

**SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR,
INFORMATIKA SZAK**



**SAPIENTIA
ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM**

SkinnyB: Kalóriakövető applikáció

DIPLOMADOLGOZAT

Témavezető:
dr. Jánosi-Rancz Katalin Tünde,
Egyetemi adjunktus

Végzős hallgató:
Tövisi Zoltan

2023

**UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,
SPECIALIZAREA INFORMATICĂ**



UNIVERSITATEA SAPIENTIA

SkinnyB: Aplicație de nutritie

LUCRARE DE DIPLOMĂ

Coordonator științific: Absolvent:
dr. Jánosi-Rancz Katalin Tünde, Tövisi Zoltan
Lector universitar

2023

SAPIENTIA HUNGARIAN UNIVERSITY OF
TRANSYLVANIA
FACULTY OF TECHNICAL AND HUMAN SCIENCES
COMPUTER SCIENCE SPECIALIZATION



SAPIENTIA HUNGARIAN UNIVERSITY OF TRANSYLVANIA

SkinnyB: Calorie tracker application

BACHELOR THESIS

Scientific advisor: dr. Jánosi-Rancz Katalin Tünde,
Lecturer Student: Tövisi Zoltan

2023

LUCRARE DE DIPLOMĂ

Coordonator științific:
Dr. Jánosi-Rancz Katalin Tünde

Candidat: Tövisi Zoltán
Anul absolvirii: 2023

a) Tema lucrării de licență:

Tema lucrării este dezvoltarea unei aplicații de nutriție și fitness, care ajută oamenii la numărarea caloriilor, utilizând tehnologiile Java Spring Boot, React.js și bază de date PostgreSQL.

b) Problemele principale tratate:

Principalele probleme, pe care le tratează sunt următoarele: scădere în grutate, numărarea caloriilor și adoptarea unui stil de viață sănătos. Acestea fiind probleme relevante, cu care se confruntă tot mai mulți oameni.

c) Desene obligatorii:

Desenele obligatorii din proiect sunt: diagrame usecase și diagrame de secvență.

d) Softuri obligatorii:

Aplicație web bazată pe tehnologii Java Spring Boot și React.js, care realizează o aplicație de nutriție și fitness, care face posibil, ca utilizatorii să își poată număra caloriile, aportul de apă, activitățile fizice și somnul într-un mod convenabil. În afară de cele enumerate, utilizatorii mai beneficiază de rețete sănătoase și pot vizualiza statistice pe baza caloriilor introduse de ei.

e) Bibliografia recomandată:

- Bibliografia recomandată, care este utilizată în lucrare este următoarea:
Review on Reactjs, Bhupati Venkat Sai Indla, Yogeshchandra Puranik, International Journal of Trend in Scientific Research and Development(IJTSRD)

- Introduction to Spring Boot, K. Siva Prasad Reddy, Beginning Spring Boot 2: Applications and Microservices with the Spring Framework

<https://www.javatpoint.com/spring-boot-crud-operations>

<https://www.javatpoint.com/spring-boot-tutorial>

<https://www.javatpoint.com/reactjs-tutorial>

f) Termene obligatorii de consultații: Studentul a început dezvoltarea aplicației în septembrie 2022 și ne am întâlnit la fiecare 2-3 săptămâni

g) Locul și durata practicii: Universitatea „Sapientia” din Cluj-Napoca,
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș, sala / laboratorul 327A
Primit tema la data de: mai 2022
Termen de predare: 2 iulie 2023

Semnătura Director Departament

Semnătura coordonatorului

Semnătura responsabilului
programului de studiu

Semnătura candidatului

Declarație

Subsemnatul/a TÖVISI ZOLTAN, absolvent(ă) al/a specializării INFORMATICA, promoția 2023..., cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea, TARGU-MURES,
Data: 2023.06.15

Absolvent

Semnătura.....Zoltan

Kivonat

A szakdolgozatom célja egy kalóriák követő applikáció létrehozása. Ezt Java Spring Boot és React technológiákkal implementáltam, PostgreSQL adatbázis használatával. Az applikáció azt teszi lehetővé a felhasználók számára, hogy nyomon kövessék a bevitt kalóriák számát, és azokat a tevékenységeket, amelyek által kalóriákat égetnek el, illetve mennyi vizet isznak és mennyit alszanak, mivel ez is az egészséges életmódhoz tartozik, ami ugyanúgy fontos fogyás szempontjából. Az applikációm egy egyszerű, beszédes felhasználói felülettel rendelkezik, hogy megkönnyítse a felhasználók dolgát, a napi kalóriabevitel, aktivitások, víz bevitel, illetve alvás naplózását. Az applikáció a felhasználók számára lehetőséget biztosít, hogy étkezéseiket, italaikat, hozzáadhassák egyszerűen a nevük, elfogyasztott mennyiség vagy kalória tartalmuk alapján. Továbbá lehetőségük van az aktivitásai rögzítésére is, itt is a neve, időtartama és leégetett kalóriák által. Az applikációm automatikusan kiszámolja a felhasználó napi maximális kalóriabevitelét, a megadott adatok alapján, amiket a felhasználó a regisztráláskor ad meg egy form kitöltésével. A szakdolgozatomban be fogom mutatni az applikáció tervezésének és implementálásának lépéseinek megvalósítását. Ismertem a Java Spring Boot és React keretrendszerök használatát, valamint a PostgreSQL adatbázis konfigurációját. A létrehozott kalóriák követő applikáció hasznos eszköz lehet mindenknak, akik lefogyni szeretnének, és ebből a célból szeretnék a napi kalóriabevitelüket és aktivitásait nyomon követni, és ez által egy egészséges életmódot folytassanak és lefogyjanak.

Kulcsszavak: kalória, naplózás, lefogyás, egészséges életmód.

Rezumat

Scopul lucrării mele de licență a fost crearea unei aplicații de urmărire a caloriilor. Am implementat aceasta utilizând tehnologiile Java Spring Boot și React, folosind o bază de date PostgreSQL. Aplicația permite utilizatorilor să-și monitorizeze numărul de calorii consumate și activitățile prin care ard calorii, precum și cantitatea de apă băută și timpul petrecut în somn, deoarece acestea sunt, de asemenea, aspecte importante pentru un stil de viață sănătos și pentru pierderea în greutate. Aplicația mea dispune de o interfață simplă și intuitivă pentru a facilita utilizatorilor înregistrarea consumului zilnic de calorii, activități, aport de apă și înregistrarea somnului. Aplicația oferă utilizatorilor posibilitatea de a adăuga mesele și băuturile consumate, simplu, pe baza numelui, cantității consumate sau conținutului caloric. De asemenea, utilizatorii au posibilitatea de a înregistra activitățile lor, inclusiv numele, durata și numărul de calorii arse. Aplicația calculează automat aportul zilnic maxim de calorii al utilizatorului, pe baza datelor furnizate de acesta într-un formular completat la înregistrare. În lucrarea mea de licență voi prezenta implementarea etapelor de proiectare și dezvoltare a aplicației. Voi descrie utilizarea framework-urilor Java Spring Boot și React, precum și configurarea bazei de date PostgreSQL. Aplicația de urmărire a caloriilor creată poate fi un instrument util pentru cei care doresc să slăbească și doresc să-și monitorizeze aportul zilnic de calorii și activități pentru a adopta un stil de viață sănătos și a pierde în greutate.

Cuvinte cheie: calorii, jurnalizare, pierdere în greutate, stil de viață sănătos.

Abstract

The aim of my thesis was to create a calorie tracking application. I implemented it using Java Spring Boot and React technologies, with a PostgreSQL database. The application allows users to track their calorie intake and the activities that burn calories, as well as their water intake and sleep duration, as these are also important aspects of a healthy lifestyle and weight loss. My application features a simple and user-friendly interface to facilitate users in logging their daily calorie intake, activities, water consumption, and sleep. The application enables users to easily add their meals and beverages by providing the name, quantity, or calorie content. Users also have the option to record their activities, including the name, duration, and calories burned. The application automatically calculates the user's daily maximum calorie intake based on the data provided during registration through a form. In my thesis, I will present the implementation of the design and development stages of the application. I will discuss the usage of Java Spring Boot and React frameworks, as well as the configuration of the PostgreSQL database. The created calorie tracking application can be a useful tool for those who want to lose weight and track their daily calorie intake and activities to maintain a healthy lifestyle and achieve their weight loss goals.

Keywords: calories, logging, weight loss, healthy lifestyle.

