

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

КУРСОВОЙ ПРОЕКТ
ЗАЩИЩЕН С ОЦЕНКОЙ
РУКОВОДИТЕЛЬ

Ст.преподаватель

Е.О. Шумова

должность, уч. степень, звание

подпись, дата

инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОМУ ПРОЕКТУ

Разработка приложения для организации взаимодействия объектов при
заданных критериях

по дисциплине: ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР.

Z1431

М.Д. Быстров

подпись, дата

инициалы, фамилия

Санкт-Петербург 2023

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

Задание
на курсовой проект по дисциплине
«Объектно-ориентированное программирование»

Студенту группы Z1431 Быстров М.Д.
№ группы Ф.И.О.

Тема «Разработка приложения для организации взаимодействия объектов при заданных критериях»

Исходные данные: Разработка иерархии классов для обеспечения работы гостиницы

Проект должен содержать:

- анализ предметной области
- разработку классов
- разработку тестового приложения
- оформление пояснительной записки по результатам выполнения проекта
- создание презентации к проекту

Срок сдачи законченного проекта _____

Руководитель проекта _____ ст.преп. Е.О.Шумова

Дата выдачи задания 01.09.2023 г.

Оглавление

Введение.....	6
1. Постановка задачи.....	7
1.1 Анализ предметной области.....	7
1.1.1 Основные сущности.....	7
1.1.2 Вспомогательные сущности.....	7
1.2 Формулировка технического задания	8
2. Проектирование классов	10
2.1 Классы сущностей.....	10
2.1.1 Класс «Room»	10
2.1.2 Класс «Bedroom»	10
2.1.3 Класс «Bed».....	11
2.1.4 Класс «Customer»	11
2.1.5 Класс «Accomodation»	12
2.1.6 Диаграмма классов.....	12
2.2 Классы хранения данных.....	13
2.2.1 Класс «BaseContext»	13
2.2.2 Класс «HostelDbContext».....	14
2.2.3 Диаграмма классов.....	15
2.3 Управляющие классы.....	16
2.3.1 Класс «IRequirement».....	16
2.3.2 Класс «CapacityRequirement».....	16
2.3.3 Класс «RoomTypeRequirement».....	16
2.3.4 Класс «BedRequirement».....	16
2.3.5 Класс «BedroomRequirement»	17
2.3.6 Класс «BathroomRequirement»	17
2.3.7 Класс «FloorNumberRequirement»	18
2.3.8 Класс «AreaRequirement»	18
2.3.9 Класс «RequirementSet».....	19
2.3.10 Класс «RequirementSetBuilder».....	19
2.3.11 Класс «RequirementRoomProvider»	19

2.3.12	Диаграмма классов.....	21
2.4	Интерфейсные классы.....	22
2.4.1	Класс «RoomForm»	22
2.4.2	Класс «CustomersForm»	23
2.4.3	Класс «AccomodationForm»	24
2.4.4	Класс «EditEntityForm».....	24
2.4.5	Диаграмма классов.....	25
2.5	Диаграмма классов	26
2.6	Используемые паттерны проектирования	27
2.6.1	Паттерн «Singleton»	27
2.6.2	Паттерн «Builder» («Строитель»).....	27
3.	Разработка приложения	28
3.1	Разработка интерфейса приложения	28
3.1.1	Главный экран программы	28
3.1.2	Окно «Гости»	31
3.1.3	Диалоговое окно «Редактировать».....	32
3.1.4	Окно «Заселения».....	33
3.2	Реализация классов	34
3.2.1	Реализация класса «BaseDbContext»	34
3.2.2	Реализация класса «HostelDbContext»	36
3.2.3	Реализация классов, реализующих интерфейс «IRequirement»	38
3.2.4	Реализация класса «RequirementSet».....	40
3.2.5	Реализация класса «RequirementSetBuilder»	40
3.2.6	Реализация класса «RequirementSetProvider».....	41
3.2.7	Реализация класса «RoomForm»	42
3.2.8	Реализация класса «CustomersForm».....	46
3.2.9	Реализация класса «AccomodationForm»	48
3.2.10	Реализация класса «EditEntityForm»	50
4.	Тестирование	55
	Заключение.....	64
	Список использованных источников	66

Приложение 1 Исходный код программы	67
---	----

Введение

Предметная область курсового проекта – обеспечение работы гостиницы. В ходе выполнения проекта должно быть спроектировано и реализовано приложение для учета и распределения по свободным номерам приезжих гостей.

Приложение должно отвечать следующим требованиям: работа с базой данных, графический интерфейс, использование концепции ООП и паттернов проектирования.

В разделах «Постановка задачи», «Проектирование классов» настоящей пояснительной записки содержатся: описание определенных на основе анализа предметной области сущностей, информация о выбранных технологиях и инструментах разработки, диаграммы разработанных иерархий классов, перечисление используемых паттернов проектирования.

Далее, в разделе «Разработка приложения» описаны детали реализации приложения, приводятся изображения, содержащие пользовательский интерфейс тестового приложения.

В разделе «Тестирование» продемонстрирована работа приложения с использованием различных наборов тестовых данных.

В Приложении 1 размещен полный исходный код реализованной программы.

1. Постановка задачи

1.1 Анализ предметной области

Предметной областью курсового проекта является работа гостиницы в части регистрации и заселения клиентов. Процесс регистрации начинается при первом получении информации о клиенте, далее регистрируется факт заселения и выселения клиента.

По итогам анализа предметной области выделены следующие основные сущности:

1.1.1 Основные сущности

- Клиент (постоялец, гость) гостиницы
- Номер гостиницы
- Спальня номера
- Кровать в спальне

1.1.2 Вспомогательные сущности

- Поставщик данных БД
- Требования к номеру
 - Требование к вместимости
 - Требование к спальным местам
 - Требование к количеству комнат
 - Требование к количеству ванных комнат
 - Требование к этажу номера
 - Требование к площади номера
 - Требования к классу номера
- Набор требований к номеру
- Строитель набора требований к номеру
- Поставщик набора номеров в соответствии с требованиями к номеру

1.2 Формулировка технического задания

Спроектировать и реализовать программу для учета гостей гостиницы и данных об их проживании.

Программа должна иметь графический пользовательский интерфейс (GUI), выполняться в операционной системе MS Windows.

Программа должна содержать несколько оконных форм.

Главная оконная форма должна содержать сетки с данными об основных сущностях предметной области:

- Номера
- Спальни
- Спальные места

В программе должно быть реализовано меню со следующими пунктами:

- Гости
- Заселения
- Заселения по комнате
- Выбор файла БД

С помощью пункта меню «Гости» пользователю должен быть предоставлен доступ к управлению справочником постояльцев гостиницы, с помощью пункта меню «Заселения», «Заселения по комнате» - доступ к управлению заселениями постояльцев (всеми или для конкретного выбранного номера).

На главном экране программы должны быть представлены элементы управления, с помощью которых пользователь программы сможет выполнять поиск подходящего гостиничного номера по различным критериям.

При заселении постояльца в номер должны быть произведены необходимые проверки возможного нарушения целостности данных (например, один гостиничный номер не может быть забронирован одновременно более чем один раз).

Язык программирования – C#. Среда разработки – Visual Studio.

Технология для создания пользовательского графического интерфейса –
Windows Forms.

2. Проектирование классов

2.1 Классы сущностей

2.1.1 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)

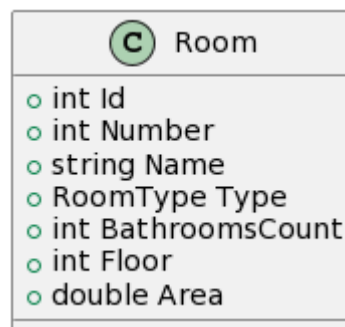


Рисунок 1 Класс Room

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.

2.1.2 Класс «Bedroom»

Класс «Bedroom» описывает спальню, принадлежащую номеру.

Описание полей:

- Id – уникальный идентификатор комнаты

- RoomId – идентификатор комнаты, которой принадлежит спальня
- Area – площадь спальни

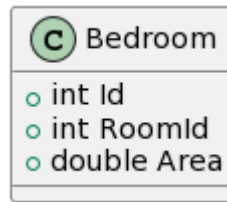


Рисунок 2 Класс Bedroom

2.1.3 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)

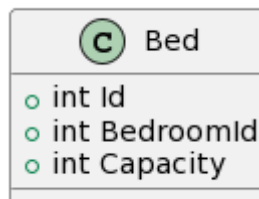


Рисунок 3 Класс Bed

2.1.4 Класс «Customer»

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

Описание полей:

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



Рисунок 4 Класс Customer

2.1.5 Класс «Accommodation»

Класс «Accommodation» описывает заселение постояльца в номер.

Описание полей:

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

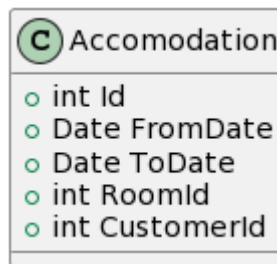


Рисунок 5 Класс Accommodation

2.1.6 Диаграмма классов

Поскольку сущности предметной области не связаны между собой отношениями, отражаемыми диаграммой классов, диаграмма классов для классов, описывающих сущности предметной области, приведена не будет.

2.2 Классы хранения данных

2.2.1 Класс «BaseDbContext»

Описание полей:

- `_databaseFullFileName` – путь к файлу базы данных

Описание методов:

- `setDatabaseFullFileName` – установить путь к файлу базы данных
- `getDatabaseFullFileName()` – получить путь к файлу базы данных
- `selectDatabaseFile()` – предоставить пользователю выбрать файл базы данных (либо имя нового файла для хранения данных)
- `getRooms` – получить номера
- `getRoom` – получить номер по идентификатору
- `addRoom` – добавить новый номер
- `updateRoom` – обновить существующий номер
- `deleteRoom` – удалить номер
- `getAccommodations` – получить все данные заселений
- `getAccommodation` – получить заселение по идентификатору
- `addAccommodation` – создать новое заселение
- `updateAccommodation` – обновить заселение
- `deleteAccommodation` – удалить заселение
- `getBedrooms` – получить спальни
- `getRoomBedrooms` – получить спальни по идентификатору номера
- `getBedroom` – получить спальню по идентификатору
- `addBedroom` – добавить спальню
- `updateBedroom` – обновить спальню
- `deleteBedroom` – удалить спальню
- `getBeds` – получить кровати
- `getBedroomBeds` – получить кровати по идентификатору спальни
- `getBed` – получить кровать по идентификатору
- `addBed` – добавить кровать

- updateBed – обновить кровать
- deleteBed – удалить кровать
- getCustomers – получить постояльцев
- getCustomer – получить постояльца по идентификатору
- addCustomer – добавить постояльца
- updateCustomer – обновить постояльца
- deleteCustomer – удалить постояльца
- clearDatabase – очистить базу данных

2.2.2 Класс «HostelDbContext»

Описание полей:

- _instance – экземпляр контекста (статическое)

Описание методов:

- getInstance – (статическое) – получить экземпляр контекста базы данных
- getVacantRooms – получить доступные комнаты на период дат
- createRoomAccommodation – создать заселение постояльца в комнату
- getRoomAccommodationOnDate – получить заселение в комнату на дату
- isRoomVacantOnDate – проверить, свободна ли комната на дату

В классе HostelDbContext реализован паттерн «Singleton». Назначение и устройство паттерна будет описано в разделе 2.5 «Используемые паттерны проектирования».

2.2.3 Диаграмма классов

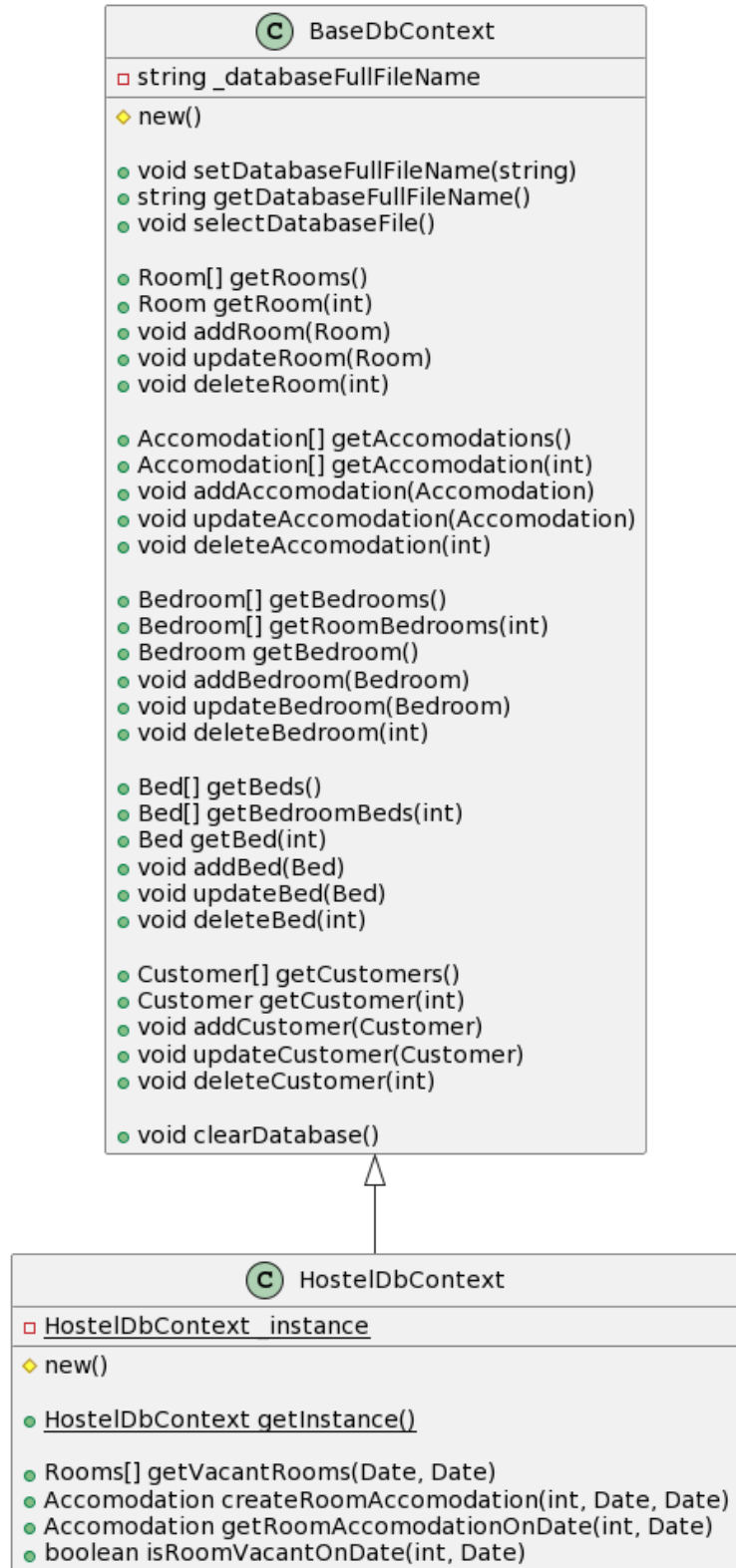


Рисунок 6 Диаграмма классов

2.3 Управляющие классы

2.3.1 Класс «IRequirement»

Класс «IRequirement» является интерфейсом для объектов, позволяющих проверить номер гостиницы на соответствие определенным требованиям.

