UNIVERSITATEA POLITEHNICA BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

PROIECT DE DIPLOMĂ

Aplicatie android pentru nutritie si antrenament

Grameni Stere-Alexandru

Coordonator științific:

S.L. dr. Ing. Carmen Odubasteanu

BUCUREŞTI

2018

Contents

Aplicațiile pentru nutriție și antrenament sunt cunoscute pentru pasionații fitness. Având în vedere dorința utilizatorilor de a duce o viață cât mai sănătoasă, dar și de a atinge câteva scopuri personale legate de greutatea corporală și aspectul fizic, aceste aplicații sunt utilizate tot mai mult. O astfel de aplicație este o cale ușoară pentru utilizatori de a-și monitoriza alimentația și de a avea acces la un program de antrenament complex.

Scopul acestui proiect este de a dezvolta o aplicație prietenoasă pentru utilizatori, atât pentru persoanele experimentate în domeniul fitness, cât și pentru persoanele care sunt la primele contacte cu această lume. Aplicația este dezvoltată pentru telefoanele cu sistem de operare android. Folosind această aplicație, utilizatorii pot reține ce au mâncat în fiecare zi, astfel vor afla numărul de calorii, proteine, carbohidrați și grăsimi consumate. Totodată, aplicația îi pune la dispoziție utilizatorului un program de antrenament, în funcție de scopurile acestuia și de nivelul fizic la care se află.

1 INTRODUCERE

1.1. Context

Stilul de viață sedimentar este adoptat de tot mai mulți oameni, aceștia având un loc de muncă bazat pe muncă de birou, nu foarte activ din punct de vedere fizic. Astfel, mulți oameni ajung să se înscrie la o sală de fitness pentru necesarul de activitate. Ritmul de creștere a pieței fitness este de 20-25% anual, iar un român cheltuie aproximativ 11 euro anual, față de 5 euro în anul 2012.[1]

Domeniul fitness este un domeniu care cere disciplină, motivație și dedicare. O persoană interesată de acest domeniu trebuie să utilizeze un antrenament potrivit pentru nivelul lui fizic, scopurile acestuia și adaptat pentru fiecare metabolism. Totodată, acea persoană trebuie să respecte și o dietă strictă, care se rezumă la un conținut calorit, împărțit în cele 3 tipuri de macro-nutrienți: carbohodrati, proteine și grăsimi. Fiecare macro-nutrient trebuie să fie prezent în dietă, într-o anumită cantitate.

Urmărind un astfel de program de antrenament și o dietă, totul se reduce la un joc al numerelor. De aceea, o persoană interesată de fitness trebuie să țină evidența nutriției și al antrenamentului, pentru a putea observa eventuale greșeli și a le corecta, sau pentru a putea observa progresul.

1.2. Problema

Industria fitness este în continuă creștere, dar nu se rezumă doar la nutriție și antrenament. Multe companii de suplimente sportive crează produse folositoare din punct de vedere nutritiv, dar le crează o imagine de suplimente “minune”. Ca o consecință, o mare parte a populației își va crea idea că fără aceste suplimente, nu se poate atinge un obiectiv îndrăzneț. Un alt factor de luat în considerare este publicitatea. Participanții la concursuri de fitness și culturism folosesc substanțe chimice cu efecte anabolice pentru a-și îmbunătăți performanțele sportive sau pentru a-și îmbunătăți aspectul fizic. Acești sportivi sunt văzuți ca modele de începători, dar când nu obțin aceleași rezultate folosind suplimente sau antrenamente minune, aceștia sunt demotivați și tentați să renunțe.

O altă problemă este informarea persoanelor aflate la început. Ele nu au cunoștințele pentru a dezvolta un program de antrenament sau un plan de nutriție, iar în mediul online informația potrivită nu este organizată și ușor de găsit, astfel încât un începător să înțeleagă ce se urmărește prin cifrele la care se rezumă tot programul. Multe informații din mediul online pun accent pe material, și nu pe cantitate. Sunt nenumărate articole despre ce se poate consuma într-o dietă, dar nu și despre cantitatea corectă.

A treia problemă este reprezentată de observarea greșelilor și constanța în rezultate. În momentul în care un rezultat nu este obținut, există o greșeală, dar, de obicei, este greu de găsit.

1.3. Obiective

Obiectivul proiectului este de a crea o aplicație pentru telefoane cu sistem de operare android, prin care utilizatorul este ajutat să își atingă scopul. Aplicația își propune să ofere utilizatorilor instrumentele necesare și de a le arăta mijloacele cele mai importante: nutriția și antrenamentul. Utilizatorul va afla că suplimentele sportive nu sunt decât componente nutritive și nu va cădea în capcana industriei media. Aplicația va prelucra datele de pornire ale utilizatorului și îi va oferi un program de antrenament bine structurat, complex și potrivit pentru obiectivele acestuia. Totodată, se va crea un plan nutrițional împărțit pe macro-nutrienți.

Astfel, persoanele care nu au acces la informație, vor putea realiza ce își propun fără a apela la un antrenor personal sau un nutriționist. Prin folosirea constantă a acestui proiect, clienții vor ajunge să înțeleagă principiile pe care trebuie să se bazeze și vor putea să creeze ei un astfel de program adaptat pentru nevoile și metabolismul lor.

1.4. Soluția propusă

Pentru a atrage cât mai mulți utilizatori în industria fitness, se va dezvolta o aplicație mobilă pentru android, pornind cu versiunea 4.4 – KitKat. În acest moment, 90.1% din telefoanele cu sistem de operare android pot rula pe această versiune, astfel, aplicația poate fi folosită de aproape toți utilizatorii de android.

Fiecare utilizator se va înregistra și autentifica. Crearea unui program de antrenament și al unui plan de nutriție se face în momentul în care utilizatorul va completa datele din profilul acestuia. Pentru fiecare persoană în parte se cer vârstă, înălțimea, greutatea, greutatea dorită și numărul de zile în care se pot efectua antrenamente. Odată cu înaintarea acestui formular, se va asocia un antrenament profilului utilizatorului și se vor stabili limitele superioare pentru conținutul caloric zilnic.

Pentru îndeplinirea părții nutritive, aplicația va pune la dispoziție o funcționalitate de tip notebook, prin intermediul căreia clienții vor putea nota, zilnic, alimentele consumate. Pentru majoritatea alimentelor existențe deja în datele aplicației se cunoaște numărul de calorii provenite din fiecare macro-nutrient. Astfel se simplifică utilizarea acestei funcții, singurul criteriu de căutare ce trebuie introdus fiind numele produsului alimentar și gramajul consumat.

1.5. Rezultatele obținute

În urma implementării acestui proiect, am obținut o aplicație ușor de utilizat, cu o interfață simplă. Folosind un telefon mobil cu conexiune la date mobile, oricine se poate înregistra și folosi aplicația.

Beneficiarii acestui proiect sunt persoanele care nu au suficientă experiență în domeniu, acestea obținând instrucțiuni personalizate, dar și persoanele experimentate, care pot folosi funcția pentru monitorizarea alimentației zilnic.

2 ANALIZA CERINȚELOR

Este bine cunoscut faptul că activitățile fizice și o dietă echilibrată sunt criterii importante pentru a duce un stil de viață sănătos. Cu toate acestea, tehnologiile noi au simplificat foarte mult viața și au creat o zonă de comoditate. Adăugând și deplasarea numai cu autoturismele personale sau mijloacele de transport în comun, un loc de muncă de birou și mult timp petrecut pe scaun în fața calculatoarelor, se ajunge la un stil de viață sedentar. Organizația Mondială a Sănătății estimează că sedentarismul

ocupă locul patru la nivel mondial in clasamentul factorilor care pot produce decese și că anual, trei milioane de oameni își pierd viața din cauza lipsei de mișcare. [2]

Sedentarismul este cu atât mai periculos cu cât este adoptat de la o vârstă fragedă. Lipsa activității sportive duce la mai puține calorii arse și mai multe stocate sub formă de țesut adipos. Astfel copilul va petrece mai mult timp în casă și mai puțin timp jucându-se cu ceilalți copii, în consecință apar și probleme sociale. [3] Cu cât sedentarismul este adoptat de la o vârstă mică, cu atât se va renunța mai greu la el și se va accepta o schimbare din acest punct de vedere.

Statistic, 49% dintre români nu fac niciodată sport și 28% fac activități fizice rar. Procentul este mai mic la nivel european, astfel 39% dintre europeni nu fac sport și 21% rareori.

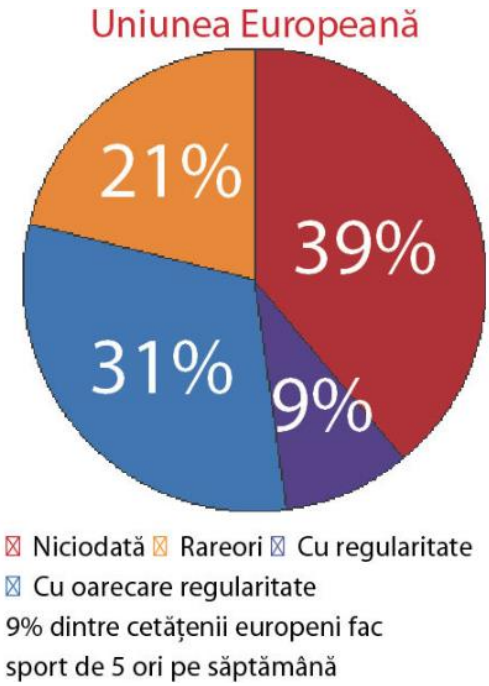

Fig 1[4] – Analiză activitate sportivă pentru persoane europene

Printre efectele sedentarismului se numără riscul crescut de boli cardiovasculare, diabet zaharat și anumite tipuri de cancer. Totuși, cel mai cunoscut efect al sedentarismului este reprezentat de obezitate. [5]

Obezitatea se manifestă prin depozitarea excesivă a țesutului adipos(greutate cu 25% peste greutatea normală). Aproape 30% din populația planetei este supraponderală sau obeză(peste 2.1 miliarde de oameni). Anual, 5% din numărul

deceselor înregistrate la nivel mondial sunt provocate de obezitate. [6] Principalul factor care duce la obezitate este consumul caloric mărit în contrast cu lipsa activităților fizice.

