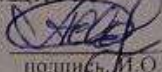


Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

Институт информационных технологий и анализа данных

Допускаю к защите

Руководитель  А.С. Дорофеев
подпись, И.О. Фамилия

Проектирование базы данных и разработка приложения для работы с ней

наименование темы

Пояснительная записка
к курсовому проекту

по дисциплине

«Базы данных»

1.015.00.00 – ПЗ

обозначение документа

Разработал студент группы АСУБ-20-2



Подпись

Арбакова А.В.

И.О. Фамилия

Нормоконтроль

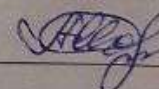
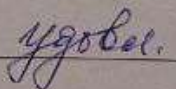


Подпись

Дорофеев А.С.

И.О. Фамилия

Курсовой проект защищен с оценкой



18.10.2023г.

Иркутск 2022

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

ЗАДАНИЕ НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ

По курсу Базы данных

Студенту Арбаковой Анастасии Вячеславовне

Тема проекта Проектирование базы данных и разработка приложения для работы с ней

Исходные данные БД «Авиакомпаний» 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.

Используемая технология: клиент-серверное приложение

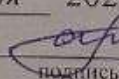
Рекомендуемая литература:

1. Базы данных : учеб. пособие / А. С. Дорофеев; Иркут. гос. техн. ун-т. – Иркутск: Изд-во ИрГТУ, 2008. – 99 с. : а-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.

Графическая часть на — листах.

Дата выдачи задания “ 3 ” октября 2022 г.


Задание получил


подпись

Арбакова А.В.
И.О.Фамилия

Дата представления проекта руководителю

Руководитель курсового проектирования

“ 23 ”

подпись

декабря 2022 г.

Дорофеев А.С.
И.О.Фамилия

Содержание

Введение	4
Основная часть	5
1 Объектная модель задачи.....	5
2 Структура инфологической модели и результаты ее нормализации	10
3 Логическая и физическая модели данных.....	13
3.1 Логическая модель данных	13
3.2 Физическая модель данных	13
4 Описание базы данных на сервере.....	14
5 Формы входных и выходных документов	17
6 Инструкция пользователя	22
7 Описание и результаты тестов	25
Заключение.....	36
Список использованных источников	37
Приложение А Листинг.....	38

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 управления 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 «Авиакомпаний» 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.

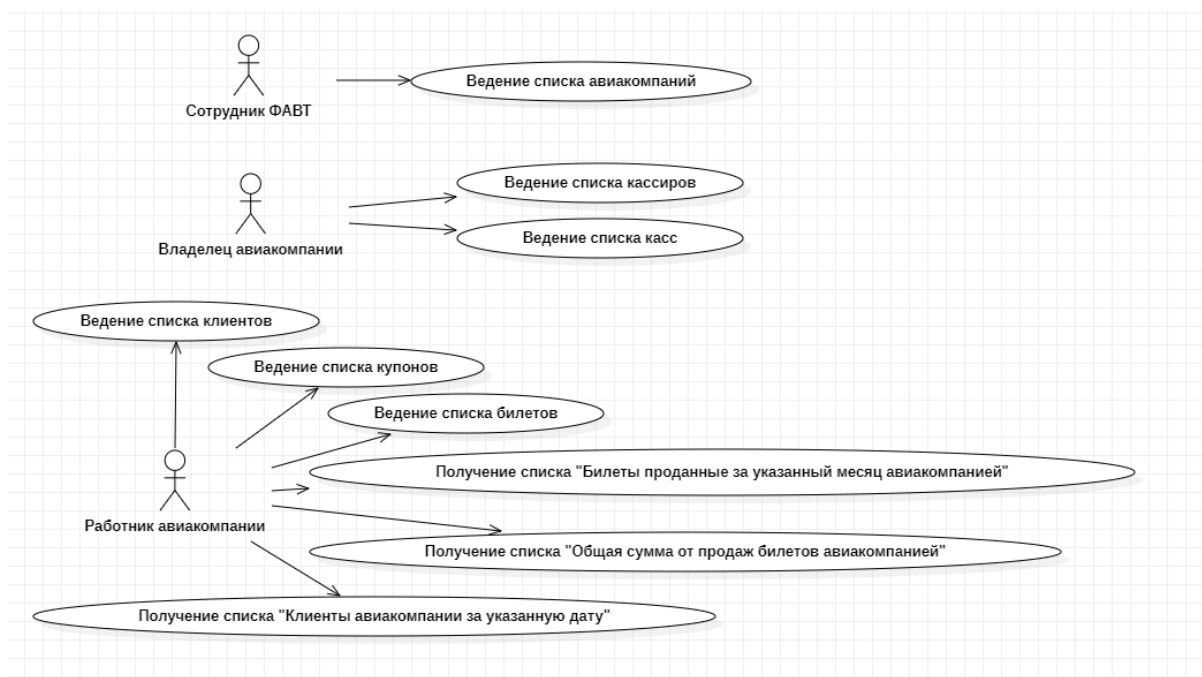


Рисунок 1.1 – Диаграмма сценариев (Use Case)

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

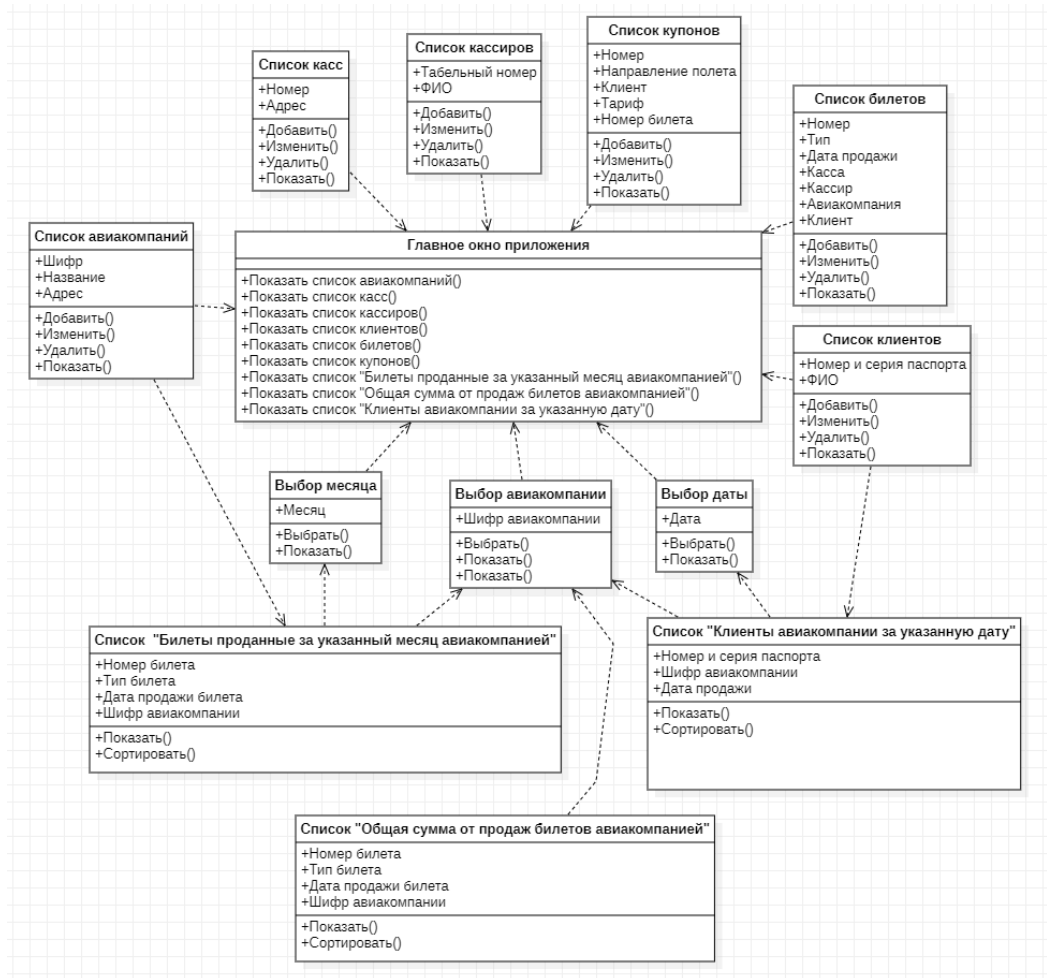


Рисунок 1.2 – Диаграмма классов (Интерфейс приложения)

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

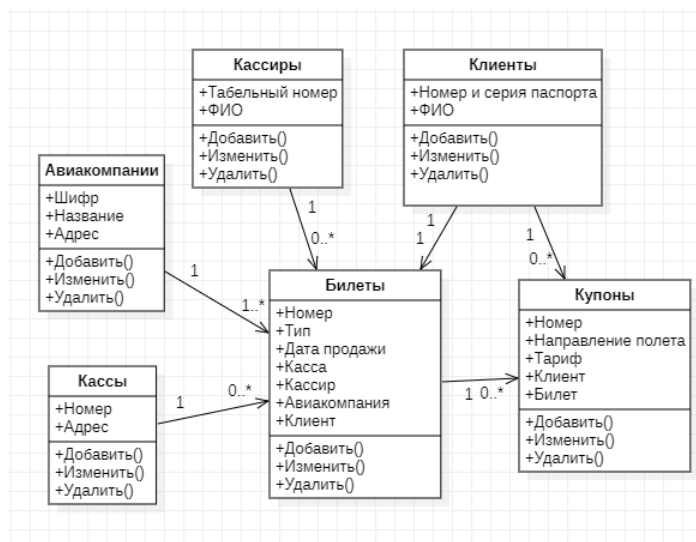


Рисунок 1.3 – Диаграмма классов (Данные)

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

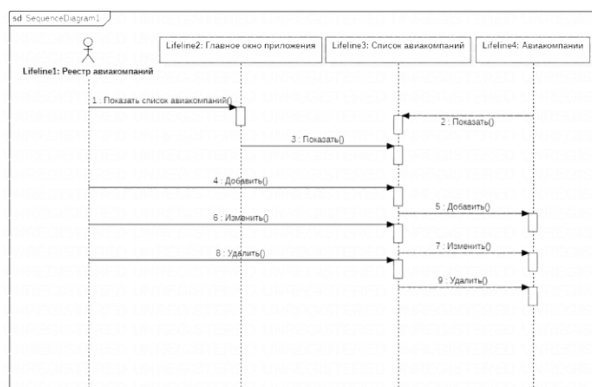


Рисунок 1.4 – Диаграмма последовательностей для прецедента «Ведение списка авиакомпаний»

