Liceul Tehnologic „Școala Națională de Gaz” Mediaș

Proiect pentru susținerea atestatului profesional

„CUNOAȘTEREA DE SINE”

2018

Profesor coordoator: Sîrbu Margareta Elev: Ciui Bianca Laura

Cuprins:

[1. Argument – Motivul și scopul proiectului 3](#_Toc513375988)

[2. Prezentarea generală a temei 4](#_Toc513375989)

[3. Resurse hardware și software necesare vizualizării proiectului 5](#_Toc513375990)

[4. Aspecte teoretice necesare realizării aplicației 6](#_Toc513375991)

[4.1. Programarea orientată pe obiect 6](#_Toc513375992)

[4.2. Baza de date Access 8](#_Toc513375993)

[4.3. Limbajul C# și platforma .NET 9](#_Toc513375994)

[5. Descrierea aplicației și a etapelor de creare 10](#_Toc513375995)

[6. Bibliografie și webografie 33](#_Toc513375996)

# Argument – Motivul și scopul proiectului

Numele atestatului meu este „Cunoașterea de sine”. Am ales această temă deoarece, pentru mine, este un subiect atractiv, iar informațiile necesare creării proiectului sunt ușor de găsit în cărți sau chiar pe Internet.

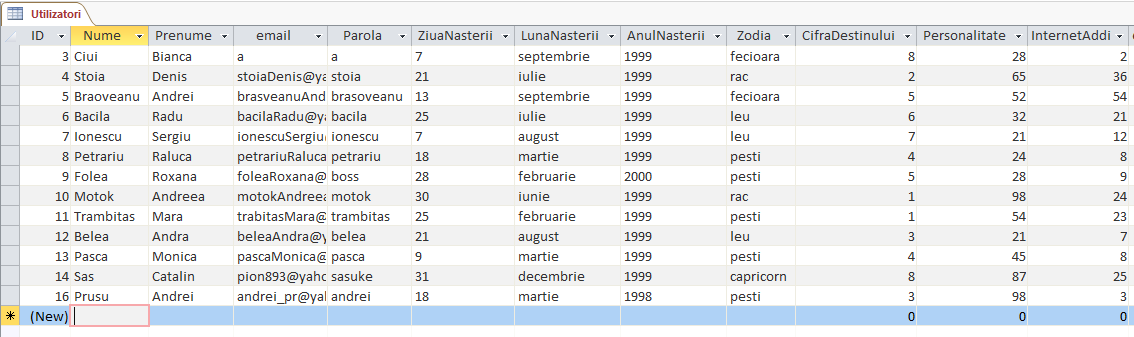
Motivul principal pentru care am creat acest proiect este interesul accentuat al oamenilor din jurul meu legat de astrologie și numerologie, fapt ce m-a determinat să mă documentez mai mult despre cele două domenii. De asemenea, în cadrul proiectului meu am inclus și un test de personalitate și unul despre dependența de internet, cele două fiind aspecte care mă fascinează. „Cine suntem?”, „De ce suntem așa?” și „Oare internetul are nevoie de mine, sau eu de internet?” sunt câteva întrebări la care aș vrea să primesc răspunsul, și consider că nu sunt singura care-și dorește asta.

Scopul proiectului meu este să-l fac pe cel care urmează pașii necesari înscrierii și testării să afle ceva nou despre el și, poate, să își pună și el, la rândul său, aceste întrebări. De asemenea, proiectul servește ca atestat profesional la informatică și poate deveni material didactic pentru generațiile următoare.

# Prezentarea generală a temei

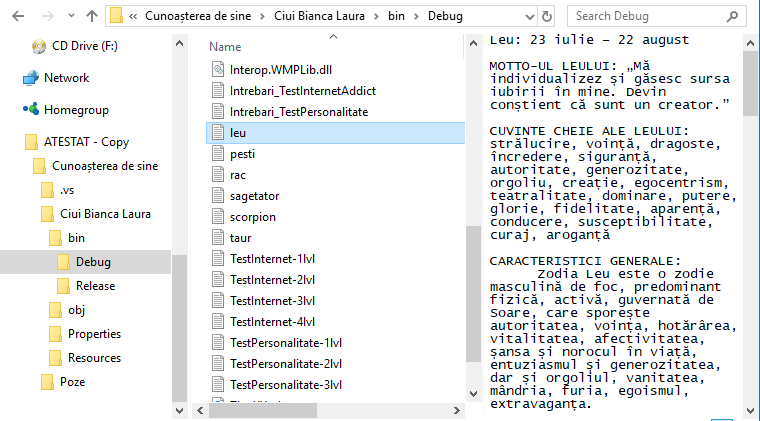
Tema atestatului meu este o aplicație care cuprinde o bază de date Access simplă, care reține date elementare despre utilizatori, legată de programul principal care are în vedere calcularea zodiei utilizatorului în funcție de data nașterii introdusă și calcularea cifrei destinului, iar mai apoi afișarea informațiilor esențiale pentru categoria respectivă.

Deoarece aplicația mea nu reține prea multe date despre utilizator, nu am avut nevoie decât de o tabelă – Utilizatori. Desigur, dacă s-ar dori extinderea aplicației mele la nivel de site sau de magazin online tematic, cu tricouri cu zodii, să zicem, baza de date ar trebui extinsă.



Programul meu se deschide cu o fereastră de autentificare care funcționează împreună cu baza de date din Access denumită „Utilizatori” (localizată pe calea Cunoașterea de sine\Ciui Bianca Laura\bin\Debug\Utilizatori.accdb pentru a face posibilă logarea și reținerea datelor personale ale celui care utilizează aplicația. În cazul în care nu ai un cont, se poate accesa fereastra pentru crearea unui cont nou care cuprinde informațiile necesare calculării zodiei și a cifrei destinului și afișării lor.

Afișarea rezultatelor pentru fiecare dintre funcțiile proiectului meu se face cu ajutorul unor fișiere aflate în același loc ca și baza mea de date:



Mai apoi, putem naviga în meniul care ne pune la dispoziție atât două teste (unul de personalitate și unul pe tema „Internet Addict”) și informații legate de modul de calcul și repartizare în categoriile specifice fiecărui test, cât și instrucțiunile necesare pentru a ajunge la un rezultat cât ai aproape de adevăr. Rezultatele sunt automat introduse și în baza de date, imediat ce se generează un rezultat.

# Resurse hardware și software necesare vizualizării proiectului

Pentru a putea vizualiza această aplicație, avem nevoie de un calculator care are instalat Microsoft Visual Studio 2015, sau orice altă versiune mai nouă care permite recunoașterea tuturor comenzilor. Versiunea menționată mai sus a fost utilizată în crearea, testarea și perfecționarea proiectului meu, iar fără acest program, deschiderea proiectului este imposibilă.



Resurse hardware necesare:

* 1.6 GHz sau mai mult
* 1 GB RAM
* 10 GB spațiu liber pe disk
* 5400 RPM
* DirectX 9 – placă video compatibilă

Resurse hardware necesare (oricare din următoarele):

* Windows 10
* Windows 8.1
* Windows 8
* Windows 7 SP 1
* Windows Server 2012 R2
* Windows Server 2012
* Windows Server 2008 R2 SP1

# Aspecte teoretice necesare realizării aplicației

Atestatul conține noțiuni de bază de cod SQL, cod C# și C++ și de baze de date Access pentru gestionarea elementelor din proiectul în Visual Studio 2015.

## Programarea orientată pe obiect

Programarea orientată pe obiect (POO și, în engleză, denumită Object Oriented Programming sau OOP) este o paradigmă de programare, axată pe ideea încapsulării, adică grupării datelor şi codului care operează asupra lor, într-o singură structură. Un alt concept important asociat programării orientate obiect este polimorfismul, care permite abstractizări care fac posibilă o descriere conceptuală mai simplă a soluţiei.

