**Содержание**

[**Введение 4**](#_Toc148302856)

[**Основная часть 5**](#_Toc148302857)

[1 Объектная модель задачи 5](#_Toc148302858)

[2 Структура инфологической модели и результаты ее нормализации 10](#_Toc148302859)

[3 Логическая и физическая модели данных 13](#_Toc148302860)

[3.1 Логическая модель данных 13](#_Toc148302861)

[3.2 Физическая модель данных 13](#_Toc148302862)

[4 Описание базы данных на сервере 14](#_Toc148302863)

[5 Формы входных и выходных документов 17](#_Toc148302864)

[6 Инструкция пользователя 22](#_Toc148302865)

[7 Описание и результаты тестов 25](#_Toc148302866)

[**Заключение 36**](#_Toc148302867)

[**Список использованных источников 37**](#_Toc148302868)

[**Приложение А Листинг 38**](#_Toc148302869)

# **Введение**

В настоящее время практически во всех сферах человеческой деятельности применяются информационные технологии, неотъемлемой частью которой являются базы данных. База данных (БД) – это совокупность данных предназначенных для машинной обработки; единая система данных, организованная по определенным правилам, которые предусматривают общие принципы описания, хранения и обработки данных.

Для работы с базами данных и решения проблем обработки информации используются современные компьютеры с соответствующими системами управлениями базами данных (СУБД). Система управления базами данных – программа, позволяющая сформировать базу данных, вносить в нее изменения, производить поиск требуемых данных по запросам и обрабатывать хранящиеся данные. СУБД также обеспечивает многопользовательский доступ к данным, что позволяет ей обслуживать одновременно тысячи пользователей.

**Цель курсового проекта:** Курсовой проект предназначен для получения более глубоких навыков по проектированию структуры БД; проектированию, написанию и отладке приложений для ведения БД и разработке разнообразных запросов к БД. Курсовой проект знакомит с многозвенной архитектурой, Internet приложениями, мобильными приложениями, современными СУБД.

**Вариант:** 15

**Технология:** Клиент-серверное приложение

**Задание:** БД «Авиакомпании» Необходимо хранить информацию об авиакомпаниях (шифр, название, адрес), кассах (номер, адрес), кассирах (табельный номер, ФИО). Клиенты (номер и серия паспорта, ФИО) приобретают билеты (номер, тип, дата продажи, касса, кассир, авиакомпания). Билет содержит не более четырех купонов (номер, направление полета, тариф, клиент). Выходные документы:

1. Билеты, проданные за указанный месяц указанной авиакомпании.
2. Общая сумма от продаж билетов каждой авиакомпании.
3. Список клиентов авиакомпаний на заданную дату.

# **Основная часть**

1. **Объектная модель задачи**

Объектная модель задачи выполняется с помощью пакета StarUML и включает создание трех типов диаграмм:

* диаграммы сценариев, или использования (Use Case);
* диаграммы классов (Classes);
* диаграммы последовательности (Sequence).

Построим диаграмму сценариев, в которой необходимо определить роли пользователей системы (актеров) и их функции (прецеденты). Для этого рассмотрим предполагаемых пользователей базы данных на рисунке 1.1, которыми являются реестр авиакомпаний, владелец авиакомпании и работник авиакомпании.

Функцией реестра авиакомпаний является ведение списка авиакомпаний. Функции владельца авиакомпании заключаются в ведение списков кассиров и касс. В функции работника авиакомпании входит введение списка клиентов, списка купонов и списка билетов, а выходными документами являются получение списков: билеты, проданные за указанный месяц указанной авиакомпании; общая сумма от продаж билетов каждой авиакомпании; список клиентов авиакомпаний на заданную дату.

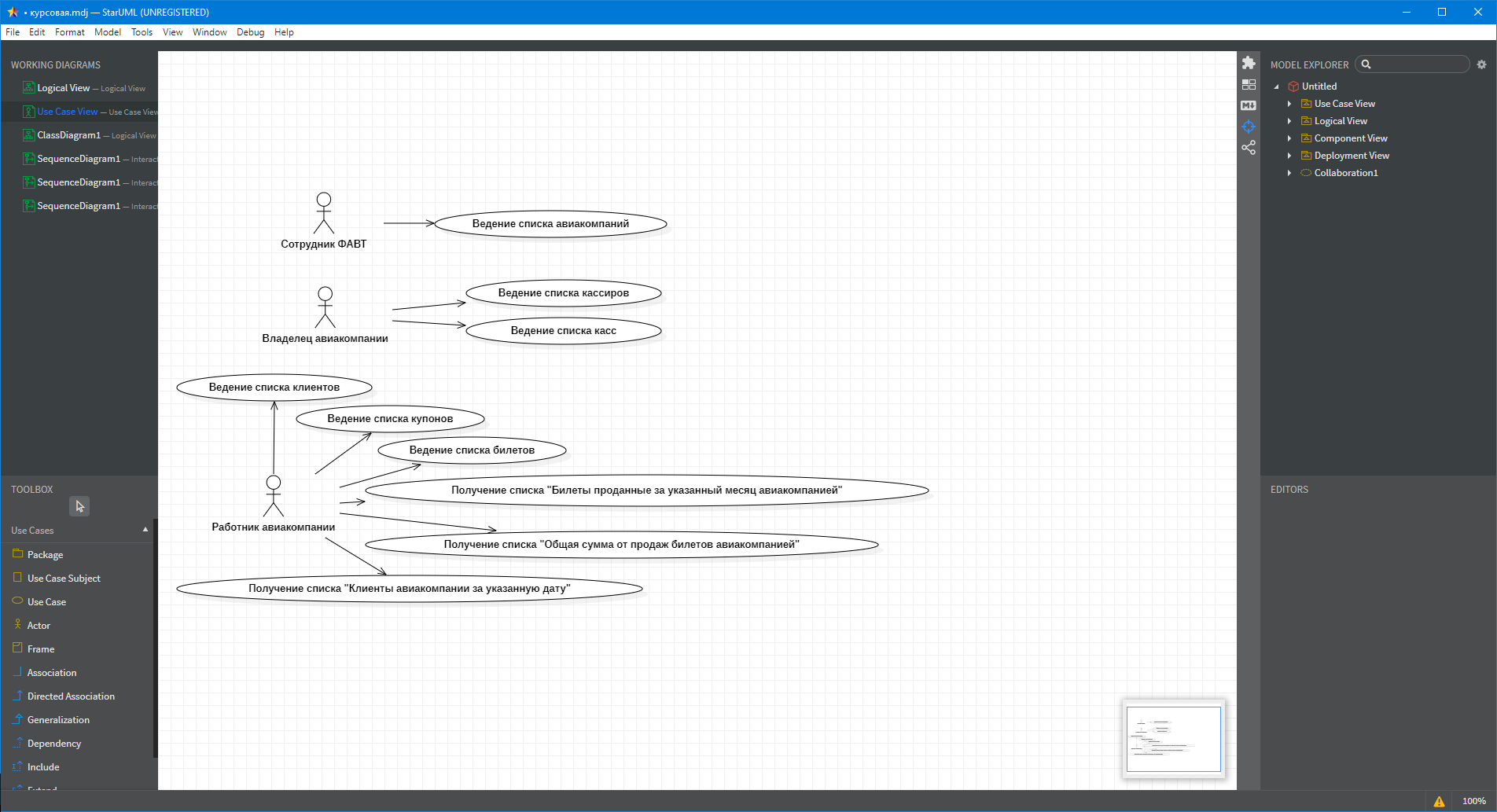


Рисунок 1.1 – Диаграмма сценариев (Use Case)

Далее создадим две диаграммы классов. Первая диаграмма, изображенная на рисунке 1.2, содержит классы интерфейса проектируемого приложения.

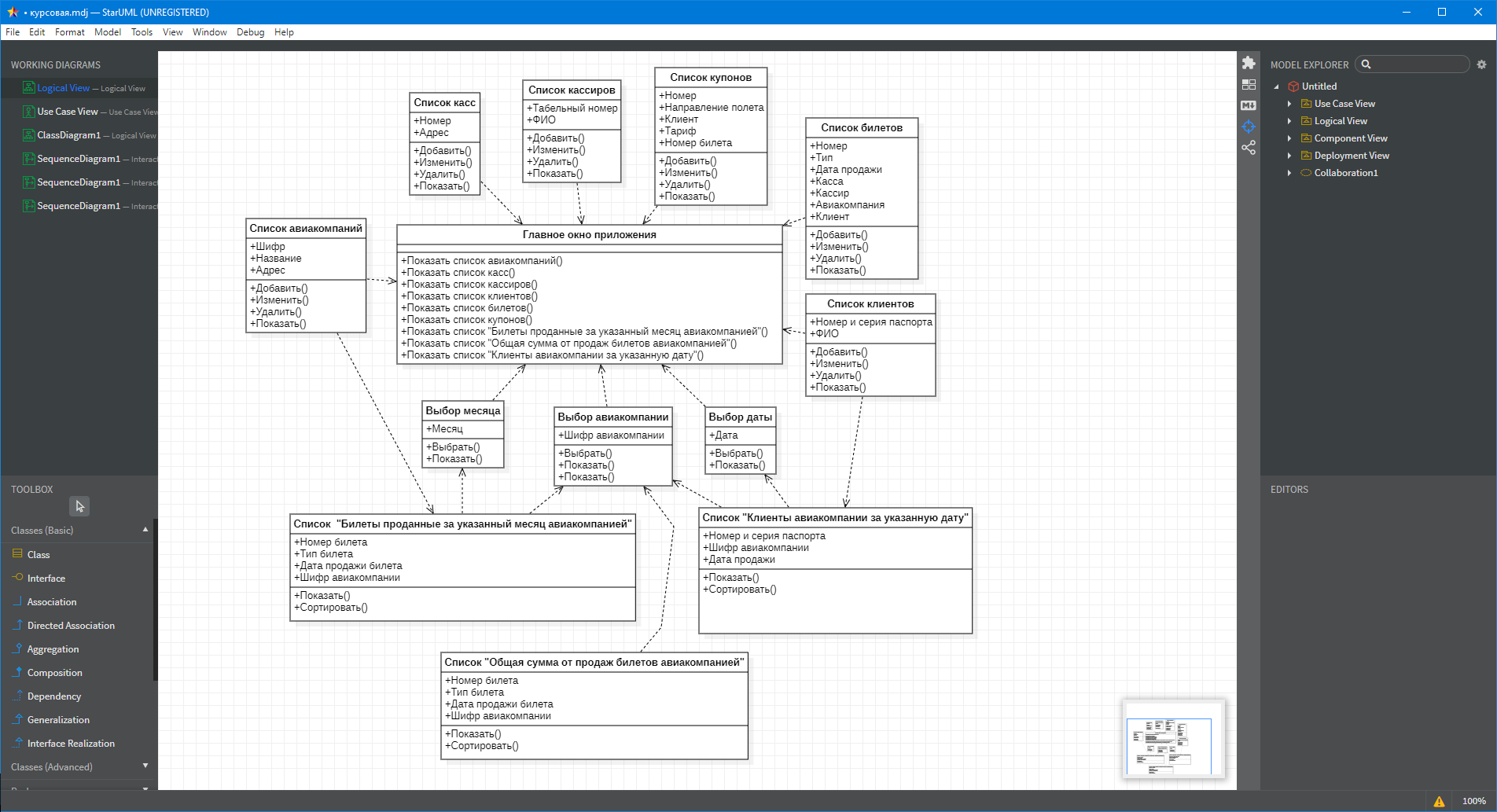


Рисунок 1.2 – Диаграмма классов (Интерфейс приложения)

Вторая диаграмма (рис. 1.3) – это данные, сущности базы данных. В построенной диаграмме классов отображены все формы будущего приложения и их взаимосвязь.

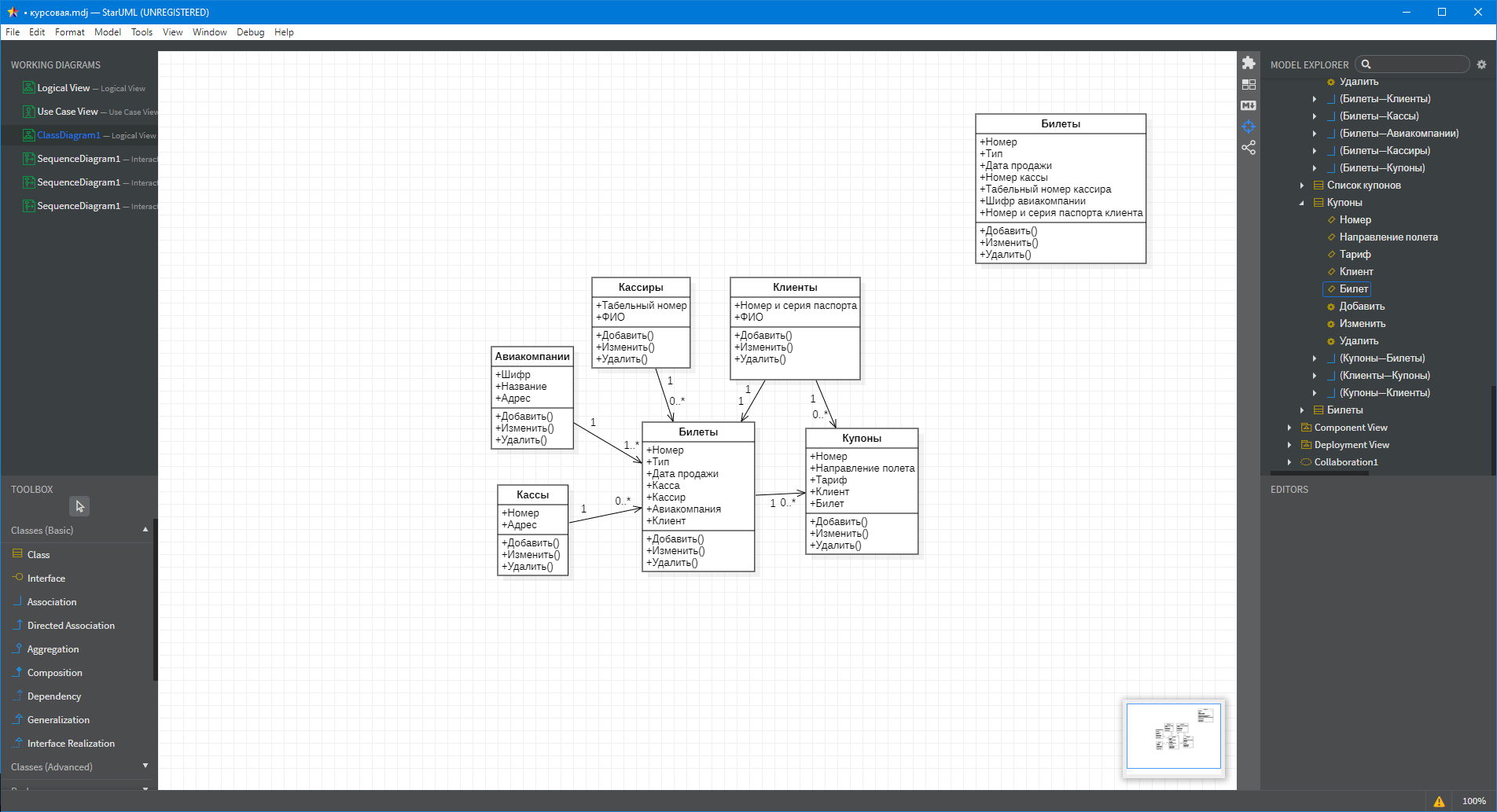


Рисунок 1.3 – Диаграмма классов (Данные)

Следующими диаграммами при построении объектной модели являются диаграммы последовательностей, изображенные на рис. 1.4 – 1.12. Диаграммы последовательностей создаются для каждого прецедента.

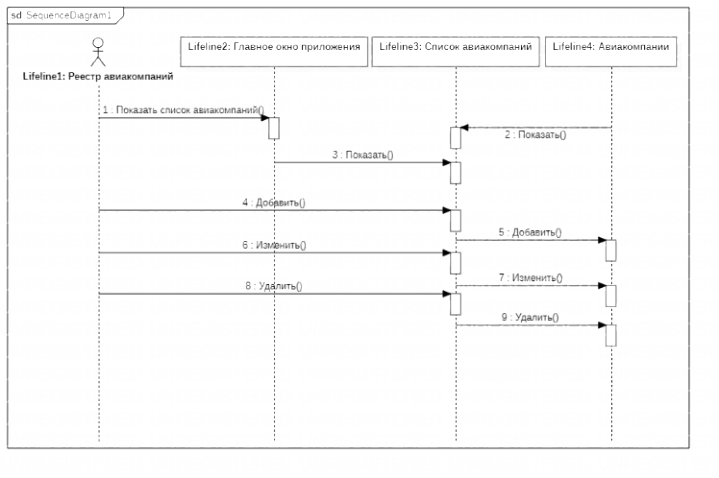


Рисунок 1.4 – Диаграмма последовательностей для прецедента «Ведение списка авиакомпаний»

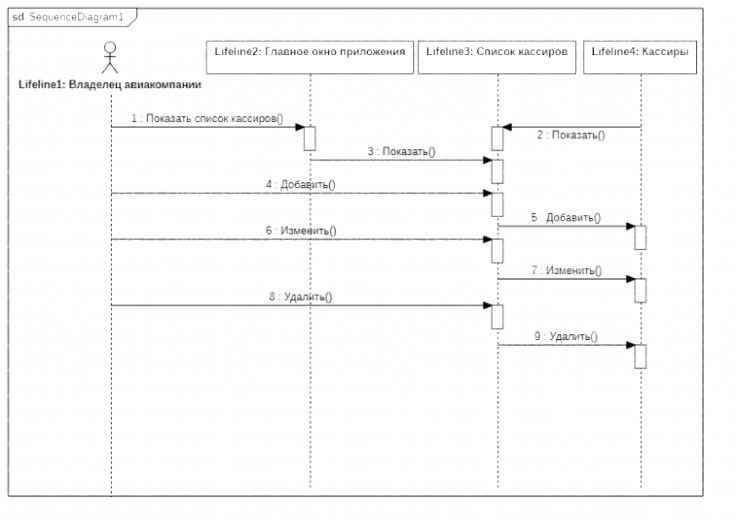


Рисунок 1.5 – Диаграмма последовательностей для прецедента «Ведение списка кассиров»

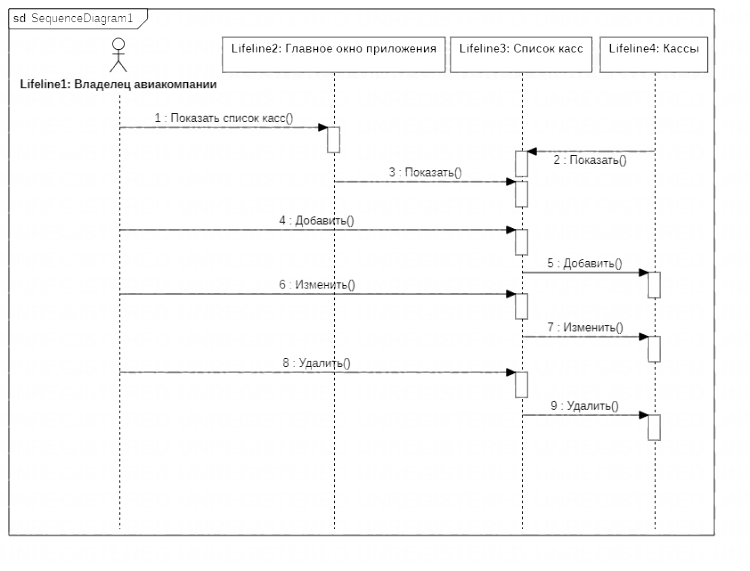


Рисунок 1.6 – Диаграмма последовательностей для прецедента «Ведение списка касс»



Рисунок 1.7 – Диаграмма последовательностей для прецедента «Ведение списка клиентов»

