***Группа ИТ-2707. Леженко С.А.***

Разработать приложение для автоматизации учёта сотрудников в отделе кадров предприятия.

1. Приложение должно содержать информацию о сотруднике: фамилия, имя, отчество, должность, номер приказа о приеме на работу и номер приказа об увольнении.
2. В базе данных должна храниться фотография сотрудника.
3. Доступ к данным должен быть сделан через хранимые процедуры.
4. Необходимо ограничить доступ к данным путём раздачи прав пользователям системы.

***CreateDB.sql***

--Создаём базу

CREATE DATABASE Employees;

GO

USE Employees;

GO

--Создаём таблицу фамилий

CREATE TABLE Surnames (

id int IDENTITY (1, 1) NOT NULL

CONSTRAINT PK\_Surmanes PRIMARY KEY (id),

name varchar(50) NOT NULL

CONSTRAINT UniSurnames UNIQUE (name)

);

GO

--Создаём таблицу имен

CREATE TABLE Names (

id int IDENTITY (1, 1) NOT NULL

CONSTRAINT PK\_Names PRIMARY KEY (id),

name varchar(50) NOT NULL

CONSTRAINT UniNames UNIQUE (name)

);

GO

--Создаём таблицу отчеств

CREATE TABLE Patronymics (

id int IDENTITY (1, 1) NOT NULL

CONSTRAINT PK\_Patronymic PRIMARY KEY (id),

name varchar(50) NOT NULL

CONSTRAINT UniPatronymic UNIQUE (name)

);

GO

--Создаём таблицу должностей

CREATE TABLE Positions (

id int IDENTITY (1, 1) NOT NULL

CONSTRAINT PK\_Position PRIMARY KEY (id),

name varchar(255) NOT NULL

CONSTRAINT UniPosition UNIQUE (name)

);

GO

--Создаём таблицу работников

CREATE TABLE Employees (

id int IDENTITY (1, 1) NOT NULL

CONSTRAINT PK\_Empliyee PRIMARY KEY (id),

idSurname int

CONSTRAINT FK\_Surname FOREIGN KEY (idSurname)

REFERENCES Surnames (id),

idName int

CONSTRAINT FK\_Name FOREIGN KEY (idName)

REFERENCES Names (id),

idPatronymic int

CONSTRAINT FK\_Patronymic FOREIGN KEY (idPatronymic)

REFERENCES Patronymics (id),

idPosition int

CONSTRAINT FK\_Position FOREIGN KEY (idPosition)

REFERENCES Positions (id),

admission varchar(30),

dismissal varchar(30),

photo varchar(255)

);

GO

--Процедура добавляет фамилию и возвращает её id

CREATE PROCEDURE getIdSurname @surname varchar(50),

@idSurname int OUTPUT

AS

BEGIN

--Если фамилия заполнена

IF (@surname IS NOT NULL)

BEGIN

SELECT

@idSurname = id

FROM Surnames

WHERE name = @surname;

--Если это новая фамилия добавляем

IF (@idSurname IS NULL)

BEGIN

INSERT INTO Surnames

VALUES (@surname);

SELECT

@idSurname = id

FROM Surnames

WHERE name = @surname;

END

END

RETURN;

END

GO

--Процедура добавляет имя и возвращает его id

CREATE PROCEDURE getIdName @name varchar(50),

@idName int OUTPUT

AS

BEGIN

--Если имя заполнено

IF (@name IS NOT NULL)

BEGIN

SELECT

@idName = id

FROM Names

WHERE name = @name;

--Если это новое имя добавляем

IF (@idName IS NULL)

BEGIN

INSERT INTO Names

VALUES (@name);

SELECT

@idName = id

FROM Names

WHERE name = @name;

END

END

RETURN;

END

GO

--Процедура добавляет отчество и возвращает его id

CREATE PROCEDURE getIdPatronymic @patronymic varchar(50),

@idPatronymic int OUTPUT

AS

BEGIN

--Если отчество заполнено

IF (@patronymic IS NOT NULL)

BEGIN

SELECT

@idPatronymic = id

FROM Patronymics

WHERE name = @patronymic;

--Если это новое отчество добавляем

IF (@idPatronymic IS NULL)

BEGIN

INSERT INTO Patronymics

VALUES (@patronymic);

SELECT

@idPatronymic = id

FROM Patronymics

WHERE name = @patronymic;

END

END

RETURN;

END

GO

--Процедура добавляет должность и возвращает её id

CREATE PROCEDURE getIdPosition @position varchar(50),

@idPosition int OUTPUT

AS

BEGIN

--Если должность заполнена

IF (@position IS NOT NULL)

BEGIN

SELECT

@idPosition = id

FROM Positions

WHERE name = @position;

--Если это новая должность добавляем

IF (@idPosition IS NULL)

