UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Aplicație mobilă: FitnessGoal

propusă de

Dan Guzovatîi

Sesiunea: iunie, 2023

Coordonator științific

Lector dr. Vidrașcu Cristian

UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

Aplicație mobilă: FitnessGoal

Dan Guzovatîi

Sesiunea: iunie, 2023

Coordonator științific

Lector dr. Vidrașcu Cristian

Cuprins

Motivație

Odată cu dezvoltarea tehnologiei, oamenii au început să sufere de probleme de sănătate tot mai mari, mai ales din cauza stilului lor de viață sedentar. Prin urmare, utilizatorii de tehnologie încep să realizeze importanța unui stil de viață sănătos. Am decis că ar trebui să ne concentrăm pe exerciții fizice, pe o dietă echilibrată, pe a rămâne hidratați și, de asemenea, pe utilizarea tehnologiei pentru a ajuta oamenii să-și mențină sănătatea și bunăstarea generală.

Introducere

Lucrarea de licență prezintă o aplicație mobilă de fitness. Aplicația se distinge de celelalte aplicații care fac referire la acest domeniu, prin oferirea fiecărui utilizator în parte a unei experiențe de utilizare personalizate și adaptive. Aceasta generează un plan de exerciții fizice și mese, astfel încât utilizatorul să își poată îndeplini scopul ales într-o anumită perioadă de timp.

Aplicația elimină necesitatea achitării serviciilor unui antrenor personal, în acest mod, fiind o soluție accesibilă pentru utilizatori din punct de vedere financiar.

Majoritatea aplicațiilor existente pe piață de acest gen se confruntă cu problema de lipsă de adaptabilitate și personalizare. De cele mai multe ori, ele oferă planuri predefinite, generale, de exerciții și dietă, care nu țin cont de caracteristicile individuale ale utilizatorilor și scopurilor lor.

Astfel utilizatorii sunt mai predispuși să renunțe la planul de fitness într-o fază inițială, din cauza frustrării.

Una din contribuțiile aduse de aplicația dezvoltată de mine, în comparație cu aplicațiile deja existente pe piață, este oferirea unei experiențe personalizate și a unei soluții adaptabile al planului, bazat pe datele introduse de utilizator, cum ar fi vârsta, greutatea, înălțimea, genul și scopul (slăbire, dezvoltarea masei musculare sau a fi activ). Pe baza lor, aplicația construiește un plan de exerciții fizice și mese personalizate, având ca scop principal să ghideze utilizatorul pe drumul său de atingere a obiectivelor propuse într-un mod eficient.

Cu ajutorul algoritmilor și bazei de date, aplicația urmărește progresul utilizatorului, oferindu-i un feedback cu privire la evoluția progresului în timp real. Astfel, utilizatorii au parte de o experiență plăcută și un suport, reprezentat de un plan adaptat nevoilor lor individuale.

O altă contribuție a aplicației este interfața ușor de perceput și de utilizat pentru consumatori. Am pus accentul pe dezvoltarea unei interfețe prietenoase cu navigare

intuitivă, pentru a ajuta utilizatorul să se concentreze asupra *feature*-urilor prestate de către aplicație.

Pe piață există aplicații ce conțin exerciții fizice, aplicații pentru dietă, aplicații de *tracking* a consumului de apă. Aplicația mea se deosebește de acestea prin faptul combinării acestor funcționalități într-o singură aplicație.

Lucrarea este împărțită în 3 capitole:

Capitolul 1 cuprinde o fundamentare teoretică în domeniul fitness, relevanța acestui domeniu în viața modernă și beneficiul fiecărei funcționalități ale aplicației oferit acestuia.

Pentru a înțelege necesitatea aplicației trebuie să cunoaștem domeniul în care aceasta are un rol important.

În **Capitolul 2** am prezentat abordarea practică a aplicației și tehnologiile folosite.

În cadrul **Capitolului 3** am descris aplicația în sine. Mai exact, arhitectura aplicației, unde am prezentat structura aplicației, diagramele C4 și diagrama fluxului utilizatorului. Tot în acest capitol am explicat cum are loc colectarea datelor utilizatorului și am descris fiecare funcționalitate în parte, cum a fost gândită și algoritmii concepuți pentru a genera planul personalizat de fitness.

Capitolul 1

Fundamentare teoretică

1.1 Importanța fitnessului

Fitnessul este un factor ce indică capacitatea organismului de a-și îndeplini sarcinile în timpul zilei. El reprezintă nivelul general de sănătate, atât fizică, cât și mentală.

Fitnessul poate fi practicat de către oricine și combină într-un mod echilibrat exercițiile fizice, o alimentație sănătoasă și o hidratare corespunzătoare.

Din cauza faptului că ducem un mod de viață lipsit de mișcare, fitnessul ne poate aduce multe beneficii multor dintre noi, dezvoltând: forța musculară, rezistența la efort, flexibilitatea întregului corp.

În acest mod, fitnessul aduce consecințe favorabile sănătății noastre, cum ar fi: îmbunătățirea circulației sângelui în corp, din cauza stimulării metabolismului de a lucra mai intensiv, întărirea sistemului imunitar (prin intermediul alimentației bogate în nutrienți sănătoși), arderea stratului de grăsime în plus, prevenirea depunerilor grăsimilor pe organe, astfel prevenind bolile de sănătate.

1.2 Beneficiile fizice și mentale ale fitnessului

Fitnessul oferă multe beneficii pentru un fizic sănătos, ceea ce este reprezentat de menținerea/obținerea unei greutăți sănătoase, întărirea rezistenței, creșterea masei musculare, îmbunătățirea funcționalității inimii și plămânilor, menținerea unui nivel optim de grăsimi, astfel prevenind obezitatea.

Un beneficiu mai puțin cunoscut al fitnessului este îmbunătățirea calității somnului. Conform unei analize comprehensive, folosind metode statistice pentru a ana-

liza diferența între calitatea somnului practicând exerciții fizice și în lipsa practicării lor, cu seturi mari de date colectate de la oameni reali, purtând aparate de analiză a calității somnului, rezultatele căreia au fost descrise în articolul *How Physical Exercise Level Affects Sleep Quality? Analyzing Big Data Collected from Wearables*, am aflat că majoritatea oamenilor care au practicat exerciții fizice și-au îmbunătățit calitatea somnului.

Pe plan mintal, fitnessul poate aduce la fel de multe beneficii, dacă nu mai multe. Starea noastră emoțională depinde foarte mult de doza noastră zilnică de mișcare și calitatea nutrienților consumați.

Activitățile, care implică o stimulare a metabolismului de a lucra mai intensiv, sunt de folos pentru reducerea nivelului de stres, îmbunătățesc capacitatea individului de a intra în starea de concentrare.

Mult mai mult, aceste activități sportive stimulează creierul să elibereze substanțe precum endorfina și serotonina. Acestea au o influență directă asupra nivelului nostru de fericire.

Multe studii au arătat că anume practicarea activităților fizice au ajutat oamenii să scape de anxietate și stres.

1.3 Rolul alimentației în domeniul fitnessului

Alimentația joacă un rol important în obținerea și menținerea unui nivel potrivit de fitness. Energia și substanțele necesare pentru efectuarea exercițiilor fizice ne sunt furnizate de către nutrienții obținuți din alimentele consumate.

Hipertrofia musculară (construirea masei musculare) bazată pe descompunerea proteică (proteoliza) și construcția proteică (sinteza proteică) este un proces care are loc constant în corp. În timpul efectuării exercițiilor fizice rata de descompunere este mai mare decât rata de sinteză, iar atunci când ne odihnim procesele de reconstrucție cresc rata de sinteză peste rata de descompunere. Deci, prin urmarea acestui aport net pozitiv cresc elementele contractile (actina și miozina) și dilatarea lichidului sarcoplasmatic, cu alte cuvinte crește masa musculară.

Pentru a menține acest proces și a profita de consecințele lui trebuie să oferim organismului nostru toată gama de nutrienți necesari: carbohidrați (furnizori de energie), proteina (construirea și repararea mușchilor), vitamine (susținerea sistemului imunitar), minerale, etc.

1.4 Importanța factorilor individuali ale individului în conceperea planului de fitness

În cadrul licenței am analizat importanța factorilor individuali ai utilizatorului (vârstă, înălțime, greutate, sex, nivelul de fitness, scop) pentru conceperea unui plan eficient de fitness, care ar duce la atingerea obiectivelor sale.

Capacitatea de a schimba planul de fitness și personalizarea lui fac utilizatorul să simtă o contribuție personală în conceperea lui, astfel motivându-l să urmeze planul de fitness pe o perioadă mai îndelungată.

Vârsta este un factor care influențează tipul de exerciții pe care o persoană le poate practica, astfel pentru persoanele în vârstă ar fi recomandate exerciții fizice de o complexitate mai ușoară, îndeosebi bazate pe dezvoltarea flexibilității, cât și un număr mai mic de repetări.

Un program de fitness bine conceput ar trebui să ia în considerare nivelul de fitness inițial al persoanei în cauză, acesta fiind un factor important. Pentru persoanele cu un nivel de fitness mai inferior le-ar fi mai greu să reziste unui număr mare de repetări al exercițiilor sau exercițiilor de o complexitate înaltă. Însă pentru persoanele cu un nivel de fitness mai ridicat, exercițiile de o complexitate scăzută și un număr mic de repetări nu le-ar permite un progres constant.

Greutatea și înălțimea unei persoane, la fel, influențează numărul de repetări al exercițiilor, studiile arătând că persoanele cu o greutate mai mare au nevoie de un număr mai mic de repetări pentru a pierde în greutate, comparativ cu o persoană cu o greutate mai mică. Studiile mai arată că persoanele cu o înălțime mai mare sunt predispuse să piardă în greutate mai ușor decât persoanele mai scunde, astfel având nevoie de un număr mai mic de repetări pentru a-și îndeplini acest scop.

Sexul este un factor la fel de important, care trebuie luat în considerare, deoarece conform acestor articole: (articol1, articol2), persoanele de gen masculin sunt mai predispuse să piardă în greutate și să construiască masă musculară decât persoanele de gen feminin, deci numărul de repetări la fel diferă de la gen la gen.

