Оглавление

[1.Задание на разработку Курсовой Работы 4](#_Toc453196262)

[2.Перечень сокращений 4](#_Toc453196263)

[3.Введение 4](#_Toc453196264)

[4. Анализ предметной области 6](#_Toc453196265)

[5.Техническое задание на разработку ИС 8](#_Toc453196266)

[6. Концептуальная модель данных 9](#_Toc453196267)

[7. Логическая модель данных 11](#_Toc453196268)

[8. Физическая модель данных 13](#_Toc453196269)

[9. SQL-код и запросы для работы с базой данных 17](#_Toc453196270)

[10. Формы приложения 18](#_Toc453196271)

[11.Тестирование приложения 22](#_Toc453196272)

[12.Заключение 31](#_Toc453196273)

[Список использованных источников 32](#_Toc453196274)

[Приложение . 33](#_Toc453196275)

## 1.Задание на разработку Курсовой Работы

Создать автоматизированную информационную систему отдела кадров оздоровительного учреждения.

## 2.Перечень сокращений

АИС - автоматизированная информационная система.

СУБД – система управления базами данных.

БД - база данных.

OC- операционная система.

SQL запрос – команда на языке SQL, производящая некоторое изменение данных в базе данных.

SQL – язык структурированных запросов.

## 3.Введение

Во время работы аэропорта, как и в любой другой области, накапливается большое количество информации, в которой трудно найти необходимые данные. Для решения этой проблемы создается автоматизированная информационная система. Она позволит ускорить и упростить поиск нужной информации. Пользователю АИС будет проще понимать общую картину работы фирмы. Система может автоматически систематизировать и группировать, что позволят получать точные отчеты о работе предприятия. Использование баз данных для такой АИС оправдано тем, что они позволяют структурированного хранить данные, в них удобно выполнять различные запросы для работы с этими данными, также СУБД позволяет строить отчеты, которые могут группировать и выводить информацию в любом необходимом пользователю виде.

*MS SQL* – реляционная СУБД, функционирующая под управлением ОС Windows и имеющая интерфейс приложений Windows.

Основным компонентом является *БД*, которая включает: таблицы, запросы.

Данные хранятся в *Таблицах.* Понятие «таблица» соответствует понятию «таблица», определенному для реляционных баз данных. Для данных в таблицах можно выполнять следующие операции:

* Select(получить данные)
* Insert(добавить данные)
* Delete(удалить запись)
* Update(редактировать запись)

*Запрос* – команда на языке SQL, производящая некоторое изменение данных в базе данных.

Цели:

* Проанализировать предметную область;
* Построить концептуальную модель данных;
* Создать логическую модель данных;
* Разработать физическую модель данных;
* Разработка SQL запросов для базы данных.
* Ускорить процесс поиска информации
* Ускорить процесс приема и обработки данных

## 4. Анализ предметной области

Актуальность. Основные идеи современной информационной технологии базируются на концепции, согласно которой данные должны быть организованы в базы данных с целью адекватного отображения изменяющегося реального мира и удовлетворения информационных потребностей пользователей. Эти базы данных создаются и функционируют под управлением специальных программных комплексов, называемых системами управления базами данных (СУБД).

Увеличение объема и структурной сложности хранимых данных, расширение круга пользователей информационных систем привели к широкому распространению наиболее удобных и сравнительно простых для понимания реляционных (табличных) СУБД. Для обеспечения одновременного доступа к данным множества пользователей, нередко расположенных достаточно далеко друг от друга и от места хранения баз данных, созданы сетевые мультипользовательские версии БД основанных на реляционной структуре. В них тем или иным путем решаются специфические проблемы параллельных процессов, целостности (правильности) и безопасности данных, а также санкционирования доступа.

AИC предназначена для администраторов и кассиров аэропорта. Благодаря AИC, сотрудники отдела кадров смогу быстрее выполнять необходимые действия, AИC позволит пользователям система отсортировать и сгруппировать уже имеющиеся данные, а также сделать процесс добавления новых данных быстрее и проще. AИC позволяет производить поиск в большом объеме данных многократно быстрее, чем если бы это проводилось в ручную по базе, хранящейся на бумажных носителях. Также использование AИC позволит избегать появления ошибок в процессе работы системы и быстро их исправлять, если все же при заполнении данных произойдут некоторые ошибки.

База данных представляет собой хранилище билетов. У каждого билета есть владелец, он указан заданными фамилией, именем и отчеством. Так же у билета есть номер рейса, на который действует данный билет, а так же номер места в самолете.

По мимо базу билетом, требуется база рейсов. Для каждого рейса хранится пункт отправления и прибытия, время отправления и прибытия, цена перелета, количество свободных мест и номер самолета.

Также требуется база самолетов, где указано название самолета, название фирмы владельца и количество посадочных мест в самолете.

По номеру рейса в билете можно получить точные данные о рейсе. В свою очередь по номеру самолета в базе рейсов, можно определить какой самолет производит данный рейс и посмотреть информацию о самолете, для данного рейса.

## 5.Техническое задание на разработку ИС

Глобальные цели ИС:

* Ускорить и упростить поиск нужной информации.
* Пользователю АИС будет проще понимать общую картину работы аэропорта.
* Система может автоматически систематизировать и группировать, что позволят получать точные отчеты о работе предприятия.

Использование баз данных для такой АИС оправдано тем, что они позволяют структурированного хранить данные, в них удобно выполнять различные запросы для работы с этими данными, также СУБД позволяет строить отчеты, которые могут группировать и выводить информацию в любом необходимом пользователю виде.

Список проблем обработки данных, которые будут решены при помощи данной ИС:

* Быстрая и корректная обработка данных(добавление, удаление, группирование)
* Ускорение поиска конкретных данных
* Безопасность хранения данных

Перечень задач, решаемых системой:

* Хранение информации о рейсах, самолетах, билетах.
* Просмотр информации
* Добавление новых рейсов
* Продажа билетов

Перечень субъектов, участвующие в работе ИС:

* Кассир
* Администратор
* Потенциальные покупатели билетов

## 6. Концептуальная модель данных

Концептуальная модель - это отражение предметной области, для которой разрабатывается база данных.

Компонентами модели являются объекты и взаимосвязи. Концептуальная модель служит, средством общения между различными пользователями и поэтому разрабатывается без учета особенностей физического представления данных. Проектирование концептуальной модели основано на основе анализа решаемых на этом предприятии задач по обработке данных. Концептуальная модель включает описания объектов и их взаимосвязей, представляющих интерес в рассматриваемой предметной области. Взаимосвязи между объектами являются частью концептуальной модели и должны отображаться в базе данных. Взаимосвязь может охватывать любое число объектов.

Выделим сущности предметной области: билет, рейс, самолет.

Сущность рейса характеризуется следующими атрибутами:

пункт отправления — пункт,откуда отправляется самолет, пункт прибытия — пункт, куда прилетает самолет, время отправления — во сколько и в какой день вылетает самолет, время прибытия — во сколько и в какой день прилетает самолет, цена — цена перелета для одного человека, количество свободных мест — сколько билетов на рейс можно продать, номер самолета — номер самолета, который обслуживает данный рейс.

