



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»


КАФЕДРА ИУК4 «Программное обеспечение ЭВМ, информационные технологии»

О Т Ч Е Т

УЧЕБНАЯ ПРАКТИКА


«Ознакомительная практика»

Студент гр. ИУК4-23Б


(подпись)

(Мазина Ю. А.)
(Ф.И.О.)

Руководитель


(подпись)

(Амеличева К.А.)
(Ф.И.О.)

Оценка руководителя

30-50

баллов

(дата)

Оценка защиты

30-50

баллов

(дата)

Оценка практики

баллов


(оценка по пятибалльной шкале)

Комиссия:

(подпись)

(Гагарин Ю.Е.)

(Ф.И.О.)

(подпись)

(Амеличева К.А.)

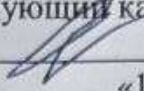
(Ф.И.О.)

(подпись)

(Пчелинцева Н.И.)

(Ф.И.О.)

Калуга, 2024

УТВЕРЖДАЮ
Заведующий кафедрой ИУК4
 (Гагарин Ю.Е.)
«14» февраля 2023 г.

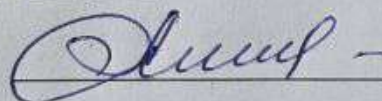
З А Д А Н И Е на УЧЕБНУЮ, ОЗНАКОМИТЕЛЬНУЮ ПРАКТИКУ

За время прохождения практики студенту необходимо:

1. Осуществить и обосновать выбор предметной области, сформулировать тему проекта и определить область его применения.
2. Осуществить выбор технологий разработки и вспомогательного программного обеспечения.
3. Определить сроки, в рамках которых будет осуществляться контроль за промежуточным состоянием проекта.
4. Разработать и реализовать проект «Отельный бизнес» по оформлению и заказу номера определенного типа на какой-то промежуток времени.
5. Подготовить отчет и защитить результаты практики.

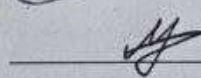
Дата выдачи задания «07» 02 2024 г.

Руководитель практики



Амеличева К.А.

Задание получил



Мазина Ю.А.

Оглавление

| | |
|---|----|
| Оглавление | 3 |
| ВВЕДЕНИЕ | 4 |
| 1. МЕТОДЫ И ИНСТРУМЕНТЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ | 5 |
| 1.1. Основание разработки | 5 |
| 1.2. Назначение разработки | 5 |
| 1.3. Технические требования | 6 |
| 1.3.1. Требования к функциональным характеристикам | 6 |
| 1.3.1.1. Состав выполняемых функций..... | 6 |
| 1.3.2. Организация входных и выходных данных..... | 7 |
| 1.3.3. Условия эксплуатации и требования к составу и параметрам | 7 |
| технических средств..... | 7 |
| 1.3.4. Требования к транспортировке и хранению | 7 |
| 1.3.5. Специальные требования | 7 |
| 1.4. Требования к программной документации..... | 8 |
| 1.5 Техническо-экономические показатели | 8 |
| 1.5.1 Требования к информационной и программной совместимости..... | 8 |
| 1.6 Порядок контроля и приемки..... | 9 |
| 1.7 Календарный план | 9 |
| 2. ПРОЕКТИРОВАНИЕ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА | 11 |
| 2.1 Принцип работы приложения | 11 |
| 2.2. Разработка структурной схемы интерфейса..... | 11 |
| 2.3. Описание логической структуры | 15 |
| 2.3.1. Загрузка программы | 15 |
| 2.3.2. Добавление новой записи..... | 16 |
| 2.3.3. Удаление записи | 16 |
| 2.3.4. Редактирование..... | 17 |
| 2.3.5. Выход из программы | 17 |
| 2.3.6. Просмотр диаграммы | 18 |
| 2.3.7. Поиск по записям..... | 18 |
| 2.4. Разработка плана тестирования | 18 |
| 3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ | 20 |
| ЗАКЛЮЧЕНИЕ | 22 |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ | 23 |
| ПРИЛОЖЕНИЕ 1 | 26 |
| ПРИЛОЖЕНИЕ 2 | 57 |
| ПРИЛОЖЕНИЕ 3 | 65 |

ВВЕДЕНИЕ

Целью учебной практики является сформировать первичные умения и навыки:

- выбора предметной области на основе имеющихся знаний в сфере разработки программного обеспечения, умение грамотно сформулировать тему проекта, область его применения и главные отличия относительно уже имеющихся проектов;
- по подбору и оценке технологий разработки и вспомогательного программного обеспечения для реализации итогового проекта в соответствии с выбранной темой;
- оценки функциональности проекта, которая будет реализована к соответствующему сроку, умение грамотно распределить роли и задачи внутри команды;
- по представлению и оценке качества разработанного приложения.

Для достижения поставленной цели решаются следующие задачи:

- разделить студентов на группы, в которых они будут реализовывать итоговые проекты согласно предметным областям, согласовать выбранные темы, назначить сроки промежуточных встреч и консультаций;
- осуществить выбор языка(-ов) программирования, фреймворков и библиотек, среду(-ы) разработки, систему контроля версий, сервис хостинга проектов, обосновать соответствующий выбор;
- подготовить доклад с использованием инструментов визуализации (презентации, графики, блок-схемы, UML-диаграммы и пр.), в котором будет представлен отчет команды о проделанной работе, общая информация о разработанном проекте, его преимущества, недостатки и перспективы дальнейшего развития.

1. МЕТОДЫ И ИНСТРУМЕНТЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ

1.1. Основание разработки

Основанием для данной работы является выполнение учебного плана.

Наименование работы: «Отельный бизнес».

Исполнители: Акимов С.В., Мазина Ю.А.

1.2. Назначение разработки

Файл проекта содержит информацию о забронированных номерах.

Приложение должно содержать и выполнять следующие действия:

- Добавление данных о новом забронированном номере
- Вывод списка забронированных номеров
- Редактирование полей (дата заезда и выезда, адрес электронной почты, номер телефона)
- Сортировка заказов по полям (дата заезда и выезда)
- Поиск по полю ФИО
- Поиск по полю Дата заезда
- Поиск по полю Тип номера
- Поиск по полю Комментарий
- Удаление заказов
- Графическое отображение информации по типу номера в виде столбчатой гистограммы (по горизонтали - тип номера, по вертикали - количество забронированных номеров)

1.3. Технические требования

1.3.1. Требования к функциональным характеристикам

1.3.1.1. Состав выполняемых функций

Разрабатываемое ПО должно обеспечивать:

- сбор и анализ информации о забронированных номерах;
- визуализацию информации о госте:
 - ФИО гостя;
 - Адрес электронной почты;
 - Номер телефона;
- визуализацию информации о записи:
 - Тип номера;
 - Дата заезда
 - Дата выезда;
 - Количество мест;
 - Комментарии к заказу;
 - Способ оплаты;
 - Стоимость;

Для устройств управления просмотром информации о забронированных номерах программа должна содержать такие поля меню, как: удалить заказ, добавить заказ, просмотреть список заказов, редактировать заказ, сортировать заказы, вывести столбчатую гистограмму, поиск по заказам.

1.3.2. Организация входных и выходных данных

Исходные файлы поступают в систему из файла, в котором хранятся данные. Данные отображаются на компьютере администратора отеля в виде таблицы, с возможностью дальнейшей работы с ними. Основным режимом использования системы - ежедневная работа. Требования к надежности. Для обеспечения надежности необходимо проверять корректность получаемых данных.

1.3.3. Условия эксплуатации и требования к составу и параметрам технических средств

Для работы системы должен быть выделен ответственный оператор. Требования к составу и параметрам технических средств уточняются на этапе эскизного проектирования системы.

1.3.4. Требования к транспортировке и хранению

Программа поставляется на USB-флэш-накопителе.

Программная документация поставляется в электронном и печатном виде.

1.3.5. Специальные требования

Программное обеспечение должно иметь дружелюбный интерфейс, рассчитанный на пользователя (в плане компьютерной грамотности) средней квалификации.

Ввиду объемности проекта задачи предполагается решать поэтапно, при этом модули ПО, созданные в разное время, должны предполагать возможность

наращивания системы и быть совместимы друг с другом, поэтому документация на принятое эксплуатационное ПО должна содержать полную информацию, необходимую для работы программистов с ним.

Язык программирования – по выбору исполнителя, должен обеспечивать возможность интеграции программного обеспечения с некоторыми видами периферийного оборудования.

1.4. Требования к программной документации

Состав технической документации должен включать в себя:

1. Техническое задание (ГОСТ 34.602-89);
2. Руководство программиста (ГОСТ 19.504-79);
3. Руководство пользования (ГОСТ 19.505-79);
4. Демонстрационный чертеж (UML).

1.5 Техническо-экономические показатели

Эффективность системы определяется удобством использования системы для контроля и управления успеваемостью студентов.

1.5.1 Требования к информационной и программной совместимости

Программа должна работать на платформах ОС Windows.

Для нормального функционирования данного программного продукта необходимо наличие:

- 1) IBM совместимый процессор 1ГГц и выше;
- 2) Операционная система Windows 7/8/10/11;
- 3) Не менее 100 Мб свободного места в памяти;

- 4) Не менее 1,5 ГБ оперативной памяти (ОЗУ);

Для нормального функционирования клиента необходимы:

- 1) Доступное бесперебойное интернет или локальное соединение;
- 2) .NET Framework Version 4.5 и выше;
- 3) Операционная система Windows 7/8/10;
- 4) Не менее 100 Мб свободного места в памяти;

1.6 Порядок контроля и приемки

После передачи Исполнителем отдельного функционального модуля программы Заказчику, последний имеет право тестировать модуль в течение 2 дней. После тестирования Заказчик должен принять работу по данному этапу или в письменном виде изложить причину отказа от принятия. В случае обоснованного отказа Исполнитель обязуется доработать модуль.

1.7 Календарный план

| № этапов | Название этапа | Сроки этапа | Чем оканчивается этап |
|----------|--|-------------------------|-------------------------------|
| 1 | Изучение предметной области. Проектирование системы. Разработка предложений по реализации системы. | 26.03.2024 – 08.03.2024 | Предложения по работе системы |
| 2 | Разработка программного | 08.03.2024 – | Программный |

| | | | |
|---|--|----------------------------|---|
| | модуля по проекту «Отельный бизнес» | 20.04.2024 | комплекс |
| 3 | Тестирование и отладка модуля | 20.04.2024 — 08.05.2024 | Готовая система «Отельный бизнес». Программная документация |

2. ПРОЕКТИРОВАНИЕ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА

2.1 Принцип работы приложения

Программа представляет собой сервис, позволяющий оформлять бронь номера в отеле определенного типа на какой-то промежуток времени. Он позволяет добавлять новые записи, редактировать информацию о выбранной записи, удалять и сохранять ее. Программа способна сортировать информацию о записях по двум из полей как по убыванию, так и по возрастанию.

В программе реализованы следующие функции:

- Добавление информации о новом забронированном номере
- Редактирование полей
- Поиск по четырем полям
- Сортировка записей по полям
- Удаление записи
- Графическое изображение информации о популярных номерах

2.2. Разработка структурной схемы интерфейса

Заказчик сформулировал следующие потребности: создать приложение «Отельный бизнес» с графическим интерфейсом, осуществить возможность добавления новых записей, просмотр статистики типов номеров, просмотр общей таблицы с заказами с возможностью ее сортировки. На основании вышесказанного была построена диаграмма действий программы (см. Приложение 3 рис.1).

Разработанная программа состоит из следующих модульных частей:

ReservationList (с. 26) – данный модуль содержит класс, который необходим для работы с записями о бронировании. Содержит такие поля как:

m_typeOfRoom – поле для описания типа номера;

m_numberOfSeats – поле количества гостей;

m_comments – поле комментариев;

m_names[3] – массив для описания полей ФИО гостя;

m_email – поле адрес электронной почты;

m_phoneNumber – поле номера телефона;

m_paymentMethod – поле для описания способа оплаты;

m_dateOfArrival – поле даты заезда;

m_dateOfDeparture – поле даты выезда;

Содержит такие методы как:

operator >> – служит для перегрузки оператора ввода из файла

operator << – служит для перегрузки оператора вывода в файл

m_price – поле для описания итоговой стоимости;

MyVector (с. 27) – данный модуль содержит собственный класс контейнер, который необходим для хранения записей. Содержит такие методы как:

AddMemory – служит для увеличения вместимости контейнера;

PushBack – служит для добавления новой записи в конец контейнера;

Remove – служит для удаления произвольной записи из контейнера;

Size – служит для возвращения значения вместимости контейнера;

Iterator (с. 29) – данный модуль содержит собственный класс-итератор, который необходим для перебора элементов контейнера и обеспечения во время перемещения по элементам доступ к каждому из них. Содержит такие методы как:

operator == – служит для перегрузки оператора сравнения;

operator != – служит для перегрузки оператора неравенства;

operator ++ – служит для перегрузки оператора инкремента;

operator* – служит для перегрузки оператора разыменования указателя;

Mainwindow (с. 30) – данный модуль является главным, так как включает в себя остальные модули и осуществляет логику приложения. Содержит такие методы как:

addBookingWindow – служит для создания окна для добавления новой записи;

addBooking – служит для считывания введенной информации из окна добавления записи и создания объекта класса ReservationList, который после добавляется в конец контейнера;

deleteBookingWidget – служит для создания окна для удаления записи;

deleteBooking – служит для удаления конкретной записи с помощью вызова метода Remove;

deleteAll – служит для удаления всех записей из таблицы;

addDataFromFile – служит для добавления записей из файла с помощью вызова перегрузки оператора ввода из файла;

addDataToFile – служит для добавления записей в файл с помощью вызова перегрузки оператора вывода в файл;

setupCoreWidgets – служит для создания и отображения основного окна и кнопок взаимодействия;

tableView – служит для создания таблицы из записей;

testLayout – служит для создания окна, которое вызывается при использовании функции поиска, с таблицей, в которой отображаются найденные записи;

arrivalIncSort – служит для сортировки записей по полю Дата заезда по возрастанию;

arrivalDecSort – служит для сортировки записей по полю Дата заезда по убыванию;

departureIncSort – служит для сортировки записей по полю Дата выезда по возрастанию;

departureDecSort – служит для сортировки записей по полю Дата выезда по убыванию;

fioSearchLayout – служит для создания окна для осуществления функции Поиск по фамилии;

fioFind – служит для поиска введенной фамилии среди записей, находящихся в таблице;

dateSearchLayout – служит для создания окна для осуществления функции Поиск по дате заезда;

dateFind – служит для поиска введенной даты заезда среди записей, находящихся в таблице;

typeSearchLayout – служит для создания окна для осуществления функции Поиск по типу номера;

typeSearchFunction – служит для поиска выбранного типа номера среди записей, находящихся в таблице;

commentSearchLayout – служит для создания окна для осуществления функции Поиск по комментарию;

commentFind – служит для поиска введенного комментария среди записей, находящихся в таблице;

diogram – служит для создания окна с диаграммой и ее построение;

arrivalDateChange – служит для создания окна для осуществления функции Редактирование по полю Дата заезда;

departureDateChange – служит для создания окна для осуществления функции Редактирование по полю Дата выезда;

phoneChange – служит для создания окна для осуществления функции Редактирование по полю Номер телефона;

emailChange – служит для создания окна для осуществления функции Редактирование по полю Адрес электронной почты;

arrivalDateChangePush – служит для замены изменяемого поля Дата заезда на новое значение;

departureDateChangePush – служит для замены изменяемого поля Дата выезда на новое значение;

phoneChangePush – служит для замены изменяемого поля Номер телефона на новое значение;

emailChangePush – служит для замены изменяемого поля Адрес электронной почты на новое значение;

createIcons – служит для интегрирования иконок в окно программы;

createMenuBar – служит для осуществления работы с кнопками взаимодействия;

2.3. Описание логической структуры

2.3.1. Загрузка программы

После запуска приложения перед пользователем открывается пустая форма с шапкой таблицы.