Описание методов:

- checkRoom – проверка комнаты на соответствие требованиям

2.3.2 Класс «CapacityRequirement»

Класс «CapacityRequirement» предназначен для проверки комнаты на соответствие требованию по вместимости (кол-во человек).

Описание полей:

- minCapacity – минимально допустимая вместимость
- maxCapacity – максимально допустимая вместимость

Описание методов:

- checkRoom – проверка комнаты на соответствие требованиям
- getMinCapacity – получить минимальную вместимость
- setMinCapacity – задать минимальную вместимость
- getMaxCapacity – получить максимальную вместимость
- setMaxCapacity - задать максимальную вместимость

2.3.3 Класс «RoomTypeRequirement»

Класс «RoomTypeRequirement» - проверка номера на соответствие типа номера требуемому

Описание полей:

- roomTypes – массив допустимых типов номеров

Описание методов:

- checkRoom – проверка комнаты на соответствие требованиям
- getRoomTypes – получить массив типов номеров
- setRoomTypes – задать массив типов номеров

2.3.4 Класс «BedRequirement»

Класс «BedRequirement» - проверка наличия и вместимости кроватей

Описание полей:

- bedCount – количество кроватей
- bedCapacity – вместимость кроватей

Описание методов:

- checkRoom – проверка комнаты на соответствие требованиям
- getBedCount – получить кол-во кроватей
- setBedCount – установить кол-во кроватей
- getBedCapacity – получить вместимость кроватей
- setBedCapacity – задать вместимость кроватей

2.3.5 Класс «BedroomRequirement»

Класс «BedroomRequirement» - требование к наличию спален

Описание полей:

- minBedroomNumber – минимальное кол-во спален
- maxBedroomNumber – максимальное кол-во спален

Описание методов:

- checkRoom – проверка комнаты на соответствие требованиям
- getMaxBedroomNumber – получить максимальное кол-во спален
- setMaxBedroomNumber – задать максимальное кол-во спален
- getMinBedroomNumber – получить минимальное кол-во спален
- setMinBedroomNumber – задать минимальное кол-во спален

2.3.6 Класс «BathroomRequirement»

Класс «BathroomRequirement» - требование к кол-ву ванных комнат

Описание полей:

- minBathroomNumber – минимально допустимое кол-во ванных комнат
- maxBathroomNumber – максимально допустимое кол-во ванных комнат

Описание методов:

- checkRoom – проверка комнаты на соответствие требованиям

- getMaxBathroomNumber - получить минимально допустимое кол-во ванных комнат
- setMaxBathroomNumber - задать минимально допустимое кол-во ванных комнат
- getMinBathroomNumber - получить максимально допустимое кол-во ванных комнат
- setMinBathroomNumber - задать максимально допустимое кол-во ванных комнат

2.3.7 Класс «FloorNumberRequirement»

Класс «FloorNumberRequirement» - требование к этажу номера

Описание полей:

- minFloorNumber – минимально допустимый этаж
- maxFloorNumber – максимально допустимый этаж

Описание методов:

- checkRoom – проверка комнаты на соответствие требованиям
- getMinFloorNumber – получить минимально допустимый этаж
- setMinFloorNumber – задать минимально допустимый этаж
- getMaxFloorNumber – получить максимально допустимый этаж
- setMaxFloorNumber – задать максимально допустимый этаж

2.3.8 Класс «AreaRequirement»

Класс «AreaRequirement» - требование к площади номера

Описание полей:

- minArea – минимально допустимая площадь
- maxArea – максимально допустимая площадь

Описание методов:

- checkRoom – проверка комнаты на соответствие требованиям
- getMinArea – получить минимально допустимую площадь
- setMinArea – задать минимально допустимую площадь
- getMaxArea – получить максимально допустимую площадь

- setMaxArea – задать максимально допустимую площадь

2.3.9 Класс «RequirementSet»

Класс «RequirementSet» представляет из себя контейнер для множества экземпляров классов, реализующих интерфейс IRequirement. Назначение класса – проверить номер на соответствие нескольким требованиям.

Описание полей:

- requirements – массив требований

Описание методов:

- checkRoom – проверка комнаты на соответствие требованиям
- setRequirements – задать набор требований
- getRequirements – получить набор требований

2.3.10 Класс «RequirementSetBuilder»

Класс «RequirementSetBuilder» является классом, спроектированным с использованием паттерна «Строитель» (“Builder”). Экземпляр класса позволяет быстро сконструировать набор требований к номеру посредством последовательного вызова методов.

Описание методов:

- addCapacityRequirement – добавить требование к вместимости
- addRoomTypeRequirement – добавить требование к типу номера
- addBedRequirement – добавить требование к спальным местам
- addFloorNumberRequirement – добавить требование к этажу
- addAreaRequirement – добавить требование к площади
- addBedroomRequirement – добавить требование к спальням
- addBathroomRequirement – добавить требование к ванным комнатам
- buildRequirementSet – построить RequirementSet

2.3.11 Класс «RequirementRoomProvider»

Класс «RequirementRoomProvider» позволяет предоставить набор номеров в соответствии с заданными требованиями.

Описание полей:

- requirements – набор требований

Описание методов:

- getRooms – получить комнаты, соответствующие требованиям
- getVacantRooms – получить только свободные для заселения комна

2.3.12 Диаграмма классов

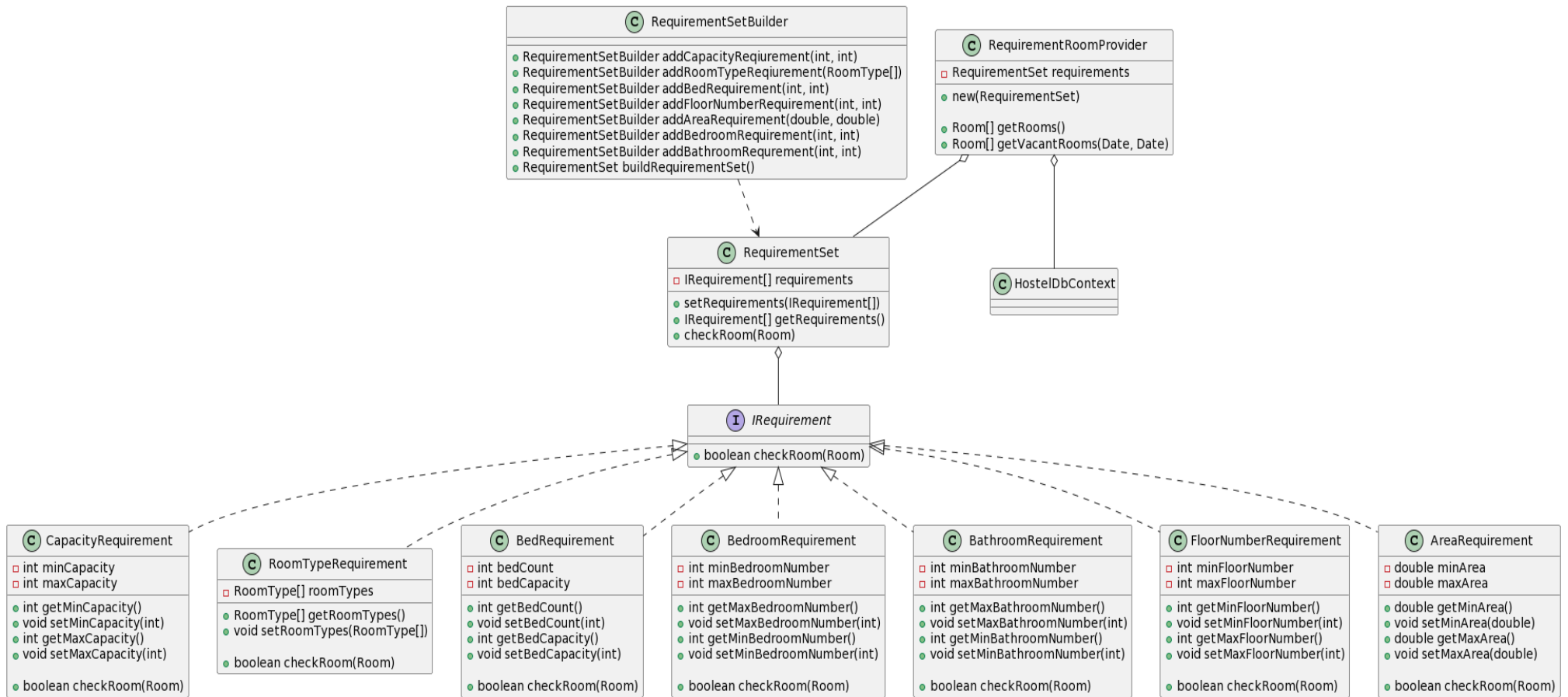


Рисунок 7 Диаграмма классов

2.4 Интерфейсные классы

2.4.1 Класс «RoomForm»

Класс «RoomForm» представляет из себя класс обработки пользовательского ввода на главной форме приложения.

Описание полей:

Первый тип полей – экранные фильтры для поиска номеров в соответствии с требованиями.

- RoomTypeField – выбор типа номера
- MinCapacityField – выбор минимальной вместимости номера
- MaxCapacityField – выбор максимальной вместимости номера
- OnePlaceBedNumberField – выбор кол-ва односпальных кроватей
- TwoPlaceBedNumberField – выбор кол-ва двуспальных кроватей
- MinBathroomNumberField – выбор минимального кол-ва ванных комнат
- MaxBathroomNumberField – выбор максимального кол-ва ванных комнат
- MinAreaNumberField – выбор минимальной площади номера
- MaxAreaNumberField – выбор максимальной площади номера
- MinFloorNumberField – выбор минимального этажа номера
- MaxFloorNumberField – выбор максимального этажа номера
- MinBedroomNumberField – выбор минимального кол-ва спален в номере
- MaxBedroomNumberField – выбор максимального кол-ва спален в номере

Также на форме присутствуют элементы отображения данных и пользовательского ввода:

- RoomGrid – сетка данных комнат
- BedroomGrid – сетка данных спален
- BedGrid – сетка данных кроватей

- AccomodationGrid – сетка данных заселений
- CreateRoomButton – кнопка «Создать номер»
- EditRoomButton – кнопка «Редактировать номер»
- DeleteRoomButton – кнопка «Удалить номер»
- CreateBedroomButton – кнопка «Создать спальню»
- EditBedroomButton – кнопка «Редактировать спальню»
- DeleteBedroomButton – кнопка «Удалить спальню»
- CreateBedButton – кнопка «Создать кровать»
- EditBedButton – кнопка «Редактировать кровать»
- DeleteBedButton – кнопка «Удалить кровать»
- CreateAccomodationButton – кнопка «Заселить гостя»
- DeleteAccomodationButton – кнопка «Отменить заселение»
- CustomersButton – кнопка «Гости»
- ApplyFiltersButton – кнопка «Применить фильтры»

Описание методов:

- getRequiredRooms – получить комнаты, соответствующие требованиям, указанным в экранных фильтрах. Вызывается при нажатии на кнопку «Применить фильтры»

2.4.2 Класс «CustomersForm»

Класс «CustomersForm» содержит управляющие элементы формы «Гости», на которой доступно редактирование персональных данных гостей гостиницы.

Описание полей:

- CustomerGrid – сетка с данными гостей
- NameField – поле для ввода ФИО
- BirthdayField – поле для ввода даты рождения
- EditButton – кнопка «Редактировать»
- CreateButton – кнопка «Создать»
- RemoveButton – кнопка «Удалить»

2.4.3 Класс «AccommodationForm»

Класс «AccommodationForm» описывает элементы формы для заселения гостя.

