győri szakképzési centrum

Jedlik Ányos
Gépipari és Informatikai
Technikum és Kollégium

9021 Győr, Szent István út 7.
+36 (96) 529-480
+36 (96) 529-448

OM: 203037/003
jedlik@jedlik.eu
www.jedlik.eu

Záródolgozat feladatkiírás

Tanuló(k) neve[1]:	Kokas Márk, Körmendi Dávid Ákos, Muth Márk József
Képzés:	nappali munkarend
Szak:	5 0613 12 03 Szoftverfejlesztő és tesztelő technikus

A záródolgozat címe:

Flexify edzéstervező program

Konzulens:	Bólya Gábor
Beadási határidő:	2024. 04. 15.

Győr, 2024. 04. 15.

Módos Gábor
igazgató

[1] Szakmajegyzékes záródolgozat esetében több szerzője is lehet a dokumentumnak, OKJ-s záródolgozatnál egyetlen személy ad le záródolgozatot.

győri szakképzési centrum

Jedlik Ányos
Gépipari és Informatikai
Technikum és Kollégium

9021 Győr, Szent István út 7.
+36 (96) 529-480
+36 (96) 529-448

OM: 203037/003
jedlik@jedlik.eu
www.jedlik.eu

Konzultációs lap[2]

	A konzultáció		Konzulens aláírása
	ideje	témája	
1.	2023.10.15.	Témaválasztás és specifikáció	
2.	2024.03.14.	Záródolgozat készültségi fokának értékelése	
3.	2023.04.10.	Dokumentáció véglegesítése	

Tulajdonosi nyilatkozat

Ez a dolgozat a saját munkánk eredménye. Dolgozatunk azon részeit, melyeket más szerzők munkájából vettünk át, egyértelműen megjelöltük.
Ha kiderülne, hogy ez a nyilatkozat valótlan, tudomásul vesszük, hogy a szakmai vizsgabizottság a szakmai vizsgáról kizár minket és szakmai vizsgát csak új záródolgozat készítése után tehetünk.

Győr, 2024. április 15.

Kokas Márk

Körmendi Dávid Ákos

Muth Márk József

[2] Szakmajegyzékes, csoportos konzultációs lap

Jedlik Ányos Gépipari és Informatikai Technikum és Kollégium

Dokumentáció

Készítették:

Kokas Márk

Körmendi Dávid Ákos

Muth Márk József

Tartalom

1. Bevezetés

1.1 A probléma és a megoldás

A Flexify egy online edzéstervező-, és követő webalkalmazás.

Az elmúlt években egyre nagyobb teret nyert az edzés, a testépítés és az ehhez hasonló tevékenységek az emberek mindennapjaiban. Számos pozitív hatása van a szervezetre, és komoly szinteket is el lehet benne érni kellő idő befektetésével és elhatározással. Azonban ehhez szükséges nyomon követni a fejlődésünket és elért eredményeinket, hogy a lehető legproduktívabbak lehessünk. Programunk ezeket a szempontokat szem előtt tartva nyújt segítséget egyaránt kezdők és középhaladók számára is.

1.2 Ötletelési fázis és a választás

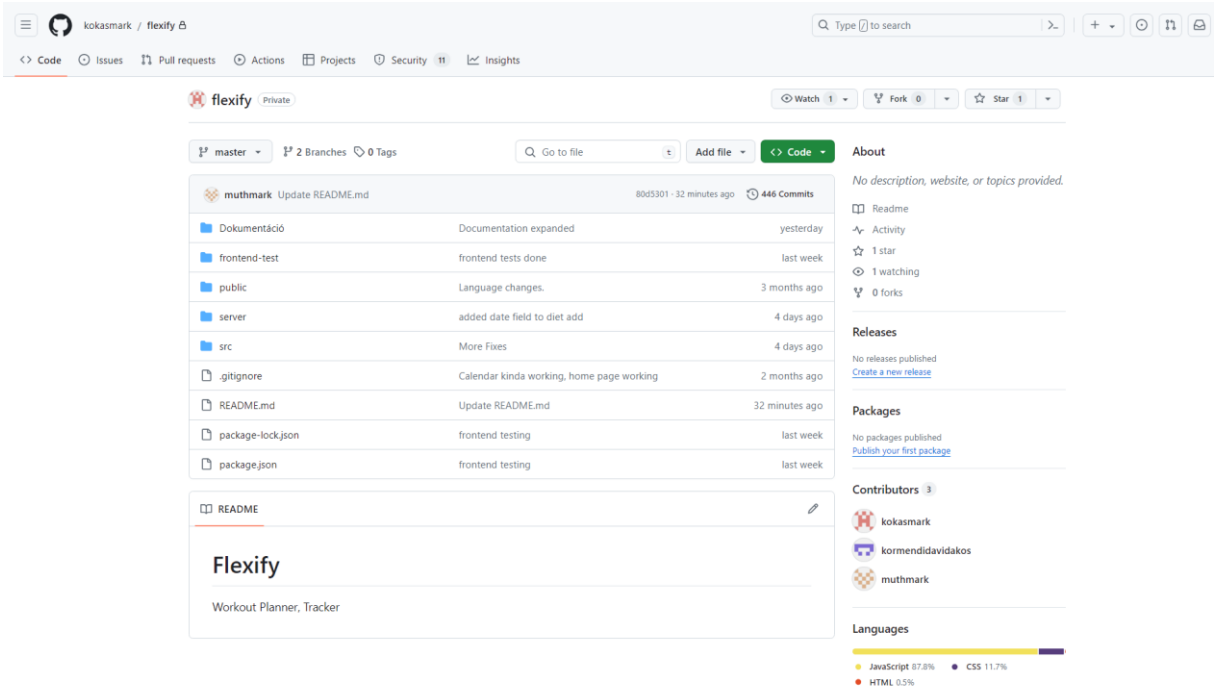
Programunk kitalálásakor több, egymástól teljesen független témát is feszegettünk, mire végül eljutottunk a végső döntéshez. Mérlegeltük a témákban rejlő lehetőségeket, a megvalósítás egyszerűségét és a személyes preferenciáinkat. Ezek alapján jutottunk el a végtermékhez, amely egy hozzánk közel álló témát feszeget, az edzést. Fontos volt számunkra, hogy egy felhasználóbarát, de emellett hasznos funkciókkal ellátott alkalmazást hozzunk létre, amely megkönnyíti az emberek számára a testépítés folyamatát.

1.3 A jövő

Miután megegyeztünk az ötletben, azonnal nekiláttunk a munkának. Rengeteg ötletünk volt, amit meg akartunk valósítani, és ahogy haladtunk előre a fejlesztésben ezek egyre csak szaporodni kezdtek. Emiatt rengeteg lehetőség rejlik a további fejlesztésekre is. Ilyen funkció például a különböző okoseszközökkel való összekapcsolás lehetősége, amely tovább könnyíti az edzések követhetőségét és további statisztikákat tud nyújtani, például az edzés közbeni pulzus stb. Egy másik fejlődési lehetőség egy szociális részleg bevezetése, ahol a felhasználók tudnak egymással kapcsolatba lépni, edzéseket egyeztetni a naptárban vagy akár egy megírt edzéstervet átadni a másiknak.

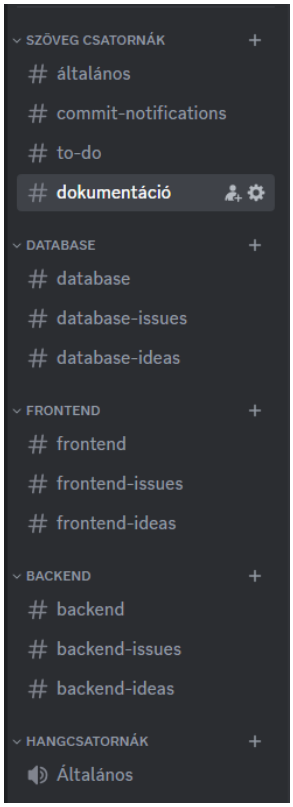
1.4 A közös munka

A programozás elkezdése előtt megbeszéltük a feladatmegosztást és a használt platformokat. A személyes preferenciákat és tapasztalatokat figyelembe véve jutottunk arra, hogy a front- és backenden külön dolgozunk. Kokas Márk volt a frontend fejlesztő, Körmendi Dávid és Muth Márk pedig közösen a backend és adatbázis feladatokat látták el. A verziókövetésre a GitHubot használtuk, amely jelentősen megkönnyítette a csapatmunka folyamatát. A GitHub repository-nk elérhető a https://github.com/kokasmark/flexify URL-en.



Erről a képről leolvasható, hogy milyen programnyelveket milyen arányban használtunk a fejlesztéskor. Az is látható, hogy a kép készítésekor 446 commitot hajtottunk végre. A mappastruktúra is megjelenik ezen az oldalon.

A kommunikáció, az ötletek egyeztetése és a hibák jelzésére a Discord nevű programot használtuk, ahol egy külön szervert hoztunk létre a vizsgaremekkel kapcsolatos munkához. Az alábbi képeken látható a szerver felépítése, valamint egy példa részlet a tervezés megbeszéléséről.



2. A program

2.1 Technikai részletek

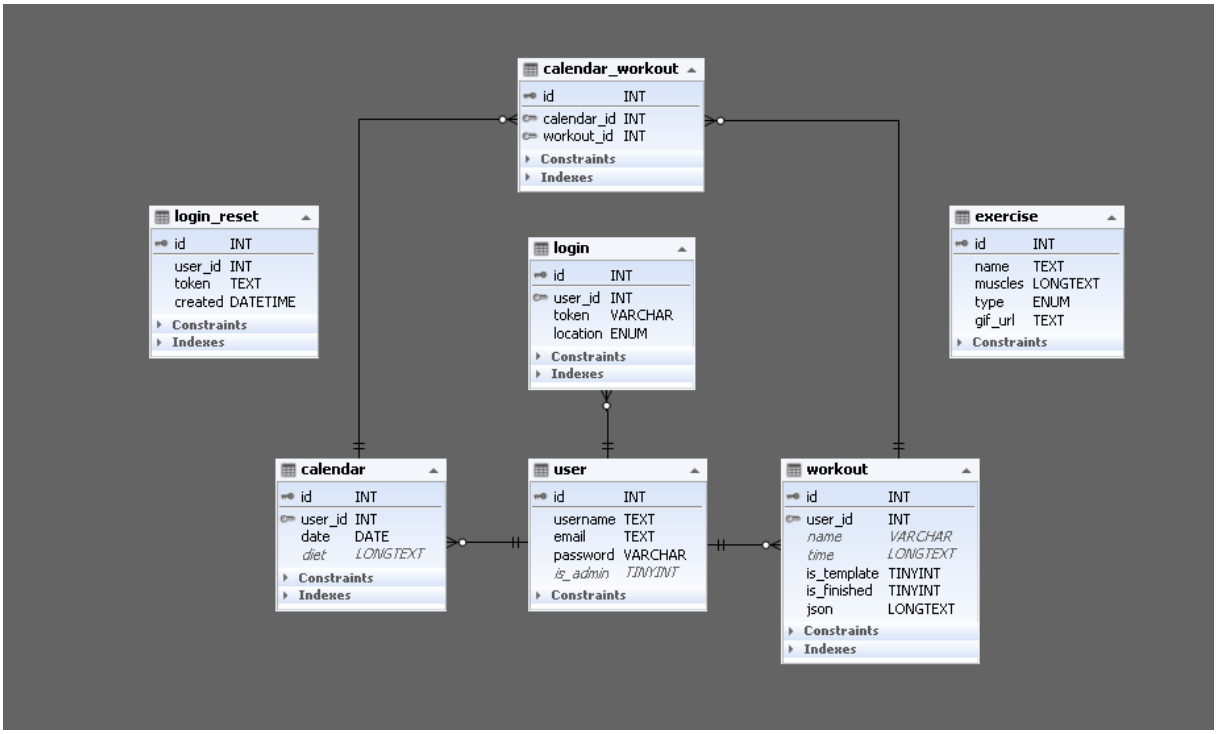
A projekt frontendként a *React* JavaScript keretrendszert használja. *A React egy deklaratív, effektív, és rugalmas JavaScript könyvtár, felhasználói felületek készítéséhez. Lehetővé teszi komplex felhasználói felületek összeállítását izolált kódrészletekből, amiket "komponenseknek" hívunk.* (React dokumentáció)

Backend fejlesztéshez Node.js-t használtunk, mivel ebben volt a legtöbb személyes tapasztalatunk, emellett a JavaScript programnyelv talaján maradhattunk, így könnyen szinkronizálni tudtuk a front- és backend fejlesztés folyamatát. Továbbá kielégített minden feltételt, ami a projektünk megvalósításához szükséges volt.

