Fejlesztői Dokumentáció

Tartalom

[Frontend 1](#_Toc194337187)

[Projekt fáljrendszer elrendezése 1](#_Toc194337188)

[Kiegészítők 2](#_Toc194337189)

[Oldal felépítés 2](#_Toc194337190)

[Lekérdezések 2](#_Toc194337191)

[Adatok 3](#_Toc194337192)

[Fajták 3](#_Toc194337193)

[Visszajelzés 3](#_Toc194337194)

[API 4](#_Toc194337195)

[Projekt fájlrendszer elrendezése 4](#_Toc194337196)

[Kiegészítők: 4](#_Toc194337197)

[Adatbázis 5](#_Toc194337198)

[Táblák 5](#_Toc194337199)

[Adatbázis megtervezése 8](#_Toc194337200)

[Felhasználó rendszer 8](#_Toc194337201)

[Órák 8](#_Toc194337202)

A Páholy rendszer egy olyan felület, ahol egy iskola dolgozói és tanulói egyaránt igénybe vehetik a rendszer által adott információkat, szolgáltatásokat könnyedén egy weboldalon keresztül, ezek között:

* Órarend
* Érdemjegyek
* Hiányzások
* Üzenetek

Az adott felhasználó lekérdezheti ezekkel kapcsolatban az adatait amit az oldal megjelenít egy rendezett módon, és akár fel is tölthet új adatokat (pl tanár beír egy jegyet)

# Frontend

A kezelőfelület a „Vue” keretrendszerrel lett megoldva ami könnyítette az adatok megjelenítését és a fejlesztés folyamatát, egy alap HTML-hez képest sokkal több funkcióval rendelkezik amit próbáltunk a legjobban kihasználni

Emellett a frontend használ más kiegészítőket is, mint például a pinia store-t(lokális adatok kezeléséhez)és az axios-t (API hívásokhoz)

## Projekt fájlrendszer elrendezése

A képen szöveg, képernyőkép, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Fontosabb mappák/fájlok:

* src: „Forrás” mappa, ez tartalmazza a kódot
* api: Lekérdezésekkel kapcsolatos kódok
* lib: Oldal elrendezéssel (route) kapcsolatos kódok
* router: oldalak/aloldalak elérési útvonalait tartalmazza
* stores: Lokálisan/memóriában elmentett adatokat kezelő kódokat tartalmaz (pinia store)
* utils: Lekérdezési kulcsokat tartalmazza (tanstack)
* views: Oldalak/aloldalak kódját tartalmazza, ezek a fájlok vannak behívva a layout mappában lévő rendező fájlok által
* lib: létrehozza a kiegészítőkhöz kellő példányokat amivel behívhatók, például az axiosClient példány amin keresztül lekérdezéseket lehet elbonyolítani

App.vue: Ez a fő fájl, ami létrehozza az oldal példányát és megjeleníti a weboldalt

## Kiegészítők

* Axios: Egy http lekérdezés klienset biztosít amivel kéréseket lehet küldeni
* Tanstack: Lekérdezések gyorsítótárba való lementését biztosítja
* Pinia: Adatok lokális kezelésére létrehozható globális tárolók
* jwt-decode: JSON Web Token dekódolásáért felel
* vuetify: bootstrap megfelelője a Vue-hoz, ad egy alap stílust a Vue tagekhez, + behoz új tageket

## Oldal felépítés

Ha nem vagy bejelentkezve a bejelentkezés oldalra dob be, egyébként meg az órarendhez

Bejelentkezve az oldalon elérhetőek a felhasználó fajtához tartozó aloldalak:

Oldal betöltésének lépései:

1. App.vue inicializálja a klienst, behívja a lib mappából a kiegészítők példányait is
2. RouterView segítségével betöltenek az url alapján, a routerben megadott komponensek
3. Betölt a layoutban megadott .vue elrendezés ami a megfelelő komponenst jeleníti meg
4. A megjelenített komponens(ek)-ben megtörténnek a lekérdezések

A „layout” .vue fájlok szükségesek, mivel így belehet tölteni különböző felületeket ugyan arra az oldalra (pl egy sidebar amit újra szeretnénk használni máshol is, de mondjuk egy másik oldalon meg egyáltalán nem akarjuk hogy megjelenjen)

A „views” mappa tartalmazza ezeket a betöltött komponenseket, funkcionalitásuk alapján vannak mappába rendezve

## Lekérdezések

A lekérdezések az Axios kliensen keresztül történnek meg, és az „api” elnevezési mappa tartalmazza őket, funkcionalitásuk alapján vannak mappába rendezve

