Könyv nyilvántartó rendszer

# Backend

## books.js:

Ez az Express.js útvonalkezelő (router) a könyvekkel kapcsolatos API-végpontokat tartalmazza:

**GET** */api/books* – Lekérdezi az összes könyvet az adatbázisból.

**POST** */api/books* – Új könyvet ad hozzá, ha még nem létezik ugyanolyan cím–szerző páros.

**DELETE** */api/books* – Törli a megadott cím–szerző párosú könyvet az adatbázisból.

A könyveket a **books** nevű MongoDB gyűjteményben tárolja (**booksCollection**). A függvény paraméterként kapja meg az adatbázis-objektumot, így könnyen újrahasználható más környezetekben is.

## books.json:

Ez a fájl előre definiált könyvrekordokat tartalmaz JSON formátumban, például:

{

"title": "A három test probléma",

"author": "Cixin Liu",

"year": 2006

}

Fő célja, hogy adatbázis inicializáláskor (pl. új gépen) feltölthessük az alapértelmezett könyvlistát a **books** kollekcióba. Ez az **init-db.js** fájlal, történik.

## index.js:

Ez az alkalmazás belépési pontja, amely:

* Inicializálja az Express alkalmazást
* Csatlakozik a MongoDB adatbázishoz
* Betölti és használja a */api/books* útvonalat
* Beállítja az alapértelmezett route-ot (/) és egy */api/test* demó végpontot (Erre nincs szükség a működéshez, nekem kellett a teszteléshez)
* Elindítja a szervert a *.env* fájlban megadott porton

Minden kérés JSON formátumban történik, és CORS engedélyezve van, hogy a frontend böngészőből elérhesse a backend-et.

## init-db.js:

Ez egy egyszer lefuttatható **adatbázis-inicializáló script**, amely:

1. Betölti a *books.json* fájlból a könyveket
2. Törli a meglévő rekordokat a *books* gyűjteményből
3. Beszúrja az új adatokat
4. Lezárja a kapcsolatot

Futtatása az „**node init-db.js**” paranccsal történik terminál használatával.

## .env:

Ez a fájl a környezeti változókat tartalmazza.

A fontos rész a **MONGODB\_URI** – A MongoDB adatbázis elérési útvonala és a **PORT**=3000. A titkos kulcs nem számít

# Frontend

A fontos, nem angulár által generált fájlok nagy része a /src/app/ mappába találhatóak.

Ezek a fájlok ezen a mappán kívűl szerepelnek:

A **styles.css** csak a globális stílusokért, a **main.ts** ami az angulár alkalmazás belépési pontja és az **index.html** ami az Angular alkalmazás HTML alapja (statikus belépési pont). Az Angular itt „ülteti be” az appot a *<app-root></app-root>* elembe.

## app.component.ts:

Ez az alkalmazás fő komponense, amely minden más komponens "kerete":

* **selector: 'app-root'**  
  Ez a komponens a index.html fájl <app-root></app-root> címkéjén keresztül jelenik meg.
* **standalone: true**  
  Ez az Angular 14+ új megközelítése, amely lehetővé teszi komponensek önálló használatát Angular modulok nélkül.
* **imports: [RouterOutlet]**  
  Az útvonalakhoz szükséges kimeneti helyet biztosítja. Itt jelennek meg a különböző oldalak (book-list, book-add, login, stb.).
* **template**  
  A HTML sablon tartalmazza:
  + Az alkalmazás címét: Könyvtár
  + Egy egyszerű navigációs menüt (EZ A /login OLDALON DIREKT NEM JELENIK MEG, MERT AZZAL KI LEHETNE KERÜLNI A BEJELENTKEZÉST)
  + A router-outlet-et, ahova az aktuális oldal tartalma betöltődik

## app.routes.ts:

Ez a fájl definiálja az alkalmazás útvonalait, és megmondja Angularnak, hogy melyik URL-re melyik komponenst töltse be.

| **Útvonal** | **Komponens** | **Funkció** |
| --- | --- | --- |
| *''* | **redirectTo: 'login'** | Alapértelmezett kezdőlap, átirányít a bejelentkezéshez |
| *'login'* | **LoginComponent** | Bejelentkező oldal |
| *'books/list'* | **BookListComponent** | Könyvek listázása |
| *'books/add'* | **BookAddComponent** | Új könyv felvitele |
| *'\*\*' (catch-all)* | **redirectTo: 'login'** | Ismeretlen útvonal esetén visszairányítás a bejelentkezéshez |

## book.model.ts:

Ez a fájl egy **TypeScript interfészt** definiál, amely a könyv adatszerkezetét írja le.

