# Általános működés:

A REST API http kéréseket fogad, melyek tartalmazzák a műveletekhez szükséges megfelelő adatokat. A kényes műveletek végpontjai védettek, autentikáció szükséges a használatukhoz.

Ilyenek a kijelentkezés, az új diák felvétele, diák adatainak módosítása, diák törlése, új csoport felvétele, csoport módosítása, csoport törlése. A regisztráció, a bejelentkezés, a diákok és csoportok lekérdezésének végpontjai publikusak, ezek használatához autentikáció nem szükséges.

Az adatokat Json formátumban fogadja és feldolgozza. A vezérlést kontrollerek valósítják meg, minden adatkezelési csoportnak ( autentikáció, diák, csoport ) külön kontrollere van, itt történik az adatfeldolgozás.

A kontrollerek modellekkel vannak kapcsolatban, amelyek az adatkezelésért felelősek. Minden adatkezelési csoportnak ( felhasználó, diák, csoport ) külön modellje van, itt történik az adatok adatbázisból kiolvasása, illetve a az adatok kiírása adatbázisba.

A modellek adatbázis táblákkal vannak kapcsolatban, melyek az adatok tárolásáért felelősek.

Az adatbázis táblák adatait a modellek kezelik.

## Env. fájlhoz szükséges adatok

**Adatbázis beállítások:**DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=ProgramvalasztoDB

DB\_USERNAME=root

DB\_PASSWORD=

**E-mail küldéshez a beállítások:**MAIL\_MAILER=smtp

MAIL\_SCHEME=smtp

MAIL\_HOST=smtp.gmail.com

MAIL\_PORT\_DEFAULT=465

MAIL\_PORT\_ALT=587

MAIL\_USERNAME=programvalaszto@gmail.com

MAIL\_PASSWORD=pniuzbnningducdw

MAIL\_FROM\_ADDRESS="programvalaszto@gmail.com"

MAIL\_FROM\_NAME="${APP\_NAME}"

## Útvonalak

Ezek a fájlok a Laravel alkalmazás API és webes útvonalaiért felelnek meg. Az alábbi kódbestand a api.php fájl tartalmát képviseli, amely a RESTful API végpontokat definiálja.

Útvonalak Részletes Elemzése

**Regisztráció és Bejelentkezés:**

Route::post("/register", [ UserController::class, "register" ]);

Cél: Új felhasználó regisztrálása.

Route::post("/login", [UserController::class, "login"]);

Cél: Felhasználó bejelentkezése.

**Email ellenőrzés:**

Route::get('email/verify/{id}/{hash}', [VerificationController::class, 'verify'])->name('verification.verify');

Cél: Email címek ellenőrzése a regisztrációs folyamat során.

**Hitelesített Felhasználói útvonalak:**

Ezek az útvonalak csak hitelesített felhasználók (akik beléptek) számára elérhetők a auth:sanctum middleware-en keresztül.

**Kilépés:**

Route::post("/logout", [ UserController::class, "logOut" ]);

**Keresések:**

Események, tagok és felhasználók keresése:

searchEvent, searchTags, searchUser funkciók.

**Felhasználók kezelése:**

Felhasználók listázása, módosítása, törlése:

getUsers, modifyUser, destroyUser funkciók.

**Események kezelése:**

**Események létrehozása, módosítása, törlése:**

newEvent, modifyEvent, destroyEvent.

**Címkék kezelése:**

Címkék létrehozása, módosítása, törlése:

newTag, modifyTag, destroyTag.

**Csoportok és Események Tag-ekhez kapcsolása:**

Eseményekhez és címkékhez kapcsolódó funkciók (attachTags, detachTag, getEventsByTag, getTagsByEvents).

**Feliratkozások kezelése:**

Feliratkozások, mint például eseményekhez való feliratkozás, módosítás és leiratkozás.

**Felhasználó és Címkék kapcsolat:**

subUserTag és unsubUserTag funkciók, amelyek lehetővé teszik, hogy a felhasználók címkéket kapjanak vagy azokról leiratkozzanak

## Migration

Az adatbázis-séma definíciói az database/migrations könyvtárban található migrációs fájlokban találhatók.

Műveletek:

up() Metódus:

A metódus célja a táblák létrehozása.

down() Metódus:

A metódus célja a migráció visszaállítása.

### create\_cache\_table

Fájl Neve: 0001\_01\_01\_000001\_create\_cache\_table.php

Feladat:

A migrációs fájl célja, hogy létrehozza a cache és cache\_locks adatbázistáblákat, amelyek a Laravel alkalmazás cache (gyorsítótár) rendszerének működéséhez szükségesek. Ezek a táblák tárolják a gyorsítótárba mentett adatokat, valamint az adatlopás és a konkurens írások elkerülése érdekében a zárakat.

Definiált Adatbázis Táblák:

cache Tábla:

key (string, primary key): A gyorsítótárban tárolt adat kulcsa. Ez biztosítja, hogy minden tárolt adat egyedi legyen.

value (mediumText): A tárolt adat értéke, amely bármilyen típusú lehet, ezért mediumText típust kapott, lehetővé téve a nagyobb adatok tárolását.

expiration (integer): Ez a mező tárolja, hogy meddig érvényes a gyorsítótárazott adat, azaz mikor jár le a cache.

cache\_locks Tábla:

key (string, primary key): A zár kulcsa, ami biztosítja, hogy minden zárolt erőforrás egyedi.

owner (string): A zár sajátos tulajdonosa, amely segíthet az azonosításban és a zárolás kezelésében, különösen, ha több felhasználó vagy folyamat próbál hozzáférni az azonos adat erőforráshoz.

expiration (integer): A zár időtartama, amely jelzi, hogy meddig érvényes a lock. Ezt az értéket súlyosbítja a cache-elésnél alkalmazott időkorlát, segítve elkerülni a halmozódást vagy a holt időszakokat.

### create\_personal\_access\_tokens\_table

Fájl Neve: 2025\_01\_09\_122629\_create\_personal\_access\_tokens\_table.php

Feladat:

Ez a migrációs fájl a personal\_access\_tokens nevű adatbázistábla létrehozásáért felelős, amely a felhasználók számára személyes hozzáférési tokenek kezelésére szolgál. Ez a funkció alapvető eleme az API hitelesítésének, lehetővé téve a felhasználók számára, hogy biztonságosan hozzáférjenek az alkalmazás erőforrásaihoz.

Definiált Adatbázis Tábla:

personal\_access\_tokens Tábla:

id (bigint, primary key): Automatikusan generált egyedi azonosító minden egyes token számára.

tokenable\_id (bigint): A felhasználó azonosítója (vagy bármilyen más tokenhez kapcsolódó entitás), amelyhez a token tartozik. Ez a mező a polymorphic kapcsolat egyik része.

tokenable\_type (string): A tokenhez tartozó entitás típusa (például User vagy Admin). Ez a mező is része a polymorphic kapcsolatnak.

name (string): A személyes token neve, amely lehetővé teszi a felhasználóknak, hogy azonosítsák a token funkcióját vagy célját.

token (string, 64 karakter hosszú, unique): A token értéke. Ez az érték egyedi, és használható az API hitelesítéséhez.

abilities (text, nullable): A tokenhez kapcsolódó képességeket leíró mező, amely opcionális. Ez lehetővé teszi a finomhangolt jogosultságok megadását a tokenhez.

last\_used\_at (timestamp, nullable): Az utolsó időpont, amikor a tokent használták. Ez segít a token érvényességi idejének és aktivitásának nyomon követésében.

expires\_at (timestamp, nullable): A token lejárati idejét tárolja. Ez lehetővé teszi a tokenek automatikus érvénytelensé válását egy előre meghatározott időpont után.

created\_at (timestamp): A token létrehozásának időpontja.

updated\_at (timestamp): A token utolsó frissítésének időpontja.

### create\_users\_table

Fájl Neve: 2025\_01\_09\_151734\_create\_users\_table.php

Feladat:

Ez a migrációs fájl a users, password\_reset\_tokens, és sessions adatbázistáblák létrehozásáért felelős, amelyek a Laravel alkalmazás felhasználói kezelésének, jelszó-visszaállításának, és munkamenet-kezelésének alapját képezik.

Definiált Adatbázis Táblák:

users Tábla:

id (bigint, primary key): Automatikusan generált egyedi azonosító minden felhasználó számára.

name (string): A felhasználó neve, amely lehetővé teszi a felhasználó azonosítását.

email (string): A felhasználó e-mail címe, amely szintén egyedi.

email\_verified\_at (timestamp, nullable): Az e-mail cím érvényesítési dátumának tárolására szolgáló mező.

password (string): A felhasználó jelszava, amelyet hash-elt formában kell tárolni a biztonság érdekében.

rememberToken (string, nullable): A "memorizálás" funkcióhoz szükséges token, amely lehetővé teszi a felhasználó számára, hogy bejelentkezve maradjon.

created\_at (timestamp): A felhasználó rekord létrehozásának időpontját tárolja.

updated\_at (timestamp): A felhasználói rekord utolsó frissítésének időpontját tárolja.

password\_reset\_tokens Tábla:

email (string, primary key): A felhasználó e-mail címe, amelyhez a jelszó-visszaállítási token tartozik. Ez a mező egyedi, és biztosítja az azonosítást.

token (string): A jelszó-visszaállításához szükséges token, amelyet a felhasználónak meg kell adnia.

created\_at (timestamp, nullable): A token létrehozásának időpontját tároló mező, amely nullable, tehát nem kötelező.

sessions Tábla:

id (string, primary key): Minden munkamenet egyedi azonosítója, amely egyedi azonosítót biztosít a session kezeléséhez.

user\_id (foreignId, nullable, index): A felhasználó azonosítója, akihez a munkamenet kapcsolódik. Nullable, azaz nem minden session szükségszerűen kapcsolódik felhasználóhoz.

ip\_address (string, 45, nullable): A felhasználó IP-címének tárolására szolgál., azaz nem kötelező.

user\_agent (text, nullable): A felhasználó eszközének vagy böngészőjének azonosítása érdekében gyűjtött adat, amely nullable.

payload (longText): A munkamenethez kapcsolódó adatokat tartalmazó mező.

last\_activity (integer, index): Az utolsó aktivitás időpontja, amely lehetővé teszi a munkamenetek időkorlátjának kezelését.

