**Структура работы:**

1. Описание проекта………………………………………...2
2. Скриншоты рабочих областей………………………….3
3. Код .xml-документов…………………………………….4
4. Исполняемый код………………………………………...6
5. Использованные ресурсы………………………………13
6. **Описание работы.**

Я сформировал навыки работы с документами в информационных системах и приложениях, рассмотрел вопросы структурированного хранения документов (базы данных, xml-файлы), организации пользовательского интерфейса для доступа к ним, создание печатных форм документов и внедрение средств защиты информации для документов (цифровая подпись, аутентификация пользователей и пр.).

В результате практикума я выполнил сквозной проект на языке программирования C#. Проект включает в себя создание небольшой подсистемы работы с документами для некоторой предметной области (магазина).

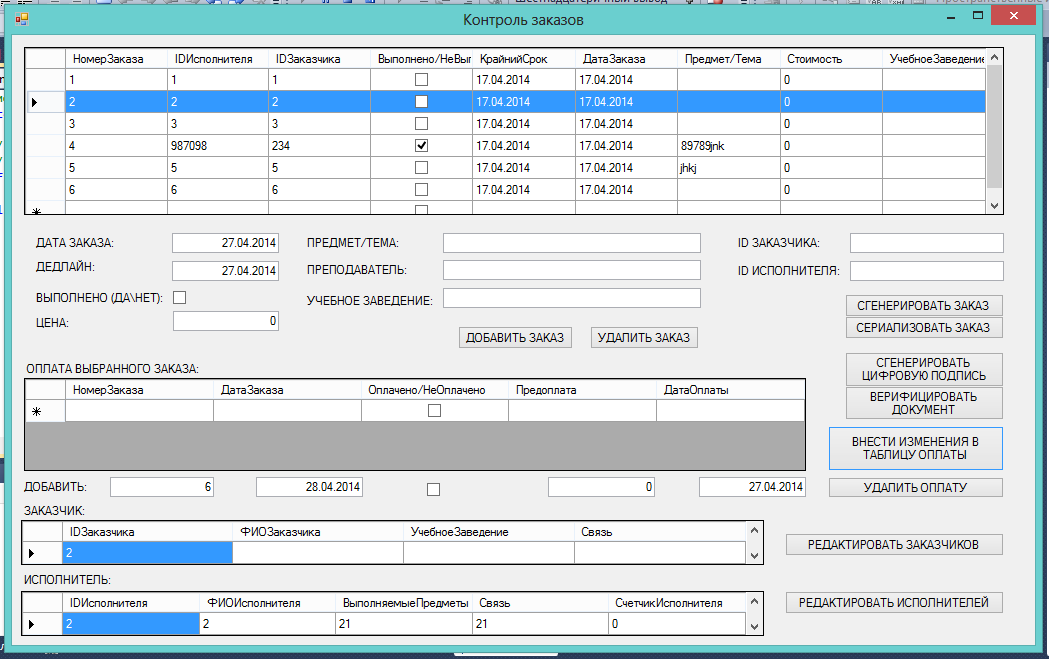
Проект включает в себя создание двух документов – «Формирование заказа» и «Оплата заказа». Заказ - заявка, оставленная заказчиком . В процессе работы была реализована интерактивная среда взаимодействия (добавления/удаления заказов, заказчиков и исполнителей, вывода заказов в отдельный Word(doc)-файл, верификация заказов). Представленные ниже скриншоты и код файлов с комментариями показывают основу работы программы.

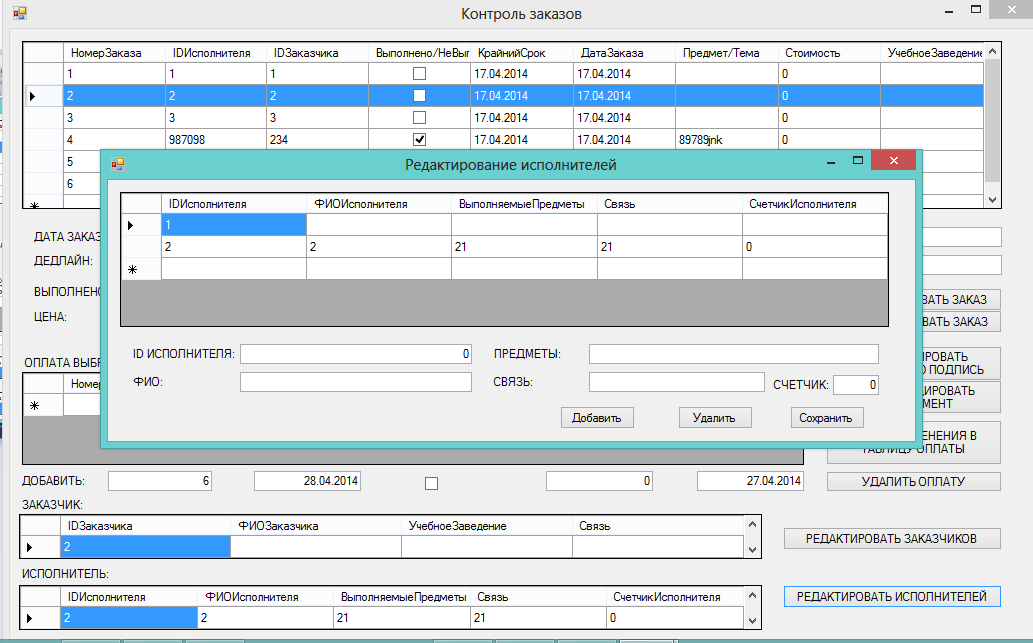
bd.xml – основной файл базы данных.

zaks.xml – база данных заказчиков.

isps.xml – база данных исполнителей.