A Node.js alaphoz kellett egy webszerver keretrendszert is választanunk. A választásunk az *express* nevű modulra jutott, a könnyen elérhető dokumentációja, az elterjedtsége, illetve egyszerűsége miatt.

Adatbázisnak a MySQL mellett döntöttünk. Fő érveink a technológia mellett többek között magukba foglalták a meglévő tapasztalatainkat, egyszerűségét és saját preferenciánkat.

2.2 Az adatbázis felépítése



2.2.1 Táblák

user – felhasználó

- *id*: elsődleges azonosító
- *username*: felhasználónév
- *email*: e-mail cím
- *password*: titkosított jelszó
- *is_admin*: adminisztrátor jogosultság

Minden felhasználót, és a hozzátartozó fiók adatokat tárolja a tábla. A *password* mezőbe a felhasználó bcrypt könyvtárral hashelt majd sózott jelszava kerül.

login – bejelentkezés

- *id*: elsődleges azonosító
- *user_id*: idegenkulcs, a bejelentkezéshez tartozó felhasználó
- *token*: bejelentkezéshez használt token
- *location*: bejelentkezés helye (web vagy mobil)

A felhasználó azonosításához használt BASE64 tokeneket tárolja a tábla. A felhasználóhoz egyidejűleg egy webes, és egy mobilos bejelentkezés tartozhat.

calendar – naptár

- *id*: elsődleges kulcs
- *user_id*: idegenkulcs, a naphoz kapcsolt felhasználó
- *date*: dátum
- *diet*: aznapi diéta, JSON formátumban

A felhasználó napjához tartozó bejegyzés. A bejegyzés jelenléte feltételezi, hogy a felhasználó aznap használta a diéta vagy az edzés funkciót.

workout – edzés

- *id*: elsődleges kulcs
- *user_id*: idegenkulcs, az edzéshez tartozó felhasználó
- *name*: edzés neve
- *time*: naptárhoz használt kezdő- és vég-időpont, JSON formátumban
- *is_tempalte*: sablon-e
- *is_finished*: befejezett-e
- *json*: edzés adatai, JSON formátumban

A felhasználó által létrehozott edzés sablonok, tervezett- vagy teljesített edzések tárolására szolgál.

calendar_workout – naptár-edzés összekötő tábla

- *id*: elsődleges kulcs
- *calendar_id*: idegenkulcs, naptár bejegyzés azonosító
- *workout_id*: idegenkulcs, edzés bejegyzés azonosító

A felhasználó által tervezett vagy teljesített edzéseket naptári bejegyzésekhez köti.

exercise – gyakorlat

- *id*: elsődleges kulcs
- *name*: név
- *muscles*: használt izmok JSON tömbben
- *type*: típus; ismétléses vagy idős
- *gif_url*: gyakorlatot személtető gif fájlneve (kiterjesztés és útvonal nélkül)

A használható gyakorlatok adatai kerülnek a táblába. Bővítése az adminisztrátori felületről lehetséges.

login_reset – jelszó-visszaállítás

- *id*: elsődleges kulcs
- *user_id*: idegenkulcs, felhasználó azonosító
- *token*: jelszó-visszaállításhoz használt token
- *created*: létrehozás ideje

Jelszó visszaállításhoz használt, véletlenszerűen generált BASE64 tokenek tárolására szolgál. A bejegyzések 10 percig használhatók.

2.3 Az adatbázistól a megjelenítésig

2.3.1 Inicializálás

Az alkalmazás elindításakor, mielőtt az API endpointok felállításra kerülnének, a program két objektumot is létrehoz, az adatbázis objektumot (*db.js*) és a gyakorlatok objektumot (*exercises.js*). Feladatuk egyszerű:

```
class DB{
    constructor(log){
        this.log = log
        this.tables = []
        this.structure = {}
        this.didInitStructure = false
    }
```

- A DB osztály felel minden adatbázis kérés elküldéséért, és a visszakapott értékek visszaadásáért, majd a kérés naplózásáért. Emellett eltárolja az adatbázis struktúráját az admin oldal megfelelő működéséért.

```
class Exercises{
    constructor(db){
        this.db = db
        this.exercises = this.loadExercises()
    }
```

- Az Exercises osztály tárolja az adatbázisban lévő gyakorlatokat. Ennek előnye, hogy nem kell minden alkalommal újra lekérni az adatbázisból. Amikor új gyakorlatot hoz létre vagy töröl egy adminisztrátor, a gyakorlatok listája is automatikusan frissül.

2.3.2 Kérés feldolgozása

```
app.get('/api/user', (req, res) => getUserDetails(new User(req, res, db, log)))
```

Miután a kérés beérkezik a szerver nyitott endpointjára, a kérés átadásra kerül egy *User objektumnak* és egy *kéréskezelő függvénynek*.

2.3.3 User osztály

```
class User{
    constructor(req, res, db, log){
        this.req = req
        this.res = res
        this.db = db
        this.log = log

        this.id = this.getUserId()
        this.loggedIn = this.isLoggedIn()
        this.admin = this.getAdmin()
        this.alreadyResponded = false
    }
```

Minden kéréshez egy User objektum tartozik. Ez többek között eltárolja a kérést küldő felhasználó azonosítóját és jogosultságait (amennyiben be van jelentkezve). Minden kérés-specifikus adatbázis műveletet ez az objektum indít, feldolgoz, és ad vissza a szervernek válaszra kész formában.

2.3.4 Kéréskezelő függvények

```
async function postLogin(user){
    log(2, '/api/login')

    const token = await user.login()
    if (token === false) return user.respondMissing()
    else if (token === 0) return user.respond(400, {reason: 'Invalid username or password'})

    user.respondSuccess({token: token})
}
```

A kéréskezelő függvényeknek három feladata van. Először is, naplózza a Bejövő kérés útvonalát. Ezután meghívja a User objektum megfelelő függvényét. Végül a függvény visszatérési értékéből visszaküldi a felhasználó felé a kérésre a választ, vagy éppen a megfelelő hibakódot és leírást.

2.3.5 Naplózó függvény

```
function log(level, message){
    if (DEBUG_LEVEL < level) return

    const log_colors = ["\x1b[31m", "\x1b[90m", "\x1b[36m", "\x1b[33m", "\x1b[32m", "\x1b[47m\x1b[30m"]
    process.stdout.write(log_colors.at(level) + "[" + moment().format('YYYY-MM-DD hh:mm:ss') + "]:\x1b[0m ")
    console.log(message)
}
```

A naplózáshoz egy saját, primitív függvényt használunk. A függvény meghívásakor beállítunk egy szintet, -1-től 4-ig. Amennyiben a *.env* fájlban beállított szint legalább akkora, mint a naplózás szintje, az üzenet (a dátummal és időponttal együtt) ki lesz írva a konzolra.

```
[2024-04-11 09:32:27]: -1-es szint
[2024-04-11 09:32:27]: 0-ás szint
[2024-04-11 09:32:27]: 1-es szint
[2024-04-11 09:32:27]: 2-es szint
[2024-04-11 09:32:27]: 3-as szint
[2024-04-11 09:32:27]: 4-es szint
```

2.4 Hitelesítés, biztonság

2.4.1 Technológia

Az oldalon történő bejelentkezéshez, majd a felhasználó azonosításához egyedi készítésű tokent használunk. Egy felhasználóhoz egyszerre egy web- és egy mobil felületről történő hitelesítéshez használt token tartozhat, ezeket az adatbázis *login* táblájában tároljuk. Ennek működési elve egyszerű: bejelentkezéskor a bejelentkezési adatok ellenőrzése után töröljük a jelenleg eltárolt tokenjét, ha létezik ilyen. Ezután eltárolunk egy frissen, véletlenszerűen generált, 64 karakter hosszúságú *BASE64* karakterláncot. Ezt visszaküldjük a felhasználónak, és a továbbiakban ezt kell a kérés fejlécébe csatolnia önmaga hitelesítésére, az *X-Token* mezőbe.

2.4.2 Jelszavak

Sikeres regisztráció vagy jelszó visszaállítás után a felhasználó jelszava titkosításon esik át. Erre a *bcrypt* modul alapértelmezett aszimmetrikus hashelését használjuk, és 10 lépésben sózunk. Ennek előnye, hogy a kapott karakterlánc nem visszafordítható egyszerű szöveggé, viszont a szöveges jelszót ismerve bejelentkezéskor vizsgálható az egyezésük. Ezt a titkosított karakterláncot a *user* tábla *password* mezőjében tároljuk.

