**Záródolgozat feladatkiírás**

Tanuló(k) neve: Szép Bence László, Tóth Bence

Képzés: nappali munkarend

Szak: 5 0613 12 03 Szoftverfejlesztő és tesztelő technikus

**A záródolgozat címe:**

**EduTron**

Konzulens: Bólya Gábor

Beadási határidő: 2023. 04. 28.

Győr, 2022. 10. 01

**Módos Gábor**igazgató

**Konzultációs lap**

|  |  |  |  |
| --- | --- | --- | --- |
|  | A konzultáció | | Konzulens aláírása |
| ideje | témája |
| 1. | 2022.10.03. | Témaválasztás és specifikáció |  |
| 2. | 2023.03.14. | Záródolgozat készültségi fokának értékelése |  |
| 3. | 2023.04.26. | 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, 2023. április 26.

Szép Bence László

Tóth Bence

Jedlik Ányos Gépipari és Informatikai

Technikum és Kollégium

**EduTron**

dokumentáció

Készítették:

Szép Bence László

Tóth Bence

Tartalomjegyzék

[1. Bevezetés 5](#_Toc133318874)

[1.1 Az elképzelés és hitelesítés 5](#_Toc133318875)

[1.2 Ötletelés 6](#_Toc133318876)

[1.3 Végeredmény 6](#_Toc133318877)

[1.4 Jövő 7](#_Toc133318878)

[1.5 Csapatmunka 8](#_Toc133318879)

[2. A program 9](#_Toc133318880)

[2.1 Programnyelvek 9](#_Toc133318881)

[2.2 Adatbázis felépítése 10](#_Toc133318882)

[2.2.1 Táblák 11](#_Toc133318883)

[2.3.1 Technológia 14](#_Toc133318884)

[2.3.2 Működése 14](#_Toc133318885)

[2.4.3 Jelszavak 15](#_Toc133318886)

[2.4.4 Jogosultságok 15](#_Toc133318887)

[3 API dokumentáció 16](#_Toc133318888)

[3.1 edutron 16](#_Toc133318889)

[3.2 edutron.user 24](#_Toc133318890)

# 1. Bevezetés

## 1.1 Az elképzelés és hitelesítés

Az EduTron egy olyan tesztkitöltő felületet biztosít felhasználóinknak, amelyben könnyedén lehet teszteket létrehozni, és kitölteni azokat.

Manapság már sok olyan oldalt találhatunk, amelyek megpróbálják ugyanezt a szolgáltatást nyújtani, de mi az EduTron létrehozásával azt a problémát próbáljuk megoldani, hogy könnyedén és egyszerűen lehessen minden funkciót kezelni, amit nyújtani tudunk Önöknek.

Így a programunk megold néhány előforduló problémát, amit a felhasználók tapasztalni szoktak ilyen oldalaknál:

* megkönnyíti a kliensek számára a tesztek létrehozását, megkeresését,
* könnyen kezelhető felületet biztosít,
* a teszt készítőinek a kérdések és válaszok csoportosítását, későbbi felhasználásukat leegyszerűsíti.

Teszteket lehet létrehozni, amelyeknek külön be lehet állítani egy kategóriát. Ezekhez még kérdéseket és válaszokat is lehet csinálni, amiket ugyanúgy lehet csoportosítani. Természetesen különböző bejelentkezési típusokat is nyújtunk (tanár, diák, intézményvezető, normál felhasználó), amiket a profil létrehozásánál be lehet állítani.

Célunk ezzel a programmal az volt, hogy az adott felhasználónak könnyedén lehessen az adott témához teszteket keresni és felhasználni annak érdekében, hogy tudását bővíthesse. A célcsoportunk főleg az iskolák lesznek, hiszen nekik már nagyobb szükségük lehet egy ilyen oldalra, de természetesen bármilyen korosztályú felhasználónak segítséget tudunk nyújtani ezen az oldalon.

## 1.2 Ötletelés

Sokat gondolkoztunk azon, hogy pontosan mit is szeretnénk megvalósítani a projektünkben.

A tanár illetve tesztkészítő oldaláról biztosítani szerettük volna azt, hogy egy könnyen átlátható felületen lehessen teszteket készíteni és közzétenni azokat a megfelelő csoportnak vagy osztálynak. A teszteket külön el lehet menteni egy későbbi alkalomra, illetve a hozzárendelt kérdéseket és válaszokat is el lehet menteni, hogy azokat is fel lehessen használni egy új tesztnél. Végül a teszteket lehet egy kategóriához rendelni a könnyebb megkeresésre való tekintettel. Magát a kategóriát is létre lehet hozni a csoportosítás érdekében.

