***Fejlesztői dokumentáció***

Tartalom

[1. Bevezetés, a téma ismertetése, témaválasztás indoklása: 1](#_Toc100239868)

[2. Fejlesztői dokumentáció 1](#_Toc100239869)

[Felhasznált technológiák 1](#_Toc100239870)

[Kódolási konvenciók 2](#_Toc100239871)

[Könyvtárszerkezet 2](#_Toc100239872)

[Adatszerkezet 5](#_Toc100239873)

[Osztályok 6](#_Toc100239874)

[Tesztelés 12](#_Toc100239875)

[3. Felhasználói dokumentáció 22](#_Toc100239876)

[4. A készítők adatai 22](#_Toc100239879)

# 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 összetettebb, 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
  + XAMPP

Asztali alkalmazás

* + Eclipse
  + Mariadb server (Adatbázis kiszolgáló)

Készítéshez használt programok

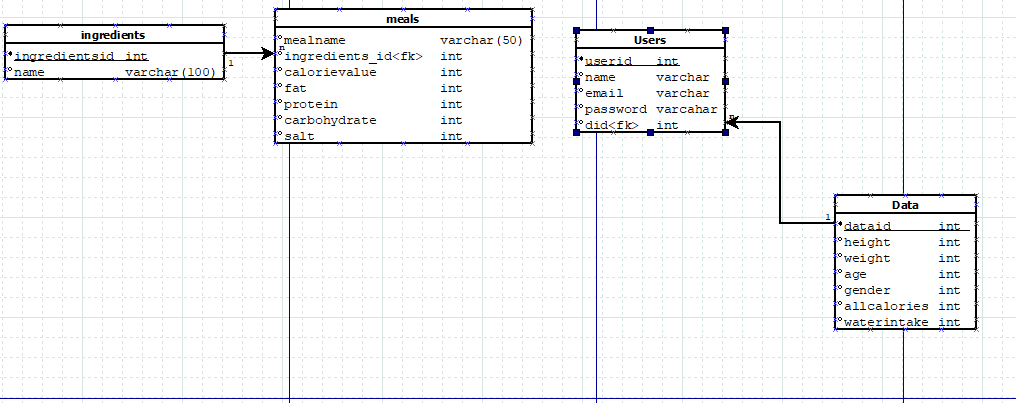
* + dia
  + Visual studio code 1.63.0
  + Isomnia 2021.5.3
  + Mariadb server
  + phpmyadmin

## 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 |

# Adatszerkezet

# 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

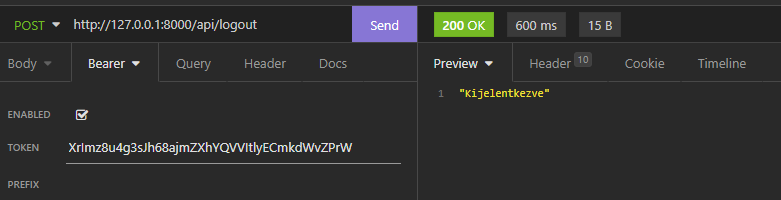
Automatikusan generált leírás

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

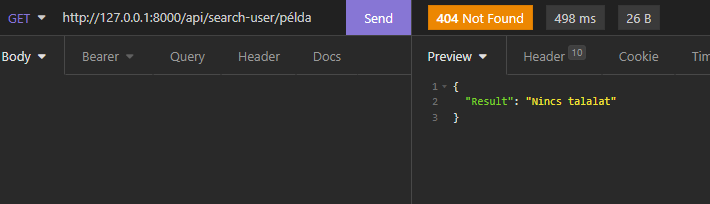
Automatikusan generált leírás

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

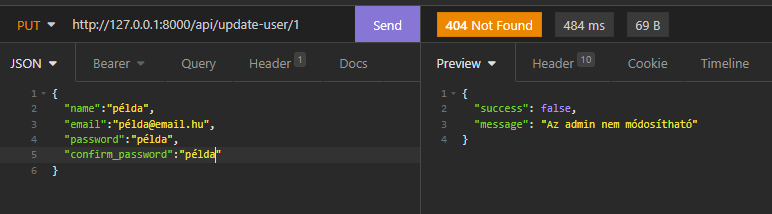
Automatikusan generált leírás

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

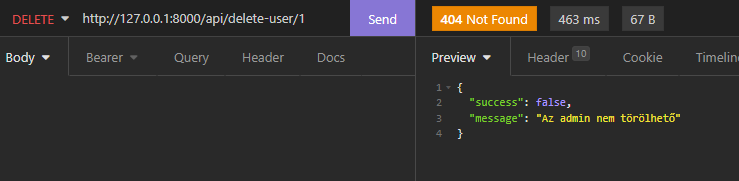
Automatikusan generált leírás

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 **Ismert hibák a kódban**

# Felhasználói dokumentáció

Webes felület „beüzemelése”:

Ha GitHubról szedi le akkor a program letöltése után szükséges a ’lifestyleapp\web\LifeStyle’ állományban a parancssorban egy npm install parancs futtatása. Illetve a ’lifestyleapp\api\LifeStyle’ helyen egy composer install parancs futtatása. A ’lifestyleapp\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 ’lifestyleapp\api\LifeStyle’ mappában parancssor segítségével indítunk egy ’php artisan serve’ parancsot. Hasonlóan cselekszünk a ’lifestyleapp\web\LifeStyle’ útvonalon, azonban ott egy ’ng serve -o’ parancsot használunk.

# A készítők adatai