Un plan de fitness ar trebui să fie construit în jurul obiectivului pus de către persoană, el putând fi: pierdere în greutate, construirea masei musculare, a duce un mod de viață activ în general.

1.5 Importanța ajustării nutriției în funcție de scopul individual

Dacă persoana are ca scop slăbirea, atunci este esențial ca ea să se afle într-un deficit caloric (numărul caloriilor consumate $-$ numărul caloriilor arse < 0). Cantitatea de macro nutrienți, precum grăsimile și carbohidrații, trebuie micșorată. Carbohidrații consumați în cantități mari și care nu sunt arși sunt stocați sub formă de grăsime corporală.

Este importantă alegerea carbohidraților complecși în locul celor rafinați și procesați. Grăsimile oferă 9 calorii per gram consumat, fiind cele mai dense în calorii printre macronutrienți. Deci, este esențială reducerea consumului de grăsimi saturate, pe când un aport adecvat de grăsimi nesaturate este bine venit pentru menținerea sănătății.

Dacă persoana are ca scop creșterea și dezvoltarea masei musculare, atunci este binevenit să aibă o dietă cu un număr crescut de proteine, pentru a putea susține hipertrofia musculară. De asemenea, carbohidrații sunt materia primă pentru energia necesară organismului pentru a reconstrui țesuturile musculare, astfel este nevoie de un aport corespunzător și din acest aspect.

Dacă persoana are ca scop ducerea unui mod de viață sănătos și activ, atunci ar trebui să aibă o dietă ce ar echilibra cantitatea de proteine, carbohidrați și grăsimi consumată.

Indiferent de scop, o hidratare corespunzătoare este un factor crucial pentru funcționarea optimă a organismului.

1.6 Utilizarea tehnologiei și a aplicațiilor mobile pentru monitorizarea și înregistrarea progresului individual

Aplicațiile mobile sunt un instrument pentru înregistrarea progresului nutriției, activității fizice și consumului de apă, astfel oferind utilizatorilor o perspectivă mai clară a performanțelor.

Un alt avantaj al aplicațiilor mobile este posibilitatea creării unei comunități virtuale, prin intermediul căreia utilizatorii pot să își împărtășească progresul, realizările, dar și greutățile prin care trec, astfel pot să primească sprijin și motivație suplimentară. În plus, specialiștii în domeniu ar putea să le ofere sfaturi utilizatorilor aflați în impas.

Capitolul 2

Abordare practică

2.1 *Framework*-ul Flutter

2.1.1 Ce este Flutter?

Flutter este un *framework* de dezvoltare de aplicații mobile creat de către Google. A fost lansat în 2017, deci este destul de nou pe piață, însă a fost îmbrățișat de mulți *software developeri*. Permite dezvoltarea aplicațiilor mobile native pentru platformele Android și iOS. Flutter utilizează limbajul de programare Dart, acesta fiind un limbaj modern, dezvoltat de Google, care este optimizat pentru construirea interfețelor utilizator și aplicații mobile reactive.

2.1.2 De ce am folosit Flutter?

Unul dintre motivele principale pentru care am ales Flutter este dezvoltarea multiplatformă:

Flutter oferă capacitatea de a dezvolta o aplicație, scriind un cod unic, care poate rula atât pe Android, cât și pe iOS. Deci se elimină nevoia de a dezvolta și menține doua aplicații separate pentru fiecare platformă.

Un alt motiv pentru care am ales Flutter este pentru performanța sa nativă, oferită prin intermediul motorului de randare, Skia.

Un alt avantaj al Flutter este reîmprospătarea instantanee, acesta fiind un proces prin care modificările din cod sunt vizualizate în timpul dezvoltării, fără a fi nevoie de recompilare sau repornire a aplicației.

2.2 Modelul de date utilizat pentru stocarea informațiilor

Modelul bazei de date utilizat pentru stocarea informațiilor este **modelul relațional**, iar pentru baza de date am folosit MySQL. A fost folosită o tabelă pentru a stoca informațiile despre utilizator precum numele, mailul, înălțimea, greutatea, nivelul de fitness, gen, data nașterii, etc. Pentru stocarea informațiilor despre mese, exerciții, postări, s-au folosit tabele separate.

Am întâmpinat probleme la alegerea stocării planului de exerciții și de mese, deci pentru a evita redundanța și inconsistența datelor am venit cu soluția de a crea câte două tabele pentru fiecare plan, drept consecință, normalizând baza de date.

2.3 Memorarea *token*-ului utilizatorului logat

Pentru a memora *token*-ul utilizatorului logat am folosit definiția de *Shared Preferences*, care permite aplicației să memoreze preferința sub formă de cheie-valoare. În cazul sistemului de operare Android aceasta este stocată în folderul *"shared-prefs"*.

Capitolul 3

FitnessGoal App

3.1 Arhitectura aplicației

3.1.1 Diagrama de context

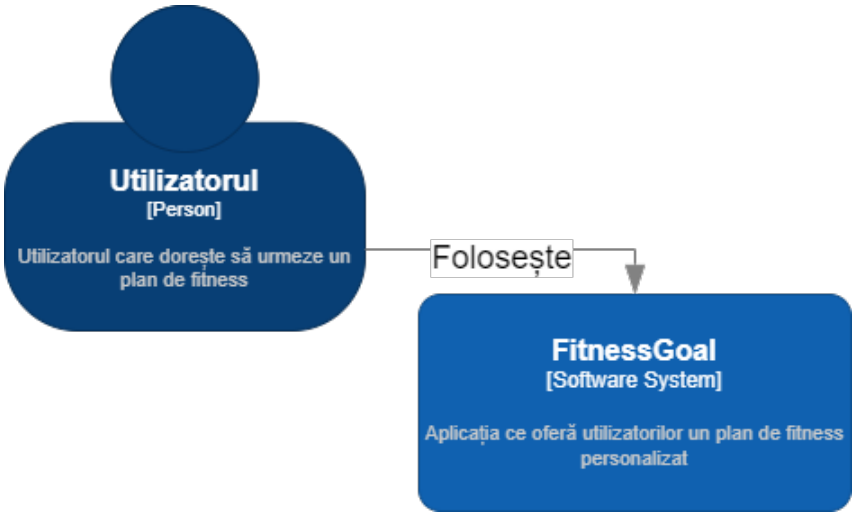


Diagrama C4 Nivelul 1

Imaginea de mai sus reprezintă diagrama de context a aplicației și are rolul de a oferi o imagine generală asupra sistemului *software* și interacțiunea sa cu entitățile externe. Există o singură entitate terță, utilizatorul, care folosește *feature*-urile aplicației.

3.1.2 Diagrama de containere

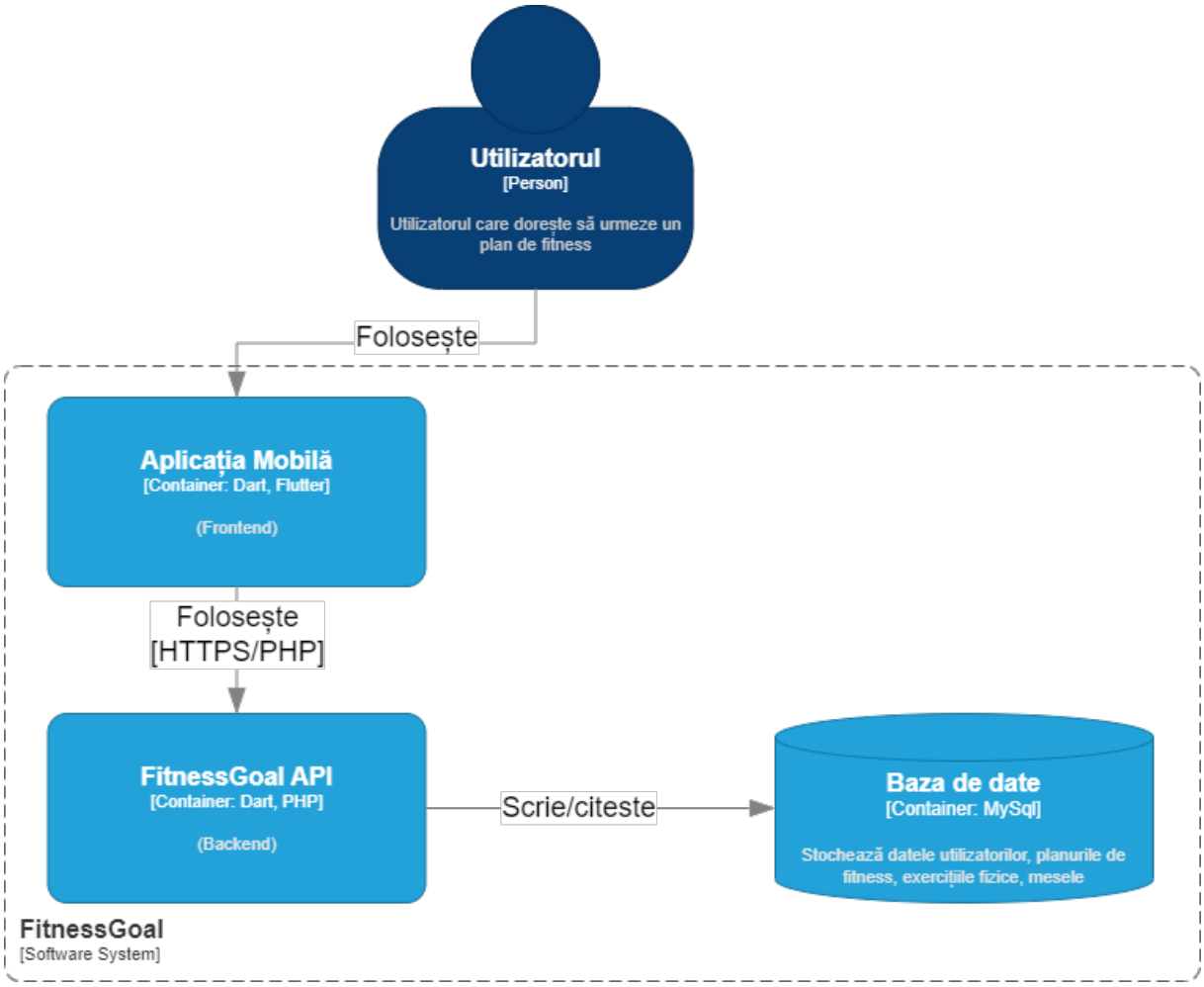


Diagrama C4 Nivelul 2

Sistemul *software* este compus din containerele: Aplicația Mobilă, *FitnessGoal API* și Baza de date.