Описание полей:

- AccommodationGrid – сетка с данными заселений
- DeleteAccommodationButton – кнопка для удаления заселения
- FromDateField – поле начальной даты заселения
- ToDateField – поле конечной даты заселения
- CustomerIdField – поле идентификатора гостя
- CustomerNameField – поле ФИО гостя
- CreateAccommodationButton – кнопка «Создать заселение»
- CreateCustomerButton – кнопка «Гости» (переход к форме «Гости»)

Описание методов:

- createAccommodation – создание заселения. Используется экземпляр HostelDbContext

2.4.4 Класс «EditEntityForm»

Класс «EditEntityForm» представляет из себя класс формы для создания/редактирования сущностей БД пользователем (номера, спальни, кровати). Наполнение формы осуществляется динамически на основании полей сущности.

Описание полей:

- entity – при редактировании – данные редактируемой сущности
- SaveButton – кнопка «Сохранить»

Описание методов:

- saveEntity – сохранить/обновить сущность в БД. Используется HostelDbContext

2.4.5 Диаграмма классов

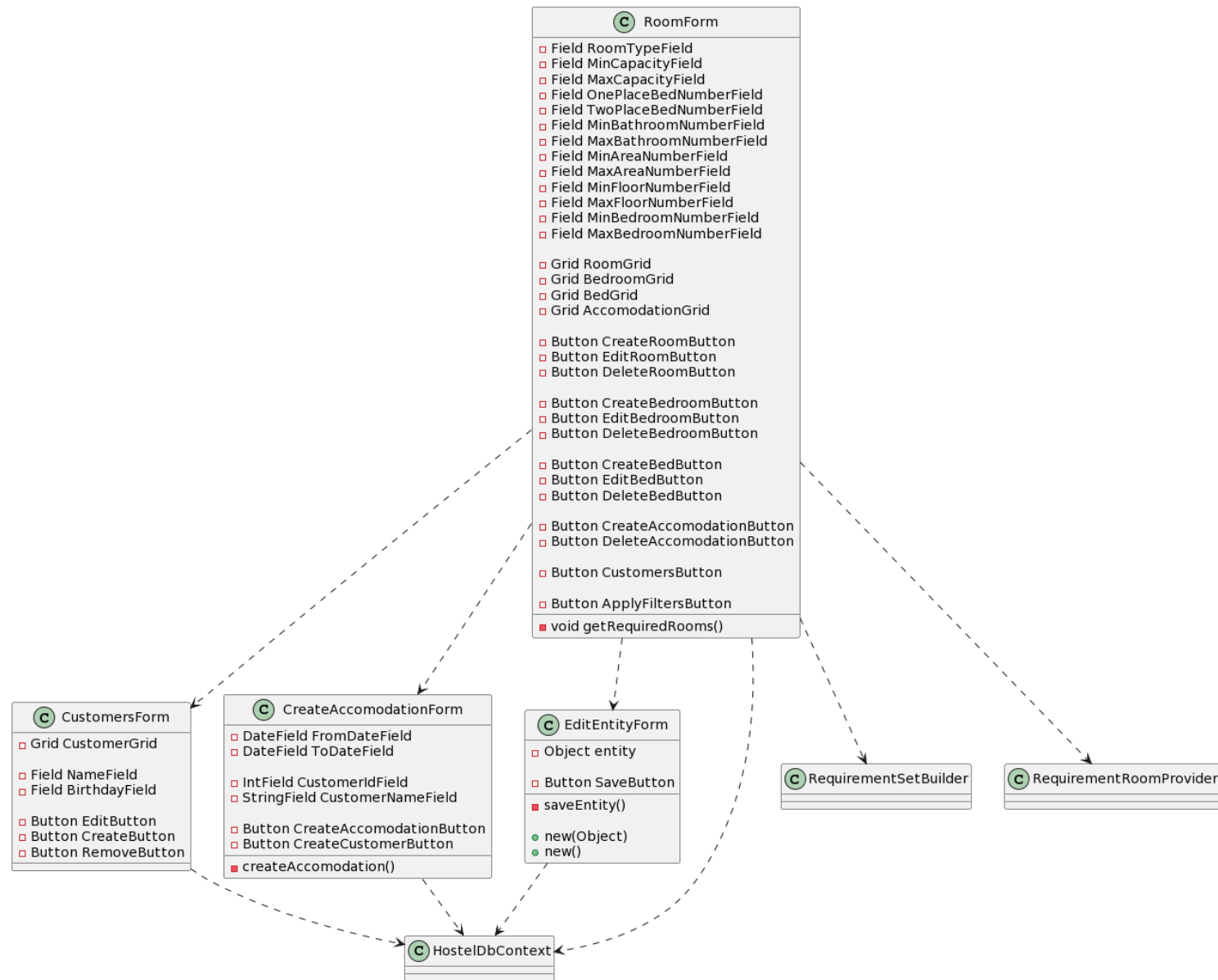


Рисунок 8 Диаграмма классов

2.5 Диаграмма классов

В дополнение к отдельным диаграммам классов для каждой группы классов на рисунке 9 приведена общая диаграмма классов, на которой классы отображены в сокращенном виде.

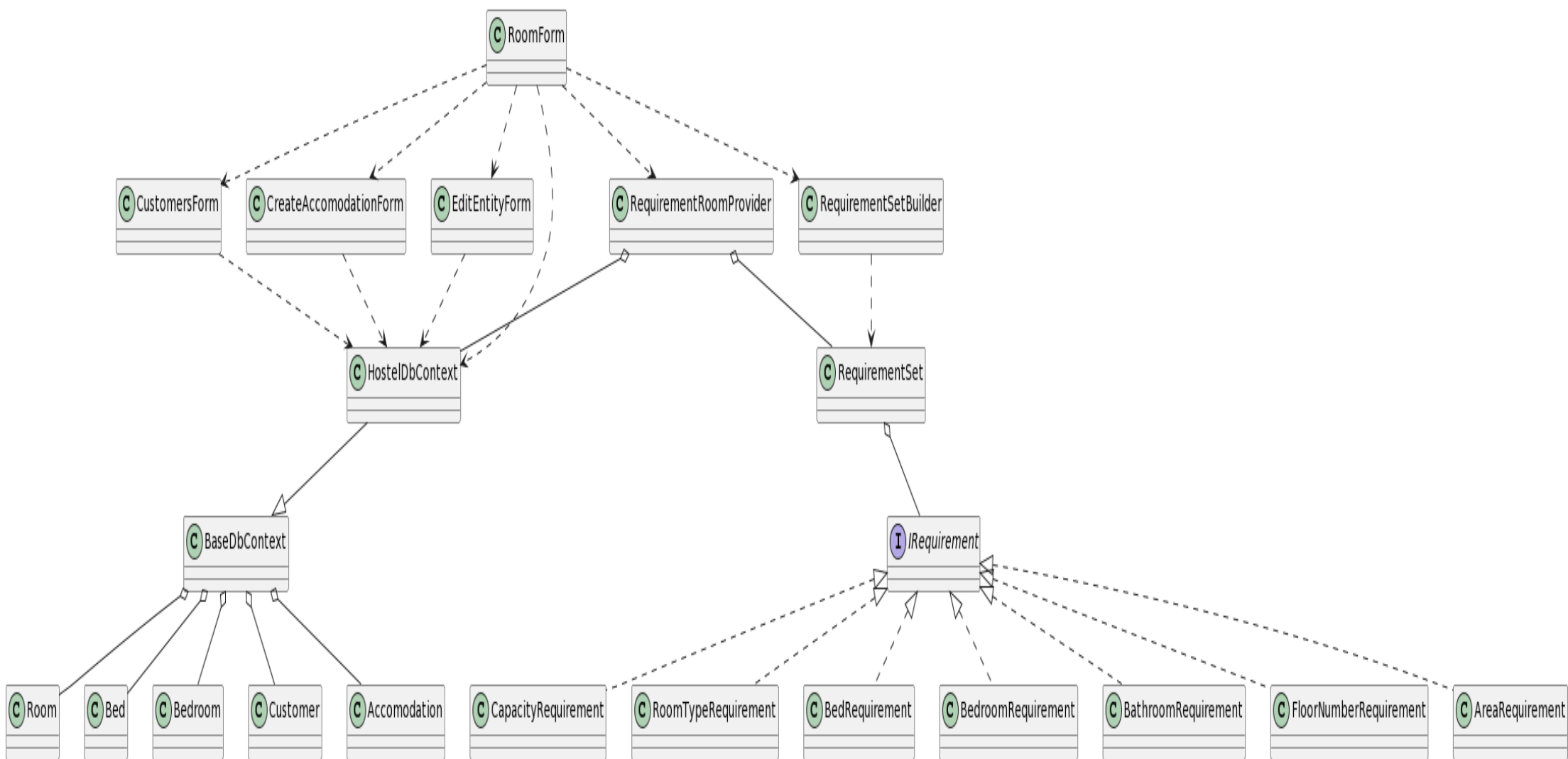


Рисунок 9 Общая диаграмма классов приложения

2.6 Используемые паттерны проектирования

2.6.1 Паттерн «Singleton»

С помощью паттерна Singleton спроектирован класс `HostelDbContext`. Диаграмма класса для паттерна показана на рисунке 6. Суть паттерна заключается в существовании только одного экземпляра класса на протяжении работы программы. Реализуется паттерн с помощью статических методов, полей и примитивов синхронизации потоков.

Использование паттерна позволяет получить доступ к объекту из любого места в приложении, в данном случае к поставщику данных из БД, что позволит избежать множественных конкурентных чтений/записей в файл БД, а также упростит архитектуру приложения.

2.6.2 Паттерн «Builder» («Строитель»)

Паттерн «Builder» позволяет упростить конструирование сложных объектов путем вызова соответствующих методов. Это позволяет избежать множественных вызовов методов конструируемого объекта (для предоставления зависимостей) после его создания, либо передачи большого количества переменных в конструктор, если класс конструируемого объекта спроектирован таким образом, чтобы невозможно было создать экземпляр без предоставления всех необходимых зависимостей. Класс, реализованный в соответствии с паттерном, упрощает конструирование. Как правило, каждый метод, предназначенный для настройки, возвращает указатель на сам экземпляр «Builder», что позволяет вовсе не создавать лишних локальных переменных. В курсовом проекте паттерн применен при проектировании класса `HostelDbContext`. Диаграмма классов для паттерна показана на рис.7.

Часто паттерн используется в тех случаях, когда необходимо конструировать схожие (но разные по типу, данным) объекты разным способом. В таком случае существует несколько «Builder»-классов, которые реализуют один интерфейс или унаследованы от одного базового класса. В данном курсовом проекте этот подход в использовании паттерна не применен.

3. Разработка приложения

3.1 Разработка интерфейса приложения

Программа имеет графический интерфейс, построенный с помощью фреймворка WinForms, являющегося частью фреймворка .NET. Фреймворк представляет из себя «обертку» над Win32 API, позволяющую создавать GUI-приложения для Windows, работающие в среде с управляемой памятью.

3.1.1 Главный экран программы

Спроектированный интерфейс главного экрана программы представлен на рисунке 10. В соответствии с техническим заданием на главном экране представлены сетки данных, элементы интерфейса, выполняющих функции фильтров, а также пункты меню, выполненные в виде кнопок в правой части интерфейса.

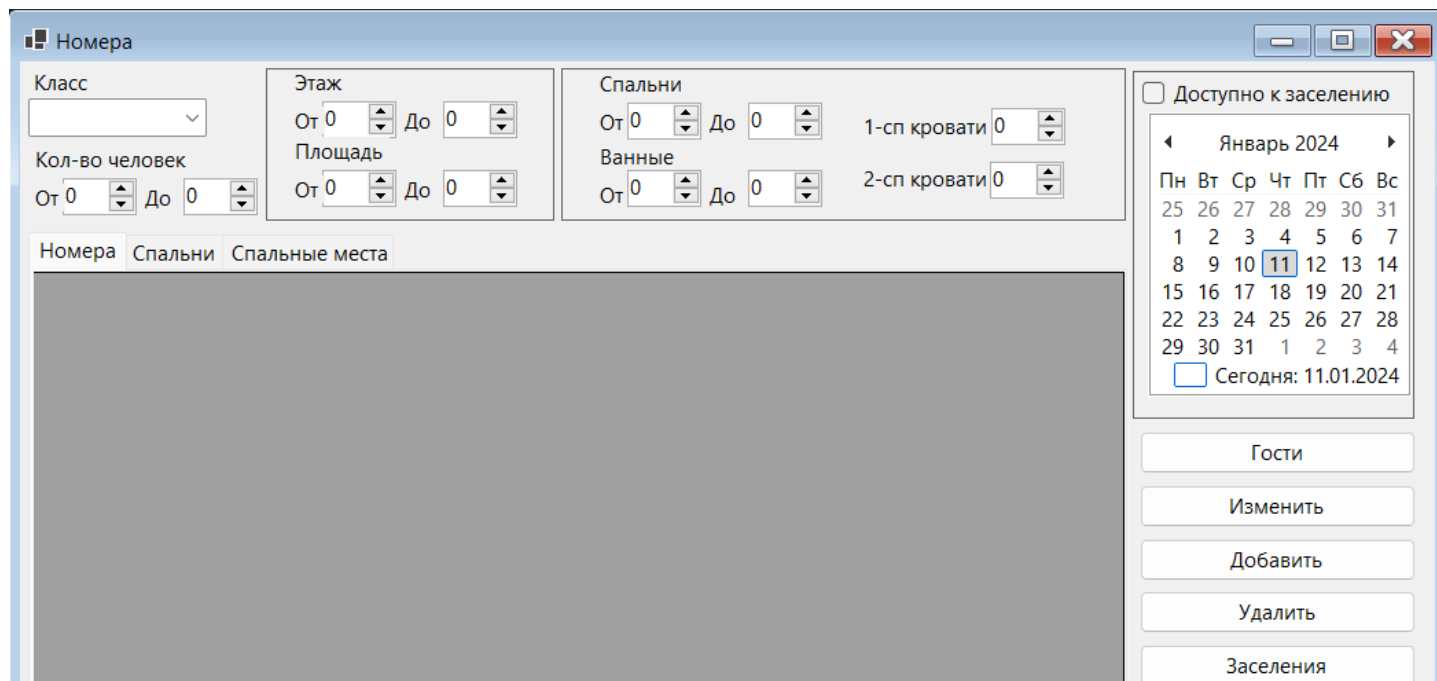


Рисунок 10 Главный экран программы

На главном экране программы представлены данные о номерах, доступных для заселения гостей. Для просмотра данных о номерах на форме расположены три сетки с данными на трех вкладках: «Номера», «Спальни», «Спальные места». Данные на второй и третьей вкладке отображаются в контексте выбранной строки на первой и второй вкладке соответственно.

