<https://infojegyzet.hu/webszerkesztes/dokumentacio/>

<https://infojegyzet.hu/webszerkesztes/zarodolgozatmintak/recept24/recept24_vizsgaremek_dokumentacio.pdf>

tartalomjegyzék

* Bevezetés, a téma ismertetése, témaválasztás indoklása, szakmai célkitűzés
* Fejlesztői dokumentáció
* Felhasználói dokumentáció
* Összefoglalás, köszönetnyilvánítás
* Irodalomjegyzék

Fejlesztői dokumentáció

Fejlesztői környezet

A telock szoftverünk elkészítéséhez különböző fejlesztőeszközöket használunk, amelyek segítenek a fejlesztésben és az adatok kezelésében. A Visual Studio Code egy könnyen használható kódszerkesztő, amely segít a weboldal kialakításában. Az adatok kezelésére a Neon-t használjuk, ez egy serverless PostgreSQL adatbázis. A szoftvert Vercel segítségével futtatjuk. Ezeket a fejlesztőeszközöket választottuk, mert tanultunk velük, és ismerjük őket. A széles körű elérhetőségük és az, hogy ingyenesen elérhetők, tovább erősíti döntésünket. Segítségükkel hatékonyan fejleszthetünk és tesztelhetünk weboldalunkat, és könnyedén kezelhetjük az adatbázist.

Domain és tárhely

Az online elérhető szoftverünk megvalósithatósága miatt szükség volt olyan Szerver nélküli (serverless) hoszting oldalra amellyel a NextJS alapú szoftverünket futtatni tudtuk. Azért választottuk a Vercelt, mert ingyenes domain címet és futtatást biztosít

Github és Git környezet

A kiírt igénynek eleget téve és közös munkánk zökkenőmentes végzése miatt, kialakítottunk egy GitHub környezetet a fejlesztésünknek. A környezetet az iskolánk által biztosított email címmel hoztuk létre, majd egy közös repository-t hoztunk létre „vizsgaremek” néven. Így a fejlesztés során végrehajtott változtatások könnyen követhetőkké váltak.

Egyéni környezeten, a https://git-scm.com/ helyről telepitett eszköz lehetőséget adott a saját gépünk és a felhős tárhely közötti Git szabályainak megfelelő kapcsolatot. Az elkészült munka, tartalmazza az adatbázist, a programkódot és a dokumentációt, a következő GitHub repositoryban érhető el: https://github.com/nagygabor123/vizsgaremek. Itt megtalálható az összes szükséges anyag, a projekt teljes körű megértéséhez.

Kialakított adatszerkezet és részletes bemutatása

A telock szoftverünk adatbázisa több táblát tartalmaz. Ezek a táblák strukturáltak és szervezettek, hogy hatékonyan kezeljék és tárolják az adatokat.

Adatbázis táblái

## 1. admins tábla

Ez a tábla az adminisztrátorok adatait tárolja.

Oszlopok:

• admin\_id: Egyedi azonosító (PRIMARY KEY, SERIAL)

• full\_name: Teljes név (VARCHAR(255), NOT NULL)

• password: Jelszó (VARCHAR(255), NOT NULL)

• position: Pozíció (VARCHAR(255), NOT NULL)

• osztalyfonok: Osztályfőnöki szerepkör (VARCHAR(255), NOT NULL)

• short\_name: Rövid név (VARCHAR(8), NOT NULL)

Az admins tábla az iskola alkalmazottainak adatait tárolja. Minden alkalmazottnak van egy egyedi azonosítója (admin\_id), teljes neve (full\_name), titkosított jelszava (password), beosztása (position), osztályfőnöki státusza (osztalyfonok) és rövid neve (short\_name). A tábla alapvető fontosságú a rendszer biztonságához, mivel itt tárolódnak a jogosultsági szintek. Az adminok kezelhetik a diákokat, csoportokat és az órarendeket.

## 2. students tábla

Ez a tábla a diákok adatait tartalmazza.

Oszlopok:

• student\_id: Egyedi diákazonosító (VARCHAR(20), PRIMARY KEY)

• full\_name: Teljes név (VARCHAR(255), NOT NULL)

• class: Osztály (VARCHAR(255), NOT NULL)

• rfid\_tag: RFID azonosító (VARCHAR(50), UNIQUE, NOT NULL)

• access: Hozzáférési jogosultság (VARCHAR(50), NOT NULL)

A students tábla az iskola diákjainak adatait tartalmazza. Minden diákot egyedi azonosító (student\_id) és RFID tag (rfid\_tag) azonosít. A tábla tárolja a diákok teljes nevét (full\_name), osztályát (class) és hozzáférési szintjét (access). Ez a tábla központi szerepet játszik a szekrények használatában és a jelenlét nyilvántartásában. Az RFID tag segítségével azonosítható a diák a szekrényeknél.

