

```
1
2   ```
3 https://www.youtube.com/watch?v=nqngC7WMZzo&list=PLVB63JG1YVK4pyhFPvaJJqIU8LmEzEsle&index=13
4   ```
5   ```0:24:00```
6   # 0 Előtte kell egy ```XAMPP``` és egy ```Composer``` install
7
8   1. Indítsd el a XAMPP Contorl Panelt és ebből az Apache-ot és a MySQL-t és a
   PHPMyadmint!
9
10  ```0:18:40```
11  # 1 feladat: Készítsen egy üres projektet a választott backend keretrendszer
   segítségével. A projekt neve legyen ENaploBackend!
12  ```0:19:05```
13  1. Az adott mappában, projekt neve legyen ```ENaploBackend``` így
   ```c:\vue-gyak-enaplo``` kiadjuk a CLI parancsot ```composer create-project
 laravel/laravel ENaploBackend```
14
15 2. Lépj be a projektmappába ```cd ENaploBackend```
16
17 4. Jobb klikk a mappán, helyi menü, Open Folder as PhpStorm project
18
19 5. .env file szerkesztése, és beállítása a MySQL-ra, és az adatbázis neve:
   ```enaplo```
20
21  ```
22  DB_CONNECTION=mysql
23  DB_HOST=127.0.0.1
24  DB_PORT=3306
25  DB_DATABASE=enaplo
26  DB_USERNAME=root
27  DB_PASSWORD=
28  ```
29
30  ```0:30:10```
31
32  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 nekünk az adatbázist!
33  (Ha így nem meg, csinálj egy adatbázist ```enaplo``` néven, pl. php myadninnal
   ```http://localhost/phpmyadmin/``` vagy heide-val.)
34 7. ellenőrizd pl. a ```http://localhost/phpmyadmin/``` -al, hogy lett-e adatbáis!
35
36 ```0:41:30```
37
38 # 2 feladat: Készítse el a „students” és a „grades” táblákat az alábbiak szerint.
39
40 ```
41 a. A „students” táblát adatbázis migráció segítségével hozza létre!
 (kiegészíthető a keretrendszer által megkövetelt oszlopokkal)
42 - id: egész szám, a tanuló azonosítója, elsődleges kulcs, automatikusan kap
 értéket
43 - name: szöveg mely a tanuló nevét tárolja, maximális hossza 255 karakter
 lehet
44 - active: boolean, az alapértelmezett értéke legyen „true”, azt adja meg,
 hogy a tanulónak aktív jogviszonya van-e
45 - eduid: szöveg, 11 karakteres, a tanuló oktatási azonosítóját tárolja
46
47 b. A „grades” táblát adatbázismigráció segítségével hozza létre! (kiegészíthető
 a keretrendszer által megkövetelt oszlopokkal)
48 - id: egész szám, a osztályzat azonosítója, elsődleges kulcs, automatikusan
 kap értéket
49 - student_id: idegen kulcs, a tanulóra, akihez a jegy tartozik
50 - grade: egész szám, értékelés
51 - weight: egész szám, értékelés súlya
52 - comment: szöveg, 50 karakter, az értékelés magyarázata
53 ```
54 ```0:42:54```
55 8. Parancssorból adjuk ki a: ```php artisan make:model Student --controller
 --resource --requests --migration```
56 Ekkor létrejön a:
```