1. **Скриншоты рабочих областей.**





1. **Код xml-документов.**

bd.xml

<?xml version="1.0" standalone="yes"?>

<NewDataSet>

<xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">

<xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:UseCurrentLocale="true">

<xs:complexType>

<xs:choice minOccurs="0" maxOccurs="unbounded">

<xs:element name="Заказы">

<xs:complexType>

<xs:sequence>

<xs:element name="НомерЗаказа" type="xs:int" />

<xs:element name="IDИсполнителя" type="xs:int" minOccurs="0" />

<xs:element name="IDЗаказчика" type="xs:int" minOccurs="0" />

<xs:element name="Выполнено\_x002F\_НеВыполнено" type="xs:boolean" minOccurs="0" />

<xs:element name="КрайнийСрок" type="xs:dateTime" minOccurs="0" />

<xs:element name="ДатаЗаказа" type="xs:dateTime" />

<xs:element name="Предмет\_x002F\_Тема" type="xs:string" minOccurs="0" />

<xs:element name="Стоимость" type="xs:int" minOccurs="0" />

<xs:element name="УчебноеЗаведение" type="xs:string" minOccurs="0" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="ЗаписьОплаты">

<xs:complexType>

<xs:sequence>

<xs:element name="НомерЗаказа" type="xs:int" />

<xs:element name="ДатаЗаказа" type="xs:dateTime" />

<xs:element name="Оплачено\_x002F\_НеОплачено" type="xs:boolean" minOccurs="0" />

<xs:element name="Предоплата" type="xs:int" minOccurs="0" />

<xs:element name="ДатаОплаты" type="xs:dateTime" minOccurs="0" />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:choice>

</xs:complexType>

<xs:unique name="Constraint1" msdata:PrimaryKey="true">

<xs:selector xpath=".//Заказы" />

<xs:field xpath="НомерЗаказа" />

<xs:field xpath="ДатаЗаказа" />

</xs:unique>

<xs:unique name="ЗаписьОплаты\_Constraint1" msdata:ConstraintName="Constraint1" msdata:PrimaryKey="true">

<xs:selector xpath=".//ЗаписьОплаты" />

<xs:field xpath="НомерЗаказа" />

<xs:field xpath="ДатаЗаказа" />

</xs:unique>

<xs:keyref name="СвязьЗаказаСОплатойЗаказа" refer="Constraint1">

<xs:selector xpath=".//ЗаписьОплаты" />

<xs:field xpath="НомерЗаказа" />

<xs:field xpath="ДатаЗаказа" />

</xs:keyref>

</xs:element>

</xs:schema>

<Заказы>

<НомерЗаказа>1</НомерЗаказа>

<IDИсполнителя>1</IDИсполнителя>

<IDЗаказчика>1</IDЗаказчика>

<Выполнено\_x002F\_НеВыполнено>false</Выполнено\_x002F\_НеВыполнено>

<КрайнийСрок>2014-04-17T00:00:00+04:00</КрайнийСрок>

<ДатаЗаказа>2014-04-17T00:00:00+04:00</ДатаЗаказа>

<Предмет\_x002F\_Тема />

<Стоимость>0</Стоимость>

<УчебноеЗаведение />

</Заказы>

<Заказы>

<НомерЗаказа>2</НомерЗаказа>

<IDИсполнителя>2</IDИсполнителя>

<IDЗаказчика>2</IDЗаказчика>

<Выполнено\_x002F\_НеВыполнено>false</Выполнено\_x002F\_НеВыполнено>

<КрайнийСрок>2014-04-17T00:00:00+04:00</КрайнийСрок>

<ДатаЗаказа>2014-04-17T00:00:00+04:00</ДатаЗаказа>

<Предмет\_x002F\_Тема />

<Стоимость>0</Стоимость>

<УчебноеЗаведение />

</Заказы>

<Заказы>

<НомерЗаказа>3</НомерЗаказа>

<IDИсполнителя>3</IDИсполнителя>

<IDЗаказчика>3</IDЗаказчика>

<Выполнено\_x002F\_НеВыполнено>false</Выполнено\_x002F\_НеВыполнено>

<КрайнийСрок>2014-04-17T00:00:00+04:00</КрайнийСрок>

<ДатаЗаказа>2014-04-17T00:00:00+04:00</ДатаЗаказа>

<Предмет\_x002F\_Тема />

<Стоимость>0</Стоимость>

<УчебноеЗаведение />

</Заказы>

<Заказы>

<НомерЗаказа>4</НомерЗаказа>

<IDИсполнителя>987098</IDИсполнителя>

<IDЗаказчика>234</IDЗаказчика>

<Выполнено\_x002F\_НеВыполнено>true</Выполнено\_x002F\_НеВыполнено>

<КрайнийСрок>2014-04-17T00:00:00+04:00</КрайнийСрок>

<ДатаЗаказа>2014-04-17T00:00:00+04:00</ДатаЗаказа>

<Предмет\_x002F\_Тема>89789jnk</Предмет\_x002F\_Тема>

<Стоимость>0</Стоимость>

<УчебноеЗаведение />

</Заказы>

<Заказы>

<НомерЗаказа>5</НомерЗаказа>

<IDИсполнителя>5</IDИсполнителя>

<IDЗаказчика>5</IDЗаказчика>

<Выполнено\_x002F\_НеВыполнено>false</Выполнено\_x002F\_НеВыполнено>

<КрайнийСрок>2014-04-17T00:00:00+04:00</КрайнийСрок>

<ДатаЗаказа>2014-04-17T00:00:00+04:00</ДатаЗаказа>

<Предмет\_x002F\_Тема>jhkj</Предмет\_x002F\_Тема>

<Стоимость>0</Стоимость>

<УчебноеЗаведение />

</Заказы>

<Заказы>

<НомерЗаказа>6</НомерЗаказа>

<IDИсполнителя>6</IDИсполнителя>

<IDЗаказчика>6</IDЗаказчика>

<Выполнено\_x002F\_НеВыполнено>false</Выполнено\_x002F\_НеВыполнено>

<КрайнийСрок>2014-04-17T00:00:00+04:00</КрайнийСрок>

<ДатаЗаказа>2014-04-17T00:00:00+04:00</ДатаЗаказа>

<Предмет\_x002F\_Тема />

<Стоимость>0</Стоимость>

<УчебноеЗаведение />

</Заказы>

</NewDataSet>

isp.xml

<?xml version="1.0" standalone="yes"?>

<NewDataSet>

<xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">

<xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:UseCurrentLocale="true">

<xs:complexType>

<xs:choice minOccurs="0" maxOccurs="unbounded">

<xs:element name="Исполнители">

<xs:complexType>

<xs:sequence>

<xs:element name="IDИсполнителя" type="xs:int" />

<xs:element name="ФИОИсполнителя" type="xs:string" minOccurs="0" />

<xs:element name="ВыполняемыеПредметы" type="xs:string" minOccurs="0" />

<xs:element name="Связь" type="xs:string" minOccurs="0" />

<xs:element name="СчетчикИсполнителя" type="xs:int" minOccurs="0" />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:choice>