Сущность билет характеризуется следующими атрибутами:

номер рейса - номер рейса для которого действителен данный билет, фамилия,имя,отчество - данные о человеке,которые воспользуется данным билетом.

Сущность самолет характеризуется следующими атрибутами:

название — название самолета, фирма-владелец — фирма которая владеет данным самолетом, количество мест — сколько людей вмещает самолет.

На основании вышесказанного была построена концептуальная модель базы данных. Концептуальная модель предметной области представлена на рисунке.

Рисунок 1.

## 7. Логическая модель данных

Версия концептуальной модели, которая может быть обеспечена конкретной СУБД, называется логической моделью. Процесс построения логической модели базы данных должен опираться на определённую модель данных (реляционная, сетевая, иерархическая), которая определяется типом предполагаемой для реализации информационной системы СУБД. В нашем случае база данных создается создаётся в среде Microsoft Access и будет представлять собой реляционную базу данных.

Реляционная модель характеризуется своей простотой структуры данных, удобным для пользователя табличным представлением и возможностью использования формального аппарата алгебры отношений и реляционного исчисления для манипулирования данными..

В реляционных моделях данных объекты и взаимосвязи между ними представляются с помощью таблиц. Каждая таблица представляет один объект и состоит из строк и столбцов. Таблица в реляционной модели называется отношением.

Атрибут (поле) – любой столбец в таблице.

Домен – множество значений, которые может принимать атрибут.

Кортежи (записи) – строки таблицы.

Таблицы связаны между собой при помощи ключевых полей.

Ключ – это поле, позволяющее однозначно идентифицировать запись в таблице. Ключ может быть простым (состоит из одного поля) или составным (из нескольких полей).

В реляционных базах данных логическое проектирование приводит к разработке схемы данных, которая представлена на рисунке.

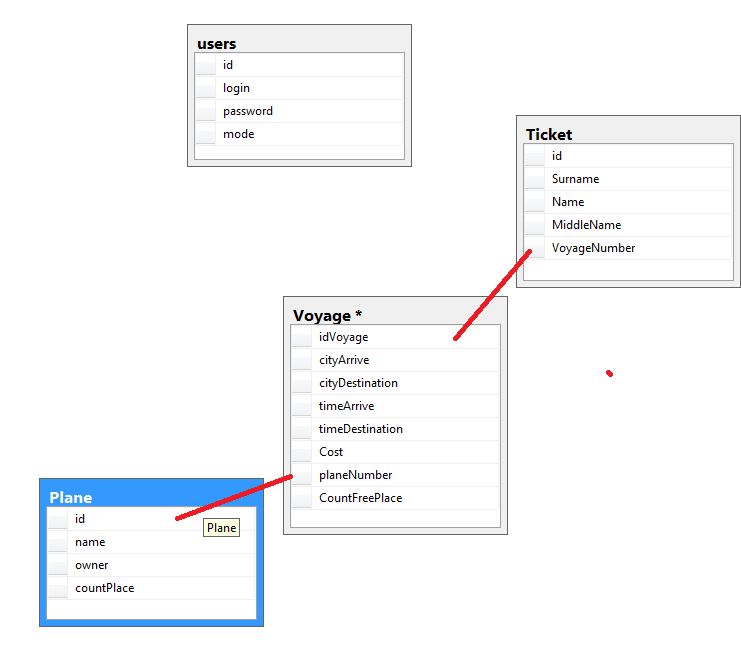


Рисунок 2.

Схема содержит как структурную, так и семантическую информацию. Структурная информация связана с объявлением отношений. Семантическая информация выражается множеством известных функциональных зависимостей между атрибутами отношений, объявленными в схеме. В связи с этим возникает вопрос о корректности представленной схемы. Корректной считается схема, в которой отсутствуют нежелательные функциональные зависимости между атрибутами.

Для приведения реляционной базы данных к корректному состоянию используется процесс нормализации отношений, который состоит в группировке элементов данных в ряде отношений.

## 8. Физическая модель данных

Физическое проектирование базы данных - процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах; на этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты.

Физическое проектирование последним этапом создания проекта базы данных, при выполнении которого проектировщик принимает решения о способах реализации разрабатываемой базы данных. Во время предыдущего этапа проектирования была определена логическая структура базы данных (которая описывает отношения и ограничения в рассматриваемой прикладной области). Хотя эта структура не зависит от конкретной целевой СУБД, она создается с учетом выбранной модели хранения данных, например реляционной, сетевой или иерархической. Однако, приступая к физическому проектированию базы данных, прежде всего необходимо выбрать конкретную целевую СУБД. Поэтому физическое проектирование неразрывно связано с конкретной СУБД. Между логическим и физическим проектированием существует постоянная обратная связь, так как решения, принимаемые на этапе физического проектирования с целью повышения производительности системы, способны повлиять на структуру логической модели данных.

Этапы физического проектирования баз данных:

* перенос глобальной логической модели данных в среду целевой СУБД;
* проектирование основных отношений;
* разработка способов получения производных данных;
* реализация ограничений предметной области;
* проектирование физического представления базы данных;
* анализ транзакций;
* выбор файловой структуры;
* определение индексов;
* определение требований к дисковой памяти;
* текущий контроль и настройка операционной системы.

В данной курсовой работе в качестве СУБД была выбрана Microsoft SQL Server 2014.

Были выбраны следующие типы данных этой СУБД:

nchar(X) – строка длины X. Используется для хранения текстовой информации, например, название самолета, фамилия, имя, отчество покупателя билета, город прилета и вылета.

int– длинное целое число. Используется для первичных ключей каждой таблицы, а также для хранения цены билета, номера места, количество мест и тд.

date - cохраняет дату и время в 32-битовом длинном слове. Этот тип данных необходим, чтобы хранить отправления и прибытия самолета.

Данные хранятся в таблицах. Представление таблиц в MS SQl.

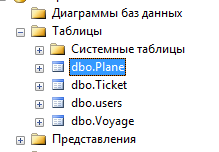


Рисунок 3.Все таблицы

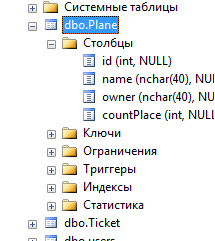


Рисунок 4.Таблица самолетов

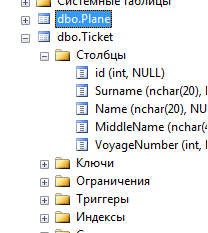


Рисунок 5. Таблица билетов

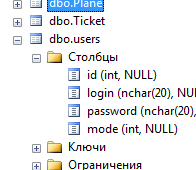


Рисунок 6. Таблица пользователей

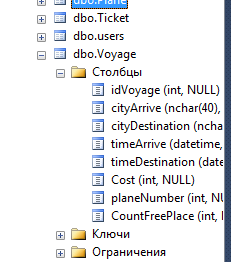


Рисунок 7.Таблица рейсов

## 9. SQL-код и запросы для работы с базой данных

SQL ─ универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных. SQL основывается на исчислении кортежей.