### create\_events\_table

Fájl Neve: 2025\_01\_09\_151748\_create\_events\_table.php

Feladat:

Ez a migrációs fájl az events nevű adatbázistábla létrehozásáért felelős, amely események adatait tárolja. A táblázat funkciója, hogy egyszerűsíti az események kezelését és lekérdezését az alkalmazásban, lehetővé téve különböző események információinak tárolását.

Definiált Adatbázis Tábla:

events Tábla:

id (bigint, primary key): Automatikusan generált egyedi azonosító minden esemény számára.

name (string): Az esemény neve, amely azonosítja az eseményt.

startDate (date, nullable): Az esemény kezdésének dátuma.

endDate (date, nullable): Az esemény befejezésének dátuma.

startTime (time, nullable): Az esemény kezdési időpontja.

endTime (time, nullable): Az esemény befejezési időpontja.

description (text, nullable): Az esemény leírása, amely tartalmazza a részletes információkat.

image (string, nullable): Az eseményhez kapcsolódó kép URL-je.

locationName (string, nullable): Az esemény helyszínének neve.

locationcountry (string, nullable): Az esemény helyszínének ország neve.

address (string, nullable): Az esemény pontos címe, ami opcionális.

state (integer, nullable): Az esemény állapotát jelölő mező, amely megadhatja az esemény státuszát.

weblink (string, nullable): Az eseményhez tartozó weboldal hivatkozása, amely opcionális.

latitude (decimal, nullable): Az esemény földrajzi szélességi koordinátája, amely a helyszín pontos lokalizálásához szükséges.

longitude (decimal, nullable): Az esemény földrajzi hosszúsági koordinátája, amely szintén a helyszín azonosításához tartozik.

gpx (string, nullable): A GPX fájl hivatkozása, amennyiben az eseményhez kapcsolódik.

subscribed (unsignedInteger, default: 0): Az eseményre feliratkozott felhasználók számát tároló mező, amely alapértelmezetten 0-ra van állítva.

created\_at (timestamp): Az esemény rekord létrehozásának időpontja.

updated\_at (timestamp): Az esemény rekord utolsó frissítésének időpontja.

### create\_tags\_table

Fájl Neve: 2025\_01\_09\_151810\_create\_tags\_table.php

Feladat:

Ez a migrációs fájl a tags nevű adatbázistábla létrehozásáért felelős, amely a címkék (tags) kezelését szolgálja az alkalmazásban. A címkék csoportosítást és kategorizálást tesznek lehetővé az események, tartalmak vagy egyéb objektumok számára.

Definiált Adatbázis Tábla:

tags Tábla:

id (bigint, primary key): Automatikusan generált egyedi azonosító minden címkéhez.

name (string): A címke neve, amely alapján az azonosítást végzik.

group (string): A címke csoportja, amely lehetővé teszi a címkék hierarchikus szerkezetbe való rendezését, vagy csoportosítását.

created\_at (timestamp): A címke rekord létrehozásának időpontját tárolja.

updated\_at (timestamp): A címke rekord utolsó frissítésének időpontját tárolja.

### create\_subscriptions\_table

Fájl Neve: 2025\_01\_09\_152058\_create\_subscriptions\_table.php

Feladat:

Ez a migrációs fájl a subscriptions nevű adatbázistábla létrehozásáért felelős, amely a felhasználók eseményekre való feliratkozásait tárolja. A táblázat lehetővé teszi a kapcsolat kialakítását a felhasználók és az események között, valamint kiegészítő információk gyűjtését a feliratkozásokkal kapcsolatosan.

Definiált Adatbázis Tábla:

subscriptions Tábla:

id (bigint, primary key): Automatikusan generált egyedi azonosító minden feliratkozáshoz.

users\_id (foreignId): A felhasználó azonosítója, aki feliratkozott az eseményre. Külső kulcs a users táblára, amely a felhasználók adatait tartalmazza. A constrained('users') megköveteli, hogy ez a mező a users táblában lévő id mező értékével egyezzen meg. Az onDelete('cascade') biztosítja, hogy ha egy felhasználót törölnek, akkor a kapcsolódó feliratkozások is automatikusan törlődnek.

events\_id (foreignId): Az esemény azonosítója, amelyre a felhasználó feliratkozott. Külső kulcs az events táblára. A constrained('events') garantálja, hogy az értéke a events táblában lévő id mező értékével egyezzen meg. Az onDelete('cascade') biztosítja, hogy ha egy eseményt törölnek, akkor a kapcsolódó feliratkozások is automatikusan törlődnek.

comment (text, nullable): Opcionális mező, amely lehetővé teszi a felhasználó számára, hogy további megjegyzést fűzzön a feliratkozásához.

sub\_date (date): A feliratkozás/ részvétel dátuma; ez a mező a feliratkozás időpontját rögzíti.

created\_at (timestamp): A rekord létrehozásának időpontját tárolja.

updated\_at (timestamp): A rekord utolsó frissítésének időpontját tárolja.

### create\_favorites\_table

Fájl Neve: 2025\_01\_09\_152141\_create\_favorites\_table.php

(Nincs alkalmazva)

Feladat:

Ez a migrációs fájl a favorites nevű adatbázistábla létrehozásáért felelős, amely a felhasználók kedvenc eseményeit tárolja. A táblázat célja, hogy lehetővé tegye a felhasználók számára, hogy megjelöljék azokat az eseményeket, amelyeket különösen fontosnak tartanak, és így könnyen hozzáférhessenek ezekhez később.

Definiált Adatbázis Tábla:

favorites Tábla:

users\_id (foreignId): A felhasználó azonosítója, aki a kedvenc eseményeket hozzáadja. Külső kulcs a users táblára, amely a felhasználók adatait tárolja.

events\_id (foreignId): Az esemény azonosítója, amely a felhasználó kedvence. Külső kulcs az events táblára, amely az esemény részleteit tartalmazza.

created\_at (timestamp): A rekord létrehozásának időpontját tárolja.

updated\_at (timestamp): A rekord utolsó frissítésének időpontját tárolja.

### create\_comments\_table

Fájl Neve: 2025\_01\_09\_152155\_create\_comments\_table.php

(Nincs alkalmazva)

Feladat:

Ez a migrációs fájl a comments nevű adatbázistábla létrehozásáért felelős, amely a felhasználók által az eseményekre írt megjegyzéseket és értékeléseket tárolja. A táblázat célja, hogy lehetővé tegye a felhasználók számára, hogy visszajelzést adjanak az eseményekről, ezzel javítva az interakciót és a közösségi élményt.

Definiált Adatbázis Tábla:

comments Tábla:

users\_id (foreignId): A felhasználó azonosítója, aki a megjegyzést írta. Külső kulcs, amely a users táblát hivatkozza.

events\_id (foreignId): Az esemény azonosítója, amelyre a megjegyzés vonatkozik. Külső kulcs, amely az events táblát hivatkozza.

comment (text, nullable): A felhasználó által írt megjegyzés szövege.

rating (integer): Az eseményre adott értékelés.

created\_at (timestamp): A rekord létrehozásának időpontját tárolja.

updated\_at (timestamp): A rekord utolsó frissítésének időpontját tárolja.

### create\_eventtags\_table

Fájl Neve: 2025\_01\_09\_152210\_create\_eventtags\_table.php

Feladat:

Ez a migrációs fájl az eventtags nevű adatbázistábla létrehozásáért felelős, amely az események és címkék közötti kapcsolatokat tárolja. Az eventtags tábla célja, hogy lehetővé tegye az események címkézését, így segítve a könnyebb keresést és kategorizálást az események között.

Definiált Adatbázis Tábla:

eventtags Tábla:

events\_id (foreignId): Az esemény azonosítója, amely a címkéhez kapcsolódik. Külső kulcs az events táblára, amely az események adatait tárolja. A constrained() metódussal létrehozott külső kulcs automatikusan hivatkozik az events tábla megfelelő mezőjére.

tags\_id (foreignId): A címke azonosítója, amely az eseményhez kapcsolódik. Külső kulcs a tags táblára, amely a címkék adatait tárolja. A constrained() metódus szintén biztosítja, hogy a hivatkozott adat érvényes legyen.

created\_at (timestamp): A rekord létrehozásának időpontját tárolja.

updated\_at (timestamp): A rekord utolsó frissítésének időpontját tárolja.

### admin\_to\_users\_table

Fájl Neve: 2025\_02\_19\_162330\_add\_admin\_to\_users\_table.php

Feladat:

Ez a migrációs fájl a users adatbázistáblához egy új mező, az admin, hozzáadásáért felelős. Ez a mező célja, hogy jelezze, egy adott felhasználó adminisztrátor-e vagy sem.

Definiált Adatbázis Tábla:

users Tábla:

admin (integer): Ez a mező egy egész számot tárol, amely a felhasználói jogosultságok szintjét jelzi. Az 1-es és 2-es érték jelenti azt, hogy a felhasználó adminisztrátor vagy szuper adminisztrátor, míg egy 0 érték a normál felhasználót jelenthet.