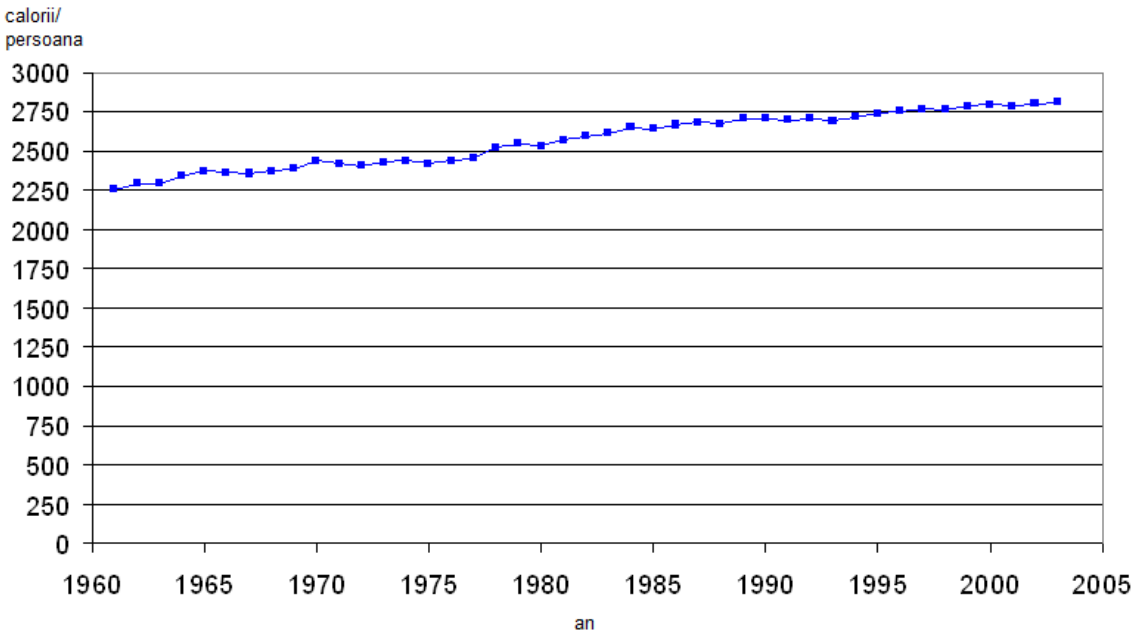

Fig 2[7] – Nivelul caloric pe cap de persoană din 1960 până în 2002

După cum se poate observa și în Fig 2, nivelul caloric mediu consumat de fiecare persoană a crescut cu peste 500 de calorii din 1960 până în prezent. Pe de altă parte, rata obezității în Statele Unite ale Americii a crescut de la 14% până la 31%, din anul 1971 până în anul 2000. În același timp, rata calorică în Statele Unite a crescut cu 335 calorii/zi la femei și 168 calorii/zi la bărbați, majoritatea caloriilor provenind din carbohidrați. După cum se observă în Fig 3, consumul caloric a crescut în fiecare regiune, atingând și valori mai mari de 3600 calorii, spre deosebire de valorile atinse în anul 1961. Între anii 1977-1995, doar în Statele Unite, consumul produselor fast-food a crescut de trei ori.

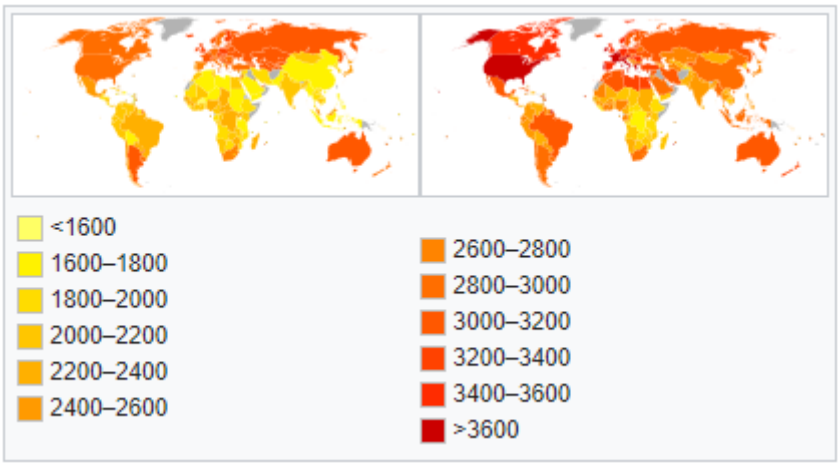

Fig 3[7] – Harta consumului de calorii/persoană în 1961(stânga) și 2002(dreapta)

Legătură dintre obezitate și alimentele de tip fast-food este îngrijorătoare, nevoia oamenilor de a consuma mese cât mai mari într-un timp cât mai scurt ducând la alegerea acestor variante. Produsele fast-food sunt alimente ce conțin cantități mari de zaharuri, sare, grăsimi și carne, astfel ele pot fi considerate bombe calorice. Din comoditate sau lipsă de timp, societatea preferă tot mai mult aceste produse, drept consecință, piața fast-food a crescut considerabil. În anul 2012, în Statele Unite ale Americii s-a înregistrat suma de 160 miliarde de dolari cheltuite de cetățeni pe alimente fast-food, în timp ce în 1970 se cheltuia doar 6 miliarde de dolari.[8]

Consumul excesiv al produselor fast-food pot avea multe efecte nedorite asupra organismului uman. Un meniu fast-food obișnuit poate conține toate caloriile necesare unei zile întregi, astfel, kilogramele în plus vor fi o consecință. Alte repercusiuni pot fi boli cardio-vasculare, diabet zaharat și infarct. Conform unui studiu realizat în anul 2008, de către Universitatea McMaster din Canada, la nivel mondial, o treime din infarcte sunt efecte ale consumului de fast-food în cantități mult prea mari.[9]

Nerespectarea principiilor nutritive poate duce și la deficitul de proteine. Proteinele sunt macro-nutrienți care, odată consumați, sunt descompuși în aminoacizi. Aminoacizii intră în sânge și au un rol important în construirea și refacerea țesutului muscular. Zilnic, o persoană adultă are nevoie de 130-140g de proteine, iar un copil de 80-100g. Totuși, aceste cifre sunt strâns legate de vârstă, masă musculară, nivelul de activitate zilnică, greutate corporală, dar și de metabolism.

Importanța proteinelor în menținerea și creșterea masei musculare duce la importanța consumului de proteine, mai ales pentru copii. Consumul redus de proteine poate crea dezechilibre în sistemul muscular, osos sau chiar hormonal, și prin urmare, poate duce chiar și la stoparea creșterii copilului. În anul 2013, aproximativ 160 de milioane de copii s-au luptat cu probleme legate de creștere.[10]

Prezența în sistemul muscular este doar o proprietate a proteinelor. Acestea sunt importante și pentru sistemul osos. Un consum redus de proteine poate slăbi sistemul osos și poate crește riscul fracturilor și fisurilor. În plus, proteinele sunt folosite și în sistemul imunitar. Producerea de globule albe este strâns legată de cantitatea de proteine care ajunge în sânge, iar aceste globule albe sunt folosite pentru a depista și elimina bacterii și viruși. Alte probleme ce pot apărea în urma unui consum scăzut de proteine sunt încetinirea metabolismului, nivelul scăzut de energie și oboseală, risc crescut de diabet și dificultăți în creșterea în masă musculară și scăderea în greutate.

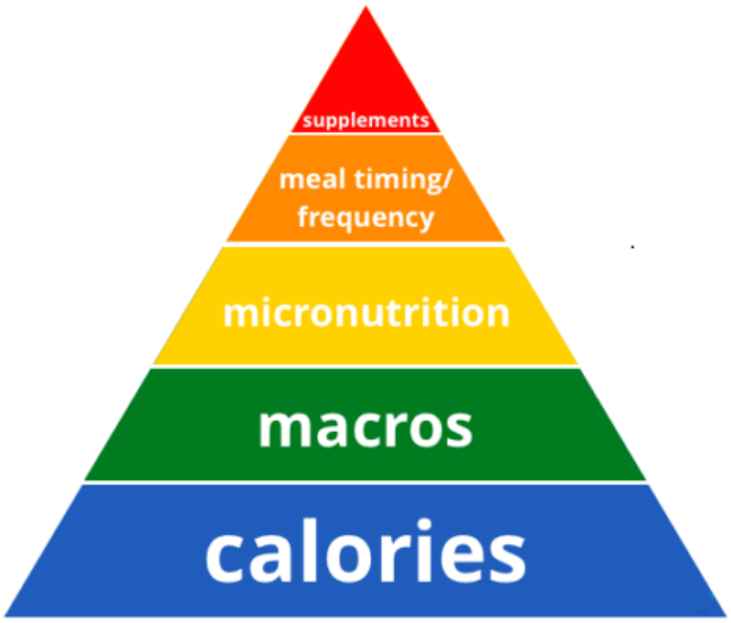

Fig 4[11] – Piramida priorităților în nutriție

Așa cum se poate observa în Fig 4, la bazele piramidei nutriționale stă nivelul caloriilor consumate, iar pe al doilea nivel este proveniența acelor calorii din cele 3 tipuri de macro-nutrienți: proteine, carbohidrați și grăsimi. Pe lângă proteine, carbohidrații și grăsimile au, de asemenea, un rol important în sănătatea organismului. Carbohidrații sunt principala sursă de energie a organismului și sunt împărțiți în două

categorii: simpli și complexi. Diferența între cele două tipuri este de timpul petrecut de organism pentru le absorbi. Carbohidrații simpli sunt tot carbohidrați complexi, dar procesați. Acestora li s-au îndepărtat fibrele și sunt absorbiți mai repede de organism.

Analizând toate argumentele prezentate mai sus, consider că este nevoie de o aplicație care să pună utilizatorul în contact cu lumea fitness, astfel încât acesta să își poată îndeplini scopurile. Industria fitness este într-o expansiune continuă, prin ajutorul cercetării se decopera în continuu cunoștințe noi legate de anatomia unama și noi procedee și metode de antrenament și nutriție, prin care se poate transforma aspectul fizic. O astfel de aplicație ar fi foarte utilă pentru utilizatorii care, din diverse motive, nu doresc să apeleze la serviciile unui antrenor personal sau la serviciile unui nutriționist. Utilizatorii ar avea acces la o bază de date cu o diversitate mare de alimente, ar putea să își creeze chiar ei un plan nutrițional, alegând alimentele preferate de ei, și ar avea acces la o serie de antrenamente bine construite.

3 STUDIU DE PIAȚĂ

3.1. Platforme existente

Piața aplicațiilor mobile pentru industria fitness a crescut semnificativ odată cu apariția smartphone-urilor și a sistemelor de operare IOS și android. În momentul de față, majoritatea smartphoneurilor vin cu aplicații fitness preinstalate. De exemplu, android versiunea 5.0 Lollipop vine cu aplicația Samsung Health[12], aplicație care monitorizează activitățile zilnice, pulsul și nivelul de stres. Aplicația poate fi folosită pentru a măsura distanțele parcurse în timpul unui antrenament cardio și va oferi, ca răspuns, detalii despre antrenament, cum ar fi numărul de calorii arse, viteza medie și viteza maximă. Principalul avantaj al acestei aplicații este tocmai numărul mare de puncte de control pentru funcțiile organismului uman. În schimb, un dezavantaj este interfața greu de folosit. Este greu de folosit aplicația la potențial maxim încă din momentul instalării aplicației, utilizatorul trebuie să se acomodeze întâi cu aceasta.

