Specificațiile Cerințelor

Software

**APLICAȚIE DE CALORIE COUNTER**

**versiunea 1.0**

**Mai, 2023**

Autori: Luchian Alexandru

Ivanov Ioan

Teslaru Victor

Grupa: 1311A

Universitatea Tehnica “Gheorghe Asachi” Iasi  
 Facultatea de Automatica si Calculatoare

Cuprins

1. Introducere

Scop

Publicul tinta si sugestii de lectura

Descriere produs

Referinte

2. Descriere generala  
 Perspectiva produsului

Functionalitati ale produsului

Categorii de utilizatori si caracteristici

Mediu de operare

Constrangeri de proiectare si implementare

Presupuneri si dependente

3. Cerinte de interfata externa

Interfete utilizator

Interfete hardware

Interfete software

Interfete de comunicare

Functionalitatea de autentificare

Functionalitatea de inregistrare

Adauga

Vezi progres

Iesire

Cerinte de performanta

4. Diagrame UML

Diagrama de utilizare a aplicatiei

Diagrama de clase

Diagrama de activitate

5. Cum sa utilizati aplicatia

6. Exemple de executie a aplicatiei

7. Exemple din codul sursa asociat

1. Introducere

# Scop

Scopul acestui document este de a oferi o descriere detaliată a cerințelor pentru aplicația noastră Calorie Counter. Aceasta este o aplicație mobilă în care poți înregistra și urmări aportul tău caloric zilnic. Poți introduce alimentele consumate, iar aplicația îți va calcula totalul caloriilor consumate. De asemenea, poți monitoriza progresul tău și obiectivele de pierdere în greutate prin intermediul unei ferestre de statistici. Aplicația nu necesită conexiune la internet, deoarece utilizează stocarea locală pentru a păstra informațiile introduse de utilizator prin intermediul interfeței.

# Publicul țintă și sugestii de lectură

Acest document este destinat atât utilizatorilor, cât și dezvoltatorilor. Utilizatorii vor găsi informații despre modul de utilizare a aplicației, iar aceștia pot citi secțiunile 3 și 4 ale acestui document. În ceea ce privește dezvoltatorii, aceștia ar trebui să citească întregul document pentru a obține o înțelegere mai profundă și mai bună a aplicației și să acorde o atenție deosebită secțiunii 2 a documentației.

# Descriere produs

Calorie Counter este o aplicație dezvoltată pe platforma .Net. Este creată într-un mod astfel încât utilizatorul să nu necesite cunoștințe avansate de calculatoare pentru a utiliza aplicația, datorită interfeței grafice simple, cu controale sugestive. Se adresează publicului care dorește să își monitorizeze aportul caloric zilnic, în vederea gestionării greutății corporale și menținerii unui stil de viață sănătos. Aplicația permite utilizatorului să introducă alimentele consumate și îi calculează totalul caloriilor consumate. De asemenea, oferă opțiunea de a urmări progresul și obiectivele de pierdere în greutate prin intermediul unei ferestre de statistici. Aplicația utilizează o interfață intuitivă și nu necesită conexiune la internet, deoarece utilizează stocarea locală pentru a păstra informațiile introduse de utilizator. Aplicația dispune de un mecanism de autentificare care diferențiază utilizatorii intre ei fiecare avand nevoi si scopuri diferite.

# Referințe

Informațiile nutriționale și designul interfeței au fost preluate din surse oficiale și resurse specializate în domeniul nutriției pentru a oferi aplicației un aspect și o funcționalitate corespunzătoare.

2. Descriere generală

# Perspectiva produsului

Calorie Counter este un proiect academic implementat ca o aplicație Windows Forms folosind limbajul de programare C# și .NET Framework 4.7.2. Datele necesare pentru această aplicație sunt stocate într-o bază de date T-SQL, în 3 tabele, una pentru alimentele si informatiile nutritionale, una pentru monitorizarea zilei curente pentru utilizator si caloriile consumate in acea zi si una pentru a calcula greutatea ideala si caloriile pe zi.