*Obiectele*

Un obiect este o reprezentare a unei entități din lumea reală asupra căruia se poate înterprinde o acțiune sau care poate întreprinde o acțiune.

Caracteristici:

* Nume;
* Atribute (date) – istanța înseamnă valorile atributelor la un moment de timp;
* Metode (servicii, operații).

*Clasele*

O clasă este o colecție de obiecte cu aceeași structură (caracteristici) și același comportament (metode sau operații).

Presupune:

* Definirea clasei;
* Implementarea clasei respective.

Principiile de bază ale POO sunt:

**Abstractizarea** **-** Este posibilitatea ca un program să ignore unele aspecte ale informaţiei pe care o manipulează, adică posibilitatea de a se concentra asupra esenţialului. Fiecare obiect în sistem are rolul unui "actor" abstract, care poate executa acţiuni, îşi poate modifica şi comunica starea şi poate comunica cu alte obiecte din sistem fără a dezvălui cum au fost implementate acele facilitaţi. Procesele, funcţiile sau metodele pot fi de asemenea abstracte, şi în acest caz sunt necesare o varietate de tehnici pentru a extinde abstractizarea:

* **Încapsularea** **-** numită şi ascunderea de informaţii: Asigură faptul că obiectele nu pot schimba starea internă a altor obiecte în mod direct (ci doar prin metode puse la dispoziţie de obiectul respectiv); doar metodele proprii ale obiectului pot accesa starea acestuia. Fiecare tip de obiect expune o interfaţă pentru celelalte obiecte care specifică modul cum acele obiecte pot interacţiona cu el.
* **Polimorfismul -** Este abilitatea de a procesa obiectele în mod diferit, în funcţie de tipul sau de clasa lor. Mai exact, este abilitatea de a redefini metode pentru clasele derivate. De exemplu pentru o clasă Figură putem defini o metodă arie. Dacă Cerc, Dreptunghi, etc. ce vor extinde clasa Figură, acestea pot redefini metoda arie.
* **Moştenirea -** Organizează şi facilitează polimorfismul şi încapsularea, permiţând definirea şi crearea unor clase specializate plecând de la clase (generale) deja definite - acestea pot impărtăşi (şi extinde) comportamentul lor, fără a fi nevoie de a-1 redefini. Aceasta se face de obicei prin gruparea obiectelor în clase şi prin definirea de clase ca extinderi ale unor clase existente. Conceptul de moştenire permite construirea unor clase noi, care păstrează caracteristicile şi comportarea, deci datele şi funcţiile membru, de la una sau mai multe clase definite anterior, numite clase de bază, fiind posibilă redefinirea sau adăugarea unor date şi funcţii noi. Se utilizează ideea: "Anumite obiecte sunt similare, dar în acelaşi timp diferite". O clasă moştenitoare a uneia sau mai multor clase de bază se numeşte clasă derivată. Esenţa moştenirii constă în posibilitatea refolosirii lucrurilor care funcţionează.

Ideea POO (Programare Orientată Obiectual) este de a crea programele ca o colecție de obiecte, unități individuale de cod care interacționează unele cu altele, în loc de simple liste de instrucțiuni sau de apeluri de proceduri. Obiectele POO sunt de obicei reprezentări ale obiectelor din viața reală (domeniul problemei), astfel încât programele realizate prin tehnica POO sunt mai ușor de înțeles, de depanat și de extins decât programele procedurale. Aceasta este adevărată mai ales în cazul proiectelor software complexe și de dimensiuni mari, care se gestionează făcând apel la ingineria programării.

## Baza de date Access

O bază de date reprezintă o colecție de informații care se referă la un anumit subiect sau scop, cum ar fi urmărirea comenzilor clienților sau menținerea ordinii într-o colecție de cărți.

Utilizând Microsoft Access, pot fi gestionate toate informațiile dintr-un singur fișier al bazei de date. Astfel, se utilizează:

* Tabele pentru memorarea datelor;
* Interogări pentru vizualizarea informațiilor selectate după anumite criterii;
* Formulare pentru vizualizarea, adăugarea și actualizarea datelor din baza de date;
* Rapoarte pentru analiza sau listarea datelor după anumite criterii.

O bază de date reprezintă fundația oricărei aplicații online care necesită memorarea datelor utilizatorilor și ea face posibil livelul tehnologizării noastre de acum și a dezvoltării continue a Internetului.

SQL (de la numele englez Structured Query Language - limbaj de interogare structurat) este un limbaj de programare specific pentru manipularea datelor în sistemele de manipulare a bazelor de date relaționale (RDBMS), iar la origine este un limbaj bazat pe algebra relațională. Acesta are ca scop inserarea datelor, interogații, actualizare și ștergere, modificarea și crearea schemelor, precum și controlul accesului la date. A devenit un standard în domeniu, fiind cel mai popular limbaj utilizat pentru crearea, modificarea, regăsirea și manipularea datelor de către SGBD-urile (Sistemele de Gestiune a Bazelor de Date) relaționale. Pe lângă versiunile standardizate ale limbajului, există o mulțime de dialecte și variante, unele proprietare, fiind specifice anumitor SGBD-uri și de asemenea conținând extensii pentru a suporta SBD-urile (Sistemele de Baze de Date) obiectuale (obiectual-relaționale).

## Limbajul C# și platforma .NET

Limbajul C# a fost dezvoltat în cadrul Microsoft de o echipă restrânsă şi are anumite caracteristici cum este de pildă suportul pentru internaţionalizare, adică posibilitatea de a scrie aplicaţii care pot fi adaptate cu uşurinţă pentru a fi utilizate în diferite regiuni ale lumii unde se vorbesc limbi diferite, fără să fie necesare pentru aceasta schimbări în arhitectura software. În strânsă legatură cu Arhitectura .NET (.NET Framework) pe care funcţionează, C# gestionează în mod automat memoria utilizată.

Eliberarea memoriei ocupate (garbage collection) de către obiectele care nu mai sunt necesare aplicaţiei este o facilitate importantă a limbajului. Programatorii nu mai trebuie să decidă singuri, aşa cum o fac, de pildă, în C++, care este locul şi momentul în care obiectele trebuie distruse. În C# se scriu de asemenea aplicaţii pentru sisteme complexe care funcţionează sub o mare varietate de sisteme de operare, cât şi pentru sisteme dedicate (embeded systems). Acestea din urmă se întind pe o arie largă, de la dispozitive portabile cum ar fi ceasuri digitale, telefoane mobile, MP3 player-e, până la dispozitive staţionare ca semafoare de trafic, sau controler-e pentru automatizarea producţiei.

Din punct de vedere sintactic, C# derivă din limbajul C++, dar include şi influenţe din alte limbaje, mai ales Java. C# este un limbaj simplu, cu circa 80 de cuvinte cheie și 12 tipuri de date predefinite. Limbajul permite programarea structurată, modulară și orientată obiect, conform principiilor moderne ale programării profesionale. Arhitectura .NET este o componentă software care oferă un mediu de programare şi de execuţie a aplicaţiilor pentru sistemele de operare Microsoft. .NET Framework este un mediu care permite dezvoltarea şi rularea aplicaţiftor şi a serviciilor Web, independente de platformă.

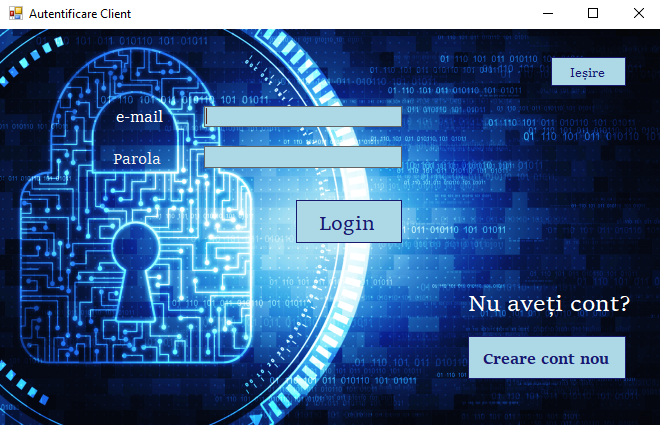
Limbajul C# a fost creat și dezvoltat de Microsoft pentru crearea codului platformei .NET, precum scopu creării limbajului C a fost acela de a implementa sistemul de operare UNIX. Platforma .NET pune la dispoziție utilizatorului o listă largă de clase organizate în biblioteci, incluse în structura limbajului C#.