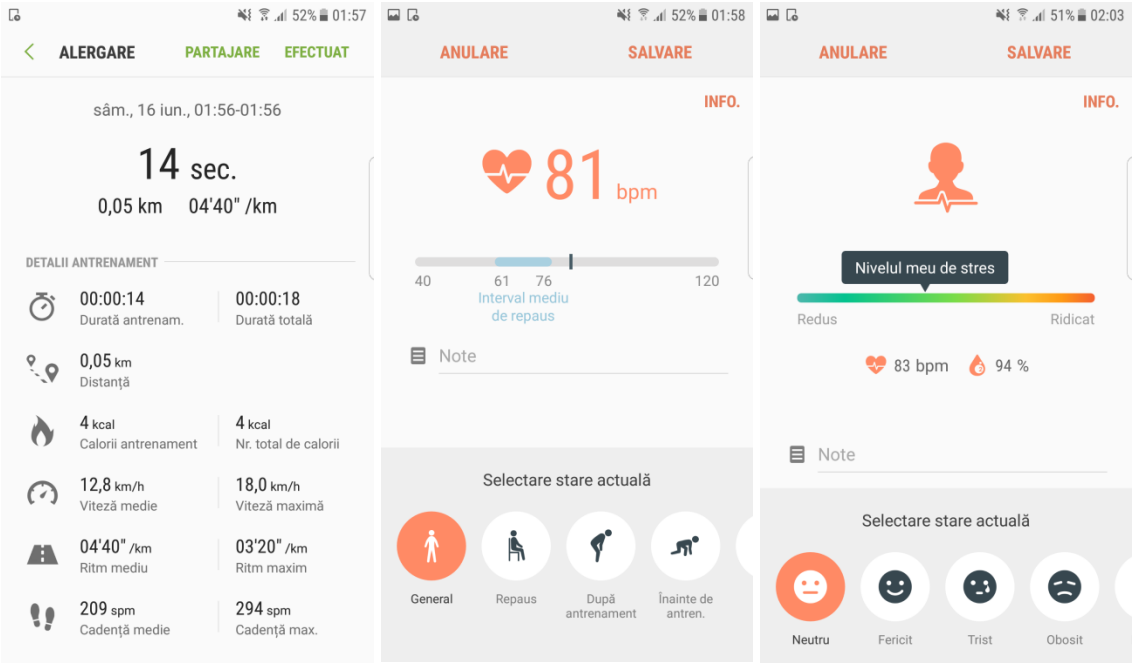

Fig 5 – Aplicație Samsung Health[12], funcții pentru puls, stres și antrenament

După doar o căutare pe Google Play[13], putem observa peste 500 de aplicații fitness, de la aplicații pentru alergat, antrenament pentru abdomen și antrenament pentru întregul corp, până la aplicații pentru monitorizare și nutriție. Putem găsi aplicații precum Six Pack în 30 Days[14], aplicație care îi oferă utilizatorului un program de antrenament pentru abdomen, în funcție de punctul de plecare al utilizatorului. Avantajul acestor aplicații este antrenamentul bine structurat pentru grupa musculară țintită, de exemplu, în Six Pack în 30 Days[14], antrenamentul este construit pentru a dezvolta zona abdominală. Dezavantajul acestor aplicații este faptul ca nu oferă suport pe parte de nutriție și, în același timp, programul de antrenament țintește o singură grupă musculară, celelalte fiind neglijate.

De departe, cea mai cunoscută și folosită aplicație fitness completă este MyFitnessPal[15]. Aceasta este folosită pentru a contoriza consumul și arderea caloriilor, precum și contorizarea antrenamentelor efectuate în sală sau în aer liber. Este o aplicație cu o foarte mare bază de date ce conține alimente, cu totalul caloric și numărul de carbohidrați, grăsimi și proteine. Aplicația este descărcată de peste 50 de milioane de utilizatori și este evaluată cu o notă de 4,6/5 din peste 1,8 milioane de recenzii. Aplicația conține aproximativ 6 milioane de alimente, dintre care peste 4 milioane pot fi introduse prin scanarea codului de bare. De asemenea, aceasta conține

produse din meniurile unor restaurante, se pot introduce propriile rețete în baza de date asociată contului utilizatorului.

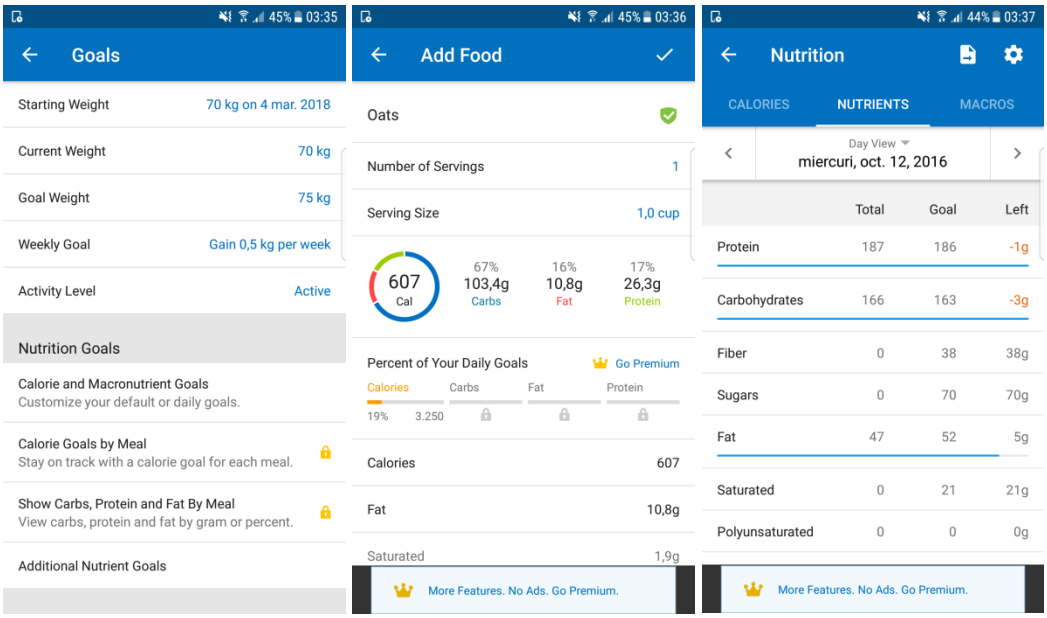

Fig 6 – Aplicație MyFitnessPall[15] – funcții de setare a scopului, adăugare de alimente și nivelul caloric zilnic

După cum putem observa în Fig 6, utilizatorul trebuie să se înregistreze, apoi trebuie să își completeze profilul, introducând câteva date despre condiția lui fizică actuală. Apoi utilizatorul trebuie să își seteza un scop și durata în care dorește să realizeze acel scop și aplicația îi va returna o soluție pentru numărul de calorii și împărțirea acestora în macro-nutrienți. Apoi zilnic, utilizatorul va adăuga în jurnal alimentele consumate, iar aplicația va întoarce nivelul caloric al acestora, precum și totalul caloric atins în acea zi, sortat după numărul de proteine, carbohidrați și grăsimi. De asemenea, în aplicație se pot introduce și antrenamentele realizate și se va estima numărul de calorii arse în acele antrenamente. La fel ca în cazul Samsung Health[12], MyFitnessPal[15] are o funcționalitate de monitorizare a unui antrenament de alergare. Interfața este foarte intuitivă și ușor de înțeles încă de la prima folosire, baza de date este foarte mare și sunt șanse foarte mici ca utilizatorul să nu găsească produsul căutat. În schimb, tocmai numărul mare de produse poate duce la derutarea utilizatorului. Aplicația este concentrată foarte mult pe partea de nutriție și nu conține suport pentru programe de antrenament.

Un alt tip de aplicație foarte popular pe piață este cel de antrenor personal de la distanță. Un exemplu este Kris Gethin Muscle Building 12-Week Trainer[16], scopul fiind acumularea unei cantități cât mai mare de masă musculară printr-un program de 12 săptămâni. Această aplicație nu oferă un program de antrenament personalizat, ci unul care a fost testat pe un grup mare de oameni cu metabolisme diferite și a funcționat. Accentul se pune însă pe efectuarea corectă a exercițiilor și pe urmărirea instrucțiunilor, de aceea este disponibil și un videoclip pentru fiecare zi de antrenament. Pentru partea nutritivă, este disponibil un calculator înglobat în aplicație, care va informa utilizatorul cu privire la consumul caloric necesar.

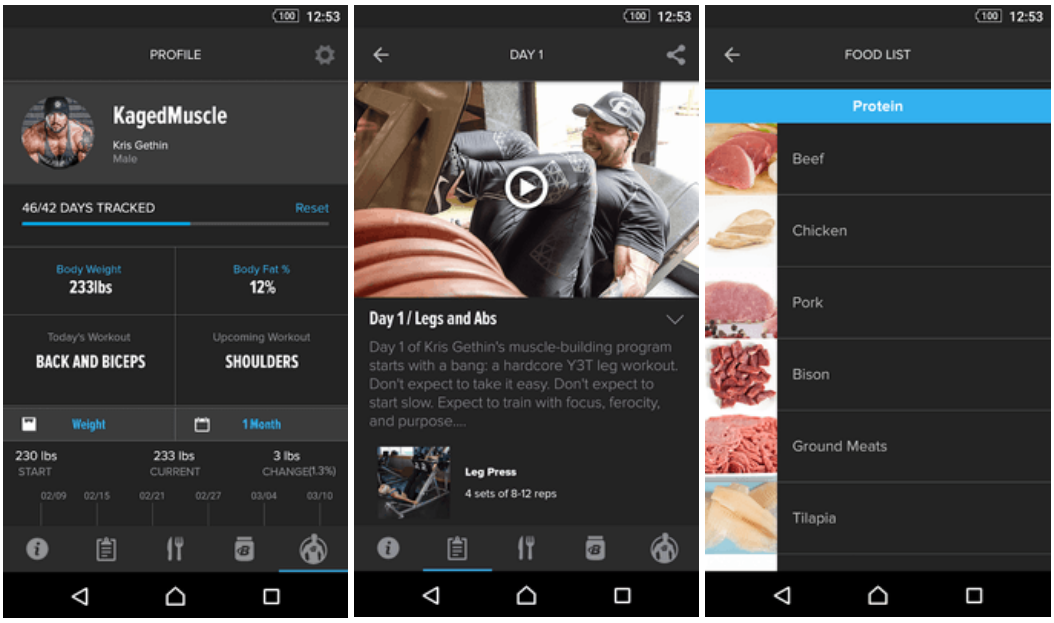

Fig 7[17] – Aplicație Kris Gethin Muscle Builder 12 Week Trainer[16]

Fig 7 evidențiază procesul prin care un utilizator își contorizează progresul, acesta primind notificări zilnic cu privire la antrenamentul pe care urmează să îl efectueze. De asemenea, utilizatorul primește o listă de alimente pe care le poate consuma, este lăsat să își construiască el dieta cu acele alimente, dar în limitele rezultatului primit de la funcția de calculator, menționată mai sus. Aplicația conține doar un program pentru 12 săptămâni, iar după terminarea acestui program, se poate reîncepe de la zero, urmând același antrenament ca înainte, cu noi date nutriționale.

