?? **throw** new PersistenceException("Гость не выбран!");

**var** customer = ReadCustomerFromForm();

customer.Id = selectedCustomer.Id;

HostelDbContext.GetInstance()

.UpdateCustomerAsync(customer).Wait();

**if** (customer.Id > 0)

{

currentCustomer = customer;

}

ExecuteCustomersQuery().Wait();

}

**catch** (Exception ex)

{

MessageBox.Show(**this**, ex.Message, "Ошибка");

}

}

**private** Customer? GetCurrentCustomer()

{

**var** selectedCustomer = CustomersGrid.CurrentRow?.DataBoundItem **as** Customer;

**return** selectedCustomer;

}

**private** **void** RemoveButton\_Click(**object** sender, EventArgs e)

{

**try**

{

**var** selectedCustomer = GetCurrentCustomer()

?? **throw** new PersistenceException("Гость не выбран!");

HostelDbContext.GetInstance()

.DeleteCustomerAsync(selectedCustomer.Id)

.GetAwaiter()

.GetResult();

ExecuteCustomersQuery().Wait();

}

**catch** (Exception ex)

{

MessageBox.Show(**this**, ex.Message, "Ошибка");

}

}

**private** **void** SelectButton\_Click(**object** sender, EventArgs e)

{

**var** customer = GetCurrentCustomer();

**if** (customer == **null**)

{

MessageBox.Show(**this**, "Гость не выбран", "Ошибка");

**return**;

}

**if** (Owner is RoomForm roomForm)

{

roomForm.SetSelectedCustomer(customer);

Close();

**return**;

}

**else**

{

MessageBox.Show(**this**, "Невозможно выбрать пользователя", "Ошибка");

}

}

**private** **void** AccomodationButton\_Click(**object** sender, EventArgs e)

{

**try**

{

**var** customer = CustomersGrid.CurrentRow.DataBoundItem **as** Customer

?? **throw** new NullReferenceException("Гость не выбран!");

**var** accForm = new AccomodationForm(filterCustomer: customer);

accForm.ShowDialog(**this**);

}

**catch** (Exception ex)

{

MessageBox.Show(**this**, ex.Message, "Ошибка");

}

}

}

}

4. ./CustomersForm.Designer.cs

﻿**namespace** HostelApp

{

**partial** **class** CustomersForm

{

*/// <summary>*

*/// Required designer variable.*

*/// </summary>*

**private** System.ComponentModel.IContainer components = **null**;

*/// <summary>*

*/// Clean up any resources being used.*

*/// </summary>*

*/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>*

**protected** **override** **void** Dispose(**bool** disposing)

{

**if** (disposing && (components != **null**))

{

components.Dispose();

}

**base**.Dispose(disposing);

}

#region Windows Form Designer generated code

*/// <summary>*

*/// Required method for Designer support - do not modify*

*/// the contents of this method with the code editor.*

*/// </summary>*

**private** **void** InitializeComponent()

{

CustomersGrid = new DataGridView();

BirthdayPicker = new DateTimePicker();

NameField = new TextBox();

label1 = new Label();

label2 = new Label();

CreateButton = new Button();

EditButton = new Button();

RemoveButton = new Button();

CancelSelectionButton = new Button();

SelectButton = new Button();

AccomodationButton = new Button();

((System.ComponentModel.ISupportInitialize)CustomersGrid).BeginInit();

SuspendLayout();

*//*

*// CustomersGrid*

*//*

CustomersGrid.Anchor = AnchorStyles.Top | AnchorStyles.Bottom | AnchorStyles.Left | AnchorStyles.Right;

CustomersGrid.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

CustomersGrid.ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.AutoSize;

CustomersGrid.Location = new Point(12, 12);

CustomersGrid.Name = "CustomersGrid";

CustomersGrid.**ReadOnly** = **true**;

CustomersGrid.RowTemplate.Height = 25;

CustomersGrid.Size = new Size(440, 263);

CustomersGrid.TabIndex = 0;

CustomersGrid.SelectionChanged += CustomersGrid\_SelectionChanged;

*//*

*// BirthdayPicker*

*//*

BirthdayPicker.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

BirthdayPicker.Format = DateTimePickerFormat.**Short**;

BirthdayPicker.Location = new Point(131, 358);

BirthdayPicker.Name = "BirthdayPicker";

BirthdayPicker.Size = new Size(270, 23);

BirthdayPicker.TabIndex = 1;

*//*

*// NameField*

*//*

NameField.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

NameField.Location = new Point(131, 319);

NameField.Name = "NameField";

NameField.Size = new Size(270, 23);

NameField.TabIndex = 2;

*//*

*// label1*

*//*

label1.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

label1.AutoSize = **true**;

label1.Location = new Point(28, 322);

label1.Name = "label1";

label1.Size = new Size(37, 15);

label1.TabIndex = 3;

label1.Text = "ФИО:";

*//*

*// label2*

*//*

label2.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

label2.AutoSize = **true**;

label2.Location = new Point(28, 364);

label2.Name = "label2";

label2.Size = new Size(93, 15);

label2.TabIndex = 4;

label2.Text = "Дата рождения:";

*//*

*// CreateButton*

*//*

CreateButton.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

CreateButton.Location = new Point(91, 404);

CreateButton.Name = "CreateButton";

CreateButton.Size = new Size(75, 23);

CreateButton.TabIndex = 5;

CreateButton.Text = "Создать";

CreateButton.UseVisualStyleBackColor = **true**;

CreateButton.Click += CreateButton\_Click;

*//*

*// EditButton*

*//*

EditButton.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

EditButton.Location = new Point(203, 404);

EditButton.Name = "EditButton";

EditButton.Size = new Size(75, 23);

EditButton.TabIndex = 6;

EditButton.Text = "Изменить";

EditButton.UseVisualStyleBackColor = **true**;

EditButton.Click += EditButton\_Click;

*//*

*// RemoveButton*

*//*

RemoveButton.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

RemoveButton.Location = new Point(313, 404);

RemoveButton.Name = "RemoveButton";

RemoveButton.Size = new Size(75, 23);

RemoveButton.TabIndex = 7;

RemoveButton.Text = "Удалить";

RemoveButton.UseVisualStyleBackColor = **true**;

RemoveButton.Click += RemoveButton\_Click;

*//*

*// CancelSelectionButton*

*//*

CancelSelectionButton.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

CancelSelectionButton.Location = new Point(313, 451);

CancelSelectionButton.Name = "CancelSelectionButton";

CancelSelectionButton.Size = new Size(75, 23);

CancelSelectionButton.TabIndex = 8;

CancelSelectionButton.Text = "Отмена";

CancelSelectionButton.UseVisualStyleBackColor = **true**;

CancelSelectionButton.Click += CancelButton\_Click;

*//*

*// SelectButton*

*//*

SelectButton.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

SelectButton.Location = new Point(91, 451);

SelectButton.Name = "SelectButton";

SelectButton.Size = new Size(75, 23);

SelectButton.TabIndex = 9;

SelectButton.Text = "Выбрать";

SelectButton.UseVisualStyleBackColor = **true**;

SelectButton.Click += SelectButton\_Click;

*//*

*// AccomodationButton*

*//*

AccomodationButton.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

AccomodationButton.Location = new Point(12, 281);

AccomodationButton.Name = "AccomodationButton";

AccomodationButton.Size = new Size(75, 23);

AccomodationButton.TabIndex = 10;

AccomodationButton.Text = "Заселения";

AccomodationButton.UseVisualStyleBackColor = **true**;

AccomodationButton.Click += AccomodationButton\_Click;

*//*

*// CustomersForm*

*//*

AutoScaleDimensions = new SizeF(7F, 15F);

AutoScaleMode = AutoScaleMode.Font;

ClientSize = new Size(464, 486);

Controls.**Add**(AccomodationButton);

Controls.**Add**(SelectButton);

Controls.**Add**(CancelSelectionButton);

Controls.**Add**(RemoveButton);

Controls.**Add**(EditButton);

Controls.**Add**(CreateButton);

Controls.**Add**(label2);

Controls.**Add**(label1);

Controls.**Add**(NameField);

Controls.**Add**(BirthdayPicker);

Controls.**Add**(CustomersGrid);

Name = "CustomersForm";

Text = "Гости";

((System.ComponentModel.ISupportInitialize)CustomersGrid).EndInit();

ResumeLayout(**false**);

PerformLayout();

}

#endregion

**private** DataGridView CustomersGrid;

**private** DateTimePicker BirthdayPicker;

**private** TextBox NameField;

**private** Label label1;

**private** Label label2;

**private** Button CreateButton;

**private** Button EditButton;

**private** Button RemoveButton;

**private** Button CancelSelectionButton;

**private** Button SelectButton;

**private** Button AccomodationButton;

}

}

5. ./EditEntityForm.cs

﻿**using** System.CodeDom;

**using** System.ComponentModel;

**using** System.ComponentModel.DataAnnotations;

**using** System.Diagnostics.Eventing.Reader;

**using** System.DirectoryServices.ActiveDirectory;

**using** System.Reflection;

**namespace** HostelApp

{

**public** **partial** **class** EditEntityForm : Form

{

**private** Dictionary<**string**, Control> propertyNameControlMap = new();

**private** **void** InitEditControls()

{

**foreach** (**var** prop **in** Entity.GetType()

.GetProperties()

.**Where**(p => p.CanWrite

&& p.CanRead)

.**Where**(p => p.GetCustomAttribute<KeyAttribute>() == **null**))

{

**var** label = new Label();

label.Text = prop.GetCustomAttribute<DisplayNameAttribute>()?.DisplayName ?? prop.Name;

**var** control = GetControlByProperty(prop, prop.GetValue(Entity));

MainContentLayoutPanel.Controls.**Add**(label);

MainContentLayoutPanel.Controls.**Add**(control);

MainContentLayoutPanel.SetFlowBreak(control, **true**);

propertyNameControlMap[prop.Name] = control;

}

}

**private** Control GetControlByProperty(PropertyInfo property, **object**? **value**)

{

**var** propertyType = property.PropertyType;

**if** (propertyType == typeof(**string**))

{

**var** textBox = new TextBox();

textBox.Text = **value** **as** **String** ?? **string**.Empty;

**return** textBox;

}

**else** **if** (propertyType == typeof(**int**)

|| propertyType == typeof(**long**)

|| propertyType == typeof(**double**))

{

**var** numeric = new NumericUpDown();

numeric.Maximum = **decimal**.MaxValue;

**if** (propertyType == typeof(**double**))

{

numeric.Increment = (**decimal**)0.01;

numeric.DecimalPlaces = 2;

}

**else**

{

numeric.Increment = 1;

}

numeric.**Value** = Convert.ToDecimal(**value** ?? 0);

**return** numeric;

}

**else** **if** (propertyType.IsEnum)

{

**var** comboBox = new ComboBox();

**foreach** (**var** en **in** **Enum**.GetValues(propertyType))

{

comboBox.Items.**Add**(en);

}

comboBox.SelectedItem = **value**;

**return** comboBox;

}

**else** **if** (propertyType == typeof(DateTime))

{

**var** picker = new DateTimePicker();

**if** (property.GetCustomAttribute<DisplayFormatAttribute>()?.DataFormatString == "short")

{

picker.Format = DateTimePickerFormat.**Short**;

}

**var** dateTime = **value** != **null** ? (DateTime)**value** : DateTime.UtcNow.Date;

picker.**Value** = dateTime;

**return** picker;

}

**throw** new Exception("Невозможно создать Control");

}

**private** **void** GetFromControls()

{

**foreach** (**var** (name, control) **in** propertyNameControlMap)

{

**var** prop = Entity.GetType().GetProperty(name)

?? **throw** new NullReferenceException();

**if** (control is TextBox textBox)

{

prop.SetValue(Entity, textBox.Text);

}

**else** **if** (control is ComboBox comboBox)

{

prop.SetValue(Entity, comboBox.SelectedItem);

}

**else** **if** (control is NumericUpDown numeric)

{

**object** **value**;

**if** (prop.PropertyType == typeof(**double**))

{

**value** = Convert.ToDouble(numeric.**Value**);

}

**else** **if** (prop.PropertyType == typeof(**int**))

{

**value** = Convert.ToInt32(numeric.**Value**);

}

**else**

{

**value** = Convert.ToInt64(numeric.**Value**);

}

prop.SetValue(Entity, **value**);

}

**else** **if** (control is DateTimePicker picker)

{

prop.SetValue(Entity, picker.**Value**);