Tartalomjegyzék

| | |
|--|-----------|
| 1. Bevezető | 11 |
| 2. Elméleti megalapozás és szakirodalmi tanulmány | 13 |
| 2.0.1. Beépített szoftverek, könyvtárak | 13 |
| 2.0.2. Java Spring Boot | 13 |
| 2.0.3. JPA Repository | 14 |
| 2.0.4. React.js | 15 |
| 2.0.5. PostgreSQL | 16 |
| 2.0.6. Hibernate | 16 |
| 3. A rendszer specifikációja | 18 |
| 3.0.1. Rendszer követelmények | 18 |
| 3.0.2. Felhasználói követelmények | 18 |
| 3.0.3. Use case diagram | 19 |
| 4. Tervezés | 24 |
| 4.0.1. A rendszer architektúrája | 24 |
| 4.0.2. Adatbázis terv | 25 |
| 4.0.3. Verziókövetés | 26 |
| 5. Kivitelezés | 27 |
| 5.0.1. Szekvencia diagramok | 27 |
| 6. Alkalmazás áttekintése | 30 |
| 6.0.1. Login | 30 |
| 6.0.2. Home page | 31 |
| 6.0.3. Dashboard | 31 |
| 6.0.4. FoodEntry hozzáadás | 32 |
| 6.0.5. Statisztikák | 33 |
| 6.0.6. Receptek | 33 |
| 6.0.7. Profil | 34 |
| 7. Összefoglaló | 35 |
| Köszönetnyilvánítás | 37 |
| Ábrák jegyzéke | 38 |

| | |
|-------------------------------------|-----------|
| Táblázatok jegyzéke | 39 |
| Irodalomjegyzék | 40 |
| Függelék | 41 |
| F.1. Az Overleaf felülete | 41 |

1. fejezet

Bevezető

A napi kalóriabevitel és aktivitások követése, naplózása nagyon fontos az egészséges életmód szempontjából. Leginkább azoknak fontos a kalóriabevitelüket számolni, naplózni, akiknek a céljuk a lefogyás. Ez valóban így van, de még fontos lehet azoknak is, akik meg akarják őrizni a jelenlegi testsúlyukat. Ilyenkor is a maximális kalóriabevitel, amit be kell tartson az illető, vagy meg fog hízni. Nos, ennek a célnak az elérésében segít az én kalóriakövető applikációm, aminek segítségével a felhasználók naplózni tudják a napi kalóriabevitelüket, de azelőtt persze az applikáció ki is számolja, hogy az adott felhasználó hány kalóriát kéne elfogyasszon napi szinten. Ezen kívül még a napi alvás időtartamát és a napi elfogyasztott vizet is naplózni lehet, mivel ezek is részei egy egészséges életmódnak és nagyon fontosak fogyás szempontjából is, főleg az, hogy hidratálva legyen a felhasználó, mivel a dehidratáltság éhség érzettel jár, miközben nem éhes, hanem szomjas az illető.

Az applikáció megvalósításához a Java Spring Boot-ot választottam, mivel ez az egyik legjobb tool a web applikációk backendjéhez és React.js-t a frontend részhez, mivel ez egy nagyon népszerű és könnyen elsajátítható Javascript library. Az adattárolásra pedig a PostgreSQL adatbázist választottam. használva a Hibernate-et, ami a Java programozási nyelvekhez egy tool, ami az osztályokból generál egy adatbázist, nem kell kézzel megsinálni. Ezen kívül még JpaRepository-t is használtam, egy API, ami tartalmaz CRUD műveleteket, ezért ezeket sem kellett én megírjam.

Az applikáció felhasználóbarát felülettel rendelkezik, ami egyszerű navigációt biztosít és intuitív funkcionálisokat a felhasználók számára. A felhasználók könnyen tudják naplózni az elfogyasztott ételeket, négy kategóriába sorolva őket: reggeli, ebéd, vacsora és nasi. Továbbá, rögzíteni tudják a fizikai aktivitásait, amivel kalóriákat égetnek, tehát a maximális kalóriabevitel száma növekedni fog. Ezen kívül tudják rögzíteni a vízmennyiséget, amit megisznak, valamint az alvási órákat is. Természetesen, ezeket mind ki is tudja törlni, ha esetleg hibázott, vagy csak meg akarta nézni, hogy az adott étel hány kalória. Törlés után, nyilván újra tud naplózni valamit, ha akarja.

A dolgozat során bemutatásra fog kerülni az applikáció és az adatbázis tervezése és implementálásának lépései. A Java Spring Boot, Hiberante-el együtt és React keretrendszerök használatát is részletesen fogom ismertetni.

Az így született kalóriakövető applikáció hasznos lehet azoknak, akik le akarnak fogyni, tehát követniük kell a napi kalóriabevitelüket, hogy kalóriadeficitben legyenek, tehát több kalóriát égessenek, mint amennyit bevisznek. Továbbá, hogy figyeljenek arra is, mennyi vizet isznak és mennyit alszanak, mert ezek is részei az egészséges életmódnak, ami kulcsfontosságú.

2. fejezet

Elméleti megalapozás és szakirodalmi tanulmány

2.0.1. Beépített szoftverek, könyvtárak

Dolgozatom jelen részében, nézzük meg milyen szoftvereket, technológiákat használtam az applikációm megvalósításához, amiket már egyébként említettem. A backend Java-ban, pontosabban Spring Boot-ban van implementálva, PostgreSQL adatbázis és Java Hibernate framework és JpaRepository használatával. A frontendet pedig React.js-ben van implementálva.

2.0.2. Java Spring Boot

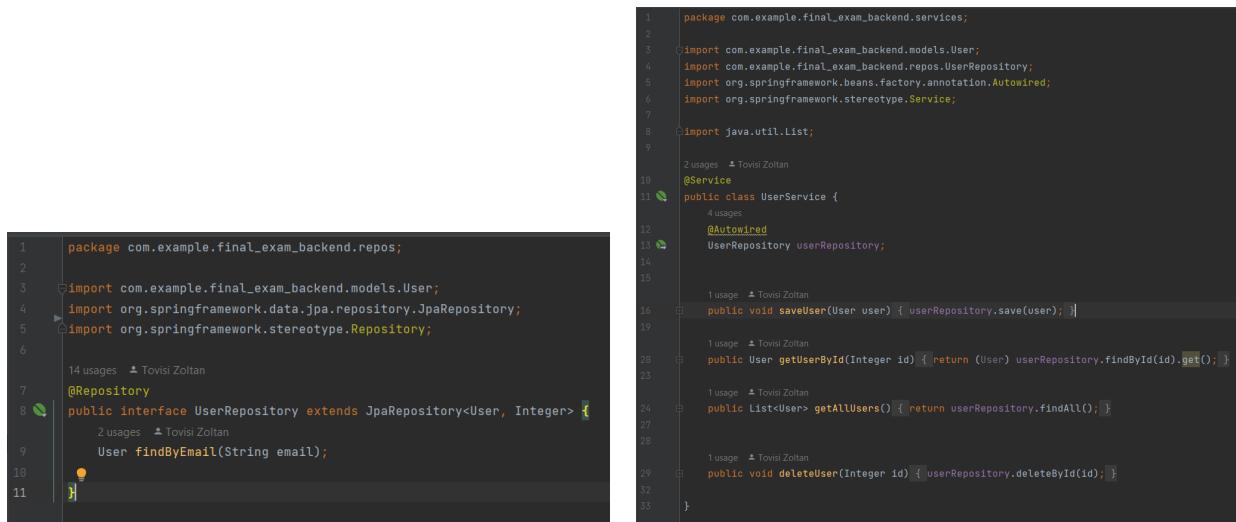
A Spring Boot [jav21c] az egyik legjobb tool a Spring framework webapplikációk backend taskjainak implementálására.

```
@Entity
@Table(name = "UserData")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    public Integer id;
    @Column
    public String name;
    2 usages
    @Column
    public String email;
    2 usages
    @Column
    public double height;
    2 usages
    @Column
    public double currentWeight;
```

2.1. ábra. Spring Boot példakód

Itt látható egy Java osztály, ami egy táblának felel meg az adatbázisban az @Entity, illetve @TableName annotációknak köszönhetően.

2.0.3. JPA Repository



The screenshot shows two side-by-side code editors. The left editor displays the `UserRepository.java` interface:

```
1 package com.example.final_exam_backend.repos;
2
3 import com.example.final_exam_backend.models.User;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.stereotype.Repository;
6
7 @Repository
8 public interface UserRepository extends JpaRepository<User, Integer> {
9     User findByEmail(String email);
10 }
11 }
```

The right editor displays the `UserService.java` class:

```
1 package com.example.final_exam_backend.services;
2
3 import com.example.final_exam_backend.models.User;
4 import com.example.final_exam_backend.repos.UserRepository;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 import java.util.List;
9
10 @Service
11 public class UserService {
12     @Autowired
13     UserRepository userRepository;
14
15     public void saveUser(User user) { userRepository.save(user); }
16
17     public User getUserId(Integer id) { return userRepository.findById(id).get(); }
18
19     public List<User> getAllUsers() { return userRepository.findAll(); }
20
21     public void deleteUser(Integer id) { userRepository.deleteById(id); }
22
23 }
24
25
26
27
28
29
30
31
32
33 }
```

Annotations like `@Service`, `@Autowired`, and `@Repository` are visible, along with various imports and usage comments.

2.2. ábra. JPA Repository

A JpaRepository egy JPA (Java Persistence API) specifikus kiterjesztése. Tartalmazza a teljes CrudRepository és PagingAndSortingRepository-kat. Tehát ez egy API-t tartalmaz CRUD műveletekre.

Ehhez egy interface-et kell létrehozzunk, ami örökli a JpaRepository-t, ezután pedig egy JpaRepository típusú objektumot létrehozunk, és az által tudjuk elérni a CRUD [jav21b] műveleteket belőle, például save, delete, getAll, findAll stb. Ezt szemléltetem a 2.2. ábrán.