Aplicația Mobilă (*Frontend*) oferă utilizatorului o interfață intuitivă, pentru o experiență cât mai plăcută.

FitnessGoal API (Backend) asigură funcționalitatea aplicației și comunicarea cu baza de date.

Baza de date stochează informațiile și le gestionează eficient pentru o funcționare eficientă a aplicației.

3.1.3 Diagrama de componente: Aplicația Mobilă FitnessGoal

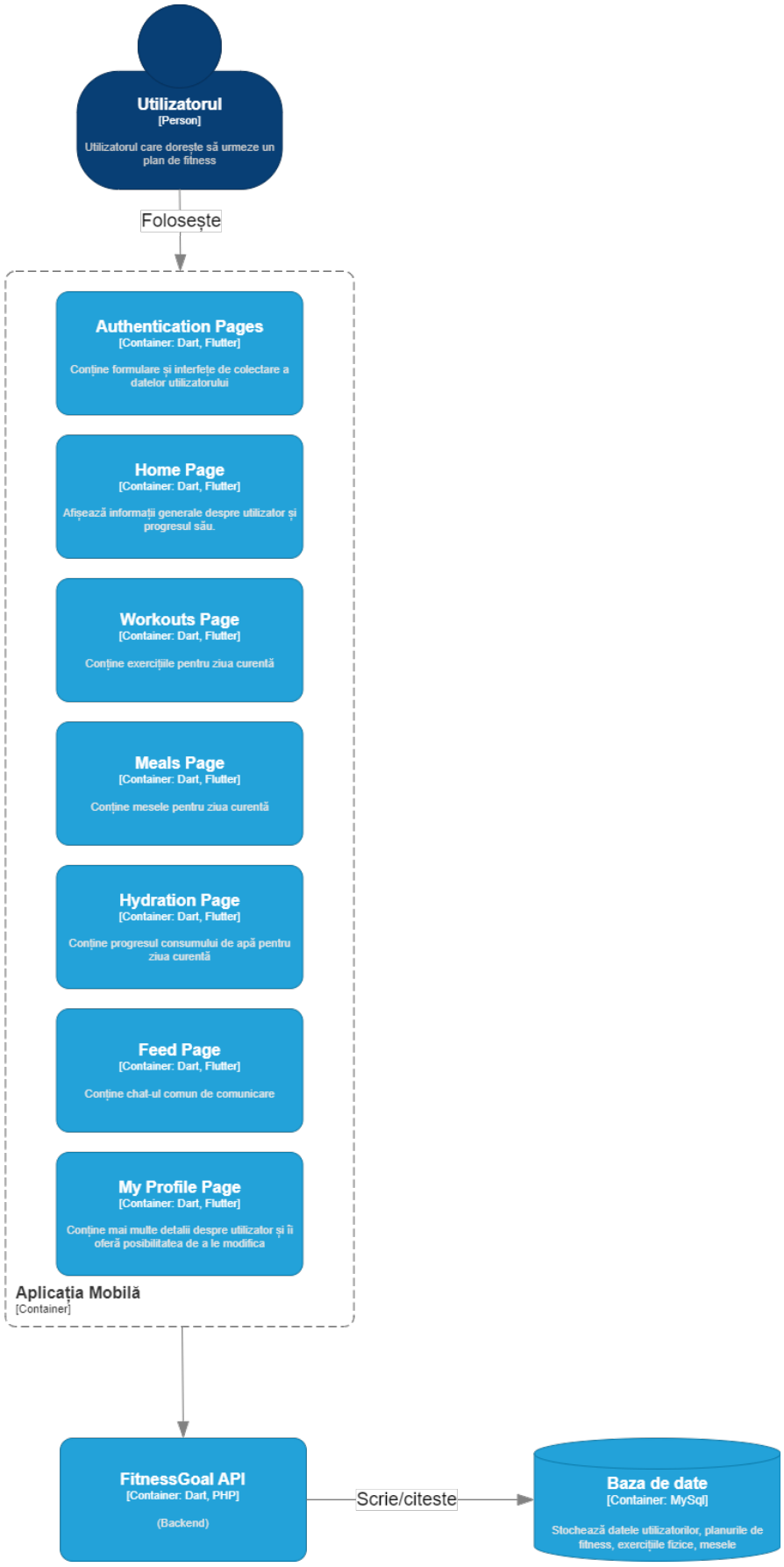


Diagrama C4 Nivelul 3 Aplicația FitnessGoal

Această diagramă cuprinde componentele interfeței de utilizator, având rol de navigare prin aplicație, reprezentând partea de client a acesteia.

Paginile de autentificare colectează datele utilizatorilor prin intermediul formularelor, apoi acestea sunt transmise sistemului de autentificare.

Pagina *Home* oferă utilizatorului o interfață grafică ce conține progresul său de-a lungul îndeplinirii planului de fitness, un citat motivațional și câteva informații generale despre utilizator.

Pagina *Workouts* conține toate exercițiile din ziua respectivă și un buton, care îl redirecționează pe utilizator către pagina destinată adăugării de exerciții.

Pagina *Meals* conține mesele generate de aplicație pentru trei zile, începând cu ziua de azi.

Pagina *Hydration* afișează progresul consumului de apă și posibilitatea de a-l actualiza.

Pagina *Feed* este interfața pentru mesajele utilizatorilor și oferă posibilitatea filtrării și adăugării postărilor.

Pagina *My Profile* conține date amănunțite despre utilizator și permite actualizarea lor, în plus oferă grafice ale progresului dezvoltării înălțimii și greutății utilizatorului.

3.1.4 Diagrama de componente: FitnessGoal API

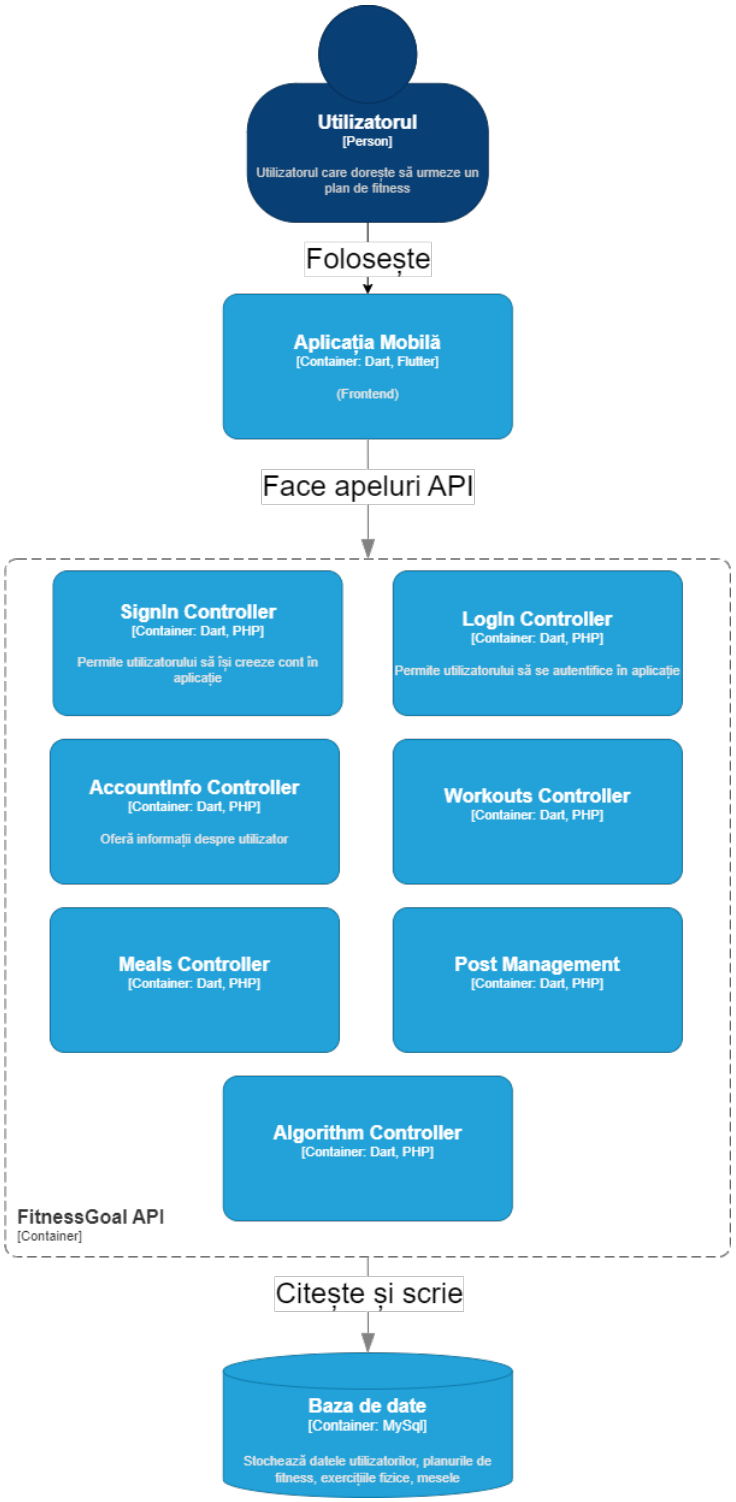


Diagrama C4 Nivelul 3 FitnessGoal API

FitnessGoal API sau Serverul aplicației conține componente de *backend* prin intermediul cărora se comunică cu baza de date și se generează planul de fitness.

Sign In Controller și *Log In Controller* oferă funcționalitatea de autentificare în cadrul aplicației. Prin intermediul *Sign In Controller* se introduc datele utilizatorului în baza de date. *Log In Controller* citește din baza de date și loghează utilizatorul în contul

său.