}

**else**

{

**throw** new Exception("Не удалось получить значение");

}

}

}

**public** EditEntityForm(**Object** entity)

{

InitializeComponent();

Entity = entity;

InitEditControls();

}

**public** **object** Entity { **get**; }

**private** **void** SaveButton\_Click(**object** sender, EventArgs e)

{

GetFromControls();

DialogResult = DialogResult.OK;

Close();

}

}

}

6. ./EditEntityForm.Designer.cs

﻿**namespace** HostelApp

{

**partial** **class** EditEntityForm

{

*/// <summary>*

*/// Required designer variable.*

*/// </summary>*

**private** System.ComponentModel.IContainer components = **null**;

*/// <summary>*

*/// Clean up any resources being used.*

*/// </summary>*

*/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>*

**protected** **override** **void** Dispose(**bool** disposing)

{

**if** (disposing && (components != **null**))

{

components.Dispose();

}

**base**.Dispose(disposing);

}

#region Windows Form Designer generated code

*/// <summary>*

*/// Required method for Designer support - do not modify*

*/// the contents of this method with the code editor.*

*/// </summary>*

**private** **void** InitializeComponent()

{

MainContentLayoutPanel = new FlowLayoutPanel();

SaveButton = new Button();

SuspendLayout();

*//*

*// MainContentLayoutPanel*

*//*

MainContentLayoutPanel.Anchor = AnchorStyles.Top | AnchorStyles.Bottom | AnchorStyles.Left | AnchorStyles.Right;

MainContentLayoutPanel.Location = new Point(12, 12);

MainContentLayoutPanel.Name = "MainContentLayoutPanel";

MainContentLayoutPanel.Size = new Size(367, 399);

MainContentLayoutPanel.TabIndex = 0;

*//*

*// SaveButton*

*//*

SaveButton.Anchor = AnchorStyles.Bottom | AnchorStyles.Left | AnchorStyles.Right;

SaveButton.Location = new Point(160, 417);

SaveButton.Name = "SaveButton";

SaveButton.Size = new Size(75, 23);

SaveButton.TabIndex = 1;

SaveButton.Text = "Сохранить";

SaveButton.UseVisualStyleBackColor = **true**;

SaveButton.Click += SaveButton\_Click;

*//*

*// EditEntityForm*

*//*

AutoScaleDimensions = new SizeF(7F, 15F);

AutoScaleMode = AutoScaleMode.Font;

ClientSize = new Size(391, 450);

Controls.**Add**(SaveButton);

Controls.**Add**(MainContentLayoutPanel);

Name = "EditEntityForm";

Text = "Редактировать";

ResumeLayout(**false**);

}

#endregion

**private** FlowLayoutPanel MainContentLayoutPanel;

**private** Button SaveButton;

}

}

7. ./Entities/Accomodation.cs

﻿**using** HostelApp.Persistence;

**using** System.ComponentModel;

**namespace** HostelApp.Entities

{

**public** **class** Accomodation : Entity

{

[DisplayName("ИД номера")]

**public** **int** RoomId { **get**; **set**; }

[DisplayName("Наименование номера")]

**public** **string** RoomName => HostelDbContext

.GetInstance()

.GetRoomAsync(RoomId)

.GetAwaiter()

.GetResult()?.Name

?? **string**.Empty;

[DisplayName("ИД гостя")]

**public** **int** CustomerId { **get**; **set**; }

[DisplayName("Имя гостя")]

**public** **string** CustomerName => HostelDbContext

.GetInstance()

.GetCustomerAsync(CustomerId)

.GetAwaiter()

.GetResult()?.FullName

?? **string**.Empty;

[DisplayName("Дата С")]

**public** DateTime FromDate { **get**; **set**; }

[DisplayName("Дата ПО")]

**public** DateTime ToDate { **get**; **set**; }

}

}

8. ./Entities/Bed.cs

﻿**using** System.ComponentModel;

**namespace** HostelApp.Entities

{

**public** **class** Bed : Entity

{

[DisplayName("ИД спальни")]

**public** **int** BedroomId { **get**; **set**; }

[DisplayName("Вместимость")]

**public** **int** Capacity { **get**; **set**; }

}

}

9. ./Entities/Bedroom.cs

﻿**using** System.ComponentModel;

**namespace** HostelApp.Entities

{

**public** **class** Bedroom : Entity

{

[DisplayName("ИД комнаты")]

**public** **int** RoomId { **get**; **set**; }

[DisplayName("Площадь")]

**public** **double** Area { **get**; **set**; }

}

}

10. ./Entities/Codes/RoomType.cs

﻿**namespace** HostelApp.Entities.Codes

{

**public** **enum** RoomType

{

Все = 0,

Стандарт,

Апартаменты,

Бизнес,

Люкс

}

}

11. ./Entities/Customer.cs

﻿**using** System.ComponentModel;

**using** System.ComponentModel.DataAnnotations;

**namespace** HostelApp.Entities

{

**public** **class** Customer : Entity

{

[DisplayName("ФИО")]

**public** **string** FullName { **get**; **set**; } = **string**.Empty;

[DisplayName("Дата рождения")]

[DisplayFormat(DataFormatString = "short")]

**public** DateTime BirthDate { **get**; **set**; }

[DisplayName("Возраст")]

**public** **int** Age { **get** => DateTime.UtcNow.Year - BirthDate.Year; }

**public** **override** **string**? ToString()

{

**return** $"({Id}): {FullName}";

}

}

}

12. ./Entities/Entity.cs

﻿**using** System.ComponentModel;

**using** System.ComponentModel.DataAnnotations;

**namespace** HostelApp.Entities

{

**public** **class** Entity

{

[DisplayName("ИД")]

[Key]

**public** **int** Id { **get**; **set**; }

}

}

13. ./Entities/Room.cs

﻿**using** HostelApp.Entities.Codes;

**using** System.ComponentModel;

**namespace** HostelApp.Entities

{

**public** **class** Room : Entity

{

[DisplayName("Код номера")]

**public** **int** Number { **get**; **set**; }

[DisplayName("Наименование")]

**public** **string** Name { **get**; **set**; } = **string**.Empty;

[DisplayName("Тип")]

**public** RoomType RoomType { **get**; **set**; }

[DisplayName("Кол-во ванных")]

**public** **int** BathroomsCount { **get**; **set**; }

[DisplayName("Этаж")]

**public** **int** Floor { **get**; **set**; }

[DisplayName("Площадь")]

**public** **double** Area { **get**; **set**; }

**public** **override** **string**? ToString()

{

**return** $"({Id}) {RoomType} №{Number} **\"**{Name}**\"**";

}

}

}

14. ./Exceptions/PersistenceException.cs

﻿**using** System.Runtime.Serialization;

**namespace** HostelApp.Exceptions

{

**public** **class** PersistenceException : Exception

{

**public** PersistenceException()

{

}

**public** PersistenceException(**string**? message) : **base**(message)

{

}

**public** PersistenceException(**string**? message, Exception? innerException) : **base**(message, innerException)

{

}

**protected** PersistenceException(SerializationInfo info, StreamingContext context) : **base**(info, context)

{

}

}

}

15. ./Extensions/HostelDbContextExtensions.cs

﻿**using** HostelApp.Entities;

**using** HostelApp.Persistence;

**using** System.Diagnostics;

**namespace** HostelApp.Extensions

{

**public** **static** **class** HostelDbContextExtensions

{

**public** **static** **async** Task GenerateTestDataSetAsync(**this** HostelDbContext context)

{

**if** (**string**.IsNullOrWhiteSpace(context.GetDatabaseFullFileName()))

{

**var** fileName = Path.GetRandomFileName();

context.SetDatabaseFullFileName(fileName);

}

**using** **var** writeSession = context.BeginSession();

**var** rooms = **await** context.GetRoomsAsync();

rooms.Clear();

**for** (**int** i = 0; i < 100; i++)

{

**var** customer = GenerateRandomCustomer();

**await** context.AddCustomerAsync(customer);

}

**for** (**int** i = 0; i <= 100; i++)

{

**var** room = GenerateRandomRoom();

**await** context.AddRoomAsync(room);

**if** (i % 2 == 0)

{

**var** acc = GenerateRandomAccomodation(room.Id);

**await** context.CreateRoomAccomodationAsync(

acc.RoomId,

acc.FromDate,

acc.ToDate,

acc.CustomerId);

}

**var** bedrooms = GenerateRandomBedroomList(room.Area);

bedrooms.**ForEach**(**async** b =>

{

b.RoomId = room.Id;

**await** context.AddBedroomAsync(b);

**var** beds = GenerateRandomBedList();

beds.**ForEach**(**async** bed =>

{

bed.BedroomId = b.Id;

**await** context.AddBedAsync(bed);

});

});

}

}

**private** **static** Room GenerateRandomRoom()

{

**var** random = new Random();

**var** room = new Room()

{

*// 20.xx - 100.xx*

Area = (**double**)(**int**)(((random.NextDouble() \* 80) + 20) \* 100) / 100,

RoomType = (Entities.Codes.RoomType)random.Next(1, 5),

BathroomsCount = random.Next(1, 3),

Floor = random.Next(1, 15)

};

room.Number = **int**.Parse($"{room.Floor}{random.Next(1, 99)}");

room.Name = $"Номер {room.Number}";

**return** room;

}

**private** **static** List<Bedroom> GenerateRandomBedroomList(**double** area)

{

List<Bedroom> ret = new();

**var** random = new Random();

**var** count = random.Next(1, 5);

**double** sumOfArea = 0;

**for** (**int** i = 0; i < count; i++)

{

**double** bedroomArea;

**if** (i < count - 1)

{

bedroomArea = (area / count) \* (random.NextDouble() \* 0.2 + 1);

sumOfArea += bedroomArea;

}

**else**

{

bedroomArea = area - sumOfArea;

}

bedroomArea = (**double**)(**int**)(bedroomArea \* 100) / 100;

**var** bedroom = new Bedroom()

{

Area = bedroomArea

};

ret.**Add**(bedroom);

}

**return** ret;

}

**private** **static** List<Bed> GenerateRandomBedList()

{

**var** random = new Random();

List<Bed> ret = new();

**var** count = random.Next(1, 3);

**for** (**int** i = 0; i < count; i++)

{

**var** bed = new Bed()

{

Capacity = random.Next(1, 3)

};

ret.**Add**(bed);

}

**return** ret;

}

**private** **static** Customer GenerateRandomCustomer()

{

**var** random = new Random();

**var** utcNow = DateTime.UtcNow;

**var** ageYears = random.Next(1, 100);

**var** ageDays = random.Next(1, 365);

**var** birthDate = utcNow.AddYears(-ageYears).AddDays(ageDays).Date;

Customer ret = new()

{

BirthDate = birthDate,

FullName = GenerateRandomFullName()

};

**return** ret;

}

**private** **static** **string**[] NAMES = new **string**[]

{

"Иван",

"Максим",

"Димон",

"Егор",

"Андрей",

"Аркадий",

"Руслан",

"Денис"

};

**private** **static** **string**[] SURNAMES = new **string**[]

{

"Петров",

"Иванов",

"Сидоров",

"Микушов",

"Денисов",

"Андреев",

"Снегов",

"Жук",

"Кружков"

};

**private** **static** **string**[] SECOND\_NAMES = new **string**[]

{

"Денисович",

"Русланович",

"Аркадьевич",

"Андреевич",

"Егорович",

"Дмитриевич",

"Максимович",

"Иванович"

};

**private** **static** **string** GenerateRandomFullName()

{

**var** random = new Random();

**var** fullName = $"{SURNAMES[random.Next(0, SURNAMES.Length)]} " +

$"{NAMES[random.Next(0, NAMES.Length)]} " +

$"{SECOND\_NAMES[random.Next(0, SECOND\_NAMES.Length)]}";

**return** fullName;

}

**private** **static** Accomodation GenerateRandomAccomodation(**int** roomId)

{

**var** random = new Random();

**var** customers = HostelDbContext

.GetInstance()

.GetCustomersAsync()

.GetAwaiter()

.GetResult();

**var** onDate = DateTime.UtcNow.Date;

**var** fromDateOffset = random.Next(0, 20) - 10;

**var** accomodationLength = random.Next(1, 20);

**var** acc = new Accomodation()

{

RoomId = roomId,

CustomerId = customers[random.Next(0, customers.Count)].Id,

FromDate = onDate.AddDays(fromDateOffset),

ToDate = onDate.AddDays(fromDateOffset).AddDays(accomodationLength)