3.2. Tehnologii folosite

Implementarea aplicației este împărțită în două niveluri: front-end și back-end. Pentru front-end am folosit sistemul de operare Android, Java, XML și Gradle, iar pentru back-end am folosit Maven, Java, Framework-ul Spring și XML. Pentru baze de date am folosit sistemul de gestiune PostgreSQL, iar pentru sincronizarea tabelelor din baza de date cu entitățile din back-end am folosit Hibernate. Legătura dintre back-end și front-end este făcută prin requesturi HTTP, datele transmise prin request și response fiind făcute date în format JSON.

3.2.1. XML

XML este prescurtarea de la extensible markup language și este un limbaj de marcare. Un alt limbaj de marcare este HTML, dar, spre deosebire de acesta, XML nu are etichete deja definite. Un document XML este format din etichete, iar acestea formează o structură arborescentă. În această structură, o etichetă deschisă trebuie închisă, dar numai după ce toate etichetele deschise în interiorul aceștia au fost închise.

```
<!--Acesta este un comentariu -->
<root value="atribut">
    <tag1>
        <tag2> A </tag2>
        <tag3> B </tag3>
    </tag1>
</root>
```

Fig 8 – Exemplu XML

După cum se vede și în Fig 8, eticheta root este închisă doar după ce se închide eticheta tag1, iar tag1 este închisă după tag2 și tag3. Elementele XML pot avea atribute, cu valoarea între ghilimele. Din prima linie din Fig 8 se poate observa și modul în care introducem un comentariu în XML. XML este cel mai folosit limbaj de marcare pentru fișierele de configurare. Am folosit XML pentru fișierul de configurare Maven, pentru fișierul Manifest din proiectul Android și pentru a compune ecranele din proiectul Android.

3.2.2. Java

Cel mai folosit limbaj de programare din acest proiect este Java. Am folosit acest limbaj atât în front-end, cât și în back-end. Java este un limbaj orientat pe obiecte și este inspirat din limbajul C.

```
public class Hello
{
    public static void main(String[] args)
    {
        System.out.println("Hello Java");
    }
}
```

Fig 9[18] – Java Hello Java

În Fig 9 este scris codul pentru a afișa textul „Hello Java”. Un obiect în Java este o entitate(instanță) a unei clase, iar o clasă este o colecție de obiecte cu aceleași atribute. Orice program java are o metodă main, din care pornește rularea. O clasă Java este compusă din atribute și metode și trebuie să conțină un constructor prin care se poate instanția un obiect. O clasă poate extinde o altă clasă, astfel, noua clasă va moșteni toate atributele și metodele clasei părinte. În Java, orice clasă extinde, implicit, clasa Object. În caz că nu este specificat vreun constructor pentru o clasă, pentru a instanția un obiect se va folosi un constructor implicit, fără parametrii. Acest constructor este generat automat. Putem crea și clase abstracte. Acestea sunt clase foarte generale, ce nu au nevoie de atribute sau de implementări pentru metode. O interfață în Java este un caz particular de clasă abstractă, prin care se elimină moștenirea multiplă. Sunt totuși, câteva diferențe între o interfață și o clasă abstractă. O clasă poate implementa mai multe interfețe, dar nu poate moșteni mai multe clase abstracte și, o interfață nu poate conține date, în timp ce o clasă abstractă poate. Adresa unui obiect Java este reținută în referința obiectului și nu în pointer, cum este cazul în C. Pe nivelul de back-end, există multe alternative pentru Java, cum ar fi C++. Am ales să folosesc Java pentru că este un limbaj foarte popular, complex și bine documentat. Pe partea front-end, Java este limbajul implicit folosit pentru dezvoltarea de aplicații Android, nu există alternativă pentru acest nivel.

```java
public class Exemplu2  extends Exemplu implements Serializable{
    //atribute
    private int number;
    private String text;

    //constructor implicit
    public Exemplu2() {
    }

    public Exemplu2(int number, String text) {
        this.number = number;
        this.text = text;
    }

    //metode
    public int getNumber() {
        return number;
    }
    public void setNumber(int number) {
        this.number = number;
    }
    public String getText() {
        return text;
    }
    public void setText(String text) {
        this.text = text;
    }
}
```

Fig 10 – Java

În Fig 10 este un cod Java, prin care am exemplificat folosirea de atribute, metode și constructori. Clasa Exemplu2 extinde clasa Exemplu și îi preia toate metodele și atributele, și implementează interfața Serializable.

3.2.3. Gradle

Gradle este sistemul folosit de android pentru importarea dependențelor necesare proiectului. Folosind Gradle, android pune build-ul pentru aplicație. Pentru sistemul de operare Android, Gradle este sistemul implicit folosit, nu există o alternativă.

3.2.4. Android

Android este un sistem de operare pentru dispozitivele mobile. Acesta folosește un fișier XML denumit Manifest pentru setarea permisiunilor aplicației, pentru definirea activity-urilor și pentru setarea informațiilor esențiale, cum ar fi numele aplicației și

versiunea. Un activity este interacțiunea utilizatorului cu aplicația, de exemplu, un ecran. O aplicație android poate avea mai multe activity-uri. Similar cu metora main din Java, un program Android pornește de la metoda onCreate() dintr-un activity. Pentru fiecare Activity se poate crea și un fișier de layout, scris în XML, unde sunt definite toate componentele ce vor face parte din acel ecran.

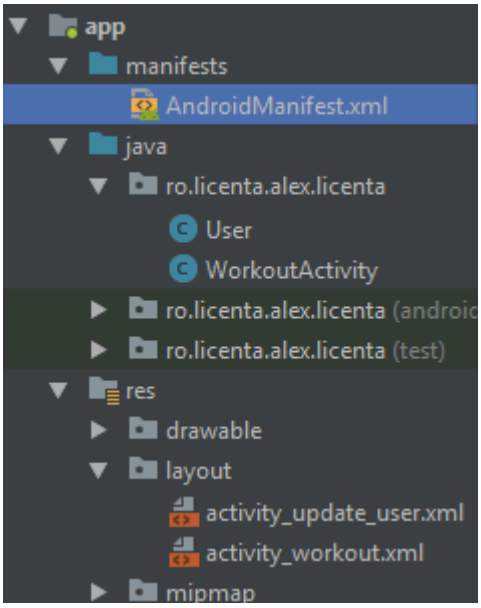

Fig 11 – Proiect Android

În Fig 11 este structura unui proiect Android. În rădăcina proiectului se află pachetele pentru manifest, cod java și resurse. În sursele Java putem introduce codul aplicației, precum și cod pentru teste unitare. În pachetul pentru resurse sunt introduse imaginile proiectului, cum ar fi Icon-ul pentru lansarea aplicației, sunt introduse fișierele XML caracteristice activity-urilor și sunt setate constantele aplicației, după cum este exemplificat în Fig 12.

```
<resources>
    <string name="app_name">licenta</string>
    <string name="username_label">Username</string>
    <string name="password_label">Password</string>
    <string name="new_password_label">New Password</string>
</resources>
```

Fig 12 – Android Constante

O alternativă pentru Android este sistemul IOS, însă am ales Android pentru că numărul dispozitivelor ce rulează Android este mai mare decât numărul celor ce rulează IOS.

3.2.5. Maven

Maven este folosit pentru administrarea unui proiect software. Astfel, folosind Maven, putem pune build-uri, putem importa dependențe în proiect, putem crea documentația unui proiect. Pentru toate acestea, Maven folosește un fișier XML, denumit POM(Project Object Model).

```
<project xmlns = "http://maven.apache.org/POM/4.0.0"
   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation = "http://maven.apache.org/POM/4.0.0
   http://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>

   <groupId>com.companyname.project-group</groupId>
   <artifactId>project</artifactId>
   <version>1.0</version>
</project>
```

Fig 13[19] – Maven Pom

În figura de mai sus este reprezentat un fișier pom.xml din maven. GroupId este id-ul grupului proiectului. În general, acesta este unic într-o organizație. ArtifactId este id-ul proiectului, iar version este versiunea proiectului. În acest mod se pot adăuga și dependențe, folosind eticheta <dependency> și prin definirea celor 3, groupId, artifactId și version.

3.2.6. Java Spring Framework

Pentru serverul aplicației am utilizat framework-ul Spring din Java. Acesta este cunoscut pentru introducerea noțiunii de Dependency Injection. Prin Dependency Injection, Spring reușește să creeze o legătură între clase, păstrându-le totuși cât de poate de independente unele de altele. Acest proces este realizat folosind Bean-uri. Un Bean în Spring este un obiect definit în contextul aplicației și instanțiat și, prin Dependency Injection, este injectat în secvența de cod dorită. Implicit, un Bean instanțiat este folosit oriunde este cerut un astfel de Bean, deoarece scope-ul lui este implicit singleton. Dacă dorim să folosim mereu o instanță nouă a acelui Bean, trebuie să îi schimbăm scope-ul din singleton în prototype. Există două metode prin care putem injecta un Bean, prima este prin constructor, a două este prin setter.

```java
public class ConstrInject {

    private final Day day;
    private User user;

    @Autowired
    public ConstrInject(Day day) {
        this.day = day;
    }

    @Autowired
    public void setUser(User user) {
        this.user = user;
    }
}
```

Fig 14 – Dependency Injection

În Fig 14 este exemplificat Dependency Injection, cu ajutorul injectării prin constructor sau prin setter. Am utilizat Spring pentru că este cel mai folosit framework în rândul serverelor pentru aplicații web și datorită implementărilor pentru serviciul REST.

3.2.7. Baza de date

În ceea ce privește utilizarea bazelor de date, am folosit PostgreSQL. PostgreSQL este un sistem open-source pentru gestionarea bazelor de date după standardele SQL. Acesta este setat ca server pe un port. Pentru a manipula instanțele motorului PostgreSQL am folosit platforma pgAdmin deoarece interfața este simplistă, ușor de înțeles și de folosit. Putem considera PostgreSQL ca fiind serverul și pgAdmin clientul. Toate modurile de lucru cu baza de date pot fi realizate din platforma pgAdmin, astfel testarea serverului a fost realistata cu ușurință.