Az \_id mező opcionális (?), mert a frontendről történő új könyv létrehozásakor még nem áll rendelkezésre, de a szerver válaszában már benne van. (ezt a MongoDB amúgy is magától generálja)

## book.service.ts:

Ez a service komponens biztosítja a frontend és a backend közötti kommunikációt a könyvekkel kapcsolatban.

A deleteBook() metódus nem az id-t használja, hanem a könyv **cím + szerző** kombinációját a törléshez – ez összhangban van a backenddel. (A fenti ok miatt nem lehet id, mert az opcionális)

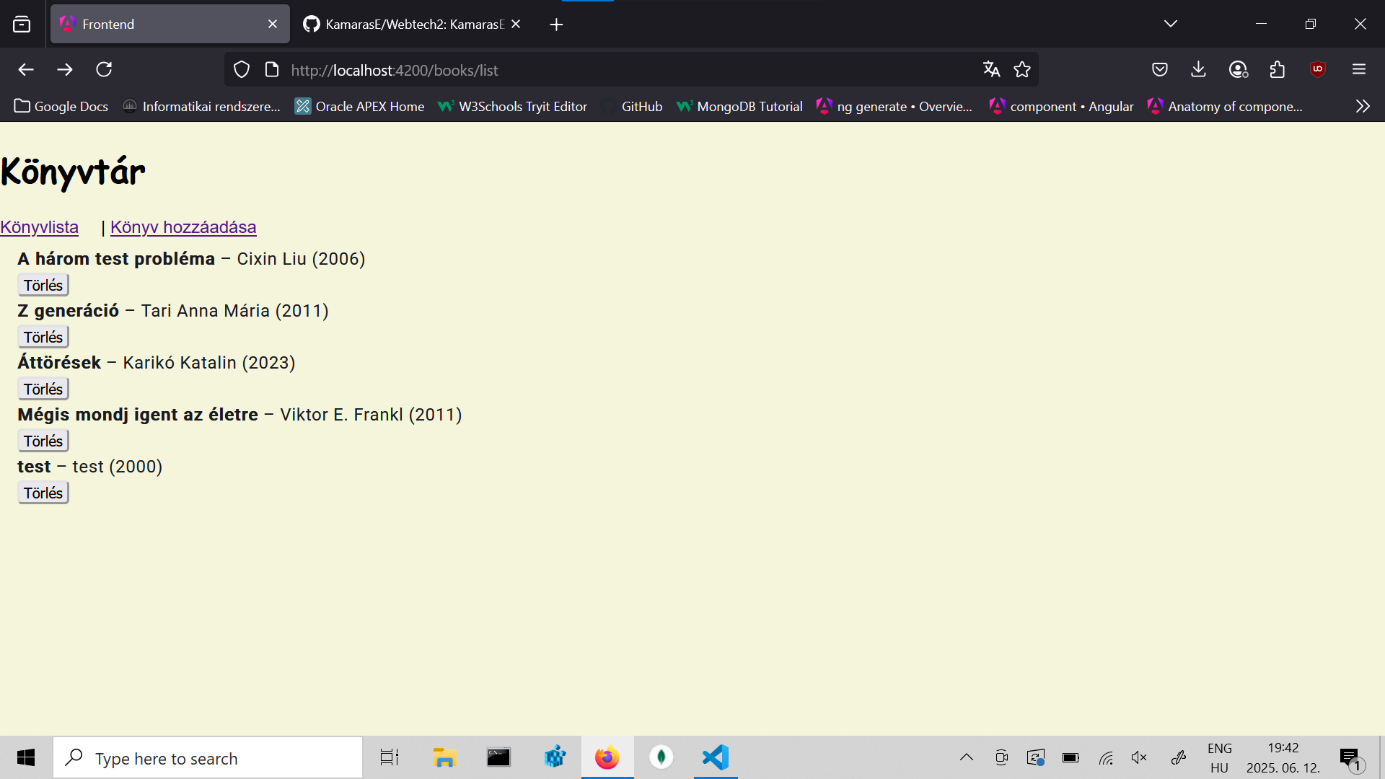
## book-list.component.ts:

A komponens feladata a könyvek kilistázása és törlése. A backendről érkező könyvlistát jeleníti meg, és lehetőséget nyújt az egyes rekordok törlésére.

## book-list.component.html:

Ez a sablon minden könyvet egy mat-list-item elemmel jelenít meg, valamint mellé helyez egy törlés gombot.

