

**Министерство образования Российской Федерации**

**Пензенский государственный университет**

**Кафедра «Математическое обеспечение и применение ЭВМ»**

Пояснительная записка к курсовому проекту

по дисциплине

«Объектно-ориентированное программирование».

Разработка программы с использованием объектно-ориентированного  
подхода.

ИС «Касса аэрофлота»

Автор работы:

студенты группы 23ВП2

Серегин К.С.

Принял:

к.т.н. Афонин А.Ю.

Работа сдана \_\_\_\_\_

Оценка \_\_\_\_\_

Пенза 2025

## Листы с заданием на курсовое проектирование

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ  
Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
«Пензенский государственный университет»  
(ФГБОУ ВО «Пензенский государственный университет»)

Кафедра «Математическое обеспечение и применение ЭВМ»

«УТВЕРЖДАЮ»

Зав. кафедрой  Козлов А.Ю.

"\_\_" \_\_\_\_ 2025 г.

**ЗАДАНИЕ**  
на курсовое проектирование по дисциплине  
«Объектно-ориентированное программирование»

Студенту Сергину Кирилл Сергеевичу Группа 23ВП2

**Тема проекта:** Разработка программы с использованием объектно-ориентированного подхода. ИС «Касса аэрофлота».

**Исходные данные (технические требования) на проектирование**

1. Разработать программу БД «Касса аэрофлота».
2. Данные по базе включают: а) Название авиакомпании; б) Стоимость; в) Номер места; г) Номер ряда; д) Номер рейса; е) Номер самолёта; ж) Класс полёта.
3. Программа должна выполнять следующие действия: создание базы данных, удаление базы данных, сохранение текущей базы данных в файл, добавление записей, удаление записей, редактирование записей, поиск и сортировка записей, фильтрация записей по определенному критерию.
4. Обязательные требования к программе: многомодульность, использование сложных типов данных, использование коллекции для организации базы данных. Старт приложения должен сопровождаться всплывающим окном с информацией об авторе и темой проекта.
5. Визуальный интерфейс приложения реализовать с использованием графических библиотек.
6. Среда разработки ПО: Microsoft Visual Studio 2019/2022, VSCode
7. Язык программирования: C#
8. Программное обеспечение должно быть полностью отлажено и протестировано, и должно функционировать под управлением ОС Windows 10 и выше.

## Объём работ по курсу

### 1. Расчётная часть

- 1) Анализ требований.
- 2) Выбор и освоение инструментальных средств анализа и проектирования
- 3) Определение структуры и функций приложения
- 4) Разработка диаграмм описания приложения
- 5) Реализация приложения на языке C# в среде Microsoft Visual Studio
- 6) Отладка и тестирование приложения

### 2. Графическая часть

- 1) Схема данных
- 2) Диаграмма вариантов использования
- 3) Диаграмма классов

### 3. Экспериментальная часть

- 1) Отладка компонентов приложения и их взаимодействия
- 2) Функциональное тестирование приложения


## Срок выполнения проекта по разделам

1	Анализ требований	к	18 февраля	2025 г.
2	Определение структуры и функций приложения	к	1 марта	2025 г.
3	Разработка UML диаграмм приложения	к	15 марта	2025 г.
4	Реализация приложения	к	12 апреля	2025 г.
5	Отладка и тестирование приложения	к	29 апреля	2025 г.
6	Оформление пояснительной записки проекта	к	17 мая	2025 г.
7	Защита курсовой работы	к	24 мая	2025 г.


Дата выдачи задания « 4 » февраля 2025 г.  
Дата защиты проекта « 5 » апреля 2025 г.

Руководитель  /к.т.н. А.Ю. Афонин/

Задание получил « 4 » февраля 2025 г.

Студент  /К.С. Серегин/

05.04.25



## Содержание

Содержание.....	3
Введение.....	4
1 Постановка задачи .....	5
2 Выбор решения.....	6
2.1 Определение необходимых модулей программы.....	6
2.2 Определение структуры файла базы данных.....	7
3 Описание разработки программы .....	9
3.1 Кодирование .....	9
3.2 Диаграмма компонентов.....	10
4 Отладка и тестирование .....	11
4.1. Функциональное тестирование .....	11
4.2. Тестирование приложения .....	11
5 Описание программы.....	18
5.1 Разработка приложения CourseWork_OOP_Seregin.exe .....	18
Выводы.....	20
Список использованных источников .....	22
Приложение А – Код программы .....	24
Приложение В – Руководство пользователя .....	36

## **Введение**

База данных представляет собой организованный набор данных, хранящийся в цифровом формате в компьютерной системе. Она предназначена для эффективного хранения, обработки и извлечения информации, что делает её важным инструментом в различных сферах. Использование баз данных позволяет систематизировать большие объёмы информации и облегчает её обработку [1].

В современных базах данных информация обычно структурирована в виде таблиц, состоящих из строк и столбцов. Такая организация данных позволяет эффективно управлять ими, выполнять модификации, обновления и контролировать доступ.

Процесс разработки базы данных включает проектирование структуры данных, определение их взаимосвязей и методов доступа к информации. Основная цель создания базы данных — обеспечение удобного, надёжного и быстрого хранения данных, соответствующего потребностям организации.

При проектировании базы данных важно учитывать такие аспекты, как создание оптимизация запросов, масштабируемость и производительность системы. Также необходимо анализировать требования программного обеспечения, чтобы база данных обеспечивала стабильную работу всей системы [2].

Грамотно спроектированная база данных способствует получению актуальной и достоверной информации, улучшает эффективность работы и повышает надёжность системы. В частности, разработка базы данных для касс аэрофлот играет ключевую роль в управлении операциями, повышая доступность данных и способствуя улучшению качества обслуживания клиентов и оптимизации внутренних процессов.

## 1 Постановка задачи

Создание программы «Касса Аэрофлота». Необходимые данные включают:

- номер рейса,
- номер самолёта,
- номер места,
- номер ряда,
- класс полёта,
- стоимость,
- название авиакомпании.

Для реализации необходимо определить структуру базы данных, предусмотреть надёжную валидацию данных, вводимых пользователем, чтобы исключить ошибки при добавлении или редактировании записей (например, проверка формата номера рейса, допустимых значений класса полета и стоимости билета). Это обеспечит корректную работу всей базы данных и повысит устойчивость приложения к некорректным действиям пользователя.

Приложение должно поддерживать добавление, удаление и редактирование записей, сохранения, загрузки базы данных, а также обеспечивать возможность сортировки, поиска и фильтрации списков по заданным критериям. Помимо этого, должна быть реализована функция удаления списков.

Для организации базы данных необходимо использовать многомодульную структуру, сложные типы данных и коллекции. При запуске приложения должно отображаться окно с информацией об авторе и теме проекта.

Важно создать интуитивно понятный интерфейс с чётко обозначенными пунктами меню, определить его тип и распределить зоны для отображения меню и таблицы с данными в консоли. Также следует изучить механизмы работы с файлами, чтобы полноценно реализовать функциональность программы [3].

Основная цель разработки — создание удобного и функционального интерфейса, обеспечивающего простое и эффективное выполнение ключевых задач программы.

## 2 Выбор решения

Для решения поставленной задачи будет использоваться 2 основных модуля программы: модуль, ответственный за пользовательский интерфейс, модуль ответственный за работу с БД. Данные модули приведены на диаграмме классов (рисунок 1). На данной диаграмме отражены основные взаимодействия классов программы между собой.

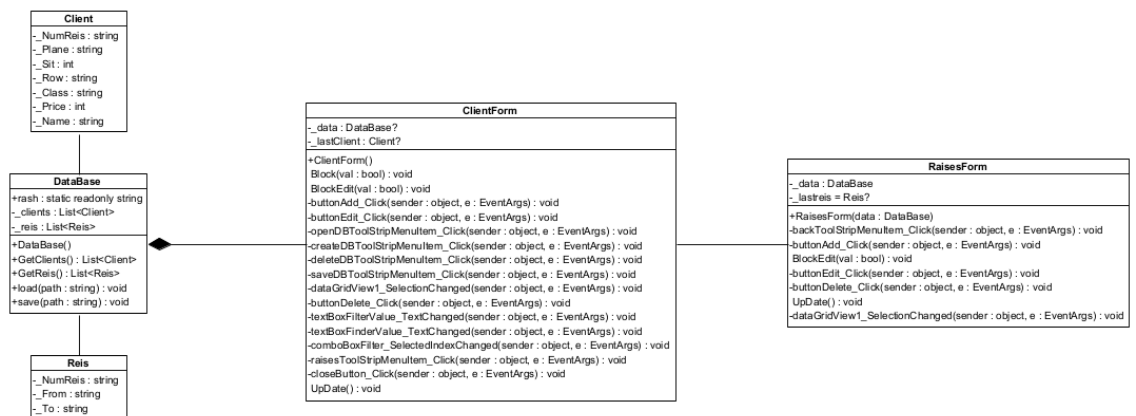


Рисунок 1 – Диаграмма классов

### 2.1 Определение необходимых модулей программы

Модуль для работы с коллекциями данных предоставляет методы для сохранения коллекций объектов в файл и загрузки информации из сохраненного файла. Включает методы load для загрузки данных из файла в формате JSON и save для записи текущих данных в файл. Кроме того, можно добавить, что данный класс использует сериализацию и десериализацию JSON через JsonSerializer для обработки данных. Код данного модуля приведен ниже [4].

```
public class DataBase
{
    public static readonly string rash = ".DataBase";
    private List<Client> _clients;
    public List<Client> GetClients() { return _clients; }
    private List<Reis> _reis;
    public List<Reis> GetReis() { return _reis; }
    public DataBase()
    public void load(string path)
    public void save(string path)
}
```

В программе присутствует модуль для реализации взаимодействия пользователя и коллекции. Для этого был реализован удобный пользовательский

интерфейс, в котором реализованы основные операции. Пример реализации данного модуля приведен ниже.

```
public partial class ClientForm : Form
{
    DataBase? _data;
    Client? _lastClient;
    public ClientForm()
    void Block(bool val)
    void BlockEdit(bool val)
    private void buttonAdd_Click(object sender, EventArgs e)
    private void buttonEdit_Click(object sender, EventArgs e)
    private void openDBToolStripMenuItem_Click(object sender, EventArgs e)
    private void createDBToolStripMenuItem_Click(object sender, EventArgs e)
    private void deleteDBToolStripMenuItem_Click(object sender, EventArgs e)
    private void saveDBToolStripMenuItem_Click(object sender, EventArgs e)
    private void dataGridView1_SelectionChanged(object sender, EventArgs e)
    private void buttonDelete_Click(object sender, EventArgs e)
    private void textBoxFilterValue_TextChanged(object sender, EventArgs e)
    private void textBoxFinderValue_TextChanged(object sender, EventArgs e)
    private void raisesToolStripMenuItem_Click(object sender, EventArgs e)
    private void closeButton_Click(object sender, EventArgs e)
    private void ClientForm_Load(object sender, EventArgs e)
    void Update()
    private void textBoxAddRow_TextChanged(object sender, EventArgs e)
}
```

Модель интерфейса программы представлена на рисунке 2.

