**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:

Beadási határidő: 2023. 04. 28.

Győr, 2023. 04.18

**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. | 2023.04.18. | 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 és szakmai vizsgát csak új záródolgozat készítése után tehetünk.

Győr, 2023. április 18.

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](#_Toc133391076)

[1.1 A probléma és megoldás 5](#_Toc133391077)

[1.2 Ötletelés 5](#_Toc133391078)

[1.3 Végeredmény 6](#_Toc133391079)

[1.4 Jövőbeli tervek 6](#_Toc133391080)

[1.5 Csapatmunka megvalósítása 7](#_Toc133391081)

[2. Fejlesztői dokumentáció 8](#_Toc133391082)

[2.1 Technikai részek 8](#_Toc133391083)

[2.2 Az adatbázis felépítése 8](#_Toc133391084)

[2.2.1 Táblák 9](#_Toc133391085)

[2.3 Authentikáció és biztonság 10](#_Toc133391086)

[2.4 Backend dokumentáció 11](#_Toc133391087)

[2.4.1 Config mappa 11](#_Toc133391088)

[2.4.2 Controllerek 11](#_Toc133391089)

[2.4.2.3 User controller 16](#_Toc133391090)

# 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 bárhonnan 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 szerettük 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 barátibbá 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 élvezetessebbé 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

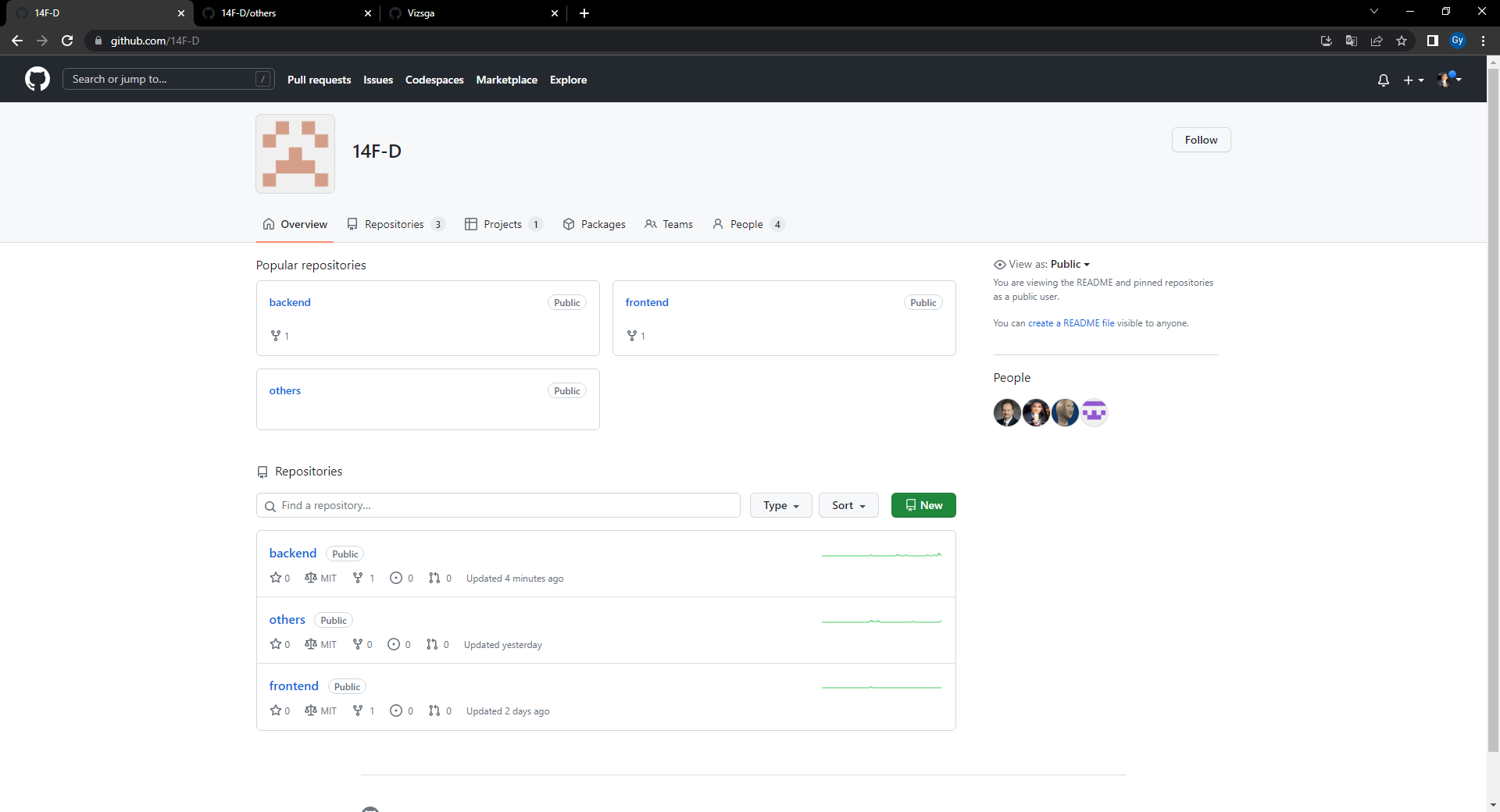
A project elkezdése előtt, nem tartottuk a legfontosabbnak, hogy kőbe véssük azt, hogy ki, mit fog csinálni az adott projectben. Mindenki dolgozott, minden részlegen, mind a hárman részt vettünk a „fullstack” webfejlesztésben, így sikerült elosztani, a részleteket.

