



Neumann János Egyetem
Műszaki és Informatikai
Kar

NAPLÓ

/gyakorlat beadandó/

BALOGH NORBERT
I2I25Q

TORONTÁLI ESZTER
J3GE0B

2024

Tartalomjegyzék

1.	AZ ELLENŐRZÉSHEZ SZÜKSÉGES LINKEK	3
3.	PROJECT FELÉPÍTÉSE	4
4.	ALKALMAZÁS BEMUTATÁSA	6
5.	RESTFUL API.....	12

1. Az ellenőrzéshez szükséges linkek

GitHub link: <https://github.com/NickDale/naplo>

GitHub felhasználók:

- Torontáli Eszter – *GHEszti*
- Balogh Norbert – *NickDale*

A feladat felosztásokat és követéseket a GitHub projekt segítségével valósítottuk meg.

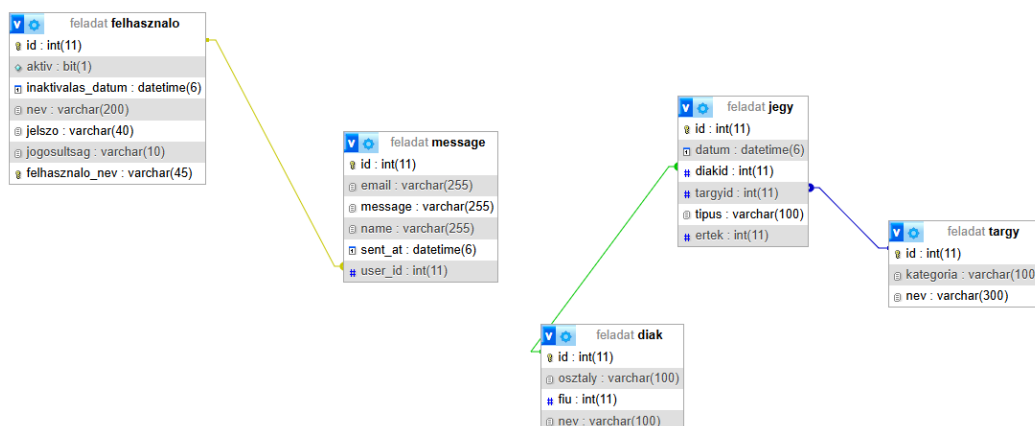
<https://github.com/users/NickDale/projects/3>

2. Feladat meghatározása

Egyre több iskolában használnak elektronikus naplót. Ebben a naplóban – sok más egyéb mellett – rögzítik az osztályzatokat is. A feladatban szereplő adatbázis a Városvégi Gimnázium elektronikus naplójának egy tanév első négy hónapjában bejegyzett jegyeit tartalmazza. Tudjuk, hogy az adathalmazban nincs két azonos nevű személy.

A jegy és a tárgy tábla között egy a többhöz kapcsolat van, mivel egy tárgyhöz több jegy is kapcsolódhat, azonban egy jegy az biztosan csak egy tárgyhöz tartozik. A jegy diák táblák között szintén egy a többhöz kapcsolat, mivel egy diákhoz több jegy is tartozik, de egy jegy az egy konkrét diákhoz tartozik.

A következő kép szemlélteti a feladat során elkészített adatbázis diagramja, az egyes táblák közötti kapcsolattal.



3. Project felépítése

A project megvalósítása során a következő technológiákat, nyelveket, toolokat használtuk.

- Spring boot 3.4.0
- JAVA 17
- maven
- MySQL adatbázis
- Használt dependencyk
 - o Spring Data JPA az adatbázis leképezéséhez és az adatbázis műveletekhez
 - o Spring Security a jogosultság kezeléshez
 - o Thymeleaf template engine a megjelenítéshez

A spring bootnak köszönhetően a *JPA* és a *Thymeleaf* konfigurációt elegendő volt az *application.propertiesben* megadni és a keretrendszer létrehozta a szükséges beaneket, konfigurációkat.