Aplicația principală utilizează două fișiere DLL (Dynamic Link Library), unul pentru gestionarea bazei de date numit DB\_DLL, un modul care facilitează interacțiunea între aplicație și baza de date T-SQL și unul pentru funcționalitatea de contorizare a caloriilor, numit Facade\_DLL, un modul care utilizează design pattern-ul Facade pentru a defini fiecare categorie de alimente și funcționalitatea asociată.

Există module de testare unitară pentru Calorie Counter și pentru Database. Punctul de intrare al aplicației este reprezentat de CalorieCounter.exe.

# Funcționalități ale produsului

Calorie Counter oferă următoarele funcții majore pentru utilizatorii finali:

1. Un mecanism de autentificare pentru o mai bună securitate;
2. Posibilitatea de a înregistra și monitoriza aportul caloric zilnic;
3. Posibilitatea de a introduce alimentele consumate și de a calcula totalul caloriilor consumate;
4. Posibilitatea de a vizualiza progresul și de a stabili obiective de pierdere în greutate prin intermediul unei ferestre de statistici;

# Categorii de utilizatori și caracteristici

Aplicația este concepută pentru ușurința în utilizare, astfel încât orice utilizator care dorește să își monitorizeze aportul caloric zilnic ar trebui să poată utiliza aplicația. Aceasta intenționează să ofere funcționalități pentru utilizatorul obișnuit, care poate fi reprezentat de oricine, indiferent de nivelul de cunoștințe în domeniul nutriției sau al calculatoarelor.

# Mediu de operare

Deoarece Contorul de Calorii este dezvoltat folosind .NET Framework, este compatibil doar cu versiunile de sistem de operare Windows începând cu Windows 7. De asemenea, este necesară instalarea .NET Framework 4.7.2 sau a unei versiuni mai noi pe sistemul de operare.

# Constrângeri de proiectare și implementare

Deoarece Calorie Counter este proiectat și implementat într-un singur semestru ca proiect pentru disciplina Ingineria Programării la universitatea noastră, este posibil ca timpul să fie principala limitare în acest ciclu de dezvoltare, precum și constrângerea tehnologică de a dezvolta totul pentru platforma .NET, amplificată de lipsa noastră de experiență în limbajul C# și în proiectarea și documentarea software.

# Presupuneri și dependențe

Vom presupune că va exista cel puțin un cont de administrator preconfigurat, altfel aplicația ar fi imposibil de utilizat. Ulterior, administratorul poate oferi acces și altor utilizatori.

3. Cerințe de interfață externă

# Interfețe utilizator

La pornirea aplicației, utilizatorul va fi întâmpinat de o fereastră de autentificare cu două câmpuri în care să introducă datele de autentificare ale contului. Dacă datele introduse nu corespund unui utilizator existent, va fi afișat un mesaj de eroare. Dacă datele introduse sunt corecte, fereastra principală a programului este afișată. Exista de asemenea optiunea inregistrarii unui nou utilizator, care va trebui sa ofere informatii legate de varsta, greutatea inaltimea, sexul si nivelul sau de activitate, ulterior stabilind un scop. Fereastra conține un progress bar ce reprezinta bugetul caloric si care se modifica in functie de alimentele consumate in ziua respectiva si un buton pentru adaugarea unei mese noi.

# Interfețe hardware

Interfața hardware trebuie să includă un mouse, o tastatură și un monitor de afișare. Nu este necesară nicio altă componentă hardware de mare capacitate, deoarece aplicația poate rula pe orice calculator cu versiunea 4.7.2 sau mai nouă a .NET Framework. Pentru a vă autentifica în fereastra principală, trebuie să utilizați tastatura pentru a tasta numele de utilizator și parola, iar apoi puteți folosi click-ul stânga al mouse-ului pentru a naviga cu ușurință în cadrul aplicației.