A tanuló illetve a teszteket kitölteni vágyó felhasználók oldalán a tesztekre való jelentkezés és a könnyű hozzáférhetőség voltak a fő céljaink. Tesztek megtalálására tanuló esetén jelentkezik az adott osztály csoportjába és azon belül minden tesztet megtalál, amit a tanár kirakott, a szabályos felhasználó esetén pedig egy kódot kell beírni azért, hogy a teszt kitöltésére könnyen lehessen feliratkozni. Ezt azért tartjuk fontosnak, mert más oldalakon sok akadályt szokott okozni az, hogy a felhasználónak külön rá kell keresnie az adott tesztre, hiszen könnyedén előfordulhat az a gond, hogy több ilyen teszt is létezik már, ezzel fokozva a felhasználó frusztráltságát. Ilyen funkcióval kevés teendővel gyorsan meg lehet találni az adott tesztet.

## 1.3 Végeredmény

A meglévő terveink mellett egyéb funkciókat is sikerült beépíteni a programba.

Ilyenek például:

* Felhasználók a tesztek létrehozásán felül egyedi kérdéseket is létrehozhatnak, ezzel lehetővé téve a kérdések testreszabhatóságát a tesztek között
* Felhasználónak lehetősége nyílik megtekinteni a már kitöltött tesztjei eredményét, valamint egyéb tesztekkel kapcsolatos statisztikáit

A megvalósított tervek mellett viszont látunk fejlesztési lehetőségeket is a jövőre nézve, mint például egy mobilalkalmazás elkészítése.

## 1.4 Jövő

A programunk tervezésekor számos ötlettel álltunk elő. Az végleges verzióban viszont mérlegelnünk kellett, hogy mit sikerül beleépíteni a programba és mit nem. Ezeket, a jelenlegi verzióból kimaradt ötleteket a jövőben azonban mindenképp bele szeretnénk építeni. Úgy gondoljuk, hogy ezeket fontos megemlíteni, hiszen ezek megmutatják azt az irányt, amerre a projektünkkel el szeretnénk jutni.

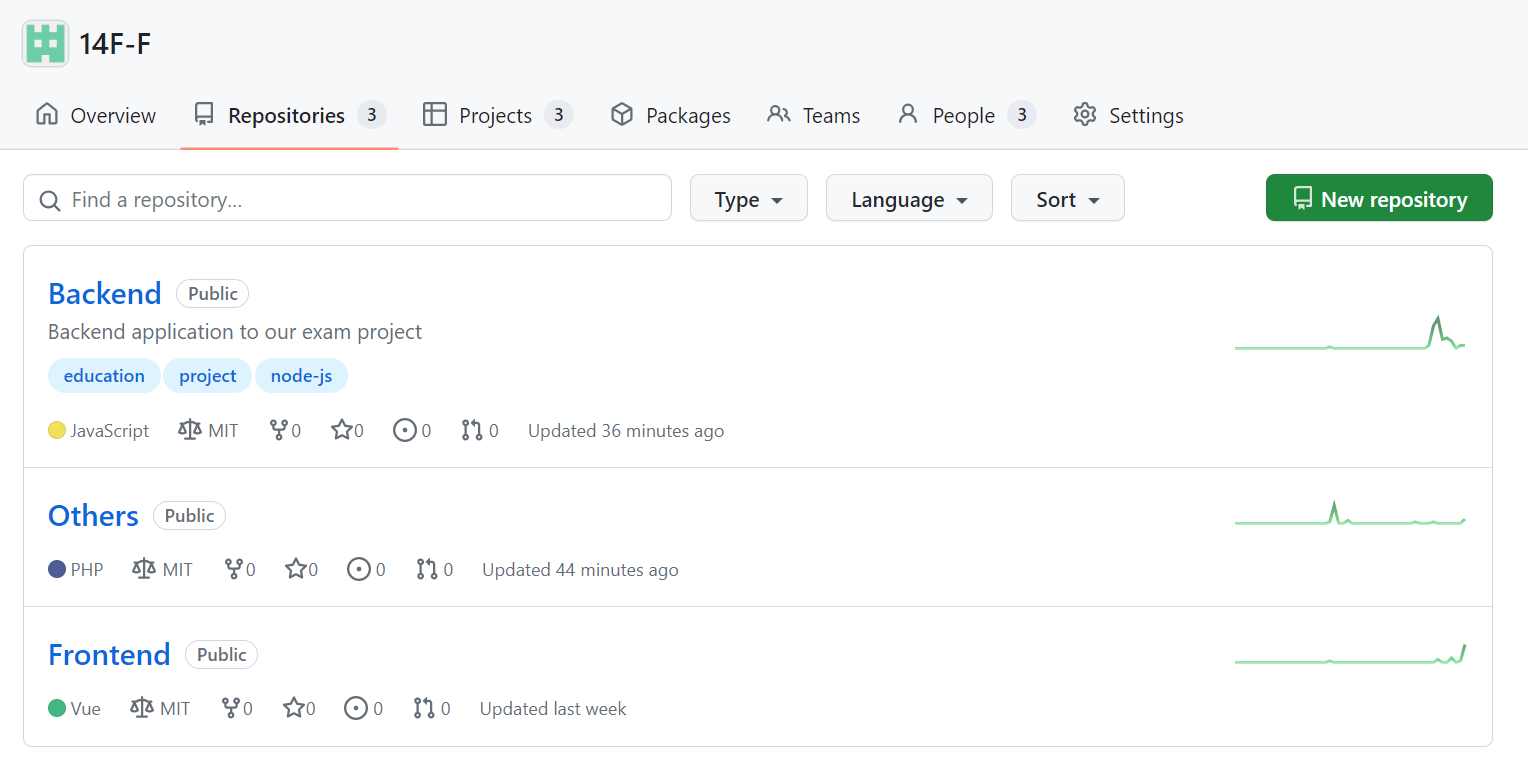
Az egyik legfontosabb ötletünk egy mobilalkalmazás fejlesztése, amely lehetővé teszi diákok számára, hogy teszteket töltsenek ki mobiltelefonjukon keresztül bárhol, ahol van internetkapcsolat. Mivel a weboldalunk már teljesen reszponzív, ezért úgy gondoljuk, hogy az alkalmazás létrehozása lehetővé teszi, hogy az összes funkció jobban elérhető legyen, és bárki könnyedén hozzáférhetne a tesztekhez.