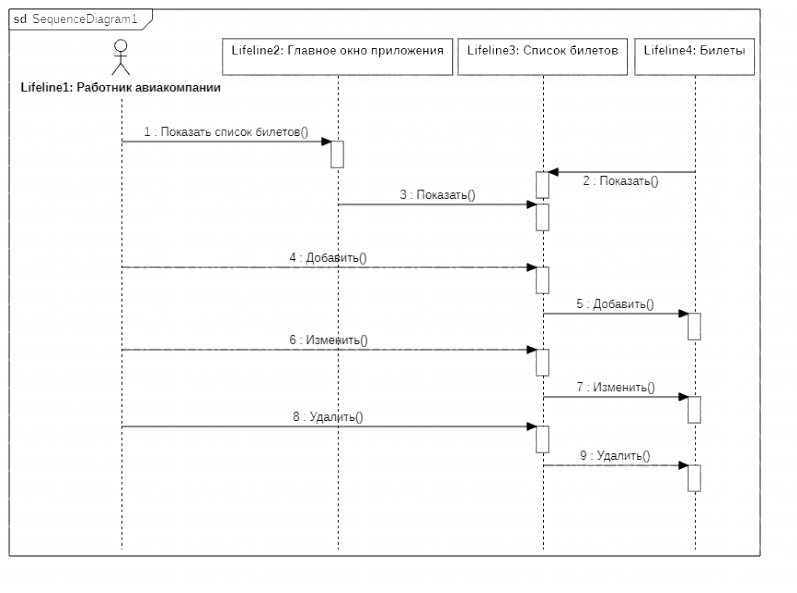


Рисунок 1.8 – Диаграмма последовательностей для прецедента «Ведение списка билетов»



Рисунок 1.9 – Диаграмма последовательностей для прецедента «Ведение списка купонов»

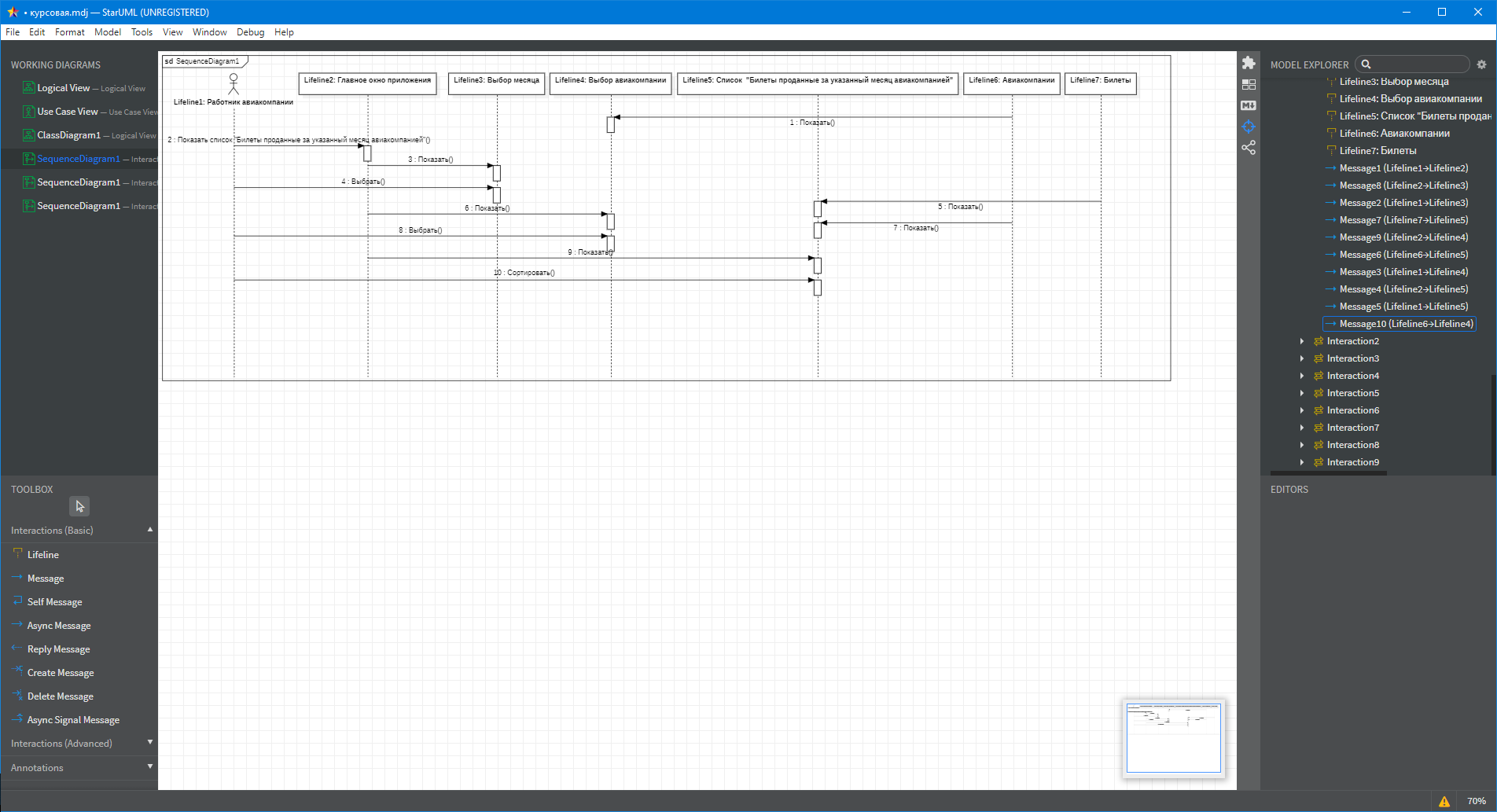


Рисунок 1.10 – Диаграмма последовательностей для прецедента «Получение списка "Билеты, проданные за указанный месяц авиакомпанией"»

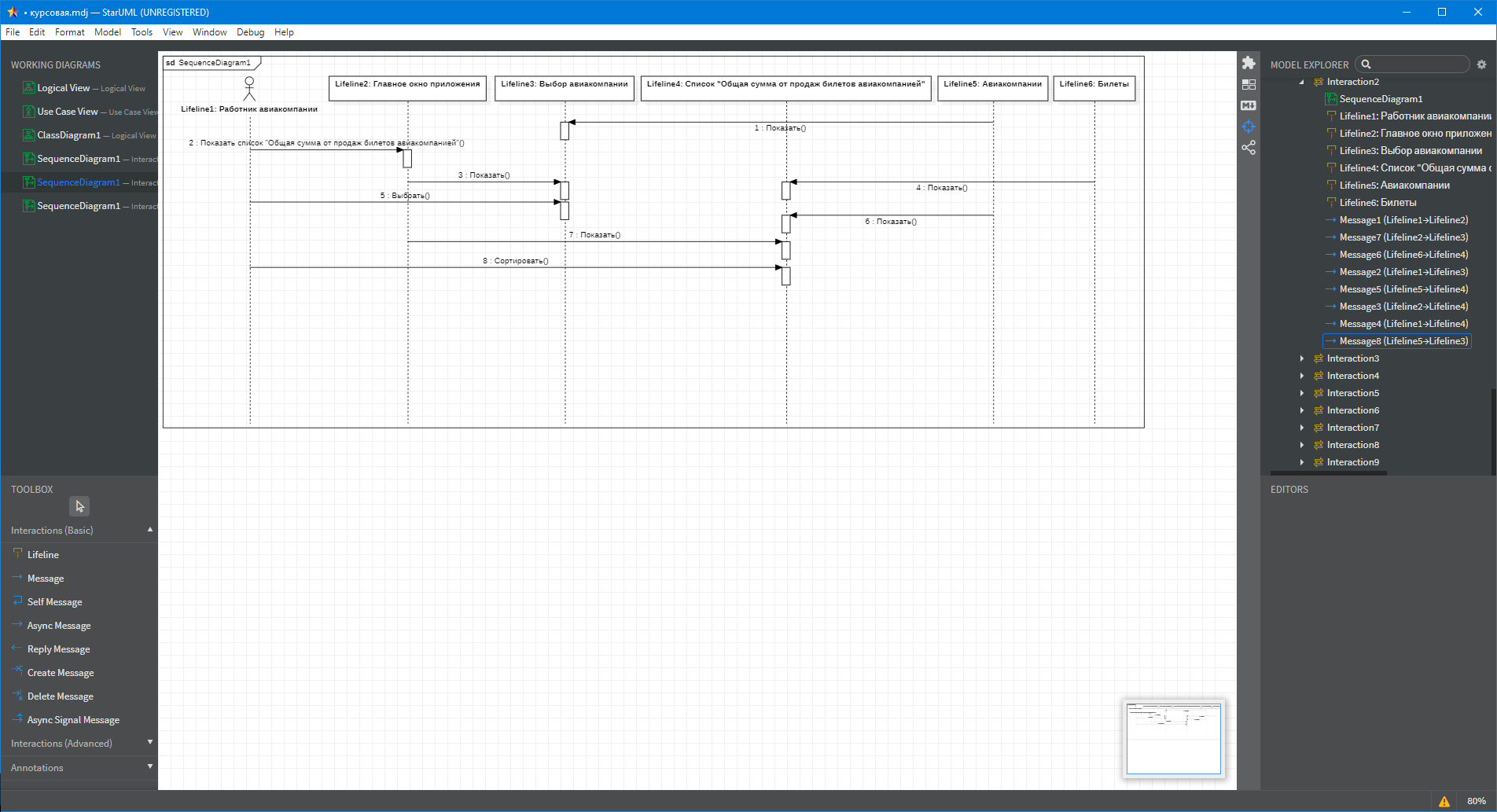


Рисунок 1.11 – Диаграмма последовательностей для прецедента «Получение списка "Общая сумма от продаж билетов авиакомпанией"»

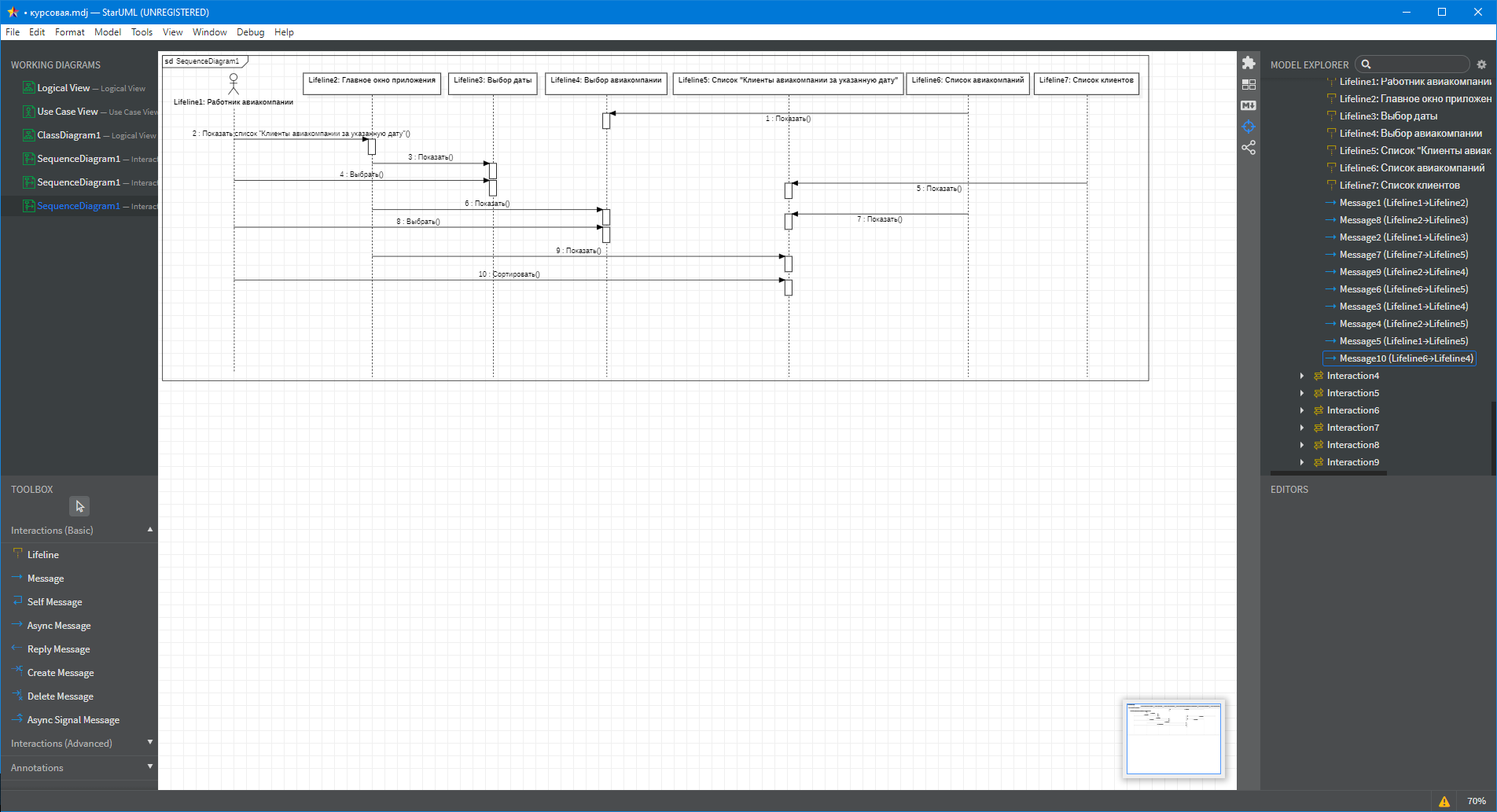


Рисунок 1.12 – Диаграмма последовательностей для прецедента «Получение списка "Клиенты авиакомпании за указанную дату"»

1. **Структура инфологической модели и результаты ее нормализации**

Для предметной области, определим сущности, описанные в задании:

1. Авиакомпания
2. Касса
3. Кассир
4. Клиент
5. Билет
6. Купон

Определим взаимосвязи пары сущностей, между которыми можно установить связь (рис. 2.1).

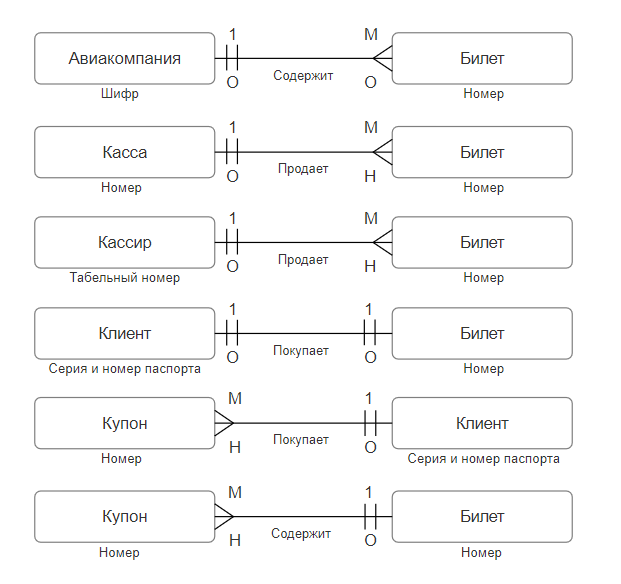


Рисунок 2.1 – Пары сущностей

Далее, согласно правилам преобразования ER-диаграмм, в отношения записываем набор получившихся отношений.

1. По четвертому правилу:

Авиакомпания (Шифр)

Билет (Номер, Шифр авиакомпании (FK))

1. По пятому правилу:

Касса (Номер)

Билет (Номер)

Список касс (Номер билета (FK), Номер кассы (FK))

1. По пятому правилу:

Кассир (Табельный номер)

Билет (Номер)

Список кассиров (Номер билета (FK), Табельный номер кассира (FK))

1. По первому правилу:

Клиент (Номер и серия паспорта)

Билет (Номер)

Список клиентов (Номер билета (FK), Номер и серия паспорта (FK))

1. По пятому правилу:

Купон (Номер)

Клиент (Номер и серия паспорта)

Список купонов (Номер купона (FK), Номер и серия паспорта (FK))

1. По пятому правилу:

Купон (Номер)

Билет (Номер)

Список билетов (Номер билета (FK), Номер купона (FK))

1НФ определение первичного ключа таблиц:

Авиакомпания (Шифр (PK), Название, Адрес)

Касса (Номер (PK), Адрес)

Кассир (Табельный номер (PK), ФИО)

Клиент (Номер и серия паспорта (PK), ФИО, Номер купона (FK))

Билет (Номер (PK), Тип, Дата продажи, Номер кассы (FK), Табельный номер кассира (FK), Шифр авиакомпании (FK), Номер и серия паспорта клиента (FK))

Купон (Номер (PK), Направление полета, Тариф, Номер билета (FK), Номер и серия паспорта клиента (FK))

2НФ выявление полей, функционально зависимых от части составного ключа:

Шифр авиакомпании –> Название авиакомпании

Шифр авиакомпании –> Индекс авиакомпании

Шифр авиакомпании –> Город авиакомпании

Шифр авиакомпании –> Улица авиакомпании

Шифр авиакомпании –> Дом авиакомпании

Номер кассы –> Индекс кассы

Номер кассы –> Город кассы

Номер кассы –> Улица кассы

Номер кассы –> Дом кассы

Табельный номер кассира –> Фамилия кассира

Табельный номер кассира –> Имя кассира

Табельный номер кассира –> Отчество кассира

Номер и серия паспорта клиента –> Фамилия клиента

Номер и серия паспорта клиента –> Имя клиента

Номер и серия паспорта клиента –> Отчество клиента

Номер билета –> Тип

Номер билета –> День продажи

Номер билета –> Месяц продажи

Номер билета –> Год продажи

Номер купона –> Направление полета

Номер купона –> Тариф

3НФ формирование таблиц (рис. 2.2):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Шифр авиакомпании (PK) |  | Номер билета (PK) |  | Номер и серия паспорта (PK) |
| Название авиакомпании  Индекс авиакомпании  Город авиакомпании  Улица авиакомпании  Дом авиакомпании |  | Тип  Номер и серия паспорта (FK)  Шифр авиакомпании (FK)  Номер кассы (FK)  Табельный номер кассира (FK)  День продажи  Месяц продажи  Год продажи |  | Фамилия клиента  Имя клиента  Отчество клиента |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер кассы (PK) |  | Табельный номер кассира (PK) |  | Номер купона (PK) |
| Индекс кассы  Город кассы  Улица кассы  Дом кассы |  | Фамилия кассира  Имя кассира  Отчество кассира |  | Направление  Тариф  Номер и серия паспорта клиента (FK)  Номер билета (FK) |

Рисунок 2.2 – Таблицы

1. **Логическая и физическая модели данных**
   1. **Логическая модель данных**

Логическое проектирование структуры базы данных выполняется на основе объектной модели задачи и иллюстрирует сущности, а также их взаимоотношения между собой. Целью построения логической модели является получение графического представления логической структуры исследуемой предметной области. На рисунке 3.1 представлена логическая модель базы данных, выполненная с помощью DBDesigner.

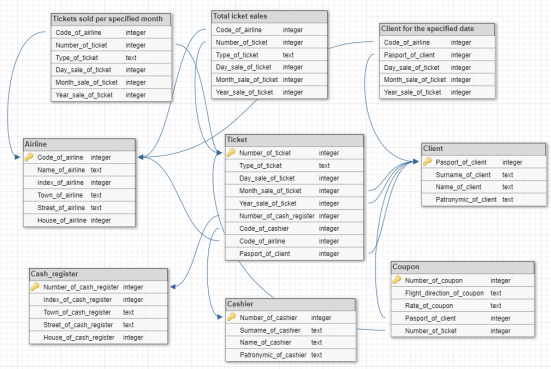


Рисунок 3.1 – Логическая модель данных

* 1. **Физическая модель данных**

Для клиент-серверного приложения, на основании спроектированной логической модели, генерируется физическая модель данных. Физическая модель базы данных определяет, каким образом представляются данные, и содержит все детали, необходимые СУБД для создания базы данных. На рисунке 3.2 представлена физическая модель базы данных.



Рисунок 3.2 – Физическая модель данных

1. **Описание базы данных на сервере**

При выполнении данной курсовой работы, база данных имеет вид и состоит из таблиц:

1. Airline – содержит данные об авиакомпаниях.
2. Cash\_register – содержит данные о кассах.
3. Cashier – содержит данные о кассирах.
4. Client – содержит данные о клиентах.
5. Coupon – содержит данные о купонах.
6. Ticket – содержит данные о билетах.

Таблица Airline содержит: Code\_of\_airline (Шифр авиакомпании), Name\_of\_airline (Название авиакомпании), Index\_of\_airline (Индекс, часть адреса авиакомпании), Town\_of\_airline (Город, часть адреса авиакомпании), Street\_of\_airline (Улица, часть адреса авиакомпании), House\_of\_airline (Дом, часть адреса авиакомпании). Первичным ключам в таблице является Code\_of\_airline (Шифр авиакомпании). Index\_of\_airline (Индекс, часть адреса авиакомпании) является уникальным.