</xs:complexType>

<xs:unique name="Constraint1" msdata:PrimaryKey="true">

<xs:selector xpath=".//Исполнители" />

<xs:field xpath="IDИсполнителя" />

</xs:unique>

</xs:element>

</xs:schema>

<Исполнители>

<IDИсполнителя>1</IDИсполнителя>

</Исполнители>

<Исполнители>

<IDИсполнителя>2</IDИсполнителя>

<ФИОИсполнителя>2</ФИОИсполнителя>

<ВыполняемыеПредметы>21</ВыполняемыеПредметы>

<Связь>21</Связь>

<СчетчикИсполнителя>0</СчетчикИсполнителя>

</Исполнители>

</NewDataSet>

zaks.xml

<?xml version="1.0" standalone="yes"?>

<NewDataSet>

<xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">

<xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:UseCurrentLocale="true">

<xs:complexType>

<xs:choice minOccurs="0" maxOccurs="unbounded">

<xs:element name="Заказчики">

<xs:complexType>

<xs:sequence>

<xs:element name="IDЗаказчика" type="xs:int" />

<xs:element name="ФИОЗаказчика" type="xs:string" minOccurs="0" />

<xs:element name="УчебноеЗаведение" type="xs:string" minOccurs="0" />

<xs:element name="Связь" type="xs:string" minOccurs="0" />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:choice>

</xs:complexType>

<xs:unique name="Constraint1" msdata:PrimaryKey="true">

<xs:selector xpath=".//Заказчики" />

<xs:field xpath="IDЗаказчика" />

</xs:unique>

</xs:element>

</xs:schema>

<Заказчики>

<IDЗаказчика>1</IDЗаказчика>

<ФИОЗаказчика />

<УчебноеЗаведение />

<Связь />

</Заказчики>

<Заказчики>

<IDЗаказчика>2</IDЗаказчика>

<ФИОЗаказчика />

<УчебноеЗаведение />

<Связь />

</Заказчики>

<Заказчики>

<IDЗаказчика>3</IDЗаказчика>

<ФИОЗаказчика />

<УчебноеЗаведение />

<Связь />

</Заказчики>

<Заказчики>

<IDЗаказчика>4</IDЗаказчика>

<ФИОЗаказчика />

<УчебноеЗаведение />

<Связь />

</Заказчики>

</NewDataSet>

1. Исполняемый код.

using System;

using System.IO;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Collections;

//Ссылка на создание объекта Office

using Word = Microsoft.Office.Interop.Word;

//Для сериализации

using System.Runtime.Serialization.Formatters.Binary;

//Для создания цифровой подписи

using System.Security.Cryptography;

namespace OurProject\_Part1

{

public partial class Form1 : Form

{

DataSet ds = new DataSet();//Для таблицы Заказов и ЗаписейОплаты

DataSet zaks = new DataSet();//Для таблицы Заказчиков

DataSet isps = new DataSet();//Для таблицы Исполнителей

int nom;//Номер текущего заказа

DateTime date;//Дата текущего заказа

public Form1()

{

InitializeComponent();

textBox2.Text = DateTime.Now.ToShortDateString();//Дата заказа

textBox3.Text = "0";//Цена

textBox4.Text = DateTime.Now.ToShortDateString();//Дедлайн заказа

textBox13.Text = DateTime.Now.ToShortDateString();//Дата оплаты

textBox7.Text = DateTime.Now.ToShortDateString();//Дата заказа

//Проверяем наличие файлов

if (!File.Exists("bd.xml"))

CreateBD(); ;

if (!File.Exists("isps.xml"))

CreateI(); ;

if (!File.Exists("zaks.xml"))

CreateZ(); ;

// считываем информацию из файлов с данными

ds.ReadXml("bd.xml", XmlReadMode.ReadSchema);

zaks.ReadXml("zaks.xml", XmlReadMode.ReadSchema);

isps.ReadXml("isps.xml", XmlReadMode.ReadSchema);

// установка источника данных для DataGridView

dataGridView1.DataSource = ds.Tables["Заказы"];

dataGridView4.DataSource = ds.Tables["ЗаписьОплаты"];

// отмена генерации столбцов DataGridView2

dataGridView2.AutoGenerateColumns = false;

// заполнение структуры таблицы заказа для dataGridView2

// из столбцов таблицы набора данных

// последовательное создание столбцов элемента управления

foreach (DataColumn dc in zaks.Tables["Заказчики"].Columns)

{

// создание нового столбца

DataGridViewTextBoxColumn dgvc = new DataGridViewTextBoxColumn();

// установка заголовка столбца из столбца таблицы

dgvc.HeaderText = dc.Caption;

// добавление столбца в коллекцию столбцов DataGridView

dataGridView2.Columns.Add(dgvc);

}

// отмена генерации столбцов DataGridView3

dataGridView3.AutoGenerateColumns = false;

// заполнение структуры таблицы исполнителей для dataGridView3

// из столбцов таблицы набора данных

// последовательное создание столбцов элемента управления

foreach (DataColumn dc in isps.Tables["Исполнители"].Columns)

{

// создание нового столбца

DataGridViewTextBoxColumn dgvc = new DataGridViewTextBoxColumn();

// установка заголовка столбца из столбца таблицы

dgvc.HeaderText = dc.Caption;

// добавление столбца в коллекцию столбцов DataGridView

dataGridView3.Columns.Add(dgvc);

}

}

//База данных заказов и оплаты заказов

public static void CreateBD()

{

// создание набора данных

DataSet ds = new DataSet();

DataColumn dc;

// создаем таблицу заказов

// добавляем таблицу в список таблиц набора данных

ds.Tables.Add(new DataTable("Заказы"));

// атрибут

dc = new DataColumn("НомерЗаказа", Type.GetType("System.Int32"));

ds.Tables["Заказы"].Columns.Add(dc);

