Федеральное государственное бюджетное образовательное учреждение высшего образования

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**Департамент анализа данных,**

**принятия решений и финансовых технологий**

**Пояснительная записка к междисциплинарной курсовой работе**

на тему:

**Разработка информационно-справочной системы**

**«Прокат автомобилей»**

Выполнил:

студент группы ПИ2-2

Номоконов Иван Александрович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

Научный руководитель:

доцент, к.т.н. Горелов Сергей Витальевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

2019

**Оглавление**

[**Введение 2**](#_30j0zll)

[**1. Постановка задачи 2**](#_1fob9te)

[**2. Описание предметной области 3**](#_3znysh7)

[**3. Актуальность автоматизации 4**](#_2et92p0)

[**4. Описание программы 4**](#_tyjcwt)

[4.1. Алгоритмические решения 4](#_3dy6vkm)

[4.2. Описание интерфейса программы 7](#_1t3h5sf)

[4.3. Состав приложения 14](#_4d34og8)

[**5. Назначение и состав классов программы 14**](#_2s8eyo1)

[5.1. Диаграмма классов 14](#_17dp8vu)

[5.2. Класс клиента отеля 14](#_3rdcrjn)

[5.3. Служебные классы 15](#_26in1rg)

[5.4. Формы 17](#_lnxbz9)

[**Заключение 17**](#_35nkun2)

[**Список литературы 17**](#_1ksv4uv)

[**Приложение. Исходный код программы 18**](#_44sinio)

**Введение**

Целью курсовой работы является приобретение навыков в анализе предметной области и в разработке сложного Windows-приложения на языке программирования C# в среде Visual Studio. Язык С# является достаточно мощным языком для разработки приложений для ПК. Разработка приложения позволит усовершенствовать знания этого языка, а также методов проектирования и анализа предметной области.

Анализ предметной области позволяет разработать информационную модель данных задачи. Во время анализа производится выбор наиболее важных объектов из предметной области и процессов автоматизации, а также разработка новых подходов и функций для решения проблем предметной области.

Также важной целью курсовой работы является разработка пояснительной записки, отвечающей таким требованиям, как полнота, лаконичность, техническая корректность и т.д. Техническая документация является составной частью программного продукта и, следовательно, определяет его качество. Поэтому приобретение навыков в разработке технической документации, а в данном случае пояснительной записки, также является важной задачей.

**1. Постановка задачи**

1. В курсовом проекте должна быть разработана информационная модель предметной области, представленная в виде пользовательских классов или таблиц БД.
2. Должно быть разработано несколько форм пользовательского интерфейса.
3. Разработчик самостоятельно определяет интерфейс программы и ее функциональность, однако для получения максимальной оценки приложение в обязательном порядке независимо от предметной области, указанной в задании, должно выполнять следующие операции:

* Отображать в сетках DataGridView данные предметной области:
* Для информационной модели, основанной на списках, на момент первого запуска программы допускается отсутствие файла. В этом случае списки объектов должны быть созданы в программе.
* Для информационной модели, основанной на БД, таблицы должны быть предварительно заполнены записями.
* Реализовать *добавление* в источник данных нового объекта, *удаление* объекта из источника данных, *редактирование* объекта в источнике данных.
* Реализовать фильтрацию записей источника данных, удовлетворяющих введенному пользователем сложному критерию.
* Реализовать сортировку записей источника данных.
* Для информационной модели, основанной на списках, сохранять источник данных в файле, используя XML-сериализацию**.**
* Для информационной модели, основанной на таблицах БД, обновлять изменения источника данных в базе данных.
* После сохранения данных при запуске программы загрузить данные из файла или базы данных.
* Используя меню или панель инструментов, вызвать приложение Блокнот для просмотра справки о программе: файл Help.txt текущего каталога программы.
* Создать пункт меню «Об авторе».
* Разработать несколько полезных пользователю функций для отображения статистических данных, например, средних, максимальных или минимальных значений, данных для построения гистограммы или графика и т.п.

1. Программа не должна завершаться аварийно: сообщения о некорректном вводе данных, противоречивых или недопустимых значениях данных, при отсутствии данных по функциональному запросу пользователя и других нештатных ситуациях отображать в окнах сообщений.
2. Программа должна быть читабельной и содержать полезные комментарии.
3. Список должен быть реализован в виде коллекции, например, динамического массива типа List<T> или BindingList<T>. Программа не должна завершаться аварийно: сообщения о некорректном вводе данных, противоречивых или недопустимых значениях данных, при отсутствии данных по функциональному запросу пользователя и других нештатных ситуациях отображать в окнах сообщений.
4. Программа должна быть читабельной и содержать полезные комментарии.

**2. Описание предметной области**

Предметной областью данного проекта является фирма, которая предоставляет автомобили в аренду. В работе фирмы присутствуют процессы, которые можно автоматизировать: бронирование автомобиля (изменение условий бронирования, отказ от бронирования), добавление нового автомобиля для аренды (добавляется администратором), изменение характеристик автомобиля и условий его аренды, удаление автомобиля, оставление клиентами отзывов и пожеланий, назначение нового администратора фирмы. Предоставление автомобилей в аренду является для заказчика основным видом деятельности, поэтому необходимо реализовать расчет и вывод статистических данных о доходах и деятельности фирмы.

В данном проекте автоматизированы все процессы, указанные выше.

**3. Актуальность автоматизации**

Клиенты фирмы являются основным источником дохода для заказчика, следовательно, их учет позволит оптимизировать затраты на некоторые виды предлагаемых в аренду автомобиле, а значит, увеличить прибыль фирмы. Также, важным для фирмы является мнение ее клиентов об арендованных автомобилях, поэтому возможность добавления пользователями приложения отзывов крайне актуальна.

Во время развития фирмы, она может приобретать новые автомобили и изменять цены на старые, следовательно, для того чтобы прайс лист приложения оставался актуальным, очень важно автоматизировать процесс добавления новых транспортных средств для аренды, а также их изменения и удаления.

Автоматизация указанных выше процессов деятельности фирмы позволит сократить непроизводительные методы использования труда работников, повысить производительность труда и, как следствие, увеличить объем предоставляемых услуг и качество обслуживания, что приведет к появлению новых клиентов и увеличению прибыли. Поэтому разработка приложения для автоматизации деятельности данной фирмы является актуальной.

**4. Описание программы**

**4.1. Алгоритмические решения**

Данные приложения сохраняются в виде списка соответствующего класса в нескольких файлах:

* Файл Cars.xml используется для хранения актуальных автомобилей для аренды;
* Файл Books.xml используется для хранения данных о бронировании автомобилей;
* Файл Users.xml используется для хранения данных о пользователях приложения;
* Файл Reports.xml используется для хранения данных об отзывах клиентов фирмы;

При входе в приложение пользователь сразу может выбрать автомобиль для бронирования, нажав на соответствующую сроку в таблице DataGridView. Для открытия формы бронирования необходимо авторизоваться или пройти регистрацию.

После авторизации откроется форма бронирования, при заполнении который от нажатия на кнопку забронировать бронь сохранится в файл Books.xml. Поля имени и email заполняются автоматически, в зависимости от текущего пользователя.

Если авторизованный пользователь является администратором, то при нажатии на строку в таблице автомобилей на начальном экране, откроется форма изменения и удаления выбранного автомобиля.

При внесении изменений в характеристики автомобиля или в условия его аренды появится кнопка сохранить изменения, при нажатии которых данные об автомобиле будут изменены в файле Cars.xml. При нажатии кнопки «Удалить» автомобиль будет удален из списка автомобилей в файле Cars.xml.

Просмотреть актуальные брони можно в личном кабинете, для входа в который необходимо авторизоваться (или зарегистрироваться).

В личном кабинете можно изменить или удалить актуальные брони, если до их начала осталось не менее 5-ти дней, в противном случае бронь можно только удалить. Изменению доступны только даты аренды.

Если пользователь является администратором, то ему доступны бронирования всех пользователей, и он может их изменить, если бронь еще не началась. Администратор может назначать других пользователей администраторами, для этого в личном кабинете есть таблица пользователей с возможностью изменения. Также, администратор в личном кабинете может добавлять автомобили и просматривать статистику.