Листинг таблицы Airline:

CREATE TABLE [dbo].[Airline] (

[Code\_of\_airline] INT NOT NULL,

[Name\_of\_airline] NVARCHAR (50) NOT NULL,

[Index\_of\_airline] INT NOT NULL,

[Town\_of\_airline] NVARCHAR (50) NULL,

[Street\_of\_airline] NVARCHAR (50) NULL,

[House\_of\_airline] NVARCHAR (50) NULL,

PRIMARY KEY CLUSTERED ([Code\_of\_airline] ASC),

UNIQUE NONCLUSTERED ([Index\_of\_airline] ASC)

);

Таблица Cash\_register содержит: Number\_of\_cash\_register (Номер кассы), Index\_of\_cash\_register (Индекс, часть адреса кассы), Town\_of\_cash\_register (Город, часть адреса кассы), Street\_of\_cash\_register (Улица, часть адреса кассы), House\_of\_cash\_register (Дом, часть адреса авиакомпании). Первичным ключам в таблице является Number\_of\_cash\_register (Номер кассы). Index\_of\_cash\_register (Индекс, часть адреса кассы) является уникальным.

Листинг таблицы Cash\_register:

CREATE TABLE [dbo].[Cash\_register] (

[Number\_of\_cash\_register] INT NOT NULL,

[Index\_of\_cash\_register] INT NOT NULL,

[Town\_of\_cash\_register] NVARCHAR (50) NULL,

[Street\_of\_cash\_register] NVARCHAR (50) NULL,

[House\_of\_cash\_register] NVARCHAR (50) NULL,

PRIMARY KEY CLUSTERED ([Number\_of\_cash\_register] ASC),

UNIQUE NONCLUSTERED ([Index\_of\_cash\_register] ASC)

);

Таблица Cashier содержит: Code\_of\_cashier (Табельный номер кассира), Surname\_of\_cashier (Фамилия кассира), Name\_of\_cashier (Имя кассира), Patronymic\_of\_cashier (Отчество кассира). Первичным ключам в таблице является Code\_of\_cashier (Табельный номер кассира).

Листинг таблицы Cashier:

CREATE TABLE [dbo].[Cashier] (

[Code\_of\_cashier] INT NOT NULL,

[Surname\_of\_cashier] NVARCHAR (50) NOT NULL,

[Name\_of\_cashier] NVARCHAR (50) NOT NULL,

[Patronymic\_of\_cashier] NVARCHAR (50) NULL,

PRIMARY KEY CLUSTERED ([Code\_of\_cashier] ASC)

);

Таблица Client содержит: Pasport\_of\_client (Номер и серия паспорта клиента), Surname\_of\_client (Фамилия клиента), Name\_of\_client (Имя клиента), Patronymic\_of\_client (Отчество клиента). Первичным ключам в таблице является Pasport\_of\_client (Номер и серия паспорта клиента).

Листинг таблицы Client:

CREATE TABLE [dbo].[Client] (

[Pasport\_of\_client] NVARCHAR (50) NOT NULL,

[Surname\_of\_client] NVARCHAR (50) NOT NULL,

[Name\_of\_client] NVARCHAR (50) NOT NULL,

[Patronymic\_of\_client] NVARCHAR (50) NULL,

PRIMARY KEY CLUSTERED ([Pasport\_of\_client] ASC)

);

Таблица Coupon содержит: Number\_of\_coupon (Номер купона), Flight\_direction\_of\_coupon (Направление полета), Rate\_of\_coupon (Тариф купона), Pasport\_of\_client (Номер и серия паспорта клиента), Number\_of\_ticket (Номер билета). Первичным ключам в таблице является Number\_of\_coupon (Номер купона). Внешними ключами являются: Pasport\_of\_client (Номер и серия паспорта клиента) и Number\_of\_ticket (Номер билета).

Листинг таблицы Coupon:

CREATE TABLE [dbo].[Coupon] (

[Number\_of\_coupon] INT NOT NULL,

[Flight\_direction\_of\_coupon] NVARCHAR (50) NOT NULL,

[Rate\_of\_coupon] NVARCHAR (50) NOT NULL,

[Pasport\_of\_client] NVARCHAR (50) NOT NULL,

[Number\_of\_ticket] INT NOT NULL,

PRIMARY KEY CLUSTERED ([Number\_of\_coupon] ASC),

CONSTRAINT [FK\_Coupon\_ToClient] FOREIGN KEY ([Pasport\_of\_client]) REFERENCES [dbo].[Client] ([Pasport\_of\_client]) ON DELETE CASCADE,

CONSTRAINT [FK\_Coupon\_ToTicket] FOREIGN KEY ([Number\_of\_ticket]) REFERENCES [dbo].[Ticket] ([Number\_of\_ticket]) ON DELETE CASCADE

);

Таблица Ticket содержит: Number\_of\_ticket (Номер билета), Type\_of\_ticket (Тип билета), Day\_sale\_of\_ticket (День продажи билета), Month\_sale\_of\_ticket (Месяц продажи билета), Year\_sale\_of\_ticket (Год продажи билета), Number\_of\_cash\_register (Номер кассы), Code\_of\_cashier (Табельный номер кассира), Code\_of\_airline (Шифр авиакомпании) и Pasport\_of\_client (Номер и серия паспорта клиента). Первичным ключам в таблице является Number\_of\_ticket (Номер билета). Внешними ключами являются: Number\_of\_cash\_register (Номер кассы), Code\_of\_cashier (Табельный номер кассира), Code\_of\_airline (Шифр авиакомпании) и Pasport\_of\_client (Номер и серия паспорта клиента). Проверочные ограничения для Day\_sale\_of\_ticket (День продажи билета) от 1 до 31, для Month\_sale\_of\_ticket (Месяц продажи билета) от 1 до 12 и для Year\_sale\_of\_ticket (Год продажи билета) от 2000 до 2023.

Листинг таблицы Ticket:

CREATE TABLE [dbo].[Ticket] (

[Number\_of\_ticket] INT NOT NULL,

[Type\_of\_ticket] NVARCHAR (50) NOT NULL,

[Day\_sale\_of\_ticket] INT NOT NULL,

[Month\_sale\_of\_ticket] INT NOT NULL,

[Year\_sale\_of\_ticket] INT NOT NULL,

[Number\_of\_cash\_register] INT NOT NULL,

[Code\_of\_cashier] INT NOT NULL,

[Code\_of\_airline] INT NOT NULL,

[Pasport\_of\_client] NVARCHAR (50) NOT NULL,

PRIMARY KEY CLUSTERED ([Number\_of\_ticket] ASC),

CONSTRAINT [FK\_Ticket\_ToClient] FOREIGN KEY ([Pasport\_of\_client]) REFERENCES [dbo].[Client] ([Pasport\_of\_client]),

CONSTRAINT [FK\_Ticket\_ToCash\_register] FOREIGN KEY ([Number\_of\_cash\_register]) REFERENCES [dbo].[Cash\_register] ([Number\_of\_cash\_register]) ON DELETE CASCADE,

CONSTRAINT [FK\_Ticket\_ToCashier] FOREIGN KEY ([Code\_of\_cashier]) REFERENCES [dbo].[Cashier] ([Code\_of\_cashier]) ON DELETE CASCADE,

CONSTRAINT [FK\_Ticket\_ToAirline] FOREIGN KEY ([Code\_of\_airline]) REFERENCES [dbo].[Airline] ([Code\_of\_airline]) ON DELETE CASCADE,

CONSTRAINT [CK\_Ticket\_Day\_sale\_of\_ticket] CHECK ([Day\_sale\_of\_ticket] <= (31)

AND [Day\_sale\_of\_ticket] >= (1)),

CONSTRAINT [CK\_Ticket\_Month\_sale\_of\_ticket] CHECK ([Month\_sale\_of\_ticket] <= (12)

AND [Month\_sale\_of\_ticket] >= (1)),

CONSTRAINT [CK\_Ticket\_Year\_sale\_of\_ticket] CHECK ([Year\_sale\_of\_ticket] <= (2023)

AND [Year\_sale\_of\_ticket] >= (2000))

);

Следует сделать необходимые изменения и дополнения к серверной части, а именно:

1. Генераторы – это специальный объект базы данных, который генерирует уникальные последовательные числа. Эти числа могут быть использованы в качестве идентификаторов. В данной работе генераторы не подлежали изменениям и дополнениям.
2. Ошибки всех видов интерпретируются как исключения – ситуации, которые не должны возникать при нормальном выполнении программы. В данной работе исключения не подлежали изменениям и дополнениям.
3. Хранимые процедуры – фрагмент программного кода, который хранится на сервере базы данных и выполняется по запросу клиента. В данной работе хранимые процедуры не подлежали изменениям и дополнениям.
4. Представления, или просмотры, представляют собой временные, производные таблицы и являются объектами базы данных, информация в которых не хранится постоянно, как в базовых таблицах, а формируется динамически при обращении к ним. В данной работе представлениями были определены получаемые списки.
5. **Формы входных и выходных документов**

Копии экранных форм входных и выходных документов представлены на рисунках 5.1 – 5.9.

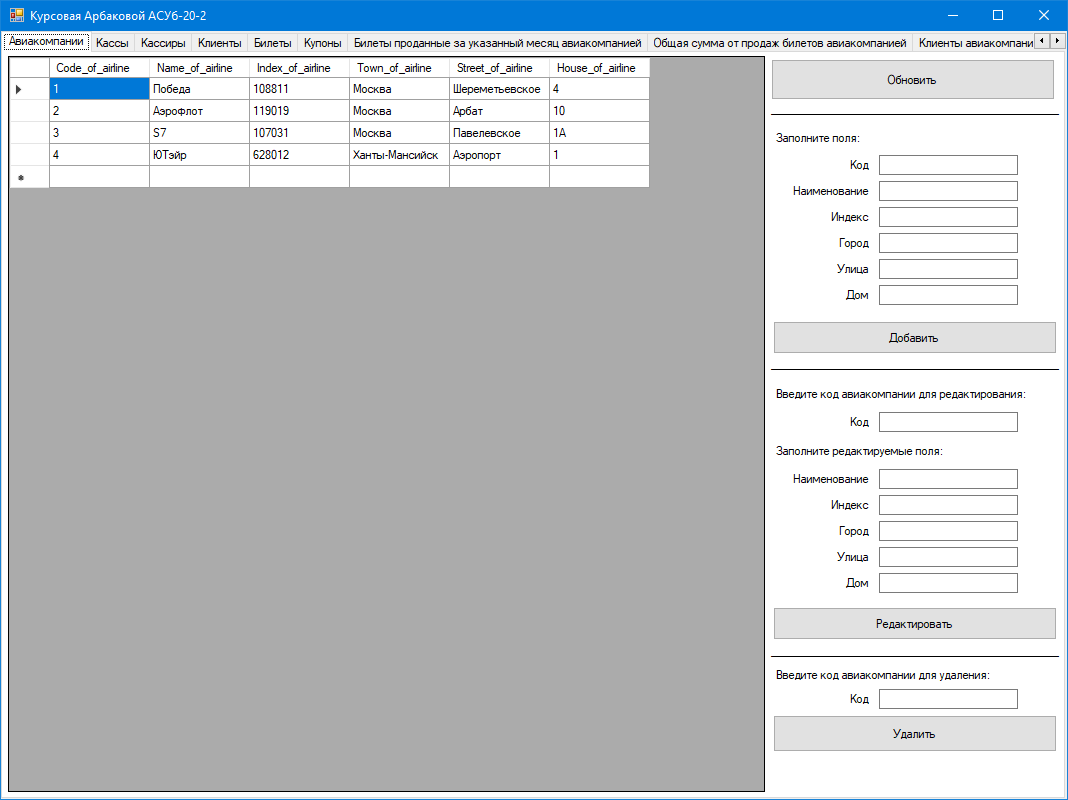


Рисунок 5.1 – Экранная форма «Авиакомпании»

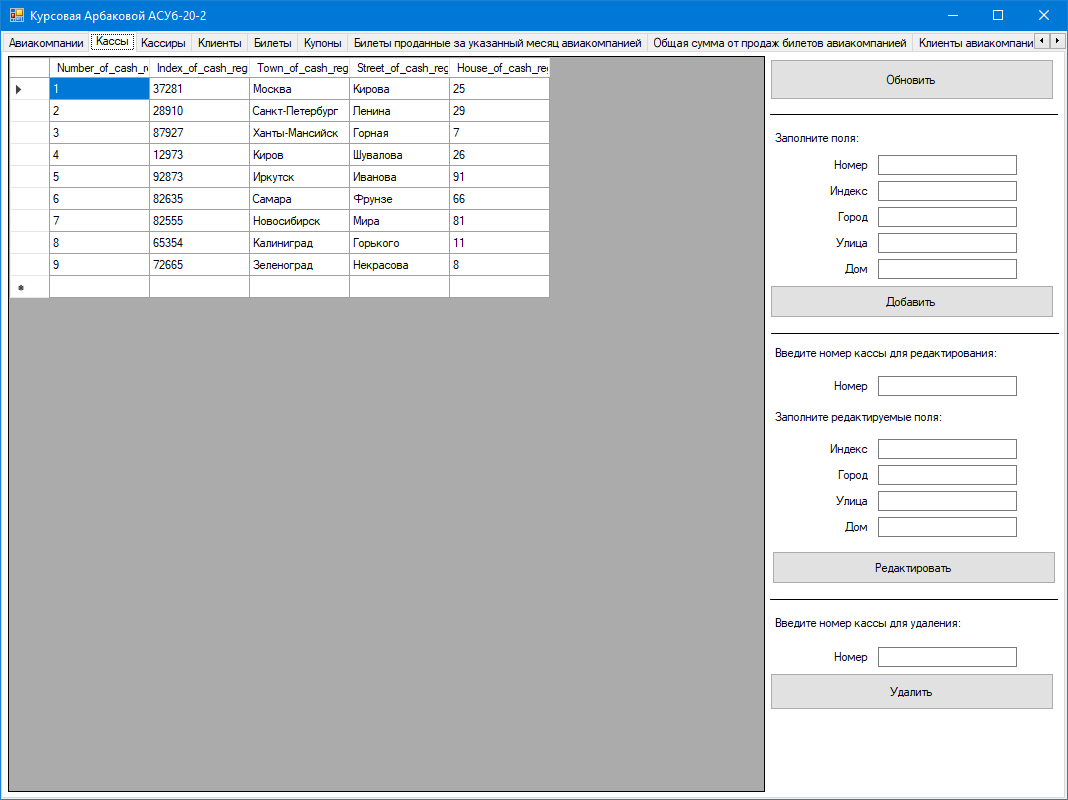


Рисунок 5.2 – Экранная форма «Кассы»

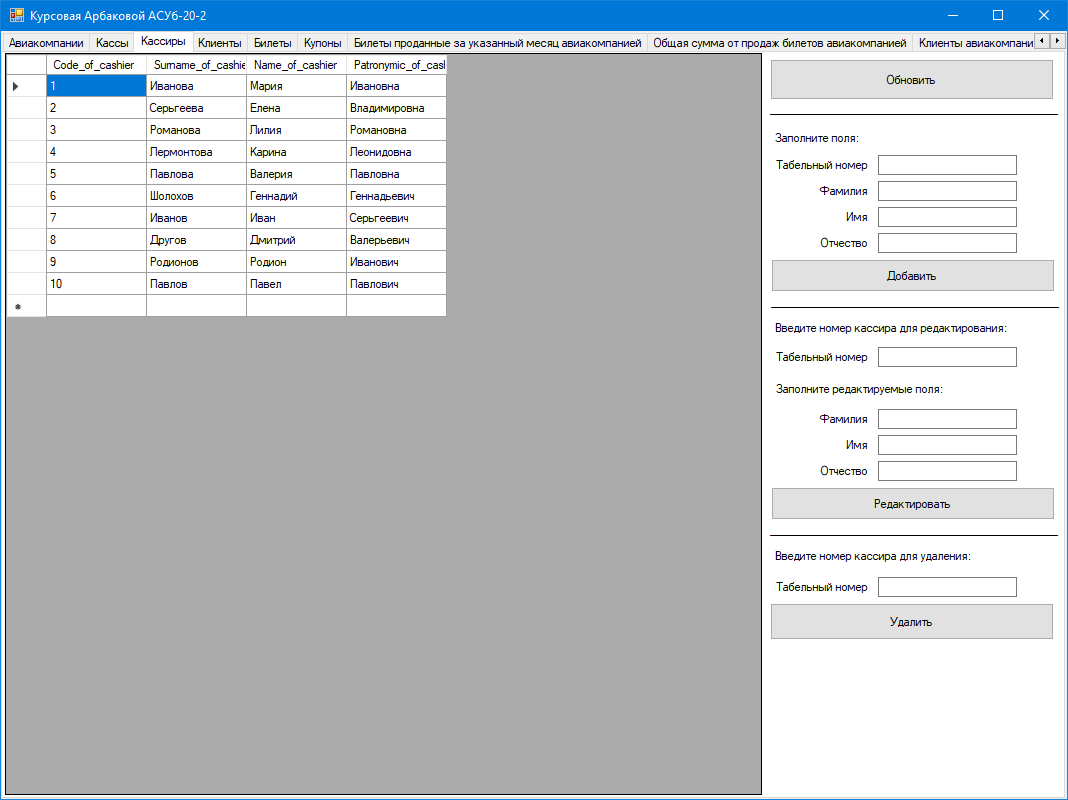


Рисунок 5.3 – Экранная форма «Кассиры»

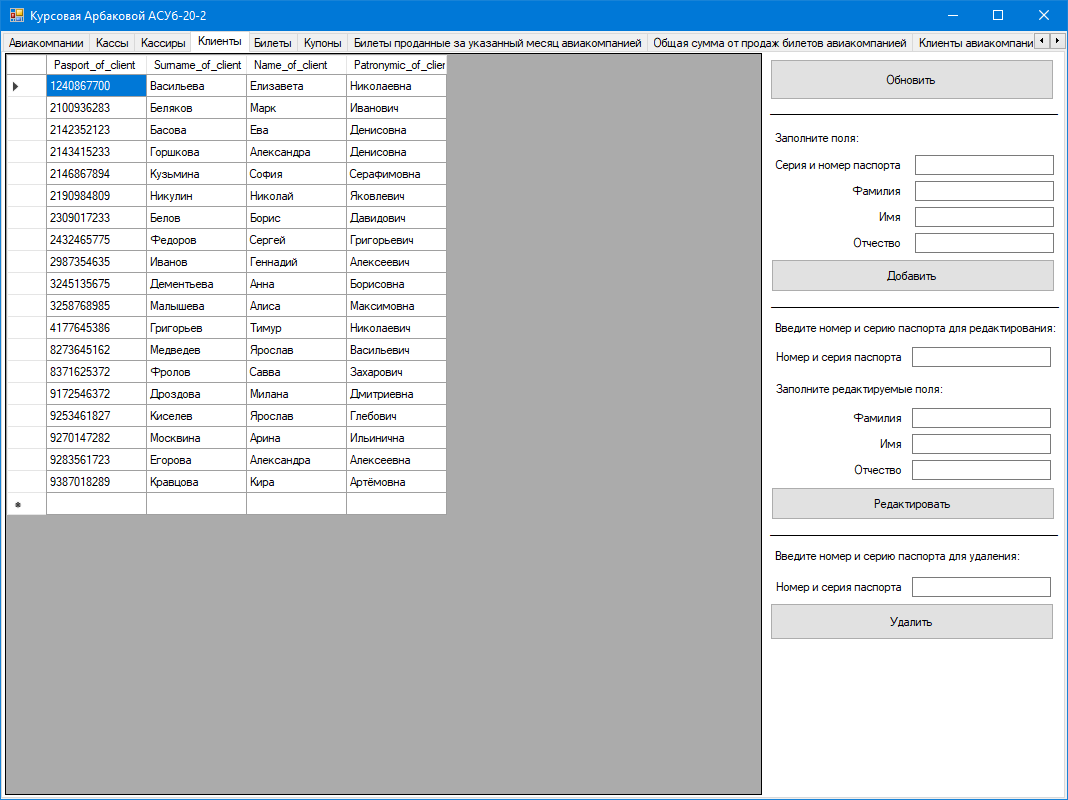


Рисунок 5.4 – Экранная форма «Клиенты»

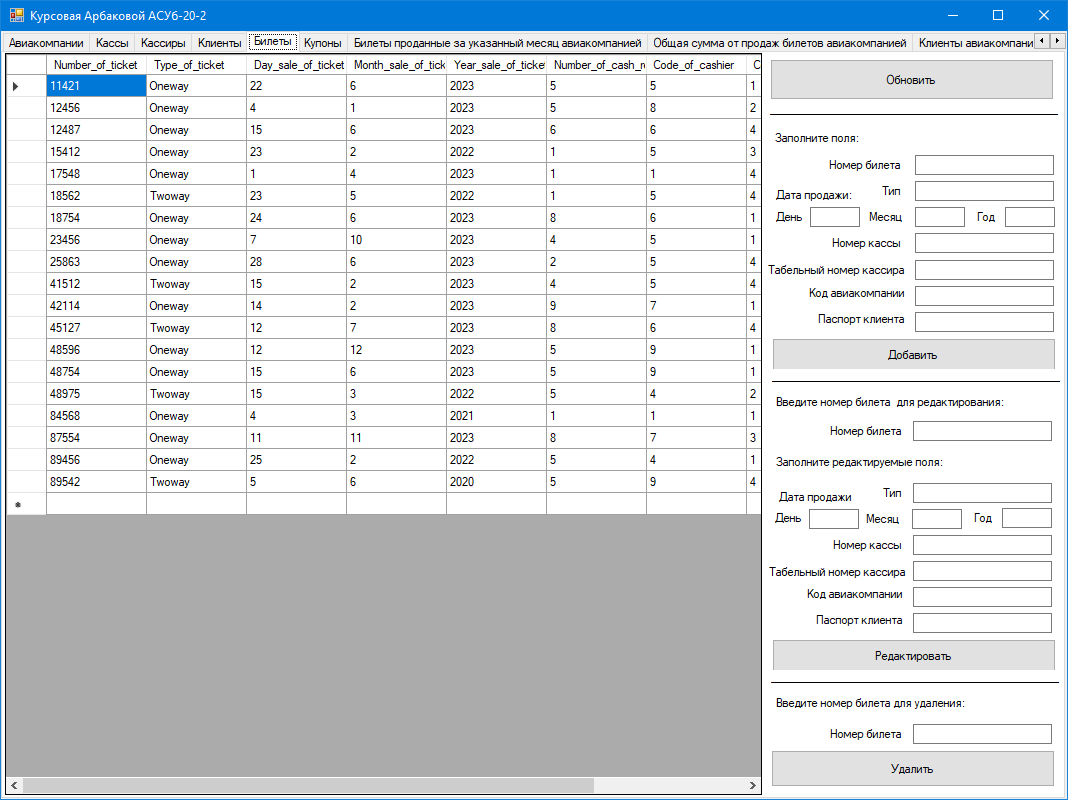


Рисунок 5.5 – Экранная форма «Билеты»

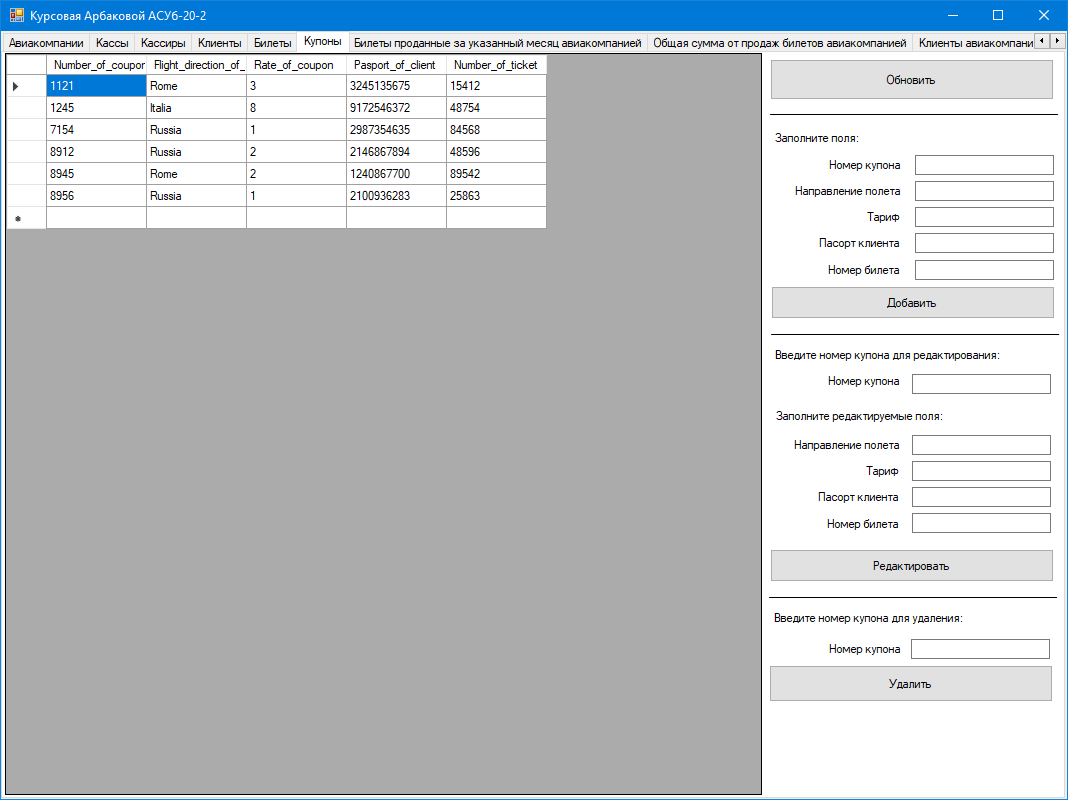


Рисунок 5.6 – Экранная форма «Купоны»

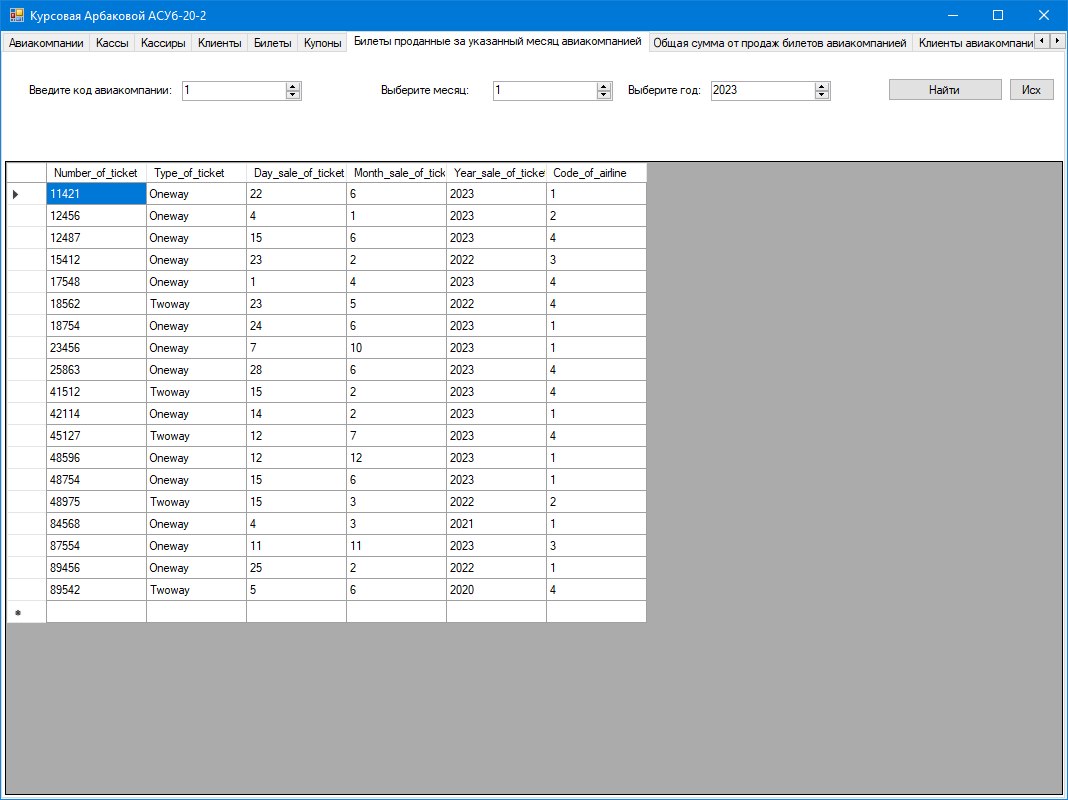


Рисунок 5.7 – Экранная форма «Билеты, проданные за указанный месяц авиакомпанией»

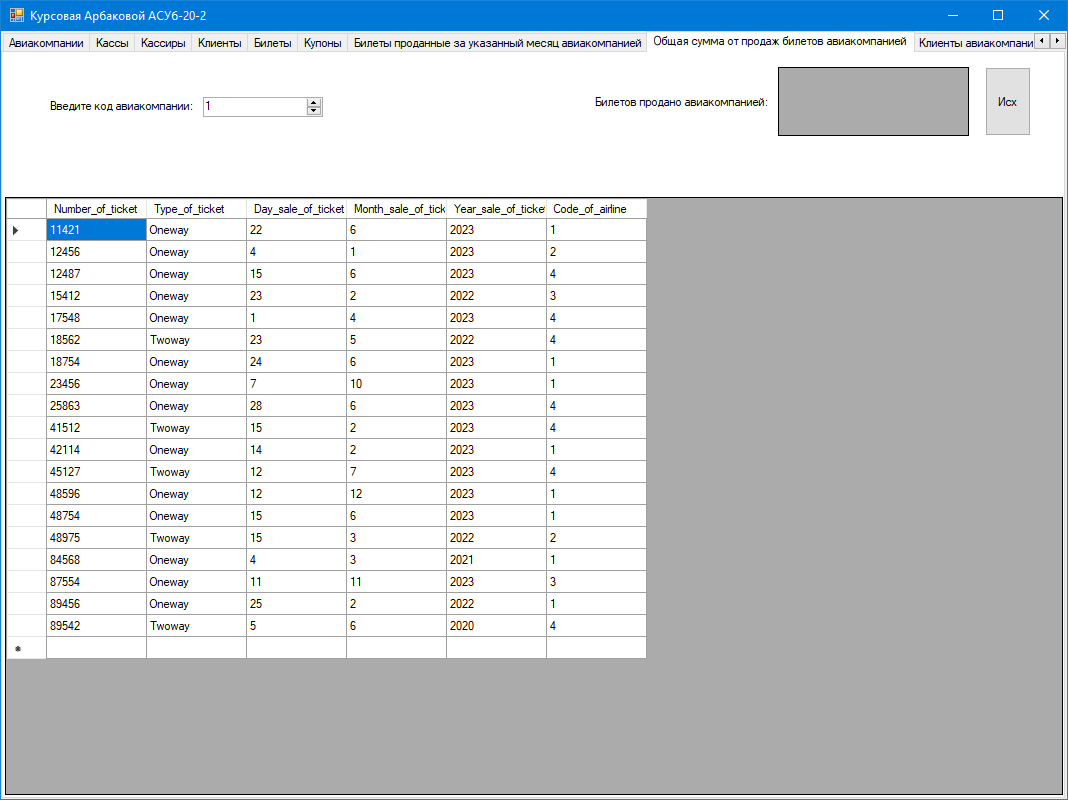


Рисунок 5.8 – Экранная форма «Общая сумма от продаж билетов авиакомпанией»

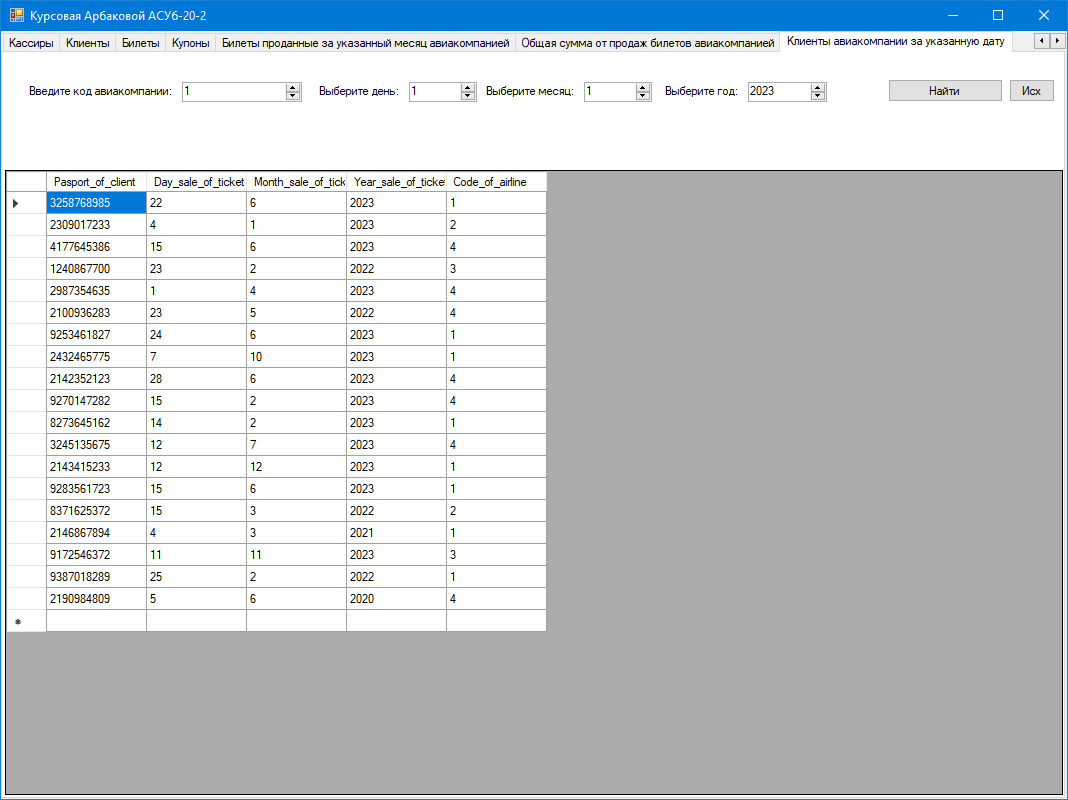


Рисунок 5.9 – Экранная форма «Клиенты авиакомпании за указанную дату»

1. **Инструкция пользователя**
2. Запуск

При запуске программы, если база данных подключена верно, то нажмите кнопку «ОК» в диалоговом окне с текстом «Подключение установлено!», иначе проверьте подключение базы данных.

1. Экранная форма «Авиакомпании»

Следом откроется окно «Курсовая» с открытой экранной формой «Авиакомпании» (рис. 5.1). В ограждённом слева поле появятся данные об авиакомпаниях, включающие в себя шифр авиакомпании, название авиакомпании, индекс, город, улица и дом адреса авиакомпании. В правой части формы расположены кнопки «Обновить», «Добавить», «Редактировать» и «Удалить» с соответствующими им полями ввода, разделенные горизонтальными линиями на разделы.

* 1. Кнопка «Обновить»

При нажатии кнопки «Обновить» ограждённое поле слева с данными обновится.

* 1. Кнопка «Добавить»

Для ввода новой записи в базу данных авиакомпаний следует ввести данные в текстовые поля правее от: «Код», «Наименование», «Индекс», «Город», «Улица» и «Дом», расположенные ниже текста «Заполните поля:», и нажать кнопку «Добавить». При успешном заполнении текстовых полей и нажатии кнопки «Добавить» откроется диалоговое окно с текстом «Строка успешно добавлена!», иначе при наличии ошибок «Входная строка имела неверный формат».

* 1. Кнопка «Редактировать»

Для редактирования строки в базе данных авиакомпаний следует ввести «Код» авиакомпании в соответствующее ему правее текстовое поле, и заполнить поля «Наименование», «Индекс», «Город», «Улица» и «Дом» авиакомпании, и нажать «Редактировать». При успешном заполнении текстовых полей и нажатии кнопки «Редактировать» откроется диалоговое окно с текстом «Строка успешно отредактирована!», иначе при наличии ошибок «Входная строка имела неверный формат».

* 1. Кнопка «Удалить»

Для удаления строки в базе данных авиакомпаний следует ввести «Код» авиакомпании в соответствующее ему правее текстовое поле, и нажать кнопку «Удалить». При успешном заполнении текстового поля и нажатии кнопки «Удалить» откроется диалоговое окно с текстом «Строка успешно удалена!», иначе при наличии ошибок «Входная строка имела неверный формат».

1. Экранная форма «Кассы»

При выборе пункта «Кассы», в верхней панели, откроется окно формы «Кассы» (рис. 5.2). В ограждённом слева поле появятся данные о кассах, включающие в себя номер кассы, индекс, город, улица и дом адреса кассы. В правой части формы расположены кнопки «Обновить», «Добавить», «Редактировать» и «Удалить» с соответствующими им полями ввода, разделенные горизонтальными линиями на разделы.

* 1. Кнопка «Обновить» – выполняется аналогично пункту 2.1.
  2. Кнопка «Добавить» – выполняется аналогично пункту 2.2.
  3. Кнопка «Редактировать» – выполняется аналогично пункту 2.3.
  4. Кнопка «Удалить» – выполняется аналогично пункту 2.4.

1. Экранная форма «Кассиры»

При выборе пункта «Кассиры», в верхней панели, откроется окно формы «Кассиры» (рис. 5.3). В ограждённом слева поле появятся данные о кассирах, включающие в себя табельный номер кассира, фамилию, имя и отчество кассира. В правой части формы расположены кнопки «Обновить», «Добавить», «Редактировать» и «Удалить» с соответствующими им полями ввода, разделенные горизонтальными линиями на разделы.

* 1. Кнопка «Обновить» – выполняется аналогично пункту 2.1.
  2. Кнопка «Добавить» – выполняется аналогично пункту 2.2.
  3. Кнопка «Редактировать» – выполняется аналогично пункту 2.3.
  4. Кнопка «Удалить» – выполняется аналогично пункту 2.4.

1. Экранная форма «Клиенты»

При выборе пункта «Клиенты», в верхней панели, откроется окно формы «Клиенты» (рис. 5.4). В ограждённом слева поле появятся данные о клиентах, включающие в себя номер и серию паспорта, фамилию, имя и отчество клиента. В правой части формы расположены кнопки «Обновить», «Добавить», «Редактировать» и «Удалить» с соответствующими им полями ввода, разделенные горизонтальными линиями на разделы.

* 1. Кнопка «Обновить» – выполняется аналогично пункту 2.1.
  2. Кнопка «Добавить» – выполняется аналогично пункту 2.2.
  3. Кнопка «Редактировать» – выполняется аналогично пункту 2.3.
  4. Кнопка «Удалить» – выполняется аналогично пункту 2.4.

1. Экранная форма «Билеты»

При выборе пункта «Билеты», в верхней панели, откроется окно формы «Билеты» (рис. 5.5). В ограждённом слева поле появятся данные о билетах, включающие в себя номер билета, тип билета, дату продажи билета (день, месяц, год), номер кассы, номер кассира, шифр авиакомпании. В правой части формы расположены кнопки «Обновить», «Добавить», «Редактировать» и «Удалить» с соответствующими им полями ввода, разделенные горизонтальными линиями на разделы.

* 1. Кнопка «Обновить» – выполняется аналогично пункту 2.1.
  2. Кнопка «Добавить» – выполняется аналогично пункту 2.2.
  3. Кнопка «Редактировать» – выполняется аналогично пункту 2.3.
  4. Кнопка «Удалить» – выполняется аналогично пункту 2.4.

1. Экранная форма «Купоны»

При выборе пункта «Купоны», в верхней панели, откроется окно формы «Купоны» (рис. 5.6). В ограждённом слева поле появятся данные о купонах, включающие в себя номер купона, направление полета, тариф, паспорт клиента и номер билета. В правой части формы расположены кнопки «Обновить», «Добавить», «Редактировать» и «Удалить» с соответствующими им полями ввода, разделенные горизонтальными линиями на разделы.

* 1. Кнопка «Обновить» – выполняется аналогично пункту 2.1.
  2. Кнопка «Добавить» – выполняется аналогично пункту 2.2.
  3. Кнопка «Редактировать» – выполняется аналогично пункту 2.3.
  4. Кнопка «Удалить» – выполняется аналогично пункту 2.4.

1. Экранная форма «Билеты, проданные за указанный месяц авиакомпанией»

При выборе пункта «Билеты, проданные за указанный месяц авиакомпанией», в верхней панели, откроется окно формы «Билеты, проданные за указанный месяц авиакомпанией» (рис. 5.7). В ограждённом ниже поле появятся данные о номере билета, типе, дате продажи и шифре авиакомпании.

Выберете код авиакомпании, месяц и год в соответствующих полях, и нажмите кнопку «Найти». При успешном вводе и нажатии кнопки база данных соответствующе обновится. При нажатии кнопки «Исх» база данных обновится в первоначальный вид.

1. Экранная форма «Общая сумма от продаж билетов авиакомпанией»

При выборе пункта «Общая сумма от продаж билетов авиакомпанией», в верхней панели, откроется окно формы «Общая сумма от продаж билетов авиакомпанией» (рис. 5.8). В ограждённом ниже поле появятся данные о номере билета, типе, дате продажи и шифре авиакомпании.

Выберете код авиакомпании в соответствующем поле, и нажмите кнопку «Найти». При успешном вводе и нажатии кнопки база данных соответствующе обновится, а поле правее от «Билетов продано авиакомпанией:» отобразит количество проданных билетов. При нажатии кнопки «Исх» база данных обновится в первоначальный вид.

1. Экранная форма «Клиенты авиакомпании за указанную дату»

При выборе пункта «Клиенты авиакомпании за указанную дату», в верхней панели, откроется окно формы «Клиенты авиакомпании за указанную дату» (рис. 5.9). В ограждённом ниже поле появятся данные о номере и серии паспорта клиента, дате продажи и шифре авиакомпании.

Выберете код авиакомпании, день, месяц и год в соответствующих полях, и нажмите кнопку «Найти». При успешном вводе и нажатии кнопки база данных соответствующе обновится. При нажатии кнопки «Исх» база данных обновится в первоначальный вид.

1. **Описание и результаты тестов**

Тест форм «Авиакомпании», «Кассы», «Кассиры», «Клиенты», «Билеты» и «Купоны» проведен на примере формы «Авиакомпании», так как все 6 форм имеют аналогичное поведение.

Тест 1 для действий «Добавить» и «Обновить»:

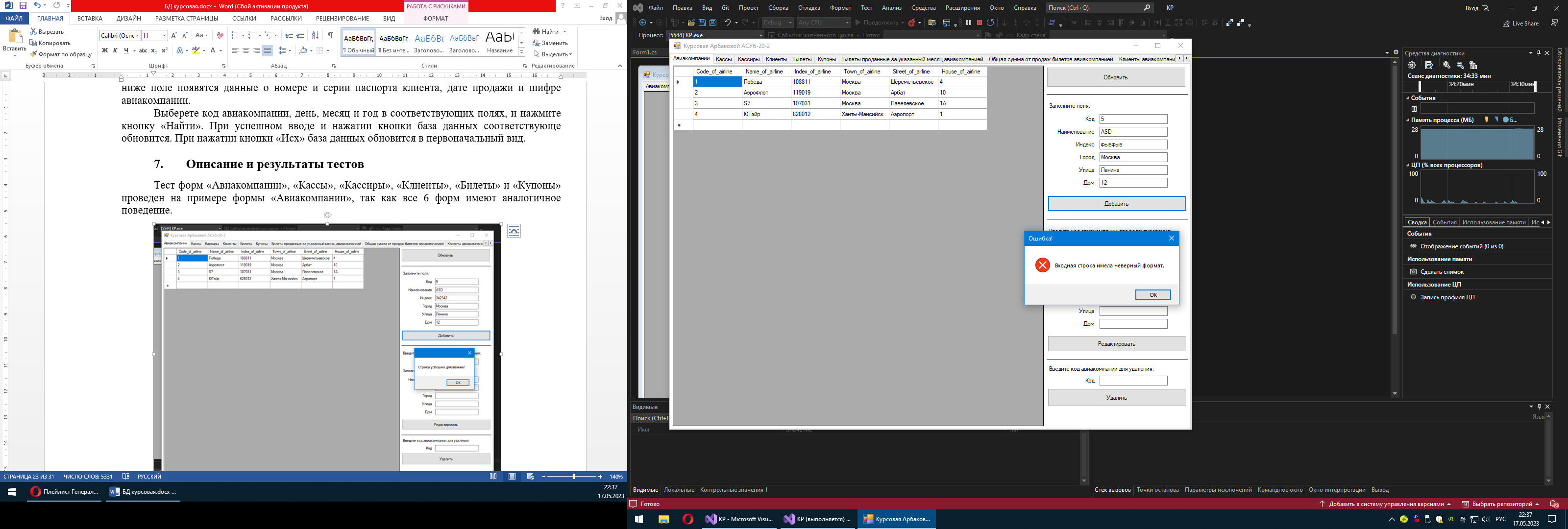


Рисунок 7.1 – Неверный ввод для действия «Добавить»

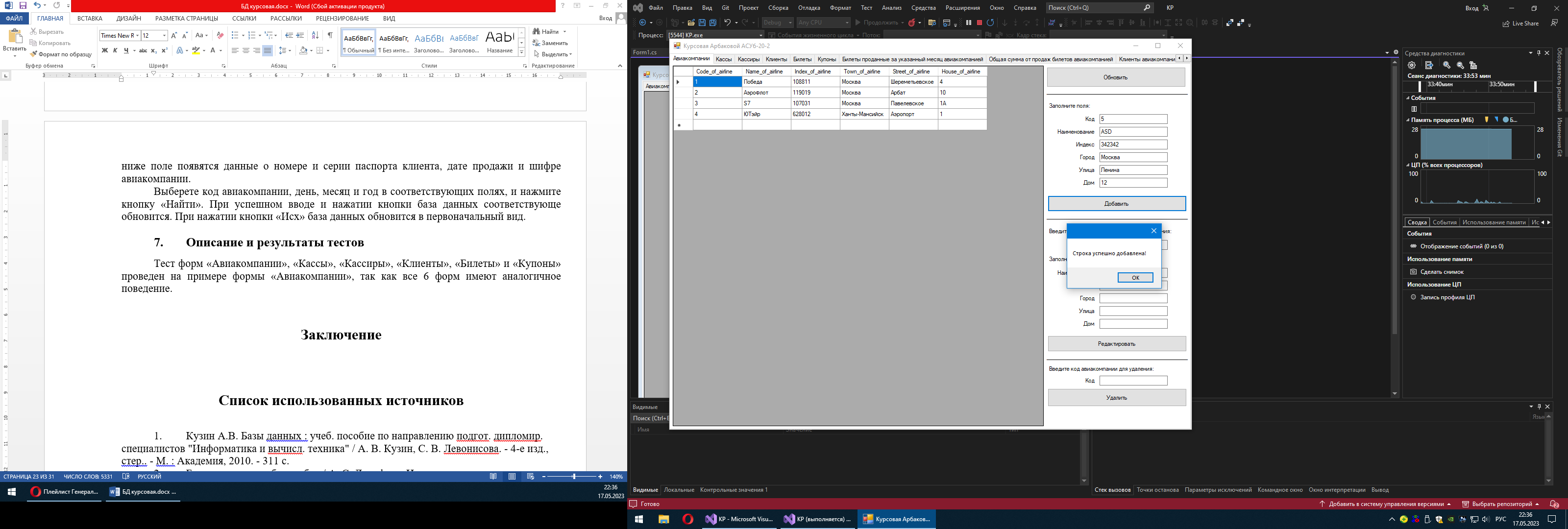


Рисунок 7.2 – Верный ввод для действия «Добавить»

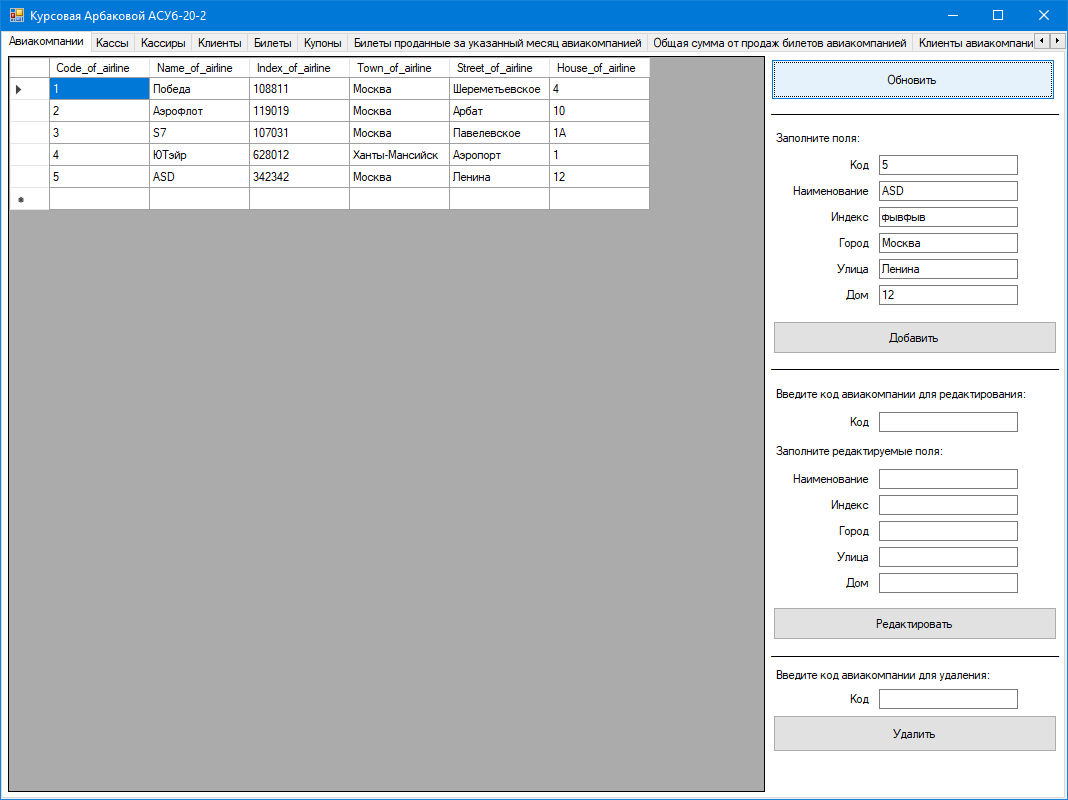


Рисунок 7.3 – Действие «Обновить»

Тест 2 для действий «Редактировать» и «Обновить»:

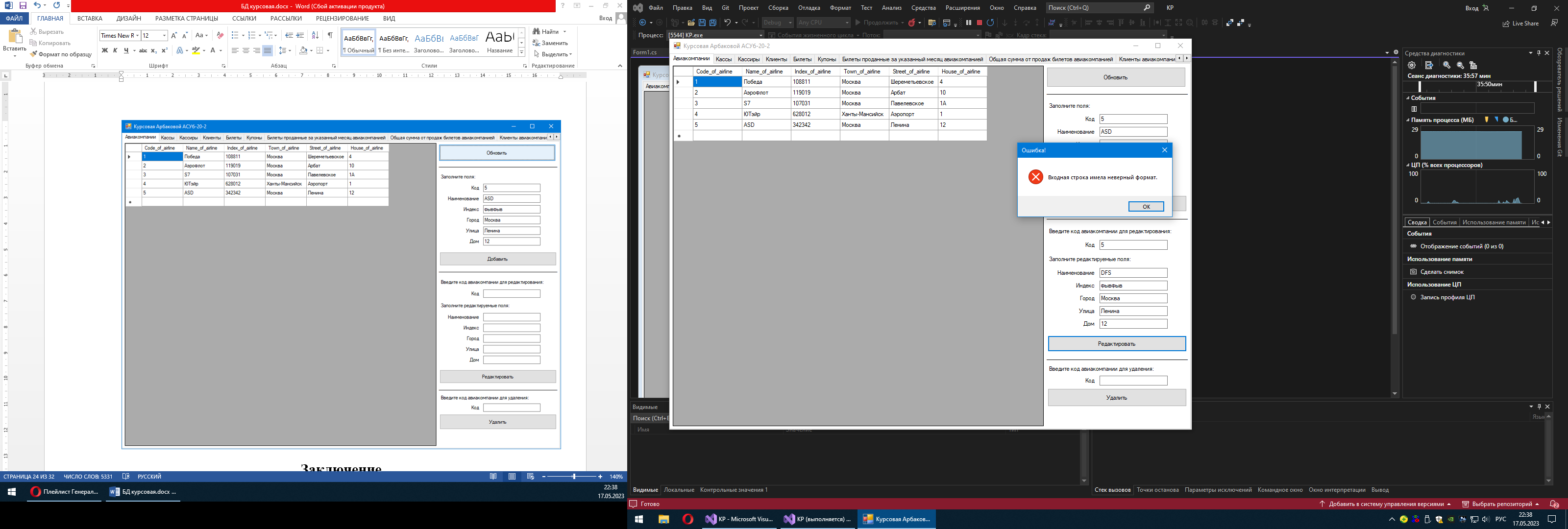


Рисунок 7.4 – Неверный ввод для действия «Редактировать»

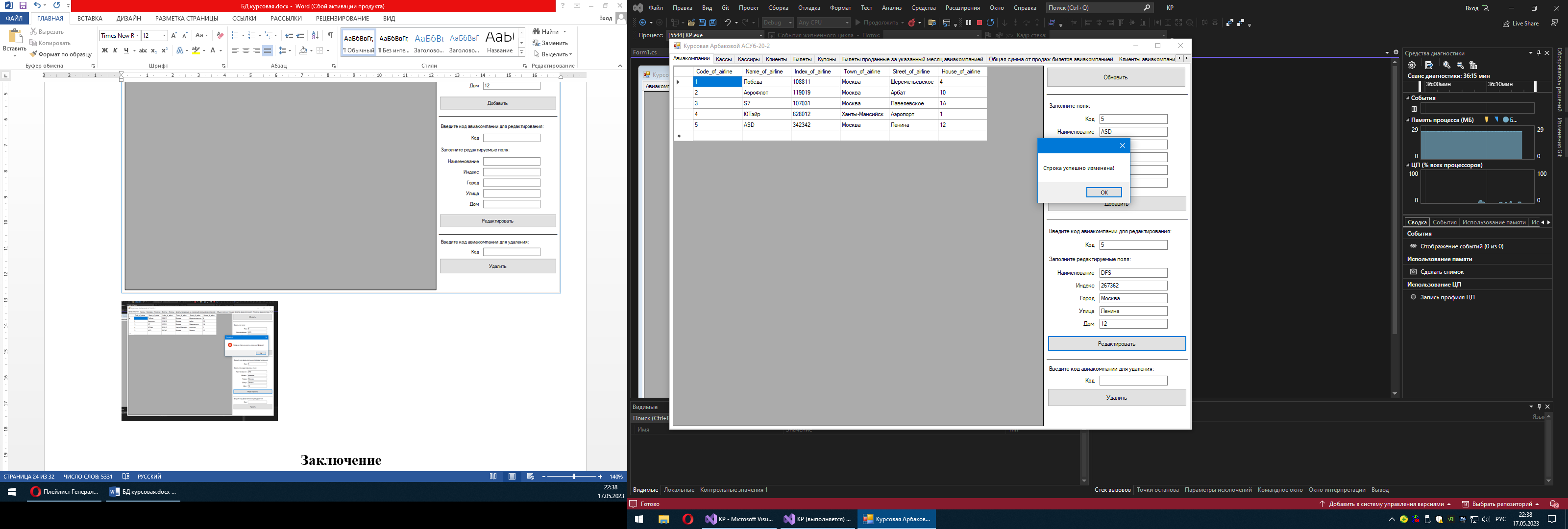


Рисунок 7.5 – Верный ввод для действия «Редактировать»

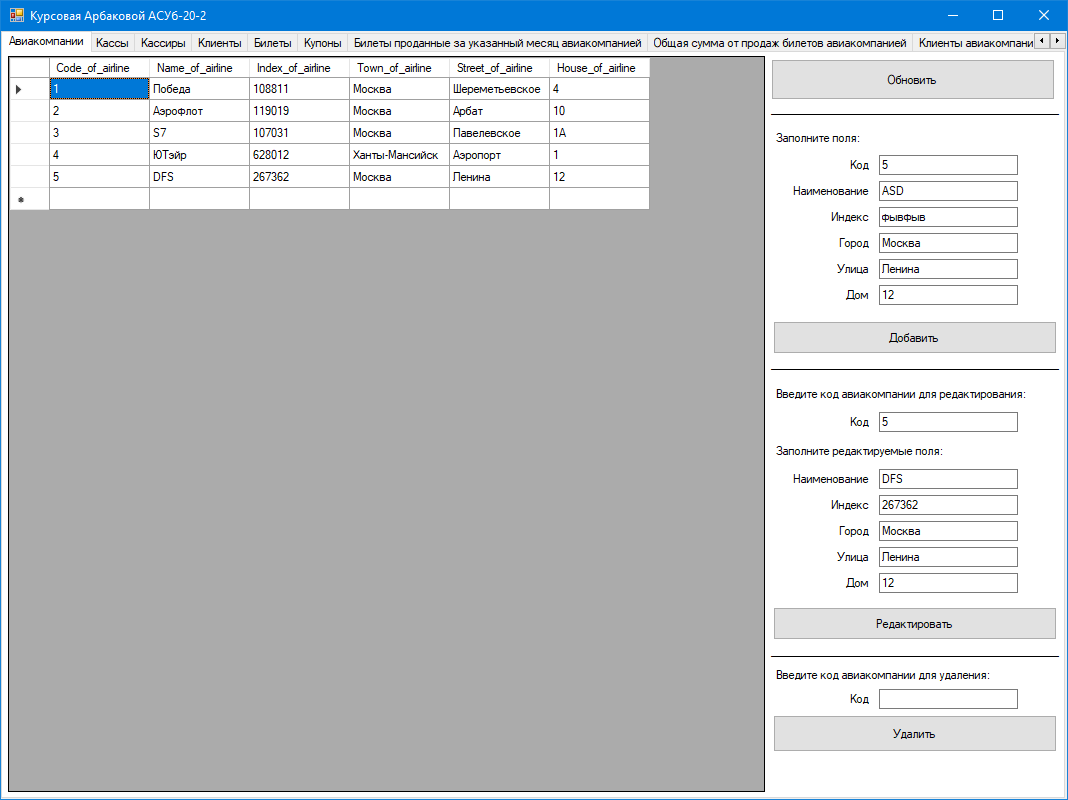


Рисунок 7.6 – Действие «Обновить»

Тест 3 для действий «Удалить» и «Обновить»:

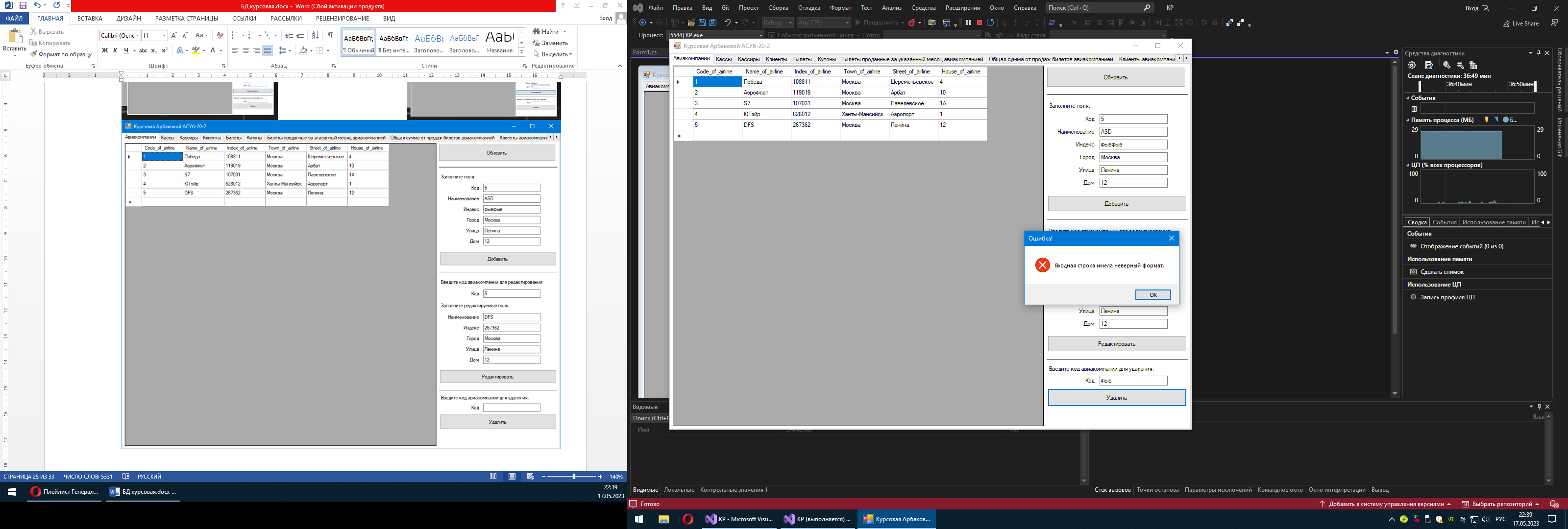


Рисунок 7.7 – Неверный ввод для действия «Удалить»