2.0.4. React.js

A React.js [jav21a] egy ingyenes open-source frontend JavaScript library, felhasználói felületek építéséhez, komponensek alapján. A segítségével, single-page, mobil, vagy server-alapú applikációkat lehet implementálni.

```
1 import {BrowserRouter} from "react-router-dom"
2 import { GoogleOAuthProvider } from '@react-oauth/google';
3
4 import React from 'react';
5 import './index.css';
6 import './App.css';
7 import App from './App';
8 import {AuthProvider} from './providers/AuthProvider';
9 import {createRoot} from 'react-dom/client';
10
11 const root = createRoot(document.getElementById("root"))
12 root.render(
13   <React.StrictMode>
14     <GoogleOAuthProvider clientId="827107914359-gg5543putd8a6n157pnu549kbd954g5q.apps.googleusercontent.com">
15       <AuthProvider>
16         <BrowserRouter>
17           <App />
18         </BrowserRouter>
19       </AuthProvider>
20     </GoogleOAuthProvider>
21   </React.StrictMode>
22 );
```

2.3. ábra. React példakód

Az én applikációm is szerver alapú, ebben a kódrészletben lehet látni, ahogy a render() függvény megjeleníti őt.

```
<Routes>
  <Route path="/" element={<Home />} />
  <Route path="/register" element={<Register />} />
  { // only authorized users have access to the following routes
    authenticated &&
    <>
      <Route exact path="/dashboard" element={<Dashboard />} />
      <Route exact path="/statistics" element={<Statistics />} />
      <Route exact path="/meals" element={<Meals />} />
      <Route exact path="/profile" element={<Profile />} />

      <Route exact path="/salmongreenbeans" element={<SalmonGreenBeans />} />
      <Route exact path="/broccolisteak" element={<BroccoliSteaks />} />
      <Route exact path="/airsquash" element={<AirSquash />} />
      <Route exact path="/fishandveggies" element={<FishAndVeggies />} />
      <Route exact path="/shrimpletteucerwraps" element={<ShrimpLettuceWraps />} />
      <Route exact path="/beankaletoast" element={<BeanKaleToast />} />
      <Route exact path="/garannahsalachicken" element={<GarannahSaladChicken />} />
      <Route exact path="/califlower_tacos" element={<CaliflowerTacos />} />
      <Route exact path="/flankstake" element={<MarinatedFlankStake />} />
      <Route exact path="/porkwithbutternutquash" element={<PorkWithButternutQuash />} />
      <Route exact path="/chickenfajitas" element={<SheetPanChickenFajitas />} />
      <Route exact path="/shrimpceviche" element={<ShrimpCeviche />} />
      <Route exact path="/califlower_soup" element={<CaliflowerSoup />} />
      <Route exact path="/californiarollsalad" element={<CaliforniaRollSalad />} />
      <Route exact path="/shakshuka" element={<Shakshuka />} />
    </>
  }
  <Route
    path="*"
    element={<Navigate to="/" replace />}
  />
</Routes>
```

```
7   return (
8     <nav className="nav">
9       <div>
10         <a href="/">
11           
12         </a>
13       </div>
14       <ul>
15         <CustomLink to="/"> HOME </CustomLink>
16         &authenticated && (
17           <>
18             <CustomLink to="/dashboard"> DASHBOARD </CustomLink>
19             <CustomLink to="/statistics"> STATISTICS </CustomLink>
20             <CustomLink to="/meals"> MEALS </CustomLink>
21             <CustomLink to="/profile"> PROFILE </CustomLink>
22           </>
23         )
24       &authenticated ? (
25         <div className="profileContainer">
26           <img src={profile.picture} alt="profilePic" className="profilePic" />
27           <button className="logout" onClick={logout}> LOGOUT </button>
28         </div>
29       ) :
30         <div className="authButtons">
31           <button onClick={login}> LOGIN </button>
32         </div>
33       </ul>
34     </nav>
35   );
36 
```

2.4. ábra. React routing

Az applikációmban react routing-ot használok, a 2.4. ábrán látható, ahogy megadom, hogy melyik path, melyik js komponensre vigyen, azután pedig beállítom a routingot. minden gomb megnyomásával létrejön a path, ami szerint az applikáció navigál az adott js komponensre.

2.0.5. PostgreSQL

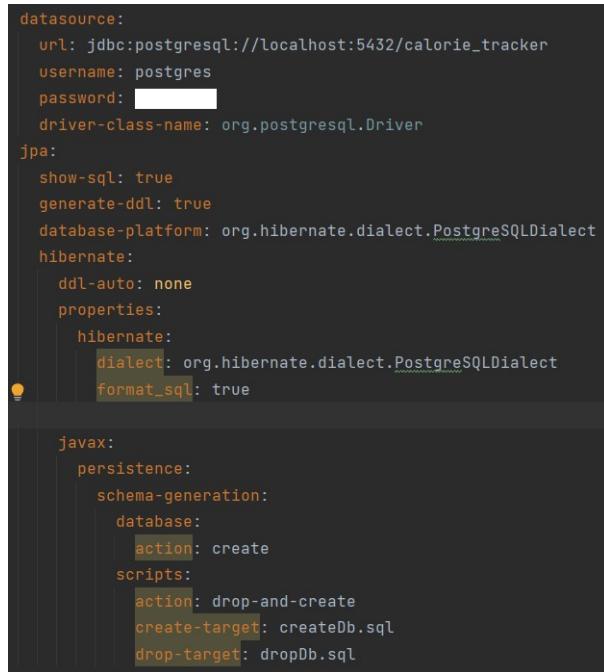
A PostgreSQL egy ingyenes open-source adatbáziskezelő rendszer, amelyben relációs adatbázisokat lehet létrehozni és kezelní az SQL nyelv segítségével.

```
-- Table: public.user_data
-- DROP TABLE IF EXISTS public.user_data;
CREATE TABLE IF NOT EXISTS public.user_data
(
    id integer NOT NULL DEFAULT 'nextval(''user_data_id_seq''::regclass)',
    access_token character varying(255) COLLATE pg_catalog."default",
    activity_level character varying(255) COLLATE pg_catalog."default",
    age integer,
    birth_date character varying(255) COLLATE pg_catalog."default",
    current_weight double precision,
    email character varying(255) COLLATE pg_catalog."default",
    finished_onboarding boolean,
    goal_date character varying(255) COLLATE pg_catalog."default",
    goal_weight double precision,
    height double precision,
    name character varying(255) COLLATE pg_catalog."default",
    sex character varying(255) COLLATE pg_catalog."default",
    start_weight double precision,
    weekly_goal integer,
    address character varying(255) COLLATE pg_catalog."default",
    CONSTRAINT user_data_pkey PRIMARY KEY (id)
)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS public.user_data
OWNER to postgres;
```

2.5. ábra. SQL példakód

Itt látható egy SQL kód, amivel létrehozom a userdata nevű táblát

2.0.6. Hibernate



```
datasource:
  url: jdbc:postgresql://localhost:5432/calorie_tracker
  username: postgres
  password: [REDACTED]
  driver-class-name: org.postgresql.Driver
jpa:
  show-sql: true
  generate-ddl: true
  database-platform: org.hibernate.dialect.PostgreSQLDialect
  hibernate:
    dialect: org.hibernate.dialect.PostgreSQLDialect
    format_sql: true
jaxb:
  persistence:
    schema-generation:
      database:
        action: create
      scripts:
        action: drop-and-create
        create-target: createDb.sql
        drop-target: dropDb.sql
```

2.6. ábra. Hibernate bekonfigurálása

Itt látható, ahogy rácsatlakozok az adatbázisra, illetve beállítom a hibernate-et, hogy Ő generálja ki a tábláimat, a Java osztályaimból. Ezt használva, nem kellett én kézzel létrehozzam a táblákat.

```

22
23     create table User (
24         id integer not null,
25         activityLevel varchar(255),
26         age integer,
27         birthDate varchar(255),
28         currentWeight float(53),
29         email varchar(255),
30         finishedOnboarding boolean,
31         goalDate varchar(255),
32         goalWeight float(53),
33         height float(53),
34         name varchar(255),
35         sex varchar(255),
36         startWeight float(53),
37         weeklyGoal integer,
38         primary key (id)
39     );
40
41     create table WaterEntry (
42         id serial not null,
43         dateTime timestamp(6) not null,
44         type varchar(255),
45         amount integer,
46         consumedAt timestamp(6),
47         user_id integer not null,
48         primary key (id)
49     );

```

```

22
23     create table User (
24         id integer not null,
25         activityLevel varchar(255),
26         age integer,
27         birthDate varchar(255),
28         currentWeight float(53),
29         email varchar(255),
30         finishedOnboarding boolean,
31         goalDate varchar(255),
32         goalWeight float(53),
33         height float(53),
34         name varchar(255),
35         sex varchar(255),
36         startWeight float(53),
37         weeklyGoal integer,
38         primary key (id)
39     );
40
41     create table WaterEntry (
42         id serial not null,
43         dateTime timestamp(6) not null,
44         type varchar(255),
45         amount integer,
46         consumedAt timestamp(6),
47         user_id integer not null,
48         primary key (id)
49     );

```

2.7. ábra. Generált script létrehozásra

```

alter table if exists FoodEntry
| drop constraint if exists FKeojmhhf2kugktxmmbvj8xice;

alter table if exists SleepEntry
| drop constraint if exists FKrs8jbh8ypkcm9hpsl4sdmds4g;

alter table if exists WaterEntry
| drop constraint if exists FKeo7q66fmi06gtxpelh3xe6evs;

alter table if exists WorkoutEntry
| drop constraint if exists FK9pdqns0dgtfymxl6paunacctv;

drop table if exists FoodEntry cascade;

drop table if exists SleepEntry cascade;

drop table if exists User cascade;

drop table if exists WaterEntry cascade;

drop table if exists WorkoutEntry cascade;

alter table if exists FoodEntry
| drop constraint if exists FKeojmhhf2kugktxmmbvj8xice;

alter table if exists SleepEntry
| drop constraint if exists FKrs8jbh8ypkcm9hpsl4sdmds4g;

```

2.8. ábra. Generált script törlésre

A Hibernate nem csak csupán az adatbázist generálja ki, hanem a scriptet is, amiből létrejön az adatbázis, ezt a 2.7. ábrán láthatjuk. Ezen kívül, még egy scriptet, amivel ki lehet törölni az egész adatbázist, vagy csak egy táblát belőle, ez látható a 2.8. ábrán.