# Interfete software

Pentru a putea rula aplicația, sistemul utilizatorului trebuie să ruleze pe un sistem de operare Windows 7 sau o versiune mai recentă și trebuie să aibă instalată versiunea 4.7.2 sau o versiune mai nouă a .NET Framework.

# Interfețe de comunicare

Nu este necesară o standard de comunicare pentru ca aplicația să funcționeze, deoarece nu necesită conexiune la internet pentru a rula.

# Funcționalitatea de autentificare

Aplicația permite unui utilizator existent să se autentifice în contul său într-un mod rapid și securizat, prin introducerea datelor de autentificare existente, nume de utilizator și parolă. Apăsând butonul "Trimite", dacă datele de autentificare sunt corecte, utilizatorul este redirecționat către fereastra principală a aplicației.

# Funcționalitatea de inregistrare

Aplicatia permite unui utilizator nou sa isi creeze un cont in care ofera informatii legate de varsta, greutatea, inaltimea, sexul, nivelul de activitate si scopul pe care il are. Apăsând butonul "Inregistrare" utilizatorul este redirecționat către fereastra principală a aplicației.

# Adauga

După selectarea radio button-ului corespunzator mesei ce urmeaza sa fie introdusa si apăsarea butonului, o nouă fereastră va apărea. Aceasta va conține o optiune de cautare de alimente din baza de date si selectarea unui gramaj in functie de care vor fi afisate informatii nutritionale, urmand ca prin confirmare, caloriile mesei sa fie sustrase din bugetul caloric total.

# Vezi progres

Fereastra principala are si capacitatea de a arata un grafic al caloriilor consumate in zilele anterioare cu scopul de a arata evolutia utilizatorului in drumul sau spre o viata mai sanatoasa.

# Iesire

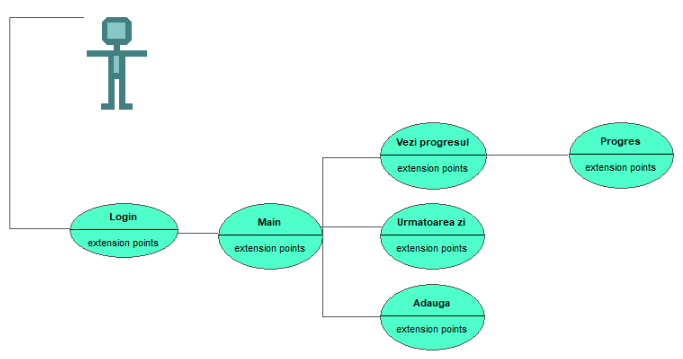
La apăsare, închide aplicația.

# Cerințe de performanță

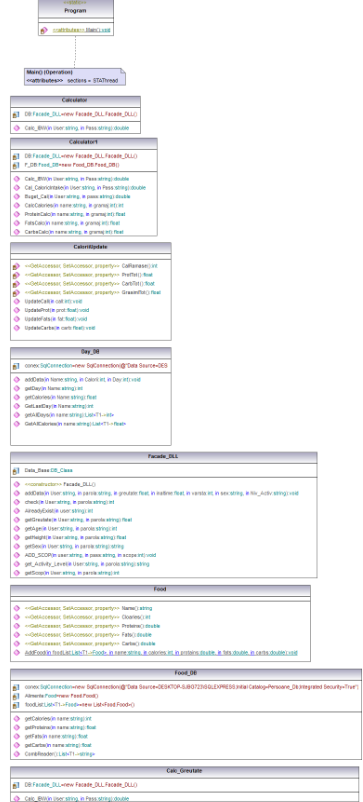
Performanța aplicației este la un nivel ridicat, timpul fiind nesemnificativ pentru utilizator.