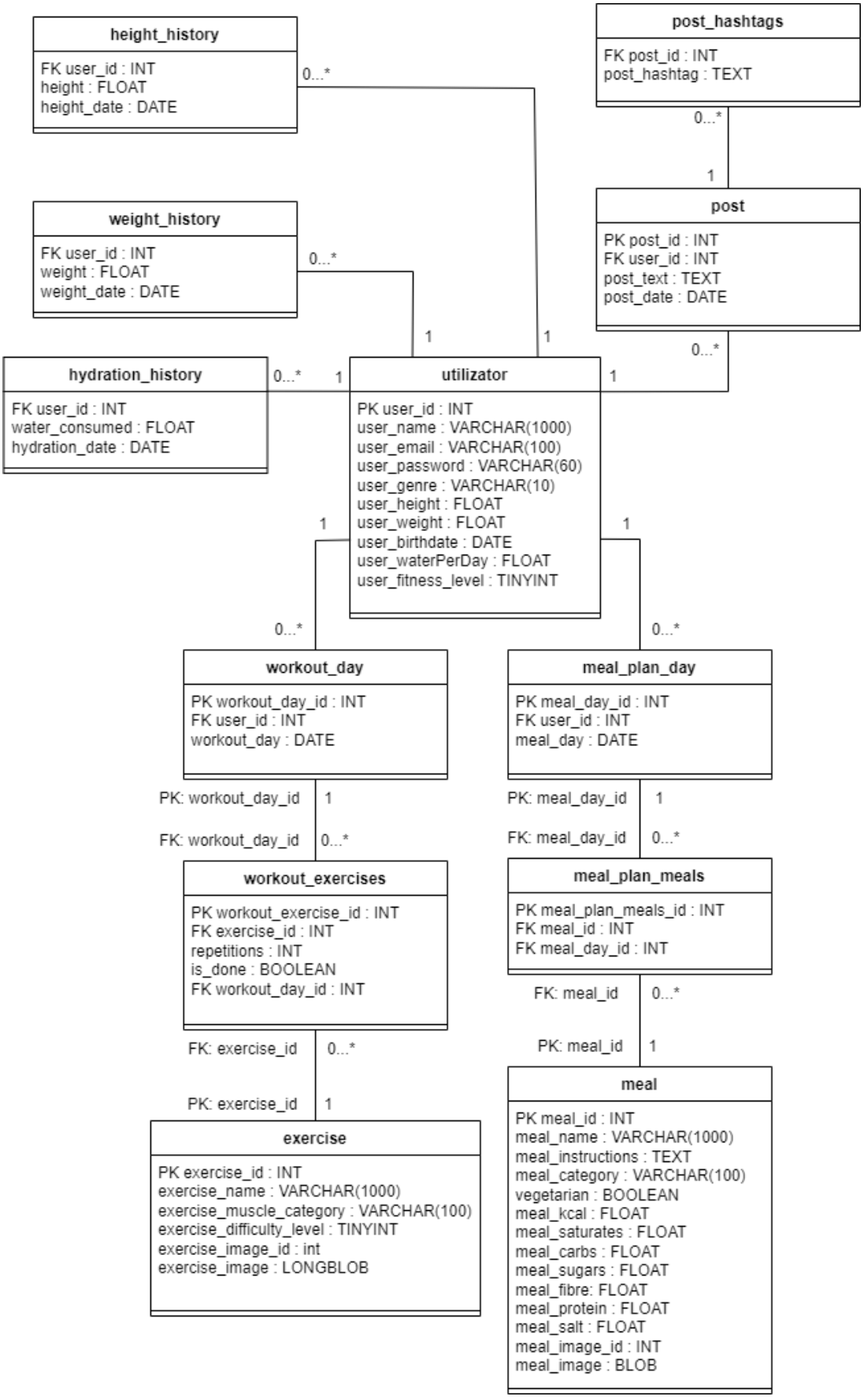
Account Info Controller oferă informații despre utilizator din baza de date.

Workouts Controller citește din baza de date planul de exerciții și îl trimite către interfață.

Meals Controller citește planul de fitness de mese, iar apoi îl oferă utilizatorului.

Algorithm Controller reprezintă algoritmii de generare a planului de fitness bazat pe caracteristicile utilizatorului.

3.1.5 Structura bazei de date



Structura bazei de date

Baza de date este structurată în jurul tabelei utilizator, care menține relații de tip *one-to-many* cu celelalte entități.

În tabela *"height-history"* este stocat progresul schimbării înălțimii tuturor utilizatorilor în timp. Ea are drept cheie străină id-ul utilizatorilor respectivi. Celelalte coloane din această tabelă sunt *"height"* și *"height-date"*, combinația cărora stochează înregistrarea înălțimii dintr-o zi anume. Respectiv un utilizator poate avea maxim o actualizare pe zi a înălțimii sale.

În tabela *"weight-history"*, similar ca în tabela descrisă anterior, este stocat progresul schimbării greutății tuturor utilizatorilor în timp. Coloana *"user-id"* joacă rol de cheie străină, coloanele *"weight"* și *"weight-date"* memorează o actualizare a progresului greutății utilizatorului.

În tabela *"hydration-history"* se păstrează progresul în timp a consumului de apă a tuturor utilizatorilor. Coloana *"user-id"* este cheie străină în această tabelă. Similar cazului tabelelor menționate anterior, combinația *"water-consumed"* și *"hydration-date"* stochează maxim o înregistrare pe zi.

Entitatea de tip *"post"* poate aparține unui singur utilizator, iar la rândul său, utilizatorul poate avea mai multe postări. Atributul *"post-text"* stochează textul postării în sine, iar atributul *"post-date"* stochează data la care a fost făcută postarea în cauză.

Postările pot avea mai multe *hashtag*-uri. Coloana *"post-id"* joacă rol de cheie străină în tabela *"post-hashtags"*, astfel memorând *hashtag*-urile postărilor.

Pentru a stoca planul de exerciții fizice am decis ca utilizatorul să aibă 0 sau mai multe zile de *"workout"*, stocate în tabela *"workout-day"*, care la rândul lor, conțin mai multe seturi de exerciții, stocate în tabela *"workout-exercises"*. Atributul *"exercise-id"* este cheia străină, datorită căreia avem acces la entități de tipul *"exercise"*. Tabela mai conține atribute precum *"repetitions"* care memorează numărul de repetări pentru un exercițiu anume și *"is-done"*, pentru a ști dacă exercițiul a fost deja executat. Atributul *"workout-day-id"* stochează cheia străină, pentru a ști cărei zile aparține setul de exerciții dat.

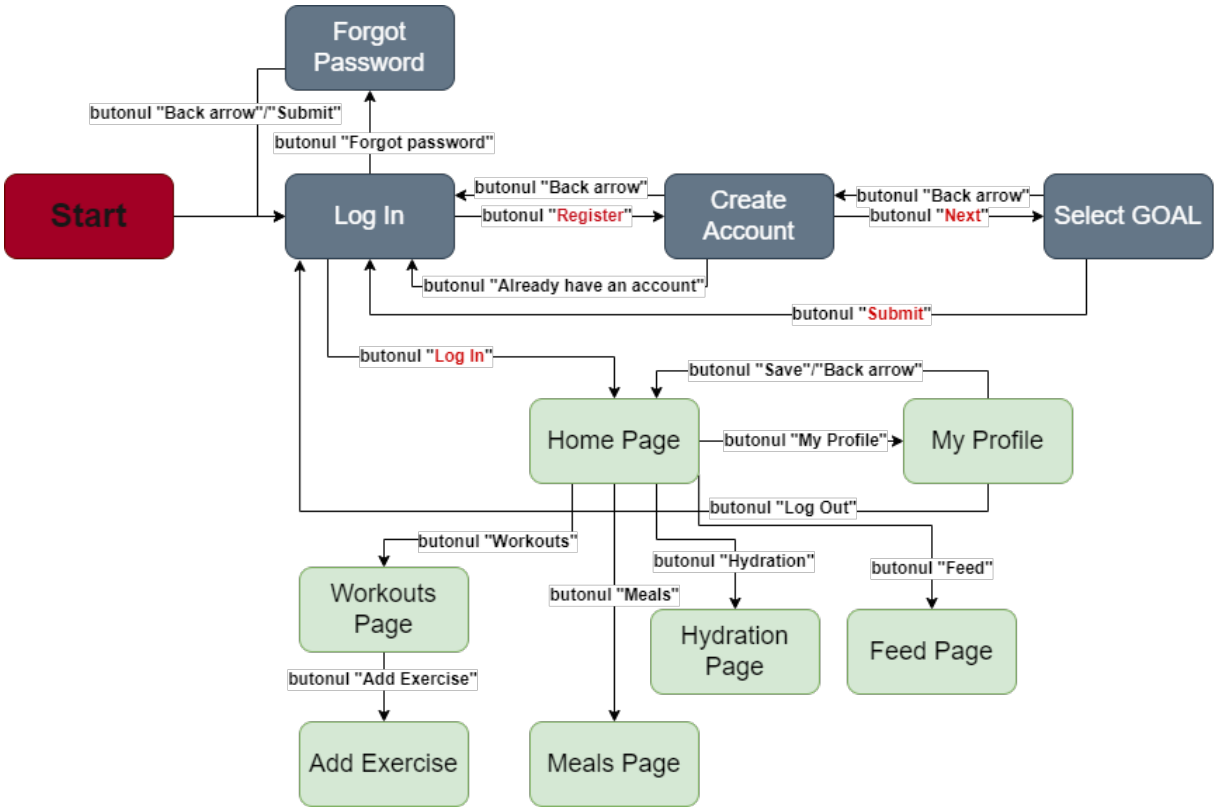
Tabela *"exercise"* stochează informații despre exercițiul în sine, precum, numele, categoria musculară afectată, nivelul de dificultate, și imaginea în care este redat un exemplu de îndeplinire al acelui exercițiu.

Pentru a stoca planul de mese am folosit o logică similară, utilizatorul poate avea 0 sau mai multe zile de *"meals"* stocate în tabela *"meal-plan-day"*, care la rândul lor conțin mai multe mese sub forma entității *"meal-plan-meals"*, care conține atributul

"meal-id", fiind cheie străină pentru a avea acces la entități de tipul *"meal"*. Mai avem cheia străină *"meal-day-id"*, pentru a ști cărei zile aparține masa respectivă.

