



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

Tanulók neve: Pálfi Gyula Milán, Kincses Tamás, Halász Péter  
Képzés: nappali munkarend  
Szak: 5 0613 12 03 Szoftverfejlesztő és tesztelő technikus

### A záródolgozat címe:

## Budget Calculator

Konzulens: Nits László  
Beadási határidő: 2023. 04. 28.

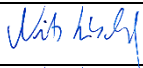
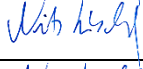
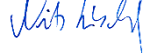
Győr, 2023. 04.28

---

**Módos Gábor**

igazgató

## Konzultációs lap

	A konzultáció		Konzulens aláírása
	ideje	témája	
1.	2022.10.01.	Témaválasztás és specifikáció	
2.	2023.02.01.	Záródolgozat készültségi fokának értékelése	
3.	2023.04.28.	Dokumentáció véglegesítése	

## Tulajdonosi nyilatkozat


Ez a dolgozat a saját munkánk eredménye. Dolgozatunk azon részeit, melyeket más sz. erzők munkájából vettünk át, egyértelműen megjelöltük.

Ha kiderülne, hogy ez a nyilatkozat valótlan, tudomásul vesszük, hogy a szakmai vizsgabizottság a szakmai vizsgáról kizár és szakmai vizsgát csak új záródolgozat készítése után tehetünk.

Győr, 2023. április 28.

Tanulók aláírásai:

  
Pálfi Gyula Milán

  
Kincses Tamás

  
Halász Péter

Jedlik Ányos Gépipari és Informatikai  
Technikum és Kollégium

Budget Calculator  
dokumentáció

Készítették:

Halász Péter

Kincses Tamás

Pálfi Gyula Milán

# Tartalomjegyzék

1. Bevezetés .....	5
1.1 A probléma és megoldás .....	5
1.2 Ötletelés .....	5
1.3 Végeredmény .....	6
1.4 Jövőbeli tervek .....	6
1.5 Csapatmunka megvalósítása .....	7
2. Fejlesztői dokumentáció .....	8
2.1 Technikai részek .....	8
2.2 Az adatbázis felépítése .....	8
2.2.1 Táblák .....	9
2.3 Authentikáció és biztonság .....	10
2.4.1 A technológia .....	10
2.4.2 Működése .....	10
2.4.3 Jelszavak .....	10
2.4.4 Jogosultságok .....	11
2.4 Backend dokumentáció .....	12
2.4.1 Config mappa .....	12
2.4.2 Controllerek .....	12
2.5 Frontend dokumentáció .....	26
2.5.1 Struktúra .....	26
2.6 Teszt .....	27
3. Felhasználói kézikönyv .....	28
3.1 Navigációs menü .....	28
3.1.1 Kezdőlap .....	28
3.2.2 Regisztráció és bejelentkezés .....	28
3.2.3 News és About oldal .....	30
3.2.4 Exchange oldal .....	31
3.2.5 Budget oldal .....	32
3.2 Fejlesztői futtatási kézikönyv .....	33
3.3 Demo .....	33

# 1. Bevezetés

## 1.1 A probléma és megoldás

A „BudgetCalculator” egy online, pénzügyi, segéd webalkalmazás.

Manapság elég nehéz az embereknek nyomon követnie hogy mennyit költenek vagy mennyit költhetnek még. A mi alkalmazásunkkal ez nagyon egyszerűvé válhat. Tablázat-szerűen nyomon követhetik, hogy mennyi volt az adott személynek kiadása/bevétele. Bármikor felvihet a felhasználó a profiljára egy adott összeget és ez segít összegezni a kívánt értékeket.

Nem ez az egyetlen funkciója az alkalmazásunknak. Árfolyamokat is lehet követni. Adott fülön, ki lehet választani, hogy milyen pénznemet szeretnénk keresni, esetleg átváltani és megmutatja az adott pénznem értékét, forintban vagy amit kiválaszt a felhasználó.

Összesítve:

- Az alkalmazás segít a költségek kezelésében,
- Árfolyam nyomon követés

A weboldal bárholnan elérhető interneten keresztül, akár mobil eszközökről, akár PC-ről is. A regisztráció, belépés és a böngészés is online történik, nem kell semmit letölteni az adott eszközre.

## 1.2 Ötletelés

Ebben a fejezetben a megfogalmazódott terveinket részletezzük.

A felhasználó részére biztosítani szeretnénk volna azt, hogy egy könnyen átlátható és menedzselhető, kiadások és bevételek oldalt hozzunk létre. Egy-egy felvitt költséghez, hozzárendelhetnek, típust és dátumot, hogy könnyen átlátható és kereshető legyen.

Az árfolyam figyelő részen, pedig egy vizuálisan vonzó oldalt próbálunk létrehozni, ahol egyszerűen és érthetően kereshet a felhasználó, az elérhető pénznemek között.

## 1.3 Végeredmény

Az elkészült program a terveink szerint készült, egy-két apróbb eltéréssel.

Eredetileg nem terveztünk grafikonokat, de úgy ítéltük, hogy vizuálisan ezek sokkal nagyobb segítséget nyújtanak a felhasználói élmény növelésében és a felhasználó számára is átláthatóbb az oldal. A mobil alkalmazásról le kellett mondanunk, komplexitás és idő problémák miatt, helyette reszponzív oldalt csináltunk, de ez nem jelenti azt, hogy kivitelezhetetlen lenne a jövőben ezt elkészíteni.

## 1.4 Jövőbeli tervek

A project kezdetekor, számtalan ötletünk volt, hogy hogyan tehetnénk felhasználó barátabbá az oldalt, de sajnos idő és megfelelő segítség hiányában nem volt erre esélyünk. Mérlegelnünk kellett azt, hogy az oldalunknak funkcionálisnak kell lennie a határidőre. A jelenlegi verzióból kimaradt ötleteket, viszont nem vetettük el, számtalan lehetőségünk és koncepciónk van, amivel az oldalt még élvezetesebbé és hasznosabbá tennénk.

Az egyik fontosabb tervünk a mobilalkalmazás elkészítése lenne, hogy a felhasználó, bárhol és bármikor, akár vásárlás után rögtön, hozzáadni képes legyen, nehogy véletlen elfelejtődjön és nehogy problémát okozzon a felhasználó jövőbeli tervezésében.

Ehhez a mobilalkalmazáshoz hozzá kapcsolva, lehetőség lenne az automatikus adat felvitelre is, amennyiben a felhasználó hozzájárul és hozzáadja a bankkártya adatait így az online vagy csak szimplán bankkártyával történő fizetés esetén, az információ automatikusan elmentésre kerül és a felhasználónak nem kell ezzel foglalkoznia, hogy manuálisan felvigye ezt.