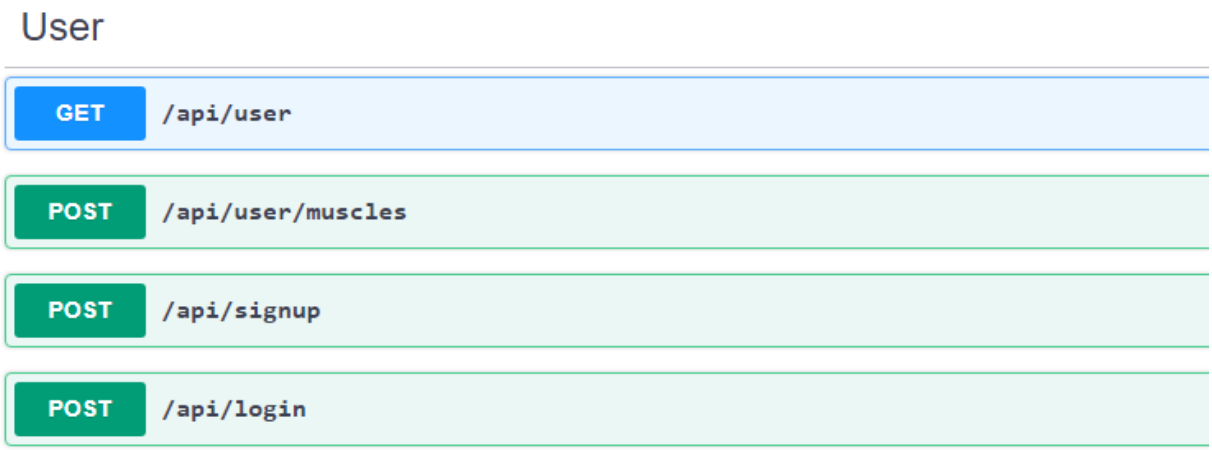
2.4.3 Jogosultságok

A programban két jogosultsági szint található: felhasználó és admin. Felhasználói jogosultságot bejelentkezés után kap a felhasználó. Ezzel az oldal funkcióit teljes egészében tudja használni, de értelemszerűen az admin oldalt nem használhatja, így csak a saját felhasználói adatait érheti el. Ezzel ellentétben admin jogosultságot csak úgy kaphat valaki, ha a *user* tába *is_admin* mezőjébe 1-es érték tartozik hozzá. Ezt az admin oldalról lehet megtenni. Az admin felhasználók is ugyanúgy használhatják a programot, de ők rendelkeznek az admin oldal menüponttal is.

2.5 API dokumentáció

Az API hívások (a bejelentkezés és regisztrálás funkciókon kívül), tokent használnak a felhasználó hitelesítésre. Ennek megadása a hívás fejlécében, az *X-Token* mezőben történik. Ez a hívások leírásában nem fog megjelenni. Az előbb említett (bejelentkezés és regisztráció) funkciók kivételével, minden híváshoz jogosultság szükséges. Ezeknek két szintje van, user és admin. Az admin felhasználók mindkét szinttel rendelkeznek. A hívások leírásában a minimális jogosultság kerül csak feltüntetésre. Minden hívás visszatér egy boolean success paraméterrel. Értéke a funkció végrehajtásának sikerességétől függ. Ez sem lesz a továbbiakban feltűntetve.

2.5.1 User



getUserDetails

- Útvonal: /api/user
- Kérés típusa: GET
- Jogosultság: user
- Válasz:
    - felhasználónév (string)
    - email (string)

postUserMuscles

- Útvonal: /api/user/muscles
- Kérés típusa: POST
- Jogosultság: user
- Paraméter: vizsgált időtartam napokban
- Válasz: használt izmok

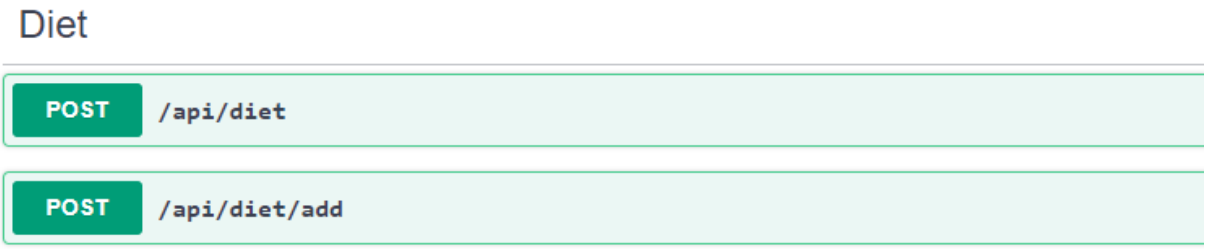
postUserRegister

- Útvonal: /api/signup
- Kérés típusa: POST
- Jogosultság: -
- Paraméter: felhasználónév, email, jelszó
- Válasz:
    - token (string)

postUserLogin

- Útvonal: /api/login
- Kérés típusa: POST
- Jogosultság: -
- Paraméter: felhasználónév vagy email, jelszó
- Válasz:
    - token (string)

2.5.2 Diet



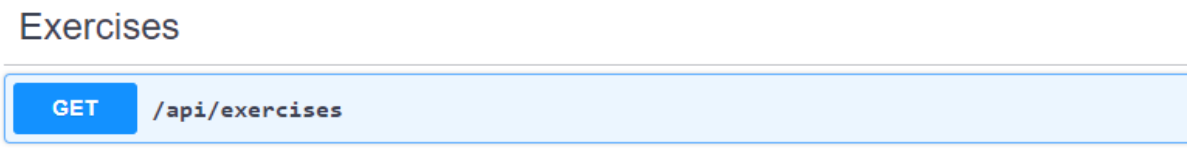
postDietQuery

- Útvonal: /api/diet
- Kérés típusa: POST
- Jogosultság: user
- Paraméter: vizsgált nap
- Válasz:
    - json (JSON objektum):
        - breakfast (JSON vektor)
        - lunch (JSON vektor)
        - dinner (JSON vektor)
        - snacks (JSON vektor)
        - a fentebb említett vektorok JSON objektumokat tartalmaznak, amelyek rendelkeznek egy *name* és egy *calories* mezővel

postDietAdd

- Útvonal: /api/diet/add
- Kérés típusa: POST
- Jogosultság: user
- Paraméter: étkezések adatai
- Válasz: -

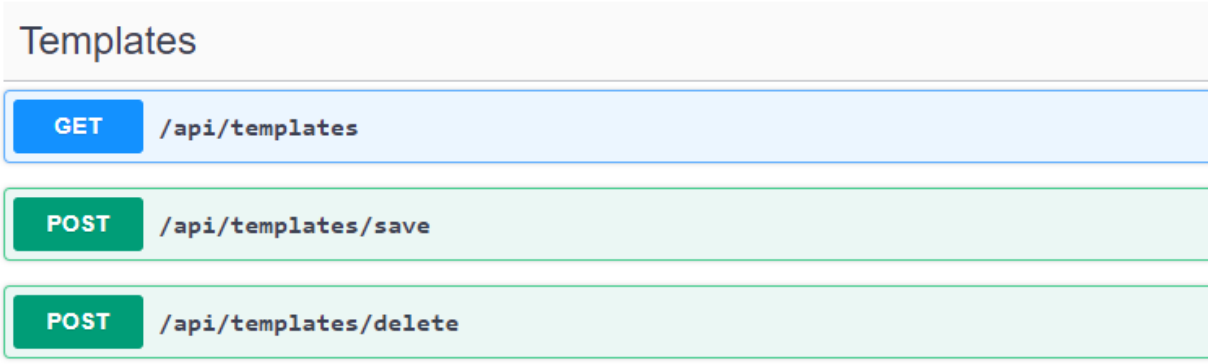
2.5.3 Exercises



getExercises

- Útvonal: /api/exercises
- Kérés típusa: GET
- Jogosultság: user
- Válasz:
    - json (JSON tömb) a tárolt gyakorlatokkal

2.5.4 Templates



getUserTemplates

- Útvonal: /api/templates
- Kérés típusa: GET
- Jogosultság: user
- Válasz:
    - templates (JSON tömb, JSON objektumokkal):
        - id (number) sablon azonosítója
        - name (string) sablon neve
        - json (JSON vektor) gyakorlatok azonosítója és ismétlési adatok

postSaveTemplate

- Útvonal: /api/templates/save
- Kérés típusa: POST
- Jogosultság: user
- Paraméter: sablon neve, adatai
- Válasz: -

postDeleteTemplate

- Útvonal: /api/templates/delete
- Kérés típusa: POST
- Jogosultság: user
- Paraméter: sablon azonosítója
- Válasz: -

2.5.5 Workouts

getWorkoutsFinished

- Útvonal: /api/workouts/finished
- Kérés típusa: GET
- Jogosultság: user
- Válasz:
    - dates (JSON vektor)

postWorkoutsDates

- Útvonal: /api/workouts/dates
- Kérés típusa: POST
- Jogosultság: user
- Paraméter: vizsgált hónap
- Válasz:
    - dates (JSON vektor):
        - (string) edzés dátuma

postWorkoutsData

- Útvonal: /api/workouts/data
- Kérés típusa: POST
- Jogosultság: user
- Paraméter: vizsgált nap
- Válasz:
    - data (JSON objektum)
        - id (number) edzés azonosítója
        - name (string) edzés neve
        - json (JSON objektum) gyakorlatok és ismétlési adatok
        - time (JSON objektum) edzés kezdete és vége
        - isFinished (boolean)

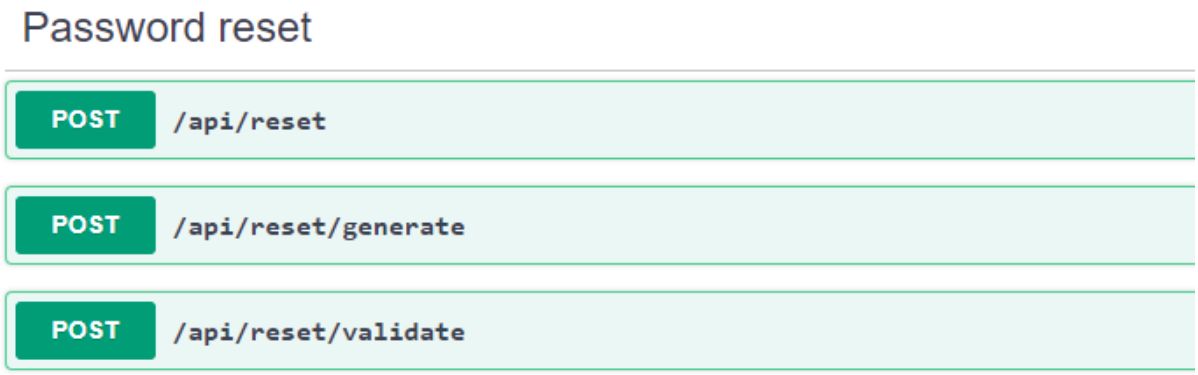
postSaveWorkout

- Útvonal: /api/workouts/data
- Kérés típusa: POST
- Jogosultság: user
- Paraméter: edzés adatai
- Válasz:
    - id (number) edzés azonosítója

postFinishWorkout

- Útvonal: /api/workouts/data
- Kérés típusa: POST
- Jogosultság: user
- Paraméter: edzés azonosítója
- Válasz: -

2.5.6 Password reset



postResetPassword

- Útvonal: /api/reset
- Kérés típusa: POST
- Jogosultság: -
- Paraméter: új jelszó, jelszó visszaállítási token
- Válasz: -

postResetGenerate

- Útvonal: /api/reset/generate
- Kérés típusa: POST
- Jogosultság: -
- Paraméter: felhasználónév vagy email
- Válasz: -

postResetValidate

- Útvonal: /api/reset/validate
- Kérés típusa: POST
- Jogosultság: -
- Paraméter: jelszó visszaállítási token
- Válasz:
    - success (boolean) validálás eredménye