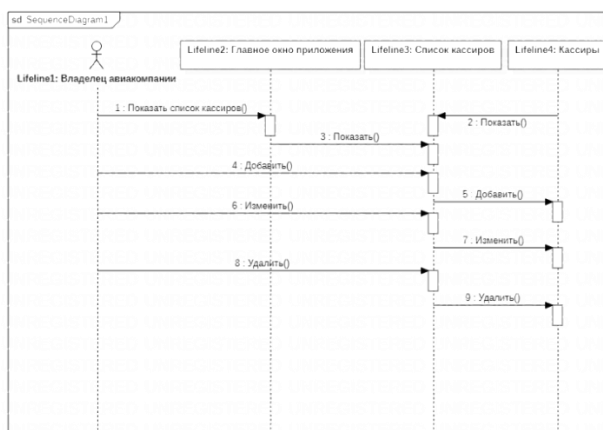


Рисунок 1.5 – Диаграмма последовательностей для прецедента «Ведение списка кассиров»

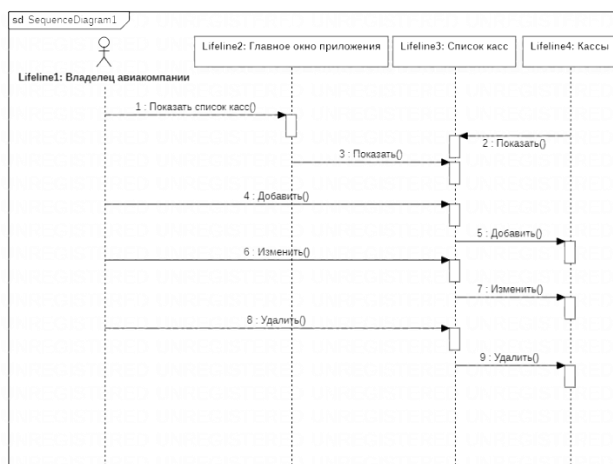


Рисунок 1.6 – Диаграмма последовательностей для прецедента «Ведение списка касс»

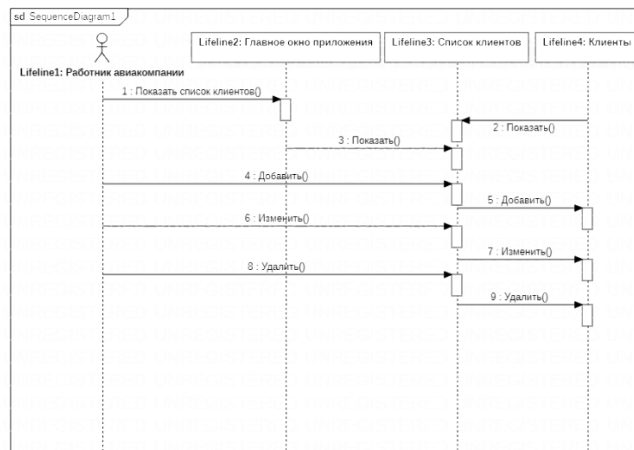


Рисунок 1.7 – Диаграмма последовательностей для прецедента «Ведение списка клиентов»

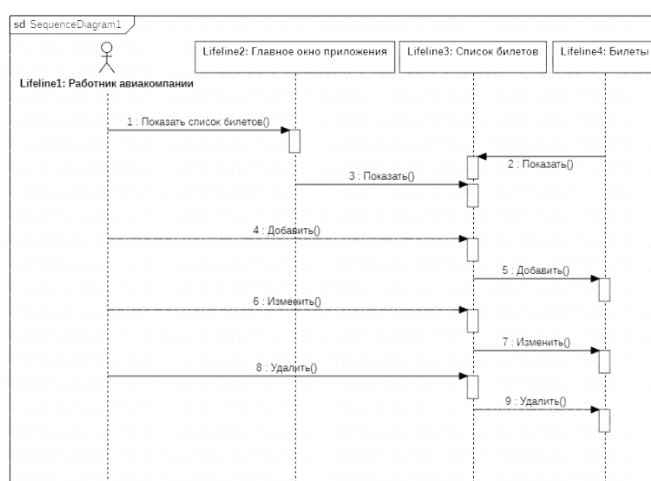


Рисунок 1.8 – Диаграмма последовательностей для прецедента «Ведение списка билетов»

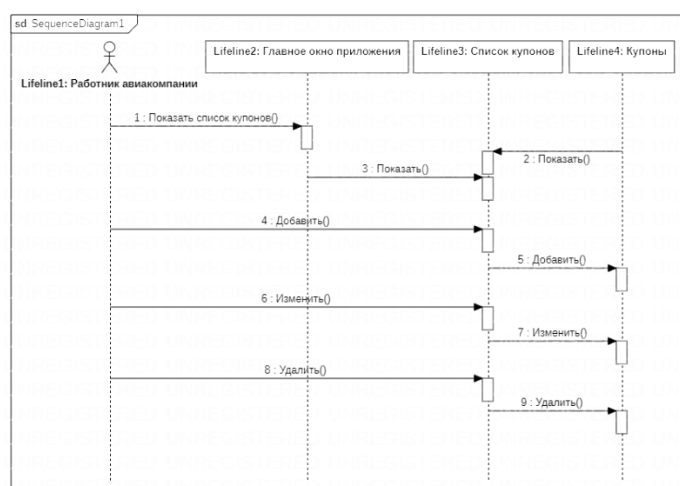


Рисунок 1.9 – Диаграмма последовательностей для прецедента «Ведение списка купонов»

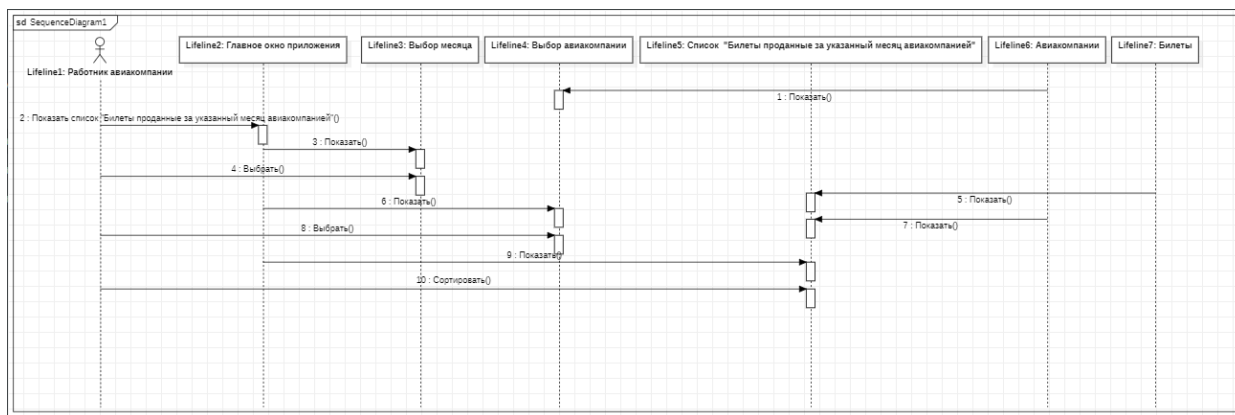


Рисунок 1.10 – Диаграмма последовательностей для прецедента «Получение списка "Билеты, проданные за указанный месяц авиакомпанией"»

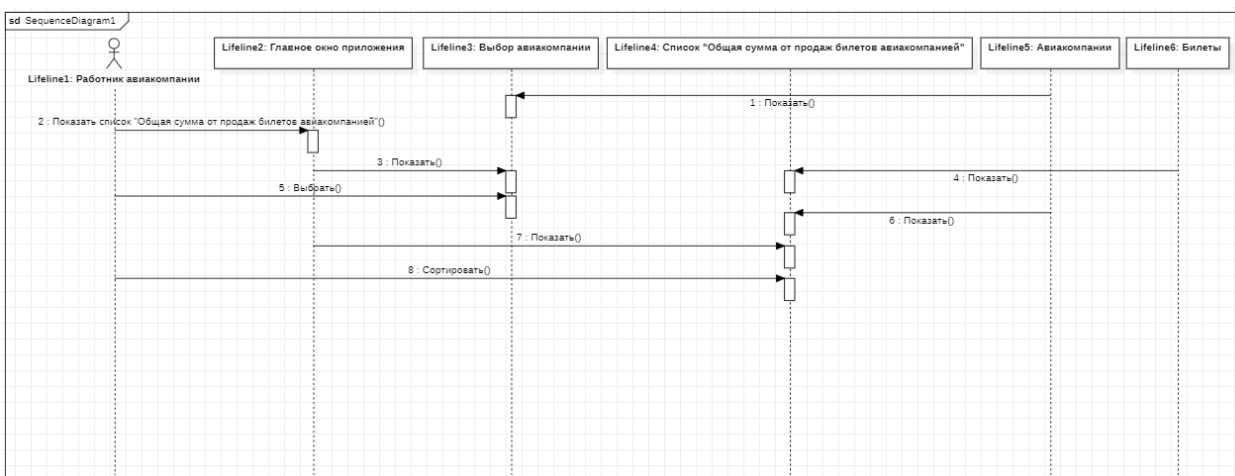


Рисунок 1.11 – Диаграмма последовательностей для прецедента «Получение списка "Общая сумма от продаж билетов авиакомпанией"»

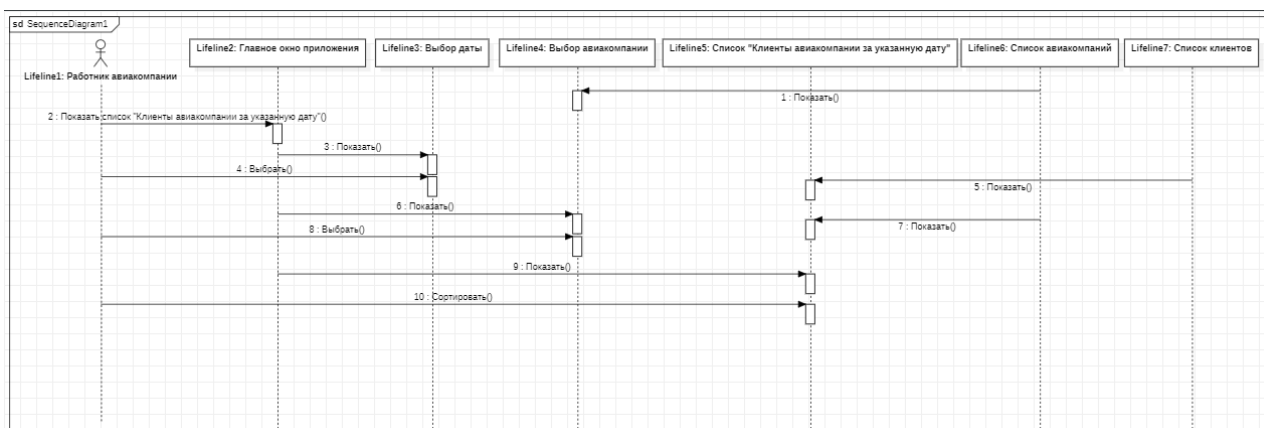


Рисунок 1.12 – Диаграмма последовательностей для прецедента «Получение списка "Клиенты авиакомпании за указанную дату"»

2 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).