Изначально, SQL был основным способом работы пользователя с базой данных и выполнял примитивные операции.

Со временем, SQL усложнился, стал приобретать черты, свойственные языкам программирования.

При всех своих изменениях, SQL остаётся единственным механизмом связи между прикладным программным обеспечением и базой данных. В то же время, современные СУБД, а, также, информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов.

Для работы с базой данных необходимо чтобы таблицы были заполнены информацией. Запрос на заполнение таблицы voyage будет выглядеть таким образом:

Insert into Voyage (idVoyage, cityArrive, cityDestination, timeArrive, timeDestination, Cost,planeNumber,CountFreePlace) Values ('Moscow','Tokyo','2015-06-15 22:00', '2015-06-16 05:00', 5000, 5, 0);

Запрос на заполнение таблицы ticket будет выглядеть таким образом:

Insert into Ticket (id,Surname,Name,MiddleName,VoyageNumber) Values ( 1,’Давыдов’, ‘Денис’,’Вячеславович’,10).

Запрос на выборку всех пользователей: Select \* From users

Запрос на выборку всех рейсов: Select \* From Voyage

Запрос на выборку всех самолетов: Select \* From Plane

## 10. Формы приложения

Окно авторизации пользователя.

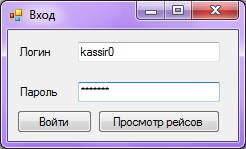


Рисунок 8.

Окно просмотра рейсов для пользователей

## 

Рисунок 9.

Окно продажи билетов

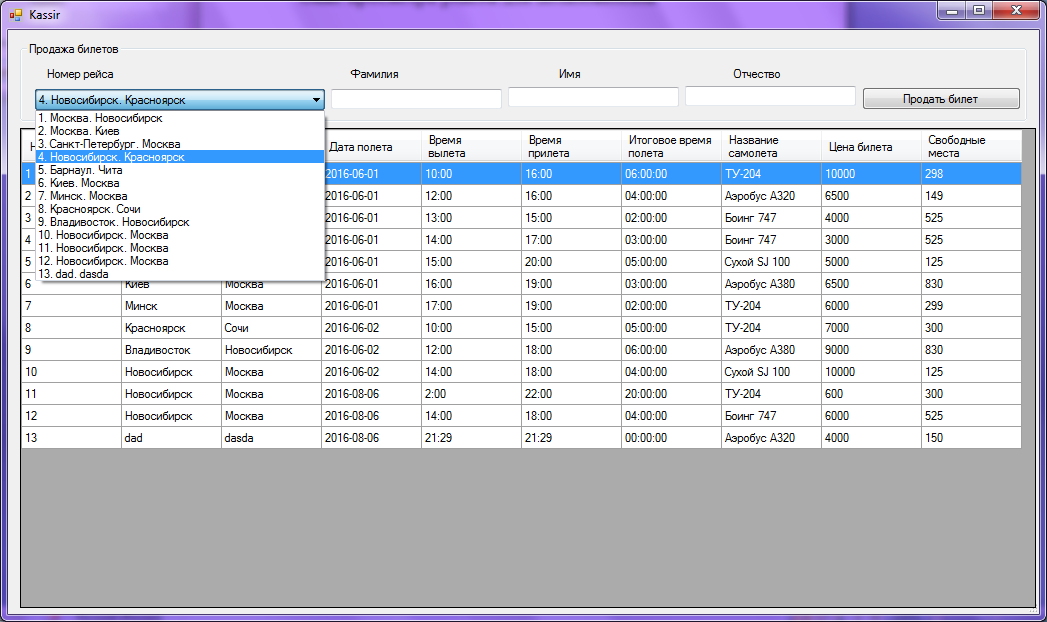


Рисунок 10.

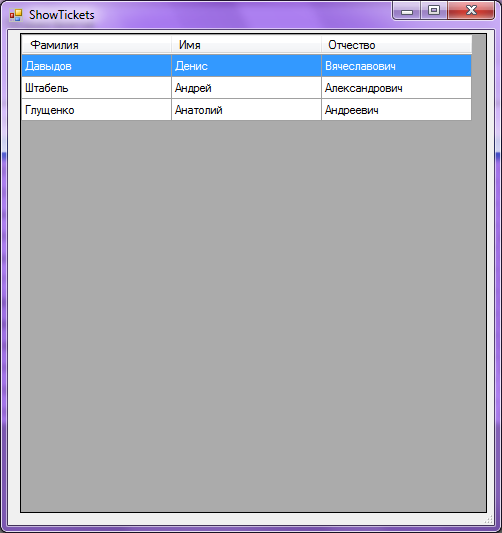


Рисунок 11. Окно просмотра проданных билетов для заданного рейса

Меню администратора

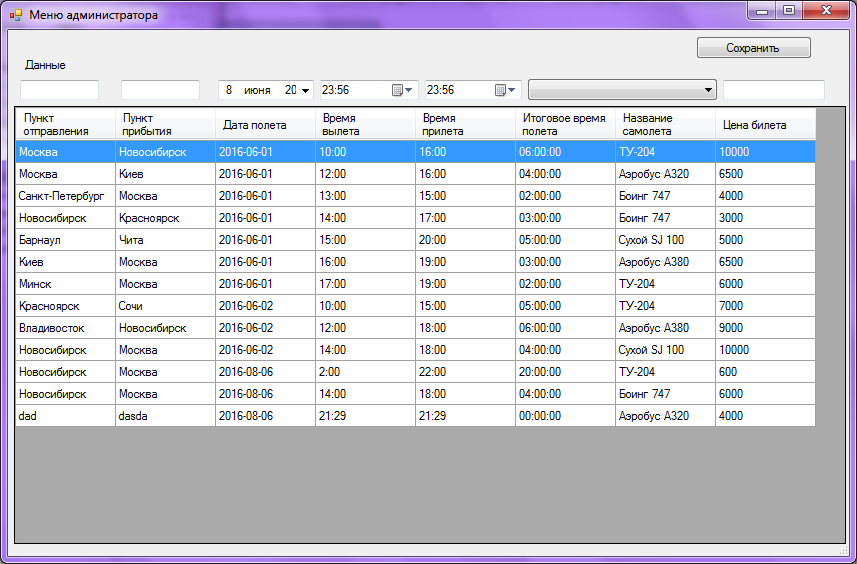


Рисунок 12.

## 11.Тестирование приложения

Для начала проверим содержимое таблиц.