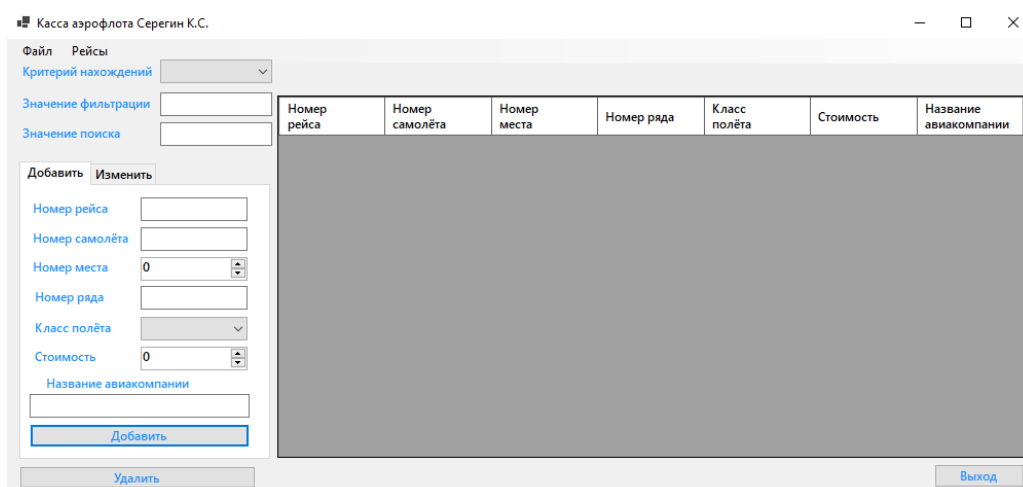


Рисунок 2 – Модель пользовательского интерфейса

Пользователь имеет возможность создать новую коллекцию и удалить уже существующую. Есть возможность изменить существующее поле и вывести информацию всей коллекции после данного действия.

## 2.2 Определение структуры файла базы данных

В качестве средства для хранения данных выбрана JSON сериализация в файл. Структура файла базы данных основывается на двух основных



коллекциях: коллекции объектов Client (клиенты) и коллекции объектов Reis (рейсы). Каждый объект Client содержит информацию о клиенте, включая номер рейса, тип самолета, место в салоне, класс обслуживания, цену билета и имя пассажира. В свою очередь, объект Reis хранит информацию о рейсе, включая его номер, пункт отправления и пункт назначения.

Ниже, на рисунке 3, приведена модель базы данных, используемая в курсовой работе.

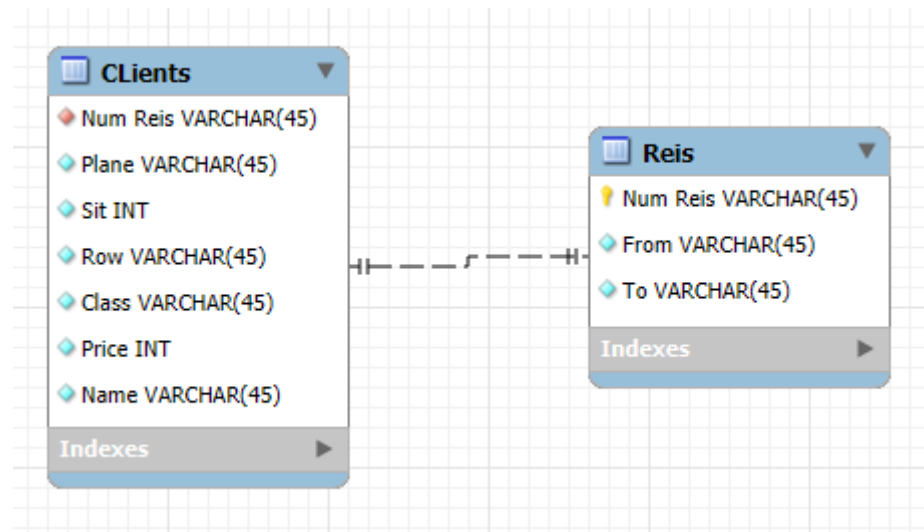


Рисунок 3 – Модель базы данных

### 3 Описание разработки программы

#### 3.1 Кодирование

В ходе выполнения курсовой работы было разработана программа «Касса аэрофлота», код которого приведен в приложении А.

Для алгоритма работы открытия БД была разработана диаграмма деятельности (рисунок 4) [5].

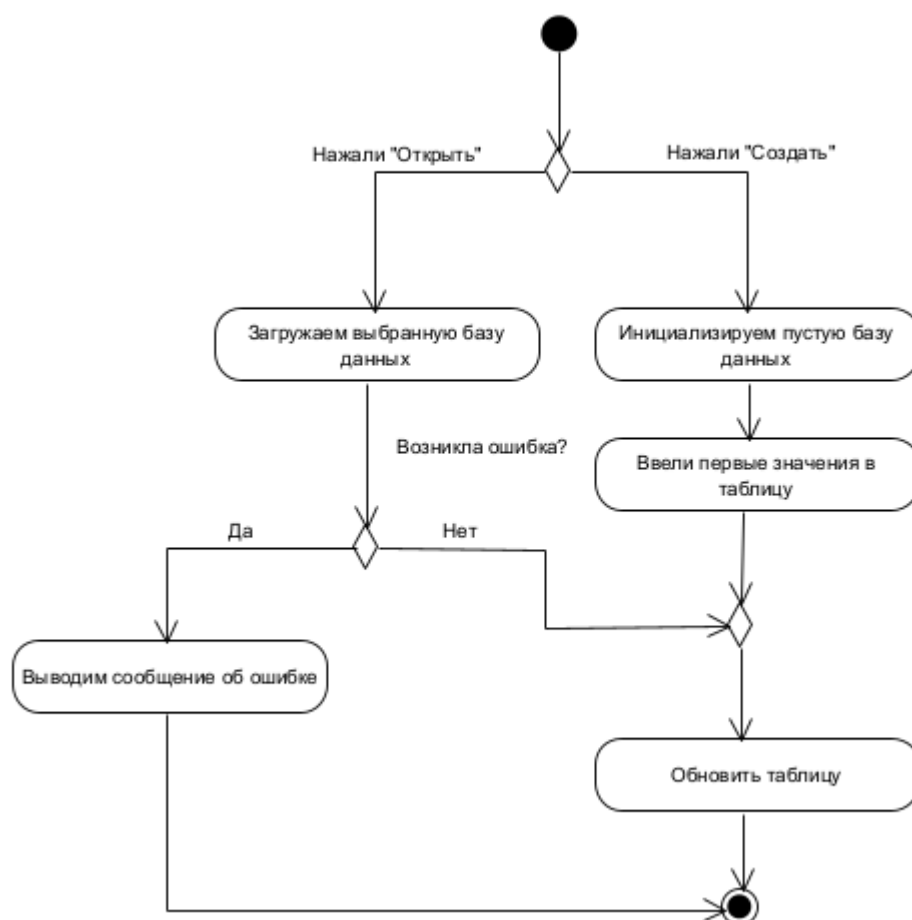


Рисунок 4 – Диаграмма деятельности

На диаграмме можно увидеть, что в начале программа смотрит что выбрал пользователь: “Открыть” – для выбора существующей базы данных или “Создать” – для создания новой базы данных. В первом случае программа проверяет на корректность файл, если все хорошо, то обновляем таблицу, в противном случае выводим сообщение об ошибке. Во втором случае программа

инициализирует пустую базу данных, в которую пользователь должен ввести данные для последующей работы с ней.

### 3.2 Диаграмма компонентов

В процессе выполнения курсового проекта была составлена диаграмма компонентов, которая отображает разбиение программной системы на структурные компоненты и связи между ними (рисунок 5)

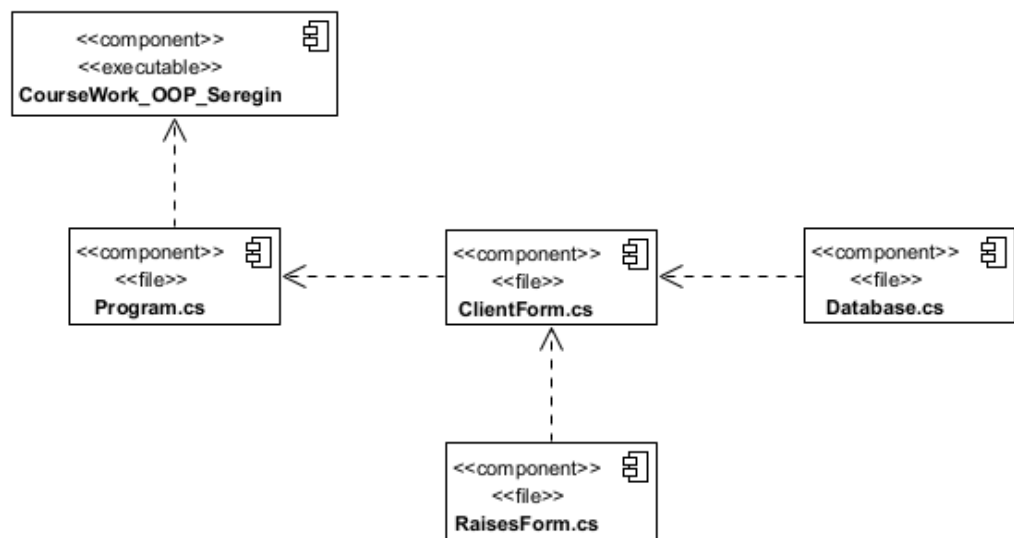


Рисунок 5 – Диаграмма компонентов

Описание компонентов приведено в таблице 1.