// атрибут

dc = new DataColumn("IDИсполнителя", Type.GetType("System.Int32"));

ds.Tables["Заказы"].Columns.Add(dc);

// атрибут

dc = new DataColumn("IDЗаказчика", Type.GetType("System.Int32"));

ds.Tables["Заказы"].Columns.Add(dc);

// атрибут

dc = new DataColumn("Выполнено/НеВыполнено", Type.GetType("System.Boolean"));

ds.Tables["Заказы"].Columns.Add(dc);

// атрибут

dc = new DataColumn("КрайнийСрок", Type.GetType("System.DateTime"));

ds.Tables["Заказы"].Columns.Add(dc);

// атрибут

dc = new DataColumn("ДатаЗаказа", Type.GetType("System.DateTime"));

ds.Tables["Заказы"].Columns.Add(dc);

// атрибут

dc = new DataColumn("Предмет/Тема", Type.GetType("System.String"));

ds.Tables["Заказы"].Columns.Add(dc);

// атрибут

dc = new DataColumn("Стоимость", Type.GetType("System.Int32"));

ds.Tables["Заказы"].Columns.Add(dc);

// атрибут

dc = new DataColumn("УчебноеЗаведение", Type.GetType("System.String"));

ds.Tables["Заказы"].Columns.Add(dc);

// описание первичного ключа - массива ссылок на столбцы таблицы

// первичным ключом будет номер

DataColumn[] key = new DataColumn[2] {ds.Tables["Заказы"].Columns["НомерЗаказа"], ds.Tables["Заказы"].Columns["ДатаЗаказа"] };//

ds.Tables["Заказы"].PrimaryKey = key;

// создаем таблицу заказов

// добавляем таблицу в список таблиц набора данных

ds.Tables.Add(new DataTable("ЗаписьОплаты"));

// атрибут

dc = new DataColumn("НомерЗаказа", Type.GetType("System.Int32"));

ds.Tables["ЗаписьОплаты"].Columns.Add(dc);

// атрибут

dc = new DataColumn("ДатаЗаказа", Type.GetType("System.DateTime"));

ds.Tables["ЗаписьОплаты"].Columns.Add(dc);

// атрибут

dc = new DataColumn("Оплачено/НеОплачено", Type.GetType("System.Boolean"));

ds.Tables["ЗаписьОплаты"].Columns.Add(dc);

// атрибут

dc = new DataColumn("Предоплата", Type.GetType("System.Int32"));

ds.Tables["ЗаписьОплаты"].Columns.Add(dc);

// атрибут

dc = new DataColumn("ДатаОплаты", Type.GetType("System.DateTime"));

ds.Tables["ЗаписьОплаты"].Columns.Add(dc);

// описание первичного ключа - массива ссылок на столбцы таблицы

// первичным ключом будет номер

key = new DataColumn[2] { ds.Tables["ЗаписьОплаты"].Columns["НомерЗаказа"], ds.Tables["ЗаписьОплаты"].Columns["ДатаЗаказа"] };//

ds.Tables["ЗаписьОплаты"].PrimaryKey = key;

// создание связи между таблицами

// указывается имя отношения и два массива связанных полей - для родительской и дочерних таблиц

DataRelation rel = new DataRelation("СвязьЗаказаСОплатойЗаказа",

new DataColumn[] { ds.Tables["Заказы"].Columns["НомерЗаказа"], ds.Tables["Заказы"].Columns["ДатаЗаказа"] },

new DataColumn[] { ds.Tables["ЗаписьОплаты"].Columns["НомерЗаказа"], ds.Tables["ЗаписьОплаты"].Columns["ДатаЗаказа"] });

//добавляем связь в список связей набора данных

ds.Relations.Add(rel);

ds.WriteXml("bd.xml", XmlWriteMode.WriteSchema);

}

public static void CreateI()

{

// создание набора данных

DataSet ds = new DataSet();

DataColumn dc;

// создаем таблицу исполнителей

// добавляем таблицу в список таблиц набора данных

ds.Tables.Add(new DataTable("Исполнители"));

// атрибут

dc = new DataColumn("IDИсполнителя", Type.GetType("System.Int32"));

ds.Tables["Исполнители"].Columns.Add(dc);

// атрибут

dc = new DataColumn("ФИОИсполнителя", Type.GetType("System.String"));

ds.Tables["Исполнители"].Columns.Add(dc);

// атрибут

dc = new DataColumn("ВыполняемыеПредметы", Type.GetType("System.String"));

ds.Tables["Исполнители"].Columns.Add(dc);

// атрибут

dc = new DataColumn("Связь", Type.GetType("System.String"));

ds.Tables["Исполнители"].Columns.Add(dc);

// атрибут

dc = new DataColumn("СчетчикИсполнителя", Type.GetType("System.Int32"));

ds.Tables["Исполнители"].Columns.Add(dc);

// описание первичного ключа - массива ссылок на столбцы таблицы

// первичным ключом будет номер

DataColumn[] key = new DataColumn[1] { ds.Tables["Исполнители"].Columns["IDИсполнителя"] };

ds.Tables["Исполнители"].PrimaryKey = key;

ds.WriteXml("isps.xml", XmlWriteMode.WriteSchema);

}

public static void CreateZ()

{

// создание набора данных

DataSet ds = new DataSet();

// создаем таблицу заказов

DataTable customers = new DataTable("Заказчики");

// добавляем таблицу в список таблиц набора данных

ds.Tables.Add(customers);

// формируем список столбцов таблицы заказчиков

// для каждого столбца указывается имя столбца и тип данных

DataColumn dc = new DataColumn("IDЗаказчика", Type.GetType("System.Int32"));

ds.Tables["Заказчики"].Columns.Add(dc);

// атрибут

dc = new DataColumn("ФИОЗаказчика", Type.GetType("System.String"));

ds.Tables["Заказчики"].Columns.Add(dc);

// атрибут

dc = new DataColumn("УчебноеЗаведение", Type.GetType("System.String"));

ds.Tables["Заказчики"].Columns.Add(dc);

// атрибут

dc = new DataColumn("Связь", Type.GetType("System.String"));

ds.Tables["Заказчики"].Columns.Add(dc);