Limbajul C# include două categorii de tipuri de date și anume tipuri valorice (spre exemplu: tipuri numerice care includ variabilele întregi, bool, decimale, mai apoi variabilele caracter, structuri, enumerări) și tipuri referință.

Elementele fundamentale programării C# sunt principiile de bază ale programării orientate obiect. O serie largă de tipuri noi de date a fost adăugată limbajului C# derivat din cel C++ pentru realizarea unor secvențe de cod „sigure”. Spre exemplu, unele funcții au fost adăugate (interfațe și delegări), altele au fost diversificare (structurile sau tipul struct), iar altele au fost modificare radical (tipul string) sau chiar eliminate (moștenirea multiplă și pointerii pentru funcții). Funcțiile preluate integral (fără vreo modificare) din C++ sunt considerate „nesigure”.

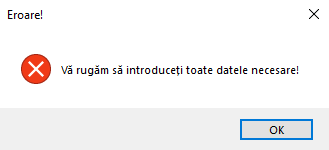
# Descrierea aplicației și a etapelor de creare

Aplicația mea debutează cu formularul de autentificare (Autentificare.cs). Fromularul conține două textBox-uri pentru introducerea datelor utilizatorului (emailTextBox și ParolaTextBox), în cazul în care acesta are un cont, și trei butoane: unul pentru a intra în cont (LoginButton), unul pentru deschiderea formularului de creare a unui nou cont (CreareContButton) și unul de pentru ieșirea și închiderea aplicației (iesireButton).

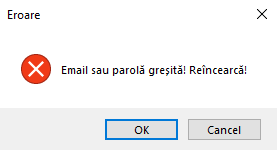


În momentul în care butonul LoginButton este apăsat, se compară informațiile introduse în cele două textBox-uri cu informațiile existente în baza de date inclusă în proiect. Ambele câmpuri trebuie completate și, de asemenea, trebuie să conțină valori-pereche din baza de date pentru a intra în contul respectiv.

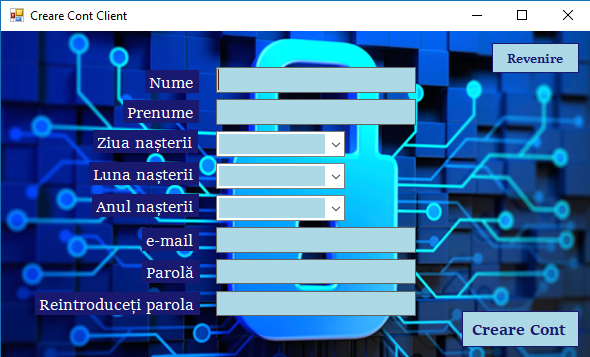
În cazul în care unul dintre cele două textBox-uri rămâne necompletat, va apărea pe ecran mesajul:



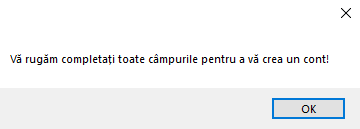
În cazul în care email-ul și parola nu există în baza de date într-o asemenea combinație, pe ecran va apărea mesajul:



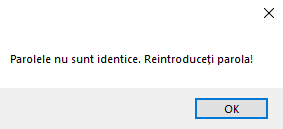
În cazul în care nu avem un cont, la apăsarea butonului Autentificare.CreareContButton se deschide o nouă pagină, un formular de înscriere (CreareContClient.cs). Aici avem un buton de revenire (IesireButton) în cazul în care vrem să abandonăm înscrierea și, cel mai important, avem textBox-urile și comboBox-urile care trebuie completate cu valori pentru a umple toate câmpurile obligatorii din baza de date (NumeTextBox, PrenumeTextBox, ZiuaComboBox, LunaComboBox, AnulComboBox, emailTextBox, ParolaTextBox și, pentru a ne asigura că nu am scris greșit parola și să nu ne putem accesa contul, ReintroducetiParolaTextBox).



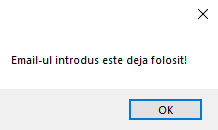
Dacă există un câmp care rămâne necompletat, contul nu se va crea, iar pe ecran va apărea mesajul:



În cazul în care se introduc valori diferite pentru cele două câmpuri referitoare la parolă, pe ecran va apărea un mesaj de atenționare, contul nu va fi creat și cele două textBox-uri pentru parolă se vor reseta.



De asemenea, se ia în calcul varianta ca noul utilizator sa încerce să folosească un email care deja există în baza de date. În acest caz pe ecran va apărea un mesaj de atenționare iar emailTextBox se va reseta, lăsându-l pe utilizator să introducă un alt email.



Pentru asta, am avut nevoie de o căutare în baza de date după email-ul introdus în emailTextBox; dacă în urma căutării, s-a returnat un ID diferit de 0, înseamnă că acel email există deja în baza de date. Mai jos avem codul pentru căutarea necesară:

//aici se face o căutare în DB după email-ul introdus, să vedem dacă acesta există deja

int idcontrol = 0;

string idcontr = "0";

string sqlcautare = "SELECT ID FROM Utilizatori " +"WHERE email='" + emailTextBox.Text+"'";

con.Open();

OleDbCommand cmdcautare = new OleDbCommand(sqlcautare, con);

OleDbDataReader rdrcautare = cmdcautare.ExecuteReader();

while (rdrcautare.Read())

{

idcontr = rdrcautare["ID"].ToString();

}

con.Close();

idcontrol = int.Parse(idcontr);//aici avem ID-ul utilizatorului, dacă acesta există

if (idcontrol!=0)

{

MessageBox.Show("Email-ul introdus este deja folosit!");

emailTextBox.Text = "";

}

După completarea corectă a tuturor câmpurilor, la apăsarea butonului de creare cont din formularul CreareContClient.cs (CreareContClient.CreareContButton), se realizează conexiunea cu baza de date a proiectului și un nou cont este creat, doar dacă cele două câmpuri legate de parolă au valorile introduse identice. Mai departe avem calea conexiunii și codul pentru introducerea valorilor completate în câmpuri în baza noastră de date:

con.ConnectionString = @"Provider = Microsoft.ACE.OLEDB.12.0; Data Source = " + Environment.CurrentDirectory + @"\Utilizatori.accdb";

//calea este aceeași pentru orice SQL folosit

//dacă parolele sunt ientice, se realizează preluarea datelor în DB

con.Open();

string sql = @"insert into Utilizatori (Nume,Prenume,email,Parola,ZiuaNasterii, LunaNasterii,AnulNasterii) " + " values ('" + NumeTextBox.Text + "','" + PrenumeTextBox.Text + "','" + emailTextBox.Text + "','" + ParolaTextBox.Text + "','" + ZiuaComboBox.Text + "','" + LunaComboBox.Text + "','" + AnulComboBox.Text + "')";

OleDbCommand cmd = new OleDbCommand(sql, con);

cmd.ExecuteNonQuery();

MessageBox.Show("Multumim pentru inregistrare!");

con.Close();

//s-a realizat înregistrarea și preluarea datelor în DB, se revine la Formul de autentificare

Close();

Autentificare f = new Autentificare();

f.Show();

Revenind acum la formularul Autentificare.cs, codul pentru a intra în aplicația implicită, în funcție de corectitudinea informațiilor introduse seamănă cu cel de la crearea contului fiindcă necesită, de asemenea, conexiunea la baza de date și un SQL de selectare:

con.Open();

string sql = "select ID from Utilizatori " +"where email='" + emailTextBox.Text + "'" + " and Parola='" + ParolaTextBox.Text + "'";

