**Math Solve**

Fejlesztői dokumentáció

# Bevezetés

Ez a dokumentáció a projekt fejlesztésében közvetlenül részt vevő csapattagoknak, valamint az újonnan érkező fejlesztőknek szól. A dokumentum segít abban, hogy mindenki pontosan megértse a rendszer felépítését, a használt eszközöket, valamint a fejlesztési és tesztelési folyamatokat.

## Dokumentáció célja:

* Részletes architekturális áttekintést nyújtani.
* Leírni a fejlesztői környezet és eszközök beállítását.
* Ismertetni a kódszerkezetet, a mappastruktúrát és a moduláris felépítést.
* Bemutatni a tesztelési stratégiát és az automatizált tesztelési folyamatokat.

## Admin felhasználó:

* Fiók név: pinteadani88@gmail.com
* Password: admin12345

# Projekt architektúra és mappastruktúra

## Technológiai Stack:

A projekt több rétegből épül fel:

* **Frontend:**
  + **Vue 3** – A projekt alapja, a felhasználói felület építéséhez használt modern JavaScript framework.
  + **Vuetify** – A felhasználói felület gyors fejlesztését segítő material design komponens könyvtár**.**
  + **Vitest** –A tesztelési keretrendszer, amely biztosítja az alkalmazás egységtesztelését.
  + **Concurrently** –A frontend és backend egyszerre történő futtatásához szükséges eszköz.
  + **Pinia** –Állapotkezelésre szolgáló könyvtár a Vue 3-ban, amely biztosítja a globális állapot kezelését az alkalmazásban.
* **API:**
  + **Axios** –HTTP kérés küldésére és válaszok kezelésére használt könyvtár.
  + **TanStack Query** – Aszinkron adatlekérések és szerver oldali állapotkezelés kezelésére szolgáló könyvtár.
* **Backend:**
  + **Express** – A backend szerver keretrendszer, amely az alkalmazás logikáját és API-jait kezeli.
  + **Sequelize** – ORM (Object-Relational Mapping) könyvtár, amely lehetővé teszi az adatbázis kezelést JavaScript-ben.
  + **Dotenv** – Környezeti változók kezelésére használt könyvtár.
  + **Swagger** – API dokumentáció generálására és tesztelésére használt eszköz.
  + **Multer** – A fájlok feltöltéséhez használt middleware.
* **Database:**
  + **Mysql** – Adatok tárolására való környezet.

## Mappastruktúra:

Ezen struktúra részletes leírása segíti a fejlesztőket abban, hogy gyorsan megtalálják a releváns kódrészeket.

[[mappa struktúrák](structure.txt)]

# Fejlesztői környezet beállítása

## Előfeltételek:

* Telepítsd a [Node.js-t](https://nodejs.org/), ha még nincs telepítve.
* Telepítsd a [Git-t](https://git-scm.com/), ha még nincs telepítve.
* Telepítsd a [XAMP](https://www.apachefriends.org), ha még nincs telepítve.

## Projekt indítása

* Indítsd el a start.bat fájlt ami a projekt mappájában találsz
* Írd be a 2-es számot

# Kódszerkezet és moduláris felépítés

**Frontend:**

* A komponensek egyértelműen elkülönülnek, minden komponens saját mappában található.
* Az állapotkezelés a Pinia segítségével történik, így a globális állapot könnyen kezelhető.
* A Vue komponensekben a logika és a megjelenítés elkülönítése segíti az új funkciók gyors integrálását.

**Backend:**

* Az Express.js szerver felelős az API végpontok kezeléséért.
* A middleware-ek és router-ek használata biztosítja a kód moduláris felépítését.
* Az adatbázis műveleteket a Sequelize ORM kezeli, így a modellek és az adatbázis kapcsolatok egy helyen kezelhetők.
* A hibakezelés és logging megfelelően megoldott, hogy könnyen nyomon követhetőek legyenek a problémák a futás során.