Fontosnak tartom megemlíteni az iskolai rendszerekkel való összekötést. Szerintünk nagyon hasznos lehetne, ha az adott iskolák beleegyezésével felhasználhatnánk adatbázisukat és összekötnénk őket a tesztekkel. Így például az adott tanulónak elég lenne egy iskolát megadnia, és így a kiválasztott osztályon belül hozzáférhet olyan tesztekhez, amiket annak az osztálynak raktak ki, és kitöltheti őket bármikor. Ezt nagyon nehéz lett volna megvalósítani az adott időintervallumban, de úgy hisszük, ez egy hasznos funkció lenne akkor, ha a programunkat sokan fogják használni.

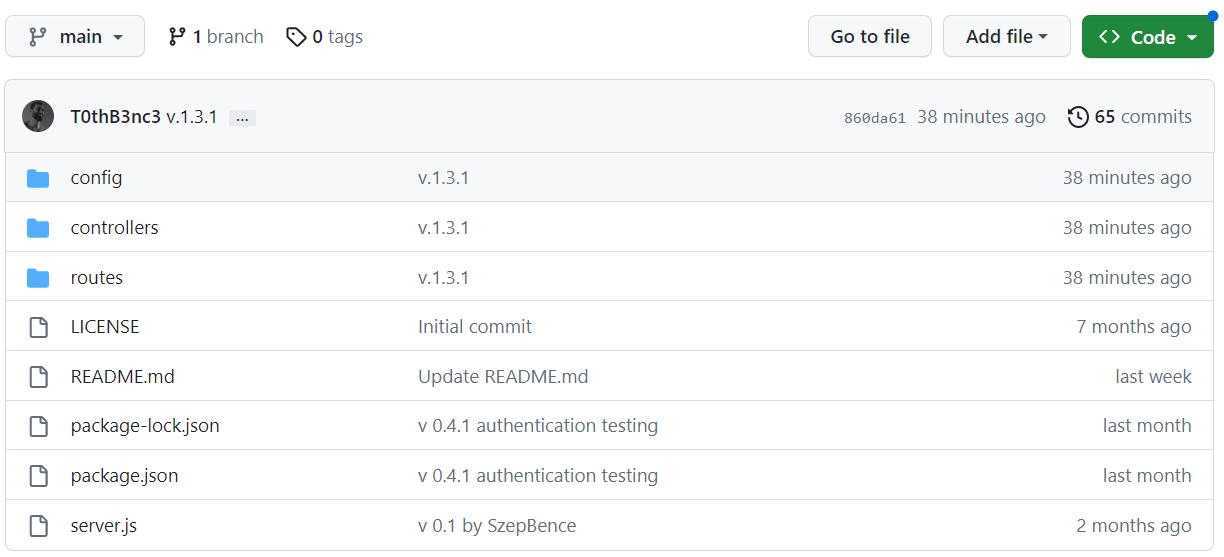
Gondoltunk még egy esetleges kvíz csatát is beépíteni egy többjátékos rendszerrel. Ezt szinte lehetetlen lett volna megvalósítani, mert ilyen rendszert egyáltalán nem ismer egyikünk sem. Ez egy olyan játékos megoldás lehetne, ami szerintünk tetszene a diákoknak és más felhasználóknak is, akik esetleg egy kihívást szeretne magának. Annyiból állna, hogy egy tesztkitöltő csinál egy szerverszobát, amibe be lehetne kapcsolódni. Ezeket lehet állítani publikusra, hogy bárkivel össze tudja mérni tudását, illetve privátra, ha barátaival vagy ismerőseivel szeretné ezt megtenni külső beavatkozás nélkül. A játék közben a program figyelne a választ megadása közben eltelt időre, és hogy helyes-e a kiválasztott válasz. Pontozási rendszer lenne, így a rendszer könnyen kiválasztja a győztest.