В верхней части интерфейса расположены фильтры для подбора номеров в соответствии с требованиями гостя. Справа от фильтров расположен календарь, при использовании которого отображаются только номера, доступные для заселения на выбранный период проживания.

Для отображения сетки с данными использован элемент интерфейса DataGridView. Элемент позволяет отображать данные с в виде таблицы. На форме элемент представлен в виде сеток с данными комнат, спален, спальных мест.

Для реализации выпадающего списка использован элемент интерфейса ComboBox. Элемент позволяет предоставить пользователю выбор из нескольких вариантов. На форме представлен в виде выпадающих списков фильтров.

Для ввода числовых значений используется элемент NumericUpDown, который позволяет пользователю вводить числовые значения. Также используется для отображения фильтров.

Для отображения календаря используется элемент MonthCalendar. Элемент используется для выбора доступных для заселения дат.

Вкладки реализованы с помощью элемента TabControl. Элемент позволяет разделить пользовательский интерфейс на несколько вкладок. Внутри вкладок на форме расположены сетки с данными.

Для пунктов меню используется элемент интерфейса Button. Кнопки на форме расположены в правой части формы.

Кнопки имеют следующий функционал:

- «Гости»

Кнопка открывает окно со списком гостей.

- «Изменить»

Кнопка открывает диалоговое окно, позволяющее изменить выбранную в сетке запись.

- «Добавить»

Кнопка открывает диалоговое окно, позволяющее добавить запись в активную сетку.

- «Удалить»

Кнопка удаляет выбранную запись в активной сетке.

- «Заселения»

Кнопка открывает окно со списком существующих заселений.

- «Заселения по комнате»

Действие кнопки аналогично действию кнопки «Заселения», но окно открывается с данными только по выбранному номеру.

- «Выбрать файл БД»

Открывается диалоговое окно выбора файла базы данных. Можно как выбрать существующий файл, так и ввести имя нового файла.

- «Экспорт БД»

Кнопка открывает окно выбора файла. Выбранное имя файла будет использовано для создания копии файла базы данных.

- «Очистить БД»

При нажатии кнопки все данные в текущем файле базы данных удаляются.

- «Тестовая БД»

При нажатии кнопки все данные в текущей БД удаляются, база данных наполняется сгенерированным набором данных.

В строке состояния в нижней части окна отображается имя текущего пользователя.

3.1.2 Окно «Гости»

На окне «Гости» расположены данные и элементы управления для учета данных гостей гостиницы. Функционал оконной формы в соответствии с техническим заданием соответствует управлению справочником – реализует операции чтения, создания, обновления, удаления данных.

На сетке в верхней части формы представлен список пользователей.

По кнопке «Заселения» открывается форма «Заселения» с данными заселений выбранного пользователя.

Ниже располагаются элементы управления, позволяющие создать нового гостя или отредактировать существующего: поле ввода ФИО, поле ввода даты рождения, кнопки «Создать», «Изменить», «Удалить».

Кнопка «Выбрать» при нажатии устанавливает выбранного пользователя как текущего в сеансе работы с программой.

При нажатии кнопки «Отмена» окно «Заселения» закрывается.

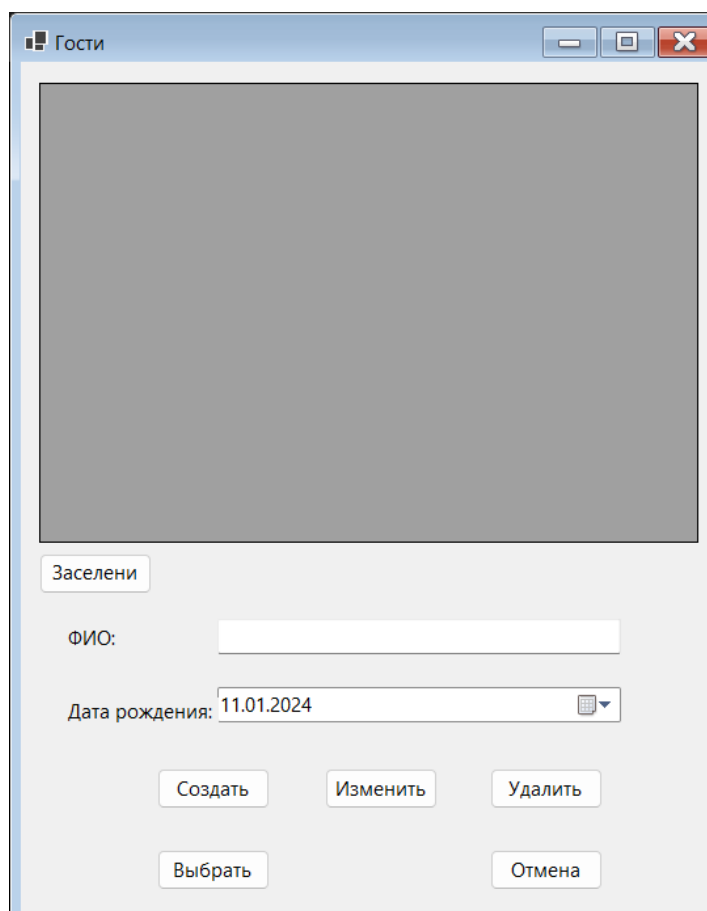


Рисунок 11 Окно «Гости»

На форме для вывода сетки с данными пользователей использован элемент интерфейса DataGridView.

Для выбора даты рождения использован элемент DateTimePicker.

Для ввода имени использован элемент TextBox, который позволяет вводить строку текста.

Также для обработки пользовательского ввода использован элемент Button.

3.1.3 Диалоговое окно «Редактировать»

Для редактирования или создания сущностей создано диалоговое окно «Редактировать». Поля на форме зависят от свойств объекта, передаваемого в конструктор при создании экземпляра класса формы.

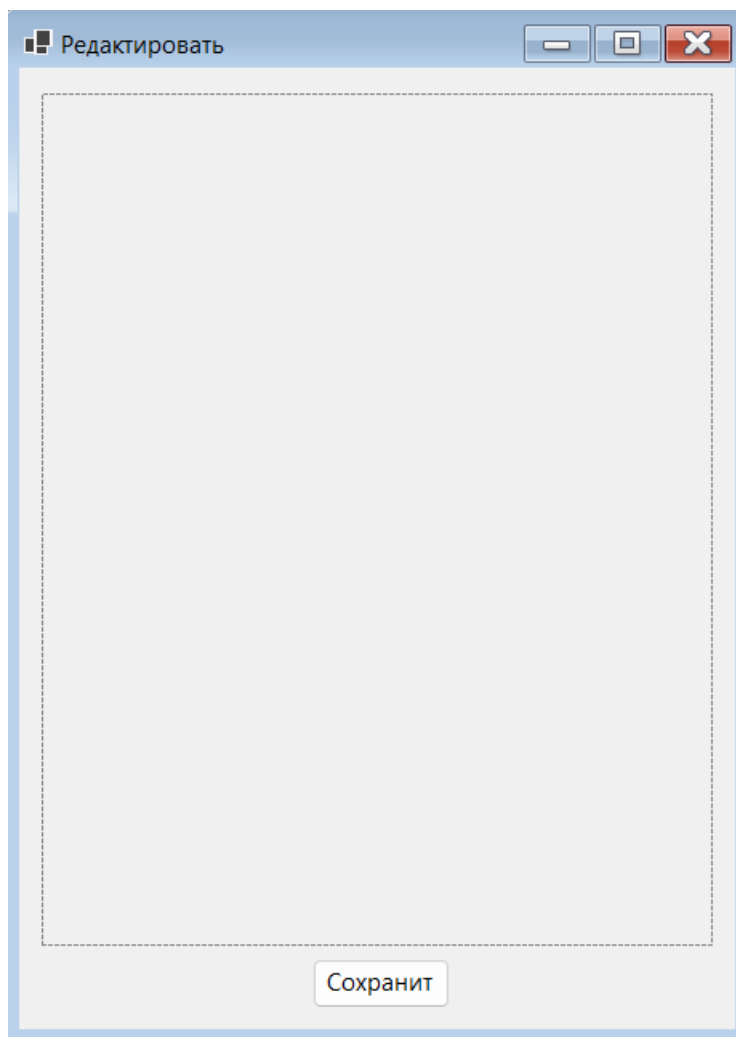


Рисунок 12 Диалоговое окно редактирования

3.1.4 Окно «Заселения»

С помощью окна «Заселения» пользователю представляется возможность просмотра, создания и редактирования периодов заселения гостей в номерах. На форме присутствуют: сетка с данными заселений, кнопка удаления заселения, поля поиска номера и гостя, выпадающие списки для выбора номера и гостя, календари для выбора периода дат, кнопка «Заселить» для заселения выбранного гостя в выбранный номер.

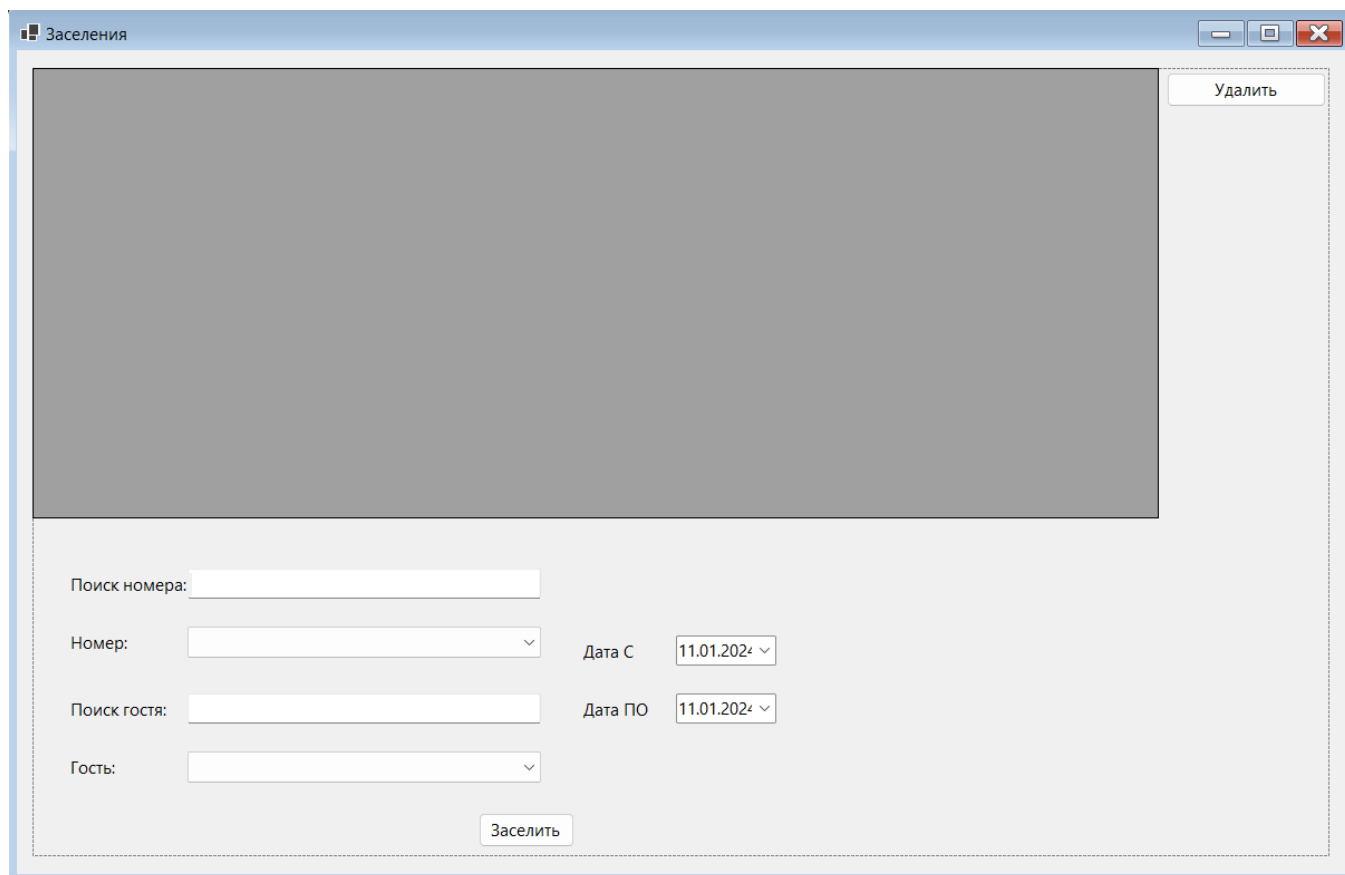


Рисунок 13 Окно «Заселения»

Для взаимодействия пользователя с формой использованы следующие элементы графического интерфейса: DataGridView, TextBox, ComboBox, DateTimePicker, Button, TextBox, Button. Их назначение было описано ранее.

