

APLICATIE PENTRU UN STIL DE VIATA SANATOS

**UNIVERSITATEA TEHNICA “GHEORGHE ASACHI” IASI
FACULTATEA DE AUTOMATICA SI CALCULATOARE
SPECIALIZAREA CTI
DISCIPLINA BAZE DE DATE**

BARBUTA DELIA ELENA

GRUPA 1307A

CUPRINS

- 1.Tema proiectului
- 2.Descrierea problemei
- 3.Proiectarea bazei de date:
 - 3.1. Descrierea tabelelor
 - 3.2. Modelul logic
 - 3.3. Modelul relational
 - 3.4. Descrierea constrangerilor
4. Limbaje si tehnologii folosite la implementare
5. Detalierea aplicatiei
6. Tranzactii

1. Tema

Tema proiectului este proiectarea unei aplicatii care permite vizualizarea sporturilor practicate, a alimentelor consumate de utilizator si a unei retete personalizate generata periodic, cu scopul de a ajuta la imbunatatirea stilului de viata.

2. Descrierea problemei

Majoritatea persoanelor nu acorda atentia necesara stilului de viata, acesta fiind motivul pentru care am dorit sa creez o aplicatie usor de utilizat, care sa ajute utilizatorii in deprinderea unor obiceiuri sanatoase, precum selectionarea alimentelor consumate si practicarea sportului in mod regulat. Principalele informatii stocate de aplicatie sunt:

- Informatii despre contul de utilizator: folosite pentru a te putea autentifica si pentru a seta in aplicatie obiectivul dorit. Fiecare persoana isi seteaza o singura data greutatea dorita si timpul in care doreste sa ajunga la aceasta. Timpul in care se doreste ajungerea la greutatea respectiva trebuie exprimat in zile.
- Sporturi: lista prestabilita de sporturi, pe care clientii o pot utiliza. Fiecare dintre acestea vine cu denumire, intensitate(marcata de la 1 la 5) si kcal consumate intr-un minut.
- Ingrediente: o lista preselectata de ingrediente, de unde utilizatorul poate alege alimentele consumate in ziua curenta. Aplicatia retine informatii cu privire la numarul de kcal pe 100g si un indice pentru a ajuta utilizatorul sa inteleaga daca alimentul consumat este sanatos sau nu.
- Retete: o lista de retete ce sunt recomandate utilizatorilor in functie de obiectivele lor. Fiecare vine cu durata(minute), dificultate(marcata de la 1 la 5), kcal si data la care va fi sugerata utilizatorului. Deoarece ne dorim ca retetele sa fie usor abordabile, durata lor nu va depasi 180 de minute.
- Inregistrarea zilnica a activitatii: in fiecare zi, utilizatorul isi poate inregistra sporturile efectuate si durata acestora, stabilind astfel numarul de calorii arse.
- Inregistrarea zilnica a alimentelor consumate: in fiecare zi, utilizatorul isi poate inregistra alimentele consumate introducand gramajul acestora. Pentru simplitate, acesta va fi va fi multiplu de 100.

3.1. Descrierea tabelor

Tabelele necesare aplicatiei sunt:

1. **accounts:** este tabela in care vom retine informatii referitoare la conturile existente in aplicatie

accounts:

id - primary key

last_name - varchar(15)
first_name - varchar(15)
email - varchar(40)
birth_date - date
weight - numeric(3)

2. **user_settings:** tabela in care fiecare utilizator isi va seta greutatea dorita si timpul in care doreste sa ajunga la aceasta

user_settings:
id - foreign key
desired_weight - numeric(3)
desired_time - numeric(3)

3. **ingredients:** tabela in care vor fi stocate alimente, din care utilizatorul le va putea alege pe cele pe care le consuma, dar si din care vor fi preluate unele pentru retetele personalizate din aplicatie.

ingredients:
id - primary key
name - varchar(15)
kcal - numeric(3)
healthy - numeric(1)

4. **sports:** tabela ce contine sporturi si informatii despre acestea.

sports:
id - primary key
name - varchar(15)
kcal - numeric(3)
intensity - numeric(1)