## 3. lockers tábla

Ez a tábla a szekrények állapotát tartalmazza.

Oszlopok:

• locker\_id: Egyedi azonosító (SERIAL, PRIMARY KEY)

• status: Szekrény állapota (TEXT, CHECK ("be", "ki"), NOT NULL)

A lockers tábla az iskola szekrényeinek állapotát tárolja. Minden szekrénynek van egy egyedi azonosítója (locker\_id) és állapota (status), ami lehet "be" (zárva) vagy "ki" (nyitva). A tábla nyomon követi, hogy melyik szekrények vannak jelenleg használatban. A szekrények állapota dinamikusan változik a diákok használata során.

## 4. locker\_relationships tábla

Ez a tábla a szekrények és RFID tagek kapcsolatát tárolja.

Oszlopok:

• relationship\_id: Egyedi azonosító (SERIAL, PRIMARY KEY)

• rfid\_tag: RFID azonosító (VARCHAR(50), FOREIGN KEY a students táblára)

• locker\_id: Szekrény azonosító (INT, FOREIGN KEY a lockers táblára)

Ez a tábla a diákok és a szekrények közötti kapcsolatot tárolja. Minden kapcsolatnak van egy egyedi azonosítója (relationship\_id), és tárolja a diák RFID tagját (rfid\_tag) és a szekrény azonosítóját (locker\_id). Ez a tábla biztosítja, hogy egy diák csak egy szekrényt használhasson, és egy szekrény csak egy diákhoz tartozzon. A kapcsolatok törlődnek, ha a diák vagy a szekrény törlődik.

## 5. csoportok tábla

A tanulói csoportokat tartalmazza.

Oszlopok:

• group\_id: SERIAL, PRIMARY KEY

• group\_name: VARCHAR(255), NOT NULL

A csoportok tábla az iskolai csoportokat (pl. osztályok, szakkörök) tárolja. Minden csoportnak van egy egyedi azonosítója (group\_id) és neve (group\_name). Ez a tábla szolgál alapul a tanórák és tevékenységek szervezéséhez. A csoportok segítségével logikai egységekbe lehet szervezni a diákokat különböző tevékenységekhez.

## 6. student\_groups tábla

A diákokat és csoportokat összekapcsoló tábla.

Oszlopok:

• student\_group\_id: SERIAL, PRIMARY KEY

• student\_id: VARCHAR(20), FOREIGN KEY a students táblára

• group\_id: INT, FOREIGN KEY a csoportok táblára

## Ez a tábla a diákok és a csoportok közötti kapcsolatot tárolja. Minden kapcsolatnak van egyedi azonosítója (student\_group\_id), és tárolja a diák azonosítóját (student\_id) és a csoport azonosítóját (group\_id). Egy diák több csoportba is tartozhat, és egy csoportban több diák is lehet. Ez a tábla lehetővé teszi a rugalmas csoportbeosztást.

## 7. timetables tábla

Az órarendi bejegyzéseket tartalmazza.

Oszlopok:

• timetable\_id: SERIAL, PRIMARY KEY

• admin\_id: INT, FOREIGN KEY az admins táblára

• group\_name: VARCHAR(255), NOT NULL

• day\_of\_week: TEXT, CHECK ('monday', 'tuesday', 'wednesday', 'thursday', 'friday'), NOT NULL

• start\_time: TIME, NOT NULL

• end\_time: TIME, NOT NULL

A timetables tábla az órarendeket tárolja. Minden órarendnek van egyedi azonosítója (timetable\_id), és tárolja a létrehozó admin azonosítóját (admin\_id), a csoport nevét (group\_name), a napot (day\_of\_week), valamint a kezdő és végidőt (start\_time, end\_time). Ez a tábla alapvető fontosságú az iskola napi működéséhez, mivel itt tárolódnak az összes óra időpontjai.

## 8. group\_relations tábla

Az órarendek és csoportok kapcsolatait tárolja.