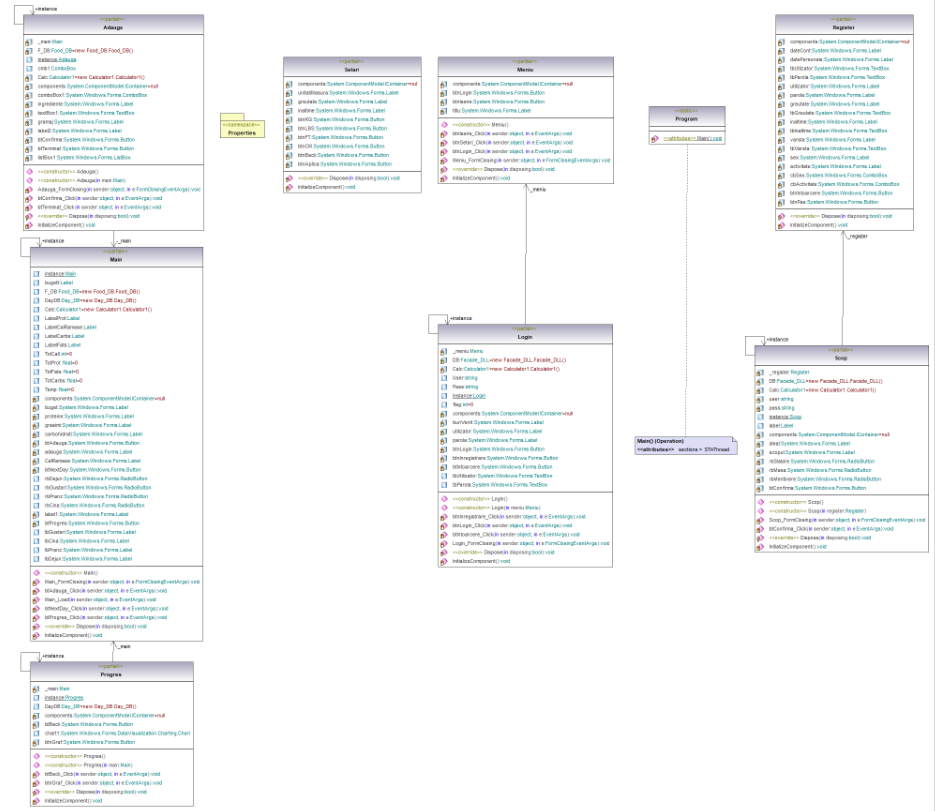
4. Diagrame UML

# Diagrama de utilizare a aplicației

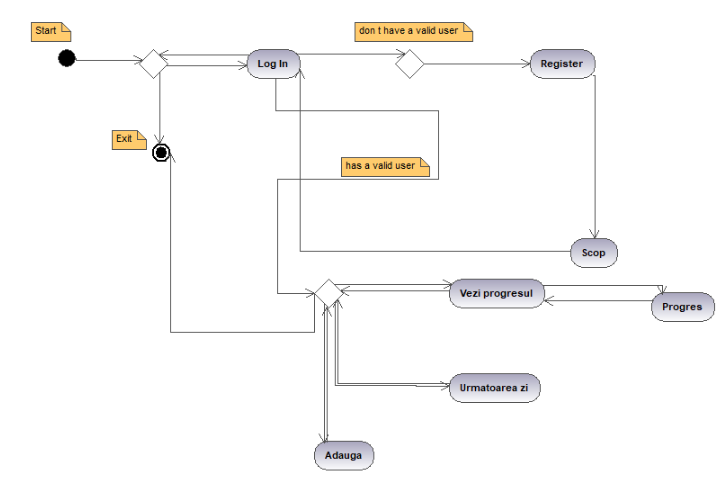


# Diagrama de clasă





# Diagrama de activitate



5. Cum să utilizați aplicația

Aplicația este concepută pentru a fi utilizată de oricine, indiferent de nivelul de cunoștințe. Fiecare utilizator poate să se autentifice în contul său sau sa creeze un cont nou. Această acțiune este foarte ușoară deoarece interfața este intuitivă.