## 1.5 Csapatmunka

A programozás megkezdése előtt fontos volt, hogy pontosan osszuk fel a feladatokat. Mivel mi az egyik olyan csapat vagyunk, ami két főből áll, úgy gondoltuk az lenne a legegyszerűbb, ha felosztjuk a programot a backend részére és a frontend részére. Természetesen mindketten foglalkoztunk a backend résszel és a frontend résszel, folyamatosan kérdeztük egymást, ha elakadtunk. Ennek megvalósítása érdekében a GitHub felületet használtuk, ahol nyomon követhető minden feltöltés.



Itt az látható, hogy a feladatokat 3 részre osztottuk fel. A Backend és Frontend repositoryk-ban a hozzájuk tartozó fájlok tartoznak, az Others repositoryban pedig minden olyan egyéb fájl, amit felhasználtunk leadásaink során.



Ezen a képen a Backend repository látható.

*(kép a Frontend reporol)*

Itt pedig a Frontend repository látható.

# 2. A program

## 2.1 Programnyelvek

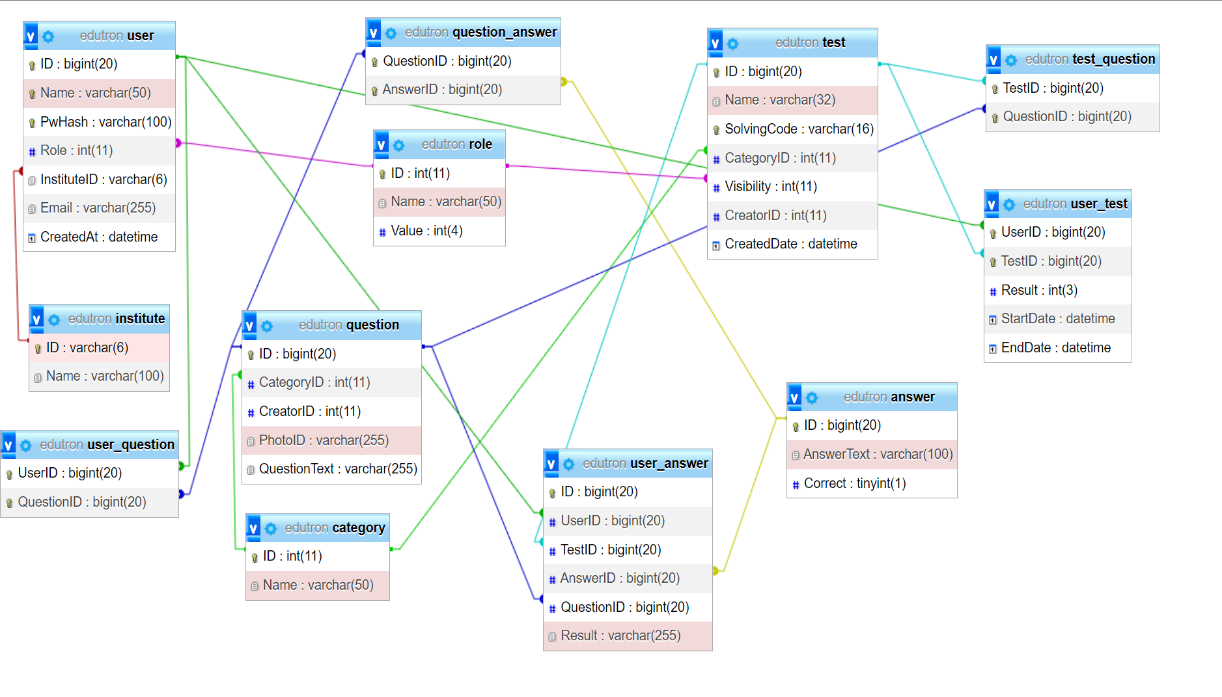
A projekt backendje Node.js keretrendszerben történt. *NodeJs egy olyan futtató környezet, ami lehetőséget nyújt JavaScriptben írt programook futtatására szervereken. JavaScriptet leginkább a kliens oldalon szokták használni, a böngészőn keresztül. Ahhoz, hogy megértsük miképp futatthatunk JavaScript kódot szerveren, először nézzük meg, miképp valósul meg ugyanez a böngészőn keresztül. A böngésző applikáció fejlesztők számára olyan motorokat kell biztosítani (a Google Chrome esetén a v8 engine), amik a JavaScript kódot gépi nyelvként interpretálják és le is futtatják. Alapvetően a NodeJs is ugyanazt a v8 engine-t használja, némi módosítással. Most felmerülhet bennünk az a kérdés, hogy miben jobb, mint a Ruby, PHP, asp.net. Nyilvánvalóan ez csak az adott szituációtól függ, ezért egyes esetekben hátrány lehet NodeJs-t használni. Annyi könnyebséget nyújt, ha már dolgoztunk frontend oldalon, nem kell egy teljesen új nyelvet megtanulni ahhoz, hogy a backend-et is implementáljuk.* (Szegedi Egyetem, 2019)



