| BKSZC Pogány Frigyes Technikum | |
| --- | --- |
| Karbantartói checklist  Szoftverfejlesztő és -tesztelő  vizsgaremek  2025 május | |
| Készítette:  Szőnyi Bence  Rozmán Balázs | Konzulens:  - |

# Feladat Leírása és Bevezetés

Ez a weboldal egy kanban alapú feladatkezelő rendszer, amely lehetővé teszi a felhasználók számára, hogy saját feladataikat nyomon kövessék, rendszerezzék és menedzseljék. Az oldal teljes mértékben a felhasználók interakcióira épül – nélkülük gyakorlatilag nem létezik –, hiszen a célja az, hogy egy rugalmas és hatékony eszközt biztosítson a munkafolyamatok követésére.

Az alkalmazás egyik legfontosabb funkciója, hogy több felhasználóval is meg lehet osztani a táblákat, így egy csapat közösen dolgozhat ugyanazon projekten. Ezáltal a rendszer nemcsak egyéni feladatkezelésre, hanem csapatmunkára is kiválóan alkalmas.

## A Projekt Eredete

Az ötlet eredetileg Tóth Józseftől származik, aki felvetette, hogy az iskolánk portáján vagy más helyszíneken ne csak egy füzetbe írják fel, hogy melyik teremben milyen eszközök vagy berendezések hibásodtak meg. Ehelyett egy digitális megoldás segítene a hibák követésében és gyorsabb megoldásában.

Ennek alapján született meg a gondolat, hogy egy olyan kanban rendszerű webalkalmazást hozzunk létre, amely bárhol alkalmazható, ahol fontos a feladatok átláthatósága és hatékony kezelése az érdek.

# Felhasználói Dokumentáció

## Bejelentkezés és Regisztráció

Amikor az oldal betöltődik, **először a bejelentkezési (Login) oldalra érkezünk**. Ez az oldal kiindulópontként szolgál, és amíg nem jelentkezünk be, **az oldal többi része elérhetetlen marad**.

A felhasználók két lehetőség közül választhatnak:

* **Bejelentkezés (Login)** egy meglévő fiókkal
* **Regisztráció (Register)** egy új fiók létrehozásához

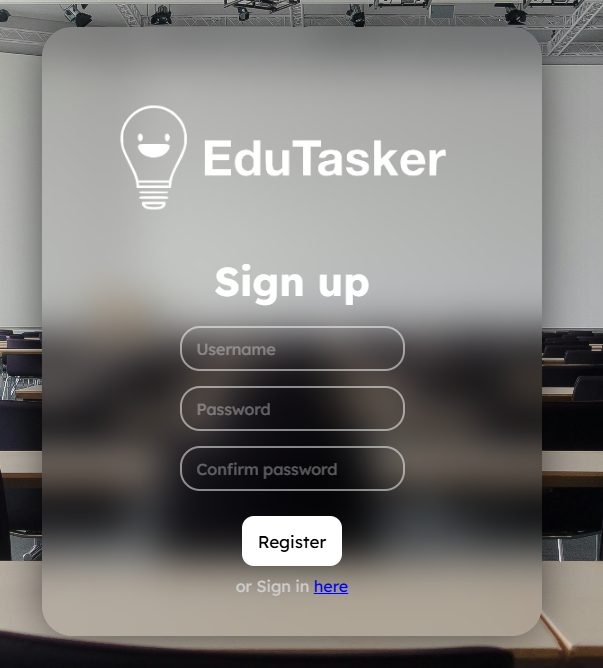
### Login és Register gombok

* A **Login gomb** lehetővé teszi a meglévő fiókba való belépést.
* A **Register gomb** átirányít a regisztrációs oldalra, ahol egy új fiók hozható létre.

**Regisztráció menete:** A regisztrációs oldalon a felhasználó megadja az alapvető adatokat (pl. felhasználónév, jelszó), majd a rendszer elmenti az adatokat az adatbázisba. A sikeres regisztráció után visszatérhetünk a **Login** oldalra, és bejelentkezhetünk az új fiókunkkal.

**Bejelentkezés után:**

* A **jobb felső sarokban megjelenik az üdvözlés** (a felhasználó nevével) és egy **Logout gomb**, amely lehetőséget biztosít a kijelentkezésre.
* Átirányítást kapunk a **Dashboardra**, amely a fő kezelőfelületként szolgál.



## Dashboard – Kezelőfelület

A bejelentkezést követően a **Dashboard** oldal fogad minket. Ez egy **áttekintő oldal**, ahol a felhasználók az elérhető tábláikat kezelhetik.

### Elérhető funkciók a Dashboardon:

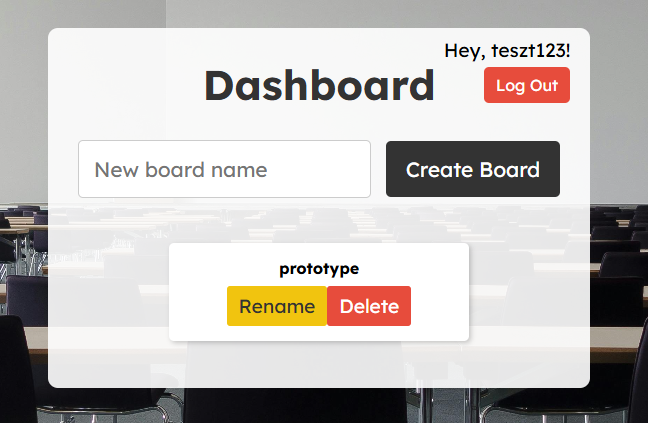
* **Log out gomb** – Kijelentkezés a fiókból
* **Create Board gomb** – Új tábla létrehozása
* **Táblák listázása** – Az összes létrehozott tábla megjelenítése

**Amíg nincs létrehozott tábla, addig nem jelenik meg semmilyen tábla a listában.** Ha létrehozunk egy táblát, akkor az azonnal megjelenik a Dashboardon.