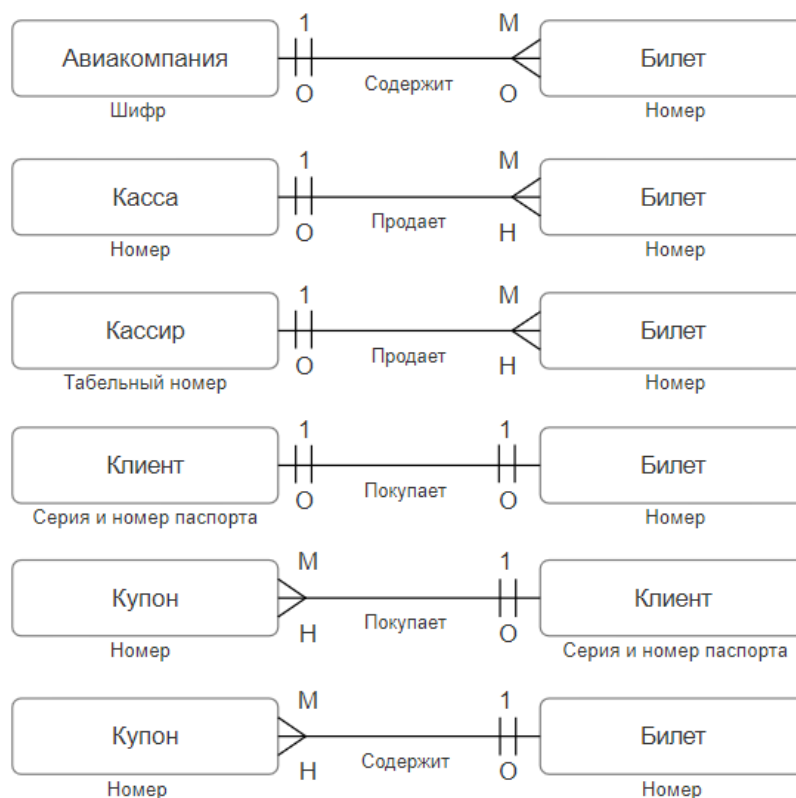


Рисунок 2.1 – Пары сущностей

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

1. По четвертому правилу:

Авиакомпания (Шифр)

Билет (Номер, Шифр авиакомпании (FK))

2. По пятому правилу:

Касса (Номер)

Билет (Номер)

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

3. По пятому правилу:

Кассир (Табельный номер)

Билет (Номер)

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

4. По первому правилу:

Клиент (Номер и серия паспорта)

Билет (Номер)

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

5. По пятому правилу:

Купон (Номер)

Клиент (Номер и серия паспорта)

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

6. 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):

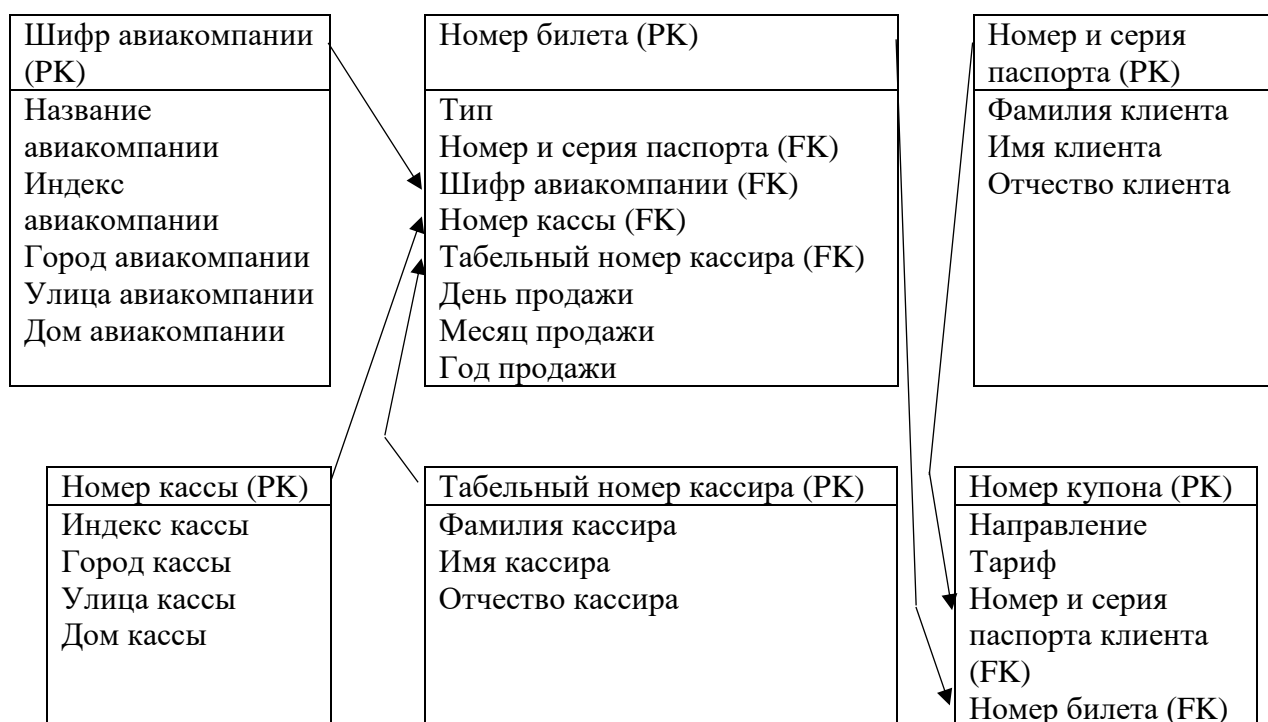


Рисунок 2.2 – Таблицы

3 Логическая и физическая модели данных

3.1 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.

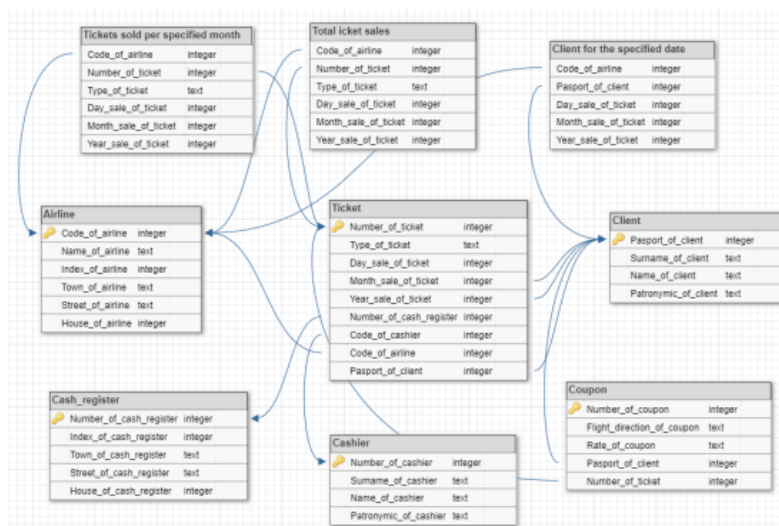


Рисунок 3.1 – Логическая модель данных

3.2 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

4 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))
);

