Diák felhő

Fejlesztői dokumentáció

2023

Kovács Dorina Angelika

Kovács Dorina Angelika
Külkereskedelmi Technikum

2023. 03. 29.

Cél

Egy tanuló számára nyilvántartás kialakítása, mely biztosíja számára aznap kapott jegyeinek eltárolását, az órákon készített képek feltöltését, teendőinek felvételét.

A backend-et PHP alapú Laravel keretrendszer biztosítja, az adatbázist pedig MariaDb szerver.

Felhasznált technológiák

Backend

- Laravel API (PHP keretrendszer)
- laravel-sanctum (Autentikációs csomag)
- Mariadb server (Adatbázis kiszolgáló)

Készítéshez használt programok

- dia
- Visual studio code 1.76.2
- insomnia 2023.1.0
- phpmyadmin

Frontend

Kódolási konvenció

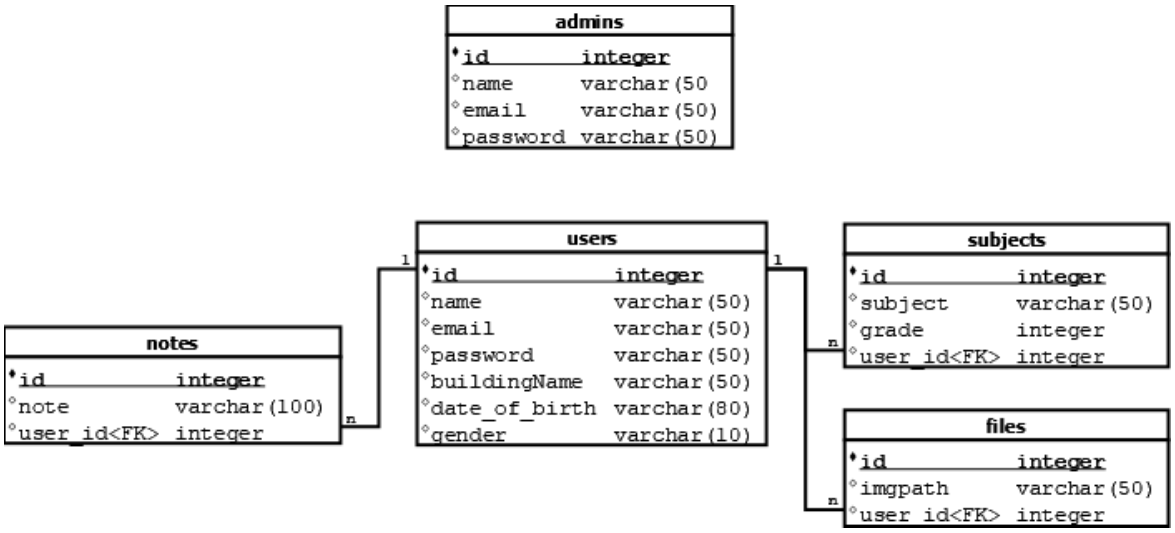
Git verziókezelővel használtam

Alapkönyvtárak

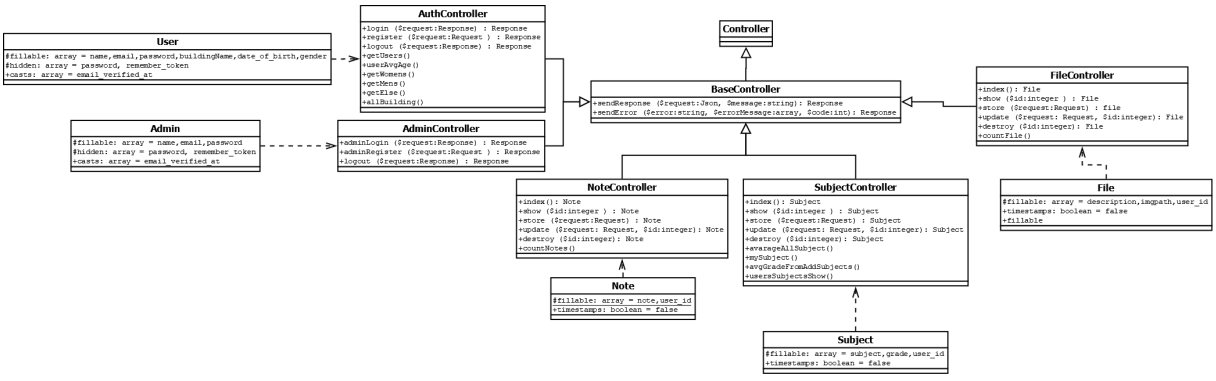
- **database**
- **db_terv**
- **Projekt_backend**
- **Projekt_docs**
- **insomnia**