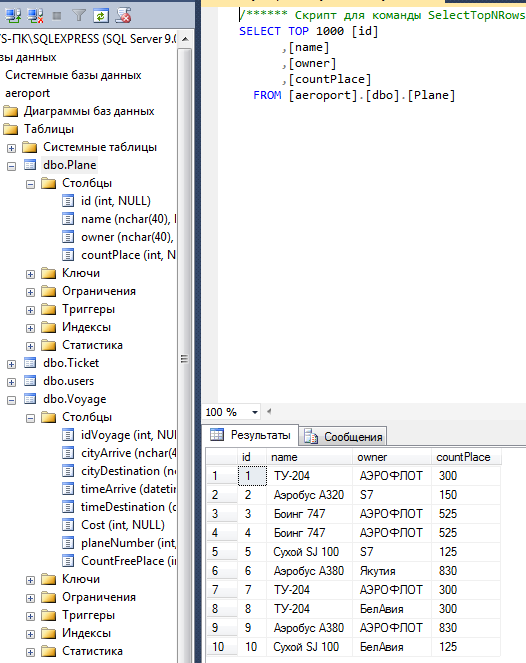


Рисунок 13. Таблица самолетов

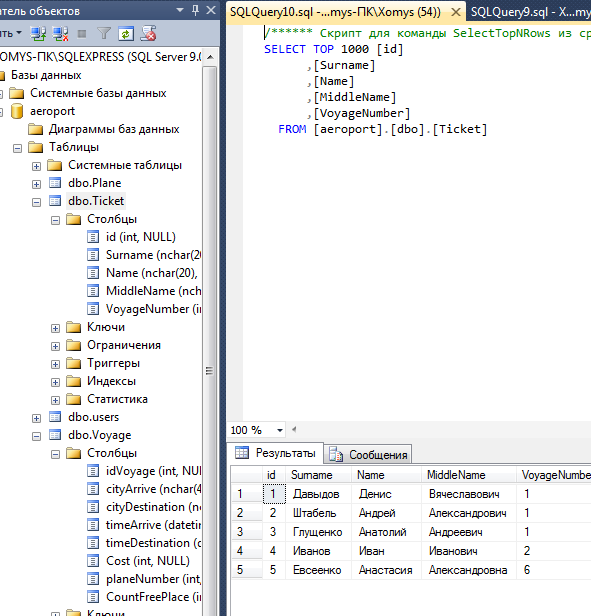


Рисунок 14. Таблица билетов

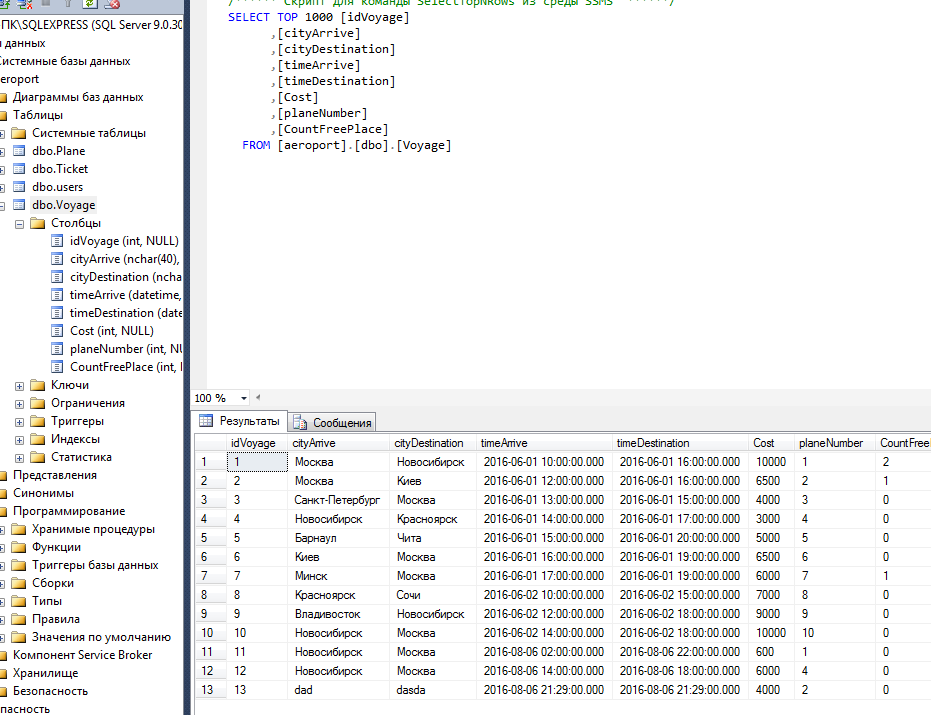


Рисунок 15. Таблица рейсов

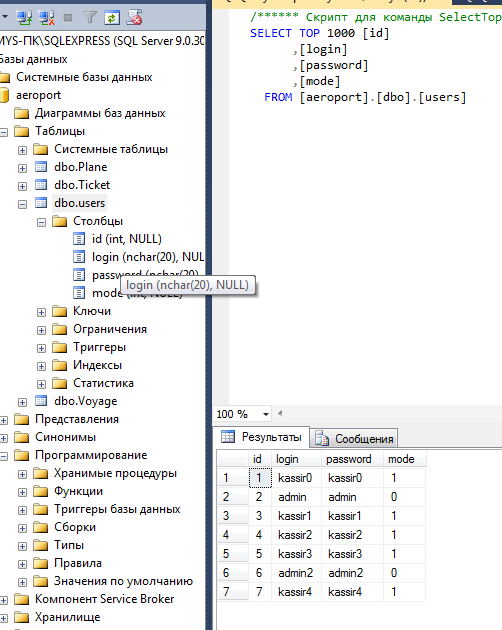
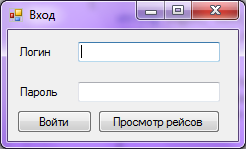


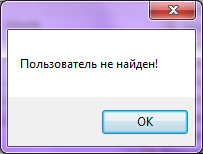
Таблица 16.

При запуске приложения видимо окно авторизации.

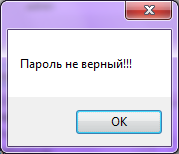


Тут необходимо ввести логин и пароль.

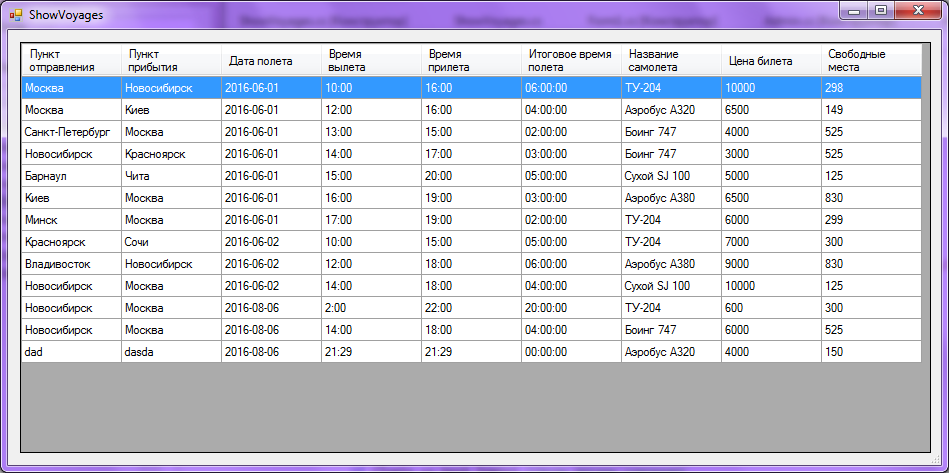
Если введен несуществующий логин, то программа сообщит об этом.