// описание первичного ключа - массива ссылок на столбцы таблицы

// первичным ключом будет комбинация номера и даты

DataColumn[] key = new DataColumn[1] { ds.Tables["Заказчики"].Columns["IDЗаказчика"] };

ds.Tables["Заказчики"].PrimaryKey = key;

ds.WriteXml("zaks.xml", XmlWriteMode.WriteSchema);

}

//Выборка данных при клике на строчку в таблице "Заказы"

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

{

//Выводим информацию о заказах по заказчику

// сохранение номера выбранного заказа из ячейки строки

// номер строки приходит в качестве параметра

if (dataGridView1.RowCount > 1)

{

nom = (int)dataGridView1.Rows[e.RowIndex].Cells["IDЗаказчика"].Value;

ZagrZak(nom);

nom = (int)dataGridView1.Rows[e.RowIndex].Cells["IDИсполнителя"].Value;

ZagrIsp(nom);

}

}

//Загрузка таблицы исполнителей

private void ZagrIsp(int nomer)

{

try

{

// очистка DataGridView, т.к. изменяется

dataGridView3.Rows.Clear();

// выбор выбранного заказа и заполнение второго DataGridView

DataRow[] drs3 = isps.Tables["Исполнители"].Select("IDИсполнителя=" + nomer);

// последовательное заполнение информации о найденных записях

// в DataGridView для записей заказов

foreach (DataRow dr in drs3)

{

DataGridViewRow dgvr = new DataGridViewRow();

dgvr.CreateCells(dataGridView3,dr["IDИсполнителя"], dr["ФИОИсполнителя"], dr["ВыполняемыеПредметы"], dr["Связь"], dr["СчетчикИсполнителя"]);

dataGridView3.Rows.Add(dgvr);

}

}

catch

{

MessageBox.Show("Вы не ввели данные об исполнителе заказа! Данные отсутствуют");

}

}

//Загрузка таблицы заказчиков

private void ZagrZak(int nomer)

{

try

{

// очистка DataGridView, т.к. изменяется

dataGridView2.Rows.Clear();

// выбор выбранного заказа и заполнение второго DataGridView

DataRow[] drs2 = zaks.Tables["Заказчики"].Select("IDЗаказчика=" + nomer);

// последовательное заполнение информации о найденных записях

// в DataGridView для записей заказов

foreach (DataRow dr in drs2)

{

DataGridViewRow dgvr = new DataGridViewRow();

dgvr.CreateCells(dataGridView2, dr["IDЗаказчика"], dr["ФИОЗаказчика"], dr["УчебноеЗаведение"], dr["Связь"]);

dataGridView2.Rows.Add(dgvr);

}

}

catch

{

MessageBox.Show("Вы не ввели данные о заказчике! Данные отсутствуют");

}

}

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

{

ds.WriteXml("bd.xml", XmlWriteMode.WriteSchema);

isps.WriteXml("isps.xml", XmlWriteMode.WriteSchema);

zaks.WriteXml("zaks.xml", XmlWriteMode.WriteSchema);

}

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

{

// если заказ не был выбран, удалять нечего

if (dataGridView1.SelectedRows.Count == 0)

return;

//получение номера текущего выбранного заказа

nom = (int)dataGridView1.SelectedRows[0].Cells["НомерЗаказа"].Value;

DateTime date = (DateTime)dataGridView1.SelectedRows[0].Cells["ДатаЗаказа"].Value;

// выбираем все записи, соответствующие текущему заказу

DataRow[] drs = ds.Tables["Заказы"].Select("НомерЗаказа=" + nom + " and ДатаЗаказа='"+date+"'");

// удаляем все найденные заказы из таблиц набора данных

for (int i = drs.Length - 1; i >= 0; i--)

drs[i].Delete();

// поиск заказа по ключу

//ds.Tables["Заказы"].Rows.Find(new object[] {(object)nom,(object)idIsp,(object)idZ}).Delete();

// удаление заказа из набора данных

// DataGridView заказов обновиться автоматически

// очистка DataGridView2 и DataGridView3 с записями о заказе(заказчики и исполнители) – т.к. выбранного заказа уже нет

dataGridView2.Rows.Clear();

dataGridView3.Rows.Clear();

}

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

{

// генерация данных о заказе

// создание нового заказа

DataRow newrow = ds.Tables["Заказы"].NewRow();

// заполнение атрибутов заказа

// для определения номера заказа можно узнать

// количество строк в таблице заказов

newrow["НомерЗаказа"] = ds.Tables["Заказы"].Rows.Count + 1;

nom = (int)newrow["НомерЗаказа"];

newrow["ДатаЗаказа"] = DateTime.Parse(textBox2.Text.ToString());

if (textBox9.Text.ToString().CompareTo("")==0)

newrow["IDИсполнителя"]=nom;

else

newrow["IDИсполнителя"] = int.Parse(textBox9.Text.ToString());

if (textBox10.Text.ToString().CompareTo("") == 0)

newrow["IDЗаказчика"] = nom;

else

newrow["IDЗаказчика"] = int.Parse(textBox10.Text.ToString());

newrow["Выполнено/НеВыполнено"] = Boolean.Parse(checkBox1.Checked.ToString());

newrow["КрайнийСрок"] = DateTime.Parse(textBox4.Text.ToString());

newrow["Предмет/Тема"] = (string)textBox8.Text;

newrow["Стоимость"] = int.Parse(textBox3.Text.ToString());

newrow["УчебноеЗаведение"] = (string)textBox5.Text;

// запоминаем дату и номер заказа

nom = (int)newrow["НомерЗаказа"];

// записываем созданную запись в таблицу

ds.Tables["Заказы"].Rows.Add(newrow);

// очищаем DataGridView для исполнителей и заказчиков – т.к. создается новый заказ,

// у которого еще нет записей

dataGridView2.Rows.Clear();

dataGridView3.Rows.Clear();

// работа с DataGridView, показываеющей таблицу заказов

// отмена выделения всех выбранных строк в DataGridView

foreach (DataGridViewRow dgvr in dataGridView1.SelectedRows)

dgvr.Selected = false;