A frontend fejlesztéshez pedig Vue.js keretrendszert használtunk. *A Vue egy JavaScript-keretrendszer felhasználói felületek építésére. A szabványos HTML-re, CSS-re és JavaScript-re épül, és deklaratív és komponens-alapú programozási modellt biztosít, amely segít hatékonyan fejleszteni a felhasználói felületeket, legyenek azok egyszerűek vagy összetettek.* (vuejs.org, angolról fordítva)



## 2.2 Adatbázis felépítése



## 2.2.1 Táblák

**Test** – Tesztek

* *ID:* elsődleges azonosító
* *Name:* teszt neve
* *SolvingCode:* teszt kódja
* *CategoryID:* idegenkulcs a Category táblával, a felhasználó által választott kategória kerül mentésre
* *Visibility:* láthatóság, kinek milyen jogosultsága van látni a tesztet
* *CreatorID:* idegenkulcs a User táblával, a felhasználót köti össze vele
* *CreatedDate:* létrehozás dátuma

**Answer** – Válaszok

* *ID:* elsődleges azonosító
* *AnswerText:* a válasz szövege
* *Correct:* helyes-e a válasz vagy sem

**Question** – Kérdések

* *ID:* elsődleges azonosító
* *CategoryID:* idegenkulcs a Category táblával, a felhasználó által választott kategória kerül mentésre
* *CreatorID:* idegenkulcs a User táblával, a felhasználót köti össze vele
* *PhotoID:* esetleges fénykép azonosítója
* *QuestionText:* a kérdés szövege

**User** – Felhasználó

* *ID:* elsődleges azonosító
* *Name:* felhasználó teljes neve
* *PwHash:* backenden titkosításra került a felhasználó által megadott jelszó
* *Role:* felhasználó szerepköre
* *InstituteID:* esetleges intézmény azonosítója
* *Email:* felhasználó e-mail címe
* *CreatedAt:* regisztráció dátuma

**Category** – Kategória

* *ID:* elsődleges azonosító
* *Name:* kategória neve

**Institute** – Intézmény

* *ID:* elsődleges azonosító
* *Name:* intézmény neve

**Role** – Szerepkör

* *ID:* elsődleges azonosító
* *Name:* szerepkör neve
* *Value:* szerepkör azonosítója

**Question\_Answer** – Kérdéseket válasszal összekötő tábla

* *QuestionID:* idegenkulcs, kérdés azonosítója
* *AnswerID:* idegenkulcs, válasz azonosítója

**Test\_Question** – Tesztekhez kérdésekkel összekötő tábla

* *TestID:* idegenkulcs, teszt azonosítója
* *QuestionID:* idegenkulcs, kérdés azonosítója