Felülettervek

Adatmodell



API UML terv



Végpont

Metódus	Végpont	Azonosítás	Leírás
POST	/login	nem	Felhasználó bejelentkezése
POST	/register	nem	Felhasználó regisztrációja
POST	/logout	igen	Felhasználó kijelentkezés
POST	/adminReg	nem	Admin felhasználó regisztrációja
POST	/adminLog	nem	Admin felhasználó bejelentkezés
GET	/user	nem	Tanulók lekérdezése
GET	/sum	nem	Tanulók megszámlálása
GET	/age	nem	Tanulók átlag életkorának lekérdezése
GET	/womens	nem	Női felhasználók megszámlálása
GET	/mens	nem	Férfi felhasználók megszámlálása
GET	/else	nem	Egyéb felhasználók megszámlálása
GET	/allBuilding	nem	Regisztrált felhasználók megadott Intézményeinek lekérdezése
GET	/subject	igen	Tanuló tantárgyának megjelenítése
POST	/subjects	igen	Tanuló tantárgy/jegy felvétel
PUT	/updateSubject /{id}	igen	Tanuló tantárgy/jegy módosítása
DELETE	/ deleteSubject/{id}	igen	Tanuló tantárgy/jegy törlése
GET	/argAll	igen	Tanuló felvett összes jegyeinek átlagának lekérdezése
GET	/mySubject	igen	Tanuló felvett tantárgyai lekérdezése
GET	/grade	igen	Tanuló tantárgyakra bontott átlag lekérdezés
GET	/usersSubjects	nem	felvett tantárgyak összesítve való lekérdezése
GET	/note	igen	Tanuló teendőinek megjelenítése
POST	/notes	igen	Tanuló teendőinek felvétele
DELETE	/deleteNotes/{id}	igen	Tanuló teendőinek törlésé
GET	/countNote	igen	Tanuló teendőinek megszámlálás lekérdezése
GET	/image	igen	Tanuló feltöltött képeinek megjelenítése
POST	/images	igen	Tanuló kép felvétele
DELETE	/deleteImages /{id}	igen	Tanuló kép törlése
GET	/countfile	igen	Tanuló feltöltött képeinek mennyi

Általános működés:

A program REST API http kérést fogad, amely a műveletekhez szükséges adatokat tartalmaz.
Egyes műveletek végpontjai védettek, a használatukhoz authentikáció szükséges. Ilyen például a kijelentkezés, tantárgy/kép/teendő felvétele, azoknak a módosítása és törlése egyaránt.
A felhasználó regisztrációja, bejelentkezése, a felhasználók adatainak lekérdezése admin számára publikusak, authentikáció nem szükséges hozzá.
Az adatok Json formátumban fogadja és dolgozza fel azt. Minden adatkezelő csoportnak külön kontrollere van (user, admin, subject, file, note). Itt történik az adatok feldolgozása.
A kontrollerek modellekkel állnak kapcsolatban, melyek az adatok kezeléséért felelősek (user, admin, subject, file, note). Az adatbázisból az adatok kiolvasásáért, kiírásáért felel az összes modell. A resources a modellek összes attribútum tömbjét adja vissza, de ezt osztály egyéni megvalósítása felülírhatja.
A modellek adatbázis táblákkal vannak kapcsolatban, mely az adatok tárolásáért felelnek. A táblák adatait a modellek kezelik

Osztályok

BaseController

Feladata a végzett műveletek válaszainak küldése. Sikeres művelet esetén a sikeres üzenetet ad vissza, ellenkező esetben sikertelen művelet esetén hibaüzenetet küld.