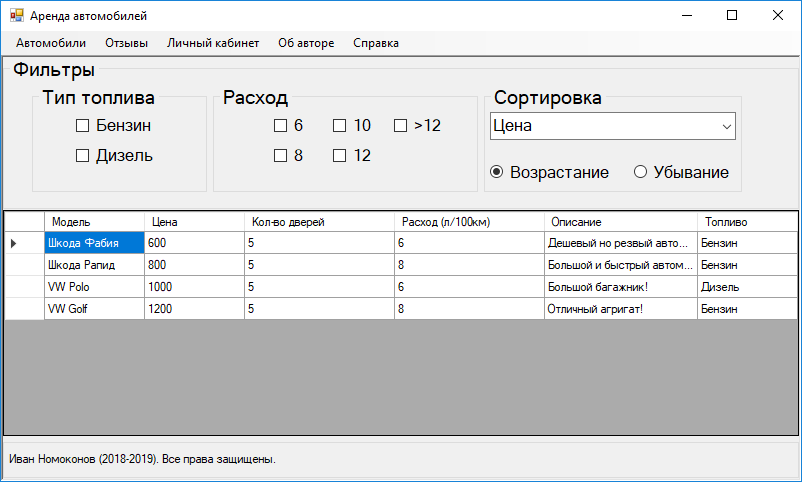
В разделе отзывов все пользователи могут добавить отзыв, предварительно авторизовавшись. На форме добавления отзыва необходимо выбрать автомобиль, к которому относится отзыв, а также указать оценку. Поля имени и email заполняются автоматически, в зависимости от текущего пользователя.

**4.2. Описание интерфейса программы**

В приложении для реализации интерфейса используется 13 элементов управления и компонентов:

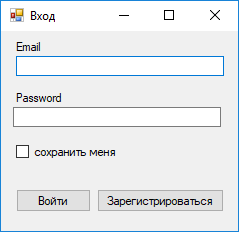
* DataGridView;
* Button;
* TextBox;
* Label;
* CheckBox
* RadioButton;
* GroupBox;
* DateTimePicker;
* MenuStrip;
* NumericUpDown;
* ComboBox;
* Panel.

На основных страницах приложения присутствует меню (MenuStrip) для перехода между основными страницами приложения.



На начальной странице (рис. 1) приложения присутствуют четыре GroupBox для фильтров и сортировки автомобилей, а также таблица dataGridView для вывода автомоблей в зависимости от фильтров и сортировки.

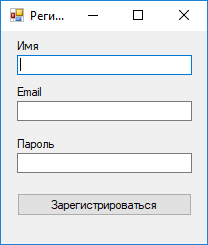
Для выбора фильтра необходимо отметить соответствующий CheckBox. Для активации сортировки необходима с помощью ComboBox выбрать поле, по которому нужно выполнить сортировку, а затем выбрать направление сортировки с помощью RadioButton.



При нажатии на ячейку таблицы автомобилей или при нажатии на пункт меню «Личный кабинет» откроется форма для входа пользователя. Авторизация необходима для оформления заказа.

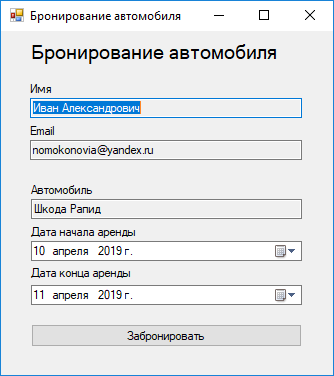
Если отметить CheckBox «сохранить меня», то пользователь сохранится после входа и при следующем открытии приложения будет авторизован автоматически.

Для регистрации нового пользователя необходимо на жать на кнопку «Зарегистрироваться». Откроется форма регистрации, где необходимо заполнить все поля для успешной регистрации.



После нажатия кнопки «Зарегистрироваться» будет создан новый пользователь и форма регистрации закроется. Вы сразу сможете войти в новый аккаунт.

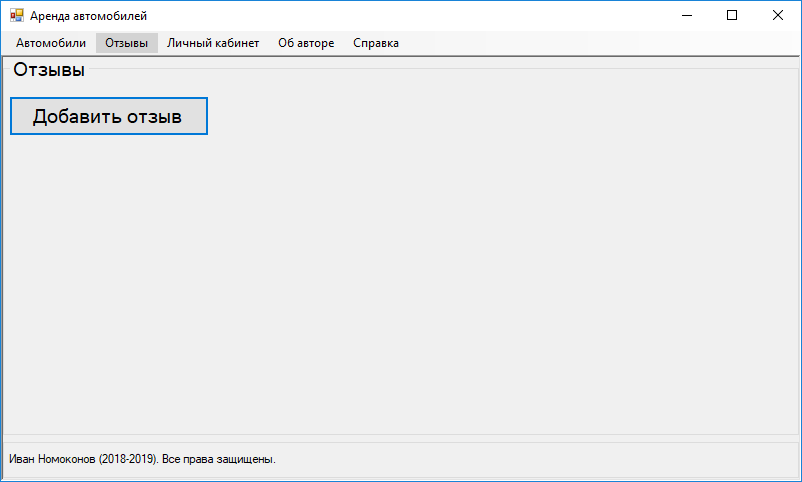
Если пользователь авторизован, то при нажатии на ячейку в таблице автомобилей откроется форма бронирования автомобиля (если пользователь не является администратором).



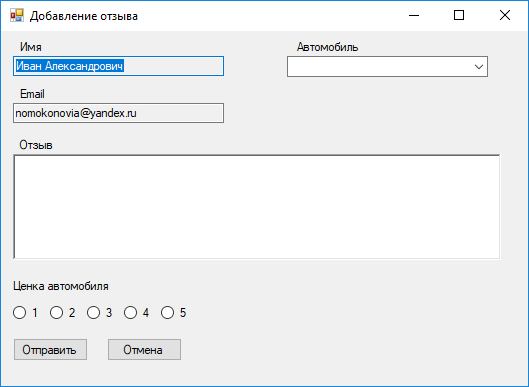
Имя, Email заказчика, а также модель автомобиля будут заполнены автоматически. На форме можно выбрать даты начала и конца аренды. Если дата конца будет меньше даты начала, то высветится соответствующая подсказка. Если в указанные даты автомобиль занят, будет высветится советующая подсказка. Подсказки высвечиваются после нажатия кнопки забронировать и убираются в случае любого изменения любой даты.

После успешного бронирования, новый заказ можно просмотреть, изменить и удалить в личном кабинете.

При нажатии на вкладку меню «Отзывы» откроется страница отзывов.



При нажатии на кнопку добавить отзыв, если пользователь авторизован откроется форма добавления отзыва, иначе откроется форма авторизации.



На форме добавления отзыва необходимо заполнить поле «Отзыв» и выбрать автомобиль к которому относится отзыв из выпадающего списка.

Поля Имя и Email заполняются автоматически в зависимости от текущего пользователя.

Также, обязательно необходимо поставить оценку автомобилю, иначе при попытке отправить отзыв выскочит подсказка «Заполните все поля».

