 **Universitatea Tehnică ”Gheorghe Asachi” din Iași Facultatea de Automatică și Calculatoare**

**Domeniul Calculatoare și Tehnologia Informației Specializarea Calculatoare Tehnologia Informației**

**BAZE DE DATE**

**Numele și prenumele, grupa Mesina Maria 1307A**

**Profesor indrumator Cătălin Mironeanu**

**1. Introducere**

Proiectul realizat implementeaza functionalitata unui calculator caloric. Acesta ofera facilitati precum stocarea inregistrarilor care contin alimentele consumate, cu posibilitatea ulterioara de modificare / stergere, afisarea inregistrarilor dupa zi, calculul numarului necesar de calorii si calculul sumei de calorii efectiv consumate, adaugarea produselor noi.

La rularea aplicatiei, utilizatorul este intampinat de pagina de logare, unde se poate conecta cu un account deja existent sau poate crea unul nou. In cazul logarii reusite, acesta este redirectionat pe o alta pagina, unde poate vizualiza, edita si sterge datele inregistrare. In coltul stanga-sus este dispus un widget calendar, pentru a permite accesul la inregistrarile din alta zi decat cea curenta. Se permite accesul la date calendaristice in intervalul [*data crearii accountului, data curenta*]. In dreapta acestui widget se afla 2 casete: una contine numarul de calorii recomandat pe zi, iar cealalta - numarul actual consumat, conform inregistrarilor din ziua respectiva. Valoarea recomandata de calorii este calculata cu formula Mifflin St Jeor:

- Men BMR = (10 x weight in kg) + (6.25 x height in cm) - (5 x age in years) + 5

- Women BMR = (10 x weight in kg) + (6.25 x height in cm) - (5 x age in years) – 161,

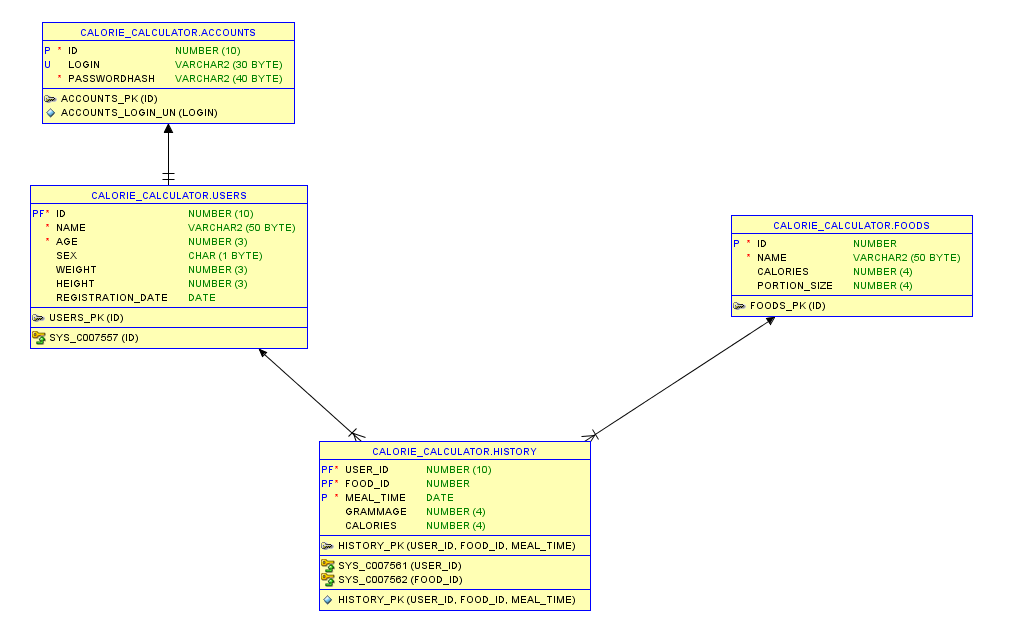
**2.Tehnologii utilizate**

Aplicatia este scrisa in limbajul de programare python, in spate datele sunt stocate intr-o baza de date Oracle, iar interfata grafica pusa la disponibilitatea utilizatorului este creata prin intermediului toolkit-ului Qt. De asemenea, a fost folosita aplicatia Qt Designer pentru deisignul mai usor (prin drag-and-drop) a ferestrelor. Aceasta aplicatie genereza fisiere .ui, care sunt incarcate in program prin intermediul unui loader specializat, pus la dispozitie de biblioteca PySide2.