Metódusok:

sendResponse()

Bemenő paraméterek: $result, $message (Egyedi üzenet a felhasználónak).

Kimenő adatok: Response(Sikeres művelet esetén adat).

sendError()

Bemenő paraméterek: $error(Php által legenerált hibaüzenet a felhasználó felé), $errorMessage(Egyedi hibaüzenet) , $code(Üzentben megjelentendő http kód).

Kimenő adatok: Response(Sikeres művelet esetén hibaüzenetet és a hiba kódját adja vissza).

AuthController

Az AuthController feladata az új felhasználók felvétele, felhasználók autentikációja, felhasználók kijelentkeztetése, felhasználók adatainak lekérdezése admin számára.

Metódusok:

login()

A felhasználó azonosítása email cím és jelszó alapján. Sikeres autentikáció esetén generál egy tokent a felhasználó számára es eltárolja a personal_access_tokens adatbázis táblában, majd megkapja a sendResponse() metódus a saját üzenettel együtt.

Bejövő paraméterek: $request(a bejelentkezéshez szükséges adatok kérése : email, password)

kimenő adatok: token , name, saját üzenet.

register()

Feladata az új felhasználók felvétele. Az adatok sikeres validációja, majd érvényesítése után, felveszi a users adatbázis tábla mezőibe.

Bejövő paraméterek: $request(a regisztrációhoz szükséges adatok kérése : name, emial, password, confrim_password, buildingName, gender, date_of_birth).

kimenő adatok: name, saját üzenet.

logout()

Feladata a felhasználó kijelentkeztetése és token törlése a personal_access_tokens adabzázis táblából.

Bejövő paraméterek: $request(felhasználó token kérése).

kimenő adat: saját üzenet.

getUsers()

Az adatbázis táblában szereplő felhasználók listáját adja vissza.

Bejövő paramétere: Nincs

kimenő adat: User (user modell mely tartalmazza a tanulók adatbázisból lekérdezett adatait)

countUsers()

A users adatbázis táblában eltárolt felhasználók megszámlálása.

Bejövő paramétere: Nincs

kimenő adat: Adatbázisból, users táblából lekérdezett felhasználók megszámlált eredménye count medódus használatával.

userAvgAge()

Admin oldalhoz való felhasználás érdekében a felhasználók átlag korának lekérdezése.

Bejövő paramétere: Nincs

kimenő adat: Adatbázisból, users táblából lekérdezett (metódus használat: ROUND(AVG(YEAR(CURDATE())-Year())) felhasználói átlag életkor.

getWomens()

Női felhasználók lekérdezése Admin oldalhoz való felhasználás érdekében

Bejövő paramétere: Nincs

kimenő adat: Adatbázisból, users táblából lekérdezett nők ki listázása majd annak megszámlálása id alapján count metódussal.

getMans()

Férfi felhasználók lekérdezése Admin oldalhoz való felhasználás érdekében

Bejövő paramétere: Nincs

kimenő adat: Adatbázisból, users táblából lekérdezett nők ki listázása majd annak megszámlálása id alapján count metódussal.

getElse()

Egyéb felhasználók lekérdezése Admin oldalhoz való felhasználás érdekében

Bejövő paramétere: Nincs

kimenő adat: Adatbázisból, users táblából lekérdezett nők ki listázása majd annak megszámlálása id alapján count metódussal.

allBuilding()

Feladata: Felhasználók által megadott intézmények lekérdezése.

Bejövő paramétere: Nincs

kimenő adat: Adatbázisból, users táblából buildingName-re szűrt majd groupBy segítségével csoportosított adatok megjelenítése.