**User\_Answer** – Felhasználó válaszait összekötő tábla

* *ID:* elsődleges azonosító
* *UserID:* idegenkulcs, felhasználó azonosítója
* *TestID:* idegenkulcs, teszt azonosítója
* *AnswerID:* idegenkulcs, válasz azonosítója
* *QuestionID:* idegenkulcs, kérdés azonosítója
* *Result:* a felhasználó eredménye

**User\_Question** – Felhasználó kérdéseit összekötő tábla

* *UserID:* idegenkulcs, felhasználó azonosítója
* *QuestionID:* idegenkulcs, kérdés azonosítója

**User\_Test** – Felhasználó tesztjeit összekötő tábla

* *UserID:* idegenkulcs, felhasználó azonosítója
* *TestId:* idegenkulcs, teszt azonosítója
* *Result:* a felhasználó eredménye
* *StartDate:* kezdés ideje
* *EndDate:* befejezés ideje

## 2.3.1 Technológia

A weboldalon történő bejelentkezéshez a JWT (JSON Web Token) szabványt használjuk, melyet a backend User rétegében található osztályban definiálunk. Ebben az osztályban határozzuk meg a tokenben lévő adatokat, mint például a lejárati idejét vagy a felhasználó jogait. *A JWT egy olyan szabvány, amely kompakt és önálló módszert határoz meg az információk biztonságos továbbításához az ügyfél és a szerver között, mint JSON objektumot. A kompakt méret megkönnyíti a tokenek átadását URL-en, POST paraméteren vagy egy http fejlécen keresztül. Továbbá, mivel önállóak, tartalmazzák a felhasználóval kapcsolatos összes szükséges információt, így az adatbázist nem kell többször lekérdezni.* (Ilusionity, 2022)

## 2.3.2 Működése

**Backend oldalon:**

A token létrehozása a genToken funkció hozza létre amit a createUser(register) függvényen hívunk meg. A loggedIn függvény meghívásakor ellenőrzés után e-mail és a felhasználó jogosultsága alapján hozza létre a hitelesítéshez szükséges tokent.

**Frontend oldalon:**

készülőben van

## 2.4.3 Jelszavak

Sikeres regisztráció után a backendre érkező jelszó titkosítására kerül sor, melyet ebben a formában tárolunk el az adatbázisban. A lekérések során ez a titkosított jelszó semmiféle formában nem kerül visszaküldésre a frontend irányába az esetleges támadások végett. A felhasználó által megadott titkosítatlan jelszó nem kerül tárolásra az adatbázisban.

## 2.4.4 Jogosultságok

Háromféle jogosultság alapján vannak a felhasználók besorolva: *Guest, User, Admin*. Regisztrációkor választható, hogy edzőként vagy kliensként szeretné-e használni fiókját, admin joggal csak a rendszergazda rendelkezhet.

**Guest** típusú felhasználó:

A felhasználó egy generált azonosítóval rendelkezik, adatai, kitöltött tesztjeinek eredményei nem kerülnek tárolásra.

**User** típusú felhasználó:

A felhasználó regisztrációt követően böngészhet a tesztek között, jelentkezhet rájuk, illetve megtekintheti jelentkezéseit. Ezek mellett készíthet saját tesztet is.

**Admin** típusú felhasználó:

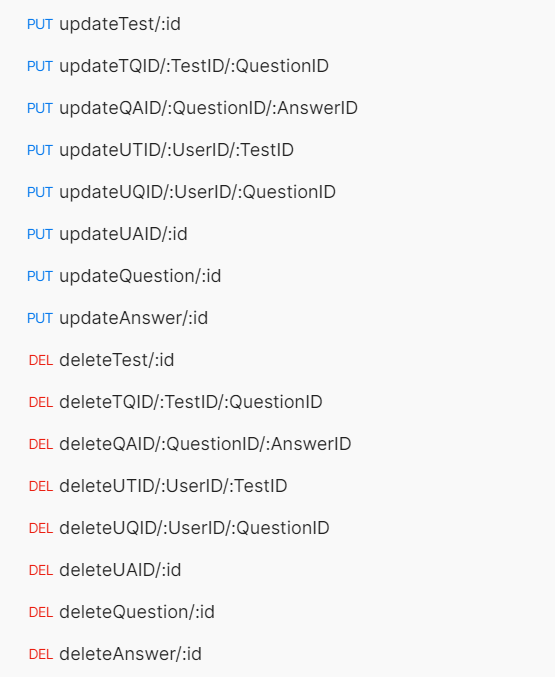
Az Admin típusú felhasználó felvehet újabb kategóriákat, illetve tag-eket egy külön erre célra létrehozott felületen.