3. fejezet

A rendszer specifikációja

3.0.1. Rendszer követelmények

Az alkalmazás használatához a következő rendszerkövetelmények szükségesek:

- Internetkapcsolattal rendelkező számítógép, laptop, vagy okostelefon
- Támogatott webböngésző, például Google Chrome, Mozilla Firefox, Safari vagy Microsoft Edge

Az alkalmazás webalapú, ezért nincs szükség külön telepítésre. A felhasználók a kompatibilis webböngészőjüket használva hozzáférhetnek az alkalmazáshoz.

3.0.2. Felhasználói követelmények

Az applikációt csak Google fiókkal lehet használni, ezért a felhasználóknak kell rendelkezniük egy ilyen fiókkal.

Miután bejelentkeztek a Google fiókkal, ki kell tölteniük egy onboarding form-ot kell kitölteniük a következő adatokkal:

- Név
- Magasság
- Jelenlegi testsúly
- Kívánt testsúly
- Meddig szeretné elérni célját
- Nem
- Életkor
- Fizikai aktivitás szint

Ahhoz, hogy egy új WorkoutEntry-t hozzanak létre, a következőket kell beírják:

- Workout típusa
- Leégetett kalóriák
- Workout időtartama

Ahhoz, hogy egy új SleepEntry-t hozzanak létre, a következőket kell beírják:

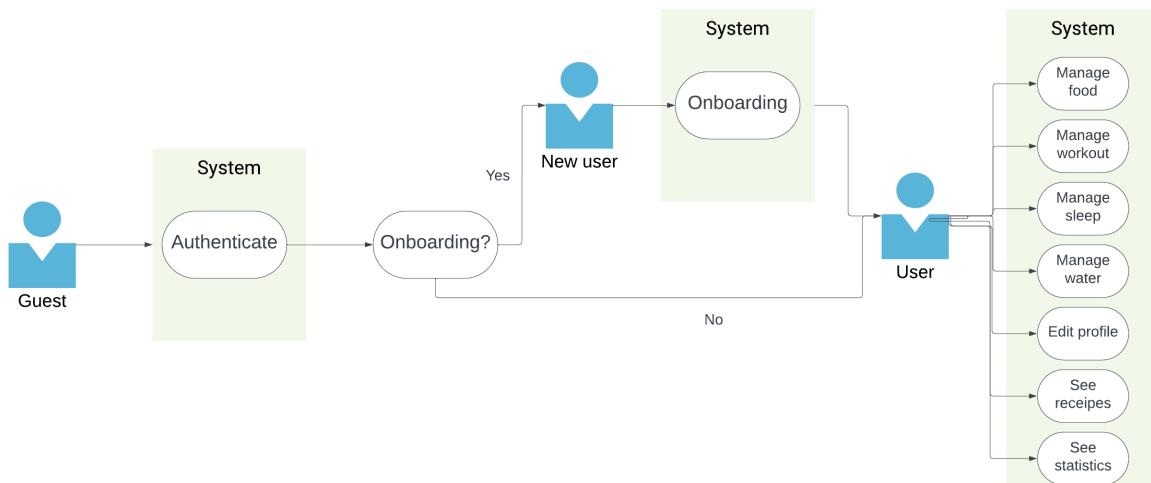
- Alvás időtartama

Ahhoz, hogy egy új FoodEntry-t hozzanak létre, a következőket kell beírják:

- Étel neve
- Étel kalóriatartalma

Ezen kívül lehetőségük van rákeresni ételekre, a rendszer pedig megmondja, hogy az hány kalória, ha beírja hány grammot fogyasztott. Továbbá, hozzá is tudnak adni egy ételt, ha az nincs meg, ide pedig a nevét és kalóriatartalmát per 100 gramm kell beírni.

3.0.3. Use case diagram



3.1. ábra. Use case diagram

A 3.1 ábra szemlélti az alkalmazás folyamatát, az alábbiakban pedig részletesebben leírok minden lépést:

3.1. táblázat. Sign-in Use Case

| | |
|----------------------|--|
| UC-0001 | Sign-in |
| Aktorok | Vendég |
| Leírás | A vendég egy létező Google fiókkal be tud jelentkezni |
| Előfeltételek | <ul style="list-style-type: none"> • A vendég még nincs bejelentkezve • A vendégnek van egy létező Google fiókja |
| Utófeltételek | <ul style="list-style-type: none"> • A vendég most egy felhasználó • A felhasználó a Home oldalra navigál |
| Normál folyam | <ul style="list-style-type: none"> • A felhasználó bejelentkezik a Google fiókjával • A felhasználó be van jelentkeztetve és a Home oldalra van küldve |

3.2. táblázat. Onboarding Use Case

| | |
|----------------------|---|
| UC-0002 | Onboarding |
| Aktorok | Új felhasználó |
| Leírás | Az új felhasználó releváns egészségügyi és fitness adatokat kell megadjon |
| Előfeltételek | <ul style="list-style-type: none"> • A felhasználó be van jelentkezve • Ez nem az első alkalom, amikor bejelentkezett az applikációba |
| Utófeltételek | <ul style="list-style-type: none"> • Az új felhasználó most egy normál felhasználó • A felhasználó a Home oldalra navigál |
| Normál folyam | <ul style="list-style-type: none"> • A felhasználó kitölti az onboarding formot és elküldi • A felhasználó be van jelentkeztetve és a Home oldalra van küldve • A felhasználó most megtekintheti és szerkesztheti a napi fitnesz információját |

3.3. táblázat. Alvás Mennyiség Kezelés Use Case

| | |
|----------------------|---|
| UC-0003 | Alvás Mennyiség Kezelés |
| Aktorok | Felhasználó |
| Leírás | A felhasználó hozzáadhat/törölhet SleepEntry-ket |
| Előfeltételek | <ul style="list-style-type: none"> • A felhasználó be van jelentkezve • A Dashboard oldalon van |
| Utófeltételek | <ul style="list-style-type: none"> • Változik az adatbázis |
| Normál folyam | <ul style="list-style-type: none"> • A felhasználó megnyomja a + gombot • Az adatbázis bővül a megadott adatokkal |

3.4. táblázat. Workout Kezelés Use Case

| | |
|----------------------|---|
| UC-0004 | Workout Kezelés |
| Aktorok | Felhasználó |
| Leírás | A felhasználó hozzáadhat/törölhet WorkoutEntry-ket |
| Előfeltételek | <ul style="list-style-type: none"> • A felhasználó be van jelentkezve • A Dashboard oldalon van |
| Utófeltételek | <ul style="list-style-type: none"> • Változik az adatbázis • Változik a maximális kalóriabevitel |
| Normál folyam | <ul style="list-style-type: none"> • A felhasználó megnyomja a + gombot • Az adatbázis bővül a megadott adatokkal • A napi maximális kalóriabevitel változik |

3.5. táblázat. Víz Bevitel Kezelés Use Case

| | |
|----------------------|---|
| UC-0005 | Víz Bevitel Kezelés |
| Aktorok | Felhasználó |
| Leírás | A felhasználó hozzáadhat/törölhet WaterEntry-ket |
| Előfeltételek | <ul style="list-style-type: none"> • A felhasználó be van jelentkezve • A Dashboard oldalon van |
| Utófeltételek | <ul style="list-style-type: none"> • Változik az adatbázis |
| Normál folyam | <ul style="list-style-type: none"> • A felhasználó megnyomja a + gombot • Az adatbázis bővül a megadott adatokkal |

3.6. táblázat. Étel Bevitel Kezelés Use Case

| | |
|----------------------|---|
| UC-0006 | Étel Bevitel Kezelés |
| Aktorok | Felhasználó |
| Leírás | A felhasználó hozzáadhat/törölhet WaterEntry-ket |
| Előfeltételek | <ul style="list-style-type: none"> • A felhasználó be van jelentkezve • A Dashboard oldalon van |
| Utófeltételek | <ul style="list-style-type: none"> • Változik az adatbázis • A napi maximális kalóriabevitel változik |
| Normál folyam | <ul style="list-style-type: none"> • A felhasználó megnyomja a + gombot • Az adatbázis bővül a megadott adatokkal |

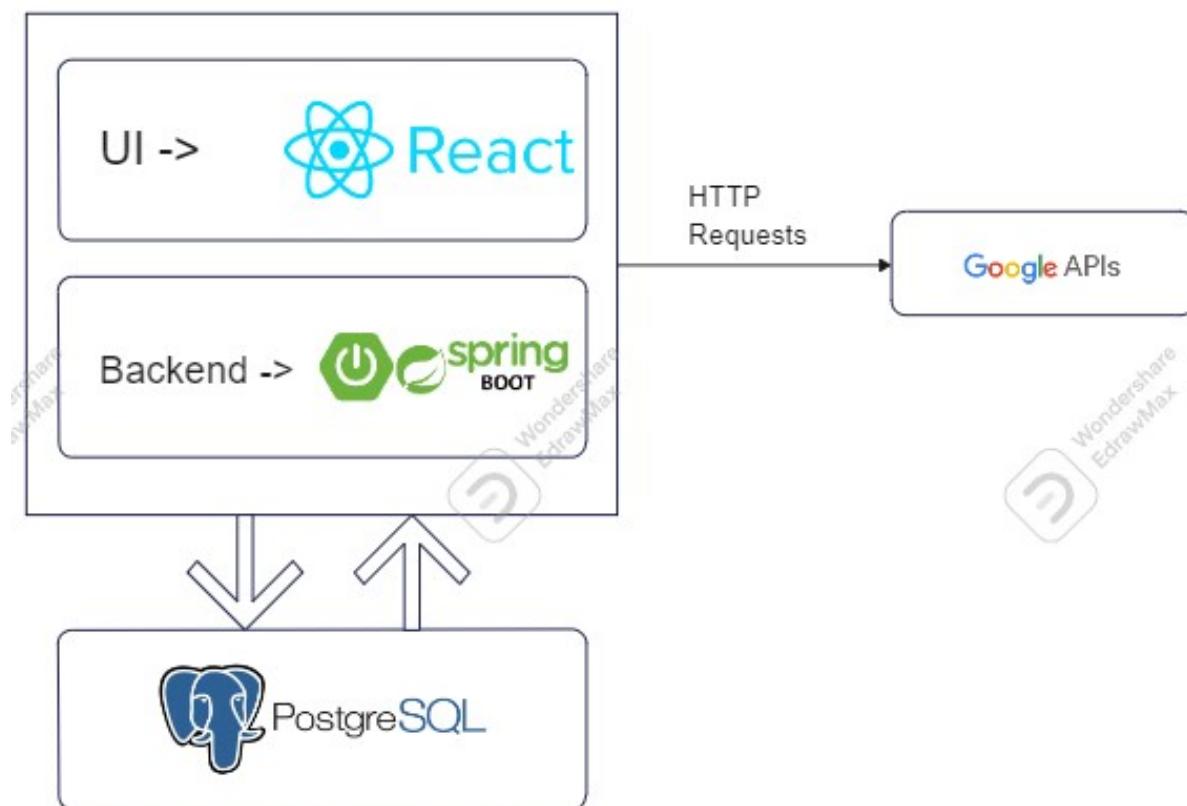
3.7. táblázat. Profil szerkesztés Use Case