PostgreSQL poate fi înlocuit cu orice server destinat gestionării bazelor de date, însă am ales să lucrez cu el pentru acest proiect deoarece Maven asigură o conectare ușoară la acesta, doar prin dependențele specificate fișierul pom.xml, ca în Fig 15.

```xml
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>9.3-1102-jdbc41</version>
</dependency>
```

Fig 15 – Dependente PostgreSQL

3.2.8 Hibernate

Pentru a mapa entitățile din baza de date cu obiectele instantiate în Java am folosit Hibernate. Hibernate este un ORM(Object-Relațional Mapping) ce mapează clasele Java pe tabelele din baza de date și mapează tipurile de date din Java pe tipurile de date din SQL. Folosind Java Spring, există două metode prin care putem configura Hibernate pentru mapare. Prima metodă este folosind un fișier XML pentru conectarea la baza de date, specificând URL-ul, platforma, numele de utilizator și parola pentru baza de date. După acest pas, tot într-un fișier XML, specificăm fiecare clasă din Java pe care dorim să o mapăm cu tabela din bază. A doua metodă este să folosim adnotări.

```
@Entity
@Table(name = "food")
public class Food implements Serializable{

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENC
    @SequenceGenerator(name = "sequenceGenerator")
    private Integer id;

    @Size(max = 50)
    @Column(name = "name")
    private String name;

    @Column(name = "calories")
    private Float calories;

    @Column(name = "proteins")
    private Float proteins;

    @Column(name = "carbohydrates")
    private Float carbohydrates;

    @Column(name = "fats")
    private Float fats;
```

Fig 16 – Mapare Hibernate

În Fig 16 putem vedea modul de mapare prin adnotări. Folosind adnotările @Entity și @Table specificăm pe ce tabelă să mapăm, apoi, prin @Column, atribuim fiecărei variabile din clasă câte o coloană din tabela respectivă. Adnotarea @Id este folosită pentru a mapa cheia primară a tabelei, iar @GeneratedValue și @SequenceGenerator au scopul de a crea un Id unic la fiecare adăugare în bază. Am

ales să folosesc Hibernate pentru că, în combinație cu Maven, este probabil cea mai bună alegere pentru o aplicație Java. Pentru a utiliza Hibernate într-o aplicație Java Spring, este suficient să introducem dependențele corecte în fișierul pom.xml, similar cu modul în care am integrat PostgreSQL în aplicație.

4 SOLUȚIA PROPUSĂ

4.1. Descrierea soluției

Soluția propusă pentru rezolvarea problemelor se prezintă sub formă unei aplicații android. Aceasta conține patru cazuri principale de utilizare, care sunt bine conectate între ele, dar ușor de înțeles și de folosit, așa cum vom prezența în diagrama de mai jos.

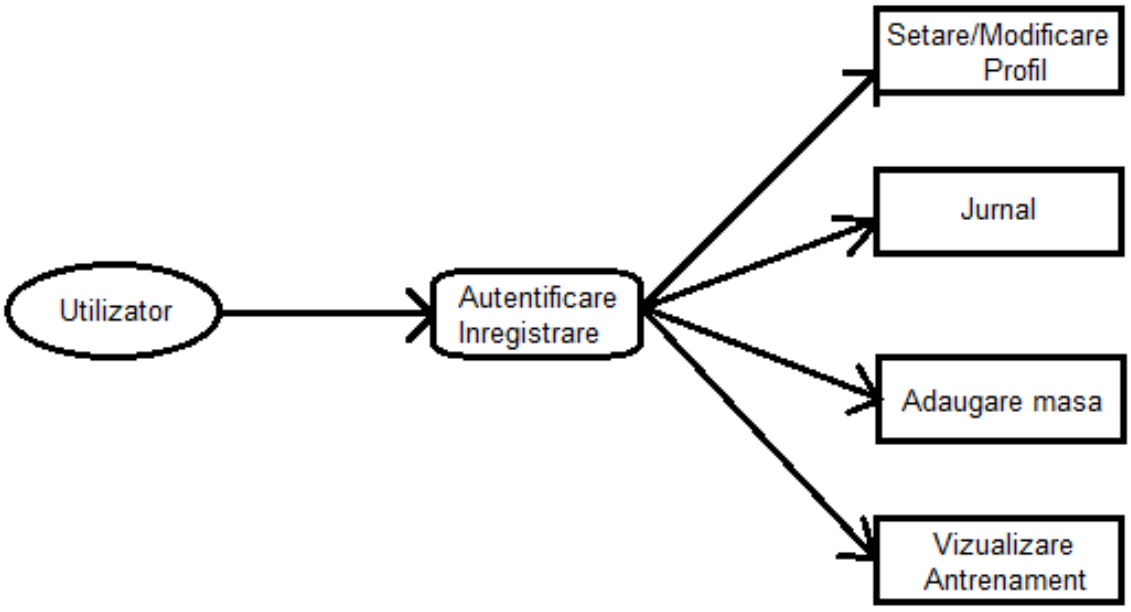

Fig 17 – Diagrama cazurilor de utilizare

Utilizatorul are câteva date personale care sunt valabile doar pentru el, astfel, el trebuie să se înregistreze, apoi să se autentifice de fiecare dată când va folosi aplicația.

Așa cum se vede și în Fig 18, pentru înregistrare, utilizatorul alege un nume, o parolă și setează și un cod, pentru cazul în care uită parola. În caz că uită parola, utilizatorul trebuie să introducă numele utilizator, codul și noua parolă, apoi este întors în pagina de autentificare.

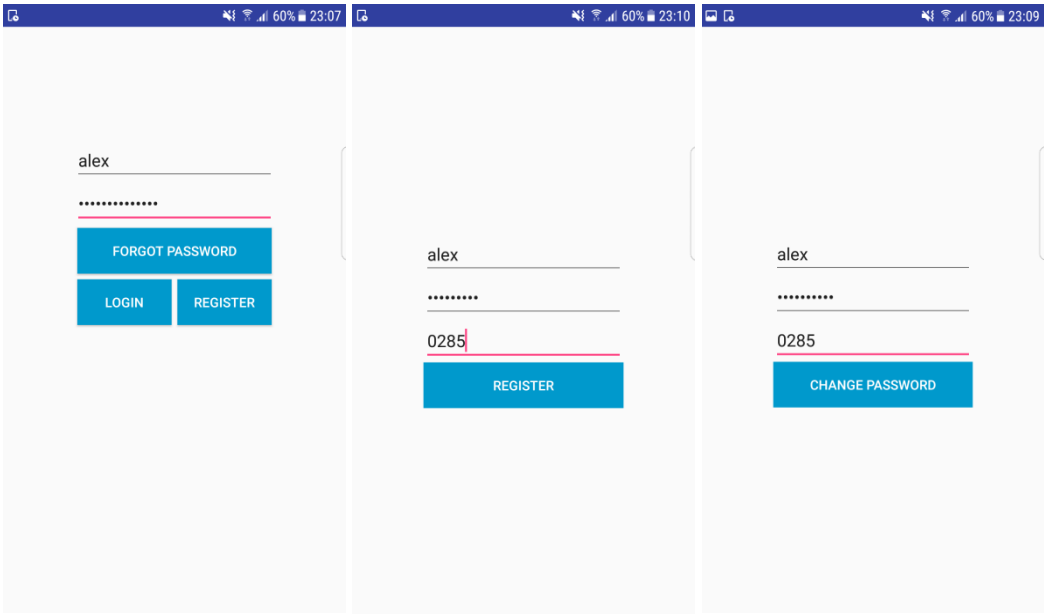

Fig 18 – Autentificare, Înregistrare, Schimbare parolă

După autentificare, utilizatorul este trimis în ecranul principal al aplicației, un meniu prin care poate naviga prin cele 4 funcționalități principale. Acestea sunt completarea și modificarea profilului, vizualizarea programului de antrenament împărțit pe zilele săptămânii, căutarea alimentelor în baza de date și introducerea acestora în jurnal și căutarea zilei dorite în jurnalul de nutriție. Cea mai importantă pe care utilizatorul trebuie să o completeze prima dată este setarea datelor din profilul utilizatorului. Acesta trebuie să completeze date privind vârsta, înălțimea în centimetri, greutatea corporală în kilograme, greutatea corporală la care dorește să ajungă, măsurată tot în kilograme și numărul de zile pe săptămână în care se poate antrena. Aceste date pot fi modificare oricând pe parcursul programului, chiar este indicat, pentru un progres cât mai precis.

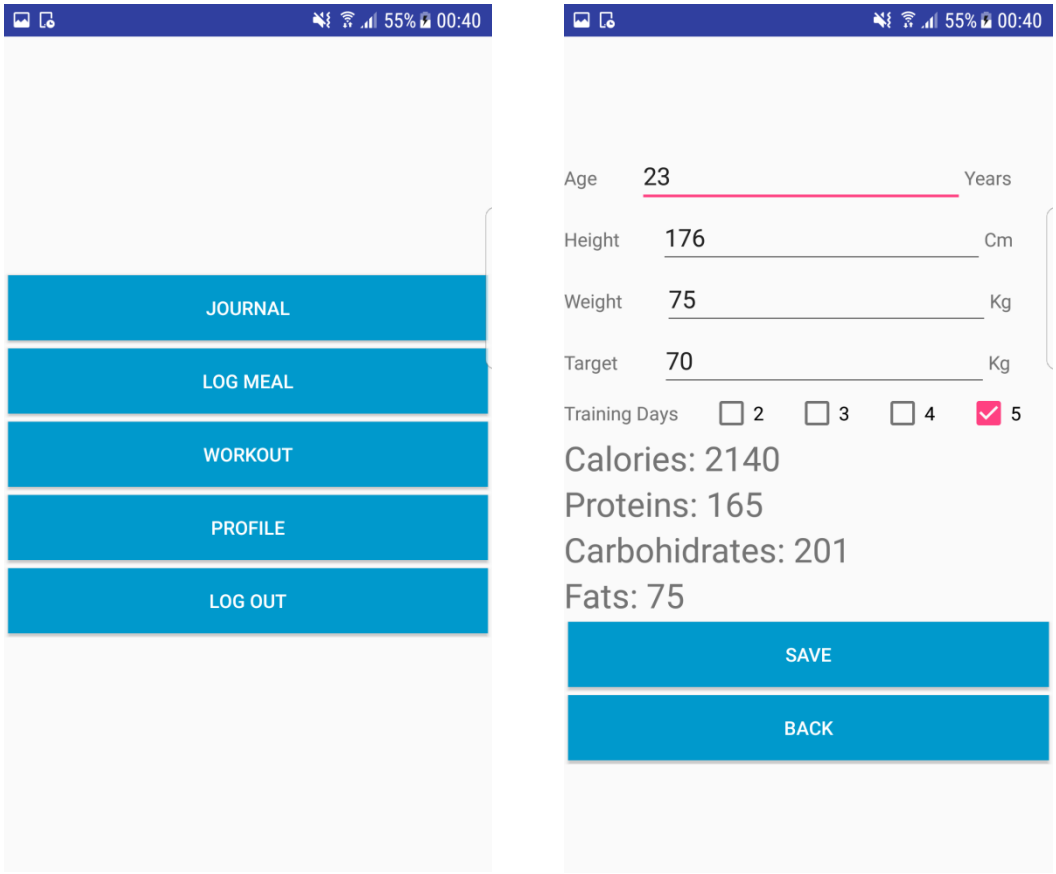

Fig 19 – Meniu și Profil

Prima dată când utilizatorul completează datele din profil, se crează nu program de antrenament pentru acesta și un plan nutrițional. De fiecare dată când utilizatorul își actualizează profilul, se actualizează dieta acestuia și programul de antrenament. Antrenamentul este strâns legat de numărul de zile de antrenament și dorința utilizatorului de a pierde grăsime sau de a crește în masă musculară. Pentru partea nutritivă, aplicația folosește 4 formule matematice pentru a calcula numărul de calorii, proteine, carbohidrați și grăsimi.

