Лабораторная работа №10

Экономическая информационная система.

*Постановка задачи*: Разработайте приложение с использованием элементов Label, Button, TextBox, ComboBox, Panel, GroupBox, LinkLabel, ErrorProvider, RadioButton, CheckBox. Приложение должно позволять добавлять, удалять и редактировать информацию. Информация должна храниться в одном или нескольких файлах. Переключение между объектами проводить кнопками «Вперед» и «Назад» с указанием текущего номера объекта (или с помощью TrackBar). Управление основными объектами реализовать с помощью классов или структур. Главное окно - проданный билет. В билете указывается ФИО, место, рейс, номер автобуса, точка начала, точка завершения. У каждого рейса есть список автобусов, которые курсируют, расписание (массив элементов: дата-время выезда и номер автобуса, ФИО водителя), набор остановочных пунктов и время движения между пунктами (время приезда и время выезда из каждого пункта).

*Входные данные*:

* «Info.xml»

*Выходные данные*:

* Данные записываются в файл «Info.XML»

*Рабочие переменные*:

* index, max, list, doc, xnode, xRoot, fio, startpoint, finishpoint, trip1, pointlist, childnode, tRip, ticket, xDoc, tickets, ticketElem, idElem, fioElem, tripElem, stPointElem, fPointElem, idText, fioText,tripText, stPointText, fPointText, date, time , bus , points, driver, tripLocations, bus1, driver1, i , array, currentticket

*Описание стандартных функций и библиотек*

* Console.WriteLine() – вывод данных на экран;
* Console.ReadLine() – получение введенных данных.

*Классы и процедуры, созданные автором*

* class Ticket;
* class Autobus;
* class Driver;
* class Trip;
* class MainForm;
* public TicketList();
* public void AddTicket(Ticket ticket);
* public void DeleteTicket(Ticket ticket);
* public Ticket findById(int id);
* public void ReplaceInfoTicket(Ticket ticket);
* public Ticket GetPrevious(int id);
* public Ticket GetNext(int id);
* public int GetMaxID();
* public List<Ticket> ReadTickets();
* public void SaveTickets(List<Ticket> allTickets);
* public TripList();
* public void AddTrip(Trip trip);
* public void DeleteTrip(Trip trip);
* public Trip findBytripLocations(string tripLocations);
* public List<Trip> ReadTrips();
* public DriverList();
* public void AddDriver(Driver driver);
* public void DeleteDriver(Driver driver);
* public Driver findByFio(string fio);
* public List<Driver> ReadDriver();
* public BusList();
* public void AddBus(Autobus bus);
* public void DeleteBus(Autobus bus);
* public Autobus findByNumber(string busNumber);
* public List<Autobus> ReadAutobus();
* private void BuildtripBox();
* private void MainForm\_Load(object sender, EventArgs e);
* public void FillForm(Ticket ticket);
* private void tripBox\_SelectedIndexChanged(object sender, EventArgs e);
* private void prevButton\_Click(object sender, EventArgs e);
* private void nextButton\_Click(object sender, EventArgs e);
* private void newTicket\_Click(object sender, EventArgs e);
* private void saveButton\_Click(object sender, EventArgs e);
* private void deleteButton\_Click(object sender, EventArgs e);

*Текст программы:*

using System;

using System.Collections.Generic;

using System.Xml;

namespace Lab\_10

{

class TicketList//класс списка билетов

{

public List<Ticket> allTicket;

public TicketList()

{

allTicket = new List<Ticket>();

}

public void AddTicket(Ticket ticket)//функция для добавления билета

{

allTicket.Add(ticket);

}

public void DeleteTicket(Ticket ticket)//функция для удаления билета

{

allTicket.Remove(ticket);

}

public Ticket findById(int id)

{

return allTicket.Find(x => x.ID == id);

}

public void ReplaceInfoTicket(Ticket ticket)//метод для обновленения информации

{

int index = allTicket.FindIndex(x => x.ID == ticket.ID);

allTicket[index] = ticket;

}

public Ticket GetPrevious(int id)//получение предыдущего билета

{

int index = allTicket.FindIndex(x => x.ID == id);

if (index > 0)

{

return allTicket[index - 1];

}

return null;