```

57 ```
58 c:\vue-gyak-enaplo\ENaploBackend\app\Models\Student.php
59 c:\vue-gyak-enaplo\ENaploBackend\app\Http\Controllers\StudentController.php
60 c:\vue-gyak-enaplo\ENaploBackend\app\Http\Requests\StoreStudentRequest.php
61 c:\vue-gyak-enaplo\ENaploBackend\app\Http\Requests\UpdateStudentRequest.php
62 c:\vue-gyak-enaplo\ENaploBackend\database\Migrations\2024_05_20_011822_create_students_
63 _table.php
64 ```
65 9. vagy inkább legyen az egyszerűbb: ```php artisan make:model Student -a``` , ez
66 csinál mást is, de egyszerűbb a parancs!!! - ez kell!!!
67 ```
68 INFO Model [C:\vue-gyak-enaplo\ENaploBackend\app\Models\Student.php] created
69 successfully.
70 INFO Factory
71 [C:\vue-gyak-enaplo\ENaploBackend\database\factories\StudentFactory.php] created
72 successfully.
73 INFO Migration
74 [C:\vue-gyak-enaplo\ENaploBackend\database\Migrations\2024_05_20_013801_create_stud
75 ents_table.php] created successfully.
76 INFO Seeder [C:\vue-gyak-enaplo\ENaploBackend\database\seeders\StudentSeeder.php]
77 created successfully.
78 INFO Request
79 [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Requests\StoreStudentRequest.php]
80 created successfully.
81 INFO Request
82 [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Requests\UpdateStudentRequest.php]
83 created successfully.
84 INFO Controller
85 [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Controllers\StudentController.php]
86 created successfully.
87 INFO Policy [C:\vue-gyak-enaplo\ENaploBackend\app\Policies\StudentPolicy.php]
88 created successfully.
89 ```
90 Majd kapunk student seeder PHP-t, amit be kell másolni!!!
91 ```0:49:29```
92 Ide kell majd egy ilyen:
93 ```
94 <?php
95
96 namespace Database\Seeders;
97
98 use App\Models\User;
99 // use Illuminate\Database\Console\Seeds\WithoutModelEvents;
100 use Illuminate\Database\Seeder;
101
102 class DatabaseSeeder extends Seeder
103 {
104 /**
105 * Seed the application's database.
106 */
107 public function run(): void
108 {
109 $this->call([
110 StudentSeeder::class
111]);
112 }
113 }
114 ```
115 Arra kell nagyon figyelni, hogy a létrehozott migration neve úgy legyen létrehozva,
116 ahogy a feladatban meg van adva!!!
117
118 10. Ebbe a fájlba:
119 ```c:\vue-gyak-enaplo\ENaploBackend\database\Migrations\2024_05_20_013801_create_stude
120 nts_table.php``` bele kell tenni az adott mezőket!
121 ```
122 a. A „students” táblát adatbázis migráció segítségével hozza létre!
123 (kiegészíthető a keretrendszer által megkövetelt oszlopokkal)
124 - id: egész szám, a tanuló azonosítója, elsődleges kulcs, automatikusan kap
125 értéket
126 - name: szöveg mely a tanuló nevét tárolja, maximális hossza 255 karakter

```

```

109 lehet
 - active: boolean, az alapértelmezett értéke legyen „true”, azt adja meg,
 hogy a tanulónak aktív jogviszonya van-e
110 - eduid: szöveg, 11 karakteres, a tanuló oktatási azonosítóját tárolja
111 ```
112 Így lesz benne!!!
113 ```
114 public function up(): void
115 {
116 Schema::create('students', function (Blueprint $table) {
117 $table->id();
118 $table->string('name', 255);
119 $table->boolean('active')->default(true);
120 $table->string('eduid',11);
121 $table->timestamps();
122 });
123 }
124 ```
125
126 11. Jön a második tábla ```php artisan make:model Grade -a```
127 Figyelni kell, hogy nem többesszámban írjuk a tábla nevét, és NAGYBETŰVEL!
128
129 ```
130 b. A „grades” táblát adatbázismigráció segítségével hozza létre! (kiegészíthető
131 a keretrendszer által megkövetelt oszlopokkal)
132 - id: egész szám, a osztályzat azonosítója, elsődleges kulcs, automatikusan
133 kap értéket
134 - student_id: idegen kulcs, a tanulóra, akihez a jegy tartozik
135 - grade: egész szám, értékelés
136 - weight: egész szám, értékelés súlya
137 - comment: szöveg, 50 karakter, az értékelés magyarázata
138 ```
139 ekkor létrejönnek az alábbi fájlok:
140 ```
141 INFO Model [C:\vue-gyak-enaplo\ENaploBackend\app\Models\Grade.php] created
142 successfully.
143 INFO Factory
144 [C:\vue-gyak-enaplo\ENaploBackend\database\factories\GradeFactory.php] created
145 successfully.
146 INFO Migration
147 [C:\vue-gyak-enaplo\ENaploBackend\database\Migrations\2024_05_20_015800_create_grad
148 es_table.php] created successfully.
149 INFO Seeder [C:\vue-gyak-enaplo\ENaploBackend\database\seeders\GradeSeeder.php]
150 created successfully.
151 INFO Request
152 [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Requests\StoreGradeRequest.php] created
153 successfully.
154 INFO Request
155 [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Requests\UpdateGradeRequest.php]
156 created successfully.
157 INFO Controller
158 [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Controllers\GradeController.php]
159 created successfully.
160 INFO Policy [C:\vue-gyak-enaplo\ENaploBackend\app\Policies\GradePolicy.php]
161 created successfully.
162 ```
163
164 Ebbe a fájlba:
165 ```c:\vue-gyak-enaplo\ENaploBackend\database\Migrations\2024_05_20_015800_create_grade
166 s_table.php``` bele kell tenni az adott mezőket!
167
168 ```
169 public function up(): void
170 {
171 Schema::create('grades', function (Blueprint $table) {
172 $table->id();
173 $table->foreignIdFor(\App\Models\Student::class);
174 $table->integer('grade');
175 $table->integer('weight');
176 $table->string('comment',50);
177 $table->timestamps();
178 });
179 }
180 ```

```

