***LifeStyle***

Fejlesztői dokumentáció

Szoftverfelesztő és –tesztelő technikus

2022

Tartalomjegyzék

[1. Bevezetés, a téma ismertetése, témaválasztás indoklása: 2](#_Toc101952487)

[Fejlesztői dokumentáció 2](#_Toc101952488)

[2. Felhasznált technológiák 2](#_Toc101952489)

[1.1 Kódolási konvenciók 3](#_Toc101952490)

[1.2 Könyvtárszerkezet 3](#_Toc101952491)

[3. Backend (Laravel) 4](#_Toc101952492)

[1.3 Végpontok 4](#_Toc101952493)

[1.4 Osztályok 5](#_Toc101952494)

[**1.4.1** BaseController 5](#_Toc101952495)

[1.4.2 AuthController 6](#_Toc101952496)

[1.4.3 UserController 6](#_Toc101952497)

[1.4.4 DataController 7](#_Toc101952498)

[1.4.5 MealController 8](#_Toc101952499)

[1.4.6 User 8](#_Toc101952500)

[1.4.7 Data 8](#_Toc101952501)

[1.4.8 Meal 9](#_Toc101952502)

[1.4.9 User 9](#_Toc101952503)

[4. Tesztelés 9](#_Toc101952504)

[1.5 Backend Authentication 9](#_Toc101952505)

[1.5.1 UserTest 12](#_Toc101952506)

[1.5.2 Data 15](#_Toc101952507)

[1.5.3 Meals 17](#_Toc101952508)

[5. Ismert hibák a kódban 19](#_Toc101952511)

[6. Asztali alkalmazás 20](#_Toc101952512)

[7. Osztályok 20](#_Toc101952513)

[1.6 LifeStyle 20](#_Toc101952514)

[1.7 Controller 20](#_Toc101952515)

[1.8 ViewController 20](#_Toc101952516)

[1.9 RapiController 22](#_Toc101952517)

[1.10 DataModel 24](#_Toc101952518)

[1.11 MealModel 24](#_Toc101952519)

[1.12 UserModel 24](#_Toc101952520)

[1.13 RapiModel 24](#_Toc101952521)

[1.14 ViewModel 26](#_Toc101952522)

[8. Webes felület 26](#_Toc101952523)

[1.15 Indítás 26](#_Toc101952524)

[1.16 Megjegyzés 27](#_Toc101952525)

[9. Felhasználása 27](#_Toc101952526)

[10. Frontend mappaszerkezet 28](#_Toc101952527)

[11. Komponensek 29](#_Toc101952528)

[1.17 BMi komponens 29](#_Toc101952529)

[1.18 Home komponens 29](#_Toc101952530)

[1.19 Login komponens 29](#_Toc101952531)

[1.20 Meal komponens 29](#_Toc101952532)

[1.21 Register komponens 30](#_Toc101952533)

[1.22 About komponens 30](#_Toc101952534)

[12. Felhasználói dokumentáció 30](#_Toc101952535)

[1.23 Webes felület „beüzemelése”: 30](#_Toc101952536)

# Bevezetés, a téma ismertetése, témaválasztás indoklása:

A projekt célja egy olyan felhasználóbarát alkalmazás létrehozása amely segít az embereknek, hogy egy rendezettebb és egészségesebb életet élhessenek. Különböző kalkulátorokon keresztül tudjuk monitorozni, hogy éppen, hogy is állunk a követendő célunk elérésével. Illetve lehet benne saját felhasználót létrehozni a még egyedibb és konkrétabb és pontosabb, illetve, hogy lehessen tárolni a saját adatokat (bmi adatok bevitt víz mennyiség stb). Nagyon szerettünk volna egy olyan appot ami kicsit összeszdettebb mint a többi és nem kell mindig máshova átlépni ha mondjuk egy BMI-t szeretnénk kiszámolni, hanem minden egy helyen van.