}

public Ticket GetNext(int id)//получение следующего билета

{

int index = allTicket.FindIndex(x => x.ID == id);

if (index < allTicket.Count - 1)

{

return allTicket[index + 1];

}

return null;

}

public int GetMaxID()//получение максимального значения id(конечный билет)

{

int max = 0;

foreach (Ticket ticket in allTicket)

{

if (max < ticket.ID)

{ max = ticket.ID; }

}

return max;

}

public List<Ticket> ReadTickets()

{

List<Ticket> list = new List<Ticket>();

XmlDocument doc = new XmlDocument();

doc.Load("Info.xml");

XmlElement xRoot = doc.DocumentElement;

// проход всех узлов в корневом элементе

foreach (XmlNode xnode in xRoot)

{

if (xnode.Name == "Tickets")

{

foreach (XmlNode node in xnode)

{

int id = 0; string fio = ""; string trip1 = ""; string startPoint = ""; string finishPoint = "";

// проходим все дочерние узлы элемента

List<string> pointsList = new List<string>();

foreach (XmlNode childnode in node.ChildNodes)

{

if (childnode.Name == "id") { id = Int32.Parse(childnode.InnerText); }

if (childnode.Name == "Fio") { fio = childnode.InnerText; }

if (childnode.Name == "Trip") { trip1 = childnode.InnerText; }

if (childnode.Name == "StartPoint") { startPoint = (childnode.InnerText); }

if (childnode.Name == "FinishPoint") { finishPoint = childnode.InnerText; }

}

Trip triP = MainForm.triplist.findBytripLocations(trip1);

Ticket ticket = new Ticket(id, fio, triP, startPoint, finishPoint);

list.Add(ticket);

}

}

}

return list;

}

public void SaveTickets(List<Ticket> allTickets)

{

XmlDocument xDoc = new XmlDocument();

xDoc.Load("Info.xml");

XmlElement xRoot = xDoc.DocumentElement;

XmlNodeList childnodes = xRoot.SelectNodes("\*");

foreach (XmlNode n in childnodes)

{

if (n.Name == "Tickets")

{

xRoot.RemoveChild(n);

}

}

XmlElement tickets = xDoc.CreateElement("Tickets");

foreach (Ticket ticket in allTickets)

{

// создаем новый элемент ticket

XmlElement ticketElem = xDoc.CreateElement("Ticket");

// создаем элементы

XmlElement idElem = xDoc.CreateElement("id");

XmlElement fioElem = xDoc.CreateElement("Fio");

XmlElement tripElem = xDoc.CreateElement("Trip");

XmlElement stPointElem = xDoc.CreateElement("StartPoint");

XmlElement fPointElem = xDoc.CreateElement("FinishPoint");

// создаем текстовые значения для элементов и атрибута

XmlText idText = xDoc.CreateTextNode(ticket.ID.ToString());

XmlText fioText = xDoc.CreateTextNode(ticket.Fio);

XmlText tripText = xDoc.CreateTextNode(ticket.Trip.TripLocations);

XmlText stPointText = xDoc.CreateTextNode(ticket.Startpoint);

XmlText fPointText = xDoc.CreateTextNode(ticket.FinishPoint);

//добавляем узлы

idElem.AppendChild(idText);

fioElem.AppendChild(fioText);

tripElem.AppendChild(tripText);

stPointElem.AppendChild(stPointText);

fPointElem.AppendChild(fPointText);

ticketElem.AppendChild(idElem);

ticketElem.AppendChild(fioElem);

ticketElem.AppendChild(tripElem);

ticketElem.AppendChild(stPointElem);

ticketElem.AppendChild(fPointElem);

tickets.AppendChild(ticketElem);

}

xRoot.AppendChild(tickets);

xDoc.Save("Info.xml");

}

}

class Ticket//класс для билетов

{

public int ID { get { return id; } set { id = value; } }

public string Fio { get { return fio; } set { fio = value; } }

public Trip Trip { get { return trip; } set { trip = value; } }

public string Startpoint { get { return startPoint; } set { startPoint = value; } }

public string FinishPoint { get { return finishPoint; } set { finishPoint = value; } }

private int id;//id билета

private string fio;//фамилия

private Trip trip;

private string startPoint;//начальная точка поездки

private string finishPoint;//последняя точка поездки

public Ticket(int id, string fio, Trip trip, string startPoint, string finishPoint)//конструктор для заполнения информации

{

this.id = id;

this.fio = fio;

this.trip = trip;

this.startPoint = startPoint;

this.finishPoint = finishPoint;

}

public Ticket(int id)//конструктор с id

{

this.id = id;

}

}

}

using System.Collections.Generic;

using System.Xml;

namespace Lab\_10

{

class TripList//класс списка рейсов

{

public List<Trip> allTrip;

public TripList()

{

allTrip = new List<Trip>();

}

public void AddTrip(Trip trip)//функция для добавления рейса

{

allTrip.Add(trip);

}

public void DeleteTrip(Trip trip)//функция для удаления рейса

{

allTrip.Remove(trip);

}

public Trip findBytripLocations(string tripLocations)

{

return allTrip.Find(x => x.TripLocations == tripLocations);

}

public List<Trip> ReadTrips()

{

List<Trip> list = new List<Trip>();

XmlDocument doc = new XmlDocument();

doc.Load("Info.xml");

XmlElement xRoot = doc.DocumentElement;

// обход всех узлов в корневом элементе

foreach (XmlNode xnode in xRoot)

{

if (xnode.Name == "Trip")

{

int id = 0; string date = ""; string time = ""; string bus = ""; ; string[,] points ; string driver = ""; string tripLocation = "";

// получаем атрибут id

if (xnode.Attributes.Count > 0)

{

XmlNode attr = xnode.Attributes.GetNamedItem("id");

if (attr != null)

{

id = int.Parse(attr.Value);

}

}

// обходим все дочерние узлы элемента

List<string> pointsList = new List<string>();

foreach (XmlNode childnode in xnode.ChildNodes)

{

if (childnode.Name == "Date") { date = childnode.InnerText; }

if (childnode.Name == "Time") { time = childnode.InnerText; }

if (childnode.Name == "Bus") { bus = (childnode.InnerText); }

if (childnode.Name == "Point")

{

pointsList.Add(childnode.InnerText);

}

if (childnode.Name == "Driver")

{

driver = childnode.InnerText;

}

if (childnode.Name == "TripLocations") { tripLocation = childnode.InnerText; }

}

Autobus bus1 = MainForm.buslist.findByNumber(bus);

Driver driver1 = MainForm.driverlist.findByFio(driver);

points = new string[pointsList.Count, 3];

for(int i = 0;i<pointsList.Count;i++)

{

string[] array = pointsList[i].Split(',');

points[i, 0] = array[0];

points[i, 1] = array[1];

points[i, 2] = array[2];

}

Trip trip = new Trip(id, date, time, bus1, points, driver1, tripLocation);

list.Add

(trip);

}

}

return list;

}

}

class Trip//класс рейсов

{

public int ID { get { return id; } set { id = value; } }

public string Date { get { return date; } set { date = value; } }

public string Time { get { return time; } set { time = value; } }

public Autobus Buses { get { return bus;} set { bus = value; } }

public string[,] Points { get { return points; } set { points = value; } }

public Driver Driver { get { return driver; } set { driver = value; } }

public string TripLocations { get { return tripLocations; } set { tripLocations = value; } }

private int id;//id поездки

private string date;//дата поездки

private string time;//время поездки

private Autobus bus;

private string[,] points;//двумерный массив с остановками и временем прибытия/отбытия

private Driver driver;//водитель автобуса

private string tripLocations;//проезжаемые места

public Trip(int id, string date, string time, Autobus bus, string [,] points, Driver driver, string tripLocations)//конструктор для заполнения информации

{

this.id = id;

this.date = date;

this.time = time;

this.bus = bus;

this.points = points;

this.driver = driver;

this.tripLocations = tripLocations;

}

public Trip (int id)//конструктор с id

{

this.id = id;

}

}

}

using System.Collections.Generic;

using System.Xml;

namespace Lab\_10

{

class DriverList//класс списка водителей

{

public List<Driver> allDrivers;

public DriverList()

{

allDrivers = new List<Driver>();

}

public void AddDriver(Driver driver)//функция реализущая добавление водителя

{

allDrivers.Add(driver);