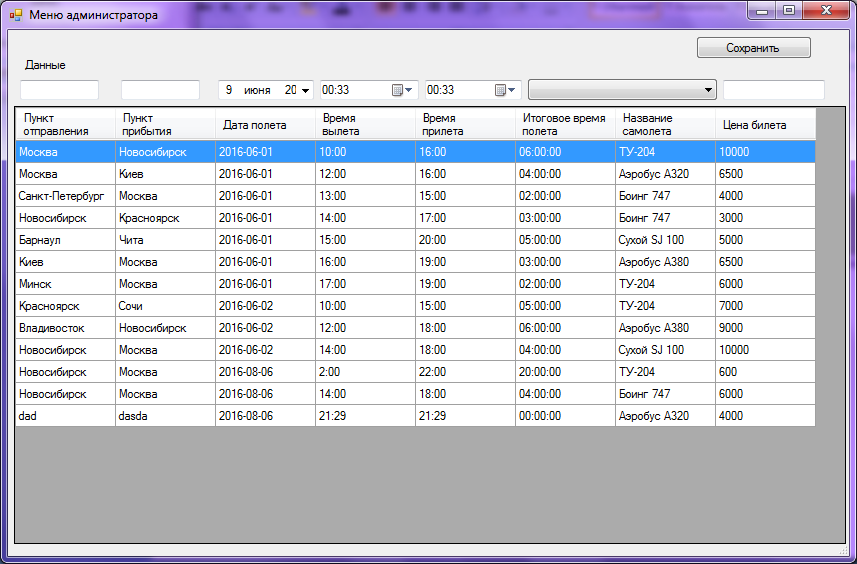
Если введен не правильный пароль, то программа тоже сообщит об этом.



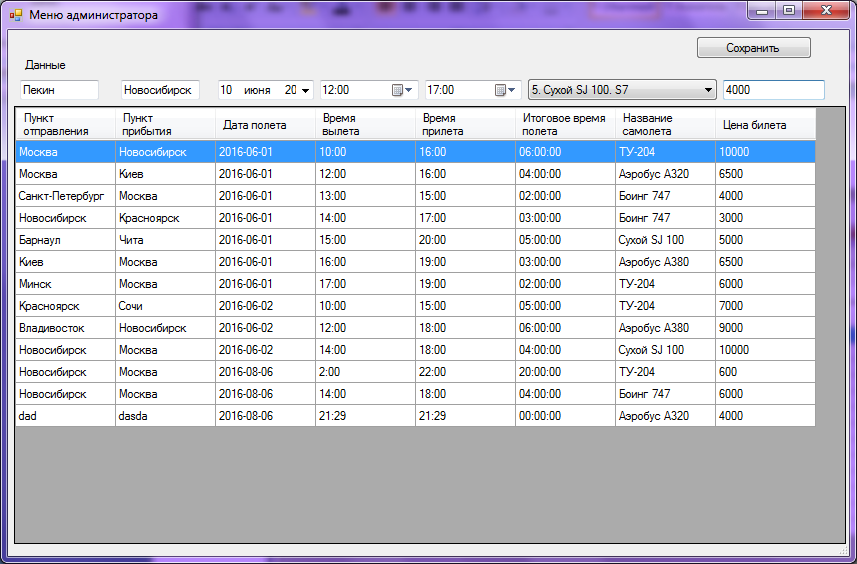
Если пользователь не является кассиром или администратором, а хочет посмотреть расписание рейсов, то он нажимает кнопку «Просмотр рейсов».



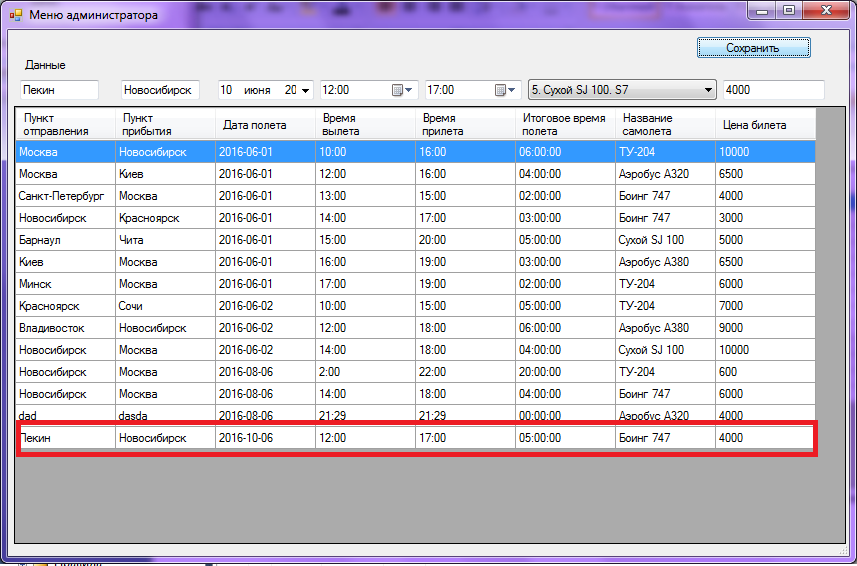
Если пользователь ввел свои логин и пароль и он является администратором, то откроется меню администратора.



Где можно просматривать существующие рейсы и добавлять новые.

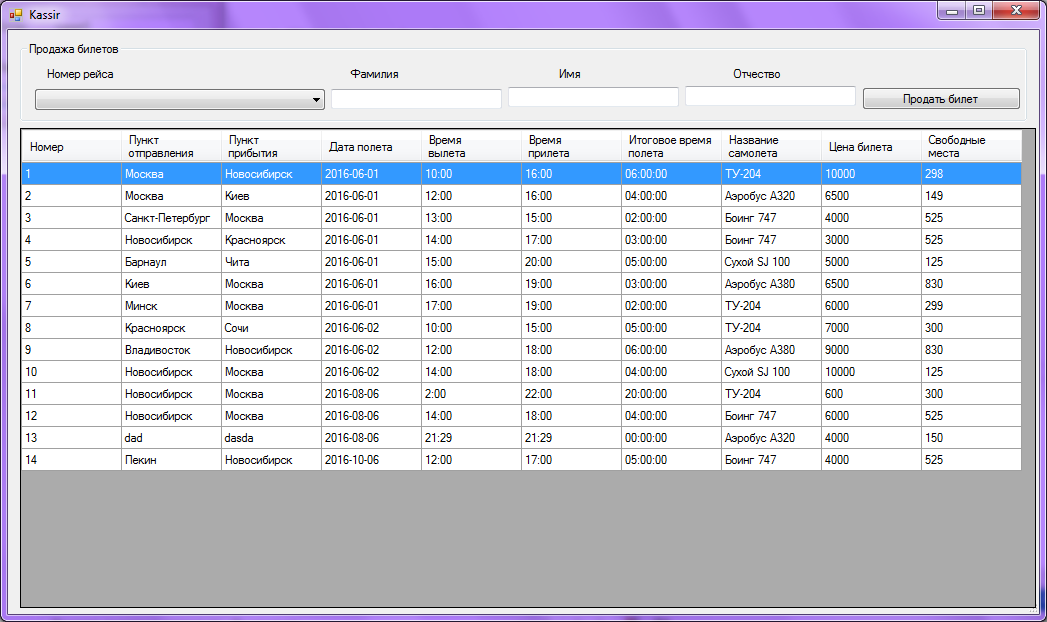


Введем данные о новом рейсе и сохраним его.