5. **recipes:** tabela in care se vor stoca diferite retete. Aceasta are un camp "recipe_date", in care va fi setata data la care reteta va aparea in contul utilizatorului caruia ii este destinata.

recipes:
id - primary key
name - varchar(15)
time - numeric(3)
difficulty - numeric(1)
kcal - numeric(4)
account_id - foreign key
recipe_date - date

6. **account_ingredients:** tabela ce apare in urma relatiei de many-to-many dintre tabelele accounts si ingredients. Aici vom stoca informatii referitoare la alimentele consumate de utilizator intr-o zi.

account_ingredients:

id - primary key

accounts_id - foreign key

ingredients_id - foreign key

weight - numeric(3)

date - date

7. **account_sports:** tabela ce apare in urma relatiei de many-to-many dintre tabelele accounts si sports. In aceasta vom retine informatii cu privire la sporturile practicate de utilizator intr-o zi.

account_sports:

id - primary key

accounts_id - foreign key

sports_id - foreign key

duration- numeric(3)

date - date

8. **ingredient_recipes:** tabela ce apare in urma relatiei de many-to-many dintre tabelele ingredients si recipes. O vom popula cu informatii referitoare la retete si ingredientele necesare acestora.

ingredient_recipes:

id - primary key

recipes_id - foreign key

ingredients_id - foreign key

weight - numeric(3)

kcal - numeric(4)

3.2. Modelul logic

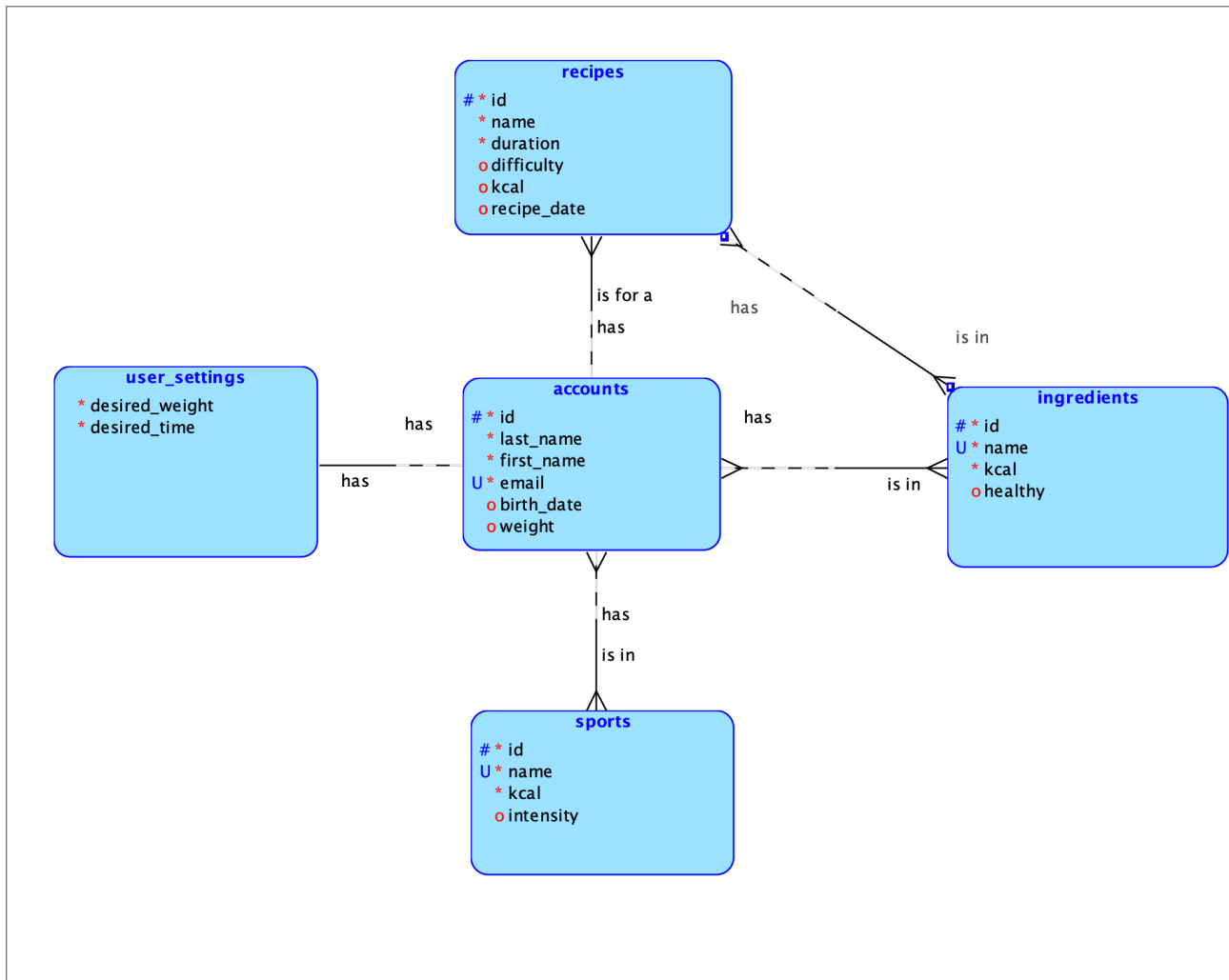
In cadrul proiectului exista toate cele trei tipuri de relatii: one-to-one, one-to-many si many-to-many.