3.2 Реализация классов

3.2.1 Реализация класса «BaseDbContext»

Класс «BaseDbContext» является базовым классом, предоставляющим возможность хранения данных. При разработке для хранения данных использовались средства сериализации объектов в JSON, содержащиеся в стандартной библиотеке платформы .NET (System.Text.Json.JsonSerializer). Для хранения данных использовались параметризованные коллекции во внутреннем классе RootScheme:

Листинг 1

```
1. public class RootScheme
2. {
3.     public List<Room> Rooms { get; set; } = new();
4.
5.     public List<Customer> Customers { get; set; } = new();
6.
7.     public List<Bedroom> Bedrooms { get; set; } = new();
8.
9.     public List<Bed> Beds { get; set; } = new();
10.
11.    public List<Accommodation> Accomodations { get; set; } = new();
12. }
```

Каждый из полей класса RootScheme представляет собой коллекцию с данными сущностей, которые нужны для работы программы. Класс BaseDbContext реализует методы, описанные в п. 2.2.1 с помощью добавления, удаления и редактирования объектов в свойствах объекта типа RootScheme. Внутренний метод GetEntities позволяет получить коллекцию, содержащую объекты определенного типа. Методы UpdateEntities, DeleteEntity реализованы по тому же принципу.

Листинг 2

```
1. private async Task<List<T>> GetEntities<T>() where T : Entity
2. {
3.     var scheme = await FetchData();
4.
5.     foreach (var prop in scheme.GetType().GetProperties())
6.     {
7.         if (prop.PropertyType == typeof(List<T>))
```

```

8.         {
9.             return ((List<T>)prop.GetValue(scheme!))
10.                .OrderBy(e => e.Id)
11.                .ToList();
12.        }
13.    }
14.
15.    throw new NullReferenceException();
16.}

```

Метод является явно параметризованным, т.е. при его использовании явно указывается требуемый тип. Пример использования содержится в методе `GetRooms`.

Листинг 3

```

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

```

На примере метода `GetRooms` видно, что реализация, основанная на рефлексии, позволяет реализовать большинство методов для взаимодействия с базой данных с помощью вызова одной функции.

При работе с объектом класса для сохранения данных в файл БД необходимо вызывать функцию `SaveChanges()`. Для упрощения работы был создан внутренний класс `WorkingSession`, реализующий интерфейс `IDisposable`. При объявлении переменных классов, реализующих интерфейс `IDisposable`, с ключевым словом `using` метод `Dispose` будет вызван при выходе из области видимости переменной.

Листинг 4

```

1. public async Task SaveChanges()
2. {
3.     await SaveData();
4. }
5.
6. private Task SaveData()
7. {
8.     using var fileStream = new FileStream(
9.         _databaseFullFileName,
10.        FileMode.Open,
11.        FileAccess.Write);
12.

```

```

13.     JsonSerializer.Serialize(fileStream, _scheme);
14.
15.     return Task.CompletedTask;
16. }
17.
18. public WorkingSession BeginSession() => new WorkingSession(this);
19.
20. public class WorkingSession : IDisposable
21. {
22.     private readonly BaseDbContext _baseDbContext;
23.
24.     public WorkingSession(BaseDbContext baseDbContext)
25.     {
26.         this._baseDbContext = baseDbContext;
27.     }
28.
29.     public void Dispose()
30.     {
31.         _baseDbContext.SaveChanges().Wait();
32.     }
33. }

```

3.2.2 Реализация класса «HostelDbContext»

Задача класса – предоставить выполнения операций с данными, связанными требованиями бизнес-процессов или предметной области. Также класс реализует паттерн «Singleton».

Листинг 5

```

1. public class HostelDbContext : BaseDbContext
2. {
3.     private static readonly HostelDbContext _instance = new
        HostelDbContext();
4.
5.     static HostelDbContext()
6.     {
7.     }
8.
9.     private HostelDbContext()
10.    {
11.    }
12.
13.    public static HostelDbContext GetInstance()
14.    {
15.        return _instance;
16.    }
17. }

```

Реализация паттерна выполнена с помощью статического поля и статического инициализатора. Такой подход гарантирует создание экземпляра класса в момент запуска программы до первого возможного вызова метода `GetInstance`, ввиду этого проверка на существование экземпляра и его создание с использованием примитивов синхронизации в методе `GetInstance` не требуется.

Метод `GetVacantRooms` позволяет получить только доступные для заселения комнаты в диапазоне дат. Алгоритм: запрашиваются все заселения, пересекающиеся с запрошенными датами, сохраняется набор идентификаторов комнат, имеющих конфликтующие периоды проживания, затем отбираются только те комнаты, идентификаторы которых отсутствуют в собранном наборе.

Листинг 6

```
1. public async Task<List<Room>> GetVacantRooms (
2.     DateTime fromDate,
3.     DateTime toDate)
4. {
5.     var clearFromDate = fromDate.Date;
6.     var clearToDate = toDate.Date;
7.
8.     var rooms = await GetRoomsAsync ();
9.
10.    var accomodations = (await GetAccomodationsAsync ())
11.        .Where(acc =>
12.            {
13.                return acc.FromDate < toDate && acc.ToDate > fromDate;
14.            })
15.        .Select(acc => acc.RoomId)
16.        .ToHashSet ();
17.
18.    var result = rooms.Where(r =>
19.        {
20.            return !accomodations.Contains(r.Id);
21.        });
22.
23.    return result.ToList ();
24. }
```

Метод CreateRoomAccommodation позволяет создать заселение, выполнив проверки на пересечение интервалов с помощью метода GetVacantRooms.

Листинг 7

```
1. public async Task<Accommodation> CreateRoomAccommodationAsync(  
2.     int roomId,  
3.     DateTime fromDate,  
4.     DateTime toDate,  
5.     int customerId)  
6. {  
7.     var clearFromDate = fromDate.Date;  
8.     var clearToDate = toDate.Date;  
9.  
10.    if (clearFromDate >= clearToDate)  
11.    {  
12.        throw new ApplicationException("Даты указаны неверно");  
13.    }  
14.  
15.    var vacant = await GetVacantRooms(clearFromDate, clearToDate);  
16.  
17.    if (!vacant.Any(r => r.Id == roomId))  
18.    {  
19.        throw new ApplicationException($"Комната занята на даты  
    {clearFromDate}-{clearToDate}");  
20.    }  
21.  
22.    var acc = new Accommodation()  
23.    {  
24.        CustomerId = customerId,  
25.        FromDate = clearFromDate,  
26.        ToDate = clearToDate,  
27.        RoomId = roomId  
28.    };  
29.  
30.    await AddAccommodationAsync(acc);  
31.  
32.    return acc;  
33. }
```

3.2.3 Реализация классов, реализующих интерфейс «IRequirement»

Классы, реализующие интерфейс «IRequirement», могут использоваться для проверки гостиничного номера на соответствие определенным требованиям. Для реализации интерфейса необходимо создать метод CheckRoom.

Листинг 8

```
1. public interface IRequirement
2. {
3.     public Task<bool> CheckRoom(Room room);
4. }
```

Выполняемые в реализующих интерфейс классах проверки не содержат сложных алгоритмов и в основном состоят из обращений к базе данных либо из обработки значений свойств переданного в метод объекта Room. В качестве примера реализации интерфейса можно рассмотреть класс BedRequirement. Класс предназначен для проверки минимального количества спальных мест заданной вместимости в номере гостиницы. В методе CheckRoom производится поиск всех спален номера, затем для каждой спальни запрашиваются спальные места. В конце полученные спальные места отфильтровываются по вместимости в соответствии с параметрами проверки. Проверка пройдена (возвращается true), если количество найденных спальных мест не меньше, чем требуемое пользователем.

Листинг 9

```
1. public class BedRequirement : IRequirement
2. {
3.     public int BedCount { get; set; }
4.
5.     public int BedCapacity { get; set; }
6.
7.     public async Task<bool> CheckRoom(Room room)
8.     {
9.         var context = HostelDbContext.GetInstance();
10.
11.         var beds = (await context.GetRoomBedroomsAsync(room.Id))
12.             .Aggregate(
13.                 new List<Bed>(),
14.                 (list, bedroom) =>
15.                 {
16.                     list.AddRange(context
17.                         .GetBedroomBedsAsync(bedroom.Id)
18.                         .GetAwaiter()
19.                         .GetResult());
20.
21.                     return list;
22.                 }
23.             );
24.         return list.Count >= BedCount;
25.     }
26. }
```

```

22.         })
23.         .Where(b => b.Capacity == BedCapacity);
24.
25.         return beds.Count() >= BedCount;
26.     }
27. }

```

Остальные проверки выполнены аналогичным образом. Их код можно увидеть в приложении с исходным кодом.

3.2.4 Реализация класса «RequirementSet»

Класс предназначен для хранения нескольких объектов классов, реализующих интерфейс `IRequirement`, и проверки гостиничного номера на соответствие с помощью всех содержащихся проверок.

Проверка происходит при вызове метода `CheckRoom`. Проверки последовательно вызываются, переданный в качестве аргумента метода объект гостиничного номера передается в метод проверки. Как только одна из проверок не выполняется успешно, проверка заканчивается неудачно. Только если все проверки пройдены успешно, возвращается значение `true`.

Листинг 10

```

1. public async Task<bool> CheckRoom(Room room)
2. {
3.     foreach (var requirement in Requirements)
4.     {
5.         if (!(await requirement.CheckRoom(room)))
6.         {
7.             return false;
8.         }
9.     }
10.
11.     return true;
12. }

```

3.2.5 Реализация класса «RequirementSetBuilder»

Класс реализует паттерн «Builder». Задача класса – создание экземпляров класса «RequirementSet». Конструирование результата работы класса осуществляется посредством последовательного вызова методов, отвечающих за создание определенной проверки, одной из тех, что реализуют интерфейс

«IRequirement». В каждом из методов создается экземпляр класса проверки определенного типа и добавляется в список проверок. В качестве примера приведены методы `AddBathroomRequirement`, `BuildRequirementSet`. Первый метод является примером метода, добавляющего одну из проверок в список (проверка на наличие ванных комнат в гостиничном номере). Второй метод – финальный метод, вызывающийся в конце работы с объектом класса, он возвращает построенный на основе списка объект класса «RequirementSet».

Листинг 11

```
1. public RequirementSetBuilder AddBathroomRequirement(  
2.     int minBathrooms,  
3.     int maxBathrooms)  
4. {  
5.     var requirement = new BathroomRequirement()  
6.     {  
7.         MinBathroomNumber = minBathrooms,  
8.         MaxBathroomNumber = maxBathrooms  
9.     };  
10.  
11.     _requirements.Add(requirement);  
12.  
13.     return this;  
14. }  
15.  
16. public RequirementSet BuildRequirementSet()  
17. {  
18.     return new RequirementSet(_requirements);  
19. }
```

3.2.6 Реализация класса «RequirementSetProvider»

Класс является вспомогательным; его задача – предоставить гостиничные номера в соответствии с предоставленным набором проверок.

В методе `GetRooms` из базы данных запрашиваются все номера, которые затем проверяются с помощью набора проверок, возвращаются только те номера, которые соответствуют требованиям. В методе `GetVacantRooms` таким же образом обрабатываются только те номера, которые доступны для заселения в период дат.

Листинг 12

```

1. public async Task<List<Room>> GetRoomsAsync()
2. {
3.     var context = HostelDbContext.GetInstance();
4.
5.     var rooms = (await context.GetRoomsAsync())
6.         .Where(r => _requirementSet.CheckRoom(r)
7.             .GetAwaiter()
8.             .GetResult())
9.         .ToList();
10.
11.     return rooms;
12. }
13.
14. public async Task<List<Room>> GetVacantRoomsAsync(
15.     DateTime fromDate,
16.     DateTime toDate)
17. {
18.     var context = HostelDbContext.GetInstance();
19.
20.     var rooms = (await context.GetVacantRooms(
21.         fromDate,
22.         toDate))
23.         .Where(r => _requirementSet.CheckRoom(r)
24.             .GetAwaiter()
25.             .GetResult())
26.         .ToList();
27.
28.     return rooms;
29. }

```

3.2.7 Реализация класса «RoomForm»

Класс RoomForm является наследником класса фреймворка WinForms System.Windows.Forms.Form. Класс является интерфейсным, содержит описание элементов пользовательского графического интерфейса и обработку пользовательского ввода.

Для отображения данных на форме присутствуют три сетки с данными, расположенные на различных вкладках. При обновлении данных в сетках используются значения фильтров, расположенных в верхней части формы. Значения фильтров учитываются в методе ExecuteRoomQuery, который выполняет обновление данных на первой вкладке формы. Методы ExecuteBedroomQuery, ExecuteBedQuery выполняются после метода

ExecuteRoomQuery и обновляют данные на второй и третьей вкладках соответственно.