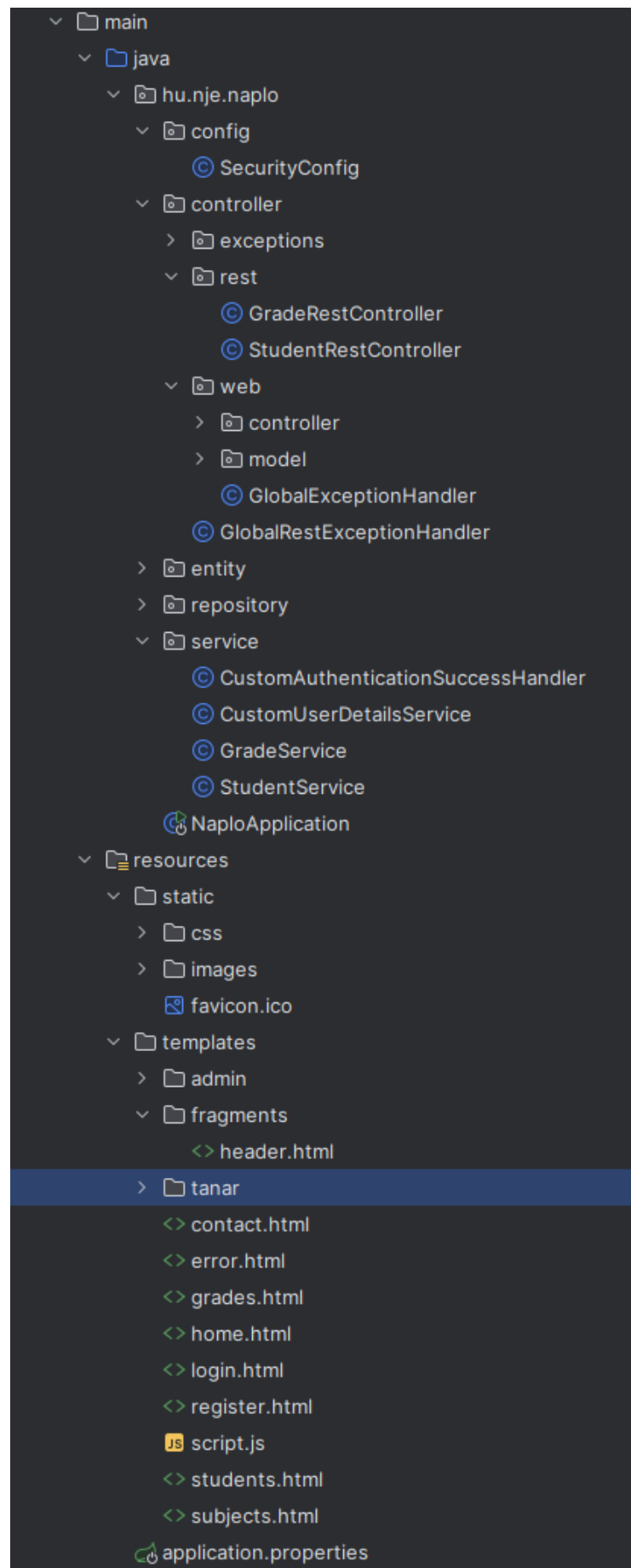
Az adatbázis elérési URL-t a feladatkiírásban feltüntetett módon állítottuk be, így a *MYSQL_HOST* könnyen lecserélhető teszteléshez egy másik elérési útvonalra.

A projektet legjobb tudásunk szerint igyekeztünk strukturáltan felépíteni, ahol MVC-t, vagyis a vizuális megjelenítést szolgáló API-kat és a REST API-kat is elkülönítettük egy mástól.

A REST API-k kaptak egy */api/....* előtagot is. Ezek a REST standardnak megfelelően *application/json* formátumban adnak vissza adatokat a megfelelő HTTP státuszkódok használata mellett.

A REST API-kat a *@RestController*, míg a felület által használtakat a *@Controller* Spring komponensek biztosítják.