### Táblakezelési lehetőségek:

Minden tábla mellett három gomb található:

1. **Megnyitás (Open Board)** – A tábla részletes nézetének megnyitása
2. **Átnevezés (Rename Board)** – A tábla nevének módosítása
3. **Törlés (Delete Board)** – A tábla végleges törlése az adatbázisból



## 

## Kanban Tábla Felület

Ha rákattintunk egy táblára, belépünk a **Kanban tábla nézetbe**, amely három oszlopra van felosztva a feladatok állapotának megfelelően:

* **To-Do** (Tennivalók)
* **In Progress** (Folyamatban lévő feladatok)
* **Complete** (Befejezett feladatok)

### Feladatkezelési lehetőségek a táblán belül:

* **Új feladat hozzáadása** – A megfelelő oszlopban létrehozhatunk új feladatokat.
* **Feladat áthelyezése** – A feladatokat egyik oszlopból a másikba húzással vagy gombok segítségével áthelyezhetjük.
* **Feladat szerkesztése** – Egy meglévő feladat címét vagy leírását módosíthatjuk.
* **Feladat törlése** – Egy adott feladat végleges törlése.

## Táblák Megosztása Más Felhasználókkal

A tábla alsó részén található a **megosztási funkció**, amely lehetővé teszi más felhasználók számára a tábla közös szerkesztését.

### Megosztás lépései:

1. A felületen megjelenik az összes elérhető felhasználó listája.
2. Minden felhasználó mellett található egy **"Megosztás" (Share) gomb**.
3. Ha megnyomjuk a gombot, az adott felhasználó hozzáférést kap a táblához.
4. **Ha a megosztás már megtörtént, a gomb inaktívvá válik.**

### Megosztás visszavonása:

* Ha egy táblát már megosztottunk egy felhasználóval, megjelenik egy **"Megvonás" (Revoke) gomb**, amellyel visszavonhatjuk az illető hozzáférését.

## Összegzés

* Az oldal **Login oldallal** indul, ahol **bejelentkezhetünk vagy regisztrálhatunk**.
* A **Dashboardon** hozhatunk létre és kezelhetünk táblákat.
* A táblán belül **feladatokat szerkeszthetünk, törölhetünk és átrendezhetünk**.
* A táblákat **megoszthatjuk másokkal**, így közösen dolgozhatunk rajtuk.

# Fejlesztői dokumentáció

# Technológiai Stack

A projekt fejlesztése során a következő **modern webtechnológiákat** és eszközöket használtuk:

### Frontend (Felhasználói felület):

* **HTML5** – Az oldal szerkezetének kialakítására.
* **CSS3** – A vizuális megjelenés és stílus biztosítására.
* **JavaScript** – Dinamikus funkciók és interakciók megvalósítására.
* **React** – A komponens alapú fejlesztés és az interaktív felhasználói élmény érdekében.

### Backend (Szerveroldal és adatkezelés):

* **Node.js** – A szerveroldali logika és API-k kiszolgálására.
* **Express.js** – A backend hatékony kezelésére és a kliens-szerver kommunikáció biztosítására.

### Adatbázis:

* **MySQL** – Az adatok tárolásához és kezeléséhez.
* **KanbanDB** – Az alkalmazás saját adatbázisa, amelyet **a letölthető mappák között megtalálsz**. **Importálás után az adatbázis automatikusan működni fog**.

# 

# Funkcionalitás

Az oldal célja **egy egyszerű és hatékony, profilhoz kötött feladatkövető rendszer biztosítása**. A legfontosabb funkciók közé tartoznak:

**Felhasználói profilok** – Minden felhasználó saját fiókkal rendelkezik.  
 **Kanban tábla használata** – A feladatok három oszlopba (To-Do, In Progress, Complete) rendezhetők.  
 **Feladatok hozzáadása, szerkesztése és törlése** – A felhasználók kezelhetik saját feladataikat.  
 **Csapatmunka támogatása** – A táblák megoszthatók más felhasználókkal.  
 **Felhasználóbarát kezelőfelület** – Egyszerű és intuitív design.

# Telepítési és Indítási Lépések

----------------------------------------------------------------------------------------------------------------------------------------------------

Github repository:[Borsocska/](https://github.com/Borsocska/Megbukas-muveszete)EduTasker

## Előfeltételek

A projekt megfelelő működéséhez néhány előkészülettel kell kezdeni. Az alábbi szoftverek és eszközök szükségesek a rendszer telepítéséhez és futtatásához.

### 1. Node.js és npm

A projekt futtatásához telepítened kell a [Node.js](https://nodejs.org/) keretrendszert. Az **ajánlott verzió legalább 14**, de lehetőleg **16 vagy magasabb** verziót használj.