# Fejlesztői dokumentáció

# Felhasznált technológiák

Backend

* + Laravel API ( PHP keretrendszer )
  + Laravel - sanctum ( Autentikációs kiegészítő csomag )
  + Mariadb server ( Adatbázis kiszolgáló )

Frontend

* + Angular
  + Bootstrap
  + CSS
  + XAMP

Asztali alkalmazás

* + Eclipse
  + Mariadb server (Adatbázis kiszolgáló)

Készítéshez használt programok

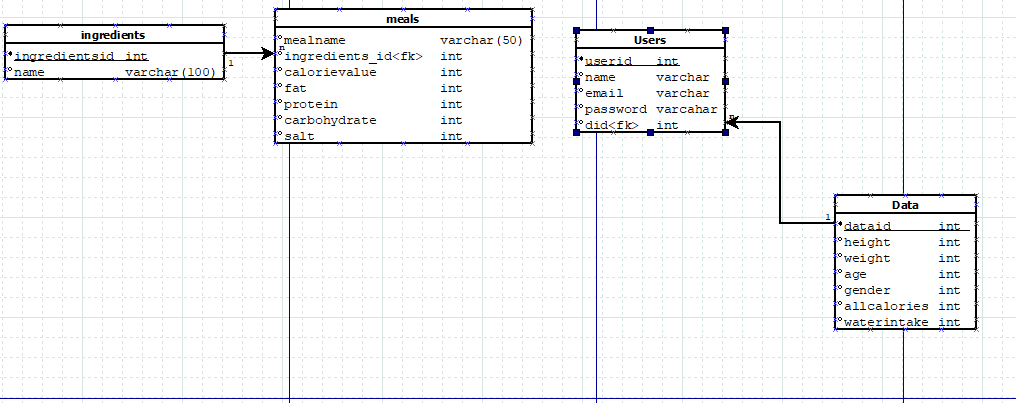
* + dia
  + Visual studio code 1.63.0
  + Isomnia 2021.5.3
  + Mariadb server
  + phpmyadmin

## Kódolási konvenciók

A kódot git verziókezelővel használjuk.

## Könyvtárszerkezet

* + adatbázis
  + api
  + asztali
  + doc
  + mobil
  + web

FelületterveK

# Backend (Laravel)

## Végpontok

|  |  |  |  |
| --- | --- | --- | --- |
| **Végpont** | **Metódus** | **Azonosítás** | **Leírás** |
| /register | post | nem | Felhasználó regisztrációja |
| /login | post | nem | Felhasználó bejelentkezése |
| /logout | post | igen | Felhasználó kijelentkezése |
| **USER** | | | |
| /show-user{id} | get | igen | Egy felhasználó lekérdezése id alapján |
| /search-user{id} | get | igen | Egy felhasználó lekérdezése név alapján |
| /show-all-user | get | igen | Az összes felhasználó lekérdezése |
| /update-user{id} | put | igen | Felhasználó adatainak módosítása |
| /delete-user{id} | delete | igen | Felhasználó törlése |
| **DATA** | | | |
| /store-data | post | igen | Felhasználó személyes adatainak létrehozása |
| /show-data{id} | get | igen | Felhasználó adatainak megjelenítése |
| /update-data{id} | put | igen | Felhaszáló adatainak módosítása |
| /delete-data{id} | delete | igen | Felhasználó adatainak törlése |
| **MEAL** | | | |
| /store-meal | post | igen | Új étel felvétele |
| /show-meal{id} | get | igen | Étel lekérdezése id alapján |
| /search-meal/{name} | get | igen | Étel lekérdezése Név alapján |
| /show-all-meal | get | igen | Összes étel lekérdezése |
| /update-meal/{id} | put | igen | Étel adatainak módosítása |
| /delete-meal/{id} | delete | igen | Étel törlése id alapján |

## Osztályok

### BaseController

Feladata a bejelentkezési műveletek válaszának küldése. Sikeres művelete esetén a generált azonosító tokent visszaküldi, sikertelen művelete esetén hibaüzenetet küld.

Metódusok:

SendResponse()

-bejövő paraméterek: $result (A generált token), $message (A felhasználónak szánt üzenet).

-kimenő adatok: Response (Ha a bejelentkezés sikeres token).

SendError()

-bejövő paraméterek: $error (A PHP által generált hibaüzenet), $errorMessage (Saját hibaüzenet), $code (A válaszban küldendő PHP kód).

