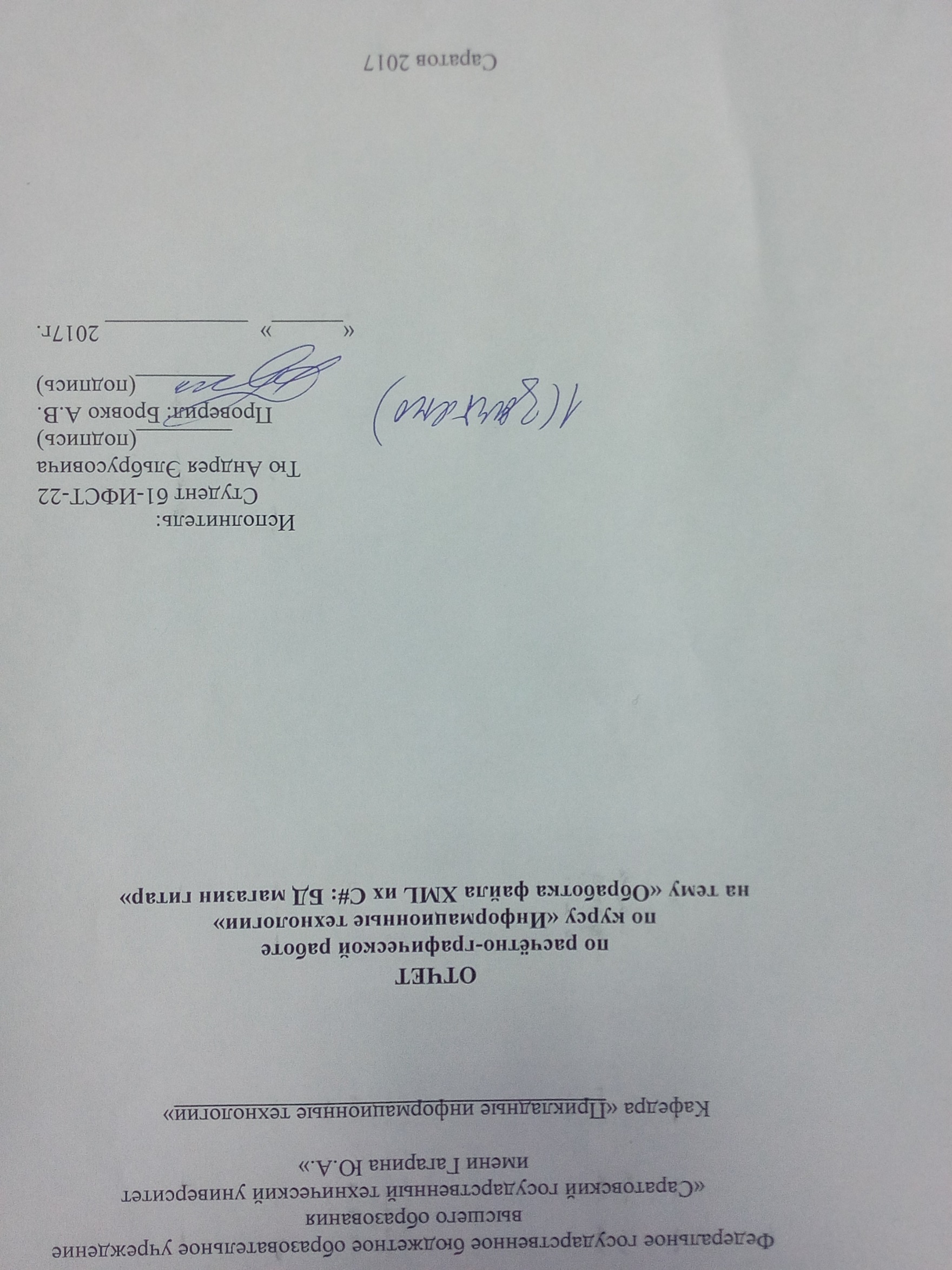
****

**Задание на расчетно-графическую работу.**

1. Выбрать предметную область из предложенного списка или предложить свою, согласовав ее с преподавателем.
2. Разработать структуру файла XML для хранения информации об объектах предметной области.
3. Разработать DTD или Schema XML (по выбору студента) для созданного файла XML.
4. Разработать консольное приложение на языке C# для обработки информации, хранящейся в файле XML. Приложение должно предоставлять пользователю меню с возможностью просмотра, добавления, удаления и изменения информации, хранящейся в файле XML. Приложение должно дружественно реагировать на ошибки ввода пользователя (при неправильном вводе должно выдаваться сообщение и предоставляться возможность ввести информацию заново).