Az oldalon monetizációs lehetőséget is látunk, reklámokon keresztül. Mivel elég sok „tőzsde” jellegű funkcionalitás létezik az oldalunkon, ezért lehetőséget látunk arra, hogy egy nagyon „broker” oldal szponzorálja az alkalmazásunk azon részét, ahol pénznemekkel dolgozhat a felhasználó, esetleg „külsős” reklámozásra is van lehetőség.

A pénznem átváltó oldalt is át lehetne dolgozni, hogy ne csak létező pénznemre tudjon rákeresni a felhasználó, hanem úgy nevezett „Stock” vagy „Részvényekkel” is képes legyen megismerkedni.

## 1.5 Csapatmunka megvalósítása

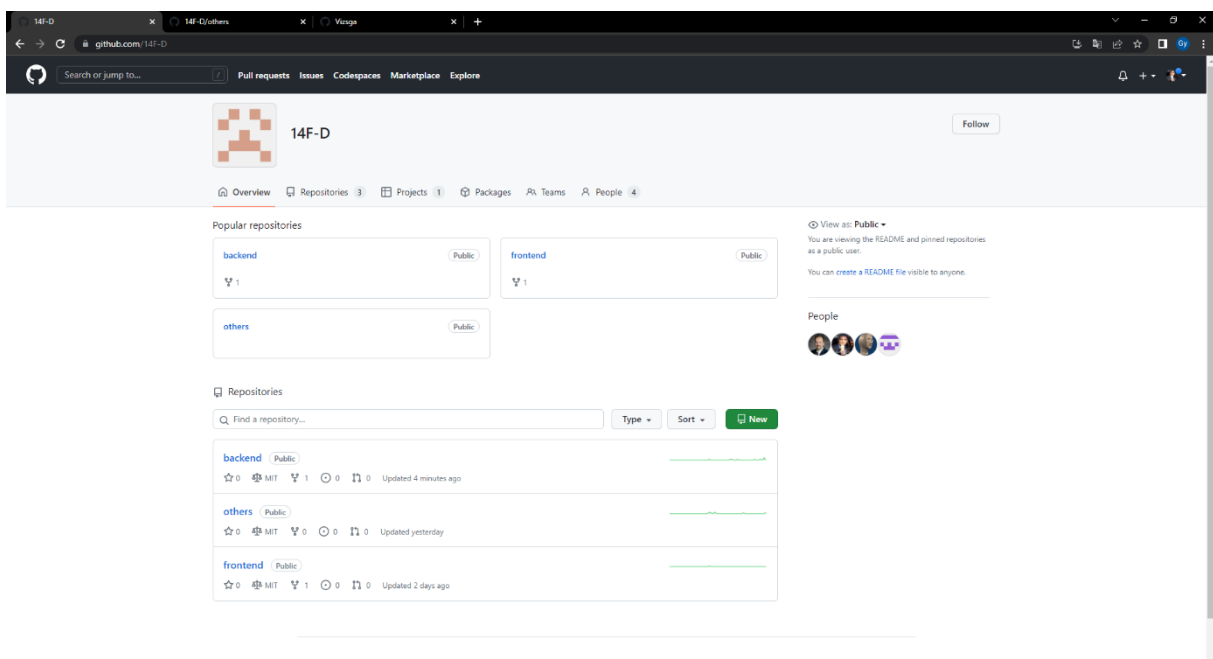
A project elkezdése előtt, nem tartottuk a legfontosabbnak, hogy kőbe vessük azt, hogy ki, mit fog csinálni az adott projectben. Mindenki dolgozott, minden részlegen, mind a hárman részt vettünk a „fullstack” webfejlesztésben, így sikerült elosztani, a részleteket.

A GitHub organizációnk elérhető a <https://github.com/14F-D> URL-en.

Az esetek 90%-ban közösen dolgoztunk, vagy legalább 2 ember kommunikált egy fájl írása közben. Egyedül ritkán dolgoztunk, szükségünk volt egymás segítségére és ha nem feltétlenül egyezett valakinek az ötlete az többiekével, akkor ott további konzultációra volt szükség amúgy is és általában újra kellett dolgozni az adott feladatot.

A GitHub-on kívül használtunk más programokat is a kommunikáció megkönnyítésére. A fő program amit használtunk az a Discord volt, ugyanis ott nagyon egyszerűen lehet egy online szerveren hanghívásban dolgozni. Az adott ember képernyőjét meg is lehet osztani a hívásban lévő többi résztvevővel is, így egyértelműen meg lehet mutatni a másiknak, hogy mit is csinált éppen az illető. Ezeken felül, több hasznos funkciója is van a Discordnak, mint például a kép és fájl küldés, írásos üzenetek küldése és több résztvevős hívások.

Minden feladat során külön branch-ben dolgoztunk, melyeket később, a helyes működés biztosítása után, Pull request segítségével töltöttünk fel az adott (pl.: frontend, backend) ágba. Ezzel elkerültük a már működő programkód hibással való felülírását, valamint így egyszerűen tudtunk egy időben dolgozni a programon.

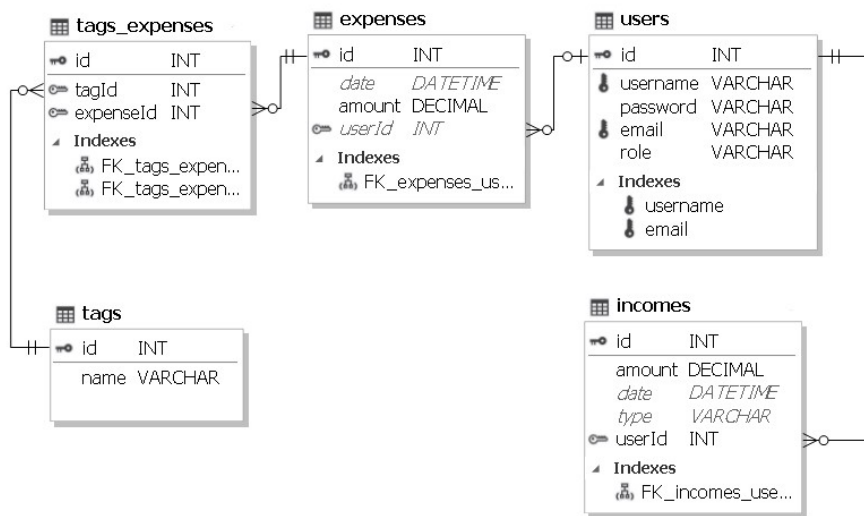


## 2. Fejlesztői dokumentáció

### 2.1 Technikai részek

A weboldalunk adatbázisa MySql technológiával készült, amire egy Node.js backendet építettünk. A frontend része pedig Vue.js nyelven lett megírva. A project kezdete előtt, mi arra a döntésre jutottunk, hogy azokat a technológiákat fogjuk használni, amelyiket mi a legkényelmesebbnek láttunk és egyszerű használni. Ez a három technológia volt nekünk a legjobban prezentálva az iskolában és ezeket értettük meg a legjobban, így úgy éreztük ezekkel lesz a legjobb esélyünk arra, hogy sikeres legyen a vizsgánk.