};

**return** acc;

}

}

}

16. ./Extensions/RoomTypeExtensions.cs

﻿**using** HostelApp.Entities.Codes;

**namespace** HostelApp.Extensions

{

**public** **static** **class** RoomTypeExtensions

{

**public** **static** **string** GetRoomTypeDescription(**this** RoomType roomType)

{

**return** roomType **switch**

{

RoomType.Стандарт => "Эконом",

RoomType.Апартаменты => "Апартаменты",

RoomType.Бизнес => "Бизнес-класс",

RoomType.Люкс => "Люкс",

RoomType.Все => "",

\_ => **throw** new MissingMemberException(nameof(roomType)),

};

}

}

}

17. ./Persistence/BaseDbContext.cs

﻿**using** HostelApp.Entities;

**using** System.Text.Json;

**namespace** HostelApp.Persistence

{

**public** **partial** **class** BaseDbContext

{

**private** **string** \_databaseFullFileName = **string**.Empty;

**private** RootScheme? \_scheme;

**public** **void** SetDatabaseFullFileName(**string** databaseFullFileName)

{

\_databaseFullFileName = databaseFullFileName;

}

**public** **string** GetDatabaseFullFileName() => \_databaseFullFileName;

**public** Task SelectDatabaseFile()

{

**using** OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.CheckFileExists = **false**;

**var** result = openFileDialog.ShowDialog();

**if** (result == DialogResult.OK)

{

\_databaseFullFileName = openFileDialog.FileName;

\_scheme = **null**;

}

**return** Task.CompletedTask;

}

**public** **async** Task ClearDatabaseFile()

{

**var** fileName = GetDatabaseFullFileName();

File.Delete(fileName);

**await** InitDatabase();

\_scheme = **null**;

**await** FetchData();

}

**public** **async** Task CopyDatabaseFile()

{

**await** SaveChanges();

**using** OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.CheckFileExists = **false**;

**var** result = openFileDialog.ShowDialog();

**if** (result == DialogResult.OK)

{

**var** saveFileName = openFileDialog.FileName;

File.Copy(GetDatabaseFullFileName(), saveFileName);

}

}

**private** Task InitDatabase()

{

**var** data = new RootScheme();

**using** **var** fileStream = new FileStream(\_databaseFullFileName, FileMode.Create);

JsonSerializer.Serialize(fileStream, data);

**return** Task.CompletedTask;

}

**private** **async** Task<RootScheme> FetchData()

{

**if** (\_scheme == **null**)

{

**if** (!File.Exists(\_databaseFullFileName)

|| new FileInfo(\_databaseFullFileName).Length == 0)

{

**await** InitDatabase();

}

**using** **var** fileStream = new FileStream(\_databaseFullFileName, FileMode.Open);

\_scheme = JsonSerializer.Deserialize<RootScheme>(fileStream)

?? **throw** new NullReferenceException();

}

**return** \_scheme;

}

**private** **async** Task<List<T>> GetEntities<T>() **where** T : Entity

{

**var** scheme = **await** FetchData();

**foreach** (**var** prop **in** scheme.GetType().GetProperties())

{

**if** (prop.PropertyType == typeof(List<T>))

{

**return** ((List<T>)prop.GetValue(scheme)!)

.OrderBy(e => e.Id)

.ToList();

}

}

**throw** new NullReferenceException();

}

**private** **async** Task UpdateEntities<T>(List<T> entities) **where** T : Entity

{

**var** scheme = **await** FetchData();

**foreach** (**var** prop **in** scheme.GetType().GetProperties())

{

**if** (prop.PropertyType == typeof(List<T>))

{

prop.SetValue(scheme, entities);

}

}

}

**private** **async** Task<T?> GetEntity<T>(**int** id) **where** T : Entity

{

**var** entities = **await** GetEntities<T>();

**var** entity = entities.**Where**(e => e.Id == id).FirstOrDefault();

**return** entity;

}

**private** **async** Task UpdateEntity<T>(T entity) **where** T : Entity

{

**var** entities = **await** GetEntities<T>();

**var** toUpdate = entities.**Where**(e => e.Id == entity.Id).FirstOrDefault()

?? **throw** new KeyNotFoundException();

**var** removed = entities.**Remove**(toUpdate);

**if** (!removed)

{

**throw** new ApplicationException("Remove");

}

entities.**Add**(entity);

**await** UpdateEntities(entities);

}

**private** **async** Task AddEntity<T>(T entity) **where** T : Entity

{

**var** entities = **await** GetEntities<T>();

**if** (entity.Id != 0)

{

**var** existing = entities.**Where**(e => e.Id == entity.Id).FirstOrDefault();

**if** (existing != **null**)

{

**throw** new ApplicationException("Duplicate key");

}

}

**else**

{

**var** data = **await** GetEntities<T>();

**var** lastEntity = data.MaxBy(e => e.Id);

entity.Id = (lastEntity?.Id ?? 0) + 1;

}

entities.**Add**(entity);

**await** UpdateEntities(entities);

}

**private** **async** Task DeleteEntity<T>(**int** id) **where** T : Entity

{

**var** scheme = **await** FetchData();

**foreach** (**var** prop **in** scheme.GetType().GetProperties())

{

**if** (prop.PropertyType == typeof(List<T>))

{

**var** entities = prop.GetValue(scheme) **as** List<T>

?? **throw** new NullReferenceException();

**var** removed = entities.RemoveAll(e => e.Id == id);

**if** (removed == 0)

{

**throw** new ApplicationException("Can't remove");

}

}

}

}

**public** **async** Task SaveChanges()

{

**await** SaveData();

}

**private** Task SaveData()

{

**using** **var** fileStream = new FileStream(

\_databaseFullFileName,

FileMode.Open,

FileAccess.Write);

JsonSerializer.Serialize(fileStream, \_scheme);

**return** Task.CompletedTask;

}

**public** WorkingSession BeginSession() => new WorkingSession(**this**);

**public** **class** WorkingSession : IDisposable

{

**private** **readonly** BaseDbContext \_baseDbContext;

**public** WorkingSession(BaseDbContext baseDbContext)

{

**this**.\_baseDbContext = baseDbContext;

}

**public** **void** Dispose()

{

\_baseDbContext.SaveChanges().Wait();

}

}

}

}

18. ./Persistence/BaseDbContext\_Operations.cs

﻿**using** HostelApp.Entities;

**using** HostelApp.Exceptions;

**namespace** HostelApp.Persistence

{

**public** **partial** **class** BaseDbContext

{

**public** **async** Task<List<Room>> GetRoomsAsync() => **await** GetEntities<Room>();

**public** **async** Task<Room?> GetRoomAsync(**int** id) => **await** GetEntity<Room>(id);

**public** **async** Task AddRoomAsync(Room room) => **await** AddEntity(room);

**public** **async** Task UpdateRoomAsync(Room room) => **await** UpdateEntity(room);

**public** **async** Task DeleteRoomAsync(**int** id)

{

**if** ((**await** GetAccomodationsAsync())

.Any(a => a.RoomId == id))

{

**throw** new PersistenceException(

"Для номера существуют заселения! Удаление невозможно");

}

**await** DeleteEntity<Room>(id);

(**await** GetRoomBedroomsAsync(id))

.**ForEach**(**async** b => **await** DeleteBedroomAsync(b.Id));

}

**public** **async** Task<List<Accomodation>> GetAccomodationsAsync() => **await** GetEntities<Accomodation>();

**public** **async** Task<Accomodation?> GetAccomodationAsync(**int** id) => **await** GetEntity<Accomodation>(id);

**public** **async** Task AddAccomodationAsync(Accomodation accomodation) => **await** AddEntity(accomodation);

**public** **async** Task UpdateAccomodationAsync(Accomodation accomodation) => **await** UpdateEntity(accomodation);

**public** **async** Task DeleteAccomodationAsync(**int** id) => **await** DeleteEntity<Accomodation>(id);

**public** **async** Task<List<Bedroom>> GetBedroomsAsync() => **await** GetEntities<Bedroom>();

**public** **async** Task<Bedroom?> GetBedroomAsync(**int** id) => **await** GetEntity<Bedroom>(id);

**public** **async** Task AddBedroomAsync(Bedroom bedroom) => **await** AddEntity(bedroom);

**public** **async** Task UpdateBedroomAsync(Bedroom bedroom) => **await** UpdateEntity(bedroom);

**public** **async** Task DeleteBedroomAsync(**int** id)

{

**await** DeleteEntity<Bedroom>(id);

(**await** GetBedroomBedsAsync(id))

.**ForEach**(**async** b => **await** DeleteBedAsync(b.Id));

}

**public** **async** Task<List<Bedroom>> GetRoomBedroomsAsync(**int** roomId) =>

(**await** GetBedroomsAsync()).**Where**(b => b.RoomId == roomId).ToList();

**public** **async** Task<List<Bed>> GetBedsAsync() => **await** GetEntities<Bed>();

**public** **async** Task<Bed?> GetBedAsync(**int** id) => **await** GetEntity<Bed>(id);

**public** **async** Task AddBedAsync(Bed bed) => **await** AddEntity(bed);

**public** **async** Task UpdateBedAsync(Bed bed) => **await** UpdateEntity(bed);

**public** **async** Task DeleteBedAsync(**int** id) => **await** DeleteEntity<Bed>(id);

**public** **async** Task<List<Bed>> GetBedroomBedsAsync(**int** bedroomId) =>

(**await** GetBedsAsync()).**Where**(b => b.BedroomId == bedroomId).ToList();

**public** **async** Task<List<Customer>> GetCustomersAsync() => **await** GetEntities<Customer>();

**public** **async** Task<Customer?> GetCustomerAsync(**int** id) => **await** GetEntity<Customer>(id);

**public** **async** Task AddCustomerAsync(Customer customer) => **await** AddEntity(customer);

**public** **async** Task UpdateCustomerAsync(Customer customer) => **await** UpdateEntity(customer);

**public** **async** Task DeleteCustomerAsync(**int** id) => **await** DeleteEntity<Customer>(id);

}

}

19. ./Persistence/HostelDbContext.cs

﻿**using** HostelApp.Entities;

**namespace** HostelApp.Persistence

{

**public** **class** HostelDbContext : BaseDbContext

{

**private** **static** **readonly** HostelDbContext \_instance = new HostelDbContext();

**static** HostelDbContext()

{

}

**private** HostelDbContext()

{

}

**public** **static** HostelDbContext GetInstance()

{

**return** \_instance;

}

*/// <summary>*

*/// Получить свободные комнаты на дату*

*/// </summary>*

**public** **async** Task<List<Room>> GetVacantRooms(

DateTime fromDate,

DateTime toDate)

{

**var** clearFromDate = fromDate.Date;

**var** clearToDate = toDate.Date;

**var** rooms = **await** GetRoomsAsync();

**var** accomodations = (**await** GetAccomodationsAsync())

.**Where**(acc =>

{

**return** acc.FromDate < toDate && acc.ToDate > fromDate;

})

.**Select**(acc => acc.RoomId)

.ToHashSet();

**var** result = rooms.**Where**(r =>

{

**return** !accomodations.Contains(r.Id);

});

**return** result.ToList();

}

*/// <summary>*

*/// Создать заселение*

*/// </summary>*

**public** **async** Task<Accomodation> CreateRoomAccomodationAsync(

**int** roomId,

DateTime fromDate,

DateTime toDate,

**int** customerId)

{

**var** clearFromDate = fromDate.Date;

**var** clearToDate = toDate.Date;

**if** (clearFromDate >= clearToDate)

{

**throw** new ApplicationException("Даты указаны неверно");

}

**var** vacant = **await** GetVacantRooms(clearFromDate, clearToDate);

**if** (!vacant.Any(r => r.Id == roomId))

{

**throw** new ApplicationException($"Комната занята на даты {clearFromDate}-{clearToDate}");

}

**var** acc = new Accomodation()

{

CustomerId = customerId,

FromDate = clearFromDate,

ToDate = clearToDate,

RoomId = roomId

};

**await** AddAccomodationAsync(acc);

**return** acc;

}

*/// <summary>*

*/// Получить заселение в комнату на дату*

*/// </summary>*

**public** **async** Task<Accomodation?> GetRoomAccomodationOnDate(

**int** roomId,

DateTime onDate)