}

public void DeleteDriver(Driver driver)//функция реализущая добавление водителя

{

allDrivers.Remove(driver);

}

public Driver findByFio(string fio)

{

return allDrivers.Find(x => x.FIO == fio);

}

public List<Driver> ReadDriver()

{

List<Driver> list = new List<Driver>();

XmlDocument doc = new XmlDocument();

doc.Load("Info.xml");

XmlElement xRoot = doc.DocumentElement;

// обход всех узлов в корневом элементе

foreach (XmlNode xnode in xRoot)

{

if (xnode.Name == "Driver")

{

int id = 0; string fio = "";

// получаем атрибут id

if (xnode.Attributes.Count > 0)

{

XmlNode attr = xnode.Attributes.GetNamedItem("id");

if (attr != null)

id = int.Parse(attr.Value);

}

// обходим все дочерние узлы элемента

foreach (XmlNode childnode in xnode.ChildNodes)

{

if (childnode.Name == "Fio") { fio = childnode.InnerText; }

}

Driver driver = new Driver(id, fio);

list.Add(driver);

}

}

return list;

}

}

class Driver//класс водителя

{

public int ID { get { return id; } set { id = value; } }

public string FIO { get { return fio; } set { fio = value; } }

private int id;//id водителя

private string fio;//ФИО водителя

public Driver(int id, string fio)//конструктор для добавления информации

{

this.id = id;

this.fio = fio;

}

public Driver(int id)//конструктор с id

{

this.id = id;

}

}

}

using System.Collections.Generic;

using System.Xml;

namespace Lab\_10

{

class BusList//класс со списком автобусов

{

public List<Autobus> allBus;

public BusList()

{

allBus = new List<Autobus>();

}

public void AddBus(Autobus bus)//функция для добавления автобуса

{

allBus.Add(bus);

}

public void DeleteBus(Autobus bus)//функция для удаления автобуса

{

allBus.Remove(bus);

}

public Autobus findByNumber(string busNumber)

{

return allBus.Find(x => x.BusNumber == busNumber);

}

public List<Autobus> ReadAutobus()

{

List<Autobus> list = new List<Autobus>();

XmlDocument doc = new XmlDocument();

doc.Load("Info.xml");

XmlElement xRoot = doc.DocumentElement;

// обход всех узлов в корневом элементе

foreach (XmlNode xnode in xRoot)

{

if (xnode.Name == "Autobus")

{

int id = 0; string busnumber = "";

// получаем атрибут id

if (xnode.Attributes.Count > 0)

{

XmlNode attr = xnode.Attributes.GetNamedItem("id");

if (attr != null)

id = int.Parse(attr.Value);

}

// обходим все дочерние узлы элемента

foreach (XmlNode childnode in xnode.ChildNodes)

{

if (childnode.Name == "BusNumber") { busnumber = childnode.InnerText; }

}

Autobus autobus = new Autobus(id, busnumber);

list.Add(autobus);

}

}

return list;

}

}

class Autobus

{

public int ID { get { return id ; } set { id = value; } }

public string BusNumber { get { return busNumber; } set { busNumber = value; } }

private int id;//id автобуса

private string busNumber;//номер автобуса

public Autobus(int id, string busNumber)//конструктор для добавления информации

{

this.id = id;

this.busNumber= busNumber;

}

public Autobus(int id)//конструктор с id

{

this.id = id;

}

public Autobus()//конструктор без параметров.вызываем его когда создается экземпляр с помощью оператора new без передачи аргументов

{

}

}

}

using System;

using System.Windows.Forms;

namespace Lab\_10

{

internal partial class MainForm : Form

{

internal static TripList triplist = new TripList();

internal static TicketList ticketlist = new TicketList();

internal static DriverList driverlist = new DriverList();

internal static BusList buslist = new BusList();

int currentId;//текущий id

public MainForm()

{

InitializeComponent();

buslist.allBus = buslist.ReadAutobus();

driverlist.allDrivers = driverlist.ReadDriver();

triplist.allTrip = triplist.ReadTrips();

ticketlist.allTicket = ticketlist.ReadTickets();

currentId = ticketlist.allTicket[0].ID;

}

private void BuildtripBox()