Intre tabelele accounts si user_settings, relatia este one-to-one, deoarece un cont poate avea un singur set de configuratii, iar un set de configuratii apartine unui singur cont.

Intre tabelele accounts si recipes, relatia este de one-to-many, intrucat un cont poate primi mai multe retete personalizate, dar o reteta apartine unui singur cont.

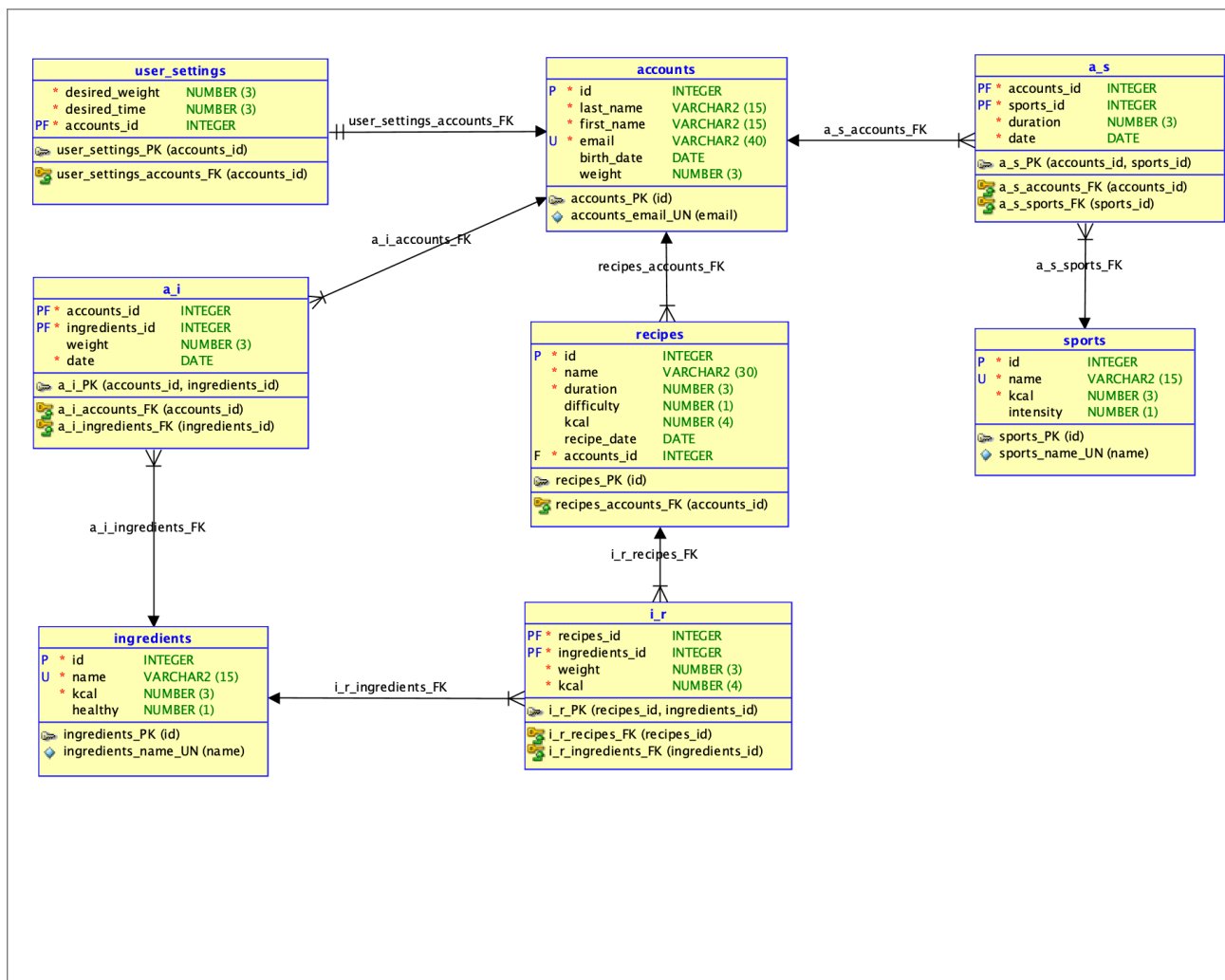
Intre tabelele accounts si sports si accounts si ingredients, relatiile sunt de many-to many, deoarece un cont poate practica mai multe sporturi si