$$\begin{cases} calories = weight * 2.2 * a + b \\ proteins = weight * 2.2 \\ fats = weight \\ carbohidrates = (calories - proteins * 4 - fats * 9)/4 \end{cases} \quad (1)$$

În ecuația (1), variabila „a” depinde de numărul de antrenamente pe care le face utilizatorul in fiecare săptămână. Pentru 5 zile, variabila „a” este egală cu 16, pentru 4 zile este egală cu 15, pentru 3 zile este 14 si pentru 2 zile este egala cu 13. Aceste valori

măsoară nivelul de activitate, astfel că în funcție de numărul de antrenamente, utilizatorul poate avea nevoie de mai multe sau mai puține calorii. Variabila „b” depinde de scopul utilizatorului. Dacă acesta dorește să câștige câteva kilograme de masă musculară, variabila „b” este egală cu 500. Dacă acesta dorește să piardă câteva kilograme, „b” va fi -500, iar dacă dorește să rămână la aceeași greutate corporală, „b” este egal cu 0. Prin numărul 500 se compune un surplus sau un deficit caloric. În medie, 3500 de calorii sunt echivalente cu 500 de grame de greutate corporală, astfel că printr-un surplus sau un deficit de 500 de calorii pe zi, se pot câștiga sau pierde 500 de grame pe o perioadă de o săptămână. Acesta este progresul ideal pe care îl poate avea un utilizator, dacă este ținut constant. Constanta 2.2 este folosită pentru a trece greutatea corporală din kilograme în pounds, iar constantele 4 și 9 sunt folosite pentru a transforma macro-nutrienții în calorii, sau pentru a transforma caloriile în macro-nutrienți.

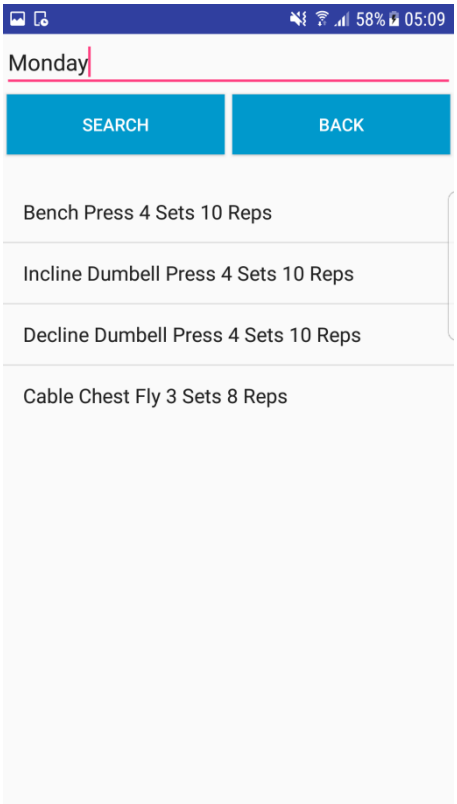

Fig 20 – Program de Antrenament

În Fig 20 găsim modalitatea prin care utilizatorul poate accesa programul de antrenament. Acesta alege să între în WORKOUT din meniu(Fig 19) și va găsi un câmp ce trebuie completat cu ziua pentru care vrea să vizualizeze exercițiile propuse și va apăsa pe butonul SEARCH. Rezultatul primit va fi o listă de exerciții pe care utilizatorul trebuie să le execute. În listă se specifică și numărul de serii pentru fiecare exercițiu, precum și numărul repetărilor pentru fiecare serie. Greutatea cu care se execută exercițiul rămâne la alegerea utilizatorului, deoarece aceasta variază în funcție de nivelul de pregătire al acestuia. Antrenamentul variază, în primul rând, în funcție de numărul de antrenamente săptămânale. De exemplu, dacă utilizatorul se antrenează de cinci ori pe săptămâna, antrenamentul zilnic va fi mai scurt, va lucra mai puține grupe musculare decât într-un antrenament pentru 2 zile pe săptămână. Similar și pentru obiectivul dorit de utilizator. Dacă acesta dorește să piardă în greutate, antrenamentul va fi mai scurt decât un antrenament pentru masă musculară, deoarece, pentru masă musculară, și numărul de calorii pe care le poate consuma utilizatorul este mai mare.

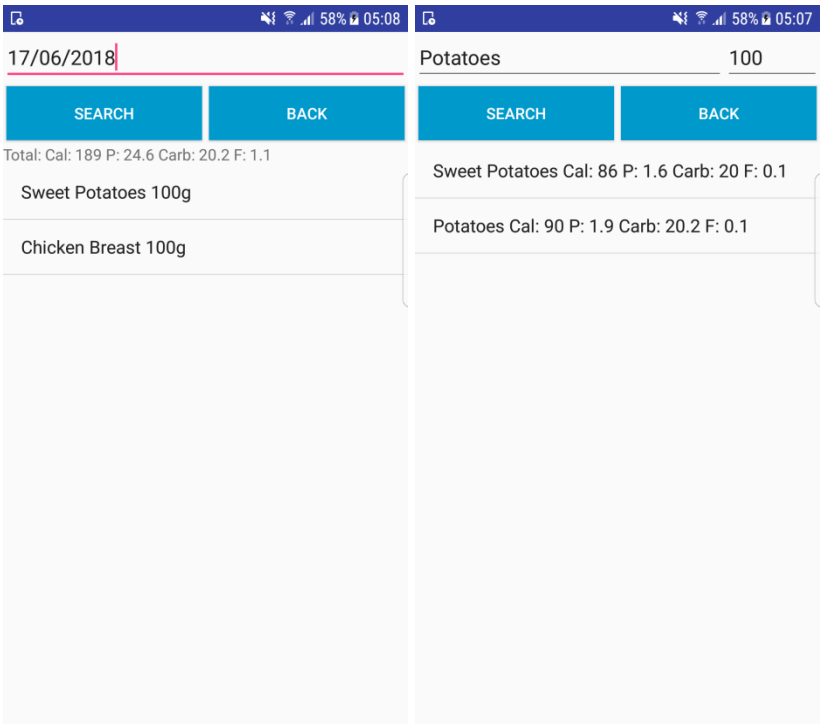

Fig 21 – Jurnal și Adăugare de alimente

O altă funcționalitate implementată în aplicație este extragerea alimentelor din baza de date și introducerea în jurnalul utilizatorului. Această funcționalitate poate fi apelată din meniu(Fig 19) apăsând pe butonul LOG MEAL. După cum apare și în Fig 21, în acest ecran trebuie să completăm denumirea alimentului, sau o denumire cât mai apropiată, și trebuie să completăm și cantitatea consumată, în grame. După căutare, apare o listă de alimente ce au denumirea apropiată cu cea căutată. Pentru fiecare aliment, vor apărea și numărul de calorii, proteine, carbohidrați și grăsimi, pentru gramajul introdus. La apăsarea unui element din lista de alimente, aceasta va intra în jurnalul utilizatorului, pentru a fi monitorizară. Totuși, utilizatorul trebuie să consume doar alimentele care se încadrează în limitele impuse și să nu depășească nivelul caloric zilnic.

Ultima parte a acestei aplicații este jurnalul. În acest ecran, utilizatorul va revedea tot ce a consumat în cursul unei zile. Pentru această funcționalitate, utilizatorul trebuie să completeze câmpul pentru ziua pe care vrea să o revadă și, în urma căutării, se va întoarce o listă cu toate alimentele consumate în acea zi. Alimentele apar cu numele acestora și cu cantitatea consumată. În același timp, după căutare, deasupra listei, se va afișa și un mesaj de informare, privind numărul de calorii consumare în acea zi, numărul de proteine, de carbohidrați și de grăsimi. Astfel, de fiecare dată când vrea să verifice nivelul caloric al unei zile, o poate face. În același timp, această funcționalitate poate fi folosită și pentru a verifica cantitatea calorică rămasă în ziua curentă. Se introduce data curentă în câmp și apoi se verifică dacă numerele afișate în mesaj sunt mai mici decât nivelul caloric din profilul utilizatorului.

4.2. Prezentarea fișierelor sursă

Pentru back-end am creat un serviciu REST implementat cu tehnologiile specificate mai sus. În fișierul pom.xml am introdus dependențele utilizate. Apoi, pentru fiecare tabelă din baza de date am creat câte o serie de clase Entitate-DTO-Repository-Service-RestController în urma căreia am realizat operațiile CRUD(Create, Read, Update, Delete). Configurările aplicației legate de port și baza de date sunt introduse în fișierul application.properties. Pe parte de front-end am creat ecranele aplicației, iar pentru fiecare apăsare de buton am setat un eveniment prin care apelez serviciul REST din

back-end. Dependențele nivelului front-end și activity-urile sunt specificate în fișierul Manifest.xml, iar în string.xml sunt setate șirurile de caractere constante folosite drept text introdus pe buton sau pe etichetele câmpurilor pentru input.

5 DETALII DE IMPLEMENTARE

Pentru a implementa nivelul back-end al aplicației am folosit o arhitectură Representational State Transfer(REST). Prin REST am creat o serie de metode pe care le-am folosit pentru a crea operații request/response peste protocolul HTTP folosind metodele HTTP Get, Post, Put și Delete. Pentru a accesa aceste metode, am mapat controllerul pe URL, după care am mapat metoda pe URI și pe metoda HTTP. Pentru un serviciu REST, datele primite sau trimise prin HTTP trebuie să fie reprezentate în JSON sau XML.

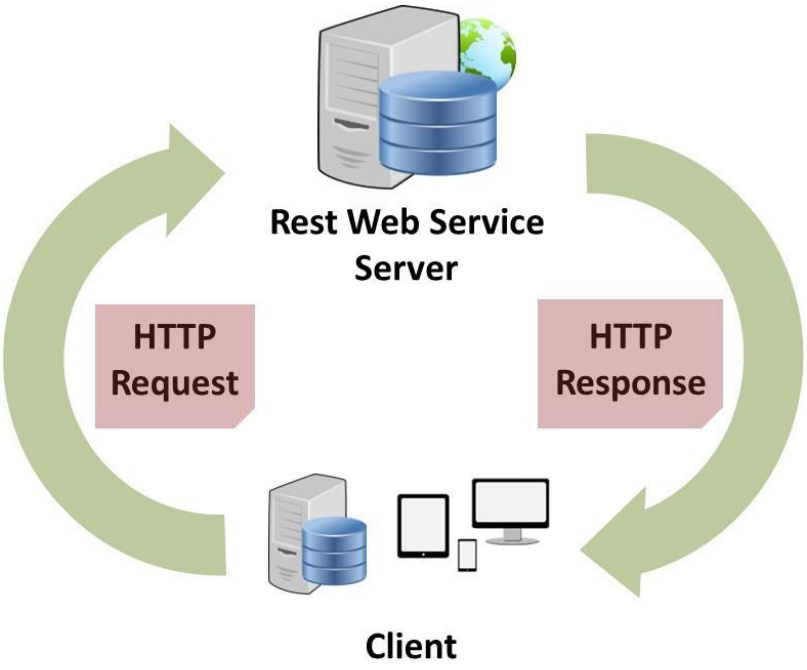

Fig 22[20] – REST

În Fig 20 este exemplificat modul în care interacționează un client cu un serviciu REST. Clientul face un request prin URL si metoda HTTP. Prin URL se găsește controllerul și metoda din acesta, apoi se executa funcționalitatea dorită. În caz ca se trimite un

body HTTP, acesta este de tip JSON sau XML. După executarea funcționalității, se întoarce un response către client, sub aceleași forme.