BEGIN

INSERT INTO Positions

VALUES (@position);

SELECT

@idPosition = id

FROM Positions

WHERE name = @position;

END

END

RETURN;

END

GO

--Процедура вставки нового работника

CREATE PROCEDURE addEmployee @surname varchar(50),

@name varchar(50),

@patronymic varchar(50),

@position varchar(255),

@admission varchar(30),

@dismissal varchar(30),

@photo varchar(255)

AS

BEGIN

DECLARE @idSurname int;

DECLARE @idName int;

DECLARE @idPatronymic int;

DECLARE @idPosition int;

EXECUTE getIdSurname @surname,

@idSurname OUTPUT;

EXECUTE getIdName @name,

@idName OUTPUT;

EXECUTE getIdPatronymic @patronymic,

@idPatronymic OUTPUT;

EXECUTE getIdPosition @position,

@idPosition OUTPUT;

INSERT INTO Employees

VALUES (@idSurname, @idName, @idPatronymic, @idPosition, @admission, @dismissal, @photo);

END

GO

--Процедура изменения работника по id

CREATE PROCEDURE updateEmployeeById @id int,

@surname varchar(50),

@name varchar(50),

@patronymic varchar(50),

@position varchar(255),

@admission varchar(30),

@dismissal varchar(30),

@photo varchar(255)

AS

BEGIN

IF (@id IS NOT NULL)

BEGIN

DECLARE @idSurname int;

DECLARE @idName int;

DECLARE @idPatronymic int;

DECLARE @idPosition int;

EXECUTE getIdSurname @surname,

@idSurname OUTPUT;

EXECUTE getIdName @name,

@idName OUTPUT;

EXECUTE getIdPatronymic @patronymic,

@idPatronymic OUTPUT;

EXECUTE getIdPosition @position,

@idPosition OUTPUT;

UPDATE Employees

SET idSurname = @idSurname,

idName = @idName,

idPatronymic = @idPatronymic,

idPosition = @idPosition,

admission = @admission,

dismissal = @dismissal,

photo = @photo

WHERE id = @id;

END

RETURN;

END

GO

--Процедура удаления работника по id

CREATE PROCEDURE deleteEmployeeById @id int

AS

BEGIN

DELETE FROM Employees

WHERE id = @id;

RETURN;

END

GO

--Показать работников

CREATE PROCEDURE getEmployees

AS

BEGIN

SELECT

Employees.id,

Surnames.name AS 'Surname',

Names.name AS 'Name',

Patronymics.name AS 'Patronymic',

Positions.name AS 'Position',

admission AS 'Admission',

dismissal AS 'Dismissal',

photo AS 'Photo'

FROM (((Employees

INNER JOIN Surnames

ON idSurname = Surnames.id)

INNER JOIN Names

ON idName = Names.id)

INNER JOIN Patronymics

ON idPatronymic = Patronymics.id)

INNER JOIN Positions

ON idPosition = Positions.id

RETURN;

END

GO

--Добавляем работников

EXECUTE addEmployee 'Аношкин',

'Василий',

'Иванович',

'Директор',

'№1 от 01.01.2017',

NULL,

'.\photos\anoshkinVA.jpg';

EXECUTE addEmployee 'Каренина',

'Анна',

'Петровна',

'Бухгалтер',

'№2 от 01.01.2017',

NULL,

'.\photos\kareninaAP.jpg';

EXECUTE addEmployee 'Прекрасная',

'Василиса',

'Васильевна',

'Менеджер по продажам',

'№3 от 02.01.2017',

NULL,

'.\photos\prekrasnajaVV.jpg';

EXECUTE addEmployee 'Рыжий',

'Петр',

'Игнатьевич',

'Грузчик',

'№6 от 06.01.2017',

'№12 от 19.02.2017',

'.\photos\ruzhiyPI.jpg';

EXECUTE addEmployee 'Беспалый',

'Игорь',

'Сергеевич',

'Экспедитор - грузчик',

'№18 от 24.02.2017',

NULL,

'.\photos\bespaliyIS.jpg';

--Добавляем пользователей в SQL Server.

EXECUTE sp\_addlogin @loginame = 'director',

@passwd = 'director';

EXECUTE sp\_addlogin @loginame = 'buhgalter',

@passwd = 'buhgalter';

EXECUTE sp\_addlogin @loginame = 'allusers',

@passwd = 'allusers';

--Добавляем пользователей в базу данных.