### 2.2 Az adatbázis felépítése





## 2.2.1 Táblák

### Users – Felhasználók

- id: elsődleges azonosító
- username: felhasználónév
- password: jelszó
- email: email cím
- role: admin vagy standard felhasználó

Tesztelés miatt le lett generálva 40 standard és 3-5 admin felhasználó

### Expenses – Kiadások

- id: elsődleges azonosító
- amount: kiadás mennyisége
- date: dátum
- userId: idegen kulcs a Users táblával, az adott felhasználó kiadásai

### Incomes – Bevételek

- id: elsődleges azonosító
- amount: kiadás mennyisége
- date: dátum
- type: típus
- userId: idegen kulcs a Users táblával, az adott felhasználó kiadásai

### Tags – Címke

- id: elsődleges azonosító
- name: címke név

### Tags\_Expenses – Tags-Expenses összekötő tábla

- id: elsődleges azonosító
- tagId: idegen kulcs a tag táblához
- expenseId: idegen kulcs az expenses táblához

## 2.3 Authentikáció és biztonság

### 2.4.1 A technológia

A weboldalon történő bejelentkezéshez az express-session session kezelőt használjuk, melyet a backend server.js fájlban definiálunk. Ebben határozzuk meg a session lejáratát és a titkos kulcsát. Az express-session egy Node.js modul, amely lehetővé teszi az időzített munkamenetek kezelését az Express keretrendszerben. A modul egyedi azonosítókat generál a felhasználói munkamenetek számára, amelyeket tárol és használ a kliens oldalon történő kérések feldolgozásakor. Az "express-session" funkciói közé tartozik a munkamenetek élettartamának beállítása, az autentikáció és az engedélyezési ellenőrzés. (ChatGPT,2023)

### 2.4.2 Működése

#### Backend oldalon:

A session létrehozása a user.controller nevű a login kérésben kezdődik. A login függvény meghívásakor az adatok ellenőrzése után a felhasználó adataival létrehozunk a hitelesítéshez szükséges session-t.

#### Frontend oldalon:

Bejelentkezéskor a frontenden a stores mappában található user.js fájlban lévő login függvény kerül meghívásra, mely a felhasználó által megadott felhasználónév és jelszó párosítást küldi tovább a backend felé. A válaszként kapott session sütit a süti közé, illetve az aktuálisan bejelentkezett felhasználó szükséges adatai a böngésző lokális tárolóján kapnak helyet.

A HTTP kérések során a backenden az auth.js fájlban található két függvény ellenőrzi, hogy a felhasználónak van-e joga elérni azt az adott útvonalat.

### 2.4.3 Jelszavak

Sikeres regisztráció után a backendre érkező jelszó bcrypt npm csomag használatával titkosítva kerül tárolásra az adatbázisban. A felhasználó által megadott titkosítatlan jelszó nem kerül tárolásra az adatbázisban. A lekérések során ez a titkosított jelszó semmiféle formában nem kerül visszaküldésre a frontend irányába az esetleges támadások végett

## 2.4.4 Jogosultságok

Kétféle jogosultság alapján vannak a felhasználók besorolva: Standard és Admin. Regisztrációkor a felhasználó automatikusan standard joggal kerül tárolásra. Admin joggal csak a rendszergazdák rendelkezhetnek.

**Standard** típusú felhasználó:

A felhasználó regisztrációt követően kiadásokat és bevételeket adhat hozzá az ő eddig eltárolt adataihoz, emellett akarata szerint módosíthatja és böngészheti a már tárolt kiadásait és bevételeit.

**Admin** típusú felhasználó:

Az Admin típusú felhasználó felvehet újabb kategóriákat, emellett törölhet, vagy kérésre módosíthatja azokat.

## 2.4 Backend dokumentáció

### 2.4.1 Config mappa

```
JS db.js ×
config > JS db.js > ...
1  const mysql = require("mysql");
2
3  const connection = mysql.createPool({
4    host: '127.0.0.1',
5    user: 'root',
6    password: '',
7    database: 'budgetcalculator',
8    dateStrings: true
9  });
10
11 module.exports = connection;
```

Itt hívjuk meg a „budgetcalculator” nevű adatbázisunkat.

### 2.4.2 Controllerek

#### 2.4.2.1 Expenses controller

Elérési útvonal: /expenses

**GetAllExpenses(req, res)**

Elérési útvonal: /

Kérés típusa: HttpGet

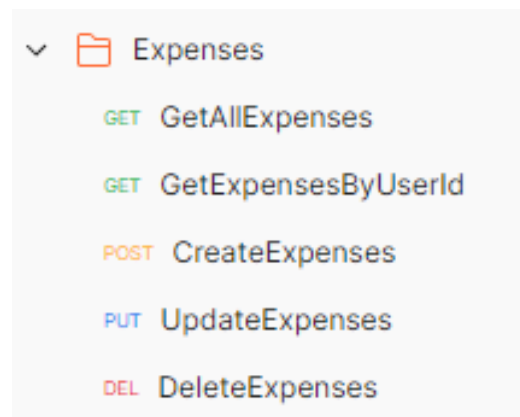
Paraméter: request, response

Jogosultság: Admin

Feladata: Kilistázza az összes kiadást

Válasz: „expenses” tömb, ha sikeres (data),

message: „Unknown error” 500-as hiba esetén



**GetExpensesByUserId(req, res)**

Elérési útvonal: /:id

Kérés típusa: HttpGet

Paraméter: request, response

Jogosultság: Admin, Standard

Feladata: Kilistázza az adott id-vel rendelkező felhasználó kiadásait

Válasz: az adott id „expenses” tömbje, ha sikeres (data),

message: „Unknown error” 500-as hiba esetén

message: „Not found” 404-es hiba esetén

**create(req, res)**

Elérési útvonal: /create

Kérés típusa: HttpPost

Paraméter: request, response

Jogosultság: Admin, Standard

Feladata: Létrehoz egy új kiadást és hozzárendel egy id-t

Válasz: újonnan létrehozott id, „newExpenses” tömb, ha sikeres,

message: „Unknown error” 500-as hiba esetén

### **update(req, res)**

Elérési útvonal: /update/:id

Kérés típusa: HttpPut

Paraméter: request, response

Jogosultság: Admin, Standard

Feladata: Módosítja a kiadást a bejelentkezett felhasználónak

Válasz: az adott idnek az „expenses” tömbje ha sikeres,

message: „Unknown error” 500-as hiba esetén

message: „Not found expense with id: (id)” 404-es hiba esetén

### **delete(req, res)**

Elérési útvonal: /delete/:id

Kérés típusa: HttpDelete

Paraméter: request, response

Jogosultság: Admin, Standard

Feladata: Kitörli a kiválasztott kiadást a bejelentkezett felhasználónak

Válasz: message: „Expense was successfully deleted!” ha sikeres, törlés

message: „Unknown error” 500-as hiba esetén

message: „Not found expense with id: (id)” 404-es hiba esetén

```
const expenses = require('../controllers/expenses.controller');
router.get('/expenses', adminAuth, expenses.getAllExpenses);
router.get('/expenses/:id', userAuth, expenses.getExpensesById);
router.post('/expenses/create', userAuth, expenses.create);
router.put('/expenses/update/:id', userAuth, expenses.update);
router.delete('/expenses/delete/:id', userAuth, expenses.delete);
```