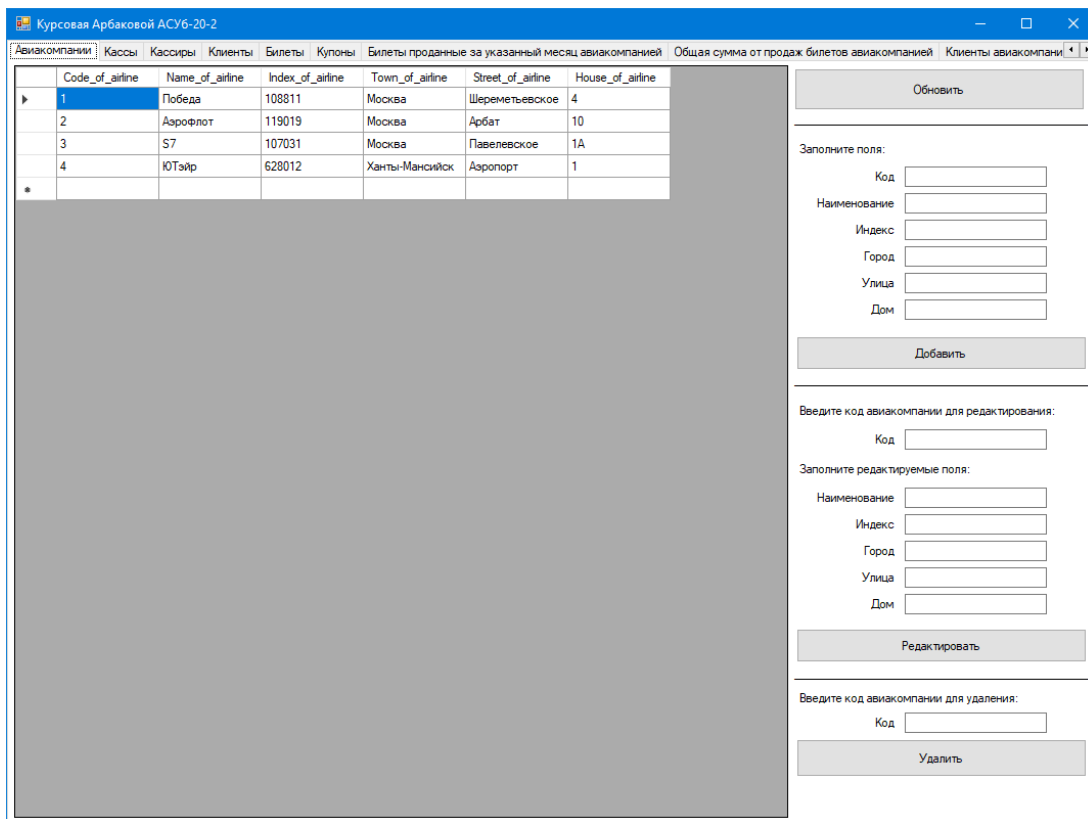
```

);
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.

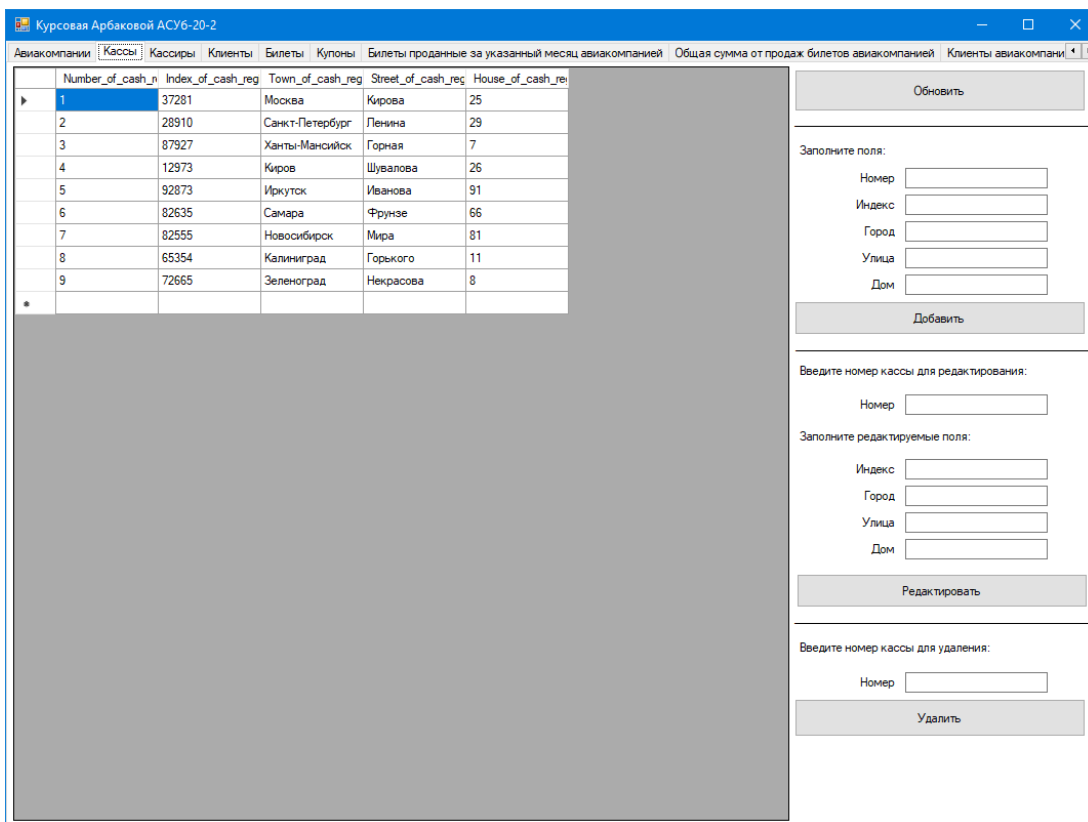
5 Формы входных и выходных документов

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



	Code_of_airline	Name_of_airline	Index_of_airline	Town_of_airline	Street_of_airline	House_of_airline
▶	1	Победа	108811	Москва	Шереметьевское	4
	2	Аэрофлот	119019	Москва	Арбат	10
	3	S7	107031	Москва	Павелевское	1A
	4	ЮТэйр	628012	Ханты-Мансийск	Аэропорт	1
*						

Рисунок 5.1 – Экранная форма «Авиакомпаний»



	Number_of_cash_n	Index_of_cash_reg	Town_of_cash_reg	Street_of_cash_reg	House_of_cash_reg
▶	1	37281	Москва	Кирова	25
	2	28910	Санкт-Петербург	Ленина	29
	3	87927	Ханты-Мансийск	Горная	7
	4	12973	Киров	Шувалова	26
	5	92873	Иркутск	Иванова	91
	6	82635	Самара	Фрунзе	66
	7	82555	Новосибирск	Мира	81
	8	65354	Калининград	Горького	11
	9	72665	Зеленоград	Некрасова	8
*					

Рисунок 5.2 – Экранная форма «Кассы»

Курсовая Арбаковой АСУ6-20-2

Авиакомпания Кассы Кассиры Клиенты Билеты Купоны Билеты проданные за указанный месяц авиакомпанией Общая сумма от продаж билетов авиакомпанией Клиенты авиакомпании

	Code_of_cashier	Surname_of_cashier	Name_of_cashier	Patronymic_of_cashier
▶	1	Иванова	Мария	Ивановна
	2	Сергеева	Елена	Владимировна
	3	Романова	Лилия	Романовна
	4	Пермونتова	Карина	Леонидовна
	5	Павлова	Валерия	Павловна
	6	Шолохов	Геннадий	Геннадьевич
	7	Иванов	Иван	Сергеевич
	8	Другов	Дмитрий	Валерьевич
	9	Родионов	Родион	Иванович
	10	Павлов	Павел	Павлович
*				

Обновить

Заполните поля:

Табельный номер

Фамилия

Имя

Отчество

Добавить

Введите номер кассира для редактирования:

Табельный номер

Заполните редактируемые поля:

Фамилия

Имя

Отчество

Редактировать

Введите номер кассира для удаления:

Табельный номер

Удалить

Рисунок 5.3 – Экранная форма «Кассиры»

Курсовая Арбаковой АСУ6-20-2

Авиакомпания Кассы Кассиры Клиенты Билеты Купоны Билеты проданные за указанный месяц авиакомпанией Общая сумма от продаж билетов авиакомпанией Клиенты авиакомпании

	Pasport_of_client	Surname_of_client	Name_of_client	Patronymic_of_client
▶	1240867700	Васильева	Елизавета	Николаевна
	2100936283	Беляков	Марк	Иванович
	2142352123	Басова	Ева	Денисовна
	2143415233	Горшкова	Александра	Денисовна
	2146867894	Кузьмина	София	Серафимовна
	2190984809	Никитин	Николай	Яковлевич
	2309017233	Белов	Борис	Давидович
	2432465775	Федоров	Сергей	Григорьевич
	2987354635	Иванов	Геннадий	Алексеевич
	3245135675	Дементьева	Анна	Борисовна
	3258768985	Мальшева	Алиса	Максимовна
	4177645386	Григорьев	Тимур	Николаевич
	8273645162	Медведев	Ярослав	Васильевич
	8371625372	Фролов	Савва	Захарович
	9172546372	Дроздова	Милана	Дмитриевна
	9253461827	Киселев	Ярослав	Глебович
	9270147282	Москина	Арина	Ильинична
	9283561723	Егорова	Александра	Алексеевна
	9387018289	Кравцова	Кира	Артёмовна
*				

Обновить

Заполните поля:

Серия и номер паспорта

Фамилия

Имя

Отчество

Добавить

Введите номер и серию паспорта для редактирования:

Номер и серия паспорта

Заполните редактируемые поля:

Фамилия

Имя

Отчество

Редактировать

Введите номер и серию паспорта для удаления:

Номер и серия паспорта

Удалить

Рисунок 5.4 – Экранная форма «Клиенты»

Курсовая Арбаковой АСУ6-20-2

Авиакомпания Кассы Кассиры Клиенты Билеты Купоны Билеты проданные за указанный месяц авиакомпанией Общая сумма от продаж билетов авиакомпанией Клиенты авиакомпании

	Number_of_ticket	Type_of_ticket	Day_sale_of_ticket	Month_sale_of_ticket	Year_sale_of_ticket	Number_of_cashier	Code_of_cashier	C
▶	11421	Oneway	22	6	2023	5	5	1
	12456	Oneway	4	1	2023	5	8	2
	12487	Oneway	15	6	2023	6	6	4
	15412	Oneway	23	2	2022	1	5	3
	17548	Oneway	1	4	2023	1	1	4
	18562	Twoway	23	5	2022	1	5	4
	18754	Oneway	24	6	2023	8	6	1
	23456	Oneway	7	10	2023	4	5	1
	25863	Oneway	28	6	2023	2	5	4
	41512	Twoway	15	2	2023	4	5	4
	42114	Oneway	14	2	2023	9	7	1
	45127	Twoway	12	7	2023	8	6	4
	48596	Oneway	12	12	2023	5	9	1
	48754	Oneway	15	6	2023	5	9	1
	48975	Twoway	15	3	2022	5	4	2
	84568	Oneway	4	3	2021	1	1	1
	87554	Oneway	11	11	2023	8	7	3
	89456	Oneway	25	2	2022	5	4	1
	89542	Twoway	5	6	2020	5	9	4