2.5.7 Admin

getAdminTables

- Útvonal: /api/admin/tables
- Kérés típusa: GET
- Jogosultság: admin
- Válasz:
    - tables (JSON vektor):
        - (string) táblanév

postAdminData

- Útvonal: /api/admin/data
- Kérés típusa: POST
- Jogosultság: admin
- Paraméter: táblanév
- Válasz:
    - json (JSON objektum):
        - headers (JSON vektor) oszlopnevek
        - body (JSON vektor) sorok adatai

postAdminUpdate

- Útvonal: /api/admin/update
- Kérés típusa: POST
- Jogosultság: admin
- Paraméter: táblanév, adat azonosítója, adat
- Válasz: -

postAdminDelete

- Útvonal: /api/admin/delete
- Kérés típusa: POST
- Jogosultság: admin
- Paraméter: táblanév, adat azonosítója
- Válasz: -

postAdminInsert

- Útvonal: /api/admin/insert
- Kérés típusa: POST
- Jogosultság: admin
- Paraméter: táblanév, adat
- Válasz: -

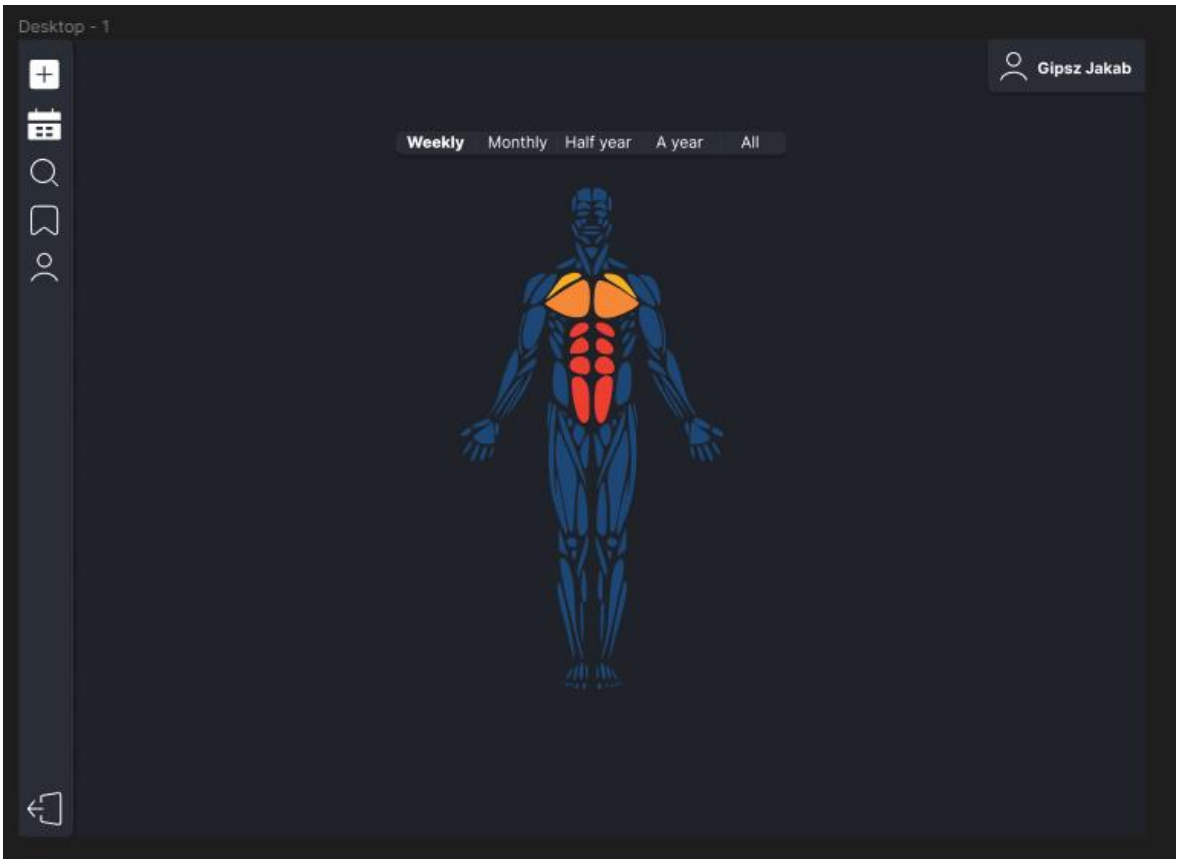
2.6 Frontend dokumentáció

2.6.1 Technológia választás

Választásunk a React keretrendszerre esett, modern szintaktikájával és hatalmas mennyiségű tanulható anyagával az interneten egyszerűvé tette a weboldal UI(User Interface) és UX(User Experience) fejlesztését.

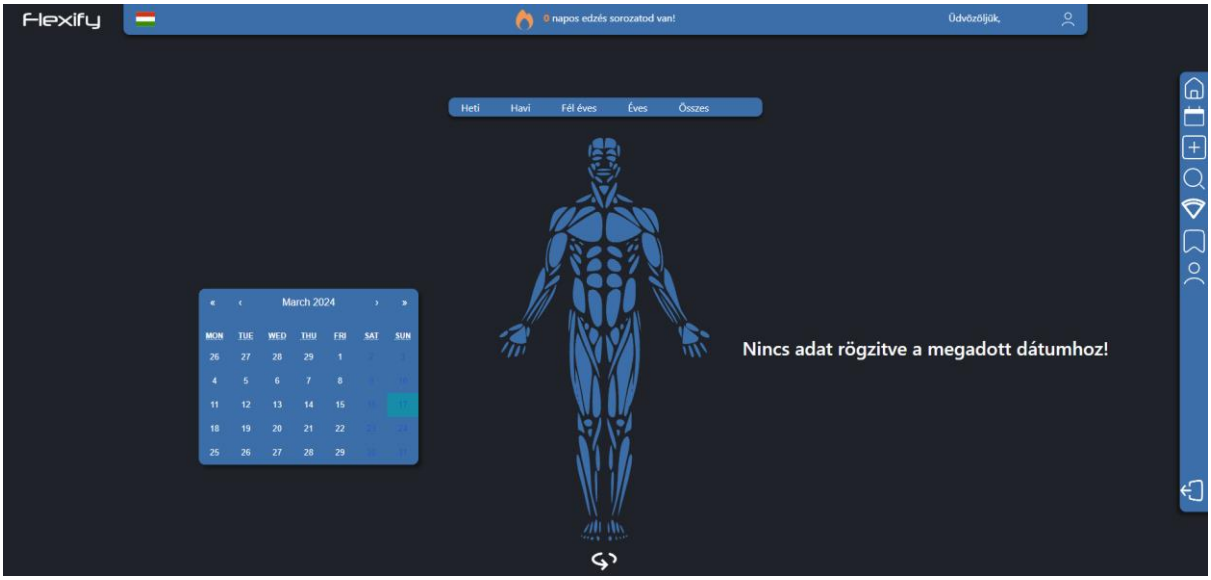
2.6.2 Korai tervezetek

A weboldal első terveit a Figma nézet tervező szoftver segítségével készítettük el. Ezek elképzeléseink alapjait foglalták össze.



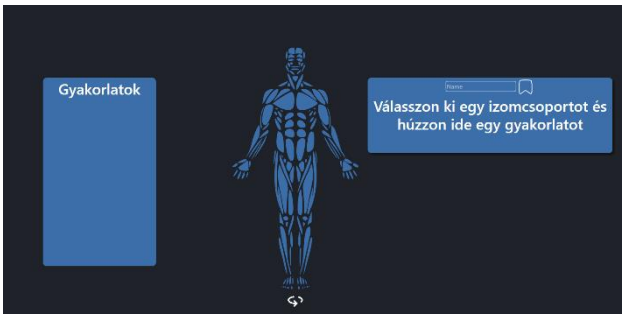
Első tervek Figmában

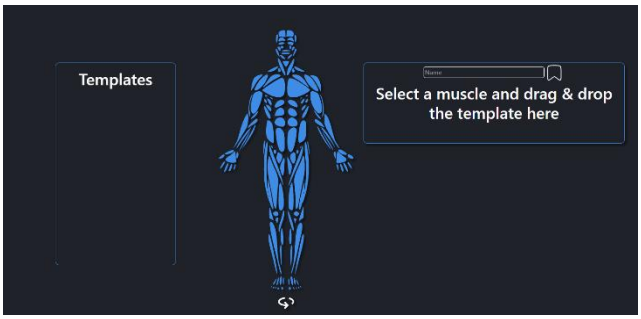
2.6.3 Korai verziók



Főoldalunk sokkal több részletet tartalmazott az előző verziókban, de ezt az egyértelműség és a letisztultság érdekében átterveztük a mai állapotára.

Több oldal is nagy változásokon esett át az első működő verzió óta, de volt pár, amelyek kisebb esztétikai változtatásokon kívül hűek maradtak az eredeti elképzeléshez.

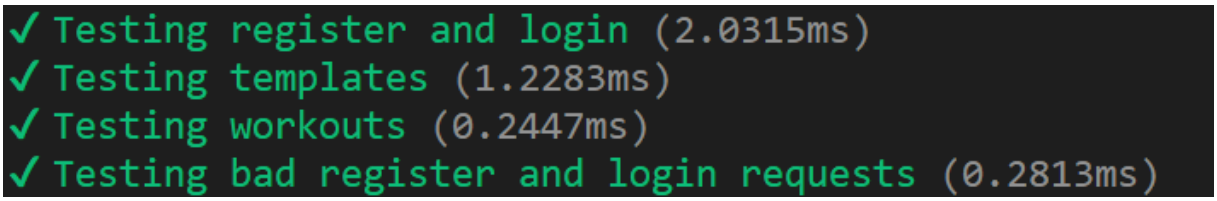




Az előző verziók edzés készítője(Bal oldalt) és a jelenlegi verzió edzés tervezője(Jobb oldalt)

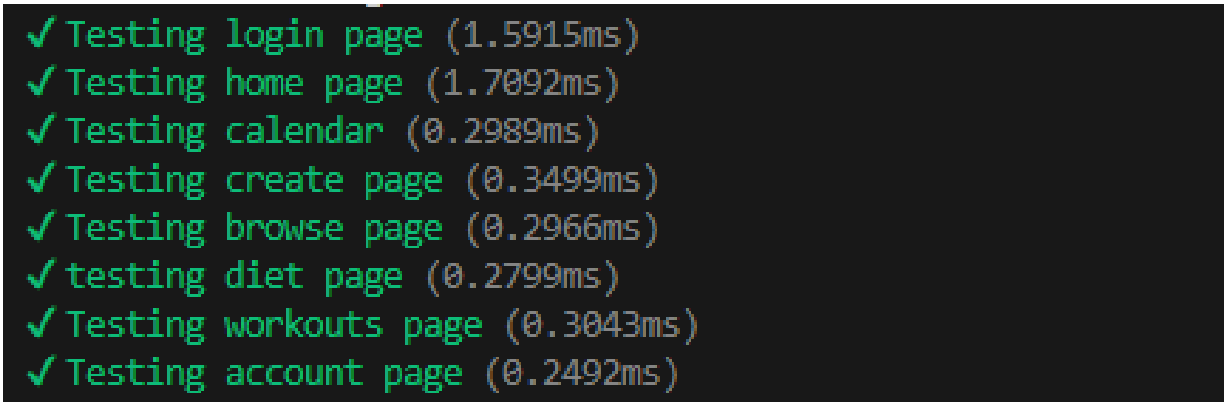
3. Tesztek

3.1 Backend tesztek



A backend teszteket egy futtatáskor generált teszt fiókon végezzük, amelyek feltöltődnek teszt adatokkal, így semmilyen formában nincs hatással a rendes felhasználók adataira. Ezzel a módszerrel tudjuk tesztelni a regisztrációt és a bejelentkezést is egyaránt, majd utána felhasználni a generált felhasználót a további funkciók tesztelésére is.

