L3: Node.js laborgyakorlatok

Első rész: node.js projekt

node.js projekt (közös)

A jelen laborfeladattól kezdődően minden repóban egy darab építkező projektet szeretnénk látni. Ennek értelmében **ne bontsuk többet külön alfolderekbe** a feladatokat. Csak egy projekt legyen, s ez induljon a repó gyökeréből. Alakítsuk a tárolónkon levő projektünket *node.js* projektté az `npm` parancssor segítségével.

1. Inicializáljunk `npm` segítségével egy új projektet a repó **gyökerében** (nem `lab3` alfolderben vagy hasonló).
2. Mozgassuk el a kliensoldali kódunkat egy `static` almappába, ahonnan `express` segítségével szolgáljuk a kliensnek. **Minden használt szerveroldali függőséget adjunk meg a `package.json`-ünkben** (használjuk az `npm i --save <packagename>` vagy `npm i --save-dev <packagename>` parancsokat az automatikus frissítésért).
3. Konfiguráljuk az *ESLint* statikus kódellenőrzőt (jelen laborfeladattal kezdődően **elfogadási feltétel**, hogy ez az eszköz ne találjon hibákat kódunkban):
   - Telepítsük lokális projektünkbe az ESLintet, valamint az AirBnB szabályzatot:
     `npm i --save-dev eslint eslint-config-airbnb-base eslint-plugin-import`
   - Bizonyosodjunk meg, hogy a következő 2 állomány egyezik a példarepó gyökerében megadott 2 állománnyal:
     `.eslintrc.yml` és `.eslintignore`
   - Telepítsük a hivatalos *ESLint* Visual Studio Code **extension**t. Így konzolról is futtathatjuk az eszközt (`npx eslint .` a gyökérből), illetve a VSCode is jelez hibákkal.

Második rész: formfeldolgozás

Formfeldolgozás

- Ebben a feladatban klienstől érkezett információkat szükséges feldolgozni és validálni a szerveroldalon. A kliensoldal legyen egy vagy több express `static`-kal felszolgált HTML/CSS oldal, amely tartalmazza az alábbiak alapján megadott formokat.
- **Az adatok validálását végezzük el backenden is**, mivel Postmanből bárki leadhat validálatlan adatot. Hibás adatok esetén küldjünk vissza érthető magyarázatot és **megfelelő HTTP státuszkódot**. Használjunk a formok leadásakor a helyzetnek **megfelelő HTTP metódust**.
- Állományfeltöltés esetén tároljuk az állományokat egy új almappában (ezt helyezzük `.gitignore`-ba, hogy a verziókövetőnk ne kövesse). Más feltöltött adatot tárolhatunk memóriában **logikusan strukturált objektumok formájában**, vagy JSON állományban. Loggoljunk minden lépést megfelelően. Adatbázis használata megengedett, de nem kötelező (ha mégis használnánk, minden szükséges felépítési szkript legyen a repón a reprodukálhatóság érdekében).

3.1. Rendezvényszervező

- Készítsünk egy formot, amellyel adminok rendezvényeket vezethetnek be a rendszerbe. Egy rendezvényhez tartozik név, időintervallum, helyszín, szervező személyek listája (alapértelmezetten üres). A hívás térítsen vissza egy generált egyedi ID-t, amellyel utólag visszakereshető a bejegyzés.
- Készítsünk egy formot, amellyel egy szervező csatlakozhat egy rendezvényhez a neve és a rendezvény egyedi ID-ja alapján. Ugyanezen formon legyen lehetséges visszalépni a szervezési listából. Kezeljük megfelelően a hibákat (pl. ha csatlakozna de már csatlakozott korábban, stb.).
- Készítsünk egy formot, amellyel egy szervező egy rendezvényhez fényképeket tölthet fel. Ehhez meg kell adja a saját nevét és a rendezvény egyedi ID-ját. Ha feltöltéskor a felhasználó még nem szervező a bizonyos rendezvényen, jelezzünk megfelelő hibát.