consuma mai multe ingrediente, in timp ce un sport poate fi practicat de mai multe persoane si un ingredient poate fi consumat de mai multi utilizatori. Intre tabelele ingredients si recipes, relatia este many-to-many. Un ingredient poate fi folosit in mai multe retete, iar o reteta poate contine mai multe ingrediente.



3.3 Modelul relational

Relatia de many-to-many dintre tabelele accounts si ingredients din modelul logic a fost sparta in doua relatii de one-to-many. Rezulta astfel o tabela de legatura, ce va retine ambele chei primare, accounts_id, din tabela accounts si ingredients_id, din tabela ingredients. Un mecanism similar va fi aplicat si in cazul relatiilor dintre tabelele accounts si sports si ingredients si recipes. In modelul relational Primary key-urile vor fi generate de baza de date printr-un mecanism de tip autoincrement. De asemenea, in tabela recipes, pentru atributul recipe_date a fost creat un trigger ce nu permite inserarea unei date calendaristice mai mica decat data curenta.



3.4 Descrierea constrangerilor

Pentru a evita introducerea datelor eronate de catre utilizatori, in cadrul bazei de date s-au stabilit diferite constrangeri:

1. Constrangeri de tip Check

Aceste tipuri de constrangeri au fost folosite pentru a nu permite introducerea unor denumiri prea scurte(in cazul atributelor last_name, first_name, din tabela accounts si atributelor name din tabellele recipes, sports si ingredients) sau a unor valori negative(pentru campurile weight , din tabellele a_i si i_r, duration, din tabela a_s, kcal din i_r, ingredients si recipes si desired_weight si desired_time, din user_settings).

Prin intermediul conditiilor de tip Check am restrictionat introducerea valorilor in tabela a_i, campul weight, doar la valori ce sunt multiplu de 100 si am conditionat valorile din campul duration, tabela recipes sa fie mai mici decat 180. De asemenea, au fost facute verificari pentru campuri precum healthy(ingredients) sau difficulty(recipes), astfel incat acestea sa contina doar valori dintr-o lista preselectata.

Pentru verificarea datei de nastere si a emailului s-au folosit de asemenea constrangeri de tip Check.