OleDbCommand cmd = new OleDbCommand(sql, con);

OleDbDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

ct++;

id = int.Parse(rdr[0].ToString());

User\_id.id = id;

}

if (ct > 0)

{

MessageBox.Show("Autentificare reușită!", "Bine ai venit!", MessageBoxButtons.OK, MessageBoxIcon.None);

MeniuPrincipal f = new MeniuPrincipal();

f.Show();

Hide();

}

else

{

MessageBox.Show("Email sau parola gresite! Reincearca!", "Eroare", MessageBoxButtons.OKCancel, MessageBoxIcon.Error);

//se golesc textboxurile pentru următoarea serie de valori introduse

emailTextBox.Text = ParolaTextBox.Text = "";

}

con.Close();

Se observă existența variabilei id, cu rolul de a extrage informații despre utilizatorul conectat în următoarele informații. Variabila este plasată în Program.cs pentru a fi recunoscută global, în toate formularele:

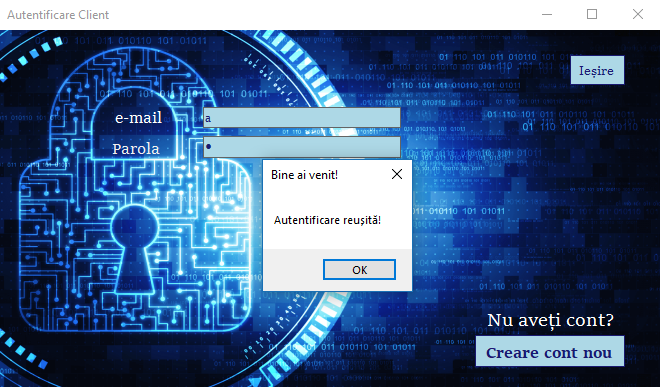
static class User\_id

{

public static int id;

}

După ce autentificarea s-a realizat corect, ne va apărea pe ecran următorul mesaj:



Apoi vom avea în față fereastra meniului principal (MeniuPrincipal.cs) care conține un toolStripMenu cu un buton de ieșire din aplicație (iesireToolStripMenuItem) și un buton de informații cu două opțiuni – despre zodie (InfoZodie.cs) și despre cifra destinului (InfoCifraDestinului.cs):



De asemenea aici avem un label de întâmpinare care preia din baza de date, în funcție de id-ul reținut în momentul autentificării, numele utilizatorului curent pentru o întâmpinare călduroasă și individualizată. Tot aici mai găsim și un label important (incearcaLabel) care face legătura cu formularului (AlegereaTestului.cs) responsabil cu prezentarea și cu deschiderea celor două teste – cel de personalitate (TestPersonalitate.cs) și testul „Internet Addict” (TestInternetAddict.cs).

În cadrul acestui formular (MeniuPrincipal.cs), în codul din spate, se preiau valori din baza de date și se calculează zodia și cifra destinului utilizatorului cu ajutorul datei nașterii și se actualizează valorile atributelor „Zodia” și „CifraDestinului” pe linia corespunzătoare utilizatorului, din baza de date.

Un exemplu de preluare de dată din baza de date (calea conexiunii este aceeași pentru toate deschiderile de conexiune, iar preluarea anului și a lunii necesită un cod similar):

OleDbCommand cmdziua = new OleDbCommand(sqlziua, con);

OleDbDataReader rdrziua = cmdziua.ExecuteReader();

while (rdrziua.Read())

{ //preluarea valorii zilei din DB

string ziua = rdrziua["ZiuaNasterii"].ToString();

idziua = Int32.Parse(ziua);//conversie de la string la integer

}

Calculul zodiei se realizează pe baza zilei nașterii și a lunii, stocate în variabilele de tip întreg idziua și respectiv idluna (care reține varianta numerică a lunii, nu cea scrisă literal, ca de exemplu „martie” – varianta reținută în baza de date). După secvența de calcul a zodiei, aceasta se stochează în variabila de tip string numită idzodia (aceasta va intra în construcția numelui fișierului selectat individual pentru fiecare utilizator în funcție de zodia în care se încadrează). Mai jos este o parte din secvența de calcul, reținere a zodiei și formare a fișierului respectiv:

public static string fisierzodie = "";//îl vom folosi pentru a încărca în richtextbox fișierul cu zodia potrivită

...

//aici este secvența care alege zodia corectă, în funcție de lună și zi

if ((idluna == 1 && idziua >= 20) || (idluna == 2 && idziua <= 18))

idzodia = "varsator";

if ((idluna == 2 && idziua >= 19) || (idluna == 3 && idziua <= 20))

idzodia = "pesti";

if ((idluna == 3 && idziua >= 21) || (idluna == 4 && idziua <= 20))

idzodia = "berbec";

...

fisierzodie = idzodia+".txt";

Mai jos avem actualizarea zodiei în baza de date, după calcularea ei:

// actualizare în DB a zodiei, imediat cum o aflăm

con.Open();

string sqlUpdateZodia = @"UPDATE Utilizatori set Zodia='" + idzodia + "' WHERE ID=" + User\_id.id;

OleDbCommand cmdUpdateZodia = new OleDbCommand(sqlUpdateZodia, con);

cmdUpdateZodia.ExecuteNonQuery();

con.Close();

Calculul cifrei destinului se realizează pe baza datei nașterii complete, prin adunarea repetată, succesivă a tuturor cifrelor care alcătuiesc data respectivă, până când se obține o singură cifră.

public static string fisiercifra = "";"";//îl vom folosi pentru a încărca în richtextbox fișierul cu cifra destinului potrivită

...

//calculul efectiv al Ciferi Destinului

int suma = idanul + idluna + idziua;

while (suma > 9)

suma = sumacifre(suma);//sumacifre returnează suma cifrelor unui număr dat

CifraDest = suma;

//aici se alege fișierul corespunzător Cifrei Destinului

fisiercifra = CifraDest.ToString() + ".txt";

Mai jos avem și actualizarea cifrei destinului în baza de date:

//actualizare în DB a Cifrei Destinului, imediat cum o aflăm

con.Close();

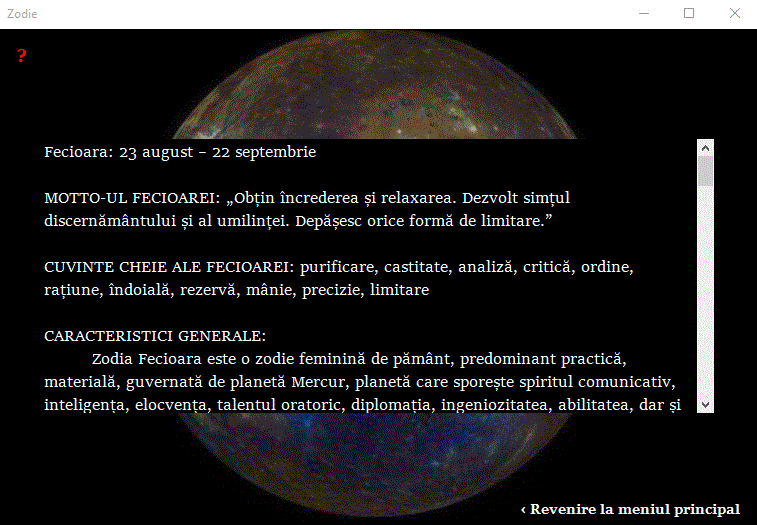
con.Open();

string sqlCifDest = @"UPDATE Utilizatori set CifraDestinului=" + CifraDest.ToString()+" WHERE ID="+User\_id.id;

OleDbCommand cmdCifDest = new OleDbCommand(sqlCifDest, con);

cmdCifDest.ExecuteNonQuery();

Butonul zodieButton (cel pătrat din stânga din ultima poză - MeniuPrincipal.cs) face legătura cu formularul Zodie.cs, responsabil cu afișarea fișierului corespondent zodiei utilizatorului în zodieRichTextBox. În funcție de zodia utilizatorului, pe fundal se va încărca o poză cu planeta specifică acesteia. În acest formular avem două label-uri, unul pentru redeschiderea meniului principal (MeniuPrincipal.cs) și altul, cu textul „?”, care este responsabil pentru deschiderea formului cu informații despre zodii și o mică istorie a lor (InfoZodie.cs).