```

162 }
163 ```
164
165 ```0:57:16```
166
167 12. Futtassuk a CLI: ```php artisan migrate``` parancsot, és létrejönnek a táblák az
adatbázisban (Le is fut!!!)!
168
169 ```
170 INFO Running migrations.
171
172 2024_05_20_013801_create_students_table 54.22ms DONE
173 2024_05_20_015800_create_grades_table 31.55ms DONE
174 ```
175
176 13. Kimaradt a két tábla kapcsolata, ezért módosítom a
```c:\vue-gyak-enaplo\ENaploBackend\database\Migrations\2024_05_20_015800_create_grade
s_table.php``` fájlt! Bele kell tenni a ```->constrained()``` függvényt!
177
178     ```
179     public function up(): void
180     {
181         Schema::create('grades', function (Blueprint $table) {
182             $table->id();
183             $table->foreignIdFor(\App\Models\Student::class)->constrained();
184             $table->integer('grade');
185             $table->integer('weight');
186             $table->string('comment', 50);
187             $table->timestamps();
188         });
189     }
190     ```
191
192 14. Majd futtatni egy migration rollbacket! (visszavonni a migrációt, ami futtatja a
```c:\vue-gyak-enaplo\ENaploBackend\database\Migrations\2024_05_20_015800_create_grade
s_table.php``` fájlban lévő ```down()``` függvényt, ezzel törli az adatbázisból a
táblákat, amiket megadtam migration-ként!
193 A ```CLI``` parancs: ```php artisan migrate:rollback```
194 ```
195 INFO Rolling back migrations.
196
197 2024_05_20_015800_create_grades_table 165.39ms DONE
198 2024_05_20_013801_create_students_table 32.98ms DONE
199 ```
200
201 ```
202 public function down(): void
203 {
204 Schema::dropIfExists('grades');
205 }
206 ```
207
208 15. Futtassuk a CLI: ```php artisan migrate``` parancsot, és létrejönnek a táblák az
adatbázisban (Le is fut!!!)!
209 (Vizsgán töröljük nyugodtan akár az adatbázist és futtassuk újra a migrációt! - ha nem
jut eszünkbe. Éles adatbázison, nem csinálunk ilyet!)
210
211 ```
212 INFO Running migrations.
213
214 2024_05_20_013801_create_students_table 58.42ms DONE
215 2024_05_20_015800_create_grades_table 439.30ms DONE
216 ```
217
218 16. Ekkor újra elkészül a két tábla, és az adatbázis ```migrations``` táblájába újra
bekerülnek, és látszik, a ```batch``` oszlopban, hogy hányszor készültek el.
219
220 Teljes szövegek
221 id migration batch
222 1 0001_01_01_000000_create_users_table 1
223 2 0001_01_01_000001_create_cache_table 1
224 3 0001_01_01_000002_create_jobs_table 1

```

```

225 6 2024_05_20_013801_create_students_table 2
226 7 2024_05_20_015800_create_grades_table 2
227
228 17. Ha mégis törölnéd az adatbázist, akkor phpMyAdmin/műveletek/adatbázis eldobása,
YES ``http://localhost/phpmyadmin/index.php?route=/database/operations&db=enaplo``
229
230 18. Az egész adatbázis újragenerálása ``CLI``: ``php artisan migrate``
231 ``
232 c:\vue-gyak-enaplo\ENaploBackend>php artisan migrate
233
234 WARN The database 'enaplo' does not exist on the 'mysql' connection.
235
236 Would you like to create it? (yes/no) [yes]
237 ☐ yes
238
239 INFO Preparing database.
240
241 Creating migration table 29.37ms DONE
242
243 INFO Running migrations.
244
245 0001_01_01_000000_create_users_table 164.66ms DONE
246 0001_01_01_000001_create_cache_table 108.11ms DONE
247 0001_01_01_000002_create_jobs_table 187.28ms DONE
248 2024_05_20_013801_create_students_table 29.50ms DONE
249 2024_05_20_015800_create_grades_table 146.65ms DONE
250 ``
251
252 # 3 feladat: Illessze be a megfelelő helyre a forrásként kapott Seeder fájlokat
valamint a megfelelő sorrendben hivatkozzon rájuk a DatabaseSeeder.php fájlban.
253
254 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!
255
256 20. A ``DatabaseSeeder.php`` fájlba behivatkozom a két seedert, először azt ami nem
hivatkozik másik táblára, majd ami hivatkozik!!!
257 ``
258 public function run(): void
259 {
260 $this->call([
261 StudentSeeder::class,
262 GradeSeeder::class
263]);
264 }
265 ``
266
267 21. Majd kiadom a ``CLI``: ``php artisan db:seed`` parancsot!!! és feltöltődnek az
adott táblák adattal!!!
268
269 ``
270 c:\vue-gyak-enaplo\ENaploBackend>php artisan db:seed
271
272 INFO Seeding database.
273
274 Database\Seeders\StudentSeeder RUNNING
275 Database\Seeders\StudentSeeder 167 ms DONE
276
277 Database\Seeders\GradeSeeder RUNNING
278 Database\Seeders\GradeSeeder 98 ms DONE
279 ``
280
281 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!
282 ``http://localhost/phpmyadmin/index.php?route=/database/operations&db=enaplo``
283 ``CLS``: ``php artisan migrate``
284 ``CLI``: ``php artisan db:seed``
285
286
287 # 4 feladat: Készítse el az alábbi API végpontokat! 1:42:00
288 ``
- Minden végpont JSON formátumban adja vissza a kimenetet. Hiba esetén
jelezzze a hiba okát, HTTP státusz kóddal és egy JSON-ben a hiba részleteivel

```

```

289 (a JSON-t generálhatja a keretrendszer, vagy a fejlesztő is, amennyiben
megfelel a megadott paramétereknek)
290 - GET /api/grades
291 Adja vissza azokat a jegyeket melyek beírhatóak a rendszerbe
292 {
293 "data": [
294 {"grade" :1},
295 {"grade" :2},
296 {"grade" :3},
297 {"grade" :4},
298 {"grade" :5},
299],
300 "success": true,
301 "message": ""
302 }
303 - GET /api/students
304 Adja vissza az összes tanulót és azok adatait. Az eredmény egy tanuló
objektumokból álló lista legyen az alábbi formában:
305 {
306 "data": [
307 {
308 "id":1,
309 "name":"Pista",
310 "eduid":"7777777777"
311 },
312],
313 "success": true,
314 "message": ""
315 }
316 - GET /api/students/:id
317 Adja vissza egy tanuló összes adatát. Az eredményt egy objektumként adja
vissza.
318 A kérésben az „:id” paraméterként szerepel, melyben egy tanuló „id”-t ad
át a hívó.
319 {
320 "data": [
321 {
322 "id":1,
323 "name":"Pista",
324 "eduid":"7777777777",
325 "grades":[
326 "grade":?,
327 "weight":?,
328 "comment":?
329]
330 },
331],
332 "success": true,
333 "message": ""
334 }
335 - POST /api/grades
336 A kérés törzse egy JSON objektum, mely tartalmaz egy új osztályzat
adatait, melyek a következők:
337 {
338 "student_id" : 1
339 "grade" : 2
340 "weight" : 1
341 "comment" : "Új osztályzat"
342 }
343 ```
344
345 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_persona
l_access_tokens_table.php)
346 ```
347 c:\vue-gyak-enaplo\ENaploBackend\routes\api.php
348 ```
349
350 24. Ami benne van egy bejegyzés, azt töröljük!
351 ```

```