{

foreach(Trip trip in triplist.allTrip)

{

tripBox.Items.Add(trip.TripLocations);

}

}

private void MainForm\_Load(object sender, EventArgs e)//при загрузке формы заполняем базовую информацию

{

BuildtripBox();

FillForm(ticketlist.findById(currentId));

}

public void FillForm(Ticket ticket)//заполняем text/combo box'ы

{

this.Text = $"Билет №{currentId} ";

currentId = ticket.ID;

FIObox.Text = ticket.Fio;

for(int i = 0;i<ticket.Trip.Points.Length/5;i++)

{

placeBox.Items.Add(ticket.Trip.Points[i, 0]);

FinishBox.Items.Add(ticket.Trip.Points[i, 0]);

}

placeBox.Text = ticket.Startpoint;

FinishBox.Text = ticket.FinishPoint;

tripBox.Text = ticket.Trip.TripLocations;

AutoBusbox.Text = ticket.Trip.Buses.BusNumber;

Datebox.Text = ticket.Trip.Date;

TimeBox.Text = ticket.Trip.Time;

Driverbox.Text = ticket.Trip.Driver.FIO;

}

private void tripBox\_SelectedIndexChanged(object sender, EventArgs e)

{

placeBox.Items.Clear();//при каждой смене рейса очищаем поля

FinishBox.Items.Clear();

Trip trip = triplist.findBytripLocations(tripBox.Text);

TimeTable.Rows.Clear();

TimeTable.Rows.Add(5);//формируем расписание

for (int i = 0; i < trip.Points.Length / 3; i++)

{

int j = 0;

placeBox.Items.Add(trip.Points[i, 0]);

FinishBox.Items.Add(trip.Points[i, 0]);

TimeTable.Rows[i].Cells[j].Value = trip.Points[i, 0];

j++;

TimeTable.Rows[i].Cells[j].Value = trip.Points[i, 1];

j++;

TimeTable.Rows[i].Cells[j].Value = trip.Points[i, 2];

}

try

{

placeBox.Text = ticketlist.findById(currentId).Startpoint;

FinishBox.Text = ticketlist.findById(currentId).FinishPoint;

}

catch

{

placeBox.SelectedIndex = 1;

FinishBox.SelectedIndex = 1;

}

AutoBusbox.Text = trip.Buses.BusNumber;

Datebox.Text = trip.Date;

TimeBox.Text = trip.Time;

Driverbox.Text = trip.Driver.FIO;

}

private void prevButton\_Click(object sender, EventArgs e)//переходим на предыдущий билет

{

if(ticketlist.GetPrevious(currentId)!=null)

{

Ticket currentticket = ticketlist.GetPrevious(currentId);

currentId = currentticket.ID;

FillForm(currentticket);

}

}

private void nextButton\_Click(object sender, EventArgs e)//переходим на следующий билет

{

if(ticketlist.GetNext(currentId)!=null)

{

Ticket currentticket = ticketlist.GetNext(currentId);

currentId = currentticket.ID;

FillForm(currentticket);

}

}

private void newTicket\_Click(object sender, EventArgs e)//при нажатии на кнопку создается новый билет

{

if (FIObox.Text != "")

{

Ticket currentticket = new Ticket(ticketlist.GetMaxID() + 1);

ticketlist.AddTicket(currentticket);

currentId = currentticket.ID;

this.Text = "Билет №" + currentId;

tripBox.SelectedIndex = 0;

FIObox.Text = "";

placeBox.SelectedIndex = 0;

FinishBox.SelectedIndex = 0;

AutoBusbox.Text = "";

TimeBox.Text = "";

Datebox.Text = "";

Driverbox.Text = "";

tripBox\_SelectedIndexChanged(sender, e);

}

else

{

MessageBox.Show("Для добавления нового билета заполните данные");

}

}

private void saveButton\_Click(object sender, EventArgs e)//при нажатии кнопки сохраняем правки в документ

{

if(FIObox.Text!="" || placeBox.Text!=""||FinishBox.Text!="")

{

Ticket ticket = new Ticket(currentId);

ticket.Fio = FIObox.Text;

ticket.Startpoint = placeBox.Text;

ticket.FinishPoint = FinishBox.Text;

ticket.Trip = triplist.findBytripLocations(tripBox.Text);

ticketlist.ReplaceInfoTicket(ticket);

ticketlist.SaveTickets(ticketlist.allTicket);

