**Vežba 4**

**Tema: ADO.NET**

* **ADO.NET снабдевачи подацима (Data Provider-и)**
* **Основни објекти ADO.NET Data Provider-a**
* **Креирање конекције ка бази података из програмског кода**

**Zadatak 1**

НАЗИВ РАДНОГ ЗАДАТКА: **Израда десктоп апликације:** **Библиотека**

Пројектована је база за памћење и обраду података о расположивим ресурсима библиотеке и њеним корисницима. Структура базе је дата на слици у прилогу. Управа библиотеке се труди да стално набавља нове наслове, али и расходује старе – похабане или изгубљене наслове. Да не би дошло до претрпавања података, тражено је да се омогући преглед и брисање аутора (у ситуацији када библиотека не поседује ни један наслов тог аутора). Исказана је потреба да се у бази чувају и додатни подаци o формату у коме је књига штампана.

На основу спецификације дате у **Прилогу задатка** урадити следеће:

* Дизајнирати десктоп апликацију у циљу задовољења захтева;
* Повезати десктоп апликацију са базом података;
* Креирати упите за манипулацију подацима у бази података;
* Тестирати функционалности десктоп апликације;
* Модификовати структуру базе података у складу са захтевима корисника;
* Написати упутство корисницима о коришћењу апликације;
* ЕР дијаграм, бекап базе и апликацију запамтити на десктопу рачунара, у фолдеру **Матурски\_испит\_4ЕИТ\Име\_Презиме\4ЕИТ\_A2**, где се као име и презиме уноси име ученика.

Предвиђено време за израду задатка је 180 минута.

По истеку максималног времена задатак се прекида и бодује се оно што је до тада урађено.

У оквиру времена за израду задатка ученик може да одустане од даљег рада, при чему се бодује оно што је до тада урађено.

Прилоге преузети из фолдера **4ЕИТ** на десктопу рачунара.

Дијаграм почетне структуре базе дат је као **Прилог\_4ЕИТ\_А21**.

Резервна копија почетног стања базе дата је као **Прилог\_4ЕИТ\_А22**.

Архива икона дата је као **Прилог\_4ЕИТ\_А00**.

**ПРИЛОГ:**



Развојним планом апликације предвиђено је да се уради следеће:

Креирати форму која податке о свим ауторима табеларно приказује у ***listView*** контроли.

Кликом на неки од редова, податке о аутору треба приказати у осталим пољима форме.

Кликом на дугме ***Брисање*** у линији са алатима, обрисати изабраног аутора из базе, а потом освежити приказ у ***listView-у***.

Кликом на дугме ***Излаз*** у линији алата прве форме, затворити целу апликацију.

Отворити кратко корисничко упутство кликом на дугме ***О апликацији***.



Дугме ***Анализа*** отвара другу форму на којој је потребно приказати колико су пута дела изабраног аутора била изнајмљена током задатог временског периода. Временски период се задаје у односу на текући датум, као изабрани број година уназад. Табела и график се приказују кликом на дугме ***Прикажи***.

Кликом на ***Изађи*** вратити корисника на почетну форму.



Осмислити проширење базе којим се за сваки примерак књиге омогућава памћење формата у коме је књига издата. Формат може да буде тврди повез, меки повез, илустовано издање...

Проширити базу тако да се за читаоца памтити датум последњег плаћања чланарине и ограничити да се чланарина не може да плаћа ретроактивно (датум плаћања није старији од текућег датума).

Креирати дијаграм и бекап стања проширене базе.