În cadrul formularului Zodie.cs, odată cu încărcarea acestuia pe ecran, se execută instrucțiunile de cod necesare pentru încărcarea textului despre zodie și a pozei de fundal:

//aici se încarcă fișierul cu zodia corect, după luna și ziua nașterii

StreamReader fzod = new StreamReader(MeniuPrincipal.fisierzodie);

zodieRichTextBox.Text = fzod.ReadToEnd();

//aici, în funcție de fișierul curent, se actualizează imaginea de fundal cu planeta caracteristică zodiei

if (MeniuPrincipal.fisierzodie == "berbec.txt")

this.BackgroundImage = Properties.Resources.soare;

if (MeniuPrincipal.fisierzodie == "taur.txt")

this.BackgroundImage = Properties.Resources.luna;

...

Butonul CifraDestinuluiButton (cel pătrat din dreapta formularului MeniuPrincipal.cs) face legătura cu formularul CifraDestinului.cs, responsabil cu afișarea fișierului cifrei destinului corespondent utilizatorului, în cifraRichTextBox. În funcție de cifra destinului a utilizatorului, pe fundal se va încărca, din nou, o poză cu planeta guvernantă cifrei. În acest formular avem două label-uri, unul pentru redeschiderea meniului principal (MeniuPrincipal.cs) și altul, cu textul „?”, care este responsabil pentru deschiderea formului cu informații despre cifra destinului și o mică istorie numerologiei (InfoCifraDestinului.cs). Formularul arată așa:



În cadrul formularului CifraDestinului.cs, odată cu încărcarea acestuia pe ecran, se execută instrucțiunile de cod necesare pentru încărcarea textului despre cifră și a pozei de fundal:

//aici se afișează fișierul corespunzător pentru utilizator

StreamReader fcif = new StreamReader(MeniuPrincipal.fisiercifra);

cifraRichTextBox.Text = fcif.ReadToEnd();

//aici, în funcție de fișierul curent, se actualizează imaginea de fundal cu planeta caracteristică cifrei destinului

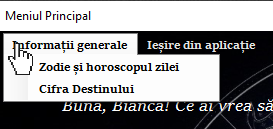
if (MeniuPrincipal.fisiercifra == "1.txt")

this.BackgroundImage = Properties.Resources.soare;

if (MeniuPrincipal.fisiercifra == "2.txt")

this.BackgroundImage = Properties.Resources.luna;

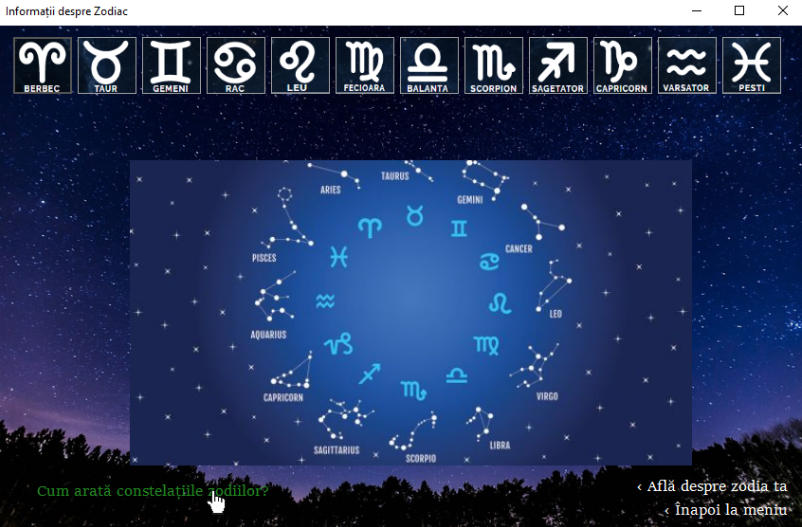
Să vorbim și despre formularele InfoZodie.cs și InfoCifraDestinului.cs. Mai jos avem butoanele care aparțin MenuStrip-ului din MeniuPrincipal.cs; ele sunt responsabile cu deschiderea acestor două formuri.



Să începem cu InfoZodie.cs. Aici avem un label cu o scurtă istorie despre zodiac și cum a fost inventat, avem alte două label-uri din partea de jos a formularului (de revenire la meniul principal – label1 – și de legătură cu formularul Zodie.cs – label5). Formularul arată ca cel de mai jos:



Dacă utilizatorul dorește să vadă cum arată constelațiile specifice zodiilor, poate face lucrul acesta trecând cu mouse-ul peste textul din partea de jos-stânga. Textul din spate (cel cu povestea despre zodiac) dispare, lăsând să apară în prim-plan o poză cu constelațiile, cum se întâmplă și se vede mai jos:



Dacă utilizatorul dorește să citească horoscopul zilei, poate face asta apăsând pe butoanele din partea de sus a ecranului, care fac legătura cu un site. Când utilizatorul trece cu mouse-ul pe deasupra butoanelor, apare pe ecran un mesaj specific zodiei de pe buton:



În momentul în care se face click pe unul dintre butoanele cu zodii pe ele, se deschide pagina următoare, cu horoscopul zilei, desigur, cu zodia butonului respectiv:



Pentru a face legătura între butoanele acestea și site-urile corespunzătoare, am avut nevoie de biblioteca System.Diagnostics; legătura s-a realizat astfel:

//instrucțiuni care fac legătura între butoane si site-urile caracteristice fiecăruia

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

{

Process.Start("https://www.horoscop.ro/berbec/");

}

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

{

Process.Start("https://www.horoscop.ro/taur/");

}

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

{

Process.Start("https://www.horoscop.ro/gemeni/");

}

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

{

Process.Start("https://www.horoscop.ro/rac/");

}

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

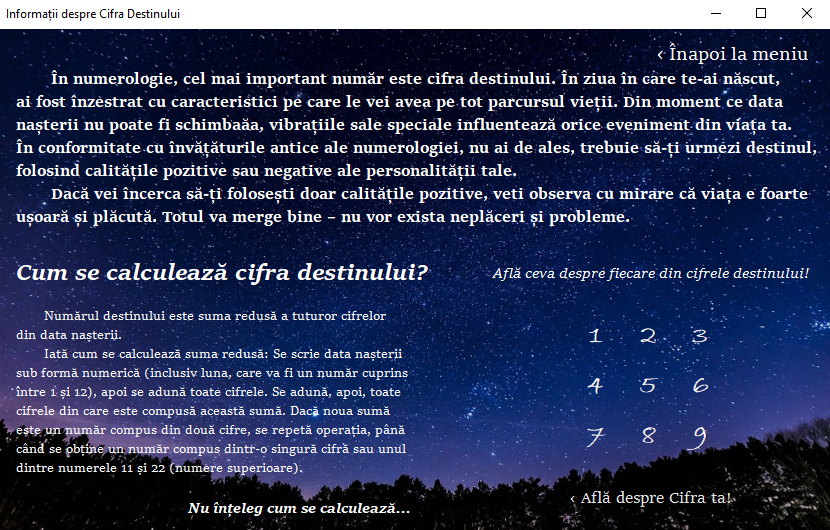
{

Process.Start("https://www.horoscop.ro/leu/");

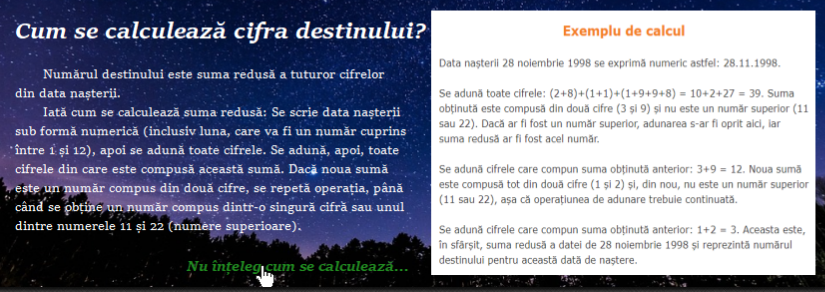
}

...