{

**var** clearOnDate = onDate.Date;

**var** accomodation = (**await** GetAccomodationsAsync()).**Where**(acc =>

{

**return** acc.RoomId == roomId

&& acc.FromDate <= onDate

&& acc.ToDate > onDate;

}).FirstOrDefault();

**return** accomodation;

}

}

}

20. ./Persistence/RequirementRoomProvider.cs

﻿**using** HostelApp.Entities;

**using** HostelApp.Requirements;

**namespace** HostelApp.Persistence

{

**public** **class** RequirementRoomProvider

{

**private** **readonly** RequirementSet \_requirementSet;

**public** RequirementRoomProvider(RequirementSet requirementSet)

{

\_requirementSet = requirementSet;

}

**public** **async** Task<List<Room>> GetRoomsAsync()

{

**var** context = HostelDbContext.GetInstance();

**var** rooms = (**await** context.GetRoomsAsync())

.**Where**(r => \_requirementSet.CheckRoom(r)

.GetAwaiter()

.GetResult())

.ToList();

**return** rooms;

}

**public** **async** Task<List<Room>> GetVacantRoomsAsync(

DateTime fromDate,

DateTime toDate)

{

**var** context = HostelDbContext.GetInstance();

**var** rooms = (**await** context.GetVacantRooms(

fromDate,

toDate))

.**Where**(r => \_requirementSet.CheckRoom(r)

.GetAwaiter()

.GetResult())

.ToList();

**return** rooms;

}

}

}

21. ./Persistence/RootScheme.cs

﻿**using** HostelApp.Entities;

**namespace** HostelApp.Persistence

{

**public** **class** RootScheme

{

**public** List<Room> Rooms { **get**; **set**; } = new();

**public** List<Customer> Customers { **get**; **set**; } = new();

**public** List<Bedroom> Bedrooms { **get**; **set**; } = new();

**public** List<Bed> Beds { **get**; **set**; } = new();

**public** List<Accomodation> Accomodations { **get**; **set**; } = new();

}

}

22. ./Program.cs

**namespace** HostelApp

{

**internal** **static** **class** Program

{

*/// <summary>*

*/// The main entry point for the application.*

*/// </summary>*

[STAThread]

**static** **void** Main()

{

*// To customize application configuration such as set high DPI settings or default font,*

*// see https://aka.ms/applicationconfiguration.*

ApplicationConfiguration.Initialize();

Application.Run(new RoomForm());

}

}

}

23. ./Requirements/AreaRequirement.cs

﻿**using** HostelApp.Entities;

**using** HostelApp.Persistence;

**namespace** HostelApp.Requirements

{

**public** **class** AreaRequirement : IRequirement

{

**public** **double** MinArea { **get**; **set**; } = 0;

**public** **double** MaxArea { **get**; **set**; } = 0;

**public** **async** Task<**bool**> CheckRoom(Room room)

{

**var** context = HostelDbContext.GetInstance();

**var** area = (**await** context.GetRoomBedroomsAsync(room.Id)).Sum(x => x.Area);

**return** area >= MinArea

&& area <= MaxArea

|| MaxArea <= 0;

}

}

}

24. ./Requirements/BathroomRequirement.cs

﻿**using** HostelApp.Entities;

**namespace** HostelApp.Requirements

{

**public** **class** BathroomRequirement : IRequirement

{

**public** **int** MinBathroomNumber { **get**; **set**; } = 0;

**public** **int** MaxBathroomNumber { **get**; **set**; } = 0;

**public** Task<**bool**> CheckRoom(Room room)

{

**return** Task.FromResult(room.BathroomsCount >= MinBathroomNumber

&& (room.BathroomsCount <= MaxBathroomNumber

|| MaxBathroomNumber <= 0));

}

}

}

25. ./Requirements/BedRequirement.cs

﻿**using** HostelApp.Entities;

**using** HostelApp.Persistence;

**namespace** HostelApp.Requirements

{

**public** **class** BedRequirement : IRequirement

{

**public** **int** BedCount { **get**; **set**; }

**public** **int** BedCapacity { **get**; **set**; }

**public** **async** Task<**bool**> CheckRoom(Room room)

{

**var** context = HostelDbContext.GetInstance();

**var** beds = (**await** context.GetRoomBedroomsAsync(room.Id))

.Aggregate(

new List<Bed>(),

(list, bedroom) =>

{

list.AddRange(context

.GetBedroomBedsAsync(bedroom.Id)

.GetAwaiter()

.GetResult());

**return** list;

})

.**Where**(b => b.Capacity == BedCapacity);

**return** beds.Count() >= BedCount;

}

}

}

26. ./Requirements/BedroomRequirement.cs

﻿**using** HostelApp.Entities;

**using** HostelApp.Persistence;

**namespace** HostelApp.Requirements

{

**public** **class** BedroomRequirement : IRequirement

{

**public** **int** MinBedroomNumber { **get**; **set**; } = 0;

**public** **int** MaxBedroomNumber { **get**; **set**; } = 0;

**public** **async** Task<**bool**> CheckRoom(Room room)

{

**var** context = HostelDbContext.GetInstance();

**var** bedroomCount = (**await** context.GetRoomBedroomsAsync(room.Id)).Count;

**return** bedroomCount >= MinBedroomNumber

&& (bedroomCount <= MaxBedroomNumber

|| MaxBedroomNumber <= 0);

}

}

}

27. ./Requirements/CapacityRequirement.cs

﻿**using** HostelApp.Entities;

**using** HostelApp.Persistence;

**namespace** HostelApp.Requirements

{

**public** **class** CapacityRequirement : IRequirement

{

**public** **int** MinCapacity { **get**; **set**; }

**public** **int** MaxCapacity { **get**; **set**; }

**public** **async** Task<**bool**> CheckRoom(Room room)

{

**var** context = HostelDbContext.GetInstance();

**var** capacity = 0;

**foreach** (**var** bedroom **in** **await** context.GetRoomBedroomsAsync(room.Id))

{

**foreach** (**var** bed **in** **await** context.GetBedroomBedsAsync(bedroom.Id))

{

capacity += bed.Capacity;

}

}

**return** capacity >= MinCapacity

&& (capacity <= MaxCapacity

|| MaxCapacity == 0);

}

}

}

28. ./Requirements/FloorNumberRequirement.cs

﻿**using** HostelApp.Entities;

**namespace** HostelApp.Requirements

{

**public** **class** FloorNumberRequirement : IRequirement

{

**public** **int** MinFloorNumber { **get**; **set**; } = 0;

**public** **int** MaxFloorNumber { **get**; **set**; } = 0;

**public** Task<**bool**> CheckRoom(Room room)

{

**var** result = room.Floor >= MinFloorNumber

&& (room.Floor <= MaxFloorNumber

|| MaxFloorNumber <= 0);

**return** Task.FromResult(result);

}

}

}

29. ./Requirements/IRequirement.cs

﻿**using** HostelApp.Entities;

**namespace** HostelApp.Requirements

{

**public** **interface** IRequirement

{

**public** Task<**bool**> CheckRoom(Room room);

}

}

30. ./Requirements/RequirementSet.cs

﻿**using** HostelApp.Entities;

**namespace** HostelApp.Requirements

{

**public** **class** RequirementSet : IRequirement

{

**public** List<IRequirement> Requirements { **get**; **set**; }

**public** RequirementSet(List<IRequirement> requirements)

{

Requirements = requirements;

}

**public** **async** Task<**bool**> CheckRoom(Room room)

{

**foreach** (**var** requirement **in** Requirements)

{

**if** (!(**await** requirement.CheckRoom(room)))

{

**return** **false**;

}

}

**return** **true**;

}

}

}

31. ./Requirements/RequirementSetBuilder.cs

﻿**using** HostelApp.Entities.Codes;

**namespace** HostelApp.Requirements

{

**public** **class** RequirementSetBuilder

{

**private** **readonly** List<IRequirement> \_requirements = new();

**public** RequirementSetBuilder AddCapacityRequrement(

**int** minCapacity,

**int** maxCapacity = 0)

{

**var** requirement = new CapacityRequirement()

{

MinCapacity = minCapacity,

MaxCapacity = maxCapacity

};

\_requirements.**Add**(requirement);

**return** **this**;

}

**public** RequirementSetBuilder AddRoomTypeRequirement(

IEnumerable<RoomType> roomTypes)

{

**var** requirement = new RoomTypeRequirement()

{

RoomTypes = roomTypes.ToList()

};

\_requirements.**Add**(requirement);

**return** **this**;

}

**public** RequirementSetBuilder AddBedRequirement(

**int** bedCapacity,

**int** bedNumber)

{

**var** requirement = new BedRequirement()

{

BedCapacity = bedCapacity,

BedCount = bedNumber

};

\_requirements.**Add**(requirement);

**return** **this**;

}

**public** RequirementSetBuilder AddFloorNumberRequirement(

**int** minFloor,

**int** maxFloor)

{

**var** requirement = new FloorNumberRequirement()

{

MinFloorNumber = minFloor,

MaxFloorNumber = maxFloor

};

\_requirements.**Add**(requirement);

**return** **this**;

}

**public** RequirementSetBuilder AddAreaRequirement(

**double** minArea,

**double** maxArea)

{

**var** requirement = new AreaRequirement()

{

MinArea = minArea,

MaxArea = maxArea

};

\_requirements.**Add**(requirement);

**return** **this**;

}

**public** RequirementSetBuilder AddBedroomRequirement(

**int** minBedrooms,

**int** maxBedrooms)

{

**var** requirement = new BedroomRequirement()

{

MinBedroomNumber = minBedrooms,

MaxBedroomNumber = maxBedrooms

};

\_requirements.**Add**(requirement);

**return** **this**;

}

**public** RequirementSetBuilder AddBathroomRequirement(

**int** minBathrooms,

**int** maxBathrooms)

{

**var** requirement = new BathroomRequirement()

{

MinBathroomNumber = minBathrooms,

MaxBathroomNumber = maxBathrooms

};

\_requirements.**Add**(requirement);

**return** **this**;

}

**public** RequirementSet BuildRequirementSet()

{

**return** new RequirementSet(\_requirements);

}

}

}

32. ./Requirements/RoomTypeRequirement.cs

﻿**using** HostelApp.Entities;

**using** HostelApp.Entities.Codes;

**namespace** HostelApp.Requirements

{

**public** **class** RoomTypeRequirement : IRequirement

{

**public** List<RoomType> RoomTypes { **get**; **set**; } = new List<RoomType>();

**public** Task<**bool**> CheckRoom(Room room) =>

Task.FromResult(RoomTypes.Contains(room.RoomType)

|| RoomTypes.Contains(RoomType.Все)

|| RoomTypes.All(t => !**Enum**.IsDefined(t)));

}

}

33. ./RoomForm.cs

**using** HostelApp.Entities;

**using** HostelApp.Entities.Codes;

**using** HostelApp.Extensions;

**using** HostelApp.Persistence;

**using** HostelApp.Requirements;

**namespace** HostelApp

{

**public** **partial** **class** RoomForm : Form

{

**private** Customer? \_selectedCustomer = **null**;

**public** **void** SetSelectedCustomer(Customer? selectedCustomer)

{

\_selectedCustomer = selectedCustomer;

**if** (\_selectedCustomer != **null**)

{

CurrentCustomerLabel.Text =

$"ҥ곹詠㮱򼺠{\_selectedCustomer.FullName} ({\_selectedCustomer.Id})";

}

**else**

{

CurrentCustomerLabel.Text =

$"ҥ곹詠㮱򼺠�⻡𠭢;

}

}

public RoomForm()

{

InitializeComponent();

}

private async Task ExecuteRoomQuery()

{

var requirementSet = new RequirementSetBuilder()

.AddRoomTypeRequirement(

new RoomType[] { (RoomType)RoomTypeField.SelectedIndex })

.AddFloorNumberRequirement(

(int)MinFloorNumberField.Value,

(int)MaxFloorNumberField.Value)

.AddAreaRequirement(

(double)MinAreaField.Value,

(double)MaxAreaField.Value)

.AddCapacityRequrement(

(int)MinCapacityField.Value,

(int)MaxCapacityField.Value)

.AddBathroomRequirement(

(int)MinBathroomNumberField.Value,

(int)MaxBathroomNumberField.Value)

.AddBedRequirement(

1,

(int)OnePlaceBedNumberField.Value)

.AddBedRequirement(

2,

(int)TwoPlaceBedNumberField.Value)