Clasa principala a programului, GUI, este derivata din QStackedWidget, care permite crearea unui stive de widgeturi, denumite si pagini, dintre care, la un moment dat de timp, este vizibila una singura. Celelalte pagini sunt instante ale claselor QWidget si QDialog. In acest mod, a fost posibila includerea in proiect a 3 pagini si 2 dialoguri cu utilizatorul. In plus, a fost definita o implementare a clasei abstracte QabstractTableModel care impreuna cu un QTableView permite manipularea inregistrarilor utilizatorului curent.

Implementarea functionalitatii butoanelor/line edit-urilor se realizeaza cu ajutorul API-ului pus la dispozitie de Qt. Astfel, la fiecare actiune a utilizatorului, buton/edit-ul respectiv emite un semnal, precum clicked sau finishedEditing. Acestora li se poate asigna o functie de callback, numita si Slot. Acest model se dovedeste a fi unul de succes, deoarece se poate adapta necesitatilor aplicatiilor.

**3. Structura si inter-relationarea tabelelor**



* Tabelul ACCOUNTS contine datele necesare pentru logarea utilizatorului: loginul ales si parola criptata.
* Tabelul USERS contine informatii suplimentare despre utilizator necesare pentru calculul valorii recomandate de calorii, precum si data inregistrarii pentru a limita introducerea inregistrarilor anterioare acestei dati.
* Tabelul FOODS contine numele produsului, valoarea energetica exprimata in kCal si greutatea medie a unei portii/cantitati. Aceasta din urma este folosita pentru a facilita introduce inregistrarilor din interfata grafica
* Tabelul HISTORY stocheaza inregistrarile tuturor userilor. O inregistrare este formata din id-ul userului, id-ul alimentului consumat, ora mesei, cantitatea consumata (in grame) si numarul de calorii (calculat automat in interfata, in baza id-ului produsului si informatiile sale din tabelul FOODS)
* Intre tabelele ACCOUNTS – USERS exita o legutura ONE-TO-ONE
* Intre tabelele USERS – HISTORY exista o legutra MANY-TO-MANY
* Intre tabelele FOODS – HISTORY exista o legatura MANY-TO-MANY
* Se stabileste o legatura indirecta USERS – FOODS de tip MANY-TO-MANY. Un utilizator poate consuma diverse alimente si un aliment poate fi consumat de mai multi utilizatori distincti.

**4. Constrangeri utilizate**

Pentru toate campurile din toate tabelele s-a impus constrangerea NOT NULL.