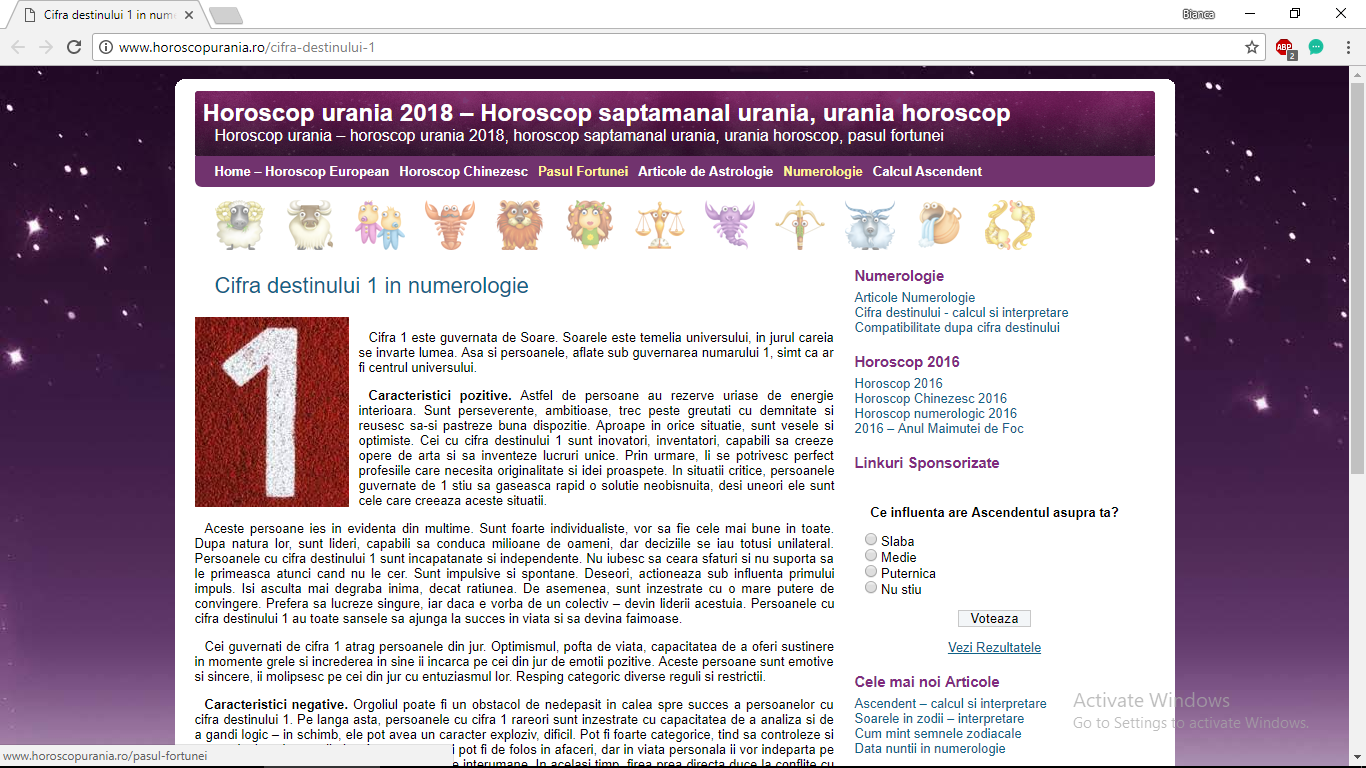
Mai apoi avem formularul InfoCifraDestinului.cs. Aici, la fel ca în formularul de dinainte, avem câteva label-uri cu un scurt text despre numerologie și cu o explicație legată de modul de calcul al cifrei destinului. De asemenea avem un label care, la apăsare, ne duce înapoi la MeniuPrincipal.cs și altul care face legătura cu formularul CifraDestinului.cs. Formularul arată așa:



Dacă utilizatorul nu este convins de modul de calcul, poate trece cu mouse-ul peste label-ul din partea de jos a ecranului – label5 – și un exemplu concret va apărea pe ecran, iar butoanele din partea dreaptă a ecranului vor deveni invizibile ca mai jos:



După ce mouse-ul părăsește acel label, poza dispare și apar din nou butoanele cu cifre, legate de câte un site despre fiecare cifră a destinului și label-ul de legătură cu formularul CifraDestinului.cs; astfel, dacă se apasă butonul button1, se va deschide pagina corespunzătoare cifrei 1:



Pentru ca asta să se întâmple, am avut nevoie din nou, ca mai devreme, de biblioteca System.Diagnostics, iar mai jos avem o parte din secvența de cod responsabilă pentru deschiderea paginilor:

//instrucțiuni de legătura a butoanelor la site-urile potrivite

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

{

Process.Start("http://www.horoscopurania.ro/cifra-destinului-1");

}

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

{

Process.Start("http://www.horoscopurania.ro/cifra-destinului-2");

}

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

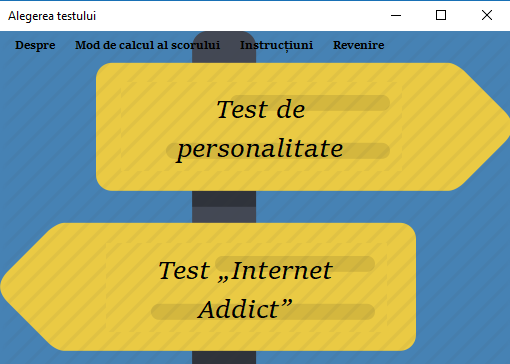
{

Process.Start("http://www.horoscopurania.ro/cifra-destinului-3");

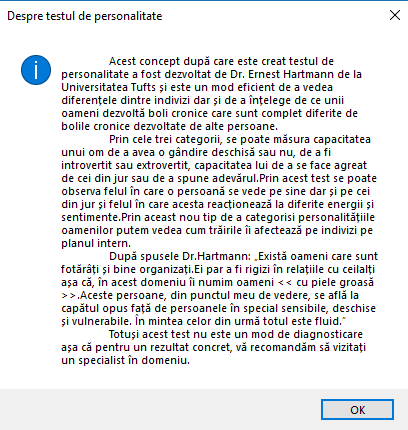
}

...

Labelul incearcaLabel din formularul MeniulPrincipal.cs deschide, la apăsare, formularul AlegereaTestului.cs. Acest formular conține un menuStrip1 cu butoane care deschid MessageBox-uri cu informații despre cele două teste, modul de calcul, instrucțiunile necesare pentru a primi un rezultat fidel realității și butonul de revenire la MeniuPrincipal.cs. Mai jos avem aspectul formularului AlegereaTestului.cs:



La deschiderea celor două opțiuni ale butonului despreToolStripMenuItem, putem afla informații despre cele două teste:

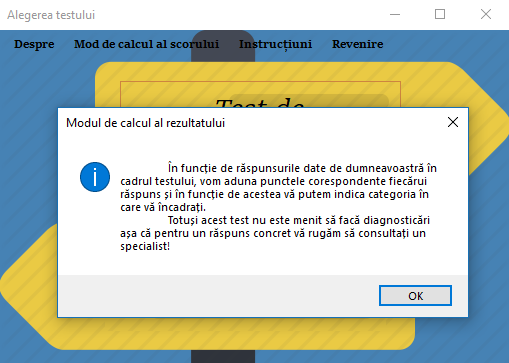


Pentru pregătirea rezultatelor celor două teste, am avut nevoie de o variabilă declarată global, ca să o putem folosi și modifica și în formularele TestPersonalitate.cs și TestInternetAddict.cs.