-kimenő adatok: Response (Sikertelen bejelentkezés esetén hibaüzenet és a hiba kódja).

### AuthController

Feladata az új felhasználók felvétele, autentikáció, felhasználók kijelentkeztetése.

Metódusok:

signup()

-bejövő paraméterek: $request (A regisztációhoz szükséges adatok lérése name, email, password, confirm\_password).

Adatok validálása, majd sikeres validálás után bejegyzi a users adatbázis tábla megfelelő mezőibe.

-kimenő adatok: Saját üzenet.

signin()

Feladata a felhasználók azonosítása email és jelszó alapján. Sikeres autentikáció esetén generál egy tokent a felhasználó számára és bejegyzi a personal\_access\_token adatbázis tábla megfelelő mezőibe, majd átadja a BaseController sendResponse() metódusának a saját üzenettel együtt.

-bejövő paraméterek: $request(a bejelentkezéshez szükséges adatok a kérésben, email, password).

-kimenő adatok: token, saját üzenet.

logout()

Feladata a felhasználók kijelentkeztetése és a token törlése a táblából.

-bejövő paraméterek: $request (A kijelentkezéshez szükséges adatok a kérésben, token).

-kimenő adatok: saját üzenet.

### UserController

Feladata a meglévő felhasználó adatainak módosítása, meglévő felhasználó keresése, meglévő felhasználó törlése.

Metódusok:

show()

Visszaadja egy kiválasztott felhasználó adatait.

- bejövő paraméterek: $id (A kiválasztott felhasználó azonosítója)

- kimenő adatok: User (A kiválasztott felhasználó adatait tartalmazó modell).

showAll()

Visszaadja az összes felhasználó adatait.

- kimenő adatok: User (A kiválasztott felhasználó adatait tartalmazó modell).

update()

Frissíti az adatbázisban tárolt egy felhasználó adatait.

- bejövő paraméterek: $request, $id (A felhasználó adatait tartalmazó kérés és a felhasználó azonosítója).

- kimenő adatok: User (A Felhasználó adatait tartalmazó modell).

search()

Név alapján kikeres egy vagy több felhasználót az adatbázisból.

- bejövő paraméterek: $name (A felhasználó neve).

- kimenő adatok: User (Az egyezést mutató felhasználók adatait tartalmazó modell).

destroy()

Feladata a kiválasztott felhasználó törlése az adatbázisból.

- bejövő paraméterek: $id (A kiválasztott felhasználó azonosítója).

- kimenő adatok: User (A kiválasztott felhasználó adatait tartalmazó modell).

### DataController

Feladata a meglévő felhasználó adatainak módosítása, meglévő felhasználó adatainak keresése, meglévő felhasználó adatainak törlése.

Metódusok:

storeData()

Eltárolja az adatbázisban a felhasználó adatait.

- bejövő paraméterek: request (A felhasználó adatait tartalmazó kérés).

- kimenő adatok: Data (A felhasználó adatait tartalmazó modell).

show()

Visszaadja egy kiválasztott felhasználó adatait.

- bejövő paraméterek: $id (A kiválasztott felhasználó azonosítója)

- kimenő adatok: Data (A kiválasztott felhasználó adatait tartalmazó modell).

showAll()

Visszaadja az összes felhasználó adatait.

- kimenő adatok: Data (A kiválasztott felhasználó adatait tartalmazó modell).

update()

Frissíti az adatbázisban tárolt egy felhasználó adatait.

- bejövő paraméterek: $request, $id (A felhasználó adatait tartalmazó kérés és a felhasználó azonosítója).

- kimenő adatok: Data (A Felhasználó adatait tartalmazó modell).

destroy()

Feladata a kiválasztott felhasználó adatainak törlése az adatbázisból.

- bejövő paraméterek: $id (A kiválasztott felhasználó adatának azonosítója).

- kimenő adatok: Data (A kiválasztott felhasználó adatait tartalmazó modell).

### MealController

Feladata új étel létrehozása, a meglévő étel adatainak módosítása, meglévő étel keresése, meglévő étel törlése.

Metódusok:

store()

Eltárolja az adatbázisban a étel adatait.

- bejövő paraméterek: request (Az étel adatait tartalmazó kérés).