USE Employees;

EXECUTE sp\_grantdbaccess @loginame = 'director';

EXECUTE sp\_grantdbaccess @loginame = 'buhgalter';

EXECUTE sp\_grantdbaccess @loginame = 'allusers';

--Выдаём права на выполнение процедур

GRANT EXECUTE ON getEmployees TO director, buhgalter, allusers;

GRANT EXECUTE ON updateEmployeeById TO director, buhgalter;

GRANT EXECUTE ON addEmployee TO director, buhgalter;

GRANT EXECUTE ON deleteEmployeeById TO director;

***Program.cs***

using System;

using System.Linq;

using System.Windows.Forms;

namespace EmployeeCatalog

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainForm());

}

}

}

***App.config***

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

<configuration>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5.2" />

</startup>

<connectionStrings>

<add name="Employees" connectionString="Data Source=STYTAN-PC\SQLEXPRESS1;Initial Catalog=Employees"/>

</connectionStrings>

</configuration>

***Employee.cs***

using System;

using System.Linq;

namespace EmployeeCatalog

{

public class Employee

{

public int id { set; get; }

public string surname { set; get; }

public string name { set; get; }

public string patronymic { set; get; }

public string position { set; get; }

public string admission { set; get; }

public string dismissal { set; get; }

public string photo { set; get; }

}

}

***formLogin.cs***

/\*

\* Created by SharpDevelop.

\* User: sergey.lezhenko

\* Date: 22.08.2017

\* Time: 12:32

\*

\* To change this template use Tools | Options | Coding | Edit Standard Headers.

\*/

using System;

using System.Windows.Forms;

namespace EmployeeCatalog

{

/// <summary>

/// Запрос логина и пароля для входа в БД

/// </summary>

public partial class formLogin : Form

{

public formLogin()

{

InitializeComponent();

}

}

}

***MainForm.cs***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Configuration;

namespace EmployeeCatalog

{

public partial class MainForm : Form

{

/// <summary>

/// Форма входа

/// </summary>

formLogin fmLogin;

/// <summary>

/// Соединение с базой

/// </summary>

private SqlConnection connect;

public MainForm()

{

InitializeComponent();

this.Load += new EventHandler(MainForm\_Load);

}

private void MainForm\_Load(Object seneder, EventArgs e)

{

do

{

//Запрашиваем логин и пароль

fmLogin = new formLogin();

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

{

//Если не ОК - завершаемся

Environment.Exit(0);

}

//пока не подключимся к базе

} while (!SetupDbConnect());

//Обновляем таблицу

GetDataFormDB();

labelUser.Text += fmLogin.textBox1.Text;

}

/// <summary>

/// Проверяет возможность соединения с базой под заданными учетными данными

/// </summary>

/// <returns>true - соединение успешно</returns>

private bool SetupDbConnect()

{

//Задаём параметры подключения

connect = new SqlConnection(ConfigurationManager.ConnectionStrings["Employees"].ConnectionString);

//Задаём учётные данные

var pass = new System.Security.SecureString();

fmLogin.maskedTextBox1.Text.ToList().ForEach(pass.AppendChar);

pass.MakeReadOnly();

connect.Credential = new SqlCredential(fmLogin.textBox1.Text, pass);

//Проверяем возможность соединения

try

{

connect.Open();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return false;

}

finally { connect.Close(); }

return true;

}

/// <summary>

/// Получает данные из БД и заполняет таблицу

/// </summary>

private void GetDataFormDB()

{

SqlDataReader reader = null;

try

{

connect.Open();

SqlCommand com = new SqlCommand

{

//Процедура возвращает таблицу сотрудников

CommandText = "EXECUTE getEmployees",

Connection = connect

};

reader = com.ExecuteReader();

//Заполняем список значениями из базы

List<Employee> table = new List<Employee>();

while (reader.Read())

{

table.Add(new Employee

{

id = (int)reader["id"],

surname = (reader.IsDBNull(1) ? "" : reader.GetString(1)),

name = (reader.IsDBNull(2) ? "" : reader.GetString(2)),

patronymic = (reader.IsDBNull(3) ? "" : reader.GetString(3)),

position = (reader.IsDBNull(4) ? "" : reader.GetString(4)),

admission = (reader.IsDBNull(5) ? "" : reader.GetString(5)),

dismissal = (reader.IsDBNull(6) ? "" : reader.GetString(6)),

photo = (reader.IsDBNull(7) ? "" : reader.GetString(7)),

});

}

//Заполняем DataGridView

dataGridView1.DataSource = table;

//Скрываем столбцы id и photo

dataGridView1.Columns["id"].Visible = false;

dataGridView1.Columns["photo"].Visible = false;

}

catch (Exception e)

{

MessageBox.Show(e.Message, "DB Error");

}

finally

{

reader.Close();

connect.Close();

}

}

/// <summary>

/// Обновляет фото при выборе сотрудника

/// </summary>

void dataGridView1\_SelectionChanged(object sender, EventArgs e)

{

picBoxPhoto.ImageLocation = (string)(dataGridView1.CurrentRow.Cells["photo"].Value ?? "");

picBoxPhoto.Load();