Oszlopok:

• relation\_id: SERIAL, PRIMARY KEY

• timetable\_id: INT, FOREIGN KEY a timetables táblára

• group\_id: INT, FOREIGN KEY a csoportok táblára

Ez a tábla az órarendek és a csoportok közötti kapcsolatot tárolja. Minden kapcsolatnak van egyedi azonosítója (relation\_id), és tárolja az órarend azonosítóját (timetable\_id) és a csoport azonosítóját (group\_id). Ez a tábla biztosítja, hogy egy órarend több csoporthoz is tartozhasson, és egy csoportnak több órarendje is lehet.

## 9. system\_status tábla

A rendszer aktuális állapotát tartalmazza.

Oszlopok:

• id: SERIAL, PRIMARY KEY

• status: VARCHAR(10), NOT NULL

A system\_status tábla a rendszer globális állapotát tárolja. Jelenleg csak egy rekordot tartalmaz, ami a rendszer állapotát mutatja ("nyithato" vagy egyéb érték). Ez a tábla szolgál a rendszer nyitva/zárva állapotának nyilvántartására. Az érték változtatásával a rendszer adminisztrátorai befolyásolhatják a rendszer elérhetőségét.

## 10. year\_schedule tábla

A tanév eseményeit tartalmazza.

Oszlopok:

• year\_schedule\_id: SERIAL, PRIMARY KEY

• type: VARCHAR(255), NOT NULL

• nev: VARCHAR(255), NOT NULL

• which\_day: DATE, NOT NULL

• replace\_day: VARCHAR(255), NOT NULL

A year\_schedule tábla az iskola éves ütemtervét tárolja. Minden eseménynek van típusa (type), neve (nev), dátuma (which\_day) és helyettesítő napja (replace\_day). Ez a tábla tartalmazza a tanév fontos dátumait, mint a tanévkezdés és tanévzárás. Az információkat a rendszer a naptáriszinkronizációhoz használja.

## 11. ring\_times tábla

A csengetési időpontokat tárolja.

Oszlopok:

• id: SERIAL, PRIMARY KEY

• start\_time: TIME, NOT NULL

• end\_time: TIME, NOT NULL

A ring\_times tábla az iskola harangszó időpontjait tárolja. Minden időpontnak van egyedi azonosítója (id), kezdő (start\_time) és végidőpontja (end\_time). Ez a tábla határozza meg a napirend szerinti harangszókat, amelyek az órák kezdetét és végét jelzik. Az időpontok pontos beállítása fontos az iskola napi ritmusának meghatározásához.

Api végpontok:

**URL:** POST /api/setup/ascToDatabase?school\_id={az adott isokal azonosítója}

**Leírás:** Ez az API végpont lehetővé teszi egy XML fájl feltöltését, annak feldolgozását, majd a kinyert adatok több lépésben történő tárolását adatbázisba. Az API az XML adatokat tantárgyak, tanárok, csoportok és órarend formájában dolgozza fel.

Kérés

Metódus: POST

Tartalom típusa: multipart/form-data

Query paraméter: school\_id (SZÜKSÉGES - az iskola azonosítója)

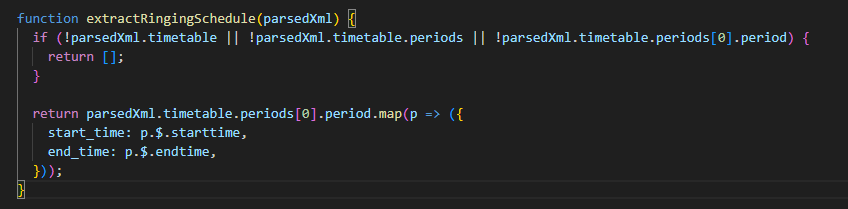
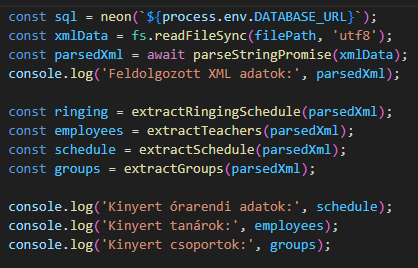
Törzs: A kérés egy XML fájlt kell, hogy tartalmazzon.