Lekérdezések:

* admin: admin felülettel kapcsolatos lekérdezések (pl.: órarend feltöltés, felhasználó feltöltés)
* auth: bejelentkezéssel kapcsolatos lekérdezések
* hazik: feladat felülettel kapcsolatos lekérdezések (feladatok lekérdezése, válasz felküldés)
* hianyzasok: hiányzásokkal kapcsolatos lekérdezések (hiányzások lekérdezése, hiányzás beírás)
* jegyek: érdemjegyekkel kapcsolatos lekérdezések (érdemjegyek lekérdezése, érdemjegyek beírás)
* orarend: órarenddel kapcsolatos lekérdezések (órarend lekérdezés)
* szulo: szülő oldali lekérdezések (minden féle lekérdezés szülő oldalra átírva, hogy ne a szülő adatait kérdezze le, hanem a kiválasztott gyerek adatait)
* uzenetek: üzenetekkel kapcsolatos lekérdezések (üzenet lekérdezés)

Minden lekérdezés egy „useQuery” vagy „useMutation” függvényben van a tanstack miatt, így azokkal az adatokkal könnyebb dolgozni, pl láthatjuk a kérés státuszát (sikerült, folyamatban, nem sikerült), és könnyen lehet újra kérni ezeket az adatokat (refetchQueries függvénnyel)

## Adatok

A legtöbb adat már elrendezve jön meg az API-tól, viszont sok helyen még az oldal kódján belül is rendezésre kerül. A vue egyszerűvé teszi a megjelenítését v-forral, sok táblázat cellái megvannak magában az oldalban írva egyszer, mint egy sablon, beolvassa a lekérdezett adatot és megjeleníti adattagonként

A Vue egyik fontos része a „ref” megnevezés amivel egy referencia objektumot hozhatunk létre, ezért nagyon hasznos új adatok felvitelénél még a lokális tároláshoz, majd abból kiolvasásához, és akár több helyről is belehet olvasni ugyan azt az adatot úgy hogy ha változik akkor az a többi helyen is megváltozik

## Fajták

Typescript-ben lett megírva minden lekérdezés és számítás kód, így sok helyen kell fajtákat használni, a lekérdezés fájlok mellett a „kerdezes\_nev.d.ts” nevű fájlokban találhatóak a visszakapott adatok fajtáinak deklarálása

## Visszajelzés

A képen szöveg, képernyőkép, embléma, zöld látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Az „errorHandler” és „statusHandler” tároló felel a lekérdezések visszajelzésére, ami bárhonnan meghívható és a „setError” és „setStatus” függvénnyel beküldhető egy szöveg amit megjelenít az oldal alján. Az App.vue-ban helyezkedik el a megjelenítő kód hogy bárhol látszódjon, akár még aloldal váltás után is a visszajelzés

# API

Az adatbázishoz tartozó API lehetővé teszi hogy lekéréseket kérhessen a weboldal, egy kontrolált módon, esetleges hiba esetén pedig lekezelje azokat és meg

## Projekt fájlrendszer elrendezése

A képen szöveg, képernyőkép, szoftver, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Fontosabb mappák/fájlok:

* \_mocks\_: teszteléshez használt adatbázis másolat létrehozó fájlt tartalmazza
* tests: tesztek
* routes: végpontokat tartalmazza
* controllers: irányítók amik egyenesen a routes-ból vannak meghívva
* services: függvények amik az adatbázisból lekérdezett adatokkal számítanak, legtöbbször csak magát az adatot küldik tovább
* repositories: a táblákhoz való lekérdezéseket tartalmazza
* models: sequelize-al itt vannak meg adva az adatbázis táblái és köttetései
* app.js: Ez a fő fájl, ami létrehozza az adatbázist/megnézi hogy létezik e az adatbázis, meghívja a routes-ból a végpontokat

## Kiegészítők:

* express: web applikáció keretrendszer, lehetővé teszi a végpontok definiálását
* nodemon: automatikusan újra indítja az applikációt egy változtatás után
* jest: tesztelési keretrendszer
* sequelize: ORM adatbázis létrehozását teszi lehetővé
* bcrypt: titkosítháshoz
* jsonwebtoken: token generáló rendszer, felhasználók ellenőrzésére

## Végpont dokumentáció

Végpontokhoz egy kigenerált dokumentációt készítettünk el, amely elmagyarázza az egyes végpontokhoz beküldendő adat formátumát, és válaszait, „/documentation” végponton elérhető webes formában