При нажатии на кнопку «Файлы» открывается окно, в котором можно выбрать действие «Добавить из файла» или «Сохранить в файл».

При нажатии на кнопку «Сортировка» открывается окно, в котором можно выбрать направление сортировки «По дате заезда по возрастанию», «По дате заезда по убыванию», «По дате выезда по возрастанию» или «По дате выезда по убыванию», после чего управление передается соответственно процедуре «arrivalIncSort», «arrivalDecSort», «departureIncSort» или «departureDecSort».

При нажатии на кнопку «Поиск» открывается окно, в котором можно выбрать направление поиска «По фамилии», «По дате заезда», «По комментарию» или «По типу номера».

При нажатии на кнопку «Редактирование» открывается окно, в котором можно выбрать поле, которое необходимо изменить, «Дата заезда», «Дата выезда», «Email» или «Номер телефона».

При нажатии на кнопку «Помощь» открывается окно, в котором можно выбрать действие «Руководство пользователя».

При нажатии на кнопку «Выйти» пользователь закрывает приложение.

При нажатии на кнопку «Добавить бронь» открывается окно для создания новой записи.

При нажатии на кнопку «Диаграмма» открывается окно для отображения диаграммы.

При нажатии на кнопку «Удалить бронь» открывается окно для удаления записи.

При нажатии на кнопку «Удалить все записи» удаляются все записи из таблицы.

2.3.2. Добавление новой записи

Добавить новые записи в таблицу можно двумя способами: добавить записи из файла и добавить запись с клавиатуры.

Для добавления записей из файла необходимо нажать кнопку «Файлы» → «Добавить из файла». Далее все записи из файла отобразятся в таблице (см. Приложение 2 рис. 1).

Для добавления записи с клавиатуры необходимо нажать кнопку «Добавить бронь», после управление передается процедуре «addBookingWindow». Далее заполнить все поля: «ФИО», «Дата заезда», «Дата выезда», «Тип номера», «Количество гостей», «Email», «Номер телефона», «Комментарии», «Способ оплаты», затем управление передается процедуре «addBooking» (см. Приложение 2 рис. 2). Введенная запись отобразится в таблице.

2.3.3. Удаление записи

Удалить записи можно двумя способами: удалить конкретную запись и удалить все записи.

Для удаления конкретной записи необходимо нажать кнопку «Удалить бронь», после чего управление передается процедуре «deleteBookingWidget». Далее в появившемся окне ввести номер записи, которую нужно удалить, затем управление передается процедуре «deleteBooking» (см. Приложение 2 рис. 3). Выбранная бронь будет удалена из таблицы.

Для удаления всех записей необходимо нажать кнопку «Удалить все записи». Далее в появившемся окне выбрать «Да, удалить» или «Отмена», чтобы оставить записи, затем управление передается процедуре «deleteAll» (см. Приложение 2 рис. 4).

2.3.4. Редактирование

Для редактирования записи необходимо нажать кнопку «Редактирование» и выбрать поле, которое нужно изменить: «Дата заезда», «Дата выезда», «Email» или «Номер телефона», после чего управление передается соответственно процедуре «arrivalDateChange», «departureDateChange», «emailChange» или «phoneChange». Далее в появившемся окне ввести номер записи и измененное поле и нажать кнопку «Редактировать» (см. Приложение 2 рис. 5), затем управление передается соответственно процедуре «arrivalDateChangePush», «departureDateChangePush», «phoneChangePush» или «emailChangePush». Измененная запись отобразится в таблице.

2.3.5. Выход из программы

Для выхода из программы необходимо нажать кнопку «Выйти». Далее в появившемся окне выбрать действие «Выйти» или «Сохранить и выйти», если выход из программы больше не требуется, то просто закрыть окно или нажать кнопку «Отмена» (см. Приложение 2 рис. 6).

2.3.6. Просмотр диаграммы

Для просмотра диаграммы необходимо нажать кнопку «Диаграмма», после чего управление передается процедуре «diogram». Далее в появившемся окне графически отобразится столбчатая гистограмма (по горизонтали - тип номера, по вертикали - количество забронированных номеров) (см. Приложение 2 рис. 7).

2.3.7. Поиск по записям

Для поиска по записям необходимо нажать кнопку «Поиск» и выбрать поле, по которому он будет осуществляться: «Фамилия», «Дата заезда», «Тип номера» или «Комментарии», после чего управление передается соответственно процедуре «fioSearchLayout», «dateSearchLayout», «typeSearchLayout» или «commentSearchLayout». Далее в появившемся окне ввести запись, которую нужно найти (см. Приложение 2 рис. 8), затем управление передается соответственно процедуре «fioFind», «dateFind», «typeSearchFunction» или «commentFind». В появившемся окне будут выведена таблица со всеми найденными записями.

2.4. Разработка плана тестирования

- Запустите программу
- Добавьте новую запись, нажав «Добавить бронь» или «Файлы» → «Добавить из файла»
- При выборе «Добавить бронь» в появившихся полях укажите ФИО, дату заезда, дату выезда, тип номера, количество гостей, адрес электронной почты, номер телефона, комментарии, способ оплаты:

| ФИО | Дата заезда | Дата выезда | Тип номера | Кол-во гостей | Email | Номер телефона | Комментарии | Способ оплаты |
|----------------------------|----------------|----------------|------------|---------------|------------------|-----------------|-------------|---------------|
| Иванов Иван Иванович | 29.06.20 24 | 30.06.20 24 | Стандарт | 2 | ivanov@gmail.com | 880053533 53 | нет | наличные |

- Нажмите кнопку «Добавить»
- Повторите предыдущие пункты с другими данными еще два раза
- Нажмите кнопку «Сортировка» → «По дате заезда по возрастанию». Записи в таблице будут отсортированы по дате заезда по возрастанию
- Нажмите кнопку «Поиск» → «По типу номера». Выберите тип номера «Стандарт». В появившемся окне в таблице будут отображены записи с таким типом номера.
- Нажмите кнопку «Редактирование» → «Дата выезда». Выберите запись 1 и введите новую дату выезда. Измененная запись будет отображена в таблице.
- Нажмите кнопку «Диаграмма». В появившемся окне графически отобразится столбчатая гистограмма (по горизонтали - тип номера, по вертикали - количество бронированных номеров).
- Нажмите кнопку «Удалить бронь». Введите запись 1. Выбранная бронь будет удалена из таблицы.
- Нажмите кнопку «Выйти» → «Сохранить и выйти». Записи из таблицы будут сохранены в файл.

3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

1. Как добавить запись?

Для создания записи необходимо щелкнуть по кнопке "Добавить бронь" в правом нижнем углу окна программы. Далее необходимо заполнить все поля появившегося окна "Добавление брони": фамилия, имя, отчество, адрес электронной почты, номер телефона (без пробелов и разделительных знаков), комментарий (если комментария нет, то ввести "нет" или оставить поле пустым, если комментарий состоит из нескольких слов, то вводить без пробелов), дата заезда и выезда, количество гостей (не менее одного), тип номера и способ оплаты выбрать из предлагаемого перечня. После заполнения необходимых данных о брони нажать на кнопку "Добавить". Ваша новая запись успешно сохранена и сразу отобразится в таблице.

2. Как добавить запись из файла?

Для добавления записи из файла необходимо щелкнуть по кнопке "Файлы" в левом верхнем углу окна программы, затем выбрать действие "Добавить из файла". Далее записи из файла отобразятся в таблице.

3. Как редактировать запись?

Для редактирования записи необходимо щелкнуть по кнопке "Редактирование" в левом верхнем углу окна программы и выбрать поле записи, которое нужно изменить. Далее в появившемся окне ввести номер записи, которую нужно отредактировать, и ввести измененную запись.

4. Как удалить запись из таблицы?

Для удаления записи из таблицы необходимо щелкнуть по кнопке "Удалить бронь" в правом нижнем углу окна программы. Далее в появившемся окне ввести номер нужной записи (номер отображается в левой части таблицы). После заполнения поля нажать на кнопку "Удалить". Запись успешно удалена из таблицы.

5. Как сортировать записи?

Для сортировки записей необходимо щелкнуть по кнопке "Сортировка" в верхнем левом углу окна программы. Далее выбрать направление сортировки из предложенного перечня. Отсортированные записи отобразятся в таблице.

6. Как произвести поиск по записям?

Для поиска по записям необходимо щелкнуть по кнопке "Поиск" в верхнем левом углу окна программы. Далее выбрать направление поиска из предложенного перечня. Найденные записи отобразятся в появившемся окне.

7. Как сохранить записи таблицы в файл?

Для сохранения записей таблицы в файл необходимо щелкнуть по кнопке "Файлы" в левом верхнем углу окна программы, затем выбрать действие "Сохранить в файл". Все записи из таблицы будут сохранены в файл.

8. Как посмотреть диаграмму?

Для просмотра диаграммы необходимо щелкнуть по кнопке "Диаграмма" в правом нижнем углу окна программы. Далее в появившемся окне графически отобразится столбчатая гистограмма (по горизонтали - тип номера, по вертикали - количество забронированных номеров).

9. Как удалить все записи?

Для удаления всех записей необходимо щелкнуть по кнопке "Удалить все" в правом нижнем углу программы. Далее в появившемся окне выбрать действие: "Да, удалить", после чего все записи из таблицы будут удалены, или "Отмена", после чего все останется неизменным.

10. Как сохранить записи?

Для сохранения записей необходимо щелкнуть по кнопке "Файлы" в левом верхнем углу программы, затем выбрать действие "Сохранить в файл". Все записи будут сохранены в файл.

ЗАКЛЮЧЕНИЕ

В ходе проведения учебной практики бакалавра был разработан проект «Отельный бизнес».

В результате учебной практики бакалавра были выполнены следующие задачи:

- разделить студентов на группы, в которых они будут реализовывать итоговые проекты согласно предметным областям, согласовать выбранные темы, назначить сроки промежуточных встреч и консультаций;
- осуществить выбор языка(-ов) программирования, фреймворков и библиотек, среду(-ы) разработки, систему контроля версий, сервис хостинга проектов, обосновать соответствующий выбор;
- подготовить доклад с использованием инструментов визуализации (презентации, графики, блок-схемы, UML-диаграммы и пр.), в котором будет представлен отчет команды о проделанной работе, общая информация о разработанном проекте, его преимущества, недостатки и перспективы дальнейшего развития.

В ходе разработки и реализации были получены следующие навыки:

- выбор предметной области на основе имеющихся знаний в сфере разработки программного обеспечения, умение грамотно сформулировать тему проекта, область его применения и главные отличия относительно уже имеющихся проектов;
- подбор и оценка технологий разработки и вспомогательного программного обеспечения для реализации итогового проекта в соответствии с выбранной темой;
- оценка функциональности проекта, которая будет реализована к соответствующему сроку, умение грамотно распределить роли и задачи внутри команды;

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Основная литература

1. Моделирование информационных ресурсов [Электронный ресурс]: учебно-методический/ Составитель Огнев Э.Н. - Кемерово : Кемеровский государственный университет культуры и искусств, 2013. - 36 с. : ил., табл. - URL: <http://biblioclub.ru/index.php?page=book&id=274218>
2. Коваленко, Ю.В. Информационно-поисковые системы [Электронный ресурс]: учебно-методическое пособие / Ю.В. Коваленко, Т.А. Сергиенко. — Омск: Омская юридическая академия, 2017. — 38 с.— Режим доступа: <http://www.iprbookshop.ru/66817.html>
3. Маюрникова, Л. А. Основы научных исследований в научно-технической сфере [Электронный ресурс] : учебно-методическое пособие / Л. А. Маюрникова, С. В. Новосёлов. — Кемерово : Кемеровский технологический институт пищевой промышленности, 2009. — 123 с. — Режим доступа: <http://www.iprbookshop.ru/14381.html>
4. Вайнштейн, М. З. Основы научных исследований [Электронный ресурс] : учебное пособие / М. З. Вайнштейн, В. М. Вайнштейн, О. В. Кононова. — Йошкар-Ола : Марийский государственный технический университет, Поволжский государственный технологический университет, ЭБС АСВ, 2011. — 216 с. — Режим доступа: <http://www.iprbookshop.ru/22586.html>
5. Мокий, М.С. Методология научных исследований [Текст]: учебник / М.С. Мокий, А.Л. Никифоров, В.С. Мокий. - М.: Юрайт, 2015. - 255 с.

6. Рогов, В.А. Методика и практика технических экспериментов [Текст]: учеб.пособие / В.А. Рогов, А.В. Антонов, Г.Г. Поздняк. – М.: Академия, 2005. – 288 с.

Дополнительная литература

7. Щербаков, А. Интернет-аналитика [Электронный ресурс]: поиск и оценка информации в web-ресурсах: практическое пособие / А. Щербаков. - М.: Книжный мир, 2012. - 78 с. - URL: <http://biblioclub.ru>
8. Моделирование систем [Текст]: учебник для вузов / С.И. Дворецкий, Ю.Л. Муромцев, В.А. Погонин, А.Г. Схиртладзе. – М.: Академия, 2009. – 320 с.
9. Порсев, Е. Г. Организация и планирование экспериментов [Электронный ресурс] : учебное пособие / Е. Г. Порсев.— Новосибирск : Новосибирский государственный технический университет, 2010. — 155 с. — Режим доступа: <http://www.iprbookshop.ru/45415.html>
10. Мюллер. Джон Пол, Семпф. Билл, Сфер. Чак. С# для чайников.: Пер. с англ. - СПб.: ООО "Диалектика", 2019. - 608 с. : ил. - Парал. тит. англ. 11.
11. Абрамян М. Э. Visual C# на примерах. — СПб.: БХВ-Петербург, 2008. — 496 с.: ил.
12. Петцольд Ч. Программирование с использованием Microsoft Windows Forms. Мастер-класс/Пер. с англ. – М.: Русская редакция; СПб.: Питер, 2006. – 432 стр.: ил.
13. Балена Франческо, Димауро Джузеппе Современная практика программирования на Microsoft Visual Basic и Visual C#/Пер. с англ. – М.: Издательско-торговый дом «Русская редакция», 2006. – 640 стр.: ил.
14. Васильев, Алексей. Программирование на C# для начинающих. Основные сведения/Алексей Васильев. – Москва: Эксмо, 2018. – 592 с. – (Российский компьютерный бестселлер).