.AddBedroomRequirement(

(int)MinBedroomNumberField.Value,

(int)MaxBedroomNumberField.Value)

.BuildRequirementSet();

var roomProvider = new RequirementRoomProvider(requirementSet);

var fromDate = VacantCalendar.SelectionStart;

var toDate = VacantCalendar.SelectionEnd;

List<Room> rooms;

if (IsOnlyVacantField.Checked

&& fromDate != DateTime.MinValue

&& toDate != DateTime.MinValue)

{

rooms = await roomProvider.GetVacantRoomsAsync(

fromDate,

toDate);

}

else

{

rooms = await roomProvider.GetRoomsAsync();

}

RoomGrid.DataSource = rooms;

}

private async Task ExecuteBedroomQuery()

{

var dataSource = Enumerable.Empty<Bedroom>().ToList();

if (RoomGrid.CurrentRow != null)

{

var room = RoomGrid.CurrentRow.DataBoundItem as Room;

if (room != null)

{

dataSource = await HostelDbContext

.GetInstance()

.GetRoomBedroomsAsync(room.Id);

}

}

BedroomGrid.DataSource = dataSource;

}

private async Task ExecuteBedQuery()

{

var dataSource = Enumerable.Empty<Bed>().ToList();

if (BedroomGrid.CurrentRow != null)

{

var bedroom = BedroomGrid.CurrentRow.DataBoundItem as Bedroom;

if (bedroom != null)

{

dataSource = await HostelDbContext

.GetInstance()

.GetBedroomBedsAsync(bedroom.Id);

}

}

BedGrid.DataSource = dataSource;

}

private void RoomForm\_Load(object sender, EventArgs e)

{

var context = HostelDbContext.GetInstance();

if (string.IsNullOrWhiteSpace(context.GetDatabaseFullFileName()))

{

using var session = context.BeginSession();

context.SetDatabaseFullFileName(Path.GetTempFileName());

context.GenerateTestDataSetAsync().GetAwaiter().GetResult();

}

ExecuteRoomQuery().GetAwaiter().GetResult();

InitFilters();

}

private void InitFilters()

{

var enumValues = Enum

.GetValues<RoomType>();

RoomTypeField.DataSource = enumValues;

}

private void RoomTypeField\_SelectedValueChanged(object sender, EventArgs e)

{

ExecuteRoomQuery().GetAwaiter().GetResult();

}

private void FilterChanged(object sender, EventArgs e)

{

ExecuteRoomQuery().GetAwaiter().GetResult();

}

private void monthCalendar1\_DateChanged(object sender, DateRangeEventArgs e)

{

FilterChanged(sender, e);

}

private void RoomGrid\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void RoomGrid\_DataBindingComplete(object sender, DataGridViewBindingCompleteEventArgs e)

{

ExecuteBedroomQuery().GetAwaiter().GetResult();

}

private void RoomGrid\_SelectionChanged(object sender, EventArgs e)

{

ExecuteBedroomQuery().GetAwaiter().GetResult();

}

private void BedroomGrid\_DataBindingComplete(object sender, DataGridViewBindingCompleteEventArgs e)

{

ExecuteBedQuery().GetAwaiter().GetResult();

}

private void BedroomGrid\_SelectionChanged(object sender, EventArgs e)

{

ExecuteBedQuery().GetAwaiter().GetResult();

}

private void CustomersButton\_Click(object sender, EventArgs e)

{

var customers = new CustomersForm();

customers.ShowDialog(this);

}

private void EditButton\_Click(object sender, EventArgs e)

{

var currentGrid = GetActiveGrid();

if (currentGrid == null)

{

MessageBox.Show(this, "Ġ�堭堢ûᰠ�, "θ衪ࢩ;

return;

}

var currentObject = currentGrid.CurrentRow.DataBoundItem;

if (currentObject != null)

{

var edit = new EditEntityForm(currentObject);

var result = edit.ShowDialog(this);

if (result == DialogResult.OK)

{

using (var session = HostelDbContext.GetInstance().BeginSession())

{

if (currentGrid == RoomGrid)

{

HostelDbContext.GetInstance().UpdateRoomAsync((Room)currentObject)

.GetAwaiter().GetResult();

}

else if (currentGrid == BedroomGrid)

{

HostelDbContext.GetInstance().UpdateBedroomAsync((Bedroom)currentObject)

.GetAwaiter().GetResult();

}

else if (currentGrid == BedGrid)

{

HostelDbContext.GetInstance().UpdateBedAsync((Bed)currentObject)

.GetAwaiter().GetResult();

}

}

currentGrid.Update();

currentGrid.Refresh();

}

}

else

{

MessageBox.Show(this, "Ġ�堭堢ûᰠ�, "θ衪ࢩ;

}

}

private DataGridView? GetActiveGrid()

{

if (TabControl.SelectedTab == RoomTab)

{

return RoomGrid;

}

else if (TabControl.SelectedTab == BedroomTab)

{

return BedroomGrid;

}

else if (TabControl.SelectedTab == BedTab)

{

return BedGrid;

}

return null;

}

private void AddButton\_Click(object sender, EventArgs e)

{

var currentGrid = GetActiveGrid();

if (currentGrid == null)

{

MessageBox.Show(this, "Ġ�堭堢ûᰠ�, "θ衪ࢩ;

return;

}

object? currentObject = null;

if (currentGrid == RoomGrid)

{

currentObject = new Room();

}

else if (currentGrid == BedroomGrid)

{

var room = RoomGrid.CurrentRow.DataBoundItem as Room;

currentObject = new Bedroom()

{

RoomId = room?.Id ?? 0

};

}

else if (currentGrid == BedGrid)

{

var bedroom = BedroomGrid.CurrentRow.DataBoundItem as Bedroom;

currentObject = new Bed()

{

BedroomId = bedroom?.Id ?? 0

};

}

if (currentObject != null)

{

var edit = new EditEntityForm(currentObject);

var result = edit.ShowDialog(this);

if (result == DialogResult.OK)

{

using (var session = HostelDbContext.GetInstance().BeginSession())

{

if (currentGrid == RoomGrid)

{

HostelDbContext.GetInstance()

.AddRoomAsync((Room)currentObject).Wait();

}

else if (currentGrid == BedroomGrid)

{

HostelDbContext.GetInstance()

.AddBedroomAsync((Bedroom)currentObject).Wait();

}

else if (currentGrid == BedGrid)

{

HostelDbContext.GetInstance()

.AddBedAsync((Bed)currentObject).Wait();

}

}

if (currentGrid == RoomGrid)

{

ExecuteRoomQuery().Wait();

}

else if (currentGrid == BedroomGrid)

{

ExecuteBedroomQuery().Wait();

}

else if (currentGrid == BedGrid)

{

ExecuteBedQuery().Wait();

}

}

}

else

{

MessageBox.Show(this, "Ġ�堭堢ûᰠ�, "θ衪ࢩ;

}

}

private void AccomodationButton\_Click(object sender, EventArgs e)

{

var room = RoomGrid.CurrentRow.DataBoundItem as Room;

var form = new AccomodationForm(

defaultCustomer: \_selectedCustomer,

defaultRoom: room);

form.ShowDialog(this);

}

private void RoomAccomodationButton\_Click(object sender, EventArgs e)

{

var room = RoomGrid.CurrentRow.DataBoundItem as Room;

var form = new AccomodationForm(

filterRoom: room,

defaultCustomer: \_selectedCustomer);

form.ShowDialog(this);

}

private void SelectDbButton\_Click(object sender, EventArgs e)

{

var context = HostelDbContext.GetInstance();

var prevFileName = context.GetDatabaseFullFileName();

try

{

context.SaveChanges().Wait();

context.SelectDatabaseFile().Wait();

ExecuteRoomQuery().Wait();

}

catch

{

context.SetDatabaseFullFileName(prevFileName);

ExecuteRoomQuery().Wait();

MessageBox.Show(this, "ͥ 󤠫 𻲼 ��", "θ衪ࢩ;

}

}

**private** **void** SaveDbButton\_Click(**object** sender, EventArgs e)

{

**try**

{

**var** context = HostelDbContext.GetInstance();

context.CopyDatabaseFile().Wait();

}

**catch** (Exception ex)

{

MessageBox.Show(**this**, ex.Message, "θ衪ࢩ;

}

}

private void ClearDbButton\_Click(object sender, EventArgs e)

{

try

{

var context = HostelDbContext.GetInstance();

context.ClearDatabaseFile().Wait();

ExecuteRoomQuery().Wait();

}

catch (Exception ex)

{

MessageBox.Show(this, ex.Message, "θ衪ࢩ;

}

}

**private** **void** TestBdButton\_Click(**object** sender, EventArgs e)

{

**try**

{

**var** context = HostelDbContext.GetInstance();

context.ClearDatabaseFile().Wait();

context.GenerateTestDataSetAsync().Wait();

context.SaveChanges().Wait();

ExecuteRoomQuery().Wait();

}

**catch** (Exception ex)

{

MessageBox.Show(**this**, ex.Message, "θ衪ࢩ;

}

}

private void IsVacantField\_CheckedChanged(object sender, EventArgs e)

{

var value = IsOnlyVacantField.Checked;

if (value)

{

VacantCalendar.Enabled = true;

}

else

{

VacantCalendar.Enabled = false;

}

ExecuteRoomQuery().Wait();

}

private void DeleteButton\_Click(object sender, EventArgs e)

{

try

{

var context = HostelDbContext.GetInstance();

if (TabControl.SelectedTab == RoomTab)

{

var entity = RoomGrid.CurrentRow.DataBoundItem as Room

?? throw new NullReferenceException("ݫ嬥�䫿 󤠫孨ÿ �⻡𠭢);

context.DeleteRoomAsync(entity.Id).Wait();

}

**else** **if** (TabControl.SelectedTab == BedroomTab)

{

**var** entity = BedroomGrid.CurrentRow.DataBoundItem **as** Bedroom

?? **throw** new NullReferenceException("ݫ嬥�䫿 󤠫孨ÿ �⻡𠭢);

context.DeleteBedroomAsync(entity.Id).Wait();

}

else if (TabControl.SelectedTab == BedTab)

{

var entity = BedGrid.CurrentRow.DataBoundItem as Bed

?? throw new NullReferenceException("ݫ嬥�䫿 󤠫孨ÿ �⻡𠭢);

context.DeleteBedAsync(entity.Id).Wait();

}

**else**

{

**throw** new ApplicationException("ݫ嬥�䫿 󤠫孨ÿ �⻡𠭢);

}

ExecuteRoomQuery().Wait();

}

catch (Exception ex)

{

MessageBox.Show(this, ex.Message, "θ衪ࢩ;

}

}

}

}

34. ./RoomForm.Designer.cs

﻿**namespace** HostelApp

{

**partial** **class** RoomForm

{

*/// <summary>*

*/// Required designer variable.*

*/// </summary>*

**private** System.ComponentModel.IContainer components = **null**;

*/// <summary>*

*/// Clean up any resources being used.*

*/// </summary>*

*/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>*

**protected** **override** **void** Dispose(**bool** disposing)

{

**if** (disposing && (components != **null**))

{

components.Dispose();

}

**base**.Dispose(disposing);

}

#region Windows Form Designer generated code

*/// <summary>*

*/// Required method for Designer support - do not modify*

*/// the contents of this method with the code editor.*

*/// </summary>*

**private** **void** InitializeComponent()

{

RoomGrid = new DataGridView();

RoomTypeField = new ComboBox();

label1 = new Label();

label2 = new Label();

label3 = new Label();

label4 = new Label();

MinCapacityField = new NumericUpDown();

MaxCapacityField = new NumericUpDown();

OnePlaceBedNumberField = new NumericUpDown();

TwoPlaceBedNumberField = new NumericUpDown();

label5 = new Label();

label6 = new Label();

label7 = new Label();

MaxFloorNumberField = new NumericUpDown();

MinFloorNumberField = new NumericUpDown();

label8 = new Label();

label9 = new Label();

MaxAreaField = new NumericUpDown();

MinAreaField = new NumericUpDown();

label10 = new Label();

label11 = new Label();

label12 = new Label();

MaxBathroomNumberField = new NumericUpDown();

MinBathroomNumberField = new NumericUpDown();

label13 = new Label();

label14 = new Label();

label15 = new Label();

MaxBedroomNumberField = new NumericUpDown();

MinBedroomNumberField = new NumericUpDown();

label16 = new Label();

label17 = new Label();

label18 = new Label();

splitContainer1 = new SplitContainer();

panel1 = new Panel();