Neumann János Egyetem - Informatika Tanszék
Balogh Norbert - I2I25Q
Torontáli Eszter - J3GE0B
2024

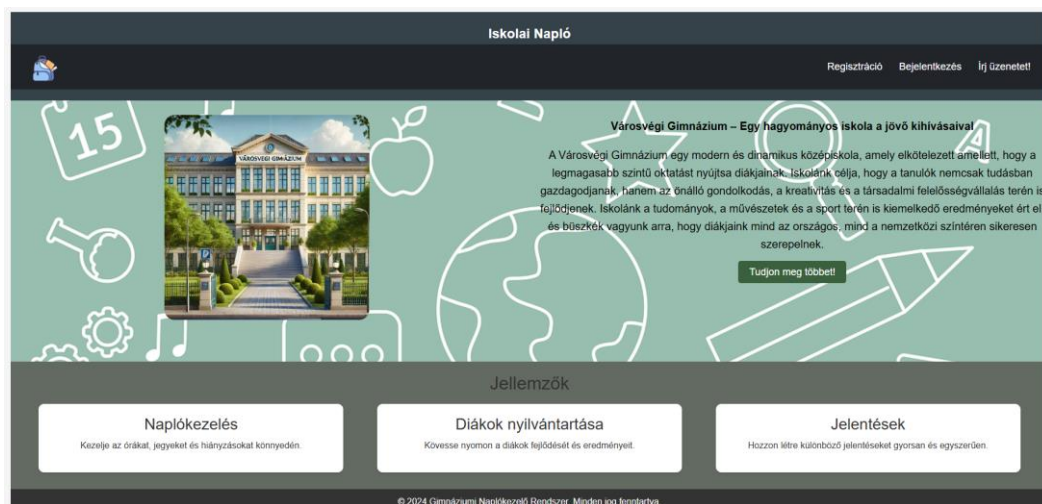


4. Alkalmazás bemutatása

4.1. Home page

Az alkalmazást indítását követően a /, /home útvonalon bejön a kezdőoldal, ahol a gimnázium rövid bemutatását láthatjuk.

A kezdőoldal mindenki számára elérhető, innen lehet elérni a további funkciókat is, mint a bejelentkezést, a regisztrációt és az üzenetküldést



4.2.Regisztráció

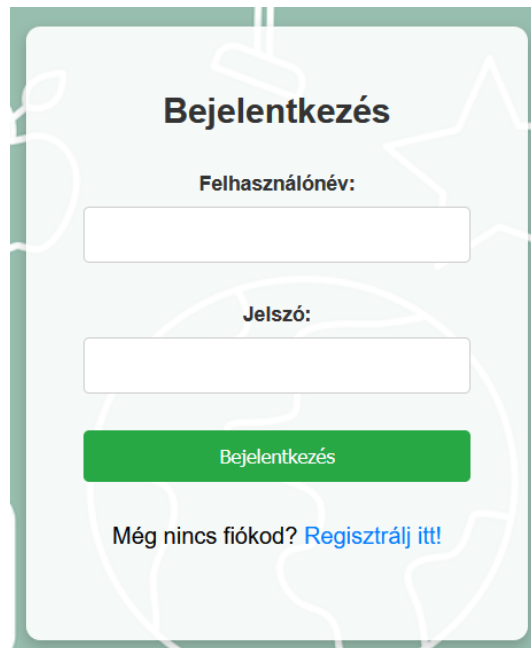
Jelentkezzen be itt!' (Already have an account? [Log in here!](#)))."/>

1. ábra - felhasználó regisztráció

A regisztrációs form sikeres kitöltését követően az adatok mentésre kerülnek az adatbázis felhasználó táblájába. Ezt követően a regisztrált felhasználó már be is tud jelentkezni a rendszerbe.

4.3. Bejelentkezés

Regisztráció után bejelentkezésnél a felhasználó jogosultsági körét is azonosítjuk. Helytelen adatok megadása esetén sikertelen a bejelentkezés.

