Fejlesztői Dokumentáció

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 fáljrendszer 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, és a script nem is nyúl hozzájuk, beolvassa a lekérdezett adatot és megjeleníti adattagonként

A vue egyik fontos része a „ref” megnevezés amivel egy referencia objektumot hozhatunk létre aminek bármilyen fajta értéke lehet, ezért nagyon hasznos új adatok felvitelénél még a lokális tároláshoz, majd abból kiolvasásához

## Fajták

Typescriptben 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” store 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 irányított módon

## Projekt fáljrendszer elrendezése

A képen szöveg, képernyőkép, szoftver, Betűtípus látható

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

* \_mocks\_: teszteléhez használt adatbázis másolat létrehozó fáljt 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
* repositoreis: a táblákhoz való lekérdezéseket tartalmazza
* models: sequilize-al itt deklarálódnak 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 mentés után
* jest: tesztelési keretrendszer
* sequelize: ORM adatbázis létrehozását teszi lehetővé
* bycrypt: titkosítháshoz
* jsonwebtoken: token generáló rendszer, felhasználók validálására

# Adatbázis

Az adatbázis nem rendes SQL-ben lett megírva hanem a sequielize segítségével, legfőbb előnye hogy json objektum szerűen lehet módosítani, és egyszerű benne megírni az adatbázist

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