VacantCalendar = new MonthCalendar();

IsOnlyVacantField = new CheckBox();

TabControl = new TabControl();

RoomTab = new TabPage();

BedroomTab = new TabPage();

BedroomGrid = new DataGridView();

BedTab = new TabPage();

BedGrid = new DataGridView();

CustomersButton = new Button();

statusStrip1 = new StatusStrip();

CurrentCustomerLabel = new ToolStripStatusLabel();

EditButton = new Button();

AddButton = new Button();

AccomodationButton = new Button();

RoomAccomodationButton = new Button();

SelectDbButton = new Button();

SaveDbButton = new Button();

ClearDbButton = new Button();

TestBdButton = new Button();

DeleteButton = new Button();

((System.ComponentModel.ISupportInitialize)RoomGrid).BeginInit();

((System.ComponentModel.ISupportInitialize)MinCapacityField).BeginInit();

((System.ComponentModel.ISupportInitialize)MaxCapacityField).BeginInit();

((System.ComponentModel.ISupportInitialize)OnePlaceBedNumberField).BeginInit();

((System.ComponentModel.ISupportInitialize)TwoPlaceBedNumberField).BeginInit();

((System.ComponentModel.ISupportInitialize)MaxFloorNumberField).BeginInit();

((System.ComponentModel.ISupportInitialize)MinFloorNumberField).BeginInit();

((System.ComponentModel.ISupportInitialize)MaxAreaField).BeginInit();

((System.ComponentModel.ISupportInitialize)MinAreaField).BeginInit();

((System.ComponentModel.ISupportInitialize)MaxBathroomNumberField).BeginInit();

((System.ComponentModel.ISupportInitialize)MinBathroomNumberField).BeginInit();

((System.ComponentModel.ISupportInitialize)MaxBedroomNumberField).BeginInit();

((System.ComponentModel.ISupportInitialize)MinBedroomNumberField).BeginInit();

((System.ComponentModel.ISupportInitialize)splitContainer1).BeginInit();

splitContainer1.Panel1.SuspendLayout();

splitContainer1.Panel2.SuspendLayout();

splitContainer1.SuspendLayout();

panel1.SuspendLayout();

TabControl.SuspendLayout();

RoomTab.SuspendLayout();

BedroomTab.SuspendLayout();

((System.ComponentModel.ISupportInitialize)BedroomGrid).BeginInit();

BedTab.SuspendLayout();

((System.ComponentModel.ISupportInitialize)BedGrid).BeginInit();

statusStrip1.SuspendLayout();

SuspendLayout();

*//*

*// RoomGrid*

*//*

RoomGrid.AllowUserToAddRows = **false**;

RoomGrid.AllowUserToDeleteRows = **false**;

RoomGrid.AllowUserToOrderColumns = **true**;

RoomGrid.AllowUserToResizeRows = **false**;

RoomGrid.Anchor = AnchorStyles.Top | AnchorStyles.Bottom | AnchorStyles.Left | AnchorStyles.Right;

RoomGrid.ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.AutoSize;

RoomGrid.Location = new Point(-3, 0);

RoomGrid.Name = "RoomGrid";

RoomGrid.**ReadOnly** = **true**;

RoomGrid.RowTemplate.Height = 25;

RoomGrid.Size = new Size(697, 412);

RoomGrid.TabIndex = 0;

RoomGrid.CellContentClick += RoomGrid\_CellContentClick;

RoomGrid.DataBindingComplete += RoomGrid\_DataBindingComplete;

RoomGrid.SelectionChanged += RoomGrid\_SelectionChanged;

*//*

*// RoomTypeField*

*//*

RoomTypeField.FormattingEnabled = **true**;

RoomTypeField.Location = new Point(5, 20);

RoomTypeField.Name = "RoomTypeField";

RoomTypeField.Size = new Size(114, 23);

RoomTypeField.TabIndex = 1;

RoomTypeField.SelectedValueChanged += FilterChanged;

*//*

*// label1*

*//*

label1.AutoSize = **true**;

label1.Location = new Point(5, 4);

label1.Name = "label1";

label1.Size = new Size(39, 15);

label1.TabIndex = 2;

label1.Text = "Класс";

*//*

*// label2*

*//*

label2.AutoSize = **true**;

label2.Location = new Point(6, 46);

label2.Name = "label2";

label2.Size = new Size(94, 15);

label2.TabIndex = 5;

label2.Text = "Кол-во человек";

*//*

*// label3*

*//*

label3.AutoSize = **true**;

label3.Location = new Point(6, 67);

label3.Name = "label3";

label3.Size = new Size(21, 15);

label3.TabIndex = 6;

label3.Text = "От";

*//*

*// label4*

*//*

label4.AutoSize = **true**;

label4.Location = new Point(76, 67);

label4.Name = "label4";

label4.Size = new Size(22, 15);

label4.TabIndex = 7;

label4.Text = "До";

*//*

*// MinCapacityField*

*//*

MinCapacityField.Location = new Point(27, 64);

MinCapacityField.Name = "MinCapacityField";

MinCapacityField.Size = new Size(47, 23);

MinCapacityField.TabIndex = 10;

MinCapacityField.ValueChanged += FilterChanged;

*//*

*// MaxCapacityField*

*//*

MaxCapacityField.Location = new Point(104, 64);

MaxCapacityField.Name = "MaxCapacityField";

MaxCapacityField.Size = new Size(47, 23);

MaxCapacityField.TabIndex = 11;

MaxCapacityField.ValueChanged += FilterChanged;

*//*

*// OnePlaceBedNumberField*

*//*

OnePlaceBedNumberField.Location = new Point(273, 21);

OnePlaceBedNumberField.Maximum = new **decimal**(new **int**[] { 10, 0, 0, 0 });

OnePlaceBedNumberField.Name = "OnePlaceBedNumberField";

OnePlaceBedNumberField.Size = new Size(47, 23);

OnePlaceBedNumberField.TabIndex = 12;

OnePlaceBedNumberField.ValueChanged += FilterChanged;

*//*

*// TwoPlaceBedNumberField*

*//*

TwoPlaceBedNumberField.Location = new Point(272, 50);

TwoPlaceBedNumberField.Maximum = new **decimal**(new **int**[] { 10, 0, 0, 0 });

TwoPlaceBedNumberField.Name = "TwoPlaceBedNumberField";

TwoPlaceBedNumberField.Size = new Size(47, 23);

TwoPlaceBedNumberField.TabIndex = 13;

TwoPlaceBedNumberField.ValueChanged += FilterChanged;

*//*

*// label5*

*//*

label5.AutoSize = **true**;

label5.Location = new Point(188, 23);

label5.Name = "label5";

label5.Size = new Size(78, 15);

label5.TabIndex = 14;

label5.Text = "1-сп кровати";

*//*

*// label6*

*//*

label6.AutoSize = **true**;

label6.Location = new Point(188, 52);

label6.Name = "label6";

label6.Size = new Size(78, 15);

label6.TabIndex = 15;

label6.Text = "2-сп кровати";

*//*

*// label7*

*//*

label7.AutoSize = **true**;

label7.Location = new Point(14, -1);

label7.Name = "label7";

label7.Size = new Size(34, 15);

label7.TabIndex = 16;

label7.Text = "Этаж";

*//*

*// MaxFloorNumberField*

*//*

MaxFloorNumberField.Location = new Point(112, 17);

MaxFloorNumberField.Maximum = new **decimal**(new **int**[] { 20, 0, 0, 0 });

MaxFloorNumberField.Name = "MaxFloorNumberField";

MaxFloorNumberField.Size = new Size(47, 23);

MaxFloorNumberField.TabIndex = 20;

MaxFloorNumberField.ValueChanged += FilterChanged;

*//*

*// MinFloorNumberField*

*//*

MinFloorNumberField.Location = new Point(35, 17);

MinFloorNumberField.Maximum = new **decimal**(new **int**[] { 20, 0, 0, 0 });

MinFloorNumberField.Name = "MinFloorNumberField";

MinFloorNumberField.Size = new Size(47, 23);

MinFloorNumberField.TabIndex = 19;

MinFloorNumberField.ValueChanged += FilterChanged;

*//*

*// label8*

*//*

label8.AutoSize = **true**;

label8.Location = new Point(84, 20);

label8.Name = "label8";

label8.Size = new Size(22, 15);

label8.TabIndex = 18;

label8.Text = "До";

*//*

*// label9*

*//*

label9.AutoSize = **true**;

label9.Location = new Point(14, 20);

label9.Name = "label9";

label9.Size = new Size(21, 15);

label9.TabIndex = 17;

label9.Text = "От";

*//*

*// MaxAreaField*

*//*

MaxAreaField.Location = new Point(112, 55);

MaxAreaField.Maximum = new **decimal**(new **int**[] { 150, 0, 0, 0 });

MaxAreaField.Name = "MaxAreaField";

MaxAreaField.Size = new Size(47, 23);

MaxAreaField.TabIndex = 25;

MaxAreaField.ValueChanged += FilterChanged;

*//*

*// MinAreaField*

*//*

MinAreaField.Location = new Point(35, 55);

MinAreaField.Maximum = new **decimal**(new **int**[] { 150, 0, 0, 0 });

MinAreaField.Name = "MinAreaField";

MinAreaField.Size = new Size(47, 23);

MinAreaField.TabIndex = 24;

MinAreaField.ValueChanged += FilterChanged;

*//*

*// label10*

*//*

label10.AutoSize = **true**;

label10.Location = new Point(84, 58);

label10.Name = "label10";

label10.Size = new Size(22, 15);

label10.TabIndex = 23;

label10.Text = "До";

*//*

*// label11*

*//*

label11.AutoSize = **true**;

label11.Location = new Point(14, 58);

label11.Name = "label11";

label11.Size = new Size(21, 15);

label11.TabIndex = 22;

label11.Text = "От";

*//*

*// label12*

*//*

label12.AutoSize = **true**;

label12.Location = new Point(14, 37);

label12.Name = "label12";

label12.Size = new Size(59, 15);

label12.TabIndex = 21;

label12.Text = "Площадь";

*//*

*// MaxBathroomNumberField*

*//*

MaxBathroomNumberField.Location = new Point(118, 55);

MaxBathroomNumberField.Maximum = new **decimal**(new **int**[] { 10, 0, 0, 0 });

MaxBathroomNumberField.Name = "MaxBathroomNumberField";

MaxBathroomNumberField.Size = new Size(47, 23);

MaxBathroomNumberField.TabIndex = 30;

MaxBathroomNumberField.ValueChanged += FilterChanged;

*//*

*// MinBathroomNumberField*

*//*

MinBathroomNumberField.Location = new Point(41, 55);

MinBathroomNumberField.Maximum = new **decimal**(new **int**[] { 10, 0, 0, 0 });

MinBathroomNumberField.Name = "MinBathroomNumberField";

MinBathroomNumberField.Size = new Size(47, 23);

MinBathroomNumberField.TabIndex = 29;

MinBathroomNumberField.ValueChanged += FilterChanged;

*//*

*// label13*

*//*

label13.AutoSize = **true**;

label13.Location = new Point(90, 61);

label13.Name = "label13";

label13.Size = new Size(22, 15);

label13.TabIndex = 28;

label13.Text = "До";

*//*

*// label14*

*//*

label14.AutoSize = **true**;

label14.Location = new Point(20, 61);

label14.Name = "label14";

label14.Size = new Size(21, 15);

label14.TabIndex = 27;

label14.Text = "От";

*//*

*// label15*

*//*

label15.AutoSize = **true**;

label15.Location = new Point(20, 40);

label15.Name = "label15";

label15.Size = new Size(49, 15);

label15.TabIndex = 26;

label15.Text = "Ванные";

*//*

*// MaxBedroomNumberField*

*//*

MaxBedroomNumberField.Location = new Point(118, 18);

MaxBedroomNumberField.Maximum = new **decimal**(new **int**[] { 10, 0, 0, 0 });

MaxBedroomNumberField.Name = "MaxBedroomNumberField";

MaxBedroomNumberField.Size = new Size(47, 23);

MaxBedroomNumberField.TabIndex = 35;

MaxBedroomNumberField.ValueChanged += FilterChanged;

*//*

*// MinBedroomNumberField*

*//*

MinBedroomNumberField.Location = new Point(41, 18);

MinBedroomNumberField.Maximum = new **decimal**(new **int**[] { 10, 0, 0, 0 });