# Adatbázis

Az adatbázis nem nyers SQL-ben lett megírva hanem a sequelize segítségével, legfőbb előnye hogy json objektum formában lehet megadni és módosítani a táblákat, és az adatok lekérdezése és feltöltése is egyszerűbben megoldható

## Táblák

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Minden fájl egy külön tábla mezőit és tulajdonságait tartalmazza, ez alapján készíti el az SQL-es adatbázist

Táblák rövid leírásai:

* absences: tanulók hiányzását rögzíti

1. ID: azonosító (elsődleges kulcs)
2. studentID: hiányzó diák azonosítója
3. teacherID: hiányzást naplózó tanár azonosítója
4. date: hiányzás dátuma
5. excused: igazolás státusza
6. lessonID: hiányzott óra azonosítója

* admins: adminisztrátori fiókok adatait rögzíti

1. ID: azonosító (elsődleges kulcs)
2. name: adminisztrátor teljes neve
3. phone: adminisztrátor telefon száma
4. email: adminisztrátor email címe
5. userID: felhasználó(user) táblában lévő fiókhoz kötött bejelentkezési adatok azonosítója

* assignmentfiles: feladatokhoz csatolt fájlokat rögzíti

1. ID: azonosító (elsődleges kulcs)
2. assignmentID: feladat azonosítója amihez tartozik a fájl
3. buffer: fájl
4. mimetype: fájl kiterjesztése
5. filename: fájl neve

* assignments: tanárok által kitűzött feladatokat rögzíti

1. ID: azonosító (elsődleges kulcs)
2. teacherID: adminisztrátor teljes neve
3. desc: feladat leírása
4. deadline: feladat határideje
5. uploadDate: feladat kitűzésének dátuma

* classdisruptions: helyettesített/módosult órák adatait rögzíti

1. ID: azonosító (elsődleges kulcs)
2. date: helyettesített/módsítás dátuma
3. groupID: csoport azonosítója akikhez az óra lett rendelve
4. start\_Hour: óra kezdete órában (óra:perc)
5. start\_Minute: óra kezdete percben (óra:perc)
6. length: óra hossza
7. day: melyik napon van az óra (hétfő,kedd, stb…)
8. subjectName: óra tantárgya

* completedassignments: feladatokra beadott választ rögzíti

1. ID: azonosító (elsődleges kulcs)
2. assignmentID: feladat azonosítója amihez tartozik a válasz
3. studentID: feladatot leadó tanuló azonosítója
4. date: feladat leadásának dátuma
5. textAnswer: leadott válasz szöveges tartalma
6. status:leadás státusza (leadva, nincs leadva, határidő lejárt)

* completedassignmentfiles: elkészült feladatokhoz csatolt fájlokat rögzíti

1. ID: azonosító (elsődleges kulcs)
2. assignmentID: feladat azonosítója amihez tartozik a fájl
3. buffer: fájl
4. mimetype: fájl kiterjesztése
5. filename: fájl neve

* groups: csoportokat rögzíti

1. ID: azonosító (elsődleges kulcs)
2. name: csoport neve

* guardians: szülő fiókok adatait rögzíti

1. ID: azonosító (elsődleges kulcs)
2. name: szülő teljes neve
3. phone: szülő telefon száma
4. email: szülő email címe
5. userID: felhasználó(user) táblában lévő fiókhoz kötött bejelentkezési adatok azonosítója

* guardianstudents: kapcsoló tábla (szülő<gyerek)

1. GuardianID: szülő azonosító (idegenkulcs)
2. StudentID: gyerek(diák) azonosító (idegenkulcs)

* lessons: órákat rögzíti

1. ID: azonosító (elsődleges kulcs)
2. groupID: csoport azonosítója akikhez az óra lett rendelve
3. start\_Hour: óra kezdete órában (óra:perc)
4. start\_Minute: óra kezdete percben (óra:perc)
5. length: óra hossza
6. day: melyik napon van az óra (hétfő,kedd, stb…)
7. subjectName: óra tantárgya

* marks: jegyeket rögzíti

1. ID: azonosító (elsődleges kulcs)
2. teacherID: jegyet naplózó tanár
3. studentID: jegyet kapó tanuló
4. Value: jegy értéke (1,2,3,4,5)
5. Multiplier: hány százalékos a jegy
6. subjectName: tantárgy amiből kapta a jegyet
7. date: jegy beírásának dátuma

* messagereceivers: kapcsoló tábla (üzenet-felhasználó)