Таблица 1 – Компоненты

Компоненты	Назначение
Program.cs	Главная программа
ClientForm.cs	Файл с формой для добавления и редактирования клиентов
DataBase.cs	Файл с классом базы данных
RaisesForm.cs	Файл с формой для добавления и редактирования рейсов

## 4 Отладка и тестирование

### 4.1. Функциональное тестирование

Функциональное тестирование – это процесс проверки программного обеспечения на соответствие требованиям и спецификации продукта. Оно сосредоточено на внешнем поведении системы, не принимая во внимание её внутреннюю структуру. В ходе функционального тестирования анализируется правильность и полнота реализации определённых функций, а также соответствие продукта запросам клиента или пользователей. [6]

Каждая функция системы тестируется путём предоставления соответствующих входных данных, анализа выходных данных и сопоставления реальных результатов с ожидаемыми. Тестирование может осуществляться как вручную, так и с использованием автоматизированных инструментов.

### 4.2. Тестирование приложения

В курсовом проекте было выполнено функциональное тестирование разработанного программного обеспечения. Результаты тестирования приведены в таблице 2.

Таблица 2 – Функциональное тестирование

Состав теста	Ожидаемый результат	Наблюдаемый результат
Создать базу данных	Создание новой базы данных и открытие взаимодействия с ней	Создана новая база данных и открыто взаимодействие с ней (рисунок 6)
Добавить клиента с полями, имеющую пустое поле	Появление окна с ошибкой и информацией о ней	Появилось окно с ошибкой и информацией о ней. (рисунок 7)

Продолжение таблицы 2

Добавить клиента с корректными данными	Появление в БД созданного клиента	Появился в БД созданного клиента (рисунок 8)
Переименовать кампанию у клиента	Изменение информации о клиенте	Изменилась информация в таблице(рисунок 9)
Сохранить БД в файл	Появление файла сохранения	Появился файл сохранения (рисунок 10)
Удалить текущую БД	Стирается таблица	Таблица стёрлась (рисунок 11)
Открыть БД	Появление сохранённых ранее данных	Появились данные (рисунок 12)
Поиск по критерию в БД	Подсветка жёлтым цветом соответствующих данных и вывод количества подходящих данных	Подсветились данные вывод количества (рисунок 13)
Фильтрация по критерию в БД	Стираются неподходящие данные и вывод количества подходящих данных	Стерлись неподходящие данные и вывод количества (рисунок 14)

Ниже, на рисунках 6-14, приведены скриншоты, отражающие результаты работы программы в процессе тестирования.



Рисунок 6 – Тест “Создать базу данных”

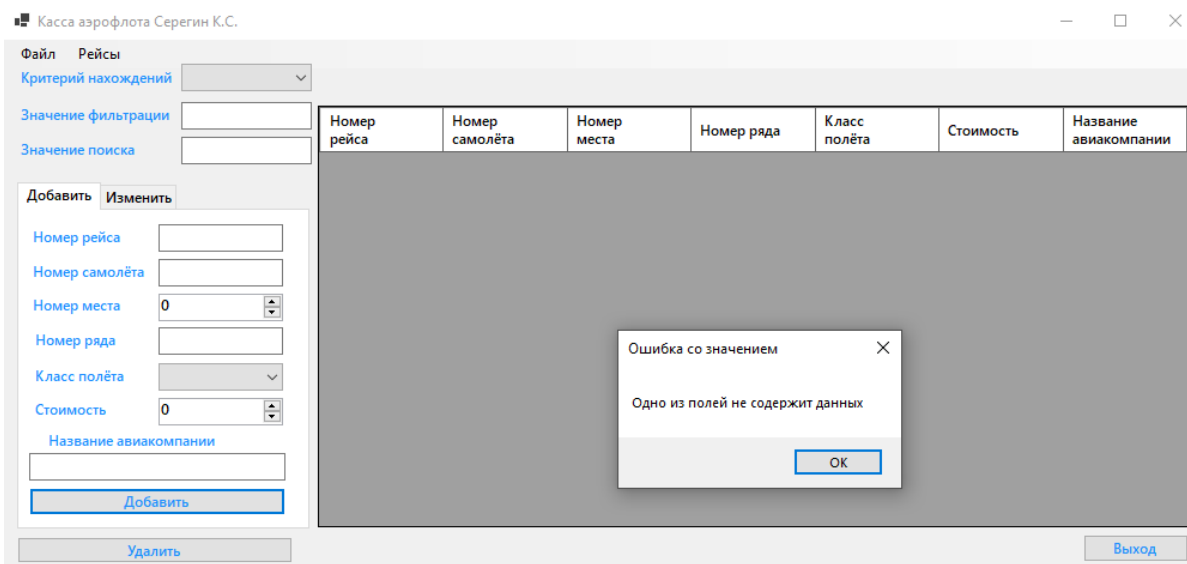


Рисунок 7 – Тест “Добавить клиента с полями, имеющую пустое поле”

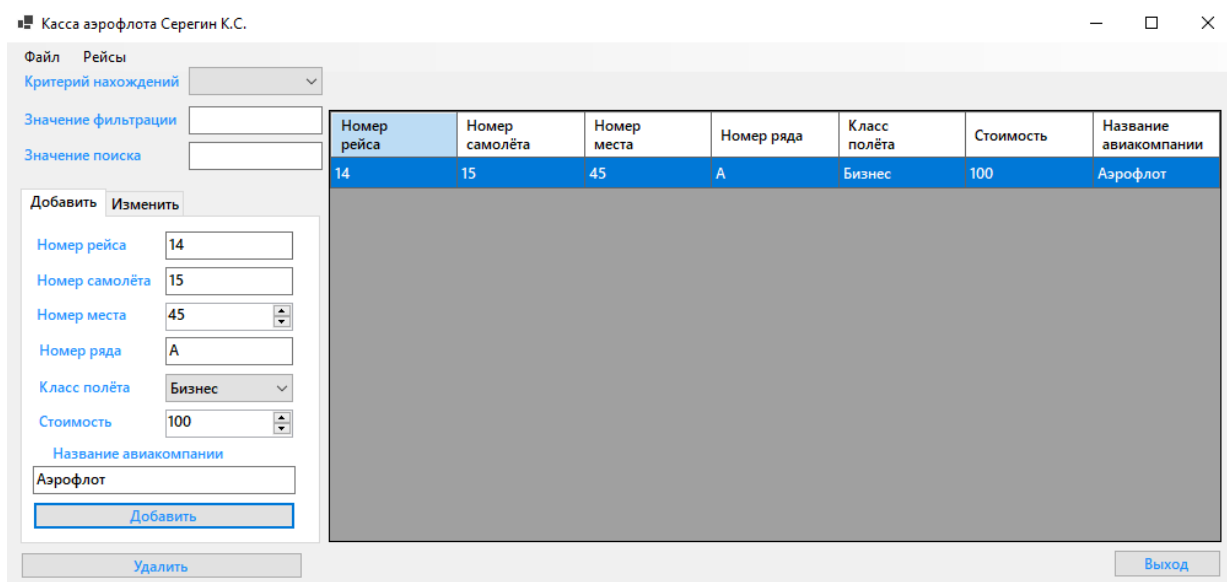


Рисунок 8 – Тест “Добавить клиента с корректными данными”

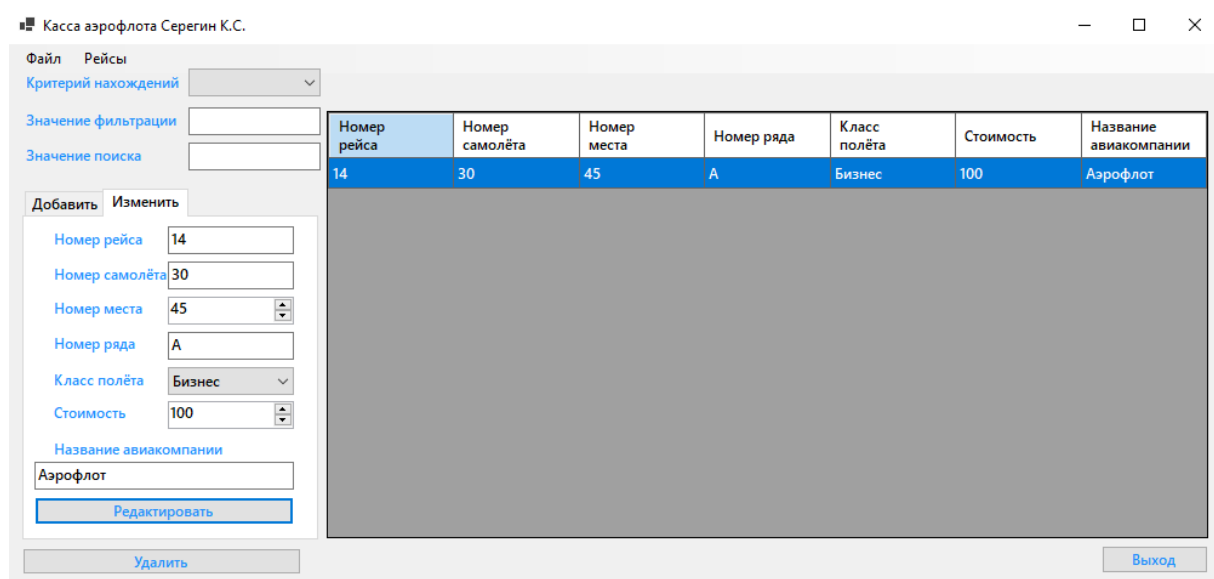


Рисунок 9 – Тест “Переименовать кампанию у клиента”