Tabela *"meals"* conține înregistrări despre mesele disponibile, precum : nume, instrucțiuni, categorie, dacă e vegetariană, cantitatea de kilocalorii, saturați, carbohidrați, zaharuri, fibre, proteină, sare și imaginea produsului final.

3.1.6 Diagrama *user flow*



Fluxul utilizatorului

Diagrama de mai sus ne ajută să înțelegem posibilul flux al utilizatorului în timpul folosirii aplicației.

Faza de autentificare

Pentru început utilizatorul este direcționat către pagina Log In, unde întâlnim butoanele principale:

- *"Forgot Password"*, care ne redirecționează pe pagina *"Forgot Password"*, unde ne putem recupera accesul la cont în cazul în care am uitat parola, prin intermediul adresei de email.

- *"Register"*, care ne redirecționează către pagina *"Create Account"*, unde utilizatorul introduce datele sale personale precum: nume, email, parolă, înălțime, greutate, gen, data nașterii, dacă e vegetarian, ulterior prin intermediul butonului *"Next"* ajungem pe pagina *"Select GOAL"*, unde utilizatorul își alege scopul său principal, pe care vrea să îl atingă, și nivelul său de fitness. În urma completării tuturor informațiilor, la apăsarea butonului *"Submit"*, utilizatorul este redirecționat înapoi spre pagina de *"Log In"*.

- *"Log In"*, apăsarea căruia ne redirecționează spre pagina principală a aplicației, *"Home"*.

Folosirea implicită a aplicației

După faza de autentificare, utilizatorul este redirecționat pe pagina de *"Home"*. De acolo putem naviga pe celelalte pagini principale prin intermediul *Bottom Navigation Bar*, în diagramă am indicat doar accesul din pagina de Home către celelalte pagini pentru a fi mai ușor de înțeles.

Din pagina *Workouts*, accesând un exercițiu, avem *pop-up*-ul cu informația respectivului exercițiu și următoarele butoane:

- *"Delete"*, care șterge exercițiul respectiv din baza de date și face un *refresh* la pagina de *"Workouts"*.

- *"Done"*, care actualizează exercițiul respectiv din baza de date ca fiind realizat și face un *refresh* la pagina de *"Workouts"*.

- *"Close"*, care ne întoarce pe pagina de *"Workouts"*.

Pentru a modifica programul de exerciții din ziua de azi, utilizatorul poate naviga către pagina de adăugare de exerciții prin apăsarea butonului *"Add Exercise"*.

Pagina *"Add Exercise"* conține toate exercițiile disponibile din baza de date și oferă posibilitatea de a filtra respectivele exerciții după denumirea lor și denumirea zonei musculare de acțiune. În urma apăsării butonului exercițiului dorit apare un *pop-up*, în interiorul căruia avem informații despre exercițiul pe care dorim să îl adăugăm, precum: imaginea care ne sugerează forma de execuție a exercițiului dat, zona musculară de acțiune, și un *input* pentru numărul de repetări dorit. Prin apăsarea singurului buton disponibil din acest *pop-up*, *"Add"*, exercițiul va fi adăugat pentru ziua de antrenament din prezent și ne va întoarce către lista de exerciții din pagina *"Add Exercise"*.

Similar paginii de *"Workouts"*, pagina *"Meals"* conține butoane, reprezentând mesele din ziua respectivă, ziua de mâine și ziua de după. La apăsarea unui buton corespunzător mesei vom declanșa un *pop-up*, unde avem prezentă informația despre masa respectivă, precum: imaginea rezultatului final, nume, ingrediente, instrucțiuni, informații nutriționale. La apăsarea butonului *"Close"*, utilizatorul este reîntors la pagina *"Meals"*.

Pagina *"Hydration"* conține butonul *"Add quantity"*, care declanșează un *"pop-up"*, care permite utilizatorului adăugarea unei cantități de apă consumate. Acesta contine următoarele butoane:

- *"100 ml cup "*, care adaugă în baza de date, la progresul consumului de apă, cantitatea de 100 ml, apoi ne redirecționează către pagina de *"Hydration"*.
- *"200 ml cup "*, care adaugă în baza de date, la progresul consumului de apă, cantitatea de 200 ml, apoi ne redirecționează către pagina de *"Hydration"*.
- *"400 ml cup "*, care adaugă în baza de date, la progresul consumului de apă, cantitatea de 400 ml, apoi ne redirecționează către pagina de *"Hydration"*.
- *"Add"*, care adaugă în baza de date, la progresul consumului de apă, cantitatea introdusă de utilizator în *field*-ul de mai sus.

Pe pagina de *"Feed"* putem observa următoarele butoane:

- *"Refresh"*, care face un *refresh* postărilor, și încarcă din baza de date lista actualizată de postări.
- *"Make a post"*, apăsarea căruia declanșează un *pop-up*, unde utilizatorul poate să își introducă textul postării și *hashtag*-urile dorite, iar prin apăsarea butonului *"Post"*, postarea este introdusă în baza de date, iar utilizatorul este redirecționat către pagina *"Feed"*.
- *"Filter"*, care oferă posibilitatea de a filtra postările existente după *hashtag*. La apăsarea lui apare un *pop-up*, unde putem introduce *hashtag*-urile după care vrem să facem filtrarea, iar la apăsarea butonului *"Filter"*, suntem redirecționați pe pagina *"Feed"*, unde avem lista actualizată de postări.

Pagina *"My Profile"* poate fi accesată din toate paginile principale ale aplicației. Ea conține câmpuri de completare spre modificare a informațiilor utilizatorului. Buto-

nul *"Save"* actualizează baza de date cu informațiile introduse și ne redirecționează pe pagina *"Home"*.

Acțiunea butonului *"Log Out"* este de a deconecta utilizatorul din contul său și îl redirecționează pe pagina *"Log In"*.

3.2 Colectarea datelor utilizatorului

Colectarea datelor inițiale ale utilizatorului (nume, gen, data nașterii, greutate, înălțime, scop, nivel de fitness) are loc în faza de înregistrare a respectivului utilizator prin intermediul formularelor și interfețelor de utilizator.

Este necesară colectarea acestor date în faza inițială, deoarece la înregistrarea utilizatorului are loc rularea algoritmilor de generare a planurilor de fitness (de exerciții fizice și mese), cât și determinarea cantității de apă recomandate per zi.

Ulterior progresul utilizatorului este colectat zi de zi, prin îndeplinirea exercițiilor, consumului de apă și actualizarea datelor personale (greutate, înălțime).

3.3 Planul de *workouts*

3.3.1 Stocarea în baza de date

Planul de exerciții generat este stocat în baza de date utilizând două tabele, una pentru stocarea id-ului utilizatorului și zilele următoare și cealaltă conținând id-ul din tabelul descris anterior, id-ul exercițiilor asignate pentru ziua respectivă, un indicator, care ne sugerează dacă exercițiul a fost îndeplinit și numărul de repetări.

Exercițiile sunt categorizate în funcție de grupurile musculare țintă și nivelului de dificultate.

3.3.2 Algoritmul folosit pentru generarea numărului inițial de repetări

Pentru început am avut nevoie de un algoritm de generare al numărului de repetări de exerciții, considerând datele utilizatorului.

În cadrul algoritmului am luat în considerare scopul, greutatea, înălțimea, genul, nivelul de fitness și vârsta utilizatorului.

În primul rând am împărțit algoritmul pe cazuri după scop, în modul următor:

- *"Slăbire"* - în cazul în care persoana este de gen feminin s-a folosit următoarea formulă de calcul al repetărilor:
  `repetitions = floor(3500/(userHeight+userWeight)) * userFitnessLevel;`
  Pentru persoane de genul masculin s-a folosit formula:
  `repetitions = floor(3000/(userHeight+userWeight)) * userFitnessLevel;`
  Apoi numărul de repetări este actualizat în funcție de vârsta utilizatorului.

  Dacă aceasta se află în intervalul 30-70 de ani, atunci numărul de repetări se reduce cu 10 procente.

  Dacă aceasta trece vârsta de 70 de ani, atunci numărul de repetări se reduce cu 30 de procente.

- *"Creșterea masei musculare"* - în cazul în care persoana aparține genului feminin, se aplică următoarea formulă de calcul:
  `repetitions = floor(userWeight / 4) * userFitnessLevel;`
  `repetitions = floor(repetitions * 0.9);`
  În cazul utilizatorilor de gen masculin se folosește aceeași formulă de calcul, doar că nu mai facem reducerea numărului de repetări cu 10 procente.

  Apoi numărul de repetări este actualizat în funcție de vârsta utilizatorului.

  Dacă aceasta se află în intervalul 30-70 de ani, atunci numărul de repetări se reduce cu 20 de procente.

  Dacă aceasta trece vârsta de 70 de ani, atunci numărul de repetări se reduce cu 50 de procente.

  Urmărim o reducere mai mare de repetări odată cu creșterea vârstei, comparativ cu scopul anterior, deoarece la o vârstă înaintată procesul de regenerare al mușchilor nu mai funcționează la fel de eficient, deci o supraîncărcare a mușchilor ar duce mai mult la distrugerea masei musculare.

- *"A fi activ"* - aici nu se mai face o diferențiere după gen, formula aplicată în toate cazurile fiind:
  `repetitions = floor(400 / userAge) * userFitnessLevel;`

Deci, cu cat vârsta utilizatorului este mai înaintată, cu atât numărul de repetări va fi mai mic, din cauza diferenței de metabolism.

Rolul parametrilor personali ai utilizatorilor pentru determinarea numărului de repetări este menționat și detaliat în Capitolul 1 - Fundamentare teoretică, subcapitolul 1.4 - Importanța factorilor individuali ale individului în conceperea planului de fitness..

Numărul de repetări este determinat și de tipul de antrenament, după cum am menționat și în fundamentarea teoretică, **hipertrofia** este un factor important în construcția mușchilor.

Adaptarea directă a fibrelor musculare (într-o oarecare măsură elastice) cere ca stimulul să fie specific și constant. Orice deviere de la stimul schimbă particularitățile metodicii de antrenament, astfel hipertrofia și tipul acesteia este direct proporțională cu cerințele:

- Antrenamentul **culturiștilor** se bazează pe un număr mai voluminos de exerciții și repetări, însă cu o intensitate scăzută.
- Antrenamentul sportivilor din *powerlifting* se bazează pe un număr mai mic de exerciții și repetări, însă cu o intensitate mai ridicată.