**Ход выполнения работы.**

Исполнителем была выбрана предметная область «БД магазин гитар» из предложенного списка методических указаний по расчётно-графической работе.

Был разработан XML файл, имеющий следующую структуру:

<?xml version="1.0" encoding="utf-8" ?>

<!DOCTYPE Guitars [

<!ELEMENT Guitars (Acoustic\*, Electric\*, Bass\*)>

<!ELEMENT Acoustic (Guitar)\*>

<!ATTLIST Acoustic Type CDATA #REQUIRED>

<!ELEMENT Electric (Guitar)\*>

<!ATTLIST Electric Type CDATA #REQUIRED>

<!ELEMENT Bass (Guitar)\*>

<!ATTLIST Bass Type CDATA #REQUIRED>

<!ELEMENT Guitar (MadeBy, Model, DeckWood, FingerboardWood, Weight, Length, Price)>

<!ATTLIST Guitar ID CDATA #REQUIRED>

<!ELEMENT MadeBy (#PCDATA)>

<!ELEMENT Model (#PCDATA)>

<!ELEMENT DeckWood (#PCDATA)>

<!ELEMENT FingerboardWood (#PCDATA)>

<!ELEMENT Weight (#PCDATA)>

<!ELEMENT Length (#PCDATA)>

<!ELEMENT Price (#PCDATA)>

]>

<Guitars>

<Acoustic Type="Western">

<Guitar ID="0">

<sa></sa>

<MadeBy>Colombo</MadeBy>

<Model>LM3800</Model>

<DeckWood>Сосна</DeckWood>

<FingerboardWood>Сосна</FingerboardWood>

<Weight>1.0</Weight>

<Length>0.95</Length>

<Price>75.95$</Price>

</Guitar>

</Acoustic>

<Electric Type="Stratocaster">

<Guitar ID="1">

<MadeBy>Gibson</MadeBy>

<Model>Les Paul Black Beauty Custom</Model>

<DeckWood>Махагон</DeckWood>

<FingerboardWood>Махагон</FingerboardWood>

<Weight>5.0</Weight>

<Length>1.2</Length>

<Price>7000.00$</Price>

</Guitar>

</Electric>

</Guitars>

**Для работы с файлом была создана программа на языке программирования C#.**

**Программа реализует архитектуру MVP, где View представлен в виде класса View, Presenter в виде XMLWorker, а Model в виде XML-файла, представленного выше.**

**View:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RGR

{

class View

{

private XMLWorker xmlWorker;

public View(XMLWorker xw)

{

if (xw != null) xmlWorker = xw;

else throw new ArgumentException();

}

public void ShowMainMenu()

{

Console.WriteLine("Choose the option : " + Environment.NewLine +

"1. Show all guitars" + Environment.NewLine +

"2. Add new guitar" + Environment.NewLine +

"3. Update guitar" + Environment.NewLine +

"4. Delete guitar" + Environment.NewLine +

"5. Exit");

}

public void ShowGuitars()

{

LinkedList<Guitar> list = xmlWorker.GetAllGuitars();

foreach (Guitar guitar in list)

{

Console.WriteLine(guitar);

Console.WriteLine("\_\_--||=================||--\_\_");

}

}

public void AddGuitar()

{

string MadeBy = "";

string Model = "";

string DeckWood = "";

string FingerboardWood = "";

string Weight = "";

string Length = "";

string Price = "";

string Variation = "";

string Type = "";

GetValue("type of guitar (acoustic, electric, etc)", out Variation);

GetValue("type of deck (western, classic, straticaster, etc)", out Type);

GetValue("manufacturer",out MadeBy);