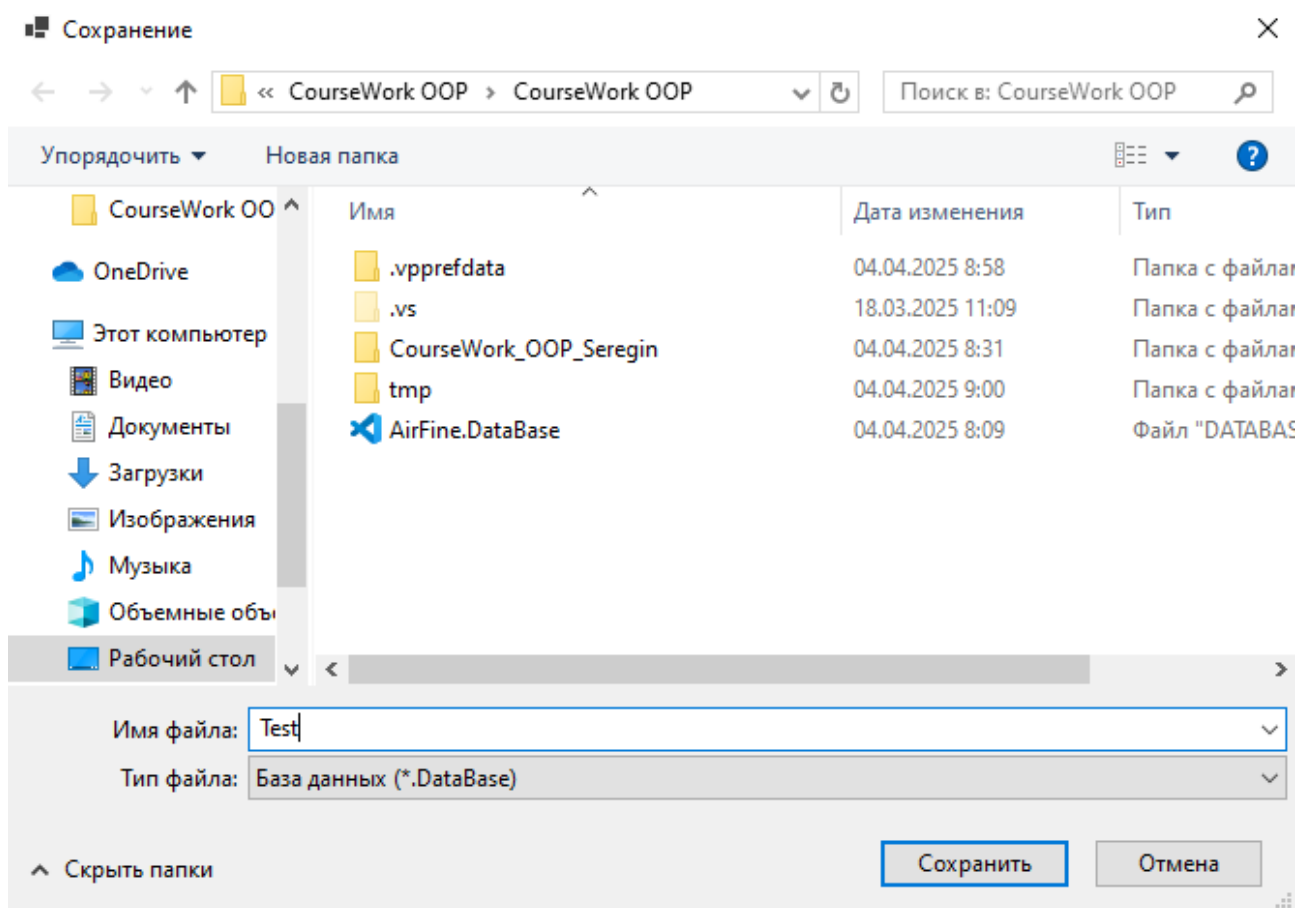


Рисунок 10 – Тест “Сохранить БД в файл”

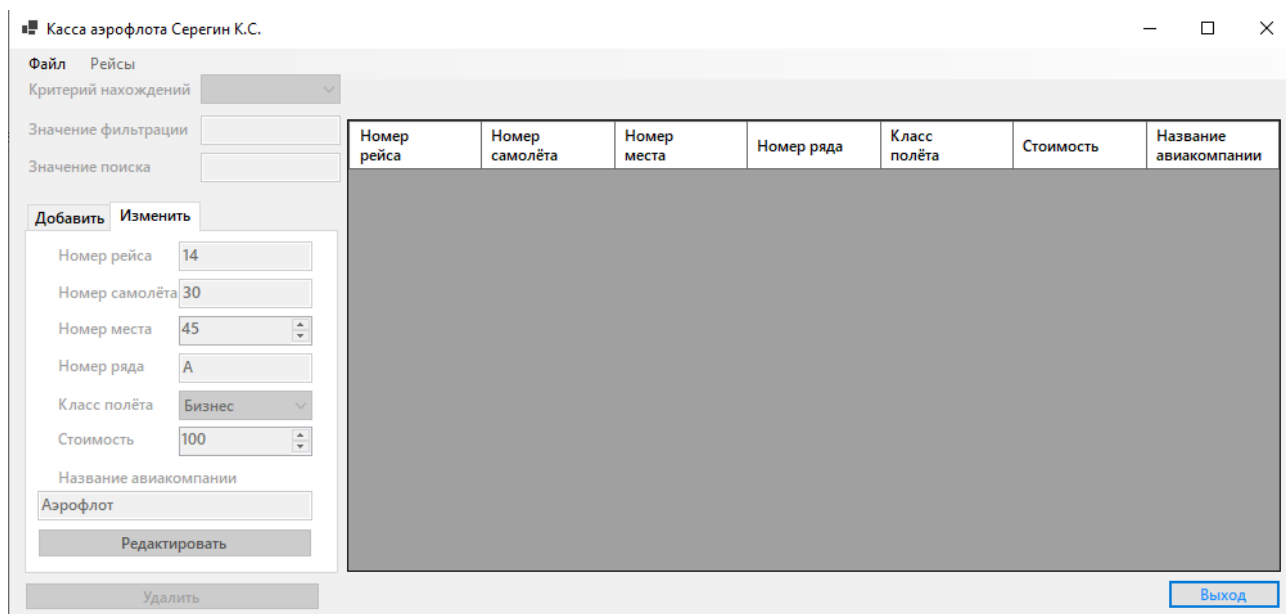


Рисунок 11 – Тест “Удалить текущую БД”



Касса аэрофлота Серегин К.С.

Файл Рейсы

Критерий найденный

Значение фильтрации

Значение поиска

Добавить Изменить

Номер рейса

Номер самолёта

Номер места

Номер ряда

Класс полёта

Стоимость

Название авиакомпании

Добавить

Удалить

Выход

Номер рейса	Номер самолёта	Номер места	Номер ряда	Класс полёта	Стоимость	Название авиакомпании
14	30	45	A	Бизнес	100	Аэрофлот

Рисунок 12 – Тест “Открыть БД”

Касса аэрофлота Серегин К.С.

Файл Рейсы

Критерий найденный

Значение фильтрации

Значение поиска

Найдено 1 совпадений из 5

Добавить Изменить

Номер рейса

Номер самолёта

Номер места

Номер ряда

Класс полёта

Стоимость

Название авиакомпании

Добавить

Удалить

Выход

Номер рейса	Номер самолёта	Номер места	Номер ряда	Класс полёта	Стоимость	Название авиакомпании
14	30	45	A	Бизнес	100	Аэрофлот
145	15	45	A	Бизнес	100	Аэрофлот
14	14	45	A	Бизнес	100	Аэрофлот
14	17	45	A	Бизнес	100	Аэрофлот
31	17	45	A	Бизнес	100	Аэрофлот

Рисунок 13 – Тест “Поиск по критерию в БД”

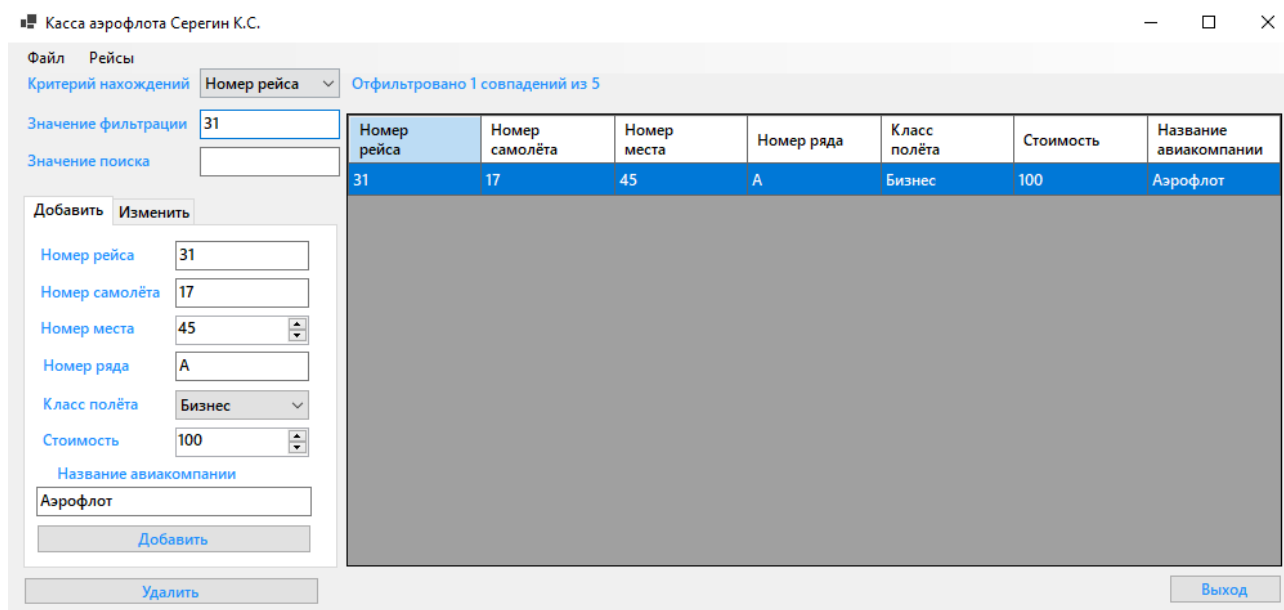


Рисунок 14 – Тест “Фильтрация по критерию в БД”

В ходе выполнения тестирования несовпадения ожидаемого и наблюдаемого результата не выявлены. Следовательно, можно сделать вывод, что программа работает корректно и готова к использованию.

## **5 Описание программы**

### **5.1 Разработка приложения CourseWork\_OOP\_Seregin.exe**

Программа CourseWork\_OOP\_Seregin.exe является основным модулем для работы с базой данных, предназначенной для хранения информации о рейсах и клиентах. Программа разработана с учетом удобства пользователя и интуитивно понятного интерфейса, что позволяет легко выполнять операции по добавлению, редактированию, удалению записей, а также осуществлять поиск, фильтрацию и сортировку данных. Также предусмотрена возможность сохранения и загрузки базы данных, что обеспечивает долговременное хранение и актуальность информации [7].

При запуске программы, пользователю демонстрируется заставка с приветствием, после чего появляется главное окно с меню, из которого можно выбрать требуемую операцию. Программа ожидает действия пользователя, и при выборе одного из пунктов меню, управление передается в соответствующие функции, реализованные в файлах FormClient.cs.

Программа предоставляет пользователю возможность создавать новую базу данных, для чего достаточно выбрать пункт меню «Создать базу данных» в вкладке «Файл». Для работы с уже существующими данными, предусмотрена функция открытия базы данных. В этом случае пользователь выбирает пункт «Открыть базу данных», после чего ему предложат выбрать файл базы данных из директории. Это дает возможность загрузить ранее сохраненную информацию и продолжить работу с ней без потери данных.