Листинг 13

```
1. private async Task ExecuteRoomQuery()
2. {
3.     var requirementSet = new RequirementSetBuilder()
4.         .AddRoomTypeRequirement(
5.             new RoomType[] { (RoomType)RoomTypeField.SelectedIndex })
6.         .AddFloorNumberRequirement(
7.             (int)MinFloorNumberField.Value,
8.             (int)MaxFloorNumberField.Value)
9.         .AddAreaRequirement(
10.            (double)MinAreaField.Value,
11.            (double)MaxAreaField.Value)
12.         .AddCapacityRequirement(
13.            (int)MinCapacityField.Value,
14.            (int)MaxCapacityField.Value)
15.         .AddBathroomRequirement(
16.            (int)MinBathroomNumberField.Value,
17.            (int)MaxBathroomNumberField.Value)
18.         .AddBedRequirement(
19.            1,
20.            (int)OnePlaceBedNumberField.Value)
21.         .AddBedRequirement(
22.            2,
23.            (int)TwoPlaceBedNumberField.Value)
24.         .AddBedroomRequirement(
25.            (int)MinBedroomNumberField.Value,
26.            (int)MaxBedroomNumberField.Value)
27.         .BuildRequirementSet();
28.
29.     var roomProvider = new RequirementRoomProvider(requirementSet);
30.
31.     var fromDate = VacantCalendar.SelectionStart;
32.     var toDate = VacantCalendar.SelectionEnd;
33.
34.     List<Room> rooms;
35.
36.     if (IsOnlyVacantField.Checked
37.         && fromDate != DateTime.MinValue
38.         && toDate != DateTime.MinValue)
39.     {
40.         rooms = await roomProvider.GetVacantRoomsAsync(
41.             fromDate,
42.             toDate);
43.     }
```

```

44.     else
45.     {
46.         rooms = await roomProvider.GetRoomsAsync();
47.     }
48.
49.     RoomGrid.DataSource = rooms;
50. }
51.
52. private async Task ExecuteBedroomQuery()
53. {
54.     var dataSource = Enumerable.Empty<Bedroom>().ToList();
55.
56.     if (RoomGrid.CurrentRow != null)
57.     {
58.         var room = RoomGrid.CurrentRow.DataBoundItem as Room;
59.
60.         if (room != null)
61.         {
62.             dataSource = await HostelDbContext
63.                 .GetInstance()
64.                 .GetRoomBedroomsAsync(room.Id);
65.         }
66.     }
67.
68.     BedroomGrid.DataSource = dataSource;
69. }
70.
71. private async Task ExecuteBedQuery()
72. {
73.     var dataSource = Enumerable.Empty<Bed>().ToList();
74.
75.     if (BedroomGrid.CurrentRow != null)
76.     {
77.         var bedroom = BedroomGrid.CurrentRow.DataBoundItem as Bedroom;
78.
79.         if (bedroom != null)
80.         {
81.             dataSource = await HostelDbContext
82.                 .GetInstance()
83.                 .GetBedroomBedsAsync(bedroom.Id);
84.         }
85.     }
86.
87.     BedGrid.DataSource = dataSource;
88. }

```

В качестве примера обработки пользовательского ввода подойдет обработка нажатия пользователем кнопки «Добавить». Метод

AddButton_Click запускает диалоговое окно редактирования объекта, определив активную сетку с данными. После выхода пользователя из диалогового окна добавленный объект сохраняется в БД, сетка обновляется.

Листинг 14.

```
1. private void AddButton_Click(object sender, EventArgs e)
2. {
3.     var currentGrid = GetActiveGrid();
4.
5.     if (currentGrid == null)
6.     {
7.         MessageBox.Show(this, "Данные не выбраны", "Ошибка");
8.
9.         return;
10.    }
11.
12.    object? currentObject = null;
13.
14.    if (currentGrid == RoomGrid)
15.    {
16.        currentObject = new Room();
17.    }
18.    else if (currentGrid == BedroomGrid)
19.    {
20.        var room = RoomGrid.CurrentRow.DataBoundItem as Room;
21.
22.        currentObject = new Bedroom()
23.        {
24.            RoomId = room?.Id ?? 0
25.        };
26.    }
27.    else if (currentGrid == BedGrid)
28.    {
29.        var bedroom = BedroomGrid.CurrentRow.DataBoundItem as Bedroom;
30.
31.        currentObject = new Bed()
32.        {
33.            BedroomId = bedroom?.Id ?? 0
34.        };
35.    }
36.
37.    if (currentObject != null)
38.    {
39.        var edit = new EditEntityForm(currentObject);
40.
41.        var result = edit.ShowDialog(this);
42.    }
```

```

43.         if (result == DialogResult.OK)
44.         {
45.             using (var session =
HostelDbContext.GetInstance().BeginSession())
46.             {
47.                 if (currentGrid == RoomGrid)
48.                 {
49.                     HostelDbContext.GetInstance()
50.                         .AddRoomAsync((Room)currentObject).Wait();
51.                 }
52.                 else if (currentGrid == BedroomGrid)
53.                 {
54.                     HostelDbContext.GetInstance()
55.                         .AddBedroomAsync((Bedroom)currentObject).Wait();
56.                 }
57.                 else if (currentGrid == BedGrid)
58.                 {
59.                     HostelDbContext.GetInstance()
60.                         .AddBedAsync((Bed)currentObject).Wait();
61.                 }
62.             }
63.
64.             if (currentGrid == RoomGrid)
65.             {
66.                 ExecuteRoomQuery().Wait();
67.             }
68.             else if (currentGrid == BedroomGrid)
69.             {
70.                 ExecuteBedroomQuery().Wait();
71.             }
72.             else if (currentGrid == BedGrid)
73.             {
74.                 ExecuteBedQuery().Wait();
75.             }
76.         }
77.     }
78.     else
79.     {
80.         MessageBox.Show(this, "Данные не выбраны", "Ошибка");
81.     }
82. }

```

Обработка взаимодействия пользователя с остальными элементами интерфейса выполнена аналогичным образом.

3.2.8 Реализация класса «CustomersForm»

Класс CustomersForm описывает пользовательский интерфейс окна, содержащего данные гостей гостиницы и элементы управления для

редактирования данных гостей. Метод `CreateButton_Click` обрабатывает нажатие пользователя на кнопку «Создать», и, используя метод `ReadCustomerFromForm`, создает нового пользователя в базе данных.

Листинг 15

```
1. private void CreateButton_Click(object sender, EventArgs e)
2. {
3.     try
4.     {
5.         var customer = ReadCustomerFromForm();
6.
7.         HostelDbContext.GetInstance()
8.             .AddCustomerAsync(customer)
9.             .GetAwaiter()
10.            .GetResult();
11.
12.         if (customer.Id > 0)
13.         {
14.             currentCustomer = customer;
15.         }
16.
17.         ExecuteCustomersQuery().GetAwaiter().GetResult();
18.     }
19.     catch (PersistenceException ex)
20.     {
21.         MessageBox.Show(this, ex.Message, "Ошибка");
22.     }
23. }
24.
25. private Customer ReadCustomerFromForm()
26. {
27.     var customer = new Customer()
28.     {
29.         FullName = NameField.Text,
30.         BirthDate = BirthDatePicker.Value
31.     };
32.
33.     if (string.IsNullOrEmpty(customer.FullName))
34.     {
35.         throw new PersistenceException("Имя не введено");
36.     }
37.
38.     if (customer.BirthDate < DateTime.UtcNow.AddYears(-150)
39.         || customer.BirthDate > DateTime.UtcNow)
40.     {
41.         throw new PersistenceException("Неправильная дата рождения");
42.     }
```

```
43.  
44.     return customer;  
45. }
```

Обработка изменения данных пользователя выполнена аналогично.

Метод SelectButton_Click устанавливает выбранного гостя как текущего в сеансе работы с программой.

Листинг 16

```
1. private void SelectButton_Click(object sender, EventArgs e)  
2. {  
3.     var customer = GetCurrentCustomer();  
4.  
5.     if (customer == null)  
6.     {  
7.         MessageBox.Show(this, "Гость не выбран", "Ошибка");  
8.  
9.         return;  
10.    }  
11.  
12.    if (Owner is RoomForm roomForm)  
13.    {  
14.        roomForm.SetSelectedCustomer(customer);  
15.  
16.        Close();  
17.  
18.        return;  
19.    }  
20.    else  
21.    {  
22.        MessageBox.Show(this, "Невозможно выбрать пользователя", "Ошибка");  
23.    }  
24. }
```

3.2.9 Реализация класса «AccomodationForm»

Класс является интерфейсным и служит для просмотра и редактирования данных заселений гостей в номера. На форме располагаются: сетка с данными заселений, поля поиска номера и гостя, выпадающие списки выбора номера и гостя.

Форма имеет различные варианты запуска: при запуске могут быть указаны гость либо номер для фильтрации данных, а также пользователь и номер для выбора по умолчанию. Реализован такой подход с помощью

определения необязательных аргументов конструктора, а также с помощью внутренних защищенных полей.

Метод `SetCustomersComboBoxSource` выполняет создание элементов выпадающего списка для выбора гостя с использованием для поиска введенное в поле поиска значение.

Листинг 17

```
1. private void SetCustomersComboBoxSource()
2. {
3.     var searchText = SearchNameField.Text.ToLower();
4.
5.     var customers = HostelDbContext
6.         .GetInstance()
7.         .GetCustomersAsync()
8.         .GetAwaiter()
9.         .GetResult()
10.        .Where(c => c.FullName.ToLower().Contains(searchText)
11.            || c.Id.ToString().ToLower().Contains(searchText)
12.            || string.IsNullOrEmpty(searchText))
13.        .ToList();
14.
15.    CustomerComboBox.Items.Clear();
16.    CustomerComboBox.Items.AddRange(customers.ToArray());
17.
18.    if (string.IsNullOrEmpty(searchText)
19.        && _defaultCustomer != null
20.        && CustomerComboBox.Items.Contains(_defaultCustomer))
21.    {
22.        CustomerComboBox.SelectedItem = _defaultCustomer;
23.    }
24.}
```

В конструкторе формы содержится логика обработки фильтров. При переданных объектах фильтров возможность поиска и выбора значений на форме ограничивается. Учет фильтров при запросе данных в БД производится в методе `ExecuteAccommodationQuery`.

Листинг 18

```
1. public AccommodationForm(
2.     Customer? defaultCustomer = null,
3.     Customer? filterCustomer = null,
4.     Room? defaultRoom = null,
```

```

5.      Room? filterRoom = null)
6.  {
7.      InitializeComponent();
8.
9.      _defaultCustomer = defaultCustomer;
10.     _filterCustomer = filterCustomer;
11.     _defaultRoom = defaultRoom;
12.     _filterRoom = filterRoom;
13.
14.     if (_filterRoom != null)
15.     {
16.         SearchRoomField.Enabled = false;
17.         RoomComboBox.Enabled = false;
18.         _defaultRoom = _filterRoom;
19.     }
20.
21.     if (_filterCustomer != null)
22.     {
23.         SearchNameField.Enabled = false;
24.         CustomerComboBox.Enabled = false;
25.         _defaultCustomer = _filterCustomer;
26.     }
27.
28.     SetCustomersComboBoxSource();
29.     SetRoomsComboBoxSource();
30.     ExecuteAccommodationQuery();
31. }
32.
33. private void ExecuteAccommodationQuery()
34. {
35.     var accommodations = HostelDbContext
36.         .GetInstance()
37.         .GetAccommodationsAsync()
38.         .GetAwaiter()
39.         .GetResult()
40.         .Where(acc => (acc.CustomerId == _filterCustomer?.Id
41.             || _filterCustomer == null)
42.             && (acc.RoomId == _filterRoom?.Id
43.             || _filterRoom == null))
44.         .ToList();
45.
46.     AccommodationGrid.DataSource = accommodations;
47. }

```

3.2.10 Реализация класса «EditEntityForm»

Класс представляет из себя форму, на которой содержатся сгенерированные элементы редактирования свойств переданного на вход объекта.

В методе `InitEditControls` на форму добавляются элементы пользовательского интерфейса для редактирования каждого из свойств объекта, доступного для чтения и записи. При этом свойства – идентификаторы (имеющие атрибут `KeyAttribute`) не редактируются, т.к. они являются первичным ключом, который при создании объекта генерируется автоматически, а далее не может быть изменен.

Тип добавляемых на форму элементов зависит от типа свойства объекта. Создание элемента происходит в методе `GetControlByProperty`.

Считывание данных из элементов в свойства объекта происходит в методе `GetFromControls`.