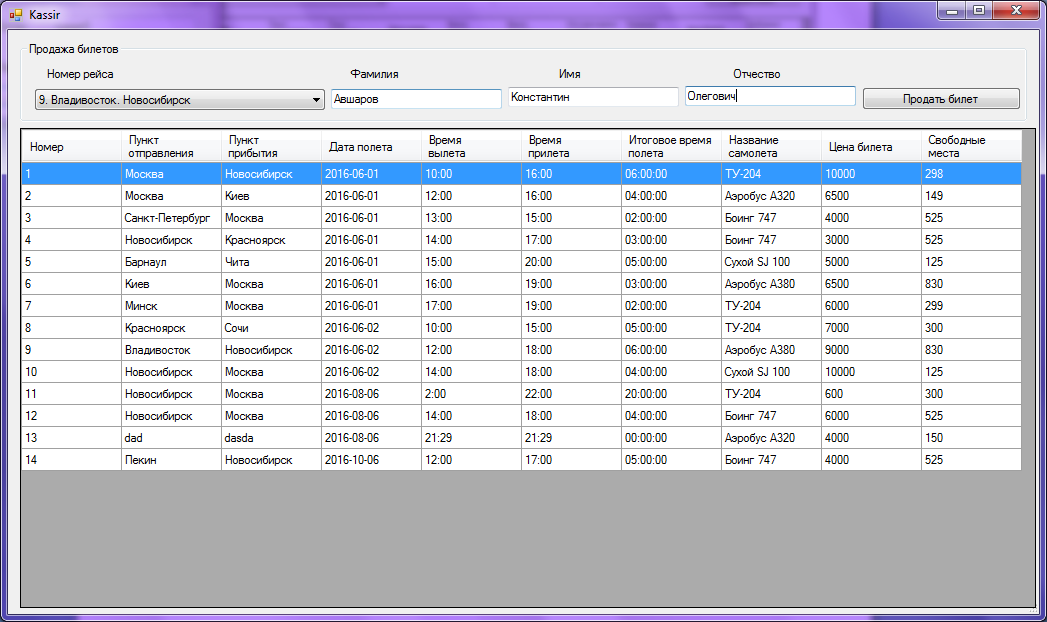


Видим, что данные добавились правильно.

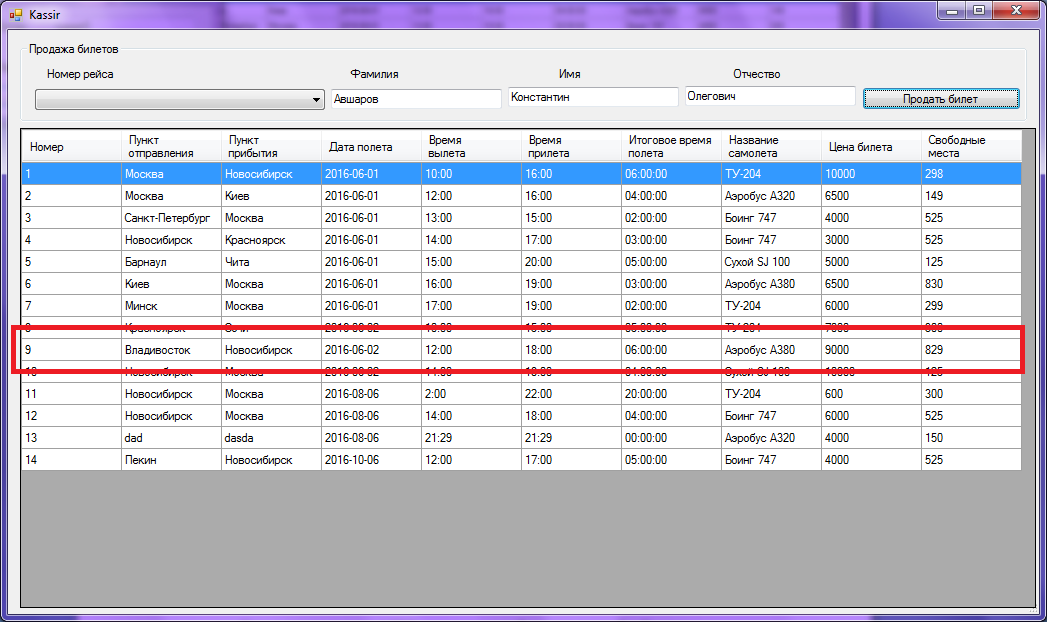
Зайдем в меню кассира.



Здесь можно просматривать информацию о рейсах и продавать билеты.



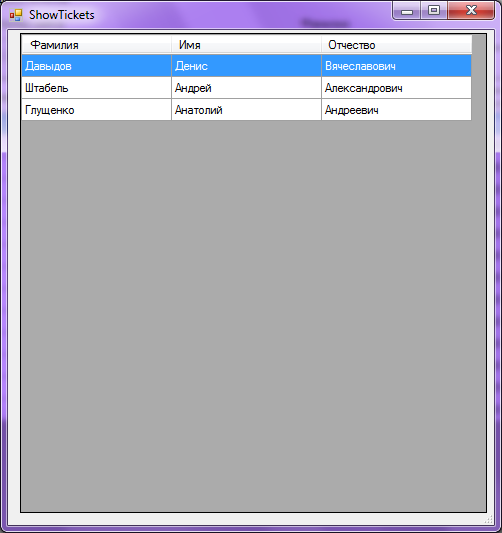
Продадим один билет на рейс «Владивосток – Новосибирск». Заметим, что количество свободных мест сейчас равно 830.



После продажи количество мест уменьшилось.

Теперь можем открыть список проданных билетов для любого рейса нажав два раза на него.

Выберем рейс номер 1.



Видим, что на данный рейс продано 3 билета.

## 12.Заключение

В ходе выполнения работы удалось выполнить следующие задачи:

* Произвели анализ предметной области
* Ознакомились и получили опыт работы в СУБД *Microsoft SQL Server*
* Ознакомились и получили опыт работы с языком SQL
* Спроектировали АИС для заданной предметной области
* Реализовали БД
* Изучили методы защиты информации

Полученную АИС нельзя назвать полноценной системой, мы брали упрощенную схему работы оздоровительного учреждения, но полученный опыт и навыки позволят в дальнейшем проектировать полноценные системы, которые будут покрывать все аспекты решаемых задач.