2. Constrangeri de tip NOT NULL

Aceste constrangeri au fost folosite pentru a marca esentialele campuri, ce nu pot lipsi de la inserare, pentru o entitate. Aceasta constrangere s-a aplicat tuturor id-urilor si majoritatii atributelor din tabele. De exemplu: `date(a_i)`, `duration(a_s)`, `last_name(accounts)`, `weight(i_r)`, `kcal(ingredients, sports)`, `name(recipe)`, `desired_weight(user_settings)`.

3. Constrangeri de tip Unique

Atributele de tip Primary Key respecta implicit constrangerea de tip Unique. In afara de acestea, atributul email, din tabela accounts si name, din tabelele ingredients si sports mai respecta aceasta constrangere.

4. Constrangeri de tip Primary Key

Fiecare tabel in parte prezinta un camp de tip index cu auto-incrementare care respecta o constrangere de tip Primary Key, in afara de tabelele `a_i` si `a_s`.

5. Constrangeri de tip Foreign Key

Aceste constrangeri sunt utilizate pentru a modela diferite tipuri de relatii .

- Many-to-many: in tabela `a_i`, `accounts_id` si `ingredients_id`, in tabela `a_s`, `accounts_id` si `sports_id`, in tabela `i_r`, `ingredients_id` si `recipes_id`,
- One-to-many: in tabela `recipes`, `accounts_id`
- One-to-one: in tabela `user_settings`, `accounts_id`

4. Limbaje si tehnologii folosite la implementare

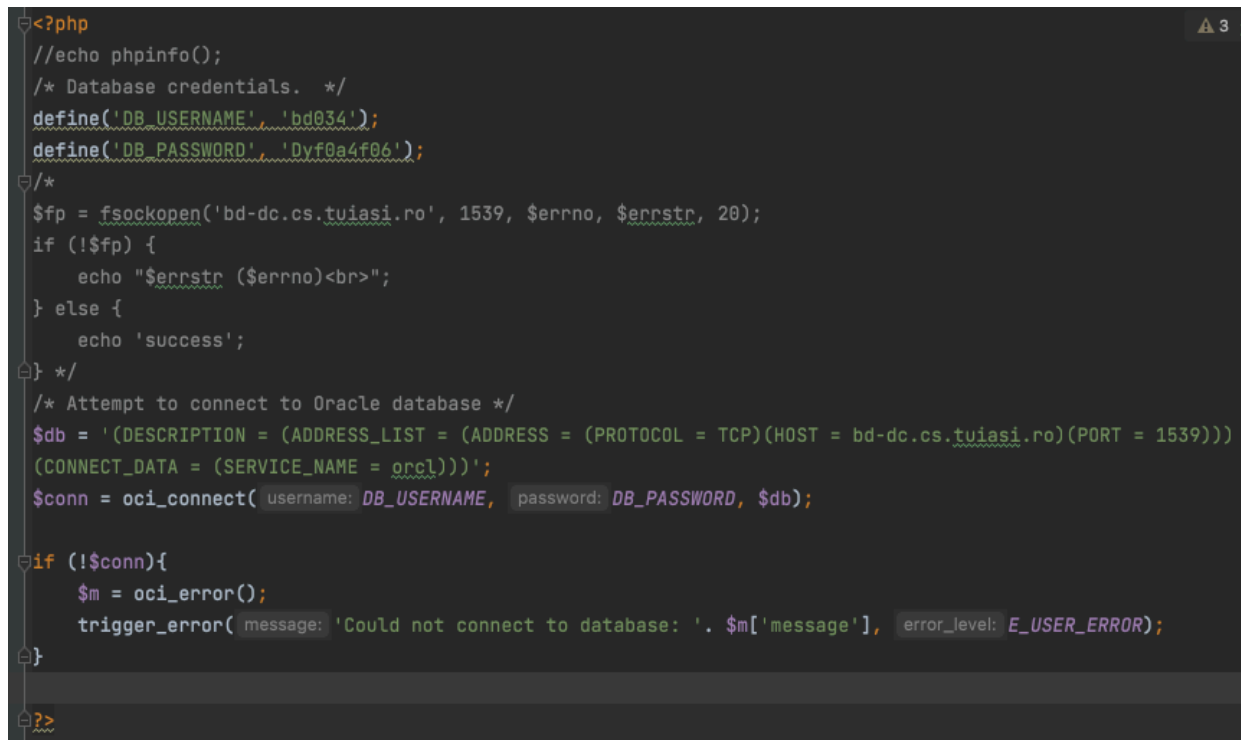
Baza de date creata a fost folosita in cadrul unei aplicatii de tip Web. Back-end-ul aplicatiei a fost realizat in limbajul PHP v8.1.1, prin intermediul programului XAMPP. A fost necesara instalarea unei extensii, `oci8`, prin intermediul careia este posibila comunicarea cu o baza de date de tip Oracle.