Листинг 19

```
1. private void InitEditControls()
2. {
3.     foreach (var prop in Entity.GetType()
4.         .GetProperties()
5.         .Where(p => p.CanWrite
6.             && p.CanRead)
7.         .Where(p => p.GetCustomAttribute<KeyAttribute>() == null))
8.     {
9.         var label = new Label();
10.
11.         label.Text =
            prop.GetCustomAttribute<DisplayNameAttribute>()?.DisplayName ?? prop.Name;
12.
13.         var control = GetControlByProperty(prop, prop.GetValue(Entity));
14.
15.         MainContentLayoutPanel.Controls.Add(label);
16.         MainContentLayoutPanel.Controls.Add(control);
17.
18.         MainContentLayoutPanel.SetFlowBreak(control, true);
19.
20.         propertyNameControlMap[prop.Name] = control;
21.     }
22. }
23.
24. private Control GetControlByProperty(PropertyInfo property, object? value)
25. {
26.     var propertyType = property.PropertyType;
27.
28.     if (propertyType == typeof(string))
29.     {
```

```

30.         var textBox = new TextBox();
31.
32.         textBox.Text = value as String ?? string.Empty;
33.
34.         return textBox;
35.     }
36.     else if (propertyType == typeof(int)
37.         || propertyType == typeof(long)
38.         || propertyType == typeof(double))
39.     {
40.         var numeric = new NumericUpDown();
41.
42.         numeric.Maximum = decimal.MaxValue;
43.
44.         if (propertyType == typeof(double))
45.         {
46.             numeric.Increment = (decimal)0.01;
47.             numeric.DecimalPlaces = 2;
48.         }
49.         else
50.         {
51.             numeric.Increment = 1;
52.         }
53.
54.         numeric.Value = Convert.ToDecimal(value ?? 0);
55.
56.         return numeric;
57.     }
58.     else if (propertyType.IsEnum)
59.     {
60.         var comboBox = new ComboBox();
61.
62.         foreach (var en in Enum.GetValues(propertyType))
63.         {
64.             comboBox.Items.Add(en);
65.         }
66.
67.         comboBox.SelectedItem = value;
68.
69.         return comboBox;
70.     }
71.     else if (propertyType == typeof(DateTime))
72.     {
73.         var picker = new DateTimePicker();
74.
75.         if
(property.GetCustomAttribute<DisplayFormatAttribute>()?.DataFormatString ==
"short")
76.         {

```

```

77.         picker.Format = DateTimePickerFormat.Short;
78.     }
79.
80.     var dateTime = value != null ? (DateTime)value :
        DateTime.UtcNow.Date;
81.
82.     picker.Value = dateTime;
83.
84.     return picker;
85. }
86.
87.     throw new Exception("Невозможно создать Control");
88. }
89.
90. private void GetFromControls()
91. {
92.     foreach (var (name, control) in propertyNameControlMap)
93.     {
94.         var prop = Entity.GetType().GetProperty(name)
95.             ?? throw new NullReferenceException();
96.
97.         if (control is TextBox textBox)
98.         {
99.             prop.SetValue(Entity, textBox.Text);
100.        }
101.        else if (control is ComboBox comboBox)
102.        {
103.            prop.SetValue(Entity, comboBox.SelectedItem);
104.        }
105.        else if (control is NumericUpDown numeric)
106.        {
107.            object value;
108.
109.            if (prop.PropertyType == typeof(double))
110.            {
111.                value = Convert.ToDouble(numeric.Value);
112.            }
113.            else if (prop.PropertyType == typeof(int))
114.            {
115.                value = Convert.ToInt32(numeric.Value);
116.            }
117.            else
118.            {
119.                value = Convert.ToInt64(numeric.Value);
120.            }
121.
122.            prop.SetValue(Entity, value);
123.        }
124.        else if (control is DateTimePicker picker)

```

```
125.         {
126.             prop.SetValue(Entity, picker.Value);
127.         }
128.     else
129.     {
130.         throw new Exception("Не удалось получить значение");
131.     }
132. }
133. }
```

4. Тестирование

При запуске программы открывается главное окно:

Рисунок 14 Главное окно программы

На рисунке 14 показано окно программы с базой данных, заполненной тестовыми данными (кнопка «Тестовая БД»).

На рисунке 15 продемонстрировано изменение существующих данных с помощью диалогового окна «Редактировать» (кнопка «Изменить»). Изменено наименование гостиничного номера и его класс.

Номера

Класс
Все

Кол-во человек
От 0 До 0

Этаж
От 0 До 0

Площадь
От 0 До 0

Номера Спальни Спальные места

	Код номера	Наименование	Тип
▶	747	Тест Номер 747	Стандарт
	1270	Номер 1270	Люкс
	1362	Номер 1362	Апартаменты
	472	Номер 472	Апартаменты

Спальни

Редактировать

Код номера 747

Наименование Тест Номер 747

Тип Стандарт

Кол-во ванных 1

Этаж 7

Площадь 72,04

Рисунок 15 Изменение данных

На рисунках 16-18 показано тестирование создания и удаления данных. Тестирование произведено с объектами, хранящими данные о спальнях мест.

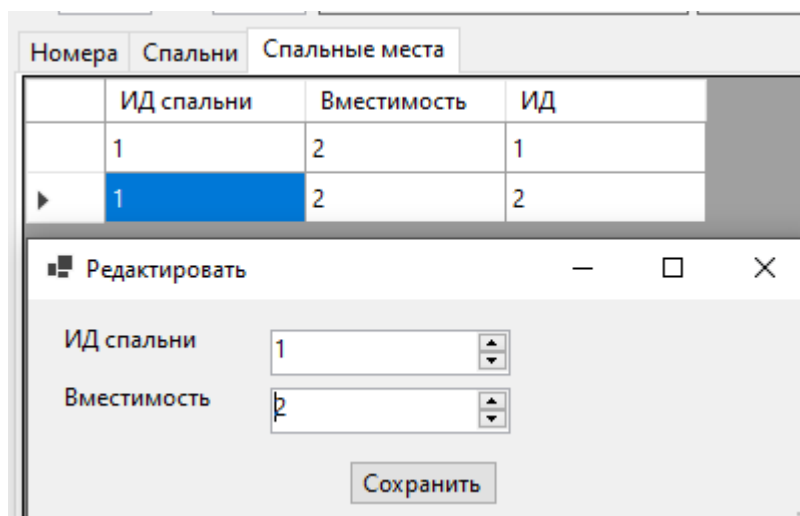


Рисунок 16 Изначальные данные и окно добавления

Номера	Спальни	Спальные места	
	ИД спальни	Вместимость	ИД
	1	2	1
	1	2	2
▶	1	2	384

Рисунок 17 Добавленные данные

Номера	Спальни	Спальные места	
	ИД спальни	Вместимость	ИД
	1	2	1
▶	1	2	384

Рисунок 18 Данные после удаления

На рисунке 19 показано тестирование выбора пользователя как текущего для сессии.

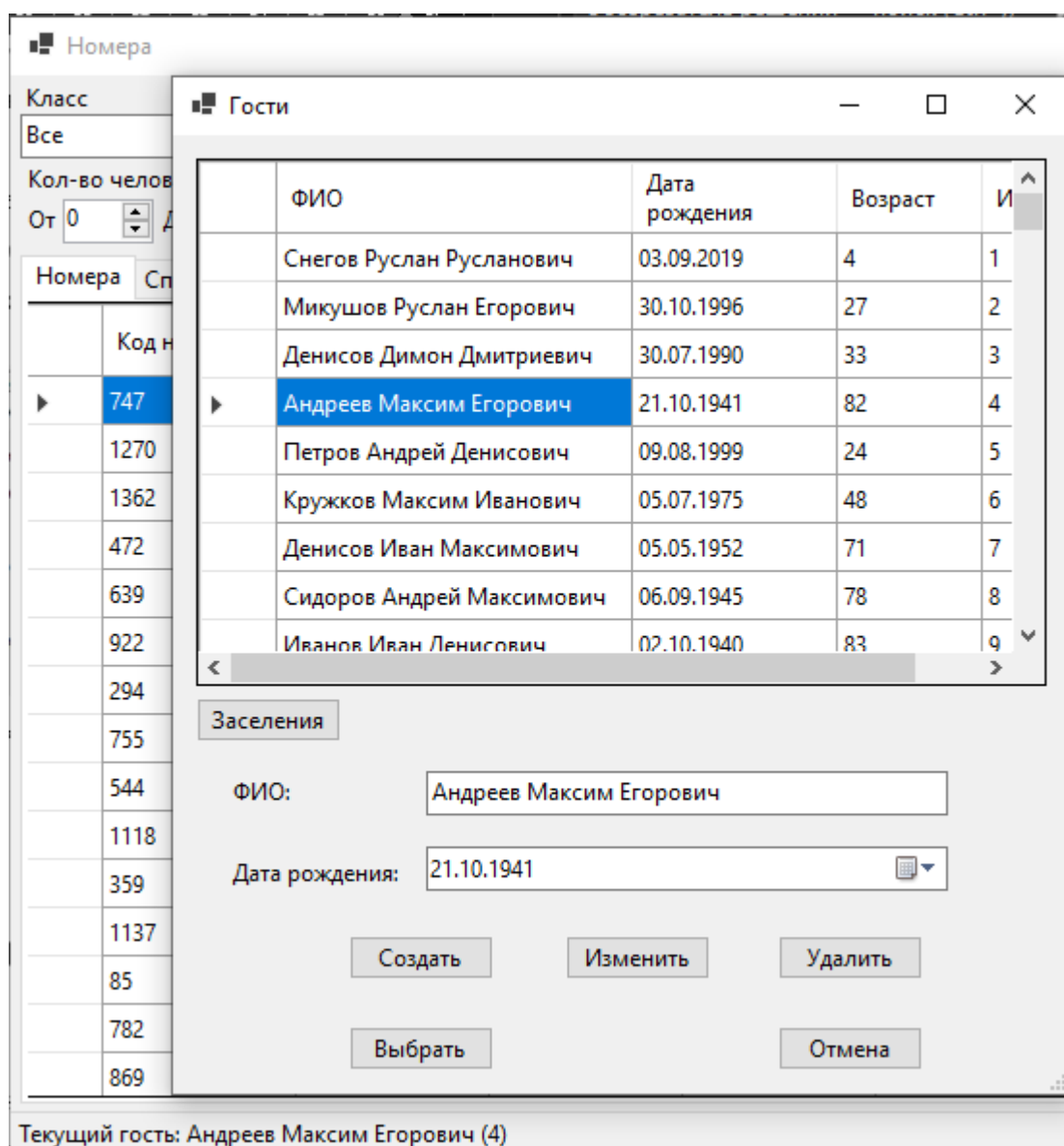


Рисунок 19 Выбор текущего пользователя

При выборе текущего пользователя он выбирается по умолчанию как гость для заселения при открытии окна «Заселения» с главного экрана программы. На рисунке 20 демонстрируется корректная работа.

Номера

Класс: Все

Кол-во человек: От 0 До 0

Этаж: От 0 До 0

Площадь: От 0 До 0

Номера | Спальни | Спальные места

Код номера	Наименование	Тип
747	Тест Номер 747	Ст
1270	Номер 1270	Л
1362	Номер 1362	А
472	Номер 472	А
639	Номер 639	А
922	Номер 922	Ст
294	Номер 294	А
755	Номер 755	Л
544	Номер 544	Л
1118	Номер 1118	Л
359	Номер 359	А
1137	Номер 1137	Ст
85	Номер 85	Ст
782	Номер 782	А
869	Номер 869	Л

Текущий гость: Андреев Максим Егорович (4)

Заселения

ИД номера	Наименование номера	ИД гостя	Имя гостя
1	Тест Номер 747	85	Андреев Егор
3	Номер 1362	22	Денисов Русл
5	Номер 639	77	Сидоров Арка
7	Номер 294	77	Сидоров Арка
9	Номер 544	84	Петров Андре
11	Номер 359	77	Сидоров Арка
13	Номер 85	33	Андреев Иван
15	Номер 869	40	Микушов Ива
17	Номер 67	30	Микушов Ма
19	Номер 344	87	Снегов Иван

Поиск номера:

Номер: (4) Апартаменты №472 "Номер 472" ▼

Поиск гостя:

Гость: (4): Андреев Максим Егорович ▼

Заселить

Рисунок 20 Открытие окна «Заселения» с пользователем по умолчанию

На рисунке 21 показан открытие формы «Заселения» с формы «Гости». Показаны заселения только выбранного гостя, выбор гостя в выпадающем списке недоступен.

Гости

	ФИО	Дата рождения	Возраст	ИД
	Снегов Руслан Русланович	03.09.2019	4	1
	Микушов Руслан Егорович	30.10.1996	27	2
▶	Денисов Димон Дмитриевич	30.07.1990	33	3
	Александр Михайлович...	21.10.1941	82	4

Заселения

	ИД номера	Наименование номера	ИД гостя	Имя гостя
▶	101	Номер 1359	3	Денисов Димон Д

Поиск номера:

Номер: ▼

Поиск гостя:

Гость: ▼

Заселить

Рисунок 21 Открытие формы «Заселения» с фильтром по гостю

На рисунке 22 показана работа текстового поиска номера. Элементы выпадающего списка сформированы на основе значения, введенного в поле поиска.

■ Заселения

	ИД номера	Наименование номера	ИД гостя	Имя гост
▶	101	Номер 1359	3	Денисов,

Поиск номера:

Номер:

Рисунок 22 Работа поиска номера

При попытке заселить в номер одновременно более одного гостя срабатывает проверка на пересечение дат (рис. 23).

	ИД номера	Наименование номера	ИД гостя	Имя гостя	Дата С	Дата ПО	ИД
▶	101	Номер 1359	3	Денисов Димон Дмитриевич	11.12.2023	29.12.2023	51

Поиск номера:

Номер:

Дата С:

Дата ПО:

Ошибка

Комната занята на даты 28.12.2023 0:00:00-29.12.2023 0:00:00

ОК Заселить

Рисунок 23 Проверка на пересечения периодов заселения

На рисунке 24 показан результат создания заселения с корректными данными.

