Лабораторная работа №13

Стандартные диалоговые формы и меню.

*Постановка задачи*: В разработанное ранее приложение, в форму настроек стандартные диалоги выбора места хранения файлов и/или папки программы. Информация о новом расположении файлов программы должна запоминаться и использоваться. С помощью элемента Timer приложение должно периодически проводить сохранение данных без участия пользователя. Добавьте иконку NotifyIcon в системный лоток. Создайте меню на NotifyIcon с как минимум двумя действиями. Добавьте в приложение статусную строку, в которую должна выводиться информация о результатах работы программы. Например, слова «Успешно сохранено», «Папка данных успешно изменена», «Ошибка сохранения», «Введены неверные данные».

*Входные данные*:

* Data.xml;
* Settings.xml
* Authorization.xml

*Выходные данные*:

* Данные записываются в файл Data.xml, а настройки в Settings.xml.

*Рабочие переменные*:

* allCars, list, docCars, xRoot, id, model, num, price, deposit, address, xnode, attr, childnode, car, allServices, docClients, name, allClients, docClients, FIO, period, addservices, client, xDoc, clientElem, idAttr, fioElem, modelElem, priceElem, depositElem, addressElem, periodElem, addservicesElem, idText, fioText, modelText, priceText, depositText, addressText, periodText, addservicesText, index, max, fios, currentFio, clients, cars, addServices, currentId, currentClient, newClient, next nameModel, allFIOs, clientsList, clientForm.

*Классы и процедуры, созданные автором*

* class Car – класс для создания объектов-машин;
* GetModel() – возвращает название машины;
* GetPrice() ­– возвращает цену;
* GetDeposit() – возвращает залог;
* GetAddress() – возвращает адрес;
* class Cars – класс для специального списка машин;
* ReadCars() – метод создания списка машин(из файла);
* GetAllCars() – метод получения списка машин;
* class AddService – класс для доп.услуг;
* GetID() – возвращает идентификатор услуги;
* GetName() – возвращает название услуги;
* GetPrice() ­– возвращает цену услуги;
* class AddServices – класс для списка доп.услуг;
* ReadServices() – метод создания списка доп.услуг (из файла);
* FindByName() – возвращает услугу по её названию;
* GetAddServices() – возвращает список услуг;
* class Client – класс для создания клиентов;
* GetID() – возвращает идентификатор;
* GetFIO() – возвращает ФИО;
* GetPeriod() – возвращает срок аренды;
* GetCarModel() – возвращает название машины;
* GetCarPrice() – возвращает цену аренды;
* GetCarDeposit() – возвращает залог;
* GetCarAddress() – возвращает адрес пункта обслуживания;
* GetAddServices() – возвращает список выделенных доп.услуг;
* SetFIO() – устанавливает ФИО;
* SetCarModel() – устанавливает название машины;
* SetCarPrice() – устанавливает цену аренды;
* SetCarDeposit() – устанавливает залог;
* SetCarAddress() – устанавливает адрес пункта;
* SetPeriod() – устанавливает срок аренды;
* SetAddServices() – устанавливает список выбранных доп.услуг;
* GetAddServicesString() – возвращает строку с идентификаторами допюуслуг;
* GetPrevious() – получение предыдущего клиента;
* GetNext() – получение следующего клиента;
* GetMaxID() – получение последнего клиента;
* findByID() – нахлждение клиента по id;
* findByFIO() – нахлждение клиента по фио;
* GetAllFIO() – вывод списка ФИО всех клиентов;
* class ClientForm – класс формы клиента;
* ClientForm­\_Load() – загрузка формы клиента;
* FillForm() – заполнение формы;
* BuildCarsBox() – заполнение списка машин;
* BuildPriceBox() – заполнение списка цен;
* BuildDepositsBox() – заполнение списка залогов;
* BuildPeriodsBox() – заполнение списка сроков;
* BuildAddressesBox() – заполнение списка адресов;
* BuildAddServicesListBox() – заполнение списка доп.услуг;
* SaveThisForm() – сохранение обновленной формы;
* saveButton\_Click() – сохранение в файл;
* deleteClient\_Click() – удаление клиента;
* newClient\_Click() – создание нового клиента;
* next\_Click() – переход к след. Клиенту;
* previous\_Click() – переход к предыдущему клиенту;
* backToListButton\_Click() – переход к списку клиентов;
* calcButton() – подсчет итоговой суммы;
* ClientForm\_MouseMove() – картинка машины;
* class Authorization – класс для авторизации;
* class ClientsList – класс формы со списком клиентов;
* class FileOperations – класс для операций с файлом;
* class Rents – класс списка аренд;
* class Rent – класс аренды;
* class Addresses – класс для адресов компании;
* class Settings – класс настроек;
* class SettingsForm – класс формы настроек.
* Class About – класс формы о программе.
* Class Clients – класс клиентов.

*Текст программы:*

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;

namespace Lab10\_rent\_a\_car

{

public partial class AddressEdit : Form

{

string inName;

public AddressEdit()//инициализация

{

InitializeComponent();

try

{

FillForm();

inName = Data.CurrentAddress.Name;

}

catch

{

streetBox.Text = ""; houseBox.Text = "";

}

}

internal void FillForm() //запонение формы адреса

{

string[] array = Data.CurrentAddress.Name.Split(',');

streetBox.Text = array[0].Substring(4, array[0].Length - 4);

houseBox.Text = array[1].Substring(4, array[1].Length - 4);

}

private void saveButton\_Click(object sender, EventArgs e)//сохранение адреса

{

if (streetBox.Text == "" || houseBox.Text == "")

{

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Заполнены не все поля");

MessageBox.Show("Для сохранения заполните поле формы.", "Упс!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

SaveThisForm();

Data.Status = false;

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Есть несохраненные данные");

MessageBox.Show("Данные успешно изменены.", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

}

}

private void SaveThisForm()//сохранение формы

{

if (Data.Addresses.FindByName(inName) == null)

{

Data.CurrentAddress.Name = $"ул. {streetBox.Text}, д. {houseBox.Text}";

Data.CurrentAddress.Deleted = false;

Data.Addresses.AddAddress(Data.CurrentAddress);

}

else

{

Data.Addresses.FindByName(inName).Name = $"ул. {streetBox.Text}, д. {houseBox.Text}";

}

}

private void deleteButton\_Click(object sender, EventArgs e)//удаление адреса

{

if (MessageBox.Show("Вы точно хотите удалить данный адрес?", "!!!", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

Data.Addresses.FindByName(Data.CurrentAddress.Name).Deleted = true;

Data.CurrentAddress = null;

Data.Status = false;

this.Close();

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Есть несохраненные данные");

}

}

private void AddressEdit\_FormClosed(object sender, FormClosedEventArgs e)//внесение изменений в форму родителя

{

AddressesList addressesList = this.Owner as AddressesList;

addressesList.FillGrid();

}

private void AddressEdit\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Close();

try

{

AddressesList addList = this.Owner as AddressesList;

addList.Close();

}

catch

{

}

}

}

}

}

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;

namespace Lab10\_rent\_a\_car

{

public partial class AddressesList : Form

{

public AddressesList()

{

InitializeComponent();

FillGrid();

}

internal void FillGrid()

{

int i = 0;

AddressesGrid.Rows.Clear();

foreach (Address address in Data.Addresses.AllAddresses)

{

if (!address.Deleted)

{

AddressesGrid.Rows.Add();

int j = 0;

AddressesGrid.Rows[i].Cells[j].Value = address.Name; j++;

i++;

}

}

}

private void AddressesGrid\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

if (e.ColumnIndex != -1 & e.RowIndex != -1)

{

Address address = Data.Addresses.FindByName(AddressesGrid.Rows[e.RowIndex].Cells[0].Value.ToString());

Data.CurrentAddress = address;

AddressEdit addressEdit = new AddressEdit();

addressEdit.Owner = this;

addressEdit.ShowDialog();

}

}

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

{

Data.CurrentAddress = new Address();

AddressEdit addressEdit = new AddressEdit();

addressEdit.Owner = this;

addressEdit.ShowDialog();

}

private void AddressesList\_FormClosed(object sender, FormClosedEventArgs e)

{

MainForm mainForm = this.Owner as MainForm;

if (!Data.Status) { mainForm.Text = "Аренда\*"; }

}

private void AddressesList\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Close();

}

}

}

}

using System.Collections.Generic;

using System.Xml;

namespace Lab10\_rent\_a\_car

{

class AddServices

{

List<AddService> allServices;

public AddServices()

{

allServices = new List<AddService>();

}

public void AddService(AddService service) { allServices.Add(service); }

public AddService FindByName(string name){ return allServices.Find(x => x.Name == name); }

public AddService FindByID(int id) { return allServices.Find(x => x.ID == id); }

public List<AddService> AllServices { get { return allServices; } }

public void SaveServices()//сохранение данных всех клиентов в xml-файл

{

FileOperations.SaveAddServices(allServices);

}

public void DelService(AddService addService) { allServices.Remove(addService); }

public void ReplaceInfoService(AddService addService)//метод для обновления информации

{

int index = allServices.FindIndex(x => x.ID == addService.ID);

allServices[index] = addService;

}

public int GetMaxID()//получение максимального значения id

{

int max = 0;

foreach (AddService addService in allServices)

{

if (max < addService.ID) max = addService.ID;

}

return max;

}

}

class AddService

{

private int id;

private string name;

private decimal price;

private bool deleted;

public AddService(int id)

{

this.id = id;

}

public AddService(int id, string name, decimal price, bool deleted)

{

this.id = id;

this.name = name;

this.price = price;

this.deleted = deleted;

}

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

public string Name { get { return name; } set { name = value; } }

public decimal Price{ get { return price; } set { price = value; } }

public bool Deleted { get { return deleted; } set { deleted = value; } }

}

}

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;

namespace Lab10\_rent\_a\_car

{

public partial class AddServicesList : Form

{

public AddServicesList()

{

InitializeComponent();

FillGrid();

}

internal void FillGrid()

{

int i = 0;

AddServicesGrid.Rows.Clear();

foreach (AddService addService in Data.AddServices.AllServices)

{

if (!addService.Deleted)

{

AddServicesGrid.Rows.Add();

int j = 0;

AddServicesGrid.Rows[i].Cells[j].Value = addService.Name; j++;

AddServicesGrid.Rows[i].Cells[j].Value = addService.Price; j++;

i++;

}

}

}

private void AddServicesGrid\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

string name = "";

if (e.ColumnIndex != -1 & e.RowIndex != -1)

{

name = AddServicesGrid.Rows[e.RowIndex].Cells[0].Value.ToString();

Data.CurrentService = Data.AddServices.FindByName(name);

ServiceView serviceView = new ServiceView();

serviceView.Owner = this;

serviceView.ShowDialog();

}

}

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

{

ServiceEdit serviceEdit = new ServiceEdit();

serviceEdit.Owner = this;

serviceEdit.ShowDialog();

}

private void AddServicesList\_FormClosed(object sender, FormClosedEventArgs e)

{

MainForm mainForm = this.Owner as MainForm;

if (!Data.Status) { mainForm.Text = "Аренда\*"; }

}

private void AddServicesList\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Close();

}

}

}

}

using System;

using System.Windows.Forms;

using System.Xml;

namespace Lab10\_rent\_a\_car

{ public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

MainForm mainForm = new MainForm();

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

{

if (FileOperations.CheckLogin(loginField.Text, passwordField.Text))

{

FileOperations.ReadUsers();

Users.CurrentUser = Users.FindByLogin(loginField.Text);

this.Hide();

mainForm.ShowDialog();

this.Show();

Close();

}

else

MessageBox.Show("Введен неправильный логин или пароль!", "Внимание!!!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

Close();

}

if (e.KeyCode == Keys.Enter)

{

loginButton\_Click(sender,e);

}

if (e.KeyCode == Keys.Down)

{

passwordField.Select();

}

if (e.KeyCode == Keys.Up)

{

loginField.Select();

}

}

}

}

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.Text.RegularExpressions;

namespace Lab10\_rent\_a\_car

{

public partial class ClientEdit : Form

{

internal ClientEdit()//новый

{

InitializeComponent();

Data.CurrentClient = new Client();

Data.Clients.AddClient(Data.CurrentClient);

}

internal ClientEdit(Client client)//существующий

{

InitializeComponent();

FillForm(client);

}

internal void FillForm(Client client)//заполнение

{

FIOBox.Text = client.FIO;

passBox.Text = client.Passport;

licenseBox.Text = client.License;

if (client.Category.Contains(catA.Text)) catA.Checked = true; else catA.Checked = false;

if (client.Category.Contains(catB.Text)) catB.Checked = true; else catB.Checked = false;

if (client.Category.Contains(catC.Text)) catC.Checked = true; else catC.Checked = false;

if (client.Category.Contains(catD.Text)) catD.Checked = true; else catD.Checked = false;

if (client.Category.Contains(catE.Text)) catE.Checked = true; else catE.Checked = false;

addressBox.Text = client.Address;

}

private void SaveThisForm()//сохранение обновленной (или нет) формы

{

Data.CurrentClient.FIO = FIOBox.Text;

Data.CurrentClient.Passport = passBox.Text;

Data.CurrentClient.License = licenseBox.Text;

Data.CurrentClient.Category = "";

if (catA.Checked) { Data.CurrentClient.Category += "A"; }

if (catB.Checked) { Data.CurrentClient.Category += "B"; }

if (catC.Checked) { Data.CurrentClient.Category += "C"; }

if (catD.Checked) { Data.CurrentClient.Category += "D"; }

if (catE.Checked) { Data.CurrentClient.Category += "E"; }

Data.CurrentClient.Address = addressBox.Text;

Data.CurrentClient.Saved = true;

if (Data.Clients.SavedClients.FindIndex(x => x.FIO == Data.CurrentClient.FIO) == -1)

Data.Clients.SavedClients.Add(Data.CurrentClient);

Data.Clients.ReplaceInfoClient(Data.CurrentClient);

}

private void saveButton\_Click(object sender, EventArgs e)//сохранение

{

if(FIOBox.Text == "" || passBox.Text == "" || licenseBox.Text == "" || addressBox.Text == "")

{

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Заполнены не все поля");

MessageBox.Show("Для сохранения заполните все поля формы.", "Упс!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

SaveThisForm();

Data.Status = false;

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Есть несохраненные данные");

MessageBox.Show("Данные успешно изменены.", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

ClientView clientView = this.Owner as ClientView;

if(clientView != null) clientView.Close();

}

}

private void passBox\_KeyPress(object sender, KeyPressEventArgs e)//вводить только цифры

{

if (Char.IsDigit(e.KeyChar) || Char.IsWhiteSpace(e.KeyChar) || Char.IsControl(e.KeyChar))

return;

else

e.Handled = true;

}

private void licenseBox\_KeyPress(object sender, KeyPressEventArgs e)//вводить только цифры

{

passBox\_KeyPress(sender, e);

}

private void FIOBox\_KeyPress(object sender, KeyPressEventArgs e)//ввод только букв

{

if (Char.IsLetter(e.KeyChar) || Char.IsWhiteSpace(e.KeyChar) || Char.IsControl(e.KeyChar))

return;

else

e.Handled = true;

}

private void ClientEdit\_FormClosed(object sender, FormClosedEventArgs e)//изменения в форму родителя

{

try

{

ClientView clientView = this.Owner as ClientView;

if (FIOBox.Text == "" || passBox.Text == "" || passBox.Text == "" || licenseBox.Text == "" || addressBox.Text == "")

{

Data.Clients.DelClient(Data.CurrentClient);

clientView.Close();

}

else

{

clientView.FillForm(Data.CurrentClient);

}

}

catch

{

MainForm mainForm = this.Owner as MainForm;

mainForm.FillClients();

}

}

private void passBox\_Leave(object sender, EventArgs e)//проверка правильности ввода паспорта

{

Regex regex = new Regex(@"^\d{4}\s\d{6}$");

if (!regex.IsMatch(passBox.Text))

{

MessageBox.Show("Введен неправильный формат данных.", "Внимание!!!", MessageBoxButtons.OK, MessageBoxIcon.Warning);

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Введены неверные данные");

}

}

private void licenseBox\_Leave(object sender, EventArgs e)//проверка правильности ввода прав

{

Regex regex = new Regex(@"^\d{4}\s\d{6}$");

if (!regex.IsMatch(licenseBox.Text))

{

MessageBox.Show("Введен неправильный формат данных.", "Внимание!!!", MessageBoxButtons.OK, MessageBoxIcon.Warning);

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Введены неверные данные");

}

}

private void ClientEdit\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Close();

try

{

ClientView clientView = this.Owner as ClientView;

clientView.Close();

}

catch

{

}

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab10\_rent\_a\_car

{

class Clients

{

List<Client> allClients = new List<Client>();

List<Client> savedClients;

public Clients()

{

allClients = new List<Client>();

savedClients = new List<Client>();

}

public void AddClient(Client client) { allClients.Add(client); if (client.Saved){ savedClients.Add(client); } }

public void DelClient(Client client) { allClients.Remove(client); if (client.Saved) { savedClients.Remove(client); } }

public List<Client> AllClients { get { return allClients; } }

public List<Client> SavedClients { get { return savedClients; } }

public Client FindByFIO(string fio)//метод для нахождения клиента по id

{

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

}

public void SaveClients()//сохранение данных всех клиентов в xml-файл

{

FileOperations.SaveClients(allClients);

}

public void ReplaceInfoClient(Client client)//метод для обновления информации

{

int index = allClients.FindIndex(x => x.FIO == client.FIO);

allClients[index] = client;

}

}

class Client

{

private string fio; private string passport;

private string license; private string category;

private string address; private bool saved;

private List<int> rents;

public Client()

{

}

public Client (string fio, string passport, string license, string category, string address,bool saved, List<int> rents)

{

this.fio = fio; this.passport = passport;

this.license = license; this.category = category;

this.address = address; this.saved = saved;

this.rents = rents;

}

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

public string Passport { get { return passport; } set { passport = value; } }

public string License { get { return license; } set { license = value; } }

public string Category { get { return category; } set { category = value; } }

public string Address { get { return address; } set { address = value; } }

public bool Saved { get { return saved; } set { saved = value; } }

public List<int> Rents { get { return rents; } set { rents = value; } }

}

}

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;

namespace Lab10\_rent\_a\_car

{

public partial class ClientView : Form

{

public ClientView()//инициализация

{

InitializeComponent();

FillForm(Data.CurrentClient);

}

internal void FillForm (Client client)//заполнение формы просмотра клиента

{

FIOBox.Text = client.FIO;

PassBox.Text = client.Passport;

licenseBox.Text = client.License;

if (client.Category.Contains(catA.Text)) catA.Checked = true; else catA.Checked = false;

if (client.Category.Contains(catB.Text)) catB.Checked = true; else catB.Checked = false;

if (client.Category.Contains(catC.Text)) catC.Checked = true; else catC.Checked = false;

if (client.Category.Contains(catD.Text)) catD.Checked = true; else catD.Checked = false;

if (client.Category.Contains(catE.Text)) catE.Checked = true; else catE.Checked = false;

addressBox.Text = client.Address;

if (client.Rents != null)

{

rentsListBox.Items.Clear();

foreach (int rentID in client.Rents)

{

rentsListBox.Items.Add($"Договор №{rentID}");

}

}

else { rentsListBox.Items.Clear(); }

}

private void editButton\_Click(object sender, EventArgs e)//открытие формы изменения

{

ClientEdit clientEdit = new ClientEdit(Data.CurrentClient);

clientEdit.Owner = this;

clientEdit.ShowDialog();

}

private void deleteButton\_Click(object sender, EventArgs e)//удаление

{

if (MessageBox.Show("Вы точно хотите удалить данного клиента?", "!!!", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

Data.Status = false;

Data.Clients.FindByFIO(Data.CurrentClient.FIO).Saved = false;

Data.Clients.SavedClients.Remove(Data.CurrentClient);

Data.CurrentClient = null;

Data.Clients.SaveClients();

this.Close();

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Есть несохраненные данные");

}

}

public void ClientView\_FormClosed(object sender, FormClosedEventArgs e)//изменения в форму родителя

{

MainForm mainForm = this.Owner as MainForm;

mainForm.FillClients();

}

private void rentsListBox\_SelectedIndexChanged(object sender, EventArgs e)//открытие формы соответсвующей аренды

{

if (rentsListBox.SelectedItem != null)

{

string myID = "";

//подготовка формы аренды

myID = rentsListBox.SelectedItem.ToString().Substring(9);

Data.CurrentRent = Data.Rents.FindByID(int.Parse(myID));

MainForm mainForm = this.Owner as MainForm;

RentView rentView = new RentView();

rentView.Owner = mainForm;

rentView.ShowDialog();//открытие формы просмотра аренды

}

}

private void ClientView\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Close();

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml;

using System.Xml.Linq;

namespace Lab10\_rent\_a\_car

{

static class FileOperations

{

public static string GetStorage()

{

XmlDocument doc = new XmlDocument();

doc.Load("Storage.xml");

XmlElement xRoot = doc.DocumentElement;

return xRoot.InnerText;

}

public static void SaveStorage(string storage)

{

XmlDocument doc = new XmlDocument();

XmlElement elem = doc.CreateElement("Storage");

elem.InnerText = storage;

doc.AppendChild(elem);

doc.Save("Storage.xml");

}

public static bool CheckLogin(string myLogin, string myPassword)

{

bool check = false;

XmlDocument docAuthor = new XmlDocument();

docAuthor.Load(Data.Storage + "DB/Settings.xml");

XmlElement xRoot = docAuthor.DocumentElement;

// обход всех узлов в корневом элементе

foreach (XmlNode xnode in xRoot)

{

if (xnode.Name == "Users")

{

foreach (XmlNode node in xnode.ChildNodes)

{

string login = ""; string password = "";

// обходим дочерний узел элемента

foreach (XmlNode childnode in node.ChildNodes)

{

if (childnode.Name == "login")

{

login = childnode.InnerText;

}

if (childnode.Name == "password")

{

password = childnode.InnerText;

}

}

if (login == myLogin && password == myPassword) { check = true; }

}

}

}

return check;

}

public static void ReadUsers()

{

XmlDocument docAuthor = new XmlDocument();

docAuthor.Load(Data.Storage + "DB/Settings.xml");

XmlElement xRoot = docAuthor.DocumentElement;

// обход всех узлов в корневом элементе

foreach (XmlNode xnode in xRoot)

{

if (xnode.Name == "Users")

{

foreach (XmlNode node in xnode.ChildNodes)

{

string login = ""; string password = ""; string status = "";

// обходим дочерний узел элемента

foreach (XmlNode childnode in node.ChildNodes)

{

if (childnode.Name == "login")

{

login = childnode.InnerText;

}

if (childnode.Name == "password")

{

password = childnode.InnerText;

}

if (childnode.Name == "status")

{

status = childnode.InnerText;

}

}

Users.allUsers.Add(new User(login, password, status));

}

}

}

}

public static Objects ReadObjects()

{

Objects list = new Objects();

XmlDocument docObjects = new XmlDocument();

docObjects.Load(Data.Storage + "DB/RefBooks.xml");

XmlElement xRoot = docObjects.DocumentElement;

// обход всех узлов в корневом элементе

foreach (XmlNode xnode in xRoot)

{

if (xnode.Name == "Objects")

{

foreach (XmlNode node in xnode.ChildNodes)

{

int id = 0; string model = ""; string num = ""; decimal price = 0;

decimal mdeposit = 0; bool deleted = false;

int quantity = 0; int inRent = 0; int inStock = 0; int repair = 0;

int minT = 0; int maxT = 0; string description = ""; string damage = "";

string measure = ""; List<int> rents = new List<int>();

foreach (XmlNode childnode in node.ChildNodes)

{

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

if (childnode.Name == "name") { model = childnode.InnerText; }

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

if (childnode.Name == "price") { price = decimal.Parse(childnode.InnerText); }

if (childnode.Name == "rents")

{

if (childnode.InnerText.Contains(","))

{

string[] array = childnode.InnerText.Split(',');

foreach (string elem in array) { rents.Add(int.Parse(elem)); }

}

if (childnode.InnerText != "" & !childnode.InnerText.Contains(","))

{

rents.Add(int.Parse(childnode.InnerText));

}

}

if (childnode.Name == "deleted") { deleted = bool.Parse(childnode.InnerText); }

if (childnode.Name == "quantity") { quantity = int.Parse(childnode.InnerText); }

if (childnode.Name == "inRent") { inRent = int.Parse(childnode.InnerText); }

if (childnode.Name == "inStock") { inStock = int.Parse(childnode.InnerText); }

if (childnode.Name == "repair") { repair = int.Parse(childnode.InnerText); }

if (childnode.Name == "minT") { minT = int.Parse(childnode.InnerText); }

if (childnode.Name == "maxT") { maxT = int.Parse(childnode.InnerText); }

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

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

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

}

Object obj = new Object(id, model, num, price, mdeposit, deleted, quantity, inRent, inStock,

repair, minT, maxT, description, damage, measure, rents);

list.AddObject(obj);

}

}

}

return list;

}

public static void SaveObjects(List<Object> allObjects)

{

XmlDocument xDoc = new XmlDocument();

XmlElement xRoot;

try

{

xDoc.Load(Data.Storage + "DB/RefBooks.xml");

xRoot = xDoc.DocumentElement;

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

foreach (XmlNode n in childnodes)

{

if (n.Name == "Objects")

{

xRoot.RemoveChild(n);

}

}

}

catch

{

xRoot = xDoc.CreateElement("Data");

xDoc.AppendChild(xRoot);

}

XmlElement objs = xDoc.CreateElement("Objects");

foreach (Object obj in allObjects)

{

XmlElement objElem = xDoc.CreateElement("Object");

// создаем элементы

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

XmlElement nameElem = xDoc.CreateElement("name");

XmlElement numElem = xDoc.CreateElement("num");

XmlElement priceElem = xDoc.CreateElement("price");

XmlElement rentsElem = xDoc.CreateElement("rents");

XmlElement deletedElem = xDoc.CreateElement("deleted");

XmlElement quantityElem = xDoc.CreateElement("quantity");

XmlElement inRentElem = xDoc.CreateElement("inRent");

XmlElement inStockElem = xDoc.CreateElement("inStock");

XmlElement repairElem = xDoc.CreateElement("repair");

XmlElement minTElem = xDoc.CreateElement("minT");

XmlElement maxTElem = xDoc.CreateElement("maxT");

XmlElement descriptionElem = xDoc.CreateElement("description");

XmlElement damageElem = xDoc.CreateElement("damage");

XmlElement measureElem = xDoc.CreateElement("measure");

// создаем текстовые значения для элементов и атрибута

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

XmlText nameText = xDoc.CreateTextNode(obj.Name);

XmlText numText = xDoc.CreateTextNode(obj.Num);

XmlText priceText = xDoc.CreateTextNode(obj.Price.ToString());

string rents = "";

if(obj.Rents!=null)

{

if (obj.Rents.Count != 0)

{

foreach (int num in obj.Rents)

{

rents += num + ",";

}

rents = rents.Substring(0, rents.Length - 1);

}

}

XmlText rentsText = xDoc.CreateTextNode(rents);

XmlText deletedText = xDoc.CreateTextNode(obj.Deleted.ToString());

XmlText quantityText = xDoc.CreateTextNode(obj.Quantity.ToString());

XmlText inRentText = xDoc.CreateTextNode(obj.InRent.ToString());

XmlText inStockText = xDoc.CreateTextNode(obj.InStock.ToString());

XmlText repairText = xDoc.CreateTextNode(obj.Repair.ToString());

XmlText minTText = xDoc.CreateTextNode(obj.MinT.ToString());

XmlText maxTText = xDoc.CreateTextNode(obj.MaxT.ToString());

XmlText descriptionText = xDoc.CreateTextNode(obj.Description);

XmlText damageText = xDoc.CreateTextNode(obj.Damage);

XmlText measureText = xDoc.CreateTextNode(obj.Measure);

//добавляем узлы

idElem.AppendChild(idText);

nameElem.AppendChild(nameText);

numElem.AppendChild(numText);

priceElem.AppendChild(priceText);

rentsElem.AppendChild(rentsText);

deletedElem.AppendChild(deletedText);

quantityElem.AppendChild(quantityText);

inRentElem.AppendChild(inRentText);

inStockElem.AppendChild(inStockText);

repairElem.AppendChild(repairText);

minTElem.AppendChild(minTText);

maxTElem.AppendChild(maxTText);

descriptionElem.AppendChild(descriptionText);

damageElem.AppendChild(damageText);

measureElem.AppendChild(measureText);

objElem.AppendChild(idElem);

objElem.AppendChild(nameElem);

objElem.AppendChild(numElem);

objElem.AppendChild(priceElem);

objElem.AppendChild(rentsElem);

objElem.AppendChild(deletedElem);

objElem.AppendChild(quantityElem);

objElem.AppendChild(inRentElem);

objElem.AppendChild(inStockElem);

objElem.AppendChild(repairElem);

objElem.AppendChild(minTElem);

objElem.AppendChild(maxTElem);

objElem.AppendChild(descriptionElem);

objElem.AppendChild(damageElem);

objElem.AppendChild(measureElem);

objs.AppendChild(objElem);

}

xRoot.AppendChild(objs);

xDoc.Save(Data.Storage + "DB/RefBooks.xml");

}

public static AddServices ReadServices()

{

AddServices list = new AddServices();

XmlDocument docClients = new XmlDocument();

docClients.Load(Data.Storage + "DB/RefBooks.xml");

XmlElement xRoot = docClients.DocumentElement;

// обход всех узлов в корневом элементе

foreach (XmlNode xnode in xRoot)

{

if (xnode.Name == "AddServices")

{

foreach (XmlNode node in xnode.ChildNodes)

{

int id = 0; string name = ""; int price = 0; bool deleted = false;

foreach (XmlNode childnode in node.ChildNodes)

{

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

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

if (childnode.Name == "price") { price = int.Parse(childnode.InnerText); }

if (childnode.Name == "deleted") { deleted = bool.Parse(childnode.InnerText); }

}

AddService addService = new AddService(id, name, price, deleted);

list.AddService(addService);

}

}

}

return list;

}

public static void SaveAddServices(List<AddService> allServices)

{

XmlDocument xDoc = new XmlDocument();

XmlElement xRoot;

try

{

xDoc.Load(Data.Storage + "DB/RefBooks.xml");

xRoot = xDoc.DocumentElement;

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

foreach (XmlNode n in childnodes)

{

if (n.Name == "AddServices")

{

xRoot.RemoveChild(n);

}

}

}

catch

{

xRoot = xDoc.CreateElement("Data");

xDoc.AppendChild(xRoot);

}

XmlElement services = xDoc.CreateElement("AddServices");

foreach (AddService service in allServices)

{

// создаем новый элемент client

XmlElement serviceElem = xDoc.CreateElement("AddService");

// создаем элементы

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

XmlElement nameElem = xDoc.CreateElement("name");

XmlElement priceElem = xDoc.CreateElement("price");

XmlElement deletedElem = xDoc.CreateElement("deleted");

// создаем текстовые значения для элементов и атрибута

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

XmlText nameText = xDoc.CreateTextNode(service.Name);

XmlText priceText = xDoc.CreateTextNode(service.Price.ToString());

XmlText deletedText = xDoc.CreateTextNode(service.Deleted.ToString());

//добавляем узлы

idElem.AppendChild(idText);

nameElem.AppendChild(nameText);

priceElem.AppendChild(priceText);

deletedElem.AppendChild(deletedText);

serviceElem.AppendChild(idElem);

serviceElem.AppendChild(nameElem);

serviceElem.AppendChild(priceElem);

serviceElem.AppendChild(deletedElem);

services.AppendChild(serviceElem);

}

xRoot.AppendChild(services);

xDoc.Save(Data.Storage + "DB/RefBooks.xml");

}

public static Addresses ReadAddresses()

{

Addresses list = new Addresses();

XmlDocument docClients = new XmlDocument();

docClients.Load(Data.Storage + "DB/RefBooks.xml");

XmlElement xRoot = docClients.DocumentElement;

// обход всех узлов в корневом элементе

foreach (XmlNode xnode in xRoot)

{

if (xnode.Name == "CompAddresses")

{

foreach (XmlNode childnode in xnode.ChildNodes)

{

string address = ""; bool deleted = false;

foreach (XmlNode node in childnode.ChildNodes)

{

if(node.Name == "name")

{

address = node.InnerText;

}

if (node.Name == "deleted")

{

deleted = bool.Parse(node.InnerText);

}

}

list.AddAddress(new Address(address, deleted));

}

}

}

return list;

}

public static void SaveAddresses(List<Address> allAddresses)

{

XmlDocument xDoc = new XmlDocument();

XmlElement xRoot;

try

{

xDoc.Load(Data.Storage + "DB/RefBooks.xml");

xRoot = xDoc.DocumentElement;

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

foreach (XmlNode n in childnodes)

{

if (n.Name == "CompAddresses")

{

xRoot.RemoveChild(n);

}

}

}

catch

{

xRoot = xDoc.CreateElement("Data");

xDoc.AppendChild(xRoot);

}

XmlElement addresses = xDoc.CreateElement("CompAddresses");

foreach (Address address in allAddresses)

{

XmlElement addressElem = xDoc.CreateElement("CompAddress");

XmlElement nameElem = xDoc.CreateElement("name");

XmlElement deletedElem = xDoc.CreateElement("deleted");

XmlText nameText = xDoc.CreateTextNode(address.Name);

XmlText deletedText = xDoc.CreateTextNode(address.Deleted.ToString());

nameElem.AppendChild(nameText);

deletedElem.AppendChild(deletedText);

addressElem.AppendChild(nameElem);

addressElem.AppendChild(deletedElem);

addresses.AppendChild(addressElem);

}

xRoot.AppendChild(addresses);

xDoc.Save(Data.Storage + "DB/RefBooks.xml");

}

public static Rents ReadRents()

{

Rents list = new Rents();

XmlDocument docClients = new XmlDocument();

docClients.Load(Data.Storage + "DB/Data.xml");

XmlElement xRoot = docClients.DocumentElement;

// обход всех узлов в корневом элементе

foreach (XmlNode xnode in xRoot)

{

if (xnode.Name == "Rents")

{

foreach (XmlNode node in xnode.ChildNodes)

{

int ID = 0; string clientFIO = ""; string name = ""; decimal price = 0; string depositType = "";

decimal depositMoney = 0; string deposit = ""; string measure = ""; double period = 0;

string compAddress = ""; string secondAddress = ""; List<int> addservices = new List<int>();

bool saved = false; DateTime dateTime = new DateTime(); string status = ""; string description = "";

foreach (XmlNode childnode in node.ChildNodes)

{

if (childnode.Name == "ID") { ID = int.Parse(childnode.InnerText); }

if (childnode.Name == "ClientFIO") { clientFIO = childnode.InnerText; }

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

if (childnode.Name == "Price") { price = decimal.Parse(childnode.InnerText); }

if (childnode.Name == "DepositType") { depositType = childnode.InnerText; }

if (childnode.Name == "Deposit")

{

if(depositType == "money") { depositMoney = decimal.Parse(childnode.InnerText); }

if(depositType == "doc" || depositType == "product"){ deposit = childnode.InnerText; }

}

if (childnode.Name == "Measure") { measure = childnode.InnerText; }

if (childnode.Name == "Period") { period = double.Parse(childnode.InnerText); }

if (childnode.Name == "CompAddress") { compAddress = childnode.InnerText; }

if (childnode.Name == "SecondAddress") { secondAddress = childnode.InnerText; }

if (childnode.Name == "AddServices")

{

if (childnode.InnerText != "")

{

string[] array = childnode.InnerText.Split(',');

foreach (string elem in array) { addservices.Add(int.Parse(elem)); }

}

}

if (childnode.Name == "Saved") { saved = bool.Parse(childnode.InnerText); }

if (childnode.Name == "Date")

{

string[] array = childnode.InnerText.Split('.');

dateTime = new DateTime(int.Parse(array[2]), int.Parse(array[1]), int.Parse(array[0]));

}

if (childnode.Name == "Time")

{

string[] array = childnode.InnerText.Split(':');

dateTime = dateTime.AddHours(int.Parse(array[0]));

dateTime = dateTime.AddMinutes(int.Parse(array[1]));

}

if(childnode.Name == "Status") { status = childnode.InnerText; }

if (childnode.Name == "Description") { description = childnode.InnerText; }

}

Rent rent = new Rent(ID, clientFIO, name, price, depositType,depositMoney, deposit, period, measure,

compAddress, secondAddress, addservices, saved, dateTime, status, description);

list.AddRent(rent);

}

}

}

return list;

}

public static void SaveRents(List<Rent> allRents)//сохранение данных всех клиентов в xml-файл

{

XmlDocument xDoc = new XmlDocument();

XmlElement xRoot;

try

{

xDoc.Load(Data.Storage + "DB/Data.xml");

xRoot = xDoc.DocumentElement;

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

foreach (XmlNode n in childnodes)

{

if (n.Name == "Rents")

{

xRoot.RemoveChild(n);

}

}

}

catch

{

xRoot = xDoc.CreateElement("Data");

xDoc.AppendChild(xRoot);

}

XmlElement rents = xDoc.CreateElement("Rents");

foreach (Rent rent in allRents)

{

// создаем новый элемент

XmlElement rentElem = xDoc.CreateElement("Rent");

// создаем элементы

XmlElement IDElem = xDoc.CreateElement("ID");

XmlElement clFIOElem = xDoc.CreateElement("ClientFIO");

XmlElement nameElem = xDoc.CreateElement("Name");

XmlElement priceElem = xDoc.CreateElement("Price");

XmlElement depositTypeElem = xDoc.CreateElement("DepositType");

XmlElement depositElem = xDoc.CreateElement("Deposit");

XmlElement measureElem = xDoc.CreateElement("Measure");

XmlElement periodElem = xDoc.CreateElement("Period");

XmlElement compAddElem = xDoc.CreateElement("CompAddress");

XmlElement secondAddressElem = xDoc.CreateElement("SecondAddress");

XmlElement addservicesElem = xDoc.CreateElement("AddServices");

XmlElement savedElem = xDoc.CreateElement("Saved");

XmlElement dateElem = xDoc.CreateElement("Date");

XmlElement timeElem = xDoc.CreateElement("Time");

XmlElement statusElem = xDoc.CreateElement("Status");

XmlElement descriptionElem = xDoc.CreateElement("Description");

// создаем текстовые значения для элементов и атрибута

XmlText clIDText = xDoc.CreateTextNode(rent.ID.ToString());

XmlText clFIOText = xDoc.CreateTextNode(rent.ClientFIO);

XmlText nameText = xDoc.CreateTextNode(rent.Name);

XmlText priceText = xDoc.CreateTextNode(rent.Price.ToString());

XmlText depositTypeText = xDoc.CreateTextNode(rent.DepositType);

XmlText depositText = xDoc.CreateTextNode("");

if(rent.DepositMoney != 0){depositText = xDoc.CreateTextNode(rent.DepositMoney.ToString()); }

else { depositText = xDoc.CreateTextNode(rent.Deposit.ToString()); }

XmlText measureText = xDoc.CreateTextNode(rent.Measure);

XmlText periodText = xDoc.CreateTextNode(rent.Period.ToString());

XmlText compAddText = xDoc.CreateTextNode(rent.CompAddress);

XmlText secondAddressText = xDoc.CreateTextNode(rent.SecondAddress);

XmlText addservicesText = xDoc.CreateTextNode("");

if (rent.AddServicesString != null || rent.AddServicesString != "")

addservicesText = xDoc.CreateTextNode(rent.AddServicesString);

XmlText savedText = xDoc.CreateTextNode(rent.Saved.ToString());

XmlText dateText = xDoc.CreateTextNode($"{rent.DateTime.Day}.{rent.DateTime.Month}.{rent.DateTime.Year}");

XmlText timeText = xDoc.CreateTextNode($"{rent.DateTime.Hour}:{rent.DateTime.Minute}");

XmlText statusText = xDoc.CreateTextNode(rent.Status);

XmlText descriptionText = xDoc.CreateTextNode(rent.Description);

//добавляем узлы

IDElem.AppendChild(clIDText);

clFIOElem.AppendChild(clFIOText);

nameElem.AppendChild(nameText);

priceElem.AppendChild(priceText);

depositTypeElem.AppendChild(depositTypeText);

depositElem.AppendChild(depositText);

measureElem.AppendChild(measureText);

periodElem.AppendChild(periodText);

compAddElem.AppendChild(compAddText);

secondAddressElem.AppendChild(secondAddressText);

addservicesElem.AppendChild(addservicesText);

savedElem.AppendChild(savedText);

dateElem.AppendChild(dateText);

timeElem.AppendChild(timeText);

statusElem.AppendChild(statusText);

descriptionElem.AppendChild(descriptionText);

rentElem.AppendChild(IDElem);

rentElem.AppendChild(clFIOElem);

rentElem.AppendChild(nameElem);

rentElem.AppendChild(priceElem);

rentElem.AppendChild(depositTypeElem);

rentElem.AppendChild(depositElem);

rentElem.AppendChild(measureElem);

rentElem.AppendChild(periodElem);

rentElem.AppendChild(compAddElem);

rentElem.AppendChild(secondAddressElem);

rentElem.AppendChild(addservicesElem);

rentElem.AppendChild(savedElem);

rentElem.AppendChild(dateElem);

rentElem.AppendChild(timeElem);

rentElem.AppendChild(statusElem);

rentElem.AppendChild(descriptionElem);

rents.AppendChild(rentElem);

}

xRoot.AppendChild(rents);

xDoc.Save(Data.Storage + "DB/Data.xml");

}

public static Clients ReadClients()

{

Clients list = new Clients();

XmlDocument docClients = new XmlDocument();

docClients.Load(Data.Storage + "DB/Data.xml");

XmlElement xRoot = docClients.DocumentElement;

// обход всех узлов в корневом элементе

foreach (XmlNode xnode in xRoot)

{

if (xnode.Name == "Clients")

{

foreach (XmlNode node in xnode.ChildNodes)

{

string FIO = ""; string passport = ""; string license = "";

string category = ""; string address = ""; bool saved = false;

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

foreach (XmlNode childnode in node.ChildNodes)

{

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

if (childnode.Name == "Passport") { passport = childnode.InnerText; }

if (childnode.Name == "License") { license = childnode.InnerText; }

if (childnode.Name == "Category") { category = childnode.InnerText; }

if (childnode.Name == "Address") { address = childnode.InnerText; }

if (childnode.Name == "Saved") { saved = bool.Parse(childnode.InnerText); }

if (childnode.Name == "Rents")

{

if (childnode.InnerText != "")

{

string[] array = childnode.InnerText.Split(',');

foreach (string elem in array) { rents.Add(int.Parse(elem)); }

}

}

}

Client client = new Client(FIO, passport, license, category, address, saved, rents);

list.AddClient(client);

}

}

}

return list;

}

public static void SaveClients(List<Client> allClients)//сохранение данных всех клиентов в xml-файл

{

XmlDocument xDoc = new XmlDocument();

XmlElement xRoot;

try

{

xDoc.Load(Data.Storage + "DB/Data.xml");

xRoot = xDoc.DocumentElement;

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

foreach (XmlNode n in childnodes)

{

if (n.Name == "Clients")

{

xRoot.RemoveChild(n);

}

}

}

catch

{

xRoot = xDoc.CreateElement("Data");

xDoc.AppendChild(xRoot);

}

XmlElement clients = xDoc.CreateElement("Clients");

foreach (Client client in allClients)

{

// создаем новый элемент client

XmlElement clientElem = xDoc.CreateElement("Client");

// создаем элементы

XmlElement FIOElem = xDoc.CreateElement("FIO");

XmlElement passportElem = xDoc.CreateElement("Passport");

XmlElement licenseElem = xDoc.CreateElement("License");

XmlElement categoryElem = xDoc.CreateElement("Category");

XmlElement addressElem = xDoc.CreateElement("Address");

XmlElement savedElem = xDoc.CreateElement("Saved");

XmlElement rentsElem = xDoc.CreateElement("Rents");

// создаем текстовые значения для элементов

XmlText FIOText = xDoc.CreateTextNode(client.FIO);

XmlText passportText = xDoc.CreateTextNode(client.Passport);

XmlText licenseText = xDoc.CreateTextNode(client.License);

XmlText categoryText = xDoc.CreateTextNode(client.Category);

XmlText addressText = xDoc.CreateTextNode(client.Address);

XmlText savedText = xDoc.CreateTextNode(client.Saved.ToString());

string rents = "";

if (client.Rents != null )

{

if (client.Rents.Count != 0)

{

foreach (int num in client.Rents) { rents += num + ","; }

rents = rents.Substring(0, rents.Length - 1);

}

}

XmlText rentsText = xDoc.CreateTextNode(rents);

//добавляем узлы

FIOElem.AppendChild(FIOText);

passportElem.AppendChild(passportText);

licenseElem.AppendChild(licenseText);

categoryElem.AppendChild(categoryText);

addressElem.AppendChild(addressText);

savedElem.AppendChild(savedText);

rentsElem.AppendChild(rentsText);

clientElem.AppendChild(FIOElem);

clientElem.AppendChild(passportElem);

clientElem.AppendChild(licenseElem);

clientElem.AppendChild(categoryElem);

clientElem.AppendChild(addressElem);

clientElem.AppendChild(savedElem);

clientElem.AppendChild(rentsElem);

clients.AppendChild(clientElem);

}

xRoot.AppendChild(clients);

xDoc.Save(Data.Storage + "DB/Data.xml");

}

public static Settings ReadSettings()

{

Settings settings = new Settings();

XmlDocument docClients = new XmlDocument();

docClients.Load(Data.Storage + "DB/Settings.xml");

XmlElement xRoot = docClients.DocumentElement;

foreach (XmlNode xnode in xRoot)

{

bool dprice = false;

bool ddeposit = false;

int dperiod = 0;

if (xnode.Name == "Dprice")

{

if(xnode.InnerText == "0") { dprice = false; }

if(xnode.InnerText == "1") { dprice = true; }

settings.AddSetting(dprice);

}

if (xnode.Name == "Ddeposit")

{

if (xnode.InnerText == "0") { ddeposit = false; }

if (xnode.InnerText == "1") { ddeposit = true; }

settings.AddSetting(ddeposit);

}

if (xnode.Name == "Dperiod")

{

dperiod = int.Parse(xnode.InnerText);

settings.DefaultPeriod = dperiod;

}

if(xnode.Name == "Round")

{

settings.Round = int.Parse(xnode.InnerText);

}

}

return settings;

}

public static void SaveSettings(Settings Settings)

{

XmlDocument xDoc = new XmlDocument();

XmlElement xRoot;

try

{

xDoc.Load(Data.Storage + "DB/Settings.xml");

xRoot = xDoc.DocumentElement;

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

foreach (XmlNode n in childnodes) { xRoot.RemoveChild(n); }

}

catch

{

xRoot = xDoc.CreateElement("Settings");

xDoc.AppendChild(xRoot);

}

XmlElement users = xDoc.CreateElement("Users");

foreach (User user in Users.allUsers)

{

XmlElement userElem = xDoc.CreateElement("User");

XmlElement login = xDoc.CreateElement("login");

XmlElement password = xDoc.CreateElement("password");

XmlElement status = xDoc.CreateElement("status");

XmlText loginText = xDoc.CreateTextNode(user.Login);

XmlText passwordText = xDoc.CreateTextNode(user.Password);

XmlText statusText = xDoc.CreateTextNode(user.Status);

login.AppendChild(loginText);

password.AppendChild(passwordText);

status.AppendChild(statusText);

userElem.AppendChild(login);

userElem.AppendChild(password);

userElem.AppendChild(status);

users.AppendChild(userElem);

}

XmlElement price = xDoc.CreateElement("Dprice");

XmlElement deposit = xDoc.CreateElement("Ddeposit");

XmlElement period = xDoc.CreateElement("Dperiod");

XmlElement round = xDoc.CreateElement("Round");

XmlText priceText = xDoc.CreateTextNode("0");

if (Settings.BoolSettings[0]) { priceText = xDoc.CreateTextNode("1"); }

XmlText depositText = xDoc.CreateTextNode("0");

if (Settings.BoolSettings[1]) { depositText = xDoc.CreateTextNode("1"); }

XmlText periodText = xDoc.CreateTextNode(Settings.DefaultPeriod.ToString());

XmlText roundText = xDoc.CreateTextNode(Settings.Round.ToString());

price.AppendChild(priceText);

deposit.AppendChild(depositText);

period.AppendChild(periodText);

round.AppendChild(roundText);

xRoot.AppendChild(price);

xRoot.AppendChild(deposit);

xRoot.AppendChild(period);

xRoot.AppendChild(round);

xRoot.AppendChild(users);

xDoc.Save(Data.Storage + "DB/Settings.xml");

}

}

}

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace Lab10\_rent\_a\_car

{

public partial class MainForm : Form

{

public static StatusStrip stStrip;

public MainForm()

{

InitializeComponent();

//чтение всех данных с файла

Data.Storage = FileOperations.GetStorage();

Data.Rents = FileOperations.ReadRents();

Data.AddServices = FileOperations.ReadServices();

Data.Addresses = FileOperations.ReadAddresses();

//Data.Cars = FileOperations.ReadCars();

Data.Objects = FileOperations.ReadObjects();

Data.Clients = FileOperations.ReadClients();

Data.Settings = FileOperations.ReadSettings();

Data.Status = true;

Data.Rents.SavedRents.Reverse();

Data.Rents.AllRents.Reverse();

BuildNotifyIcon();

rentsPagesBox.SelectedItem = "50";

clientsPagesBox.SelectedItem = "50";

stStrip = statusStrip;

rentStatusBox.Text = "Все";

}

private void BuildNotifyIcon()//инициализация NotifyIcon

{

this.ShowInTaskbar = true;

notifyIcon.ContextMenuStrip = contextMenu;

}

internal void MainForm\_Load(object sender, EventArgs e)//заполнение формы

{

FillRents();

FillClients();

}

int firstRentIndex = 0;

int lastRentIndex = 0;

internal void FillRents()//запонения таблицы с арендами

{

if (rentStatusBox.Text == "Все")

{

nextRentButton.Enabled = true;

prevRentButton.Enabled = true;

rentsPagesBox.Enabled = true;

int i = 0;

RentsGrid.Rows.Clear();

foreach (Rent rent in Data.Rents.SavedRents)//перебор всех аренд для заполнения

{

if (i < int.Parse(rentsPagesBox.Text))

{

if (RentsGrid.Rows.Count < Data.Rents.SavedRents.Count) { RentsGrid.Rows.Add(); }

int j = 0;

RentsGrid.Rows[i].Cells[j].Value = rent.ID; j++;

RentsGrid.Rows[i].Cells[j].Value = rent.ClientFIO; j++;

RentsGrid.Rows[i].Cells[j].Value = rent.Name; j++;

RentsGrid.Rows[i].Cells[j].Value = rent.DateTime.ToString(); j++;

if (rent.Measure == "дней")

{

RentsGrid.Rows[i].Cells[j].Value = rent.DateTime.AddDays(rent.Period).ToString();

}

if (rent.Measure == "часов")

{

DateTime nDate = rent.DateTime.AddHours(rent.Period);

RentsGrid.Rows[i].Cells[j].Value = nDate.ToString();

}

if (rent.Measure == "месяцев")

{

RentsGrid.Rows[i].Cells[j].Value = rent.DateTime.AddMonths((int)rent.Period).ToString();

}

j++;

RentsGrid.Rows[i].Cells[j].Value = $"{rent.Period} {rent.Measure}"; j++;

RentsGrid.Rows[i].Cells[j].Value = rent.CompAddress; j++;

RentsGrid.Rows[i].Cells[j].Value = rent.Status; j++;

i++;

lastRentIndex = i;

}

}

if (!Data.Status) { this.Text = "Аренда\*"; }

else this.Text = "Аренда";

}

if (rentStatusBox.Text == "Активные" || rentStatusBox.Text == "Забронированные" || rentStatusBox.Text == "Завершенные"

|| rentStatusBox.Text == "Отмененные")

{

nextRentButton.Enabled = false;

prevRentButton.Enabled = false;

rentsPagesBox.Enabled = false;

string st = "";

if (rentStatusBox.Text == "Активные") { st = "Активная"; }

if (rentStatusBox.Text == "Забронированные") { st = "Забронирована"; }

if (rentStatusBox.Text == "Завершенные") { st = "Завершена"; }

if (rentStatusBox.Text == "Отмененные") { st = "Отменена"; }

int i = 0;

RentsGrid.Rows.Clear();

foreach (Rent rent in Data.Rents.SavedRents)//перебор всех аренд для заполнения

{

if (rent.Status == st)

{

if (RentsGrid.Rows.Count < Data.Rents.SavedRents.Count) { RentsGrid.Rows.Add(); }

int j = 0;

RentsGrid.Rows[i].Cells[j].Value = rent.ID; j++;

RentsGrid.Rows[i].Cells[j].Value = rent.ClientFIO; j++;

RentsGrid.Rows[i].Cells[j].Value = rent.Name; j++;

RentsGrid.Rows[i].Cells[j].Value = rent.DateTime.ToString(); j++;

if (rent.Measure == "дней")

{

RentsGrid.Rows[i].Cells[j].Value = rent.DateTime.AddDays(rent.Period).ToString();

}

if (rent.Measure == "часов")

{

DateTime nDate = rent.DateTime.AddHours(rent.Period);

RentsGrid.Rows[i].Cells[j].Value = nDate.ToString();

}

if (rent.Measure == "месяцев")

{

RentsGrid.Rows[i].Cells[j].Value = rent.DateTime.AddMonths((int)rent.Period).ToString();

}

j++;

RentsGrid.Rows[i].Cells[j].Value = $"{rent.Period} {rent.Measure}"; j++;

RentsGrid.Rows[i].Cells[j].Value = rent.CompAddress; j++;

RentsGrid.Rows[i].Cells[j].Value = rent.Status; j++;

i++;

lastRentIndex = rent.ID;

}

}

if (!Data.Status) { this.Text = "Аренда\*"; }

else this.Text = "Аренда";

}

//RentsGrid.Sort( rentsID, System.ComponentModel.ListSortDirection.Descending);

}

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

{

if (rentStatusBox.Text == "Все")

{

if (Data.Rents.SavedRents.Count > lastRentIndex)

{

RentsGrid.Rows.Clear();

int i = 0;

int k;

firstRentIndex = lastRentIndex - int.Parse(rentsPagesBox.Text);

for (k = lastRentIndex; k < lastRentIndex + int.Parse(rentsPagesBox.Text); k++)

{

if (k < Data.Rents.SavedRents.Count)

{

RentsGrid.Rows.Add();

int j = 0;

RentsGrid.Rows[i].Cells[j].Value = Data.Rents.SavedRents[k].ID; j++;

RentsGrid.Rows[i].Cells[j].Value = Data.Rents.SavedRents[k].ClientFIO; j++;

RentsGrid.Rows[i].Cells[j].Value = Data.Rents.SavedRents[k].Name; j++;

RentsGrid.Rows[i].Cells[j].Value = Data.Rents.SavedRents[k].DateTime.ToString(); j++;

if (Data.Rents.SavedRents[k].Measure == "дней")

{

RentsGrid.Rows[i].Cells[j].Value = Data.Rents.SavedRents[k].DateTime.AddDays(Data.Rents.SavedRents[k].Period).ToString();

}

if (Data.Rents.SavedRents[k].Measure == "часов")

{

DateTime nDate = Data.Rents.SavedRents[k].DateTime.AddHours(Data.Rents.SavedRents[k].Period);

RentsGrid.Rows[i].Cells[j].Value = nDate.ToString();

}

if (Data.Rents.SavedRents[k].Measure == "месяцев")

{

RentsGrid.Rows[i].Cells[j].Value = Data.Rents.SavedRents[k].DateTime.AddMonths((int)Data.Rents.SavedRents[k].Period).ToString();

}

j++;

RentsGrid.Rows[i].Cells[j].Value = $"{Data.Rents.SavedRents[k].Period} {Data.Rents.SavedRents[k].Measure}"; j++;

RentsGrid.Rows[i].Cells[j].Value = Data.Rents.SavedRents[k].CompAddress; j++;

RentsGrid.Rows[i].Cells[j].Value = Data.Rents.SavedRents[k].Status; j++;

i++;

}

}

lastRentIndex = k;

}

}

}

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

{

if (rentStatusBox.Text == "Все")

{

if (firstRentIndex >= 0)

{

RentsGrid.Rows.Clear();

int i = 0;

int k;

lastRentIndex = firstRentIndex + int.Parse(rentsPagesBox.Text);

for (k = firstRentIndex; k < lastRentIndex; k++)

{

if (k < Data.Rents.SavedRents.Count && Data.Rents.SavedRents[k] != null)

{

RentsGrid.Rows.Add();

int j = 0;

RentsGrid.Rows[i].Cells[j].Value = Data.Rents.SavedRents[k].ID; j++;

RentsGrid.Rows[i].Cells[j].Value = Data.Rents.SavedRents[k].ClientFIO; j++;

RentsGrid.Rows[i].Cells[j].Value = Data.Rents.SavedRents[k].Name; j++;

RentsGrid.Rows[i].Cells[j].Value = Data.Rents.SavedRents[k].DateTime.ToString(); j++;

if (Data.Rents.SavedRents[k].Measure == "дней")

{

RentsGrid.Rows[i].Cells[j].Value = Data.Rents.SavedRents[k].DateTime.AddDays(Data.Rents.SavedRents[k].Period).ToString();

}

if (Data.Rents.SavedRents[k].Measure == "часов")

{

DateTime nDate = Data.Rents.SavedRents[k].DateTime.AddHours(Data.Rents.SavedRents[k].Period);

RentsGrid.Rows[i].Cells[j].Value = nDate.ToString();

}

if (Data.Rents.SavedRents[k].Measure == "месяцев")

{

RentsGrid.Rows[i].Cells[j].Value = Data.Rents.SavedRents[k].DateTime.AddMonths((int)Data.Rents.SavedRents[k].Period).ToString();

}

j++;

RentsGrid.Rows[i].Cells[j].Value = $"{Data.Rents.SavedRents[k].Period} {Data.Rents.SavedRents[k].Measure}"; j++;

RentsGrid.Rows[i].Cells[j].Value = Data.Rents.SavedRents[k].CompAddress; j++;

RentsGrid.Rows[i].Cells[j].Value = Data.Rents.SavedRents[k].Status; j++;

i++;

}

}

firstRentIndex -= int.Parse(rentsPagesBox.Text);

}

}

}

int firstClientIndex = 0;

int lastClientIndex = 0;

internal void FillClients()//запонение таблицы клиентов

{

int i = 0;

ClientsGrid.Rows.Clear();

foreach (Client client in Data.Clients.SavedClients)

{

if (i < int.Parse(clientsPagesBox.Text))

{

ClientsGrid.Rows.Add();

int j = 0;

ClientsGrid.Rows[i].Cells[j].Value = client.FIO; j++;

ClientsGrid.Rows[i].Cells[j].Value = client.Passport; j++;

ClientsGrid.Rows[i].Cells[j].Value = client.License; j++;

ClientsGrid.Rows[i].Cells[j].Value = client.Category; j++;

ClientsGrid.Rows[i].Cells[j].Value = client.Address; j++;

i++;

lastClientIndex = i;

}

}

if (!Data.Status) { this.Text = "Аренда\*"; }

else { this.Text = "Аренда"; }

ClientsGrid.Sort( FIO, System.ComponentModel.ListSortDirection.Ascending);

}

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

{

if (firstClientIndex >= 0)

{

ClientsGrid.Rows.Clear();

int i = 0;

int k;

lastClientIndex = firstClientIndex + int.Parse(clientsPagesBox.Text);

for (k = firstClientIndex; k < lastClientIndex; k++)

{

try

{

ClientsGrid.Rows.Add();

int j = 0;

ClientsGrid.Rows[i].Cells[j].Value = Data.Clients.SavedClients[k].FIO; j++;

ClientsGrid.Rows[i].Cells[j].Value = Data.Clients.SavedClients[k].Passport; j++;

ClientsGrid.Rows[i].Cells[j].Value = Data.Clients.SavedClients[k].License; j++;

ClientsGrid.Rows[i].Cells[j].Value = Data.Clients.SavedClients[k].Category; j++;

ClientsGrid.Rows[i].Cells[j].Value = Data.Clients.SavedClients[k].Address; j++;

i++;

}

catch { ClientsGrid.Rows.RemoveAt(ClientsGrid.Rows.Count - 1); }

}

firstClientIndex -= int.Parse(clientsPagesBox.Text);

}

}

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

{

if (Data.Clients.SavedClients.Count > lastClientIndex)

{

ClientsGrid.Rows.Clear();

int i = 0;

int k;

firstClientIndex = lastClientIndex - int.Parse(clientsPagesBox.Text);

for (k = lastClientIndex; k < lastClientIndex + int.Parse(clientsPagesBox.Text); k++)

{

try

{

ClientsGrid.Rows.Add();

int j = 0;

ClientsGrid.Rows[i].Cells[j].Value = Data.Clients.SavedClients[k].FIO; j++;

ClientsGrid.Rows[i].Cells[j].Value = Data.Clients.SavedClients[k].Passport; j++;

ClientsGrid.Rows[i].Cells[j].Value = Data.Clients.SavedClients[k].License; j++;

ClientsGrid.Rows[i].Cells[j].Value = Data.Clients.SavedClients[k].Category; j++;

ClientsGrid.Rows[i].Cells[j].Value = Data.Clients.SavedClients[k].Address; j++;

i++;

}

catch { ClientsGrid.Rows.RemoveAt(ClientsGrid.Rows.Count - 1); }

}

lastClientIndex = k;

}

}

private void CarsMenuBox\_Click(object sender, EventArgs e)//открытие таблицы машин

{

if (Users.CurrentUser.Status == "admin")

{

ObjectsList carsList = new ObjectsList();

carsList.Owner = this;

carsList.ShowDialog();

}

else

MessageBox.Show("Доступ ограничен вашим статусом");

}

private void AddressesMenuBox\_Click(object sender, EventArgs e)//открытие таблицы адресов компании

{

if (Users.CurrentUser.Status == "admin")

{

AddressesList addressesList = new AddressesList();

addressesList.Owner = this;

addressesList.ShowDialog();

}

else

MessageBox.Show("Доступ ограничен вашим статусом");

}

private void ServicesMenuBox\_Click(object sender, EventArgs e)//открытие таблицы доп.сервисов

{

if(Users.CurrentUser.Status == "admin")

{

AddServicesList servicesList = new AddServicesList();

servicesList.Owner = this;

servicesList.ShowDialog();

}

else

MessageBox.Show("Доступ ограничен вашим статусом");

}

private void RentsGrid\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

string myID = "";

if (e.ColumnIndex != -1 & e.RowIndex != -1)

{

//подготовка формы аренды

myID = RentsGrid.Rows[e.RowIndex].Cells[0].Value.ToString();

Data.CurrentRent = Data.Rents.FindByID(int.Parse(myID));

RentView rentView = new RentView();

rentView.Owner = this;

rentView.ShowDialog();//открытие формы просмотра аренды

}

}

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

{

Data.CurrentRent = new Rent(Data.Rents.GetMaxID() + 1);//новая аренда

RentEdit rentEdit = new RentEdit();

rentEdit.Owner = this;

rentEdit.Show();

}

private void ClientsGrid\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

string fio = "";

if (e.ColumnIndex != -1 & e.RowIndex != -1)