Ezen típusú felhasználók mellett más, egyéb típusú felhasználók rögzítésére is lehetőség nyílik, amiket az Admin típusú felhasználók rögzítenek az adatbázisban.

# 3 API dokumentáció

## 3.1 edutron

**getAllfromTable**

* Kérés típusa: Get
* Paraméter: tableName
* Jogosultság: Admin, User
* Feladat: kilistáz minden adatot az adott táblán belül
* ****Válasz: összes adat JSON formátumban

**testByUser**

* Kérés típusa: Get
* Paraméter: userId
* Jogosultság: Admin, User
* Feladat: tesztek felhasználók szerint csoportosítva
* Válasz: adott felhasználók tesztjei JSON formátumban

**testByCategory**

* Kérés típusa: Get
* Paraméter: categoryId
* Jogosultság: Admin, User
* Feladat: tesztek kategóriák szerint csoportosítva
* Válasz: adott kategória tesztjei JSON formátumban

**getTestByID**

* Kérés típusa: Get
* Paraméter: id
* Jogosultság: Admin, User
* Feladat: minden tesztet visszaad a megadott id alapján
* Válasz: teszt adatai JSON formátumban

**getQuestionByTestID**

* Kérés típusa: Get
* Paraméter: testId
* Jogosultság: Admin, User
* Feladat: minden kérdést visszaad teszt ID-je alapján
* Válasz: adott teszt kérdései JSON formátumban

**getAnswerByQuestionID**

* Kérés típusa: Get
* Paraméter: questionId
* Jogosultság: Admin, User
* Feladat: minden választ visszaad kérdés ID-je alapján
* Válasz: adott kérdés válaszai JSON formátumban

**getCategoryByID**

* Kérés típusa: Get
* Paraméter: nincs
* Jogosultság: Admin, User
* Feladat: minden kategórát visszaad a megadott id alapján
* Válasz: minden tesztet visszaad a megadott id alapján

**createTest**

* Kérés típusa: Post
* Paraméter: nincs
* Jogosultság: Admin, User
* Feladat: egy teljesen új tesztet hoz létre a megadott adatokkal
* Válasz: létrejött tesztet visszaadja

**createQuestion**

* Kérés típusa: Post
* Paraméter: nincs
* Jogosultság: Admin, User
* Feladat: egy teljesen új kérdést hoz létre a megadott adatokkal
* Válasz: létrejött kérdést visszaadja

**createAnswer**

* Kérés típusa: Post
* Paraméter: nincs
* Jogosultság: Admin, User
* Feladat: egy teljesen új választ hoz létre a megadott adatokkal
* Válasz: létrejött választ visszaadja

**addTQID**

* Kérés típusa: Post
* Paraméter: nincs
* Jogosultság: Admin, User
* Feladat: test\_question kapcsolótáblába rakja az adott ID-t
* Válasz: létrejött ID-t visszaadja

**addQAID**

* Kérés típusa: Post
* Paraméter: nincs
* Jogosultság: Admin, User
* Feladat: question\_answer kapcsolótáblába rakja az adott ID-t
* Válasz: létrejött ID-t visszaadja

**addUTID**

* Kérés típusa: Post
* Paraméter: nincs
* Jogosultság: Admin, User
* Feladat: user\_test kapcsolótáblába rakja az adott ID-t
* Válasz: létrejött ID-t visszaadja

**addUQID**

* Kérés típusa: Post
* Paraméter: nincs
* Jogosultság: Admin, User
* Feladat: user\_question kapcsolótáblába rakja az adott ID-t
* Válasz: létrejött ID-t visszaadja

**addUAID**

* Kérés típusa: Post
* Paraméter: nincs
* Jogosultság: Admin, User
* Feladat: user\_answer kapcsolótáblába rakja az adott ID-t
* Válasz: létrejött ID-t visszaadja

**updateTest**

* Kérés típusa: Put
* Paraméter: id
* Jogosultság: Admin, User
* Feladat: megadott id alapján megkeresi a tesztet és megváltoztatja az adatait
* Válasz: megváltozott értékek JSON formátumban

**updateQuestion**

* Kérés típusa: Put
* Paraméter: id
* Jogosultság: Admin, User
* Feladat: megadott id alapján megkeresi a kérdést és megváltoztatja a szövegét
* Válasz: megváltozott érték JSON formátumban