GetValue("model of this guitar",out Model);

GetValue("wood of deck", out DeckWood);

GetValue("wood of grif", out FingerboardWood);

GetDoubleData("weight", out Weight);

GetDoubleData("length", out Length);

GetDoubleData("price", out Price);

xmlWorker.AddGuitar(new Guitar(Variation, Type, MadeBy, Model, DeckWood, FingerboardWood, Weight, Length, Price));

}

private void GetValue(string substring, out string data)

{

Console.Write($"Enter the {substring} : ");

data = Console.ReadLine();

}

public void GetDoubleData(string substring, out string data)

{

do

{

GetValue(substring, out data);

} while (!double.TryParse(data, out double g) || g < 0);

}

public void UpdateGuitar()

{

Console.WriteLine("Choose the guitar to update or press Enter to cancel : ");

ShowGuitars();

string iD = Console.ReadLine();

if (iD == null) return;

if(!int.TryParse(iD, out int ID) || ID < 0)

{

Console.WriteLine("Bad data for ID");

return;

}

Dictionary<string, string> tagName = new Dictionary<string, string>

{

{"1", "MadeBy"},

{"2", "Model"},

{"3", "DeckWood"},

{"4", "FingerboardWood"},

{"5", "Weight"},

{"6", "Length"},

{"7", "Price"}

};

Console.WriteLine("Choose the tag to update :" + Environment.NewLine +

"1. Made by" + Environment.NewLine +

"2. Model" + Environment.NewLine +

"3. Deck wood" + Environment.NewLine +

"4. Fingerboard wood" + Environment.NewLine +

"5. Weight" + Environment.NewLine +

"6. Length" + Environment.NewLine +

"7. Price");

string TagName = Console.ReadLine();

switch (TagName)

{

case "1":

case "2":

case "3":

case "4":

case "5":

case "6":

case "7":

Console.WriteLine("Enter the new value :");

string data = Console.ReadLine();

xmlWorker.UpdateGuitar(ID, tagName[TagName], data);

break;

default:

break;

}

}

public static void NoTagFoundMessage()

{

Console.WriteLine("No such tag found");

}

public void DeleteGuitar()

{

Console.WriteLine("Choose the guitar to delete : ");

ShowGuitars();

string iD = Console.ReadLine();

if (!int.TryParse(iD, out int ID) || ID < 0)

{

Console.WriteLine("Bad data for ID");

return;

}

if (!xmlWorker.DeleteByID(ID)) Console.WriteLine("Wrong ID entry");

}

}

}

**XMLWorker:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml;

using System.Xml.Linq;

using System.Xml.Schema;

namespace RGR

{

class XMLWorker

{

public XDocument xdoc;

public string path;

private int LastID;

public XMLWorker()

{

path = "Guitars.xml";

xdoc = XDocument.Load(path);

XmlReaderSettings setting = new XmlReaderSettings();

setting.DtdProcessing = DtdProcessing.Parse;

setting.ValidationType = ValidationType.DTD;

setting.ValidationEventHandler += new ValidationEventHandler(ValidationCallBack);

XmlReader reader = XmlReader.Create(path, setting);

while (reader.Read());

LastID = -1;

foreach (XElement xType in xdoc.Root.Elements())

{

foreach (XElement xGuitar in xType.Elements())

{

xGuitar.Attribute("ID").SetValue((++LastID).ToString());

}

}

}

private static void ValidationCallBack(object sender, ValidationEventArgs e)

{

Console.WriteLine("Validation Error: {0}", e.Message);

}

public LinkedList<Guitar> GetAllGuitars()

{

LinkedList<Guitar> list = new LinkedList<Guitar>();

foreach (XElement xType in xdoc.Root.Elements())

{

foreach (XElement xGuitar in xType.Elements())

{

list.AddLast(new Guitar(int.Parse(xGuitar.Attribute("ID").Value),

xGuitar.Parent.Name.ToString(),

xGuitar.Parent.Attribute("Type").Value,

xGuitar.Element("MadeBy").Value,

xGuitar.Element("Model").Value,

xGuitar.Element("DeckWood").Value,

xGuitar.Element("FingerboardWood").Value,

xGuitar.Element("Weight").Value,

xGuitar.Element("Length").Value,

xGuitar.Element("Price").Value));