```

352 Route::get('/user', function (Request $request) {
353 return $request->user();
354 })->middleware('auth:sanctum');
355 ```
356 25. Vegyünk fel egy új bejegyzést a ```
357 c:\vue-gyak-enaplo\ENaploBackend\routes\api.php ``` -ba!
358 ```
359 use App\Http\Controllers\GradeController;
360
361 Route::apiResource('grades', GradeController::class)
362 ->only(['index']);
363 ```
364 26. Nézzük meg, hogy bejegyződött-e! CMD: ```php artisan rout:list``` , és ha szerepel
365 benne az ```api/grades``` , akkor OK!
366 ```
367 c:\vue-gyak-enaplo\ENaploBackend>php artisan rout:list
368
369 GET|HEAD /
370 POST _ignition/execute-solution ignition.executeSolution > Spatie...
371 GET|HEAD _ignition/health-check ignition.healthCheck > Spatie\Laravel...
372 POST _ignition/update-config ignition.updateConfig > Spatie\Larav...
373 GET|HEAD api/grades grades.index > GradeController@index
374 GET|HEAD sanctum/csrf-cookie sanctum.csrf-cookie > Laravel\Sanctum > ...
375 GET|HEAD up
376
377 Showing [7] routes
378 ```
379 ```1:52:00```
380
381 27. Valósítsuk meg a működését! Kell egy USE és egy return az index() függvénybe! (Itt
382 bele kerül kézzel az adat és a ```status: 200```-as státusz kód is!!!! (ez az OK-t
383 jelenti)
384 ```c:\vue-gyak-enaplo\ENaploBackend\app\Http\Controllers\GradeController.php```-ben.
385 ```
386 ```
387 public function index()
388 {
389 return Response::json(
390 [
391 "data" => [
392 ["grade">1],
393 ["grade">2],
394 ["grade">3],
395 ["grade">4],
396 ["grade">5],
397],
398 "success" => true,
399 "message" => ""
400], 200
401);
402 }
403 ```
404 28. Megnézzük, hogy működ-e! CLI: ```PHP artisan serve```
405 ```
406 c:\vue-gyak-enaplo\ENaploBackend>php artisan serve
407
408 INFO Server running on [http://127.0.0.1:8000].
409
410 Press Ctrl+C to stop the server
411 ```
412 ```
413 29. Böngészőben megnyitjuk a: ```http://127.0.0.1:8000/api/grades``` és az első kész
414 is van!
415
416 30. Második API végpont: ```GET /api/students``` , a routes\api.php-ben felveszünk egy
417 új bejegyzést!
418 ```

```

```

418 c:\vue-gyak-enaplo\ENaploBackend\routes\api.php
419 ```
420 fájlban, a:
421 ```
422 use App\Http\Controllers\StudentController;
423
424 Route::apiResource('students', StudentController::class)
425 ->only(['index']);
426 ```
427
428 31. Student Contorller index függvényét kell létrehozni!!!!
429
430 32. Ő írja legy, hogy 1 student hogy jelenjen meg: Célszerű parancssorból: CLI:
431 ```php artisan make:resource StudentResource```
432 ```
433 c:\vue-gyak-enaplo\ENaploBackend>php artisan make:resource StudentResource
434
435 INFO Resource
436 [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\StudentResource.php] created
437 successfully.
438 ```
439
440 33. A sok student hogyan legyen formázva, Célszerű parancssorból: CLI: ```php artisan
441 make:resource StudentCollection```
442 ```
443 c:\vue-gyak-enaplo\ENaploBackend>php artisan make:resource StudentCollection
444
445 INFO Resource collection
446 [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\StudentCollection.php]
447 created successfully.
448 ```
449
450 34. Felvesszük, hogy hogyan néz ki egy student (a return mögötti részt töröljük és oda
451 írjuk!):
452 ```
453 C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\StudentResource.php
454 ```
455 ide:
456 ```
457 public function toArray(Request $request): array
458 {
459 return [
460 'id' => $this ->id,
461 'name' => $this ->name,
462 'eduid' => $this -> eduid
463];
464 }
465 ```
466
467 35. Megoldjuk, hogy kiíratódjanak a Studentek a Collectionban! (a return mögötti részt
468 töröljük és oda írjuk!)
469 ```
470 C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\StudentCollection.php
471 ```
472 ide:
473 ```
474 public function toArray(Request $request): array
475 {
476 return [
477 'data' => $this->collection,
478 'success' => true,
479 'message' => ''
480];
481 }
482 ```

```



```

482 36. És még be kell hivatkozni a StudentContorllert
483 ``` c:\vue-gyak-enaplo\ENaploBackend\app\Http\Controllers\StudentController.php```
484 Ennek
485
486 ```
487 public function index()
488 {
489 return new StudentCollection(Student::all());
490 }
491 ```
492
493 37. Harmadik API végpont: ```GET /api/students/:id``` , a routes\api.php-ben
 felvesszünk egy új bejegyzést!
494 ```
495 c:\vue-gyak-enaplo\ENaploBackend\routes\api.php
496 ```
497 fájlban, felvesszük a index mellé a show-t is!:
498
499 ```
500 Route::apiResource('students', StudentController::class)
501 ->only(['index', 'show']);
502 ```
503
504 38. Majd a
505 ```c:\vue-gyak-enaplo\ENaploBackend\app\Http\Controllers\StudentController.php```
 fájlba megírjuk a SHOW függvényt:
506 public function show(Student $student)
507 {
508 return new StudentResource($student);
509 }
510 39. De ez nem jó, mert nem ez a feladat!!!! Hanem létrehozunk egy
    ```StudentDetailedResource.php``` fájlt!
511 CLI: ```php artisan make:resource StudentDetailedResource```
512 ```
513 PS C:\vue-gyak-enaplo\ENaploBackend> php artisan make:resource StudentDetailedResource
514
515     INFO Resource
    [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\StudentDetailedResource.php]
    created successfully.
516
517 ```
518 40. És ezt használjuk a StudentContorller.php SHOW függvényében, tehát 38 ki, 40
    -be!!!!
519 ```
520     public function show(Student $student)
521     {
522         return new StudentDetailedResource($student);
523     }
524 ```
525
526 41. A
527 ```C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\StudentDetailedResource.php```
    ben az alapműködés helyett felépítjük azt, amit a feladat kér!
528 ```StudentDetailedResource.php```
    fájlban:
529 ```
530 public function toArray(Request $request): array
531 {
532     return [
533         'data' => [
534             'id' => $this->id,
535             'name' => $this->name,
536             'eduid' => $this->eduid,
537             'grades' => $this->grades
538         ],
539         'success' => true,
540         'message' => ''
541     ];
542 }
543 ```
544 42. de ez még nem jó, mert a ```grades```-t nem tölti be, ekkor rakunk bele elgy

```

lekérdezést, ami még mindig nem elég jó:

```
545
546 ```
547     public function toArray(Request $request): array
548     {
549         return [
550             'data' => [
551                 'id' => $this->id,
552                 'name' => $this->name,
553                 'eduid' => $this->eduid,
554                 'grades' => Grade::where('student_id', $this->id)->get()
555             ],
556             'success' => true,
557             'message' => ''
558         ];
559     }
560 ```
561 43. Ezért létrehozunk egy ```GradeResource.php```-t CLI: ```php artisan make:resource
GradeResource```
562
563 PS C:\vue-gyak-enaplo\ENaploBackend> php artisan make:resource GradeResource
564
565 INFO Resource
566 [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\GradeResource.php] created
567 successfully.
568
569 44. Meg akkor már kell egy GradeCollection.php is kell!!!
570 CLI: ```php artisan make:resource GradeCollection```
571
572 PS C:\vue-gyak-enaplo\ENaploBackend> php artisan make:resource GradeCollection
573
574 INFO Resource collection
575 [C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\GradeCollection.php] created
576 successfully.
577
578 45. ezekben definiáljuk, hogy hogyan nézzenek ki!!!! A
579 ```C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\GradeResource.php```-ban ezt
580 írjuk:
581
582 ```
583     public function toArray(Request $request): array
584     {
585         return [
586             'grade' => $this->grade,
587             'weight' => $this->weight,
588             'comment' => $this->comment
589         ];
590     }
591 ```
592
593 46. Majd megjelenítjük egy a ```GradeCollection.php```-ban!
594
595 C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\GradeCollection.php``` - ott ezt
596 írjuk:
597
598 ```
599
600 47. Javítjuk a
601 ```C:\vue-gyak-enaplo\ENaploBackend\app\Http\Resources\StudentDetailedResource.php```
602 ben az alapműködés helyett felépítjük azt, amit a feladat kér!
603 ```StudentDetailedResource.php```
604 fájlban:
605
606 ```
607     return [
608         'data' => [
609             'id' => $this->id,
610             'name' => $this->name,
611             'eduid' => $this->eduid,
612             'grades' => new GradeCollection(Grade::where('student_id',
613                 $this->id)->get())
614         ]
615     ]
616 ```
```

```

606         ],
607         'success' => true,
608         'message' => ''
609     ];
610
611     ```
612 48. De ennél még van egy egyszerűbb és szebb megoldás is, ha ezeket a kapcsolatokat
    felvesszük a modellekénél, akkor jobb lesz!!!
613 A ```c:\vue-gyak-enaplo\ENaploBackend\app\Models\Student.php```-be felveszek egy
    grades() függvényt:
614     ```
615     class Student extends Model
616     {
617         use HasFactory;
618
619         public function grades()
620         {
621             return $this->hasMany(Grade::class);
622         }
623     }
624     ```
625
626 49. Negyedik API végpont: ```POST /api/grades
627     A kérés törzse egy JSON objektum, mely tartalmaz egy új osztályzat
    adatait, melyek a következők:
628     {
629         "student_id" : 1
630         "grade" : 2
631         "weight" : 1
632         "comment" : "Új osztályzat"
633     }
634     ```
635
636 50. Vegyünk fel egy új bejegyzést a Felvesszük a ```'store'``` kiegészítést a
637 ```c:\vue-gyak-enaplo\ENaploBackend\routes\api.php``` -ba!
638     ```
639     use App\Http\Controllers\GradeController;
640
641     Route::apiResource('grades', GradeController::class)
642         ->only(['index', 'store']);
643     ```
644 51. A ```GradeController.php```-ben van egy ```Store()``` függvény, ami a
    ```StoreGradeRequest.php```-re hivatkozik, ott átírjuk az ```authorize()``` függvény
 return-jét true-ra```!
646
647 ```GradeController.php```-file:
648 ```
649 public function store(StoreGradeRequest $request)
650 {
651 //
652 }
653 ```
654
655 ```StoreGradeRequest.php```-file:
656 ```
657 public function authorize(): bool
658 {
659 return true;
660 }
661 ```
662
663 52. ```2:53:20``` A ```StoreGradeRequest.php``` -ben lévő ```rules()``` függvényében
 megírjuk a következőt!
664 ```c:\vue-gyak-enaplo\ENaploBackend\app\Http\Requests\StoreGradeRequest.php```-ben:
665 ```
666 public function rules(): array
667 {
668 return [
669 'student_id' => 'required',

```

```

672 'grade' => 'required',
673 'weight' => 'required',
674 'comment' => 'required'
675];
676 }
677 ```
678
679 53. Hogy lehessen megadni tömegesen adatokat, bele kell írni a
680 ```c:\vue-gyak-enaplo\ENaploBackend\app\Models\Grade.php``` -ba, hogy:
681 ```
682
683 class Grade extends Model
684 {
685 use HasFactory;
686
687 protected $guarded=[];
688 }
689 ```
690 54. A ```c:\vue-gyak-enaplo\ENaploBackend\app\Http\Controllers\GradeController.php```
691 -ben megírjuk a ```Store()``` függvényt!
692 ```
693 public function store(StoreGradeRequest $request)
694 {
695 $data = $request->validated();
696
697 $grade = Grade::create($data);
698
699 return Response::json([
700 'data' => new GradeResource($grade),
701 'success' => true,
702 'message' => 'Created'
703],201
704);
705 ```
706
707 55. A StoreGradeRequest.php-be írunk egy felüldefiniált ```failedValidation()```
708 függvényt, hogy tesztelhessük, hogy miért nem jó a progink.
709 ```
710 use Illuminate\Support\Facades\Response;
711
712 protected function failedValidation(Validator $validator)
713 {
714 throw new HttpResponseException(
715 Response::json([
716 'data' => $validator->errors(),
717 'success' => false,
718 'message' => 'Validation error'
719],400)
720);
721 }
722 ```
723
724 56. Nyisd meg a PostMAN-t, és
725 ```
726 post, http://localhost:8000/api/grades
727
728
729 Body, raw, json
730
731 {
732 "student_id":1,
733 "grade":2,
734 "weight":1,
735 "comment": "Új osztályzat"
736 }
737 SEND
738
739 Ha minden jó, akkor beírja ezt a json-t az adatbázis grades táblájába!
740 http://localhost/phpmyadmin/index.php?route=/sql&pos=0&db=enaplo&table=grades
741

```

```

742 'grades' - tábla
743 id student_id grade weight comment created_at updated_at
744 1 1 4 1 Órai munka 2024-05-21 23:49:49 2024-05-21
23:49:49
745 2 1 3 2 Laravel Témazáró 2024-05-21 23:49:49 2024-05-21
23:49:49
746 3 1 5 1 Felelet 2024-05-21 23:49:49 2024-05-21
23:49:49
747 4 2 2 1 Órai munka 2024-05-21 23:49:49 2024-05-21
23:49:49
748 5 2 5 2 Laravel Témazáró 2024-05-21 23:49:49 2024-05-21
23:49:49
749 6 1 2 1 Új osztályzat 2024-05-22 17:43:37 2024-05-22
17:43:37
750
751 ...
752
753 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 írunk megoldást az api.php-ben!
``->missing(function(Request $request){return Response::json``-t írunk mindkét
route-hoz!
754
755 ...
756 <?php
757 use App\Http\Controllers\GradeController;
758 use App\Http\Controllers\StudentController;
759 use Illuminate\Http\Request;
760 use Illuminate\Support\Facades\Response;
761 use Illuminate\Support\Facades\Route;
762
763 Route::apiResource('grades', GradeController::class)
764 ->only(['index', 'store'])
765 ->missing(function(Request $request){
766 return Response::json([
767 'data' => null,
768 'success' => false,
769 'message' => 'Grade not exists'
770],404);
771 });
772
773 Route::apiResource('students', StudentController::class)
774 ->only(['index', 'show'])
775 ->missing(function(Request $request){
776 return Response::json([
777 'data' => null,
778 'success' => false,
779 'message' => 'Student not exists'
780],404);
781 });
782
783 ...
784
785 58. Nyisd meg a PostMAN-t, és
786 ...
787 get, http://127.0.0.1:8000/api/students/3
788
789 Body
790
791 SEND
792
793 Ezt adja vissza:
794 {
795 "data": null,
796 "success": false,
797 "message": "Student not exists"
798 }
799 ...
800
801
802
803 # 5.feladat A POST /api/types végpont ellenőrizze, hogy a bemeneti információk
megfelelőek-e az alábbi elven.

```

```

804 `` - student_id: kötelező kitölteni és csak olyan értéket fogad el, mely a
 „group” táblában létező id.
805 - grade: kötelező és egész szám
806 - weight: kötelező és egész szám
807 - comment: kötelező és szöveg
808
809 Validációs probléma esetén JSON formátumú válaszban jelezze azt a backend a
 frontend felé.
810 Amennyiben az adatok megfelelőek, vegye fel adatbázisba a osztályzatot és
 adja azt vissza, 201-es státusz kóddal és „Created” üzenettel.
811 ```
812
813 59. ``3:26:00`` A ``StoreGradeRequest.php`` -ben lévő ``rules()`` függvényében
 felvesszük a típusokat is!
814 ``c:\vue-gyak-enaplo\ENaploBackend\app\Http\Requests\StoreGradeRequest.php``-ben:
815 ```
816 public function rules(): array
817 {
818 return [
819 'student_id' => 'required|exists:students,id',
820 'grade' => 'required|integer',
821 'weight' => 'required|integer',
822 'comment' => 'required|string',
823];
824 }
825 ```
826
827 60. Próbáljuk ki, hogy működik-e! Nyisd meg a PostMAN-t, és
828 ```
829 post, http://localhost:8000/api/grades
830
831 Body, raw, json
832
833 {
834 "student_id":10,
835 "grade":"2a",
836 "weight":"1e",
837 "comment": "Hiba"
838 }
839 SEND
840
841 EZ a válasz jön vissza!!!!
842 {
843 "data": {
844 "student_id": [
845 "The selected student id is invalid."
846],
847 "grade": [
848 "The grade field must be an integer."
849],
850 "weight": [
851 "The weight field must be an integer."
852]
853 },
854 "success": false,
855 "message": "Validation error"
856 }
857 ```
858
859 # 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.
860 ```
861 A nézet a "GET /" végpont betöltésekor jelenjen meg.
862 ```
863
864 Segítségül használhatók:
865 internetkapcsolat, amely használható:
866 átalános keresésre;

```

```

870 online dokumentáció elérésére;
871 projekt keretek létrehozására, csomagok telepítésére.
872 ...
873
874 61. Lecseréljük a ```c:\vue-gyak-enaplo\ENaploBackend\routes\web.php```-ben a default
 view-et,
875 ```
876 <?php
877
878 use Illuminate\Support\Facades\Route;
879
880 Route::get('/', function () {
881 return view('grades ');
882 });
883 ```
884
885 62. Csinálunk egy view-et,
886 ```c:\vue-gyak-enaplo\ENaploBackend\resources\views\grades.blade.php```-fájlt!
887 ebbe csinálunk egy HTML-t!
888 ```
889 <!doctype html>
890 <html lang="hu">
891 <head>
892 <title>Enaplo</title>
893 </head>
894 <body>
895 <h1>Osztályzatok</h1>
896 <hr>
897
898 @foreach($grades as $grade)
899 {{$grade->grade}} (x{{$grade->weight}}) [{{$grade->comment}}]
900 @endforeach
901
902 </body>
903 </html>
904 ```
905
906 63. Majd ezt kibővítjük, hogy a tanulók nevei is benne legyenek, de előtte kell egy
907 Grade->Student hivatkozás is, mert hiába vesszük fel a mév mezőt, nem jelennek meg a
908 nevek!
909 ```Grade.php```:
910 ```
911 public function student()
912 {
913 return $this->belongsTo(Student::class);
914 }
915 ```
916
917 64. Betesszük a neveket is a jegyek alá a HTML-be
918 ```
919 <!doctype html>
920 <html lang="hu">
921 <head>
922 <title>Enaplo</title>
923 </head>
924 <body>
925 <h1>Osztályzatok</h1>
926 <hr>
927
928 @foreach($grades as $grade)
929 {{$grade->grade}} (x{{$grade->weight}}) [{{$grade->comment}}]
930
931 {{$grade->student->name}}
932
933
934 @endforeach
935
936 </body>
937 </html>

```