| | |
|----------------------|---|
| UC-0007 | Profil szerkesztés |
| Aktorok | Felhasználó |
| Leírás | A felhasználó szerkesztheti a profilját |
| Előfeltételek | <ul style="list-style-type: none"> • A felhasználó be van jelentkezve • A Profile oldalon van |
| Utófeltételek | <ul style="list-style-type: none"> • Változik az adatbázis • A napi maximális kalóriabevitel és fitness adatok változnak |
| Normál folyam | <ul style="list-style-type: none"> • A felhasználó megnyomja az Edit profile gombot • A felhasználó megváltoztatja a kívánt adatokat • Az adatbázis megváltozik a megadott adatok alapján • A felhasználó maximális kalóriabevitele újra ki lesz számolva |

4. fejezet

Tervezés

4.0.1. A rendszer architektúrája



4.1. ábra. Architektura diagram

Az applikációm több technológiát használva készült. Ezek kommunikálnak egymás-sal, s így születik meg a végső eredmény. A 4.1. ábra szemlélteti, hogyan van elosztva az applikáció működtetése, egyesek általam készítve, mások pedig kívülálló forrásokból.

4.0.2. Adatbázis terv

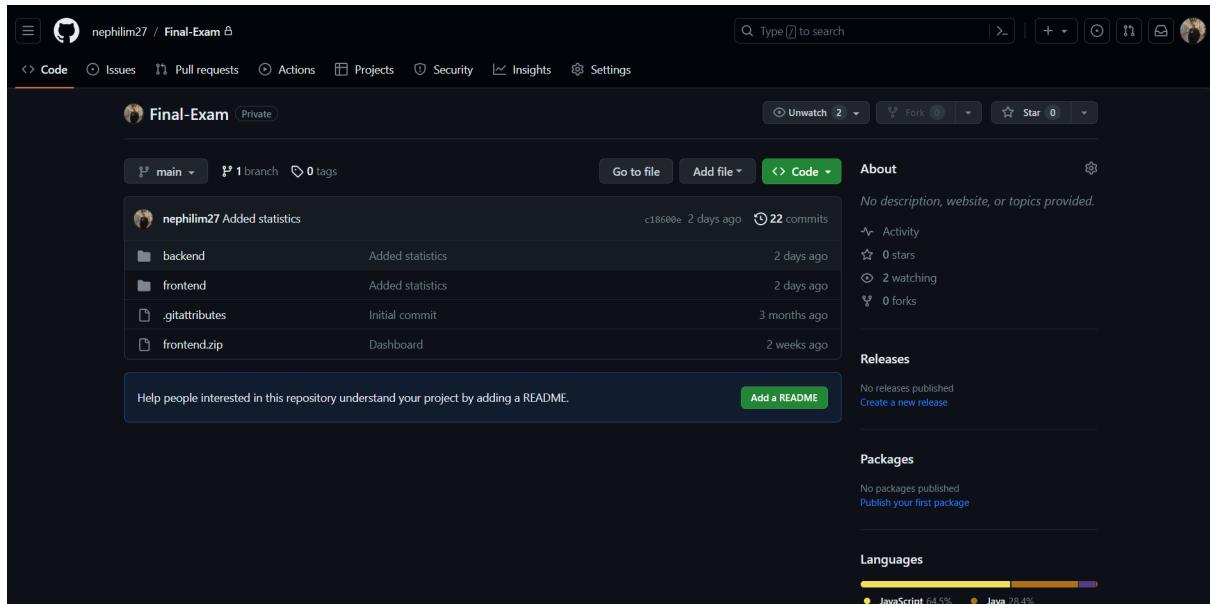


4.2. ábra. Adatbázis diagram

A 4.2 ábrán látható az adatbázisom terve. Amint már említettem, ez PostgreSQL-ben van létrehozva, egy erőteljes adatbáziskezelő rendszer, ami lehetővé teszi, hogy egy széleskörű adattípusokkal dolgozzunk.

Ebben az adatbázisban tárolom el a felhasználók adatait, beleértve a fitness adatokat is, amik mind nagyon relevánsak az alkalmazás optimális működésének szempontjából. Ezen kívül itt van eltárolva minden entry, amit benaplóz, azaz minden étel, víz mennyisége, fizikai aktivitások és alvás időtartama. Ezen kívül van még egy Foods tábla, ott vannak eltárolva az ételek, amikre rá lehet keresni az applikációban és úgy rákeresni az ételekre és azáltal benaplózni.

4.0.3. Verziókövetés



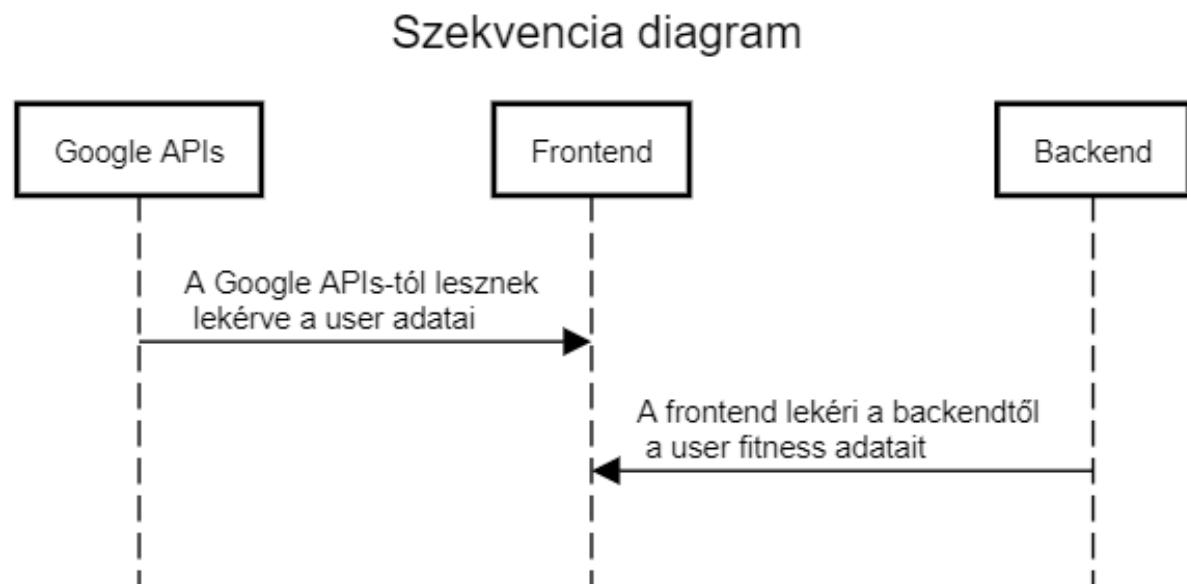
4.3. ábra. Verzió követés

Az applikáció implementálása során, verziókövetést is végeztem, erre a GitHub-ot használtam. Ez az egyik legnépszerűbb verziókövető rendszer a fejlesztők körében. A 4.3. ábrán látható a repository, ahol az applikációm van, [ide](#) kattintva pedig meg is lehet tekinteni.