}

}

return list;

}

public void AddGuitar(Guitar guitar)

{

bool f = false;

foreach (XElement xType in xdoc.Root.Elements(guitar.Variation1))

{

if (xType.Attribute("Type").Value == guitar.Type1)

{

xType.Add(new XElement("Guitar", new XAttribute("ID", ++LastID),

new XElement("MadeBy", guitar.MadeBy1),

new XElement("Model", guitar.Model1),

new XElement("DeckWood", guitar.DeckWood1),

new XElement("FingerboardWood", guitar.FingerboardWood1),

new XElement("Weight", guitar.Weight1),

new XElement("Length", guitar.Price1),

new XElement("Price", guitar.Price1)));

f = true;

break;

}

}

if (!f)

{

xdoc.Root.Add(new XElement(guitar.Variation1, new XAttribute("Type", guitar.Type1),

new XElement("Guitar", new XAttribute("ID", ++LastID),

new XElement("MadeBy", guitar.MadeBy1),

new XElement("Model", guitar.Model1),

new XElement("DeckWood", guitar.DeckWood1),

new XElement("FingerboardWood", guitar.FingerboardWood1),

new XElement("Weight", guitar.Weight1),

new XElement("Length", guitar.Price1),

new XElement("Price", guitar.Price1))));

}

}

public bool DeleteByID(int ID)

{

bool result = false;

foreach (XElement xType in xdoc.Root.Elements())

{

foreach (XElement xGuitar in xType.Elements())

{

if (xGuitar.Attribute("ID").Value == ID.ToString())

{

xGuitar.Remove();

result = true;

break;

}

}

}

return result;

}

public void UpdateGuitar(int ID, string TagName, string NewData)

{

bool flag = true;

foreach (XElement xType in xdoc.Root.Elements())

{

foreach (XElement xGuitar in xType.Elements())

{

if (xGuitar.Attribute("ID").Value == ID.ToString())

{

xGuitar.Element(TagName).SetValue(NewData);

flag = false;

}

}

}

if (flag) View.NoTagFoundMessage();

}

}

}

**Класс Guitar, представляющий тег <Guitar> в виде объекта с соответствующими полями и имеющий реализацию ToString для уровня представления:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RGR

{

class Guitar

{

private int ID;

private string Variation;

private string Type;

private string MadeBy;

private string Model;

private string DeckWood;

private string FingerboardWood;

private string Weight;

private string Length;

private string Price;

public Guitar(int ID, string Variation, string Type, string madeBy, string model, string deckWood, string fingerboardWood, string weight, string length, string price)

: this(Variation, Type, madeBy, model, deckWood, fingerboardWood, weight, length, price)

{

ID1 = ID;

}

public Guitar(string Variation, string Type, string madeBy, string model, string deckWood, string fingerboardWood, string weight, string length, string price)

{

Variation1 = Variation;

Type1 = Type;

MadeBy1 = madeBy;

Model1 = model;

DeckWood1 = deckWood;

FingerboardWood1 = fingerboardWood;

Weight1 = weight;

Length1 = length;

Price1 = price;

}

public string MadeBy1 { get => MadeBy; set => MadeBy = value; }

public string Model1 { get => Model; set => Model = value; }

public string DeckWood1 { get => DeckWood; set => DeckWood = value; }

public string FingerboardWood1 { get => FingerboardWood; set => FingerboardWood = value; }

public string Weight1 { get => Weight; set => Weight = value; }

public string Length1 { get => Length; set => Length = value; }

public string Price1 { get => Price; set => Price = value; }

public string Type1 { get => Type; set => Type = value; }

public int ID1 { get => ID; set => ID = value; }

public string Variation1 { get => Variation; set => Variation = value; }

public override string ToString()

{

return $"{Variation} {Type} Guitar:" + Environment.NewLine +

$" Made by : {MadeBy}" + Environment.NewLine +

$" Model : {Model}" + Environment.NewLine +

$" Deck wood : {DeckWood}" + Environment.NewLine +

$" Fingerboard wood : {FingerboardWood}" + Environment.NewLine +

$" Weight : {Weight}" + Environment.NewLine +

$" Length : {Length}" + Environment.NewLine +

$" Price : {Price}";