Ellenőrizheted a telepített verziót az alábbi parancsokkal:

node -v

npm -v

Ha még nincs telepítve, letöltheted a hivatalos weboldalról, majd telepítheted a Node Package Managert (**npm**) is. Az első lépésként futtasd:

npm i

Ez letölti és telepíti a projekt futtatásához szükséges csomagokat.

### 2. MySQL adatbázis

A projekt az **adatok tárolásához MySQL adatbázist használ**, így elengedhetetlen, hogy a MySQL szerver futó állapotban legyen.

Ha még nincs telepítve, letöltheted innen: [MySQL Letöltés](https://dev.mysql.com/downloads/).

Győződj meg róla, hogy az adatbázis szerver fut, és a projektben megadott konfigurációnak megfelelő adatbázis elérhető.

### 3. Kódszerkesztő

A fejlesztéshez és szerkesztéshez **Visual Studio Code (VS Code)** vagy egy hasonló modern szerkesztő javasolt. Letöltés: [VS Code](https://code.visualstudio.com/)

## Indítási Lépések

### 1. Adatbázis beállítása

Nyisd meg a MySQL adatbázis kezelődet (pl. MySQL Workbench vagy phpMyAdmin), majd importáld a projekt db almappájában található adatbázist.

### 2. Frontend indítása

A frontend a felhasználói felület megjelenítéséért felel. A futtatásához:

1. Nyiss egy terminált vagy parancssort, majd lépj be a **frontend** mappába:  
   cd frontend
2. Indítsd el a fejlesztői szervert:  
   npm run dev

Ezután az oldal elérhető lesz a megadott localhost címen (pl. http://localhost:3000).

### 3. Backend indítása

A backend felel az adatkezelésért és a szerveroldali műveletekért.

1. Nyiss egy új terminált, majd lépj be a **backend** mappába:  
   cd backend
2. Telepítsd a szükséges csomagokat:  
   npm i
3. Indítsd el a szervert:  
   npm start

A backend ekkor elérhető lesz a megadott szervercímen (pl. http://localhost:3000 vagy http://localhost:5000, attól függően, hogy milyen beállításokat használsz).

# **Pontos specifikáció**

## 1. Áttekintés

A rendszer célja egy felhasználóbarát Kanban táblás feladatkezelő alkalmazás létrehozása, amely lehetővé teszi a feladatok vizuális kezelését és nyomon követését. Minden felhasználó rendelkezik saját táblával, azonban támogatja a közös, csapatmunkára kialakított táblákat is. A feladatok mozgatása az oszlopok között egy intuitív drag-and-drop felület segítségével történik.

## 2. Funkcionális követelmények

### 2.1 Felhasználói kezelés

* **Regisztráció és Bejelentkezés:** A felhasználók regisztrálhatnak, majd bejelentkezhetnek a rendszerbe. A hitelesítés során a rendszer biztosítja a jogszerű hozzáférést a megfelelő felhasználói táblákhoz.

### 2.2 Kanban táblák kezelése

* **Egyéni táblák:** Minden felhasználó rendelkezik legalább egy személyes Kanban táblával, ahol saját feladatait kezelheti.
* **Közös táblák:** Lehetőség van olyan projektek létrehozására, ahol több felhasználó együtt dolgozhat, megosztva ugyanazt a Kanban táblát.

### 2.3 Feladatok kezelése

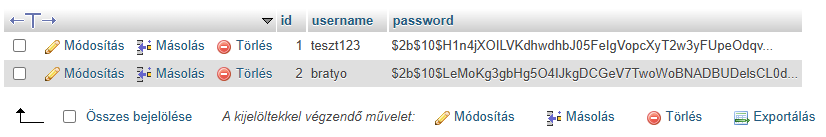
* **Feladat létrehozása:** Új feladat létrehozása, amelyet nyomon lehet követni
* **Feladat módosítás és törlés:** A felhasználó módosíthatja vagy törölheti a már létrehozott feladatokat.
* **Drag-and-drop felület:** A feladatkártyák átmozgatása az oszlopok között, az aktuális állapot (pl. Teendő → Folyamatban → Elvégzett) vizuális változásával.

# Adattáblák Definíciói

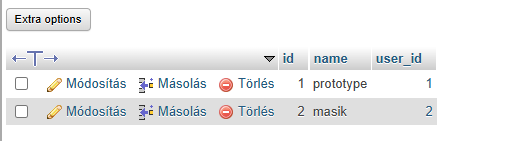
### *JELEN ESETBEN VANNAK BENNÜK TESZTADATOK, EZEKRŐL KÉSŐBB*

A. users tábla

* **Cél:** A rendszerben regisztrált felhasználók tárolása.
* **Struktúra:**
  + id: Egész típusú, automatikusan növekvő elsődleges kulcs.
  + username: Maximum 100 karakter hosszú szöveg, kötelező érték.
  + password: Maximum 255 karakter hosszú szöveg, kötelező (a jelszó titkosított változatát tartalmazza).

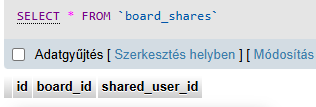
B. boards tábla

* **Cél:** Az egyéni vagy közösen használt Kanban táblák adatait tartalmazza.
* **Struktúra:**
  + id: Automatikusan növekvő elsődleges kulcs.
  + name: A tábla neve, maximum 100 karakter, nem lehet üres.
  + user\_id: Felhasználó azonosítója, mely a tábla tulajdonosát jelöli. Ez az érték kapcsolódik a users tábla id oszlopához (külső kulcs).