**4.3. Состав приложения**

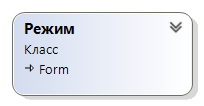
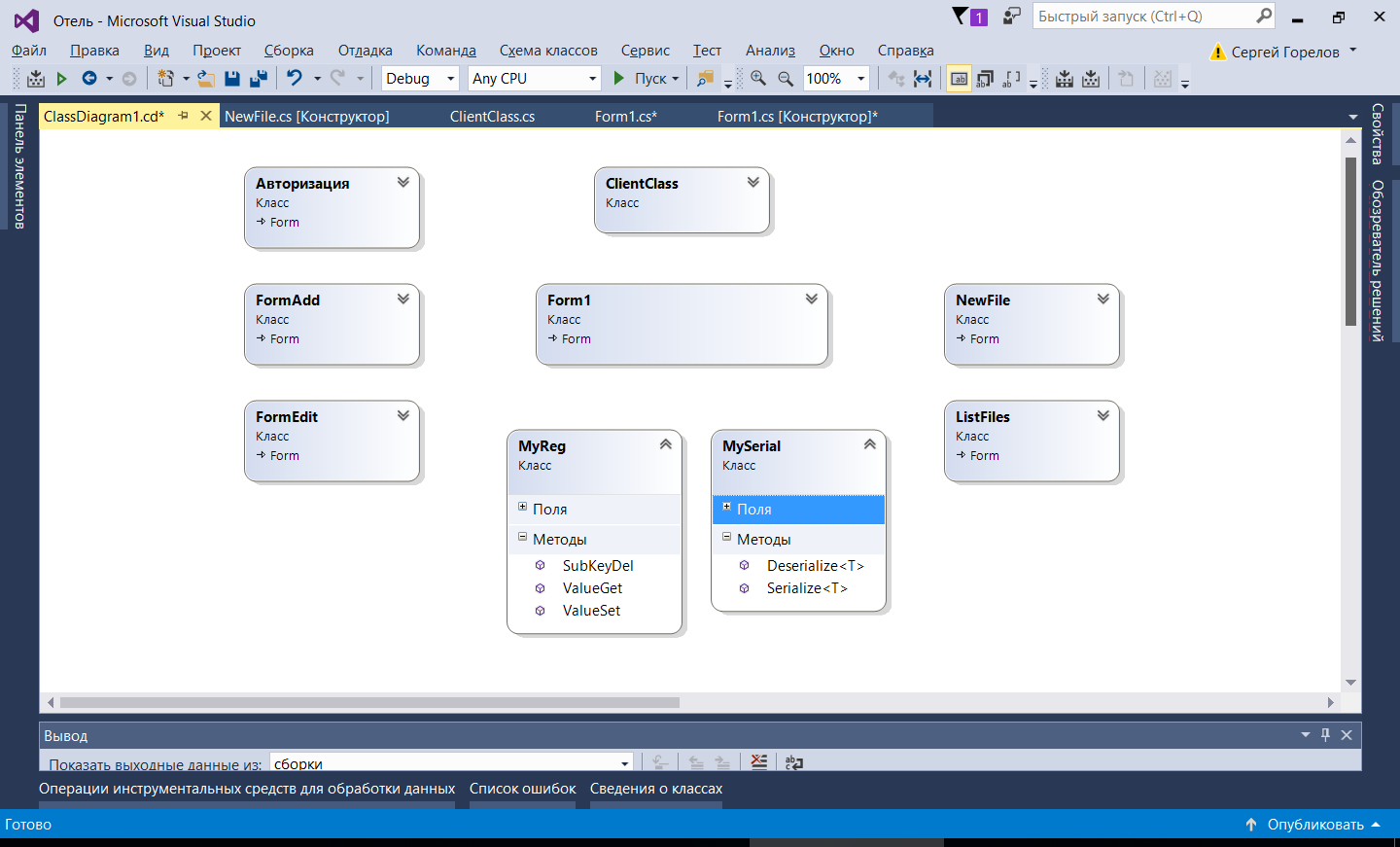
В состав приложения входят следующие файлы и папки:

* Client.exe – программа;
* Manual – папка с инструкцией:
* Info.txt – инструкция;
* Клиенты – папка с базами клиентов:
* БазаДанных1.xml
* БазаДанных2.xml

**5. Назначение и состав классов программы**

**5.1. Диаграмма классов**

Разрабатываемая программа состоит из нескольких классов, содержащих множество полей, методов и свойств. Взаимосвязь основных классов приложения можно увидеть на диаграмме, изображенной на рис. 5.1. Стрелками изображены ассоциации.



***Рис. 5.1.*** *Диаграмма классов*

**5.2. Класс клиента отеля**

Определение класса **ClientClass**клиента отеля представлено в Приложении. В классе описаны следующие открытые свойства и конструкторы:

* Reserv – свойство, позволяющее возвращать и задавать номер брони клиента;
* Room – свойство, позволяющее возвращать и задавать номер забронированной комнаты для клиента.
* Surname – свойство, позволяющее возвращать и задавать фамилию клиента;
* Name – свойство, позволяющее возвращать и задавать имя клиента;
* CheckInDate – свойство, позволяющее возвращать и задавать дату заезда клиента;
* CheckOutDate – свойство, позволяющее возвращать и задавать дату выезда клиента;
* Telefon – свойство, позволяющее возвращать и задавать номер телефона клиента;
* VIP – свойство, позволяющее возвращать и задавать признак VIP клиента;
* NonGrata – свойство, позволяющее возвращать и задавать признак нежелательного клиента;
* ClientClass() – конструктор по умолчанию;
* ClientClass(*параметры*) – параметрический конструктор, осуществляющий инициализацию свойств класса.

**5.3. Служебные классы**

К служебным классам относятся:

* **MyReg** – класс для работы с разделом CurrentUser системного реестра.

Все открытые поля и методы класса **статические**.

*Поля:*

**string subKeyName** – имя подраздела. По умолчанию – "Hotel".

**string valueName** – имя значения (ключ). По умолчанию – "File".

*Методы:*

**string ValueGet()** – метод читает из реестра значение с именем valueName в подразделе subKeyName.

*Возвращаемое значение:*

"*значение*", если значение прочитано;

"", если подраздела или значения нет или зафиксирована ошибка.

**bool ValueSet(string *value*)** – метод записывает значение *value* с именем valueName в подразделе subKeyName.

*value* – записываемое значение.

*Возвращаемое значение:*

true – успешно записано;

false – не записано из-за ошибки.

**bool SubKeyDel()** – метод удаляет значение с именем valueName в подразделе subKeyName.

*Возвращаемое значение:*

true - успешно удалили подраздел Hotel и значение File;

false - не удалили из-за ошибки.

Чтобы использовать класс в других приложениях, необходимо полям subKeyName и valueName присвоить свои значения.

* **MySerial<T>** – универсальный класс для записи в XML-файл и чтения из XML-файла сериализованных данных типа Т. Класс является универсальным. В качестве источника данных используется список типа **List<T>.** Методы класса являются статическими.

*Методы класса:*

* **bool Serialize(string *file*, List<T> *list*)** – метод сериализует список объектов ***list*** типа **Т** в файл ***file***.

*Параметры:*

T – тип класса предметной области;

file – выходной XML-файл;

list – коллекция типа List<T>.

*Возвращаемое значение:*

true – норма;

false – не сериализовано.

В программе установлены следующие значения параметров:

file = Form1.pathCurFile – относительный путь к файлу:

list = Form1.list – список объектов с данными.

* **List<T> Deserialize(string *file*)** – метод десериализует список объектов типа **Т** (читает из XML-файла).

*Параметры:*

T – тип класса предметной области;

file – исходный XML-файл.

*Возвращаемое значение:*

список типа List<T> – норма;

null – не десериализовано.

* **Program** – класс, содержащий метод Main, который позволяет запустить программу, открывая главную форму Form1.

**5.4. Формы**

Ниже представлены пользовательские классы, наследующие базовую функциональность от класса Form.

* **Form1**– класс, содержащий все основные обработчики событий и методы (см. рис. 4.2, 4.8, 4.9).
* **Режим** – класс, осуществляющий вход в основную форму (см. рис. 4.1).
* **FormAdd** – класс, содержащий обработчики событий для добавления новых записей в таблицу (см. рис. 4.6).
* **FormEdit** – класс, содержащий обработчики событий для редактирования выбранной записи в таблице (см. рис. 4.6).
* **ListFiles** – класс, содержащий обработчики событий для выборки и открытия файла, а так же для копирования и удаления файлов (см. рис. 4.4).
* **NewFile** – класс, содержащий обработчики событий для перехода к новому файлу (см. рис. 4.3).

**Заключение**

Разработав Windows приложение, был выполнен ряд поставленных задач. Справочно-информационная система написана на языке программирования С# с использованием классов, методов и свойств. Для реализации некоторых задач потребовалось обратиться к объектам операционной системы: системный реестр, файловая система, процессы.

Программа может модернизироваться и обновляться. К примеру, может быть расширен класс клиента путем добавления паспортных данных, денежных сумм предоплаты и итоговой платы, примечания и т.д.

Программа может быть расширена статистическими и финансовыми функциями.

**Список литературы**

1. Горелов С.В., Волков А.Г. Разработка Windows-приложений. Часть 1. Учебное пособие. Образовательный портал Финансового университета. 2018.
2. Горелов С.В. Разработка Windows-приложений. Часть 2. Учебное пособие. Образовательный портал Финансового университета. 2018.
3. Г. Шилдт. Полный справочник по C#. - М.: «Вильямс», 2004.
4. Официальный сайт Microsoft: [Интернет-ресурс]. URL: https://msdn.microsoft.com org.