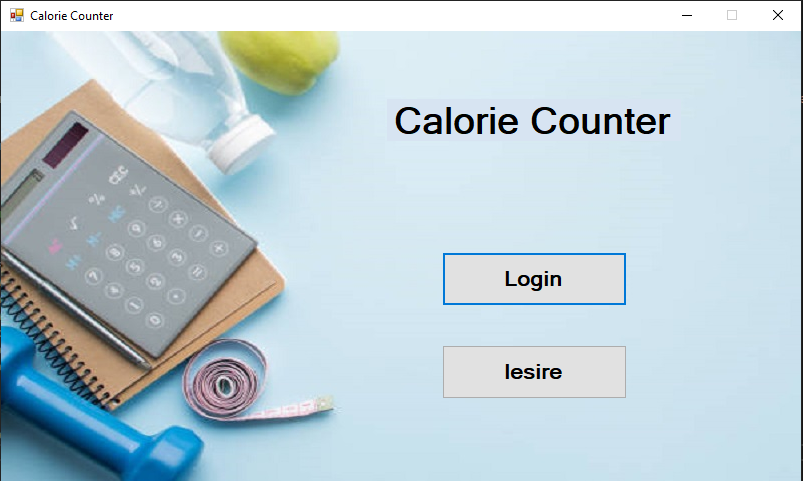
După inregistrare, unde utilizatorul ofera informatii legate de varsta, greutatea, inaltimea, sexul, nivelul de activitate si scopul pe care il are acesta este redirecționat către fereastra principală a aplicației.

Fereastra conține bugetul caloric personalizat, caloriile ramase care se modifica in functie de alimentele consumate in ziua respectiva, informatii nutritionale, calorii individuale pentru fiecare masa si un buton pentru adaugarea unei mese noi.

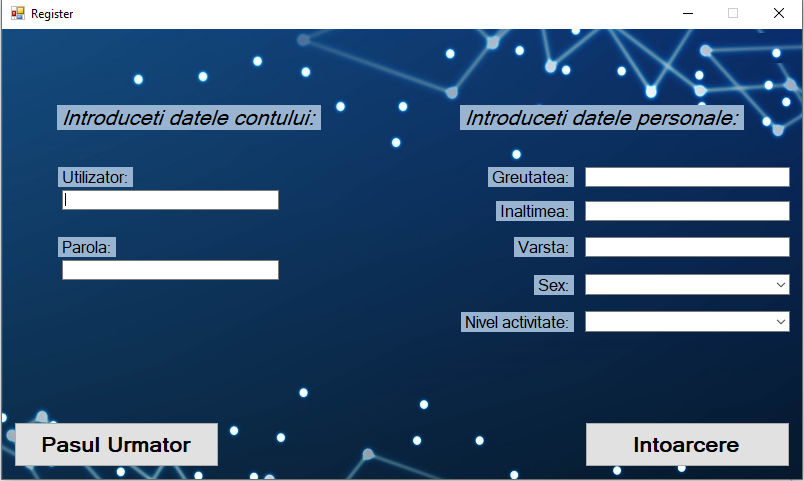
După selectarea radio button-ului corespunzator mesei ce urmeaza sa fie introdusa si apăsarea butonului, o nouă fereastră va apărea. Aceasta va conține o optiune de cautare de alimente din baza de date si selectarea unui gramaj in functie de care vor fi afisate informatii nutritionale, urmand ca prin confirmare, caloriile mesei sa fie sustrase din bugetul caloric total.