### 2.4.2.2 Incomes controller

Elérési útvonal: /incomes

#### **GetAllIncomes(req, res)**

Elérési útvonal: /

Kérés típusa: HttpGet

Paraméter: request, response

Jogosultság: Admin

Feladata: Kilistázza az összes bevételt

Válasz: „incomes” tömb, ha sikeres (data),

message: „Unknown error” 500-as hiba esetén

#### **GetIncomesByUserId(req, res)**

Elérési útvonal: /:id

Kérés típusa: HttpGet

Paraméter: request, response

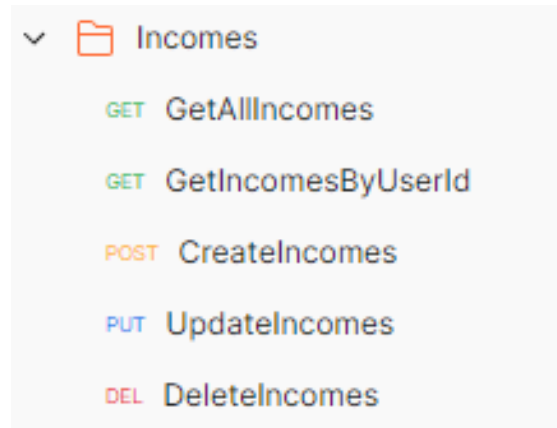
Jogosultság: Admin, Standard

Feladata: Kilistázza az adott id-vel rendelkező felhasználó bevételeit

Válasz: az adott idnek az „incomes” tömbje, ha sikeres (data),

message: „Unknown error” 500-as hiba esetén

message: „Not found” 404-es hiba esetén



**create(req, res)**

Elérési útvonal: /create

Kérés típusa: HttpPost

Paraméter: request, response

Jogosultság: Admin, Standard

Feladata: Létrehoz egy új kiadást és hozzárendel egy id-t

Válasz: újonnan létrehozott id és „newIncomes” tömb ha sikeres,

message: „Unknown error” 500-as hiba esetén

**update(req, res)**

Elérési útvonal: /update/:id

Kérés típusa: HttpPut

Paraméter: request, response

Jogosultság: Admin, Standard

Feladata: Módosítja a bevételt a bejelentkezett felhasználónak

Válasz: az adott idnek az „incomes” tömbje, ha sikeres,

message: „Unknown error” 500-as hiba esetén,

message: „Not found income with id: (id)” 404-es hiba esetén



### **delete(req, res)**

Elérési útvonal: /delete/:id

Kérés típusa: HttpDelete

Paraméter: request, response

Jogosultság: Admin, Standard

Feladata: Kitörli a kiválasztott bevételt a bejelentkezett felhasználónak

Válasz: „Income was successfully deleted!”, ha sikeres, törlés

message: „Unknown error” 500-as hiba esetén,

message: „Not found income with id: (id)” 404-es hiba esetén

```
const incomes = require('../controllers/incomes.controller');
router.get('/incomes', adminAuth, incomes.getAllIncomes);
router.get('/incomes/:id', userAuth, incomes.getIncomesById);
router.post('/incomes/create', userAuth, incomes.create);
router.put('/incomes/update/:id', userAuth, incomes.update);
router.delete('/incomes/delete/:id', userAuth, incomes.delete);
```

### **2.4.2.3 User controller**

Elérési útvonal: /users

#### **getAllUsers(req, res)**

Elérési útvonal: /

Kérés típusa: HttpGet

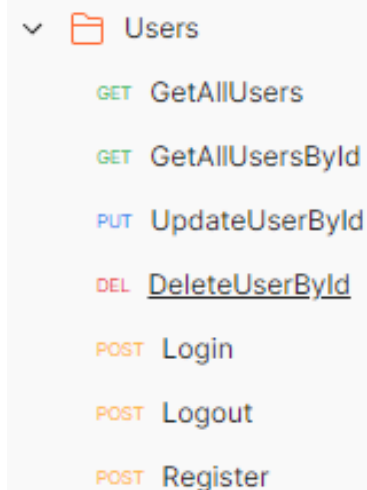
Paraméter: request, response

Jogosultság: Admin

Feladata: Kijelölt az összes regisztrált felhasználót

Válasz: „users” tömb, ha sikeres (data),

message: „Unknown error” 500-as hiba esetén



Users

- GET GetAllUsers
- GET GetAllUsersById
- PUT UpdateUserById
- DEL DeleteUserById
- POST Login
- POST Logout
- POST Register

**getUsersById(req, res)**

Elérési útvonal: /:id

Kérés típusa: HttpGet

Paraméter: request, response

Jogosultság: Admin, Standard

Feladata: Kिलistázza a megadott id-vel rendelkező felhasználót

Válasz: „users” tömb, ha sikeres (data),

message: „Unknown error” 500-as hiba esetén

message: „Not found” 404-es hiba esetén

**register(req, res)**

Elérési útvonal: /register

Kérés típusa: HttpPost

Paraméter: request, response

Jogosultság: Admin, Standard

Feladata: Regisztrál egy felhasználót. Hozzárendel egy id-t, a felhasználó által megadott felhasználónevet, email-t és jelszót, hashelve (titkosítva).

Válasz: message: „An error occurred” 500-as hiba esetén

message: „User registered successfully” 200-as response esetén és elmenti a felhasználó adatait (felhasználónév, jelszó, email)

### **login(req, res)**

Elérési útvonal: /login

Kérés típusa: HttpPost

Paraméter: request, response

Jogosultság: Admin, Standard

Feladata: Bejelentkezteti a felhasználót. Ellenőrzi a megadott felhasználónevet és jelszót és ha sikeres az adatmegadás, akkor bejelentkezik.

Válasz: Létrehoz egy session-t, a felhasználó id-jével, felhasználónevével és szerepkörével

message: „Username and passwords are required” 400-as hiba esetén

(A felhasználó üresen hagyta a mezők bármelyikét, nem történik bejelentkezés)

message: „Invalid username or password” 401-es hiba esetén

(A felhasználó rossz adatokat adott meg, nem történik bejelentkezés)

message: „Logged in successfully” sikeres bejelentkezés esetén és tovább küldi a user tömböt

### **logout(req, res)**

Elérési útvonal: /logout

Kérés típusa: HttpPost

Paraméter: request, response

Jogosultság: Admin, Standard

Feladata: Kijelentkezteti a felhasználót.