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 fáljküldés, 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.



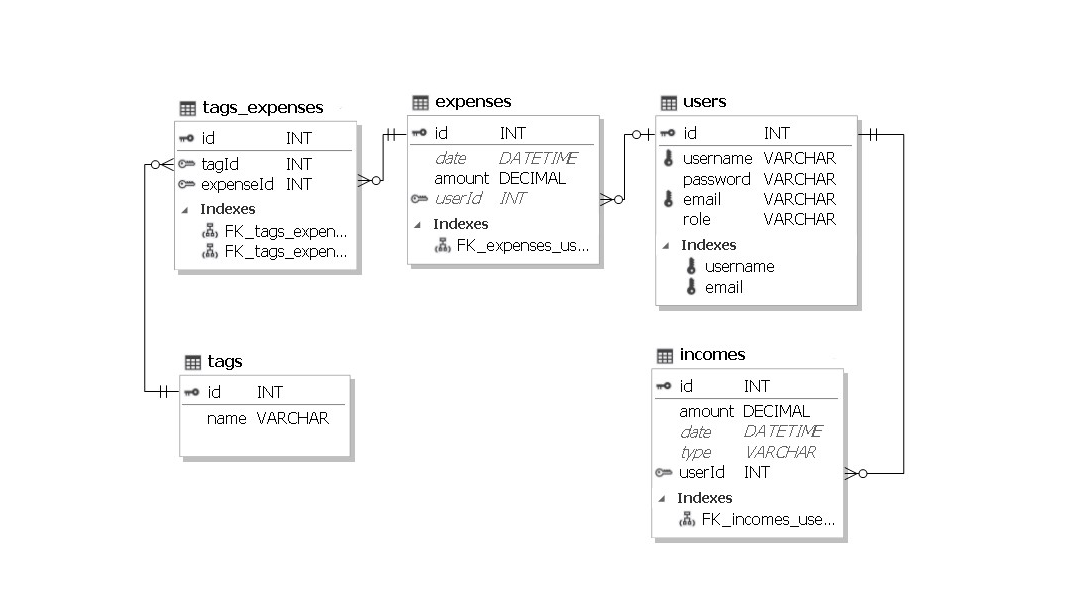
Az organizációnk mappa struktúrája

# 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, igy 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

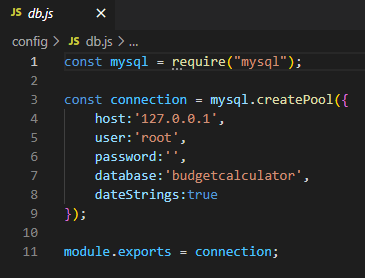
* id: elsődleges azonosító
* tagId: idegen kulcs a tag táblához
* expenseId: idegen kulcs az expenses táblához