Кроме того, программа поддерживает функцию сохранения базы данных. Пользователи могут выбрать пункт меню «Сохранить базу данных», чтобы сохранить текущие изменения в выбранной директории. При этом появится окно с выбором места и имени файла для сохранения, что обеспечивает удобство работы и сохранность данных.

Программа предоставляет возможность добавлять новые записи в базу данных. Для этого нужно выбрать пункт «Добавить», заполнить форму

необходимыми данными и сохранить запись. Для редактирования существующих записей достаточно выбрать нужную строку в таблице, нажать «Редактировать» и внести необходимые изменения. Удаление записи выполняется просто — нужно выбрать строку и нажать «Удалить».

Для эффективной работы с большим количеством записей программа также поддерживает поиск, фильтрацию и сортировку данных. Для поиска необходимо выбрать критерий, ввести значение и программа автоматически найдет все подходящие записи, подсвечивая их для удобства. Фильтрация данных позволяет отобразить только те записи, которые соответствуют заданному критерию. Сортировка позволяет упорядочить записи по различным параметрам, что значительно облегчает работу с большими объемами информации.

Все эти функции обеспечивают гибкость и удобство работы с базой данных, что делает программу мощным инструментом для хранения и обработки информации о рейсах и клиентах [9].

## **Выводы**

В рамках курсового проекта была разработана программа для работы с базой данных, которая включает операции по добавлению, изменению, фильтрации, удалению рейсов и клиентов. Основной задачей программы является управление данными о рейсах и клиентах, их сохранение и дальнейшее использование. Вся информация сохраняется в виде файлов, что позволяет поддерживать актуальность базы данных и использовать её в дальнейшем.

Программа поддерживает работу с коллекциями данных и использует объектно-ориентированный подход для реализации функциональности. Создание базы данных, добавление и удаление записей, редактирование информации о клиентах и рейсах, а также фильтрация, поиск и сортировка данных, становятся доступными благодаря интуитивно понятному пользовательскому интерфейсу, построенному на технологии Windows Forms. Данные сохраняются в формате JSON, что делает их легко доступными для дальнейшего использования или анализа.

Процесс разработки программы позволил нам приобрести навыки в создании программных решений на языке C# в среде разработки Microsoft Visual Studio. Мы расширили навык проектировать и реализовывать функциональность для работы с базой данных, взаимодействовать с пользователем через графический интерфейс и обеспечивать сохранность данных. Этот опыт укрепил наши знания в области объектно-ориентированного программирования и позволил приобрести практические навыки в проектировании и разработке программного обеспечения с использованием современных технологий и инструментов [10].

В ходе работы над проектом также была проведена отладка приложения, его функциональное тестирование, что позволило выявить и устранить возможные ошибки. Результат работы соответствует техническим требованиям и полностью готов к использованию в реальных условиях.

Процесс разработки программы включал в себя несколько этапов: от анализа требований до окончательной реализации и тестирования. Особое внимание было уделено правильной организации данных и созданию удобного интерфейса для пользователя.

## **Список использованных источников**

1. Работа с базами данных в C# и .NET: учебное пособие / А.А. Лебедев – М.: ДМК Пресс, 2019. – 368 с.
2. Разработка баз данных в C#: учебное пособие / Е.Н. Сидоров – М.: Изд-во МГТУ, 2020. – 352 с.
3. Visual C# на примерах / М.Э. Абрамян – М.: ДМК Пресс, 2008. – 560 с.
4. Программирование на C#: учебное пособие / В.В. Поляков – М.: Юрайт, 2021. – 448 с.
5. Моделирование информационных систем. Unified Modeling Language. Учебное пособие для вузов: учебное пособие / И.Ю. Матюшичев, А.В. Флегонтов, Лань, - 2020 – 352 с.
6. Тестирование программного обеспечения: Учебное пособие: учебное пособие / Ю.В. Морозова, ТУСУР, 2019. — 120 с.
7. C# для начинающих / Б.И. Пахомов – СПб. : БХВ-Петербург, 2020. – 416 с.
8. Windows Forms в C#: учебное пособие / Д.В. Кузнецов – СПб.: Питер, 2020. – 320 с.
9. Разработка на языке C# приложений с графическим интерфейсом (использование Windows Forms): учеб. пособие / С.Э. Мурадханов – Москва : Изд-во «МИСиС», 2019. – 396 с.
10. Лучшие практики программирования на C# – URL: <https://habr.com/ru/articles/438796/> Дата обращения: 04.04.2025.
11. Использование Windows Forms для создания GUI-приложений в C# – URL: <https://habr.com/ru/articles/491474/> Дата обращения: 04.04.2025.
12. Разработка Windows-приложений на C#: учеб. пособие / Н.А. Осипов – СПб. : Питер, 2021. – 320 с.
13. 5.1 Windows Forms и .NET – URL: <https://www.flenov.info/books/read/biblia-csharp/49> Дата обращения: 04.04.2025.

14. Как подключить Entity Framework Core к Windows Forms C# .Net Core?  
– URL: <https://qna.habr.com/q/1070648> Дата обращения: 04.04.2025.

15. Практическое руководство по разработке приложений Windows Forms на C# / И.И. Смирнов – М.: ДМК Пресс, 2019. – 288 с.



## Приложение А – Код программы

### Program.cs

```
namespace CourseWork_OOP_Seregin
{
    internal static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            // To customize application configuration such as set high DPI settings
            or default font,
            // see https://aka.ms/applicationconfiguration.
            ApplicationConfiguration.Initialize();
            Application.Run(new Form2());
        }
    }
}
```

### DataBase.cs

```
using System.Text.Json;
using System.Text.Json.Serialization;

namespace CourseWork_OOP_Seregin
{
    public class DataBase
    {
        public static readonly string rash = ".DataBase";

        private List<Client> _clients;
        public List<Client> GetClients() { return _clients; }

        private List<Reis> _reis;
        public List<Reis> GetReis() { return _reis; }

        public DataBase()
        {
            _clients = new List<Client>();
            _reis = new List<Reis>();
        }

        public void load(string path)
        {
            // Абстрактные классы для работы с текстовыми потоками в .NET.
            TextReader? reader = null;
            try
            {
                reader = new StreamReader(path); // Создаем объект пример
                if (reader == null) throw new ArgumentNullException();
                string? tmpLineReis = reader.ReadLine();
                string? tmpLineClient = reader.ReadLine();
                if (tmpLineReis == null || tmpLineClient == null) throw new
                ArgumentNullException();
                List<Reis>? reises =
                JsonSerializer.Deserialize<List<Reis>>(tmpLineReis);
                List<Client>? clients =
                JsonSerializer.Deserialize<List<Client>>(tmpLineClient);
                if (reises == null || clients == null) throw new
                ArgumentNullException();
                _reis = reises;
            }
            catch { }
        }
    }
}
```

```

        _clients = clients;
    }
    finally
    {
        if (reader != null)
            reader.Close();
    }
}

public void save(string path)
{
    TextWriter? writer = null;
    try
    {
        var Reises = JsonSerializer.Serialize(_reis,
Context.Default.ListReis);
        var clients = JsonSerializer.Serialize(_clients,
Context.Default.ListClient);
        writer = new StreamWriter(path);
        writer.Write(Reises + "\n" + clients);
    }
    finally
    {
        if (writer != null)
            writer.Close();
    }
}

}

public record Client(
    string _NumReis,
    string _Plane,
    int _Sit,
    string _Row,
    string _Class,
    int _Price,
    string _Name
);

public record Reis(
    string _NumReis,
    string _From,
    string _To
);

[JsonSerializable(typeof(List<Client>))]
[JsonSerializable(typeof(List<Reis>))]
internal partial class Context : JsonSerializerContext
{
}
}

```

## RaisesForm.cs

```

using System.Text.RegularExpressions;

namespace CourseWork_OOP_Seregin
{
    public partial class RaisesForm : Form
    {
        DataBase _data;
        Reis? _lastreis;
        public RaisesForm(DataBase data)
        {
            InitializeComponent();
            _data = data;
            UpDate();
            BlockEdit(true);
        }
    }
}

```

```

    }

    private void backToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Close();
    }

    private void buttonAdd_Click(object sender, EventArgs e)
    {
        string Reis = Regex.Replace(textBoxAddReis.Text, @"\s+", " ");
        string From = Regex.Replace(textBoxAddFrom.Text, @"\s+", " ");
        string To = Regex.Replace(textBoxAddTo.Text, @"\s+", " ");

        if (Reis == "" || From == "" || To == "")
        {
            MessageBox.Show("Одно из полей не содержит данных", "Ошибка с значением");
            return;
        }

        foreach (Reis reis in _data.GetReis())
        {
            if (reis._NumReis == Reis)
            {
                MessageBox.Show("Такой рейс уже есть", "Ошибка с значением");
                return;
            }
        }
        _data.GetReis().Add(new Reis(Reis, From, To));
        Update();
    }

    void BlockEdit(bool val)
    {
        label1.Enabled = !val;
        label2.Enabled = !val;
        label3.Enabled = !val;
        textBoxEditTo.Enabled = !val;
        textBoxEditFrom.Enabled = !val;
        textBoxEditReis.Enabled = !val;
        buttonEdit.Enabled = !val;
        buttonDelete.Enabled = !val;
    }

    private void buttonEdit_Click(object sender, EventArgs e)
    {
        if (_lastreis == null) throw new ArgumentNullException();
        string Reis = Regex.Replace(textBoxEditReis.Text, @"\s+", " ");
        string From = Regex.Replace(textBoxEditFrom.Text, @"\s+", " ");
        string To = Regex.Replace(textBoxEditTo.Text, @"\s+", " ");

        if (Reis == "" || From == "" || To == "")
        {
            MessageBox.Show("Одно из полей не содержит данных", "Ошибка с значением");
            return;
        }

        if (_lastreis._NumReis != Reis)
        {
            foreach (Reis reis in _data.GetReis())
            {
                if (reis._NumReis == Reis)
                {
                    MessageBox.Show("Такой рейс уже есть", "Ошибка с значением");
                    return;
                }
            }
        }
    }