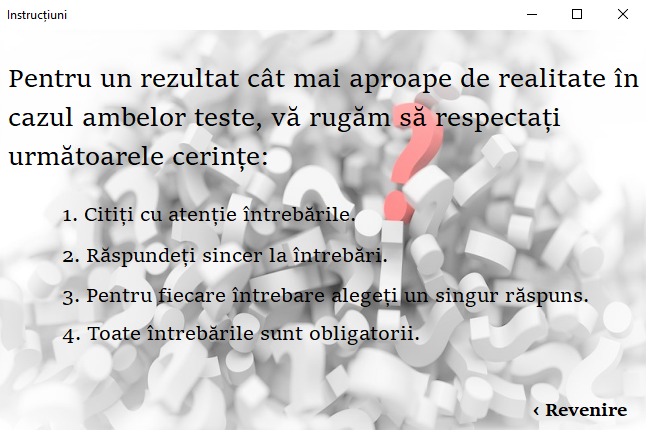
//acest total este folosit pentru ambele teste, pentru generarea rezultatelor

public static int total = 0;

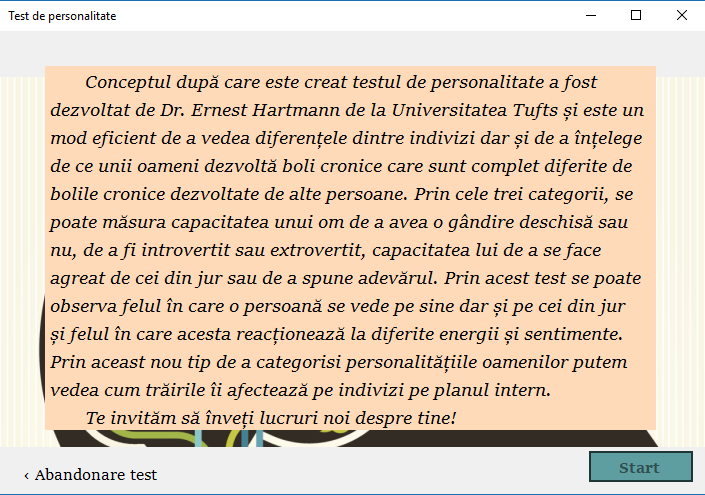
Modul de calcul al scorului poate fi vizualizat la apăsarea butonului modDeCalculAlScoruluiToolStripMenuItem, ca mai jos:



La apăsarea butonului din MenuStrip denumit instructiuniToolStripMenuItem se deschide formularul Instructiuni.cs, unde sunt prezentate cele patru reguli care trebuie respectate pentru a ajunge la un rezultat fidel realității.



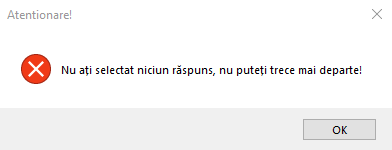
Butonul testPersonalitateButton din formularul AlegereaTestului.cs deschide un nou formular: TestPersonalitate.cs. Label-ul label1 este vizibil pe tot parcursul testului pentru a putea abandona progresul făcut până atunci, iar totalul se anulează. La începutul testului ne apare pe ecran o scurtă istorie a conceptului după care a fost creat acest test, butonul de start (StartButton) fiind singurul vizibil, ca mai jos:



Când butonul StartButton este apăsat, acesta dispare și apare unul nou – AltaIntrebareButton (care are ca scop avansarea în test și trecerea la următoarea întrebare; întrebările sunt preluate din fișierul denumit „Intrebari\_TestPersonalitate”, aflat lângă toate celelalte fișiere care rețin informații despre rezultate, întrebări și zodii/cifre ale destinului) – și richTextBox1 și groupBox1 care afișează întrebările, respectiv variantele de răspuns:



Fiecărui răspuns îi corespunde un anume punctaj care se adună în AlegereaTestului.total. E necesară alegerea unui răspuns pentru fiecare întrebare pentru a ajunge la rezultat și astfel, dacă pentru o întrebare nu se selectează nicio variantă de răspuns, pe ecran se va afișa:



Pentru încărcarea unei noi întrebări din listă, vom avea nevoie de următoarele declarări și un subprogram care e responsabil cu încărcarea unei noi întrebări:

List<string> linii = new List<string>();

int i = -1;//pentru liniile din listă

//încărcarea fiecărei întrebări pe rând

private void IncarcareContinut()

{

richTextBox1.Text = linii[++i];

}

Mai jos avem o parte din codul pentru butonul AltaIntrebareButton în cazul în care acesta este apăsat:

//totalizarea punctelor pentru fiecare întrebare

if (radioButton1.Checked == true)

AlegereaTestului.total += 0;

if (radioButton2.Checked == true)

AlegereaTestului.total += 1;

if (radioButton3.Checked == true)

AlegereaTestului.total += 2;

if (radioButton4.Checked == true)

AlegereaTestului.total += 3;

if (radioButton4.Checked == true)

AlegereaTestului.total += 4;

//cât timp nu s-a ajuns la ultima întrebare

if (i < linii.Count)

IncarcareContinut();

if (i==15) //când se ajunge la ultima întrebare

{

AltaIntrebareButton.Visible = false;

FinalizareTestButton.Visible = true;

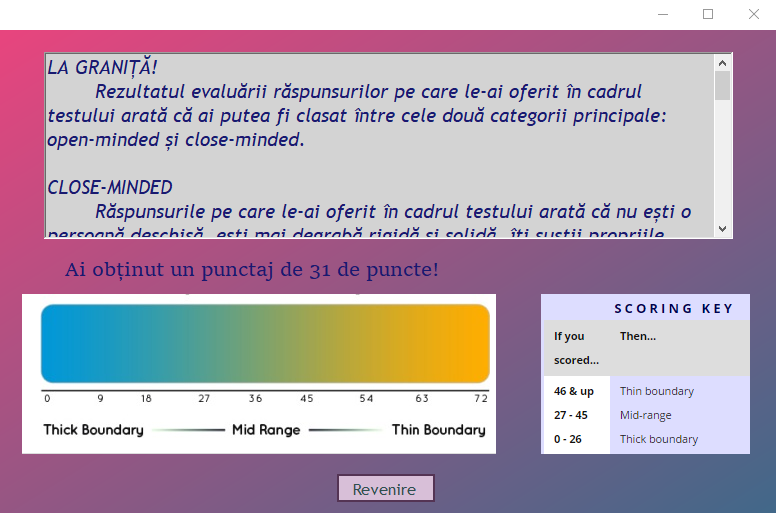
groupBox1.Visible = false;

}

Când se ajunge la ultima întrebare, butonul AltaIntrebareButton devine invizibil, butonul FinalizareTestButton devine vizibil, iar în loc de o nouă întrebare, apare pe ecran un mesaj de mulțumire pentru participare, ca mai jos:



La apăsarea butonului de finalizare, se deschide formularul RezultatePersonalitate.cs.



Acest formular (RezultatePersonalitate.cs) conține un richTextBox1 în care se încarcă automat rezultatul din fișierele aflate în resursele proiectului, în funcție de punctajul reținut în AlegereaTestului.total.