// установка выбора вновь созданного заказа

// последняя строка DataGridView – это строка для ручного ввода новой

// записи, поэтому последняя значимая строка – предпоследняя

dataGridView1.Rows[dataGridView1.Rows.Count - 2].Selected = true;

}

//Редактирование заказчиков

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

{

Form2 zaks=new Form2(ref this.zaks);

if (zaks.Visible == false)

{

zaks.Show();

}

}

//Редактирование исполнителей

private void button3\_Click(object sender, EventArgs e)

{

Form3 isps = new Form3(ref this.isps);

if (isps.Visible == false)

{

isps.Show();

}

}

//Добавление Оплаты в Таблицу "ЗаписьОплаты"

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

{

// генерация данных о заказе

// создание нового заказа

try

{

DataRow newrow = ds.Tables["ЗаписьОплаты"].NewRow();

// заполнение атрибутов заказа

// для определения номера заказа можно узнать

// количество строк в таблице заказов

newrow["НомерЗаказа"] = int.Parse(textBox1.Text.ToString());

newrow["ДатаЗаказа"] = DateTime.Parse(textBox7.Text.ToString());

newrow["Оплачено/НеОплачено"] = Boolean.Parse(checkBox2.Checked.ToString());

newrow["Предоплата"] = int.Parse(textBox12.Text.ToString());

newrow["ДатаОплаты"] = DateTime.Parse(textBox13.Text.ToString());

// записываем созданную запись в таблицу

ds.Tables["ЗаписьОплаты"].Rows.Add(newrow);

// работа с DataGridView, показываеющей таблицу заказов

// отмена выделения всех выбранных строк в DataGridView

foreach (DataGridViewRow dgvr in dataGridView4.SelectedRows)

dgvr.Selected = false;

// установка выбора вновь созданного заказа

// последняя строка DataGridView – это строка для ручного ввода новой

// записи, поэтому последняя значимая строка – предпоследняя

dataGridView4.Rows[dataGridView4.Rows.Count - 2].Selected = true;

}

catch

{

MessageBox.Show("Ошибка номера заказа!");

}

}

//Удаление выбранной оплаты(если ошибочно внесена)

private void button4\_Click(object sender, EventArgs e)

{

// если заказ не был выбран, удалять нечего

if (dataGridView4.SelectedRows.Count == 0)

return;

//получение номера текущего выбранного заказа

nom = (int)dataGridView4.SelectedRows[0].Cells["НомерЗаказа"].Value;

DateTime date = (DateTime)dataGridView4.SelectedRows[0].Cells["ДатаЗаказа"].Value;

// выбираем все записи, соответствующие текущему заказу

DataRow[] drs = ds.Tables["ЗаписьОплаты"].Select("НомерЗаказа=" + nom + " and ДатаЗаказа='" + date + "'");

// удаляем все найденные заказы из таблиц набора данных ЗаписОплаты

for (int i = drs.Length - 1; i >= 0; i--)

drs[i].Delete();

}

//Генерация заказа в Word-файл

private void button5\_Click(object sender, EventArgs e)

{

// определяем, какой заказ выбран для генерации документа

if (dataGridView1.SelectedRows.Count == 0)

return;

//получение номера текущего выбранного заказа

nom = (int)dataGridView1.SelectedRows[0].Cells["НомерЗаказа"].Value;

//получение даты текущего выбранного заказа

date = (DateTime)dataGridView1.SelectedRows[0].Cells["ДатаЗаказа"].Value;

// поиск заказа по ключу

DataRow dr = ds.Tables["Заказы"].Rows.Find(new object[] { (object)nom, (object)date });

// создаем объект заказа для последующей сериализации

Zakaz c = new Zakaz((int)dr["IDЗаказчика"],(int)dr["IDИсполнителя"],(int)dr["НомерЗаказа"],(DateTime)dr["ДатаЗаказа"],(string)dr["Предмет/Тема"],(int)dr["Стоимость"]);

// выбираем все ЗаписьОплаты, соответствующие выбранному Заказу

DataRow[] drs = ds.Tables["ЗаписьОплаты"].Select("НомерЗаказа=" + nom + " and ДатаЗаказа='" + date + "'");

// добавляем информацию в объект заказа

foreach (DataRow d in drs)

{

Oplata z = new Oplata((int)d["НомерЗаказа"], (int)d["Предоплата"], (bool)d["Оплачено/НеОплачено"], (DateTime)d["ДатаОплаты"]);

c.AddOplata(z);

}

// работа с документами Word

// создание объекта-приложения

Word.Application app = new Word.Application();

// создание и добавление объекта-документа MS Word

Word.Document doc = app.Documents.Add();

// создание параграфа с заголовком (указание номера и даты заказа)

Word.Paragraph p = doc.Content.Paragraphs.Add();

// задание текста параграфа

p.Range.Text = "Заказ №" + c.НомерЗаказа + " от " + c.ДатаЗаказа;

// указание, что шрифт должен быть полужирным

p.Range.Font.Bold = 1;

// центрирование абзаца

p.Format.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

// устанавливаем межабзацный отступ

p.Format.SpaceAfter = 20;

p.Range.InsertParagraphAfter();

// вставка параграфа с указанием Заказчика

p = doc.Content.Paragraphs.Add();

p.Range.Text = "ID Заказчика: " + c.IDЗаказчика;

p.Format.Alignment = Word.WdParagraphAlignment.wdAlignParagraphLeft;

p.Format.SpaceAfter = 20;

p.Range.InsertParagraphAfter();

// вставка параграфа с указанием Исполнителя

p = doc.Content.Paragraphs.Add();

p.Range.Text = "ID Исполнителя: " + c.IDИсполнителя;

p.Format.Alignment = Word.WdParagraphAlignment.wdAlignParagraphLeft;

p.Format.SpaceAfter = 20;

p.Range.InsertParagraphAfter();

// вставка параграфа с указанием ПредметаТемы

p = doc.Content.Paragraphs.Add();

p.Range.Text = "Предмет/Тема: " + c.ПредметТема;

p.Format.Alignment = Word.WdParagraphAlignment.wdAlignParagraphLeft;

p.Format.SpaceAfter = 20;

p.Range.InsertParagraphAfter();