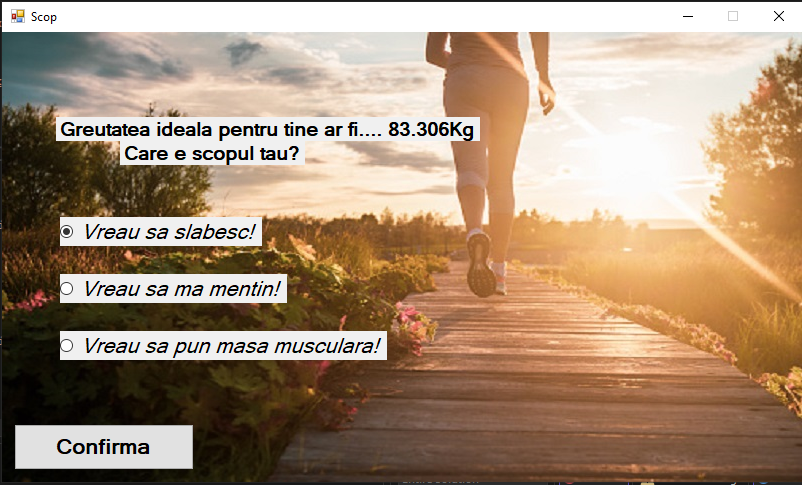
Fereastra principala are si capacitatea de a arata un grafic al caloriilor consumate in zilele anterioare cu scopul de a arata evolutia utilizatorului in drumul sau spre o viata mai sanatoasa

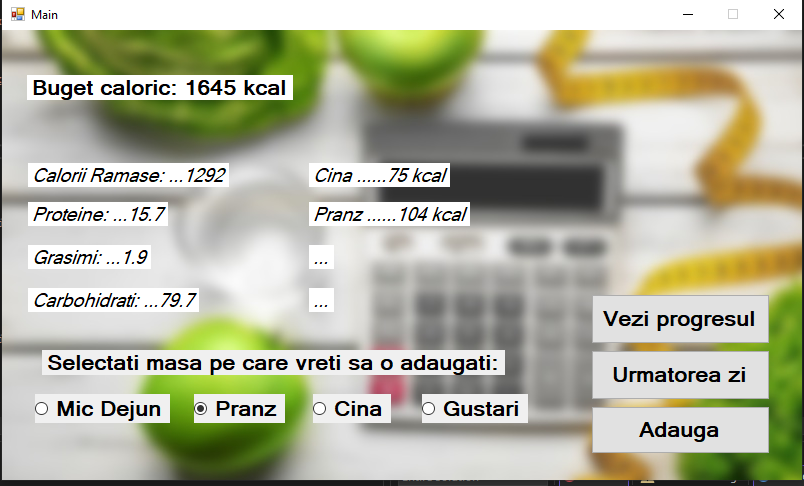
6. Exemple de execuție a aplicației



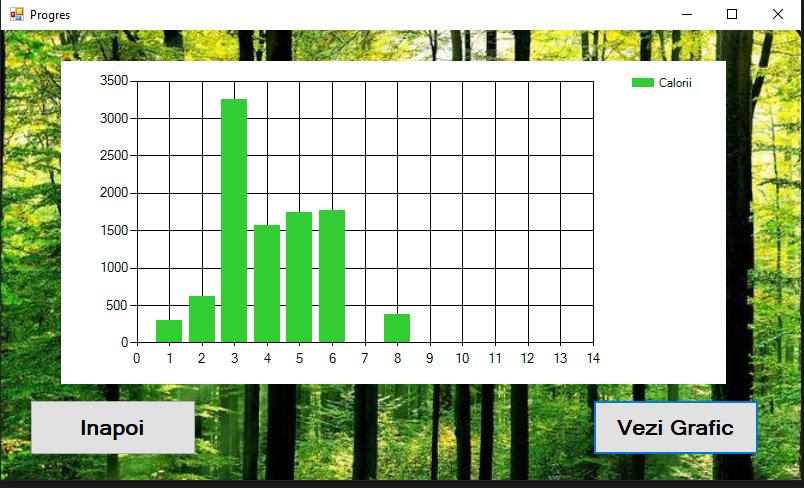












7. Exemple din codul sursă asociat

# Calculator1.cs

using System;

using System.Collections.Generic;

using System.Data.Common;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Facade\_DLL;

namespace Calculator1

{

public class Calculator1

{

private Facade\_DLL.Facade\_DLL DB = new Facade\_DLL.Facade\_DLL();

private Food\_DB.Food\_DB F\_DB = new Food\_DB.Food\_DB();

public double Calc\_IBW(string User, string Pass)