3.2. Vonattársaság

- Készítsünk egy formot, amellyel adminok vonatjáratokat vezethetnek be a rendszerbe. A járatok tartalmazzák a következőket: honnan, hova, a hét melyik napján, órájában, jegy ára, vonat típusa - gyors, regionális, stb. A hívás térítsen vissza egy generált egyedi ID-t, amellyel utólag visszakereshető a bejegyzés.
- Készítsünk egy formot, amellyel egy felhasználó helyet foglalhat egy bizonyos járaton, ennek egyedi ID-ját alkalmazva. Ha megadott ID-jú járat nem létezik, adjunk megfelelő hibaüzenetet.
- Készítsünk egy formot, amellyel egy vendég kereshet járatok között. Megadhat egy kiindulópontot, célpontot, minimum árat és maximum árat. A visszatérített információ nem szükséges HTML/CSS-sel formázva legyen.

3.3. Lakáshirdetések

- Készítsünk egy formot, amellyel felhasználók lakáshirdetéseket tölthetnek fel a rendszerbe. Egy hirdetés tartalmazza a lakás címét (város, negyed), felszínterületét, árát, szobák számát, feltöltés

dátumát, stb. A hívás térítsen vissza egy generált egyedi ID-t, amellyel utólag visszakereshető a bejegyzés.

- Készítsünk egy formot, amellyel egy felhasználó képeket tölthet fel egy hirdetéshez, az egyedi ID-ját alkalmazva. Ha megadott ID-jú hirdetés nem létezik, adjunk megfelelő hibaüzenetet.
- Készítsünk egy formot, amellyel egy vendég kereshet lakáshirdetések között. Megadhat egy várost, negyedet, minimum árat és maximum árat. A visszatérített információ nem szükséges HTML/CSS-sel formázva legyen.

3.4. Asztalfoglaló vendéglőknek

- Készítsünk egy formot, amellyel egy tulajdonos bevezethet egy vendéglőt a rendszerbe. Egy vendéglő tartalmazza a címét (város, utca, szám külön), telefonszámot, nyitvatartást, stb. A hívás térítsen vissza egy generált egyedi ID-t, amellyel utólag visszakereshető a bejegyzés.
- Készítsünk egy formot, amellyel egy tulajdonos képeket tölthet fel egy vendéglőhöz, az egyedi ID-ját alkalmazva. Ha megadott ID-jú vendéglő nem létezik, adjunk megfelelő hibaüzenetet.
- Készítsünk egy formot, amellyel egy vendég asztalt foglalhat egy vendéglőnél, megadva a nevét, a vendéglő egyedi ID-ját és a foglalás időpontját. Kezeljük a helyzetet amikor nyitvatartáson kívül probál valaki foglalni. Tároljuk a foglalásokat.

3.5. Online napló

- Készítsünk egy formot, amellyel egy tanár létrehozhat új tantárgyakat. Egy tantárgyat körülír az egyedi kódja, leírása, feladatok listája (eleinte üres).
- Készítsünk egy formot, amellyel egy tanár feladatot hozhat létre egy már létező tantárgyhoz. Szükséges információk: a tantárgy egyedi azonosítója (ha még nem létezik ilyen tantárgy, jelezzünk hibát), a feladat rövid leírása, határideje, illetve egy részletes leírás PDF formátumú állomány formájában (csak PDF-eket engedjünk feltölteni).
- Készítsünk egy formot, amellyel egy tanár letörölhet egy tantárgyat az egyedi kódja alapján. Törlés esetén törlődjön a tantárgyhoz tartozó összes feladat is.

3.6. Használt autó kereskedés

- Készítsünk egy formot, amellyel felhasználók használt autókat tölthetnek fel a rendszerbe. Egy hirdetés tartalmazza az autó márkáját, városát, árát, feltöltés dátumát, stb. A hívás térítsen vissza egy generált egyedi ID-t, amellyel utólag visszakereshető a bejegyzés.
- Készítsünk egy formot, amellyel egy felhasználó képeket tölthet fel egy hirdetéshez, az egyedi ID-ját alkalmazva. Ha megadott ID-jú hirdetés nem létezik, adjunk megfelelő hibaüzenetet.
- Készítsünk egy formot, amellyel egy vendég kereshet hirdetések között. Megadhat egy márkát, várost, minimum árat és maximum árat. A visszatérített információ nem szükséges HTML/CSS-sel formázva legyen.