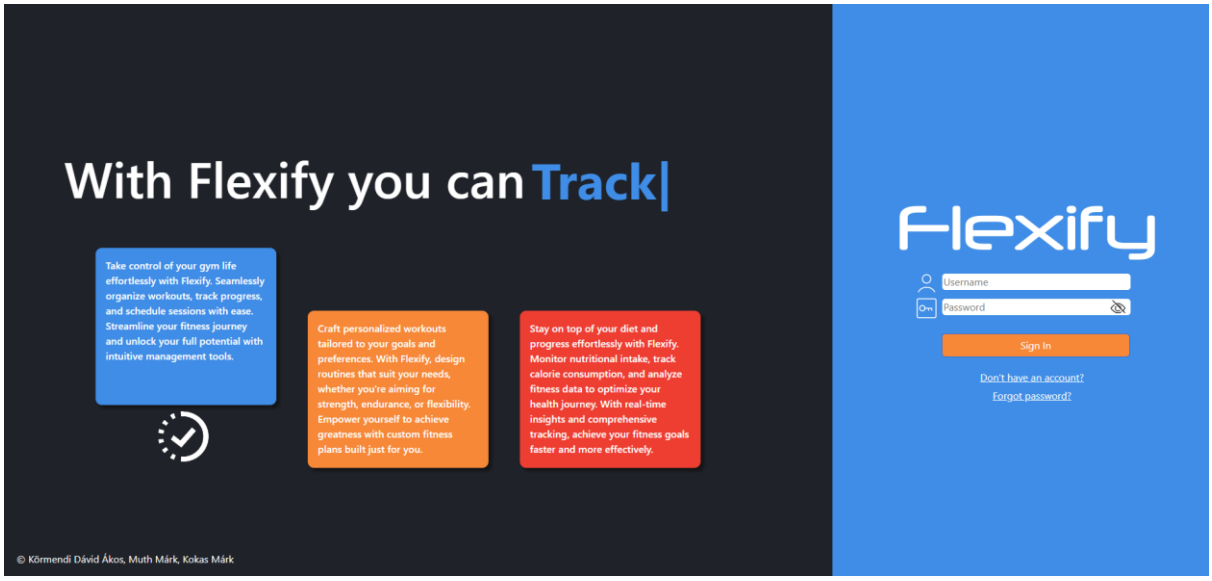
3.2 Frontend tesztek



A frontend tesztelését a Selenium webes bővítmény segítségével végezzük. A Selenium a megírt programkódot futtatva automatikusan a böngészőben végrehajtja az utasításokat, és teszteli a webalkalmazás frontend oldali funkcióit.

4. UI és UX Felépítése

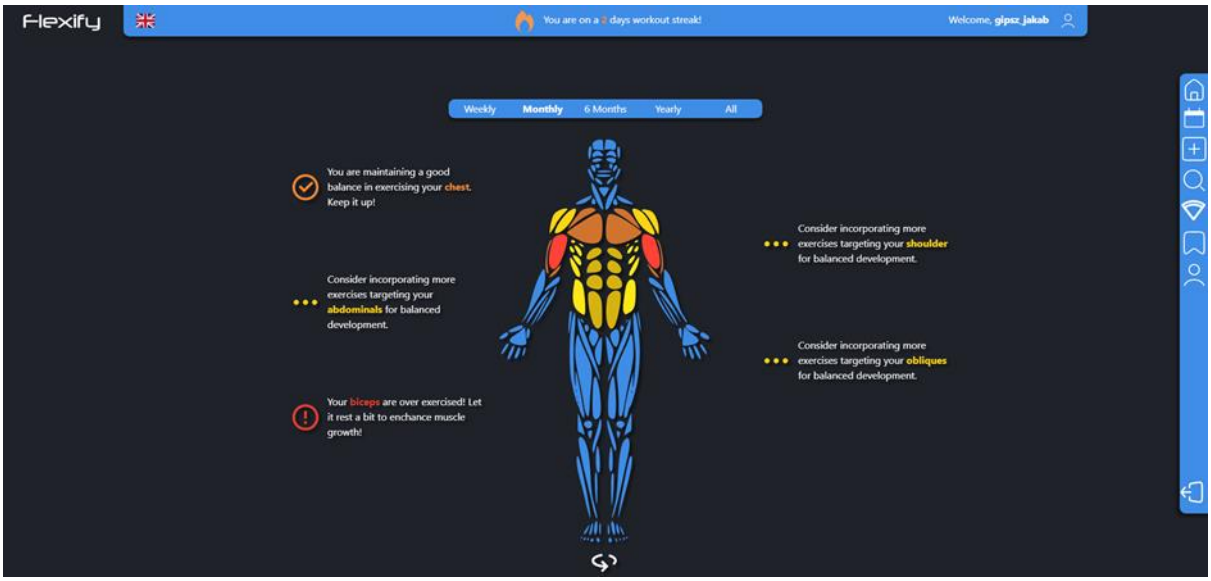
4.1 Bejelentkezés és Regisztrálás

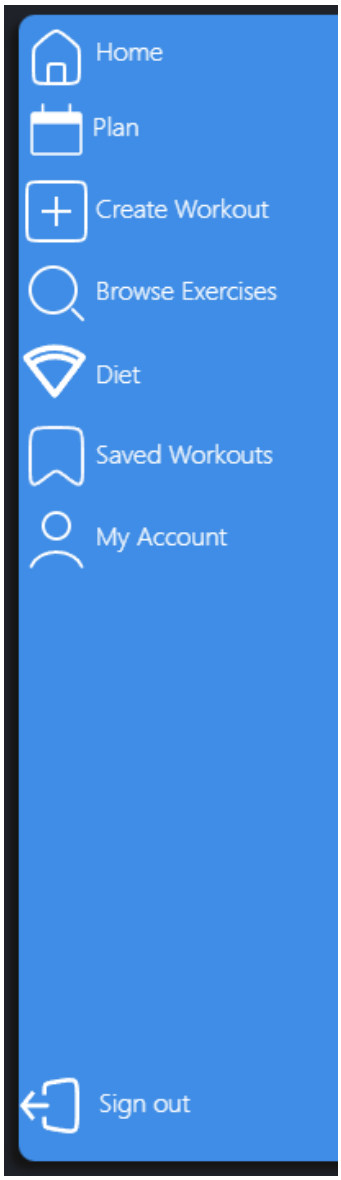


A bejelentkezés és regisztrálás oldalak fontos szerepet foglalnak el oldalunk mechanizmusában, nem csak azért, mert ezek által tud egy felhasználó hozzáférést szerezni a különböző funkciókhoz, hanem mivel ez az első oldal, ami megjelenik nekik. A bejelentkezés és regisztrációs oldalakat animációkkal és információkkal töltve terveztük meg, hogy egy új felhasználó érdeklődését felkeltsük amikor felkeresi az oldalt.

A felhasználók email címük vagy felhasználó nevük megadásával tudnak bejelentkezni. Bejelentkezés után a weboldal összes funkciója elérhető válik számukra.

4.2 Főoldal





Bejelentkezés után minden felhasználót először a Főoldal(Home) köszönt. Innen a felhasználó minden másik funkcióhoz el tud jutni a navigációs oldalsáv segítségével.

Edzésterv(Plan): Egy naptár nézet segítségével a felhasználó egyszerűen és egyértelműen tudja megtervezni edzéseinek időpontját és követni teljesítményét.

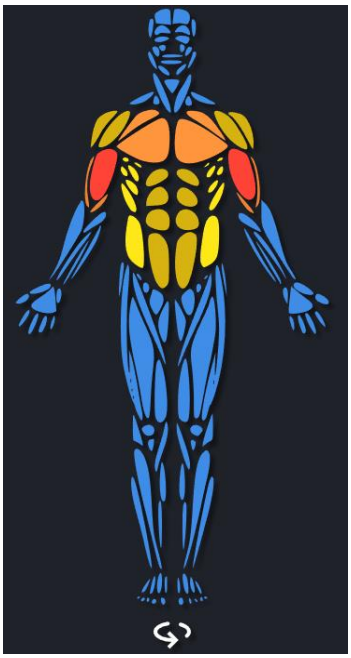
Edzés létrehozása(Create Workout): Egy egyszerű drag & drop oldal ahol a felhasználó edzés terveket készíthet az adatbázisunkban található megannyi gyakorlatból.

Gyakorlatok(Browse Exercises): A felhasználó izomcsoportokra felosztva tekintheti meg az egyes gyakorlatokat melyekhez mind egy rövid segítő videó tartozik amely megmutatja hogy, hogy egészséges azt az adott gyakorlatot végezni.

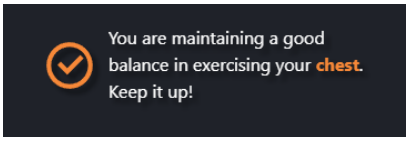
Étkezés(Diet): A felhasználó itt rögzítheti a három főétkezés és egyéb fogyasztását egy napon. Előző napok étkezését is itt tudja megtekinteni egy felhasználóbarát felületen.

Mentett Edzéseim(Saved Workouts): Ezen az oldalon találja meg a felhasználó a létrehozott edzésterveit és kezelheti azokat.

Fiók Beállítások (My Account): A felhasználó adatait tartalmazó oldal melyen többek között a felhasználó által preferált anatómiai beállítás változtatható.

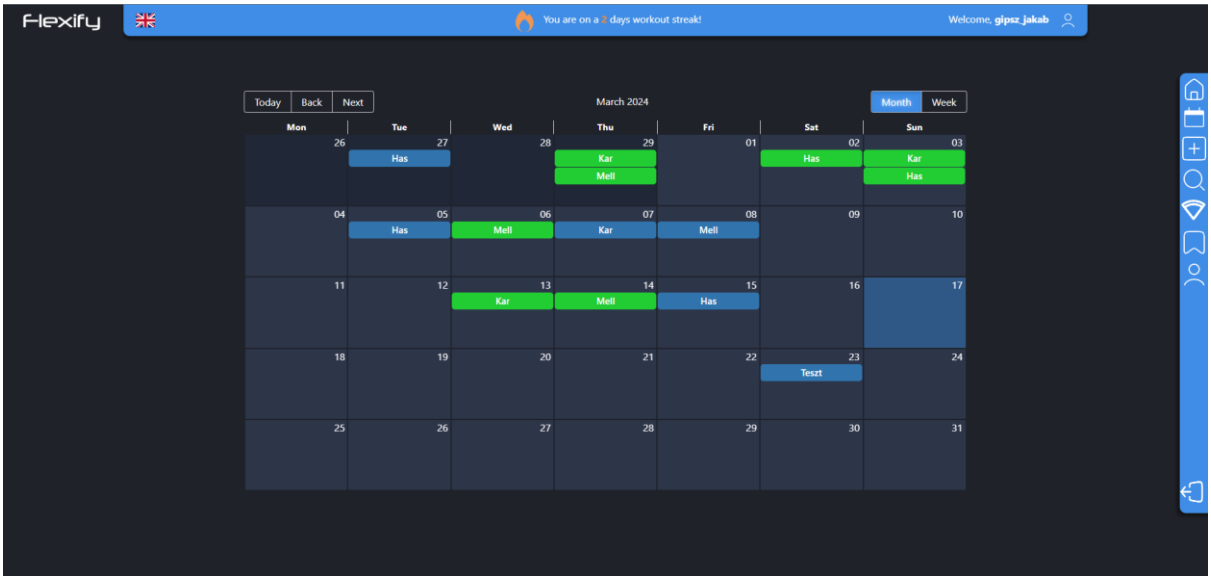


A főoldal közepén található ábra különböző intervallumokon belüli edzések, edzett izomcsoportjainak megoszlását szemlélteti. Alul edzés, túledzés és egészséges megoszlásba sorolja az izomcsoportokat melyhez nem csak vizuális, hanem verbális visszajelzést is ad.