**Приложение. Исходный код программы**

**Файл Program.cs:**

using System;

using System.Windows.Forms;

namespace Отель

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

**Файл Режим.cs:**

using System;

using System.Windows.Forms;

namespace Отель

{

public partial class Режим : Form

{

public static int mode = 0; // Режим работы: 1 - только просмотр; 2 - всё; 0 - выход.

public Режим()

{

InitializeComponent();

}

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

{

if (groupBox1.Visible)

{

if (textBox1.Text == "123" && textBox2.Text == "123")

{

mode = 2;

Close();

return;

}

else

{

MessageBox.Show("Не верно введен логин или пароль.");

textBox1.Focus();

return;

}

}

mode = 1;

Close();

return;

}

//---------------------------------------------------

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

{

Close();

}

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

{

groupBox1.Visible = false;

Text += " Версия программы " + Form1.vers;

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

groupBox1.Visible = true;

textBox1.Focus();

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

groupBox1.Visible = false;

}

}

}

**Файл ClientClass.cs:**

public class **ClientClass**

{

public int Reserv { get; set; }

public int Room { get; set; }

public string Surname { get; set; }

public string Name { get; set; }

public DateTime CheckInDate { get; set; }

public DateTime CheckOutDate { get; set; }

public string Telefon { get; set; }

public bool VIP { get; set; }

public bool NonGrata { get; set; }

public **ClientClass**()

{

this.CheckInDate = DateTime.Now;

this.CheckOutDate = DateTime.Now;

}

public **ClientClass**(int Reserv, int Room, string Surname, string Name,

DateTime CheckInDate, DateTime CheckOutDate, string Tele

{

this.Reserv = Reserv;

this.Room = Room;

this.Surname = Surname;

this.Name = Name;

this.CheckInDate = CheckInDate;

this.CheckOutDate = CheckOutDate;

this.Telefon = Telefon;

this.VIP = VIP;

this.NonGrata = NonGrata;

}

}

**Файл Form1.cs**

using System;

using System.Windows.Forms;

using System.IO;

using System.Diagnostics;

using System.Drawing;

using System.Collections.Generic;

namespace Отель

{

public partial class Form1 : Form

{

public string folderClients = "Клиенты"; // Папка в текущем каталоге с файлами клиентов.

public static string nameFile = ""; // Имя текущего файла клиентов.

public static string pathCurFile = ""; // Папка "Клиенты" + имя текущего файла клиентов.

public static List<ClientClass> list = new List<ClientClass>(); // Список клиентов.

public static bool edit = false; // Редактировался ли список (таблица) для ее

// последующего сохранения в файле.

public static string textButton = "Завершить работу с сохранением списка клиентов";

public static bool newFile = false; // Определен ли новый файл клиентов.

public static string vers = "3.0"; // Версия программы.

public Режим formPSWD;

public int rowAdd = 1; // Наличие пустой строки в DataGridView для добавления нового объекта.

//---------------------------------------------------

public Form1()

{ InitializeComponent(); }

//---------------------------------------------------

// Запускаем окно с выбором режима работы.

// Выполняем настройки.

// Проверяем различные сочетания состояния рееста и наличия файлов клиентов.

// Загружаем клиентов либо из тестового списка , либо из текущего файла.

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

{

formPSWD = new Режим();

formPSWD.ShowDialog();

if (Режим.mode == 0)

{

Close();

return;

}

Text += " Версия программы " + vers;

if (Режим.mode == 1) // Только просмотр.

{

dataGridView1.AllowUserToAddRows = false;

dataGridView1.AllowUserToDeleteRows = false;

dataGridView1.ReadOnly = true;

b\_AddNew.Enabled = false;

b\_Edit.Enabled = false;

b\_DeleteRow.Enabled = false;

b\_Cleare.Enabled = false;

b\_NewFileClient.Enabled = false;

textButton = "Завершить работу";

}

label3.Text = "";

b\_SaveList.Text = textButton;

if (dataGridView1.Rows.Count == 1) // Таблица еще пустая.

rowAdd = 1; // Есть строка для добавления объекта

else

rowAdd = 0; // Нет строки для добавления объекта

// Сформировать следующие переменные:

/\*

\* flagFalesInFolder = 0 - нет файлов в папке "Клиенты";

\* flagFalesInFolder = 1 - есть файлы в папке "Клиенты";

\*

\* flagNameFile = 0 - нет имени файла в реестре;

\* flagNameFile = 1 - есть имя файла в реестре;

\*/

int flagNameFile = 0;

int flagFalesInFolder = 0;

// Чтение списка файлов в папке Клиенты.

string[] files = null;

if (!Directory.Exists(folderClients))

{

Directory.CreateDirectory(folderClients);

flagFalesInFolder = 0;

}

else

{

DirectoryInfo di = new DirectoryInfo(folderClients);

FileInfo[] filesDir = di.GetFiles();

files = new string[filesDir.Length];

// Преобразуем из FileInfo в string/

for (int i = 0; i < filesDir.Length; i++)

files[i] = filesDir[i].Name;

if (files.Length > 0) // Файлы клиентов есть.

flagFalesInFolder = 1;

else

flagFalesInFolder = 0;

}

// Чтение имени текущего файла из реестра.

nameFile = MyReg.ValueGet();

if (nameFile == "")

flagNameFile = 0;

else

flagNameFile = 1;

// Анализируем сочетание flagNameFile и flagFalesInFolder.

if (flagNameFile == 1 && flagFalesInFolder == 1)

{

pathCurFile = Path.Combine(folderClients, nameFile);

if (Array.IndexOf(files, nameFile) == -1)

{

MyReg.SubKeyDel(); // Удалить имя файла из реестра.

flagNameFile = 0;

b\_ListFiles\_Click(null, null); //------------->>>>

return;

}

else if ((list = MySerial<ClientClass>.Deserialize(pathCurFile)) != null)

{

pathCurFile = Path.Combine(folderClients, nameFile);

label3.Text = nameFile;

if (Режим.mode == 2)

b\_SaveList.Text = textButton + " в \"" + nameFile + "\"";

clientClassBindingSource.DataSource = list;

edit = false;

return;

}

else

{

pathCurFile = "";

MessageBox.Show("Открыть файл и загрузить список клиентов не удалось.",

"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

flagNameFile = 0;

return;

}

}

if (flagNameFile == 1 && flagFalesInFolder == 0)

{

MyReg.SubKeyDel(); // Удалить имя файла из реестра.

flagNameFile = 0;

}

if (flagNameFile == 0 && flagFalesInFolder == 1)

{

b\_ListFiles\_Click(null, null);

edit = false;

return;

}

if (flagNameFile == 0 && flagFalesInFolder == 0)

{

DialogResult res = MessageBox.Show("Файл не найден! \n\n" +

"Загрузить тестовый список клиентов?",

"Сделайте выбор", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (res == DialogResult.Yes)

{

list.Add(new ClientClass(1, 251, "Иванов", "Петр",

new DateTime(2018, 06, 16), new DateTime(2018, 06, 26),

"+7 (916) 333-4455", false, false));

list.Add(new ClientClass(2, 252, "Петров", "Иван",

new DateTime(2018, 06, 16), new DateTime(2018, 06, 26),

"+7 (916) 444-4455", false, false));

list.Add(new ClientClass(3, 251, "Козлов", "Буян",

new DateTime(2017, 06, 06), new DateTime(2017, 06, 15),

"+7 (916) 555-4455", false, true));

list.Add(new ClientClass(4, 253, "Лепселидзе", "Григорий",

new DateTime(2018, 2, 05), new DateTime(2018, 2, 13),

"+7 (916) 555-4455", true, false));

}

clientClassBindingSource.DataSource = list;

edit = false;

}

}

//---------------------------------------------------

// Отображаем форму со списком файлов.

private void b\_ListFiles\_Click(object sender, EventArgs e)

{

ListFiles form = new ListFiles();

form.form1 = this;

form.ShowDialog();

}

//---------------------------------------------------

// Отображаем форму для перехода к новому файлу.

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

{

NewFile form = new NewFile();

form.form1 = this;

form.ShowDialog();

if (newFile)

{

if (MessageBox.Show("Текущие клиенты из таблицы запомнены в файле " +

Form1.nameFile + ".\n\n" +

"Очистить таблицу клиентов для нового файла?",

"Внимание", MessageBoxButtons.YesNo,

MessageBoxIcon.Question) == DialogResult.Yes)

{

list.Clear();

dataGridView1.Rows.Clear();

//clientClassBindingSource.ResetBindings(false);

}

edit = true;

newFile = false;

}

}

//---------------------------------------------------

// Отображаем форму для добавления нового клиента.

private void b\_AddNew\_Click(object sender, EventArgs e)

{

FormAdd form = new FormAdd();

form.form1 = this;

form.ShowDialog();

if (edit)

clientClassBindingSource.ResetBindings(false);

}

//---------------------------------------------------

// Отображаем форму для редактирования клиента.

private void b\_Edit\_Click(object sender, EventArgs e)

{

if (dataGridView1.Rows.Count == rowAdd)

{

MessageBox.Show("Нельзя редактировать пустую таблицу.", "ОШИБКА");

return;

}

if (dataGridView1.CurrentCell == null)

{

MessageBox.Show("Для редактирования клиента его строку нужно выделить.",

"ОШИБКА");

return;

}

int y = dataGridView1.CurrentRow.Index;

FormEdit form = new FormEdit();

form.form1 = this;

form.ShowDialog();

}

//------------------------------------------------------------------

// Отображаем форму для удаления клиента.

private void b\_DeleteRow\_Click(object sender, EventArgs e)

{

int y = dataGridView1.CurrentRow.Index;

if (dataGridView1.CurrentRow.IsNewRow || y < 0)

{

MessageBox.Show("Для удаления клиента его строку нужно выделить.",

"ОШИБКА");

return;

}

string fam = (string)dataGridView1.CurrentRow.Cells["surnam"].Value;

if (MessageBox.Show("Вы действительно хотите удалить клиента " + fam + "?",

"Внимание", MessageBoxButtons.YesNo,

MessageBoxIcon.Question) == DialogResult.No)

return;

dataGridView1.Rows.RemoveAt(y);

MessageBox.Show("Клиент " + fam + " удален!", "Внимание");

//clientClassBindingSource.ResetBindings(false);

}

//---------------------------------------------------

// Обработчик события Click для отбора клиентов по совокупности критериев.

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

{

b\_ВсеКлиенты\_Click(null, null);

dataGridView1.CurrentCell = null;

int count = dataGridView1.Rows.Count - rowAdd;

for (int i = 0; i < count && dataGridView1.Rows[i].Visible == true; i++)

if (RowFind(i))

dataGridView1.Rows[i].Visible = true;

else

dataGridView1.Rows[i].Visible = false;

}

//---------------------------------------------------

// Метод проверяет поля переданной строки

// на одновременное равенство всем критериям.

private bool RowFind(int i)

{

if (reservBox.Text != "")

if (dataGridView1["reserv", i].Value.ToString() != reservBox.Text)

return false;

if (roomBox.Text != "")

if (dataGridView1["room", i].Value.ToString() != roomBox.Text)

return false;

if (surnameBox.Text != "")

if (dataGridView1["surnam", i].Value.ToString() != surnameBox.Text)

return false;

if (telefonBox.Text != "+7 ( ) -")

if ((string)dataGridView1["telefon", i].Value != telefonBox.Text)

return false;

if (inDateBox.Text != "")

if ((DateTime)dataGridView1["checkInDate", i].Value !=

DateTime.Parse(inDateBox.Text))

return false;

if (outDateBox.Text != "")

if ((DateTime)dataGridView1["checkOutDate", i].Value !=

DateTime.Parse(outDateBox.Text))

return false;

if (VIPcheckBox.Checked)

if (!(bool)dataGridView1["vIP", i].Value)

return false;

if (NONcheckBox.Checked)

if (!(bool)dataGridView1["nonGrata", i].Value)

return false;

return true;

}

//---------------------------------------------------

// Отображение всех клиентов списка.

private void b\_ВсеКлиенты\_Click(object sender, EventArgs e)

{

int count = dataGridView1.Rows.Count - rowAdd;

for (int i = 0; i < count; i++)

dataGridView1.Rows[i].Visible = true;

}

//---------------------------------------------------

// Запуск Блокнота

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

{

Process p = Process.Start("NotePad.exe", "Manual\\Info.txt");

}

bool close = false; // Для Form1\_FormClosed: true - выход был по кнопке b\_SaveClientsToFile.

//------------------------------------------------------------------

// Если есть тестовые записи, то предлагается их сохранить в файле.

// Если таблица клиентов была изменена, то список сохраняется в файле.

private void b\_SaveClientsToFile\_Click(object sender, EventArgs e)

{

close = true;

if (Режим.mode == 2) // Полные возможности.

{

if (pathCurFile != "")

{

if (edit)

{

// Запомнить текущих клиентов в текущем файле.

if (!MySerial<ClientClass>.Serialize(pathCurFile, list))

{

MessageBox.Show("Текущих клиентов в файле запомнить не удалось!\n" +

"Чтобы их не потерять, создайте новый файл.",

"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

}

else

{

// Файла нет. Это первое сохранение.

MessageBox.Show("Первый запуск программы. \n\n" +

"Файла для сохранения клиентов нет.\n" +

"Чтобы его создать введите имя файла.",

"Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Information);

NewFile form = new NewFile();

form.form1 = this;

form.ShowDialog();

if (pathCurFile != "")

{

if (!MySerial<ClientClass>.Serialize(pathCurFile, list))

{

MessageBox.Show("Текущих клиентов в файле запомнить не удалось!\n" +

"Чтобы их не потерять, создание нового файла отменяется.",

"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

if (MessageBox.Show("Файл не выбран!\n\n" +

"Завершить программу без сохранения?", "Сделайте выбор.",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question) == DialogResult.No)

return;

}

}

}

Close();

}

//-------------------------------------------------------------------

// Очистка таблицы клиентов.

private void b\_Clear\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Вы действительно хотите очистить таблицу клиентов?",

"Внимание", MessageBoxButtons.YesNo,

MessageBoxIcon.Question) == DialogResult.Yes)

{

dataGridView1.Rows.Clear();

list.Clear();

edit = true;

//clientClassBindingSource.ResetBindings(false);

}

}

//-------------------------------------------------------------------

// Сортировка списка по свойству, соответствующему выбранному заголовку.

private void dataGridView1\_ColumnHeaderMouseClick(object sender,

DataGridViewCellMouseEventArgs e)

{

if (dataGridView1.Rows.Count > rowAdd)

{

switch (dataGridView1.Columns[e.ColumnIndex].Name)

{

case "reserv":

list.Sort(delegate (ClientClass a1, ClientClass a2)

{ return a1.Reserv.CompareTo(a2.Reserv); });

break;

case "room":

list.Sort(delegate (ClientClass a1, ClientClass a2)

{ return a1.Room.CompareTo(a2.Room); });

break;

case "surnam":

list.Sort(delegate (ClientClass a1, ClientClass a2)

{ return a1.Surname.CompareTo(a2.Surname); });

break;

case "name":

list.Sort(delegate (ClientClass a1, ClientClass a2)

{ return a1.Name.CompareTo(a2.Name); });

break;

case "checkInDate":

list.Sort(delegate (ClientClass a1, ClientClass a2)

{ return a1.CheckInDate.CompareTo(a2.CheckInDate); });

break;

case "checkOutDate":

list.Sort(delegate (ClientClass a1, ClientClass a2)

{ return a1.CheckOutDate.CompareTo(a2.CheckOutDate); });

break;

}

clientClassBindingSource.ResetBindings(false);

}

}

//-------------------------------------------------------------------

// Отображение календаря DateTimePicker.

private void b\_dateIn\_Click(object sender, EventArgs e)

{

dateTimePicker1.Location = ((Button)sender).Location;

dateTimePicker1.Visible = true;

}

//-------------------------------------------------------------------

// Отображение календаря DateTimePicker.

private void b\_dateOut\_Click(object sender, EventArgs e)

{

dateTimePicker2.Location = ((Button)sender).Location;

dateTimePicker2.Visible = true;

}

//-------------------------------------------------------------------

// Копирование даты в текстовое поле при потере фокуса календарем.

private void dateTimePicker1\_Leave(object sender, EventArgs e)

{

inDateBox.Text = dateTimePicker1.Text;

dateTimePicker1.Visible = false;

}

//-------------------------------------------------------------------

// Копирование даты в текстовое поле при потере фокуса календарем.

private void dateTimePicker2\_Leave(object sender, EventArgs e)

{

outDateBox.Text = dateTimePicker2.Text;

dateTimePicker2.Visible = false;

}

//-------------------------------------------------------------------

// Очищение всех критериев отбора клиентов.

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

{

reservBox.Text = "";

roomBox.Text = "";

surnameBox.Text = "";

inDateBox.Text = "";

outDateBox.Text = "";

telefonBox.Text = "";

VIPcheckBox.Checked = false;

NONcheckBox.Checked = false;

}

//---------------------------------------------------------------------

// Срабатывает при любом завершении программы.

// Отделяем завершение по системной кнопке X от Button.

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

if (close || Режим.mode == 1) // close=true - было завершение по Button

e.Cancel = false; // Закрыть форму

else if (edit == false)

e.Cancel = false; // Закрыть форму

else if (MessageBox.Show("Выйти без сохранения списка клиентов?",

"Внимание", MessageBoxButtons.YesNo,

MessageBoxIcon.Question) == DialogResult.Yes)

e.Cancel = false; // Закрыть форму

else

e.Cancel = true; // Отменить закрытие формы

}

//-------------------------------------------------------------

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

{

MessageBox.Show("Программу разработал Горелов С.В. в качестве примера для курсовой работы студентов.",

"Версия " + vers, MessageBoxButtons.OK, MessageBoxIcon.Information);

}

//---------------------------------------------------------------------------------------

// Проверка даты отъезда по отношению к дате заезда.

// Происходит при перемещении фокуса на другую строку.

private void dataGridView1\_RowValidating(object sender, DataGridViewCellCancelEventArgs e)

{

e.Cancel = false;

DateTime dt1 = (DateTime)dataGridView1["checkInDate", e.RowIndex].Value;

DateTime dt2 = (DateTime)dataGridView1["checkOutDate", e.RowIndex].Value;

// Проверяем поля даты.

if (dt1 > dt2)

{

MessageBox.Show("Дата заезда не может быть позже даты отъезда", "ОШИБКА", MessageBoxButtons.OK, MessageBoxIcon.Error);

e.Cancel = true;

}

}

//---------------------------------------------------------------------------------------

// Проверка допустимости нового значения в ячейке при перемещении фокуса на другую ячейку этой же строки.

private void dataGridView1\_CellValidating(object sender, DataGridViewCellValidatingEventArgs e)

{

e.Cancel = false;

string valueNew = e.FormattedValue.ToString().Trim();

string nameValue = dataGridView1.Columns[e.ColumnIndex].Name; // Имя поля

int vInt;

DateTime dt;

switch (nameValue)

{

case "reserv":

if (!int.TryParse(valueNew, out vInt))

{

MessageBox.Show("Поле \"№ брони\" должно содержать числовое значение!", "Ошибка ввода данных", MessageBoxButtons.OK, MessageBoxIcon.Error);

e.Cancel = true;

}

return;

case "room":

if (!int.TryParse(valueNew, out vInt))

{

MessageBox.Show("Поле \"№ комнаты\" должно содержать числовое значение!", "Ошибка ввода данных", MessageBoxButtons.OK, MessageBoxIcon.Error);

e.Cancel = true;

}

return;

case "surnam":

if (valueNew == "")

{

MessageBox.Show("Поле \"Фамилия\" не должно быть пустым!",

"Ошибка ввода данных", MessageBoxButtons.OK, MessageBoxIcon.Error);

e.Cancel = true;

}

return;

case "name":

if (valueNew == "")

{

MessageBox.Show("Поле \"Имя\" не должно быть пустым!",

"Ошибка ввода данных",

MessageBoxButtons.OK, MessageBoxIcon.Error);

e.Cancel = true;

}

return;

case "checkInDate":

if (valueNew == "" || !DateTime.TryParse(valueNew, out dt))

{

MessageBox.Show("Поле \"Дата заезда\" должно содержать дату в формате дд.мм.гггг !", "Ошибка ввода данных",

MessageBoxButtons.OK, MessageBoxIcon.Error);

e.Cancel = true;

}

return;

case "checkOutDate":

if (valueNew == "" || !DateTime.TryParse(valueNew, out dt))

{

MessageBox.Show("Поле \"Дата выезда\" должно содержать дату в формате дд.мм.гггг !", "Ошибка ввода данных",

MessageBoxButtons.OK, MessageBoxIcon.Error);

e.Cancel = true;

}

return;

}

}

//--------------------------------------------------------------------------

// Изменить цвет у VIP и NonGrata.

// Этот обработчик вызывается при срабатывании привязки,

// поэтому здесь обновляется закраска.

private void dataGridView1\_DataBindingComplete(object sender,

DataGridViewBindingCompleteEventArgs e)

{

for (int i = 0; i < dataGridView1.Rows.Count - rowAdd; i++)

{

edit = true;

if ((bool)(dataGridView1["nonGrata", i].Value) == true)

{

dataGridView1["surNam", i].Style.BackColor = Color.LightCyan;

dataGridView1["name", i].Style.BackColor = Color.LightCyan;

}

else

{

dataGridView1["surNam", i].Style.BackColor = Color.White;

dataGridView1["name", i].Style.BackColor = Color.White;

}

if ((bool)(dataGridView1["vIP", i].Value) == true)

{

dataGridView1["surNam", i].Style.ForeColor = Color.Maroon;

dataGridView1["name", i].Style.ForeColor = Color.Maroon;

}

else

{

dataGridView1["surNam", i].Style.ForeColor = Color.Black;

dataGridView1["name", i].Style.ForeColor = Color.Black;

}

}

}

//--------------------------------------------------------------------------

// Изменить цвет у VIP и NonGrata.

// Этот обработчик вызывается при попытке изменить значение ячейки

// с использованием базового интерфейса сетки.

// Раз есть попытка, значит флажок переброшен: меняем цвет.

private void dataGridView1\_CellBeginEdit(object sender,

DataGridViewCellCancelEventArgs e)

{

if (dataGridView1.Columns[e.ColumnIndex].Name == "nonGrata")

{

if ((bool)(dataGridView1["nonGrata", e.RowIndex].Value) == false)

{

dataGridView1["surNam", e.RowIndex].Style.BackColor = Color.LightCyan;

dataGridView1["name", e.RowIndex].Style.BackColor = Color.LightCyan;

}

else

{

dataGridView1["surNam", e.RowIndex].Style.BackColor = Color.White;

dataGridView1["name", e.RowIndex].Style.BackColor = Color.White;

}

}

if (dataGridView1.Columns[e.ColumnIndex].Name == "vIP")

{

if ((bool)(dataGridView1["vIP", e.RowIndex].Value) == false)

{

dataGridView1["surNam", e.RowIndex].Style.ForeColor = Color.Maroon;

dataGridView1["name", e.RowIndex].Style.ForeColor = Color.Maroon;

}

else

{

dataGridView1["surNam", e.RowIndex].Style.ForeColor = Color.Black;

dataGridView1["name", e.RowIndex].Style.ForeColor = Color.Black;

}

}

}

}

}

**Файл NewFile.cs:**

using System;

using System.IO;

using System.Windows.Forms;

namespace Отель

{

public partial class NewFile : Form

{

public Form1 form1;

public NewFile()

{

InitializeComponent();

}

//-----------------------------------------------------------

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

{

if (!Directory.Exists(form1.folderClients))

Directory.CreateDirectory(form1.folderClients);

// Заполнить список именами старых файлов.

LoadListBox();

if (Form1.pathCurFile == "")

{

button1.Text = "Выбрать текущим новый файл и выйти";

}

// Убрать выделение текста

textBox1.SelectionStart = 0;

}

//-----------------------------------------------------

private void LoadListBox()

{

// Заполнить список именами всех файлов.

DirectoryInfo di = new DirectoryInfo(form1.folderClients);

FileInfo[] files = di.GetFiles();

if (files.Length != 0)

{

listBox1.Items.Clear();

foreach (var item in files)

listBox1.Items.Add(item.Name);

listBox1.SelectedIndex = listBox1.FindString(Form1.nameFile);

}

}

//-----------------------------------------------------------

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

// Только текущий файл может быть выделен.

listBox1.SelectedIndex = listBox1.FindString(Form1.nameFile);

}

//-----------------------------------------------------------

private void b\_Create\_Click(object sender, EventArgs e)

{

if (Path.GetFileNameWithoutExtension(textBox1.Text) == "" )

{

MessageBox.Show("Вы не ввели имя нового файла!", "Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (Path.GetExtension(textBox1.Text) == "") textBox1.Text += ".xml";

textBox1.Text = textBox1.Text.Trim();

if (listBox1.FindString(textBox1.Text) > 0)

{

MessageBox.Show("Вы ввели имя нового файла, который уже существует!",

"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (Form1.pathCurFile != "" && Form1.edit)

{

// Запомнить текущих клиентов.

if (!MySerial<ClientClass>.Serialize(Form1.pathCurFile, Form1.list))

{

MessageBox.Show("Текущих клиентов в файле запомнить не удалось!\n" +

"Чтобы их не потерять, создание нового файла отменяется.",

"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

Close();

return;

}

}

// Запомним в реестре имя нового файла.

MyReg.ValueSet(textBox1.Text);

// Новый текущий файл.