{

//подготовка формы клиента

fio = ClientsGrid.Rows[e.RowIndex].Cells[0].Value.ToString();

Data.CurrentClient = Data.Clients.FindByFIO(fio);

ClientView clientView = new ClientView();

clientView.Owner = this;

clientView.ShowDialog();//открытие формы просмотра клиента

}

}

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

{

ClientEdit clientEdit = new ClientEdit();

clientEdit.Owner = this;

clientEdit.ShowDialog();

}

private void SaveToolStripMenuItem\_Click(object sender, EventArgs e)//сохранение данных

{

try

{

Data.Rents.AllRents.Reverse();

Data.Rents.SavedRents.Reverse();

FileOperations.SaveStorage(Data.Storage);

Data.Rents.SaveRents(); Data.Clients.SaveClients();

Data.Objects.SaveObjects(); Data.AddServices.SaveServices();

Data.Addresses.SaveAddresses(); Data.Settings.SaveSettings();

Data.Status = true; this.Text = "Аренда";

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Данные успешно сохранены");

Data.Rents.AllRents.Reverse();

Data.Rents.SavedRents.Reverse();

}

catch

{

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Ошибка сохранения");

}

}

private void settingsToolStripMenuItem\_Click(object sender, EventArgs e)//открытие формы настроек

{

SettingsForm settingsForm = new SettingsForm();

settingsForm.ShowDialog();

}

private void saveTimer\_Tick(object sender, EventArgs e)//таймер сохранения

{

try { SaveToolStripMenuItem\_Click(sender, e); }

catch

{

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Ошибка сохранения");

}

}

private void MainForm\_FormClosing(object sender, FormClosingEventArgs e)//проверка на сохранение

{

if (!Data.Status)

{

if (MessageBox.Show("Вы точно хотите закрыть программу без сохранения изменений?", "!!!",

MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Yes)

{

e.Cancel = false;

}

else { e.Cancel = true; }

}

this.ShowInTaskbar = false;

}

private void saveItem\_Click(object sender, EventArgs e)//сохранение через notify icon

{

SaveToolStripMenuItem\_Click(sender, e);

}

private void settingsItem\_Click(object sender, EventArgs e)//открытие настроек через notify icon

{

settingsToolStripMenuItem\_Click(sender, e);

}

private void aboutItem\_Click(object sender, EventArgs e)//о программе через notify icon

{

About about = new About();

about.ShowDialog();

}

private void exitItem\_Click(object sender, EventArgs e)//выход через notify icon

{

this.Close();

}

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

{

FillRents();

}

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

{

FillClients();

}

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

{

if (Users.CurrentUser.Status == "admin")

{

UsersList usersList = new UsersList();

usersList.Owner = this;

usersList.ShowDialog();

}

else

MessageBox.Show("Доступ ограничен вашим статусом");

}

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

{

FillRents();

}

private void MainForm\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

foreach (Form f in Application.OpenForms)

{

if (f != this)

{

f.Close();

}

}

}

}

}

}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.Text.RegularExpressions;

using System.IO;

namespace Lab10\_rent\_a\_car

{

public partial class ObjectEdit : Form

{

public ObjectEdit()//новая

{

InitializeComponent();

Data.CurrentObject = new Object();

Data.CurrentObject.ID = Data.Objects.GetMaxID() + 1;

Data.Objects.AddObject(Data.CurrentObject);

inRentBox.Text = "0";

quantityBox.Text = "0";

}

internal ObjectEdit(Object obj)//существующая

{

InitializeComponent();

FillForm(obj);

}

internal void FillForm(Object obj)//заполнение

{

nameBox.Text = obj.Name;

numBox.Text = obj.Num;

priceBox.Text = obj.Price.ToString();

inRentBox.Text = obj.InRent.ToString();

inStockBox.Value = obj.InStock;

repairBox.Value = obj.Repair;

quantityBox.Text = obj.Quantity.ToString();

minTBox.Value = obj.MinT;

maxTBox.Value = obj.MaxT;

measureBox.Text = obj.Measure;

descriptionBox.Text = obj.Description;

damageBox.Text = obj.Damage;

}

private void saveButton\_Click(object sender, EventArgs e)//сохранение

{

if (nameBox.Text == "" || numBox.Text == "" || priceBox.Text == "" ||

numBox.Text == "")

{

MessageBox.Show("Для сохранения заполните все поля формы.", "Упс!", MessageBoxButtons.OK, MessageBoxIcon.Error);

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Заполнены не все поля");

}

else

{

SaveThisForm();

Data.Status = false;

MessageBox.Show("Данные успешно изменены.", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Есть несохраненные данные");

this.Close();

ObjectView objectView = this.Owner as ObjectView;

if(objectView != null) objectView.Close();

}

}

private void SaveThisForm()//сохранение обновленной (или нет) формы

{

Data.CurrentObject.Name = nameBox.Text;

Data.CurrentObject.Num = numBox.Text;

Data.CurrentObject.Price = decimal.Parse(priceBox.Text);

Data.CurrentObject.InStock = (int)inStockBox.Value;

Data.CurrentObject.Repair = (int)repairBox.Value;

Data.CurrentObject.Quantity = int.Parse(quantityBox.Text);

Data.CurrentObject.InRent = int.Parse(inRentBox.Text);

Data.CurrentObject.MinT = (int)minTBox.Value;

Data.CurrentObject.MaxT = (int)maxTBox.Value;

Data.CurrentObject.Measure = measureBox.Text;

Data.CurrentObject.Description = descriptionBox.Text;

Data.CurrentObject.Damage = damageBox.Text;

Data.CurrentObject.Deleted = false;

Data.CurrentObject.Rents = new List<int>();

LoadCarPic();

Data.Objects.ReplaceInfoObject(Data.CurrentObject);

}

private void LoadCarPic()

{

try

{

if (File.Exists(Data.Storage + $"DB/Pics/{Data.CurrentObject.Name}.jpg")) { File.Delete(Data.Storage + $"DB/Pics/{Data.CurrentObject.Name}.jpg"); }

if (File.Exists(Data.PicPath))

{

File.Move(Data.PicPath, Data.Storage + $"DB/Pics/{Data.CurrentObject.Name}.jpg");

}

}

catch { }

}

private void priceBox\_KeyPress(object sender, KeyPressEventArgs e)//ввод только цифр

{

if (Char.IsDigit(e.KeyChar) || Char.IsControl(e.KeyChar))

return;

else

e.Handled = true;

}

private void CarEdit\_FormClosed(object sender, FormClosedEventArgs e)//изменение формы родителя

{

try

{

ObjectView carView = this.Owner as ObjectView;

if (nameBox.Text == "" || numBox.Text == "" || priceBox.Text == "")

{

Data.Objects.DelObject(Data.CurrentObject);

carView.Close();

}

carView.FillForm(Data.CurrentObject);

}

catch

{

ObjectsList carsList = this.Owner as ObjectsList;

carsList.FillGrid();

}

}

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

{

openFileDialog.Filter = "png files (\*.png)|\*.png|jpeg files (\*.jpeg)|\*.jpeg|jpg files (\*jpg)|\*.jpg";

openFileDialog.ShowDialog();

Data.PicPath = openFileDialog.FileName;

MessageBox.Show("Фото успешно добавлено.");

}

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

{

try { quantityBox.Text = (inStockBox.Value + repairBox.Value + int.Parse(inRentBox.Text)).ToString(); }

catch { }

}

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

{

try { quantityBox.Text = (inStockBox.Value + repairBox.Value + int.Parse(inRentBox.Text)).ToString(); }

catch { }

}

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

{

if (measureBox.Text == "дней")

{

minTBox.Maximum = 60;

maxTBox.Maximum = 60;

measureLabel.Text = "в день";

}

if (measureBox.Text == "часов")

{

minTBox.Maximum = 48;

maxTBox.Maximum = 48;

measureLabel.Text = "в час";

}

if (measureBox.Text == "месяцев")

{

minTBox.Maximum = 12;

maxTBox.Maximum = 12;

measureLabel.Text = "в месяц";

}

}

private void ObjectEdit\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Close();

try

{

ObjectView objView = this.Owner as ObjectView;

ObjectsList objList = objView.Owner as ObjectsList;

objView.Close();

objList.Close();

}

catch

{

}

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab10\_rent\_a\_car

{

class Objects

{

List<Object> allObjects;

public Objects()

{

allObjects = new List<Object>();

}

public void AddObject(Object obj) { allObjects.Add(obj); }

public List<Object> AllObjects { get { return allObjects; } }

public void SaveObjects()//сохранение данных всех клиентов в xml-файл

{

FileOperations.SaveObjects(allObjects);

}

public void DelObject(Object obj) { allObjects.Remove(obj); }

public void ReplaceInfoObject(Object obj)//метод для обновления информации

{

int index = allObjects.FindIndex(x => x.ID == obj.ID);

allObjects[index] = obj;

}

public int GetMaxID()//получение максимального значения id

{

int max = 0;

foreach (Object obj in allObjects)

{

if (max < obj.ID) max = obj.ID;

}

return max;

}

public Object FindByName(string name)//метод для нахождения клиента по id

{

return allObjects.Find(x => x.Name == name);

}

}

class Object

{

private int id; private string name;

private string num; private decimal price;

private decimal mdeposit;

private bool deleted; private int quantity; private int inRent;

private int inStock; private int repair;

private int minT; private int maxT;

private string description; private string damage; private string measure;

private List<int> rents;

public Object() { }

public Object(int id, string name, string num, decimal price, decimal mdeposit,

bool deleted, int quantity, int inRent, int inStock, int repair,

int minT, int maxT, string description, string damage, string measure, List<int> rents)

{

this.id = id;this.name = name;this.num = num;

this.price = price; this.mdeposit = mdeposit;

this.deleted = deleted; this.quantity = quantity; this.inRent = inRent;

this.inStock = inStock; this.repair = repair; this.minT = minT; this.maxT = maxT;

this.description = description; this.damage = damage; this.measure = measure;

this.rents = rents;

}

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

public string Name { get { return name; } set { name = value; } }

public string Num { get { return num; } set { num = value; } }

public decimal Price { get { return price; } set { price = value; } }

public decimal Mdeposit { get {return mdeposit; } set { mdeposit = value; } }

public bool Deleted { get { return deleted; } set { deleted = value; } }

public int Quantity { get { return quantity; } set {quantity = value; } }

public int InRent { get { return inRent; } set {inRent = value; } }

public int InStock { get { return inStock; } set { inStock = value; } }

public int Repair { get { return repair; } set { repair = value; } }

public int MinT { get { return minT; } set { minT = value; } }

public int MaxT { get { return maxT; } set { maxT = value; } }

public string Description { get { return description; } set { description = value; } }

public string Damage { get { return damage; } set { damage = value; } }

public string Measure { get { return measure; } set { measure = value; } }

public List<int> Rents { get { return rents; }set { rents = value; } }

}

}

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;

namespace Lab10\_rent\_a\_car

{

public partial class ObjectsList : Form

{

public ObjectsList()

{

InitializeComponent();

FillGrid();

}

internal void FillGrid()

{

int i = 0;

CarsGrid.Rows.Clear();

foreach (Object obj in Data.Objects.AllObjects)

{

if (!obj.Deleted)

{

CarsGrid.Rows.Add();

int j = 0;

CarsGrid.Rows[i].Cells[j].Value = obj.Name; j++;

CarsGrid.Rows[i].Cells[j].Value = obj.Num; j++;

CarsGrid.Rows[i].Cells[j].Value = obj.Price; j++;

CarsGrid.Rows[i].Cells[j].Value = obj.Quantity; j++;

CarsGrid.Rows[i].Cells[j].Value = obj.InRent; j++;

CarsGrid.Rows[i].Cells[j].Value = obj.InStock; j++;

CarsGrid.Rows[i].Cells[j].Value = obj.Repair; j++;

i++;

}

}

}

private void CarsGrid\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

string name = "";

if (e.ColumnIndex != -1 & e.RowIndex != -1)

{

name = CarsGrid.Rows[e.RowIndex].Cells[0].Value.ToString();

Data.CurrentObject = Data.Objects.FindByName(name);

ObjectView objView = new ObjectView();

objView.Owner = this;

objView.ShowDialog();

}

}

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

{

ObjectEdit objEdit = new ObjectEdit();

objEdit.Owner = this;

objEdit.ShowDialog();

}

private void CarsList\_FormClosed(object sender, FormClosedEventArgs e)

{

MainForm mainForm = this.Owner as MainForm;

if(!Data.Status) { mainForm.Text = "Аренда\*"; }

}

private void ObjectsList\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Close();

}

}

}

}

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;

namespace Lab10\_rent\_a\_car

{

public partial class ObjectView : Form

{

public ObjectView()

{

InitializeComponent();

FillForm(Data.CurrentObject);

}

internal void FillForm(Object obj)//заполнение

{

nameBox.Text = obj.Name;

numBox.Text = obj.Num;

priceBox.Text = obj.Price.ToString();

quantityBox.Text = obj.Quantity.ToString();

inRentBox.Text = obj.InRent.ToString();

inStockBox.Text = obj.InStock.ToString();

repairBox.Text = obj.Repair.ToString();

minTBox.Text = obj.MinT.ToString();

maxTBox.Text = obj.MaxT.ToString();

descriptionBox.Text = obj.Description;

damageBox.Text = obj.Damage;

measureLabel1.Text = obj.Measure;

measureLabel2.Text = obj.Measure;

try { carPic.Image = System.Drawing.Image.FromFile($"DB/Pics/{nameBox.Text}.jpg"); }

catch { }

if (obj.Rents != null)

{

rentsListBox.Items.Clear();

foreach (int rentID in obj.Rents)

{

rentsListBox.Items.Add($"Договор №{rentID}");

}

}

else { rentsListBox.Items.Clear(); }

}

private void editButton\_Click(object sender, EventArgs e)//открытие формы изменения

{

ObjectEdit objectEdit = new ObjectEdit(Data.CurrentObject);

objectEdit.Owner = this;

objectEdit.ShowDialog();

}

private void deleteButton\_Click(object sender, EventArgs e)//удаление машины

{

if (MessageBox.Show("Вы точно хотите удалить данную машину?", "!!!", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

Data.Status = false;

Data.CurrentObject.Deleted = true;

Data.Objects.ReplaceInfoObject(Data.CurrentObject);

Data.CurrentObject = null;

this.Close();

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Есть несохраненные данные");

}

}

private void CarView\_FormClosed(object sender, FormClosedEventArgs e)//изменения в форму родителя

{

ObjectsList objectsList = this.Owner as ObjectsList;

objectsList.FillGrid();

carPic.Image.Dispose();

}

private void rentsListBox\_SelectedIndexChanged(object sender, EventArgs e)//открытие соответсвующей формы аренды

{

if(rentsListBox.SelectedItem != null)

{

string myID = "";

//подготовка формы аренды

myID = rentsListBox.SelectedItem.ToString().Substring(9);

Data.CurrentRent = Data.Rents.FindByID(int.Parse(myID));

MainForm mainForm = this.Owner.Owner as MainForm;

RentView rentView = new RentView();

if (!Data.CurrentRent.Saved) { rentView.statusBox.Text = "Удалена";

rentView.editButton.Enabled = false;

rentView.deleteButton.Enabled = false;

}

rentView.Owner = mainForm;

rentView.ShowDialog();//открытие формы просмотра аренды

}

}

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

{

RentsList rentList = new RentsList();

rentList.Owner = this;

rentList.ShowDialog();

}

private void ObjectView\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Close();

try

{

ObjectsList objList = this.Owner as ObjectsList;

objList.Close();

}

catch

{

}

}

}

}

}

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;

namespace Lab10\_rent\_a\_car

{

public partial class RentEdit : Form

{

bool svd;

int currentId;

string inStatus;

Rent srent;

internal RentEdit()//конструктор для формы новой аренды

{

InitializeComponent();

currentId = Data.CurrentRent.ID;

//Data.Rents.AllRents.Insert(0, Data.CurrentRent);

Data.Rents.AllRents.Reverse();

Data.Rents.SavedRents.Reverse();

Data.Rents.AddRent(Data.CurrentRent);

//Data.Rents.AllRents.Reverse();

this.Text = $"Аренда №{currentId}";

todayButton.Visible = true;

labelInStock.Visible = true;

inStockBox.Visible = true;

inStockBox.Text = "0";

svd = true;

docB.Checked = true;

foreach (AddService add in Data.AddServices.AllServices)

{

if (!add.Deleted) { addServicesListBox.Items.Add(add.Name + "(" + add.Price + " руб)"); }

}

datePicker1.Value = DateTime.Now;

hoursBox.Value = DateTime.Now.Hour;

minutesBox.Value = DateTime.Now.Minute;

}

internal RentEdit(Rent rent)//конструктор для открытия формы уже существующей аренды

{

InitializeComponent();

this.currentId = rent.ID;

this.Text = $"Аренда №{currentId}";

todayButton.Visible = false;

labelInStock.Visible = false;

inStockBox.Visible = false;

svd = true;

BuildAddServicesListBox();//заполнение списка доп.услуг

srent = Data.CurrentRent;

Data.Rents.AllRents.Reverse();

Data.Rents.SavedRents.Reverse();

}

private void RentEdit\_Load(object sender, EventArgs e)//загрузка формы

{

BuildCarsBox();//заполнение списка машин

BuildFIOBox();//заполнение списка фио

BuildAddressesBox();//заполнение списка адресов

if (Data.CurrentRent.ClientFIO != null)//заполнение формы существующего клиента

{

FillForm(Data.CurrentRent);

}

else//при пустом списке клиентов

{

Rent newRent = new Rent(1);

numericUpDown.Text = Data.Settings.DefaultPeriod.ToString();//применение настройки

}

Calculator(sender, e);

}

private void FillForm(Rent rent)//заполнение формы аренды

{

clientFIO.Text = rent.ClientFIO;

objectBox.Text = rent.Name;

if(objectBox.Text == null || objectBox.Text == "") { objectBox.Items.Add(rent.Name);objectBox.Text = rent.Name; }

priceBox.Text = rent.Price.ToString();

if (rent.DepositType == "doc") { docB.Checked = true; depositBox.Text = rent.Deposit; }

if (rent.DepositType == "money") { moneyB.Checked = true; depositBox.Text = rent.DepositMoney.ToString(); }

if (rent.DepositType == "product") { productB.Checked = true; depositBox.Text = rent.Deposit; }

if (!addressesBox.Items.Contains(rent.CompAddress)) { addressesBox.Items.Add(rent.CompAddress); }

addressesBox.Text = rent.CompAddress;

secondAddressBox.Text = rent.SecondAddress;

statusBox.Text = rent.Status;

inStatus = rent.Status;

datePicker1.Value = rent.DateTime;

hoursBox.Value = rent.DateTime.Hour;

minutesBox.Value = rent.DateTime.Minute;

DateTime nDate = new DateTime();

if (rent.Measure == "дней")

{

nDate = rent.DateTime.AddDays(rent.Period);

objectMeasureLabel.Text = "в день";

if (Data.Settings.BoolSettings[1]) numericUpDown.Maximum = 60;

}

if (rent.Measure == "часов")

{

nDate = rent.DateTime.AddHours(rent.Period);

objectMeasureLabel.Text = "в час";

if (Data.Settings.BoolSettings[1]) numericUpDown.Maximum = 48;

}

if (rent.Measure == "месяцев")

{

nDate = rent.DateTime.AddMonths((int)rent.Period);

objectMeasureLabel.Text = "в месяц";

if (Data.Settings.BoolSettings[1]) numericUpDown.Maximum = 12;

}

datePicker2.Value = nDate;

timePicker2.Value = nDate;

measureBox.Text = rent.Measure;

numericUpDown.Value = (decimal)rent.Period;

descriptionBox.Text = rent.Description;

for (int i = 0; i < addServicesListBox.Items.Count; i++)//простановка галочек в листбоксе

{

if (rent.AddServices.Contains(i)) { addServicesListBox.SetItemChecked(i, true); }

else { addServicesListBox.SetItemChecked(i, false); }

}

}

private void BuildCarsBox()//заполнение списка машин

{

foreach (Object obj in Data.Objects.AllObjects)

{

if (!obj.Deleted)

{

objectBox.Items.Add(obj.Name);

}

}

}

private void BuildFIOBox()//заполнение списка фио

{

foreach (Client client in Data.Clients.AllClients)

{

clientFIO.Items.Add(client.FIO);

}

}

private void BuildAddressesBox()//заполнение списка адресов

{

foreach (Address elem in Data.Addresses.AllAddresses)

{

if (elem.Name != null && !elem.Deleted)

addressesBox.Items.Add(elem.Name);

}

}

private void BuildAddServicesListBox()//заполнение списка листбокса доп.услуг

{

MainForm main = this.Owner as MainForm;

foreach (AddService add in Data.AddServices.AllServices)

{

addServicesListBox.Items.Add(add.Name + "(" + add.Price + " руб)");

}

}

private void SaveThisForm()//сохранение обновленной (или нет) формы

{

if (Data.CurrentRent.ClientFIO != clientFIO.Text && Data.CurrentRent.ClientFIO != null)

{ Data.Clients.FindByFIO(clientFIO.Text).Rents.Remove(Data.CurrentRent.ID); }

try

{

if (Data.CurrentRent.ClientFIO == null)

{

Data.Clients.FindByFIO(clientFIO.Text).Rents.Add(Data.CurrentRent.ID);

}

}catch { }

Data.CurrentRent.ClientFIO = clientFIO.Text;

if (Data.CurrentRent.Name != objectBox.Text && Data.CurrentRent.Name != null)

{ Data.Objects.FindByName(objectBox.Text).Rents.Remove(Data.CurrentRent.ID); }

if (Data.CurrentRent.Name == null)

{

if (statusBox.Text == "Активная" || statusBox.Text == "Забронирована")

{

Data.Objects.FindByName(objectBox.Text).Rents.Add(Data.CurrentRent.ID);

Data.Objects.FindByName(objectBox.Text).InRent += 1;

Data.Objects.FindByName(objectBox.Text).InStock -= 1;

}

}

Data.CurrentRent.Name = objectBox.Text;

Data.CurrentRent.Price = decimal.Parse(priceBox.Text);

if (docB.Checked)

{

Data.CurrentRent.DepositType = "doc";

Data.CurrentRent.Deposit = depositBox.Text;

}

if (moneyB.Checked)

{

Data.CurrentRent.DepositType = "money";

Data.CurrentRent.DepositMoney = decimal.Parse(depositBox.Text);

}

if (productB.Checked)

{

Data.CurrentRent.DepositType = "product";

Data.CurrentRent.Deposit = depositBox.Text;

}

Data.CurrentRent.CompAddress = addressesBox.Text;

Data.CurrentRent.SecondAddress = secondAddressBox.Text;

Data.CurrentRent.Status = statusBox.Text;

if(inStatus != statusBox.Text)

{

if(inStatus == "Активная" && statusBox.Text != "Забронирована")

{

Data.Objects.FindByName(objectBox.Text).InRent--;

Data.Objects.FindByName(objectBox.Text).InStock++;

}

if ((inStatus == "Завершена" || inStatus == "Отменена") && (statusBox.Text != "Завершена" || statusBox.Text != "Отменена"))

{

Data.Objects.FindByName(objectBox.Text).InRent++;

Data.Objects.FindByName(objectBox.Text).InStock--;

}

}

datePicker1.Value = datePicker1.Value.AddHours(-datePicker1.Value.Hour).AddMinutes(-datePicker1.Value.Minute);

datePicker1.Value = datePicker1.Value.AddHours((double)hoursBox.Value).AddMinutes((double)minutesBox.Value);

Data.CurrentRent.DateTime = datePicker1.Value;

Data.CurrentRent.Period = (double)numericUpDown.Value;

Data.CurrentRent.Measure = measureBox.Text;

Data.CurrentRent.Description = descriptionBox.Text;

Data.CurrentRent.Saved = true;

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

Data.CurrentRent.AddServices = list;

foreach (string elem in addServicesListBox.CheckedItems)

{

AddService ads;

try

{

ads = Data.AddServices.FindByName(elem.Remove(elem.IndexOf('(')));

Data.CurrentRent.AddServices.Add(ads.ID);

}

catch

{

}

}

if (Data.Rents.SavedRents.FindIndex(x => x.ID == Data.CurrentRent.ID) == -1)

{ Data.Rents.SavedRents.Add(Data.CurrentRent); }

Data.Rents.ReplaceInfoRent(Data.CurrentRent);

//if (Data.Clients.FindByFIO(clientFIO.Text).Rents != null && !Data.Clients.FindByFIO(clientFIO.Text).Rents.Contains(Data.CurrentRent.ID))

//Data.Clients.FindByFIO(clientFIO.Text).Rents.Add(Data.CurrentRent.ID);

//if (Data.Cars.findByModel(objectBox.Text).Rents != null && !Data.Cars.findByModel(objectBox.Text).Rents.Contains(Data.CurrentRent.ID))

//Data.Cars.findByModel(objectBox.Text).Rents.Add(Data.CurrentRent.ID);

}

private void saveButton\_Click(object sender, EventArgs e)//сохранение в файл

{

if (clientFIO.Text == "" || objectBox.Text == "" || priceBox.Text == "" ||

depositBox.Text == "" || addressesBox.Text == "" || secondAddressBox.Text == ""||

statusBox.Text == "" || measureBox.Text == "" )

{

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Заполнены не все поля");

MessageBox.Show("Для сохранения заполните все поля формы.", "Упс!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else if(Data.Objects.FindByName(objectBox.Text).InStock == 0 && Data.CurrentRent.ClientFIO ==null)

{

MessageBox.Show("На складе не осталось данного товара.", "Упс!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else if (Data.Objects.FindByName(objectBox.Text).InStock == 0 && srent.Name != Data.Objects.FindByName(objectBox.Text).Name)

{

MessageBox.Show("На складе не осталось данного товара.", "Упс!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

SaveThisForm();

svd = false;

Data.Status = false;

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Есть несохраненные данные");

MessageBox.Show("Данные успешно изменены.", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

RentView rentView = this.Owner as RentView;

if (rentView != null) rentView.Close();

}

}

private void Calculator(object sender, EventArgs e)

{

List<AddService> ads = Data.AddServices.AllServices;

decimal p = 0m;

foreach (string elem in addServicesListBox.CheckedItems)

{

AddService adds;

try

{

adds = Data.AddServices.FindByName(elem.Remove(elem.IndexOf('(')));

p += adds.Price;

}

catch

{

}

}

try

{

if (Data.Settings.Round == 0)

{

cost.Text = ((decimal.Parse(priceBox.Text) \* (Math.Ceiling(numericUpDown.Value))) + p).ToString() + " рублей";

}

if (Data.Settings.Round == 1)

{

cost.Text = ((decimal.Parse(priceBox.Text) \* (Math.Round(numericUpDown.Value))) + p).ToString() + " рублей";

}

if (Data.Settings.Round == 2)

{

cost.Text = ((decimal.Parse(priceBox.Text) \* (Math.Floor(numericUpDown.Value))) + p).ToString() + " рублей";

}

if (Data.Settings.Round == 3)

{

cost.Text = ((decimal.Parse(priceBox.Text) \* numericUpDown.Value) + p).ToString() + " рублей";

}

}

catch { cost.Text = "0 рублей"; }

}

private void CarPic(object sender, EventArgs e)// + рекомендованная цена

{

string name = objectBox.Text;

try { pictureBox1.Image = System.Drawing.Image.FromFile(Data.Storage + $"DB/Pics/{name}.jpg"); }

catch { }

try

{

if (Data.Settings.BoolSettings[0])

{

priceBox.Text = Data.Objects.FindByName(name).Price.ToString();

measureBox.Text = Data.Objects.FindByName(name).Measure;

numericUpDown.Value = Data.Objects.FindByName(name).MinT;

}

else

{

numericUpDown.Value = 1;

}

inStockBox.Text = Data.Objects.FindByName(name).InStock.ToString();

}

catch { }

}

private void pricesBox\_KeyPress(object sender, KeyPressEventArgs e)//вводить только числа

{

if (Char.IsDigit(e.KeyChar) || Char.IsControl(e.KeyChar))

return;

else

e.Handled = true;

}

private void periodsBox\_KeyPress(object sender, KeyPressEventArgs e)//вводить только числа

{

pricesBox\_KeyPress(sender, e);

}

private void BoundPrice(object sender, EventArgs e)//контроль ввода цены

{

try

{

string name = objectBox.GetItemText(objectBox.SelectedItem);

decimal recomP = Data.Objects.FindByName(name).Price / 2;

decimal pr;

try { pr = decimal.Parse(priceBox.Text); } catch { pr = 0; }

if (pr < recomP)

{

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Введены неверные данные");

MessageBox.Show("Вы ввели слишком низкую цену!", "Ошибка!!!", MessageBoxButtons.OK, MessageBoxIcon.Stop);

priceBox.Text = Data.Objects.FindByName(name).Price.ToString();

}

}

catch { }

}

private void RentEdit\_FormClosed(object sender, FormClosedEventArgs e)//изменения в форму родителя

{

pictureBox1.Image.Dispose();

Data.Rents.AllRents.Reverse();

Data.Rents.SavedRents.Reverse();

try

{

RentView rentView = this.Owner as RentView;

if (srent != null && srent.Equals(Data.CurrentRent))

{

svd = false;

}

if (clientFIO.Text == "" || objectBox.Text == "" || priceBox.Text == "" ||

depositBox.Text == "" || addressesBox.Text == "" || svd)

{

Data.Rents.DelRent(Data.CurrentRent);

}

else

{

rentView.FillForm(Data.CurrentRent);

}

}

catch

{

MainForm mainForm = this.Owner as MainForm;

mainForm.FillRents();

}

srent = null;

svd = true;

}

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

{

if (measureBox.Text == "часов")

{

objectMeasureLabel.Text = "в час";

DateTime nDate = new DateTime();

nDate = datePicker1.Value.AddHours((double)numericUpDown.Value);

datePicker2.Value = nDate;

timePicker2.Value = nDate;

if (Data.Settings.BoolSettings[1]) numericUpDown.Maximum = 48;

}

if (measureBox.Text == "дней")

{

objectMeasureLabel.Text = "в день";

DateTime nDate = new DateTime();

nDate = datePicker1.Value.AddDays((double)numericUpDown.Value);

datePicker2.Value = nDate;

timePicker2.Value = nDate;

if (Data.Settings.BoolSettings[1]) numericUpDown.Maximum = 60;

}

if (measureBox.Text == "месяцев")

{

objectMeasureLabel.Text = "в месяц";

DateTime nDate = new DateTime();

nDate = datePicker1.Value.AddMonths((int)numericUpDown.Value);

datePicker2.Value = nDate;

timePicker2.Value = nDate;

if (Data.Settings.BoolSettings[1]) numericUpDown.Maximum = 12;

}

Calculator(sender, e);

}

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

{

if (secondAddressBox.Text == "" || addressesBox.Items.Contains(secondAddressBox.Text))

{

secondAddressBox.Text = addressesBox.Text;

}

}

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

{

if (measureBox.Text == "часов")

{

DateTime nDate = new DateTime();

nDate = datePicker1.Value.AddHours((double)numericUpDown.Value);

datePicker2.Value = nDate;

timePicker2.Value = nDate;

}

if (measureBox.Text == "дней")

{

DateTime nDate = new DateTime();

nDate = datePicker1.Value.AddDays((double)numericUpDown.Value);

datePicker2.Value = nDate;

timePicker2.Value = nDate;

}

if (measureBox.Text == "месяцев")

{

DateTime nDate = new DateTime();

nDate = datePicker1.Value.AddMonths((int)numericUpDown.Value);

datePicker2.Value = nDate;

timePicker2.Value = nDate;

}

if(numericUpDown.Value > numericUpDown.Maximum)

{

MessageBox.Show("Значение превышает максимальный срок аренды.");

}

if (numericUpDown.Value < numericUpDown.Minimum)

{

MessageBox.Show("Значение меньше минимального срока аренды.");

}

try

{

if (numericUpDown.Value > Data.Objects.FindByName(objectBox.Text).MaxT)

{

MessageBox.Show("Значение превышает максимальный срок аренды.");

numericUpDown.Value = Data.Objects.FindByName(objectBox.Text).MaxT;

}

if (numericUpDown.Value < Data.Objects.FindByName(objectBox.Text).MinT)

{

MessageBox.Show("Значение меньше минимального срока аренды.");

numericUpDown.Value = Data.Objects.FindByName(objectBox.Text).MinT;

}

}

catch { }

Calculator(sender, e);

}

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

{

datePicker1.Value = DateTime.Now;

hoursBox.Value = DateTime.Now.Hour;

minutesBox.Value = DateTime.Now.Minute;

}

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

{

if (measureBox.Text == "часов")

{

DateTime nDate = new DateTime();

nDate = datePicker1.Value.AddHours((double)numericUpDown.Value);

datePicker2.Value = nDate;

timePicker2.Value = nDate;

datePicker1.Value = datePicker1.Value.AddHours(-datePicker1.Value.Hour).AddMinutes(-datePicker1.Value.Minute);

datePicker1.Value = datePicker1.Value.AddHours((double)hoursBox.Value).AddMinutes((double)minutesBox.Value);

}

if (measureBox.Text == "дней")

{

DateTime nDate = new DateTime();

nDate = datePicker1.Value.AddDays((double)numericUpDown.Value);

datePicker2.Value = nDate;

timePicker2.Value = nDate;

datePicker1.Value = datePicker1.Value.AddHours(-datePicker1.Value.Hour).AddMinutes(-datePicker1.Value.Minute);

datePicker1.Value = datePicker1.Value.AddHours((double)hoursBox.Value).AddMinutes((double)minutesBox.Value);

}

if (measureBox.Text == "месяцев")

{

DateTime nDate = new DateTime();

nDate = datePicker1.Value.AddMonths((int)numericUpDown.Value);

datePicker2.Value = nDate;

timePicker2.Value = nDate;

datePicker1.Value = datePicker1.Value.AddHours(-datePicker1.Value.Hour).AddMinutes(-datePicker1.Value.Minute);

datePicker1.Value = datePicker1.Value.AddHours((double)hoursBox.Value).AddMinutes((double)minutesBox.Value);

}

}

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

{

if (measureBox.Text == "часов")

{

DateTime nDate = new DateTime();

nDate = datePicker1.Value.AddHours((double)numericUpDown.Value);

datePicker2.Value = nDate;

timePicker2.Value = nDate;

datePicker1.Value = datePicker1.Value.AddHours(-datePicker1.Value.Hour).AddMinutes(-datePicker1.Value.Minute);

datePicker1.Value = datePicker1.Value.AddHours((double)hoursBox.Value).AddMinutes((double)minutesBox.Value);

}

if (measureBox.Text == "дней")

{

DateTime nDate = new DateTime();

nDate = datePicker1.Value.AddDays((double)numericUpDown.Value);

datePicker2.Value = nDate;

timePicker2.Value = nDate;

datePicker1.Value = datePicker1.Value.AddHours(-datePicker1.Value.Hour).AddMinutes(-datePicker1.Value.Minute);

datePicker1.Value = datePicker1.Value.AddHours((double)hoursBox.Value).AddMinutes((double)minutesBox.Value);

}

if (measureBox.Text == "месяцев")

{

DateTime nDate = new DateTime();

nDate = datePicker1.Value.AddMonths((int)numericUpDown.Value);

datePicker2.Value = nDate;

timePicker2.Value = nDate;

datePicker1.Value = datePicker1.Value.AddHours(-datePicker1.Value.Hour).AddMinutes(-datePicker1.Value.Minute);

datePicker1.Value = datePicker1.Value.AddHours((double)hoursBox.Value).AddMinutes((double)minutesBox.Value);

}

}

private void datePicker1\_ValueChanged(object sender, EventArgs e)

{

if (measureBox.Text == "часов")

{

DateTime nDate = new DateTime();

nDate = datePicker1.Value.AddHours((double)numericUpDown.Value);

datePicker2.Value = nDate;

timePicker2.Value = nDate;

}

if (measureBox.Text == "дней")

{

DateTime nDate = new DateTime();

nDate = datePicker1.Value.AddDays((double)numericUpDown.Value);

datePicker2.Value = nDate;

timePicker2.Value = nDate;

}

if (measureBox.Text == "месяцев")

{

DateTime nDate = new DateTime();

nDate = datePicker1.Value.AddMonths((int)numericUpDown.Value);

datePicker2.Value = nDate;

timePicker2.Value = nDate;

}

}

private void RentEdit\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Close();

try

{

RentView rentView = this.Owner as RentView;

rentView.Close();

}

catch

{

}

try

{

RentView rentView = this.Owner as RentView;

RentsList rList = rentView.Owner as RentsList;

ObjectView objectView = rList.Owner as ObjectView;

ObjectsList objList = objectView.Owner as ObjectsList;

rentView.Close();

rList.Close();

objectView.Close();

objList.Close();

}

catch

{

}

}

}

private void numericUpDown\_VisibleChanged(object sender, EventArgs e)

{

if (numericUpDown.Value > numericUpDown.Maximum)

{

MessageBox.Show("Значение превышает максимальный срок аренды.");

}

if (numericUpDown.Value < numericUpDown.Minimum)

{

MessageBox.Show("Значение меньше минимального срока аренды.");

}

}

private void moneyB\_CheckedChanged(object sender, EventArgs e)

{

if (moneyB.Checked)

{

depositBox.KeyPress += pricesBox\_KeyPress;

}

else

{

depositBox.KeyPress -= pricesBox\_KeyPress;

}

}

}

}using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab10\_rent\_a\_car

{

class Rents

{

List<Rent> allRents;

List<Rent> savedRents;

public int Count { get; set; }

public Rents()

{

allRents = new List<Rent>();

savedRents = new List<Rent>();

}

public List<Rent> AllRents { get { return allRents; } }

public List<Rent> SavedRents { get { return savedRents; } }

public Rent FindByID(int id)//метод для нахождения клиента по id

{

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

}

public void SaveRents()//сохранение данных всех клиентов в xml-файл

{

FileOperations.SaveRents(allRents);

}

public void AddRent(Rent rent)

{

allRents.Add(rent);

if(rent.Saved) { Count++; savedRents.Add(rent); }

}//метод добавления клиента в список

public void DelRent(Rent rent)

{

allRents.Remove(rent);

if (rent.Saved) { Count--; savedRents.Remove(rent); }

}

public void ReplaceInfoRent(Rent rent)//метод для обновления информации

{

int index = allRents.FindIndex(x => x.ID == rent.ID);

allRents[index] = rent;

if (rent.Saved) {

int ind = savedRents.FindIndex(x => x.ID == rent.ID);

if (ind<savedRents.Count ||ind == -1) { savedRents[ind] = rent; } }

}

public int GetMaxID()//получение максимального значения id

{

int max = 0;

foreach (Rent rent in allRents)

{

if (max < rent.ID) max = rent.ID;

}

return max;

}

}

class Rent

{

private int id;

private string clientFIO;

private string name;

private decimal price;

private string depositType;

private decimal depositMoney;

private string deposit;

private double period;

private string measure;

private string compAddress;

private string secondAddress;

private bool saved;

private DateTime dateTime;

private string status;

private string description;

private List<int> addServices = new List<int>();

public Rent (int id)

{

this.id = id;

}

public Rent (int id, string clientFIO, string name, decimal price, string depositType,

decimal depositMoney, string deposit , double period, string measure, string compAddress,

string secondAddress, List<int> addservices, bool saved, DateTime dateTime, string status, string description)

{

this.id = id;

this.clientFIO = clientFIO;

this.name = name;

this.price = price;

this.depositType = depositType;

this.depositMoney = depositMoney;

this.deposit = deposit;

this.period = period;

this.measure = measure;

this.compAddress = compAddress;

this.secondAddress = secondAddress;

this.addServices = addservices;

this.saved = saved;

this.dateTime = dateTime;

this.status = status;

this.description = description;

}

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

public string ClientFIO { get { return clientFIO; } set { clientFIO = value; } }

public string Name { get { return name; } set { name = value; } }

public double Period { get { return period; } set { period = value; } }

public string Measure { get { return measure; } set { measure = value; } }

public decimal Price { get { return price; } set { price = value; } }

public string DepositType { get { return depositType; } set { depositType = value; } }

public string Deposit { get { return deposit; } set { deposit = value; } }

public decimal DepositMoney { get { return depositMoney; } set { depositMoney = value; } }

public string CompAddress { get { return compAddress; } set { compAddress = value; } }

public string SecondAddress { get { return secondAddress; } set { secondAddress = value; } }

public List<int> AddServices { get { return addServices; } set { addServices = value; } }

public bool Saved { get { return saved; } set { saved = value; } }

public DateTime DateTime { get { return dateTime; } set { dateTime = value; } }

public string Status { get { return status; } set { status = value; } }

public string Description { get { return description; } set { description = value; } }

public string AddServicesString

{

get

{

string str = "";

try

{

foreach (int elem in addServices)

{

str += elem + ",";

}

str = str.Substring(0, str.Length - 1);

}

catch { }

return str;

}

}

public string GetTotalPrice(AddServices addServices)

{

string cost = "";

decimal p = 0m;

foreach (int elem in this.addServices)

{

try

{

AddService adds = addServices.FindByID(elem);

p += adds.Price;

}

catch

{

}

}

cost = (this.Price \* decimal.Parse(this.Period.ToString()) + p).ToString() + " рублей";

return cost;

}

}

class Addresses//класс в котором содержатся адреса компании

{

List<Address> allAddresses;

public Addresses()

{

allAddresses = new List<Address>();

}

public void AddAddress(Address address) { allAddresses.Add(address); }

public List <Address> AllAddresses { get { return allAddresses; } }

public void SaveAddresses() { FileOperations.SaveAddresses(allAddresses); }

public Address FindByName(string name)//метод для нахождения клиента по id

{

return allAddresses.Find(x => x.Name == name);

}

}

class Address

{

string name;

bool deleted;

public Address(string name, bool deleted)

{

this.name = name;

this.deleted = deleted;

}

public Address()

{

}

public string Name { get { return name; } set { name = value; } }

public bool Deleted { get { return deleted; } set { deleted = value; } }

}

}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;

namespace Lab10\_rent\_a\_car

{

public partial class RentsList : Form

{

public RentsList()

{

InitializeComponent();

FillList(Data.CurrentObject);

}

internal void FillList(Object obj)

{

int i = 0;

RentsGrid.Rows.Clear();

foreach (Rent rent in Data.Rents.AllRents)//перебор всех аренд для заполнения

{

if (rent.Name == obj.Name)

{

if (RentsGrid.Rows.Count < Data.Rents.AllRents.Count) { RentsGrid.Rows.Add(); }

int j = 0;

RentsGrid.Rows[i].Cells[j].Value = rent.ID; j++;

RentsGrid.Rows[i].Cells[j].Value = rent.ClientFIO; j++;

RentsGrid.Rows[i].Cells[j].Value = rent.Name; j++;

RentsGrid.Rows[i].Cells[j].Value = rent.DateTime.ToString(); j++;

if (rent.Measure == "дней")

{

RentsGrid.Rows[i].Cells[j].Value = rent.DateTime.AddDays(rent.Period).ToString();

}

if (rent.Measure == "часов")

{

DateTime nDate = rent.DateTime.AddHours(rent.Period);

RentsGrid.Rows[i].Cells[j].Value = nDate.ToString();

}

if (rent.Measure == "месяцев")

{

RentsGrid.Rows[i].Cells[j].Value = rent.DateTime.AddMonths((int)rent.Period).ToString();

}

j++;

RentsGrid.Rows[i].Cells[j].Value = $"{rent.Period} {rent.Measure}"; j++;

RentsGrid.Rows[i].Cells[j].Value = rent.CompAddress; j++;

if (!rent.Saved) { RentsGrid.Rows[i].Cells[j].Value = "Удалена"; j++; }

else

{

RentsGrid.Rows[i].Cells[j].Value = rent.Status; j++;

}

i++;

}

}

}

private void RentsGrid\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

string myID = "";

if (e.ColumnIndex != -1 & e.RowIndex != -1)

{

//подготовка формы аренды

myID = RentsGrid.Rows[e.RowIndex].Cells[0].Value.ToString();

Data.CurrentRent = Data.Rents.FindByID(int.Parse(myID));

RentView rentView = new RentView();

rentView.Owner = this;

if (!Data.CurrentRent.Saved)

{

rentView.statusBox.Text = "Удалена";

rentView.editButton.Enabled = false;

rentView.deleteButton.Enabled = false;

}

rentView.ShowDialog();//открытие формы просмотра аренды

}

}

private void RentsList\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Close();

try

{

ObjectView objectView = this.Owner as ObjectView;

ObjectsList objList = objectView.Owner as ObjectsList;

objectView.Close();

objList.Close();

}

catch

{

}

}

}

}

}

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;

namespace Lab10\_rent\_a\_car

{

public partial class RentView : Form

{

internal RentView()

{

InitializeComponent();

FillForm(Data.CurrentRent);

}

internal void FillForm (Rent rent)//заполнение формы просмотра аренды

{

FIOBox.Text = rent.ClientFIO;

carBox.Text = rent.Name;

priceBox.Text = rent.Price.ToString();

if ( rent.DepositMoney <= 0 ) { depositBox.Text = rent.Deposit.ToString(); }

else { depositBox.Text = rent.DepositMoney.ToString(); }

periodBox.Text = rent.Period.ToString();

measureLabel.Text = rent.Measure;

//totalBox.Text = rent.GetTotalPrice(Data.AddServices);

addressBox.Text = rent.CompAddress;

secondAddressBox.Text = rent.SecondAddress;

try { carPic.Image = System.Drawing.Image.FromFile(Data.Storage + $"DB/Pics/{rent.Name}.jpg"); }

catch { }

if(rent.Status == "Завершена")

{

editButton.Enabled = false;

endRentButton.Enabled = false;

}

statusBox.Text = rent.Status;

fdateBox.Text = rent.DateTime.ToShortDateString();

ftimeBox.Text = rent.DateTime.ToShortTimeString();

if (rent.Measure == "дней")

{

sdateBox.Text = rent.DateTime.AddDays(rent.Period).ToShortDateString();

stimeBox.Text = rent.DateTime.ToShortTimeString();

objectMeasureLabel.Text = "в день";

}

if (rent.Measure == "часов")

{

DateTime nDate = rent.DateTime.AddHours(rent.Period);

stimeBox.Text = nDate.ToShortTimeString();

sdateBox.Text = nDate.ToShortDateString();

objectMeasureLabel.Text = "в час";

}

if (rent.Measure == "месяцев")

{

sdateBox.Text = rent.DateTime.AddMonths((int)rent.Period).ToShortDateString();

stimeBox.Text = rent.DateTime.ToShortTimeString();

objectMeasureLabel.Text = "в месяц";

}

descriptionBox.Text = rent.Description;

if (addServicesListBox.Items.Count == 0) { BuildAddServicesListBox(Data.AddServices); }

for (int i = 0; i < addServicesListBox.Items.Count; i++)//простановка галочек в листбоксе

{

if (rent.AddServices.Contains(i)) { addServicesListBox.SetItemChecked(i, true); }

else { addServicesListBox.SetItemChecked(i, false); }

}

List<AddService> ads = Data.AddServices.AllServices;

decimal p = 0m;

foreach (string elem in addServicesListBox.CheckedItems)

{

AddService adds;

try

{

adds = Data.AddServices.FindByName(elem.Remove(elem.IndexOf('(')));

p += adds.Price;

}

catch

{

}

}

if (Data.Settings.Round == 0)

{

totalBox.Text = ((rent.Price \* (decimal)(Math.Ceiling(rent.Period)))+p).ToString();

}

if (Data.Settings.Round == 1)

{

totalBox.Text = ((rent.Price \* (decimal)(Math.Round(rent.Period)))+p).ToString();

}

if (Data.Settings.Round == 2)

{

totalBox.Text = ((rent.Price \* (decimal)(Math.Floor(rent.Period)))+p).ToString();

}

if (Data.Settings.Round == 3)

{

totalBox.Text = ((rent.Price \* (decimal)(rent.Period))+p).ToString();

}

}

private void BuildAddServicesListBox(AddServices addServices)//заполнение списка листбокса доп.услуг

{

foreach (AddService add in addServices.AllServices)

{

addServicesListBox.Items.Add(add.Name + "(" + add.Price + " руб)");

}

}

private void editButton\_Click(object sender, EventArgs e)//открытие формы редактирования

{

RentEdit rentEdit = new RentEdit(Data.CurrentRent);

rentEdit.Owner = this;

rentEdit.ShowDialog();

}

private void RentView\_FormClosed(object sender, FormClosedEventArgs e)

{

//обновление формы "родителя"

carPic.Image.Dispose();

try

{

MainForm mainForm = this.Owner as MainForm;

mainForm.FillRents();

}

catch { }

try

{

RentsList rentList = this.Owner as RentsList;

rentList.FillList(Data.CurrentObject);

}

catch { }

}

private void deleteButton\_Click(object sender, EventArgs e)//удаление элемента

{

if (MessageBox.Show("Вы точно хотите удалить данную аренду?", "!!!", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

Data.Status = false;

Data.CurrentRent.Saved = false;

if(Data.CurrentRent.Status == "Активная" || Data.CurrentRent.Status == "Забронирована")

{

Data.Objects.FindByName(Data.CurrentRent.Name).InStock++;

Data.Objects.FindByName(Data.CurrentRent.Name).InRent--;

}

Data.Rents.ReplaceInfoRent(Data.CurrentRent);

Data.Rents.SavedRents.Remove(Data.CurrentRent);

Data.CurrentRent = null;

this.Close();

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Есть несохраненные данные");

}

}

private void RentView\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Close();

try

{

RentsList rList = this.Owner as RentsList;

ObjectView objectView = rList.Owner as ObjectView;

ObjectsList objList = objectView.Owner as ObjectsList;

rList.Close();

objectView.Close();

objList.Close();

}

catch

{

}

}

}

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

{

if(Data.CurrentRent.Status != "Завершена")

{

if(MessageBox.Show("После завершения аренды ее невозможно будет редактировать, вы уверены?",

"",MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

if (Data.CurrentRent.Status == "Активная" || Data.CurrentRent.Status == "Забронирована")

{

Data.Objects.FindByName(Data.CurrentRent.Name).InStock++;

Data.Objects.FindByName(Data.CurrentRent.Name).InRent--;

}

Data.CurrentRent.Status = "Завершена";

this.Close();

}

}

else

{

MessageBox.Show("Аренда уже завершена!","!!!",MessageBoxButtons.OK,MessageBoxIcon.Error);

}

}

}

}

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;

namespace Lab10\_rent\_a\_car

{

public partial class ServiceEdit : Form

{

public ServiceEdit()//инициализация новой услуги

{

InitializeComponent();

Data.CurrentService = new AddService(Data.AddServices.GetMaxID() + 1);

Data.AddServices.AddService(Data.CurrentService);

}

internal ServiceEdit(AddService addService)//инициализация существующей услуги

{

InitializeComponent();

FillForm(addService);

}

internal void FillForm(AddService addService)//заполнение

{

nameBox.Text = addService.Name;

priceBox.Text = addService.Price.ToString();

}

private void saveButton\_Click(object sender, EventArgs e)//сохранение услуги

{

if (nameBox.Text == "" || priceBox.Text == "")

{

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Заполнены не все поля");

MessageBox.Show("Для сохранения заполните все поля формы.", "Упс!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

SaveThisForm();

Data.Status = false;

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Есть несохраненные данные");

MessageBox.Show("Данные успешно изменены.", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

ServiceView serviceView = this.Owner as ServiceView;

if(serviceView != null)serviceView.Close();

}

}

private void SaveThisForm()//сохранение обновленной (или нет) формы

{

Data.CurrentService.Name = nameBox.Text;

Data.CurrentService.Price = decimal.Parse(priceBox.Text);

Data.CurrentService.Deleted = false;

Data.AddServices.ReplaceInfoService(Data.CurrentService);

}

private void priceBox\_KeyPress(object sender, KeyPressEventArgs e)//вводить только числа

{

if (Char.IsDigit(e.KeyChar) || Char.IsControl(e.KeyChar))

return;

else

e.Handled = true;

}

private void ServiceEdit\_FormClosed(object sender, FormClosedEventArgs e)//изменение формы родителя

{

try

{

ServiceView serviceView = this.Owner as ServiceView;

if (nameBox.Text == "" || priceBox.Text == "")

{

Data.AddServices.DelService(Data.CurrentService);

serviceView.Close();

}

serviceView.FillForm(Data.CurrentService);

}

catch

{

AddServicesList addServicesList = this.Owner as AddServicesList;

addServicesList.FillGrid();

}

}

private void ServiceEdit\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Close();

try

{

ServiceView servView = this.Owner as ServiceView;

AddServicesList servList = servView.Owner as AddServicesList;

servView.Close();

servList.Close();

}

catch

{

}

}

}

}

}

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;

namespace Lab10\_rent\_a\_car

{

public partial class ServiceView : Form

{

public ServiceView()//инициализация

{

InitializeComponent();

FillForm(Data.CurrentService);

}

internal void FillForm(AddService addService)//заполнение

{

nameBox.Text = addService.Name;

priceBox.Text = addService.Price.ToString();

}

private void editButton\_Click(object sender, EventArgs e)//открытие формы изменения

{

ServiceEdit serviceEdit = new ServiceEdit(Data.CurrentService);

serviceEdit.Owner = this;

serviceEdit.ShowDialog();

}

private void deleteButton\_Click(object sender, EventArgs e)//удаление

{

if (MessageBox.Show("Вы точно хотите удалить данную услугу?", "!!!", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

Data.CurrentService.Deleted = true;

Data.AddServices.ReplaceInfoService(Data.CurrentService);

Data.CurrentService = null;

Data.Status = false;

this.Close();

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Есть несохраненные данные");

}

}

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

{

AddServicesList servicesList = this.Owner as AddServicesList;

servicesList.FillGrid();

}

private void ServiceView\_FormClosed(object sender, FormClosedEventArgs e)//изменения в форму родителя

{

AddServicesList servicesList = this.Owner as AddServicesList;

servicesList.FillGrid();

}

private void ServiceView\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Close();

try

{

AddServicesList servList = this.Owner as AddServicesList;

servList.Close();

}

catch

{

}

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab10\_rent\_a\_car

{

public class Settings

{

List<bool> boolSettings;

int defaultPeriod;

int round;

public Settings()

{

boolSettings = new List<bool>();

}

public void AddSetting(bool setting) { boolSettings.Add(setting); }

public List<bool> BoolSettings { get { return boolSettings; } }

public int Round { get { return round; } set { round = value; } }

public int DefaultPeriod { get { return defaultPeriod; } set { this.defaultPeriod = value; } }

public void SaveSettings() { FileOperations.SaveSettings(this); }

}

}

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.IO;

namespace Lab10\_rent\_a\_car

{

public partial class SettingsForm : Form

{

public SettingsForm()//инициализация

{

InitializeComponent();

FillForm();

}

public void FillForm()//запонение формы настроек

{

DefaultPrice.Checked = Data.Settings.BoolSettings[0];

DefaultDeposit.Checked = Data.Settings.BoolSettings[1];

DefaultPeriod.Text = Data.Settings.DefaultPeriod.ToString();

roundBox.SelectedIndex = Data.Settings.Round;

dbPathBox.Text = Data.Storage + "DB";

}

private void okButton\_Click(object sender, EventArgs e)//выход

{

if (DefaultPrice.Checked) { Data.Settings.BoolSettings[0] = true; }

if (!DefaultPrice.Checked) { Data.Settings.BoolSettings[0] = false; }

if (DefaultDeposit.Checked) { Data.Settings.BoolSettings[1] = true; }

if (!DefaultDeposit.Checked) { Data.Settings.BoolSettings[1] = false; }

Data.Settings.DefaultPeriod = int.Parse(DefaultPeriod.Text);

Data.Settings.Round = roundBox.SelectedIndex;

this.Close();

}

private void About\_Click(object sender, EventArgs e)//открытие "О программе"

{

About about = new About();

about.ShowDialog();

}

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

{

folderBrowserDialog.Description = "Выберите папку, в которую хотите копировать данные программы.";

if(folderBrowserDialog.ShowDialog() == DialogResult.OK)

{

if (Directory.Exists($"{folderBrowserDialog.SelectedPath}/DB\_Backup")) { Directory.Delete($"{folderBrowserDialog.SelectedPath}/DB\_Backup", true); }

Directory.CreateDirectory($"{folderBrowserDialog.SelectedPath}/DB\_Backup");

DirectoryCopy(Data.Storage + "DB", $"{folderBrowserDialog.SelectedPath}/DB\_Backup", true);

MessageBox.Show("Папка данных успешно скопирована");

MainForm.stStrip.Items.Add("Папка данных успешно скопирована");

}

}

private static void DirectoryCopy(string sourceDirName, string destDirName, bool copySubDirs)

{

DirectoryInfo dir = new DirectoryInfo(sourceDirName);

if (!dir.Exists)

{

throw new DirectoryNotFoundException(

"Source directory does not exist or could not be found: "

+ sourceDirName);

}

DirectoryInfo[] dirs = dir.GetDirectories();

if (!Directory.Exists(destDirName))

{

Directory.CreateDirectory(destDirName);

}

FileInfo[] files = dir.GetFiles();

foreach (FileInfo file in files)

{

string temppath = Path.Combine(destDirName, file.Name);

file.CopyTo(temppath, false);

}

if (copySubDirs)

{

foreach (DirectoryInfo subdir in dirs)

{

string temppath = Path.Combine(destDirName, subdir.Name);

DirectoryCopy(subdir.FullName, temppath, copySubDirs);

}

}

}

private void SettingsForm\_FormClosed(object sender, FormClosedEventArgs e)

{

okButton\_Click(sender, e);

}

private void SettingsForm\_KeyDown(object sender, KeyEventArgs e)

{

if(e.KeyCode == Keys.Escape)

{

this.Close();

}

}

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

{

try

{

if (folderBrowserDialog.ShowDialog() == DialogResult.OK)

{

folderBrowserDialog.Description = "Выберите папку, в которой хотите хранить данные программы.";

if (!Directory.Exists($"{folderBrowserDialog.SelectedPath}/DB")) { Directory.CreateDirectory($"{folderBrowserDialog.SelectedPath}/DB"); }

else { Directory.Delete($"{folderBrowserDialog.SelectedPath}/DB", true); Directory.CreateDirectory($"{folderBrowserDialog.SelectedPath}/DB"); }

DirectoryCopy(Data.Storage + "DB", $"{folderBrowserDialog.SelectedPath}/DB", true);

Data.Storage = folderBrowserDialog.SelectedPath + "\\";

if (Data.Storage == "\\") { Data.Storage = ""; }

Data.Status = false;

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Местоположение базы данных успешно изменено, выполните сохранение.");

MessageBox.Show("Местоположение базы данных успешно изменено");

dbPathBox.Text = Data.Storage + "DB";

}

}

catch

{

MessageBox.Show("Ошибка при изменении места хранения базы данных");

}

}

}

}

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;

namespace Lab10\_rent\_a\_car

{

public partial class UserEdit : Form

{

string inLogin;

public UserEdit()

{

InitializeComponent();

try

{

FillForm();

inLogin = Data.CurrentUser.Login;

}

catch

{

loginBox.Text = ""; passwordBox.Text = ""; managerB.Checked = true;

}

}

internal void FillForm() //запонение формы адреса

{

loginBox.Text = Data.CurrentUser.Login;

passwordBox.Text = Data.CurrentUser.Password;

if(Data.CurrentUser.Status == "admin")

{

adminB.Checked = true;

}

else

{

managerB.Checked = true;

}

}

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

{

if (loginBox.Text == "" || passwordBox.Text == "")

{

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Заполнены не все поля");

MessageBox.Show("Для сохранения заполните поле формы.", "Упс!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

SaveThisForm();

Data.Status = false;

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Есть несохраненные данные");

MessageBox.Show("Данные успешно изменены.", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

}

}

private void SaveThisForm()//сохранение формы

{

if (Users.FindByLogin(inLogin) == null)

{

Data.CurrentUser.Login = loginBox.Text;

Data.CurrentUser.Password = passwordBox.Text;

if (managerB.Checked) { Data.CurrentUser.Status = "manager"; }

if (adminB.Checked) { Data.CurrentUser.Status = "admin"; }

Users.allUsers.Add(Data.CurrentUser);

}

else

{

Users.FindByLogin(inLogin).Login = loginBox.Text;

Users.FindByLogin(inLogin).Password = passwordBox.Text;

if (managerB.Checked) { Users.FindByLogin(inLogin).Status = "manager"; }

if (adminB.Checked) { Users.FindByLogin(inLogin).Status = "admin"; }

}

}

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

{

if (MessageBox.Show("Вы точно хотите удалить данного пользователя?", "!!!", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

Users.allUsers.Remove(Users.FindByLogin(Data.CurrentUser.Login));

Data.CurrentUser = null;

Data.Status = false;

this.Close();

MainForm.stStrip.Items.Clear();

MainForm.stStrip.Items.Add("Есть несохраненные данные");

}

}

private void UserEdit\_FormClosed(object sender, FormClosedEventArgs e)

{

UsersList usersList = this.Owner as UsersList;

usersList.FillGrid();

}

private void UserEdit\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Close();

try

{

UsersList usersList = this.Owner as UsersList;

usersList.Close();

}

catch

{

}

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab10\_rent\_a\_car

{

static class Users

{

public static List<User> allUsers = new List<User>();

public static User CurrentUser;

public static User FindByLogin(string login)//метод для нахождения клиента по id

{

return allUsers.Find(x => x.Login == login);

}

}

class User

{

private string login;

private string password;

private string status;

public User(string login, string password, string status)

{

this.login = login;

this.password = password;

this.status = status;

}

public User()

{

}

public string Login { get { return login; } set { login = value; } }

public string Password { get { return password; } set { password = value; } }

public string Status { get { return status; } set { status = value; } }

}

}

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;

namespace Lab10\_rent\_a\_car

{

public partial class UsersList : Form

{

public UsersList()

{

InitializeComponent();

FillGrid();

}

internal void FillGrid()

{

int i = 0;

UsersGrid.Rows.Clear();

foreach (User user in Users.allUsers)

{

UsersGrid.Rows.Add();

int j = 0;

UsersGrid.Rows[i].Cells[j].Value = user.Login; j++;

UsersGrid.Rows[i].Cells[j].Value = user.Password; j++;

if (user.Status == "manager")

{

UsersGrid.Rows[i].Cells[j].Value = "Менеджер"; j++;

}

else

{

UsersGrid.Rows[i].Cells[j].Value = "Администратор"; j++;

}

i++;

}

}

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

{

Data.CurrentUser = new User();

UserEdit userEdit = new UserEdit();

userEdit.Owner = this;

userEdit.ShowDialog();

}

private void UsersList\_FormClosed(object sender, FormClosedEventArgs e)

{

MainForm mainForm = this.Owner as MainForm;

if (!Data.Status) { mainForm.Text = "Аренда\*"; }

}

private void UsersList\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Close();

}

}

private void UsersGrid\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

if (e.ColumnIndex != -1 & e.RowIndex != -1)

{

User user = Users.FindByLogin(UsersGrid.Rows[e.RowIndex].Cells[0].Value.ToString());

Data.CurrentUser = user;

UserEdit userEdit = new UserEdit();

userEdit.Owner = this;

userEdit.ShowDialog();

}

}

}

}