Repartizarea se face astfel, în funcție de punctaj:

//aici rezultatul, în funcție de totalul obținut, se încadrează într-o categorie

if (AlegereaTestului.total<27)

{

StreamReader f = new StreamReader("TestPersonalitate-1lvl.txt");

string s = f.ReadToEnd();

richTextBox1.Text = s;

}

if (AlegereaTestului.total >= 27 && AlegereaTestului.total<=45)

{

StreamReader f = new StreamReader("TestPersonalitate-2lvl.txt");

string s = f.ReadToEnd();

richTextBox1.Text = s;

}

if (AlegereaTestului.total>45)

{

StreamReader f = new StreamReader("TestPersonalitate-3lvl.txt");

string s = f.ReadToEnd();

richTextBox1.Text = s;

}

De asemenea, punctajul obținut în cadrul testului trebuie reținut și în baza de date. Asta se întâmplă în următoarele linii de cod, unde se actualizează baza de date:

//actualizarea datelor obținute în evaluarea răspunsurilor în DB

con.Open();

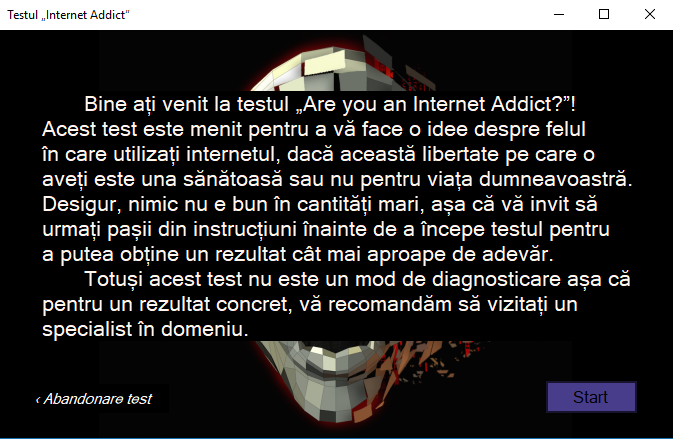
string sql = @"UPDATE Utilizatori set Personalitate=" + AlegereaTestului.total + @"WHERE ID=" + User\_id.id;

OleDbCommand cmd = new OleDbCommand(sql, con);

cmd.ExecuteNonQuery();

con.Close();

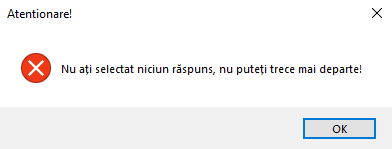
Butonul RevenireButton ne întoarce la pagina meniului principal (MeniuPrincipal.cs). Aici mai avem butonul TestInternetAddictButton care deschide formularul cu testul „Internet Addict” – TestInternetAddict.cs. Aici, la deschiderea formularului, avem un mic text introductiv, butonul de start – StartButton – și un buton vizibil pe tot parcursul testului, pentru a putea abandona testarea oricând și pentru a reseta progresul și punctajul obținut până atunci - AbandonareButton.



La apăsarea butonului StartButton, mesajul introductiv și butonul de start devin invizibile, și fac loc pentru richTextBox1 (responsabil cu afișarea întrebărilor, în ordine), groupBox1 (responsabil pentru afișarea variantelor de răspuns) și butonul AltaaIntrebareButton (responsabil pentru avansul în lista de întrebări existentă în fișierul denumit „Intrebari\_TestInternetAddict”), ca mai jos:



Fiecărui răspuns îi corespunde un anume punctaj care se adună în AlegereaTestului.total. E necesară alegerea unui răspuns pentru fiecare întrebare pentru a ajunge la rezultat și astfel, dacă pentru o întrebare nu se selectează nicio variantă de răspuns, pe ecran se va afișa:



Pentru încărcarea unei noi întrebări din listă, vom avea nevoie de următoarele declarări:

List<string> linii = new List<string>();

int i = -1;//pentru liniile din listă

//încărcarea fiecărei întrebări pe rând

private void IncarcareContinut()

{

richTextBox1.Text = linii[++i];

}

Mai jos avem o parte din codul pentru butonul AltaaIntrebareButton în cazul în care acesta este apăsat:

//totalizarea punctelor pentru fiecare întrebare

if (radioButton2.Checked == true)

AlegereaTestului.total += 1;

if (radioButton3.Checked == true)

AlegereaTestului.total += 2;

//cât timp nu se ajunde la ultima întrebare

if (i < linii.Count)

IncarcareContinut();

//când se ajunge la ultima întrebare

if (i == 14)

{

AltaaIntrebareButton.Visible = false;

FinalizareeButton.Visible = true;

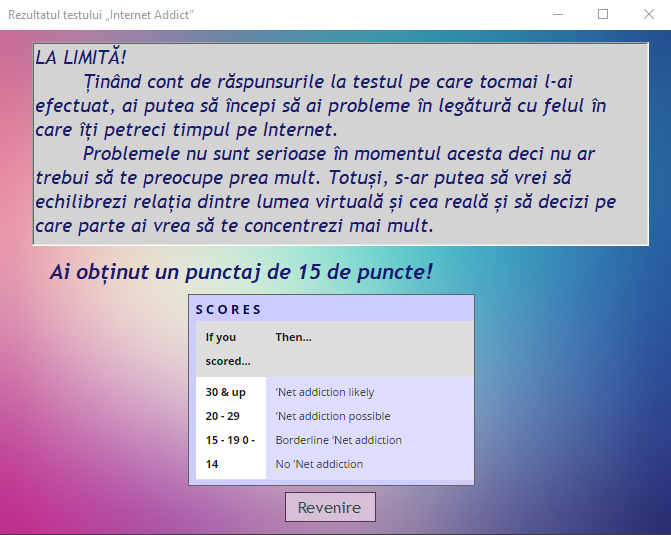
groupBox1.Visible = false;

}

Când se ajunge la ultima întrebare, butonul AltaaIntrebareButton devine invizibil, butonul FinalizareeButton devine vizibil, iar în loc de o nouă întrebare, apare pe ecran un mesaj de mulțumire pentru participare, ca mai jos:



La apăsarea butonului de finalizare, un nou formular - RezultateInternetAddict.cs - se deschide:



Acest form conține un richTextBox1 în care se încarcă automat rezultatul din fișierele aflate în resursele proiectului, în funcție de punctajul reținut în AlegereaTestului.total.

Repartizarea se face astfel, în funcție de punctaj:

//aici rezultatul, în funcție de totalul obținut, se încadrează într-o categorie, se afișează pe ecran fișierul corespunzător

if (AlegereaTestului.total <= 14)

{

StreamReader f = new StreamReader("TestInternet-1lvl.txt");

string s = f.ReadToEnd();

richTextBox1.Text = s;

}

if (AlegereaTestului.total >= 15 && AlegereaTestului.total <= 19)

{

StreamReader f = new StreamReader("TestInternet-2lvl.txt");

string s = f.ReadToEnd();

richTextBox1.Text = s;

}

if (AlegereaTestului.total >= 20 && AlegereaTestului.total<=29)

{

StreamReader f = new StreamReader("TestInternet-3lvl.txt");

string s = f.ReadToEnd();

richTextBox1.Text = s;

}

if (AlegereaTestului.total >= 30)

{

StreamReader f = new StreamReader("TestInternet-4lvl.txt");

string s = f.ReadToEnd();

richTextBox1.Text = s;

}

De asemenea, punctajul obținut în cadrul testului trebuie reținut și în baza de date. Asta se întâmplă în următoarele linii de cod:

//actualizarea datelor obținute în evaluarea răspunsurilor în DB

con.Open();

string sql = @"UPDATE Utilizatori set InternetAddict=" + AlegereaTestului.total+@" WHERE ID="+User\_id.id;

OleDbCommand cmd = new OleDbCommand(sql, con);

cmd.ExecuteNonQuery();

con.Close();

Butonul RevenireButton ne întoarce la pagina meniului principal (MeniuPrincipal.cs).

# Bibliografie și webografie

<https://ro.wikipedia.org/wiki/Programare_orientat%C4%83_pe_obiecte>

<http://www.ls-infomat.ro/user/content/e9efcsharp.pdf>

<https://www.scribd.com/document/324327179/Curs-de-C-Programare-in-Visual-C-2008-Express-Edition-pdf>

<https://ro.wikipedia.org/wiki/SQL>