5. fejezet

Kivitelezés

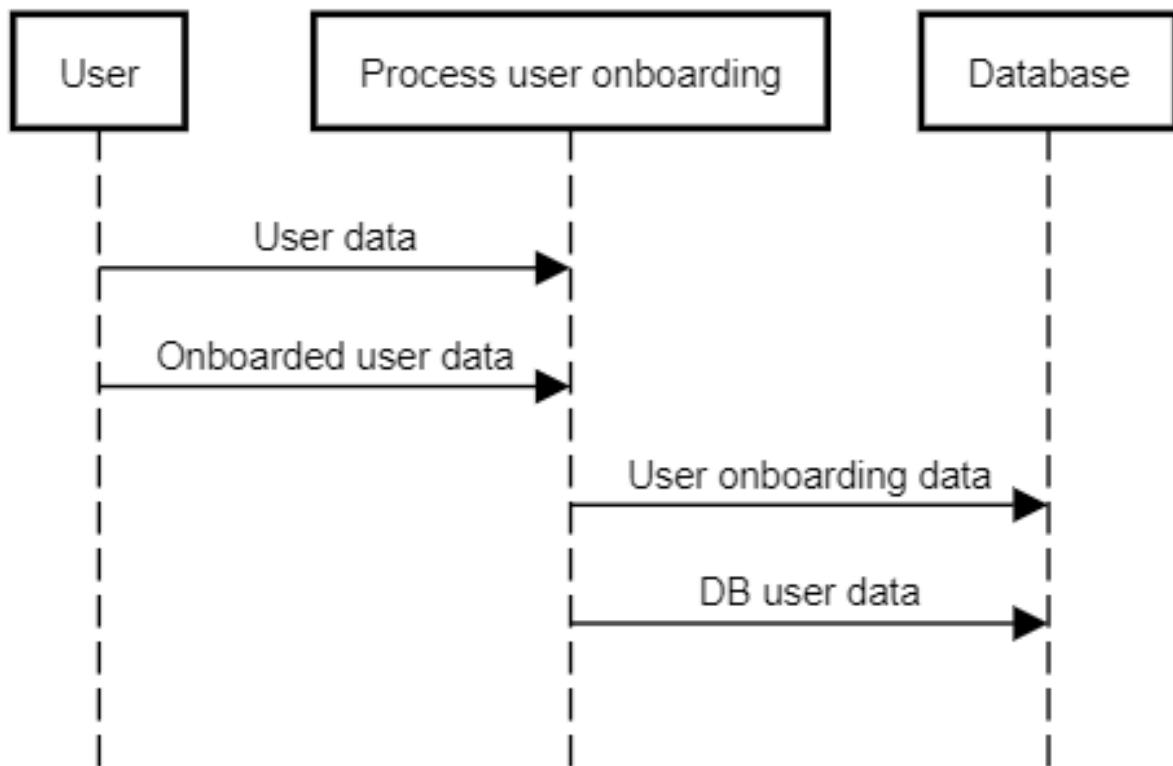
5.0.1. Szekvencia diagramok



5.1. ábra. Bejelentkezés szekvencia diagram

Az 5.1. ábra szemlélti a bejelentkezést. A Google APIs-tól le lesznek kérve a felhasználó adatai, az email cím alapján kérés küldődik a backendhez, ahonnan lekérjük a felhasználó fitnesz adatait, és így születik a profilja.

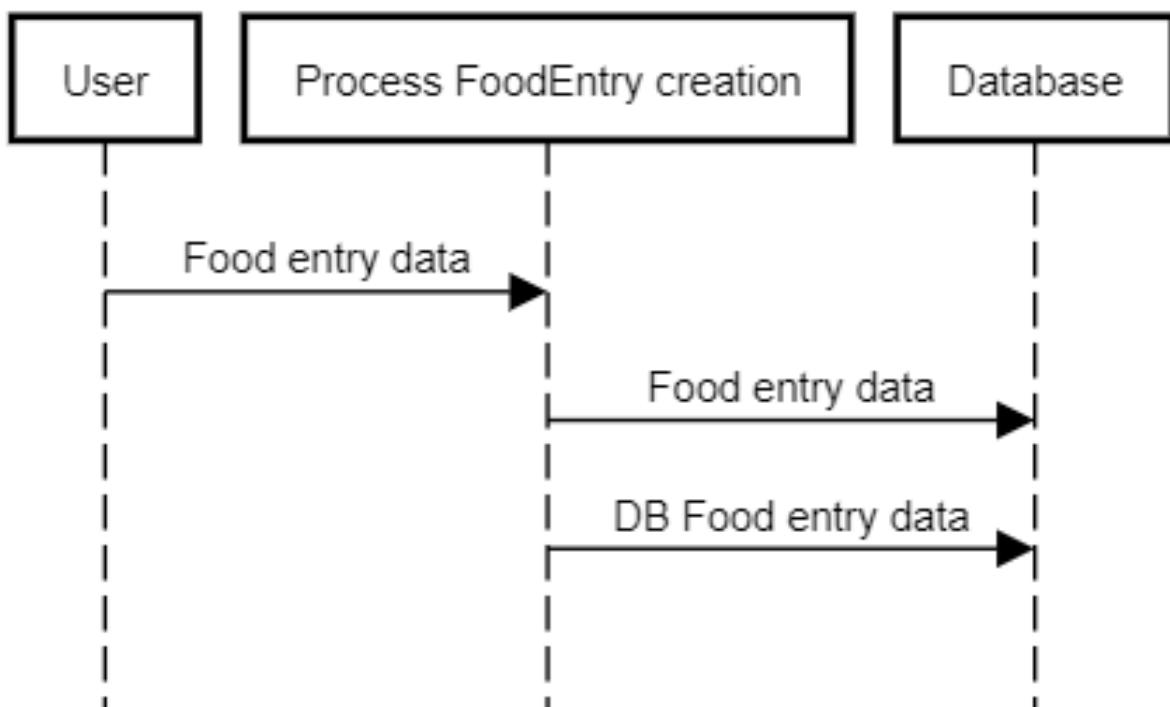
Szekvencia diagram



5.2. ábra. Onboarding szekvencia diagram

Az 5.2. ábra szemlélteti az onboarding folyamatot. Miután a felhasználó bejelentkezett a Google fiókjával, új felhasználójává válik és ki kell töltenie egy onboarding formot, a fitnesz adataival. Amiután megvannak a fitness adatok és a Google-tól lekért adatok is, akkor az onboarding folyamat átküldi őket az adatbázisba, és akkor az új felhasználó, egy már létező felhasználó lesz.

Szekvencia diagram



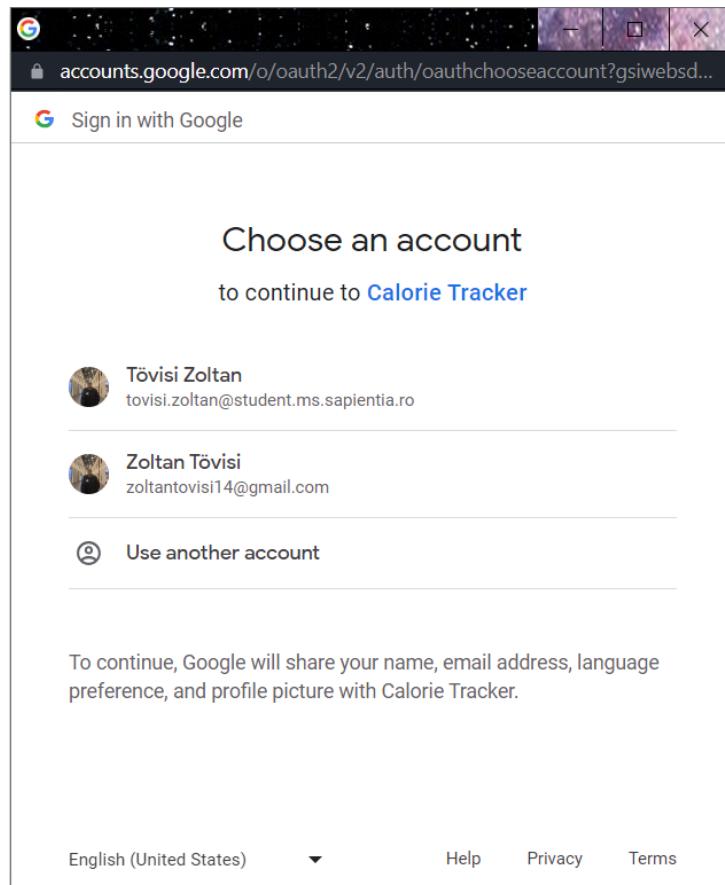
5.3. ábra. FoodEntry hozzáadás szekvencia diagram

Az 5.3. ábra szemlélteti egy FoodEntry hozzáadását. A user kitölt egy kis formot az entry adataival, az applikáció pedig ezekből létrehoz egy új food entry-t és elküldi az adatbázisnak.

6. fejezet

Alkalmazás áttekintése

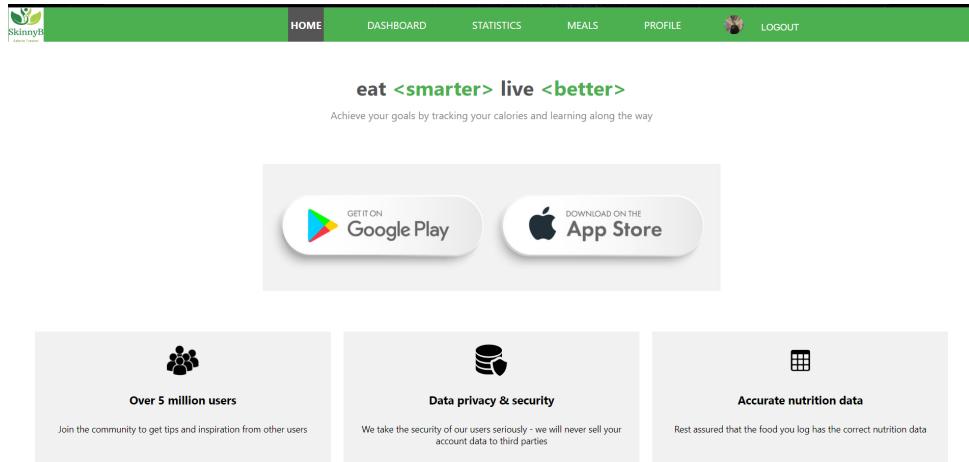
6.0.1. Login



6.1. ábra. Login

Amint ezelőtt is említettem, az alkalmazásomban egy sima Google login-nal lehet bejelentkezni, a 6.1. ábrán láthatjuk, hogy mi jelenik meg, amikor megnyomjuk a Login gombot.