MinBedroomNumberField.Name = "MinBedroomNumberField";

MinBedroomNumberField.Size = new Size(47, 23);

MinBedroomNumberField.TabIndex = 34;

MinBedroomNumberField.ValueChanged += FilterChanged;

*//*

*// label16*

*//*

label16.AutoSize = **true**;

label16.Location = new Point(90, 21);

label16.Name = "label16";

label16.Size = new Size(22, 15);

label16.TabIndex = 33;

label16.Text = "До";

*//*

*// label17*

*//*

label17.AutoSize = **true**;

label17.Location = new Point(20, 21);

label17.Name = "label17";

label17.Size = new Size(21, 15);

label17.TabIndex = 32;

label17.Text = "От";

*//*

*// label18*

*//*

label18.AutoSize = **true**;

label18.Location = new Point(20, 0);

label18.Name = "label18";

label18.Size = new Size(55, 15);

label18.TabIndex = 31;

label18.Text = "Спальни";

*//*

*// splitContainer1*

*//*

splitContainer1.BorderStyle = BorderStyle.FixedSingle;

splitContainer1.Location = new Point(157, 4);

splitContainer1.Name = "splitContainer1";

*//*

*// splitContainer1.Panel1*

*//*

splitContainer1.Panel1.Controls.**Add**(label12);

splitContainer1.Panel1.Controls.**Add**(label7);

splitContainer1.Panel1.Controls.**Add**(label9);

splitContainer1.Panel1.Controls.**Add**(label8);

splitContainer1.Panel1.Controls.**Add**(MinFloorNumberField);

splitContainer1.Panel1.Controls.**Add**(MaxFloorNumberField);

splitContainer1.Panel1.Controls.**Add**(MaxAreaField);

splitContainer1.Panel1.Controls.**Add**(label11);

splitContainer1.Panel1.Controls.**Add**(MinAreaField);

splitContainer1.Panel1.Controls.**Add**(label10);

*//*

*// splitContainer1.Panel2*

*//*

splitContainer1.Panel2.Controls.**Add**(TwoPlaceBedNumberField);

splitContainer1.Panel2.Controls.**Add**(MaxBedroomNumberField);

splitContainer1.Panel2.Controls.**Add**(OnePlaceBedNumberField);

splitContainer1.Panel2.Controls.**Add**(MinBedroomNumberField);

splitContainer1.Panel2.Controls.**Add**(label5);

splitContainer1.Panel2.Controls.**Add**(label16);

splitContainer1.Panel2.Controls.**Add**(MaxBathroomNumberField);

splitContainer1.Panel2.Controls.**Add**(label6);

splitContainer1.Panel2.Controls.**Add**(MinBathroomNumberField);

splitContainer1.Panel2.Controls.**Add**(label17);

splitContainer1.Panel2.Controls.**Add**(label18);

splitContainer1.Panel2.Controls.**Add**(label13);

splitContainer1.Panel2.Controls.**Add**(label15);

splitContainer1.Panel2.Controls.**Add**(label14);

splitContainer1.Size = new Size(548, 83);

splitContainer1.SplitterDistance = 185;

splitContainer1.TabIndex = 36;

*//*

*// panel1*

*//*

panel1.Anchor = AnchorStyles.Top | AnchorStyles.Right;

panel1.BorderStyle = BorderStyle.FixedSingle;

panel1.Controls.**Add**(VacantCalendar);

panel1.Controls.**Add**(IsOnlyVacantField);

panel1.Location = new Point(709, 5);

panel1.Name = "panel1";

panel1.Size = new Size(180, 190);

panel1.TabIndex = 37;

*//*

*// VacantCalendar*

*//*

VacantCalendar.Enabled = **false**;

VacantCalendar.Location = new Point(9, 22);

VacantCalendar.MaxSelectionCount = 31;

VacantCalendar.Name = "VacantCalendar";

VacantCalendar.TabIndex = 1;

VacantCalendar.DateSelected += monthCalendar1\_DateChanged;

*//*

*// IsOnlyVacantField*

*//*

IsOnlyVacantField.AutoSize = **true**;

IsOnlyVacantField.Location = new Point(5, 3);

IsOnlyVacantField.Name = "IsOnlyVacantField";

IsOnlyVacantField.Size = new Size(151, 19);

IsOnlyVacantField.TabIndex = 0;

IsOnlyVacantField.Text = "Доступно к заселению";

IsOnlyVacantField.UseVisualStyleBackColor = **true**;

IsOnlyVacantField.CheckedChanged += IsVacantField\_CheckedChanged;

*//*

*// TabControl*

*//*

TabControl.Anchor = AnchorStyles.Top | AnchorStyles.Bottom | AnchorStyles.Left | AnchorStyles.Right;

TabControl.Controls.**Add**(RoomTab);

TabControl.Controls.**Add**(BedroomTab);

TabControl.Controls.**Add**(BedTab);

TabControl.Location = new Point(5, 93);

TabControl.Name = "TabControl";

TabControl.SelectedIndex = 0;

TabControl.Size = new Size(702, 440);

TabControl.TabIndex = 38;

*//*

*// RoomTab*

*//*

RoomTab.Controls.**Add**(RoomGrid);

RoomTab.Location = new Point(4, 24);

RoomTab.Name = "RoomTab";

RoomTab.Padding = new Padding(3);

RoomTab.Size = new Size(694, 412);

RoomTab.TabIndex = 0;

RoomTab.Text = "Номера";

RoomTab.UseVisualStyleBackColor = **true**;

*//*

*// BedroomTab*

*//*

BedroomTab.Controls.**Add**(BedroomGrid);

BedroomTab.Location = new Point(4, 24);

BedroomTab.Name = "BedroomTab";

BedroomTab.Padding = new Padding(3);

BedroomTab.Size = new Size(694, 412);

BedroomTab.TabIndex = 1;

BedroomTab.Text = "Спальни";

BedroomTab.UseVisualStyleBackColor = **true**;

*//*

*// BedroomGrid*

*//*

BedroomGrid.Anchor = AnchorStyles.Top | AnchorStyles.Bottom | AnchorStyles.Left | AnchorStyles.Right;

BedroomGrid.ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.AutoSize;

BedroomGrid.Location = new Point(0, 0);

BedroomGrid.Name = "BedroomGrid";

BedroomGrid.**ReadOnly** = **true**;

BedroomGrid.RowTemplate.Height = 25;

BedroomGrid.Size = new Size(673, 418);

BedroomGrid.TabIndex = 0;

BedroomGrid.DataBindingComplete += BedroomGrid\_DataBindingComplete;

BedroomGrid.SelectionChanged += BedroomGrid\_SelectionChanged;

*//*

*// BedTab*

*//*

BedTab.Controls.**Add**(BedGrid);

BedTab.Location = new Point(4, 24);

BedTab.Name = "BedTab";

BedTab.Size = new Size(694, 412);

BedTab.TabIndex = 2;

BedTab.Text = "Спальные места";

BedTab.UseVisualStyleBackColor = **true**;

*//*

*// BedGrid*

*//*

BedGrid.Anchor = AnchorStyles.Top | AnchorStyles.Bottom | AnchorStyles.Left | AnchorStyles.Right;

BedGrid.ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.AutoSize;

BedGrid.Location = new Point(0, 0);

BedGrid.Name = "BedGrid";

BedGrid.**ReadOnly** = **true**;

BedGrid.RowTemplate.Height = 25;

BedGrid.Size = new Size(673, 418);

BedGrid.TabIndex = 0;

*//*

*// CustomersButton*

*//*

CustomersButton.Anchor = AnchorStyles.Top | AnchorStyles.Right;

CustomersButton.Location = new Point(713, 202);

CustomersButton.Name = "CustomersButton";

CustomersButton.Size = new Size(176, 23);

CustomersButton.TabIndex = 39;

CustomersButton.Text = "Гости";

CustomersButton.UseVisualStyleBackColor = **true**;

CustomersButton.Click += CustomersButton\_Click;

*//*

*// statusStrip1*

*//*

statusStrip1.Items.AddRange(new ToolStripItem[] { CurrentCustomerLabel });

statusStrip1.Location = new Point(0, 536);

statusStrip1.Name = "statusStrip1";

statusStrip1.Size = new Size(901, 22);

statusStrip1.TabIndex = 40;

statusStrip1.Text = "statusStrip1";

*//*

*// CurrentCustomerLabel*

*//*

CurrentCustomerLabel.Name = "CurrentCustomerLabel";

CurrentCustomerLabel.Size = new Size(152, 17);

CurrentCustomerLabel.Text = "Текущий гость: не выбран";

*//*

*// EditButton*

*//*

EditButton.Anchor = AnchorStyles.Top | AnchorStyles.Right;

EditButton.Location = new Point(713, 231);

EditButton.Name = "EditButton";

EditButton.Size = new Size(176, 23);

EditButton.TabIndex = 41;

EditButton.Text = "Изменить";

EditButton.UseVisualStyleBackColor = **true**;

EditButton.Click += EditButton\_Click;

*//*

*// AddButton*

*//*

AddButton.Anchor = AnchorStyles.Top | AnchorStyles.Right;

AddButton.Location = new Point(713, 260);

AddButton.Name = "AddButton";

AddButton.Size = new Size(176, 23);

AddButton.TabIndex = 42;

AddButton.Text = "Добавить";

AddButton.UseVisualStyleBackColor = **true**;

AddButton.Click += AddButton\_Click;

*//*

*// AccomodationButton*

*//*

AccomodationButton.Anchor = AnchorStyles.Top | AnchorStyles.Right;

AccomodationButton.Location = new Point(713, 318);

AccomodationButton.Name = "AccomodationButton";

AccomodationButton.Size = new Size(176, 23);

AccomodationButton.TabIndex = 43;

AccomodationButton.Text = "Заселения";

AccomodationButton.UseVisualStyleBackColor = **true**;

AccomodationButton.Click += AccomodationButton\_Click;

*//*

*// RoomAccomodationButton*

*//*

RoomAccomodationButton.Anchor = AnchorStyles.Top | AnchorStyles.Right;

RoomAccomodationButton.Location = new Point(713, 347);

RoomAccomodationButton.Name = "RoomAccomodationButton";

RoomAccomodationButton.Size = new Size(176, 23);

RoomAccomodationButton.TabIndex = 44;

RoomAccomodationButton.Text = "Заселения по комнате";

RoomAccomodationButton.UseVisualStyleBackColor = **true**;

RoomAccomodationButton.Click += RoomAccomodationButton\_Click;

*//*

*// SelectDbButton*

*//*

SelectDbButton.Anchor = AnchorStyles.Top | AnchorStyles.Right;

SelectDbButton.Location = new Point(713, 376);

SelectDbButton.Name = "SelectDbButton";

SelectDbButton.Size = new Size(176, 23);

SelectDbButton.TabIndex = 45;

SelectDbButton.Text = "Выбрать файл БД";

SelectDbButton.UseVisualStyleBackColor = **true**;

SelectDbButton.Click += SelectDbButton\_Click;

*//*

*// SaveDbButton*

*//*

SaveDbButton.Anchor = AnchorStyles.Top | AnchorStyles.Right;

SaveDbButton.Location = new Point(713, 405);

SaveDbButton.Name = "SaveDbButton";

SaveDbButton.Size = new Size(176, 23);

SaveDbButton.TabIndex = 46;

SaveDbButton.Text = "Экспорт БД";

SaveDbButton.UseVisualStyleBackColor = **true**;

SaveDbButton.Click += SaveDbButton\_Click;

*//*

*// ClearDbButton*

*//*

ClearDbButton.Anchor = AnchorStyles.Top | AnchorStyles.Right;

ClearDbButton.Location = new Point(713, 434);

ClearDbButton.Name = "ClearDbButton";

ClearDbButton.Size = new Size(176, 23);

ClearDbButton.TabIndex = 47;

ClearDbButton.Text = "Очистить БД";

ClearDbButton.UseVisualStyleBackColor = **true**;

ClearDbButton.Click += ClearDbButton\_Click;

*//*

*// TestBdButton*

*//*

TestBdButton.Anchor = AnchorStyles.Top | AnchorStyles.Right;

TestBdButton.Location = new Point(713, 463);

TestBdButton.Name = "TestBdButton";

TestBdButton.Size = new Size(176, 23);

TestBdButton.TabIndex = 48;

TestBdButton.Text = "Тестовая БД";

TestBdButton.UseVisualStyleBackColor = **true**;