Текст программы:

ReservationList.h:

```
#pragma once
#include <iostream>
#include <string>
#include <fstream>
using namespace std;

class ReservationList : public User{
public:
    string m_typeOfRoom{"-"};
    string m_numberOfSeats{"-"};
    string m_comments{"-"};
    string m_names[3]{"-", "-", "-"};
    string m_email{"-"};
    string m_phoneNumber{"-"};
    string m_paymentMethod{"-"};
    string m_dateOfArrival = "-";
    string m_dateOfDeparture = "-";
    string m_price{"-"};
    void printData() const;
    friend ifstream& operator >> (ifstream& ifs, ReservationList& reserv);
    friend ofstream &operator<<(ofstream &ofs, const ReservationList
&reserv);
    ReservationList(string, string , string, string , string , string ,
string ,string , string , string , string , string);
    ReservationList();
};
```

ReservationList.cpp:

```
#include <iostream>
#include <string>
#include <fstream>
#include "../ReservationList.h"

ifstream &operator>>(ifstream &ifs, ReservationList &reserv)
{
    ifs >> reserv.m_typeOfRoom >> reserv.m_dateOfArrival >>
reserv.m_dateOfDeparture >> reserv.m_numberOfSeats >> reserv.m_comments >>
reserv.m_names[0] >> reserv.m_names[1] >> reserv.m_names[2] >> reserv.m_email
>> reserv.m_phoneNumber >> reserv.m_paymentMethod >> reserv.m_price;
    return ifs;
}
```



```

}

ofstream &operator<<(ofstream &ofs, const ReservationList &reserv)
{
    ofs << reserv.m_typeOfRoom << " " << reserv.m_dateOfArrival << " " <<
    reserv.m_dateOfDeparture << " " << reserv.m_numberOfSeats << " " <<
    reserv.m_comments << " " << reserv.m_names[0] << " " << reserv.m_names[1] <<
    " " << reserv.m_names[2] << " " << reserv.m_email << " " <<
    reserv.m_phoneNumber << " " << reserv.m_paymentMethod << " " <<
    reserv.m_price << endl;
    return ofs;
}

ReservationList::ReservationList(string _typeOfRoom, string _dateOfArrival,
string _dateOfDeparture, string _numberOfSeats, string _comments, string
_surname, string _name, string _secondName, string _email, string
_phoneNumber, string _paymentMethod, string _price){
    m_typeOfRoom = _typeOfRoom;
    m_dateOfArrival = _dateOfArrival;
    m_dateOfDeparture = _dateOfDeparture;
    m_numberOfSeats = _numberOfSeats;
    m_comments = _comments;
    m_names[0] = _surname;
    m_names[1] = _name;
    m_names[2] = _secondName;
    m_email = _email;
    m_phoneNumber = _phoneNumber;
    m_paymentMethod = _paymentMethod;
    m_price = _price;
}

ReservationList::ReservationList(){}

```

MyVector.h:

```

#pragma once
#include "Iterator.h"
#include "../ReservationList.cpp"
class MyVector
{
public:
    typedef Iterator iterator;
    typedef Iterator const_iterator;
    MyVector();
    iterator begin();

```

```

        iterator end();
        const_iterator begin() const;
        const_iterator end() const;
        void AddMemory();
        void PushBack(const ReservationList &);
        void Remove(size_t);
        size_t Size();
        ReservationList &operator[](int i);

private:
        ReservationList *m_array{};
        size_t m_size{};
        size_t m_capacity{};
};

```

MyVector.cpp:

```

#include "MyVector.h"
#include <iostream>
using namespace std;
MyVector::MyVector()
{
    m_array = new ReservationList[1];
    m_size = 0;
    m_capacity = 1;
}
MyVector::iterator MyVector::begin()
{
    return iterator(m_array);
}
MyVector::iterator MyVector::end()
{
    return iterator(m_array + m_size);
}
MyVector::const_iterator MyVector::begin() const
{
    return const_iterator(m_array);
}
MyVector::const_iterator MyVector::end() const
{
    return const_iterator(m_array + m_size);
}
size_t MyVector::Size()
{

```

```

        return m_size;
    }
void MyVector::AddMemory()
{
    m_capacity *= 2;
    ReservationList *tmp = m_array;
    m_array = new ReservationList[m_capacity];
    for (size_t i = 0; i < m_size; i++)
    {
        m_array[i] = tmp[i];
    }
    delete[] tmp;
}
void MyVector::PushBack(const ReservationList &value)
{
    if (m_size >= m_capacity)
    {
        AddMemory();
    }
    m_array[m_size] = value;
    m_size++;
}
void MyVector::Remove(size_t index)
{
    for (size_t i = index + 1; i < m_size; ++i)
    {
        m_array[i - 1] = m_array[i];
    }
    m_size--;
}

ReservationList &MyVector::operator[](int i)
{
    return m_array[i];
}

```

Iterator.h:

```

#pragma once
#include "../ReservationList.h"

class Iterator {
    friend class MyVector;
public:
    Iterator(const Iterator&);

```

```

    bool operator==(const Iterator&) const;
    bool operator!=(const Iterator&) const;
    Iterator& operator++();
    ReservationList& operator*() const;
private:
    ReservationList* m_reservatonList;
    Iterator(ReservationList* p) : m_reservatonList(p) {}
};

```

Iterator.cpp:

```

#include "Iterator.h"
#include <iostream>
using namespace std;
Iterator::Iterator(const Iterator& it) : m_reservatonList(it.m_reservatonList)
{}
bool Iterator::operator==(const Iterator& it) const {
    return m_reservatonList == it.m_reservatonList;
}
bool Iterator::operator!=(const Iterator& it) const {
    return m_reservatonList != it.m_reservatonList;
}
Iterator& Iterator::operator++() {
    m_reservatonList++;
    return *this;
}
ReservationList& Iterator::operator*() const {
    return *m_reservatonList;
}

```

Mainwindow.h:

```

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QToolBar>
#include <QMenu>
#include <QWidget>
#include <QLabel>
#include <QVBoxLayout>
#include <QPushButton>
#include <QLineEdit>
#include <QDateEdit>
#include <QTableView>
#include <QStandardItem>
#include <QHeaderView>

```

```

#include <QComboBox>
#include <QSpinBox>

QT_BEGIN_NAMESPACE
namespace Ui {
class MainWindow;
}
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    // init methods
    void addBookingWindow();
    void addBooking();
    void deleteBookingWidget();
    void deleteBooking();
    void deleteAll();
    void addDataFromFile();
    void addDataToFile();
    void setupCoreWidgets();
    void tableView();
    void testLayout();
    void arrivalIncSort();
    void arrivalDecSort();
    void departureIncSort();
    void departureDecSort();
    void fioSearchLayout();
    void fioFind();
    void dateSearchLayout();
    void dateFind();
    void typeSearchLayout();
    void typeSearchFunction();
    void commentSearchLayout();
    void commentFind();
    void diogram();
    void arrivalDateChange();
    void departureDateChange();
    void phoneChange();

```

```

void emailChange();
void arrivalDateChangePush();
void departureDateChangePush();
void phoneChangePush();
void emailChangePush();
void clearAllRecords();
void exit();

private:
    Ui::MainWindow *ui;
    //ToolBar config
    QToolBar *toolbar;
    QAction *newToolBarAction;
    QAction *openToolBarAction;
    QAction *closeToolBarAction;
    //Menu config
    QMenu *fileMenu;
    QMenu *sortMenu;
    QMenu *searchMenu;
    QMenu *editMenu;
    QAction *aboutAction;
    QAction *saveAction;
    QAction *cancelAction;
    QAction *openAction;
    QAction *newAction;
    QAction *arrivalIncSortAction;
    QAction *arrivalDecSortAction;
    QAction *departureIncSortAction;
    QAction *departureDecSortAction;
    QAction *fioSearch;
    QAction *dateSearch;
    QAction *typeSearch;
    QAction *commentSearch;
    QAction *arrivalDateEdit;
    QAction *departureDateEdit;
    QAction *emailEdit;
    QAction *phoneEdit;
    //Main Widget
    QWidget *mainWidget;
    QVBoxLayout *centralWidgetLayout;
    QGridLayout *formLayout;
    QHBoxLayout *buttonsLayout;
    QLabel *nameLabel;
    QLabel *dateOfBirthLabel;

```



```

QLabel *phoneNumberLabel;
QPushButton *savePushButton;
QPushButton *newPushButton;
QPushButton *graphicPushButton;
QPushButton *clearPushButton;
QPushButton *quitPushButton;
QDateEdit *dateOfBirthEdit;
QTableView *appTable;
QStandardItemModel *model;
// AddBooking
QWidget *addBookingW;
QLineEdit *surnamenameLineEdit;
QLineEdit *nameLineEdit;
QLineEdit *fathersnameLineEdit;
QLineEdit *emailLineEdit;
QLineEdit *phoneNumberLineEdit;
QLineEdit *commentLineEdit;
QComboBox *roomType;
QDateEdit *arrivalDate;
QDateEdit *departureDate;
QSpinBox *guestsQuantity;
QComboBox *payWay;
// DeleteBooking
QWidget *deleteWidget;
QSpinBox *bookInd;
// Find widget
QWidget *fioFindWidget;
QTableView *findTable;
QStandardItemModel *findItem;
QLineEdit *inputForm;
QString *name;
QWidget *dateFindWidget;
QDateEdit *inputDateForm;
QDate *dateToFind;
QWidget *typeFindWidget;
QComboBox *inputTypeForm;
QWidget *commentFindWidget;
// Edit widget
QWidget *arrivalDateEditWidget;
QDateEdit *arrivalDateEditInput;
QWidget *departureDateEditWidget;
QDateEdit *departureDateEditInput;
QWidget *phoneEditWidget;
QLineEdit *phoneEditInput;

```

```

QWidget *emailEditWidget;
QLineEdit *emailEditInput;
// Icons
QPixmap fileIcon;
QPixmap saveIcon;
QPixmap closeIcon;
QPixmap plusIcon;
QPixmap upIcon;
QPixmap downIcon;
QPixmap xIcon;
QPixmap searchIcon;
QPixmap editIcon;
QPixmap graphicIcon;
QPixmap deleteIcon;
// init methods
void createIcons();
void createMenuBar();
};
#endif // MAINWINDOW_H

```

Mainwindow.cpp:

```

#include "mainwindow.h"
#include "../ui_mainwindow.h"
#include "MyVector.cpp"
#include "MyVector.h"
#include <QApplication>
#include <QPushButton>
#include <QSpinBox>
#include <QComboBox>
#include <QFormLayout>
#include <QAction>
#include <QtCharts>
#include <QRegularExpression>
#include <cstring>
#include <string>
#include <fstream>
#include <iostream>

MyVector bookings;

void MainWindow::createIcons() {
    fileIcon = QPixmap("../OP/file.svg");
    saveIcon = QPixmap("../OP/save.svg");
    closeIcon = QPixmap("../OP/off.svg");
    plusIcon = QPixmap("../OP/plus.svg");
    downIcon = QPixmap("../OP/down.svg");
    upIcon = QPixmap("../OP/up.svg");
    xIcon = QPixmap("../OP/x.svg");
    searchIcon = QPixmap("../OP/search.svg");
    editIcon = QPixmap("../OP/pencil.svg");
    graphicIcon = QPixmap("../OP/graphic.svg");
    deleteIcon = QPixmap("../OP/delete.svg");
}

```

```

void MainWindow::tableView() {
    appTable = new QTableView();
    model = new QStandardItemModel(1, 8, this);
    appTable->setContextMenuPolicy(Qt::CustomContextMenu);
    appTable->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);
    appTable->setEditTriggers(QAbstractItemView::NoEditTriggers);
    /** Note **/
    model->setHorizontalHeaderItem(0, new QStandardItem(QString("ФИО")));
    model->setHorizontalHeaderItem(1, new QStandardItem(QString("Дата
заезда")));
    model->setHorizontalHeaderItem(2, new QStandardItem(QString("Дата
выезда")));
    model->setHorizontalHeaderItem(3, new QStandardItem(QString("Тип
номера")));
    model->setHorizontalHeaderItem(4, new QStandardItem(QString("Кол-во
гостей")));
    model->setHorizontalHeaderItem(5, new QStandardItem(QString("Email")));
    model->setHorizontalHeaderItem(6, new QStandardItem(QString("Номер
телефона")));
    model->setHorizontalHeaderItem(7, new
QStandardItem(QString("Комментарий")));
    model->setHorizontalHeaderItem(8, new QStandardItem(QString("Способ
оплаты")));
    model->setHorizontalHeaderItem(9, new
QStandardItem(QString("Стоимость")));
    appTable->setModel(model);
    model->removeRow(0);
    for(int i = 0; i < bookings.Size(); i++){
        string fio = bookings[i].m_names[0] + " " + bookings[i].m_names[1] +
" " + bookings[i].m_names[2];
        QStandardItem *item1 = new
QStandardItem(QString::fromStdString(fio));
        QStandardItem *item2 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfArrival));
        QStandardItem *item3 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfDeparture));
        QStandardItem *item4 = new
QStandardItem(QString::fromStdString(bookings[i].m_typeOfRoom));
        QStandardItem *item5 = new
QStandardItem(QString::fromStdString(bookings[i].m_numberOfSeats));
        QStandardItem *item6 = new
QStandardItem(QString::fromStdString(bookings[i].m_email));
        QStandardItem *item7 = new
QStandardItem(QString::fromStdString(bookings[i].m_phoneNumber));
        QStandardItem *item8 = new
QStandardItem(QString::fromStdString(bookings[i].m_comments));
        QStandardItem *item9 = new
QStandardItem(QString::fromStdString(bookings[i].m_paymentMethod));
        QStandardItem *item10 = new
QStandardItem(QString::fromStdString(bookings[i].m_price));
        model->setItem(i, 0, item1);
        model->setItem(i, 1, item2);
        model->setItem(i, 2, item3);
        model->setItem(i, 3, item4);
        model->setItem(i, 4, item5);
        model->setItem(i, 5, item6);
        model->setItem(i, 6, item7);
        model->setItem(i, 7, item8);
        model->setItem(i, 8, item9);
        model->setItem(i, 9, item10);
    }
}

void MainWindow::setupCoreWidgets() {
    //testLayout();
}

```

```

mainWidget = new QWidget();
centralWidgetLayout = new QVBoxLayout();
formLayout = new QGridLayout();
buttonsLayout = new QHBoxLayout();
savePushButton = new QPushButton(xIcon, "Удалить бронь", this);
newPushButton = new QPushButton(plusIcon, "Добавить бронь", this);
graphicPushButton = new QPushButton(graphicIcon, "Диаграмма", this);
clearPushButton = new QPushButton(deleteIcon, "Удалить все", this);
quitPushButton = new QPushButton(closeIcon, "Выйти", this);
// TableView
tableView();
buttonsLayout->addStretch();
buttonsLayout->addWidget(newPushButton);
buttonsLayout->addWidget(savePushButton);
buttonsLayout->addWidget(graphicPushButton);
buttonsLayout->addWidget(clearPushButton);
buttonsLayout->addWidget(quitPushButton);

}

void MainWindow::deleteBooking() {
    if (bookings.Size() != 0) {
        int ind = bookInd->value();
        model->removeRows(0, bookings.Size());
        bookings.Remove(ind - 1);
        for (int i = 0; i < bookings.Size(); i++)
        {
            string fio = bookings[i].m_names[0] + " " +
bookings[i].m_names[1] + " " + bookings[i].m_names[2];
            QStandardItem *item1 = new
QStandardItem(QString::fromStdString(fio));
            QStandardItem *item2 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfArrival));
            QStandardItem *item3 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfDeparture));
            QStandardItem *item4 = new
QStandardItem(QString::fromStdString(bookings[i].m_typeOfRoom));
            QStandardItem *item5 = new
QStandardItem(QString::fromStdString(bookings[i].m_numberOfSeats));
            QStandardItem *item6 = new
QStandardItem(QString::fromStdString(bookings[i].m_email));
            QStandardItem *item7 = new
QStandardItem(QString::fromStdString(bookings[i].m_phoneNumber));
            QStandardItem *item8 = new
QStandardItem(QString::fromStdString(bookings[i].m_comments));
            QStandardItem *item9 = new
QStandardItem(QString::fromStdString(bookings[i].m_paymentMethod));
            QStandardItem *item10 = new
QStandardItem(QString::fromStdString(bookings[i].m_price));
            model->appendRow({item1, item2, item3, item4, item5, item6,
item7, item8, item9, item10});
        }
    }
    deleteWidget->close();
}

void MainWindow::deleteAll() {
    model->removeRows(0, bookings.Size());
    int len = bookings.Size();
    for (int i = 0; i < len; i++) {
        bookings.Remove(0);
    }
}

void MainWindow::clearAllRecords()