*								

Обновить

Заполните поля:

Номер билета

Дата продажи: Тип

День Месяц Год

Номер кассы

Табельный номер кассира

Код авиакомпании

Паспорт клиента

Добавить

Введите номер билета для редактирования:

Номер билета

Заполните редактируемые поля:

Дата продажи Тип

День Месяц Год

Номер кассы

Табельный номер кассира

Код авиакомпании

Паспорт клиента

Редактировать

Введите номер билета для удаления:

Номер билета

Удалить

Рисунок 5.5 – Экранная форма «Билеты»

Курсовая Арбаковой АСУ6-20-2

Авиакомпания Кассы Кассиры Клиенты Билеты Купоны Билеты проданные за указанный месяц авиакомпанией Общая сумма от продаж билетов авиакомпанией Клиенты авиакомпании

	Number_of_coupon	Flight_direction_of	Rate_of_coupon	Pasport_of_client	Number_of_ticket
▶	1121	Rome	3	3245135675	15412
	1245	Italia	8	9172546372	48754
	7154	Russia	1	2987354635	84568
	8912	Russia	2	2146867894	48596
	8945	Rome	2	1240867700	89542
	8956	Russia	1	2100936283	25863
*					

Обновить

Заполните поля:

Номер купона

Направление полета

Тариф

Паспорт клиента

Номер билета

Добавить

Введите номер купона для редактирования:

Номер купона

Заполните редактируемые поля:

Направление полета

Тариф

Паспорт клиента

Номер билета

Редактировать

Введите номер купона для удаления:

Номер купона

Удалить

Рисунок 5.6 – Экранная форма «Купоны»

Курсовая Арбаковой АСУ6-20-2

Авиакомпания Кассы Кассиры Клиенты Билеты Купоны Билеты проданные за указанный месяц авиакомпанией Общая сумма от продаж билетов авиакомпанией Клиенты авиакомпании

Введите код авиакомпании: 1 Выберите месяц: 1 Выберите год: 2023 Найти Иск

Number_of_ticket	Type_of_ticket	Day_sale_of_ticket	Month_sale_of_ticket	Year_sale_of_ticket	Code_of_airline
11421	Oneway	22	6	2023	1
12456	Oneway	4	1	2023	2
12487	Oneway	15	6	2023	4
15412	Oneway	23	2	2022	3
17548	Oneway	1	4	2023	4
18562	Twoway	23	5	2022	4
18754	Oneway	24	6	2023	1
23456	Oneway	7	10	2023	1
25863	Oneway	28	6	2023	4
41512	Twoway	15	2	2023	4
42114	Oneway	14	2	2023	1
45127	Twoway	12	7	2023	4
48596	Oneway	12	12	2023	1
48754	Oneway	15	6	2023	1
48975	Twoway	15	3	2022	2
84568	Oneway	4	3	2021	1
87554	Oneway	11	11	2023	3
89456	Oneway	25	2	2022	1
89542	Twoway	5	6	2020	4

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

Курсовая Арбаковой АСУ6-20-2

Авиакомпания Кассы Кассиры Клиенты Билеты Купоны Билеты проданные за указанный месяц авиакомпанией Общая сумма от продаж билетов авиакомпанией Клиенты авиакомпании

Введите код авиакомпании: 1 Билетов продано авиакомпанией: Иск

Number_of_ticket	Type_of_ticket	Day_sale_of_ticket	Month_sale_of_ticket	Year_sale_of_ticket	Code_of_airline
11421	Oneway	22	6	2023	1
12456	Oneway	4	1	2023	2
12487	Oneway	15	6	2023	4
15412	Oneway	23	2	2022	3
17548	Oneway	1	4	2023	4
18562	Twoway	23	5	2022	4
18754	Oneway	24	6	2023	1
23456	Oneway	7	10	2023	1
25863	Oneway	28	6	2023	4
41512	Twoway	15	2	2023	4
42114	Oneway	14	2	2023	1
45127	Twoway	12	7	2023	4
48596	Oneway	12	12	2023	1
48754	Oneway	15	6	2023	1
48975	Twoway	15	3	2022	2
84568	Oneway	4	3	2021	1
87554	Oneway	11	11	2023	3
89456	Oneway	25	2	2022	1
89542	Twoway	5	6	2020	4

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

Курсовая Арбаковой АСУ6-20-2

Кассиры Клиенты Билеты Купоны Билеты проданные за указанный месяц авиакомпанией Общая сумма от продаж билетов авиакомпанией Клиенты авиакомпании за указанную дату

Введите код авиакомпании: 1 Выберите день: 1 Выберите месяц: 1 Выберите год: 2023 Найти Иск

	Passport_of_client	Day_sale_of_ticket	Month_sale_of_tick	Year_sale_of_ticker	Code_of_airline
▶	3258768985	22	6	2023	1
	2309017233	4	1	2023	2
	4177645386	15	6	2023	4
	1240867700	23	2	2022	3
	2987354635	1	4	2023	4
	2100936283	23	5	2022	4
	9253461827	24	6	2023	1
	2432465775	7	10	2023	1
	2142352123	28	6	2023	4
	9270147282	15	2	2023	4
	8273645162	14	2	2023	1
	3245135675	12	7	2023	4
	2143415233	12	12	2023	1
	9283561723	15	6	2023	1
	8371625372	15	3	2022	2
	2146867894	4	3	2021	1
	9172546372	11	11	2023	3
	9387018289	25	2	2022	1
	2190984809	5	6	2020	4
*					

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

6 Инструкция пользователя