MessageBox.Show("Сохранение прошло успешно");

}

else

{

MessageBox.Show("Вы не заполнили информацию");//контролируем заполнение информации

}

}

private void deleteButton\_Click(object sender, EventArgs e)//кнопка для удаления билета

{

if (MessageBox.Show(this, "Вы точно хотите удалить билет?", "", MessageBoxButtons.YesNo) == DialogResult.Yes)//просим потверждение действия

{

ticketlist.DeleteTicket(ticketlist.findById(currentId));

currentId = ticketlist.GetMaxID();

FillForm(ticketlist.findById(currentId));

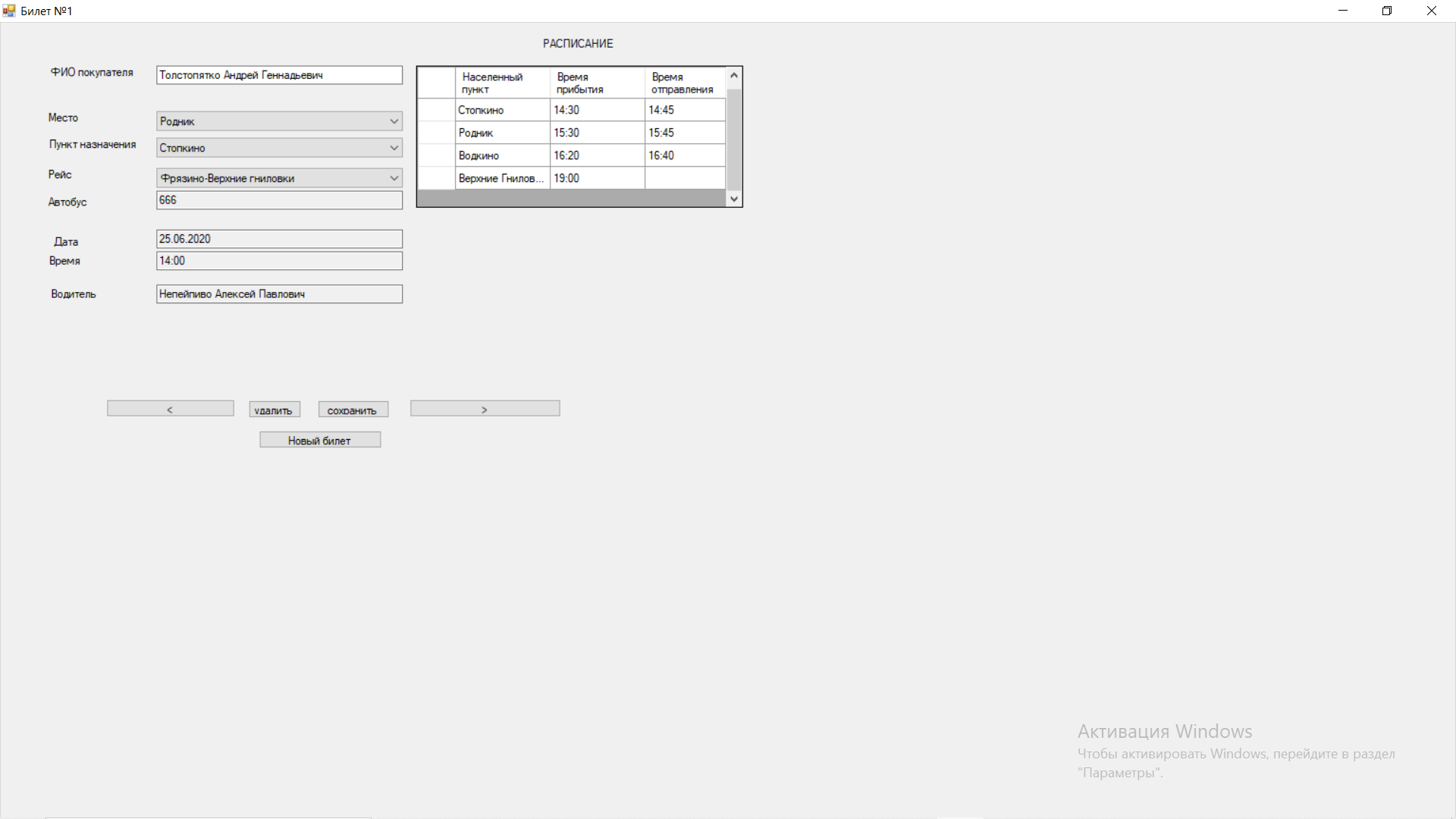
}

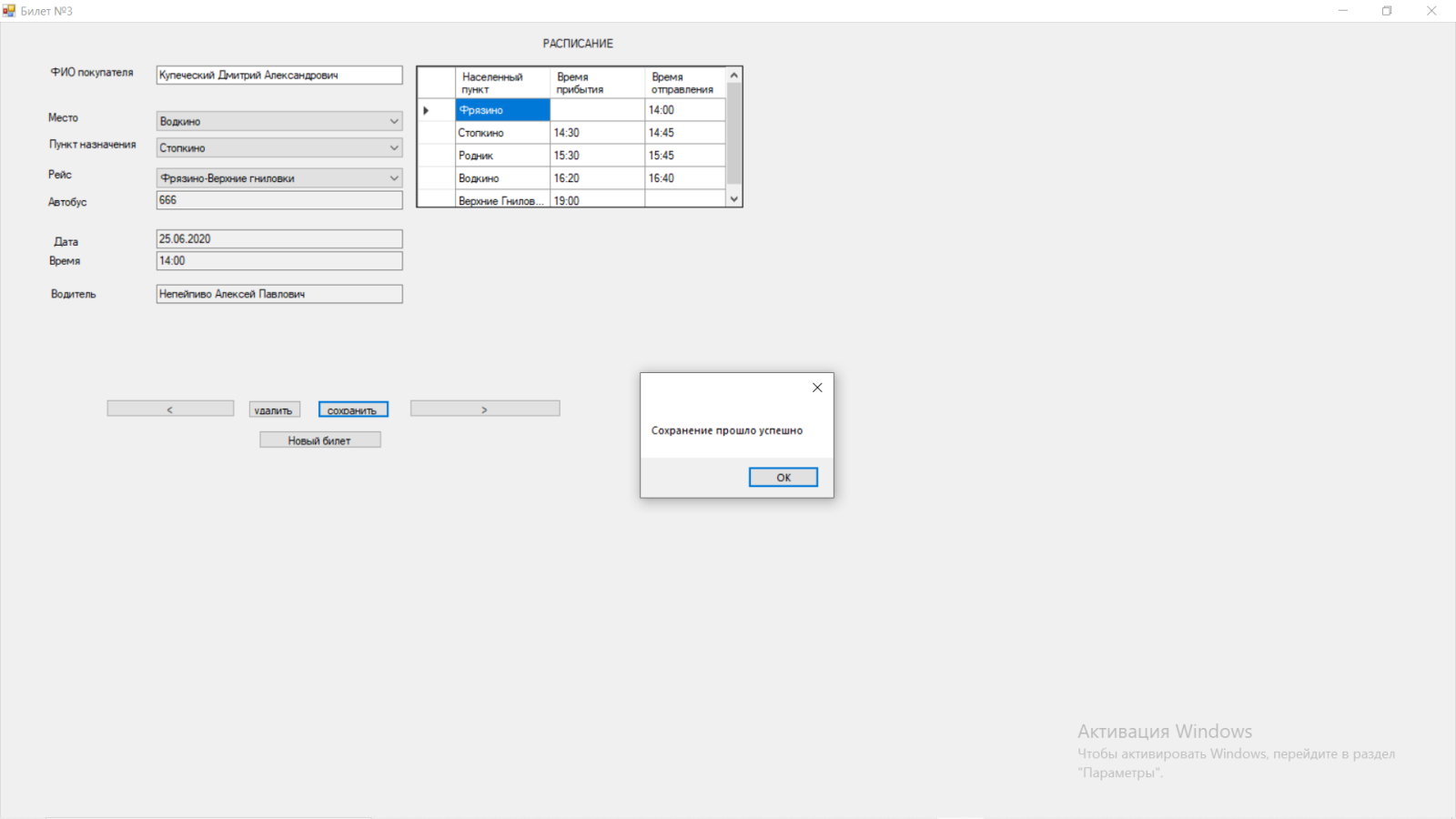
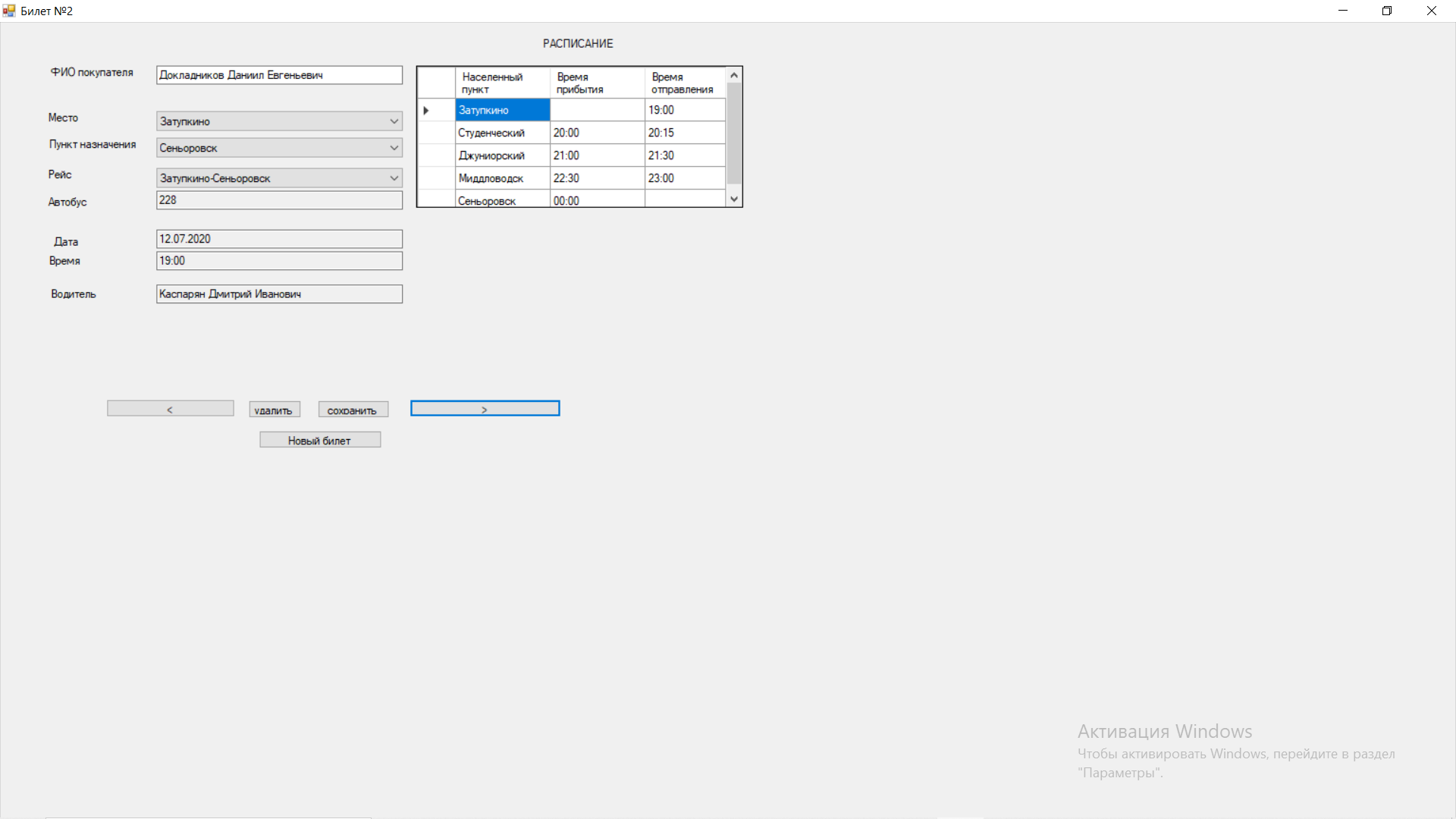
}

}

}

*Результат выполнения программы:*

**

**