C. board\_shares tábla

* **Cél:** Lehetővé teszi, hogy egy tábla több felhasználó számára is elérhető legyen, így támogatva a közös munkát.
* **Struktúra:**
  + id: Automatikusan növekvő elsődleges kulcs.
  + board\_id: A megosztani kívánt tábla azonosítója (külső kulcs, amely a boards tábla id oszlopára hivatkozik).
  + shared\_user\_id: Annak a felhasználónak az azonosítója, akivel a táblát megosztják (külső kulcs, amely a users tábla id oszlopára hivatkozik).
  + **Egyediség:** Az UNIQUE KEY unique\_board\_share (board\_id, shared\_user\_id) biztosítja, hogy ugyanazt a táblát ne lehessen ugyanazzal a felhasználóval kétszer megosztani.

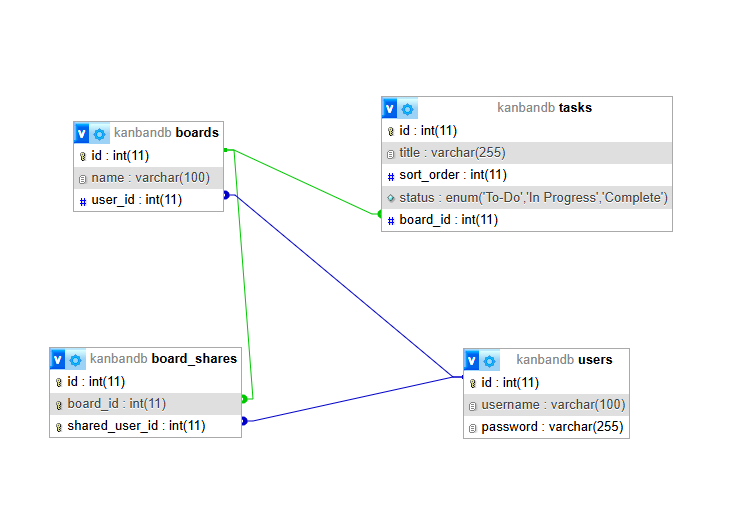


D. tasks tábla

* **Cél:** A Kanban táblákhoz tartozó feladatok kezelésére szolgál.
* **Struktúra:**
  + id: Automatikusan növekvő elsődleges kulcs.
  + title: A feladat címe, maximum 255 karakter hosszúságú, nem lehet üres.
  + sort\_order: Egész szám, amely meghatározza a feladatok sorrendjét, segítve azok rendezését a tábla oszlopán belül.
  + status: Az aktuális állapot, amely három előre definiált értéket vehet fel: 'To-Do', 'In Progress', vagy 'Complete'. Az alapértelmezett érték a 'To-Do'.
  + board\_id: Kapcsolat a megfelelő Kanban táblához; ez a mező a boards tábla id oszlopára hivatkozik (külső kulcs).

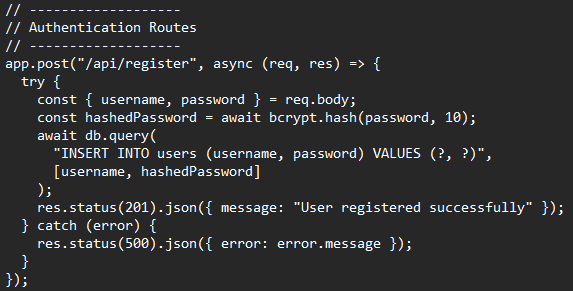
# Táblák közötti Kapcsolatok

* **Users és Boards:** Egy felhasználónak lehet saját Kanban táblája. A boards tábla user\_id oszlopa határozza meg, hogy melyik felhasználóhoz tartozik a tábla.
* **Boards és Board\_shares:** A board\_shares tábla segítségével egy tábla több felhasználó számára is megosztható. Itt két külső kulcs biztosítja a kapcsolatot: az egyik a táblára (board\_id → boards.id), a másik pedig a felhasználóra (shared\_user\_id → users.id). Az egyedi kulcs garantálja a duplikáció elkerülését.
* **Boards és Tasks:** A tasks tábla board\_id oszlopa kapcsolja a feladatokat a Kanban táblákhoz, így minden feladat tisztán a megfelelő tábla részeként jelenik meg.



### API végpontok leírása és tesztelése

**POST /api/register**

****

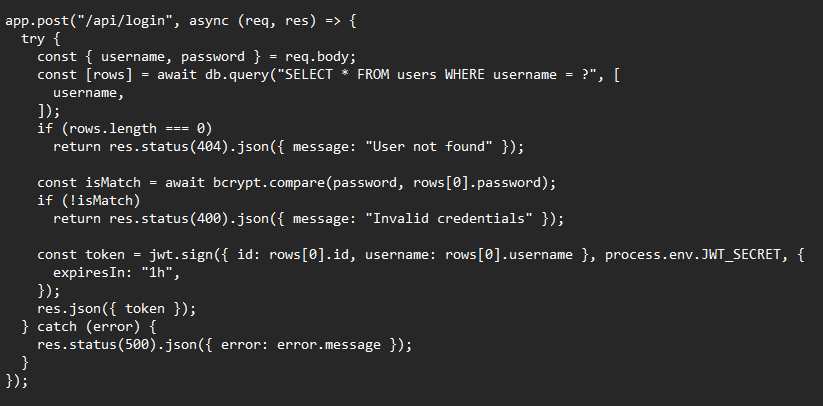
Ez a végpont új felhasználó regisztrációját szolgálja. A kliens JSON objektumként küldi el a kívánt username és password értékeket. A rendszer bcrypt algoritmust alkalmaz a jelszó titkosítására, majd elmenti az új felhasználót az adatbázisba. Siker esetén a rendszer 201-es státuszkóddal válaszol, és egy üzenetet közöl, amely szerint a regisztráció sikeres volt. Tesztelés:

curl -X POST http://localhost:5000/api/register \

-H "Content-Type: application/json" \

-d '{"username": "tesztuser", "password": "jelszo123"}'

**POST /api/login**

****

A bejelentkezési művelet során a kliens szintén JSON formátumban küldi a username és password adatokat. A rendszer ellenőrzi a megadott adatokat a felhasználói adatbázisban, majd, ha minden rendben van, létrehoz egy JWT tokent 1 órás érvényességgel, amelyet visszaad az ügyfélnek. Amennyiben a felhasználó nem létezik vagy a jelszó nem egyezik, hibakódokkal (404 vagy 400) tér vissza a rendszer. Példa:

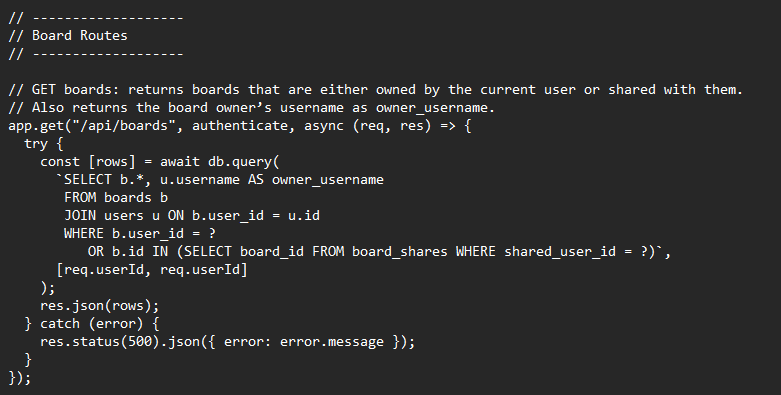
curl -X POST http://localhost:5000/api/login \

-H "Content-Type: application/json" \

-d '{"username": "tesztuser", "password": "jelszo123"}'

### 2. Board Végpontok

**GET /api/boards**

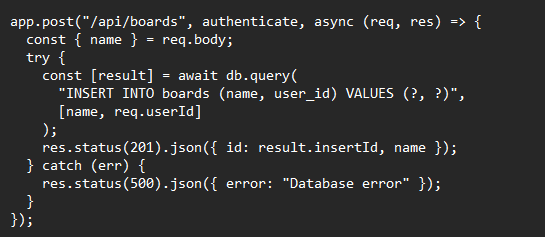
****

Ez a végpont azokat a Kanban táblákat adja vissza, amelyek a bejelentkezett felhasználóhoz tartoznak, vagy amelyek vele megosztottak. A válaszban a tábla tulajdonosának felhasználónevét is megtaláljuk, így könnyen azonosítható, hogy melyik tábla kihez tartozik. A kérelemhez szükséges a JWT token. Példa:

curl -X GET http://localhost:5000/api/boards \

-H "Authorization: Bearer <your\_jwt\_token>"

**POST /api/boards**

****

Új Kanban tábla létrehozását teszi lehetővé. A kliensnek el kell küldenie egy JSON objektumot, amely tartalmazza a tábla nevét. A rendszer automatikusan hozzárendeli az aktuális felhasználót a tábla tulajdonosaként, és 201-es státuszkóddal, valamint az új tábla adataival válaszol. Példa:

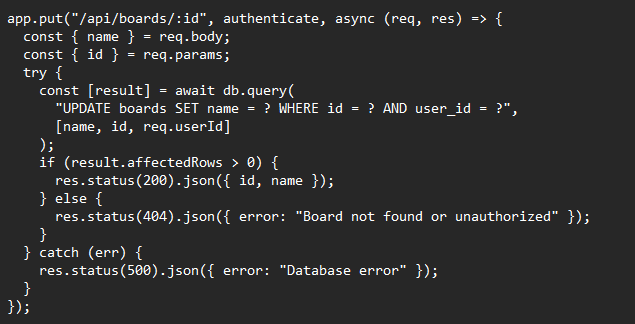
curl -X POST http://localhost:5000/api/boards \

-H "Content-Type: application/json" \

-H "Authorization: Bearer <your\_jwt\_token>" \

-d '{"name": "Új Tábla Neve"}'

**PUT /api/boards/:id**

****

Ezzel a végponttal módosítani lehet egy már létező tábla nevét. A kliens a tábla azonosítóját (URL paraméterben) és az új nevét küldi el. A rendszer ellenőrzi, hogy a tábla valóban a bejelentkezett felhasználó tulajdonában van, majd frissíti az adatokat. Sikeres módosítás esetén a 200-as státuszkódot kapjuk, ha pedig a tábla nem található, vagy a felhasználó nem jogosult a módosításra, hibaüzenettel tér vissza. Példa:

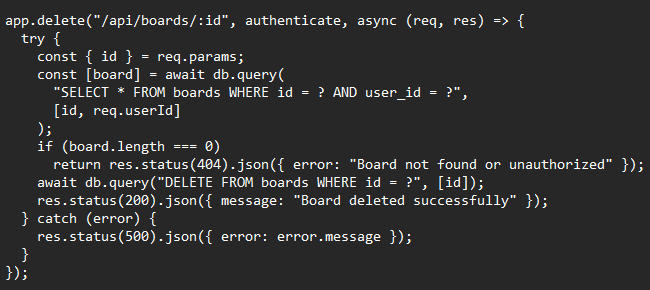
curl -X PUT http://localhost:5000/api/boards/1 \

-H "Content-Type: application/json" \

-H "Authorization: Bearer <your\_jwt\_token>" \

-d '{"name": "Frissített Tábla Név"}'

**DELETE /api/boards/:id**

****

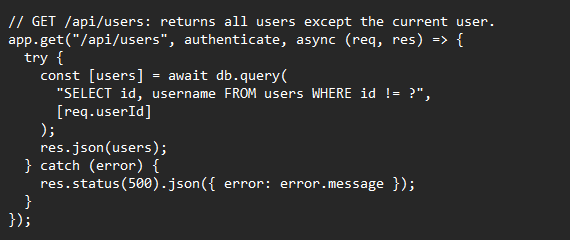
A tábla törlésére szolgáló végpont. A rendszer ellenőrzi, hogy a tábla a felhasználóhoz tartozik-e, és ha igen, törli az adatbázisból. Siker esetén a válasz 200-as státuszkóddal és egy törlési megerősítéssel érkezik. Példa:

curl -X DELETE http://localhost:5000/api/boards/1 \

-H "Authorization: Bearer <your\_jwt\_token>"

### 3. Board Megosztási Végpontok

**GET /api/users**

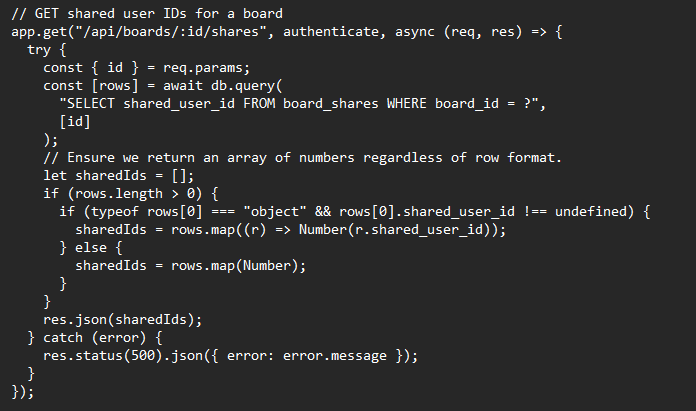
****

Ez a végpont felsorolja az összes olyan felhasználót, akik nem a bejelentkezett felhasználók. Ezt a listát általában a tábla megosztásakor használják, amikor kihez szeretnénk megosztani a táblát. Példa:

curl -X GET http://localhost:5000/api/users \

-H "Authorization: Bearer <your\_jwt\_token>"

**GET /api/boards/:id/shares**

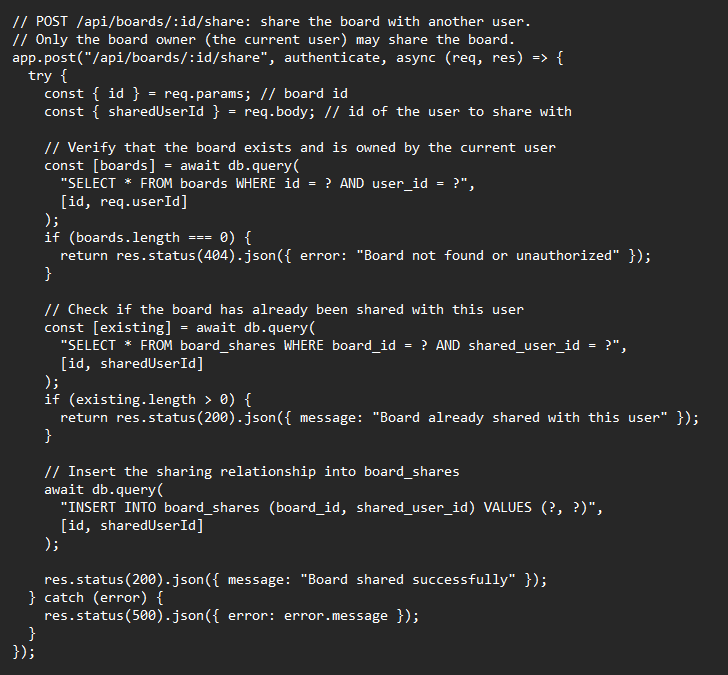
****

Ezzel az végponttal lekérdezhetjük, hogy egy adott táblát mely felhasználókkal osztottak meg. A válasz egy egyszerű számtömb, ahol minden szám egy felhasználó azonosítóját jelöli, akivel a board megosztva van. Példa:

curl -X GET http://localhost:5000/api/boards/1/shares \

-H "Authorization: Bearer <your\_jwt\_token>"