■ Заселения

	ИД номера	Наименование номера	ИД гостя	Имя гостя	Дата С	Дата ПО	ИД
	101	Номер 1359	3	Денисов Димон Дмитриевич	11.12.2023	29.12.2023	51
▶	101	Номер 1359	3	Денисов Димон Дмитриевич	29.12.2023	30.12.2023	52

Поиск номера:

Номер: ▼

Поиск гостя:

Гость: ▼

Дата С: ▼

Дата ПО: ▼

Рисунок 24 Создание заселения

На главном экране расположены фильтры. Пример работы с фильтрами показан на рисунке 25. Выбраны номера с типом «Апартаменты», на 4 или 5 этаже, площадью от 50 до 80 кв.м.

■ Номера

Класс: ▼

Кол-во человек: От До

Этаж: От До

Площадь: От До

Спальни: От До

Ванные: От До

1-сп кровати:

2-сп кровати:

☐ Доступно к заселению

Декабрь 2023

Пн	Вт	Ср	Чт	Пт	Сб	Вс
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

☐ Сегодня: 03.12.2023

Номера	Спальни	Спальные места				
Код номера	Наименование	Тип	Кол-во ванных	Этаж	Площадь	ИД
▶ 441	Номер 441	Апартаменты	2	4	73,84	18
596	Номер 596	Апартаменты	1	5	64,48	35
537	Номер 537	Апартаменты	1	5	73,05	51
454	Номер 454	Апартаменты	2	4	79,93	92

Рисунок 25 Работа фильтров

На рисунке 26 показан результат применения фильтра по дате, выбранной в календаре.

Номера

Класс

Апартаменты

Кол-во человек

От 0 До 0

Этаж

От 4 До 5

Площадь

От 50 До 80

Спальни

От 0 До 0

Ванные

От 0 До 0

1-сп кровати

0

2-сп кровати

0

Доступно к заселению

Декабрь 2023

Пн	Вт	Ср	Чт	Пт	Сб	Вс
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

Сегодня: 03.12.2023

Гости

Номера

Спальни

Спальные места

	Код номера	Наименование	Тип	Кол-во ванных	Этаж	Площадь	ИД
▶	441	Номер 441	Апартаменты	2	4	73,84	18
	454	Номер 454	Апартаменты	2	4	79,93	92

Рисунок 26 Работа фильтров с календарем

Заключение

В процессе выполнения курсового проекта было спроектировано и реализовано программное решение в соответствии с выбранной предметной областью. Применены принципы ООП (создана иерархия классов), а также шаблоны проектирования (использованы шаблоны «Singleton», «Builder»).

Созданная программа обеспечивает автоматизированный учет заселений при работе гостиницы. Программа написана на языке C# с помощью фреймворка WinForms платформы .NET.

В процессе выполнения проекта приобретены и улучшены навыки создания приложений с пользовательским графическим интерфейсом.

Преимущества созданной программы:

- Работа с базой данных, возможность экспорта и импорта файлов из графического интерфейса;
- Унифицированный интерфейс редактирования данных;
- Малая связность интерфейсной и логической части программы;
- Используемые паттерны проектирования упростили структурную сложность программного кода;
- Переиспользование компонентов: одну и ту же форму пользовательского интерфейса можно запустить из разных мест программы, её поведение будет различаться в зависимости от контекста

Недостатки программы:

- Дизайн графического интерфейса является стандартным для фреймворка WinForms и не соответствует современным требованиям к внешнему виду ПО;
- Реализация БД не имеет проверки на ограничения (Foreign Keys, Primary Keys)
- Главный экран программы перегружен элементами.

Перспективы дальнейшего развития:

- Осуществить перевод программы на работу с реляционной БД;
- Провести работу над дизайном графического пользовательского интерфейса;
- Декомпонировать иерархию окон программы, разбив главный экран на модули.

Список использованных источников

1. Чарльз Петцольд Программирование с использованием Microsoft Windows Forms. Санкт-Петербург: Питер, 2006.
2. Что нового в Windows Forms в .NET 6.0 // habr.com URL: <https://habr.com/ru/companies/microsoft/articles/590057/> (дата обращения: 03-12-2023).
3. Рихтер Джеффри CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4 изд. СПб.: Питер, 2022.

Приложение 1 Исходный код программы

```
1. ./AccommodationForm.cs
using HostelApp.Entities;
using HostelApp.Persistence;
using System.Data;
using System.Diagnostics.Eventing.Reader;

namespace HostelApp
{
    public partial class AccommodationForm : Form
    {
        private readonly Customer? _defaultCustomer;
        private readonly Customer? _filterCustomer;
        private readonly Room? _defaultRoom;
        private readonly Room? _filterRoom;

        public AccommodationForm(
            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();
            ExecuteAccommodationQuery();
        }

        private void ExecuteAccommodationQuery()
        {
            var accommodations = HostelDbContext
                .GetInstance()
                .GetAccommodationsAsync()
                .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.IsNullOrEmpty(searchText))
            .ToList();

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

        if (string.IsNullOrEmpty(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.IsNullOrEmpty(searchText))
            .ToList();

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

        if (string.IsNullOrEmpty(searchText)
            && _defaultRoom != null
            && RoomComboBox.Items.Contains(_defaultRoom))
        {

```

```

        RoomComboBox.SelectedItem = _defaultRoom;
    }
}

private void RoomSearchField_TextChanged(object sender, EventArgs
e)
{
    SetRoomsComboBoxSource();
}

private void CreateAccommodationButton_Click(object sender,
EventArgs e)
{
    try
    {
        CreateAccommodation();
    }
    catch (Exception ex)
    {
        MessageBox.Show(this, ex.Message, "Ошибка");
    }
}

private void CreateAccommodation()
{
    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()
        .CreateRoomAccommodationAsync(
            room.Id,
            fromDate,
            toDate,
            customer.Id)
        .GetAwaiter()
        .GetResult();

    ExecuteAccommodationQuery();

    foreach (DataGridViewRow item in AccommodationGrid.Rows)
    {
        if (item.DataBoundItem is Accommodation gridAcc
            && gridAcc.Id == acc.Id)
        {
            AccommodationGrid.ClearSelection();
            AccommodationGrid.CurrentCell = item.Cells[0];

            item.Cells[0].Selected = true;

```

```

        break;
    }
}

private void DeleteButton_Click(object sender, EventArgs e)
{
    try
    {
        DeleteAccommodation();
    }
    catch (Exception ex)
    {
        MessageBox.Show(this, ex.Message, "Ошибка");
    }
}

private void DeleteAccommodation()
{
    var result = MessageBox.Show(
        this,
        "Удалить заселение?",
        "Внимание",
        MessageBoxButtons.OKCancel);

    if (result != DialogResult.OK)
    {
        return;
    }

    var acc = AccommodationGrid.CurrentRow.DataBoundItem as
Accommodation
        ?? throw new NullReferenceException("Заселение не
выбрано");

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

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

    ExecuteAccommodationQuery();
}
}

2. ./AccommodationForm.Designer.cs
namespace HostelApp
{
    partial class AccommodationForm
    {
        /// <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();
        CreateAccommodationButton = 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();
        AccommodationGrid = new DataGridView();
        panel1.SuspendLayout();

        ((System.ComponentModel.ISupportInitialize)AccommodationGrid).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(CreateAccommodationButton);
        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(AccommodationGrid);
        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;
//
// CreateAccommodationButton
//
CreateAccommodationButton.Anchor = AnchorStyles.Bottom |
AnchorStyles.Left;
CreateAccommodationButton.Location = new Point(350, 501);

```



```

CreateAccommodationButton.Name = "CreateAccommodationButton";
CreateAccommodationButton.Size = new Size(75, 23);
CreateAccommodationButton.TabIndex = 9;
CreateAccommodationButton.Text = "Заселить";
CreateAccommodationButton.UseVisualStyleBackColor = true;
CreateAccommodationButton.Click +=
CreateAccommodationButton_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(panell1);
        Name = "AccomodationForm";
        Text = "Заселения";
        panell1.ResumeLayout(false);
        panell1.PerformLayout();

        ((System.ComponentModel.ISupportInitialize)AccomodationGrid).EndInit();
        ResumeLayout(false);
    }

    #endregion

    private Panel panell1;

```

```

        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 CreateAccommodationButton;
        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.GetCustomAttributes<KeyAttribute>() == null))
            {
                var label = new Label();

                label.Text =
prop.GetCustomAttributes<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/Accommodation.cs
using HostelApp.Persistence;
using System.ComponentModel;

namespace HostelApp.Entities
{
    public class Accommodation : 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.IsNullOrEmpty(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 = GenerateRandomAccommodation(room.Id);

            await context.CreateRoomAccommodationAsync(
                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 = $"Homep {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 GenerateRandomAccommodation(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 accommodationLength = 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(accommodationLength)
        };

        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.Бсе => "",
        _ => 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 GetAccommodationsAsync())
                .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<Accommodation>> GetAccommodationsAsync() =>
await GetEntities<Accommodation>();

        public async Task<Accommodation?> GetAccommodationAsync(int id) =>
await GetEntity<Accommodation>(id);

        public async Task AddAccommodationAsync(Accommodation accomodation)
=> await AddEntity(accomodation);

```

```

        public async Task UpdateAccommodationAsync(Accommodation
accommodation) => await UpdateEntity(accommodation);

        public async Task DeleteAccommodationAsync(int id) => await
DeleteEntity<Accommodation>(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 GetAccommodationsAsync())
                .Where(acc =>
                {
                    return acc.FromDate < toDate && acc.ToDate > fromDate;
                })
                .Select(acc => acc.RoomId)
                .ToHashSet();

            var result = rooms.Where(r =>
            {
                return !accommodations.Contains(r.Id);
            });

            return result.ToList();
        }
    }
}

```

```

    }

    /// <summary>
    /// Создать заселение
    /// </summary>
    public async Task<Accommodation> CreateRoomAccommodationAsync(
        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 Accommodation()
        {
            CustomerId = customerId,
            FromDate = clearFromDate,
            ToDate = clearToDate,
            RoomId = roomId
        };

        await AddAccommodationAsync(acc);

        return acc;
    }

    /// <summary>
    /// Получить заселение в комнату на дату
    /// </summary>
    public async Task<Accommodation?> GetRoomAccommodationOnDate(
        int roomId,
        DateTime onDate)
    {
        var clearOnDate = onDate.Date;

        var accommodation = (await GetAccommodationsAsync()).Where(acc
=>
        {
            return acc.RoomId == roomId
                && acc.FromDate <= onDate
                && acc.ToDate > onDate;
        }).FirstOrDefault();

        return accommodation;
    }
}

```

```

}

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 AddCapacityRequirement(
    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.Bce)
                || 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 =
                    $"고객 { _selectedCustomer.FullName }
({ _selectedCustomer.Id })";
            }
            else
            {
                CurrentCustomerLabel.Text =
                    $"고객 선택";
            }
        }

        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)
                .AddCapacityRequirement(
                    (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.IsNullOrEmpty(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)
        {

```



```

        {
            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, "G?堽堽u□?, "θ祉;
    }
}

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, "G?堽堽u□?, "θ祉;
    }
}

```

```

        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, "G\u00f9\u00f9\u00f9", "0\u00f9\u00f9\u00f9");
}
}

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, "i □□ □□ ❖❖", "❖❖");
    }
}

private void SaveDbButton_Click(object sender, EventArgs e)
{
    try
    {
        var context = HostelDbContext.GetInstance();

        context.CopyDatabaseFile().Wait();
    }
    catch (Exception ex)
    {
        MessageBox.Show(this, ex.Message, "❖❖");
    }
}

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, "❖❖");
    }
}

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, "❖❖");
    }
}

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("房间不存在");

            context.DeleteRoomAsync(entity.Id).Wait();
        }
        else if (TabControl.SelectedTab == BedroomTab)
        {
            var entity = BedroomGrid.CurrentRow.DataBoundItem as
                Bedroom
                ?? throw new NullReferenceException("卧室不存在");

            context.DeleteBedroomAsync(entity.Id).Wait();
        }
        else if (TabControl.SelectedTab == BedTab)
        {
            var entity = BedGrid.CurrentRow.DataBoundItem as Bed
                ?? throw new NullReferenceException("床位不存在");

            context.DeleteBedAsync(entity.Id).Wait();
        }
        else
        {
            throw new ApplicationException("操作失败");
        }

        ExecuteRoomQuery().Wait();
    }
    catch (Exception ex)
    {
        MessageBox.Show(this, ex.Message, "错误");
    }
}
}

```

```

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();
AccommodationButton = new Button();
RoomAccommodationButton = 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).BeginIn
it();

((System.ComponentModel.ISupportInitialize)TwoPlaceBedNumberField).BeginIn
it();

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

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

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

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

((System.ComponentModel.ISupportInitialize)MaxBathroomNumberField).BeginIn
it();

((System.ComponentModel.ISupportInitialize)MinBathroomNumberField).BeginIn
it();

((System.ComponentModel.ISupportInitialize)MaxBedroomNumberField).BeginIni
t();

((System.ComponentModel.ISupportInitialize)MinBedroomNumberField).BeginIni
t();

```

```

((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 = "Homepa";
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";

```



```

RoomAccommodationButton.UseVisualStyleBackColor = true;
RoomAccommodationButton.Click += RoomAccommodationButton_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>RoomAccommodationButton);
        Controls.Add>AccommodationButton);
        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 = "Homepa";
        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;
}
}

```