- kimenő adatok: Meal (A felhasználó adatait tartalmazó modell).

show()

Visszaadja egy kiválasztott étel adatait.

- bejövő paraméterek: $id (A kiválasztott étel azonosítója)

- kimenő adatok: Meal (A kiválasztott étel adatait tartalmazó modell).

update()

Frissíti az adatbázisban tárolt egy étel adatait.

- bejövő paraméterek: $request, $id (Az étel adatait tartalmazó kérés és a felhasználó azonosítója).

- kimenő adatok: Meal (Az étel adatait tartalmazó modell).

search()

Név alapján kikeres egy vagy több ételt az adatbázisból.

- bejövő paraméterek: $name (Az étel neve).

- kimenő adatok: Meal (Az egyezést mutató ételek adatait tartalmazó modell).

showAll()

Visszaadja az összes étel adatait.

- kimenő adatok: Meal (A kiválasztott étel adatait tartalmazó modell).

destroy()

Feladata a kiválasztott étel törlése az adatbázisból.

- bejövő paraméterek: $id (A kiválasztott étel azonosítója).

- kimenő adatok: Meal (A kiválasztott étel adatait tartalmazó modell).

### User

A regisztrált felhasználók adatait, illetve az azonosító tokent amelynek a generálása a felhasználóhoz kapcsolódik írja az adatbázis megfelelő táblájába, valamint onnan szolgáltatja szükség szerint. A User modell a users adatbázis táblával van kapcsolatban az adatokat onnan olvassa és oda írja. Az osztályt a Laravel projekt generálja, használata a Sanctum autentikációs kiegészítő csomagon keresztül történik.

### Data

Az adatbázisban felvett felhasználók adatait kezeli, a DataController megfelelő metódusai által kért adatot adja vissza az adatbázis megfelelő táblájából, illetve a kontrollerből érkező adatokat írja ki. Az adatbázis datas táblájával van kapcsolatban, az adatokat onnan olvassa és oda írja.

Mezők:

$fillable (array) elemei: height, weight, age, gender, allcalories, waterintake

A fillable mező elemei egy – egy mező a datas adatbázis táblában. Biztonsági okokból csak a felsorolt mezőket írhatja a modell az adatbázisban.

### Meal

Az adatbázisban felvett étele adatait kezeli, a MealController megfelelő metódusai által kért adatot adja vissza az adatbázis megfelelő táblájából, illetve a kontrollerből érkező adatokat írja ki. Az adatbázis meals táblájával van kapcsolatban, az adatokat onnan olvassa és oda írja.

Mezők:

$fillable (array) elemei: name, calorievalue, fat, protein, carbohydrate, salt

A fillable mező elemei egy – egy mező a meals adatbázis táblában. Biztonsági okokból csak a felsorolt mezőket írhatja a modell az adatbázisban.

### User

Az adatbázisban felvett felhasználók adatait kezeli, a UserController megfelelő metódusai által kért adatot adja vissza az adatbázis megfelelő táblájából, illetve a kontrollerből érkező adatokat írja ki. Az adatbázis users táblájával van kapcsolatban, az adatokat onnan olvassa és oda írja.

Mezők:

$fillable (array) elemei: name, email, password, data\_id,

A fillable mező elemei egy – egy mező a users adatbázis táblában. Biztonsági okokból csak a felsorolt mezőket írhatja a modell az adatbázisban.