In realizarea partii de Front-End, s-au utilizat limbajele HTML si CSS. Pentru usurinta manevrarii elementelor din interfata, am utilizat Bootstrap (<https://getbootstrap.com>), un framework ce utilizeaza limbajul CSS. Din acest framework am preluat template-uri pentru butoane, tabele, formulare, meniuri de navigare, etc. Interfata paginilor de login si register este inspirata

din interfata de la urmatorul link: <https://www.tutorialrepublic.com/php-tutorial/php-mysql-login-system.php>.

5. Detalierea aplicatiei

Conexiunea cu baza de date este realizata prin intermediul functiilor puse la dispozitie de catre extensia oci8, mai precis, cu ajutorul functiei `oci_connect()`. Aceasta primeste ca parametrii username-ul, parola si o variabila db definita cum se poate vedea in screenshot-ul de mai jos.



```
<?php
//echo phpinfo();
/* Database credentials. */
define('DB_USERNAME', 'bd034');
define('DB_PASSWORD', 'Dyf0a4f06');
/*
$fp = fsockopen('bd-dc.cs.tuiasi.ro', 1539, $errno, $errstr, 20);
if (!$fp) {
    echo "$errstr ($errno)<br>";
} else {
    echo 'success';
}
*/
/* Attempt to connect to Oracle database */
$db = '(DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP)(HOST = bd-dc.cs.tuiasi.ro)(PORT = 1539)))
(CONNECT_DATA = (SERVICE_NAME = orcl)))';
$conn = oci_connect( username: DB_USERNAME, password: DB_PASSWORD, $db);

if (!$conn){
    $m = oci_error();
    trigger_error( message: 'Could not connect to database: '. $m['message'], error_level: E_USER_ERROR);
}

?>
```

În cadrul proiectului există 2 template-uri, unul pentru footer și unul pentru header. În header se initializează sesiunea, prin intermediul funcției `session_start()` și avem câteva elemente de Front-end, printre care și butonul de logout, ce conține un link spre un fișier denumit `logout.php`. În fișierul respectiv este închisă sesiunea și se realizează redirect-ul spre pagina de login.

Înainte de intrarea propriu-zisă în aplicație, utilizatorul are două opțiuni: fie să își creeze cont nou, fie să se înregistreze, utilizând adresa de email, în contul preexistent. Cele două pagini se numesc `register` și `login`.

Sign Up

localhost/bd-proiect/register.php

Aplicații Toshiba Google YouTube Gmail

Sign Up

Please fill this form to create an account.

Email*

Nume*

Prenume*

Data nasterii

Greutate

Greutate dorita*

Timp acordat indeplinirii obiectivului*

Submit

Reset

Already have an account? [Login here.](#)

Login

localhost/bd-proiect/login.php

Aplicații Toshiba Google YouTube Gmail

Login

Please fill in your credentials to login.

Email

Login

Don't have an account? [Sign up now.](#)

Odata finalizat login-ul, utilizatorul este redirectionat spre localhost/bd-proiect/pages.user-dashboard.php, unde este pagina principala a aplicatiei.

BD App | Delia Barbuta

localhost/bd-proiect/pages/user-dashboard.php

Lista de lectură

BD APP

[Log out](#)

Calorii consumate

09-01-22

Ati consumat 1114 calorii.

Calorii ramase de consumat

09-01-22

Mai aveti 1086 de calorii de consumat.