**POST /api/boards/:id/share**

****

A tábla megosztását végző végpont; itt a kliens megadja, hogy melyik felhasználónak akarja megosztani a táblát. Csak a tábla tulajdonosa jogosult erre a műveletre. Ha a tábla már megosztva van ezzel a felhasználóval, a rendszer ezt jelzi, ellenkező esetben sikeres megosztás esetén 200-as státuszkóddal válaszol. Példa:

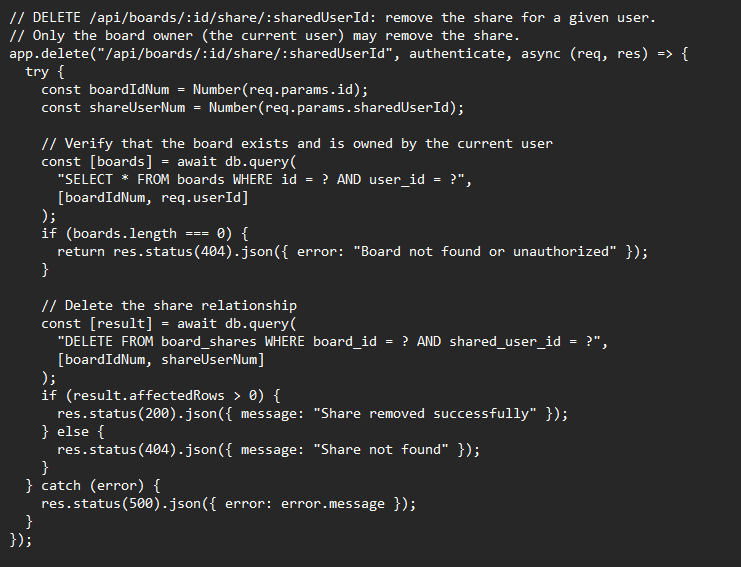
curl -X POST http://localhost:5000/api/boards/1/share \

-H "Content-Type: application/json" \

-H "Authorization: Bearer <your\_jwt\_token>" \

-d '{"sharedUserId": 4}'

**DELETE /api/boards/:id/share/:sharedUserId**

****

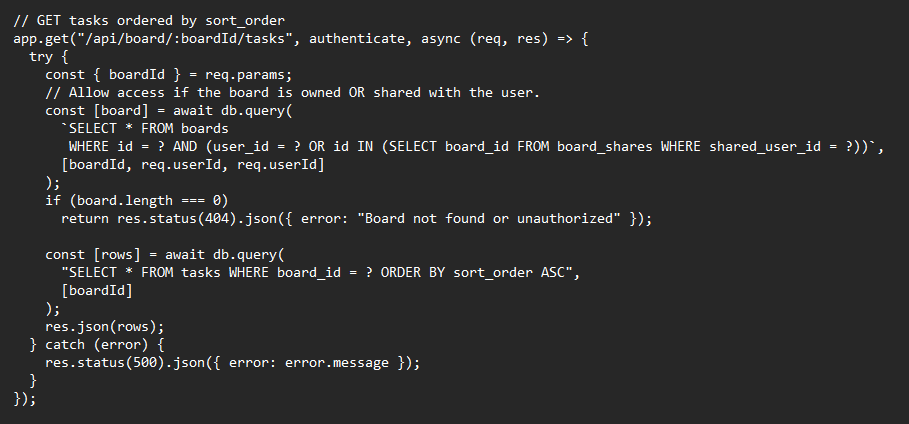
Ezzel a végponttal eltávolíthatjuk egy korábban létrehozott megosztási kapcsolatot. Ismételten, ez a művelet csak a tábla tulajdonosához tartozó jogosultság esetén végezhető el. Sikeres törlés esetén a válaszban egy megerősítő üzenet található. Példa:

curl -X DELETE http://localhost:5000/api/boards/1/share/4 \

-H "Authorization: Bearer <your\_jwt\_token>"

### 4. Feladat (Task) Végpontok

**GET /api/board/:boardId/tasks**

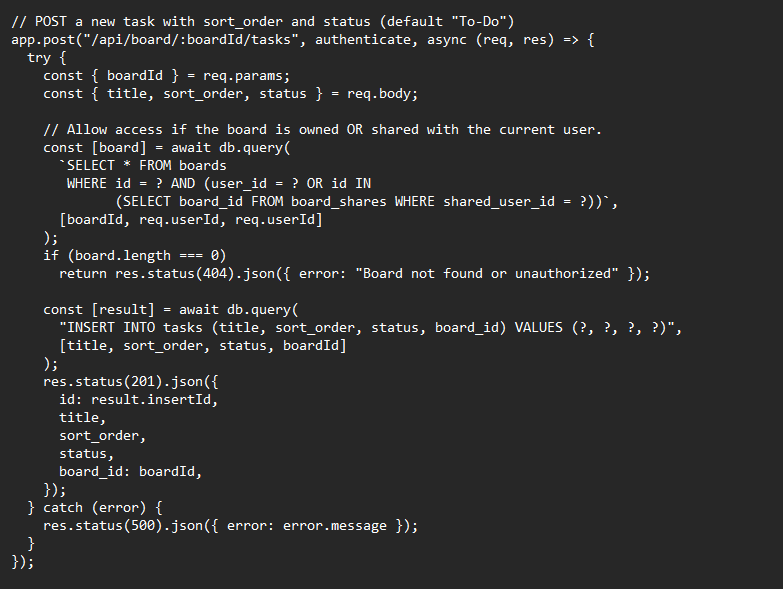
****

Ez a végpont lekéri az adott Kanban táblához tartozó összes feladatot. Először ellenőrzi, hogy a bejelentkezett felhasználó jogosult-e megtekinteni a tábla tartalmát (akár a tábla tulajdonosa, akár megosztott felhasználó). Az eredmény egy olyan lista, ahol a feladatok a sort\_order értékük szerint vannak rendezve. Példa:

curl -X GET http://localhost:5000/api/board/1/tasks \

-H "Authorization: Bearer <your\_jwt\_token>"

**POST /api/board/:boardId/tasks**

****

Új feladat hozzáadását teszi lehetővé az adott tábla számára. A kliens JSON objektumban küldi el a feladat címét, sorrendjét (sort\_order) és státuszát. Mielőtt a rendszer létrehozza a feladatot, ellenőrzi, hogy a kliens jogosult-e hozzáférni a megadott táblához. Sikeres esetben a rendszer 201-es válasszal tér vissza és a létrejött feladat részleteit adja vissza. Példa:

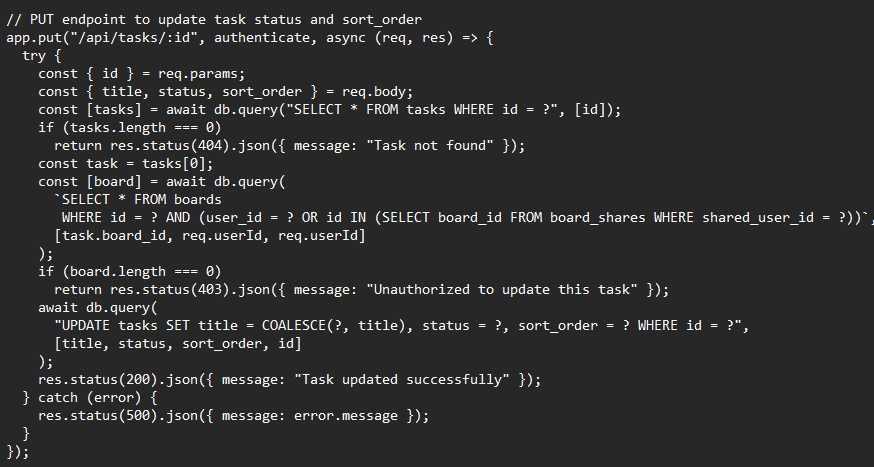
curl -X POST http://localhost:5000/api/board/1/tasks \

-H "Content-Type: application/json" \

-H "Authorization: Bearer <your\_jwt\_token>" \

-d '{"title": "Új feladat", "sort\_order": 1, "status": "To-Do"}'

**PUT /api/tasks/:id**

****

Ez a végpont egy meglévő feladat módosítására szolgál. A kliens elküldi az új értékeket (pl. feladat címe, státusz, sorrend), majd a rendszer először megkeresi a feladatot, majd ellenőrzi, hogy a művelet végrehajtható-e (a feladat a felhasználó által elérhető táblához tartozik-e). Sikeres frissítés esetén a rendszer 200-as státuszkódot ad vissza egy megerősítő üzenettel. Példa:

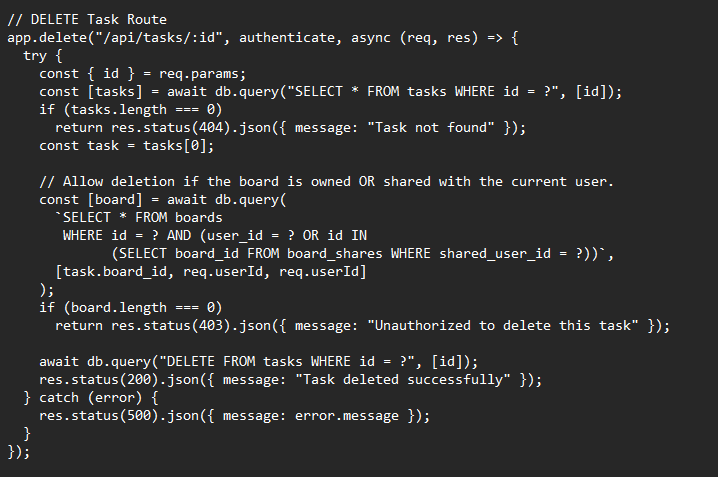
curl -X PUT http://localhost:5000/api/tasks/5 \

-H "Content-Type: application/json" \

-H "Authorization: Bearer <your\_jwt\_token>" \

-d '{"title": "Frissített feladat címe", "status": "In Progress", "sort\_order": 2}'

**DELETE /api/tasks/:id**

****

A végpont törli a megadott feladatot, amennyiben a felhasználó hozzáférést kapott az adott Kanban táblához (akár tulajdonosi, akár megosztott jogosultság alapján). Ha a törlés sikeres, a rendszer 200-as státuszkóddal és egy megerősítő üzenettel válaszol. Példa:

curl -X DELETE http://localhost:5000/api/tasks/5 \

-H "Authorization: Bearer <your\_jwt\_token>"