# Tesztelés

## Backend Authentication

Register – Jó adatokkal

A képen szöveg, képernyőkép, beltéri, fekete látható

Automatikusan generált leírás

Register – Nem megegyező jelszóval

A képen szöveg látható

Automatikusan generált leírás

Login – Jó adatokkal

A képen szöveg, képernyőkép, képernyő látható

Automatikusan generált leírás

Login – Hibás jelszó

A képen szöveg, monitor, képernyőkép, képernyő látható

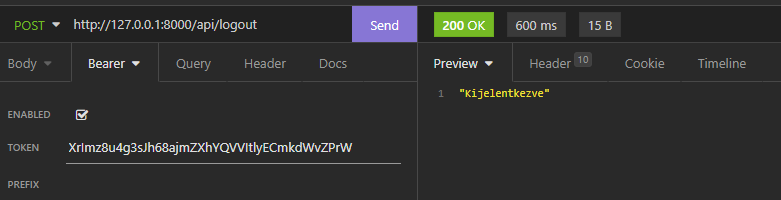
Automatikusan generált leírás

Login – hibás email-cím

A képen szöveg, képernyőkép, monitor, képernyő látható

Automatikusan generált leírás

Logout



### UserTest

show-user-létező id

A képen szöveg látható

Automatikusan generált leírás

show-user – nem létező id

A képen szöveg, képernyőkép, monitor, képernyő látható

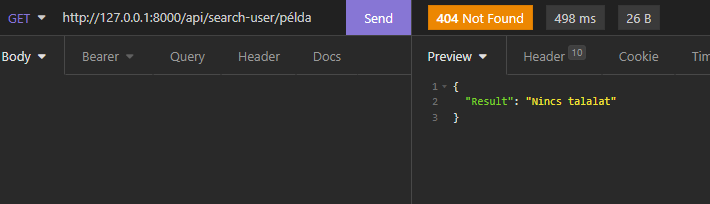
Automatikusan generált leírás

search-user – létező név

A képen szöveg, monitor, képernyő, képernyőkép látható

Automatikusan generált leírás

search-user – nem létező név



show-all-user – admin joggal

show-all-user – admin jog nélkül

A képen szöveg, képernyőkép, monitor, képernyő látható

Automatikusan generált leírás

update-user – adminként

A képen szöveg, monitor, képernyőkép, képernyő látható

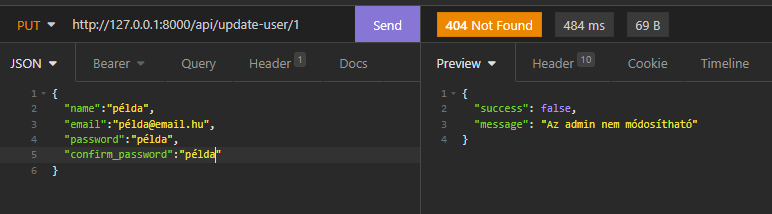
Automatikusan generált leírás

update-user - nem saját felhasználó admin jog nélkül

A képen szöveg, képernyőkép, monitor, képernyő látható

Automatikusan generált leírás

update-user – admin módosítása



delete-user – adminként

A képen szöveg látható

Automatikusan generált leírás

delete user – saját fiók, admin jog nélkül

A képen szöveg, monitor, képernyőkép, képernyő látható

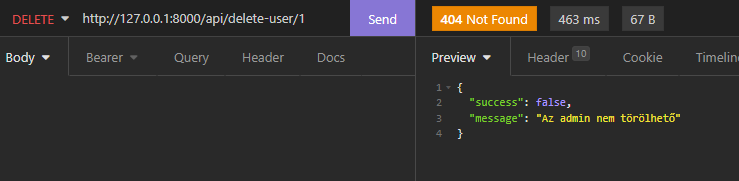
Automatikusan generált leírás

delete user – más fiók, admin jog nélkül

A képen szöveg, képernyőkép, képernyő, bezárás látható

Automatikusan generált leírás

delete user – admin felhasználó



### Data

store-data

A képen szöveg, monitor, képernyőkép, képernyő látható

Automatikusan generált leírás

show-data

A képen szöveg látható

Automatikusan generált leírás

update-data

A képen szöveg, monitor, képernyőkép látható

Automatikusan generált leírás

delete-data

A képen szöveg, monitor, fekete, képernyőkép látható

Automatikusan generált leírás

### Meals

store-meal

A képen szöveg, monitor, képernyőkép, fekete látható

Automatikusan generált leírás

show-meal

A képen szöveg, monitor, képernyőkép, fekete látható

Automatikusan generált leírás

show-all-meal

A képen szöveg látható

Automatikusan generált leírás

search-meal

A képen szöveg, monitor, képernyőkép, fekete látható

Automatikusan generált leírás

update-meal

A képen szöveg, monitor, képernyőkép, képernyő látható

Automatikusan generált leírás

delete-meal

A képen szöveg látható

Automatikusan generált leírás

# 

# 

# Asztali alkalmazás

Az asztali alkalmazást alapvetően egy beleégetett admin felhasználónak van tervezve aki tud felhasználót/ételt törölni módosítani illetve megtudja nézni a felhasználók adatait (magassság súly nem kalóriaérték stb.).