Válasz: „Logged out successfully” message, ha sikeres a kijelentkezés és tovább küldi a user tömböt

„Unauthorized”, ha sikertelen a kijelentkezés

### **update(req, res)**

Elérési útvonal: /:id

Kérés típusa: HttpPut

Paraméter: request, response

Jogosultság: Admin, Standard

Feladata: Módosítja a felhasználó adatait.

Válasz: továbbküldi az id-t és a user tömböt

message: „Unknown error” 500-as hiba esetén

message: „Not found user with id: (id)” 404-es hiba esetén

### **delete(req, res)**

Elérési útvonal: /:id

Kérés típusa: HttpDelete

Paraméter: request, response

Jogosultság: Admin, Standard

Feladata: Kitörli a felhasználót és adatait az adatbázisból

Válasz: Ha a felhasználó nem admin, csak a saját profilját tudja törölni, az admin jogosultságú felhasználók másokét is tudja törölni id alapján.

Sikeres törlés esetén: message: „User was succesfully deleted!”

message: „Not found user with id: (id)” 404-es kód esetén.

```
const users = require('../controllers/user.controller');
router.get('/users', adminAuth, users.getAllUsers);
router.post('/register', users.register);
router.put('/users/:id', userAuth, users.update);
router.delete('/users/:id', userAuth, users.delete);
router.get('/users/:id', userAuth, users.getUsersById);
router.post('/login', users.login);
router.post('/logout', users.logout);
```

**function RegisterValidate(req, res)**

Paraméter: request, response

Feladata: Hitelesíti a regisztrációt

Sikeres regisztrációhoz kötelező:

- Megadni a felhasználó nevet ami legalább 4, de maximum 15 karakter hosszú lehet
- Megadni jelszót ami legalább 8, de maximum 20 karakter hosszú lehet
- Megadni egy valid e-mail címet

Ha ezek közül bármelyik is nem megfelelő, vagy üres valamelyik mező, egy üzenetet ír ki a felhasználónak az adott problémával.

```
function RegisterValidate(req, res) {  
  //console.log(req.body)  
  if (JSON.stringify(req.body) == '{}') {  
    res.status(400).send({  
      message: 'Content cannot be empty!'  
    });  
    return true;  
  }  
  if (req.body.username == '' || req.body.username == undefined) {  
    res.status(400).send({  
      message: 'Username required!'  
    });  
    return true;  
  }  
  else if (validator.isLength(req.body.username, {min: 4, max: 15}) == false) {  
    res.status(400).send({  
      message: 'Username has to be min 4 character, max 15 character'  
    });  
    return true;  
  }  
  if (req.body.password == '' || req.body.password == undefined) {  
    res.status(400).send({  
      message: 'Password required!'  
    });  
    return true;  
  }  
  else if (validator.isLength(req.body.password, {min: 8, max: 20}) == false) {  
    res.status(400).send({  
      message: 'Password has to be min 8 character, max 20 char'  
    });  
    return true;  
  }  
  if (req.body.email == '' || req.body.email == undefined) {  
    res.status(400).send({  
      message: 'Email required!'  
    });  
    return true;  
  }  
  else if (validator.isEmail(req.body.email) == false) {  
    res.status(400).send({  
      message: 'Email is not valid'  
    });  
    return true;  
  }  
}
```

#### 2.4.2.4 Tags controller

Elérési útvonal: /tags

**getAllTags(req, res)**

Elérési útvonal: /

Kérés típusa: HttpGet

Paraméter: request, response

Feladata: Kilistázza az összes címkét

Válasz: „tags” tömb, ha sikeres (data),

message: „Unknown error” 500-as hiba esetén

**getTagById(req, res)**

Kérés típusa: HttpGet

Paraméter: request, response

Feladata: Kilistázza az adott id-vel rendelkező címkét

Válasz: az adott id „tags” tömbje, ha sikeres (data),

message: „Unknown error” 500-as hiba esetén

message: „Not Found” 404-es hiba esetén

**create(req, res)**

Elérési útvonal: /create

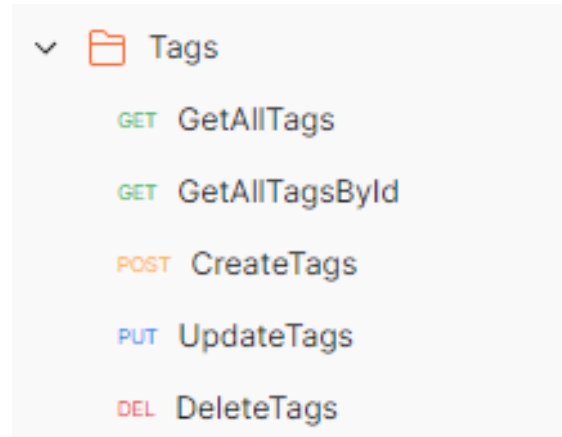
Kérés típusa: HttpPost

Paraméter: request, response

Feladata: Létrehoz egy új taget és hozzárendel egy id-t

Válasz: újonnan létrehozott id és „newTag” tömb ha sikeres,

message: „Unknown error” 500-as hiba esetén



### **update(req, res)**

Elérési útvonal: /update/:id

Kérés típusa: HttpPut

Paraméter: request, response

Feladata: Módosítja a taget, id alapján

Válasz: az adott id „tags” tömbje, ha sikeres,

message: „Unknown error” 500-as hiba esetén,

message: „Not found tag with id: (id)” 404-es hiba esetén

### **delete(req, res)**

Elérési útvonal: /delete/:id

Kérés típusa: HttpDelete

Paraméter: request, response

Feladata: Kitörli a kiválasztott taget, id alapján

Válasz: „Tag was successfully deleted!”, ha sikeres, törlés

message: „Unknown error” 500-as hiba esetén,

message: „Not found tag with id: (id)” 404-es hiba esetén

```
const tags = require('../controllers/tags.controller');
router.get('/tags', tags.getAllTags);
router.get('/tags/:id', tags.getTagById);
router.post('/tags/create', tags.create);
router.put('/tags/update/:id', tags.update);
router.delete('/tags/delete/:id', tags.delete);
```