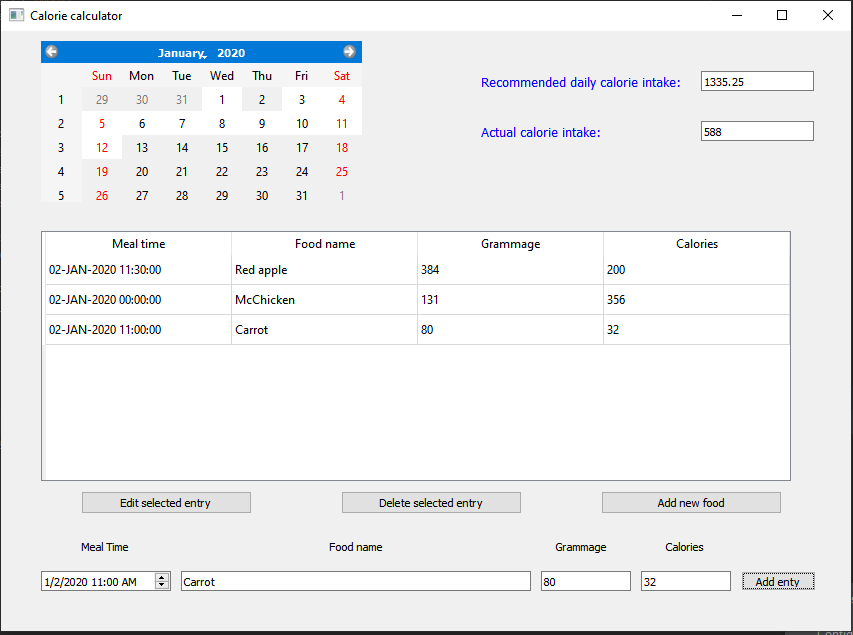
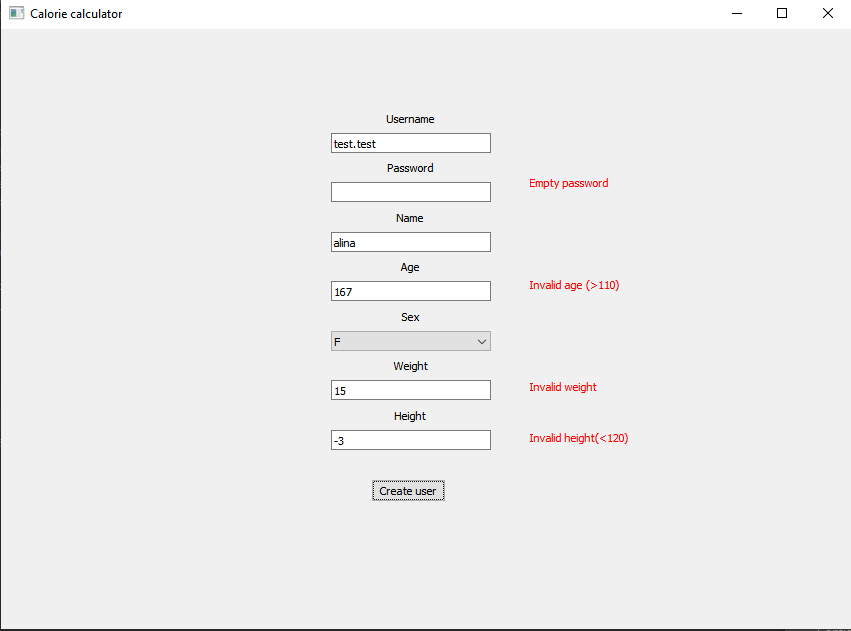
* Tabelul ACCOUNTS:
  + PRIMARY KEY– campul ID
  + UNIQUE – campul login (poate exista un singur utilizator cu un anumit login)
* Tabelul USERS:
  + PRIMARY KEY – campul ID
  + FOREIGN KEY – campul ID (informatiile suplimentare trebuie sa apartina unui utiliator existent)
  + CHECK – campurile age(valori intre 10-110), sex(valori F/M), weight(valori intre 40-200), height(valori 120-220).
* Tabelul FOODS:
  + PRIMARY KEY – campul ID
  + UNIQUE – campul name (un produs poate aparea o singura data in tabela, cu un anumit numar de calorii atasat)
* Tabelul HISTORY:
  + PRIMARY KEY – tupla (user\_id, food\_id, meal\_time) (in loc de a crea o noua inregistrare pentru a modifica cantitatea consumata de un anumit produs, utilizatorul poate edita acea inregistrare)
  + FOREIGN KEY – campurile user\_id, food\_id (atat utilizatorul, cat si produsul trebuie sa existe pentru a putea fi folosite intr-o inregistrare)

Pentru generarea id-urilor s-au folosit 2 secvente, user\_id si food\_id.

**5. Modalitatea de conectare**

Conexiunea la baza de date se realizeaza prin intermediul bibilotecii cx\_Oracle. De asemenea, a fost construita o clasa singleton DBManager care faciliteaza executia si obtinerea rezultatelor generate de instructiunile SQL.

**6. Screenshot-uri si exemple de cod**

****