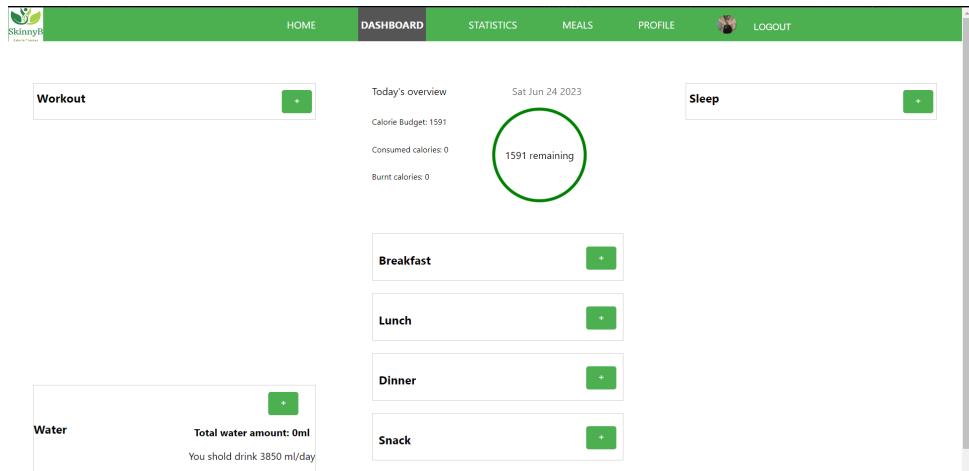
6.0.2. Home page



6.2. ábra. Home page

Miután bejelentkeztünk, a Home vagy landing page-re kerülünk. Itt látható az applikáció mottója, ami: eat smarter, live better. Ezt követően pedig pár hasznos információ, mint például, hogy a felhasználók adatai biztonságban vannak, és az ételek kalóriászáma, amit az applikáció számol ki az elfogyasztott mennyiség alapján, helyesek.

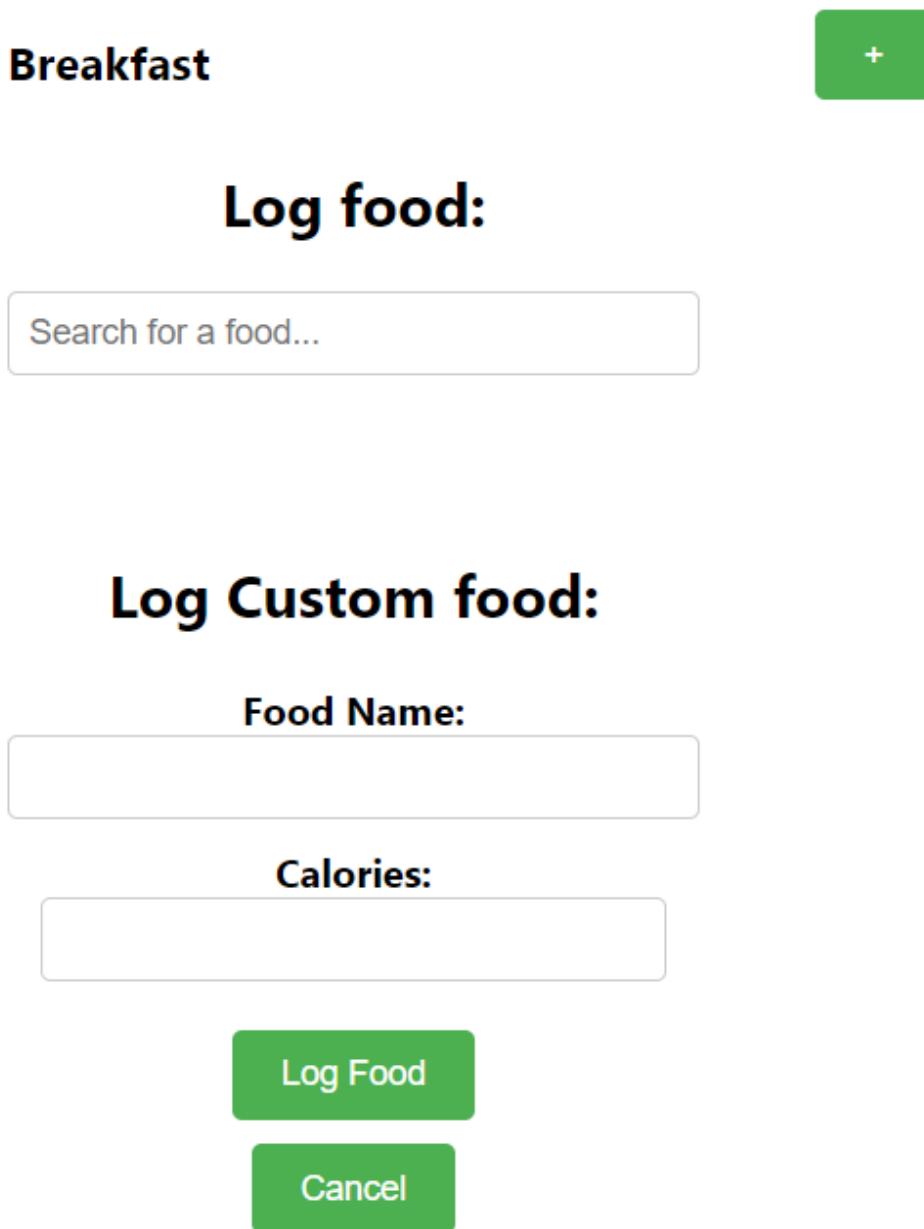
6.0.3. Dashboard



6.3. ábra. Dashboard

Ha a Dashboard-ra navigálunk, ott történik az ételek, aktivitások, víz és alvás naplózása. Ugyanott megtekinthető, hogy mennyi a maximális kalóriabevitele az adott felhasználónak, hogy mennyit fogyasztott el és mennyi maradt. Az ételek 4 szekcióba vannak felosztva: reggeli, ebéd, vacsora, nasi. Ahova bekerül egy étel, az ott fog megjelenni majd, így a nap végén a felhasználó megnézheti, hogy miből mennyit evett aznap. Természetesen a víznél is ki van írva, hogy mennyit kéne megigyon, és ott is megjelenik, hogy az adott nap mennyit ivott meg.

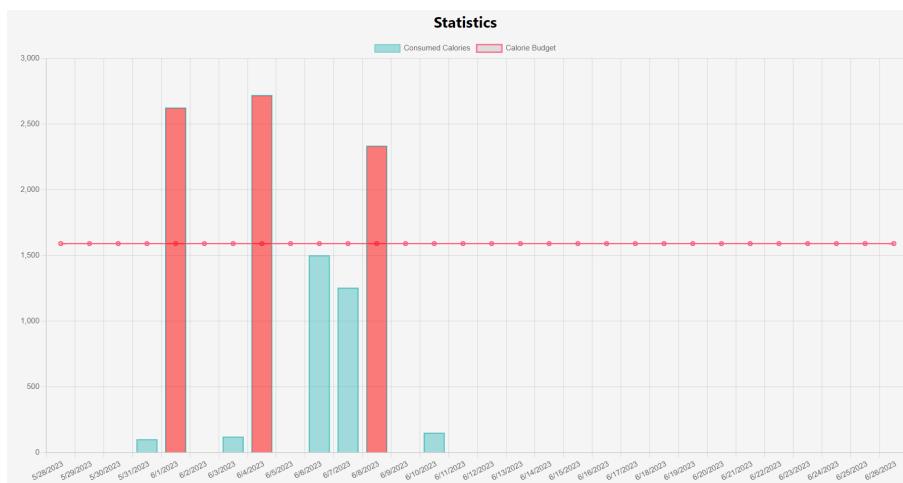
6.0.4. FoodEntry hozzáadás



6.4. ábra. FoodEntry hozzáadás

Ha egy ételt be akar naplózni, akkor az fog megjelenni, amit a 6.4. ábrán látunk. Az első részben, rá tud keresni egy adott ételre, megnyomja a gombot az adott ételnek, beírja, hogy hány grammot evett belőle, és ezáltal be lesz naplózva az adott étel. Ha nincs meg az adatbázisban az adott étel, akkor saját ételt is be lehet naplózni, oda akkor a nevét és kalóriaszámát kell beírni és a Log Food gomb megnyomásával be lesz naplózva az is.

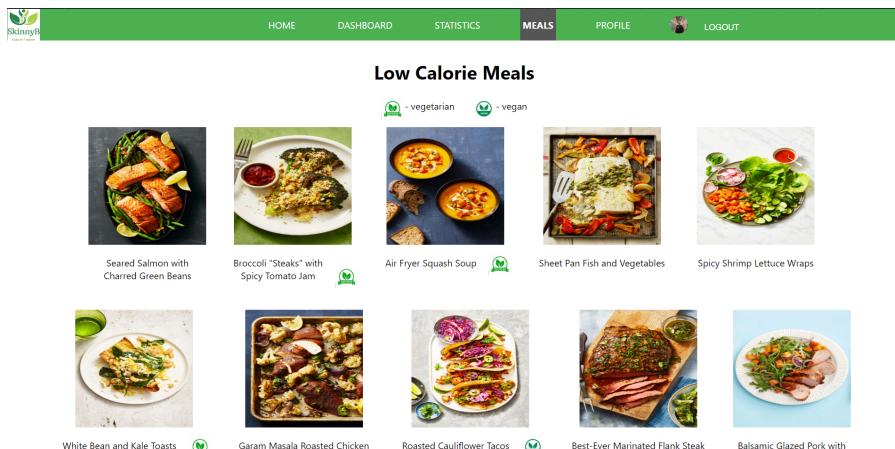
6.0.5. Statisztikák



6.5. ábra. Statisztikák

Ha a Statistics page-re navigálunk, ott megtekinthetőek az elmúlt 30 nap statisztikái. A felhasználó megtekintheti ott, hogy az elmúlt 30 napban hányszor haladta meg a maximális kalóriabevitelét. Egy piros vonalat lehet látni, ami oda van húzva. ahol a maximális kalóriabevitel száma van, és lentről, minden napnál, ahol ételek voltak benaplózva, egy oszlop jelenik meg. Ha nem haladja meg a vonalat, kék színű, ha pedig meg van haladva a vonal, azaz a maximális kalóriabevitel, akkor pirossá változik.