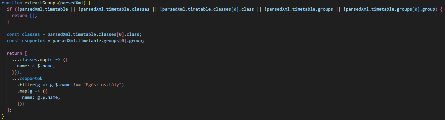
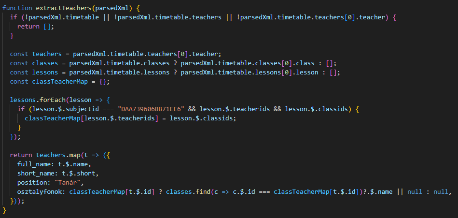
Működés és feldolgozási folyamatok:

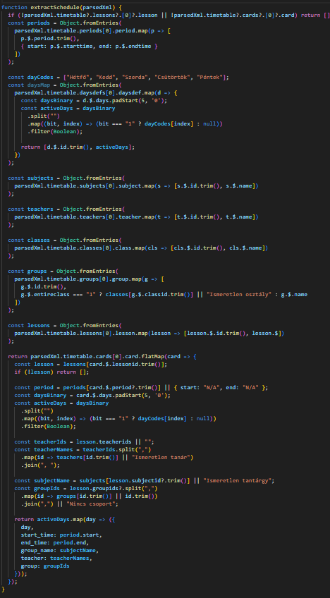
1. **Fájl feltöltés**
   * A rendszer egy multiparty formulán keresztül feldolgozza a beérkezett fájlt.
   * Ellenőrzi, hogy a fájl beérkezett-e.
2. **XML Feldolgozás**
   * A fájl tartalmát az xml2js csomag segítségével JSON formátummá alakítja.
3. **Adatok Kinyerése**
   * **Csengetési rend (ringing)**: A kezdési és befejezési időpontokat tartalmazza.
   * **Tanarok (employees)**: A tanárok nevét, rövid nevét és osztályfőnöki szerepét tartalmazza.
   * **Csoportok (groups)**: Az osztályok és csoportok neveit tartalmazza.
   * **Órarend (schedule)**: Tartalmazza a tantárgyak nevét, tanárokat, időpontokat és napokat.
4. **Adatok Elküldése** Az adatok különböző végpontokra történő elküldése:
   * /api/upload/uploadRinging?school\_id={school\_id} *(ringing tárolása)*
   * /api/upload/uploadEmployees?school\_id={school\_id} *(tanárok tárolása)*
   * /api/upload/uploadGroups?school\_id={school\_id} *(csoportok tárolása)*
   * /api/upload/uploadTimetables?school\_id={school\_id} *(órarend tárolása)*
5. **Adatbázis Készség Ellenőrzése**
   * Az API ellenőrzi, hogy az adatbázis legalább minRows mennyiségű tanárt tartalmaz-e az admins táblában.
   * Ha az adatbázis kész, a órarend törvényesen tárolásra kerül.

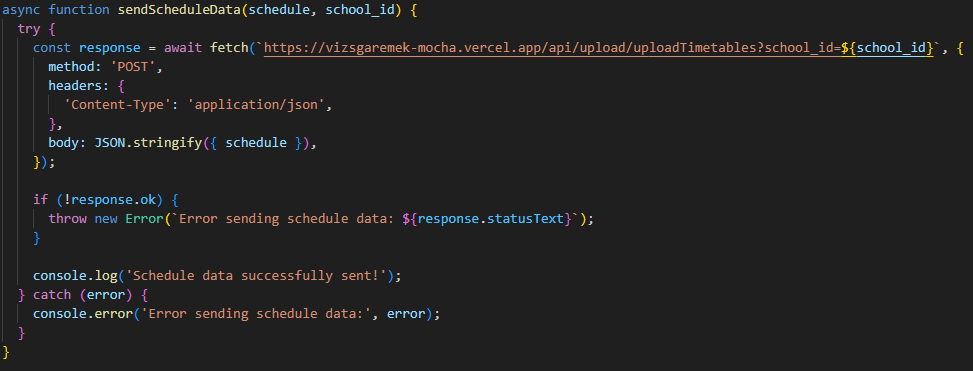
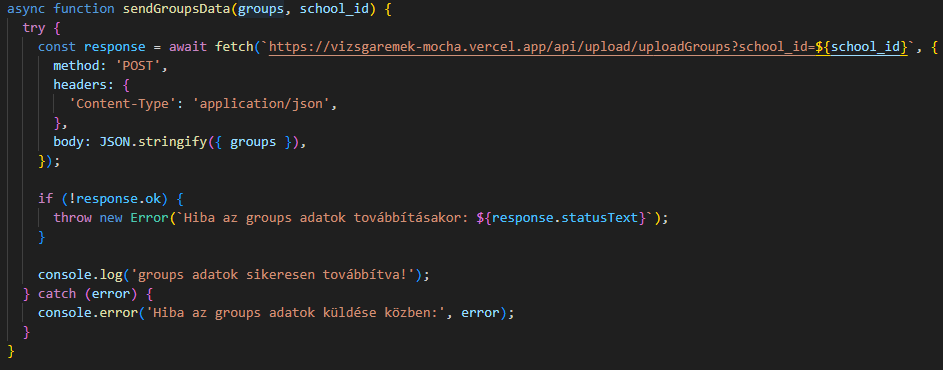
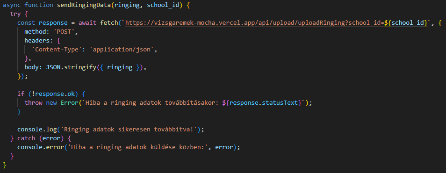
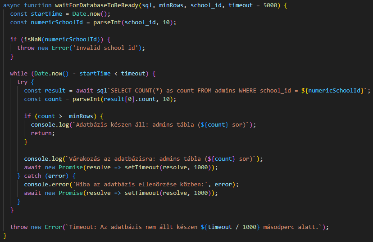
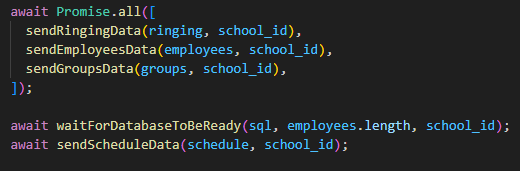
Hibakezelés:

* **Hibás fájlformátum esetén**: A rendszer visszautasítja a kérést 400 Bad Request hibakóddal.
* **Adatbázis elérési hiba esetén**: A rendszer 500 Internal Server Error hibakódot ad vissza.
* **Timeout esetén**: Ha az adatbázis 5 mp alatt nem érhető el, 500 Internal Server Error válasz történik.









**URL:** POST /api/setup/studentsToDatabase?school\_id={az adott isokal azonosítója}

**Leírás:** Ez az API-végpont lehetőséget biztosít egy CSV fájl feltöltésére, amely tartalmazza a diákok adatait. A fájl feldolgozása után az adatok bekerülnek az adatbázisba, majd további API-hívások történnek a diákok csoportjainak és szekrénykapcsolatainak feltöltésére.

Metódus: POST

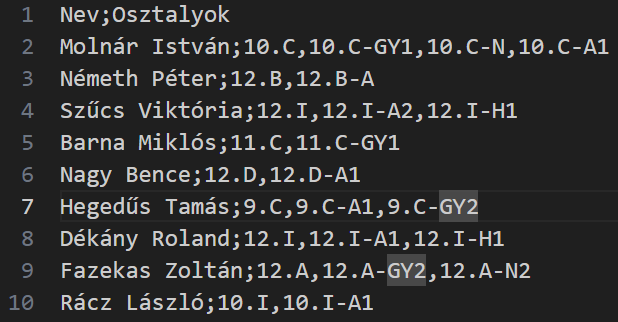
Tartalom típusa: multipart/form-data (CSV fájl)

Query paraméter: school\_id (SZÜKSÉGES - az iskola azonosítója)

Törzs: A kérés egy CSV fájlt kell, hogy tartalmazzon.

## Követelmények

A feltöltéshez egy CSV fájlra van szükség, amely a diákok adatait tartalmazza. A fájl formátum:



## ****Feldolgozás Lépései****

1. **Fájl feltöltés ellenőrzése:**
   * Ha nincs fájl feltöltve, 400 Bad Request hibát ad vissza.
2. **CSV formátum ellenőrzése:**
   * A fájl beolvasása UTF-8 formátumban.
   * A sorok érvényességének vizsgálata (minimum 2 sor kell legyen: fejlec + adatsor).
3. **Adatok feldolgozása:**
   * A fejléc oszlopainak ellenőrzése.
   * A sorok beolvasása és objektummá alakítása.
   * Egyedi azonosító generálása minden diáknak (generateStudentID()).
   * RFID-kód generálása (generateRFID()).
4. **Adatok mentése az adatbázisba:**
   * SQL INSERT parancs segítségével.
   * Ha a mentés sikeres, akkor az adatbázisból visszaellenőrzi a feltöltést (checkStudentsInserted()).
5. **Kapcsolódó API-k meghívása:**
   * uploadStudentGroups()
   * uploadLockerRelations()