## Список использованных источников

1. Кузин, А.В. Базы данных: учеб. пособие для студ. высш. учеб. Заведений / А.В. Кузин, С.В. Левонисова. – М.: Издательский центр «Академия», 2012. – 320 с.

2. Диго, С. М. Базы данных. Проектирование и создание: Учебно-методический комплекс / С. М. Диго. – М.: Изд. Центр ЕАОИ, 2008. – 171 с.

3. Олькина, Е.В. Методические указания по оформлению пояснительных записок к дипломным, курсовым проектам (работам) и отчетов по практикам в соответствии с требованиями государственных стандартов / Е.В. Олькина. – Орел: Орел-ГТУ, 2007. – 54с.

## Приложение .

**Листинг программы.**

**Файл Form1.cs (форма авторизации)**

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.Data.SqlClient;

namespace Aeroport

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

}

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

{

string login, test\_login;

string pass, test\_password;

bool check = false;

string mode;

login = textBox1.Text;

pass = textBox2.Text;

SqlConnection conn = new SqlConnection(@"Data Source=(local)\SQLEXPRESS;Initial Catalog=aeroport;Integrated Security=True;");

try

{

//пробуем подключится

conn.Open();

SqlCommand cmd = new SqlCommand("Select \* From users", conn); //создаем команду

using (SqlDataReader dr = cmd.ExecuteReader(CommandBehavior.CloseConnection)) //выполняем

{

while (dr.Read()) //читаем результат

{

string id = dr.GetValue(0).ToString().Trim();

test\_login = dr.GetValue(1).ToString().Trim();

test\_password = dr.GetValue(2).ToString().Trim();

if (login == test\_login) //если логины совпадают

{

check = true;

if (test\_password == pass) //если пароли совпадают

{

mode = dr.GetValue(3).ToString();

if (mode == "1") //проверка наличия прав администратора

{

Kassir kassir = new Kassir();

kassir.ShowDialog();

}

else

{

Admin admin = new Admin();

admin.ShowDialog();

}

Hide();

}

else

{

MessageBox.Show("Пароль не верный!!!");

}

}

}

}

conn.Close();

conn.Dispose();

}

catch (SqlException se)

{

MessageBox.Show(se.Message);

}

if (!check)

MessageBox.Show("Пользователь не найден!");

//закрвываем соединение

Close();

}

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

{

ShowVoyages show = new ShowVoyages();

show.ShowDialog();

}

}

}

**Файл kassir.cs**

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.Data.SqlClient;

namespace Aeroport

{

public partial class Kassir : Form

{

int idVoyage;

SqlConnection conn = new SqlConnection(@"Data Source=(local)\SQLEXPRESS;Initial Catalog=aeroport;Integrated Security=True;");

public Kassir()

{

InitializeComponent();

}

public string stringToDate(DateTime str)

{

string res = "";

res += str.Year + "-" + str.Month.ToString("00") + "-" + str.Day.ToString("00");

return res;

}

public string stringToTime(DateTime str)

{

string res = "";

res += str.Hour + ":" + str.Minute.ToString("00");

return res;

}

public void UpdateTable()

{

DateTime arrive, destination;

TimeSpan dif;

string line;

int i = 0;

int countPlace, countBusyPlace;

dataGridView1.Rows.Clear();

comboBox1.Items.Clear();

SqlCommand cmd = new SqlCommand("Select \*,Plane.name From Voyage join Plane on Plane.id = Voyage.planeNumber", conn); //создаем команду

using (SqlDataReader dr = cmd.ExecuteReader()) //выполняем

{

while (dr.Read()) //читаем результат

{

dataGridView1.Rows.Add();

line = dr.GetValue(0).ToString().Trim();

line += ". " + dr.GetValue(1).ToString().Trim();

line += ". " + dr.GetValue(2).ToString().Trim();

comboBox1.Items.Add(line);

dataGridView1[0, i].Value = dr.GetValue(0).ToString().Trim();

dataGridView1[1, i].Value = dr.GetValue(1).ToString().Trim();

dataGridView1[2, i].Value = dr.GetValue(2).ToString().Trim();

arrive = System.Convert.ToDateTime(dr.GetValue(3).ToString());

dataGridView1[3, i].Value = stringToDate(arrive);

dataGridView1[4, i].Value = stringToTime(arrive);

destination = System.Convert.ToDateTime(dr.GetValue(4).ToString());

dataGridView1[5, i].Value = stringToTime(destination);

dif = destination - arrive;

dataGridView1[6, i].Value = dif;

dataGridView1[7, i].Value = dr.GetValue(9).ToString().Trim();

dataGridView1[8, i].Value = dr.GetValue(5).ToString().Trim();

countPlace = System.Convert.ToInt32(dr.GetValue(11).ToString().Trim());

countBusyPlace = System.Convert.ToInt32(dr.GetValue(7).ToString().Trim());

dataGridView1[9, i].Value = countPlace - countBusyPlace;

i++;

}

}

}

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

{

conn.Open();

UpdateTable();

}

public int getId()

{

int id;

SqlCommand cmd = new SqlCommand("Select max(id) From Ticket", conn); //создаем команду

using (SqlDataReader dr = cmd.ExecuteReader()) //выполняем

{

dr.Read();

id = System.Convert.ToInt32(dr.GetValue(0).ToString().Trim());

}

return id + 1;

}

public int CountBusyPlace()

{

int res;

SqlCommand cmd = new SqlCommand("Select CountFreePlace From Voyage where idVoyage = "+(comboBox1.SelectedIndex+1).ToString(), conn); //создаем команду

using (SqlDataReader dr = cmd.ExecuteReader()) //выполняем

{

dr.Read();

res = System.Convert.ToInt32(dr.GetValue(0).ToString().Trim());

}

return res+1;

}

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

{

int id = getId();

SqlCommand cmd = new SqlCommand("Insert into Ticket (id,Surname,Name,MiddleName,VoyageNumber) Values (" + id.ToString() + ",'" + textBox1.Text + "','" + textBox2.Text + "','" + textBox3.Text + "',"+comboBox1.SelectedIndex+")", conn); //создаем команду

cmd.ExecuteNonQuery();

cmd = new SqlCommand("Update Voyage set CountFreePlace = " + CountBusyPlace().ToString() + " where idVoyage = " + (comboBox1.SelectedIndex + 1).ToString(), conn); //создаем команду

cmd.ExecuteNonQuery();

UpdateTable();

}

private void dataGridView1\_MouseDoubleClick(object sender, MouseEventArgs e)

{

ShowTickets ticket = new ShowTickets();

ticket.idVoyage = idVoyage;

ticket.ShowDialog();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

idVoyage = System.Convert.ToInt32(dataGridView1[0, e.RowIndex].Value);

}

private void dataGridView1\_CellMouseClick(object sender, DataGridViewCellMouseEventArgs e)

{

idVoyage = System.Convert.ToInt32(dataGridView1[0, e.RowIndex].Value);