AdminController

Az AdminController feladata az új felhasználók felvétele, felhasználók autentikációja, felhasználók kijelentkeztetése.

adminLogin()

A felhasználó azonosítása email cím és jelszó alapján. Sikeres autentikáció esetén generál egy tokent a felhasználó számára es eltárolja a personal_access_tokens adatbázis táblában, majd megkapja a sendResponse() metódus a saját üzenettel együtt.

Bejövő paraméterek: $request(a bejelentkezéshez szükséges adatok kérése : email, password)

kimenő adatok: token , name, saját üzenet.

adminRegister()

Feladata az új felhasználók felvétele. Az adatok sikeres validációja, majd érvényesítése után, felveszi az admins adatbázis tábla mezőibe.

Bejövő paraméterek: $request(a regisztrációhoz szükséges adatok kérése : name, emial, password, confrim_password).

kimenő adatok: name, saját üzenet.

logout()

Feladata a felhasználó kijelentkeztetése és token törlése a personal_access_tokens adabzázis táblából.

Bejövő paraméterek: $request(felhasználó token kérése).

kimenő adat: saját üzenet.

SubjectController

A SubjectController feladata a tantárgyak/jegyek felvétele/megjelenítése/szerkeztése/törlése, a felvett adatok szűrése

index()

subjects adatbázis táblában eltárolt adatok megjelenítése a felhasználó számára.

Bejövő paramétere:Nincs

kimenő adat: Subject -> feltételhez kötötten (azonosítást követően a felhasználó csak saját adatait kapja vissza, $subject és $grade).

s**tore()**

Új tantárgy és jegy eltárolása az adatbázisban.

Bejövő paramétere: $request ->tantárgyak és jegyek adatait tartalmazó kérés

kimenő adat: Subject (felvett adatokat tartalmazó modell)

show()

Egy kiválasztott tantárgy/jegy adatait adja vissza.

Bejövő paramétere: $id ->kiválasztott tantárgy/jegy azonosítója

kimenő adat: Subject (kiválasztott tantárgy/jegy adatait tartalmazó modell)

update()

Egy kiválasztott tantárgy/jegy adatait adja vissza.

Bejövő paramétere: $id ->kiválasztott tantárgy/jegy azonosítója

kimenő adat: Subject (kiválasztott tantárgy/jegy adatait tartalmazó modell)

destroy()

Egy kiválasztott tantárgy/jegy törlése az adatbázisból.

Bejövő paramétere: $id ->kiválasztott tantárgy/jegy azonosítója

kimenő adat: Subject (kiválasztott tantárgy/jegy adatait tartalmazó modell)

avarageAllSubject()

A felhasználó által eddig felvett tantárgy jegyeinek összesített átlag lekérdezése.

Bejövő paramétere: Nincs

kimenő adat: feltételhez kötötten (azonosítást követően a felhasználó csak saját adatait kapja vissza, subjects adatbázis tábla user_id-jára keresve a jegyek darabszám meghatározása, sum metódus használatával való összesítése -> $arg változóban átlagszámítás alkalmazása a megkapott adatokból, majd annak visszatérése).

mySubject()

A felhasználó által eddig felvett tantárgyai.

Bejövő paramétere: Nincs

kimenő adat: feltételhez kötötten (azonosítást követően a felhasználó csak saját adatait kapja vissza, subjects adatbázis tábla user_id-jára keres, tantárgyra szűr, majd a kapott adatokat groupBy metódussal rendezi -> $groupSub-ban eltárolt szűrt adatok megjelenítése).

avgGradeFromAddSubjects()

A felhasználó által eddig felvett jegyeinek átlag lekérdezése tantárgyaként.

Bejövő paramétere: Nincs

kimenő adat: feltételhez kötötten (azonosítást követően a felhasználó csak saját adatait kapja vissza, subjects adatbázis tábla user_id-jára keres, tantárgyra/user_id-ra szűr ->átlag meghatározása (sum(grade)/count(subject) as atlag, count(subject) as jegydb') -> majd a kapott adatokat groupBy metódussal rendezi -> $avg-ben eltárolt szűrt adatok megjelenítése).