Legătura dintre front-end și back-end este realizată prin protocolul HTTP(Hypertext Transfer Portocol). Cum am precizat și mai sus, HTTP folosește câteva metode pentru transfer. Cele mai utilizate și cunoscute sunt cele utilizate pentru a realiza operații de tip CRUD, și anume:

- GET – metodă folosită pentru a returna date. Rezultatele întoarse de această metodă trebuie să fie același, ori de câte ori se repetă căutarea cu aceiași parametri.
- POST – respectând pattern-ul, POST este folosit pentru a crea o nouă entitate. Totuși, se poate folosi și pentru a actualiza o entitate existentă.
- PUT – metodă folosită pentru a actualiza o entitate existentă. Poate fi folosită și pentru a crea o nouă entitate. Diferența dintre POST și PUT este că, prin PUT, dacă nu sunt completate toate variabilele, acestea vor primi valoarea default.
- DELETE – metodă folosită pentru ștergerea unei resurse.

În response, HTTP trimite și un cod, în funcție de rezultatul response-ului. Codurile sunt structurate astfel:

- „1xx – erori informaționale”[21]
- „2xx – răspuns reușit”[21]
- „3xx” – redirectări”[21]
- „4xx – erori ale utilizatorilor”[21]
- „5xx – erori de server”[21]

În implementarea realizată, un șir caracteristic al acțiunii pleacă din front-end. Utilizatorul completează un formular, apoi, printr-un eveniment pus pe un buton, trimite un request HTTP cu un URL. În caz că metoda folosită este POST sau PUT, requestul are și body sub formă de JSON.

```
@RestController
@RequestMapping("/food")
public class FoodRestController {

    private FoodService foodService;
    private final Logger log = LoggerFactory.getLogger(FoodRestController.class);

    public FoodRestController() {
    }

    @Autowired
    public FoodRestController(FoodService foodService) { this.foodService = foodService; }

    @GetMapping("/find/{foodId}")
    @Timed
    public ResponseEntity<FoodDTO> getOne(@PathVariable("foodId") String id) {
        log.debug("REST request to get Food");

        Food food= foodService.findById(Integer.parseInt(id));
        if(food == null) {
            return ResponseEntity.badRequest().
                    headers(HeaderUtil.createFailureAlert(ENTITY_NAME,  errorKey: "FindError",  defaultM
                    .body( t: null);
        }

        return ResponseUtil.wrapOrNotFound(Optional.of(new FoodDTO(food)),
                HeaderUtil.createAlert( message: "Food: " + food.getName(), food.getName()));
    }
```

Fig 23 – REST Controller

În Fig 23 se observă metoda de căutare a unui aliment. Prin @RestController creez un Bean de acest tip. Această adnotare diferă de @Controller prin @ResponseBody, prin care valoarea de retur a metodei este scrisă în response-ul HTTP. Prin @RequestMapping și @GetMapping mapez URL-ul din header-ul HTTP pe o metodă dintr-un controller. Pe acest exemplu, dacă în request voi avea URL-ul http://localhost:4321/fitapp/food/find/1, aplicația intră în FoodRestController prin /food, apoi în metoda getOne prin /find/1. FoodId este o variabilă de cale, setată în apelul din front-end. Prin constructor injectez un service pentru această entitate iar, în metodele controllerului în voi apela pentru fiecare operație. Rezultatul metodelor dintr-un RestController este de tip ResponseEntity, transformat dintr-o entitate DTO. În Fig 24 este răspunsul pentru URL-ul de mai sus. Observăm un obiect în format JSON și headerul cu URL, metoda și statusul HTTP.

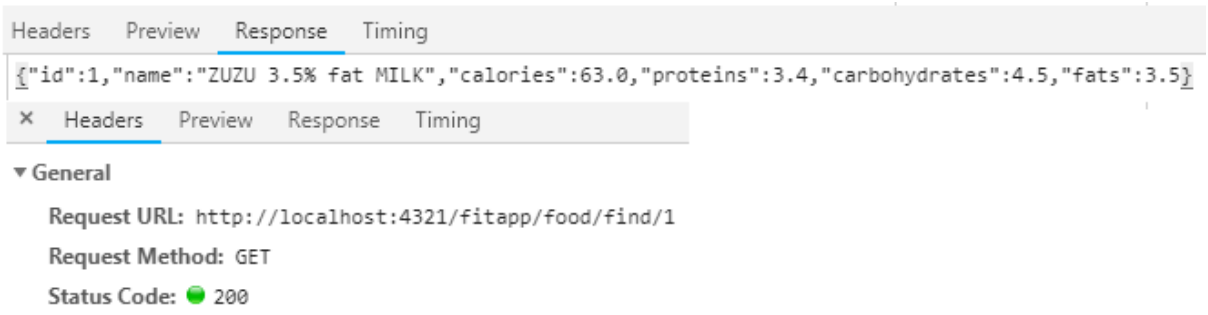

Fig 24 – RestController Response

După cum am menționat mai sus, în controller este injectat un Bean de tip Service. Beanul este definit cu adnotarea @Service pe clasă, după cum este exemplificat în Fig 25. Pentru a exemplifica cât mai clar funcționalitatea unui service, de această data am ales o adăugare în locul unei căutări. În acest service este injectat prin constructor un Bean de tipul mealRepository. Prin HTTP voi primi prin POST request de adăugare, iar în body-ul acestui request se află un JSON pt un obiect de tip MealDTO. Din controller trimit acel mealDTO către service, unde îl transform într-un obiect Meal. Adăugarea în bază se realizează după apelul metodei save din repository, apoi este întoarsă entitatea nou creață. După cum se observă, MealService este doar o interfață implementată de MealServiceImpl. În MealService sunt definite metodele de care am nevoie pentru a realiza operațiile CRUD.

```java
@Service
public class MealServiceImpl implements MealService{

    private final Logger log = LoggerFactory.getLogger(MealService.class);
    private final MealRepository mealRepository;

    @Autowired
    public MealServiceImpl(MealRepository mealRepository) { this.mealRepository = mealRepository; }

    @Override
    public Meal createMeal(MealDTO mealDTO) {
        Meal meal = new Meal();
        meal.setGrams(mealDTO.getGrams());
        meal.setMealdate(mealDTO.getMealdate());
        meal.setUserid(mealDTO.getUserid());
        meal.setFoodid(mealDTO.getFoodid());

        mealRepository.save(meal);
        log.debug("Create a new Meal: {}", meal);
        return meal;
    }
```

Fig 25 - Service

În următoarea etapă a procesului vom intra în accesul la baza de date. Acesta se face prin repository-ul injectat în service. Repository-ul este o interfață ce extinde interfața JpaRepository din pachetul springframework.data.jpa.

```
@Repository
public interface MealRepository extends JpaRepository<Meal, Integer> {

    Page<Meal> findByUserid(Pageable pageable, Integer userid);

    Page<Meal> findByMealdate(Pageable pageable, Date mealdate);

    Page<Meal> findByMealdateAndUserid(Pageable pageable, Date mealdate, Integer userid);

}
```

Fig 26 – Repository

Prin adnotarea @Repository definesc un Bean de acest tip. În JpaRepository sunt deja definite metodele pentru save, findById și findAll, iar în MealRepository am definit metodele de care mai am nevoie, pe lângă cele deja existențe în JpaRepository. Rezultatul de tip Page anunță faptul că se returnează o listă de entități, nu doar o entitate.

Pentru tot acest proces am definit două tipuri de entități. Prima este entitatea mapată pe tabela din bază prin Hibernate, în cazul din Fig 26 este vorba de Meal, iar a doua este o entitate numită MealDTO. Prima entitate este folosită pentru operațiunile cu baza de date, datorită mapării. Pin urmare aceasta va fi folosită în Repository, iar toate intrările și ieșirile din metodele repository-ului sunt de tip Meal. A doua entitate este entitatea introdusă de controller în response. Rezultatul unui apel Repository este Meal, prin urmare în service va ajunge o astfel de entitate. Tot în service se transformă parametrii primiți ca MealDTO în Meal, pentru a putea fi trimiși în Repository, apoi este returnat rezultatul un controller, unde va fi transformat înapoi în MealDTO, apoi în response. Entitățile Meal și MealDTO au aceleași atribute și aceleași metode, singura diferență dintre ele fiind maparea prin Hibernate.

Procesul parcurs mai sus reprezintă fluxul aplicației din front-end până în back-end și înapoi pentru o entitate, dar este valabil pentru toate celelalte entități. În total, pentru implementarea acestei lucrări am folosit 7 entități, după cum urmează:

- User: reprezintă informațiile utilizatorilor referitoate la partea de autentificare și înregistrare. Această entitate are trei parametrii, numele de utlizator, parola și codul pentru resetarea parolei.
- UserData: este strâns legată de entitatea User. Această entitate reprezintă datele pe care utilizatorul le completează în profil. Se crează prima dată când utilizatorul își completează profilul, apoi este actualizată la fiecare modificare, prin aceiași pași prezentați mai sus, folosind metoda HTTP PUT.
- UserDiet: reprezintă datele nutritive pe care trebuie să le respecte utilizatorul. Este creat odată cu crearea entității UserData și modificată de fiecare dată când se modifică și UserData.
- Workout: reprezintă programul de antrenament al utilizatorului. La fel ca UserDiet, este creat odată cu UserData și modificat de fiecare dată când este actualizat și UserData.
- Food: este entitatea ce reprezintă alimentele din bază. Această entitate este folosită pentru adăugarea în jurnalul alimentar al utilizatorului și conține cinci atribute: numele, caloriile, proteinele, carbohidrații și grăsimile.
- Meal: reprezintă consumarea unui aliment de către utilizator. Este strâns legată de entitatea Food, deoarece în această entitate rețin și alimentul consumat.
- Day: în această entitate sunt exercițiile pentru o zi de antrenament. Workout este legată de această entitate, deoarece în Workout se rețin zilele de antrenament.