}

}

}

**Файл admin.cs**

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.Data.SqlClient;

using System.Collections;

namespace Aeroport

{

public partial class Admin : Form

{

List<int> id = new List<int>();

int selectIndex;

SqlConnection conn = new SqlConnection(@"Data Source=(local)\SQLEXPRESS;Initial Catalog=aeroport;Integrated Security=True;");

public Admin()

{

InitializeComponent();

}

public string stringToDate(DateTime str)

{

string res = "";

res += str.Year + "-" + str.Month.ToString("00") + "-" + str.Day.ToString("00");

return res;

}

public string stringToTime(DateTime str)

{

string res = "";

res += str.Hour + ":" + str.Minute.ToString("00");

return res;

}

public string DateTimeToString(DateTime date, DateTime time)

{

string res = "";

res = date.Year + "-" + date.Month + "-" + date.Day + " " + time.Hour + ":" + time.Minute;

return res;

}

public int getId()

{

int id;

SqlCommand cmd = new SqlCommand("Select max(idVoyage) From Voyage", conn); //создаем команду

using (SqlDataReader dr = cmd.ExecuteReader()) //выполняем

{

dr.Read();

id = System.Convert.ToInt32(dr.GetValue(0).ToString().Trim());

}

return id+1;

}

public void UpdateTable()

{

dateTimePicker2.Format = DateTimePickerFormat.Custom;

dateTimePicker3.Format = DateTimePickerFormat.Custom;

DateTime arrive, destination;

TimeSpan dif;

int i = 0;

dataGridView1.Rows.Clear();

SqlCommand cmd = new SqlCommand("Select \*,Plane.name From Voyage join Plane on Plane.id = Voyage.planeNumber", conn); //создаем команду

using (SqlDataReader dr = cmd.ExecuteReader()) //выполняем

{

while (dr.Read()) //читаем результат

{

dataGridView1.Rows.Add();

dataGridView1[0, i].Value = dr.GetValue(1).ToString().Trim();

dataGridView1[1, i].Value = dr.GetValue(2).ToString().Trim();

arrive = System.Convert.ToDateTime(dr.GetValue(3).ToString());

dataGridView1[2, i].Value = stringToDate(arrive);

dataGridView1[3, i].Value = stringToTime(arrive);

destination = System.Convert.ToDateTime(dr.GetValue(4).ToString());

dataGridView1[4, i].Value = stringToTime(destination);

dif = destination - arrive;

dataGridView1[5, i].Value = dif;

dataGridView1[6, i].Value = dr.GetValue(9).ToString().Trim();

dataGridView1[7, i].Value = dr.GetValue(5).ToString().Trim();

i++;

}

}

}

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

{

conn.Open();

UpdateTable();

SqlCommand cmd = new SqlCommand("Select \* From Plane", conn);

using (SqlDataReader dr = cmd.ExecuteReader()) //выполняем

{

while (dr.Read()) //читаем результат

{

string line;

line = dr.GetValue(0).ToString().Trim();

line += ". " + dr.GetValue(1).ToString().Trim();

line += ". " + dr.GetValue(2).ToString().Trim();

comboBox1.Items.Add(line);

}

}

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

selectIndex = e.RowIndex;

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

}

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

{

string arrive = DateTimeToString(dateTimePicker1.Value, dateTimePicker2.Value);

string destination = DateTimeToString(dateTimePicker1.Value, dateTimePicker3.Value);

int id = getId();

SqlCommand cmd = new SqlCommand("Insert into Voyage (idVoyage,cityArrive,cityDestination,timeArrive,timeDestination,Cost,planeNumber,CountFreePlace) Values ("+id.ToString()+",'" + textBox1.Text + "','" + textBox2.Text + "','" + arrive + "','" + destination + "'," + textBox3.Text + "," + comboBox1.SelectedIndex + ",0)", conn); //создаем команду

cmd.ExecuteNonQuery();

UpdateTable();

}

}

}

**Файл ShowTicket.cs**

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.Data.SqlClient;

namespace Aeroport

{

public partial class ShowTickets : Form

{

public int idVoyage;

SqlConnection conn = new SqlConnection(@"Data Source=(local)\SQLEXPRESS;Initial Catalog=aeroport;Integrated Security=True;");

public ShowTickets()

{

InitializeComponent();

}

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

{

conn.Open();

int i = 0;

dataGridView1.Rows.Clear();

SqlCommand cmd = new SqlCommand("Select \* From Ticket where VoyageNumber = "+idVoyage, conn); //создаем команду

using (SqlDataReader dr = cmd.ExecuteReader()) //выполняем

{

while (dr.Read()) //читаем результат

{

dataGridView1.Rows.Add();

dataGridView1[0, i].Value = dr.GetValue(1).ToString().Trim();

dataGridView1[1, i].Value = dr.GetValue(2).ToString().Trim();

dataGridView1[2, i].Value = dr.GetValue(3).ToString().Trim();

i++;

}

}

}

}

}

**Файл ShowVoyages.cs**

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.Data.SqlClient;

namespace Aeroport

{

public partial class ShowVoyages : Form

{

public ShowVoyages()

{

InitializeComponent();

}

public string stringToDate(DateTime str)

{

string res = "";

res += str.Year + "-" + str.Month.ToString("00") + "-" + str.Day.ToString("00");

return res;

}

public string stringToTime(DateTime str)

{

string res = "";

res += str.Hour + ":" + str.Minute.ToString("00");

return res;

}

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

{

SqlConnection conn = new SqlConnection(@"Data Source=(local)\SQLEXPRESS;Initial Catalog=aeroport;Integrated Security=True;");

conn.Open();

DateTime arrive, destination;

TimeSpan dif;

int i = 0;

int countPlace, countBusyPlace;

dataGridView1.Rows.Clear();

SqlCommand cmd = new SqlCommand("Select \*,Plane.name From Voyage join Plane on Plane.id = Voyage.planeNumber", conn); //создаем команду

using (SqlDataReader dr = cmd.ExecuteReader()) //выполняем

{

while (dr.Read()) //читаем результат

{

dataGridView1.Rows.Add();

dataGridView1[0, i].Value = dr.GetValue(1).ToString().Trim();

dataGridView1[1, i].Value = dr.GetValue(2).ToString().Trim();

arrive = System.Convert.ToDateTime(dr.GetValue(3).ToString());

dataGridView1[2, i].Value = stringToDate(arrive);

dataGridView1[3, i].Value = stringToTime(arrive);

destination = System.Convert.ToDateTime(dr.GetValue(4).ToString());

dataGridView1[4, i].Value = stringToTime(destination);

dif = destination - arrive;

dataGridView1[5, i].Value = dif;

dataGridView1[6, i].Value = dr.GetValue(9).ToString().Trim();

dataGridView1[7, i].Value = dr.GetValue(5).ToString().Trim();

countPlace = System.Convert.ToInt32(dr.GetValue(11).ToString().Trim());

countBusyPlace = System.Convert.ToInt32(dr.GetValue(7).ToString().Trim());

dataGridView1[8, i].Value = countPlace - countBusyPlace;

i++;

}

}

}

}

}