**Решење**

**Процедуре у бази:**

usp\_AutorUcitaj

USE [A2]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[usp\_AutorUcitaj] Script Date: 2.11.2022. 17:29:19 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[usp\_AutorUcitaj]

AS

BEGIN

select \* from Autor as a

END

usp\_AutorBrisi

USE [A2]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[usp\_AutorBrisi]

@sifra int = null

AS

BEGIN

Delete from Autor

where AutorID = @sifra

END

usp\_Statistika

USE [A2]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[usp\_Statistika] Script Date: 2.11.2022. 19:36:44 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[usp\_Statistika]

-- Add the parameters for the stored procedure here

@autorID int = null,

@godina int = null

AS

BEGIN

select year(c.DatumUzimanja) as Godina, count(c.KnjigaID) as Broj

from Autor as a

inner join Napisali as n on a.AutorID=n.AutorID

inner join Knjiga as k on n.KnjigaID=k.KnjigaID

inner join Na\_Citanju as c on k.KnjigaID=c.KnjigaID

where a.AutorID=@autorID and year(c.DatumUzimanja)>= year(getdate()) - @godina

group by year(c.DatumUzimanja)

order by Godina

END

class Konekcija

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace A2

{

class Konekcija

{

static string connStr = @"Data Source=VESNA-STANOJEVI\SQLEXPRESS; Initial Catalog=A2; integrated security=true";

public static SqlCommand GetCommand()

{

SqlCommand cmd = new SqlCommand();

cmd.Connection = new SqlConnection(connStr);

cmd.CommandType = CommandType.StoredProcedure;

return cmd;

}

}

}

class Autor

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace A2

{

class Autor

{

public int AutorID { get; set; }

public string Ime { get; set; }

public string Prezime { get; set; }

public DateTime DatumRodjenja { get; set; }

public string PunoIme

{

get { return this.Ime + " " + this.Prezime; }

}

public void InicijalizujPolja(DataRow dr)

{

this.AutorID = (int)dr["AutorID"];

this.Ime = (string)dr["Ime"];

this.Prezime = (string)dr["Prezime"];

if (dr["DatumRodjenja"] != DBNull.Value) this.DatumRodjenja = Convert.ToDateTime(dr["DatumRodjenja"]);

}

public Autor() { }

public Autor(DataRow dr)

{

this.InicijalizujPolja(dr);

}

public static List<Autor> UcitajSve()

{

SqlCommand cmd = Konekcija.GetCommand();

cmd.CommandText = "usp\_AutorUcitaj";

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

List<Autor> lista = new List<Autor>();

try

{

cmd.Connection.Open();

da.Fill(dt);

if (dt.Rows.Count >= 1)

{

foreach (DataRow dr in dt.Rows) lista.Add(new Autor(dr));

return lista;

}

else

{

MessageBox.Show("Podaci nisu nadjeni.");

return null;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message + "Konekcija nije uspostavljena.");

return null;

}

finally

{

cmd.Connection.Close();

}

}

public bool Obrisi()

{

SqlCommand cmd = Konekcija.GetCommand();

cmd.CommandText = "usp\_AutorBrisi";

cmd.Parameters.AddWithValue("@sifra", this.AutorID);

try

{

cmd.Connection.Open();

int ok = cmd.ExecuteNonQuery();

if (ok == 1)

{

MessageBox.Show("Uspesno brisanje");

return true;

}

else

{

MessageBox.Show("Brisanje nije izvrseno.");

return false;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message + "Konekcija nije uspostavljena.");

return false;

}

finally

{

cmd.Connection.Close();

}

}

}

}

class Statistika

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace A2

{

class Statistika

{

public static DataTable StatistikaIznajmljivanja(int god, int autor)

{

SqlCommand cmd = Konekcija.GetCommand();

cmd.CommandText = "usp\_Statistika";

cmd.Parameters.AddWithValue("@godina", god);

cmd.Parameters.AddWithValue("@autorID", autor);

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

try

{

cmd.Connection.Open();

da.Fill(dt);

if (dt.Rows.Count >= 1)

{

return dt;

}

else

{

MessageBox.Show("Za traženi vremenski period za izabranog autora nije bilo iznajmljenih knjiga.");

return null;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message + "Konekcija nije uspostavljena.");

return null;

}

finally

{

cmd.Connection.Close();

}

}

}

}

class FAutor

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace A2

{

public partial class FAutor : Form

{

public FAutor()