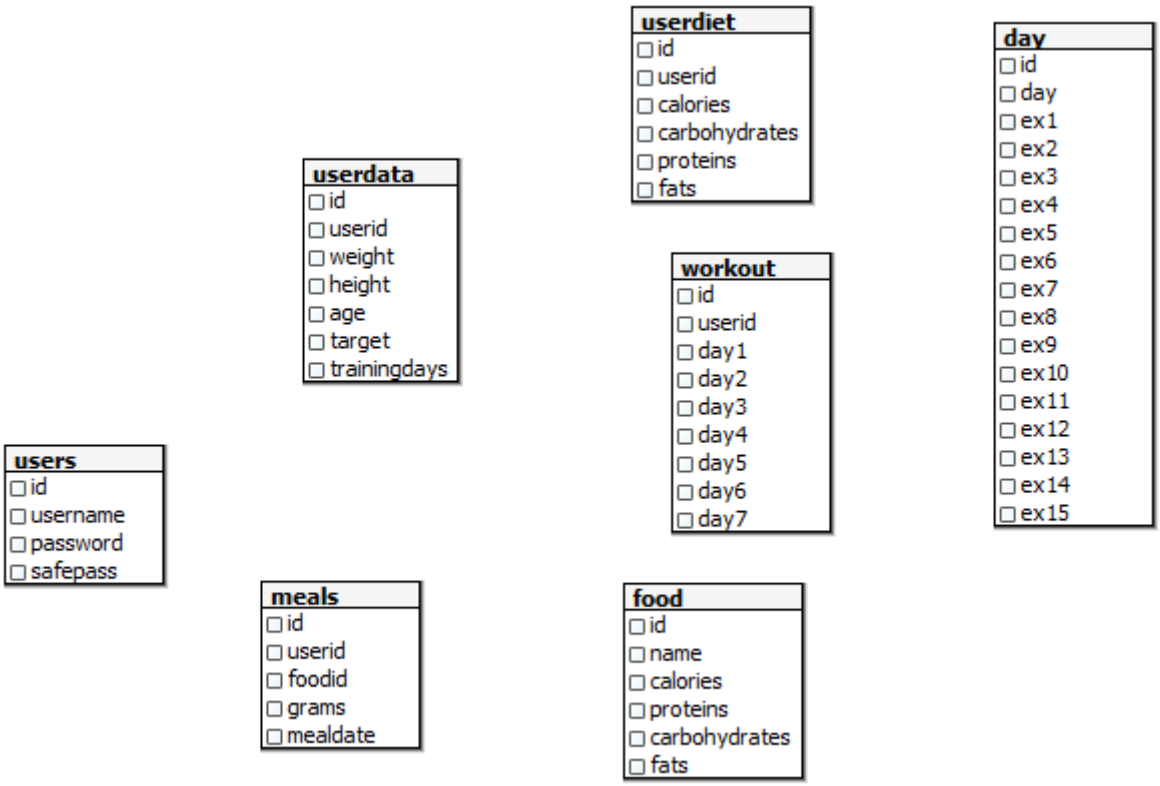

Fig 27 – Database

În Fig 27 se observă tabelele din baza de date. În continuare voi descrie câmpurile din fiecare tabelă.

- Users:
  - Id: Integer, Primary Key
  - Username: varchar(255), numele de utilizatori
  - Password: varchar(255), parola pentru fiecare utilizator
  - Safepass: varchar(255), codul pentru resetarea parolei
- Userdata:
  - Id: Integer, Primary Key
  - Userid: Integer, Foreign Key către tabela users
  - Weight: Double, greutatea corporală a utilizatorului
  - Height: Double, înaltimea utilizatorului
  - Age: Double, vârsta utilizatorului
  - Target: Double, greutatea dorită de utilizator
  - Trainingdays: Integer, numărul de antrenamente pe săptămână

- Userdiet:
    - Id: Integer, Primary Key
    - Userid: Integer, Foreign Key către tabela users
    - Calories: Double, numărul de calorii din dietă
    - Carbohydrates: Double, numărul de carbohidrați din dietă
    - Proteins: Double, numărul de proteine din dietă
    - Fats: Double, numărul de grăsimi din dietă
- Workout:
    - Id: Integer, Primary Key
    - Userid: Integer, Foreign Key către tabela users
    - Day1: Integer, Foreign Key către tabela Day, reprezintă prima zi din săptămâna de antrenament
    - Day2: Integer, Foreign Key către tabela Day, reprezintă a doua zi din săptămâna de antrenament
    - Day3: Integer, Foreign Key către tabela Day, reprezintă a treia zi din săptămâna de antrenament
    - Day4: Integer, Foreign Key către tabela Day, reprezintă a patra zi din săptămâna de antrenament
    - Day5: Integer, Foreign Key către tabela Day, reprezintă a cincea zi din săptămâna de antrenament
    - Day6: Integer, Foreign Key către tabela Day, reprezintă a sasea zi din săptămâna de antrenament
    - Day7: Integer, Foreign Key către tabela Day, reprezintă a saptea zi din săptămâna de antrenament
- Day:
    - Id: Integer, Primary Key
    - Day: Varchar(20), Numele zilei din săptămâna
    - Ex1 – Ex15: Varchar(200), reprezintă exercițiile pe care trebuie să le execute utilizatorul
- Food:
    - Id: Integer, Primary Key
    - Name: Varchar(50), denumirea alimentelor

    - Calories: Double, conținutul caloric al alimentelor
    - Carbohydrates: Double, conținutul de carbohidrați din alimente
    - Proteins: Double, conținutul de proteine din alimente
    - Fats: Double, conținutul de grăsimi din alimente
- Meals:
    - Id: Integer, Primary Key
    - Userid: Integer, Foreign Key către tabela users
    - Foodid: Integer, Foreig Key către tabela food
    - Grams: Integer, gramajul alimentelor
    - Mealdate: Date, data consumării alimentelor

6 CONCLUZII

Așa cum am menționat în capitolele anterioare, piața aplicațiilor fitness pentru platforme mobile este într-o continuă expansiune. Totodată, nevoia pentru astfel de aplicații este din ce în ce mai mare. Chiar dacă numărul aplicațiilor din industria fitness este mare, nu putem spune că există o aplicație completă, care să exceleze în toate punctele de interes pentru utilizatori. Comparând mai multe aplicații, putem observa că, luate individual, aplicațiile sunt incomplete, dar, luate împreună, se poate crea o aplicație notabila.

Prin aplicația acestei lucrări am dorit să creez o aplicație cu funcții în toate părțile industriei, astfel încât utilizatorul nu fie nevoit să descarce foarte multe aplicații. Interfața acesteia este simplistă, ușor de înțeles și de folosit. Pentru implementare am folosit doar limbaje și programe ce pot fi utilizate în mod gratuit, costul implementării rezumându-se doar la timp.

Legat de funcționalități, acestea sunt:

- Înregistrarea/autentificarea utilizatorilor
- Construirea unui program alimentar pe baza profilului utilizatorului
- Construirea unui program de antrenament pe baza profilului utilizatorului

- Introducerea alimentelor într-un jurnal și revizuirea alimentelor consumate într-o anumită zi

Un dezavantaj al aplicației o reprezintă baza de date. Accesul este la o singură bază de date, iar pentru un număr mare de utilizatori, este insuficient.

Deoarece aplicația este dezvoltată pentru a fi folosită de oricine, se pot găsi câteva îmbunătățiri ce pot fi realizate în viitor. În primul rând, programul de antrenament cuprinde doar ce exerciții trebuie să execute utilizatorul, precum și numărul de serii și repetări pentru fiecare exercițiu. Consider că, pentru persoanele care nu sunt familiarizate cu toate exercițiile, ar fi util un ghid de executare sau încărcarea unui videoclip cu o exemplificare.

De asemenea, ar fi utilă dezvoltarea unui calendar, în care utilizatorii să seteze când doresc să se antreneze, sau când este planificată o masă. După realizarea acestui calendar, utilizatorul va primi notificări în apropierea orelor din acest calendar.

Pe parte de nutriție, se pot adăuga două funcționalități. Prima ar fi mărirea numărului de alimente în baza de date, astfel utilizatorii vor avea mai multe informații. A doua îmbunătățire poate fi existența unei secțiuni din aplicație în care utilizatorul poate găsi rețete culinare cu anumite alimente cunoscute pentru efectul lor asupra sănătății.

Având în vedere toate punctele expuse mai sus, consider că mi-am atins obiectivul propus pentru această lucrare, și anume dezvoltarea unei aplicații prin care utilizatorii își pot îndeplini scopurile. Această aplicație este ușor de folosit, nu necesită cunoștințe în domeniul IT pentru folosire, fiind necesar doar înțelegerea conceptelor de calorii, proteine, carbohidrați și grăsimi.

7 BIBLIOGRAFIE

[1] http://www.businessmagazin.ro/analize/chiar-daca-romanii-fac-mai-putin-sport-decat-majoritatea-europenilor-piata-de-fitness-s-a-dublat-in-ultimii-3-ani-15968788

[2] http://www.sfatulmedicului.ro/Sanatate-prin-sport/sedentarismul-si-bolile-pe-care-le-ascunde-afla-cum-te-ajuta-miscarea-re_15619

[3] http://www.doctorulzilei.ro/inamicul-periculos-al-copiilor-sedentarismul/

[4] http://www.gandul.info/stiri/opt-romani-dintr-o-suta-fac-sport-in-mod-regulat-cum-arata-cifrele-in-ue-special-7682636

[5] http://www.sfatulmedicului.ro/Sanatate-prin-sport/sedentarismul-efecte-asupra-sanatatii_7950

[6] http://www.csid.ro/health/noutati-sanatate/peste-2-miliarde-de-persoane-sunt-obeze-la-nivel-mondial-13613628/

[7] https://ro.wikipedia.org/wiki/Obezitate

[8] https://en.wikipedia.org/wiki/Fast_food

[9] https://ro.wikipedia.org/wiki/Fast_food

[10] https://www.libertateapentrufemei.ro/articol/8-semne-si-simptome-ca-ai-un-deficit-de-proteine-154249

[11] https://ramonnastase.ro/blog/bazele-nutritiei/

[12] https://play.google.com/store/apps/details?id=com.sec.android.app.shealth&hl=ro

[13] https://play.google.com/store

[14] https://play.google.com/store/apps/details?id=sixpack.sixpackabs.absworkout

[15] https://play.google.com/store/apps/details?id=com.myfitnesspal.android

[16] https://play.google.com/store/apps/details?id=com.bodybuilding.krisgethin.musclebuilding

[17] https://www.androidout.com/item/android-apps/932549/kris-gethin-muscle-building/#

[18] https://www.w3schools.in/java-tutorial/

[19] https://www.tutorialspoint.com/maven/maven_pom.htm

[20] https://medium.com/@sagar.mane006/understanding-rest-representational-state-transfer-85256b9424aa

[21] https://ro.wikipedia.org/wiki/Hypertext_Transfer_Protocol