### create\_tag\_subscriptions\_table

Fájl Neve: 2025\_02\_23\_105012\_create\_tag\_subscriptions\_table.php

Feladat:

Ez a migrációs fájl a tag\_subscriptions nevű adatbázistábla létrehozásáért felelős, amely a felhasználók címke-feliratkozásait tárolja. Célja, hogy a felhasználók kedvenc címkéikhez való kapcsolódását nyomon kövesse, lehetővé téve számukra, hogy értesítést kapjanak az adott címkéhez kapcsolódó eseményekről.

Definiált Adatbázis Tábla:

tag\_subscriptions Tábla:

users\_id (foreignId): A felhasználó azonosítója, aki feliratkozott a címkére. Külső kulcs a users táblára, amely a felhasználói adatokat tárolja.

tags\_id (foreignId): Az a címke azonosítója, amelyre a felhasználó feliratkozott. Külső kulcs a tags táblára, amely a címkék adatait tárolja.

created\_at (timestamp): A feliratkozás időpontjának tárolására szolgál.

updated\_at (timestamp): A feliratkozás frissítésének időpontját tárolja.

### add\_deleted\_at\_to\_users\_table

Fájl Neve: 2025\_04\_06\_074012\_add\_deleted\_at\_to\_users\_table.php

Feladat:

Ez a migrációs fájl a users adatbázistáblához egy deleted\_at mező hozzáadására szolgál, amely lehetővé teszi a lágy törléseket (soft deletes) az alkalmazásban.

Definiált Adatbázis Tábla:

users Tábla:

deleted\_at (timestamp): Ez a mező tárolja a törlés időpontját, ha a felhasználót lágyan törölték. Ha az érték NULL, akkor a felhasználó aktív, míg ha van értéke, akkor ez azt jelzi, hogy a felhasználót törölték, de az adatainak megőrzése érdekében a rendszerben maradt.

## Seeder

### DatabaseSeeder

Feladata: A DatabaseSeeder osztály a Laravel adatbázis-seedelője, amely lehetővé teszi, hogy egyszerűen hozhassunk létre és tölthessünk fel alapvető adatokat az adatbázisban. Az osztályban meghatározott adatok alapvető felhasználói rekordokat tartalmaznak, amelyek a fejlesztés során referenciaként vagy tesztelésre használhatók.

Tulajdonságok és Metódusok:

public function run(): void

Leírás: Ez a metódus futtatja az adatok betöltéséhez szükséges kódot. A $table->insert([..]) hívások segítségével rekordokat adhatunk hozzá az adatbázishoz.

Implementáció:

Felhasználói rekordok beszúrása: Két rekordot helyez a users táblába:

Az admin felhasználót és a superadmin felhasználót hasonló attribútumokkal, de magasabb jogosultsági szinttel.

## Model

### User

Feladata: A felhasználói adatok kezelésére és autentikációjára szolgál. Ez az osztály tartalmazza a felhasználók tulajdonságait és kapcsolatait, lehetővé téve a Laravel rendszerében történő hitelesítést és adatkezelést.

Használat: Az User osztály a Laravel autentikációs rendszere révén lehetővé teszi a felhasználók létrehozását, frissítését, törlését, valamint a kapcsolódó műveleteket, mint például a jelszókezelést és a szerepkörök ellenőrzését.

Tulajdonságok és Metódusok:

Traits:

HasFactory: Lehetővé teszi a gyári metódusok használatát a felhasználói modell előállításához.

Notifiable: Lehetővé teszi, hogy a felhasználó értesítéseket kapjon (pl. e-mail, SMS).

HasApiTokens: Lehetővé teszi API tokenek létrehozását a felhasználók számára.

SoftDeletes: Lehetővé teszi a lágy törlés (soft delete) használatát, azaz a rekordok nem kerülnek azonnal törlésre, hanem csak „törölt” állapotba kerülnek.

protected $fillable:

Leírás: Definiálja azokat az attribútumokat, amelyeket tömegesen hozzáadhatunk (mass assignable), például a felhasználó neve, e-mail címe, jelszava és admin jogosultsága.

Tartalmazza:

'name'

'email'

'password'

'admin'

protected $hidden:

Leírás: Megadja azokat az attribútumokat, amelyeket nem szeretnénk megjeleníteni a JSON sorosítás során (pl. API válaszokban).

Tartalmazza:

'password'

'remember\_token'

protected function casts(): array:

Leírás: A modellek attribútumainak típushashálását definiálja.

Tartalmazza:

'email\_verified\_at' => 'datetime': Ez az attribútum dátumként van definiálva.

'password' => 'hashed': A jelszó automatikusan hashelt formában lesz tárolva.

public function subscriptions():

Leírás: A felhasználóhoz tartozó feliratkozásek (Subscription) lekérdezésére szolgál.

Kimeneti adatok: A hasMany relációt visszaadó lekérdezés, amely megadja, hogy a felhasználónak több feliratkozáse lehet.

public function tags(): BelongsToMany:

Leírás: A felhasználó és a címkék (Tag) közötti kapcsolatok kezelésére szolgál.

Kimeneti adatok: A belongsToMany relációt visszaadó lekérdezés a tag\_subscriptions pivot táblán keresztül, megadva a users\_id és tags\_id kulcsokat, valamint a withTimestamps() metódussal biztosítja a létrehozási és frissítési időbélyegzők mentését.

### Event

Feladata: Az események reprezentálására és kezelésére szolgáló modell. Az események lehetnek rendezvények, aktivitások, vagy bármilyen más típusú esemény, amelyhez felhasználók, megjegyzések, címkék és feliratkozások kapcsolódhatnak.

Használat: Az Event osztály segítségével az alkalmazások képesek kezelni az eseményeket, valamint a kapcsolódó műveleteket, mint például a résztvevők nyilvántartását, a hozzászólások gyűjtését és a rendezvények kategorizálását.

Tulajdonságok és Metódusok:

Traits:

HasFactory: Lehetővé teszi a gyári metódusok használatát az események egyszerű előállításához.

Kapcsolatok:

public function users()

Leírás: Visszaadja a felhasználókat, akik kapcsolódnak az eseményhez.

Kimeneti adatok: belongsToMany relációt visszaadó lekérdezés, amely lehetővé teszi, hogy több felhasználó is részt vegyen egy eseményen.

public function comments()

Leírás: Visszaadja az eseményhez tartozó megjegyzéseket.

Kimeneti adatok: hasMany relációt visszaadó lekérdezés, amely megadja, hogy az eseményhez több hozzászólás is tartozhat.

public function tags()

Leírás: Visszaadja a címkéket, amelyek az eseményhez kapcsolódnak.

Kimeneti adatok: belongsToMany relációt visszaadó lekérdezés a eventtags pivot táblán keresztül.

Pivot tábla: eventtags

Felhasználó ID: events\_id

Címke ID: tags\_id

public function subscriptions()

Leírás: Visszaadja az eseményhez kapcsolódó feliratkozásokat.

Kimeneti adatok: hasMany relációt visszaadó lekérdezés, amely megadja, hogy az eseményhez több feliratkozás is tartozhat.

### Subscription

Feladata: A felhasználók eseményekre való feliratkozásainak reprezentálására és kezelésére szolgáló modell. Az eseményekhez való feliratkozás lehetővé teszi a felhasználók számára, hogy nyomon követhessék az érdeklődési köreiket és részvételüket a különböző eseményeken.

Használat: Az Subscription osztály fontos szerepet játszik az események és felhasználók közötti kapcsolat kezelésében, lehetővé téve, hogy egy felhasználó több eseményre is feliratkozhasson, és információt tarthasson az feliratkozásával kapcsolatban, például észrevételeket és a feliratkozás dátumát.

Tulajdonságok és Metódusok:

Traits:

HasFactory: Lehetővé teszi a gyári metódusok használatát az feliratkozások egyszerű előállításához.

protected $fillable:

Leírás: Megadja azokat az attribútumokat, amelyeket tömegesen hozzáadhatunk (mass assignable).

Tartalmazza:

'users\_id': Az feliratkozó felhasználó azonosítója.

'events\_id': Az esemény azonosítója, amelyre a felhasználó feliratkozott.

'comment': Az feliratkozáshoz kapcsolódó megjegyzés.

'sub\_date': Az feliratkozás dátuma.

Kapcsolatok:

public function user()

Leírás: Visszaadja a felhasználót, aki az feliratkozást létrehozta.

Kimeneti adatok: belongsTo relációt visszaadó lekérdezés, amely lehetővé teszi, hogy egy feliratkozáshoz egy felhasználó tartozzon.

public function event()

Leírás: Visszaadja az eseményt, amelyhez az feliratkozás tartozik.

Kimeneti adatok: belongsTo relációt visszaadó lekérdezés, amely lehetővé teszi, hogy az feliratkozáshoz egy esemény tartozzon.

Egyedi kulcs: events\_id - ez a mező jelöli meg az esemény azonosítóját.

### Comment

(Nincs alkalmazva)

Feladata: A felhasználók által az eseményekhez fűzött megjegyzések reprezentálására és kezelésére szolgáló modell. A megjegyzések lehetővé teszik a felhasználók számára, hogy kifejezzék véleményüket, kérdéseiket, vagy bármilyen információt oszthassanak meg az adott eseménnyel kapcsolatban.

Használat: Az Comment osztály segítségével az alkalmazások képesek nyilvántartani az eseményekhez fűzött megjegyzéseket, amelyeket felhasználók adhatnak hozzá, ezzel növelve az interaktivitást és a felhasználói élményt.

Tulajdonságok és Metódusok:

Traits:

HasFactory: Lehetővé teszi a gyári metódusok használatát a megjegyzések egyszerű előállításához.

Kapcsolatok:

public function user()

Leírás: Visszaadja a felhasználót, aki a megjegyzést írta.