## 2.3 Authentikáció és biztonság

**@Peti**

## 2.4 Backend dokumentáció

### 2.4.1 Config mappa



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

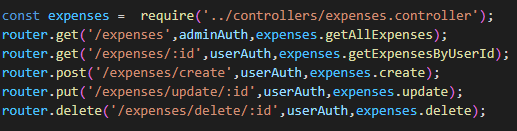
Jogosultság: Admin, Standard

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



#### 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

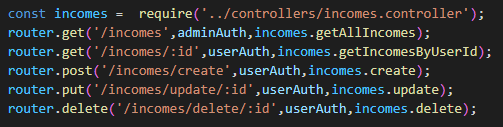
Jogosultság: Admin, Standard

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



#### 2.4.2.3 User controller

Elérési útvonal: /users

**getAllUsers(req, res)**

Elérési útvonal: /

Kérés típusa: HttpGet

Paraméter: request, response

Jogosultság: Admin

Feladata: Kilistázza az összes regisztrált felhasználót

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

message: „Unknown error” 500-as hiba esetén

**getUsersById(req, res)**

Elérési útvonal: /:id

Kérés típusa: HttpGet

Paraméter: request, response

Jogosultság: Admin, Standard

Feladata: Kilistázza 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 occured” 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

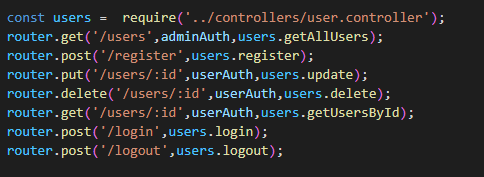
Jogosultság: Admin, Standard

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.



**function RegisterValidate(req, res)**

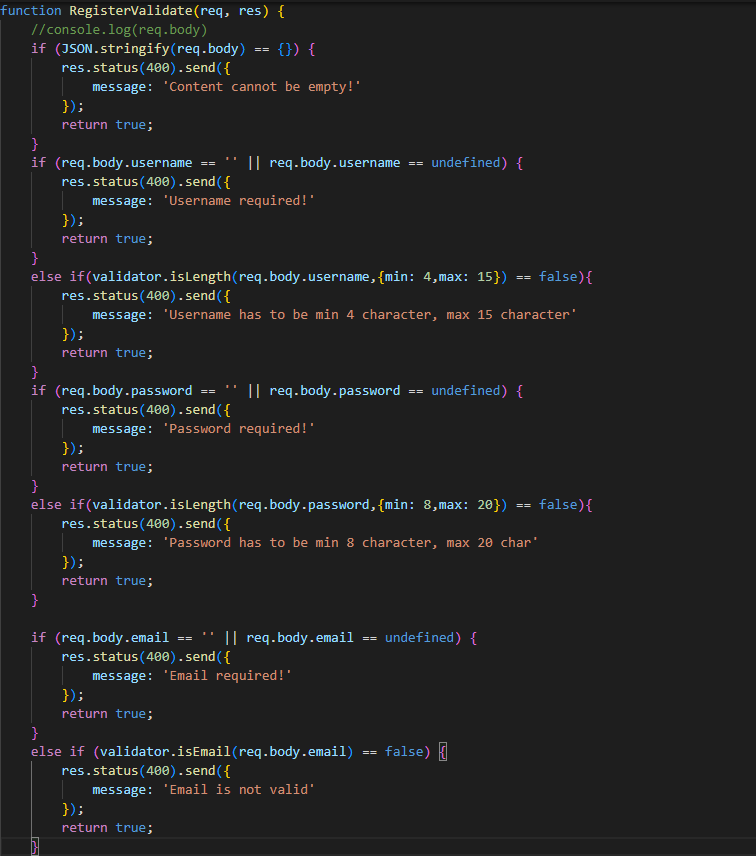
Paraméter: request, response

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.

****

#### 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

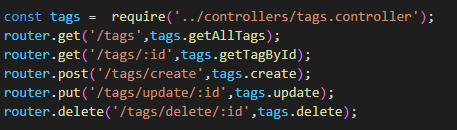
Paraméter: request, response

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



#### 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