// вставка параграфа с указанием общей суммы по Заказу

p = doc.Content.Paragraphs.Add();

p.Range.Text = "Сумма: " + c.Стоимость;

p.Format.Alignment = Word.WdParagraphAlignment.wdAlignParagraphLeft;

p.Range.Font.Size = 20;

p.Format.SpaceAfter = 20;

p.Range.InsertParagraphAfter();

// вставка параграфа с таблицей, в которой указана детальная

// информация об оплате заказа

p = doc.Content.Paragraphs.Add();

// при создании таблицы указывается ее

// количество строк и столбцов (2 и 3 параметры)

Word.Table tab = doc.Tables.Add(p.Range, 1 + c.Список.Count, 4);

// указание, что таблица должна иметь рамку

tab.Borders.Enable = 1;

// заполняем ячейки таблицы – обращение к таблице осуществляется

// с помощью функции Cell(номер строки, номер столбца)

// отметим, что нумерация строк и столбцов начинается с 1

tab.Cell(1, 1).Range.Text = "НомерЗаказа";

tab.Cell(1, 2).Range.Text = "Предоплата";

tab.Cell(1, 3).Range.Text = "Оплачено Да/Нет";

tab.Cell(1, 4).Range.Text = "Дата оплаты";

// просматриваем список купленных товаров и заполняем остальные строки таблицы

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

{

tab.Cell(i + 2, 1).Range.Text = "" + (c.Список[i] as Oplata).НомерЗаказа;

tab.Cell(i + 2, 2).Range.Text = "" + (c.Список[i] as Oplata).Предоплата;

tab.Cell(i + 2, 3).Range.Text = "" + (c.Список[i] as Oplata).Оплаченность;

tab.Cell(i + 2, 4).Range.Text = "" + (c.Список[i] as Oplata).ДатаОплаты;

}

// сохранение документа

doc.Save();

// активируем окно MS Word для просмотра сгенерированного документа

app.Visible = true;

}

//Переводим документ в бинарный файл

private void button6\_Click(object sender, EventArgs e)

{

// определяем, какой заказ следует сериализовать

// код создания объекта (Zakaz c) аналогичен предыдущему примеру

// определяем, какой заказ выбран для генерации документа

if (dataGridView1.SelectedRows.Count == 0)

return;

//получение номера текущего выбранного заказа

nom = (int)dataGridView1.SelectedRows[0].Cells["НомерЗаказа"].Value;

//получение даты текущего выбранного заказа

date = (DateTime)dataGridView1.SelectedRows[0].Cells["ДатаЗаказа"].Value;

// поиск заказа по ключу

DataRow dr = ds.Tables["Заказы"].Rows.Find(new object[] { (object)nom, (object)date });

// создаем объект заказа для последующей сериализации

Zakaz c = new Zakaz((int)dr["IDЗаказчика"], (int)dr["IDИсполнителя"], (int)dr["НомерЗаказа"], (DateTime)dr["ДатаЗаказа"], (string)dr["Предмет/Тема"], (int)dr["Стоимость"]);

// создание объекта стандартного диалогового окна

// выбора файла для сохранения

SaveFileDialog dlg = new SaveFileDialog();

// показ диалогового окна на экране и проверка, по какой кнопке

// (ОК или Отмена) было произведено его закрытие

if (dlg.ShowDialog() == DialogResult.OK)

{

// создание файлового потока, в который будет

// сериализоваться информация о заказе - создание файла

// имя файла поставляется свойством диалога сохранения файла FileName

FileStream fs = new FileStream(dlg.FileName, FileMode.Create);

// создание форматера типа Binary

// SOAP НЕ ПОДДЕРЖИВАЛСЯ!

BinaryFormatter bin=new BinaryFormatter();

// сериализация объекта-чека c. Первый параметр – файловый поток,

// второй параметр – сериализованный объект

bin.Serialize(fs, c);

// закрытие файла

fs.Close();

MessageBox.Show("Файл сформирован!");

}

}

//Формируем цифровую подпись

private void button8\_Click(object sender, EventArgs e)

{

// создаем объект выбранного заказа

// как и в примере из предыдущего раздела

// В результате имеем заполненный объект класса Заказ с именем с

// определяем, какой заказ следует сериализовать

// код создания объекта (Zakaz c) аналогичен предыдущему примеру

// определяем, какой заказ выбран для генерации документа

if (dataGridView1.SelectedRows.Count == 0)

return;

//получение номера текущего выбранного заказа

nom = (int)dataGridView1.SelectedRows[0].Cells["НомерЗаказа"].Value;

//получение даты текущего выбранного заказа

date = (DateTime)dataGridView1.SelectedRows[0].Cells["ДатаЗаказа"].Value;

// поиск заказа по ключу

DataRow dr = ds.Tables["Заказы"].Rows.Find(new object[] { (object)nom, (object)date });

// создаем объект заказа для последующей сериализации

Zakaz c = new Zakaz((int)dr["IDЗаказчика"], (int)dr["IDИсполнителя"], (int)dr["НомерЗаказа"], (DateTime)dr["ДатаЗаказа"], (string)dr["Предмет/Тема"], (int)dr["Стоимость"]);

// хэшировать будем по двоичному коду объекта класса Zakaz

// проводим бинарную сериализацию объекта c

// создаем объект-форматер

BinaryFormatter ser = new BinaryFormatter();

// создаем поток для сериализации объекта в оперативной памяти

MemoryStream ms = new MemoryStream();

// проводим сериализацию в память

ser.Serialize(ms, c);

// получаем массив байт, определяющий объект заказа,

// считывая из потока в памяти

byte[] message = new byte[ms.Length];

ms.Read(message, 0, (int)ms.Length);

// хэшируем заказ

// создаем провайдер для хэширования

SHA1 sha1 = new SHA1CryptoServiceProvider();

// проводим хэширование

byte[] hashMessage = sha1.ComputeHash(message);

// получаем цифровую подпись с помощью алгоритма DSA

DSACryptoServiceProvider dsa = new DSACryptoServiceProvider();

// цифровая подпись – это также набор байт.