# Osztályok

## LifeStyle

Ez a főosztály az asztali alkalmazásban melynek feladata elindítani az asztali alkalmazást és példányosítja a Controller osztályt.

## Controller

Megpéldányosítja a másik kettő (RapiControllert, ViewControllert ) és átadjuk neki a RapiController új példányát.

## ViewController

Az alkalmazás asztali részét működteti és kontrollálja, illetve minden olyan adatot megjelenít amit meg szeretnénk jeleníteni vele.

Osztályváltozók:

* LifeForm lifeFrm;
* ViewModel viewMdl;
* Vector<Vector<Object>> tableData;
* RapiController rapiCtr;
* String id;

ViewController ()

fogadott paraméter: rapiCtr RapiController

megpéldányosítja a ViewModel osztályt.

ActionListeners()

Alkalmazza az „action” illet „change”Listenereket a függvényeken.

initFrames()

Betölti a 2 alkalmazott ablakot illetve a hozzájuk tartozó táblázatot.

initLifeFrame()

hozzárendeli a Life(fő) ablak tulajdonságait és adatait

initTables()

feltölti a Life ablakhoz tartozó táblázatot adatokkal.

delete ()

elindítja a megfelelő törlésre megírt függvényt attól függően hogy éppen melyik táblát látjuk.

DeleteMeal()

Az általunk kiválasztott sor törlésre kerül, alul a Labelben visszajelzést kapunk a sikeres törlésünkről és újra betölti a Meal táblázatot.

DeleteUser()

Az általunk kiválasztott sor törlésre kerül, alul a Labelben visszajelzést kapunk a sikeres törlésünkről és újra betölti a User táblázatot.

update()

Elindítja a megfelelő módosításra megírt függvényt attól függően hogy éppen melyik táblát látjuk.

UpdateUser()

Az általunk kiválasztott sorban lévő módosításokat megvalósítja majd a Labelben visszajelzést kapunk a sikeres módosításunkról és újra betölti a User táblázatot.

UpdateMeal()

az általunk kiválasztott sorban lévő módosításokat megvalósítja majd a Labelben visszajelzést kapunk a sikeres módosításunkról és újra betölti a Meal táblázatot.

ShowData ()

Az általunk kiválasztott sorban lévő felhasználónak megmutatja a hozzá tartozó adatit és beletölti egy Data táblázatba.

exit()

Kijelentkezik majd bezárja az asztali alkalmazást.

## RapiController

Modellektől átveszi az adatokat és továbbítja a ViewControllernek vagy a RapiControllernek.

Változók:

RapiModel rapiMdl;

String token;

String method;

String id;

RapiController()

Megpéldányosítja a RapiModel osztályt, elmenti a tokent amivel beléptünk.

getToken()

Kinyeri a RestModelből a tokent belépést követően

kimenő adatok: token;

Logout ()

Rapimodelből meghívja a „Logout” függvényt és átadja neki paraméterként a tokent.

getUsers()

Rapimodelből meghívja a „User” függvényt és átadja neki paraméterként a tokent.

getData()

Rapimodelből meghívja a „Data” függvényt és átadja neki paraméterként a tokent.

getMeal()

Rapimodelből meghívja a „Meal” függvényt és átadja neki paraméterként a tokent.

DeleteMeal ()

Rapimodelből meghívja a „DeleteMeal” függvényt és átadja neki paraméterként a tokent és az „id-t”.

DeleteUser()

Rapimodelből meghívja a „DeleteUser” függvényt és átadja neki paraméterként a tokent és az „id-t”.

setId()

Bejövő paraméter id;

összevonja a paraméterként érkező id-t és a létrehozott osztályváltozó id-t.

UpdateUser()

Rapimodelből meghívja a „UpdateUser” függvényt és átadja neki paraméterként a tokent és az id-t.

UpdateMeal()

Rapimodelből meghívja a „UpdateMeal” függvényt és átadja neki paraméterként a tokent és az id-t.