```

```

{
    int status = QMessageBox::question(this, "Вы уверены?",
        "После подтверждения будут удалены все записи " "<p> Вы уверены, что  

хотите удалить все записи? ", "Да, удалить", "Отмена", QString(), 1, 1);
    if (status == 0) {
        deleteAll();
    }
}

void MainWindow::exit()
{
    int status = QMessageBox::question(this, "Вы уверены?",
        "Вы уверены, что хотите выйти? :( " "<p> Выберите наиболее удобный способ  

", "Выйти", "Сохранить и выйти", "Отмена", 2, 2);
    if (status == 0) {
        QApplication::quit();
    }
    else if (status == 1) {
        addDataToFile();
        QApplication::quit();
    }
}

void MainWindow::deleteBookingWidget() {
    deleteWidget = new QWidget;
    QFormLayout *inputLayout = new QFormLayout;
    QPushButton *confirm = new QPushButton(xIcon, "Удалить", this);
    bookInd = new QSpinBox;
    bookInd->setRange(1, bookings.Size());
    inputLayout->addRow("Введите номер удаляемой брони:", bookInd);
    inputLayout->addRow("", confirm);
    deleteWidget->setLayout(inputLayout);
    connect(confirm, SIGNAL(clicked(bool)), this, SLOT(deleteBooking()));
    deleteWidget->setWindowTitle("Удаление записи...");
    deleteWidget->show();
}

void MainWindow::addBooking() {
    QString surname = QString(surnamenameLineEdit->text());
    QString nameB = QString(nameLineEdit->text());
    QString fathersname = QString(fathersnameLineEdit->text());
    QString email = QString(emailLineEdit->text());
    QString phone = QString(phoneNumberLineEdit->text());
    QString comment = QString(commentLineEdit->text());
    QString type = QString(roomType->currentText());
    QDate arrivalDateB = QDate(arrivalDate->date());
    QDate departureDateB = QDate(departureDate->date());
    int quantity = guestsQuantity->value();
    QString pay = QString(payWay->currentText());
    ReservationList temp;
    if (phone.length() == 16) {
        if(surname != "") {
            temp.m_names[0] = surname.toStdString();
        }
        if (nameB != "") {
            temp.m_names[1] = nameB.toStdString();
        }
        if (fathersname != "") {
            temp.m_names[2] = fathersname.toStdString();
        }
        if (email != "") {
            temp.m_email = email.toStdString();
        }
        if (phone != "") {
            temp.m_phoneNumber = phone.toStdString();
        }
    }
}

```

```

    }
    if (comment != "") {
        temp.m_comments = comment.toStdString();
    }
    if (type != "") {
        temp.m_typeOfRoom = type.toStdString();
    }
    if (arrivalDateB.day() < 10 || arrivalDateB.month() < 10)
    {
        if(arrivalDateB.day() < 10 && arrivalDateB.month() < 10)
        {
            temp.m_dateOfArrival = "0" + to_string(arrivalDateB.day()) +
            "." + "0" + to_string(arrivalDateB.month()) + "." +
            to_string(arrivalDateB.year());
        }
        if(arrivalDateB.day() < 10 && arrivalDateB.month() >= 10)
        {
            temp.m_dateOfArrival = "0" + to_string(arrivalDateB.day()) +
            "." + to_string(arrivalDateB.month()) + "." + to_string(arrivalDateB.year());
        }
        if(arrivalDateB.day() >= 10 && arrivalDateB.month() < 10)
        {
            temp.m_dateOfArrival = to_string(arrivalDateB.day()) + "." +
            "0" + to_string(arrivalDateB.month()) + "." + to_string(arrivalDateB.year());
        }
    }
    else
    {
        temp.m_dateOfArrival = to_string(arrivalDateB.day()) + "." +
        to_string(arrivalDateB.month()) + "." + to_string(arrivalDateB.year());
    }
    if (departureDateB.day() < 10 || departureDateB.month() < 10)
    {
        if(departureDateB.day() < 10 && departureDateB.month() < 10)
        {
            temp.m_dateOfDeparture = "0" +
            to_string(departureDateB.day()) + "." + "0" +
            to_string(departureDateB.month()) + "." + to_string(departureDateB.year());
        }
        if(departureDateB.day() < 10 && departureDateB.month() >= 10)
        {
            temp.m_dateOfDeparture = "0" +
            to_string(departureDateB.day()) + "." + to_string(departureDateB.month()) +
            "." + to_string(departureDateB.year());
        }
        if(departureDateB.day() >= 10 && departureDateB.month() < 10)
        {
            temp.m_dateOfDeparture = to_string(departureDateB.day()) +
            "." + "0" + to_string(departureDateB.month()) + "." +
            to_string(departureDateB.year());
        }
    }
    else
    {
        temp.m_dateOfArrival = to_string(arrivalDateB.day()) + "." +
        to_string(arrivalDateB.month()) + "." + to_string(arrivalDateB.year());
    }
    temp.m_numberOfSeats = to_string(quantity);
    temp.m_paymentMethod = pay.toStdString();
    int d, m, y;
    char d1[11]{};
    char d2[11]{};
    string d11 = temp.m_dateOfArrival;
    string d22 = temp.m_dateOfDeparture;
    strcpy(d1, d11.c_str());

```

```

        strcpy(d2, d22.c_str());
        sscanf(d1, "%d.%d.%d", &d, &m, &y);
        int date1 = y * 365 + m * 31 + d;
        sscanf(d2, "%d.%d.%d", &d, &m, &y);
        int date2 = y * 365 + m * 31 + d;
        if (temp.m_typeOfRoom == "Стандарт")
        {
            temp.m_price = to_string((date2 - date1) * 5000);
        }
        else
        {
            temp.m_price = to_string((date2 - date1) * 10000);
        }
        bookings.PushBack(temp);
        string fio = temp.m_names[0] + " " + temp.m_names[1] + " " +
temp.m_names[2];
        QStandardItem *item1 = new
QStandardItem(QString::fromStdString(fio));
        QStandardItem *item2 = new
QStandardItem(QString::fromStdString(temp.m_dateOfArrival));
        QStandardItem *item3 = new
QStandardItem(QString::fromStdString(temp.m_dateOfDeparture));
        QStandardItem *item4 = new
QStandardItem(QString::fromStdString(temp.m_typeOfRoom));
        QStandardItem *item5 = new
QStandardItem(QString::fromStdString(temp.m_numberOfSeats));
        QStandardItem *item6 = new
QStandardItem(QString::fromStdString(temp.m_email));
        QStandardItem *item7 = new
QStandardItem(QString::fromStdString(temp.m_phoneNumber));
        QStandardItem *item8 = new
QStandardItem(QString::fromStdString(temp.m_comments));
        QStandardItem *item9 = new
QStandardItem(QString::fromStdString(temp.m_paymentMethod));
        QStandardItem *item10 = new
QStandardItem(QString::fromStdString(temp.m_price));
        model->appendRow({item1, item2, item3, item4, item5, item6, item7,
item8, item9, item10});
        addBookingW->close();
    }
    else {
        QMessageBox::warning(this, "Неверный ввод",
        "Введенный номер телефона слишком короткий " "<p> Попробуйте еще
раз", "Ок", QString(), QString(), 1, 1);
        addBookingW->close();
    }
}

void MainWindow::addBookingWindow() {
    //int quantity{};
    addBookingW = new QWidget;
    surnameLineEdit = new QLineEdit;
    QRegularExpression nameCheck("[А-Яа-я]{0,32}");
    QValidator *validator9 = new QRegularExpressionValidator(nameCheck,
this);
    surnameLineEdit->setValidator(validator9);
    nameLineEdit = new QLineEdit;
    nameLineEdit->setValidator(validator9);
    fathersnameLineEdit = new QLineEdit;
    emailLineEdit = new QLineEdit;
    fathersnameLineEdit->setValidator(validator9);
    QRegularExpression emailCheck("[a-z.0-9@]{0,36}");
    QValidator *validator7 = new QRegularExpressionValidator(emailCheck,
this);
    emailLineEdit->setValidator(validator7);
}

```

```

    phoneNumberLineEdit = new QLineEdit;
    phoneNumberLineEdit->setInputMask("+7(000)000-00-00");
    commentLineEdit = new QLineEdit;
    QRegularExpression commentCheck("[a-zA-Z-A-Я-a-я-0-9.@!?!+]{0,100}");
    QValidator *validator8 = new QRegularExpressionValidator(commentCheck,
this);
    commentLineEdit->setValidator(validator8);
    roomType = new QComboBox;
    QStringList types = {"Стандарт", "Люкс"};
    roomType->addItem(types);
    arrivalDate = new QDateEdit(QDate::currentDate());
    arrivalDate->setMinimumDate(QDate::currentDate());
    arrivalDate->setMaximumDate(QDate::currentDate().addYears(1));
    departureDate = new QDateEdit(QDate::currentDate());
    departureDate->setMinimumDate(QDate::currentDate().addDays(1));
    departureDate-
>setMaximumDate(QDate::currentDate().addDays(1).addYears(1));
    guestsQuantity = new QSpinBox;
    guestsQuantity->setRange(1, 6);
    payWay = new QComboBox;
    QStringList payWayList = {"Онлайн", "Наличными", "Картой"};
    payWay->addItem(payWayList);
    QPushButton *add = new QPushButton(plusIcon, "Добавить", this);
    connect(add, SIGNAL(clicked(bool)), this, SLOT(addBooking()));
    QFormLayout *layoutLeft = new QFormLayout;
    QFormLayout *layoutRight = new QFormLayout;
    QGridLayout *layout = new QGridLayout;
    layoutLeft->addRow("Фамилия:", surnameLineEdit);
    layoutLeft->addRow("Имя:", nameLineEdit);
    layoutLeft->addRow("Отчество:", fathersnameLineEdit);
    layoutLeft->addRow("Email:", emailLineEdit);
    layoutLeft->addRow("Номер телефона:", phoneNumberLineEdit);
    layoutLeft->addRow("Комментарий:", commentLineEdit);
    layoutRight->addRow("Тип номера:", roomType);
    layoutRight->addRow("Дата заезда:", arrivalDate);
    layoutRight->addRow("Дата выезда:", departureDate);
    layoutRight->addRow("Количество гостей:", guestsQuantity);
    layoutRight->addRow("Способ оплаты:", payWay);
    layoutRight->addRow(add);
    layout->addLayout(layoutLeft, 0, 0);
    layout->addLayout(layoutRight, 0, 1);
    addBookingW->setLayout(layout);
    addBookingW->setWindowTitle("Добавление брони...");
    addBookingW->show();
}

void MainWindow::testLayout(){
    QWidget *window = new QWidget;
    window->setWindowTitle("Найденные брони:");
    window->resize(1280, 720);
    findTable = new QTableView();
    findItem = new QStandardItemModel(1, 10, this);
    findTable->setContextMenuPolicy(Qt::CustomContextMenu);
    findTable->horizontalHeader()-
>setSectionResizeMode(QHeaderView::Stretch);
    /** Note **/
    findItem->setHorizontalHeaderItem(0, new QStandardItem(QString("ФИО")));
    findItem->setHorizontalHeaderItem(1, new QStandardItem(QString("Дата
заезда")));
    findItem->setHorizontalHeaderItem(2, new QStandardItem(QString("Дата
выезда")));
    findItem->setHorizontalHeaderItem(3, new QStandardItem(QString("Тип
номера")));
    findItem->setHorizontalHeaderItem(4, new QStandardItem(QString("Кол-во
гостей")));

```



```

        findItem->setHorizontalHeaderItem(5, new
QStandardItem(QString("Email")));
        findItem->setHorizontalHeaderItem(6, new QStandardItem(QString("Номер
телефона")));
        findItem->setHorizontalHeaderItem(7, new
QStandardItem(QString("Комментарий")));
        findItem->setHorizontalHeaderItem(8, new QStandardItem(QString("Способ
оплаты")));
        findItem->setHorizontalHeaderItem(9, new
QStandardItem(QString("Стоимость")));
        findTable->setModel(findItem);
        QFormLayout *menuLauout = new QFormLayout;
        menuLauout->addWidget(findTable);
        window->setLayout(menuLauout);
        window->show();
    }

void MainWindow::addDataFromFile()
{ // Ввод из текстового файла
    ifstream fin;
    fin.open("/home/stepan/Документы/OP/inputStudents.txt");
    int newBookings{};
    while (!fin.eof())
    {
        ReservationList temp{};
        fin >> temp;
        if(temp.m_dateOfArrival != "-"){
            bookings.PushBack(temp);
            newBookings++;
        }
    }
    fin.close();
    for(int i = bookings.Size() - newBookings; i < bookings.Size(); i++)
    {
        string fio = bookings[i].m_names[0] + " " + bookings[i].m_names[1] +
" " + bookings[i].m_names[2];
        QStandardItem *item1 = new
QStandardItem(QString::fromStdString(fio));
        QStandardItem *item2 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfArrival));
        QStandardItem *item3 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfDeparture));
        QStandardItem *item4 = new
QStandardItem(QString::fromStdString(bookings[i].m_typeOfRoom));
        QStandardItem *item5 = new
QStandardItem(QString::fromStdString(bookings[i].m_numberOfSeats));
        QStandardItem *item6 = new
QStandardItem(QString::fromStdString(bookings[i].m_email));
        QStandardItem *item7 = new
QStandardItem(QString::fromStdString(bookings[i].m_phoneNumber));
        QStandardItem *item8 = new
QStandardItem(QString::fromStdString(bookings[i].m_comments));
        QStandardItem *item9 = new
QStandardItem(QString::fromStdString(bookings[i].m_paymentMethod));
        QStandardItem *item10 = new
QStandardItem(QString::fromStdString(bookings[i].m_price));
        model->appendRow({item1, item2, item3, item4, item5, item6, item7,
item8, item9, item10});
    }
}

void MainWindow::addDataToFile() {
    ofstream fout;
    fout.open("/home/stepan/Документы/OP/inputStudents.txt");
    for (int i = 0; i < bookings.Size(); i++)