1. 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.

2. Экранная форма «Авиакомпаний»

Следом откроется окно «Курсовая» с открытой экранной формой «Авиакомпаний» (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.

2.1. Кнопка «Обновить»

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

2.2. 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».

2.3. 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».

2.4. 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».

3. 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.

3.1. Кнопка «Обновить» – выполняется аналогично пункту 2.1.

3.2. Кнопка «Добавить» – выполняется аналогично пункту 2.2.

3.3. Кнопка «Редактировать» – выполняется аналогично пункту 2.3.

3.4. Кнопка «Удалить» – выполняется аналогично пункту 2.4.

4. Экранная форма «Кассиры»

При выборе пункта «Кассиры», в верхней панели, откроется окно формы «Кассиры» (рис. 5.3). В огражденном 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.

4.1. Кнопка «Обновить» – выполняется аналогично пункту 2.1.

4.2. Кнопка «Добавить» – выполняется аналогично пункту 2.2.

4.3. Кнопка «Редактировать» – выполняется аналогично пункту 2.3.

4.4. Кнопка «Удалить» – выполняется аналогично пункту 2.4.

5. Экранная форма «Клиенты»

При выборе пункта «Клиенты», в верхней панели, откроется окно формы «Клиенты» (рис. 5.4). В огражденном 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.

5.1. Кнопка «Обновить» – выполняется аналогично пункту 2.1.

5.2. Кнопка «Добавить» – выполняется аналогично пункту 2.2.

5.3. Кнопка «Редактировать» – выполняется аналогично пункту 2.3.

5.4. Кнопка «Удалить» – выполняется аналогично пункту 2.4.

6. Экранная форма «Билеты»

При выборе пункта «Билеты», в верхней панели, откроется окно формы «Билеты» (рис. 5.5). В огражденном 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.

6.1. Кнопка «Обновить» – выполняется аналогично пункту 2.1.

6.2. Кнопка «Добавить» – выполняется аналогично пункту 2.2.

6.3. Кнопка «Редактировать» – выполняется аналогично пункту 2.3.

6.4. Кнопка «Удалить» – выполняется аналогично пункту 2.4.

7. 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.

7.1. Кнопка «Обновить» – выполняется аналогично пункту 2.1.

7.2. Кнопка «Добавить» – выполняется аналогично пункту 2.2.

7.3. Кнопка «Редактировать» – выполняется аналогично пункту 2.3.

7.4. Кнопка «Удалить» – выполняется аналогично пункту 2.4.

8. 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.

9. 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.

10. 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.

7 Описание и результаты тестов

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

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

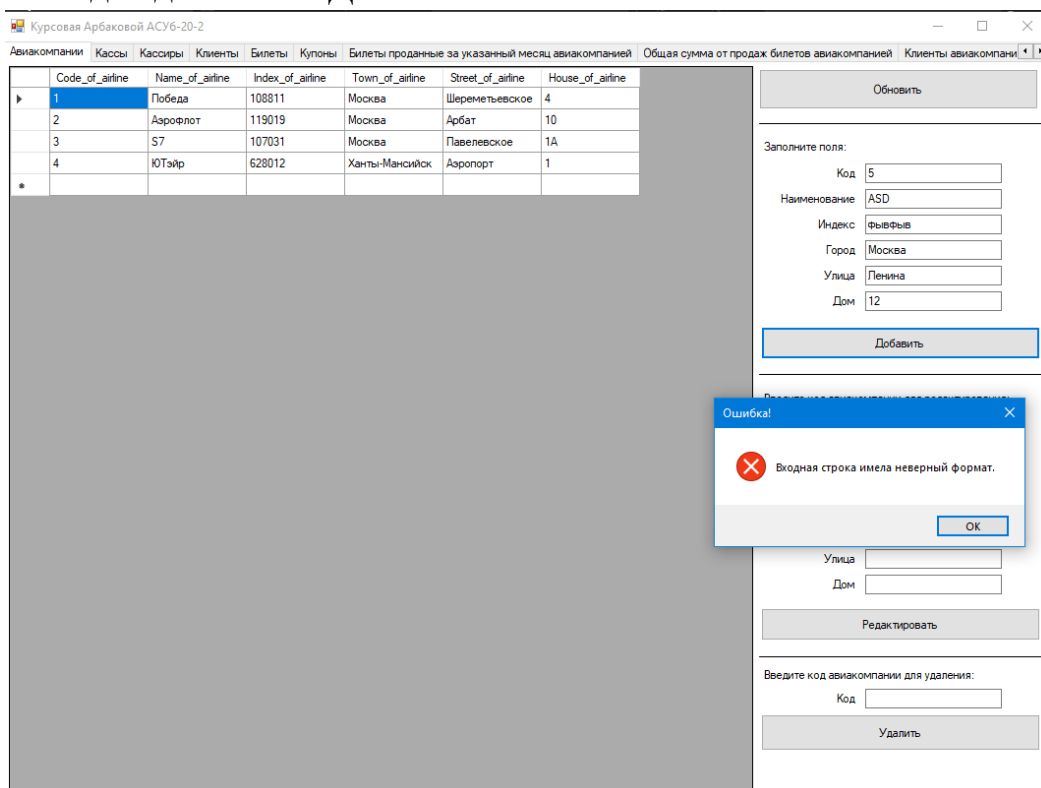


Рисунок 7.1 – Неверный ввод для действия «Добавить»

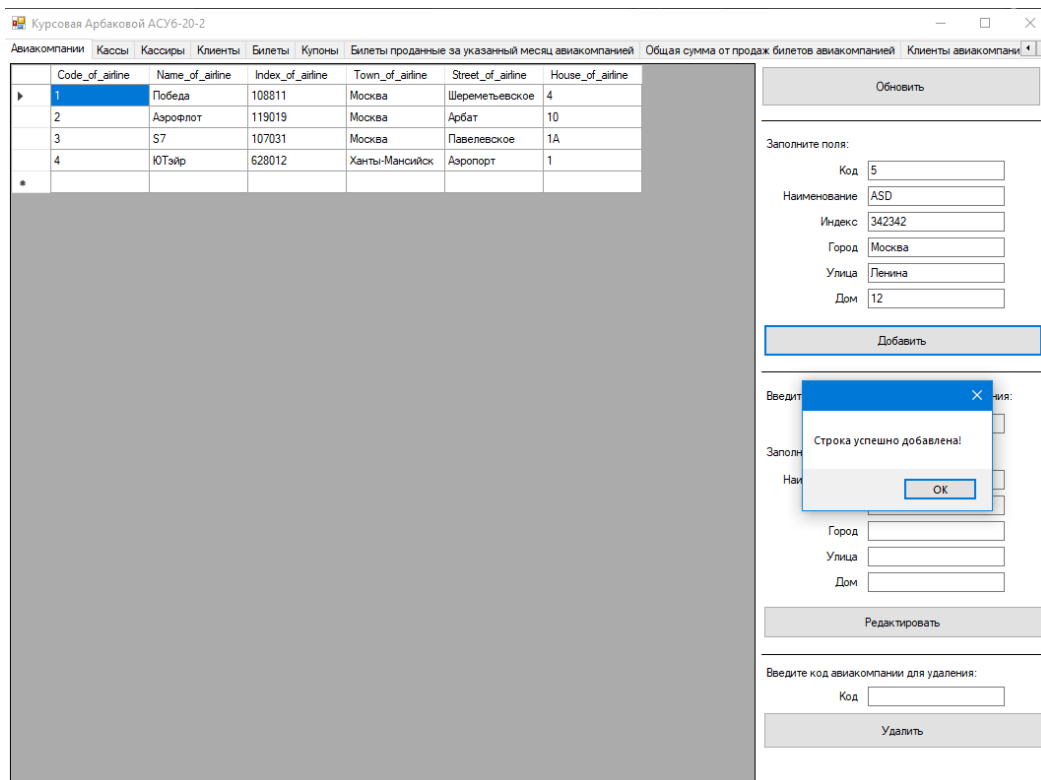


Рисунок 7.2 – Верный ввод для действия «Добавить»

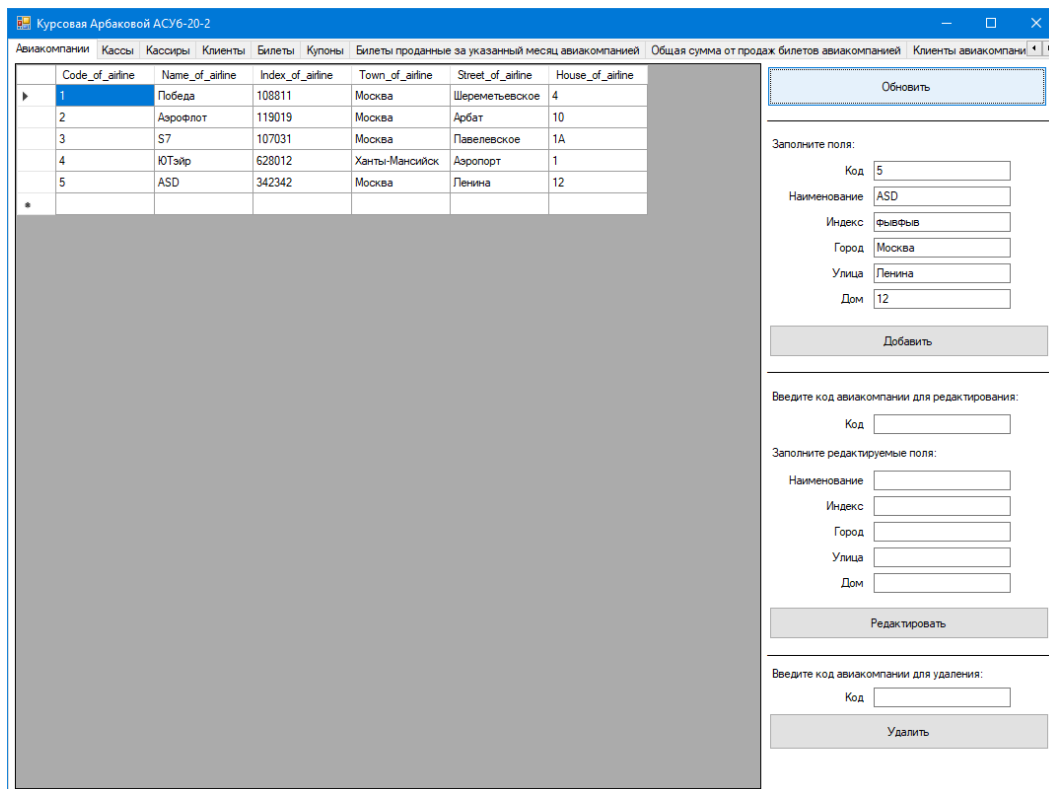


Рисунок 7.3 – Действие «Обновить»

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

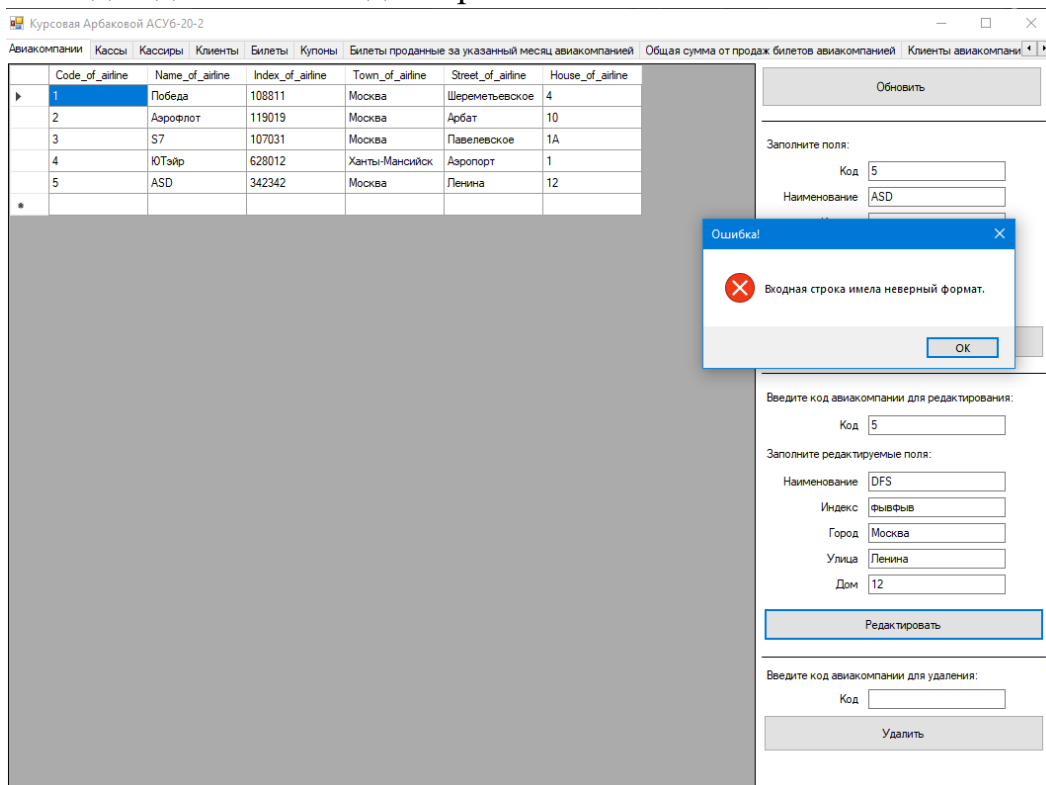


Рисунок 7.4 – Неверный ввод для действия «Редактировать»

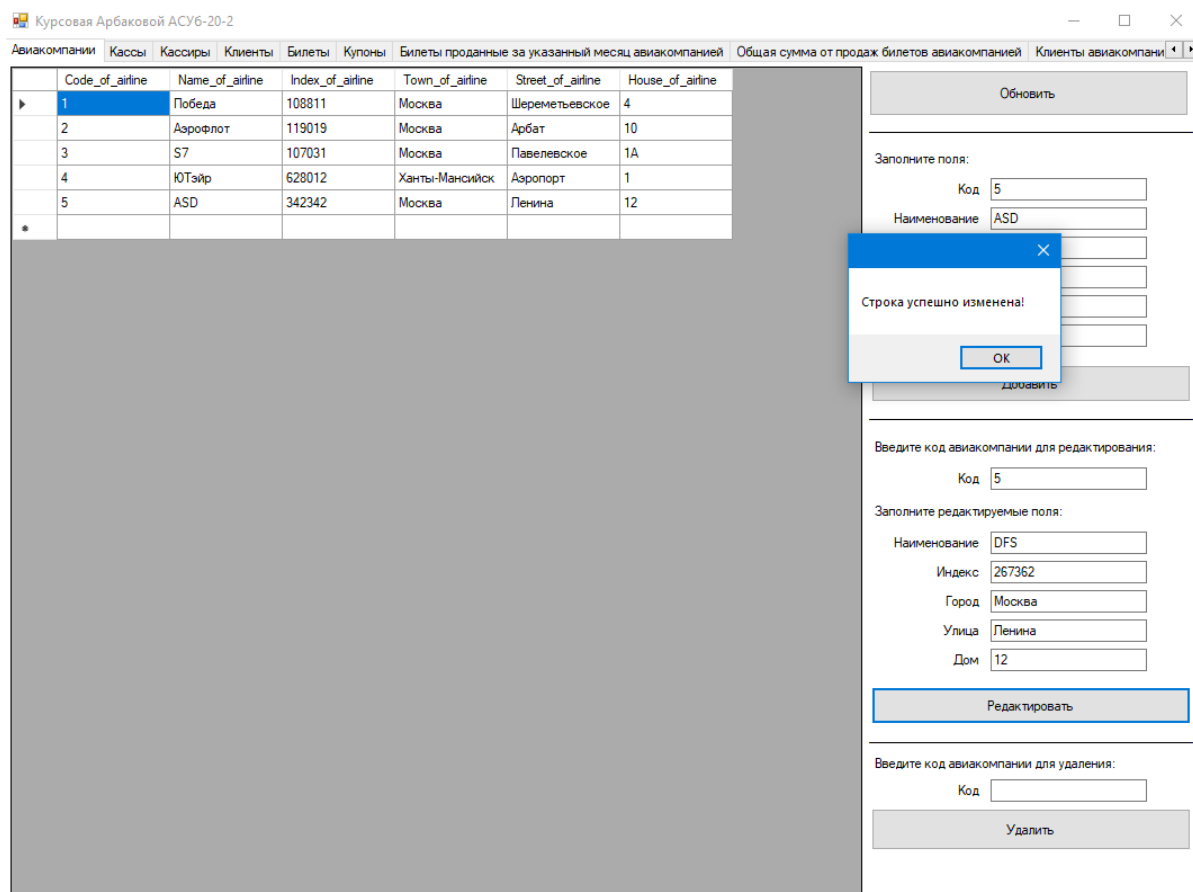


Рисунок 7.5 – Верный ввод для действия «Редактировать»

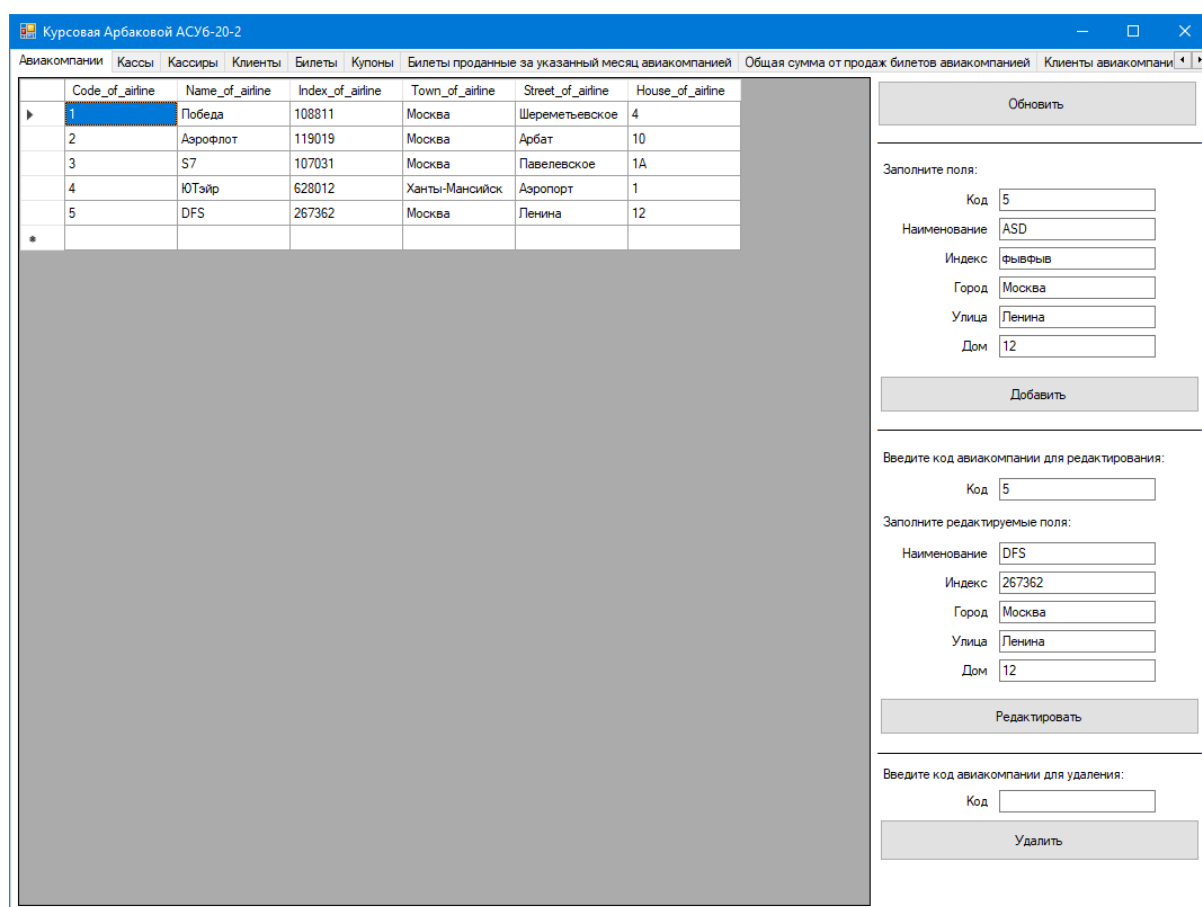


Рисунок 7.6 – Действие «Обновить»

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

Курсовая Арбаковой АСУ6-20-2

Авиакомпания Кассы Кассиры Клиенты Билеты Купоны Билеты проданные за указанный месяц авиакомпанией Общая сумма от продаж билетов авиакомпании Клиенты авиакомпании

	Code_of_airline	Name_of_airline	Index_of_airline	Town_of_airline	Street_of_airline	House_of_airline
▶	1	Победа	108811	Москва	Шереметьевское	4
	2	Аэрофлот	119019	Москва	Арбат	10
	3	S7	107031	Москва	Павелевское	1А
	4	ЮТэйр	628012	Ханты-Мансийск	Аэропорт	1
	5	DFS	267362	Москва	Ленина	12
*						

Обновить

Заполните поля:

Код

Наименование

Индекс

Город

Улица

Дом

Добавить

Ошибка!

✖ Вводная строка имела неверный формат.

OK

Улица

Дом

Редактировать

Введите код авиакомпании для удаления:

Код