## DataModel

Sablonként funkcionál a RapiModel osztály számára.

## MealModel

Sablonként funkcionál a RapiModel osztály számára.

## UserModel

Sablonként funkcionál a RapiModel osztály számára.

## RapiModel

* String DeleteMealMsg;
* String DeleteUserMsg;
* ResponseModel responseMdl;

tryLogin()

Megpróbál bejelentkezni az alkalmazásba és a megadott tokent beolvassa.

Login()

Lefuttatja a „TryLogin()” függvényt.

tryUsers()

a Backendnek elküldi a kérést és próbálja kiolvasni a felhasználók .adatait

Users()

Lefuttatja a „TryUsers()” függvényt.

tryMeals()

a Backendnek elküldi a kérést és próbálja kiolvasni a kaják adatait.

Meals()

Lefuttatja a „tryMeals()” függvényt.

tryDatas()

a Backendnek elküldi a kérést és próbálja kiolvasni a felhasználók adatainak adatait.

Datas()

Lefuttatja a „tryDatas()” függvényt.

tryLogout()

a Backendnek elküldi a kérést és próbál kijelentkezni.

Logout()

Lefuttatja a „tryLogout()” függvényt és kiírja hogy („Sikeres kijelentkezés”).

tryDeleteMeal()

Elküldi a Backendnek a törlésre szánt recordnak az id-ját és azon keresztül törli az ételt.

DeleteMeal()

Lefuttatja a „tryDeleteMeal()” függvényt és kiírja a backend által megírt tryDeleteUser().

Elküldi a Backendnek a törlésre szánt recordnak az id-ját és azon keresztül törli az ételt.

tryDeleteUser

Elküldi a Backendnek a törlésre szánt recordnak az id-ját és azon keresztül törli a felhasználót.

DeleteUser()

Lefuttatja a „tryDeleteMeal()” függvényt és kiírja a backend által megírt szöveget.

tryUpdateUser()

Elküldi a Backendnek a módosításra szánt recordnak az id-ját és azon keresztül módosítja az felhasználó táblát.

UpdateUser

Lefuttattja a „tryUpdateUser()” függvényt és módosítja a felhasználó adatait

tryUpdateMeal()

Elküldi a Backendnek a módosításra szánt recordnak az id-ját és azon keresztül módosítja az ételek táblát

UpdateMeal()

Lefuttattja a „tryUpdateMeal()” függvényt és módosítja a étel adatait

getDeleteMealMsg()

az étel törlését követően ezzel az üzenettel tér vissza az ablak (label fülén)

getDeleteUserMsg()

a felhasználó törlését követően ezzel az üzenettel tér vissza az ablak (label fülén)

## ViewModel

Ebben az osztályban vannak felvezetve a főablakban lévő táblázatnak az oszlopneveit

getUserColumnNames()

oszlopneveket külön megnevezzük és tömbbe gyűjtjük

GetMealColumnNames()

oszlopneveket külön megnevezzük és tömbbe gyűjtjük

# Webes felület

## Indítás

1. npm install
2. ng serve –o

## Megjegyzés

Amennyiben nincsen a Backend „beélesítve” akkor szükséges a backend mappájába egy .env nevű fájlt készíteni a „.enxexample” fájl mintájára illetve egy ’composer install’ parancsot írni a parancssorsorába.

# Felhasználása

1. Bejelentkezés/Regisztráció

Első alkalommal a bejelentkező felületre jutunk. Ha még nincs fiókunk akkor a regisztráció gombra kattintva jutunk el a kívánt oldalra. Itt meg kell adnunk a nevünket, e-mail címünket és az általunk választott jelszót aminek legalább 5 karakternek kell lennie. Ezután az előző oldalon betudunk jelentkezni az általunk megadott e-mail címmel és jelszóval.

1. Főoldal

A főoldalon tudunk elnavigálni a Kaják, BMI Kalkulátor, Rólunk oldalakon, illetve itt van lehetőség a kijelentkezésre is.