FileController

A FileController feladata a képek/leírások felvétele/megjelenítése/törlése, a felvett adatok szűrése

index()

Files adatbázis táblában eltárolt adatok megjelenítése a felhasználó számára.

Bejövő paramétere: Nincs

kimenő adat: feltételhez kötötten (azonosítást követően a felhasználó csak saját adatait kapja vissza, $file) .

store()

Új kép és leírás eltárolása az adatbázisban.

Bejövő paramétere: $request -> kép és leírás adatait tartalmazó kérés

kimenő adat: File (felvett adatokat tartalmazó modell)

destroy()

Egy kiválasztott kép és leírás törlése az adatbázisból / storage mappából.

Bejövő paramétere: $id ->kiválasztott kép és leírás azonosítója

kimenő adat: feltételhez kötötten-> File (kiválasztott kép és leírás adatait tartalmazó modell)

countFile()

A felhasználó által felvett képek/leírások megszámlálása.

Bejövő paramétere: Nincs

kimenő adat: feltételhez kötötten (azonosítást követően a felhasználó csak saját adatait kapja vissza, a files adatbázis táblából user_id-ra keres -> majd count metódussal id megszámlálás -> $count váltózóban eltárolt adattal visszatérés).

NoteController

A NoteController feladata a teendők felvétele/megjelenítése/törlése, a felvett adatok szűrése

index()

Notes adatbázis táblában eltárolt adatok megjelenítése a felhasználó számára.

Bejövő paramétere: Nincs

kimenő adat: feltételhez kötötten (azonosítást követően a felhasználó csak saját adatait kapja vissza, $note).

store()

Új teendő eltárolása az adatbázisban.

Bejövő paramétere: $request -> a teendő adatait tartalmazó kérés

kimenő adat: Note (felvett adatokat tartalmazó modell)

destroy()

Egy kiválasztott teendő törlése az adatbázisból.

Bejövő paramétere: $id ->kiválasztott teendő azonosítója

kimenő adat: Note (kiválasztott teendő adatait tartalmazó modell)

countNotes()

A felhasználó által felvett teendők megszámlálása.

Bejövő paramétere: Nincs

kimenő adat: feltételhez kötötten (azonosítást követően a felhasználó csak saját adatait kapja vissza, a notes adatbázis táblából user_id-ra keres -> majd count metódussal id megszámlálás -> $count váltózóban eltárolt adattal visszatérés)

Models

User

A felhasználó regisztrációját követően adatait, azonosítóját (tokent), melynek legenerálása után felhasználóhoz kötődik, az adatbázis megfelelő táblájába írja az adatokat, és szolgáltatja egyaránt.

HasMany függvény, subject/file/note modell közti kapcsolatot biztosít. A getId() metódus egy felhasználó azonosítóját adja vissza, melyet a controllerben feltételként van alkalmazva.

A modell a users adat táblával áll kapcsolatban, onnan érkeznek az adatok és oda íródnak vissza.

A modellt a Laravel generálja, Sanctum használatával authentikációs kiegészítő csomagon keresztül történik.

Mezők:

$fillable elemei: buildingName, name, email, date_of_birth, gender, password.

$timestamp : false -> modell nem használja alapértelmezetten az idő bélyeget.

$hidden: remember_token, password elemek rejtettek.

Admin

Az admin regisztrációját követően adatait, azonosítóját (tokent), melynek legenerálása után felhasználóhoz kötődik, az adatbázis megfelelő táblájába írja az adatokat, és szolgáltatja egyaránt.

A modellt a Laravel generálja, Sanctum használatával authentikációs kiegészítő csomagon keresztül történik.

Mezők:

$fillable elemei: name, email, password.

$timestamp : false -> modell nem használja alapértelmezetten az idő bélyeget.

$hidden: remember_token, password elemek rejtettek.