### 2.4.2.5 Tags\_Expenses controller

Elérési útvonal: /tagsExpenses

#### **getAll (req, res)**

Elérési útvonal: /

Kérés típusa: HttpGet

Paraméter: request, response

Feladata: Kilistázza az összes címkét és ahhoz tartozó kiadásokat

Válasz: „tagsExpenses” tömb, ha sikeres (data),

message: „Unknown error” 500-as hiba esetén

#### **getByTagId(req, res)**

Kérés típusa: HttpGet

Paraméter: request, response

Feladata: Kilistázza az adott tag id-vel rendelkező kiadást

Válasz: az adott id „tagsExpense” tömbje, ha sikeres (data),

message: „Unknown error” 500-as hiba esetén

message: „Not Found” 404-es hiba esetén

#### **getByExpenseId(req, res)**

Kérés típusa: HttpGet

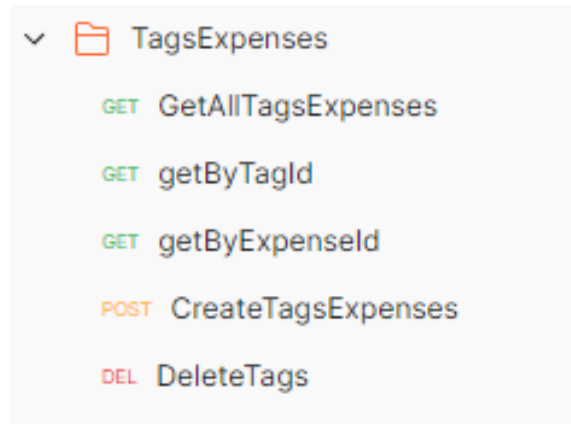
Paraméter: request, response

Feladata: Kilistázza az adott expense id-vel rendelkező címkét

Válasz: az adott id „tagsExpense” tömbje, ha sikeres (data),

message: „Unknown error” 500-as hiba esetén

message: „Not Found” 404-es hiba esetén





### **create(req, res)**

Elérési útvonal: /create

Kérés típusa: HttpPost

Paraméter: request, response

Feladata: Létrehoz egy új tag\_expense-t és hozzárendel egy id-t

Válasz: újonnan létrehozott id és „newTagsExpenses” tömb ha sikeres,

message: „Unknown error” 500-as hiba esetén

### **delete(req, res)**

Elérési útvonal: /delete/:id

Kérés típusa: HttpDelete

Paraméter: request, response

Feladata: Kitörli a kiválasztott tag\_expense-t, id alapján

Válasz: „TagExpense was successfully deleted!”, ha sikeres, törlés

message: „Unknown error” 500-as hiba esetén,

message: „Not found tagExpense with id: (id)” 404-es hiba esetén

```
const tagsExpenses = require('../controllers/tags_expenses.controller');
router.get('/tagsExpenses', tagsExpenses.getAll);
router.get('/tagsExpenses/tag/:tagId', tagsExpenses.getByTagId);
router.get('/tagsExpenses/expense/:expenseId', tagsExpenses.getByExpenseId);
router.post('/tagsExpenses/create', tagsExpenses.create);
router.delete('/tagsExpenses/delete/:id', tagsExpenses.delete);
```

## 2.5 Frontend dokumentáció

### 2.5.1 Struktúra

Az oldalunkat Vue.Js technológiával csináltuk, a különböző funkciók, különböző mappákba vannak rendezve.

Első mappa az assets mappa, amiben a képek és a script.js fájl van.

Következő a components mappa. Itt a 3 Modal.vue fájl van, ezeknek köszönhetőek a felugró ablakok, amikről részletesebben a Felhasználói kézikönyvben fogunk írni.

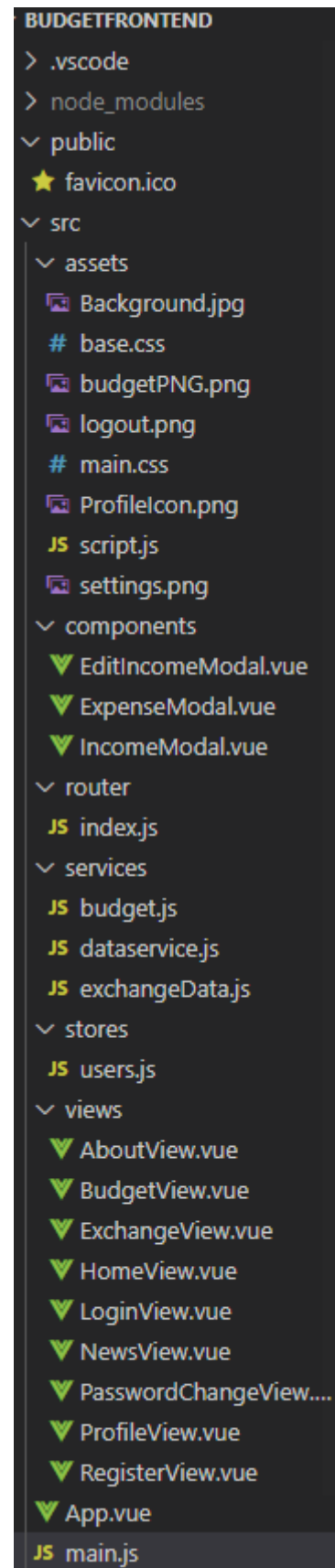
Következő a router mappa, benne az index.js fájl, ami a navigációt kezeli az oldalak között.

Következő a services mappa, benne 3 különböző fájl. A „budget.js” a BudgetView oldalon szereplő funkciókat látja el. A „dataservice.js” biztosítja a backend szerverhez való csatlakozást. Az „exchangeData.js” pedig a külső API-val kommunikál és átváltja az ExchangeView-hoz a pénznemeket.

Következik a stores mappa, benne a users.js fájl. Ez a fájl hívja meg a users API-kat és kezeli a felhasználói adatokat a frontend részen.

A views mappa a legfontosabb az összes közül, ezekben találhatóak maguk a weboldalak. Minden fájlnak a funkcionalitásáról, később, a Felhasználói kézikönyvben fogunk részletesebben írni.

Maradt még a mappákon kívül az App.vue fájlunk, ami a navbar-t kezeli, illetve a main.js, ami a plugin-ok kezelésére szolgál.



## 2.6 Teszt

A backend tesztet Jest-el végeztük. A Jest egy JavaScript alapú tesztkeretrendszer. A célja, hogy egyszerű és hatékony megoldást nyújtson a JavaScript alkalmazások teszteléséhez. A Jest számos funkcióval rendelkezik, mint például az automatikus tesztfuttatás, a snapshot tesztek és az aszinkron kódok tesztelése. A Jest támogatja az összes gyakori JavaScript keretrendszert, mint például a React, a Vue és az Angular. A tesztek futtatása egyszerűen történik a parancssorból, és az eredmények részletes jelentéssel jelennek meg.

```
const axios = require('axios')
const url = 'http://localhost:3000'

describe("OK", () => {
  test('Base route', async () => {
    const res = await axios.get(url)
    expect(res.status).toBe(200)
    expect(res.data).toEqual({message: 'ok'})
  })
})

describe("POST /api/login and /api/logout", () => {
  test("should login as standard and return data of user", async () => {
    const res = await axios.post(`${url}/api/login`, {
      username: 'TestStandard123',
      password: 'TestStandard123'
    });
    expect(res.status).toBe(200);
  });
  test("should log out", async () => {
    const res = await axios.post(`${url}/api/logout`);
    expect(res.status).toBe(200);
  });
  test("should login as admin and return data of user", async () => {
    const res = await axios.post(`${url}/api/login`, {
      username: 'TestAdmin123',
      password: 'TestAdmin123'
    });
    expect(res.status).toBe(200);
  });
});
```

**PASS** tests/user.test.js

OK

- ✓ Base route (34 ms)
- POST /api/login and /api/logout
  - ✓ should login as standard and return data of user (924 ms)
  - ✓ should log out (4 ms)
  - ✓ should login as admin and return data of user (50 ms)
- GET /api/users
  - ✓ should get all user's data if role admin (3 ms)

Test Suites: 1 passed, 1 total  
 Tests: 5 passed, 5 total  
 Snapshots: 0 total  
 Time: 1.317 s, estimated 2 s  
 Ran all test suites.