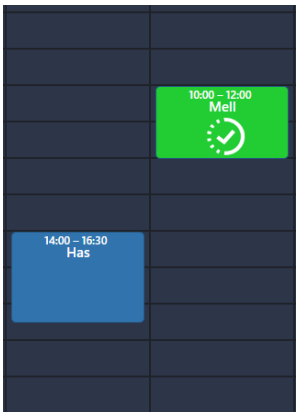


Az intervallumokat az ábra feletti sávban választhatja ki a felhasználó. Az e heti. hónapi, félévi, évi és minden leedzett edzések közül tud választani.

4.3 Edzésterv

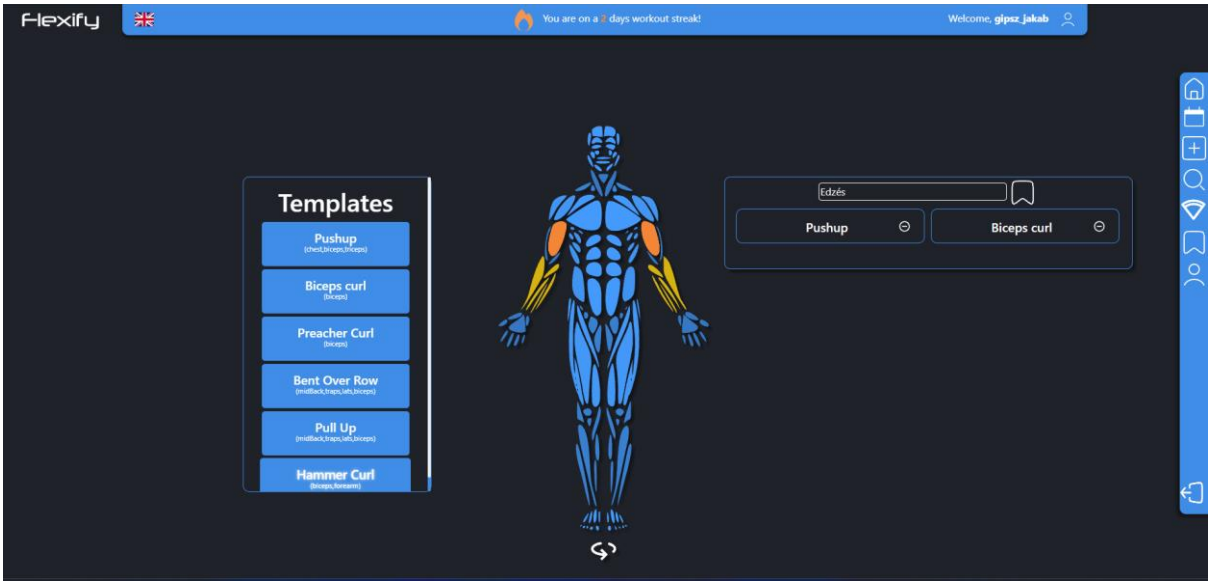


A felhasználónak ezen az oldalon van lehetősége elkészített edzés terveit időtartamhoz és naphoz kötnie. Az oldal egyszerű színkódolással különbözeti meg a még nem és a már elvégzett edzéseket.



A naptárat két nézet között lehet váltani, a havi és a heti nézet. A heti nézet órákra lebontva mutatja az edzéseket. Itt a legegyszerűbb edzést felvenni, hosszan lenyomva a kívánt kezdő órát az edzés tervezett végéig húzza a felhasználó majd a felugró panelről kiválaszthatja melyik edzést szeretné hozzáadni.

4.4 Edzés Létrehozás

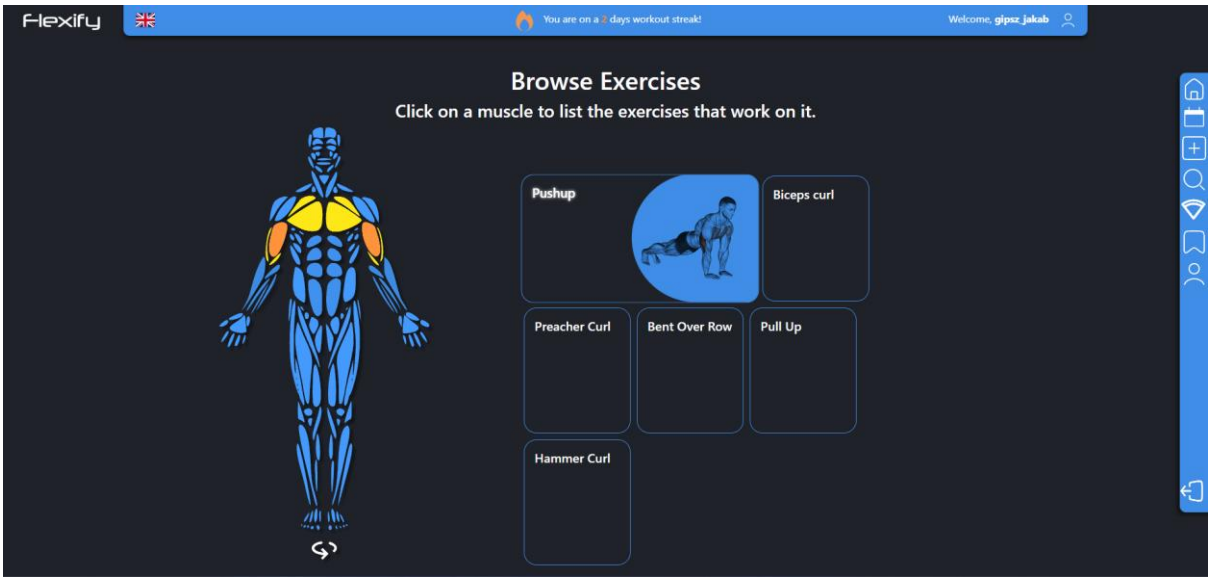


Edzés létrehozásához egy felhasználóbarát felületet terveztünk meg, melyen bárki egyszerűen el tud navigálni. Gyakorlatok ki listázásához a felhasználónak nincs más teendője csak kijelölni azt az izomcsoportot melyet edzeni szeretne, ekkor a gyakorlatok kis kártyák formájában megjelennek a bal oldali panelen.

A felhasználó amikor egy gyakorlat kártyára pozícionálja kurzorát felvillannak a test ábrán a gyakorlat által még edzett más izom csoportok.

A kártyát a felhasználó a kurzorával megfogva behúzhatja a jobb oldali panelre, ekkor az a gyakorlat hozzáadódik az edzés tervhez. Itt a felhasználó szerkesztheti az adott gyakorlat “Set”-jeit. Egyes gyakorlatok ismétlési számát, vagy ha a gyakorlat időtartamra megy akkor idejét és a használt súlyát kilogrammban megadva.

4.5 Gyakorlatok

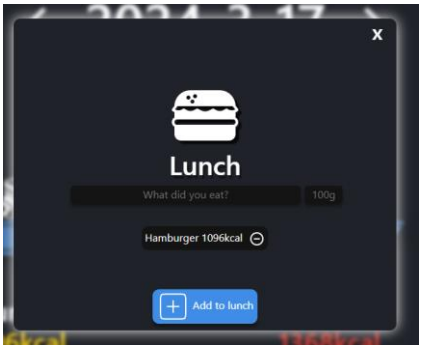


A felhasználó ezen az oldalon rövid videók segítségével megtekintheti egy modern felületen, hogy az adott gyakorlatot miként kell elvégezni az egészséges izom fejlődéshez.

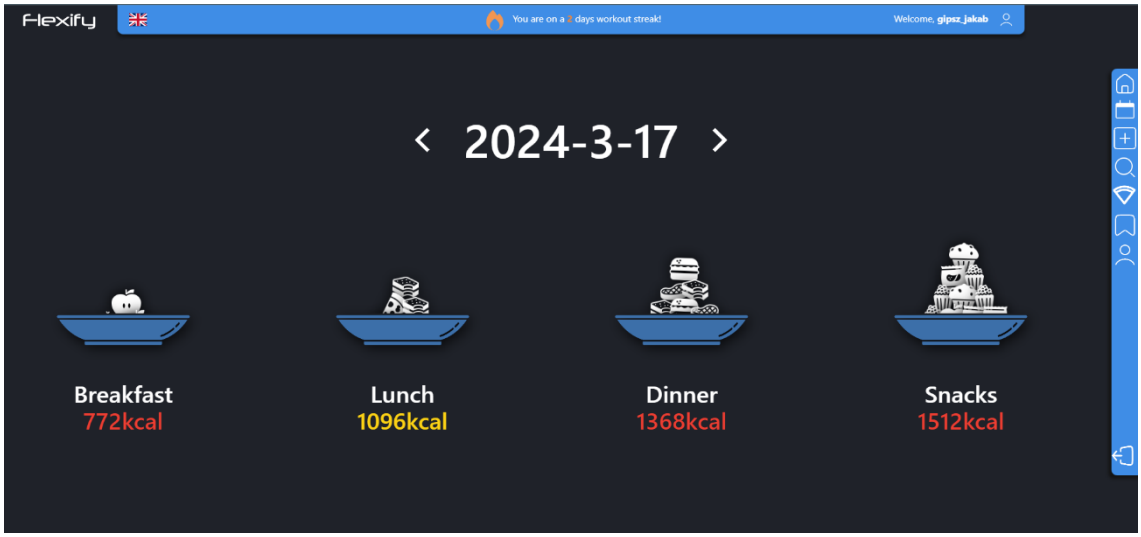
A felhasználónak itt is a test ábrán egy izomcsoportra kattintva van lehetősége kilistázni a gyakorlatokat. A gyakorlatokat kurzorával kijelölve tekintheti meg a segítő animációt és a gyakorlat által edzett további izomcsoportokat