TestBdButton.Click += TestBdButton\_Click;

*//*

*// DeleteButton*

*//*

DeleteButton.Anchor = AnchorStyles.Top | AnchorStyles.Right;

DeleteButton.Location = new Point(713, 289);

DeleteButton.Name = "DeleteButton";

DeleteButton.Size = new Size(176, 23);

DeleteButton.TabIndex = 49;

DeleteButton.Text = "Удалить";

DeleteButton.UseVisualStyleBackColor = **true**;

DeleteButton.Click += DeleteButton\_Click;

*//*

*// RoomForm*

*//*

AutoScaleDimensions = new SizeF(7F, 15F);

AutoScaleMode = AutoScaleMode.Font;

ClientSize = new Size(901, 558);

Controls.**Add**(DeleteButton);

Controls.**Add**(TestBdButton);

Controls.**Add**(ClearDbButton);

Controls.**Add**(SaveDbButton);

Controls.**Add**(SelectDbButton);

Controls.**Add**(RoomAccomodationButton);

Controls.**Add**(AccomodationButton);

Controls.**Add**(AddButton);

Controls.**Add**(EditButton);

Controls.**Add**(statusStrip1);

Controls.**Add**(CustomersButton);

Controls.**Add**(TabControl);

Controls.**Add**(panel1);

Controls.**Add**(splitContainer1);

Controls.**Add**(label2);

Controls.**Add**(label1);

Controls.**Add**(label3);

Controls.**Add**(RoomTypeField);

Controls.**Add**(label4);

Controls.**Add**(MinCapacityField);

Controls.**Add**(MaxCapacityField);

Name = "RoomForm";

Text = "Номера";

Load += RoomForm\_Load;

((System.ComponentModel.ISupportInitialize)RoomGrid).EndInit();

((System.ComponentModel.ISupportInitialize)MinCapacityField).EndInit();

((System.ComponentModel.ISupportInitialize)MaxCapacityField).EndInit();

((System.ComponentModel.ISupportInitialize)OnePlaceBedNumberField).EndInit();

((System.ComponentModel.ISupportInitialize)TwoPlaceBedNumberField).EndInit();

((System.ComponentModel.ISupportInitialize)MaxFloorNumberField).EndInit();

((System.ComponentModel.ISupportInitialize)MinFloorNumberField).EndInit();

((System.ComponentModel.ISupportInitialize)MaxAreaField).EndInit();

((System.ComponentModel.ISupportInitialize)MinAreaField).EndInit();

((System.ComponentModel.ISupportInitialize)MaxBathroomNumberField).EndInit();

((System.ComponentModel.ISupportInitialize)MinBathroomNumberField).EndInit();

((System.ComponentModel.ISupportInitialize)MaxBedroomNumberField).EndInit();

((System.ComponentModel.ISupportInitialize)MinBedroomNumberField).EndInit();

splitContainer1.Panel1.ResumeLayout(**false**);

splitContainer1.Panel1.PerformLayout();

splitContainer1.Panel2.ResumeLayout(**false**);

splitContainer1.Panel2.PerformLayout();

((System.ComponentModel.ISupportInitialize)splitContainer1).EndInit();

splitContainer1.ResumeLayout(**false**);

panel1.ResumeLayout(**false**);

panel1.PerformLayout();

TabControl.ResumeLayout(**false**);

RoomTab.ResumeLayout(**false**);

BedroomTab.ResumeLayout(**false**);

((System.ComponentModel.ISupportInitialize)BedroomGrid).EndInit();

BedTab.ResumeLayout(**false**);

((System.ComponentModel.ISupportInitialize)BedGrid).EndInit();

statusStrip1.ResumeLayout(**false**);

statusStrip1.PerformLayout();

ResumeLayout(**false**);

PerformLayout();

}

#endregion

**private** DataGridView RoomGrid;

**private** ComboBox RoomTypeField;

**private** Label label1;

**private** Label label2;

**private** Label label3;

**private** Label label4;

**private** NumericUpDown MinCapacityField;

**private** NumericUpDown MaxCapacityField;

**private** NumericUpDown OnePlaceBedNumberField;

**private** NumericUpDown TwoPlaceBedNumberField;

**private** Label label5;

**private** Label label6;

**private** Label label7;

**private** NumericUpDown MaxFloorNumberField;

**private** NumericUpDown MinFloorNumberField;

**private** Label label8;

**private** Label label9;

**private** NumericUpDown MaxAreaField;

**private** NumericUpDown MinAreaField;

**private** Label label10;

**private** Label label11;

**private** Label label12;

**private** NumericUpDown MaxBathroomNumberField;

**private** NumericUpDown MinBathroomNumberField;

**private** Label label13;

**private** Label label14;

**private** Label label15;

**private** NumericUpDown MaxBedroomNumberField;

**private** NumericUpDown MinBedroomNumberField;

**private** Label label16;

**private** Label label17;

**private** Label label18;

**private** SplitContainer splitContainer1;

**private** Panel panel1;

**private** CheckBox IsOnlyVacantField;

**private** MonthCalendar VacantCalendar;

**private** TabControl TabControl;

**private** TabPage RoomTab;

**private** TabPage BedroomTab;

**private** TabPage BedTab;

**private** DataGridView BedroomGrid;

**private** DataGridView BedGrid;

**private** Button CustomersButton;

**private** StatusStrip statusStrip1;

**private** ToolStripStatusLabel CurrentCustomerLabel;

**private** Button EditButton;

**private** Button AddButton;

**private** Button AccomodationButton;

**private** Button RoomAccomodationButton;

**private** Button SelectDbButton;

**private** Button SaveDbButton;

**private** Button ClearDbButton;

**private** Button TestBdButton;

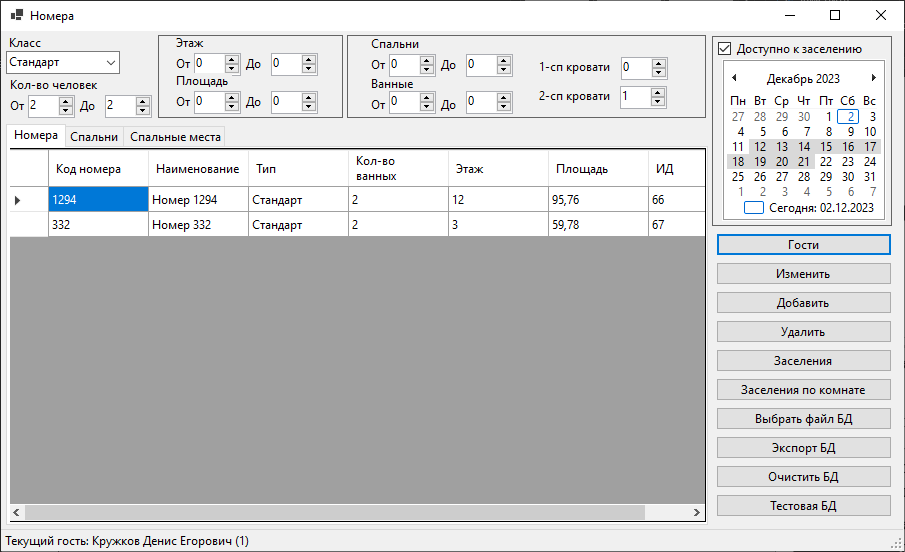
**private** Button DeleteButton;

}

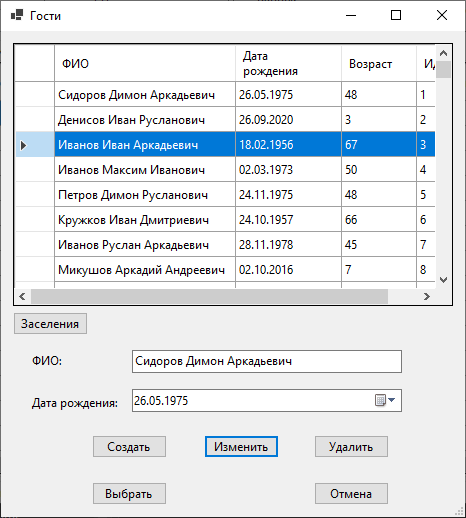
}

1. Скриншоты результатов работы (скриншоты должны отражать все ветви алгоритмов) с пояснениями

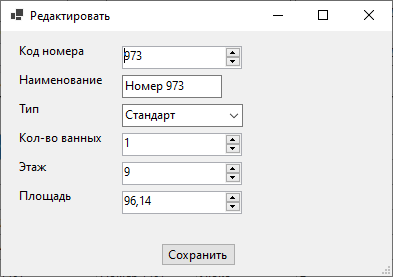
На рисунках 9-13 показан интерфейс разработанных программных форм.



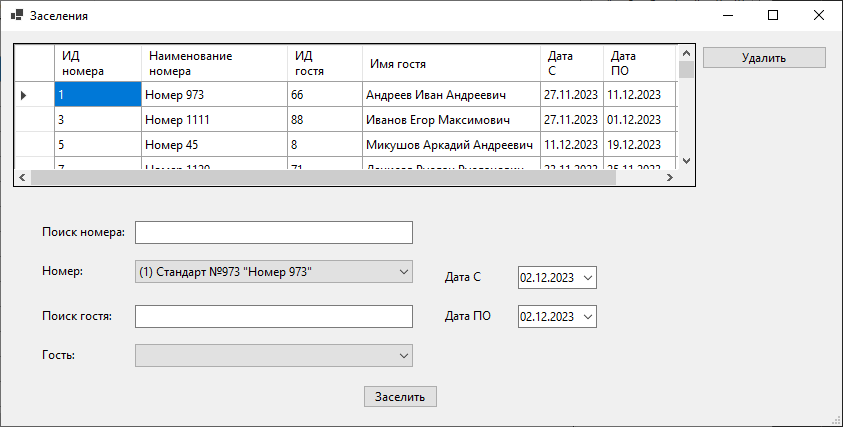
*Рисунок 9 Главное окно программы*



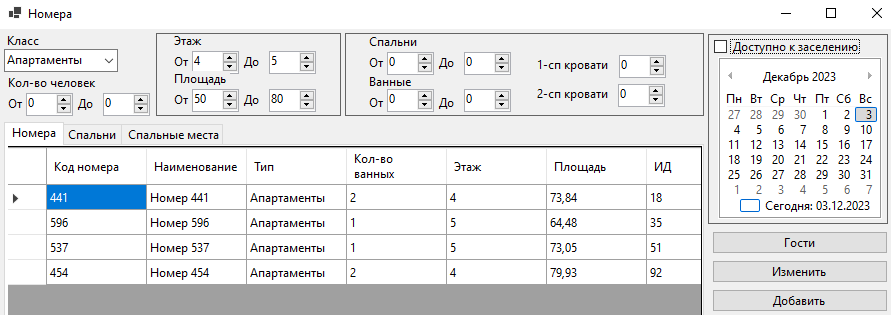
*Рисунок 10 Окно просмотра гостей*



*Рисунок 11 Окно редактирования номера*



*Рисунок 12 Окно просмотра заселений*

**

*Рисунок 13 Работа фильтров в главном окне (с помощью паттерна «Строитель»)*

ЗАКЛЮЧЕНИЕ

В результате выполнения второй лабораторной работы приобретены навыки в применении паттернов проектирования при разработке приложений с графическим пользовательским интерфейсом.

Создано приложение, позволяющее оперировать сущностями, выделенными в предметной области «Разработка иерархии классов для обеспечения работы гостиницы».

Описан процесс разработки, приведены примеры использования программы.

Цель лабораторной работы достигнута, задачи выполнены в полном объеме.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Павлов Е. В. Проектирование программных систем: методические указания к выполнению лабораторных работ / Е. В. Павлов. — Санкт-Петербург, 2023
2. Буч Г. Введение в UML от создателей языка / Грэди Буч, Джеймс Рамбо, Айвар Якобсон: пер. с англ. — ДМК Пресс, 2015 — 496 с.: ил.
3. Ларман К. Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку: пер. с англ. — М.: ИД «Вильямс», 2013. — 736 с.: ил.
4. Диаграмма активности // PlantUML URL: https://plantuml.com/ru/activity-diagram-beta (дата обращения: 08-01-2024).
5. Диаграмма классов // PlantUML URL: https://plantuml.com/ru/class-diagram (дата обращения: 08-01-2024).
6. Диаграмма последовательности // PlantUML URL: https://plantuml.com/ru/sequence-diagram (дата обращения: 08-01-2024).