Hipertrofia miofibrilară este specifică antrenamentelor cu un număr de repetări mai mic, iar cea **sarcoplasmatică** respectiv cu un număr mai mare de repetări.

Codul întregului algoritm este prezent în anexa 1.

3.3.3 Algoritmul folosit pentru alegerea exercițiilor

În cazul exercițiilor alese pentru fiecare utilizator în parte, am luat în considerare scopul și nivelul de fitness.

Inițial se determină numărul de exerciții din marile categorii de exerciții (Cardio și de Forță), în dependență de nivelul de fitness al utilizatorului în felul următor:

```
strenghExercisesNumber = 3 * userFitnessLevel;
cardioExercisesNumber = 3 * userFitnessLevel;
```

Apoi algoritmul se împarte pe cazuri de scop în felul următor:

- *"Slăbire"* - în cazul acestui scop numărul de exerciții de tip "Forță" se reduce:
  `strenghExercisesNumber = floor(strenghExercisesNumber / 2);`
  Apoi se extrage din baza de date numărul respectiv de exerciții din fiecare categorie.

- *"Creșterea masei musculare"* - în cazul acestui scop numărul de exerciții de tip "Cardio" se reduce:
  `cardioExercisesNumber = floor(cardioExercisesNumber / 2);`
  Apoi se extrage din baza de date numărul respectiv de exerciții din fiecare categorie.

- *"A fi activ"* - în cazul acestui scop numărul de exerciții de tip "Cardio" și numărul de exerciții de tip "Forță" este egal. Însă, un lucru important este ca se extrag exerciții cu un nivel de dificultate egal cu 1, dat fiind că acest scop urmărește depunerea unui efort mai *light*.

O condiție în plus este ca exercițiile selectate să nu aibă un grad de dificultate mai mare decât nivelul de fitness al utilizatorului.

Codul întregului algoritm este prezent în anexa 2.

3.3.4 Algoritmul folosit pentru generarea planului de exerciții

La conceperea algoritmului pentru generarea planului de fitness a fost nevoie de a adăuga o progresie din punct de vedere al numărului de repetări și al nivelului de fitness, deoarece în timp, organismul uman se adaptează condițiilor cărora este expus. Deci, pentru a evita stagnarea progresului am decis să adaug această progresie pe parcursul trecerii timpului.

Algoritmul parcurge perioada planului de fitness și o populează cu exerciții selectate prin intermediul algoritmului de alegere a exercițiilor și numărul de repetări determinat prin intermediul algoritmului de generare a numărului inițial de repetări.

În decursul trecerii timpului în cadrul planului de exerciții, la numărul de repetări se adaugă progresia, și la fel evoluează nivelul de fitness al utilizatorului.

O parte din algoritm:

```
1 for($day=0; $day<$workoutDuration; $day++){
2     $currentDate = $startDay->format('Y-m-d');
3     $exercises = array();
```

```
4       $exercises = chooseExercises($userGoal, $userFitnessLevel, $db);
5
6       $sql1 = "INSERT INTO workout_day(user_id, workout_day)
           VALUES('".$userId."', '".$currentDate."')";
7       if(mysqli_query($db, $sql1)){
8           $insertedId = mysqli_insert_id($db);
9           for($i=0; $i<count($exercises); $i++){
10              $sql2 = "INSERT INTO workout_exercises(exercise_id,
                   repetitions, is_done, workout_day_id)
                   VALUES('".$exercises[$i]."',
                   round('".$repetitions."'), 0, '".$insertedId."')";
11              if(mysqli_query($db, $sql2)){
12              }
13              else{
14                  return "Failure";
15              }
16          }
17          //Increase the number of repetitions over time
18          $repetitions += $progression;
19
20          //Increase the difficulty level of exercise over time
21          if ($day % 10 == 0){
22              $userFitnessLevel = min(3, $userFitnessLevel + 1);
23          }
24
25          $startDay->modify('+1 day');
26      }
27      else{
28          return "Failure";
29      }
30  }
```

Codul întregului algoritm este prezent în anexa 3.

3.3.5 Flexibilitatea și ajustabilitatea oferită de aplicație

Aplicația oferă opțiunea de a schimba planul deja generat de către algoritmi, astfel oferind utilizatorului o flexibilitate în alegerea de exerciții, numărul de repetări și intensitatea complexului de exerciții.

Prin completarea exercițiilor, utilizatorul își înregistrează progresul în cadrul aplicației, astfel putând să își urmărească progresul în timp.

3.4 Planul de *meals*

3.4.1 Stocarea în baza de date

Planul de mese generat este stocat în baza de date utilizând două tabele, asemănător planului de exerciții.

Mesele sunt categorizate în funcție de tipul mesei (mic dejun, prânz sau cină).

3.4.2 Algoritmul folosit la alegerea meselor

Pentru alegerea meselor individuale s-a luat în considerare scopul și greutatea utilizatorului. Pentru utilizatorii vegetarieni s-a luat în calcul și acest aspect în alegerea bucatelor.

Inițial s-a calculat cantitatea necesară de kilocalorii pentru ca organismul utilizatorului să își poată menține toate procesele de bază, și în același timp să piardă în greutate, prin formula oferita de *Harvard University*:

```
1    $userWeightPounds = $userWeight * 2.205;
2    $meintenanceCal = $userWeightPounds * 15;
3    $kcalLW = floor((($meintenanceCal - $userWeightPounds) - 700)/3);
```

În cazul scopului *"Slăbire"*, au fost alese mesele cu consum limitat de grăsimi ($<= 15$), carbohidrați ($<= 42$), zaharuri ($<= 8$) și kilocalorii ($<=$ *"kcalLW"*).

Un exemplu de interogare folosită de către algoritm:

```
1    $sqlBreakfast = "select meal_id from meal where vegetarian =
        '".$userVegetarian."' and meal_fat<=15 and meal_carbs<=42 and
        meal_sugars<=8 and meal_kcal<='".$kcalLW."' and meal_category
        = 'Breakfast' order by rand() limit 1";
```

În cazul obiectivului *"Creșterea masei musculare"*, mesele au fost extrase din baza de date, însă, acestea au fost ordonate descrescător după coloana ce stochează cantitatea de proteină, dat fiind faptul că ea contribuie la refacerea țesuturilor musculare.

Un exemplu de interogare folosită de către algoritm:

```
$sqlBreakfast = "select meal_id from (select * from meal where
    vegetarian = '".$userVegetarian."' and meal_category =
    'Breakfast' order by meal_protein desc limit 10) as subquery
    order by rand() limit 1";
```

Pentru scopul de *"A fi activ"* am luat în considerare doar faptul că utilizatorul e vegetarian sau nu. Nu a fost nevoie de o limitare a listei meselor din care algoritmul alege, deoarece toate mesele introduse în baza de date sunt sănătoase și intră în cadrul scopului propus.

Codul întregului algoritm este prezent în anexa 4.

3.4.3 Algoritmul folosit la generarea planului de mese

Acest algoritm populează tabela planului de mese, pentru perioada specificată, cu mesele generate de algoritmul de alegere a meselor.

Codul întregului algoritm este prezent în anexa 5.

3.5 Monitorizarea hidratării

Un alt *feature* al aplicației este monitorizarea hidratării. Din cauză că hidratarea joacă un rol important în fitness, i-am dedicat o pagină aparte în cadrul aplicației.

Aici utilizatorul poate să-și urmărească progresul hidratării, putând să actualizeze cantitatea de apă consumată, pentru ziua respectivă, pentru a atinge cantitatea recomandată de algoritm.

3.5.1 Algoritmul folosit pentru determinarea cantității de apă recomandate

```
function getWaterPerDayIntake($userWeight, $userGoal){
    $userWeightPounds = $userWeight * 2.205;
    $waterIntake = ($userWeightPounds * 0.5) / 33.814;
```

```
4       switch ($userGoal) {
5           case 'Lose Weight':
6               $waterIntake = $waterIntake + 0.25;
7               break;
8           case 'Get Muscle':
9               $waterIntake = $waterIntake + 0.35;
10              break;
11      }
12      $waterIntake = round($waterIntake, 1);
13      return $waterIntake;
14  }
```

Conform articolului din U.S. News and World Report, și mai exact regulii de bază pentru stabilirea cantității necesare de consum de apă, o persoană trebuie să consume jumătate din greutatea sa (în lbs) în uncii de apă.

În cazul nostru, utilizatorul îndeplinește și exerciții zilnice, din această cauză, pe lângă consumul de apă calculat, trebuie adăugată o cantitate adițională în felul următor:

- Deci pentru scopul *"Creșterea masei musculare"* este nevoie să adăugăm la consumul de apă 350 de mililitri în plus, pentru a fi siguri că recuperăm apa pierdută din organism în timpul efectuării exercițiilor și asigurăm procesele de sinteză a proteinei și hidratarea țesutului muscular.
- Dacă scopul utilizatorului este *"Slăbirea"*, atunci algoritmul adaugă 250 de mililitri de apă consumului recomandat, din cauză că exercițiile de cardio au ca efect pierderi de apă din organism.

3.6 Interacțiunea utilizatorilor prin *chat*

O funcționalitate importantă a aplicației este *chat*-ul comun. Acesta oferă utilizatorilor posibilitatea de a discuta subiecte de interes comun.

Utilizatorul și-ar putea expune părerea despre anumite seturi de exerciții, astfel la rândul lor alți utilizatori pot să-și actualizeze planul de exerciții în dependență de părerile populare.

Un alt beneficiu al acestei funcționalități este sprijinul oferit persoanelor care caută sfaturi în domeniul fitness.

Chat-ul poate servi drept instrument de *feedback* pentru utilizatori, unde își pot exprima părerile despre aplicație. Dezvoltatorii ar putea extrage informații valoroase pentru îmbunătățirea ulterioară a experienței utilizatorilor și de a adapta aplicația conform preferințelor și nevoilor acestora.

Un plus semnificativ adus de acest *feature* este crearea unei comunități în cadrul aplicației, prin urmare generând un sentiment de responsabilitate și motivație.