4.6 Étkezés

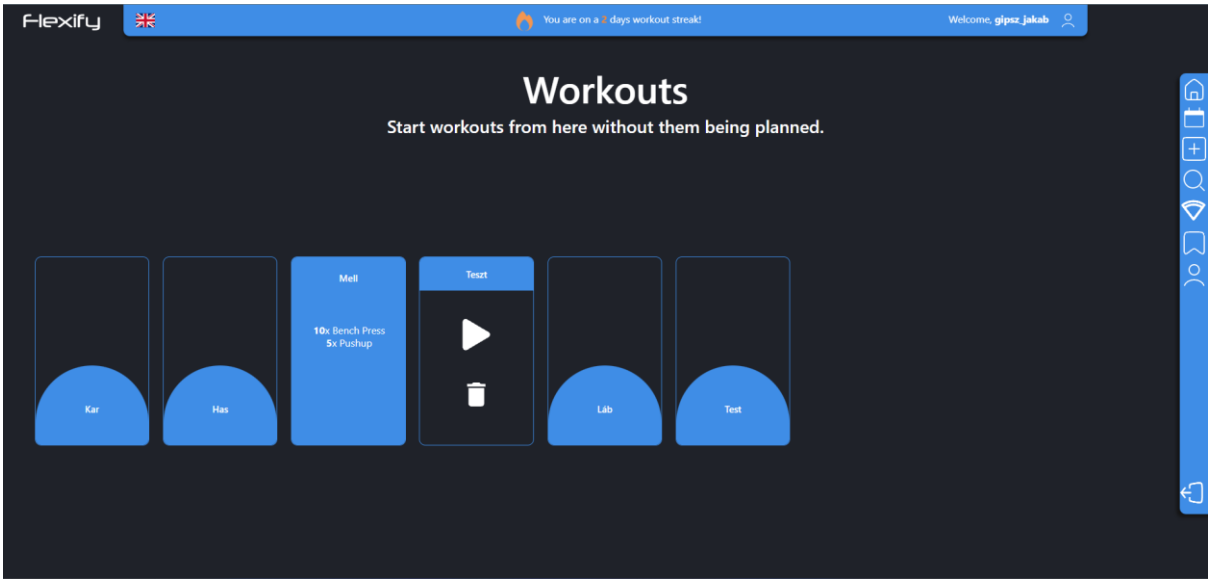
A felhasználó étkezését a három főétkezésre és egyéb elfogyasztott ételekre bontja a weboldal. A felhasználó az adott étkezés táljára kattintva egy felugró panelen adhatja meg elfogyasztottételeit, egy külső API segítségével (*https://api.api-ninjas.com/v1/nutrition*) pedig az adott étel és adag tápértékegységeit menti el a rendszer.



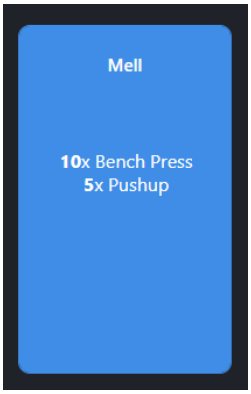
A felhasználó előző napok elfogyasztott ételeink értékeit is megtekintheti, ez elősegíti egy egészséges étrend tervezéséhez.



4.7 Mentett edzéseim



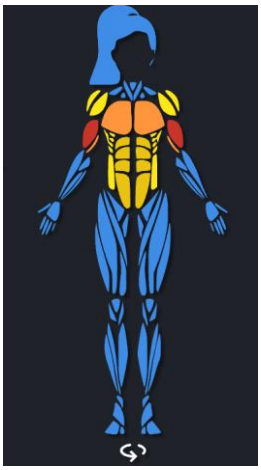
A felhasználó által létrehozott edzés tervek ezen az oldalon jelennek meg kártyák formájában. Erről az oldalról spontán edzések is indíthatók melyek nem voltak betervezve, ezek az edzések elvégzés után megjelennek a naptárban, mint kész edzések.

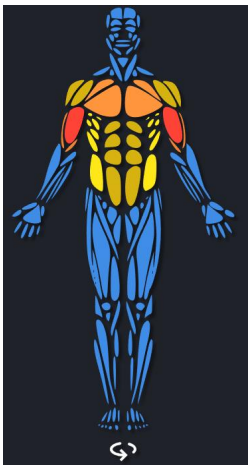


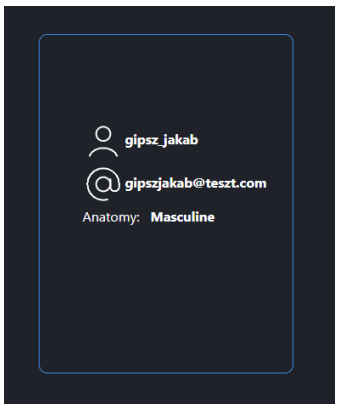
Ha a felhasználó egy kártyára pozícionálja kurzorát, a kártyán megjelenik az edzés rövid tömörített verziója így a felhasználó könnyebben megtudja különböztetni edzéseit.

4.8 Fiók Beállítások

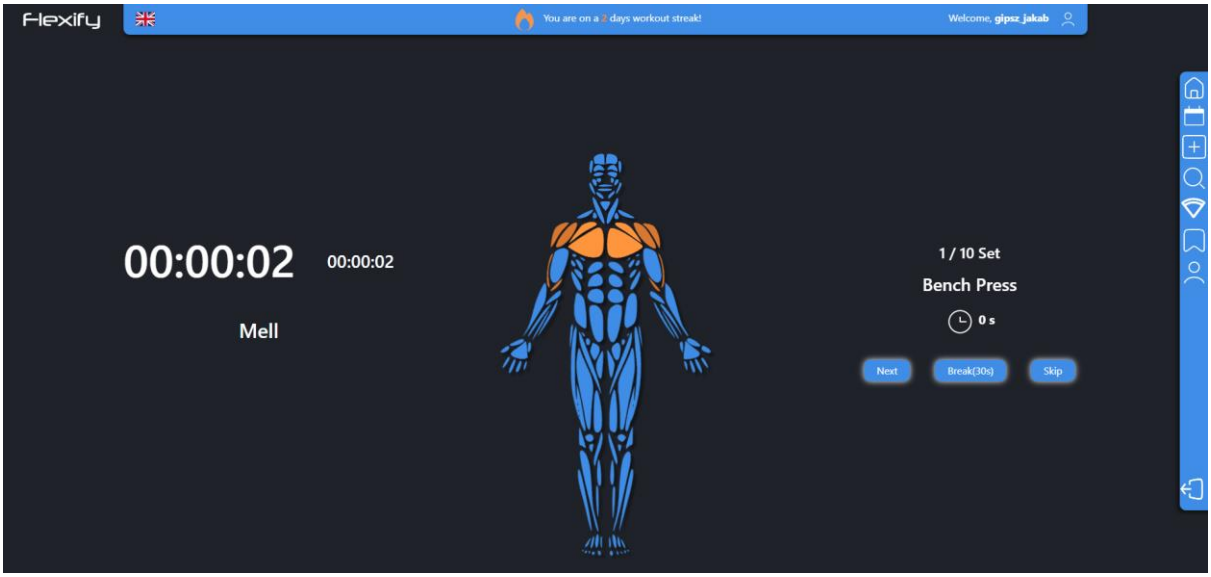
Ezen az oldalon tekintheti meg a felhasználó adatait és választhatja ki, hogy a test ábra oldalainkon milyen anatómiával szerepeljen (nőies vagy férfias).







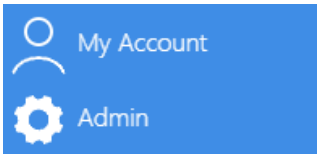
4.9 Edzés

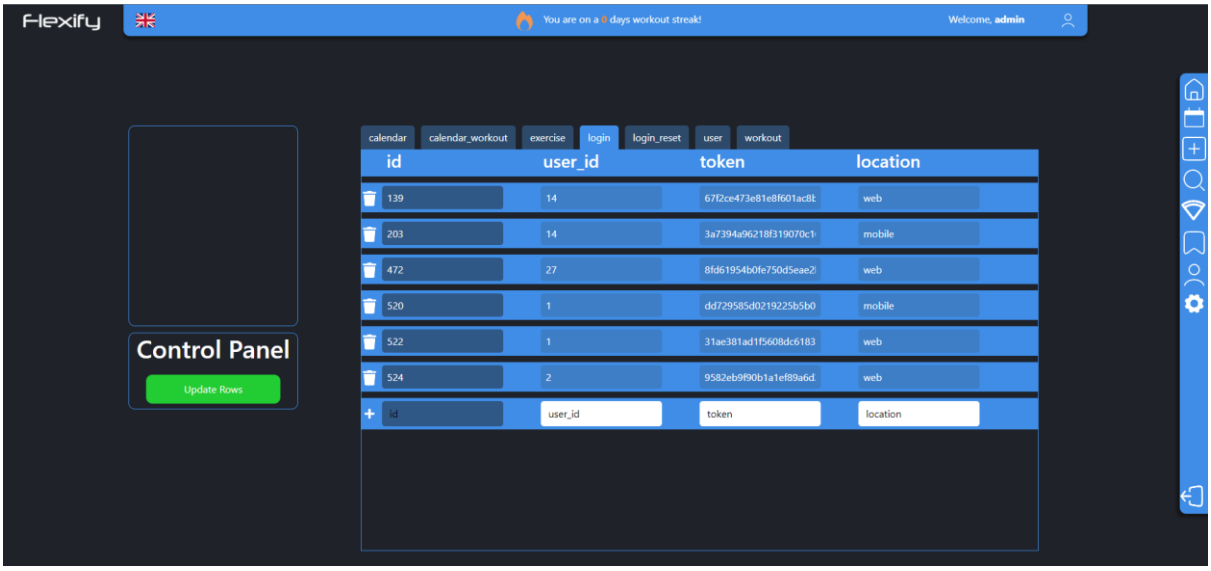


Edzéseit a webes felületen is van lehetősége elindítania felhasználónak. Ekkor az edzés oldalra navigál át és itt egy egyszerű dizájnon követheti végig edzés tervét.

4.10 Kezelő felület (Admin)

Weboldalunk tartalmaz egy kezelő felületet is melyről egyszerűen és felhasználóbarát módon lehet az adatbázist kezelni. Ha egy felhasználó admin jogokkal ellátott felhasználóval jelentkezik be a jobb oldalon elhelyezkedő navigációs sávon megjelenik egy új menüpont mely az Admin oldalra navigál.





A kezelő felület biztosítja a megfelelő jogokkal rendelkező felhasználónak az adatváltoztatást, törlést és hozzáadást egy egyszerű felületen.

Az adatbázis bármely táblája elérhető innen és módosítható, ha a felhasználónak megfelelő a jogköre.

5. Mobil alkalmazás

5.1 Miért van rá szükség?

Csapatunk inspiráció és megoldások kutatása közben több hasonló alkalmazást tekintett meg és ezekben az alkalmazásokban mind az a közös, hogy szorgalmazzák a felhasználót natív mobil alkalmazásuk letöltésére.