{

if (DB.getSex(User, Pass) == "Barbat")

{

return 50 + 0.91 \* (DB.getHeight(User, Pass) - 152.4);

}

else if (DB.getSex(User, Pass) == "Femeie")

{

return 45.5 + 0.91 \* (DB.getHeight(User, Pass) - 152.4);

}

else

{

return 0;

}

}

public double Cal\_CaloricIntake(string User, string Pass)

{

double BMR = 0;

if (DB.getSex(User, Pass) == "Barbat")

{

BMR = 66 + (13.75 \* DB.getGreutate(User, Pass)) + (5 \* DB.getHeight(User, Pass)) - (6.75 \* DB.getAge(User, Pass));

}

else if (DB.getSex(User, Pass) == "Femeie")

{

BMR = 655 + (9.56 \* DB.getGreutate(User, Pass)) + (1.85 \* DB.getHeight(User, Pass)) - (4.68 \* DB.getAge(User, Pass));

}

if (DB.get\_Activity\_Level(User, Pass) == " Scazut(sedentar , majoritatea timpului stat pe scaun)")

{

return BMR \* 1.2;

}

else if (DB.get\_Activity\_Level(User, Pass) == "Moderat(practici un sport 1-3 ori pe saptaman)")

{

return BMR \* 1.375;

}

else if (DB.get\_Activity\_Level(User, Pass) == "Activ(practici un sport mai mult de 3 ori pe saptaman si ai un stil de viata activ)")

{

return BMR \* 1.65;

}

return BMR;

}

public double Buget\_Cal(string User,string pass)

{

if(DB.getScop(User,pass) == 1)

{

return Cal\_CaloricIntake(User, pass) - 400;

}

else if(DB.getScop(User, pass) == 2)

{

return Cal\_CaloricIntake(User, pass);

}

else if(DB.getScop(User, pass) == 3)

{

return Cal\_CaloricIntake(User, pass) + 250;

}

return 1;

}

public int CalcCalories(string name,int gramaj)

{

return (gramaj \* F\_DB.getCalories(name))/100;

}

public float ProteinCalc(string name,int gramaj)

{

return (gramaj \* F\_DB.getProteins(name)) / 100;

}

public float FatsCalc(string name, int gramaj)

{

return (gramaj \* F\_DB.getFats(name)) / 100;

}

public float CarbsCalc(string name,int gramaj)

{

return (gramaj \* F\_DB.getCarbs(name)) / 100;

}

}

}

# Day\_DB.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Linq;

namespace Day\_DB

{

public class Day\_DB

{

private SqlConnection conex = new SqlConnection(@"Data Source=DESKTOP-SJBG723\SQLEXPRESS;Initial Catalog=Persoane\_Db;Integrated Security=True");

public void addData(string Name,int Calorii,int Day)

{

try

{

//Se face conexiunea la baza de date

conex.Open();

SqlCommand comm = new SqlCommand("INSERT INTO [DayDB] (Name,Calorii,Day) values(@Name,@Calorii,@Days)", conex);

comm.Parameters.AddWithValue("@Name",Name);

comm.Parameters.AddWithValue("@Calorii", Calorii);

comm.Parameters.AddWithValue("@Days",Day);

var tmp = comm.ExecuteNonQuery();

Console.WriteLine(tmp);

Console.WriteLine("Merge inserare in baza de date");

}

catch (Exception ex)

{

Console.WriteLine("Eroare la introducere date in BD: " + ex.Message);

}

finally

{

//Se inchide conexiunea

conex.Close();

}

}

public int getDay(string Name)