3.7. Online filmgyűjtemény

- Készítsünk egy formot, amellyel egy admin filmet vezethet be a rendszerbe. Szükséges információk: cím, megjelenési év, leírás, zsáner, illetve egy borítókép. A hívás térítsen vissza egy generált egyedi ID-t, amellyel utólag visszakereshető a bejegyzés.

- Készítsünk egy formot, amellyel egy felhasználó egy filmnek visszajelzést (review) adhat, a film egyedi ID-ját alkalmazva. A visszajelzés áll egy pontszámból és egy szöveges véleményezésből. Ha megadott ID-jú film nem létezik, adjunk megfelelő hibaüzenetet.
- Készítsünk egy formot, amellyel egy felhasználó kereshet/szűrhet filmek között. Megadhat egy részleges címet, zsánert, minimum évszámot és maximum évszámot. A visszatérített információ nem szükséges HTML/CSS-sel formázva legyen.

3.8. Sportközpont pályafoglaló

- Készítsünk egy formot, amellyel egy admin bevezethet egy sportpályát a rendszerbe. Egy bejegyzés tartalmazza a pálya típusát (kosárlabda, foci, stb.), órabérárát, címét, leírását. A hívás térítsen vissza egy generált egyedi ID-t, amellyel utólag visszakereshető a bejegyzés.
- Készítsünk egy formot, amellyel egy admin képeket tölthet fel egy sportpályához, az egyedi ID-ját alkalmazva. Ha megfelelő ID-jú pálya nem létezik, adjunk megfelelő hibaüzenetet.
- Készítsünk egy formot, amellyel egy diák pályát foglalhat, megadva a saját nevét, a pálya egyedi ID-ját és a foglalás időpontját. Hogyha a megadott ID-jú pálya nem létezik vagy már foglalt a megadott időintervallumban, térítsünk vissza megfelelő hibaüzenetet.

3.9. Órarendkészítő

- Készítsünk egy formot, amellyel adminok felvihetnek új tantárgyakat. Egy tantárgyat körülír az egyedi kódja, neve, hogy melyik évfolyamnak szól, hány óra kurzus, szeminárium és labor jár vele. Ha egy egyedi kóddal már létezik tantárgy, küldjünk megfelelő hibát.
- Készítsünk egy formot, amellyel egy tanár erőforrás-állományokat (pl. PDF slide-ok) tölthet fel egy tantárgyhoz, az egyedi ID-ját alkalmazva (Vigyázat: több állomány tartozhat egy tantárgyhoz). Ha megfelelő ID-jú tantárgy nem létezik, adjunk megfelelő hibaüzenetet.
- Készítsünk egy formot, amellyel egy diák csatlakozhat vagy kiléphet a tantárgyhoz/-ból, megadva a tantárgy és a felhasználó ID-ját. Hogyha a megadott ID-jú tantárgy nem létezik, vagy a diák státusza nem megfelelő (próbál csatlakozni egy tárgyhoz, de már csatlakozott korábban, vagy próbál kilépni, de nem része), térítsünk vissza megfelelő hibaüzenetet.

3.10. Online könyvtár

- Készítsünk egy formot, amellyel adminok bevezethetnek új könyveket a rendszerbe. Egy könyv egyedi azonosítója az ISBN száma, tartalmaz még címet, szerzőt, kiadási évet, rövid összefoglalót, borítóképet (állomány) és példányok számát.
- Készítsünk egy formot, amellyel egy felhasználó, megadva a nevét és egy könyv ISBN számát, kölcsönözhet egy könyvet. Ha nincs több példány a könyvtárban, kapjon hibaüzenetet. Tároljuk a szerveren, hogy melyik felhasználók mikor kölcsönöztek könyvet.
- Készítsünk egy formot, amellyel egy felhasználó, megadva a nevét és egy könyv ISBN számát, visszaszolgáltathat egy korábban kölcsönvett könyvet. Ha korábban nem vette kölcsön azt a könyvet, kapjon hibaüzenetet.