* A gombra kattintva meghívódik a deleteBook() metódus
* Hibák esetén megjelenik az errorMessage szöveg

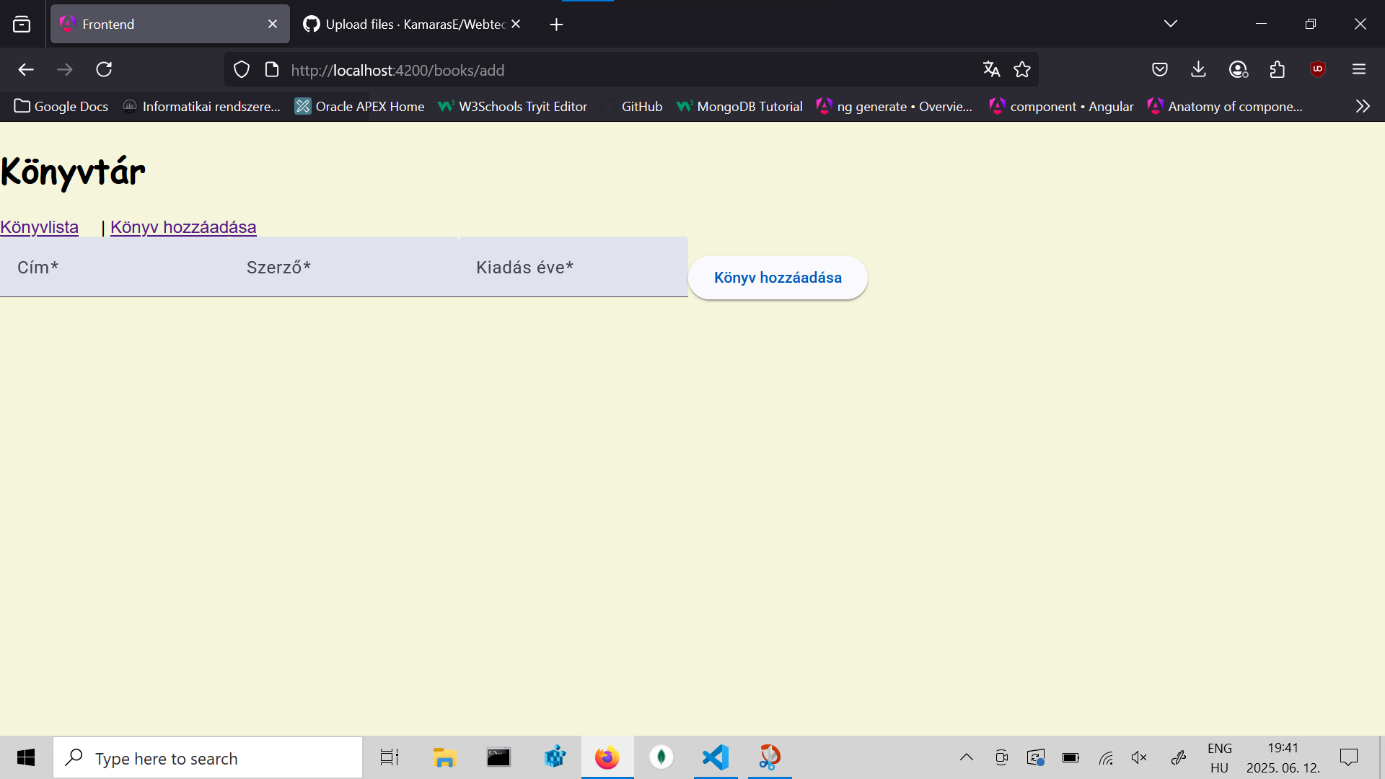
Kép: 

## book-add.component.ts:

Ez a komponens lehetőséget biztosít új könyv felvételére egy űrlap kitöltésével.

## book-add.component.html

Az űrlap három mezőt tartalmaz: könyv címe, szerzője és kiadási éve. A type="submit" gomb az onSubmit() metódust hívja meg, ha az űrlap érvényes.

Kép: 

## login.component.ts:

Ez a komponens egy egyszerű, hardcoded (előre megadott) bejelentkezést valósít meg:

* **Felhasználónév**: root
* **Jelszó**: admin

Sikeres bejelentkezés esetén átirányít a könyvlistához (/books/list), különben hibaüzenetet jelenít meg.

## login.component.html:

* Az űrlap két mezőt tartalmaz: username és password.
* A "Belépés" gomb megnyomásakor az onSubmit() függvény fut le.
* Sikertelen bejelentkezés esetén piros hibaüzenetet jelenít meg.

Kép:

