```
https://www.youtube.com/watch?v=nqngC7WMZzo&list=PLVB63JG1YVK4pyhFPvaJJqIU8LmEzEsle&index=13
```
```0:24:00```
 # 0 Előtte kell egy ```XAMPP``` és egy ```Composer``` install

1. Indítsd el a XAMPP Contorl Panelt és ebből az Apache-ot és a MySQL-t és a PHPmyadmint!

```0:18:40```
 # 1 feladat: Készítsen egy üres projektet a választott backend keretrendszer segítségével. A projekt neve legyen ENaploBackend!
```0:19:05```
1. Az adott mappában, projekt neve legyen ```ENaploBackend``` így ```c:\vue-gyak-enaplo``` kiadjuk a CLI parancsot ```composer create-project laravel/laravel ENaploBackend```

2. Lépj be a projektmappába ```cd ENaploBackend```

4. Jobb klikk a mappán, helyi menü, Open Folder as PhpStorm project

5. .env file szerkesztése, és beállítása a MySQL-ra, és az adatbázis neve: ```enaplo```

```
DB_CONNECTION=mysql
 DB_HOST=127.0.0.1
 DB_PORT=3306
 DB_DATABASE=enaplo
 DB_USERNAME=root
 DB_PASSWORD=
```

```0:30:10```

6. Ha már az env fájlban beállítottuk a MySQL szerver hozzáférést, és fut a MySQL szerver és kiadjuk a: ```php artisan migrate``` parancsot, és nyomunk egy YES-t, akkor megcsinálja nekunk az adatbázist!
(Ha így nem meg, csinálj egy adatbázist ```enaplo``` néven, pl. php myadninnal ```http://localhost/phpmyadmin/``` vagy heide-val.)
7. ellenőrizd pl. a ```http://localhost/phpmyadmin/``` -al, hogy lett-e adatbáis!

```0:41:30```

 # 2 feladat: Készítse el a „students” és a „grades” táblákat az alábbiak szerint.

```
    a.  A „students” táblát adatbázis migráció segítségével hozza létre! (kiegészíthető a keretrendszer által megkövetelt oszlopokkal)
        - id: egész szám, a tanuló  azonosítója, elsődleges kulcs, automatikusan kap értéket
        - name: szöveg mely a tanuló nevét tárolja, maximális hossza 255 karakter lehet
        - active: boolean, az alapértelmezett értéke legyen „true”, azt adja meg, hogy a tanulónak aktív jogviszonya van-e
        - eduid: szöveg, 11 karakteres, a tanuló oktatási azonosítóját tárolja

    b.  A „grades” táblát adatbázismigráció segítségével hozza létre! (kiegészíthető a keretrendszer által megkövetelt oszlopokkal)
        - id: egész szám, a osztályzat azonosítója, elsődleges kulcs, automatikusan kap értéket
        - student_id: idegen kulcs, a tanulóra, akihez a jegy tartozik
        - grade: egész szám, értékelés
        - weight: egész szám, értékelés súlya
        - comment: szöveg, 50 karakter, az értékelés magyarázata
```
```0:42:54```
8. Parancssorból adjuk ki a: ```php artisan make:model Student --controller --resource --requests --migration```
Ekkor létrejön a:

```
c:\vue-gyak-enaplo\ENaploBackend\app\Models\Student.php
c:\vue-gyak-enaplo\ENaploBackend\app\Http\Controllers\StudentController.php
c:\vue-gyak-enaplo\ENaploBackend\app\Http\Requests\StoreStudentRequest.php
c:\vue-gyak-enaplo\ENaploBackend\app\Http\Requests\UpdateStudentRequest.php
c:\vue-gyak-enaplo\ENaploBackend\database\migrations\2024_05_20_011822_create_students_table.php
```

9. vagy inkább legyen az egyszerűbb: ```php artisan make:model Student -a```, ez csinál mást is, de egyszerűbb a parancs!!! - ez kell!!!

```
   INFO  Model [C:\vue-gyak-enaplo\ENaploBackend\app\Models\Student.php] created successfully.
   INFO  Factory [C:\vue-gyak-enaplo\ENaploBackend\database\factories\StudentFactory.php] created successfully.
   INFO  Migration [C:\vue-gyak-enaplo\ENaploBackend\database\migrations/2024_05_20_013801_create_students_table.php] created successfully.
   INFO  Seeder [C:\vue-gyak-enaplo\ENaploBackend\database\seeders\StudentSeeder.php] created successfully.
   INFO  Request [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Requests\StoreStudentRequest.php] created successfully.
   INFO  Request [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Requests\UpdateStudentRequest.php] created successfully.
   INFO  Controller [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Controllers\StudentController.php] created successfully.
   INFO  Policy [C:\vue-gyak-enaplo\ENaploBackend\app\Policies\StudentPolicy.php] created successfully.
```

Majd kapunk student seede PHP-t, amit be kell másolni!!!
```0:49:29```
Ide kell majd egy ilyen:
```
<?php

namespace Database\Seeders;

use App\Models\User;
// use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     */
    public function run(): void
    {
        $this->call([
            StudentSeender::class
        ]);
    }
}
```
Arra kell nagyon figyelni, hogy a létrehozott migration neve úgy legyen létrehozva, ahogy a feladatban meg van adva!!!

10. Ebbe a fájlba: ```c:\vue-gyak-enaplo\ENaploBackend\database\migrations\2024_05_20_013801_create_students_table.php``` bele kell tenni az adott mezőket!
```
    a.  A „students” táblát adatbázis migráció segítségével hozza létre! (kiegészíthető a keretrendszer által megkövetelt oszlopokkal)
        - id: egész szám, a tanuló  azonosítója, elsődleges kulcs, automatikusan kap értéket
        - name: szöveg mely a tanuló nevét tárolja, maximális hossza 255 karakter
```
```

lehet
- active: boolean, az alapértelmezett értéke legyen „true”, azt adja meg, hogy a tanulónak aktív jogviszonya van-e
- eduid: szöveg, 11 karakteres, a tanuló oktatási azonosítóját tárolja

```
Így lesz benne!!!
```
    public function up(): void
    {
        Schema::create('students', function (Blueprint $table) {
            $table->id();
            $table->string('name', 255);
            $table->boolean('active')->default(true);
            $table->string('eduid',11);
            $table->timestamps();
        });
    }
```

11. Jön a második tábla ```php artisan make:model Grade -a```
Figyelni kell, hogy nem többesszámban írjuk a tábla nevét, és NAGYBETŰVEL!

```
    b. A „grades” táblát adatbázismigráció segítségével hozza létre! (kiegészíthető a keretrendszer által megkövetelt oszlopokkal)
        - id: egész szám, a osztályzat azonosítója, elsődleges kulcs, automatikusan kap értéket
        - student_id: idegen kulcs, a tanulóra, akihez a jegy tartozik
        - grade: egész szám, értékelés
        - weight: egész szám, értékelés súlya
        - comment: szöveg, 50 karakter, az értékelés magyarázata
```
ekkor létrejönnek az alábbi fájlok:
```
   INFO Model [C:\vue-gyak-enaplo\ENaploBackend\app\Models\Grade.php] created successfully.
   INFO Factory [C:\vue-gyak-enaplo\ENaploBackend\database\factories\GradeFactory.php] created successfully.
   INFO Migration [C:\vue-gyak-enaplo\ENaploBackend\database\migrations/2024_05_20_015800_create_grades_table.php] created successfully.
   INFO Seeder [C:\vue-gyak-enaplo\ENaploBackend\database\seeders\GradeSeeder.php] created successfully.
   INFO Request [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Requests\StoreGradeRequest.php] created successfully.
   INFO Request [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Requests\UpdateGradeRequest.php] created successfully.
   INFO Controller [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Controllers\GradeController.php] created successfully.
   INFO Policy [C:\vue-gyak-enaplo\ENaploBackend\app\Policies\GradePolicy.php] created successfully.
```

Ebbe a fájlba: ```c:\vue-gyak-enaplo\ENaploBackend\database\migrations\2024_05_20_015800_create_grades_table.php``` bele kell tenni az adott mezőket!

```
    public function up(): void
    {
        Schema::create('grades', function (Blueprint $table) {
            $table->id();
            $table->foreignIdFor(\App\Models\Student::class);
            $table->integer('grade');
            $table->integer('weight');
            $table->string('comment',50);
            $table->timestamps();
        });
```
```

```
    }
```

```0:57:16```

12. Futtassuk a CLI: ```php artisan migrate``` parancsot, és létrejönnek a táblák az adatbázisban (Le is fut!!!)!

```
   INFO  Running migrations.

  2024_05_20_013801_create_students_table ..................... 54.22ms DONE
  2024_05_20_015800_create_grades_table ....................... 31.55ms DONE
```

13. Kimaradt a két tábla kapcsolata, ezért módosítom a ```c:\vue-gyak-enaplo\ENaploBackend\database\migrations\2024_05_20_015800_create_grades_table.php``` fájlt! Bele kell tenni a ```->constrained()``` függvényt!

```
    public function up(): void
    {
        Schema::create('grades', function (Blueprint $table) {
            $table->id();
            $table->foreignIdFor(\App\Models\Student::class)->constrained();
            $table->integer('grade');
            $table->integer('weight');
            $table->string('comment',50);
            $table->timestamps();
        });
    }
```

14. Majd futtatni egy migration rollbacket! (visszavonni a migrációt, ami futtatja a ```c:\vue-gyak-enaplo\ENaploBackend\database\migrations\2024_05_20_015800_create_grades_table.php``` fájlban lévő ```down()``` függvényt, ezzel törli az adatbázisból a táblákat, amiket megadtam migration-ként!
A ```CLI``` parancs: ```php artisan migrate:rollback```
```
   INFO  Rolling back migrations.

  2024_05_20_015800_create_grades_table ..................... 165.39ms DONE
  2024_05_20_013801_create_students_table ................... 32.98ms DONE
```

```
public function down(): void
    {
        Schema::dropIfExists('grades');
    }
```

15. Futtassuk a CLI: ```php artisan migrate``` parancsot, és létrejönnek a táblák az adatbázisban (Le is fut!!!)!
(Vizsgán töröljük nyugodtan akár az adatbázist és futtassuk újra a migrációt! - ha nem jut eszünkbe. Éles adatbázison, nem csinálunk ilyet!)

```
   INFO  Running migrations.

  2024_05_20_013801_create_students_table ..................... 58.42ms DONE
  2024_05_20_015800_create_grades_table ..................... 439.30ms DONE
```

16. Ekkor újra elkészül a két tábla, és az adatbázis ```migrations``` táblájába újra bekerülnek, és látszik, a ```batch``` oszlopban, hogy hányszor készültek el.

Teljes szövegek

id	migration	batch
1	0001_01_01_000000_create_users_table	1
2	0001_01_01_000001_create_cache_table	1
3	0001_01_01_000002_create_jobs_table	1
```

6   2024_05_20_013801_create_students_table 2
7   2024_05_20_015800_create_grades_table   2

17. Ha mégis törölnéd az adatbázist, akkor phpMyAdmin/műveletek/adatbázis eldobása, YES ```http://localhost/phpmyadmin/index.php?route=/database/operations&db=enaplo```

18. Az egész adatbázis újragenerálása ```CLI```: ```php artisan migrate```
```
c:\vue-gyak-enaplo\ENaploBackend>php artisan migrate

   WARN  The database 'enaplo' does not exist on the 'mysql' connection.

  Would you like to create it? (yes/no) [yes]
 yes

   INFO  Preparing database.

  Creating migration table ..................................... 29.37ms DONE

   INFO  Running migrations.

  0001_01_01_000000_create_users_table ..................... 164.66ms DONE
  0001_01_01_000001_create_cache_table ..................... 108.11ms DONE
  0001_01_01_000002_create_jobs_table ...................... 187.28ms DONE
  2024_05_20_013801_create_students_table ...................... 29.50ms DONE
  2024_05_20_015800_create_grades_table ..................... 146.65ms DONE
```

 # 3 feladat: Illessze be a megfelelő helyre a forrásként kapott Seeder fájlokat valamint a megfelelő sorrendben hivatkozzon rájuk a DatabaseSeeder.php fájlban.

19. Seederek berakása a ``` c:\vue-gyak-enaplo\ENaploBackend\database\seeders\ ``` mappába. (Ha van benne már legyártott seederem pl. ```GradeSeeder.php``` és/vagy ```StudentSeeder.php``` , akkor ezeket töröljük!) Vagy felülírjuk!

20. A ```DatabaseSeeder.php``` fájlba behivatkozom a két seedert, először azt ami nem hivatkozik másik táblára, majd ami hivatkozik!!!
```
    public function run(): void
    {
        $this->call([
            StudentSeeder::class,
            GradeSeeder::class
        ]);
    }
```

21. Majd kiadom a ```CLI:``` ```php artisan db:seed``` parancsot!!! és feltöltődnek az adott táblák adattal!!!

```
c:\vue-gyak-enaplo\ENaploBackend>php artisan db:seed

   INFO  Seeding database.

  Database\Seeders\StudentSeeder ................................ RUNNING
  Database\Seeders\StudentSeeder ............................... 167 ms DONE

  Database\Seeders\GradeSeeder .................................. RUNNING
  Database\Seeders\GradeSeeder ................................. 98 ms DONE
```

22. Ha nem sikerül a vizsgán a SEED-elés, mert hibás a kód, akkor töröljük az adatbázist, és újra futtassuk a migrationst, és a seed-et!
```http://localhost/phpmyadmin/index.php?route=/database/operations&db=enaplo```
```CLS:``` ```php artisan migrate```
```CLI:``` ```php artisan db:seed```

 # 4 feladat: Készítse el az alábbi API végpontokat! 1:42:00
```         - Minden végpont JSON formátumban adja vissza a kimenetet. Hiba esetén jelezze a hiba okát, HTTP státusz kóddal és egy JSON-ben a hiba részleteivel

```
            (a JSON-t generálhatja a keretrendszer, vagy a fejlesztő is, amennyiben
            megfelel a megadott paramétereknek)
                    - GET /api/grades
            Adja vissza azokat a jegyeket melyek beírhatóak a rendszerbe
            {
                "data": [
                    {"grade" :1},
                    {"grade" :2},
                    {"grade" :3},
                    {"grade" :4},
                    {"grade" :5},
                ],
                "success": true,
                "message": ""
            }
        - GET /api/students
            Adja vissza az összes tanulót és azok adatait. Az eredmény egy tanuló
            objektumokból álló lista legyen az alábbi formában:
            {
                "data": [
                    {
                        "id":1,
                        "name":"Pista",
                        "eduid":"7777777777"
                    },
                ],
                "success": true,
                "message": ""
            }
        - GET /api/students/:id
            Adja vissza egy tanuló összes adatát. Az eredményt egy objektumként adja
            vissza.
            A kérésben az „:id” paraméterként szerepel, melyben egy tanuló „id”-t ad
            át a hívó.
            {
                "data": [
                    {
                        "id":1,
                        "name":"Pista",
                        "eduid":"7777777777",
                        "grades":[
                            "grade":?,
                            "weight":?,
                            "comment":?
                        ]
                    },
                ],
                "success": true,
                "message": ""
            }
        - POST /api/grades
            A kérés törzse egy JSON objektum, mely tartalmazz egy új osztályzat
            adatait, melyek a következők:
            {
                "student_id" : 1
                "grade" : 2
                "weight" : 1
                "comment" : "Új osztályzat"
            }
```

23. Kell egy API file ```CLI:``` ```php artisan install:api``` (A 11-es laravelben nincs ez a file), legalább 1 perc, mire lefut! A végén csinál egy migrations fájlt, de nem kell vele foglalkozni!
(c:\vue-gyak-enaplo\ENaploBackend\database\migrations\2024_05_22_104101_create_personal_access_tokens_table.php)

```
c:\vue-gyak-enaplo\ENaploBackend\routes\api.php
```

24. Ami benne van egy bejegyzés, azt töröljük!
```
```

````
Route::get('/user', function (Request $request) {
    return $request->user();
})->middleware('auth:sanctum');
```
25. Vegyünk fel egy új bejegyzést a ```c:\vue-gyak-enaplo\ENaploBackend\routes\api.php``` -ba!
```
use App\Http\Controllers\GradeController;

Route::apiResource('grades', GradeController::class)
    ->only(['index']);
```
26. Nézzük meg, hogy bejegyződött-e! CMD: ```php artisan rout:list```, és ha szerepel benne az ```api/grades```, akkor OK!

```
c:\vue-gyak-enaplo\ENaploBackend>php artisan rout:list

  GET|HEAD   / ..........................................................
  POST       _ignition/execute-solution ignition.executeSolution › Spatie…
  GET|HEAD   _ignition/health-check ignition.healthCheck › Spatie\Laravel…
  POST       _ignition/update-config ignition.updateConfig › Spatie\Larav…
  GET|HEAD   api/grades ..... grades.index › GradeController@index
  GET|HEAD   sanctum/csrf-cookie sanctum.csrf-cookie › Laravel\Sanctum › …
  GET|HEAD   up .........................................................

                                                    Showing [7] routes
```

```1:52:00```

27. Valósítsuk meg a működését! Kell egy USE és egy return az index() függvénybe! (Itt bele kerül kézzel az adat és a ```status: 200```-as státuszkód is!!!! (ez az OK-t jelenti)
```c:\vue-gyak-enaplo\ENaploBackend\app\Http\Controllers\GradeController.php```-ben.

```
    public function index()
    {
        return Response::json(
            [
                "data" => [
                    ["grade"=>1],
                    ["grade"=>2],
                    ["grade"=>3],
                    ["grade"=>4],
                    ["grade"=>5],
                ],
                "success" => true,
                "message" => ""
            ], 200
        );
    }

```

28. Megnézzük, hogy működi-e!  CLI: ```PHP artisan serve```
```
c:\vue-gyak-enaplo\ENaploBackend>php artisan serve

   INFO  Server running on [http://127.0.0.1:8000].

  Press Ctrl+C to stop the server

```

29. Böngészőben megnyitjuk a: ```http://127.0.0.1:8000/api/grades``` és az első kész is van!

30. Második API végpont: ```GET /api/students``` , a routes\api.php-ben felveszünk egy új bejegyzést!
```
````

c:\vue-gyak-enaplo\ENaploBackend\routes\api.php
```
fájlban, a:
```
use App\Http\Controllers\StudentController;

Route::apiResource('students', StudentController::class)
    ->only(['index']);
```

31. Student Contorller index függvényét kell létrehozni!!!!

32. Ő írja legy, hogy 1 student hogy jelenjen meg: Célszerű parancssorból: CLI: ```php artisan make:resource StudentResource```

```
c:\vue-gyak-enaplo\ENaploBackend>php artisan make:resource StudentResource

   INFO  Resource
   [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\StudentResource.php] created
   successfully.
```

33. A sok student hogyan legyen formázva, Célszerű parancssorból: CLI: ```php artisan make:resource StudentCollection```

```
c:\vue-gyak-enaplo\ENaploBackend>php artisan make:resource StudentCollection

   INFO  Resource collection
   [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\StudentCollection.php]
   created successfully.
```

34. Felvesszük, hogy hogyan néz ki egy student (a return mögötti részt töröljük és oda írjuk!):

```
C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\StudentResource.php
```
ide:

```
    public function toArray(Request $request): array
    {
        return [
            'id' => $this ->id,
            'name' => $this ->name,
            'eduid' => $this -> eduid
        ];
    }
```

35. Megoldjuk, hogy kiíratódjanak a Studentek a Collectionban! (a return mögötti részt töröljük és oda írjuk!)

```
C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\StudentCollection.php
```
ide:

```
    public function toArray(Request $request): array
    {
        return [
            'data' => $this->collection,
            'success' => true,
            'message' => ''
        ];
    }
```
```

36. És még be kell hivatkozni a StudentContorllert
``` c:\vue-gyak-enaplo\ENaploBackend\app\Http\Controllers\StudentController.php```
Ennek

```
    public function index()
    {
        return new StudentCollection(Student::all());
    }
```

37. Harmadik API végpont: ```GET /api/students/:id``` , a routes\api.php-ben felveszünk egy új bejegyzést!
```
c:\vue-gyak-enaplo\ENaploBackend\routes\api.php
```
fájlban, felvesszük a index mellé a show-t is!:

```
Route::apiResource('students', StudentController::class)
    ->only(['index', 'show']);
```

38. Majd a
```c:\vue-gyak-enaplo\ENaploBackend\app\Http\Controllers\StudentController.php```
fájlba megírjuk a SHOW függvényt:
```
    public function show(Student $student)
    {
        return new StudentResource($student);
    }
```
39. De ez nem jó, mert nem ez a feladat!!!! Hanem létrehozunk egy
```StudentDetailedResource.php``` fájlt!
CLI: ```php artisan make:resource StudentDetailedResource ```

```
PS C:\vue-gyak-enaplo\ENaploBackend> php artisan make:resource StudentDetailedResource

   INFO  Resource
   [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\StudentDetailedResource.php]
   created successfully.

```
40. És ezt használjuk a StudentContorller.php SHOW függvényében, tehát 38 ki, 40 -be!!!!
```
    public function show(Student $student)
    {
        return new StudentDetailedResource($student);
    }
```

41. A
```C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\StudentDetailedResource.php```
ben az alapműködés helyett felépítjük azt, amit a feladat kér!
```StudentDetailedResource.php```
fájlban:
```
public function toArray(Request $request): array
    {
        return [
            'data' => [
                'id' => $this->id,
                'name' => $this->name,
                'eduid' => $this->eduid,
                'grades' => $this->grades
            ],
            'success' => true,
            'message' => ''
        ];
    }
```
 42. de ez még nem jó, mert a ```grades```-t nem tölti be, ekkor rakunk bele elgy
```

lekérdezést, ami még mindíg nem elég jó:

```
  public function toArray(Request $request): array
   {
       return [
           'data' => [
               'id' => $this->id,
               'name' => $this->name,
               'eduid' => $this->eduid,
               'grades' => Grade::where('student_id', $this->id)->get()
           ],
           'success' => true,
           'message' => ''
       ];
   }
```

43. Ezért létrehozunk egy ```GradeResource.php```-t CLI: ```php artisan make:resource GradeResource```

```
PS C:\vue-gyak-enaplo\ENaploBackend> php artisan make:resource GradeResource

   INFO  Resource
   [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\GradeResource.php] created
   successfully.
```

44. Meg akkor már kell egy GradeCollection.php is kell!!!
CLI: ```php artisan make:resource GradeCollection```

```
PS C:\vue-gyak-enaplo\ENaploBackend> php artisan make:resource GradeCollection

   INFO  Resource collection
   [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\GradeCollection.php] created
   successfully.

```

45. ezekben definiáljuk, hogy hogyan nézzenek ki!!!! A ```C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\GradeResource.php```-ban ezt írjuk:

```
   public function toArray(Request $request): array
    {
        return [
            'grade' => $this->grade,
            'weight'  => $this->weight,
            'comment'  => $this->comment
        ];
    }
```

46. Majd megjelenítjük egy a ```GradeCollection.php```-ban!

```
C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\GradeCollection.php``` - ott ezt írjuk:
```

```
47. Javítjuk a ```C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\StudentDetailedResource.php``` ben az alapműködés helyett felépítjük azt, amit a feladat kér!
```StudentDetailedResource.php```
fájlban:
```

```
return [
            'data' => [
                'id' => $this->id,
                'name' => $this->name,
                'eduid' => $this->eduid,
                'grades' => new GradeCollection(Grade::where('student_id',
                $this->id)->get())
```
```

```
            ],
            'success' => true,
            'message' => ''
        ];

```
48. De ennél még van egy egyszerűbb és szebb megoldás is, ha ezeket a kapcsolatokat felvesszük a modelleknél, akkor jobb lesz!!!
A ```c:\vue-gyak-enaplo\ENaploBackend\app\Models\Student.php```-be felveszek egy grades() függvényt:
```
class Student extends Model
{
    use HasFactory;

    public function grades()
    {
        return $this->hasMany(Grade::class);
    }
}
```

49. Negyedik API végpont: ```POST /api/grades
            A kérés törzse egy JSON objektum, mely tartalmazz egy új osztályzat adatait, melyek a következők:
            {
                "student_id" : 1
                "grade" : 2
                "weight" : 1
                "comment" : "Új osztályzat"
            }
```

50. Vegyünk fel egy új bejegyzést a Felvesszük a```'store'``` kiegészítést a ``` c:\vue-gyak-enaplo\ENaploBackend\routes\api.php ``` -ba!

```
use App\Http\Controllers\GradeController;

Route::apiResource('grades', GradeController::class)
    ->only(['index', 'store']);
```
51. A ```GradeController.php```-ben van egy ```Store()``` függvény, ami a ```StoreGradeRequest.php```-re hivatkozik, ott átírjuk az ```authorize() függvény return-jét true-ra```!


```GradeController.php```-file:
```    public function store(StoreGradeRequest $request)
    {
        //
    }

```

```StoreGradeRequest.php```-file:
```
    public function authorize(): bool
    {
        return true;
    }
```

52. ```2:53:20``` A ```StoreGradeRequest.php``` -ben lévő ```rules()``` függvényében megírjuk a következőt!
```c:\vue-gyak-enaplo\ENaploBackend\app\Http\Requests\StoreGradeRequest.php```-ben:

```
    public function rules(): array
    {
        return [
            'student_id' => 'required',
```

```
            'grade'  => 'required',
            'weight'  => 'required',
            'comment'  => 'required'
        ];
    }
```

53. Hogy lehessen megadni tömegesen adatokat, bele kell írni a ```c:\vue-gyak-enaplo\ENaploBackend\app\Models\Grade.php``` -ba, hogy:

```
class Grade extends Model
{
    use HasFactory;

    protected $guarded=[];
}
```

54. A ```c:\vue-gyak-enaplo\ENaploBackend\app\Http\Controllers\GradeController.php``` -ben megírjuk a```Store()``` függvényt!

```
public function store(StoreGradeRequest $request)
    {
        $data = $request->validated();

        $grade = Grade::create($data);

        return Response::json([
            'data' => new GradeResource($grade),
                    'success' => true,
                    'message' => 'Created'
            ],201
        );
    }
```

55. A StoreGradeRequest.php-be írun egy felüldefiniált ```failedValidation()``` függvényt, hogy tesztelhessük, hogy miért nem jó a progink.

```
use Illuminate\Support\Facades\Response;


    protected function failedValidation(Validator $validator)
    {
        throw new HttpResponseException(
          Response::json([
                'data' => $validator->errors(),
                'success' => false,
                'message' => 'Validation error'
          ],400)
        );
    }
```

56. Nyisd meg a PostMAN-t, és

```
post, http://localhost:8000/api/grades


Body, raw, json

{
    "student_id":1,
    "grade":2,
    "weight":1,
    "comment": "Új osztályzat"
}
SEND

Ha minden jó, akkor beírja ezt a json-t az adatbázis grades táblájába!
http://localhost/phpmyadmin/index.php?route=/sql&pos=0&db=enaplo&table=grades
```

'grades' - tábla
id student_id grade weight comment created_at updated_at
1 1 4 1 Órai munka 2024-05-21 23:49:49 2024-05-21 23:49:49
2 1 3 2 Laravel Témazáró 2024-05-21 23:49:49 2024-05-21 23:49:49
3 1 5 1 Felelet 2024-05-21 23:49:49 2024-05-21 23:49:49
4 2 2 1 Órai munka 2024-05-21 23:49:49 2024-05-21 23:49:49
5 2 5 2 Laravel Témazáró 2024-05-21 23:49:49 2024-05-21 23:49:49
6 1 2 1 Új osztályzat 2024-05-22 17:43:37 2024-05-22 17:43:37

```

57. Másik, hogy ha ```http://127.0.0.1:8000/api/students/3``` - ra megyünk mivel nincs ilyen id a táblázatban 404-es hibával tér vissza. Erre íruk megoldást az api.php-ben! ```->missing(function(Request $request){return Response::json```-t írunk mindkét route-hoz!

```
<?php
use App\Http\Controllers\GradeController;
use App\Http\Controllers\StudentController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Response;
use Illuminate\Support\Facades\Route;

Route::apiResource('grades', GradeController::class)
    ->only(['index', 'store'])
    ->missing(function(Request $request){
        return Response::json([
            'data' => null,
            'success' => false,
            'message' => 'Grade not exists'
        ],404);
    });

Route::apiResource('students', StudentController::class)
    ->only(['index', 'show'])
    ->missing(function(Request $request){
        return Response::json([
            'data' => null,
            'success' => false,
            'message' => 'Student not exists'
        ],404);
    });

```

58. Nyisd meg a PostMAN-t, és
```
get, http://127.0.0.1:8000/api/students/3

Body

SEND

EZt adja vissza:
{
    "data": null,
    "success": false,
    "message": "Student not exists"
}
```

# 5.feladat A POST /api/types végpont ellenőrizze, hogy a bemeneti információk megfelelőek-e az alábbi elven.
```

```      - student_id: kötelező kitölteni és csak olyan értéket fogad el, mely a „group” táblában létező id.
        - grade: kötlező és egész szám
        - weight: kötelező és egész szám
        - comment: kötelező és szöveg

        Validációs probléma esetén JSON formátumú válaszban jelezze azt a backend a frontend felé.
        Amennyiben az adatok megfelelőek, vegye fel adatbázisba a osztályzatot és adja azt vissza, 201-es státusz kóddal és „Created” üzenettel.
```

59. ```3:26:00``` A ```StoreGradeRequest.php``` -ben lévő ```rules()``` függvényében felvesszük a típusokat is!
```c:\vue-gyak-enaplo\ENaploBackend\app\Http\Requests\StoreGradeRequest.php```-ben:

```
    public function rules(): array
    {
        return [
            'student_id' => 'required|exists:students,id',
            'grade'  => 'required|integer',
            'weight'  => 'required|integer',
            'comment'  => 'required|string',
        ];
    }
```

60. Próbáljuk ki, hogy működik-e! Nyisd meg a PostMAN-t, és
```
post, http://localhost:8000/api/grades


Body, raw, json

{
    "student_id":10,
    "grade":"2a",
    "weight":"1e",
    "comment": "Hiba"
}
SEND

EZ a válasz jön vissza!!!!
{
    "data": {
        "student_id": [
            "The selected student id is invalid."
        ],
        "grade": [
            "The grade field must be an integer."
        ],
        "weight": [
            "The weight field must be an integer."
        ]
    },
    "success": false,
    "message": "Validation error"
}

```

 # 6 Készítsen nézetet grades.blade.php néven, ahol a jegyek jelennek meg a mintának megfelelően meg jeleníti számozatlan felsorolásban és a hozzájuk tartozó tanulókat is szintén egy számozatlan felsorolásban.
```
    A nézet a "GET /" végpont betöltésekor jelenjen meg.
```
```
Segítségül használhatók:
    internetkapcsolat, amely használható:
        általános keresésre;
```
```

```
        online dokumentáció elérésére;
        projekt keretek létrehozására, csomagok telepítésére.
```

61. Lecseréljük a ```c:\vue-gyak-enaplo\ENaploBackend\routes\web.php```-ben a defau lt view-et,

```
<?php

use Illuminate\Support\Facades\Route;

Route::get('/', function () {
    return view('grades ');
});
```

62. Csinálunk egy view-et, ```c:\vue-gyak-enaplo\ENaploBackend\resources\views\grades.blade.php```-fájlt! ebbe csinálunk egy HTML-t!

```
<!doctype html>
<html lang="hu">
<head>
    <title>Enaplo</title>
</head>
<body>
    <h1>Osztályzatok</h1>
    <hr>
    <ul>
        @foreach($grades as $grade)
            <li>{{$grade->grade}} (x{{$grade->weight}}) [{{$grade->comment}}]</li>
       @endforeach
    </ul>
</body>
</html>
```

63. Majd ezt kibővítjük, hogy a tanulók nevei is benne legyenek, de előtte kell egy Grade->Student hivatkozás is, mert hiába vesszük fel a mév mezőt, nem jelennek meg a nevek!

```Grade.php```:
```
    public function student()
    {
        return $this->belongsTo(Student::class);
    }
```

64. Betesszük a neveket is a jegyek alá a HTML-be

```

<!doctype html>
<html lang="hu">
<head>
    <title>Enaplo</title>
</head>
<body>
    <h1>Osztályzatok</h1>
    <hr>
    <ul>
        @foreach($grades as $grade)
            <li>{{$grade->grade}} (x{{$grade->weight}}) [{{$grade->comment}}]
                <ul>
                    <li>{{$grade->student->name}}</li>
                </ul>
            </li>
       @endforeach
    </ul>
</body>
</html>
```
```

```
```