3.Kaják

Idekattintva látjuk kilistázva az ételeinket illetve itt tudunk újakat felvenni az Új hozzáadása segítségével. Itt megtudjuk adni a nevét, kalória értékét, zsír, fehérje, szénhidrát és só tartalmát. A vissza gombbal pedig visszajutunk a főoldalra.

4.BMI kalkulátor

A BMI kalkulátorra lépve pedig kitudjuk számolni a testtömegindexünket a súlyunk és a magasságunk megadásával.

# Frontend mappaszerkezet

• web/

◦ bmi/

▪ src/

• app.js

• index.html

• style.css

◦ kcal\_bevitel/

▪ src/

• app.js

• index.html

• style.css

◦ LifeStyle/

▪ .angular

▪ .vscode

▪ node\_modules

▪ src/

• app/

◦ bmi/

▪ bmi.component.css

▪ bmi.component.html

▪ bmi.component.spec.ts

▪ bmi.component.ts

◦ home/

▪ home.component.css

▪ home.component.html

▪ home.component.spec.ts

▪ home.component.ts

◦ login/

▪ login.component.css

▪ login.component.html

▪ login.component.spec.ts

▪ login.component.ts

◦ meals/

▪ meals.component.css

▪ meals.component.html

▪ meals.component.spec.ts

▪ meals.component.ts

◦ register/

▪ index.ts

▪ register.compnent.css

▪ register.compnent.html

▪ register.compnent.spec.ts

▪ register.compnent.ts

◦ shared/

▪ auth.guard.spec.ts

▪ auth.guard.ts

▪ auth.service.spec.ts

▪ auth.service.ts

▪ meals.service.spec.ts

▪ meals.service.ts

◦ app-routing.module.ts

◦ app.component.css

◦ app.component.html

◦ app.component.spec.ts

◦ app.component.ts

◦ app.module.ts

• assets

• environments

• index.html

# Komponensek

## BMi komponens

bmi.component.ts

calcIndex()

A megadott paraméterek alapján kiszámolja a testtömegindexet.

## Home komponens

[home.component.hmtl](http://home.component.hmtl)

Átjáróként funkciónál a többi komponens számára a könnyebb elérés érdekében.

## Login komponens

login.component.ts

login()

Megkezdődik a felhasználó azonosítása e-mail cím és jelszó alapján. Amennyiben az e-mail cím és a jelszó már létezik az adatbázisban sikeres bejelentkezést követően kapunk egy tokent.

userLogin()

A „bejelentkezés” gomb megnyomását követően indul el egy animáció, és ennek az időtartama van beállítva a függvényben.

## Meal komponens

meals.component.ts

getMeals()

Itt nyerjük ki a backendből az ételek tábla adatait, és töltjük be egy táblázatba.

addMeal()

Az „Új hozzáadása” gomb megnyomása után tudunk felvenni új ételeket, és a függvény segítségével eltudjuk tárolni a „getMealsben” az adatokat. Az új felvett étel megjelennek a listánkban.

## Register komponens

register.component.ts

register()

Az „auth.service.ts”-ben megírt „register” függvényt meghívja és lefuttat rajta egy „ha” függvényt a jelszó megegyezésével kapcsolatban.

## About komponens

about.component.html

Itt jelenítettük meg a csoportban résztvevők nevét és github elérhetőségüket

# Felhasználói dokumentáció

## Webes felület „beüzemelése”:

Ha GitHubról szedi le akkor a program letöltése után szükséges a ’ \web\LifeStyle’ állományban a parancssorban egy npm install parancs futtatása. Illetve a ’\api\LifeStyle’ helyen egy composer install parancs futtatása. A ’api\LifeStyle’ helyen lévő ’.env.example’ nevű fájlt lemásolni, és ’.env’-nek átnevezni és a ’DB\_DATABASE=laravel’ sorban a laravelt átírni a saját állományunkra ami ’calorieapp’.

Amennyiben a program pendrive vagy más adathordozón kerül átadásra semmi ilyen előkészületet nem igényel.

Ezt követően szükséges az XAMPP elindítása. Majd a ’\LifeStyle’ mappában parancssor segítségével indítunk egy ’php artisan serve’ parancsot. Hasonlóan cselekszünk a ’web\LifeStyle’ útvonalon, azonban ott egy ’ng serve -o’ parancsot használunk.