Удалить

Рисунок 7.7 – Неверный ввод для действия «Удалить»

Курсовая Арбаковой АСУ6-20-2

Авиакомпания Кассы Кассиры Клиенты Билеты Купоны Билеты проданные за указанный месяц авиакомпанией Общая сумма от продаж билетов авиакомпании Клиенты авиакомпании

	Code_of_airline	Name_of_airline	Index_of_airline	Town_of_airline	Street_of_airline	House_of_airline
▶	1	Победа	108811	Москва	Шереметьевское	4
	2	Аэрофлот	119019	Москва	Арбат	10
	3	S7	107031	Москва	Павелевское	1А
	4	ЮТэйр	628012	Ханты-Мансийск	Аэропорт	1
	5	DFS	267362	Москва	Ленина	12
*						

Обновить

Заполните поля:

Код

Наименование

Индекс

Город

Улица

Дом

Добавить

Введите код авиакомпании для редактирования:

Зап

Строка успешно удалена!

OK

Улица

Дом

Редактировать

Введите код авиакомпании для удаления:

Код

Удалить

Рисунок 7.8 – Верный ввод для действия «Удалить»

Курсовая Арбаковой АСУ6-20-2

Авиакомпании Кассы Кассиры Клиенты Билеты Купоны Билеты проданные за указанный месяц авиакомпании Общая сумма от продаж билетов авиакомпании Клиенты авиакомпании

	Code_of_airline	Name_of_airline	Index_of_airline	Town_of_airline	Street_of_airline	House_of_airline
▶	1	Победа	108811	Москва	Шереметьевское	4
	2	Аэрофлот	119019	Москва	Арбат	10
	3	S7	107031	Москва	Павелевское	1A
	4	ЮТэйр	628012	Ханты-Мансийск	Аэропорт	1
*						

Обновить

Заполните поля:

Код: 5

Наименование: ASD

Индекс: фывфыв

Город: Москва

Улица: Ленина

Дом: 12

Добавить

Введите код авиакомпании для редактирования:

Код: 5

Заполните редактируемые поля:

Наименование: DFS

Индекс: 267362

Город: Москва

Улица: Ленина

Дом: 12

Редактировать

Введите код авиакомпании для удаления:

Код: 5

Удалить

Рисунок 7.9 – Действие «Обновить»

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

Курсовая Арбаковой АСУ6-20-2

Авиакомпании Кассы Кассиры Клиенты Билеты Купоны Билеты проданные за указанный месяц авиакомпании Общая сумма от продаж билетов авиакомпании Клиенты авиакомпании

Введите код авиакомпании: 1 Выберите месяц: 6 Выберите год: 2023 Найти Иск

	Number_of_ticket	Type_of_ticket	Day_sale_of_ticket	Month_sale_of_tick	Year_sale_of_tick	Code_of_airline
▶	11421	Oneway	22	6	2023	1
	18754	Oneway	24	6	2023	1
	48754	Oneway	15	6	2023	1
*						

Рисунок 7.10.1 – Действие кнопки «Найти» с указанными параметрами

Курсовая Арбаковой АСУ6-20-2

Авиакомпании Кассы Кассиры Клиенты Билеты Купоны Билеты проданные за указанный месяц авиакомпании Общая сумма от продаж билетов авиакомпанией Клиенты авиакомпании

Введите код авиакомпании: 4 Выберите месяц: 6 Выберите год: 2023 Найти Иск

	Number_of_ticket	Type_of_ticket	Day_sale_of_ticket	Month_sale_of_tick	Year_sale_of_tick	Code_of_airline
▶	12487	Oneway	15	6	2023	4
	25863	Oneway	28	6	2023	4
*						

Рисунок 7.10.2 – Действие кнопки «Найти» с указанными параметрами

Курсовая Арбаковой АСУ6-20-2

Авиакомпании Кассы Кассиры Клиенты Билеты Купоны Билеты проданные за указанный месяц авиакомпании Общая сумма от продаж билетов авиакомпанией Клиенты авиакомпании

Введите код авиакомпании: 2 Выберите месяц: 1 Выберите год: 2023 Найти Иск

	Number_of_ticket	Type_of_ticket	Day_sale_of_ticket	Month_sale_of_tick	Year_sale_of_tick	Code_of_airline
▶	12456	Oneway	4	1	2023	2
*						

Рисунок 7.10.3 – Действие кнопки «Найти» с указанными параметрами

Курсовая Арбаковой АСУ6-20-2

Авиакомпании Кассы Кассиры Клиенты Билеты Купоны Билеты проданные за указанный месяц авиакомпанией Общая сумма от продаж билетов авиакомпанией Клиенты авиакомпании

Введите код авиакомпании: 2 Выберите месяц: 1 Выберите год: 2023 Найти Иск

	Number_of_ticket	Type_of_ticket	Day_sale_of_ticket	Month_sale_of_tick	Year_sale_of_ticket	Code_of_airline
▶	11421	Oneway	22	6	2023	1
	12456	Oneway	4	1	2023	2
	12487	Oneway	15	6	2023	4
	15412	Oneway	23	2	2022	3
	17548	Oneway	1	4	2023	4
	18562	Twoway	23	5	2022	4
	18754	Oneway	24	6	2023	1
	23456	Oneway	7	10	2023	1
	25863	Oneway	28	6	2023	4
	41512	Twoway	15	2	2023	4
	42114	Oneway	14	2	2023	1
	45127	Twoway	12	7	2023	4
	48596	Oneway	12	12	2023	1
	48754	Oneway	15	6	2023	1
	48975	Twoway	15	3	2022	2
	84568	Oneway	4	3	2021	1
	87554	Oneway	11	11	2023	3