```

```

        }
    }
    _data.GetReis().Remove(_lastreis);
    _lastreis = null;
    _data.GetReis().Add(new Reis(Reis, From, To));
    UpDate();
    dataGridView1_SelectionChanged(sender, e);
}

private void buttonDelete_Click(object sender, EventArgs e)
{
    if (_lastreis == null) throw new ArgumentNullException();
    _data.GetReis().Remove(_lastreis);
    _lastreis = null;
    UpDate();
    dataGridView1_SelectionChanged(sender, e);
}

void UpDate()
{
    if (_data == null) throw new ArgumentNullException();
    dataGridView1.Rows.Clear();
    foreach (Reis reis in _data.GetReis())
    {
        dataGridView1.Rows.Add(reis._NumReis, reis._From, reis._To);
    }
}

private void dataGridView1_SelectionChanged(object sender, EventArgs e)
{
    if (dataGridView1.CurrentCell != null)
    {
        BlockEdit(false);
        textBoxEditReis.Text =
Convert.ToString(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[0].Value);
        textBoxEditFrom.Text =
Convert.ToString(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[1].Value);
        textBoxEditTo.Text =
Convert.ToString(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[2].Value);
        foreach (Reis reis in _data.GetReis())
        {
            if (reis._NumReis ==
Convert.ToString(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[0].Value) &&
                reis._From ==
Convert.ToString(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[1].Value) &&
                reis._To ==
Convert.ToString(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[2].Value))
            {
                _lastreis = reis;
            }
        }
    }
    else
    {
        _lastreis = null;
        BlockEdit(true);
    }
}

```

```

    }
}

```

## ClientForm.cs

```

using System.Text.RegularExpressions;

namespace CourseWork_OOP_Seregin
{
    public partial class ClientForm : Form
    {
        DataBase? _data;
        Client? _lastClient;
        public ClientForm()
        {
            InitializeComponent();
            Block(true);
            labelFilterCount.Visible = false;
            openFileDialog1.Filter = String.Format("База данных|*{0}",
DataBase.rash);
            openFileDialog1.DefaultExt = DataBase.rash;
            saveFileDialog1.Filter = String.Format("База данных|*{0}",
DataBase.rash);
            saveFileDialog1.DefaultExt = DataBase.rash;
            BlockEdit(true);
        }
        void Block(bool val)
        {
            tabControl1.Enabled = !val;
            dataGridView1.Enabled = !val;
            openDBToolStripMenuItem.Enabled = val;
            createDBToolStripMenuItem.Enabled = val;
            deleteDBToolStripMenuItem.Enabled = !val;
            saveDBToolStripMenuItem.Enabled = !val;
            labelFilter.Enabled = !val;
            labelFilterValue.Enabled = !val;
            labelFinderValue.Enabled = !val;
            comboBoxFilter.Enabled = !val;
            raisesToolStripMenuItem.Enabled = !val;
            buttonDelete.Enabled = !val;
            textBoxFilterValue.Enabled = !val;
            textBoxFilterValue.Text = "";
            textBoxFinderValue.Enabled = !val;
            textBoxFinderValue.Text = "";
        }
        void BlockEdit(bool val)
        {
            label2.Enabled = !val;
            label4.Enabled = !val;
            label5.Enabled = !val;
            label6.Enabled = !val;
            label7.Enabled = !val;
            label8.Enabled = !val;
            label9.Enabled = !val;
            textBoxEditAerline.Enabled = !val;
            textBoxEditPlane.Enabled = !val;
            textBoxEditReis.Enabled = !val;
            textBoxEditRow.Enabled = !val;
            comboBoxEditClass.Enabled = !val;
            numericUpDownEditPrice.Enabled = !val;
            numericUpDownEditSit.Enabled = !val;
            buttonEdit.Enabled = !val;
            buttonDelete.Enabled = !val;
            textBoxFilterValue.Enabled = !val;
            textBoxFinderValue.Enabled = !val;
        }
    }
}

```

```

    }

    private void buttonAdd_Click(object sender, EventArgs e)
    {
        if (_data == null) throw new ArgumentNullException();
        string Reis = Regex.Replace(textBoxAddReis.Text, @"\s+", " ");
        string Plane = Regex.Replace(textBoxAddPlane.Text, @"\s+", " ");
        string Name = Regex.Replace(textBoxAddNameAirlines.Text, @"\s+", " ");
        string Row = Regex.Replace(textBoxAddRow.Text, @"\s+", " ");
        string Class = comboBoxAddClass.Text;
        if (Reis == "" || Plane == "" || Name == "" || Row == "" || Class == "")
        {
            MessageBox.Show("Одно из полей не содержит данных", "Ошибка со
значением");
            return;
        }
        int Sit = Convert.ToInt32(numericUpDownAddSit.Value);
        int Price = Convert.ToInt32(numericUpDownAddPrice.Value);
        if (Sit < 1 || Price < 0)
        {
            MessageBox.Show("Некорректное числовое значение", "Ошибка со
значением");
            return;
        }
        foreach (Client client in _data.GetClients())
        {
            if (client._Plane == Plane && client._NumReis == Reis && client._Sit
== Sit && client._Row == Row)
            {
                MessageBox.Show("Место уже занято", "Ошибка со значением");
                return;
            }
        }
        _data.GetClients().Add(new Client(Reis, Plane, Sit, Row, Class, Price,
Name));
        Update();
    }

    private void buttonEdit_Click(object sender, EventArgs e)
    {
        if (_data == null) throw new ArgumentNullException();
        if (_lastClient == null) throw new ArgumentNullException();
        string Reis = Regex.Replace(textBoxEditReis.Text, @"\s+", " ");
        string Plane = Regex.Replace(textBoxEditPlane.Text, @"\s+", " ");
        string Name = Regex.Replace(textBoxEditAerline.Text, @"\s+", " ");
        string Row = Regex.Replace(textBoxEditRow.Text, @"\s+", " ");
        string Class = comboBoxEditClass.Text;
        if (Reis == "" || Plane == "" || Name == "" || Row == "" || Class == "")
        {
            MessageBox.Show("Одно из полей не содержит данных", "Ошибка со
значением");
            return;
        }
        int Sit = Convert.ToInt32(numericUpDownEditSit.Value);
        int Price = Convert.ToInt32(numericUpDownEditPrice.Value);
        if (Sit < 1 || Price < 0)
        {
            MessageBox.Show("Некорректное числовое значение", "Ошибка со
значением");
            return;
        }
        if (_lastClient._Sit != Sit || _lastClient._NumReis != Reis ||
_lastClient._Plane != Plane)
            foreach (Client client in _data.GetClients())
            {

```

```

        if (client._Plane == Plane && client._NumReis == Reis &&
client._Sit == Sit && client._Row == Row)
        {
            MessageBox.Show("Место уже занято", "Ошибка со значением");
            return;
        }
    }

    _data.GetClients().Remove(_lastClient);
    _lastClient = null;
    _data.GetClients().Add(new Client(Reis, Plane, Sit, Row, Class, Price,
Name));
    UpDate();
    dataGridView1_SelectionChanged(sender, e);
}
private void openDBToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() != DialogResult.OK) return;
    _data = new DataBase();
    try
    {
        _data.load(openFileDialog1.FileName);
    }
    catch (System.InvalidOperationException)
    {
        MessageBox.Show("Открыт некорректный файл", "Ошибка");
        return;
    }
    Block(false);
    UpDate();
}
private void createDBToolStripMenuItem_Click(object sender, EventArgs e)
{
    _data = new DataBase();
    Block(false);
}
private void deleteDBToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (_data == null) throw new ArgumentNullException();
    dataGridView1.Rows.Clear();
    _data = null;
    Block(true);
}
private void saveDBToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (_data == null) throw new ArgumentNullException();
    if (saveFileDialog1.ShowDialog() != DialogResult.OK) return;
    _data.save(saveFileDialog1.FileName);
}
private void dataGridView1_SelectionChanged(object sender, EventArgs e)
{
    if (_data == null) throw new ArgumentNullException();
    if (dataGridView1.CurrentCell != null)
    {
        BlockEdit(false);
        textBoxEditReis.Text =
Convert.ToString(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[0].Val
ue);
        textBoxEditPlane.Text =
Convert.ToString(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[1].Val
ue);
        numericUpDownEditSit.Value =
Convert.ToInt32(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[2].Valu
e);
    }
}

```

```

        textBoxEditRow.Text =
Convert.ToString(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[3].Value);
        comboBoxEditClass.SelectedIndex =
comboBoxEditClass.Items.IndexOf(Convert.ToString(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[4].Value));
        numericUpDownEditPrice.Value =
Convert.ToInt32(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[5].Value);
        textBoxEditAerline.Text =
Convert.ToString(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[6].Value);
        foreach (Client client in _data.GetClients())
        {
            if (client._NumReis ==
Convert.ToString(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[0].Value) &&
                client._Plane ==
Convert.ToString(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[1].Value) &&
                client._Sit ==
Convert.ToInt32(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[2].Value) &&
                client._Row ==
Convert.ToString(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[3].Value) &&
                client._Class ==
Convert.ToString(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[4].Value) &&
                client._Price ==
Convert.ToInt32(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[5].Value) &&
                client._Name ==
Convert.ToString(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex].Cells[6].Value))
            {
                _lastClient = client;
            }
        }
        else
        {
            _lastClient = null;
            BlockEdit(true);
        }
    }
    private void buttonDelete_Click(object sender, EventArgs e)
    {
        if (_data == null) throw new ArgumentNullException();
        if (_lastClient == null) throw new ArgumentNullException();
        _data.GetClients().Remove(_lastClient);
        _lastClient = null;
        //
        dataGridView1.CurrentCell.Dispose();
        Update();
        dataGridView1_SelectionChanged(sender, e);
    }
    private void textBoxFilterValue_TextChanged(object sender, EventArgs e)
    {
        dataGridView1.Rows.Clear();
        if (_data == null) throw new ArgumentNullException();
        int count = 0;

        string filterText = textBoxFilterValue.Text.ToLower();

```



```

// Если поле фильтра пустое, сбрасываем цвет всех строк на белый
if (string.IsNullOrEmpty(filterText))
{
    labelFilterCount.Visible = false; // Скрываем счетчик, если фильтр
пустой
}

switch (comboBoxFilter.Text)
{
    case "":
        MessageBox.Show("Выберите критерий", "Ошибка");
        break;
    case "Номер рейса":
        foreach (Client client in _data.GetClients())
        {
            if
(client._NumReis.Contains(Convert.ToString(textBoxFilterValue.Text)))
            {
                dataGridView1.Rows.Add(client._NumReis, client._Plane,
client._Sit, client._Row, client._Class, client._Price, client._Name);
                count++;
            }
        }
        break;
    case "Номер самолёта":
        foreach (Client client in _data.GetClients())
        {
            if
(client._Plane.Contains(Convert.ToString(textBoxFilterValue.Text)))
            {
                dataGridView1.Rows.Add(client._NumReis, client._Plane,
client._Sit, client._Row, client._Class, client._Price, client._Name);
                count++;
            }
        }
        break;
    case "Номер места":
        foreach (Client client in _data.GetClients())
        {
            if
(Convert.ToString(client._Sit).Contains(Convert.ToString(textBoxFilterValue.Text)))
            {
                dataGridView1.Rows.Add(client._NumReis, client._Plane,
client._Sit, client._Row, client._Class, client._Price, client._Name);
                count++;
            }
        }
        break;
    case "Номер ряда":
        foreach (Client client in _data.GetClients())
        {
            if
(Convert.ToString(client._Row).Contains(Convert.ToString(textBoxFilterValue.Text)))
            {
                dataGridView1.Rows.Add(client._NumReis, client._Plane,
client._Sit, client._Row, client._Class, client._Price, client._Name);
                count++;
            }
        }
}