## 3. Felhasználói kézikönyv

Szeretnénk felhívni a figyelmet arra, hogy a következő részlegben található képek, a weblap fejlesztése közben készültek, még nem véglegesek az oldal kinézeti tényezői, de az oldal funkcionalitása nem fog változni!

### 3.1 Navigációs menü

#### 3.1.1 Kezdőlap

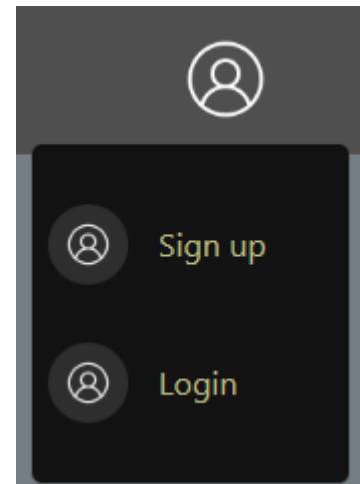
Az oldal betöltésekor a felhasználó a kezdőlapot fogja látni. Az oldal tartalmaz linkeket a további oldalakhoz, illetve egy logót a bal oldalon, ami szintén a kezdőlapra irányít vissza, illetve egy ikont a jobb oldalon, ami a profilkezeléshez irányítja a felhasználót. Itt tud bejelentkezni és regisztrálni, később adatokat módosítani. Ha a felhasználó sikeresen regisztrált és belépett az oldalra, akkor megjelenik még egy link a navigációs sávon, ami csak akkor lesz elérhető, ha be van jelentkezve az illető.



1. ÁBRA - NAVIGÁCIÓS MENÜ

#### 3.2.2 Regisztráció és bejelentkezés

A felhasználónak be kell jelentkeznie az oldalra, hogy hozzáférése legyen az oldalunk összes funkciójához, ugyanis a „Budget” oldal csak akkor jelenik meg ha a felhasználó be van jelentkezve. A jobb sarokban lévő ikonra kattintva, ez a leugró menü fog megjeleníteni.



2. ÁBRA - PROFIL FÜL  
BEJELENTKEZÉS ELŐTT

**Register**

Username  
username

Email address  
email@address.com

Password  
password123

Already have an account? [Login](#)

**REGISTER**

**Login**

Username  
username

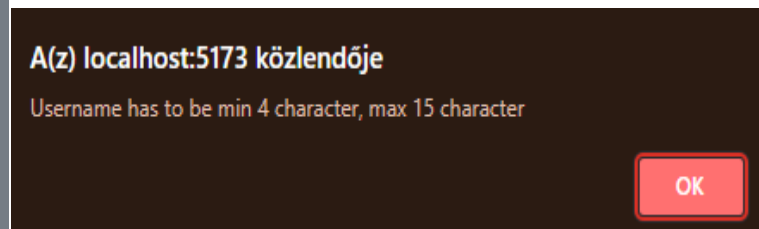
Password  
password123

You don't have an account? [Register](#)

**LOGIN**

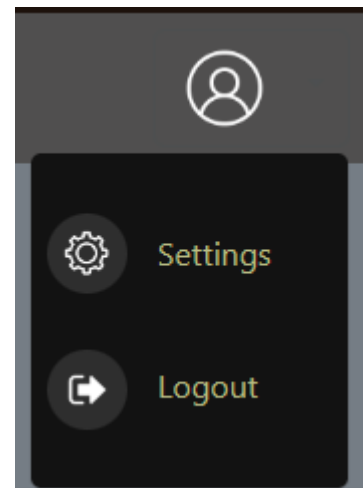
3. ÁBRA - REGISZTRÁCIÓ ÉS  
BEJELENTKEZÉS OLDAL

A felhasználó regisztrációja során kötelező megadnia egy hiteles e-mail címet, felhasználó nevet és egy jelszót. A felhasználó névnek legalább 4, de maximum 15 karakternek kell lennie. A jelszónak legalább 8, de maximum 20 karakternek kell lennie. Az emailnek pedig tartalmaznia kell legalább egy @ jelet. A lent látható képen látszódik, hogy egy alert boxot fog feldobni a felhasználónak, ha bármi hiba történik a regisztráció vagy bejelentkezés során.



4. ÁBRA - ERROR ALERT

Miután regisztrált és bejelentkezett a felhasználó megváltozik a navigációs sáv és a profil box is. Megjelenik a profil fölött a „Logout” és a „Settings” opció. Ha a felhasználó szeretné meg változtatni a felhasználó nevét, jelszavát vagy email címét, akkor ezt itt tudja megtenni.