Calorii arse

09-01-22

Ati ars 200 de calorii.

Alimente consumate

Selectati ingredientul ▼

Big Mac	200g	1006kcal	<input type="button" value="Modifica"/>	<input type="button" value="Sterge"/>
Oua	100g	108kcal	<input type="button" value="Modifica"/>	<input type="button" value="Sterge"/>

Sporturi practicate

Selectati sportul ▼

Alergat	20min	200kcal	<input type="button" value="Modifica"/>	<input type="button" value="Sterge"/>
---------	-------	---------	---	---------------------------------------

Reteta personalizata - Tocanita

Durata: 100 min
Dificultate: 3
Calorii: 355

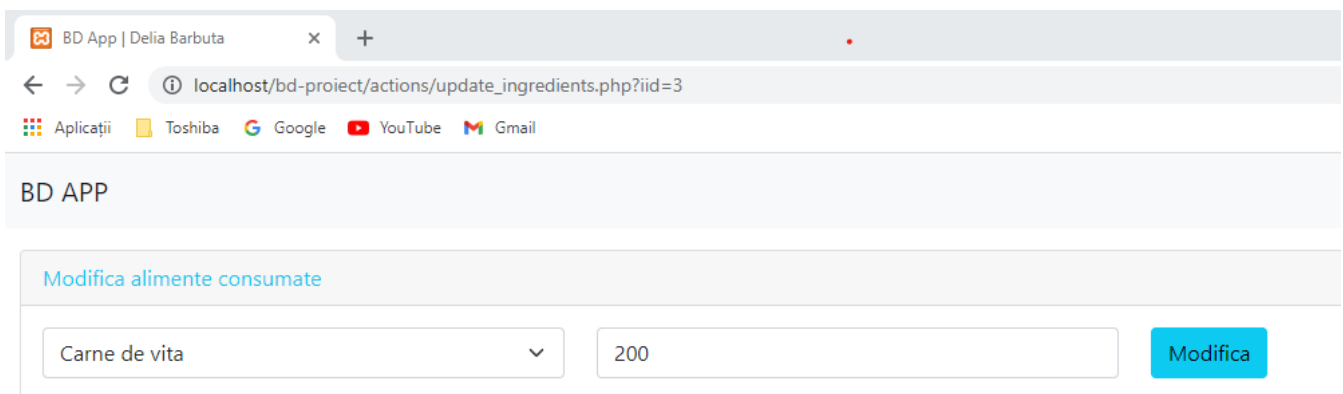
Cartof	100	87
Rosie	100	18
Carne de vita	100	250

In cadrul paginii principale sunt realizate:

- **Operatii de tip SELECT:** kaloriile consumate, kaloriile arse, lista de ingrediente si sporturi. Daca este ziua in care contului ii este asignata o reteta personalizata, cum e cazul contului din imaginea de mai sus, atunci si numele retetei, durata, dificultatea si numarul de calorii al acesteia, precum si ingredientele, cu gramajul si kaloriile lor sunt toate preluate din baza de date, prin intermediul mai multor SELECT-uri. Pentru realizarea acestora am creat o functie runQuery, in fisierul functions, ce accepta ca parametrii query-ul ce va fi executat si conexiunea si returneaza un obiect de tip statement. Functia runQuery foloseste doua functii ale extensiei oci, oci_parse() si oci_execute(). Pentru a obtine fiecare linie din baza de date, am utilizat functia oci_fetch_assoc(), ce primeste ca parametru statement-ul returnat de runQuery() si care selecteaza cate o linie din baza de date, o pune intr-o variabila si apoi o sterge din statement.
- **Operatii de tip INSERT:** operatiile de inserare se realizeaza la apasarea butonului “Adaugați” din dashboard, fie pentru alimente, fie pentru ingrediente. La apasarea butonului, se trimite datele din formular, iar user-ul este redirectionat catre pagina “/bd-proiect/actions/insert_ingredients.php” sau “/bd-proiect/actions/insert_sports.php” (in functie de butonul “Adaugați” apasat). Aceasta pagina nu contine html, ci doar realizeaza inserarea in baza de date, utilizand functia runQueryInsert. Ea este definita tot in fisierul functions si accepta aceeasi parametrii ca

runQuery, dar returneaza o variabila de tip bool, care indica daca operatia a avut succes sau nu. Desi functia se numeste runQueryInsert, ea este folosita si in cadrul operatiilor de Update si Delete.