```

```

        }
        break;
        case "Класс полёта":
            foreach (Client client in _data.GetClients())
            {
                if
(client._Class.Contains(Convert.ToString(textBoxFilterValue.Text)))
                {
                    dataGridView1.Rows.Add(client._NumReis, client._Plane,
client._Sit, client._Row, client._Class, client._Price, client._Name);
                    count++;
                }
            }
            break;
        case "Стоимость":
            foreach (Client client in _data.GetClients())
            {
                if
(Convert.ToString(client._Price).Contains(Convert.ToString(textBoxFilterValue.Text))
)
                {
                    dataGridView1.Rows.Add(client._NumReis, client._Plane,
client._Sit, client._Row, client._Class, client._Price, client._Name);
                    count++;
                }
            }
            break;
        case "Название авиакомпании":
            foreach (Client client in _data.GetClients())
            {
                if
(client._Name.Contains(Convert.ToString(textBoxFilterValue.Text)))
                {
                    dataGridView1.Rows.Add(client._NumReis, client._Plane,
client._Sit, client._Row, client._Class, client._Price, client._Name);
                    count++;
                }
            }
            break;
    }
    labelFilterCount.Visible = true;
    labelFilterCount.Text = $"Отфильтровано {count} совпадений из
{_data.GetClients().Count}";
}
private void textBoxFinderValue_TextChanged(object sender, EventArgs e)
{
    if (_data == null) throw new ArgumentNullException();
    int count = 0;

    string filterText = textBoxFinderValue.Text.ToLower();

    // Если поле фильтра пустое, сбрасываем цвет всех строк на белый
    if (string.IsNullOrEmpty(filterText))
    {
        foreach (DataGridViewRow row in dataGridView1.Rows)
        {
            row.DefaultCellStyle.BackColor = Color.White;
        }
        labelFilterCount.Visible = false; // Скрываем счетчик, если фильтр
пустой

```

```

        return; // Прерываем выполнение, чтобы не фильтровать строки
    }

    // Если фильтр не пустой, сбрасываем цвет всех строк
    foreach (DataGridViewRow row in dataGridView1.Rows)
    {
        row.DefaultCellStyle.BackColor = Color.White;
    }

    foreach (DataGridViewRow row in dataGridView1.Rows)
    {
        Client client = _data.GetClients()[row.Index];
        bool match = false;

        switch (comboBoxFilter.Text)
        {
            case "":
                MessageBox.Show("Выберите критерий", "Ошибка");
                break;
            case "Номер рейса":
                match = client._NumReis.ToLower().Contains(filterText);
                break;
            case "Номер самолёта":
                match = client._Plane.ToLower().Contains(filterText);
                break;
            case "Номер места":
                match = client._Sit.ToString().Contains(filterText);
                break;
            case "Номер ряда":
                match = client._Row.ToString().Contains(filterText);
                break;
            case "Класс полёта":
                match = client._Class.ToLower().Contains(filterText);
                break;
            case "Стоимость":
                match = client._Price.ToString().Contains(filterText);
                break;
            case "Название авиакомпании":
                match = client._Name.ToLower().Contains(filterText);
                break;
        }

        if (match)
        {
            row.DefaultCellStyle.BackColor = Color.Yellow;
            count++;
        }
    }

    labelFilterCount.Visible = true;
    labelFilterCount.Text = $"Найдено {count} совпадений из
{_data.GetClients().Count}";
}

private void raisesToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (_data == null) throw new ArgumentNullException();
    RaisesForm raisesForm = new RaisesForm(_data);
    raisesForm.ShowDialog();
}

private void closeButton_Click(object sender, EventArgs e)
{
    Application.Exit();
}

private void ClientForm_Load(object sender, EventArgs e)
{

```

```

    }
    void UpDate()
    {
        if (_data == null) throw new ArgumentNullException();
        dataGridView1.Rows.Clear();
        foreach (Client client in _data.GetClients())
        {
            dataGridView1.Rows.Add(client._NumReis, client._Plane, client._Sit,
client._Row, client._Class, client._Price, client._Name);
        }
        textBoxFilterValue.Text = "";
    }
    private void textBoxAddRow_TextChanged(object sender, EventArgs e)
    {
    }
}
}

```

## Приложение В – Руководство пользователя

Программа CourseWork\_OOP\_Seregin.exe предназначена для хранения информации рейсах и клиентах. Программа имеет интуитивно понятный интерфейс и поддерживает такие операции как добавления новой записи, удаления записи, редактирования записи, а также поиск, фильтрация и сортировка записи, сохранение существующей БД.

Для создания и отображения в окне новой БД необходимо выбрать пункт меню «Создать базу данных» во вкладке «Файл» (рисунок В.1).

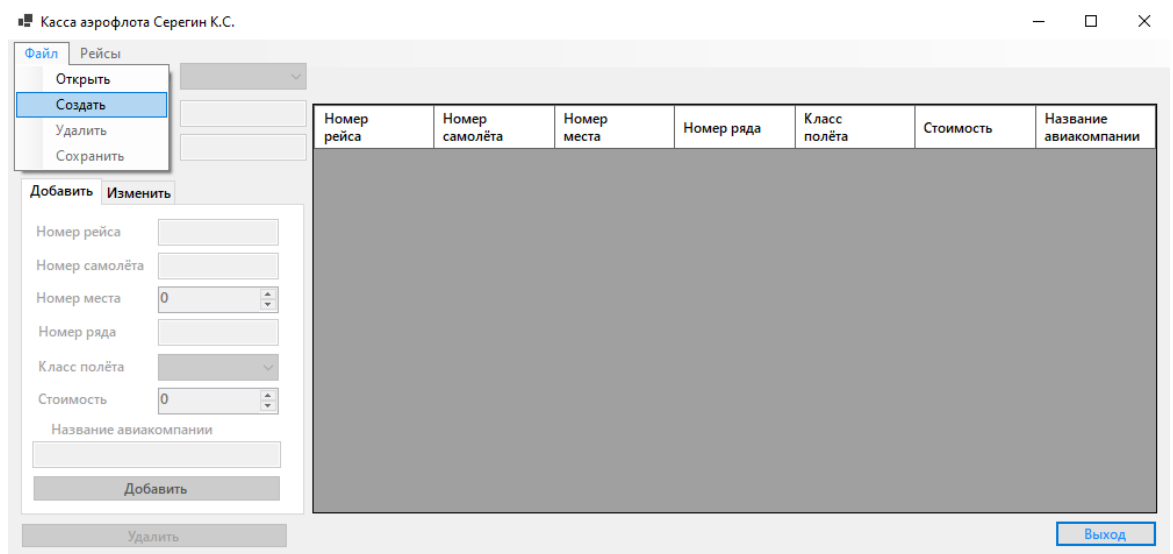
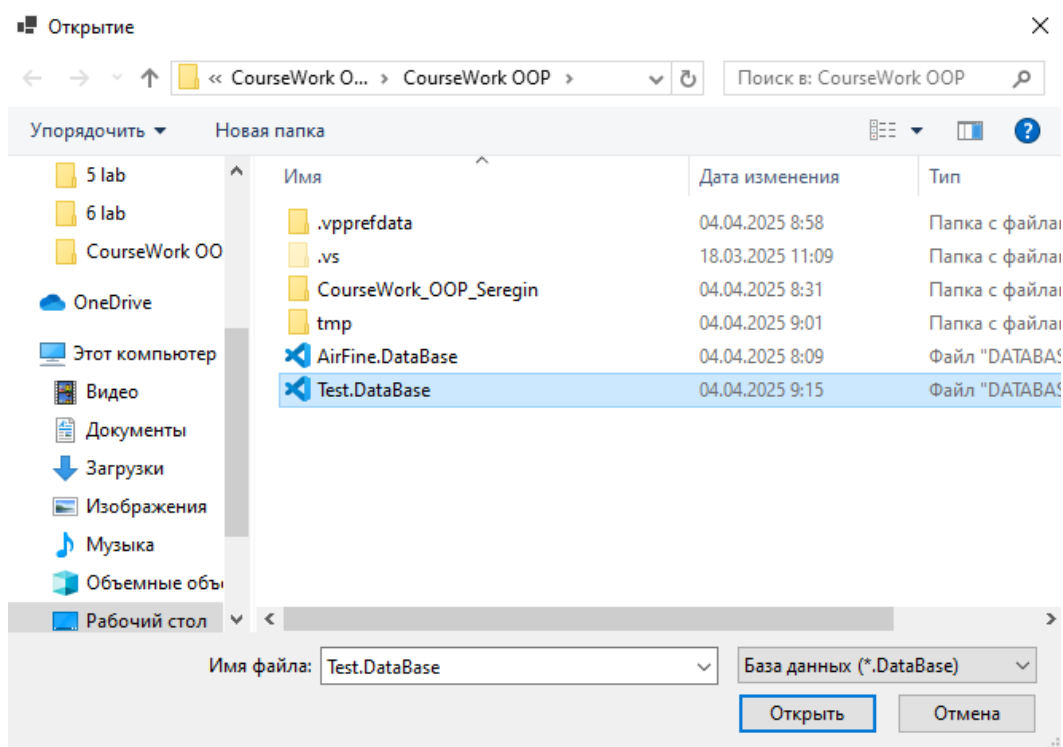