Subject

A felhasználó által felvett adatokat kezeli, SubjectControllerben megadott metódusok használatával adja vissza az adatokat a subjects adatbázis táblából , emellett a controller felől közvetített adatokat írja ki.

A belongsTo() függvény használata teremt kapcsolatot a user model közt. A subjects táblában felvett user_id a users adatbázis táblában szereplő id oszlopának felel meg.

Mezők:

$fillable elemei: subject, grade, user_id.

$timestamp : false -> modell nem használja alapértelmezetten az idő bélyeget.

File

A felhasználó által felvett adatokat kezeli, FileControllerben megadott metódusok használatával adja vissza az adatokat a files adatbázis táblából, emellett a controller felől közvetített adatokat írja ki.

A belongsTo() függvény használata teremt kapcsolatot a user model közt. A subjects táblában felvett user_id a users adatbázis táblában szereplő id oszlopának felel meg.

Mezők:

$fillable elemei: description, imgpath, user_id.

$timestamp : false -> modell nem használja alapértelmezetten az idő bélyeget.

Note

A felhasználó által felvett adatokat kezeli, NoteControllerben megadott metódusok használatával adja vissza az adatokat a notes adatbázis táblából, emellett a controller felől közvetített adatokat írja ki.

A belongsTo() függvény használata teremt kapcsolatot a user model közt. A subjects táblában felvett user_id a users adatbázis táblában szereplő id oszlopának felel meg.

Mezők:

$fillable elemei: note, user_id.

$timestamp : false -> modell nem használja alapértelmezetten az idő bélyeget.

Resources

User

Ez a metódus alapértelmezés szerint az User modell összes attribútum tömbjét adja vissza, de az osztály egyéni megvalósítása felülírja, hogy csak az erőforráshoz szükséges konkrét attribútumokat adja vissza. Ilyen az id, buildingName, name, email, date_of_birth, gender

Admin

Ez a metódus alapértelmezés szerint az Admin modell összes attribútum tömbjét adja vissza, de az osztály egyéni megvalósítása felülírja, hogy csak az erőforráshoz szükséges konkrét attribútumokat adja vissza. Ilyen az id, name, email

Subject

Ez a metódus alapértelmezés szerint az Admin modell összes attribútum tömbjét adja vissza, de az osztály egyéni megvalósítása felülírja, hogy csak az erőforráshoz szükséges konkrét attribútumokat adja vissza. Ilyen az id, subject, grade, user_id

File

Ez a metódus alapértelmezés szerint az File modell összes attribútum tömbjét adja vissza, de az osztály egyéni megvalósítása felülírja, hogy csak az erőforráshoz szükséges konkrét attribútumokat adja vissza. Ilyen az id, description, imgpath, user_id

Note

Ez a metódus alapértelmezés szerint az Note modell összes attribútum tömbjét adja vissza, de az osztály egyéni megvalósítása felülírja, hogy csak az erőforráshoz szükséges konkrét attribútumokat adja vissza. Ilyen az id, note, user_id

Fejlesztői környezet

Pl.:

- XAMPP
  - Adatbázis kezelése a mysql nevű klienssel.
  - PHP
- Visual Studio Code
- Laravel

Tesztelés

A backend tesztelését insomnián keresztül hajtottam végre. Metódusok tesztelése során az elvárt értékeket / üzeneteket adta vissza sikeresen.

Fejlesztési lehetőségek

A jövőben mobil applikáció készítése a projekthez.

Naptár fejlesztése, hogy a felhasználók beírhassák eseményeiket és azokról értesítést kapjanak a megadott napon.

Adatbázisba fellehetne venni egy új táblát (pl: type_users), amely a regisztráció során kell kiválasztani, hogy egyetemista/főiskolás, avagy általános/gimnazista a tanuló. Ez által külön lehetne szedni a felső oktatásban / általános oktatásban tanulók részére a projektet, és még jobban ’rájuk szabni’.

Összegzés

Az év elején eltervezett összes funkció megvalósítását próbáltam megoldani Backendben, ami sikerült is.