Kimeneti adatok: belongsTo relációt visszaadó lekérdezés, amely lehetővé teszi, hogy egy megjegyzéshez egy felhasználó tartozzon.

Használat: Ez a kapcsolat lehetővé teszi a felhasználói információk elérését a megjegyzésekből (pl. a megjegyzést író felhasználó neve vagy profilja).

public function event()

Leírás: Visszaadja az eseményt, amelyhez a megjegyzés tartozik.

Kimeneti adatok: belongsTo relációt visszaadó lekérdezés, amely lehetővé teszi, hogy egy megjegyzéshez egy esemény tartozzon.

Használat: Ez a kapcsolat lehetővé teszi a megjegyzés eseményhez való kapcsolódásának biztosítását, így könnyen lekérdezhető, hogy mely esemény alatt írták a megjegyzést.

## Resource

### User

Feladata: A felhasználói adatok (például az azonosítók, név, e-mail cím és adminisztrátori státusz) JSON formátumban történő átalakítása a Laravel API válaszain belül. Ez az osztály lehetővé teszi a felhasználói adatok strukturált megjelenítését és helyes kezelést.

Metódusok:

toArray(Request $request): array

Leírás: Átalakítja a felhasználói erőforrást egy tömbbé (array), amely JSON válaszként visszaadható.

Bejövő paraméterek:

$request: Az aktuális HTTP kérés, amely lehetővé teszi a további kontextus megadását.

Kimeneti adatok:

array: Az átalakított adatok tömbje. A következő mezőket tartalmazza:

"id": A felhasználó azonosítója (integer).

"name": A felhasználó neve (string).

"email": A felhasználó e-mail címe (string).

"admin": Boolean érték, amely jelzi, hogy a felhasználó adminisztrátor-e (boolean).

### Event

Feladata: Az események adatait (például az azonosítót, nevet, időpontokat, leírást és egyéb részleteket) JSON formátumban történő átalakítása a Laravel API válaszaiban. Ez az osztály lehetővé teszi az események strukturált megjelenítését egy API-n keresztül.

Metódusok:

toArray(Request $request): array

Leírás: Átalakítja az esemény erőforrást egy tömbbé (array), amely JSON válaszként visszaadható.

Bejövő paraméterek:

$request: Az aktuális HTTP kérés, amely lehetővé teszi a további kontextus megadását.

Kimeneti adatok:

array: Az átalakított adatok tömbje, amely a következő mezőket tartalmazza:

"id": Az esemény azonosítója (integer).

"name": Az esemény neve (string).

"startDate": Az esemény kezdő dátuma (string).

"endDate": Az esemény befejezésének a dátuma (string).

"startTime": Az esemény kezdési időpontja (formázott string).

"endTime": Az esemény befejezésének az időpontja (formázott string).

"description": Az esemény leírása (string).

"image": Az eseményhez tartozó kép URL-je (string).

"locationName": Az esemény helyszínének neve (string).

"locationcountry": Az esemény helyszínének országa (string).

"address": Az esemény címe (string).

"state": Az esemény állapota (string).

"weblink": Az esemény weboldala (string).

"latitude": Az esemény földrajzi szélessége (float).

"longitude": Az esemény földrajzi hosszúsága (float).

"gpx": GPX fájl hivatkozás (string).

"subscribed": Boolean érték, amely jelzi, hogy mennyi felhasználó van feliratkozva az eseményre.

formatTime(?string $time): ?string

Leírás: Segédmetódus az időpontok formázására. Ellenőrzi, hogy az időpont érvényes formátumban van-e, és ha igen, a H:i formátumban adja vissza.

Bejövő paraméterek:

$time: Az időpont, amelyet formázni kell. Nullable (lehet null).

Kimeneti adatok:

string|null: A formázott időpont string formátumban, vagy null, ha az időpont üres vagy érvénytelen.

### Tag

Feladata: A címke adatait (például az azonosítót, nevet és csoportot) JSON formátumban történő átalakítása a Laravel API válaszaiban. Ez az osztály lehetővé teszi a címkék egyszerű és strukturált megjelenítését egy API-n keresztül.

Metódusok:

toArray(Request $request): array

Leírás: Átalakítja a címke erőforrást egy tömbbé (array), amely JSON válaszként visszaadható.

Bejövő paraméterek:

$request: Az aktuális HTTP kérés, amely lehetővé teszi a további kontextus megadását.

Kimeneti adatok:

array: Az átalakított adatok tömbje, amely a következő mezőket tartalmazza:

"id": A címke azonosítója (integer).

"name": A címke neve (string).

"group": A címke csoportja (string).

### Comment

(Nincs alkalmazva)

Feladata: A megjegyzés adatait (például a felhasználó azonosítóját, az esemény azonosítóját, a megjegyzést és a minősítést) JSON formátumban történő átalakítása a Laravel API válaszaiban. Ez az osztály lehetővé teszi a megjegyzések egyszerű és strukturált megjelenítését egy API-n keresztül.

Metódusok:

toArray(Request $request): array

Leírás: Átalakítja a megjegyzés erőforrást egy tömbbé (array), amely JSON válaszként visszaadható.

Bejövő paraméterek:

$request: Az aktuális HTTP kérés, amely lehetővé teszi a további kontextus megadást.

Kimeneti adatok:

array: Az átalakított adatok tömbje, amely a következő mezőket tartalmazza:

'user\_id': A megjegyzést író felhasználó azonosítója (integer).

'event\_id': Az esemény azonosítója, amelyhez a megjegyzés kapcsolódik (integer).

'comment': A megjegyzés szövege (string).

'rating': A megjegyzéshez tartozó minősítés, ha van (integer vagy null).

## Kontrollerek

### Controller

Feladata az admin jogosultság kezelése.

Metódusok:

authorizeAdmin() –a felhasználó jogkörének leellenőrzését végzi

kimenő adatok: nincs

bejövő paraméterek: $user – felhasználó adatai

kimenő adatok: nincs

### ResponseController

Feladata: A válaszok kezelése az API-nak, beleértve a sikeres és hibás válaszok küldését, valamint az adminisztratív jogosultságok ellenőrzését.

Metódusok:

sendResponse($data, $message)

Leírás: Sikeres API válasz küldésére szolgál. A megadott adatot és üzenetet JSON formátumban adja vissza.

Bejövő paraméterek:

$data: Az API válaszban visszaadandó adat. Ez lehet bármilyen típusú, amely a JSON formátumba konvertálható.

$message: A válaszhoz tartozó üzenet, amely informálja a felhasználót a kérés eredményéről.

Kimenő adatok: Visszatér a JSON válasz, amely tartalmazza a következőket:

"success": true

"data": A megadott adat.

"message": A visszaküldött üzenet.

HTTP Státusz Kód: 200 OK

sendError($error, $errorMessages = [], $code = 404)

Leírás: Hibás API válasz küldésére szolgál. Megadja a hibaüzenetet és opcionálisan további hibaüzeneteket is.

Bejövő paraméterek:

$error: A hiba üzenete, amely informálja a felhasználót a kérés problémáiról.

$errorMessages: Opcionális. További hibák részletezése, amelyeket tömb formájában adhatunk meg.

$code: Opcionális. A HTTP státusz kód, amely alapértelmezés szerint 404.

Kimenő adatok: Visszatér a JSON válasz, amely tartalmazza a következőket:

"success": false

"message": A megadott hibaüzenet.

("errorMessage": Ha vannak további hibaüzenetek, azok kerülnek ide.)

HTTP Státusz Kód: A megadott kód (alapértelmezetten 404).

adminAuth()

Leírás: Ellenőrzi a felhasználó admin jogosultságait. Ha a felhasználó adminisztrátor (admin jogkör 2), automatikusan engedélyezi a hozzáférést.

Bejövő paraméterek: Nincs.

Kimenő adatok: Ha a felhasználó nem rendelkezik megfelelő admin jogosultsággal, hibaüzenetet küld, ellenkező esetben nincs kimenet.

Hibaüzenetek: Ha a felhasználó nem felel meg a jogosultságoknak, visszatér a sendError metódust használva a következővel:

"Azonosítási hiba": Hibaüzenet.

"Nincs jogosultság": További információ a problémáról.

HTTP Státusz Kód: 401 Unauthorized.

### VerificationController

Feladata: Az e-mail címek ellenőrzésének kezelése a felhasználók regizstrációjához. A controller lehetővé teszi a felhasználók számára az e-mail címük megerősítését egy kód vagy hash segítségével.

Metódusok:

verify(Request $request, $id, $hash)

Leírás: Ellenőrzi a felhasználó e-mail címét a megadott azonosító ($id) és hash ($hash) alapján.

Bejövő paraméterek:

$request: A HTTP kérés, amely tartalmazhat további adatokat.

$id: A felhasználó azonosítója, amely alapján az adatbázisból betöltjük a felhasználót.

$hash: A felhasználó e-mail címének titkosított formája, amelyet a belépési folyamat során használnak az e-mail ellenőrzéséhez.

Kimenő adatok:

Redirect: A felhasználót átirányítja a bejelentkezési oldalra, és különböző üzeneteket jelenít meg az eljárás eredményeként:

Ha a hash nem egyezik az e-mail cím SHA1 hash értékével: 'Érvénytelen ellenőrző link.'

Ha a felhasználó már igazolta az e-mail címét: 'Az e-mail már le van igazolva.'

Ha az e-mail cím sikeresen igazolva lett: 'Az e-mail sikeresen igazolva.'

HTTP Státusz Kód: 302 Found (mivel a felhasználót átirányítjuk).

### UserController

Feladata felhasználók lekérése, regisztrációja, bejelentkeztetése, kijelentkeztetése, módosítása, törlése, keresése, fel és leiratkozása

Metódusok:

getUsers()

Leírás: Visszaküldi az aktív felhasználók listáját az adataikkal

Bejövő paraméterek: bearer token autenikációhoz

Kimenő adatok: Aktív felhasználók

register(RegisterRequest $request)

Leírás: Regisztrál egy új felhasználót, amennyiben a megadott adatok megfelelnek a RegisterRequest által előírt feltételeknek.

Bejövő paraméterek:

"name": Felhasználónév.

"email": E-mail cím.

"password": Jelszó.

"confirm\_password": Jelszó megerősítése.

Kimenő adatok: Nincs. (Sikeres regisztráció esetén e-mailt küld a felhasználónak.)

login(LoginRequest $request)

Leírás: Ellenőrzi a felhasználó azonosítását név/email és jelszó alapján. Sikeres autentikáció esetén létre hoz egy tokent, amelyet tárol az adatbázisban, és visszaadja.

Bejövő paraméterek:

"name": Felhasználónév vagy e-mail.

"password": Jelszó.

Kimenő adatok: A sikeres bejelentkezés után visszaadott token és üzenet.

logout()

Leírás: Kijelentkezteti a felhasználót, és törli a kiválasztott tokent az adatbázisból.

Bejövő paraméterek: Bearer token az autentikációhoz.

Kimenő adatok: A sikeres kijelentkezés üzenete.

modifyUser(ModifyUserRequest $request)

Leírás: Módosítja a felhasználó adatait, amennyiben a felhasználó jogosult az adatok módosítására.

Bejövő paraméterek:

"id": Felhasználó azonosítója.

"name": Felhasználónév.

"email": E-mail cím.

"password" (opcionális): Új jelszó.

"admin" (opcionális): Jogosultságok módosítása.

Kimenő adatok: Módosított felhasználó adatai vagy hibaüzenet.

destroyUser(int $id)

Leírás: Törli a felhasználót az adatbázisból, amennyiben a felhasználó jogosult a törlésre.

Bejövő paraméterek:

int $id: A törlendő felhasználó azonosítója.

Kimenő adatok: Törölt felhasználó adatai vagy hibaüzenet.

searchUser(LoginRequest $request)

Leírás: Keres felhasználókat a megadott név alapján.

Bejövő paraméterek:

"query": Keresési kifejezés.

Kimenő adatok: A keresett felhasználók listája.

subUserTag(Request $request)

Leírás: Hozzáad címkéket a felhasználóhoz.

Bejövő paraméterek:

"tags": Hozzáadni kívánt címkék azonosítói.

Kimenő adatok: A felhasználó új címkéivel együtt visszaadott üzenet.

unsubUserTag(Request $request)

Leírás: Eltávolít címkéket a felhasználótól.

Bejövő paraméterek:

"tags": Eltávolítandó címkék azonosítói.

Kimenő adatok: Eltávolított címkék esetén visszaadott üzenet.

### EventController

Feladata: Az események kezelésének központosítása az API-n belül, beleértve az események létrehozását, lekérdezését, módosítását, törlését, keresését és címkékkel való kapcsolat kezelését.

Metódusok:

getEvents()

Leírás: Lekéri az összes eseményt az adatbázisból.

Bejövő paraméterek: Nincs.

Kimenő adatok: JSON válasz, amely tartalmazza az összes eseményt EventResource formájában, és a következő üzenetet: "Események betöltve".

HTTP Státusz Kód: 200 OK

newEvent(EventRequest $request)

Leírás: Új esemény létrehozása az API-n keresztül, ha a felhasználó admin jogosultsággal rendelkezik.

Bejövő paraméterek:

$request: A HTTP kérés, amely tartalmazza az esemény adatait.

Kimenő adatok: JSON válasz, amely a létrehozott eseményt tartalmazza EventResource formájában, és a következő üzenetet: "Esemény létrehozva".

HTTP Státusz Kód: 201 Created

modifyEvent(EventRequest $request)

Leírás: Meglévő esemény módosítása az API-n keresztül, ha a felhasználó admin jogosultsággal rendelkezik.

Bejövő paraméterek:

$request: A HTTP kérés, amely tartalmazza az esemény módosítandó adatait.

Kimenő adatok: JSON válasz a módosított eseménnyel EventResource formájában, vagy hibaüzenet, ha az esemény nem létezik.

HTTP Státusz Kódok:

200 OK (ha a módosítás sikeres)

406 Not Acceptable (ha az esemény nem található)

destroyEvent(int $id)

Leírás: Esemény törlése az API-n keresztül.

Bejövő paraméterek:

$id: Az esemény azonosítója, amelyet törölni kell.

Kimenő adatok: JSON válasz a törölt eseményről, vagy hibaüzenet, ha az esemény nem létezik.

HTTP Státusz Kódok:

200 OK (ha a törlés sikeres)

406 Not Acceptable (ha az esemény nem található)

getEventId(EventRequest $request)

Leírás: Lekéri egy esemény azonosítóját az esemény nevének megadásával.

Bejövő paraméterek:

$request: A HTTP kérés, amely tartalmazza az esemény nevét.

Kimenő adatok: Az esemény azonosítója.

HTTP Státusz Kód: 200 OK

searchEvent(Request $request)

Leírás: Események keresése a név, leírás vagy egyéb paraméterek alapján.

Bejövő paraméterek:

$request: A HTTP kérés, amely tartalmaz egy keresési lekérdezést.

Kimenő adatok: JSON válasz az illeszkedő események gyűjteményével EventResource formájában.

HTTP Státusz Kód: 200 OK

attachTags(Request $request, $eventId)

Leírás: Címkék hozzáadása egy konkrét eseményhez.

Bejövő paraméterek:

$request: A HTTP kérés, amely tartalmazza a címkék azonosítóit.

$eventId: Az esemény azonosítója, amelyhez a címkék csatolandók.

Kimenő adatok: JSON válasz a frissített eseményről, vagy hibaüzenet, ha a címkék nem találhatók.

HTTP Státusz Kódok:

200 OK (ha a címkék sikeresen hozzáadva)

404 Not Found (ha az esemény vagy a címkék nem találhatók)

detachTag($eventId, $tagId)

Leírás: Címke eltávolítása a megadott eseményről.

Bejövő paraméterek:

$eventId: Az esemény azonosítója.

$tagId: A címke azonosítója, amelyet el kell távolítani.

Kimenő adatok: JSON válasz a frissített eseményről, vagy hibaüzenet, ha az esemény nem található.

HTTP Státusz Kódok:

200 OK (ha a címke eltávolítás sikeres)

404 Not Found (ha az esemény nem található)

getTagsByEvents(int $eventId)

Leírás: Lekéri az eseményhez tartozó címkéket.

Bejövő paraméterek:

$eventId: Az esemény azonosítója.

Kimenő adatok: JSON válasz a címkékkel, vagy üzenet, ha nincs címke társítva.

HTTP Státusz Kódok:

200 OK (ha az eseményhez címkék társítva találhatók)

404 Not Found (ha az esemény nem található)

getEventsWithTags()

Leírás: Összes esemény lekérdezése címkékkel együtt.

Bejövő paraméterek: Nincs.

Kimenő adatok: JSON válasz az eseményekkel, vagy üzenet, ha nincsenek címkékkel ellátott események.

HTTP Státusz Kódok:

200 OK (ha vannak események címkékkel)

200 OK (ha nincsenek események, üzenettel)

### TagController

Feladata: A címkék kezelésének központosítása az API-n belül, beleértve a címkék létrehozását, lekérdezését, módosítását, törlését, keresését és címkékhez kapcsolódó események lekérdezését.

Metódusok:

getTags()

Leírás: Lekéri az összes címkét az adatbázisból.

Bejövő paraméterek: Nincs.

Kimenő adatok: JSON válasz, amely tartalmazza az összes címkét TagResource formájában, és a következő üzenetet: "Címkék betöltve".

HTTP Státusz Kód: 200 OK

newTag(TagRequest $request)

Leírás: Új címke létrehozása az API-n keresztül, ha a felhasználó admin jogosultsággal rendelkezik.

Bejövő paraméterek:

$request: A HTTP kérés, amely tartalmazza a címke adatait (név, csoport).

Kimenő adatok: JSON válasz a létrehozott címkével TagResource formájában, és a következő üzenetet: "Címke kiírva".

HTTP Státusz Kód: 201 Created

modifyTag(TagRequest $request)

Leírás: Meglévő címke módosítása az API-n keresztül, ha a felhasználó admin jogosultsággal rendelkezik.

Bejövő paraméterek:

$request: A HTTP kérés, amely tartalmazza a módosítandó címke adatait (azonosító, név, csoport).

Kimenő adatok: JSON válasz a módosított címkéről TagResource formájában, vagy hibaüzenet, ha a címke nem létezik.

HTTP Státusz Kódok:

200 OK (ha a módosítás sikeres)

406 Not Acceptable (ha a címke nem található)

destroyTag(int $id)

Leírás: Címke törlése az API-n keresztül.

Bejövő paraméterek:

$id: A címke azonosítója, amelyet törölni kell.

Kimenő adatok: JSON válasz a törölt címkéről TagResource formájában, vagy hibaüzenet, ha a címke nem létezik.

HTTP Státusz Kódok:

200 OK (ha a törlés sikeres)

406 Not Acceptable (ha a címke nem található)

getTagId(TagRequest $request)

Leírás: Lekéri egy címke azonosítóját a címke nevének megadásával.

Bejövő paraméterek:

$request: A HTTP kérés, amely tartalmazza a címke nevét.

Kimenő adatok: A megtalált címke azonosítója.

HTTP Státusz Kód: 200 OK

searchTag(TagRequest $request)

Leírás: Címkék keresése a név alapján.

Bejövő paraméterek:

$request: A HTTP kérés, amely tartalmaz egy keresési lekérdezést.

Kimenő adatok: JSON válasz az illeszkedő címkék gyűjteményével TagResource formájában.

HTTP Státusz Kód: 200 OK

getGroups()

Leírás: Visszaadja a címkecsoportokat, amiket a rendszer kezelésére használ.

Bejövő paraméterek: Nincs.

Kimenő adatok: JSON válasz a címkecsoportok listájával, és a következő üzenettel: "Csoportok betöltve".

HTTP Státusz Kód: 200 OK

getEventsByTag(int $tagId)

Leírás: Lekéri az eseményeket, amelyekhez a megadott címke kapcsolódik.

Bejövő paraméterek:

$tagId: A címke azonosítója, amelyhez az eseményeket lekérdezzük.

Kimenő adatok: JSON válasz az események gyűjteményével EventResource formájában, vagy üzenet, ha nincsenek események a címkéhez társítva.

HTTP Státusz Kódok:

200 OK (ha vannak események a címkéhez)

404 Not Found (ha a címke nem található)

### SubscriptionController

Feladata: A feliratkozások kezelése az API-n belül, beleértve a felhasználók eseményekre történő feliratkozását, leiratkozását és a feliratkozások adatainak kezelését.

Metódusok:

getSubscriptions()

Leírás: Lekéri az aktuális felhasználó összes feliratkozását, az eseményekkel együtt.

Bejövő paraméterek: Nincs.

Kimenő adatok: JSON válasz a feliratkozások listájával, valamint egy üzenet: "Feliratkozások betöltve."

HTTP Státusz Kód: 200 OK

subscribe(SubscribeRequest $request)

Leírás: Feliratkozás egy eseményre. Admin felhasználóknem iratkozhatnak fel.

Bejövő paraméterek:

$request: A HTTP kérés, amely tartalmazza az events\_id-t (az esemény azonosítója) és opcionálisan egy comment-et (kommentár).

Kimenő adatok: JSON válasz a létrehozott feliratkozással, valamint a következő üzenet: "Sikeres föliratkozás."

HTTP Státusz Kódok:

201 Created (ha a feliratkozás sikeres)

403 Forbidden (ha az admin felhasználó próbálkozik)

404 Not Found (ha az esemény nem található)

400 Bad Request (ha a felhasználó már fel van iratkozva)

updateSubscription(Request $request, $eventId)

Leírás: Eseményre vonatkozó feliratkozás frissítése, ahol a komment (élménybeszámoló) mező megváltoztatható.

Bejövő paraméterek:

$request: A HTTP kérés, amely tartalmazza az opcionális comment-et.

$eventId: Az esemény azonosítója, amire frissíteni szeretnénk a feliratkozást.

Kimenő adatok: JSON válasz a frissített feliratkozással, valamint a következő üzenet: "Sikeresen frissített feliratkozás."

HTTP Státusz Kódok:

200 OK (ha a frissítés sikeres)

404 Not Found (ha a felhasználó nincs feliratkozva vagy az esemény nem található)

403 Forbidden (ha az esemény kezdete nem járt le)

unsubscribe($eventId)

Leírás: Feliratkozás törlése az adott eseményről.

Bejövő paraméterek:

$eventId: Az esemény azonosítója, amelyről le szeretnénk iratkozni.

Kimenő adatok: JSON válasz nullával (vagy üres válasz), és a következő üzenet: "Sikeres leiratkozás."

HTTP Státusz Kódok:

200 OK (ha a leiratkozás sikeres)

404 Not Found (ha a felhasználó nincs feliratkozva az eseményre)

## Request

### RegisterRequest

Feladata: A HTTP kérés érvényesítése a felhasználói regisztráció során. Biztosítja, hogy a bejövő adatok megfeleljenek a megadott szabályoknak, és hogy a felhasználó regisztrálása sikeres legyen.

Metódusok:

authorize(): bool

Leírás: Ellenőrzi, hogy a felhasználó jogosult-e a regisztrációs kérés végrehajtására.

Kimeneti adatok:

true: Mivel a metódus mindig true-t ad vissza, bármilyen felhasználó jogosult a kérés végrehajtására.

rules(): array

Leírás: Visszaadja a validációs (érvényesítési) szabályokat, amelyek érvényesek a regisztrációs űrlap benyújtására.

Kimeneti adatok:

array: A validációs szabályokat tartalmazó tömb, amely a következő mezőkre vonatkozik:

name: Kötelező, legalább 3, legfeljebb 20 karakter, és egyedi a users táblában (figyelembe véve a soft delete műveleteket).

email: Kötelező, email formátumú, és egyedi a users táblában (figyelembe véve a soft delete műveleteket).

password: Kötelező, legalább 8 karakter hosszú, és tartalmaznia kell kisbetűt, nagybetűt és számot.

confirm\_password: Kötelező, a jelszó megerősítése, amelynek egyeznie kell a password mezővel.

messages(): array

Leírás: Visszaadja az egyedi hibaüzeneteket a validációs szabályokhoz, amelyeket a felhasználó láthat.

Kimeneti adatok:

array: Olyan kulcs-érték párok, ahol a kulcsok a validációs szabályok nevei, az értékek pedig a hibaüzenetek. Például:

"name.required": "Név kötelező."

"name.unique": "Létező név."

"password.regex": "A jelszónak tartalmaznia kell kisbetűt, nagybetűt és számot."

"confirm\_password.same": "Nem egyező jelszó."

failedValidation(Validator $validator)

Leírás: Kezeli a validációs hibákat, és JSON válasz formájában adja vissza azokat.

Bejövő paraméterek:

$validator: A validációs hibák részleteit tartalmazó objektum.

Kimeneti adatok:

JSON válasz, amely tartalmazza:

"success": false

"message": "Adatbeviteli hiba"

"error": A validációs hibákat tartalmazó tömb.

### LoginRequest

Feladata: A HTTP kérés érvényesítése a felhasználói bejelentkezés számára. Biztosítja, hogy a bejövő adatok megfeleljenek a megadott szabályoknak, és megfelelő válaszokat ad, ha az adatok nem érvényesek.

Metódusok:

authorize(): bool

Leírás: Ellenőrzi, hogy a felhasználó jogosult-e a kérés végrehajtására.

Kimeneti adatok:

true: Minden felhasználónak engedélyezi a kérés végrehajtását.

rules(): array

Leírás: Visszaadja a validációs (érvényesítési) szabályokat, amelyek érvényesek a bejövő kérésre.

Kimeneti adatok:

array: A validációs szabályokat tartalmazó tömb, amely a következő mezőkre vonatkozik:

name: Kötelező mező, a felhasználó neve.

password: Kötelező mező, a felhasználó jelszava.

messages(): array

Leírás: Visszaadja az egyedi hibaüzeneteket a validációs szabályokhoz, amelyeket a felhasználó láthat.

Kimeneti adatok:

array: Olyan kulcs-érték párok, ahol a kulcsok a szabályok nevei, az értékek pedig a hibaüzenetek, például:

"name.required": "Név kötelező"

"password.required": "Jelszó kötelező"

failedValidation(Validator $validator)

Leírás: Kezeli a validációs hibákat, és JSON válasz formájában adja vissza azokat.

Bejövő paraméterek:

$validator: A validációs hibák részleteit tartalmazó objektum.

Kimeneti adatok:

JSON válasz, amely tartalmazza:

"success": false

"message": "Adatbeviteli hiba"

"error": A validációs hibákat tartalmazó tömb.

### ModifyUserRequest

Feladata: A HTTP kérés érvényesítése a felhasználói adatok módosításakor. Biztosítja, hogy a bejövő adatok megfeleljenek a megadott szabályoknak és hogy a felhasználó jogosult legyen az adatok módosítására.

Metódusok:

authorize(): bool

Leírás: Ellenőrzi, hogy a felhasználó jogosult-e a kérés végrehajtására.

Kimeneti adatok:

true: Ha a felhasználó "super" vagy "admin" jogosultsággal rendelkezik, akkor a kérés engedélyezett.

false: Ha a felhasználó nem rendelkezik ezekkel a jogosultságokkal, akkor a kérés nem engedélyezett.

rules(): array

Leírás: Visszaadja a validációs (érvényesítési) szabályokat a módosítandó felhasználói adatokhoz.

Kimeneti adatok:

array: A validációs szabályokat tartalmazó tömb, amely a következő mezőkre vonatkozik:

id: Kötelező, léteznie kell a users táblában.

name: Kötelező, minimum 3, maximum 20 karakter, egyedi a users táblában (a módosítandó id figyelembevételével).

email: Kötelező, email formátumú, egyedi a users táblában (a módosítandó id figyelembevételével).

password: Opcionális, legalább 8 karakter hosszú, és kell, hogy tartalmazzon kisbetűt, nagybetűt és számot.

(A admin mező, amelyet kommentáltál, a jogosultságok kezelésére lenne hivatott, de nincs jelenleg aktiválva).

messages(): array

Leírás: Visszaadja az egyedi hibaüzeneteket a validációs szabályokhoz, amelyeket a felhasználó láthat.

Kimeneti adatok:

array: Olyan kulcs-érték párok, ahol a kulcsok a szabályok nevei, az értékek pedig a hibaüzenetek, például:

"id.required": "Felhasználó azonosító megadása kötelező."

"email.email": "Nem megfelelő formátum."

"name.required": "Név kötelező."

(Bármely más hibaüzenet, amely az öt szerszámhoz tartozik).

failedValidation(Validator $validator)

Leírás: Kezeli a validációs hibákat, és JSON válasz formájában adja vissza azokat.

Bejövő paraméterek:

$validator: A validációs hibák részleteit tartalmazó objektum.