- **Operatii de tip UPDATE:** operatiile de update se realizeaza prin apasarea butoanelor “Modificati”, fie din tabelul alimente, fie din tabelul sporturi. La apasarea unui astfel de buton, utilizatorul este redirectionat catre o pagina in care exista doar header-ul si campurile ce pot fi modificate. Id-ul produsului pe care utilizatorul doreste sa il modifice este trimis ca parametru si preluat cu ajutorul variabilei superglobale \$_GET. Dupa selectia campurilor ce vor fi modificate si apsarea butonului “Modificati”, se realizeaza redirect inapoi catre pagina user-dashboard.



BD APP

Modifica alimente consumate

Carne de vita 200 Modifica

- **Operatii de tip DELETE:** se realizeaza prin intermediul butoanelor “Stergeti”. Id-ul produsului pe care utilizatorul doreste sa il stearga este trimis ca parametru si preluat cu ajutorul variabilei superglobale \$_GET. Pagina nu contine nici in acest caz html, ceea ce creeaza pentru utilizator senzatia unui refresh al paginii, atunci cand se apasa butonul “Stergeti”, cand de fapt acesta este redirectionat catre pagina “/bd-proiect/actions/delete_ingredients.php”, iar la finalul operatiei de stergere, redirectionat inapoi catre user-dashboard.

Dupa operatiile de Update, Delete si Insert se apeleaza `oci_commit()`, cu parametru conexiunea catre baza de date, functie ce da commit modificarilor realizate.

6. TRANZACTII

In cadrul aplicatiei a fost implementata o tranzactie, in fisierul `register.php`. In momentul in care se creeaza un nou cont, utilizatorul introduce date, ce trebuie stocate in tabela `accounts`, dar si date ce trebuie stocate in tabela `user_settings`. Se va realiza primul insert in tabela `accounts`, in cadrul caruia va trebui sa preluam si id-ul randului inserat, deoarece avem nevoie de acesta in urmatorul insert. Dupa aceea, vom realiza al doilea insert in tabela `user_settings`. Daca vreo una dintre operatii nu s-a realizat corect, apelam

functia `oci_rollback()`. Doar daca ambele inserari s-au realizat corect, apelam functia `oci_commit()`, ce va da commit modificarilor in baza de date. Astfel putem asigura atomicitatea datelor.

```
else{
    //cream o tranzactie: vom face 2 inserturi, 1 in accounts si 1 in user_settings,
    //dar pentru a pastra atomicitatea datelor, vom da commit doar dupa ce se realizeaza amandoua.
    //Astfel ne putem asigura ca fie se realizeaza ambele inserturi, fie nu se realizeaza niciunul.

    $birth_date = strtoupper(date( format: "d-M-y", strtotime($birth_date)));
    $query = "INSERT INTO accounts(last_name, first_name, email, birth_date, weight)
values ('$last_name', '$first_name', '$email', '$birth_date', '$weight') RETURNING id INTO :p_val";
    //trebuie sa returnez id-ul ca sa il introduc in user settings

    $s = oci_parse($conn, $query);
    oci_bind_by_name($s, bv_name: ":p_val", &variable: $val);
    $result = oci_execute($s, mode: OCI_NO_AUTO_COMMIT);

    $query2 = "INSERT INTO user_settings(desired_weight, desired_time, accounts_id)
values ('$desired_weight', '$desired_time', '$val')";

    $s2 = oci_parse($conn, $query2);
    $result2 = oci_execute($s2, mode: OCI_NO_AUTO_COMMIT);
    if(!$result || !$result2)
        oci_rollback($conn);
    else
        oci_commit($conn);
}
```