1. MessageID: üzenet azonosító (idegenkulcs)
2. UserID: felhasználó azonosító (idegenkulcs)

* messages: üzeneteket rögzíti

1. ID: azonosító (elsődleges kulcs)
2. senderUserID: felhasználó azonosító (idegenkulcs)
3. message: üzenet tartalma
4. date: üzenet küldésének dátuma

* studentgroups: kapcsoló tábla (csoport-tanuló)

1. GroupID: csoport azonosító (idegenkulcs)
2. StudentID: tanuló azonosító (idegenkulcs)

* students: tanuló fiókok adatait rögzíti

1. ID: azonosító (elsődleges kulcs)
2. name: tanuló teljes neve
3. DoB: születési dátum
4. address: tanuló lakcíme
5. phone: tanuló telefon száma
6. email: tanuló email címe
7. userID: felhasználó(user) táblában lévő fiókhoz kötött bejelentkezési adatok azonosítója
8. OMID: oktatási azonosító

* teachers: tanár fiókok adatait rögzíti

1. ID: azonosító (elsődleges kulcs)
2. name: tanár teljes neve
3. phone: tanár telefon száma
4. email: tanár email címe
5. userID: felhasználó(user) táblában lévő fiókhoz kötött bejelentkezési adatok azonosítója

* users: fiókok bejelentkezési adatait rögzíti

1. ID: azonosító (elsődleges kulcs)
2. username: felhasználónév
3. password: jelszó
4. role: felhasználó szerepköre (diák,tanár,szülő,admin)

A képen szöveg, képernyőkép, diagram, szám látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

## Adatbázis megtervezése

Igyekeztünk minél jobban megtervezni az adatbázist hogy később a fejlesztés során ne hátráljon minket az adatbázis megoldás módja, így több helyen is más megoldásokkal kellett feljönnünk mint amit eredetileg kigondoltunk

### Felhasználó rendszer

A rendszerünk alapköve a 4 felhasználó fajta (diák,tanár,szülő,admin), tehát úgy kellett megterveznünk ezeket a táblákat hogy könnyedén validálni lehessen a bejelentkezéseket amellett hogy különböző fajta felhasználókat is kezelünk, így született meg a „users” tábla ami csak a bejelentkezési adatokat menti le egy adott fiókhoz, és validálás után a felhasználó „role” mezője alapján letudja kérdezni az adott felhasználó adatait.

### Órák

Óráknál nem akartuk az összes létező dátumra lementeni az órákat külön-külön, így csak egy „hétnyi” órát mentünk le, tehát nem kell annyi órával dolgozni. Helyettesítéseket és óra módosulásokat pedig a „classdisruption” táblán keresztül kezeljük le, ahol meglehet adni adott dátumra az ilyen információkat, amit azután frontenden a rendes órarenddel össze kombinálva meglehet jeleníteni minden hétre, duplikálva az órákat és utána rájuk húzva a módosulásokat

# Tesztelés

Teszteléshez a Jest segítségét hívtuk ami adott nekünk egy keretrendszert a backend teszteléséhet, a projekt fájl mappájában található egy „tests” nevű mappa ami tartalmazza a teszteket, a tesztek külön-külön fájlokba vannak rendezve „.test.js” végződéssel

Az adatbázis szimulálva van a projekt fájlban megadott táblák és kapcsolatok alapján, és az adataik nem mentődnek az egyes teszt futtatások után, hanem mindig újra legenerálódnak és feltöltésre kerülnek az szimulált adatbázisba

Minden egyes tesztnél az adott felhasználónak aki „szemszögéből” végezzük a kérést, annak generálunk egy token-t amit beállítunk a kérés fejlécében a kérés előtt

A képen szöveg, Betűtípus, képernyőkép, tervezés látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Az oldal egyes főbb funkcióit teszteljük le, amik így gyorsíthatják a hibák megkeresését

Az összes tesztfájlban a teszteknek megvan adva szövegesen hogy milyen tesztek és hogy mit tesztelnek le

## Hiányzás tesztek

Leteszteli a hiányzások lekérdezését az első két tesztben a végpontokon keresztül, a másik két teszt pedig a feltöltésüket, ha jól és egy rosszul megformázott objektumot küldesz fel

A várt és felküldött objektumok megegyeznek:

A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

## Jegy tesztek

Leteszteli a jegyek lekérdezését az első két tesztben a végpontokon keresztül (diák és tanár oldalról is), a másik két teszt pedig a feltöltésüket, ha jól és egy rosszul megformázott objektumot küldesz fel

A várt és felküldött objektumok megegyeznek:

A képen szöveg, képernyőkép, menü látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.