Kimeneti adatok:

JSON válasz, amely tartalmazza:

"success": false

"message": "Adatbeviteli hiba"

"error": A validációs hibákat tartalmazó tömb.

### EventRequest

Feladata: A HTTP kérés érvényesítése az események létrehozásához és frissítéséhez. Biztosítja, hogy a bejövő adatok megfeleljenek a megadott szabályoknak, és megfelelő válaszokat ad, ha az adatok nem érvényesek.

Metódusok:

authorize(): bool

Leírás: Ellenőrzi, hogy a felhasználó jogosult-e a kérés végrehajtására.

Kimeneti adatok:

true: Minden felhasználónak engedélyezi a kérés végrehajtását.

rules(): array

Leírás: Visszaadja a validation (érvényesítési) szabályokat, amelyek alkalmazandók a bejövő kérésre.

Kimeneti adatok:

array: A validációs szabályokat tartalmazó tömb. A következő mezőkre vonatkozik:

name: Kötelező, minimum 2, maximum 40 karakter, egyedi a events táblában.

startDate: Dátum formátum, a endDate előtt vagy egyenlő lehet, nem kötelező.

endDate: Dátum formátum, a startDate után vagy egyenlő lehet, nem kötelező.

startTime: Időformátum, nem kötelező.

endTime: Időformátum, nem kötelező.

description: Opcionális szöveg.

locationName, locationcountry, address, state: Opcionális szövegek.

weblink: Opcionális URL formátum.

latitude, longitude: Opcionális numerikus értékek.

gpx: Opcionális szöveg.

Megjegyzés: A name mező egyedi ellenőrzése a kérés típusától függően változik: PATCH vagy PUT metódus esetén figyelembe veszi az esemény ID-ját, hogy elkerülje a saját nevekkel való ütközést.

messages(): array

Leírás: Visszaadja az egyedi hibaüzeneteket a validációs szabályokhoz, amelyek orientálják a felhasználót az érvényesítési hibák esetén.

Kimeneti adatok:

array: Olyan kulcs-érték párok, ahol a kulcsok a szabályok nevei, az értékek pedig a hibaüzenetek, például:

"name.required": "Esemény neve elvárt."

"startDate.date": "Érvényes kezdeti dátum szükséges."

(És így tovább a többi mező hibaüzenetével).

failedValidation(Validator $validator)

Leírás: Kezeli a validációs hibákat, és JSON válasz formájában adja vissza azokat.

Bejövő paraméterek:

$validator: A validációs hibák részleteit tartalmazó objektum.

Kimeneti adatok:

JSON válasz, amely tartalmazza:

"success": false

"message": "Adatbeviteli hiba"

"error": A validációs hibákat tartalmazó tömb.

### SubscribeRequest

Feladata: A HTTP kérés érvényesítése a felhasználók eseményekhez való feliratkozásakor. Biztosítja, hogy a bejövő adatok megfeleljenek a megadott szabályoknak, és lehetővé teszi az eseményekhez való kapcsolódást.

Metódusok:

authorize()

Leírás: Ellenőrzi, hogy a felhasználó jogosult-e a feliratkozási kérés végrehajtására.

Kimeneti adatok:

true: Mivel a metódus mindig true-t ad vissza, bármilyen felhasználó jogosult a kérés végrehajtására.

rules()

Leírás: Visszaadja a validációs (érvényesítési) szabályokat, amelyek érvényesek a feliratkozási űrlap benyújtására.

Kimeneti adatok:

array: A validációs szabályokat tartalmazó tömb, amely a következő mezőkre vonatkozik:

events\_id: Kötelező, léteznie kell az events táblában (ellenőrzi, hogy az esemény az adatbázisban van-e).

comment: Opcionális, ha megadva, akkor karakterlánc formátumú.

### TagRequest

Feladata: A HTTP kérés érvényesítése a címkék létrehozásakor vagy frissítésekor. Biztosítja, hogy a bejövő adatok megfeleljenek a megadott szabályoknak, és hogy a címkék kezelése megfelelően történjen.

Metódusok:

authorize(): bool

Leírás: Ellenőrzi, hogy a felhasználó jogosult-e a címke létrehozási vagy frissítési kérés végrehajtására.

Kimeneti adatok:

true: Mivel a metódus mindig true-t ad vissza, bármilyen felhasználó jogosult a kérés végrehajtására.

rules(): array

Leírás: Visszaadja a validációs (érvényesítési) szabályokat, amelyek érvényesek a címke adataira.

Kimeneti adatok:

array: A validációs szabályokat tartalmazó tömb, amely a következő mezőkre vonatkozik:

name: Kötelező mező, minimum 2, maximum 20 karakter hosszúságú.

group: Kötelező mező.

Feltételes logika:

Ha a kérést PATCH vagy PUT metódussal küldik, a validációs szabályhoz hozzáadódik a unique:tags,name,$tagId feltétel, ahol $tagId a frissítendő címke azonosítója.

Más esetben az unique:tags,name szabály alkalmazandó, amely biztosítja, hogy az új címke neve egyedi legyen.

messages()

Leírás: Visszaadja az egyedi hibaüzeneteket a validációs szabályokhoz, amelyeket a felhasználó láthat.

Kimeneti adatok:

array: Olyan kulcs-érték párok, ahol a kulcsok a validációs szabályok nevei, az értékek pedig a hibaüzenetek. Például:

"name.required": "Címke neve elvárt."

"name.unique": "Címke már létezik ezzel a megnevezéssel."

"name.min": "Túl rövid címke név."

"name.max": "Túl hosszú címke név."

"group.required": "Csoport elvárt."

failedValidation(Validator $validator)

Leírás: Kezeli a validációs hibákat, és JSON válasz formájában adja vissza azokat.

Bejövő paraméterek:

$validator: A validációs hibák részleteit tartalmazó objektum.

Kimeneti adatok:

JSON válasz, amely tartalmazza:

"success": false

"message": "Adatbeviteli hiba"

"error": A validációs hibákat tartalmazó tömb.

## Middleware

### AdminAuthMiddleware

Feladata: A felhasználók admin jogosultságainak ellenőrzése a beérkező HTTP kérések során. Megakadályozza a nem jogosult felhasználók hozzáférését a védett erőforrásokhoz.

Metódusok:

handle($request, Closure $next)

Leírás: Ellenőrzi, hogy a felhasználó be van-e jelentkezve és rendelkezik-e admin jogosultságokkal. Ha a felhasználó nem authentikált vagy nem rendelkezik admin jogosultságokkal, a feldolgozás megszakad, és hibaüzenetet küldd vissza.

Bejövő paraméterek:

$request: A beérkező HTTP kérés.

$next: A következő middleware vagy a végpont kezelője.

Kimenő adatok:

Ha a felhasználó nincs authentikálva: JSON válasz, amely tartalmazza a következő üzenetet: 'User not authenticated' és a HTTP státusz kód: 401 Unauthorized.

Ha a felhasználó nem rendelkezik admin jogosultsággal: JSON válasz, amely tartalmaz egy hibaüzenetet:{"error": "Azonosítási hiba", "message": "Nincs jogosultság"}és a HTTP státusz kód: 403 Forbidden.

Ha a felhasználó authentikált és jogosult: A kérés átadása a következő middleware-nek vagy a végpont kezelőjének.

## Mail

### EmailVerification

Feladata: E-mail üzenetek létrehozása a felhasználói e-mail címek megerősítésére. Ez az osztály kezeli az e-mail sablonokat, az üzenet tartalmát, valamint a szükséges adatokat a megerősítő link generálásához.

Használat: Az osztály használatával egyszerűen létrehozhatunk megerősítő e-maileket, amelyet elküldhetünk az új felhasználók számára a regisztráció során.

Metódusok:

\_\_construct(User $user)

Leírás: Az osztály konstruktora, amely a felhasználó adatokat inicializálja.

Bejövő paraméterek:

$user: A felhasználói információkat tartalmazó User objektum.

envelope(): Envelope

Leírás: Visszaadja az üzenet borítékát, beleértve a tárgyat.

Kimeneti adatok:

Envelope: Az üzenet borítékát reprezentáló objektum, amely tartalmazza a subject mezőt.

content(): Content

Leírás: Visszaadja az üzenet tartalmának definícióját, beleértve a sablont és az átadott változókat.

Kimeneti adatok:

Content: Az üzenet tartalmát képviselő objektum. A tartalom a view mezőben megadott emails.verify sablont használja, amelyhez hozzárendel egy verificationUrl -t.

attachments(): array

Leírás: A visszaadott tömb az üzenethez csatolt fájlokat tartalmazza. Jelen esetben nem tartalmaz csatolmányokat, üres tömböt ad vissza.

Kimeneti adatok:

array: Az üzenethez csatolt fájlok (ha léteznek) tömbje.

## Provider

### AppServiceProvider

Feladata: A AppServiceProvider osztály a Laravel szolgáltatási regisztrátorának és bootstrapping osztályának szerepét tölti be. Itt lehetőség van különböző alkalmazás-szolgáltatások regisztrálására, valamint az alkalmazás indításakor végrehajtandó logikák definiálására. Különösen fontos szerepet játszik a felhasználói jogosultságok kezelésében is.

Tulajdonságok és Metódusok:

public function register(): void

Leírás: Az alkalmazás szolgáltatásainak regisztrálására szolgáló metódus. Itt általában a különböző alkalmazás-specifikus szolgáltatásokat regisztrálhatjuk.

Implementáció: Jelen esetben a metódus üres, de a későbbiekben itt érkezhetnek be a különböző szolgáltatások.

Jogosultság: Fejlesztői forrásként szolgál, ahol a további szolgáltatások, függőségek, vagy harmadik fél csomagok regisztrálása történhet.

public function boot(): void

Leírás: Az alkalmazás szolgáltatásainak inicializálására szolgáló metódus. Itt hajtható végre az a logika, amely végigfut az alkalmazás indításakor.