Form1.pathCurFile = Path.Combine(form1.folderClients, textBox1.Text);

Form1.nameFile = textBox1.Text;

form1.label3.Text = Form1.nameFile;

if (Режим.mode == 2)

form1.b\_SaveList.Text = Form1.textButton + " в \"" + Form1.nameFile + "\"";

Form1.newFile = true;

Close();

return;

}

}

}

**Файл ListFiles.cs:**

using System;

using System.IO;

using System.Windows.Forms;

namespace Отель

{

public partial class ListFiles : Form

{

public Form1 form1;

public ListFiles()

{

InitializeComponent();

}

//-----------------------------------------------------

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

{

if (Режим.mode == 1) // Ограниченные возможности.

{

CopyButton.Enabled = false;

DeleteButton.Enabled = false;

}

textBox1.Text = Form1.nameFile;

if (!Directory.Exists(form1.folderClients))

Directory.CreateDirectory(form1.folderClients);

// Заполнить список именами всех файлов.

LoadListBox();

}

//-----------------------------------------------------

private void LoadListBox()

{

// Заполнить список именами всех файлов.

listBox1.Items.Clear();

DirectoryInfo di = new DirectoryInfo(form1.folderClients);

FileInfo[] files = di.GetFiles();

if (files.Length == 0)

{

button1.Enabled = false;

CopyButton.Enabled = false;

DeleteButton.Enabled = false;

}

else

{

foreach (var item in files)

listBox1.Items.Add(item.Name);

if (files.Length == 1)

{

listBox1.SelectedIndex = 0;

}

}

}

//-----------------------------------------------------

private void b\_Open\_Click(object sender, EventArgs e)

{

if (listBox1.SelectedIndex == -1)

{

MessageBox.Show("Вы не выделили файл", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

Form1.nameFile = (string)listBox1.SelectedItem;

string currentFile = Path.Combine(form1.folderClients, Form1.nameFile);

if (currentFile == Form1.pathCurFile)

{

MessageBox.Show("Этот файл уже является текущим. Укажите другой файл или закройте окно.",

"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (Режим.mode == 2 && Form1.pathCurFile != "" && Form1.edit)

{

// Сохранить текущий список в текущем файле.

if (!MySerial<ClientClass>.Serialize(Form1.pathCurFile, Form1.list)) // Сохранить текущий список

{

MessageBox.Show("Открыть файл и загрузить список клиентов не удалось.",

"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

Close();

return;

}

}

Form1.pathCurFile = currentFile;

form1.label3.Text = Form1.nameFile;

if (Режим.mode == 2)

form1.b\_SaveList.Text = Form1.textButton + " в \"" + Form1.nameFile + "\"";

if ((Form1.list = MySerial<ClientClass>.Deserialize(Form1.pathCurFile)) == null)

{

MessageBox.Show("Открыть файл и загрузить список клиентов не удалось.",

"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

Form1.pathCurFile = "";

Form1.nameFile = "";

form1.label3.Text = "";

form1.b\_SaveList.Text = Form1.textButton;

Close();

return;

}

form1.clientClassBindingSource.DataSource = Form1.list;

Form1.edit = false;

// Записать в реестр имя выбранного файла в качестве текущего.

Form1.nameFile = (string)listBox1.SelectedItem;

MyReg.ValueSet(Form1.nameFile);

Close();

}

//-----------------------------------------------------

private void b\_Copy\_Click(object sender, EventArgs e)

{

if (listBox1.SelectedIndex == -1)

{

MessageBox.Show("Вы не выделили файл", "Внимание!",

MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

string selectFile = (string)listBox1.SelectedItem;

string path = Path.Combine(Environment.CurrentDirectory, form1.folderClients);

saveFileDialog1.InitialDirectory = path;

saveFileDialog1.Filter = "XML-файл|\*.xml|Все|\*.\*";

saveFileDialog1.Title = "Найдите куда и под каким именем скопировать файл "

+ selectFile;

saveFileDialog1.FileName = selectFile;

if(saveFileDialog1.ShowDialog() != DialogResult.OK)

{

return;

}

try

{

if (saveFileDialog1.FileName != "")

{

path = Path.Combine(path, selectFile);

File.Copy(path, saveFileDialog1.FileName, true);

MessageBox.Show("Файл клиентов: " + selectFile +

" скопирован в\n" + saveFileDialog1.FileName);

LoadListBox();

}

}

catch (Exception ex)

{

MessageBox.Show("Скопировать файл не удалось. \n\n" + ex.Message,

"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

//-----------------------------------------------------

private void b\_Del\_Click(object sender, EventArgs e)

{

if (listBox1.SelectedIndex == -1)

{

MessageBox.Show("Вы не выделили файл", "Внимание!",

MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

string currentFile = Path.Combine(form1.folderClients, (string)listBox1.SelectedItem);

if (currentFile == Form1.pathCurFile)

{

MessageBox.Show("Нельзя удалить текущий файл, записи которого отображаются в таблице. Укажите другой файл или закройте окно.",

"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

try

{

string path = Path.Combine(form1.folderClients, (string)listBox1.SelectedItem);

File.Delete(path);

MessageBox.Show("Файл: " + (string)listBox1.SelectedItem + "\nУДАЛЕН!" ,

"Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Information);

// Обновить список файлов.

LoadListBox();

}

catch (Exception ex)

{

MessageBox.Show("Удалить файл не удалось. \n\n" + ex.Message,

"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

**Файл FormAdd.cs:**

using System;

using System.Linq;

using System.Windows.Forms;

namespace Отель

{

public partial class FormAdd : Form

{

public Form1 form1;

int reservInt;

public FormAdd()

{ InitializeComponent(); }

//--------------------------------------------------

private void Form2\_Load(object sender, EventArgs e)

{

int max = Form1.list.Max(m => m.Reserv);

reservBox.Text = (max + 1).ToString();

reservBox.SelectionStart = 0;

// Избавляемся от времени.

DateTime dt = DateTime.Now;

dateTimePicker1.Value = new DateTime(dt.Year, dt.Month, dt.Day);

dateTimePicker2.Value = dateTimePicker1.Value;

}

//--------------------------------------------------

private void b\_Add\_Click(object sender, EventArgs e)

{

// Проверяем поле № Брони (reserv).

if (string.IsNullOrEmpty(reservBox.Text))

{

MessageBox.Show("Не задан номер брони. Введите.", "ОШИБКА");

return;

}

if (!int.TryParse(reservBox.Text, out reservInt))

{

MessageBox.Show("Номер брони должен иметь цифровой формат. Исправьте.",

"ОШИБКА");

return;

}

// Проверяем поле № комнаты (room).

if (string.IsNullOrEmpty(roomBox.Text))

{

MessageBox.Show("Не задан номер комнаты. Введите.", "ОШИБКА");

return;

}

int roomInt;

if (!int.TryParse(roomBox.Text, out roomInt))

{

MessageBox.Show("Номер комнаты должен иметь цифровой формат. Исправьте.",

"ОШИБКА");

return;

}

// Проверяем поле Фамилия.

if (string.IsNullOrEmpty(surnameBox.Text))

{

MessageBox.Show("Не введена фамилия. Введите.", "ОШИБКА");

return;

}

// Проверяем поле Имя.

if (string.IsNullOrEmpty(nameBox.Text))

{

MessageBox.Show("Не введено имя. Введите.", "ОШИБКА");

return;

}

// Проверяем поля даты.

if (dateTimePicker1.Value > dateTimePicker2.Value)

{

MessageBox.Show("Дата приезда не может быть позже даты отъезда",

"ОШИБКА");

return;

}

Form1.list.Add(new ClientClass(reservInt, roomInt, surnameBox.Text, nameBox.Text, dateTimePicker1.Value,

dateTimePicker2.Value, phoneBox.Text, checkBoxVIP.Checked, checkBoxNON.Checked));

MessageBox.Show("Клиент " + surnameBox.Text + " добавлен.");

Form1.edit = true;

}

private void b\_Cleare\_Click(object sender, EventArgs e)

{

reservBox.Text = (reservInt + 1).ToString();

reservBox.SelectionStart = 0;

roomBox.Text = "";

surnameBox.Text = "";

nameBox.Text = "";

phoneBox.Text = "";

checkBoxVIP.Checked = false;

checkBoxNON.Checked = false;

}

//--------------------------------------------------

private void b\_Cancel\_Click(object sender, EventArgs e)

{

Close();

}

}

}

**Файл FormEdit.cs:**

using System;

using System.Windows.Forms;

namespace Отель

{

public partial class FormEdit : Form

{

public Form1 form1;

int reservInt;

DataGridViewCellCollection cells;

int i;

public FormEdit()

{ InitializeComponent(); }

//--------------------------------------------------

private void Form3\_Load(object sender, EventArgs e)

{

cells = form1.dataGridView1.CurrentRow.Cells;

i = form1.dataGridView1.CurrentRow.Index;

reservBox.Text = cells["reserv"].Value.ToString();

roomBox.Text = cells["room"].Value.ToString();

surnameBox.Text = (string)cells["surnam"].Value;

nameBox.Text = (string)cells["name"].Value;

phoneBox.Text = (string)cells["telefon"].Value;

checkBoxVIP.Checked = (bool)cells["vIP"].Value;

checkBoxNON.Checked = (bool)cells["nonGrata"].Value;

dateTimePicker1.Value = (DateTime)cells["checkInDate"].Value;

dateTimePicker2.Value = (DateTime)cells["checkOutDate"].Value;

}

//--------------------------------------------------

private void b\_Edit\_Click(object sender, EventArgs e)

{

// Проверяем поле № Брони (reserv).

if (string.IsNullOrEmpty(reservBox.Text))

{

MessageBox.Show("Не задан номер брони. Введите.", "ОШИБКА");

return;

}

if (!int.TryParse(reservBox.Text, out reservInt))

{

MessageBox.Show("Номер брони должен иметь цифровой формат. Исправьте.",

"ОШИБКА");

return;

}

// Проверяем поле № комнаты (room).

if (string.IsNullOrEmpty(roomBox.Text))

{

MessageBox.Show("Не задан номер комнаты. Введите.", "ОШИБКА");

return;

}

int roomInt;

if (!int.TryParse(roomBox.Text, out roomInt))

{

MessageBox.Show("Номер комнаты должен иметь цифровой формат. Исправьте.",

"ОШИБКА");

return;

}

// Проверяем поле Фамилия.

if (string.IsNullOrEmpty(surnameBox.Text))

{

MessageBox.Show("Не введена фамилия. Введите.", "ОШИБКА");

return;

}

// Проверяем поле Имя.

if (string.IsNullOrEmpty(nameBox.Text))

{

MessageBox.Show("Не введено имя. Введите.", "ОШИБКА");

return;

}

// Проверяем поля даты.

if (dateTimePicker1.Value > dateTimePicker2.Value)

{

MessageBox.Show("Дата приезда не может быть позже даты отъезда",

"ОШИБКА");

return;

}

// Изменяем объект списка list.

Form1.list[i].Reserv = int.Parse(reservBox.Text);

Form1.list[i].Room = int.Parse(roomBox.Text);

Form1.list[i].Surname = surnameBox.Text;

Form1.list[i].Name = nameBox.Text;

Form1.list[i].Telefon = phoneBox.Text;

Form1.list[i].VIP = checkBoxVIP.Checked;

Form1.list[i].NonGrata = checkBoxNON.Checked;

Form1.list[i].CheckInDate = dateTimePicker1.Value;

Form1.list[i].CheckOutDate = dateTimePicker2.Value;

// Изменяем строку таблицы.

//cells["reserv"].Value = reservBox.Text;

//cells["room"].Value = roomBox.Text;

//cells["surnam"].Value = surnameBox.Text;

//cells["name"].Value = nameBox.Text;

//cells["telefon"].Value = phoneBox.Text;

//cells["vIP"].Value = checkBoxVIP.Checked;

//cells["nonGrata"].Value = checkBoxNON.Checked;

//cells["checkInDate"].Value = dateTimePicker1.Value;

//cells["checkOutDate"].Value = dateTimePicker2.Value;

MessageBox.Show("Клиент " + surnameBox.Text + " отредактирован.");

Form1.edit = true;

form1.clientClassBindingSource.ResetBindings(false);

Close();

}

//--------------------------------------------------

private void b\_Cancel\_Click(object sender, EventArgs e)

{

Close();

}

}

}

**Файл MyReg.cs:**

using Microsoft.Win32;

using System;

using System.Windows.Forms;

namespace Отель

{

// Класс содержит три метода для работы с системным реестром.

public class MyReg

{

public static string subKeyName = "Hotel"; // Имя подраздела.

public static string valueName = "File"; // Имя значения (ключ).

static RegistryKey localUser = Registry.CurrentUser; // Раздел реестра

static RegistryKey subKey; // Ссылка для подраздела реестра.

//---------------------------------------------------------

// Метод читает значение по ключу File в подразделе Hotel.

// Возвращаемый результат:

// "значение", если значение прочитано;

// "", если подраздела или значения нет или ошибка.

public static string ValueGet()

{

try

{

subKey = localUser.OpenSubKey(subKeyName);

if (subKey != null)

{

object obj = subKey.GetValue(valueName);

if (obj != null) // Параметр сохранен.

{

subKey.Close();

return (string)obj;

}

}

}

catch (Exception ex)

{

MessageBox.Show("Получить имя из реестра не удалось. \n\n" + ex.Message,

"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

return "";

}

//---------------------------------------------------------

// Метод записывает значение по ключу File в подразделе Hotel.

// Параметр - имя файла без пути.

// Возвращаемый результат:

// true - успешно записано;

// false - не записано из-за ошибки.

public static bool ValueSet(string value)

{

try

{

// Записать в реестр имя выбранного файла в качестве текущего.

subKey = localUser.OpenSubKey(subKeyName, true);

if (subKey == null)

subKey = localUser.CreateSubKey(subKeyName);

subKey.SetValue(valueName, value);

subKey.Close();

return true;

}

catch (Exception ex)

{

MessageBox.Show("Записать имя в реестр не удалось. \n\n" + ex.Message,

"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

}

//---------------------------------------------------------

// Метод удаляет значение по ключу File в подразделе Hotel.

// Возвращаемый результат:

// true - успешно удалили подраздел Hotel и значение File;

// false - не удалили из-за ошибки.

public static bool SubKeyDel()

{

try

{

// Удалить из реестра подраздел.

subKey = localUser.OpenSubKey(subKeyName, true);

subKey.DeleteSubKey("");

subKey.Close();

return true;

}

catch (Exception ex)

{

MessageBox.Show("Не удалось удалить из реестра подраздел 'Hotel'. \n\n"

+ ex.Message, "Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

subKey.Close();

return false;

}

}

}

}

**Файл MySerial.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.IO;

using System.Windows.Forms;

using System.Xml.Serialization;

namespace Отель

{

// Класс содержит два метода: сериализации и десериализации

// В вызовах методам передаются статические поля класса Form1:

// - Form1.pathCurFile - относительный путь к файлу:

// - Form1.list - список объектов с данными.

// - тип данных ClientClass.

class **MySerial<T>**

{

static XmlSerializer xs = new XmlSerializer(typeof(List<T>));

static FileStream fs = null;

//-------------------------------------------------------------

// Метод СЕРИАЛИЗУЕТ список объектов типа Т (сохраняет в XML-файле).

// параметры: T - тип класса предметной области;

// file - выходной XML-файл;

// list - коллекция типа List<T>.

// Возвращаемое значение: true - норма, false - не сериализовано.

public static bool Serialize(string file, List<T> list)

{

try

{

fs = new FileStream(file, FileMode.Create, FileAccess.Write,

FileShare.ReadWrite);

// Сохраним список в XML-файле

xs.Serialize(fs, list);

fs.Close();

return true;

}

catch (Exception ex)

{

MessageBox.Show("Ошибка XML-сериализации списка в файл. \n\n" + ex.Message,

"Ошибка вывода", MessageBoxButtons.OK, MessageBoxIcon.Error);

if (fs != null)

fs.Close();

return false;

}

}

//-------------------------------------------------------------

// Метод ДЕСЕРИАЛИЗУЕТ список объектов типа Т (читает из XML-файла).

// параметры: T - тип класса предметной области;

// file - исходный XML-файл.

// Возвращаемое значение: список - норма, null - не десериализовано

public static List<T> Deserialize(string file)

{

List<T> list = null;

try

{

fs = new FileStream(file, FileMode.Open, FileAccess.Read, FileShare.ReadWrite);

list = (List<T>)xs.Deserialize(fs);

fs.Close();

return list;

}

catch (Exception ex)

{

MessageBox.Show("Ошибка XML-десериализации списка из файл. \n\n" +

ex.Message,

"Ошибка чтения", MessageBoxButtons.OK, MessageBoxIcon.Error);

if (fs != null)

fs.Close();

return null;

}

}

}

}