```

```

    {
        fout << bookings[i];
    }
    fout.close();
}

void MainWindow::arrivalIncSort() {
    if (bookings.Size() != 0) {
        int d, m, y;
        for (int i = 0; i < bookings.Size() - 1; i++)
        {
            for (int j = 0; j < bookings.Size() - i - 1; j++)
            {
                char d1[11]{};
                char d2[11]{};
                string d11 = bookings[j].m_dateOfArrival;
                string d22 = bookings[j + 1].m_dateOfArrival;
                strcpy(d1, d11.c_str());
                strcpy(d2, d22.c_str());
                sscanf(d1, "%d.%d.%d", &d, &m, &y);
                int date1 = y * 365 + m * 31 + d;
                sscanf(d2, "%d.%d.%d", &d, &m, &y);
                int date2 = y * 365 + m * 31 + d;
                if (date1 > date2) // change here to change sorting direction
                {
                    ReservationList buff{};
                    buff = bookings[j];
                    bookings[j] = bookings[j + 1];
                    bookings[j + 1] = buff;
                }
            }
        }
        model->removeRows(0, bookings.Size());
        for (int i = 0; i < bookings.Size(); i++)
        {
            string fio = bookings[i].m_names[0] + " " +
bookings[i].m_names[1] + " " + bookings[i].m_names[2];
            QStandardItem *item1 = new
QStandardItem(QString::fromStdString(fio));
            QStandardItem *item2 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfArrival));
            QStandardItem *item3 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfDeparture));
            QStandardItem *item4 = new
QStandardItem(QString::fromStdString(bookings[i].m_typeOfRoom));
            QStandardItem *item5 = new
QStandardItem(QString::fromStdString(bookings[i].m_numberOfSeats));
            QStandardItem *item6 = new
QStandardItem(QString::fromStdString(bookings[i].m_email));
            QStandardItem *item7 = new
QStandardItem(QString::fromStdString(bookings[i].m_phoneNumber));
            QStandardItem *item8 = new
QStandardItem(QString::fromStdString(bookings[i].m_comments));
            QStandardItem *item9 = new
QStandardItem(QString::fromStdString(bookings[i].m_paymentMethod));
            QStandardItem *item10 = new
QStandardItem(QString::fromStdString(bookings[i].m_price));
            model->appendRow({item1, item2, item3, item4, item5, item6,
item7, item8, item9, item10});
        }
    }
}

void MainWindow::arrivalDecSort() {
    if (bookings.Size() != 0) {

```

```

int d, m, y;
for (int i = 0; i < bookings.Size() - 1; i++)
{
    for (int j = 0; j < bookings.Size() - i - 1; j++)
    {
        char d1[11]{};
        char d2[11]{};
        string d11 = bookings[j].m_dateOfArrival;
        string d22 = bookings[j + 1].m_dateOfArrival;
        strcpy(d1, d11.c_str());
        strcpy(d2, d22.c_str());
        sscanf(d1, "%d.%d.%d", &d, &m, &y);
        int date1 = y * 365 + m * 31 + d;
        sscanf(d2, "%d.%d.%d", &d, &m, &y);
        int date2 = y * 365 + m * 31 + d;
        if (date1 < date2) // change here to change sorting direction
        {
            ReservationList buff{};
            buff = bookings[j];
            bookings[j] = bookings[j + 1];
            bookings[j + 1] = buff;
        }
    }
}
model->removeRows(0, bookings.Size());
for (int i = 0; i < bookings.Size(); i++)
{
    string fio = bookings[i].m_names[0] + " " +
bookings[i].m_names[1] + " " + bookings[i].m_names[2];
    QStandardItem *item1 = new
QStandardItem(QString::fromStdString(fio));
    QStandardItem *item2 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfArrival));
    QStandardItem *item3 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfDeparture));
    QStandardItem *item4 = new
QStandardItem(QString::fromStdString(bookings[i].m_typeOfRoom));
    QStandardItem *item5 = new
QStandardItem(QString::fromStdString(bookings[i].m_numberOfSeats));
    QStandardItem *item6 = new
QStandardItem(QString::fromStdString(bookings[i].m_email));
    QStandardItem *item7 = new
QStandardItem(QString::fromStdString(bookings[i].m_phoneNumber));
    QStandardItem *item8 = new
QStandardItem(QString::fromStdString(bookings[i].m_comments));
    QStandardItem *item9 = new
QStandardItem(QString::fromStdString(bookings[i].m_paymentMethod));
    QStandardItem *item10 = new
QStandardItem(QString::fromStdString(bookings[i].m_price));
    model->appendRow({item1, item2, item3, item4, item5, item6,
item7, item8, item9, item10});
}
}

void MainWindow::departureIncSort(){
    if (bookings.Size() != 0) {
        int d, m, y;
        for (int i = 0; i < bookings.Size() - 1; i++)
        {
            for (int j = 0; j < bookings.Size() - i - 1; j++)
            {
                char d1[11]{};
                char d2[11]{};
                string d11 = bookings[j].m_dateOfDeparture;

```

```

        string d22 = bookings[j + 1].m_dateOfDeparture;
        strcpy(d1, d11.c_str());
        strcpy(d2, d22.c_str());
        sscanf(d1, "%d.%d.%d", &d, &m, &y);
        int date1 = y * 365 + m * 31 + d;
        sscanf(d2, "%d.%d.%d", &d, &m, &y);
        int date2 = y * 365 + m * 31 + d;
        if (date1 > date2) // change here to change sorting direction
        {
            ReservationList buff{};
            buff = bookings[j];
            bookings[j] = bookings[j + 1];
            bookings[j + 1] = buff;
        }
    }
    model->removeRows(0, bookings.Size());
    for (int i = 0; i < bookings.Size(); i++)
    {
        string fio = bookings[i].m_names[0] + " " +
bookings[i].m_names[1] + " " + bookings[i].m_names[2];
        QStandardItem *item1 = new
QStandardItem(QString::fromStdString(fio));
        QStandardItem *item2 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfArrival));
        QStandardItem *item3 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfDeparture));
        QStandardItem *item4 = new
QStandardItem(QString::fromStdString(bookings[i].m_typeOfRoom));
        QStandardItem *item5 = new
QStandardItem(QString::fromStdString(bookings[i].m_numberOfSeats));
        QStandardItem *item6 = new
QStandardItem(QString::fromStdString(bookings[i].m_email));
        QStandardItem *item7 = new
QStandardItem(QString::fromStdString(bookings[i].m_phoneNumber));
        QStandardItem *item8 = new
QStandardItem(QString::fromStdString(bookings[i].m_comments));
        QStandardItem *item9 = new
QStandardItem(QString::fromStdString(bookings[i].m_paymentMethod));
        QStandardItem *item10 = new
QStandardItem(QString::fromStdString(bookings[i].m_price));
        model->appendRow({item1, item2, item3, item4, item5, item6,
item7, item8, item9, item10});
    }
}

void MainWindow::departureDecSort() {
    if (bookings.Size() != 0) {
        int d, m, y;
        for (int i = 0; i < bookings.Size() - 1; i++)
        {
            for (int j = 0; j < bookings.Size() - i - 1; j++)
            {
                char d1[11]{};
                char d2[11]{};
                string d11 = bookings[j].m_dateOfArrival;
                string d22 = bookings[j + 1].m_dateOfArrival;
                strcpy(d1, d11.c_str());
                strcpy(d2, d22.c_str());
                sscanf(d1, "%d.%d.%d", &d, &m, &y);
                int date1 = y * 365 + m * 31 + d;
                sscanf(d2, "%d.%d.%d", &d, &m, &y);
                int date2 = y * 365 + m * 31 + d;
                if (date1 < date2) // change here to change sorting direction

```

```

        {
            ReservationList buff{};
            buff = bookings[j];
            bookings[j] = bookings[j + 1];
            bookings[j + 1] = buff;
        }
    }
    model->removeRows(0, bookings.Size());
    for (int i = 0; i < bookings.Size(); i++)
    {
        string fio = bookings[i].m_names[0] + " " +
bookings[i].m_names[1] + " " + bookings[i].m_names[2];
        QStandardItem *item1 = new
QStandardItem(QString::fromStdString(fio));
        QStandardItem *item2 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfArrival));
        QStandardItem *item3 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfDeparture));
        QStandardItem *item4 = new
QStandardItem(QString::fromStdString(bookings[i].m_typeOfRoom));
        QStandardItem *item5 = new
QStandardItem(QString::fromStdString(bookings[i].m_numberOfSeats));
        QStandardItem *item6 = new
QStandardItem(QString::fromStdString(bookings[i].m_email));
        QStandardItem *item7 = new
QStandardItem(QString::fromStdString(bookings[i].m_phoneNumber));
        QStandardItem *item8 = new
QStandardItem(QString::fromStdString(bookings[i].m_comments));
        QStandardItem *item9 = new
QStandardItem(QString::fromStdString(bookings[i].m_paymentMethod));
        QStandardItem *item10 = new
QStandardItem(QString::fromStdString(bookings[i].m_price));
        model->appendRow({item1, item2, item3, item4, item5, item6,
item7, item8, item9, item10});
    }
}

void MainWindow::fioFind() {
    fioFindWidget->close();
    QString name = QString(inputForm->text());
    testLayout();
    findItem->removeRow(0);
    for(int i = 0; i < bookings.Size(); i++)
    {
        string fio = bookings[i].m_names[0] + " " + bookings[i].m_names[1] +
" " + bookings[i].m_names[2];
        string f = name.toStdString();
        if(f == bookings[i].m_names[0])
        {
            QStandardItem *item1 = new
QStandardItem(QString::fromStdString(fio));
            QStandardItem *item2 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfArrival));
            QStandardItem *item3 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfDeparture));
            QStandardItem *item4 = new
QStandardItem(QString::fromStdString(bookings[i].m_typeOfRoom));
            QStandardItem *item5 = new
QStandardItem(QString::fromStdString(bookings[i].m_numberOfSeats));
            QStandardItem *item6 = new
QStandardItem(QString::fromStdString(bookings[i].m_email));
            QStandardItem *item7 = new
QStandardItem(QString::fromStdString(bookings[i].m_phoneNumber));

```

```

        QStandardItem *item8 = new
QStandardItem(QString::fromStdString(bookings[i].m_comments));
        QStandardItem *item9 = new
QStandardItem(QString::fromStdString(bookings[i].m_paymentMethod));
        QStandardItem *item10 = new
QStandardItem(QString::fromStdString(bookings[i].m_price));
        findItem->appendRow({item1, item2, item3, item4, item5, item6,
item7, item8, item9, item10});
    }
}

void MainWindow::dateFind() {
    dateFindWidget->close();
    QDate date = QDate(inputDateForm->date());
    string dateToFind = to_string(date.day()) + "." + to_string(date.month())
+ "." + to_string(date.year());
    int d, m, y;
    char d1[11]{};
    char d2[11]{};
    strcpy(d1, dateToFind.c_str());
    sscanf(d1, "%d.%d.%d", &d, &m, &y);
    int date1 = y * 365 + m * 31 + d;
    testLayout();
    findItem->removeRow(0);
    for(int i = 0; i < bookings.Size(); i++)
    {
        string d22 = bookings[i].m_dateOfArrival;
        strcpy(d2, d22.c_str());
        sscanf(d2, "%d.%d.%d", &d, &m, &y);
        int date2 = y * 365 + m * 31 + d;
        if(date1 == date2)
        {
            string fio = bookings[i].m_names[0] + " " +
bookings[i].m_names[1] + " " + bookings[i].m_names[2];
            QStandardItem *item1 = new
QStandardItem(QString::fromStdString(fio));
            QStandardItem *item2 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfArrival));
            QStandardItem *item3 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfDeparture));
            QStandardItem *item4 = new
QStandardItem(QString::fromStdString(bookings[i].m_typeOfRoom));
            QStandardItem *item5 = new
QStandardItem(QString::fromStdString(bookings[i].m_numberOfSeats));
            QStandardItem *item6 = new
QStandardItem(QString::fromStdString(bookings[i].m_email));
            QStandardItem *item7 = new
QStandardItem(QString::fromStdString(bookings[i].m_phoneNumber));
            QStandardItem *item8 = new
QStandardItem(QString::fromStdString(bookings[i].m_comments));
            QStandardItem *item9 = new
QStandardItem(QString::fromStdString(bookings[i].m_paymentMethod));
            QStandardItem *item10 = new
QStandardItem(QString::fromStdString(bookings[i].m_price));
            findItem->appendRow({item1, item2, item3, item4, item5, item6,
item7, item8, item9, item10});
        }
    }
}

void MainWindow::typeSearchFunction() {
    typeFindWidget->close();
    QString type = QString(inputTypeForm->currentText());
    testLayout();
}

```

```

        findItem->removeRow(0);
        for(int i = 0; i < bookings.Size(); i++)
        {
            if(type.toStdString() == bookings[i].m_typeOfRoom)
            {
                string fio = bookings[i].m_names[0] + " " +
bookings[i].m_names[1] + " " + bookings[i].m_names[2];
                QStandardItem *item1 = new
QStandardItem(QString::fromStdString(fio));
                QStandardItem *item2 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfArrival));
                QStandardItem *item3 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfDeparture));
                QStandardItem *item4 = new
QStandardItem(QString::fromStdString(bookings[i].m_typeOfRoom));
                QStandardItem *item5 = new
QStandardItem(QString::fromStdString(bookings[i].m_numberOfSeats));
                QStandardItem *item6 = new
QStandardItem(QString::fromStdString(bookings[i].m_email));
                QStandardItem *item7 = new
QStandardItem(QString::fromStdString(bookings[i].m_phoneNumber));
                QStandardItem *item8 = new
QStandardItem(QString::fromStdString(bookings[i].m_comments));
                QStandardItem *item9 = new
QStandardItem(QString::fromStdString(bookings[i].m_paymentMethod));
                QStandardItem *item10 = new
QStandardItem(QString::fromStdString(bookings[i].m_price));
                findItem->appendRow({item1, item2, item3, item4, item5, item6,
item7, item8, item9, item10});
            }
        }
    }

void MainWindow::commentFind() {
    commentFindWidget->close();
    QString comment = QString(inputForm->text());
    testLayout();
    findItem->removeRow(0);
    for(int i = 0; i < bookings.Size(); i++)
    {
        if(comment.toStdString() == bookings[i].m_comments)
        {
            string fio = bookings[i].m_names[0] + " " +
bookings[i].m_names[1] + " " + bookings[i].m_names[2];
            QStandardItem *item1 = new
QStandardItem(QString::fromStdString(fio));
            QStandardItem *item2 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfArrival));
            QStandardItem *item3 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfDeparture));
            QStandardItem *item4 = new
QStandardItem(QString::fromStdString(bookings[i].m_typeOfRoom));
            QStandardItem *item5 = new
QStandardItem(QString::fromStdString(bookings[i].m_numberOfSeats));
            QStandardItem *item6 = new
QStandardItem(QString::fromStdString(bookings[i].m_email));
            QStandardItem *item7 = new
QStandardItem(QString::fromStdString(bookings[i].m_phoneNumber));
            QStandardItem *item8 = new
QStandardItem(QString::fromStdString(bookings[i].m_comments));
            QStandardItem *item9 = new
QStandardItem(QString::fromStdString(bookings[i].m_paymentMethod));
            QStandardItem *item10 = new
QStandardItem(QString::fromStdString(bookings[i].m_price));