{

int day = 0;

try

{

conex.Open();

}

catch (Exception ex)

{

Console.WriteLine("Eroare la deschidere DB in Day\_DB" ,ex.Message);

}

try

{

SqlCommand comm = new SqlCommand("Select Day from DayDB where Name=@Name", conex);

comm.Parameters.AddWithValue("@Name", Name);

comm.ExecuteNonQuery();

var Day =comm.ExecuteScalar();

if(Day == null)

{

return 0;

}

else

{

day = (int)Day;

return day;

}

}catch(Exception ex)

{

Console.WriteLine("Eraore la cautare in baza de date Day\_DB" ,ex.Message);

}

finally {

conex.Close();

}

return 0;

}

public float getCalories(string Name)

{

try

{

conex.Open();

}

catch (Exception ex)

{

Console.WriteLine("Eroare la deschidere DB in Day\_DB", ex.Message);

}

try

{

SqlCommand comm = new SqlCommand("Select Day from DayDB where Name=@Name", conex);

comm.Parameters.AddWithValue("@Name", Name);

comm.ExecuteNonQuery();

var Cal = comm.ExecuteScalar();

float Cal2 = Convert.ToSingle(Cal);

return Cal2;

}

catch (Exception ex)

{

Console.WriteLine("Eraore la cautare in baza de date Day\_DB", ex.Message);

}

finally

{

conex.Close();

}

return 0;

}

public int GetLastDay(string Name)

{

int day = 0;

try

{

conex.Open();

}

catch (Exception ex)

{

Console.WriteLine("Eroare la deschidere DB in Day\_DB", ex.Message);

}

try

{

SqlCommand comm = new SqlCommand("Select Max(Day) from DayDB where Name=@Name", conex);

comm.Parameters.AddWithValue("@Name", Name);

comm.ExecuteNonQuery();

var Day = comm.ExecuteScalar();

if (Day == null)

{

return 0;

}

else

{

day = (int)Day;

return day;

}

}

catch (Exception ex)

{

Console.WriteLine("Eraore la cautare in baza de date Day\_DB", ex.Message);

}

finally

{

conex.Close();

}

return 0;

}

public List<int> getAllDays(string name)

{

List<int> dayList = new List<int>();

string query = "SELECT Day FROM [DayDB] where Name=@name"; // Replace with your actual table name

conex.Open();

SqlCommand comm = new SqlCommand(query, conex);

Console.WriteLine("Am 1");

Console.WriteLine("Am 2");

Console.WriteLine("Am 3");

comm.Parameters.AddWithValue("@name",name );

Console.WriteLine(name);

Console.WriteLine("Am 4");

SqlDataReader reader = comm.ExecuteReader();

while (reader.Read())

{

Console.WriteLine("Am 2");

int day = reader.GetInt32(0);

dayList.Add(day);

}

conex.Close();

return dayList;

}

public List<float> GetAllCalories(string name)

{

List<float> CalList = new List<float>();

string query = "SELECT Calorii FROM [DayDB] where Name=@name";

conex.Open();

SqlCommand comm = new SqlCommand(query, conex);

comm.Parameters.AddWithValue("@name", name);

SqlDataReader reader = comm.ExecuteReader();

while (reader.Read())

{

var Cal = reader.GetValue(0);

float call = Convert.ToSingle(Cal);

CalList.Add(call);

}

conex.Close();

return CalList;

}

}

}

# CaloriiUpdate.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CaloriiPersoane

{

public class CaloriiUpdate

{

private int CalRamase { get; set; }

private float ProtTot { get; set; }

private float CarbTot { get; set; }

private float GrasimiTot { get; set; }

public void UpdateCall(int call)

{

CalRamase -= call;

}

public void UpdateProt(float prot)

{

ProtTot += prot;

}

public void UpdateFats(float fat)

{

GrasimiTot += fat;

}

public void UpdateCarbs(float carb)

{

CarbTot += carb;

}

}

}