}

}

}

**Модуль реализации – класс Programm, содержащий метод Main:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml;

using System.Xml.Linq;

using System.Xml.Schema;

namespace RGR

{

class Program

{

public static void Main()

{

XMLWorker xw = new XMLWorker();

View view = new View(xw);

bool end = false;

string option = "";

while (!end)

{

//Console.Clear();

view.ShowMainMenu();

option = Console.ReadLine();

Console.Clear();

switch (option)

{

case "1":

view.ShowGuitars();

break;

case "2":

view.AddGuitar();

break;

case "3":

view.UpdateGuitar();

break;

case "4":

view.DeleteGuitar();

break;

case "5":

end = true;

break;

default:

Console.WriteLine("Wrong option");

break;

}

Console.WriteLine("Press Enter to continue. . .");

Console.ReadKey();

xw.xdoc.Save(xw.path);

}

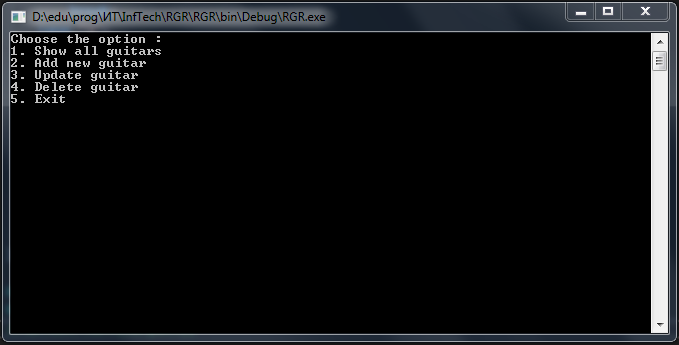
}

}

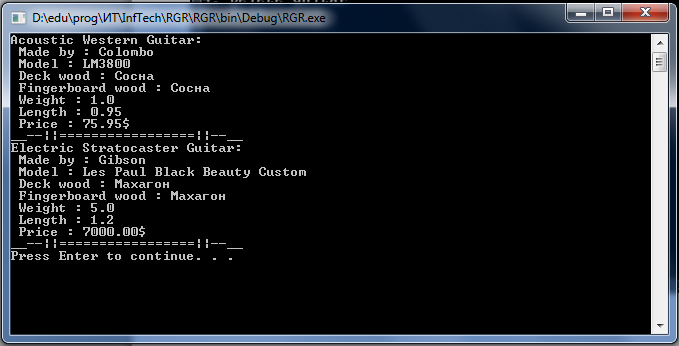
}

**Примеры работы**

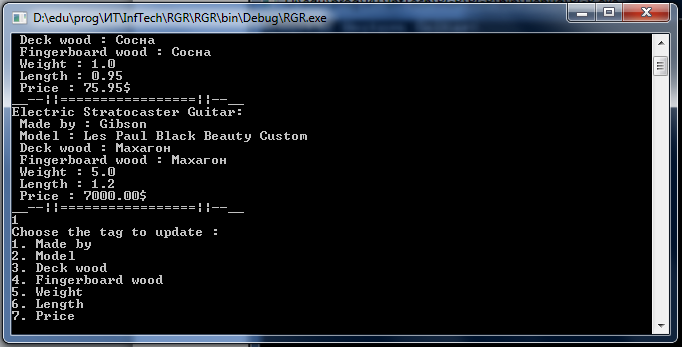
**Основное меню:**



**Просмотр элементов:**



**Редактирование элементов:**



Вывод.

В результате проделанной работы было разработано консольное приложение на языке программирования C#. Приложение позволяет обрабатывать, получать, изменять данные из XML-файла, представляющего из себя структуру Базы данных. Разработанная программа также позволяет проверять соответствие валидации XML-документа на основе DTD.

Были изучены технологии разработки XML-документов, принципы создания валидации (DTD) к разрабатываем XML-файлам, а также изучены методы работы с XML-файлами с помощью языка C#.