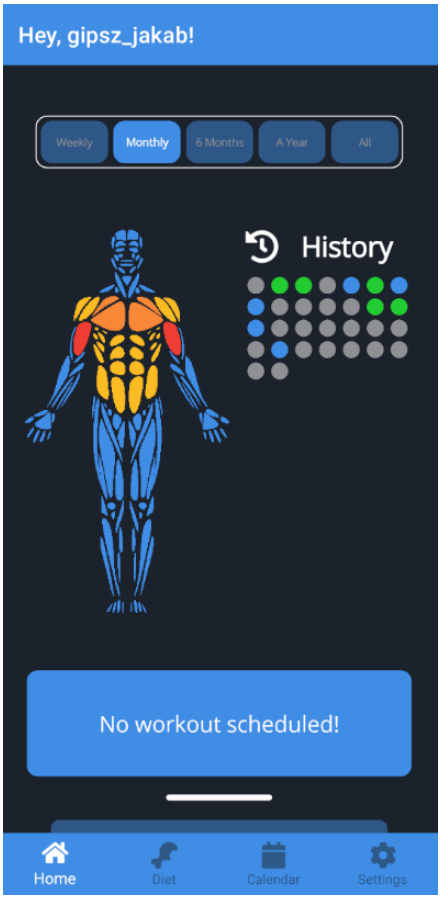
Egyszerű válasz erre az, hogy ebben a korban már hasznosabb egy natív mobil alkalmazás, mint egy reszponzívvá tett weboldal. Ezért a weboldal kis képernyőkön felajánlja mobil alkalmazásunk letöltését.

5.2 Technológia választás

Választásunk a .Net Maui keretrendszerre esett, fő okunk erre az volt, hogy ez az egyetlen ilyen fejlesztői környezet mely egyszerre többféle operációs rendszerre engedje meg az adott kód fejlesztését.

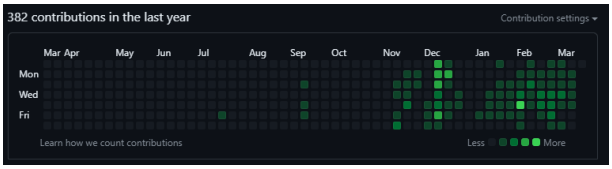
Egyetlen nehézsége volt ennek a megoldásnak, hogy ez a keretrendszer még nem teljesen elterjedt és ez a sok kisebb hibájának köszönhető, ezért egy két buckát nehezen tudtunk csak megmászni fejlesztés közben.

5.3 Főoldal



A mobil alkalmazás főoldala tartalmazza a test ábrát a webalkalmazásból, szintén a felsősávban lévő gombok segítségével képes a felhasználó intervallumot választani.

A “History” panel a Github felhasználói fiókoknál található “Contributions” ihlette. Egy kisebb fajta vizuális motiváció a felhasználónak edzései folytatásához.

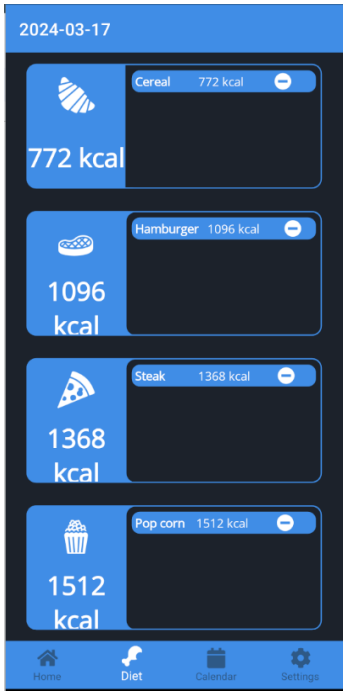


A Github contributions panelje

A Főoldalon az az napi edzés/edzések is megjelennek, ha voltak előre beírva, innen el is indíthatók ezek.

Az az napi edzés kártyája alól egy felhúzással megtekinthető az összes mentett edzés terv. A panel felcsúszik amikor a felhasználó felhúzza azt.

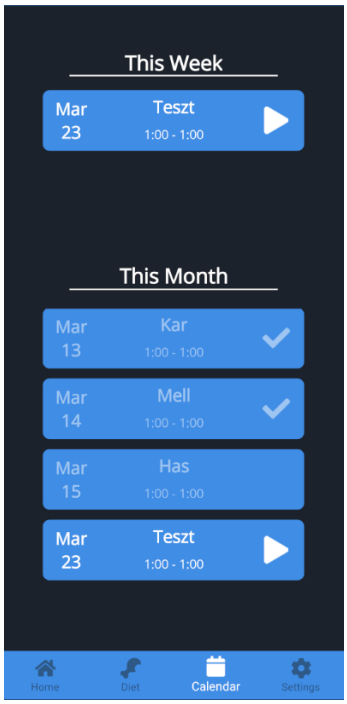
5.4 Étkezés



A mobil alkalmazás a web alkalmazáshoz hasonló módon kezeli a felhasználó étkezését.

Egy tervezett funkció még az alkalmazásban, ami a következő verzióban fog jönni a boltban vásárolt ételek Bar kódjának beszkennelésével való étel felvétel.

5.5 Naptár

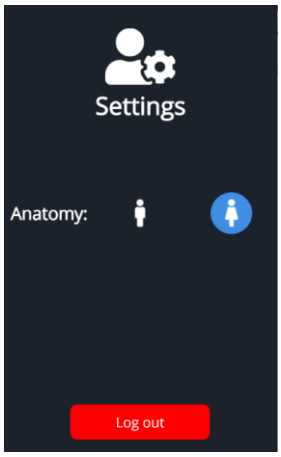


A Naptár fülön a felhasználó megtekintheti és elindíthatja előre beírt edzéseit.

Az oldal két részre osztja a beírt edzéseket, az e heti(This Week) és az összes edzés a hónapban(This Month).

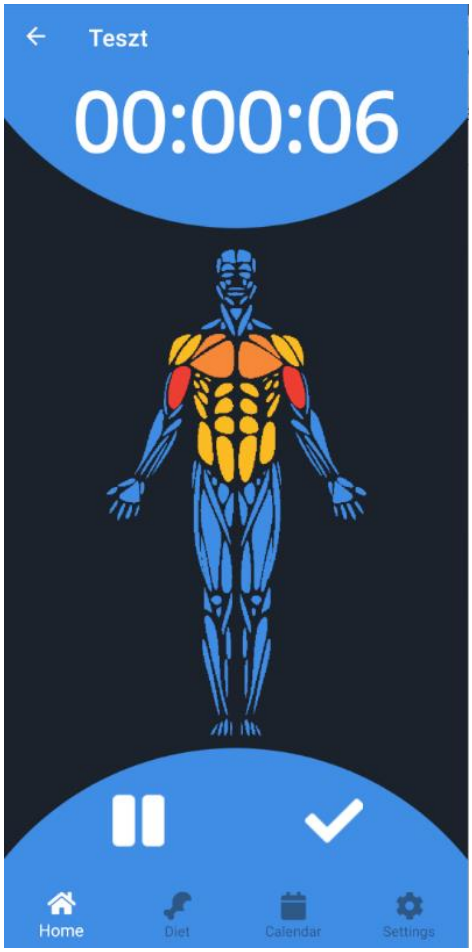
A This Week csak olyan edzéseket mutat melyek még az az napi napon vagy utána következnek be és nincsenek még leedzve.

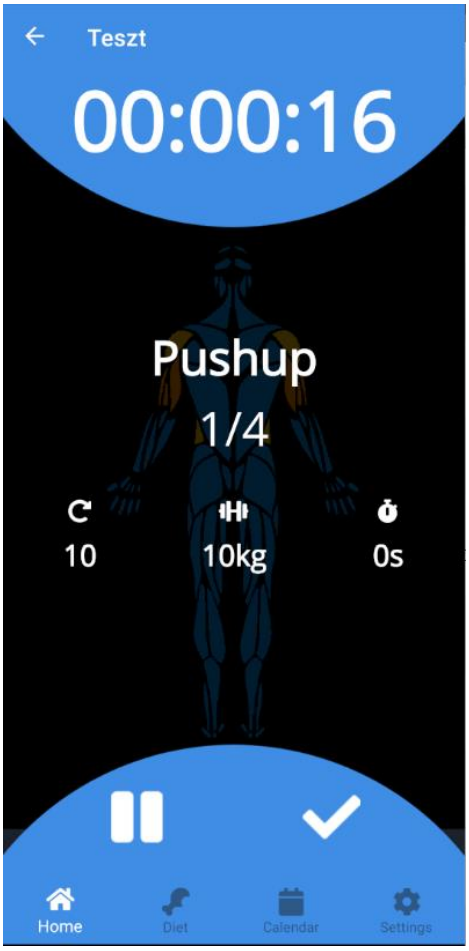
5.6 Beállítások



A jelenlegi Beállítások oldalon a felhasználó megtudja változtatni a test ábra kinézetét és ki tud jelentkezni. A közel jövőben több személyre szabási beállítás is megtalálható lesz majd itt.

5.7 Edzés





Részletek panel

Edzés elindítása után egy letisztult felület várja a felhasználót, amin könnyedén egyszerű gesztusokkal irányítható az edzés.

A képernyő közepét hosszan lenyomva felugrik a jelenlegi “Set” információja (Részletek panel).

A képernyőn való balra vagy jobbra húzással lehet léptetni az edzés “Set”-jeit ekkor is felugrik a Részletek panel.

Az edzést a Szünet gombbal meg lehet állítani ez a web alkalmazástól eltérően bármekkora pihenő lehet. A Szünet gomb mellett pedig az edzés manuális befejezése gomb található mellyel az edzés előbb letudható, ha mondjuk a felhasználó már nem tudja végig vinni valamilyen okból kifolyólag.

6. Fejlesztői futtatás

A program futtatásához a következő parancsokat kell kiadni a megfelelő helyen:

Az adatbázis létrehozásához a *flexify_final.sql* fájlt kell importálnunk a MySQL szerverünkre. Ezt bármilyen erre alkalmas eszközzel megtehetjük. Mi a phpMyAdmin importálás funkcióját vettük igénybe. Ezután a *flexify/server/.env* fájlban át kell írnunk az alábbi mezőket a megfelelő értékekre:

```
DB_HOST="<adatbázis szerver címe>"
DB_USER="<adatbázis felhasználó>"
DB_PASSWORD="<adatbázis jelszó>"
DB_DATABASE="flexify"
```

A konfigurálás után a következő parancsokat kiadva indíthatjuk el a szervert (feltételezve, hogy a parancssorunk a projekt fő mappájára mutat): `npm i`, `cd server`, `node server_final.js`. Ezen parancsok kiadása után a szerver a konzolon fog minket értesíteni a sikeres indulásról, és az adatbázishoz való sikeres csatlakozásról.

Ezután már csak a frontend elindítása maradt. Indítsunk egy újabb konzolablakot, és navigáljunk a projekt fő mappájába. Ezután el tudjuk indítani a következő paranccsal: `npm run start`

Irodalomjegyzék

Wikipedia - Discord - https://hu.wikipedia.org/wiki/Discord - 2024.04.15

React - Bevezetés a Reactbe - https://hu.legacy.reactjs.org/tutorial/tutorial.html - 2024.04.15

Rackhost - Mi az és mire jó a Node.js? - https://www.rackhost.hu/tudasbazis/informatikai-alapok/mi-az-es-mire-jo-a-node-js/ - 2024.04.15

aWh - Mi az a MySQL? - https://www.awh.hu/kb/webtarhely/mi-az-a-mysql - 2024.04.15

API Ninjas - Nutrition API - https://api-ninjas.com/api/nutrition - 2024.04.15