```

```

        findItem->appendRow({item1, item2, item3, item4, item5, item6,
item7, item8, item9, item10});
    }
}

void MainWindow::commentSearchLayout() {
    commentFindWidget = new QWidget;
    QFormLayout *inputLayout = new QFormLayout;
    inputForm = new QLineEdit;
    QRegularExpression commentCheck("[a-z-A-Z-A-Я-a-я-0-9.@!?!+]{0,100}");
    QValidator *validator8 = new QRegularExpressionValidator(commentCheck,
this);
    inputForm->setValidator(validator8);
    QPushButton *confirm = new QPushButton(searchIcon, " Найти", this);
    inputLayout->addRow("Введите комментарий:", inputForm);
    inputLayout->addRow("", confirm);
    commentFindWidget->setLayout(inputLayout);
    connect(confirm, SIGNAL(clicked(bool)), this, SLOT(commentFind()));
    commentFindWidget->setWindowTitle("Поиск по комментариям...");
    commentFindWidget->show();
}

void MainWindow::dateSearchLayout() {
    dateFindWidget = new QWidget;
    QFormLayout *inputLayout = new QFormLayout;
    inputDateForm = new QDateEdit;
    inputDateForm->setMinimumDate(QDate::fromString("01.01.2020",
"dd.MM.yyyy"));
    inputDateForm-
>setMaximumDate(QDate::currentDate().addYears(1).addMonths(1));
    QPushButton *confirm = new QPushButton(searchIcon, " Найти", this);
    inputLayout->addRow("Введите дату заезда:", inputDateForm);
    inputLayout->addRow("", confirm);
    dateFindWidget->setLayout(inputLayout);
    connect(confirm, SIGNAL(clicked(bool)), this, SLOT(dateFind()));
    dateFindWidget->setWindowTitle("Поиск по дате заезда...");
    dateFindWidget->show();
}

void MainWindow::fioSearchLayout() {
    fioFindWidget = new QWidget;
    QFormLayout *inputLayout = new QFormLayout;
    inputForm = new QLineEdit;
    QRegularExpression nameCheck("[A-Яa-я]{0,32}");
    QValidator *validator9 = new QRegularExpressionValidator(nameCheck,
this);
    inputForm->setValidator(validator9);
    QPushButton *confirm = new QPushButton(searchIcon, " Найти", this);
    inputLayout->addRow("Введите фамилию:", inputForm);
    inputLayout->addRow("", confirm);
    fioFindWidget->setLayout(inputLayout);
    connect(confirm, SIGNAL(clicked(bool)), this, SLOT(fioFind()));
    fioFindWidget->setWindowTitle("Поиск по фамилии...");
    fioFindWidget->show();
}

void MainWindow::typeSearchLayout() {
    typeFindWidget = new QWidget;
    QFormLayout *inputLayout = new QFormLayout;
    inputTypeForm = new QComboBox;
    QStringList types = {"Стандарт", "Люкс"};
    inputTypeForm->addItem(types);
    QPushButton *confirm = new QPushButton(searchIcon, " Найти", this);
    inputLayout->addRow("Выберите тип номера:", inputTypeForm);

```



```

        inputLayout->addRow("", confirm);
        typeFindWidget->setLayout(inputLayout);
        connect(confirm, SIGNAL(clicked(bool)), this,
        SLOT(typeSearchFunction()));
        typeFindWidget->setWindowTitle("Поиск по типу номера...");
        typeFindWidget->show();
    }

void MainWindow::arrivalDateChange() {
    arrivalDateEditWidget = new QWidget;
    QFormLayout *inputLayout = new QFormLayout;
    bookInd = new QSpinBox;
    bookInd->setRange(1, bookings.Size());
    arrivalDateEditInput = new QDateTimeEdit(QDate::currentDate());
    arrivalDateEditInput->setMinimumDate(QDate::currentDate());
    arrivalDateEditInput->setMaximumDate(QDate::currentDate().addYears(1));
    QPushButton *confirm = new QPushButton(editIcon, " Редактировать", this);
    inputLayout->addRow("Введите номер редактируемой брони:", bookInd);
    inputLayout->addRow("Выберите дату заезда:", arrivalDateEditInput);
    inputLayout->addRow("", confirm);
    arrivalDateEditWidget->setLayout(inputLayout);
    connect(confirm, SIGNAL(clicked(bool)), this,
    SLOT(arrivalDateChangePush()));
    arrivalDateEditWidget->setWindowTitle("Редактирование даты заезда...");
    arrivalDateEditWidget->show();
}

void MainWindow::departureDateChange() {
    departureDateEditWidget = new QWidget;
    QFormLayout *inputLayout = new QFormLayout;
    bookInd = new QSpinBox;
    bookInd->setRange(1, bookings.Size());
    departureDateEditInput = new QDateTimeEdit(QDate::currentDate());
    departureDateEditInput->setMinimumDate(QDate::currentDate());
    departureDateEditInput->setMaximumDate(QDate::currentDate().addDays(1).addYears(1));
    QPushButton *confirm = new QPushButton(editIcon, " Редактировать", this);
    inputLayout->addRow("Введите номер редактируемой брони:", bookInd);
    inputLayout->addRow("Выберите дату выезда:", departureDateEditInput);
    inputLayout->addRow("", confirm);
    departureDateEditWidget->setLayout(inputLayout);
    connect(confirm, SIGNAL(clicked(bool)), this,
    SLOT(departureDateChangePush()));
    departureDateEditWidget->setWindowTitle("Редактирование даты выезда...");
    departureDateEditWidget->show();
}

void MainWindow::phoneChange() {
    phoneEditWidget = new QWidget;
    QFormLayout *inputLayout = new QFormLayout;
    bookInd = new QSpinBox;
    bookInd->setRange(1, bookings.Size());
    phoneEditInput = new QLineEdit;
    phoneEditInput->setInputMask("+7(000)000-00-00");
    QPushButton *confirm = new QPushButton(editIcon, " Редактировать", this);
    inputLayout->addRow("Введите номер редактируемой брони:", bookInd);
    inputLayout->addRow("Введите номер телефона:", phoneEditInput);
    inputLayout->addRow("", confirm);
    phoneEditWidget->setLayout(inputLayout);
    connect(confirm, SIGNAL(clicked(bool)), this, SLOT(phoneChangePush()));
    phoneEditWidget->setWindowTitle("Редактирование номера телефона...");
    phoneEditWidget->show();
}

void MainWindow::emailChange() {

```

```

    emailEditWidget = new QWidget;
    QFormLayout *inputLayout = new QFormLayout;
    bookInd = new QSpinBox;
    bookInd->setRange(1, bookings.Size());
    emailEditInput = new QLineEdit;
    QRegularExpression emailCheck("[a-z0-9.@]{0,30}");
    QValidator *validator7 = new QRegularExpressionValidator(emailCheck,
this);
    emailEditInput->setValidator(validator7);
    QPushButton *confirm = new QPushButton(editIcon, " Редактировать", this);
    inputLayout->addRow("Введите номер редактируемой брони:", bookInd);
    inputLayout->addRow("Введите email:", emailEditInput);
    inputLayout->addRow("", confirm);
    emailEditWidget->setLayout(inputLayout);
    connect(confirm, SIGNAL(clicked(bool)), this, SLOT(emailChangePush()));
    emailEditWidget->setWindowTitle("Редактирование email...");
    emailEditWidget->show();
}

void MainWindow::arrivalDateChangePush() {
    QDate newDate = QDate(arrivalDateEditInput->date());
    int ind = bookInd->value();
    int d, m, y;
    char d1[11]{};
    char d2[11]{};
    string d11 = to_string(newDate.day()) + "." + to_string(newDate.month())
+ "." + to_string(newDate.year());
    string d22 = bookings[ind - 1].m_dateOfDeparture;
    strcpy(d1, d11.c_str());
    strcpy(d2, d22.c_str());
    sscanf(d1, "%d.%d.%d", &d, &m, &y);
    int date1 = y * 365 + m * 31 + d;
    sscanf(d2, "%d.%d.%d", &d, &m, &y);
    int date2 = y * 365 + m * 31 + d;
    if (date1 < date2) {
        if (newDate.day() < 10 || newDate.month() < 10)
        {
            if(newDate.day() < 10 && newDate.month() < 10)
            {
                bookings[ind - 1].m_dateOfArrival = "0" +
to_string(newDate.day()) + "." + "0" + to_string(newDate.month()) + "." +
to_string(newDate.year());
            }
            if(newDate.day() < 10 && newDate.month() >= 10)
            {
                bookings[ind - 1].m_dateOfArrival = "0" +
to_string(newDate.day()) + "." + to_string(newDate.month()) + "." +
to_string(newDate.year());
            }
            if(newDate.day() >= 10 && newDate.month() < 10)
            {
                bookings[ind - 1].m_dateOfArrival = to_string(newDate.day())
+ "." + "0" + to_string(newDate.month()) + "." + to_string(newDate.year());
            }
        }
        else
        {
            bookings[ind - 1].m_dateOfArrival = to_string(newDate.day()) +
"." + to_string(newDate.month()) + "." + to_string(newDate.year());
        }
        if (bookings[ind - 1].m_typeOfRoom == "Стандарт")
        {
            bookings[ind - 1].m_price = to_string((date2 - date1) * 5000);
        }
        else
    }
}

```

```

        {
            bookings[ind - 1].m_price = to_string((date2 - date1) * 10000);
        }
        model->removeRows(0, bookings.Size());
        for (int i = 0; i < bookings.Size(); i++)
        {
            string fio = bookings[i].m_names[0] + " " +
bookings[i].m_names[1] + " " + bookings[i].m_names[2];
            QStandardItem *item1 = new
QStandardItem(QString::fromStdString(fio));
            QStandardItem *item2 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfArrival));
            QStandardItem *item3 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfDeparture));
            QStandardItem *item4 = new
QStandardItem(QString::fromStdString(bookings[i].m_typeOfRoom));
            QStandardItem *item5 = new
QStandardItem(QString::fromStdString(bookings[i].m_numberOfSeats));
            QStandardItem *item6 = new
QStandardItem(QString::fromStdString(bookings[i].m_email));
            QStandardItem *item7 = new
QStandardItem(QString::fromStdString(bookings[i].m_phoneNumber));
            QStandardItem *item8 = new
QStandardItem(QString::fromStdString(bookings[i].m_comments));
            QStandardItem *item9 = new
QStandardItem(QString::fromStdString(bookings[i].m_paymentMethod));
            QStandardItem *item10 = new
QStandardItem(QString::fromStdString(bookings[i].m_price));
            model->appendRow({item1, item2, item3, item4, item5, item6,
item7, item8, item9, item10});
        }
        arrivalDateEditWidget->close();
    }
    else {
        QMessageBox::warning(this, "Неверный ввод",
            "Дата выезда не может быть раньше даты заезда " "<p> Попробуйте еще
раз", "Ок", QString(), QString(), 1, 1);
        arrivalDateEditWidget->close();
    }
}

void MainWindow::departureDateChangePush() {
    QDate newDate = QDate(departureDateEditInput->date());
    int ind = bookInd->value();
    int d, m, y;
    char d1[11]{};
    char d2[11]{};
    string d11 = bookings[ind - 1].m_dateOfArrival;
    string d22 = to_string(newDate.day()) + "." + to_string(newDate.month())
+ "." + to_string(newDate.year());;
    strcpy(d1, d11.c_str());
    strcpy(d2, d22.c_str());
    sscanf(d1, "%d.%d.%d", &d, &m, &y);
    int date1 = y * 365 + m * 31 + d;
    sscanf(d2, "%d.%d.%d", &d, &m, &y);
    int date2 = y * 365 + m * 31 + d;
    if (date1 < date2) {
        if (newDate.day() < 10 || newDate.month() < 10)
        {
            if(newDate.day() < 10 && newDate.month() < 10)
            {
                bookings[ind - 1].m_dateOfDeparture = "0" +
to_string(newDate.day()) + "." + "0" + to_string(newDate.month()) + "." +
to_string(newDate.year());
            }
        }
    }
}

```

```

        if(newDate.day() < 10 && newDate.month() >= 10)
        {
            bookings[ind - 1].m_dateOfDeparture = "0" +
to_string(newDate.day()) + "." + to_string(newDate.month()) + "." +
to_string(newDate.year());
        }
        if(newDate.day() >= 10 && newDate.month() < 10)
        {
            bookings[ind - 1].m_dateOfDeparture =
to_string(newDate.day()) + "." + "0" + to_string(newDate.month()) + "." +
to_string(newDate.year());
        }
    }
    else
    {
        bookings[ind - 1].m_dateOfArrival = to_string(newDate.day()) +
"." + to_string(newDate.month()) + "." + to_string(newDate.year());
    }
    if (bookings[ind - 1].m_typeOfRoom == "Стандарт")
    {
        bookings[ind - 1].m_price = to_string((date2 - date1) * 5000);
    }
    else
    {
        bookings[ind - 1].m_price = to_string((date2 - date1) * 10000);
    }
    model->removeRows(0, bookings.Size());
    for (int i = 0; i < bookings.Size(); i++)
    {
        string fio = bookings[i].m_names[0] + " " +
bookings[i].m_names[1] + " " + bookings[i].m_names[2];
        QStandardItem *item1 = new
QStandardItem(QString::fromStdString(fio));
        QStandardItem *item2 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfArrival));
        QStandardItem *item3 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfDeparture));
        QStandardItem *item4 = new
QStandardItem(QString::fromStdString(bookings[i].m_typeOfRoom));
        QStandardItem *item5 = new
QStandardItem(QString::fromStdString(bookings[i].m_numberOfSeats));
        QStandardItem *item6 = new
QStandardItem(QString::fromStdString(bookings[i].m_email));
        QStandardItem *item7 = new
QStandardItem(QString::fromStdString(bookings[i].m_phoneNumber));
        QStandardItem *item8 = new
QStandardItem(QString::fromStdString(bookings[i].m_comments));
        QStandardItem *item9 = new
QStandardItem(QString::fromStdString(bookings[i].m_paymentMethod));
        QStandardItem *item10 = new
QStandardItem(QString::fromStdString(bookings[i].m_price));
        model->appendRow({item1, item2, item3, item4, item5, item6,
item7, item8, item9, item10});
    }
    departureDateEditWidget->close();
}
else {
    QMessageBox::warning(this, "Неверный ввод",
" Дата выезда не может быть раньше даты заезда " "<p> Попробуйте еще
раз", "Ок", QString(), QString(), 1, 1);
    departureDateEditWidget->close();
}
}
}

```

```

void MainWindow::phoneChangePush() {
    QString newphone = QString(phoneEditInput->text());
    int ind = bookInd->value();
    if (newphone.length() == 16) {
        bookings[ind - 1].m_phoneNumber = newphone.toStdString();
        model->removeRows(0, bookings.Size());
        for (int i = 0; i < bookings.Size(); i++)
        {
            string fio = bookings[i].m_names[0] + " " +
bookings[i].m_names[1] + " " + bookings[i].m_names[2];
            QStandardItem *item1 = new
QStandardItem(QString::fromStdString(fio));
            QStandardItem *item2 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfArrival));
            QStandardItem *item3 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfDeparture));
            QStandardItem *item4 = new
QStandardItem(QString::fromStdString(bookings[i].m_typeOfRoom));
            QStandardItem *item5 = new
QStandardItem(QString::fromStdString(bookings[i].m_numberOfSeats));
            QStandardItem *item6 = new
QStandardItem(QString::fromStdString(bookings[i].m_email));
            QStandardItem *item7 = new
QStandardItem(QString::fromStdString(bookings[i].m_phoneNumber));
            QStandardItem *item8 = new
QStandardItem(QString::fromStdString(bookings[i].m_comments));
            QStandardItem *item9 = new
QStandardItem(QString::fromStdString(bookings[i].m_paymentMethod));
            QStandardItem *item10 = new
QStandardItem(QString::fromStdString(bookings[i].m_price));
            model->appendRow({item1, item2, item3, item4, item5, item6,
item7, item8, item9, item10});
        }
        phoneEditWidget->close();
    }
    else {
        QMessageBox::warning(this, "Неверный ввод",
        "Введенный номер телефона слишком короткий " "<p> Попробуйте еще
раз", "Ок", QString(), QString(), 1, 1);
        phoneEditWidget->close();
    }
}

```

```

void MainWindow::emailChangePush() {
    QString newemail = QString(emailEditInput->text());
    int ind = bookInd->value();
    bookings[ind - 1].m_email = newemail.toStdString();
    model->removeRows(0, bookings.Size());
    for (int i = 0; i < bookings.Size(); i++)
    {
        string fio = bookings[i].m_names[0] + " " + bookings[i].m_names[1] +
" " + bookings[i].m_names[2];
        QStandardItem *item1 = new
QStandardItem(QString::fromStdString(fio));
        QStandardItem *item2 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfArrival));
        QStandardItem *item3 = new
QStandardItem(QString::fromStdString(bookings[i].m_dateOfDeparture));
        QStandardItem *item4 = new
QStandardItem(QString::fromStdString(bookings[i].m_typeOfRoom));
        QStandardItem *item5 = new
QStandardItem(QString::fromStdString(bookings[i].m_numberOfSeats));
        QStandardItem *item6 = new
QStandardItem(QString::fromStdString(bookings[i].m_email));