	89456	Oneway	25	2	2022	1
	89542	Twoway	5	6	2020	4
*						

Рисунок 7.11 – Действие кнопки «Иск»

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

Курсовая Арбаковой АСУ6-20-2

Авиакомпании Кассы Кассиры Клиенты Билеты Купоны Билеты проданные за указанный месяц авиакомпанией Общая сумма от продаж билетов авиакомпанией Клиенты авиакомпании

Введите код авиакомпании: 11 Билетов продано авиакомпанией: 8 Иск

	Number_of_ticket	Type_of_ticket	Day_sale_of_ticket	Month_sale_of_tick	Year_sale_of_ticket	Code_of_airline
▶	11421	Oneway	22	6	2023	1
	18754	Oneway	24	6	2023	1
	23456	Oneway	7	10	2023	1
	42114	Oneway	14	2	2023	1
	48596	Oneway	12	12	2023	1
	48754	Oneway	15	6	2023	1
	84568	Oneway	4	3	2021	1
	89456	Oneway	25	2	2022	1
*						

Рисунок 7.12.1 – Подсчет проданных билетов

Курсовая Арбаковой АСУ6-20-2

Авиакомпании Кассы Кассиры Клиенты Билеты Купоны Билеты проданные за указанный месяц авиакомпании Общая сумма от продаж билетов авиакомпании Клиенты авиакомпании

Введите код авиакомпании: 4

Билетов продано авиакомпании: 7

Count_sale_of_tick

Исх

	Number_of_ticket	Type_of_ticket	Day_sale_of_ticket	Month_sale_of_tick	Year_sale_of_tick	Code_of_airline
▶	12487	Oneway	15	6	2023	4
	17548	Oneway	1	4	2023	4
	18562	Twoway	23	5	2022	4
	25863	Oneway	28	6	2023	4
	41512	Twoway	15	2	2023	4
	45127	Twoway	12	7	2023	4
	89542	Twoway	5	6	2020	4
*						

Рисунок 7.12.2 – Подсчет проданных билетов

Курсовая Арбаковой АСУ6-20-2

Кассиры Клиенты Билеты Купоны Билеты проданные за указанный месяц авиакомпании Общая сумма от продаж билетов авиакомпании Клиенты авиакомпании за указанную дату

Введите код авиакомпании: 1

Билетов продано авиакомпании:

Исх

	Number_of_ticket	Type_of_ticket	Day_sale_of_ticket	Month_sale_of_tick	Year_sale_of_tick	Code_of_airline
▶	11421	Oneway	22	6	2023	1
	12456	Oneway	4	1	2023	2
	12487	Oneway	15	6	2023	4
	15412	Oneway	23	2	2022	3
	17548	Oneway	1	4	2023	4
	18562	Twoway	23	5	2022	4
	18754	Oneway	24	6	2023	1
	23456	Oneway	7	10	2023	1
	25863	Oneway	28	6	2023	4
	41512	Twoway	15	2	2023	4
	42114	Oneway	14	2	2023	1
	45127	Twoway	12	7	2023	4
	48596	Oneway	12	12	2023	1
	48754	Oneway	15	6	2023	1
	48975	Twoway	15	3	2022	2
	84568	Oneway	4	3	2021	1
	87554	Oneway	11	11	2023	3
	89456	Oneway	25	2	2022	1
	89542	Twoway	5	6	2020	4
*						

Рисунок 7.13 – Действие кнопки «Исх»

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

Курсовая Арбаковой АСУ6-20-2

Кассиры Клиенты Билеты Купоны Билеты проданные за указанный месяц авиакомпанией Общая сумма от продаж билетов авиакомпанией Клиенты авиакомпании за указанную дату

Введите код авиакомпании: 1 Выберите день: 15 Выберите месяц: 6 Выберите год: 2023 [Найти] [Иск]

	Pasport_of_client	Day_sale_of_ticket	Month_sale_of_tick	Year_sale_of_tick	Code_of_airline
▶	9283561723	15	6	2023	1
*					

Рисунок 7.14.1 – Действие кнопки «Найти» с указанными параметрами

Курсовая Арбаковой АСУ6-20-2

Кассиры Клиенты Билеты Купоны Билеты проданные за указанный месяц авиакомпанией Общая сумма от продаж билетов авиакомпанией Клиенты авиакомпании за указанную дату

Введите код авиакомпании: 4 Выберите день: 5 Выберите месяц: 6 Выберите год: 2020 [Найти] [Иск]

	Pasport_of_client	Day_sale_of_ticket	Month_sale_of_tick	Year_sale_of_tick	Code_of_airline
▶	2190984809	5	6	2020	4
*					

Рисунок 7.14.2 – Действие кнопки «Найти» с указанными параметрами

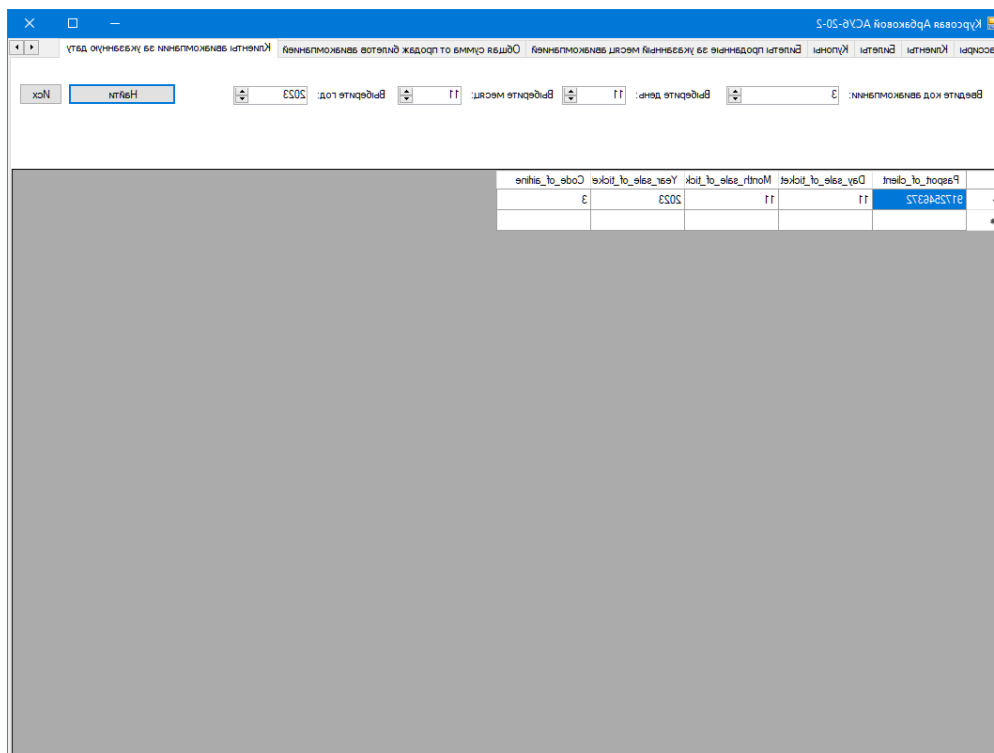


Рисунок 7.14.3 – Действие кнопки «Найти» с указанными параметрами

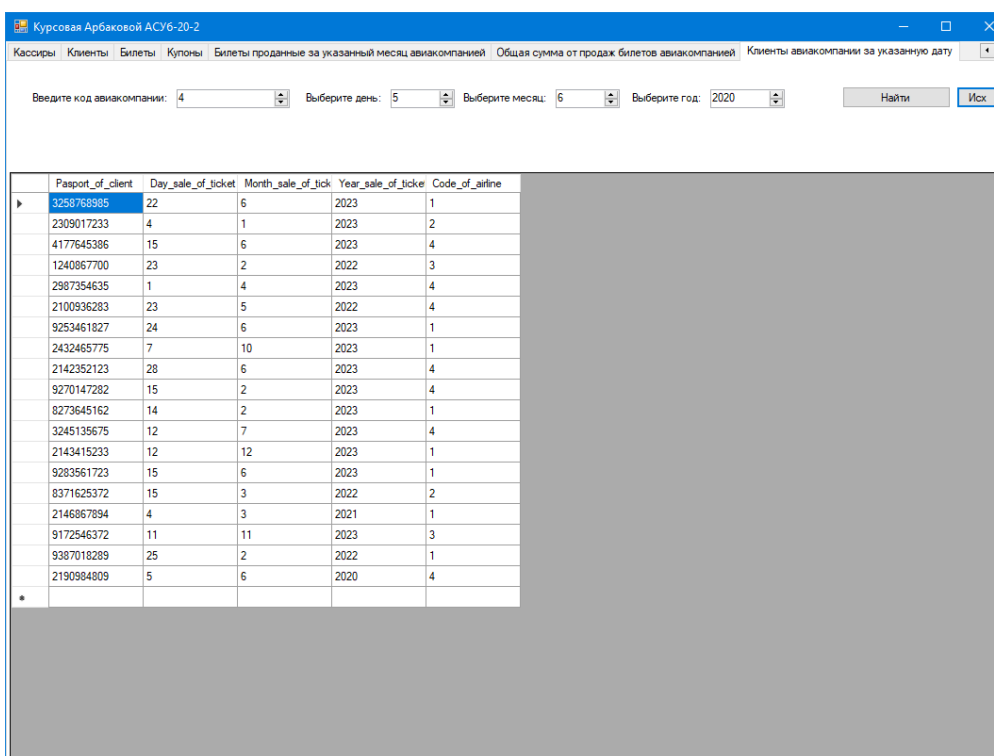


Рисунок 7.15 – Действие кнопки «Исх»

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 управления 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 технологией 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. : а-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 Studio 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);

```



```

        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!");
    }
}

```



```

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}";
}
```

```

//////////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];
    }
}
}

```