**updateAnswer**

* Kérés típusa: Put
* Paraméter: id
* Jogosultság: Admin, User
* Feladat: megadott id alapján megkeresi a választ és megváltoztatja a szövegét
* Válasz: megváltozott érték JSON formátumban

**updateTQID**

* Kérés típusa: Put
* Paraméter: TestID/QuestionID
* Jogosultság: Admin, User
* Feladat: frissíti a test\_question táblában lévő ID-t
* Válasz: megváltozott érték JSON formátumban

**updateQAID**

* Kérés típusa: Put
* Paraméter: QuestionID/AnswerID
* Jogosultság: Admin, User
* Feladat: frissíti a quesiton\_answer táblában lévő ID-t
* Válasz: megváltozott érték JSON formátumban

**updateUTID**

* Kérés típusa: Put
* Paraméter: UserID/TestID
* Jogosultság: Admin, User
* Feladat: frissíti a user\_test táblában lévő ID-t
* Válasz: megváltozott érték JSON formátumban

**updateUQID**

* Kérés típusa: Put
* Paraméter: UserID/QuestionID
* Jogosultság: Admin, User
* Feladat: frissíti a user\_question táblában lévő ID-t
* Válasz: megváltozott érték JSON formátumban

**updateUAID**

* Kérés típusa: Put
* Paraméter: id
* Jogosultság: Admin, User
* Feladat: frissíti a user\_answer táblában lévő ID-t
* Válasz: megváltozott érték JSON formátumban

**deleteTest**

* Kérés típusa: Delete
* Paraméter: id
* Jogosultság: Admin, User
* Feladat: megadott id alapján kitörli az adott tesztet
* Válasz: visszaigazoló üzenet arról, hogy törlődött a teszt

**deleteQuestion**

* Kérés típusa: Delete
* Paraméter: id
* Jogosultság: Admin, User
* Feladat: megadott id alapján kitörli az adott kérdést
* Válasz: visszaigazoló üzenet arról, hogy törlődött a kérdés

**deleteAnswer**

* Kérés típusa: Delete
* Paraméter: id
* Jogosultság: Admin, User
* Feladat: megadott id alapján kitörli az adott választ
* Válasz: visszaigazoló üzenet arról, hogy törlődött a válasz

**deleteTQID**

* Kérés típusa: Delete
* Paraméter: TestID/QuestionID
* Jogosultság: Admin, User
* Feladat: kitörli a test\_question táblából az ID-t
* Válasz: visszaigazoló üzenet arról, hogy törlődött az ID

**deleteQAID**

* Kérés típusa: Delete
* Paraméter: QuestionID/AnswerID
* Jogosultság: Admin, User
* Feladat: kitörli a question\_answer táblából az ID-t
* Válasz: visszaigazoló üzenet arról, hogy törlődött az ID

**deleteUTID**

* Kérés típusa: Delete
* Paraméter: UserID/TestID
* Jogosultság: Admin, User
* Feladat: kitörli a user\_test táblából az ID-t
* Válasz: visszaigazoló üzenet arról, hogy törlődött az ID

**deleteUQID**

* Kérés típusa: Delete
* Paraméter: UserD/QuestionID
* Jogosultság: Admin, User
* Feladat: kitörli a user\_question táblából az ID-t
* Válasz: visszaigazoló üzenet arról, hogy törlődött az ID

**deleteUAID**

* Kérés típusa: Delete
* Paraméter: id
* Jogosultság: Admin, User
* Feladat: kitörli a user\_answer táblából az ID-t
* Válasz: visszaigazoló üzenet arról, hogy törlődött az ID

## 3.2 edutron.user

**register**

* Kérés típusa: Post
* Paraméter: nincs
* Jogosultság: Admin, User
* Feladat: felhasználó létrehozása
* Válasz: üzenet a sikeres vagy sikertelen regisztrációról, visszaadva adatokkal

**updateUser**

* Kérés típusa: Put
* Paraméter: id
* Jogosultság: Admin, User
* Feladat: egy felhasználó adatait változtatja meg
* Válasz: megváltozott adatokat mutatja ki

**deleteUser**

* Kérés típusa: Delete
* Paraméter: id
* Jogosultság: Admin, User
* Feladat: kitörli az adott felhasználót
* Válasz: visszaigazoló üzenet arról, hogy törlődött a felhasználó

**loggedIn**

* Kérés típusa: Post
* Paraméter: nincs
* Jogosultság: Admin, User
* Feladat: visszaigazolja, hogy a felhasználó létezik be van jelentkezve
* Válasz: bejelentkezett felhasználó adatait visszaadja