{

InitializeComponent();

}

private void NapuniListView(List<Autor> lista)

{

listView1.Items.Clear();

foreach (Autor a in lista)

{

string[] podaci = { a.AutorID.ToString(), a.Ime, a.Prezime, a.DatumRodjenja.ToString("dd/MM/yyyy") };

ListViewItem lvi = new ListViewItem(podaci);

listView1.Items.Add(lvi);

}

}

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

{

List<Autor> autori = Autor.UcitajSve();

NapuniListView(autori);

listView1.FullRowSelect = true;

listView1.Items[0].Selected = true;

}

private void PrikaziAutora(Autor x)

{

txtSifra.Text = x.AutorID.ToString();

txtIme.Text = x.Ime;

txtPrezime.Text = x.Prezime;

txtDatum.Text = x.DatumRodjenja.ToString("dd/MM/yyyy");

}

private void OcistiPodatke()

{

txtSifra.Text = String.Empty;

txtIme.Text = String.Empty;

txtPrezime.Text = String.Empty;

txtDatum.Text = txtDatum.Mask;

}

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

{

string sifra = txtSifra.Text;

Autor izabran = new Autor();

var item = listView1.FindItemWithText(sifra, false, 0, false);

if (item != null)

{

izabran.AutorID = Convert.ToInt32(sifra);

izabran.Ime = item.SubItems[1].Text;

izabran.Prezime = item.SubItems[2].Text;

izabran.DatumRodjenja = Convert.ToDateTime(item.SubItems[3].Text);

PrikaziAutora(izabran);

DialogResult dr = MessageBox.Show("Da li si siguran da zelis da obrises izabranog autora?", "Pritisnite neko od ovih dugmadi", MessageBoxButtons.YesNo, MessageBoxIcon.Information);

if (dr == DialogResult.Yes)

{

if (izabran.Obrisi())

{

NapuniListView(Autor.UcitajSve());

OcistiPodatke();

}

}

}

else

{

MessageBox.Show("Nije pronadjen autor sa unetom sifrom");

OcistiPodatke();

}

}

private void listView1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (listView1.SelectedItems.Count == 0) return;

int sifra = Convert.ToInt32(listView1.SelectedItems[0].SubItems[0].Text);

Autor x = new Autor();

x.AutorID = sifra;

x.Ime = listView1.SelectedItems[0].SubItems[1].Text;

x.Prezime = listView1.SelectedItems[0].SubItems[2].Text;

x.DatumRodjenja = Convert.ToDateTime(listView1.SelectedItems[0].SubItems[3].Text);

PrikaziAutora(x);

}

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

{

FAnaliza frm = new FAnaliza();

frm.ShowDialog();

}

}

}

class FAnaliza

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

namespace A2

{

public partial class FAnaliza : Form

{

public FAnaliza()

{

InitializeComponent();

}

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

{

if (cmbAutor.SelectedIndex == -1)

{

MessageBox.Show("Izaberite autora");

return;

}

int god = (int)numPeriod.Value;

int autor = (int)cmbAutor.SelectedValue;

chart1.Series[0].Points.Clear(); //Brisanje prethodnog ispisa grafikona

try

{

dataGridView1.DataSource = Statistika.StatistikaIznajmljivanja(god, autor);

chart1.Series[0].XValueType = ChartValueType.Int32;

chart1.Series[0].YValueType = ChartValueType.Int32;

chart1.Series[0].XValueMember = "Godina";

chart1.Series[0].YValueMembers = "Broj";

chart1.DataSource = dataGridView1.DataSource;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

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

{

cmbAutor.DisplayMember = "PunoIme";

cmbAutor.ValueMember = "AutorID";

cmbAutor.DataSource = Autor.UcitajSve();

cmbAutor.SelectedIndex = -1;

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

}

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

{

this.Close();

}

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

{

dataGridView1.DataSource = null;

chart1.Series[0].Points.Clear(); //Brisanje prethodnog ispisa grafikona

}

}

}