Concluzii

În lucrarea de licență am abordat dezvoltarea unei aplicații mobile de fitness cu o metodă personalizată de generare de exerciții și mese zilnice, adaptabilă nevoilor utilizatorilor.

Lucrarea dată aduce beneficii în domeniul fitness, fiind o aplicație ce combină 3 factori ce influențează direct sănătatea, activitatea fizică, alimentația și hidratarea.

Tehnologia de generare a unui plan de fitness personalizat conform caracteristicilor utilizatorilor constituie o deosebire a aplicației *FitnessGoal* față de celelalte aplicații de pe piață.

În viitor aplicația ar putea fi îmbunătățită prin o populare mai mare a bazei de date cu exerciții fizice și mese nutriționale.

Cu timpul, algoritmii ar putea fi îmbunătățiți conform *feedback*-ului utilizatorilor, fiind posibilă apariția nevoii unui spectru mai larg de scopuri.

La momentul dat, aplicația oferă doar o perioadă de probă de beneficiere de funcționalități, odată cu completarea unui program de fitness, utilizatorul nu își mai poate genera un alt plan, pe viitor această funcție poate fi inclusă într-o versiune *premium* a aplicației.

Pe parcursul dezvoltării aplicației am testat *output*-ul algoritmilor de generare a planului (de exerciții și mese) personalizat conform caracteristicilor corpului meu și scopului urmărit de mine, astfel am realizat că aplicația m-a ajutat să îmi îmbunătățesc starea de sănătate și să duc un mod de viață mai activ. În acest mod am evitat sedentarismul în timpul lucrului asupra lucrării de licență.

Folosirea aplicației *FitnessGoal*, elimină necesitatea de a avea câte o aplicație instalată pentru fiecare feature în parte (de antrenament, de alimentație și de hidratare), combinându-le într-o singură aplicație, astfel simplifică experiența utilizatorului și îi economisește timp.

Ține locul unui antrenor personal, astfel elimină nevoia de a achita serviciile

unuia.

Ea oferă un plan personalizat și flexibil în dependența fiecărui utilizator. Prin urmare, din aceste motive, personal, recomand folosirea aplicației *FitnessGoal* în schimbul aplicațiilor deja existente pe piață.

Bibliografie

- Suveică Luminița - *Nutriția umană - Ghid Practic* (2020)
- Xiaoli Liu, Satu Tamminen, Topi Korhonen, Juha Röning - *How Physical Exercise Level Affects Sleep Quality? Analyzing Big Data Collected from Wearables* (2019)
- Doina Alin-Mihai - *Hipertrofia Musculară*
- Brian Keane - *The Fitness Mindset* (2017)
- Sparkx Systems - *Database Modeling in UML*
- Dart packages - *pub.dev*

Anexe

Anexa 1: Algoritmul de calculare a repetărilor

Algoritmul respectiv (ca o funcție în limbajul de programare PHP):

```
1 function calculateInitialRepetitions($userGoal, $userWeight, $
    userHeight, $userGenre, $userFitnessLevel, $userAge) {
2
3     $repetitions = 0;
4     switch ($userGoal) {
5         case "Lose Weight":
6             if($userGenre == "Female"){ //->more reps
7                 $repetitions = floor(5000/($userHeight+$userWeight))
                      * $userFitnessLevel;
8             }
9             else{ //male ->fewer reps
10                $repetitions = floor(4000/($userHeight+$userWeight))
                      * $userFitnessLevel;
11            }
12
13            if($userAge > 30){ //middle age
14                $repetitions = floor($repetitions * 0.9); // reduce
                      by 10%
15            }
16            else if($userAge > 70){ //olders
17                $repetitions = floor($repetitions * 0.7); // reduce
                      by 30%
18            }
19            break;
20        case "Get Muscle":
```

```
21              if($userGenre == "Female"){ //->less reps
22                  $repetitions = floor($userWeight / 4) * $
                        userFitnessLevel;
23                  $repetitions = floor($repetitions * 0.9); // reduce
                        by 10%
24              }
25              else{ //male ->more reps
26                  $repetitions = floor($userWeight / 4) * $
                        userFitnessLevel;
27              }
28
29              if($userAge > 30){ //middle age
30                  $repetitions = floor($repetitions * 0.8); // reduce
                        by 20%
31              }
32              else if($userAge > 70){ //olders
33                  $repetitions = floor($repetitions * 0.5); // reduce
                        by 50%
34              }
35              break;
36          case "Be Active":
37              $repetitions = floor(500 / $userAge) * $userFitnessLevel;
38              break;
39      }
40
41      return $repetitions;
42  }
```

Anexa 2: Algoritmul pentru alegerea exercițiilor

Algoritmul respectiv (ca o funcție în limbajul de programare PHP):

```
1 function chooseExercises($userGoal, $userFitnessLevel, $db){
2     $exercises = array();
3     $strenghExercisesNumber = 3 * $userFitnessLevel;
4     $cardioExercisesNumber = 3 * $userFitnessLevel;
```

```
5
6       switch ($userGoal) {
7           case "Lose Weight":
8
9               $sqlLW = "select exercise_id from exercise where
                    exercise_muscle_category='Cardio' and
                    exercise_difficulty_level <= '".$userFitnessLevel."'
                    order by rand() limit ?";
10
11              $stmt = mysqli_prepare($db, $sqlLW);
12              mysqli_stmt_bind_param($stmt, "i", $
                    cardioExercisesNumber);
13              mysqli_stmt_execute($stmt);
14
15              $resultLW = mysqli_stmt_get_result($stmt);
16              while($row = mysqli_fetch_array($resultLW)){
17                  $exercises[]=intval($row['exercise_id']);
18              }
19
20              $sqlGM = "select exercise_id from exercise where
                    exercise_muscle_category!='Cardio' and
                    exercise_difficulty_level <= '".$userFitnessLevel."'
                    order by rand() limit ? ";
21
22              $stmt = mysqli_prepare($db, $sqlGM);
23              mysqli_stmt_bind_param($stmt, "i", $
                    strenghExercisesNumber);
24              mysqli_stmt_execute($stmt);
25
26              $resultGM = mysqli_stmt_get_result($stmt);
27              while($row = mysqli_fetch_array($resultGM)){
28                  $exercises[]=intval($row['exercise_id']);
29              }
30              break;
31
32          case "Get Muscle":
```

```
33              $cardioExercisesNumber = floor($cardioExercisesNumber /
                    2);
34              $sqlLW = "select exercise_id from exercise where
                    exercise_muscle_category='Cardio' and
                    exercise_difficulty_level <= '".$userFitnessLevel."'
                    order by rand() limit ? ";
35
36              $stmt = mysqli_prepare($db, $sqlLW);
37              mysqli_stmt_bind_param($stmt, "i", $
                    cardioExercisesNumber);
38              mysqli_stmt_execute($stmt);
39
40              $resultLW = mysqli_stmt_get_result($stmt);
41              while($row = mysqli_fetch_array($resultLW)){
42                  $exercises[]=intval($row['exercise_id']);
43              }
44
45              $sqlGM = "select exercise_id from exercise where
                    exercise_muscle_category!='Cardio' and
                    exercise_difficulty_level <= '".$userFitnessLevel."'
                    order by rand() limit ? ";
46
47              $stmt = mysqli_prepare($db, $sqlGM);
48              mysqli_stmt_bind_param($stmt, "i", $
                    strenghExercisesNumber);
49              mysqli_stmt_execute($stmt);
50
51              $resultGM = mysqli_stmt_get_result($stmt);
52              while($row = mysqli_fetch_array($resultGM)){
53                  $exercises[]=intval($row['exercise_id']);
54              }
55              break;
56
57          case "Be Active":
58
```

```
59              $sqlLW = "select exercise_id from exercise where
                    exercise_muscle_category='Cardio' and
                    exercise_difficulty_level = 1 order by rand() limit ?
                    ";
60
61              $stmt = mysqli_prepare($db, $sqlLW);
62              mysqli_stmt_bind_param($stmt, "i", $
                    cardioExercisesNumber);
63              mysqli_stmt_execute($stmt);
64
65              $resultLW = mysqli_stmt_get_result($stmt);
66              while($row = mysqli_fetch_array($resultLW)){
67                  $exercises[]=intval($row['exercise_id']);
68              }
69
70              $sqlGM = "select exercise_id from exercise where
                    exercise_muscle_category!='Cardio' and
                    exercise_difficulty_level = 1 order by rand() limit
                    ?";
71
72              $stmt = mysqli_prepare($db, $sqlGM);
73              mysqli_stmt_bind_param($stmt, "i", $
                    strenghExercisesNumber);
74              mysqli_stmt_execute($stmt);
75
76              $resultGM = mysqli_stmt_get_result($stmt);
77              while($row = mysqli_fetch_array($resultGM)){
78                  $exercises[]=intval($row['exercise_id']);
79              }
80              break;
81      }
82      shuffle($exercises);
83      return $exercises;
84  }
```

Anexa 3: Algoritmul pentru generarea planului de exerciții

Algoritmul respectiv (ca o funcție în limbajul de programare PHP):

```
function buildWorkoutPlan($userId, $userGenre, $userHeight, $
    userWeight, $userAge, $userFitnessLevel, $userGoal, $db){
    $workoutDuration = 30;
    $startDay = new DateTime();
    $repetitions = calculateInitialRepetitions($userGoal, $
        userWeight, $userHeight, $userGenre, $userFitnessLevel, $
        userAge);
    $progression = 10 / $workoutDuration;

    for($day=0; $day<$workoutDuration; $day++){
        $currentDate = $startDay->format('Y-m-d');
        $exercises = array();
        $exercises = chooseExercises($userGoal, $userFitnessLevel, $
            db);

        $sql1 = "INSERT INTO workout_day(user_id, workout_day)
            VALUES('".$userId."', '".$currentDate."')";
        if(mysqli_query($db, $sql1)){
            $insertedId = mysqli_insert_id($db);
            for($i=0; $i<count($exercises); $i++){
                $sql2 = "INSERT INTO workout_exercises(exercise_id,
                    repetitions, is_done, workout_day_id)
                    VALUES('".$exercises[$i]."',
                    round('".$repetitions."'), 0, '".$insertedId."')";
                if(mysqli_query($db, $sql2)){
                }
                else{
                    return "Failure";
                }
            }
            //Increase the number of repetitions over time
            $repetitions += $progression;