```

```

        QStandardItem *item7 = new
QStandardItem(QString::fromStdString(bookings[i].m_phoneNumber));
        QStandardItem *item8 = new
QStandardItem(QString::fromStdString(bookings[i].m_comments));
        QStandardItem *item9 = new
QStandardItem(QString::fromStdString(bookings[i].m_paymentMethod));
        QStandardItem *item10 = new
QStandardItem(QString::fromStdString(bookings[i].m_price));
        model->appendRow({item1, item2, item3, item4, item5, item6, item7,
item8, item9, item10});
    }
    emailEditWidget->close();
}

void MainWindow::diagram() {
    QBarSet *set0 = new QBarSet("Стандарт");
    QBarSet *set1 = new QBarSet("Люкс");
    int standartQuantity{};
    int luxQuantity{};
    for(int i = 0; i < bookings.Size(); i++)
    {
        if (bookings[i].m_typeOfRoom == "Стандарт")
        {
            standartQuantity++;
        }
        else {
            luxQuantity++;
        }
    }
    *set0 << standartQuantity;
    *set1 << luxQuantity;
    int maxQ{};
    if (standartQuantity > luxQuantity) {
        maxQ = standartQuantity;
    }
    else {
        maxQ = luxQuantity;
    }
    QBarSeries *series = new QBarSeries();
    series->append(set0);
    series->append(set1);
    QChart *chart = new QChart();
    chart->addSeries(series);
    chart->setTitle("Количество забронированных номеров");
    chart->setAnimationOptions(QChart::SeriesAnimations);
    QString categories;
    categories = "Тип номера";
    QBarCategoryAxis *axisX = new QBarCategoryAxis();
    axisX->append(categories);
    chart->addAxis(axisX, Qt::AlignBottom);
    series->attachAxis(axisX);
    QValueAxis *axisY = new QValueAxis();
    axisY->setRange(0, maxQ + 1);
    axisY->setTickCount(maxQ + 2);
    chart->addAxis(axisY, Qt::AlignLeft);
    series->attachAxis(axisY);
    chart->legend()->setVisible(true);
    chart->legend()->setAlignment(Qt::AlignBottom);
    QChartView *chartView = new QChartView(chart);
    chartView->setRenderHint(QPainter::Antialiasing);
    chartView->resize(600, 800);
    chartView->setWindowTitle("Диаграмма");
    chartView->show();
}

```

```

void MainWindow::createMenuBar() {
    // Setup File Menu
    fileMenu = menuBar()->addMenu("Файлы");
    newAction = new QAction(fileIcon, "Добавить из файла", this);
    openAction = new QAction(saveIcon, "Сохранить в файл", this);
    fileMenu->addAction(newAction);
    fileMenu->addAction(openAction);
    fileMenu->addSeparator();
    sortMenu = menuBar()->addMenu("Сортировка");
    arrivalIncSortAction = new QAction(upIcon, "По возрастанию даты заезда",
this);
    arrivalDecSortAction = new QAction(downIcon, "По убыванию даты заезда",
this);
    departureIncSortAction = new QAction(upIcon, "По возрастанию даты
выезда", this);
    departureDecSortAction = new QAction(downIcon, "По убыванию даты выезда",
this);
    sortMenu->addAction(arrivalIncSortAction);
    sortMenu->addAction(arrivalDecSortAction);
    sortMenu->addAction(departureIncSortAction);
    sortMenu->addAction(departureDecSortAction);
    searchMenu = menuBar()->addMenu("Поиск");
    fioSearch = new QAction(searchIcon, "Поиск по фамилии", this);
    dateSearch = new QAction(searchIcon, "Поиск по дате заезда", this);
    typeSearch = new QAction(searchIcon, "Поиск по типу номера", this);
    commentSearch = new QAction(searchIcon, "Поиск по комментарию", this);
    searchMenu->addAction(fioSearch);
    searchMenu->addAction(dateSearch);
    searchMenu->addAction(typeSearch);
    searchMenu->addAction(commentSearch);
    editMenu = menuBar()->addMenu("Редактирование");
    arrivalDateEdit = new QAction(editIcon, "Редактировать дату заезда",
this);
    departureDateEdit = new QAction(editIcon, "Редактировать дату выезда",
this);
    phoneEdit = new QAction(editIcon, "Редактировать номер телефона", this);
    emailEdit = new QAction(editIcon, "Редактировать email", this);
    editMenu->addAction(arrivalDateEdit);
    editMenu->addAction(departureDateEdit);
    editMenu->addAction(phoneEdit);
    editMenu->addAction(emailEdit);
    connect(newAction, SIGNAL(triggered()), this, SLOT(addDataFromFile()));
    connect(openAction, SIGNAL(triggered()), this, SLOT(addDataToFile()));
    connect(arrivalIncSortAction, SIGNAL(triggered()), this,
SLOT(arrivalIncSort()));
    connect(arrivalDecSortAction, SIGNAL(triggered()), this,
SLOT(arrivalDecSort()));
    connect(departureIncSortAction, SIGNAL(triggered()), this,
SLOT(departureIncSort()));
    connect(departureDecSortAction, SIGNAL(triggered()), this,
SLOT(departureDecSort()));
    connect(fioSearch, SIGNAL(triggered()), this, SLOT(fioSearchLayout()));
    connect(dateSearch, SIGNAL(triggered()), this, SLOT(dateSearchLayout()));
    connect(typeSearch, SIGNAL(triggered()), this, SLOT(typeSearchLayout()));
    connect(commentSearch, SIGNAL(triggered()), this,
SLOT(commentSearchLayout()));
    connect(arrivalDateEdit, SIGNAL(triggered()), this,
SLOT(arrivalDateChange()));
    connect(departureDateEdit, SIGNAL(triggered()), this,
SLOT(departureDateChange()));
    connect(phoneEdit, SIGNAL(triggered()), this, SLOT(phoneChange()));
    connect(emailEdit, SIGNAL(triggered()), this, SLOT(emailChange()));
}

MainWindow::MainWindow(QWidget *parent)

```

```

        : QMainWindow(parent)
        , ui(new Ui::MainWindow)
    {
        ui->setupUi(this);
        setWindowTitle("Приложение отеля");
        resize(1920, 1080);
        QPixmap bkgnd("../OP/back.jpg");
        bkgnd = bkgnd.scaled(this->size(), Qt::IgnoreAspectRatio);
        QPalette palette;
        palette.setBrush(QPalette::Window, bkgnd);
        createIcons();
        setupCoreWidgets();
        createMenuBar();
        connect(newPushButton, SIGNAL(clicked(bool)), this,
            SLOT(addBookingWindow()));
        connect(savePushButton, SIGNAL(clicked(bool)), this,
            SLOT(deleteBookingWidget()));
        connect(graphicPushButton, SIGNAL(clicked(bool)), this, SLOT(diagram()));
        connect(clearPushButton, SIGNAL(clicked(bool)), this,
            SLOT(clearAllRecords()));
        connect(quitPushButton, SIGNAL(clicked(bool)), this, SLOT(exit()));
        centralWidgetLayout->addLayout(formLayout);
        centralWidgetLayout->addWidget(appTable);
        centralWidgetLayout->addLayout(buttonsLayout);
        mainWidget->setLayout(centralWidgetLayout);
        setCentralWidget(mainWidget);
        this->setPalette(palette);
    }

MainWindow::~MainWindow()
{
    delete ui;
}

```


ФайлыСортировкаПоискРедактирование

Приложение отеля

| ФИО | Дата заезда | Дата выезда | Тип номера | Кол-во гостей | Email | Номер телефона | Комментарий | Способ оплаты | Стоимость |
|---------------------------|-------------|-------------|------------|---------------|----------------------|------------------|-------------|---------------|-----------|
| 1 Иванов Иван Иванович | 29.06.2024 | 30.06.2024 | Стандарт | 2 | ivanov@gmail.com | +7(800)335-33-53 | нет | наличные | 5000 |
| 2 Новикова Елена ... | 26.02.2024 | 18.03.2024 | Люкс | 1 | novikova@gmail.com | +7(453)567-36-35 | допВода | карта | 230000 |
| 3 Смирнов Петр ... | 01.01.2024 | 5.01.2024 | Стандарт | 2 | smirnov@gmail.com | +7(567)849-37-56 | нет | наличные | 20000 |
| 4 Кузнецов Степан Ильич | 06.01.2024 | 14.01.2024 | Стандарт | 2 | kuznetsov@gmail.com | +7(567)239-45-78 | нет | наличные | 40000 |
| 5 Лопова Кристина ... | 07.02.2024 | 20.02.2024 | Люкс | 1 | ropova@gmail.com | +7(368)563-80-95 | допВода | карта | 130000 |
| 6 Васильева Екатерина ... | 21.02.2024 | 05.03.2024 | Люкс | 1 | vasilieva@gmail.com | +7(678)356-89-53 | допВода | карта | 150000 |
| 7 Петров Андрей ... | 07.03.2024 | 10.03.2024 | Люкс | 1 | petrov@gmail.com | +7(356)748-57-36 | допВода | карта | 30000 |
| 8 Попов Иван Андреевич | 15.01.2024 | 31.01.2024 | Стандарт | 2 | popov@gmail.com | +7(678)356-84-79 | нет | наличные | 80000 |
| 9 Михайлова Елена ... | 12.03.2024 | 22.03.2024 | Люкс | 1 | mikhailova@gmail.com | +7(678)356-78-39 | нет | карта | 100000 |
| 10 Лебедев Максим ... | 03.02.2024 | 16.02.2024 | Стандарт | 2 | lebedev@gmail.com | +7(123)456-78-90 | нет | наличные | 65000 |
| 11 Павлова Мария ... | 17.02.2024 | 20.02.2024 | Стандарт | 2 | pavlova@gmail.com | +7(098)765-43-21 | нет | наличные | 15000 |
| 12 Морозов Александр ... | 26.03.2024 | 31.03.2024 | Люкс | 1 | morozov@gmail.com | +7(357)356-26-89 | допВода | карта | 50000 |
| 13 Волков Николай ... | 01.04.2024 | 18.04.2024 | Люкс | 1 | volkov@gmail.com | +7(926)345-79-61 | жив. | карта | 170000 |
| 14 Федотова Елена ... | 19.04.2024 | 23.04.2024 | Люкс | 1 | fedotova@gmail.com | +7(678)274-67-89 | допВода | карта | 40000 |
| 15 Крылова Анастасия ... | 26.02.2024 | 18.03.2024 | Стандарт | 1 | kriova@gmail.com | +7(256)746-79-37 | допВода | карта | 115000 |

+Добавить бронь

✕Удалить бронь

Дататрафика

Удалить все

Выйти

Рис. 1 Вывод таблицы

Приложение отеля

ФайлыСортировкаПоискРедактирование

| ФИО | Дата заезда | Дата выезда | Тип номера | Кол-во гостей | Email | Номер телефона | Комментарий | Способ оплаты | Стоимость |
|-----|-------------|-------------|------------|---------------|-------|----------------|-------------|---------------|-----------|
|-----|-------------|-------------|------------|---------------|-------|----------------|-------------|---------------|-----------|

Добавление брони...

Фамилия:

Имя:

Отчество:

Email:

Номер телефона: +7() - -

Комментарий:

Тип номера:

Дата заезда:

Дата выезда:

Количество гостей:

Способ оплаты:

Онлайн

+ Добавить

+ Добавить бронь

✕ Удалить бронь

📊 Диаграмма

🗑 Удалить все

🚪 Выйти

Рис. 2 Форма добавления



Рис. 3 Форма удаления

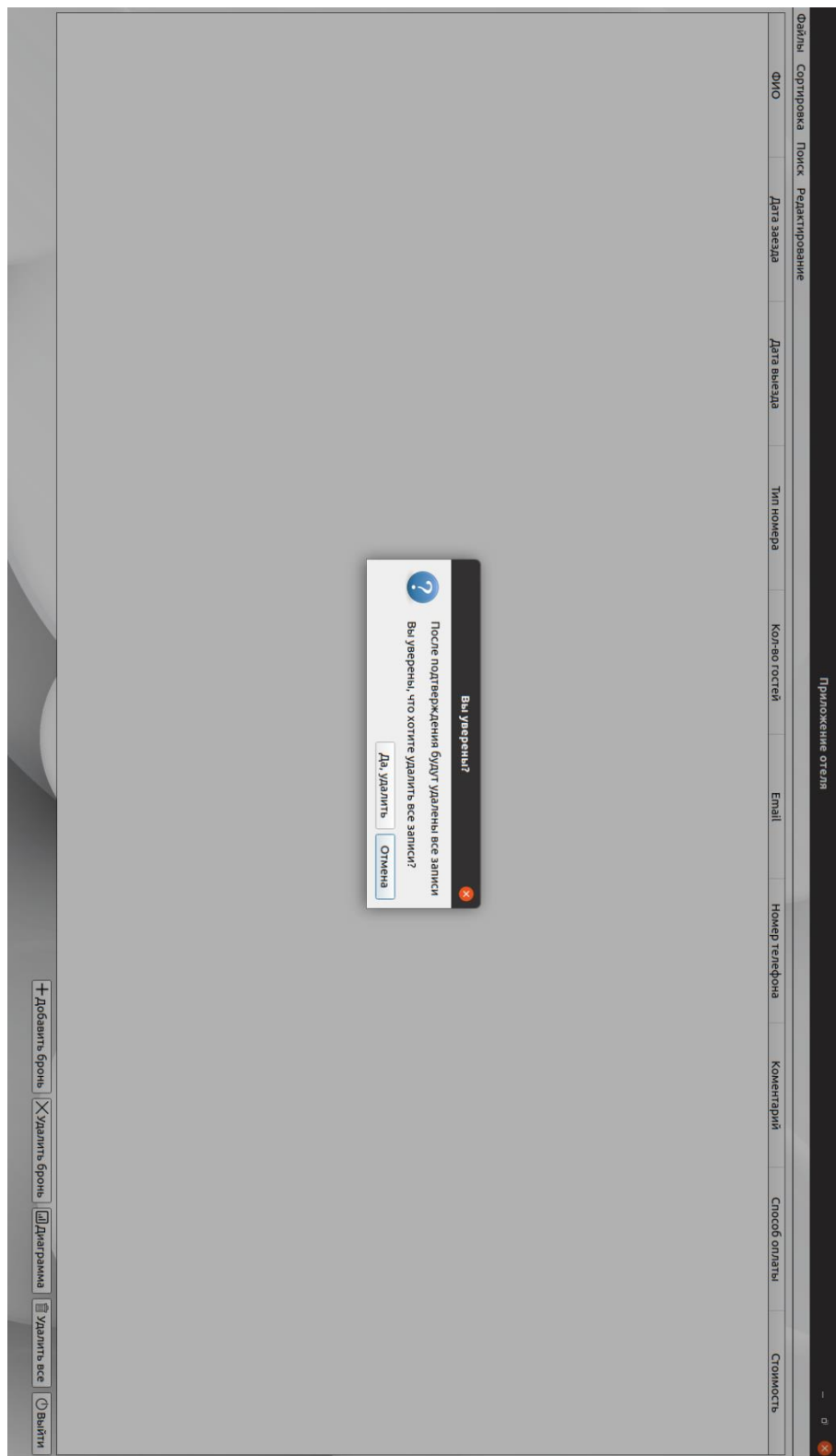


Рис. 4 Форма удаления всех записей



Рис. 5 Форма редактирования

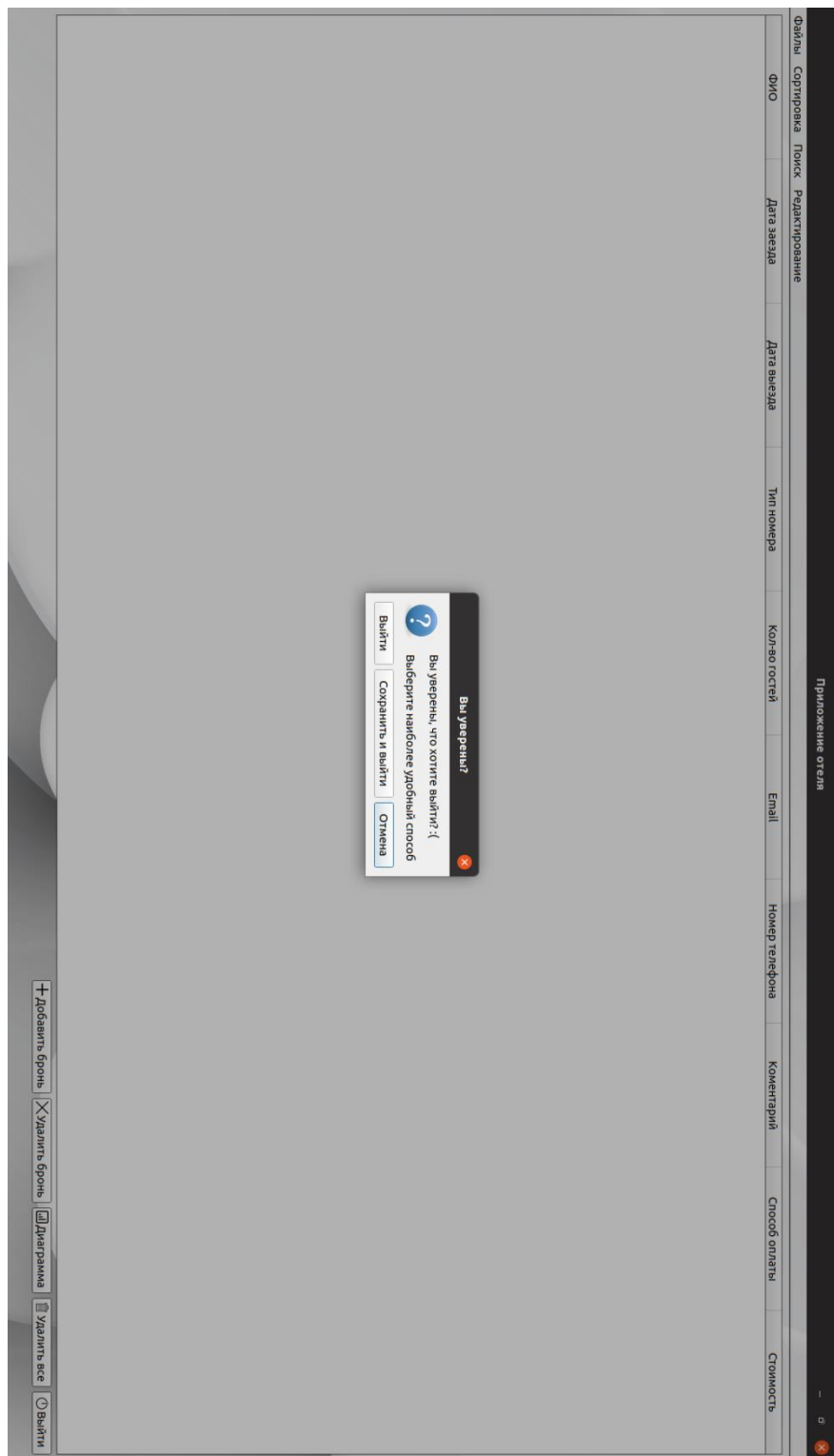
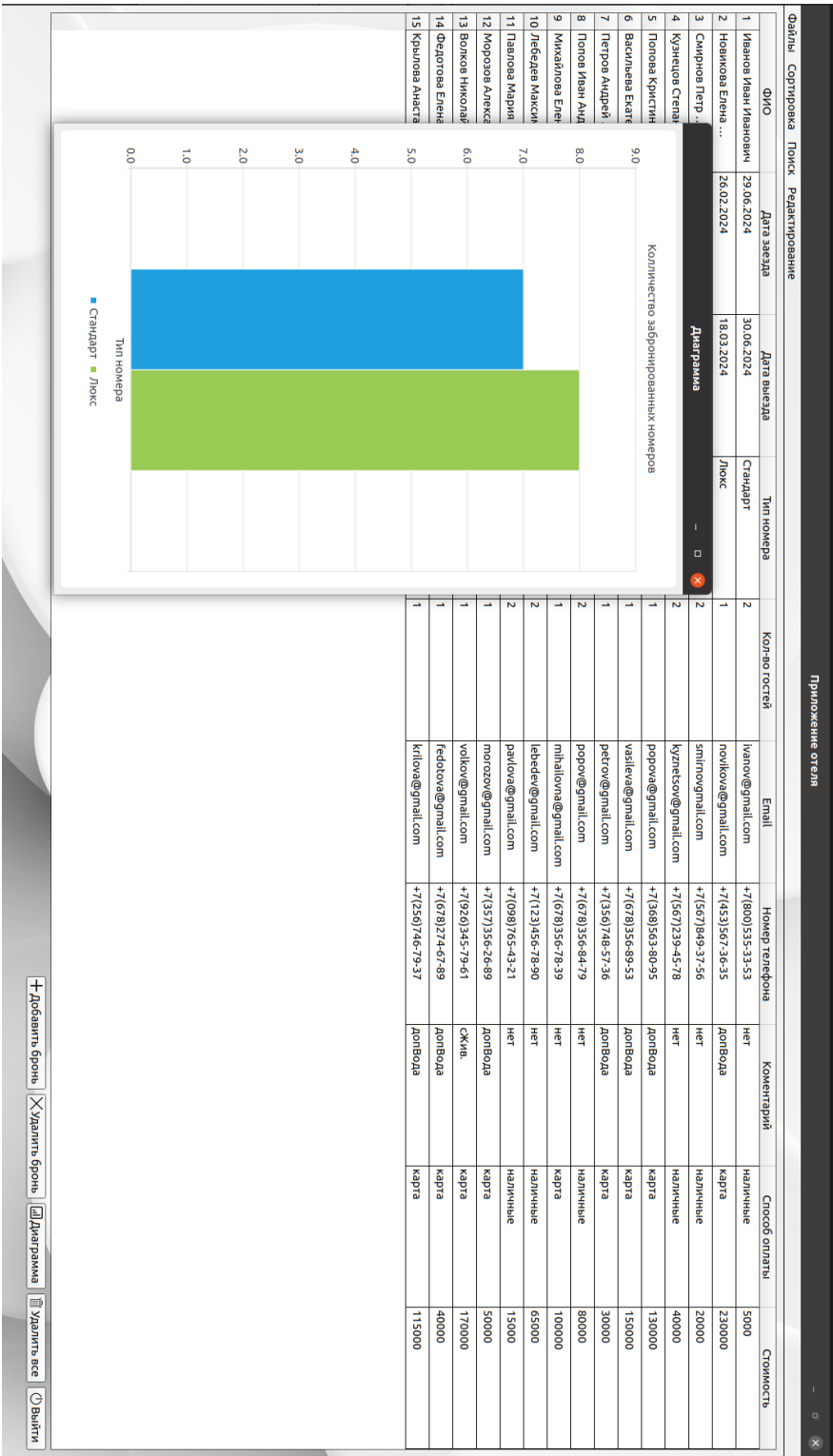


Рис. 6 Форма выхода



ФайлыСортировкаПоискРедактирование

Приложение отеля

ФИОДата заездаДата выездаТип номераКол-во гостейEmailНомер телефонаКомментарийСпособ оплатыСтоимость

Поиск по фамилии...

Введите фамилию:

Найти

+ Добавить бронь

✕ Удалить бронь

📄 Диаграмма

🗑 Удалить все

🚪 Выйти

Рис. 8 Форма поиска

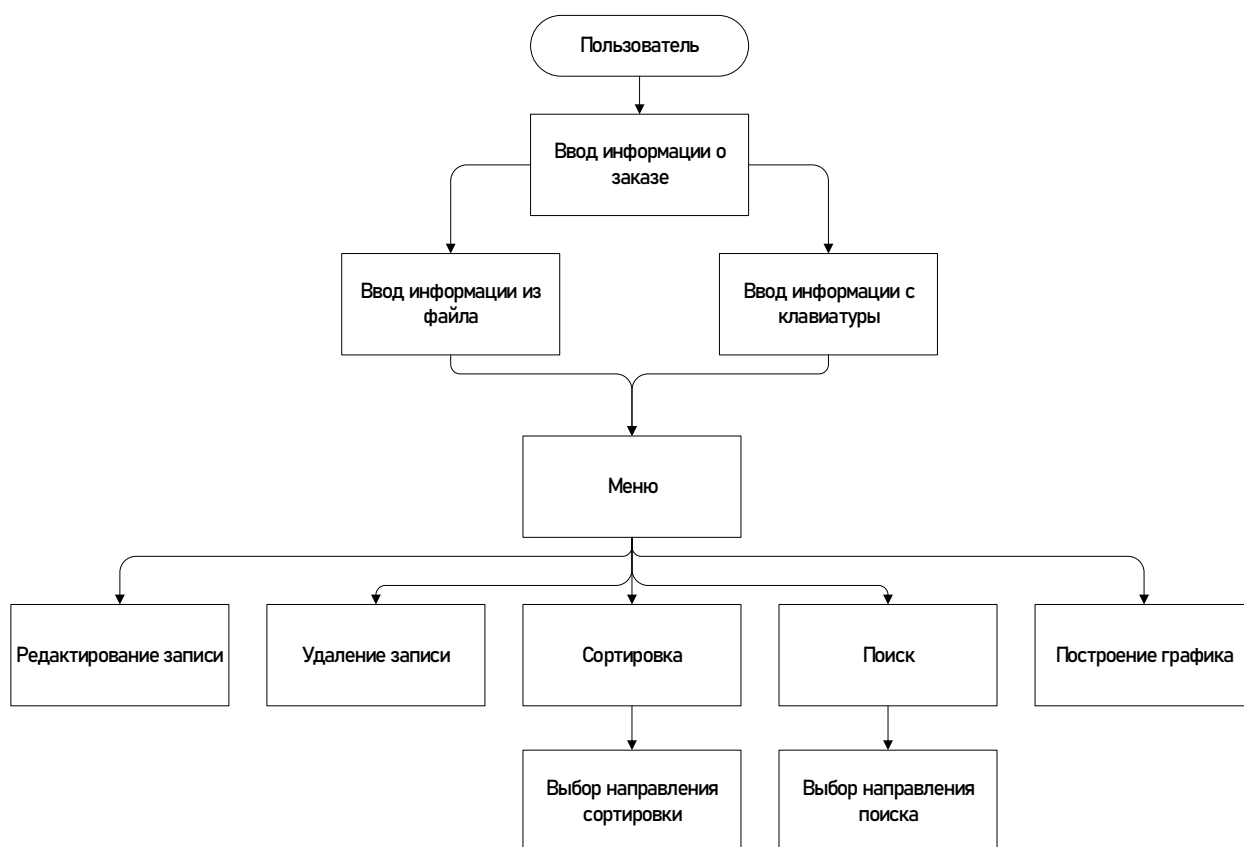


Рис. 1 Диаграмма действий приложения

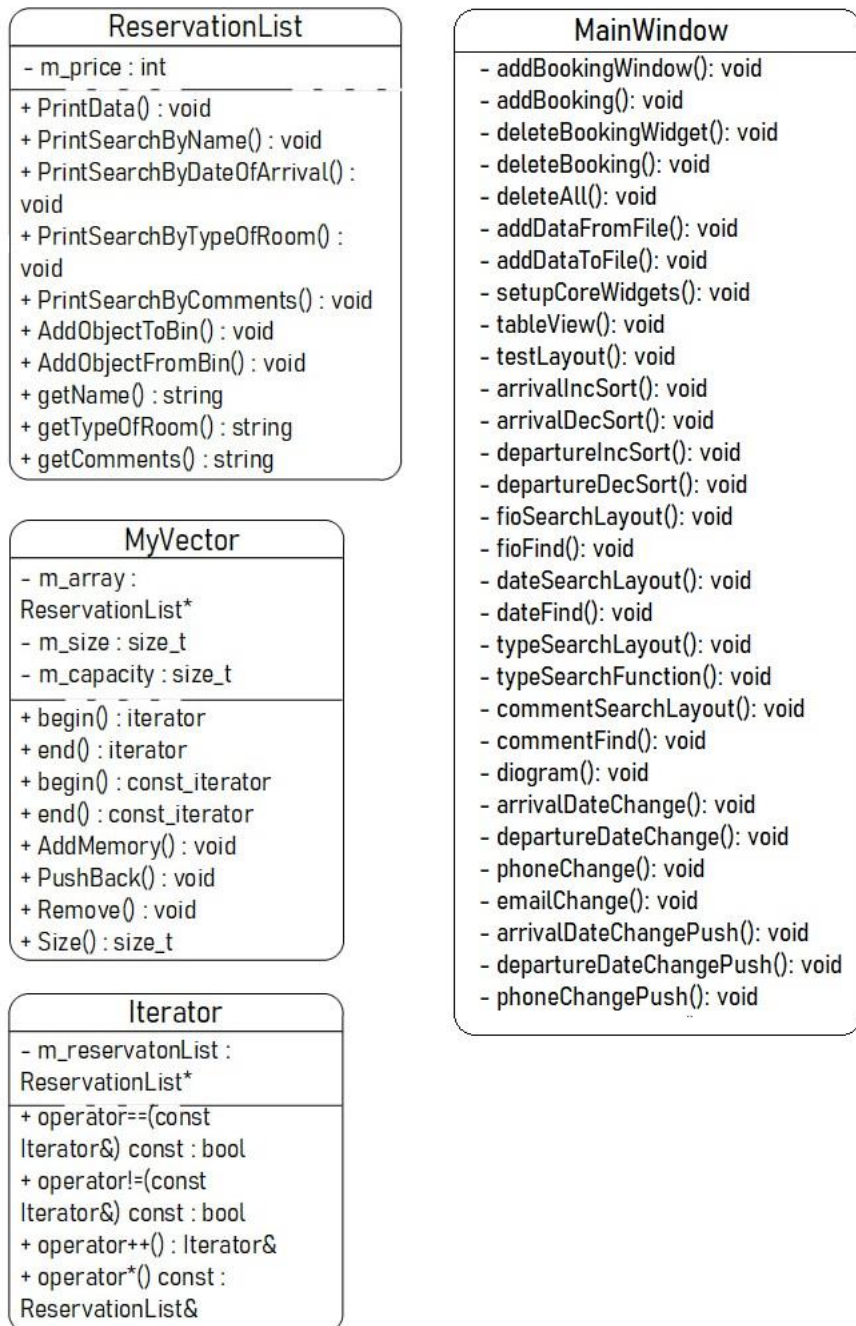


Рис.2 Диаграмма классов