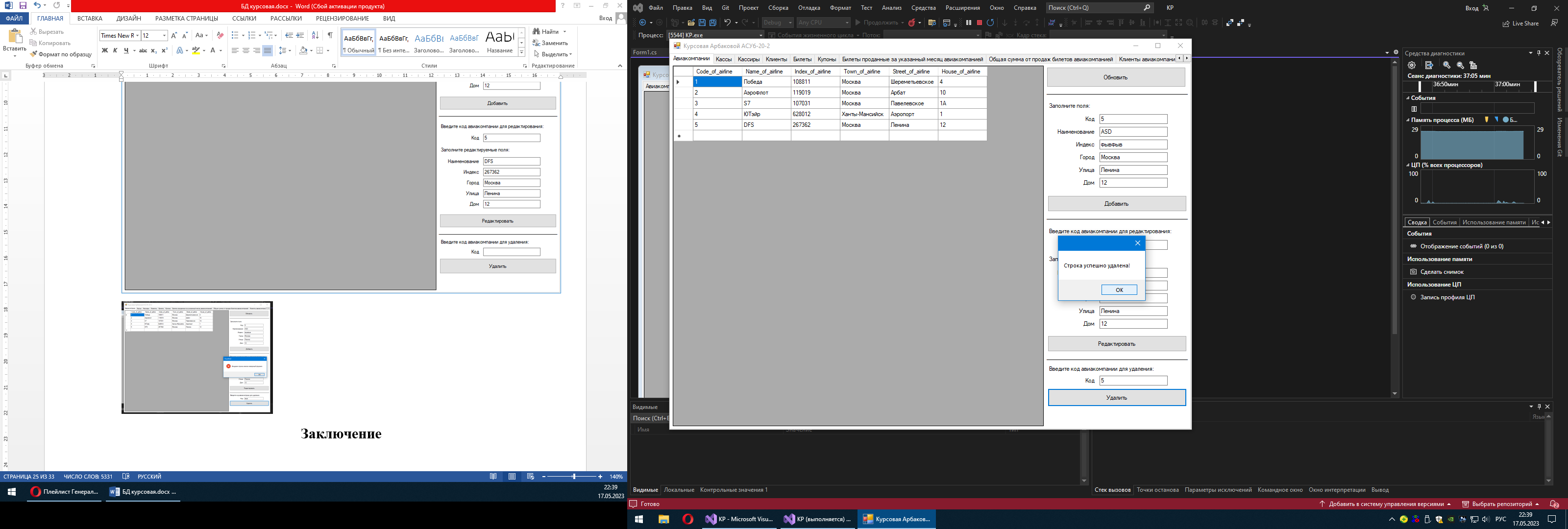


Рисунок 7.8 – Верный ввод для действия «Удалить»

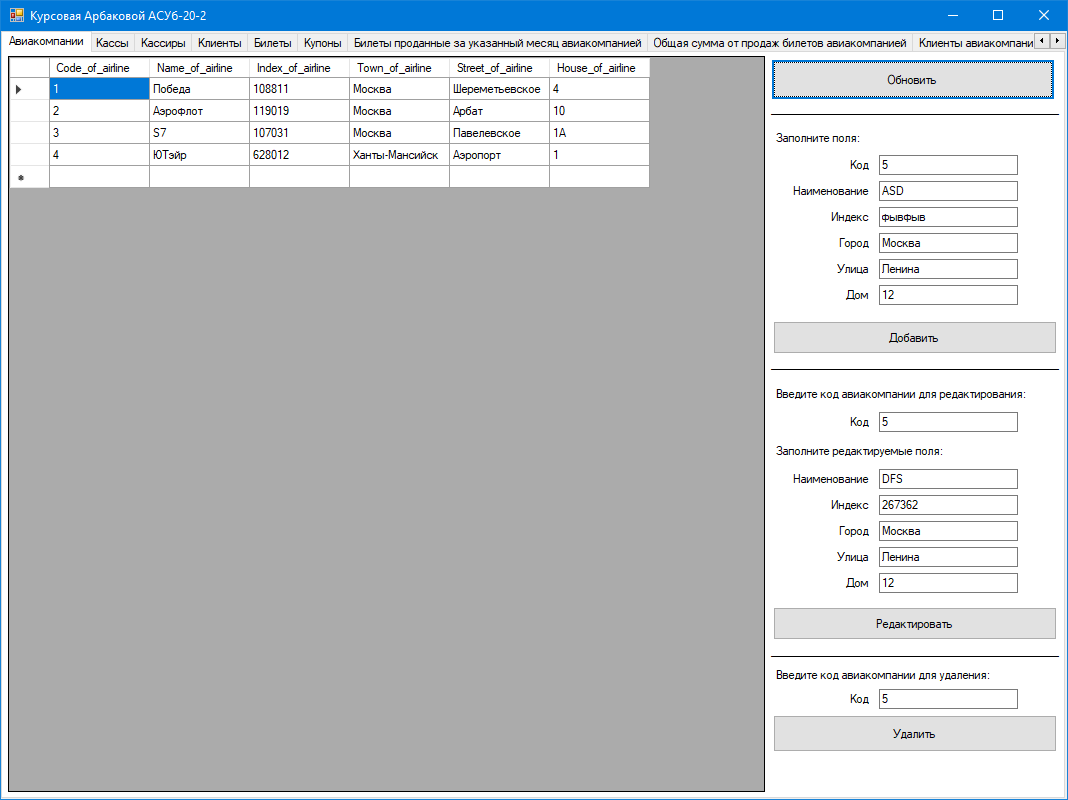


Рисунок 7.9 – Действие «Обновить»

Тест 4 для формы «Билеты, проданные за указанный месяц авиакомпанией»:

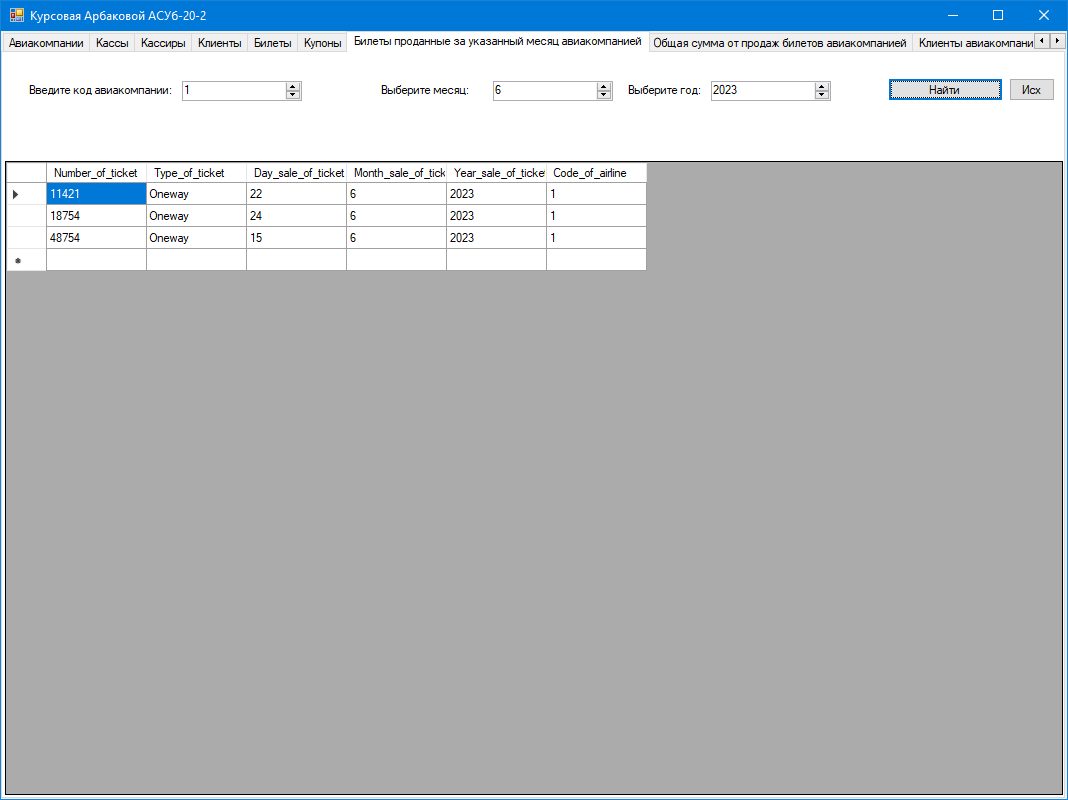


Рисунок 7.10.1 – Действие кнопки «Найти» с указанными параметрами

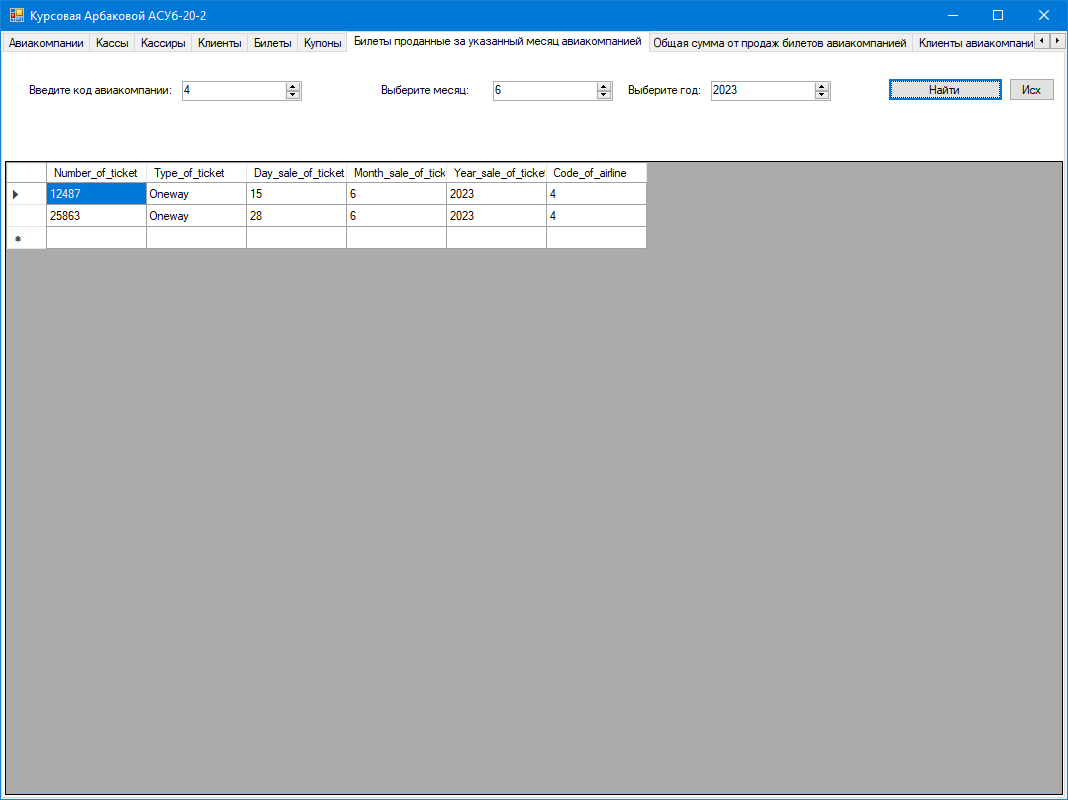


Рисунок 7.10.2 – Действие кнопки «Найти» с указанными параметрами

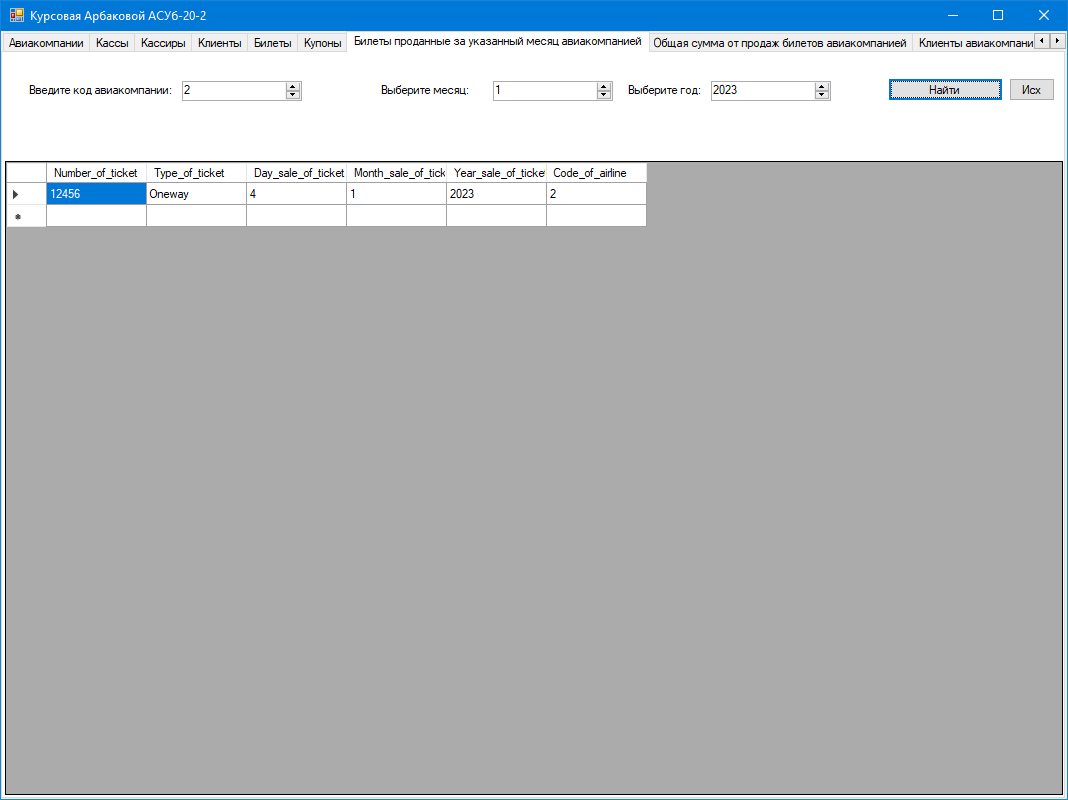


Рисунок 7.10.3 – Действие кнопки «Найти» с указанными параметрами

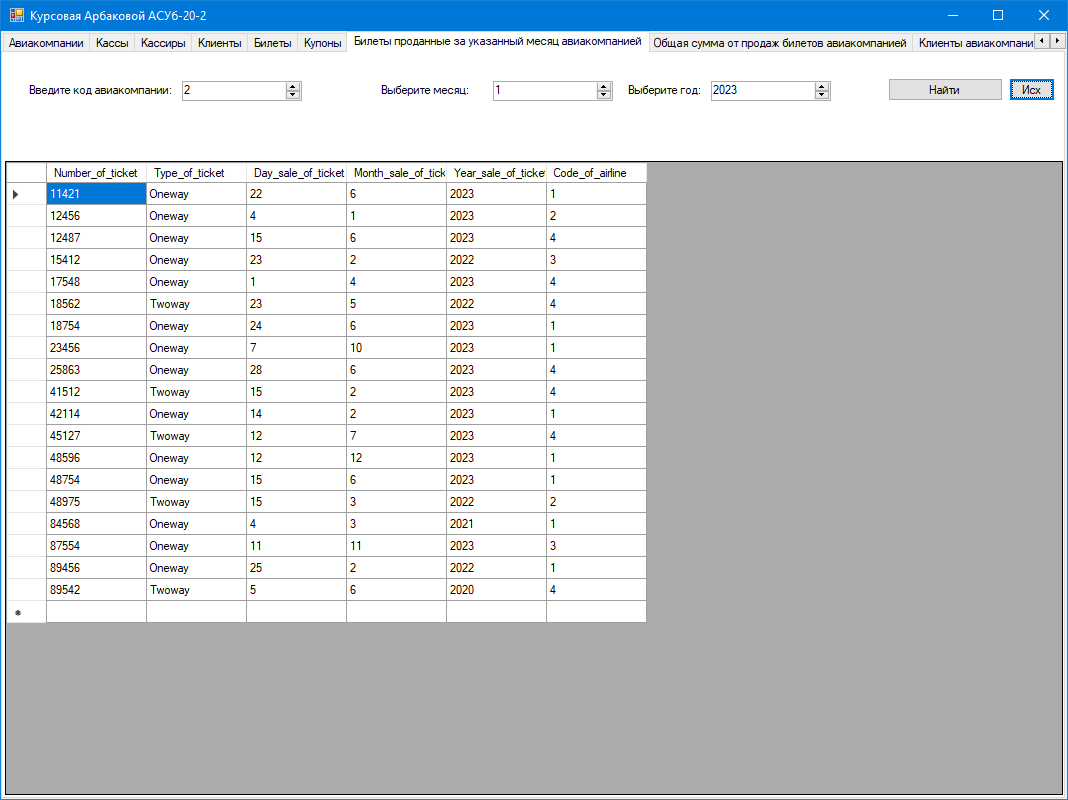


Рисунок 7.11 – Действие кнопки «Исх»

Тест 5 для формы «Общая сумма от продаж билетов авиакомпанией»:

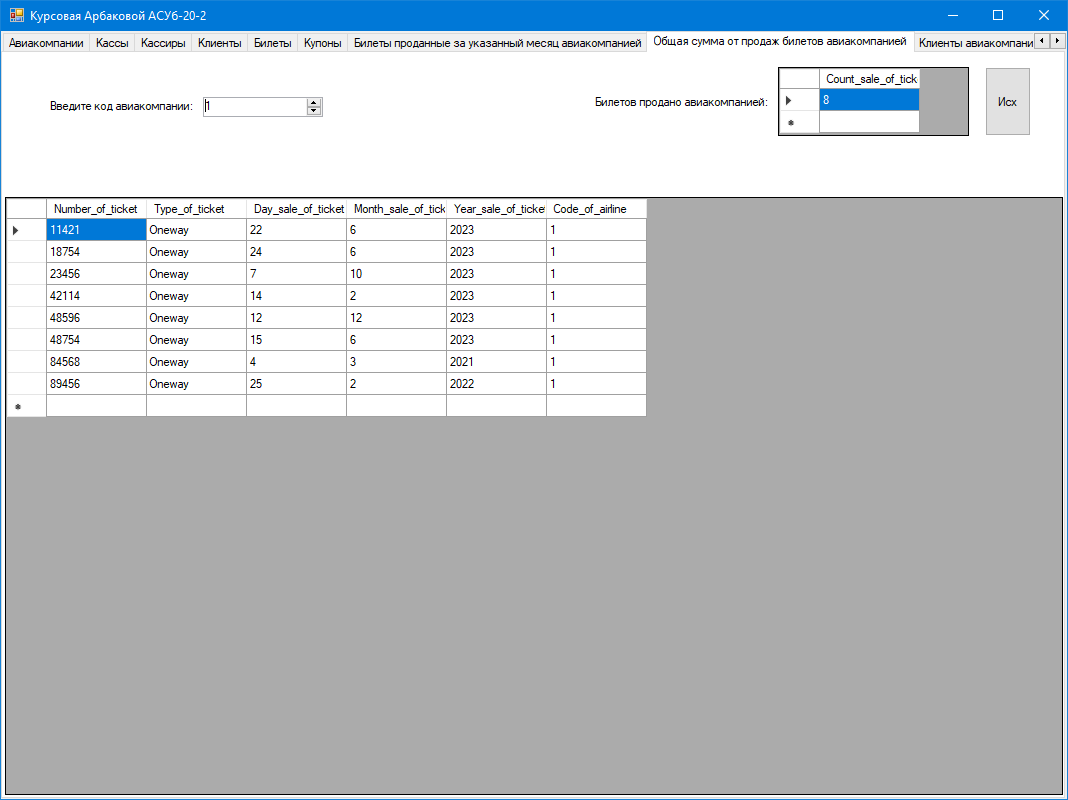


Рисунок 7.12.1 – Подсчет проданных билетов

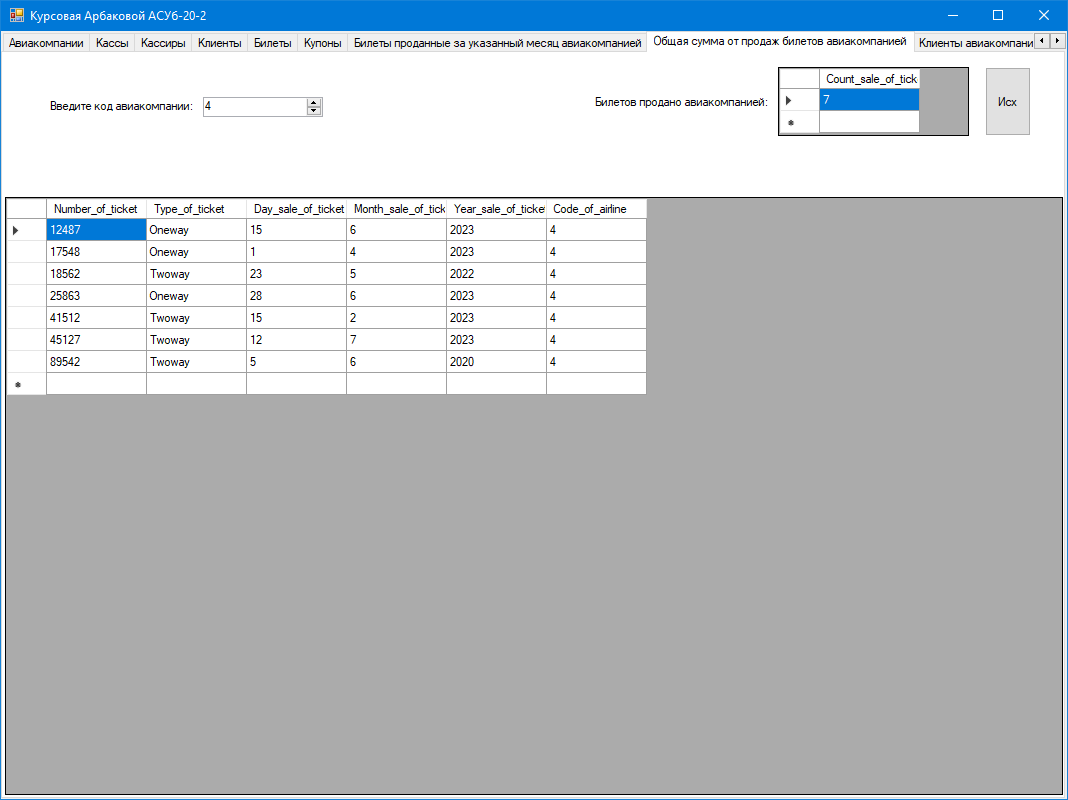


Рисунок 7.12.2 – Подсчет проданных билетов

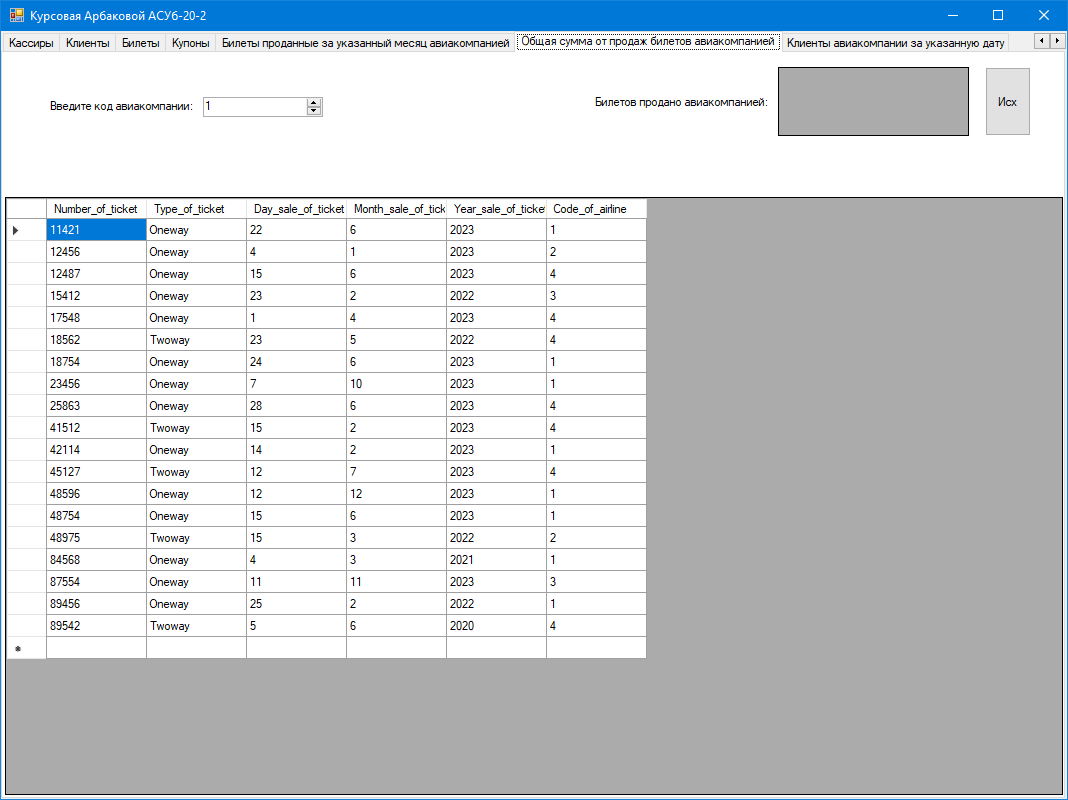


Рисунок 7.13 – Действие кнопки «Исх»

Тест 6 для формы «Клиенты авиакомпании за указанную дату»:

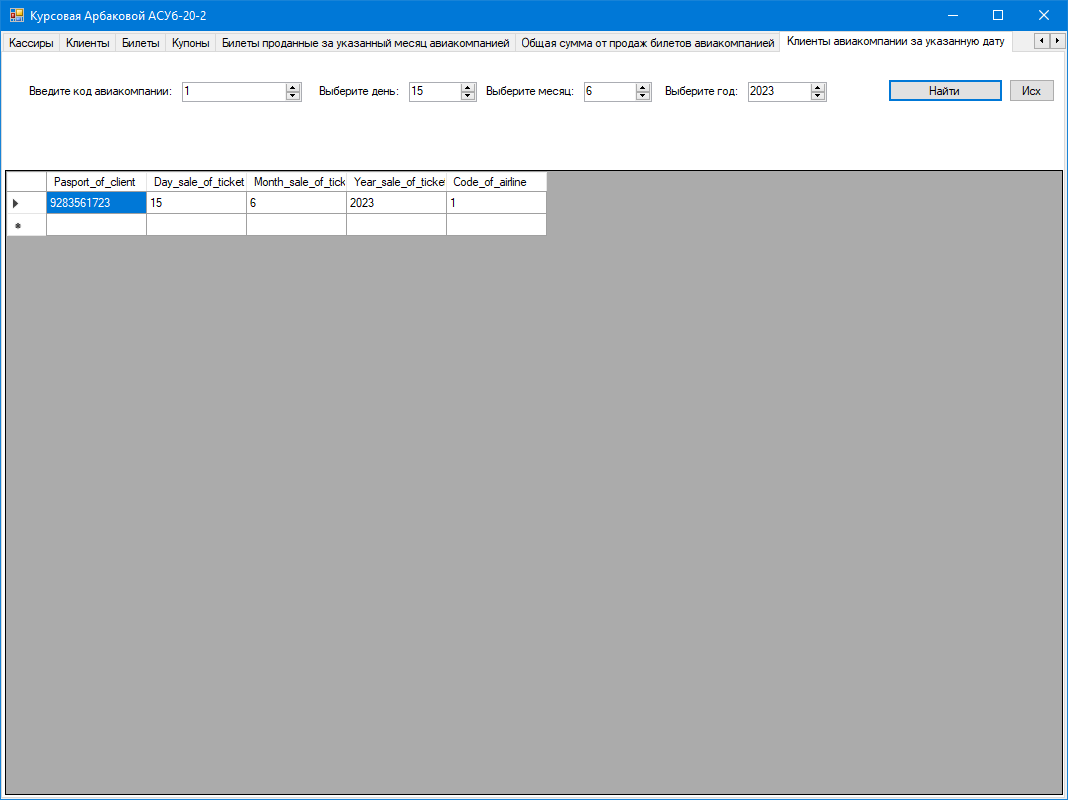


Рисунок 7.14.1 – Действие кнопки «Найти» с указанными параметрами

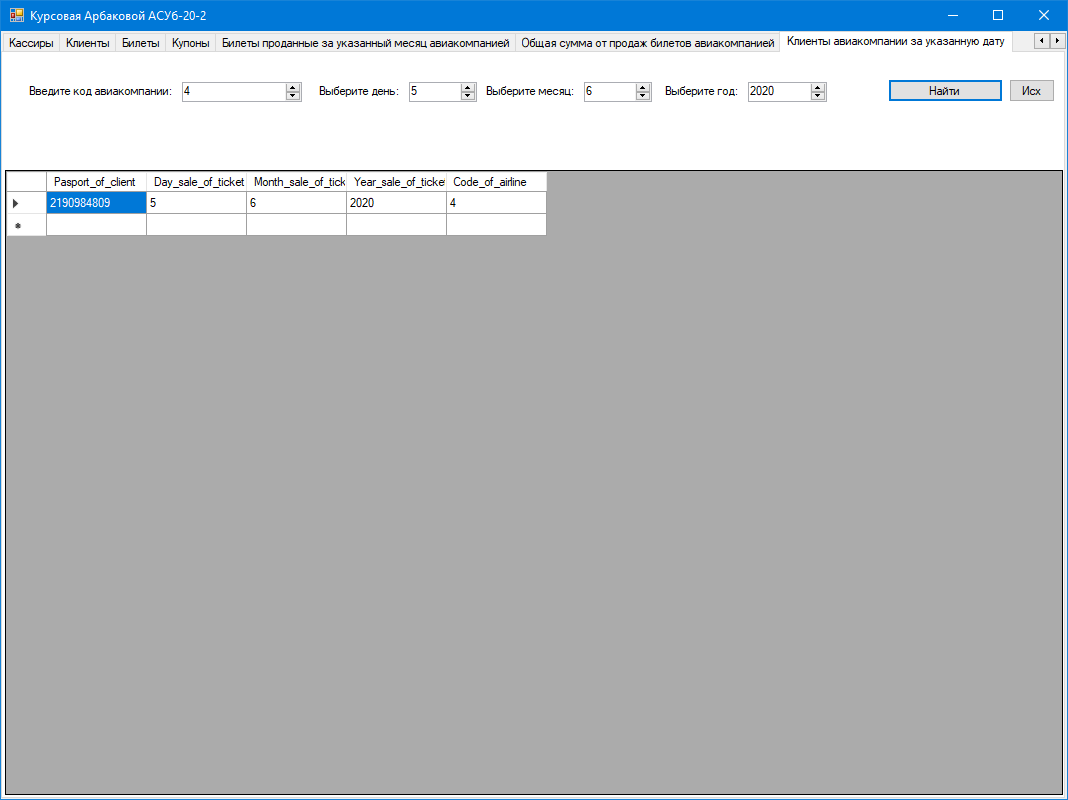


Рисунок 7.14.2 – Действие кнопки «Найти» с указанными параметрами

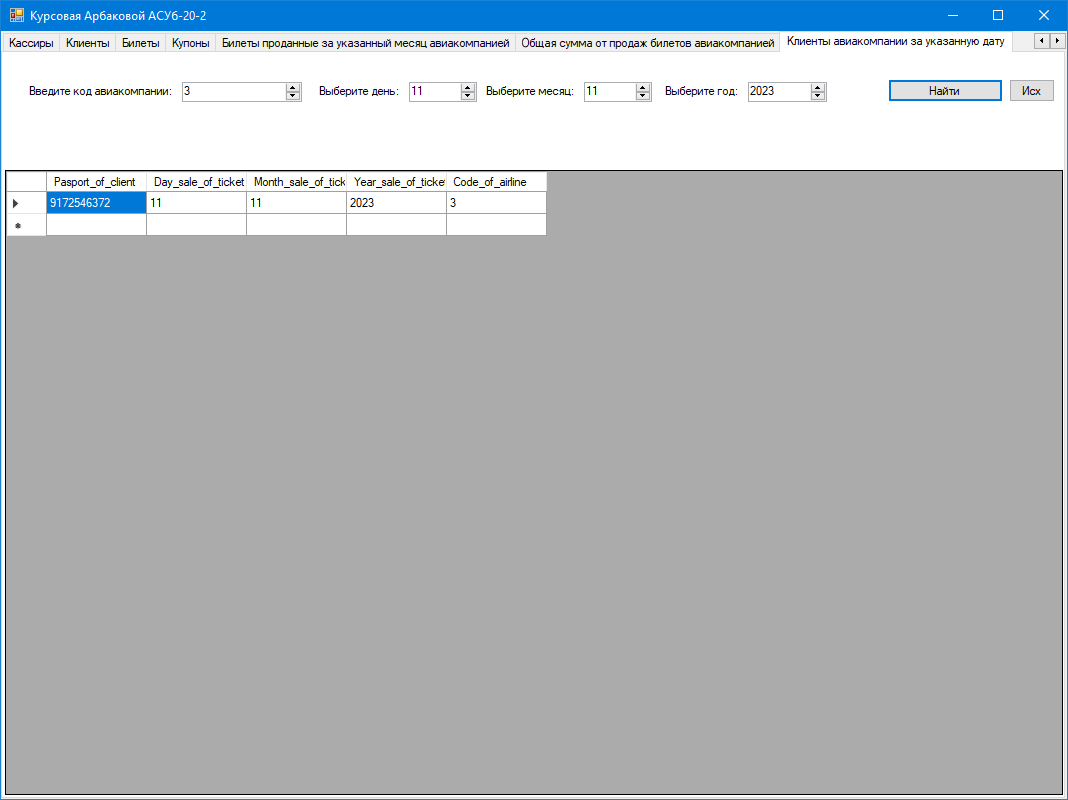


Рисунок 7.14.3 – Действие кнопки «Найти» с указанными параметрами

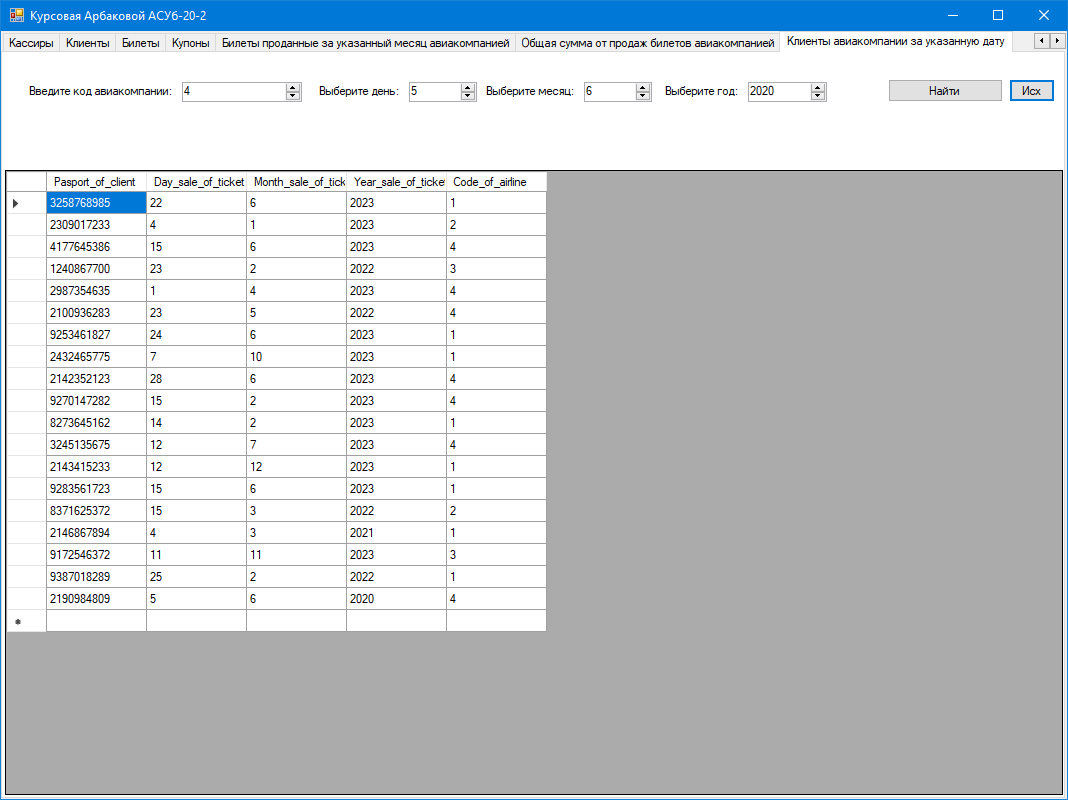


Рисунок 7.15 – Действие кнопки «Исх»

По итогу, можно сделать вывод. Тестирование показало, что приложение работает корректно и выводит требуемую информацию. Все функции приложения выполняются в соответствии с ожиданиями, интерфейс удобен для использования, а база данных функционирует без ошибок.

# **Заключение**

В ходе выполнения курсового проекта, были получены знания и навыки в области знаний баз данных. Было усвоено, что для работы с базами данных и решения проблем обработки информации используются компьютеры с системами управлениями базами данных (СУБД). Курсовой проект был выполнен в программе Microsoft Visual Studio.

Выполнена цель курсового проекта по получению навыков в проектировании структуры базы данных, и написанию и отладке приложения для ведения базы данных. Применяемой технология являлось реализованное клиент-серверное приложение. Было выполнено задание по реализации базы данных авиакомпании с необходимыми хранимыми данными о кассах, кассирах, клиентах, билетах, купонов и об авиакомпаниях. Полученными документами стали билеты, проданные за указанный месяц указанной авиакомпании, общая сумма от продаж билетов каждой авиакомпании и список клиентов авиакомпаний на заданную дату.

# **Список использованных источников**

1. Базы данных : учеб. пособие / А. С. Дорофеев; Иркут. гос. техн. ун-т . – Иркутск: Изд-во ИрГТУ, 2008. – 99 с. : a-ил.
2. Дорофеев А.С. Базы данных (09.03.01) для набора с 2019 г. [Электронный ресурс]. [2021]. URL: https://el.istu.edu/course/view.php?id=5192 (дата обращения: 10.10.2023).
3. Разработка приложений баз данных на основе современных технологий : учебное пособие для вузов по направлению "Конструкторско-технологическое обеспечение машиностроительных производств" / А. С. Дорофеев [и др.]. - Старый Оскол : ТНТ, 2020. - 275 с.
4. Ставров, С. Г. Практикум по работе с базами данных в Microsoft Visio и СУБД Microsoft SQL Server : учебное пособие / С. Г. Ставров, А. Е. Кочетков. — Иваново : ИГЭУ, 2018. — 80 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/154589 (дата обращения: 08.10.2023).
5. Волк, В. К. Базы данных : учебное пособие / В. К. Волк. — Курган : КГУ, 2018 — Часть 1 : Проектирование и программирование — 2018. — 178 с. — ISBN 978-5-4217-0472-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/177903 (дата обращения: 08.10.2023)
6. Документация по Visual Stidio URL: <https://learn.microsoft.com/ru-ru/visualstudio/windows/?view=vs-2022> (дата обращения: 10.10.2023)
7. Документация по MySQL URL: <https://dev.mysql.com/doc> (дата обращения: 10.10.2023)

# **Приложение А Листинг**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using System.Data.Common;

namespace KP

{

public partial class Form1 : Form

{

private SqlConnection sqlConnection = null;

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);

sqlConnection.Open();

if (sqlConnection.State == ConnectionState.Open)

{

MessageBox.Show("Подключение установлено!");

}

///////////////Таблица Airline

SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT \* FROM Airline", sqlConnection);

DataSet dataSet = new DataSet();

dataAdapter.Fill(dataSet);

dataGridView1.DataSource = dataSet.Tables[0];

///////////////Таблица Cash\_register

SqlDataAdapter dataAdapter1 = new SqlDataAdapter("SELECT \* FROM Cash\_register", sqlConnection);

DataSet dataSet1 = new DataSet();

dataAdapter1.Fill(dataSet1);

dataGridView2.DataSource = dataSet1.Tables[0];

///////////////Таблица Cashier

SqlDataAdapter dataAdapter2 = new SqlDataAdapter("SELECT \* FROM Cashier", sqlConnection);

DataSet dataSet2 = new DataSet();

dataAdapter2.Fill(dataSet2);

dataGridView3.DataSource = dataSet2.Tables[0];

///////////////Таблица Client

SqlDataAdapter dataAdapter3 = new SqlDataAdapter("SELECT \* FROM Client", sqlConnection);

DataSet dataSet3 = new DataSet();

dataAdapter3.Fill(dataSet3);

dataGridView4.DataSource = dataSet3.Tables[0];

///////////////Таблица Ticket

SqlDataAdapter dataAdapter4 = new SqlDataAdapter("SELECT \* FROM Ticket", sqlConnection);

DataSet dataSet4 = new DataSet();

dataAdapter4.Fill(dataSet4);

dataGridView5.DataSource = dataSet4.Tables[0];

///////////////Таблица Coupon

SqlDataAdapter dataAdapter5 = new SqlDataAdapter("SELECT \* FROM Coupon", sqlConnection);

DataSet dataSet5 = new DataSet();

dataAdapter5.Fill(dataSet5);

dataGridView6.DataSource = dataSet5.Tables[0];

///////////////Таблица Task1

SqlDataAdapter dataAdapter6 = new SqlDataAdapter("SELECT Number\_of\_ticket, Type\_of\_ticket, Day\_sale\_of\_ticket, Month\_sale\_of\_ticket, Year\_sale\_of\_ticket, Code\_of\_airline FROM Ticket", sqlConnection);

DataSet dataSet6 = new DataSet();

dataAdapter6.Fill(dataSet6);

dataGridView7.DataSource = dataSet6.Tables[0];