```

```
26              //Increase the difficulty level of exercise over time
27              if ($day % 10 == 0){
28                  $userFitnessLevel = min(3, $userFitnessLevel + 1);
29              }
30
31              $startDay->modify('+1 day');
32          }
33          else{
34              return "Failure";
35          }
36      }
37      return "Success";
38  }
```

Anexa 4: Algoritmul pentru alegerea meselor

Algoritmul respectiv (ca o funcție în limbajul de programare PHP):

```
1  function chooseMeals($userGoal, $userVegetarian, $userWeight, $db){
2      $meals = array();
3      $userWeightPounds = $userWeight * 2.205;
4      $meintenanceCal = $userWeightPounds * 15;
5      $kcalLW = floor((($meintenanceCal - $userWeightPounds) - 700)/3);
6      switch ($userGoal) {
7          case "Lose Weight":
8              if($userVegetarian == 1){
9                  $sqlBreakfast = "select meal_id from meal where
                       vegetarian = '".$userVegetarian."' and
                       meal_fat<=15 and meal_carbs<=42 and
                       meal_sugars<=8 and meal_kcal<='".$kcalLW."' and
                       meal_category = 'Breakfast' order by rand() limit
                       1";
10                 $resultBreakfast = mysqli_query($db, $sqlBreakfast);
11                 while($row = mysqli_fetch_array($resultBreakfast)){
12                     $meals[] = intval($row['meal_id']);
13                 }
```

```
14
15              $sqlLunch = "select meal_id from meal where
                   vegetarian = '".$userVegetarian."' and
                   meal_fat<=15 and meal_carbs<=42 and
                   meal_sugars<=8 and meal_kcal<='".$kcalLW."' and
                   meal_category = 'Lunch' order by rand() limit 1";
16              $resultLunch = mysqli_query($db, $sqlLunch);
17              while($row = mysqli_fetch_array($resultLunch)){
18                  $meals[] = intval($row['meal_id']);
19              }
20
21              $sqlDinner = "select meal_id from meal where
                   vegetarian = '".$userVegetarian."' and
                   meal_fat<=15 and meal_carbs<=42 and
                   meal_sugars<=8 and meal_kcal<='".$kcalLW."' and
                   meal_category = 'Dinner' order by rand() limit 1";
22              $resultDinner = mysqli_query($db, $sqlDinner);
23              while($row = mysqli_fetch_array($resultDinner)){
24                  $meals[] = intval($row['meal_id']);
25              }
26          }
27          else{
28              $sqlBreakfast = "select meal_id from meal where
                   meal_fat<=15 and meal_carbs<=42 and
                   meal_sugars<=8 and meal_kcal<='".$kcalLW."' and
                   meal_category = 'Breakfast' order by rand() limit
                   1";
29              $resultBreakfast = mysqli_query($db, $sqlBreakfast);
30              while($row = mysqli_fetch_array($resultBreakfast)){
31                  $meals[] = intval($row['meal_id']);
32              }
33
34              $sqlLunch = "select meal_id from meal meal_fat<=15
                   and meal_carbs<=42 and meal_sugars<=8 and
                   meal_kcal<='".$kcalLW."' and meal_category =
                   'Lunch' order by rand() limit 1";
```

```
35                $resultLunch = mysqli_query($db, $sqlLunch);
36                while($row = mysqli_fetch_array($resultLunch)){
37                    $meals[] = intval($row['meal_id']);
38                }
39
40                $sqlDinner = "select meal_id from meal where
                     meal_fat<=15 and meal_carbs<=42 and
                     meal_sugars<=8 and meal_kcal<='".$kcalLW."' and
                     meal_category = 'Dinner' order by rand() limit 1";
41                $resultDinner = mysqli_query($db, $sqlDinner);
42                while($row = mysqli_fetch_array($resultDinner)){
43                    $meals[] = intval($row['meal_id']);
44                }
45            }
46
47            break;
48
49        case "Get Muscle":
50
51            if($userVegetarian == 1){
52                $sqlBreakfast = "select meal_id from (select * from
                     meal where vegetarian = '".$userVegetarian."' and
                     meal_category = 'Breakfast' order by meal_protein
                     desc limit 10) as subquery order by rand() limit
                     1";
53                $resultBreakfast = mysqli_query($db, $sqlBreakfast);
54                while($row = mysqli_fetch_array($resultBreakfast)){
55                    $meals[] = intval($row['meal_id']);
56                }
57
58                $sqlLunch = "select meal_id from (select * from meal
                     where vegetarian = '".$userVegetarian."' and
                     meal_category = 'Lunch' order by meal_protein
                     desc limit 10) as subquery order by rand() limit
                     1";
59                $resultLunch = mysqli_query($db, $sqlLunch);
```

```
60              while($row = mysqli_fetch_array($resultLunch)){
61                  $meals[] = intval($row['meal_id']);
62              }
63
64              $sqlDinner = "select meal_id from (select * from
                   meal where vegetarian = '".$userVegetarian."' and
                   meal_category = 'Dinner' order by meal_protein
                   desc limit 10) as subquery order by rand() limit
                   1";
65              $resultDinner = mysqli_query($db, $sqlDinner);
66              while($row = mysqli_fetch_array($resultDinner)){
67                  $meals[] = intval($row['meal_id']);
68              }
69          }
70          else{
71              $sqlBreakfast = "select meal_id from (select * from
                   meal where meal_category = 'Breakfast' order by
                   meal_protein desc limit 10) as subquery order by
                   rand() limit 1";
72              $resultBreakfast = mysqli_query($db, $sqlBreakfast);
73              while($row = mysqli_fetch_array($resultBreakfast)){
74                  $meals[] = intval($row['meal_id']);
75              }
76
77              $sqlLunch = "select meal_id from (select * from meal
                   where meal_category = 'Lunch' order by
                   meal_protein desc limit 10) as subquery order by
                   rand() limit 1";
78              $resultLunch = mysqli_query($db, $sqlLunch);
79              while($row = mysqli_fetch_array($resultLunch)){
80                  $meals[] = intval($row['meal_id']);
81              }
82
83              $sqlDinner = "select meal_id from (select * from
                   meal where meal_category = 'Dinner' order by
                   meal_protein desc limit 10) as subquery order by
```

```
                rand() limit 1";
84              $resultDinner = mysqli_query($db, $sqlDinner);
85              while($row = mysqli_fetch_array($resultDinner)){
86                  $meals[] = intval($row['meal_id']);
87              }
88          }
89          break;
90
91      case "Be Active":
92          if($userVegetarian == 1){
93              $sqlBreakfast = "select meal_id from meal where
                    vegetarian = '".$userVegetarian."' and
                    meal_category = 'Breakfast' order by rand() limit
                    1";
94              $resultBreakfast = mysqli_query($db, $sqlBreakfast);
95              while($row = mysqli_fetch_array($resultBreakfast)){
96                  $meals[] = intval($row['meal_id']);
97              }
98
99              $sqlLunch = "select meal_id from meal where
                    vegetarian = '".$userVegetarian."' and
                    meal_category = 'Lunch' order by rand() limit 1";
100             $resultLunch = mysqli_query($db, $sqlLunch);
101             while($row = mysqli_fetch_array($resultLunch)){
102                 $meals[] = intval($row['meal_id']);
103             }
104
105             $sqlDinner = "select meal_id from meal where
                    vegetarian = '".$userVegetarian."' and
                    meal_category = 'Dinner' order by rand() limit 1";
106             $resultDinner = mysqli_query($db, $sqlDinner);
107             while($row = mysqli_fetch_array($resultDinner)){
108                 $meals[] = intval($row['meal_id']);
109             }
110         }
111         else{
```

```
112                 $sqlBreakfast = "select meal_id from meal where
                        meal_category = 'Breakfast' order by rand() limit
                        1";
113                 $resultBreakfast = mysqli_query($db, $sqlBreakfast);
114                 while($row = mysqli_fetch_array($resultBreakfast)){
115                     $meals[] = intval($row['meal_id']);
116                 }
117
118                 $sqlLunch = "select meal_id from meal where
                        meal_category = 'Lunch' order by rand() limit 1";
119                 $resultLunch = mysqli_query($db, $sqlLunch);
120                 while($row = mysqli_fetch_array($resultLunch)){
121                     $meals[] = intval($row['meal_id']);
122                 }
123
124                 $sqlDinner = "select meal_id from meal where
                        meal_category = 'Dinner' order by rand() limit 1";
125                 $resultDinner = mysqli_query($db, $sqlDinner);
126                 while($row = mysqli_fetch_array($resultDinner)){
127                     $meals[] = intval($row['meal_id']);
128                 }
129             }
130             break;
131     }
132     return $meals;
133 }
```

Anexa 5: Algoritmul pentru generarea planului de mese

Algoritmul respectiv (ca o funcție în limbajul de programare PHP):

```
1 function buildMealPlan($userId, $userWeight, $userVegetarian, $
    userGoal, $db){
2         $mealDuration = 30;
3         $startDay = new DateTime();
4
```

```
5           for($day=0; $day<$mealDuration; $day++){
6               $currentDate = $startDay->format('Y-m-d');
7               $meals = array();
8               $meals = chooseMeals($userGoal, $userVegetarian, $
                    userWeight, $db);
9
10              $sql1 = "INSERT INTO meal_plan_day(user_id, meal_day)
                    VALUES('".$userId."', '".$currentDate."')";
11              if(mysqli_query($db, $sql1)){
12                  $insertedId = mysqli_insert_id($db);
13                  for($i=0; $i<count($meals); $i++){
14                      $sql2 = "INSERT INTO meal_plan_meals(meal_id,
                            meal_day_id) VALUES('".$meals[$i]."',
                            '".$insertedId."')";
15                      if(mysqli_query($db, $sql2)){
16                      }
17                      else{
18                          return "Failure";
19                      }
20                  }
21
22                  $startDay->modify('+1 day');
23              }
24              else{
25                  return "Failure";
26              }
27          }
28          return "Success";
29      }
```