Implementáció:

Gate::before(): A Gate rendszerrel kapcsolatos jogosultságok ellenőrzéséhez használható. Az admin mező értéke 2 esetén a metódus true értéket ad vissza, így a felhasználó automatikusan minden jogosultságot megkap szuper adminként.

Gate::define('admin', ..): A admin jogosultság definiálása. Ha a felhasználó admin mezője 1, akkor a metódus true értéket ad vissza.

# Adatbázis

|  |  |  |
| --- | --- | --- |
| cache tábla | | |
| Mező | Típus / hossz | Leírás |
| key | varchar(255) | nincs alkalmazva |
| value | mediumtext | nincs alkalmazva |
| expiration | int(11) | nincs alkalmazva |

A cache tábla nincs alkalmazva

|  |  |  |
| --- | --- | --- |
| cache\_locks tábla | | |
| Mező | Típus / hossz | Leírás |
| key | varchar(255) | nincs alkalmazva |
| owner | varchar(255) | nincs alkalmazva |
| expiration | int(11) | nincs alkalmazva |

A cache\_locks tábla nincs alkalmazva

|  |  |  |
| --- | --- | --- |
| comments tábla | | |
| Mező | Típus / hossz | Leírás |
| users\_id | bigint(20) | users táblából id mint idegenkulcs |
| events\_id | bigint(20) | events táblából id mint idegenkulcs |
| comment | text utf8mb4\_unicode\_ci | felhasználók által írt hozzászólás szövege |
| rating | int(11) | felhasználó saját esemény értékelése  (nincs alkalmazva) |
| created\_at | timestamp | létrehozás dátuma és pontos ideje |
| updated\_at | timestamp | módosítás dátuma és pontos ideje |

A comments tábla nincs alkalmazva

|  |  |  |
| --- | --- | --- |
| events tábla | | |
| Mező | Típus / hossz | Leírás |
| id | bigint(20) UNSIGNED | esemény egyedi azonosítója automatikusan kapja |
| name | varchar(255) utf8mb4\_unicode\_ci | esemény megnevezése |
| startDate | date | esemény kezdetének a dátuma |
| endDate | date | esemény végének a dátuma |
| startTime | time | esemény kezdetének ideje |
| endTime | time | esemény végének ideje |
| description | text utf8mb4\_unicode\_ci | esemény leírása |
| image | varchar(255) utf8mb4\_unicode\_ci | kép hivatkozása |
| locationName | varchar(255) utf8mb4\_unicode\_ci | esemény helyének a megnevezése |
| locationcountry | varchar(255) utf8mb4\_unicode\_ci | esemény országának a megnevezése |
| address | varchar(255) utf8mb4\_unicode\_ci | esemény helyének a címe |
| state | int(11) | az eseménynek az állapota  (nincs alkalmazva) |
| weblink | varchar(255) utf8mb4\_unicode\_ci | az eseményhez tartozó weboldal linkje |
| latitude | decimal(10,7) | földrajzi szélesség a gps api számára  (nincs alkalmazva) |
| longitude | decimal(10,7) | földrajzi hosszúság a gps api számára  (nincs alkalmazva) |
| gpx | varchar(255) utf8mb4\_unicode\_ci | adat a gps számára (nincs alkalmazva |
| subscribed | int(10) | az eseményre jelenleg feliratkozott felhasználók száma |
| created\_at | timestamp | létrehozás dátuma és pontos ideje |
| updated\_at | timestamp | módosítás dátuma és pontos ideje |

|  |  |  |
| --- | --- | --- |
| eventtags tábla | | |
| Mező | Típus / hossz | Leírás |
| events\_id | bigint(20) | events táblából id mint idegenkulcs |
| tags\_id | bigint(20) | tags táblából id mint idegenkulcs |
| created\_at | timestamp | létrehozás dátuma és pontos ideje |
| updated\_at | timestamp | módosítás dátuma és pontos ideje |

|  |  |  |
| --- | --- | --- |
| favorites tábla | | |
| Mező | Típus / hossz | Leírás |
| users\_id | bigint(20) | users táblából id mint idegenkulcs |
| events\_id | bigint(20) | events táblából id mint idegenkulcs |
| created\_at | timestamp | létrehozás dátuma és pontos ideje |
| updated\_at | timestamp | módosítás dátuma és pontos ideje |

A favorites tábla nincs alkalmazva

|  |  |  |
| --- | --- | --- |
| migrations tábla | | |
| Mező | Típus / hossz | Leírás |
| id | int(10) UNSIGNED | migrációs tábla egyedi azonosítója automatikusan kapja |
| migration | varchar(255) utf8mb4\_unicode\_ci | migrációs tábla fájl neve |
| batch | int(11) | migrációs tábla lefutásának száma |

|  |  |  |
| --- | --- | --- |
| password\_reset\_tokens tábla | | |
| Mező | Típus / hossz | Leírás |
| email | varchar(255) utf8mb4\_unicode\_ci | nincs alkalmazva |
| token | varchar(255) utf8mb4\_unicode\_ci | nincs alkalmazva |
| created\_at | timestamp | nincs alkalmazva |

A password\_reset\_tokens tábla nincs alkalmazva

|  |  |  |
| --- | --- | --- |
| personal\_access\_tokens tábla | | |
| Mező | Típus / hossz | Leírás |
| id | bigint(20) UNSIGNED | token egyedi azonosítója automatikusan kapja |
| okenable\_type | varchar(255) utf8mb4\_unicode\_ci | token típusa |
| tokenable\_id | bigint(20) UNSIGNED | tokenhez tartozó azonosító jelen esetben userekhez tartoznak |
| name | varchar(255) utf8mb4\_unicode\_ci | token megnevezése |
| token | varchar(64) utf8mb4\_unicode\_ci | token hashelve |
| abilities | text utf8mb4\_unicode\_ci | tokenben kezelt jogosultságok (nincs alkalmazva) |
| last\_used\_at | timestamp | token legutóbbi felhasználásának az ideje |
| expires\_at | timestamp | token lejárati ideje (nincs alkalmazva) |
| created\_at | timestamp | létrehozás dátuma és pontos ideje |
| updated\_at | timestamp | módosítás dátuma és pontos ideje |

|  |  |  |
| --- | --- | --- |
| sessions tábla | | |
| Mező | Típus / hossz | Leírás |
| id | varchar(255) utf8mb4\_unicode\_ci | munkamenethez tartozó azonosító |
| user\_id | bigint(20) | munkamenethez tartozó felhasználó azonosítója |
| ip\_address | varchar(255) utf8mb4\_unicode\_ci | munkamenet ip címe |
| user\_agent | text utf8mb4\_unicode\_ci | munkamenet kliensalkalmazásának a megnevezése |
| payload | longtext utf8mb4\_unicode\_ci | töltet szerializált formában |
| last\_activity | bigint(11) | unixos timestamp |

|  |  |  |
| --- | --- | --- |
| subscriptions tábla | | |
| Mező | Típus / hossz | Leírás |
| id | bigint(20) UNSIGNED | feliratkozás egyedi azonosítója automatikusan kapja |
| users\_id | bigint(20) | users táblából id mint idegenkulcs |
| events\_id | bigint(20) | events táblából id mint idegenkulcs |
| comment | text utf8mb4\_unicode\_ci | felhasználó által írt élménybeszámloló |
| sub\_date | date | feliratkozás ideje |
| created\_at | timestamp | létrehozás dátuma és pontos ideje |
| updated\_at | timestamp | módosítás dátuma és pontos ideje |

|  |  |  |
| --- | --- | --- |
| tags tábla | | |
| Mező | Típus / hossz | Leírás |
| id | bigint(20) UNSIGNED | címke egyedi azonosítója automatikusan kapja |
| name | varchar(255) utf8mb4\_unicode\_ci | címke megnevezése |
| group | varchar(255) utf8mb4\_unicode\_ci | címke besorolási kategóriája (beégetett) |
| created\_at | timestamp | létrehozás dátuma és pontos ideje |
| updated\_at | timestamp | módosítás dátuma és pontos ideje |

|  |  |  |
| --- | --- | --- |
| tag\_subscriptions tábla | | |
| Mező | Típus / hossz | Leírás |
| users\_id | bigint(20) | users táblából id mint idegenkulcs |
| tags\_id | bigint(20) | tags táblából id mint idegenkulcs |
| created\_at | timestamp | létrehozás dátuma és pontos ideje |
| updated\_at | timestamp | módosítás dátuma és pontos ideje |

|  |  |  |
| --- | --- | --- |
| users tábla | | |
| Mező | Típus / hossz | Leírás |
| id | bigint(20) UNSIGNED | felhsaználó egyedi azonosítója automatikusan kapja |
| name | varchar(255) utf8mb4\_unicode\_ci | felhasználó megnevezése |
| email | varchar(255) utf8mb4\_unicode\_ci | felhasználó e-mail címe |
| email\_verified\_at | timestamp | regisztráció visszaigazolás dátuma. ha nincs visszaigazolva, akkor NULL |
| password | varchar(255) utf8mb4\_unicode\_ci | felhasználó jelszava letitkosítva |
| admin | int(11) | felhasználó jogosultsága |
| remember\_token | varchar(255) utf8mb4\_unicode\_ci | nincs alkalmazva |
| created\_at | timestamp | létrehozás dátuma és pontos ideje |
| updated\_at | timestamp | módosítás dátuma és pontos ideje |
| deleted\_at | timestamp | törlés dátuma és pontos ideje |

## Fejlesztői környezet

XAMPP

-Adatbázis kezelése a mysql nevű klienssel.

-PHP

Visual Studio Code

Laravel