///////////////Таблица Task2

SqlDataAdapter dataAdapter7 = new SqlDataAdapter("SELECT Number\_of\_ticket, Type\_of\_ticket, Day\_sale\_of\_ticket, Month\_sale\_of\_ticket, Year\_sale\_of\_ticket, Code\_of\_airline FROM Ticket", sqlConnection);

DataSet dataSet7 = new DataSet();

dataAdapter7.Fill(dataSet7);

dataGridView8.DataSource = dataSet7.Tables[0];

///////////////Таблица Task3

SqlDataAdapter dataAdapter8 = new SqlDataAdapter("SELECT Pasport\_of\_client, Day\_sale\_of\_ticket, Month\_sale\_of\_ticket, Year\_sale\_of\_ticket, Code\_of\_airline FROM Ticket", sqlConnection);

DataSet dataSet8 = new DataSet();

dataAdapter8.Fill(dataSet8);

dataGridView10.DataSource = dataSet8.Tables[0];

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

private void button1\_Click(object sender, EventArgs e)

{

try

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);

sqlConnection.Open();

SqlCommand cmd = new SqlCommand("INSERT INTO Airline VALUES (@Code\_of\_airline, @Name\_of\_airline, @Index\_of\_airline, @Town\_of\_airline, @Street\_of\_airline, @house\_of\_airline)", sqlConnection);

cmd.Parameters.AddWithValue("@Code\_of\_airline", int.Parse(textBox1.Text));

cmd.Parameters.AddWithValue("@Name\_of\_airline", textBox2.Text);

cmd.Parameters.AddWithValue("@Index\_of\_airline", int.Parse(textBox3.Text));

cmd.Parameters.AddWithValue("@Town\_of\_airline", textBox4.Text);

cmd.Parameters.AddWithValue("@Street\_of\_airline", textBox5.Text);

cmd.Parameters.AddWithValue("@House\_of\_airline", textBox6.Text);

cmd.ExecuteNonQuery();

MessageBox.Show("Строка успешно добавлена!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button2\_Click(object sender, EventArgs e)

{

try

{

SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT \* FROM Airline", sqlConnection);

DataSet dataSet = new DataSet();

dataAdapter.Fill(dataSet);

dataGridView1.DataSource = dataSet.Tables[0];

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button3\_Click(object sender, EventArgs e)

{

try

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);

sqlConnection.Open();

SqlCommand cmd = new SqlCommand("Update Airline set Name\_of\_airline=@Name\_of\_airline, Index\_of\_airline=@Index\_of\_airline, Town\_of\_airline=@Town\_of\_airline, Street\_of\_airline=@Street\_of\_airline, House\_of\_airline=@House\_of\_airline where Code\_of\_airline=@Code\_of\_airline", sqlConnection);

cmd.Parameters.AddWithValue("@Code\_of\_airline", int.Parse(textBox7.Text));

cmd.Parameters.AddWithValue("@Name\_of\_airline", textBox12.Text);

cmd.Parameters.AddWithValue("@Index\_of\_airline", int.Parse(textBox11.Text));

cmd.Parameters.AddWithValue("@Town\_of\_airline", textBox10.Text);

cmd.Parameters.AddWithValue("@Street\_of\_airline", textBox9.Text);

cmd.Parameters.AddWithValue("@House\_of\_airline", textBox8.Text);

cmd.ExecuteNonQuery();

MessageBox.Show("Строка успешно изменена!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button4\_Click(object sender, EventArgs e)

{

try

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);

sqlConnection.Open();

SqlCommand cmd = new SqlCommand("Delete Airline where Code\_of\_airline=@Code\_of\_airline", sqlConnection);

cmd.Parameters.AddWithValue("@Code\_of\_airline", int.Parse(textBox13.Text));

cmd.ExecuteNonQuery();

MessageBox.Show("Строка успешно удалена!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

private void button7\_Click(object sender, EventArgs e)

{

try

{

SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT \* FROM Cash\_register", sqlConnection);

DataSet dataSet = new DataSet();

dataAdapter.Fill(dataSet);

dataGridView2.DataSource = dataSet.Tables[0];

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button8\_Click(object sender, EventArgs e)

{

try

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);

sqlConnection.Open();

SqlCommand cmd = new SqlCommand("INSERT INTO Cash\_register VALUES (@Number\_of\_cash\_register, @Index\_of\_cash\_register, @Town\_of\_cash\_register, @Street\_of\_cash\_register, @House\_of\_cash\_register)", sqlConnection);

cmd.Parameters.AddWithValue("@Number\_of\_cash\_register", int.Parse(textBox26.Text));

cmd.Parameters.AddWithValue("@Index\_of\_cash\_register", int.Parse(textBox24.Text));

cmd.Parameters.AddWithValue("@Town\_of\_cash\_register", textBox23.Text);

cmd.Parameters.AddWithValue("@Street\_of\_cash\_register", textBox22.Text);

cmd.Parameters.AddWithValue("@House\_of\_cash\_register", textBox21.Text);

cmd.ExecuteNonQuery();

MessageBox.Show("Строка успешно добавлена!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button6\_Click(object sender, EventArgs e)

{

try

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);

sqlConnection.Open();

SqlCommand cmd = new SqlCommand("Update Cash\_register set Index\_of\_cash\_register=@Index\_of\_cash\_register, Town\_of\_cash\_register=@Town\_of\_cash\_register, Street\_of\_cash\_register=@Street\_of\_cash\_register, House\_of\_cash\_register=@House\_of\_cash\_register where Number\_of\_cash\_register=@Number\_of\_cash\_register", sqlConnection);

cmd.Parameters.AddWithValue("@Number\_of\_cash\_register", int.Parse(textBox20.Text));

cmd.Parameters.AddWithValue("@Index\_of\_cash\_register", int.Parse(textBox18.Text));

cmd.Parameters.AddWithValue("@Town\_of\_cash\_register", textBox17.Text);

cmd.Parameters.AddWithValue("@Street\_of\_cash\_register", textBox16.Text);

cmd.Parameters.AddWithValue("@House\_of\_cash\_register", textBox15.Text);

cmd.ExecuteNonQuery();

MessageBox.Show("Строка успешно изменена!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button5\_Click(object sender, EventArgs e)

{

try

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);

sqlConnection.Open();

SqlCommand cmd = new SqlCommand("Delete Cash\_register where Number\_of\_cash\_register=@Number\_of\_cash\_register", sqlConnection);

cmd.Parameters.AddWithValue("@Number\_of\_cash\_register", int.Parse(textBox14.Text));

cmd.ExecuteNonQuery();

MessageBox.Show("Строка успешно удалена!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

private void button11\_Click(object sender, EventArgs e)

{

try

{

SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT \* FROM Cashier", sqlConnection);

DataSet dataSet = new DataSet();

dataAdapter.Fill(dataSet);

dataGridView3.DataSource = dataSet.Tables[0];

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button12\_Click(object sender, EventArgs e)

{

try

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);

sqlConnection.Open();

SqlCommand cmd = new SqlCommand("INSERT INTO Cashier VALUES (@Code\_of\_cashier, @Surname\_of\_cashier, @Name\_of\_cashier, @Patronymic\_of\_cashier)", sqlConnection);

cmd.Parameters.AddWithValue("@Code\_of\_cashier", int.Parse(textBox35.Text));

cmd.Parameters.AddWithValue("@Surname\_of\_cashier", textBox34.Text);

cmd.Parameters.AddWithValue("@Name\_of\_cashier", textBox33.Text);

cmd.Parameters.AddWithValue("@Patronymic\_of\_cashier", textBox32.Text);

cmd.ExecuteNonQuery();

MessageBox.Show("Строка успешно добавлена!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button10\_Click(object sender, EventArgs e)

{

try

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);

sqlConnection.Open();

SqlCommand cmd = new SqlCommand("Update Cashier set Surname\_of\_cashier=@Surname\_of\_cashier, Name\_of\_cashier=@Name\_of\_cashier, Patronymic\_of\_cashier=@Patronymic\_of\_cashier where Code\_of\_cashier=@Code\_of\_cashier", sqlConnection);

cmd.Parameters.AddWithValue("@Code\_of\_cashier", int.Parse(textBox30.Text));

cmd.Parameters.AddWithValue("@Surname\_of\_cashier", textBox29.Text);

cmd.Parameters.AddWithValue("@Name\_of\_cashier", textBox28.Text);

cmd.Parameters.AddWithValue("@Patronymic\_of\_cashier", textBox27.Text);

cmd.ExecuteNonQuery();

MessageBox.Show("Строка успешно изменена!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button9\_Click(object sender, EventArgs e)

{

try

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);

sqlConnection.Open();

SqlCommand cmd = new SqlCommand("Delete Cashier where Code\_of\_cashier=@Code\_of\_cashier", sqlConnection);

cmd.Parameters.AddWithValue("@Code\_of\_cashier", int.Parse(textBox19.Text));

cmd.ExecuteNonQuery();

MessageBox.Show("Строка успешно удалена!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

private void button15\_Click(object sender, EventArgs e)

{

try

{

SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT \* FROM Client", sqlConnection);

DataSet dataSet = new DataSet();

dataAdapter.Fill(dataSet);

dataGridView4.DataSource = dataSet.Tables[0];

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button16\_Click(object sender, EventArgs e)

{

try

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);

sqlConnection.Open();

SqlCommand cmd = new SqlCommand("INSERT INTO Client VALUES (@Pasport\_of\_client, @Surname\_of\_client, @Name\_of\_client, @Patronymic\_of\_client)", sqlConnection);

cmd.Parameters.AddWithValue("@Pasport\_of\_client", textBox42.Text);

cmd.Parameters.AddWithValue("@Surname\_of\_client", textBox41.Text);

cmd.Parameters.AddWithValue("@Name\_of\_client", textBox40.Text);

cmd.Parameters.AddWithValue("@Patronymic\_of\_client", textBox39.Text);

cmd.ExecuteNonQuery();

MessageBox.Show("Строка успешно добавлена!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button14\_Click(object sender, EventArgs e)

{

try

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);

sqlConnection.Open();

SqlCommand cmd = new SqlCommand("Update Client set Surname\_of\_client=@Surname\_of\_client, Name\_of\_client=@Name\_of\_client, Patronymic\_of\_client=@Patronymic\_of\_client where Pasport\_of\_client=@Pasport\_of\_client", sqlConnection);

cmd.Parameters.AddWithValue("@Pasport\_of\_client", textBox38.Text);

cmd.Parameters.AddWithValue("@Surname\_of\_client", textBox37.Text);

cmd.Parameters.AddWithValue("@Name\_of\_client", textBox36.Text);

cmd.Parameters.AddWithValue("@Patronymic\_of\_client", textBox31.Text);

cmd.ExecuteNonQuery();

MessageBox.Show("Строка успешно изменена!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button13\_Click(object sender, EventArgs e)

{

try

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);

sqlConnection.Open();

SqlCommand cmd = new SqlCommand("Delete Client where Pasport\_of\_client=@Pasport\_of\_client", sqlConnection);

cmd.Parameters.AddWithValue("@Pasport\_of\_client", textBox25.Text);

cmd.ExecuteNonQuery();

MessageBox.Show("Строка успешно удалена!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

private void button19\_Click(object sender, EventArgs e)

{

try

{

SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT \* FROM Ticket", sqlConnection);

DataSet dataSet = new DataSet();

dataAdapter.Fill(dataSet);

dataGridView5.DataSource = dataSet.Tables[0];

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button20\_Click(object sender, EventArgs e)

{

try

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);

sqlConnection.Open();

SqlCommand cmd = new SqlCommand("INSERT INTO Ticket VALUES (@Number\_of\_ticket, @Type\_of\_ticket, @Day\_sale\_of\_ticket, @Month\_sale\_of\_ticket, @Year\_sale\_of\_ticket, @Number\_of\_cash\_register, @Code\_of\_cashier, @Code\_of\_airline, @Pasport\_of\_client)", sqlConnection);

cmd.Parameters.AddWithValue("@Number\_of\_ticket", int.Parse(textBox51.Text));

cmd.Parameters.AddWithValue("@Type\_of\_ticket", textBox50.Text);

cmd.Parameters.AddWithValue("@Day\_sale\_of\_ticket", int.Parse(textBox49.Text));

cmd.Parameters.AddWithValue("@Month\_sale\_of\_ticket", int.Parse(textBox67.Text));

cmd.Parameters.AddWithValue("@Year\_sale\_of\_ticket", int.Parse(textBox68.Text));

cmd.Parameters.AddWithValue("@Number\_of\_cash\_register", int.Parse(textBox48.Text));

cmd.Parameters.AddWithValue("@Code\_of\_cashier", int.Parse(textBox52.Text));

cmd.Parameters.AddWithValue("@Code\_of\_airline", int.Parse(textBox65.Text));

cmd.Parameters.AddWithValue("@Pasport\_of\_client", textBox71.Text);

cmd.ExecuteNonQuery();

MessageBox.Show("Строка успешно добавлена!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button18\_Click(object sender, EventArgs e)

{

try

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);

sqlConnection.Open();

SqlCommand cmd = new SqlCommand("Update Ticket set Type\_of\_ticket=@Type\_of\_ticket, Day\_sale\_of\_ticket=@Day\_sale\_of\_ticket, Month\_sale\_of\_ticket=@Month\_sale\_of\_ticket, Year\_sale\_of\_ticket=@Year\_sale\_of\_ticket, Number\_of\_cash\_register=@Number\_of\_cash\_register, Code\_of\_cashier=@Code\_of\_cashier, Code\_of\_airline=@Code\_of\_airline, Pasport\_of\_client=@Pasport\_of\_client where Number\_of\_ticket=@Number\_of\_ticket", sqlConnection);

cmd.Parameters.AddWithValue("@Number\_of\_ticket", int.Parse(textBox47.Text));

cmd.Parameters.AddWithValue("@Type\_of\_ticket", textBox46.Text);

cmd.Parameters.AddWithValue("@Day\_sale\_of\_ticket", int.Parse(textBox70.Text));

cmd.Parameters.AddWithValue("@Month\_sale\_of\_ticket", int.Parse(textBox69.Text));

cmd.Parameters.AddWithValue("@Year\_sale\_of\_ticket", int.Parse(textBox45.Text));

cmd.Parameters.AddWithValue("@Number\_of\_cash\_register", int.Parse(textBox44.Text));

cmd.Parameters.AddWithValue("@Code\_of\_cashier", int.Parse(textBox53.Text));

cmd.Parameters.AddWithValue("@Code\_of\_airline", int.Parse(textBox66.Text));

cmd.Parameters.AddWithValue("@Pasport\_of\_client", int.Parse(textBox72.Text));

cmd.ExecuteNonQuery();

MessageBox.Show("Строка успешно изменена!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button17\_Click(object sender, EventArgs e)

{

try

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);

sqlConnection.Open();

SqlCommand cmd = new SqlCommand("Delete Ticket where Number\_of\_ticket=@Number\_of\_ticket", sqlConnection);

cmd.Parameters.AddWithValue("@Number\_of\_ticket", textBox43.Text);

cmd.ExecuteNonQuery();

MessageBox.Show("Строка успешно удалена!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

private void button23\_Click(object sender, EventArgs e)

{

try

{

SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT \* FROM Coupon", sqlConnection);

DataSet dataSet = new DataSet();

dataAdapter.Fill(dataSet);

dataGridView6.DataSource = dataSet.Tables[0];

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button24\_Click(object sender, EventArgs e)

{

try

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);

sqlConnection.Open();

SqlCommand cmd = new SqlCommand("INSERT INTO Coupon VALUES (@Number\_of\_coupon, @Flight\_direction\_of\_coupon, @Rate\_of\_coupon, @Pasport\_of\_client, @Number\_of\_ticket)", sqlConnection);

cmd.Parameters.AddWithValue("@Number\_of\_coupon", int.Parse(textBox64.Text));

cmd.Parameters.AddWithValue("@Flight\_direction\_of\_coupon", textBox63.Text);

cmd.Parameters.AddWithValue("@Rate\_of\_coupon", textBox62.Text);

cmd.Parameters.AddWithValue("@Pasport\_of\_client", int.Parse(textBox61.Text));

cmd.Parameters.AddWithValue("@Number\_of\_ticket", int.Parse(textBox55.Text));

cmd.ExecuteNonQuery();

MessageBox.Show("Строка успешно добавлена!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button22\_Click(object sender, EventArgs e)

{

try

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);

sqlConnection.Open();

SqlCommand cmd = new SqlCommand("Update Coupon set Flight\_direction\_of\_coupon=@Flight\_direction\_of\_coupon, Rate\_of\_coupon=@Rate\_of\_coupon, Pasport\_of\_client=@Pasport\_of\_client, Number\_of\_ticket=@Number\_of\_ticket where Number\_of\_coupon=@Number\_of\_coupon", sqlConnection);

cmd.Parameters.AddWithValue("@Number\_of\_coupon", int.Parse(textBox60.Text));

cmd.Parameters.AddWithValue("@Flight\_direction\_of\_coupon", textBox59.Text);

cmd.Parameters.AddWithValue("@Rate\_of\_coupon", textBox58.Text);

cmd.Parameters.AddWithValue("@Pasport\_of\_client", int.Parse(textBox57.Text));

cmd.Parameters.AddWithValue("@Number\_of\_ticket", int.Parse(textBox54.Text));

cmd.ExecuteNonQuery();

MessageBox.Show("Строка успешно изменена!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button21\_Click(object sender, EventArgs e)

{

try

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);

sqlConnection.Open();

SqlCommand cmd = new SqlCommand("Delete Coupon where Number\_of\_coupon=@Number\_of\_coupon", sqlConnection);

cmd.Parameters.AddWithValue("@Number\_of\_coupon", textBox56.Text);

cmd.ExecuteNonQuery();

MessageBox.Show("Строка успешно удалена!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

private void button25\_Click(object sender, EventArgs e)

{

(dataGridView7.DataSource as DataTable).DefaultView.RowFilter = $"Code\_of\_airline = {numericUpDown1.Value} AND Month\_sale\_of\_ticket = {numericUpDown2.Value} AND Year\_sale\_of\_ticket = {numericUpDown3.Value}";

}

private void button26\_Click(object sender, EventArgs e)

{

///////////////Таблица Task1

SqlDataAdapter dataAdapter6 = new SqlDataAdapter("SELECT Number\_of\_ticket, Type\_of\_ticket, Day\_sale\_of\_ticket, Month\_sale\_of\_ticket, Year\_sale\_of\_ticket, Code\_of\_airline FROM Ticket", sqlConnection);

DataSet dataSet6 = new DataSet();

dataAdapter6.Fill(dataSet6);

dataGridView7.DataSource = dataSet6.Tables[0];

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

private void numericUpDown4\_ValueChanged(object sender, EventArgs e)

{

(dataGridView8.DataSource as DataTable).DefaultView.RowFilter = $"Code\_of\_airline = {numericUpDown4.Value}";

SqlDataAdapter dataAdapter = new SqlDataAdapter($"SELECT COUNT(Number\_of\_ticket) AS 'Count\_sale\_of\_ticket' FROM Ticket WHERE Code\_of\_airline = '{numericUpDown4.Value}'", sqlConnection);

DataSet dataSet = new DataSet();

dataAdapter.Fill(dataSet);

dataGridView9.DataSource = dataSet.Tables[0];

}

private void button27\_Click(object sender, EventArgs e)

{

///////////////Таблица Task2

SqlDataAdapter dataAdapter6 = new SqlDataAdapter("SELECT Number\_of\_ticket, Type\_of\_ticket, Day\_sale\_of\_ticket, Month\_sale\_of\_ticket, Year\_sale\_of\_ticket, Code\_of\_airline FROM Ticket", sqlConnection);

DataSet dataSet6 = new DataSet();

dataAdapter6.Fill(dataSet6);

dataGridView8.DataSource = dataSet6.Tables[0];

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

private void button29\_Click(object sender, EventArgs e)

{

(dataGridView10.DataSource as DataTable).DefaultView.RowFilter = $"Code\_of\_airline = {numericUpDown7.Value} AND Day\_sale\_of\_ticket = {numericUpDown8.Value} AND Month\_sale\_of\_ticket = {numericUpDown6.Value} AND Year\_sale\_of\_ticket = {numericUpDown5.Value}";

}

private void button28\_Click(object sender, EventArgs e)

{

///////////////Таблица Task3

SqlDataAdapter dataAdapter8 = new SqlDataAdapter("SELECT Pasport\_of\_client, Day\_sale\_of\_ticket, Month\_sale\_of\_ticket, Year\_sale\_of\_ticket, Code\_of\_airline FROM Ticket", sqlConnection);

DataSet dataSet8 = new DataSet();

dataAdapter8.Fill(dataSet8);

dataGridView10.DataSource = dataSet8.Tables[0];

}

}

}