# API dokumentáció és hibakezelés

A projekt **backend** API-jának dokumentációját a **Swagger** segítségével generáltuk, amely biztosítja a felhasználók és fejlesztők számára az API végpontjainak könnyű elérését és használatát.

### Mi a Swagger?

A **Swagger** (más néven OpenAPI) egy eszközkészlet, amely lehetővé teszi az API dokumentáció automatikus generálását, tesztelését és megértését. A Swagger UI egy vizuális felületet biztosít az API végpontjainak böngészésére, ahol a felhasználók megtekinthetik az összes elérhető végpontot, azok paramétereit, válaszait, és kipróbálhatják a különböző HTTP kéréseket.

### Hogyan érheted el a Swagger dokumentációt?

A Swagger API dokumentáció az alábbi linken érhető el a projekt **backend** részén:

**A helyi fejlesztési környezetben elérhető a Swagger UI az alábbi URL-en, miután a backend szerver fut:**

[**Swagger API Dokumentáció**](http://localhost:3000/api-docs)

### Használat:

1. **API végpontok böngészése** – Az összes elérhető API végpont megtekintése.
2. **API kérések kipróbálása** – A Swagger UI lehetővé teszi a különböző API kérések tesztelését, például GET, POST, PUT, DELETE kéréseket. A felhasználók egyszerűen beírhatják a kívánt paramétereket és végrehajthatják a kérést közvetlenül az UI-ból.
3. **Példák és válaszok** – Minden végponthoz tartozik egy példakérés és a választ is bemutatja a Swagger, így könnyen megérthető, hogy miként működik az adott API végpont.

## Tesztelés

A projekt tesztelése a backend alkalmazásban a [Jest](https://jestjs.io/) és [Supertest](https://www.npmjs.com/package/supertest) segítségével történik, amelyek lehetővé teszik a unit tesztelést és az API végpontok integrált tesztelését.

### Tesztelési keretrendszer

* **Jest**: A Jest egy JavaScript tesztelési keretrendszer, amely a tesztelési logikát és a tesztek futtatását kezeli. Lehetővé teszi az aszinkron műveletek kezelését, és részletes tesztelési eredményeket biztosít.
* **Supertest**: A Supertest egy HTTP aszinkron tesztelő eszköz, amely lehetővé teszi az API végpontok tesztelését. A Supertest segítségével szimulálhatunk HTTP kéréseket és ellenőrizhetjük a válaszokat.

### Tesztelési parancsok

* **A tesztek futtatása**: Elsőnek indítsd el a srart.bat fájlt és írd be a 3-as számot.

### Tesztelési struktúra

A tesztek a backend projektben a *tests* mappában találhatók. Az egyes tesztelési fájlok általában az adott funkciók vagy API végpontok tesztelésére vannak kialakítva.

* **Unit tesztek**: A backend logika egyes részleteit, mint például az adatbázis műveletek vagy egyes segéd funkciók, unit tesztekkel ellenőrizzük.
* **API tesztek**: Az API végpontokat a Supertest segítségével teszteljük. Az API tesztek szimulálják a HTTP kéréseket és ellenőrzik a válaszokat, például a státuszkódokat, a válaszadatokat és a hibakezelést.
* **Adatbázis**: Biztosítani kell hogy az adatbázis MySQL-ben létre van hozva és eléri.

### Tesztelési hibák kezelése

Amennyiben tesztelés közben hibák merülnek fel, ellenőrizni kell:

* **Az adatbázis kapcsolódik-e megfelelően**: Biztosítani kell, hogy a tesztelés előtt az adatbázis csatlakozik és inicializálódik.
* **Tesztadatok létezése**: Tesztelés előtt érdemes biztosítani, hogy az adatbázis tesztadatokkal rendelkezzen.

Ha a tesztek során egy-egy hiba felmerül, azokat a konzolban részletes üzenetek formájában ellenőrizhetjük, ami segít a problémák gyors azonosításában.