}

/// <summary>

/// Удаляет сотрудника из базы

/// </summary>

void btnDelete\_Click(object sender, EventArgs e)

{

//Если выбрана строка

if (dataGridView1.SelectedRows.Count > 0)

{

try

{

connect.Open();

SqlCommand com = new SqlCommand

{

//Процедура удаляет строку из базы по её id

CommandText = "EXECUTE deleteEmployeeById " + (int)dataGridView1.CurrentRow.Cells["id"].Value,

Connection = connect

};

MessageBox.Show(com.ExecuteNonQuery() + " record deleted");

//Обновляем таблицу

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

connect.Close();

}

GetDataFormDB();

}

}

/// <summary>

/// Добавляет сотрудника в базу

/// </summary>

void btnAdd\_Click(object sender, EventArgs e)

{

//Форма добавления/редактирования сотрудника

editForm fmEdit = new editForm();

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

{

try

{

connect.Open();

//Процедура добавляет данные сотрудника по все таблицы БД

string comText = "EXECUTE addEmployee '"

+ fmEdit.textBoxSurname.Text + "', '"

+ fmEdit.textBoxName.Text + "', '"

+ fmEdit.textBoxPatronymic.Text + "', '"

+ fmEdit.textBoxPosition.Text + "', '"

+ fmEdit.textBoxAdmission.Text + "', '"

+ fmEdit.textBoxDismissal.Text + "', '"

+ fmEdit.pictureBox1.ImageLocation + "'";

SqlCommand com = new SqlCommand

{

CommandText = comText,

Connection = connect

};

MessageBox.Show(com.ExecuteNonQuery() + " record added");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

connect.Close();

}

GetDataFormDB();

}

}

/// <summary>

/// Редактирует данные выбранного сотрудника в БД

/// </summary>

void btnEdit\_Click(object sender, EventArgs e)

{

editForm fmEdit = new editForm();

DataGridViewCellCollection dc = dataGridView1.CurrentRow.Cells;

//Заполняем форму данными редактируемого сотрудника

fmEdit.textBoxSurname.Text = (string)(dc["surname"].Value ?? "");

fmEdit.textBoxName.Text = (string)(dc["name"].Value ?? "");

fmEdit.textBoxPatronymic.Text = (string)(dc["patronymic"].Value ?? "");

fmEdit.textBoxPosition.Text = (string)(dc["position"].Value ?? "");

fmEdit.textBoxAdmission.Text = (string)(dc["admission"].Value ?? "");

fmEdit.textBoxDismissal.Text = (string)(dc["dismissal"].Value ?? "");

fmEdit.pictureBox1.ImageLocation = (string)(dc["photo"].Value ?? "");

fmEdit.pictureBox1.Load();

fmEdit.Text = "Edit employee";

fmEdit.buttonAdd.Text = "Save";

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

{

try

{

connect.Open();

//Процедура обновляет данные сотрудника во всех таблицах БД по его id

string comText = "EXECUTE updateEmployeeById "

+ (int)dc["id"].Value + ", '"

+ fmEdit.textBoxSurname.Text + "', '"

+ fmEdit.textBoxName.Text + "', '"

+ fmEdit.textBoxPatronymic.Text + "', '"

+ fmEdit.textBoxPosition.Text + "', '"

+ fmEdit.textBoxAdmission.Text + "', '"

+ fmEdit.textBoxDismissal.Text + "', '"

+ fmEdit.pictureBox1.ImageLocation + "'";

SqlCommand com = new SqlCommand

{

CommandText = comText,

Connection = connect

};

MessageBox.Show(com.ExecuteNonQuery() + " record updated");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

connect.Close();

}

GetDataFormDB();

}

}

}

}

***editForm.cs***

/\*

\* Created by SharpDevelop.

\* User: sergey.lezhenko

\* Date: 30.08.2017

\* Time: 10:52

\*

\* To change this template use Tools | Options | Coding | Edit Standard Headers.

\*/

using System;

using System.Windows.Forms;

namespace EmployeeCatalog

{

/// <summary>

/// Description of editForm.

/// </summary>

public partial class editForm : Form

{

public editForm()

{

InitializeComponent();

}

/// <summary>

/// Загружает выбранное фото

/// </summary>

void buttonLoad\_Click(object sender, EventArgs e)

{

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

pictureBox1.ImageLocation = openFileDialog1.FileName;

pictureBox1.Load();

}

}

/// <summary>

/// Не позволяет сохранить данные без имени м фамилии

/// </summary>

void buttonAdd\_Click(object sender, EventArgs e)

{

if (textBoxSurname.Text == null || textBoxName.Text == null)

{

MessageBox.Show("Surname and Name must be filled", "Warning");

this.DialogResult = DialogResult.None;

}

}

}

}