// Второй параметр метода – своеобразный ключ

byte[] signature = dsa.SignHash(hashMessage, "1.3.14.3.2.26");

// сохраняем параметры ключа в виде структурированной строки

string key = dsa.ToXmlString(true);

// сохраним подпись данного документа в бинарном файле,

// имя которого зависит от номера Заказа

BinaryWriter br = new BinaryWriter(new FileStream("Zakaz"+c.НомерЗаказа+".dat", FileMode.Create));

// сохраняем ключ в созданный файл

br.Write(key);

// сохраняем в файл цифровую подпись

// сначала количество байт

br.Write(signature.Length);

// затем саму подпись

br.Write(signature);

// закрываем файл с подписью

br.Close();

MessageBox.Show("Создана цифровая подпись!");

}

//Верифицируем - проверяем документ

private void button7\_Click(object sender, EventArgs e)

{

// создаем объект выбранного заказа

// определяем, выбран ли хотя бы один заказ

if (dataGridView1.SelectedRows.Count == 0)

return;

else

{

//получение номера текущего выбранного заказа

nom = (int)dataGridView1.SelectedRows[0].Cells["НомерЗаказа"].Value;

// проверка, имеется ли цифровая подпись выбранного чека,

// т.е. имеется ли соответствующий файл

if (!File.Exists("Zakaz" + nom + ".dat"))

{

MessageBox.Show("Ещё не создана цифровая подпись!");

return;

}

else

{

// создаем объект выбранного заказа аналогично предыдущей функции

// (объект с класса Zakaz)

MessageBox.Show(nom.ToString());

//получение номера текущего выбранного заказа

date = (DateTime)dataGridView1.SelectedRows[0].Cells["ДатаЗаказа"].Value;

// поиск заказа по ключу

DataRow dr = ds.Tables["Заказы"].Rows.Find(new object[] { (object)nom, (object)date });

// создаем объект заказа для последующей сериализации

Zakaz c = new Zakaz((int)dr["IDЗаказчика"], (int)dr["IDИсполнителя"], (int)dr["НомерЗаказа"], (DateTime)dr["ДатаЗаказа"], (string)dr["Предмет/Тема"], (int)dr["Стоимость"]);

// проводим генерацию хэш-значения для объекта c

// бинарная сериализация объекта для формирования цифровой подписи

BinaryFormatter ser = new BinaryFormatter();

// создаем поток для сериализации объекта в оперативной памяти

MemoryStream ms = new MemoryStream();

ser.Serialize(ms, c);

// получаем массив байт, определяющий объект чека

byte[] message = new byte[ms.Length];

ms.Read(message, 0, (int)ms.Length);

// создаем провайдер для хэширования

SHA1 sha1 = new SHA1CryptoServiceProvider();

// проводим хэширование

byte[] hashMessage = sha1.ComputeHash(message);

// работа с цифровой подписью - считываем подпись и

// ключ из файла и осуществляем верификацию

// читаем данные из файла-подписи

BinaryReader br = new BinaryReader(new FileStream("Zakaz" + nom + ".dat", FileMode.Open));

// читаем ключ для шифрования

string key = br.ReadString();

// читаем данные подписи

int n\_sign = br.ReadInt32();

byte[] b\_sign = br.ReadBytes(n\_sign);

br.Close();

// импортируем параметры в провайдер шифрования

DSACryptoServiceProvider dsa = new DSACryptoServiceProvider();

dsa.FromXmlString(key);

// проводим верификацию подписей

// первый параметр - сгенерированный хэш объекта

// второй параметр - ключ

// третий параметр - цифровая подпись

if (dsa.VerifyHash(hashMessage, "1.3.14.3.2.26", b\_sign))

{

MessageBox.Show("Успешная верификация!");

}

else

{

MessageBox.Show("Заказ был изменен. Ошибка верификации!");

}

}

}

}

}

//Оплата

[Serializable]

class Oplata

{

int номерЗаказа;

int предоплата;

bool оплаченность;

DateTime датаОплаты;

public Oplata()

{

номерЗаказа = 0;

предоплата = 0;

оплаченность = false;

датаОплаты = DateTime.Parse("00.00.0000");

}

public Oplata(int nom, int price, bool oplacheno, DateTime date)

{

номерЗаказа = nom;

предоплата = price;

оплаченность=oplacheno;

датаОплаты = date;

}

public int НомерЗаказа

{

get { return номерЗаказа; }

set { номерЗаказа = value; }

}

public int Предоплата

{

get { return предоплата; }

set { предоплата = value; }

}

public DateTime ДатаОплаты

{

get { return датаОплаты; }

set { датаОплаты = value; }

}

public bool Оплаченность

{

get { return оплаченность; }

set { оплаченность = value; }

}

}

//Класс для Заказа

[Serializable]

class Zakaz

{

int iDЗаказчика;

int iDИсполнителя;

int номерЗаказа;

DateTime датаЗаказа;

string предметТема;

int стоимость;

ArrayList list;

public Zakaz()

{

iDЗаказчика = 0;

iDИсполнителя = 0;

номерЗаказа = 0;

датаЗаказа = DateTime.Parse("00.00.0000");

предметТема = "";

стоимость = 0;

}

public Zakaz(int IDz, int IDi, int nom, DateTime date, string ex, int cost)

{

iDЗаказчика = IDz;

iDИсполнителя = IDi;

номерЗаказа = nom;

датаЗаказа = date;

предметТема = ex;

стоимость = cost;

list = new ArrayList();

}

public void AddOplata(Oplata z)

{

list.Add(z);

}

public int НомерЗаказа

{

get { return номерЗаказа; }

set { номерЗаказа = value; }

}

public DateTime ДатаЗаказа

{

get { return датаЗаказа; }

set { датаЗаказа = value; }

}

public int IDЗаказчика

{

get { return iDЗаказчика; }

set { iDЗаказчика = value; }

}

public int IDИсполнителя

{

get { return iDИсполнителя; }

set { iDИсполнителя = value; }

}

public string ПредметТема

{

get { return предметТема; }

set { предметТема = value; }

}

public int Стоимость

{

get { return стоимость; }

set { стоимость = value; }

}

public ArrayList Список

{

get { return list; }

set { list = value; }

}

}

}

1. **Использованная литература.**
2. Сайт msdn.com
3. Методические рекомендации по документоведению.