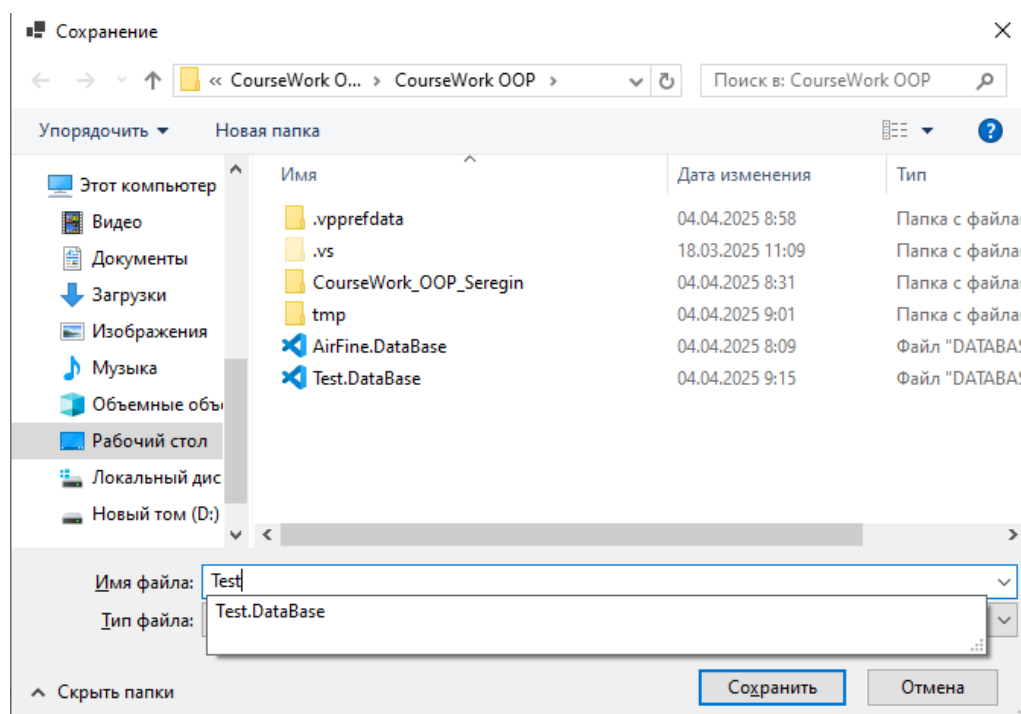
Рисунок В.1 – Создание новой БД

Для открытия и вывода на экран существующей БД необходимо выбрать пункт меню «Открыть базу данных» во вкладке «Файл» (рисунок В.1). Будет предложено выбрать файл из каталога.



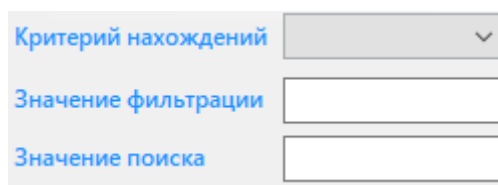
**Рисунок В.2 – Открытие существующей БД**

Для сохранения существующего каталога необходимо выбрать пункт меню «Сохранить базу данных...» во вкладке «БД». На экране появиться окно с выбором директории сохранения и именем файла.



**Рисунок В.3 – Сохранение БД в файл**

Для поиска записи или фильтрации в каталоге необходимо выбрать критерий, далее ввести в поле поиск или фильтрации значения по которым вы хотите получить данные.



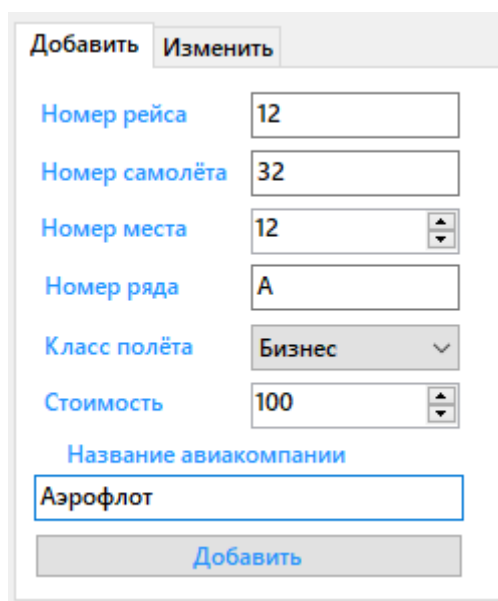
Критерий нахождений

Значение фильтрации

Значение поиска

**Рисунок В.4 – Меню фильтрации**

Для добавления новой записи необходимо выбрать «Добавить» и ввести нужные данные в появившуюся форму в соответствии с их типом данных.



Добавить Изменить

Номер рейса

Номер самолёта

Номер места

Номер ряда

Класс полёта

Стоимость

Название авиакомпании

Добавить

**Рисунок В.5 – Окно добавления**

Для редактирования данных нужно выбрать клиента в таблице, кликнув на нужную строку, затем выбрать пункт «Редактировать», ввести новые данные для редактирования.

Добавить | **Изменить**

Номер рейса: 7

Номер самолёта: 777

Номер места: 7

Номер ряда: A

Класс полёта: Бизнес

Стоимость: 100

Название авиакомпании: Аэрофлот

Редактировать

**Рисунок В.6 – Меню редактирования**

Для удаления записи необходимо выбрать строку в таблице и нажать «Удалить».

Удалить

**Рисунок В.7 – Кнопка удаления элемента**

Для сортировки нужно нажать на заголовок таблицы и данные отсортируются по выбранному параметру, для изменения критерия сортировки (по убыванию или по возрастанию) кликните повторно на заголовок.

Касса аэрофлота Серегин К.С.

Файл | Рейсы

Критерий находжений: [v]

Значение фильтрации: [ ]

Значение поиска: [ ]

Добавить | **Изменить**

Номер рейса: 12

Номер самолёта: 32

Номер места: 12

Номер ряда: A

Класс полёта: Бизнес

Стоимость: 100

Название авиакомпании: Аэрофлот

Добавить

Удалить

Номер рейса	Номер самолёта	Номер места	Номер ряда	Класс полёта	Стоимость	Название авиакомпании
7777	777	7	A	Бизнес	100	Аэрофлот
7773	777	7	A	Бизнес	100	Аэрофлот
77	777	7	A	Бизнес	100	Аэрофлот
7	777	7	A	Бизнес	100	Аэрофлот

Выход

**Рисунок В.8 – Сортировка**



Для поиска нужно выбрать критерий нахождения сверху и ввести нужно значение для поиска. Подходящие строки будут подсвечены жёлтым цветом.

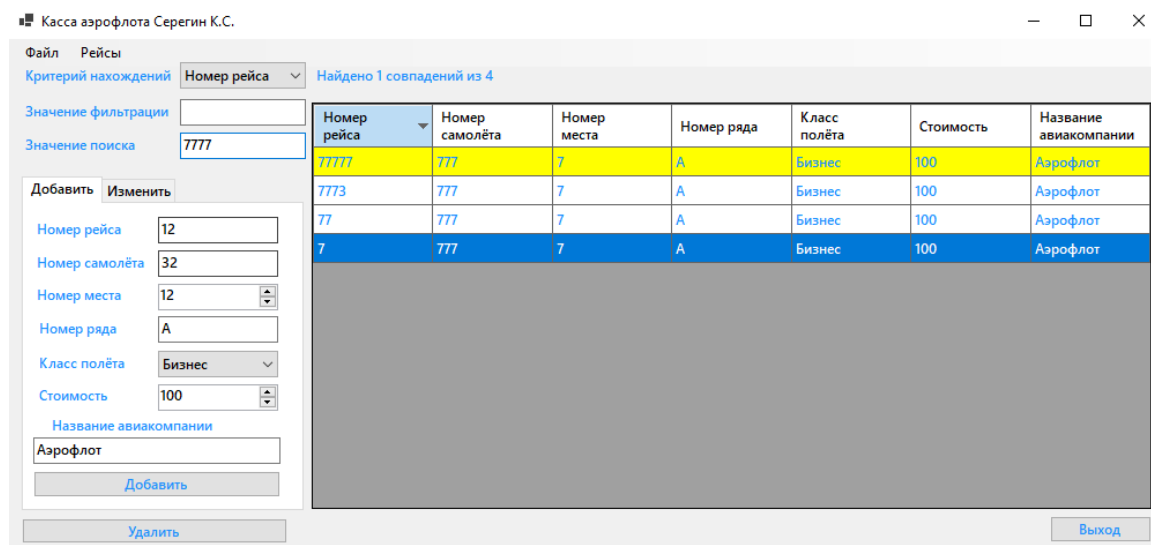


Рисунок В.9 – Поиск

Для фильтрации нужно выбрать критерий нахождения сверху и ввести нужно значение для фильтрации. Пользователю будут удалены неподходящие строки и останутся только нужные.

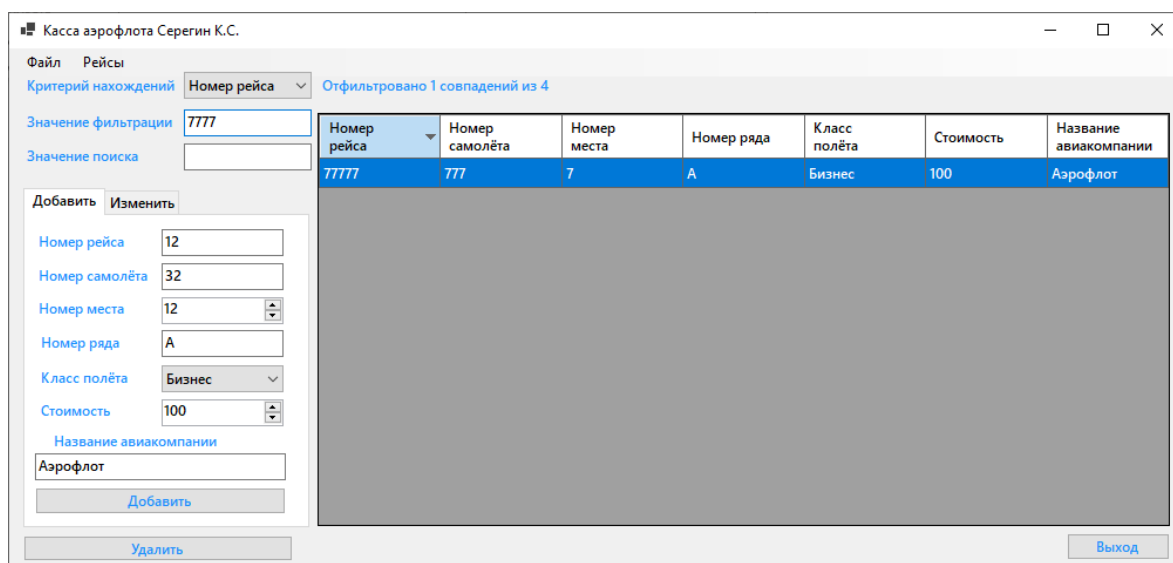


Рисунок В.9 – Поиск