6.0.6. Receptek



6.6. ábra. Receptek

Nem mindig tudjuk, hogyan étkezzünk egészségesen, és a kalóriaszám is gondot szokott okozni. Ezért, ha a felhasználóim a Meals page-re navigálnak, akkor recepteket ajánlok neki, amelyek megfelelnek az említett kritériumoknak.

The screenshot shows a user profile page from the SkinnyB website. At the top, there's a navigation bar with links for HOME, DASHBOARD, STATISTICS, MEALS, PROFILE (which is highlighted in dark grey), and LOGOUT. To the right of the navigation is a small user icon and the word 'LOGOUT'. The main content area has a light grey background and features a heading 'User Profile' in bold black font. Below the heading is a small thumbnail image of a person. Underneath the thumbnail, several user details are listed in bold black text: Name: Zoltan, Email: zoltantovisi14@gmail.com, Height: 187, Start Weight: 120, Goal Weight: 90, Goal Date: 2023.09.01, Activity Level: moderate, and Calorie Budget: 1591. At the bottom left of the content area is a green button labeled 'Edit Profile'.

6.7. ábra. Profil

6.0.7. Profil

A Profile page-re navigálva pedig, a felhasználó megtekintheti a saját adatait, illetve tudja ezeket szerkeszteni is.

7. fejezet

Összefoglaló

Diplomadolgozatom keretén belül, egy kalóriakövető applikációt hoztam létre, mivel ez nagy segítség lehet azoknak, akik le akarnak fogyni, de még azoknak is, akik ugyanannál a testsúlynál akarnak maradni. A fogyásra, az egyetlen lehetőség, a kalória deficitben való lét. Tehát, ha kevesebbet viszünk be, mint amennyit égetünk. Tehát ilyen célból az emberek meg kell tudják, hogy ők hány kalóriát kéne minden nap megegyenek, illetve számolniuk kéne ezeknek a számát. Ezért egy ilyen applikáció segít ezt könnyebbé tenni.

Az applikációm lehetőséget biztosít arra, hogy a felhasználók naplózzák a napi elfogyasztott kalóriájuknak a számát, az elfogyasztott víz mennyiségét, és a fizikai aktivitásukat, amely esetében növekedni fog a maximális kalóriabevitelnek a száma, mivel fizikai aktivitással kalóriákat égetünk. Ezen kívül még a napi megivott vizet és az éjszakai alvást is lehet naplózni, mivel ezek is részesei az egészséges életmódról és fogyásnak. Az applikáció megmondja a felhasználónak, hogy mennyi vizet kéne innia.

Azon kívül, még úgy is segítek a felhasználóknak, hogy recepteket biztosítok, amelyek kalóriaszegények. Arra is oda figyelek, hogy vegetáriánus, illetve vegán opciókat is biztosítunk, mivel jómagam is vegetáriánus vagyok, és az a tapasztalom, hogy nagyon sokszor megfelelkeznek rólunk vendéglőkben, rendezvényeken stb.

Továbbá, statisztikákat is megtekinthetnek a felhasználók, hogy az utóbbi 30 napban hányszor haladták meg a maximális kalóriabevitelüket. Ezt egy oszlopdiagram segítségével szemléltetem, ahol minden oszlop egy napon elfogyasztott kalóriák számát jelenti, ha meg van haladva a maximális kalóriabevitel, akkor az oszlop piros, ellenkező esetben kék.

Jövőbeli terveimet illetően, a következő továbbfejlesztési lehetőségekre gondoltam:

- Lehetőség összekötni okos órával
- Egy forum/blog hozzáadása, ahol a felhasználók, ahol megoszthatják a fejlődésüket, teljesítményeiket a barátaikkal
- Az applikáció 'gamify'-olása
- Személyre szabott étrendek, illetve edzés tervezés biztosítása

- AI használata, ami megjósolja a lehetséges kalóriaszámát egy ételnek, egy feltöltött kép alapján
- Review az ételekről, amiket elfogyaszt a felhasználó
- Ajánlott makrok bevitelé és minden étel makróinak a feltüntetése

Az applikáció a következő GitHub link-en érhető el:
<https://github.com/nephilim27/Final-Exam.git> vagy [ide](#) kattintva.

Köszönetnyilvánítás

Ezúttal meg szeretném köszönni mindenazonaknak, akik hozzájárultak a dolgozatom és applikáció elkészítéséhez. Első sorban köszönöm dr. Jánosi-Rancz Katalin Tündének, a vezető tanárom és nemcsak. Köszönöm, hogy segített a dokumentációval és hasznos tanácsokat adott, konstruktívan kritizálta az applikációt, ami segített azt jobbá tenni.

Köszönöm mentoraimnak az .msg systems cégtől, Szász Zoltánnak, aki a backend részen, és Baczkó Miklósnak, aki a frontend részen segített. Úgy érzem nélkülük nem fejezem volna be időben, mivel ahányszor elakadtam kisegítettek, tanácsokat adtak és vezéreltek, hogy egy szép, modern applikációt hozzak létre. Még egyszer köszönöm nekik.

Végül, de nem utolsó sorban, köszönöm Dumitrescu Helgának, az .msg systems cég site manager-ének, akinek a mentoraimat köszönhetem, ugyanis általa kerülttem kapcsolatba velük, ő tette ezt lehetővé. Még egyszer, mindenkinél köszönöm a segítséget!

Ábrák jegyzéke

| | |
|--|----|
| 2.1. Spring Boot példakód | 13 |
| 2.2. JPA Repository | 14 |
| 2.3. React példakód | 15 |
| 2.4. React routing | 15 |
| 2.5. SQL példakód | 16 |
| 2.6. Hibernate konfigurálása | 16 |
| 2.7. Generált script létrehozásra | 17 |
| 2.8. Generált script törlésre | 17 |
| 3.1. Use case diagram | 19 |
| 4.1. Architektura diagram | 24 |
| 4.2. Adatbázis diagram | 25 |
| 4.3. Verzió követés | 26 |
| 5.1. Bejelentkezés szekvencia diagram | 27 |
| 5.2. Onboarding szekvencia diagram | 28 |
| 5.3. FoodEntry hozzáadás szekvencia diagram | 29 |
| 6.1. Login | 30 |
| 6.2. Home page | 31 |
| 6.3. Dashboard | 31 |
| 6.4. FoodEntry hozzáadás | 32 |
| 6.5. Statisztikák | 33 |
| 6.6. Receptek | 33 |
| 6.7. Profil | 34 |
| F.1.1 Az Overleaf L ^A T _E X-szerkesztő | 41 |

Táblázatok jegyzéke

| | |
|--|----|
| 3.1. Sign-in Use Case | 20 |
| 3.2. Onboarding Use Case | 20 |
| 3.3. Alvás Mennyiségi Kezelés Use Case | 21 |
| 3.4. Workout Kezelés Use Case | 21 |
| 3.5. Víz Bevitel Kezelés Use Case | 22 |
| 3.6. Étel Bevitel Kezelés Use Case | 22 |
| 3.7. Profil szerkesztés Use Case | 23 |

Irodalomjegyzék

[jav21a] javaTpoint. Learn ReactJS Tutorial. Online. Available: <https://www.javatpoint.com/reactjs-tutorial>, 2021.

[jav21b] javaTpoint. Spring Boot CRUD Operations. Online. Available: <https://www.javatpoint.com/spring-boot-crud-operations>, 2021.

[jav21c] javaTpoint. Spring Boot Tutorial. Online. Available: <https://www.javatpoint.com/spring-boot-tutorial>, 2021.

Függelék

F.1. Az Overleaf felülete

The screenshot shows the Overleaf LaTeX editor interface. On the left, the file tree displays various files including `content`, `images`, `abstract.tex`, `acknowledgement.tex`, `appendices.tex`, `bevezeto.tex`, `fejezet1.tex`, `fejezet2.tex`, `fejezet3.tex`, `fejezet4.tex`, `fejezet5.tex`, `main.tex` (selected), and `osszefoglalo.tex`. A message indicates that no sections or subsections are found in this file. Below the file tree is a "File outline" section with a link to "Find out more about the file outline".

The central "Code Editor" window contains the LaTeX code for the thesis. The code includes document class definitions, input packages, language configuration, main variables, and various commands for sections, authors, and advisors. Lines 22 through 24 define the thesis title and type: `\newcommand{\dolgozattipusH}{DIPLOMADOLGOZAT}`, `\newcommand{\dolgozattipusR}{LUCRARE DE DIPLOMĂ}`, and `\newcommand{\dolgozattipusN}{BACHELOR THESIS}`.

The right side of the interface is the "Preview" panel, which shows the final document layout. It features the title "SAPIENTIA ERDÉNYI MAGYAR TUDOMÁNYEGYETEM" and "MAJOSVÁSÁTHELYI KAR, INFORMATIKA SZAK", the logo of "SAPIENTIA ERDÉNYI MAGYAR TUDOMÁNYEGYETEM", and the subtitle "INFORMATICA". The preview also includes the names of the advisor and student: "Témavezető: dr. Jánosi-Rancz Katalin Tünde, Egyetemi adjunktus" and "Végző hallgató: Tóvis Zoltán", along with the year "2023".

F.1.1. ábra. Az Overleaf L^AT_EX-szerkesztő.