5. ÁBRA - PROFIL MENÜ  
BEJELENTKEZÉS UTÁN

**Budget Calculator** Home News Budget Exchange Rates About

**Your Details**

Profile

Password

**Account**

Username  
Jimmy

Email  
palli.gyula.milan@stude

**Update Account**

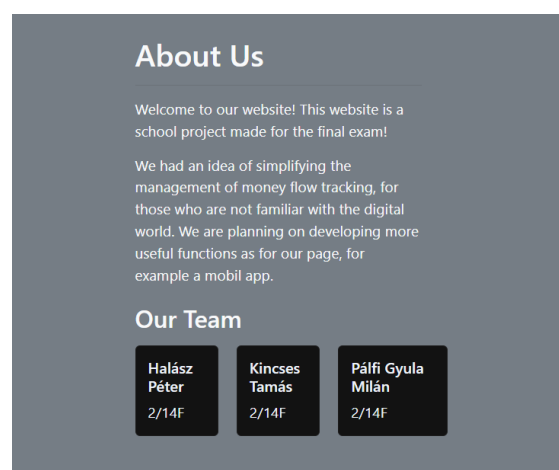
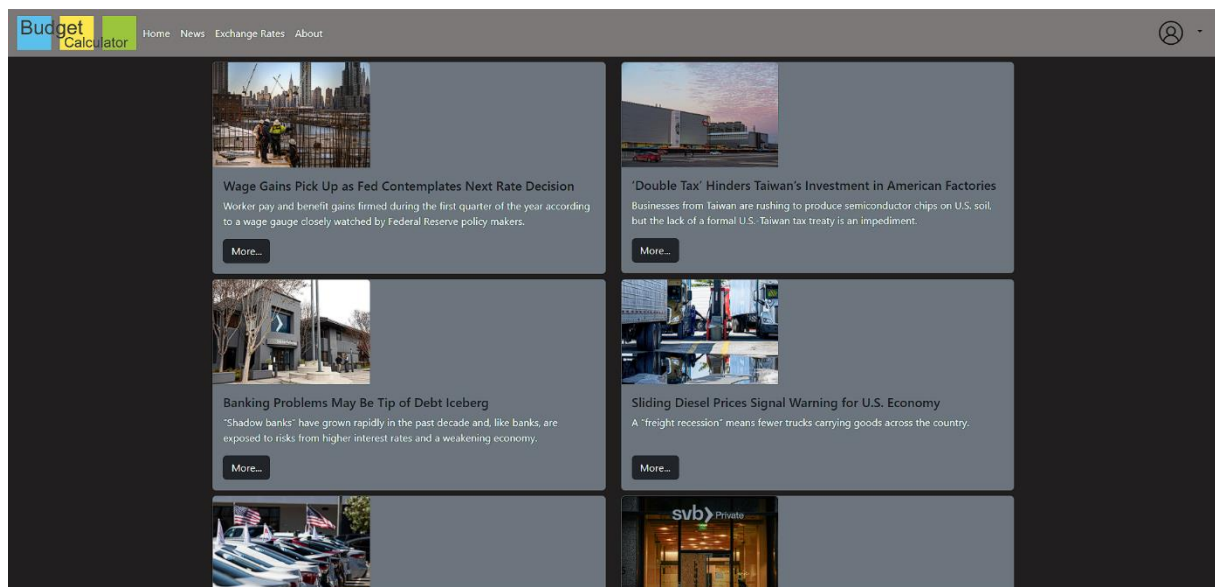
6. ÁBRA - PROFIL OPTIONS OLDAL

### 3.2.3 News és About oldal

Készítettünk két statikus oldalt, ahol pénzügyekkel foglalkozó híreket és egy kevés információt láthatnak rólunk az felhasználók.

A képen látható elrendezés alapján kilistázza a „News” oldalunk a legutóbbi híreket, a „More...” gombra kattintva egy másik oldalra irányít át (<https://www.wsj.com>), ahol részletesebben olvashatnak az adott cikkről.

Az „About” oldal pedig egy olyan oldal, ahol rólunk, az oldal készítőiről olvashatnak egy keveset.



### 3.2.4 Exchange oldal

Ez az oldal egy pénzváltó oldal, ahol a felhasználó közel 100 pénznem közül választhat, hogy mit szeretne átváltani és milyen valutában szeretné megtekinteni az értékét. Az oldalon történő lekérés egy külsős API segítségével történik ([https://apilayer.com/marketplace/exchangerates\\_data-api](https://apilayer.com/marketplace/exchangerates_data-api)), ami miatt az adott árfolyam mindig naprakész és a legújabb adatot fogja mutatni.

The screenshot displays a web interface for currency exchange. It features two dropdown menus for selecting currencies, a text input for the amount, and a display showing the conversion result. The first dropdown is set to 'EUR' and the second to 'HUF'. The amount '5' is entered in the input field. Below the input, the text 'Amount' is displayed. The conversion result is shown as '5 EUR = 1866 HUF'. A green 'Show' button is located at the bottom of the form.

Select the currency below,  
you would like to change!

EUR

Amount

5

Select the currency below,  
you would like to exchange  
into!

HUF

Amount

5 EUR = 1866 HUF

Show

### 3.2.5 Budget oldal

A budget oldal csak azután érhető el, miután a felhasználó bejelentkezett, ugyanis itt személyre szabott információkat tartalmaz az oldal. Itt tudja kezelni a felhasználó a bevételeit és kiadásait.

Mikor elsőre belép a felhasználó a táblázata üres, ugyanis neki kell felvinnie az adatokat az „Add new” gombra kattintva. Itt egy mennyiséget, típust, illetve egy dátumot kell megadnia a felhasználónak. Miután hozzáadott egy kiadást vagy bevételt, még később tudja módosítani az adatokat az „Edit” gombra kattintva, a sor végén. Egy ugyan olyan ablak fog felugrani, mint amikor felvesz egy adatot, de a gombnak a szövege „Edit income/expense” lesz. Az expenses felvételénél viszont nem típust kell kiválasztani hanem egy több címke közül tud választani és egyszerre többet is hozzárendelhet egy adathoz. Ha adatot szeretne törölni, akkor pedig a „Delete” gombra kattintva, megteheti.

Budget Calculator

HomeNewsBudgetExchange RatesAbout

Expenses

AMOUNT	TYPE	DATE		
60135	Movies Tools	2021-08-07 15:16:36	Edit	Delete
30512	Party Tools	2021-02-14 08:52:36	Edit	Delete
108707	Tools	2022-11-26 14:30:50	Edit	Delete
103178	Outdoors	2021-09-30 22:39:00	Edit	Delete
200	Movies Outdoors	2222-02-22 00:00:00	Edit	Delete
Add new				

Income

AMOUNT	TYPE	DATE		
126744	salary	2023-04-27 00:00:01	Edit	Delete
1234	bonus	2023-04-24 16:11:28	Edit	Delete
4123	transfer	2023-04-24 16:12:41	Edit	Delete
232	transfer	2023-04-24 16:21:10	Edit	Delete
50	salary	2000-10-01 00:00:00	Edit	Delete
123	transfer	2023-04-24 16:33:13	Edit	Delete
123123	salary	2000-10-02 00:00:00	Edit	Delete
123	bonus	1004-02-01 00:00:00	Edit	Delete
Add new				

New Income

Income amount:

1500

Type:

salary

Date:

2023. 04. 28.

Cancel

Add new income



## 3.2 Fejlesztői futtatási kézikönyv

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

A backendet el kell indítani, amihez a következő parancsok kellenek:

- `npm i` (telepíti a függőségeket)
- `npx nodemon ./server.js` (elindítja a szerveret a localhost:3000-es porton)

Következik a frontend rész, amihez a következő terminál parancsok kellenek:

- `npm i` (telepíti a függőségeket)
- `npm run dev` (elindítja a weboldalt)

## 3.3 Demo

Készítettünk példa felhasználókat, amelyekkel kipróbálhatók a különböző funkciók. Ezekhez a bejelentkezési adatok a következők:

Típus	Username	Jelszó
Standard	TestStandard123	TestStandard123
Admin	TestAdmin123	TestAdmin123