A bejelentkezési felület egy egyszerű, modern dizájnban. A cím "Bejelentkezés" fekete, közepes méretű betűkben van. Alatta a "Felhasználónév:" címke egy fehér, vízszintes input mezőhöz tartozik. Ezután a "Jelszó:" címke egy másik fehér, vízszintes input mezőhöz tartozik. A mezők alatt egy zöld, téglalap alakú gomb van, amelyen a "Bejelentkezés" felirat olvasható fehér betűkben. A gomb alatt a szöveg "Még nincs fiókod? [Regisztrálj itt!](#)" látható, ahol a link kék és aláhúzott.

2. ábra - bejelentkezési felület

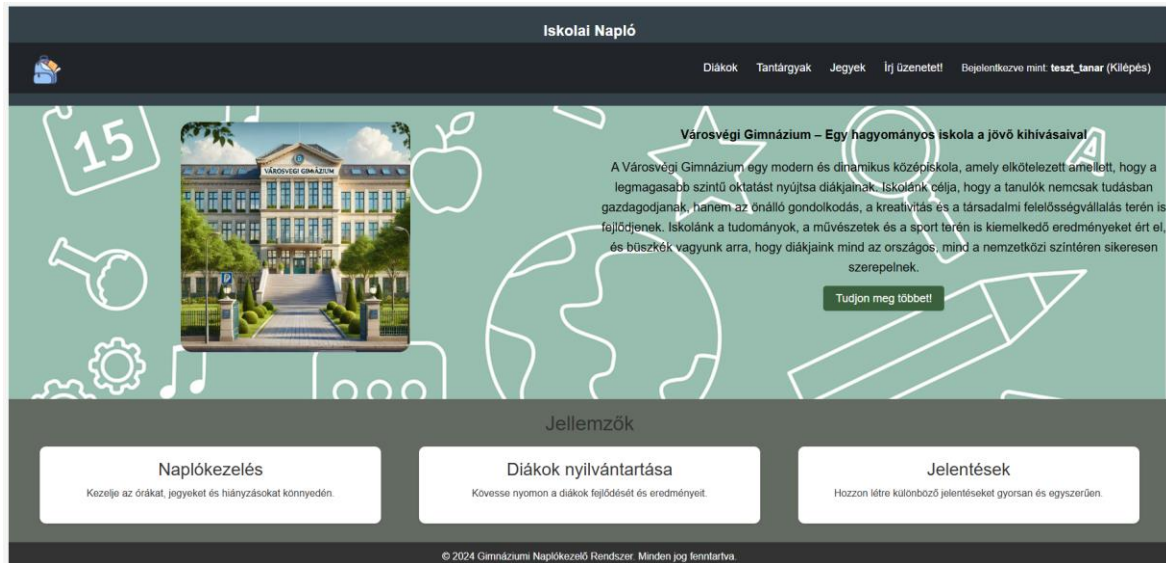
4.4. Menü kezelés

A bejelentkezést követően a bejelentkezett felhasználó jogkörének megfelelően más menüpontok jelennek meg.

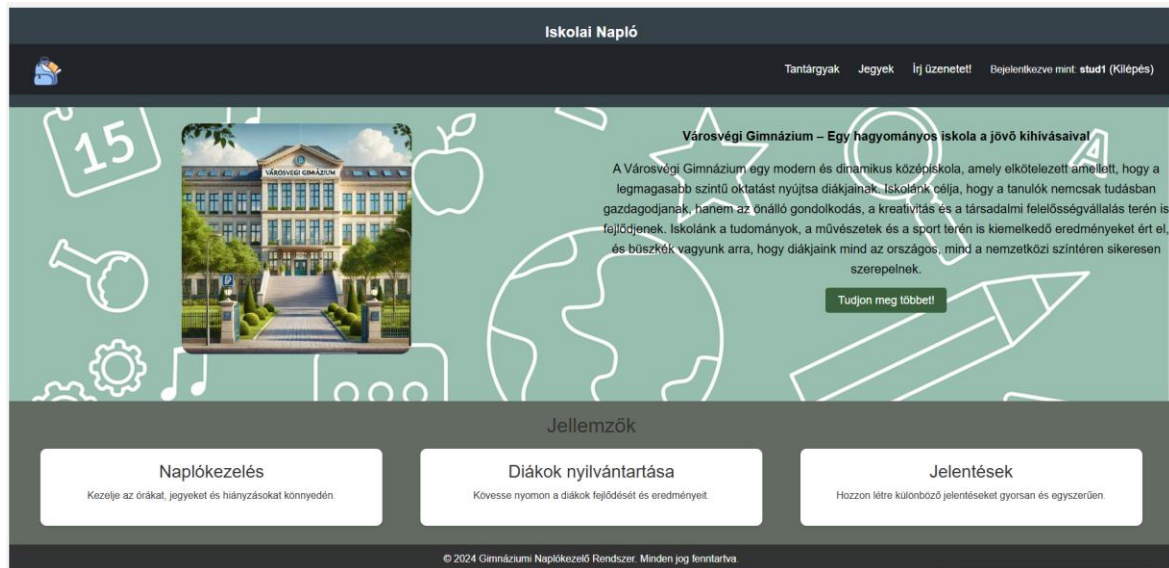
Ha tanár lép be, akkor számára a következő menüpontok jelennek meg:

- Diákok – itt a tanulókat tudja megtekinteni
- Tantárgyak – itt a tantárgyakat tudja megtekinteni
- Jegyek – a tanulóknak kiosztott jegyeket látja dátum és tantárgy szerint
- Írj üzentet – ezzel a menüponttal a tanárok is képesek üzenetet hagyni az admin számára.

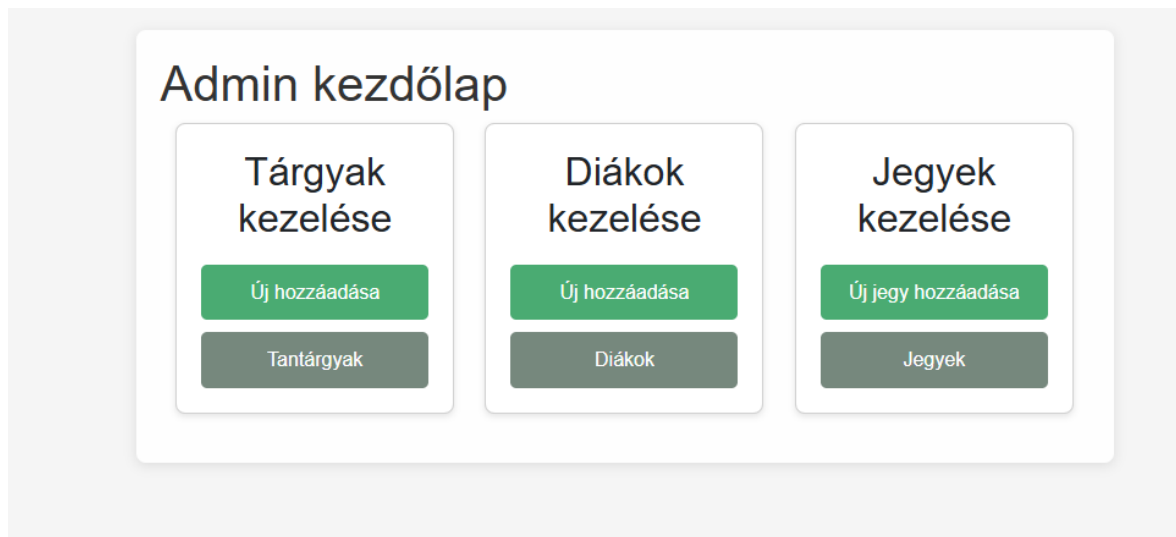
Neumann János Egyetem - Informatika Tanszék
Balogh Norbert - I2I25Q
Torontáli Eszter - J3GE0B
2024



Bejelentkezést követően eltűnik a regisztráció és a bejelentkezés menü, mert ezek értelmüket veszítik, de megjelenik, hogy ki van bejelentkezve és mellette a kilépés gomb. Ha diák jelentkezik be, akkor ő a Diákok menüpontot nem láthatja

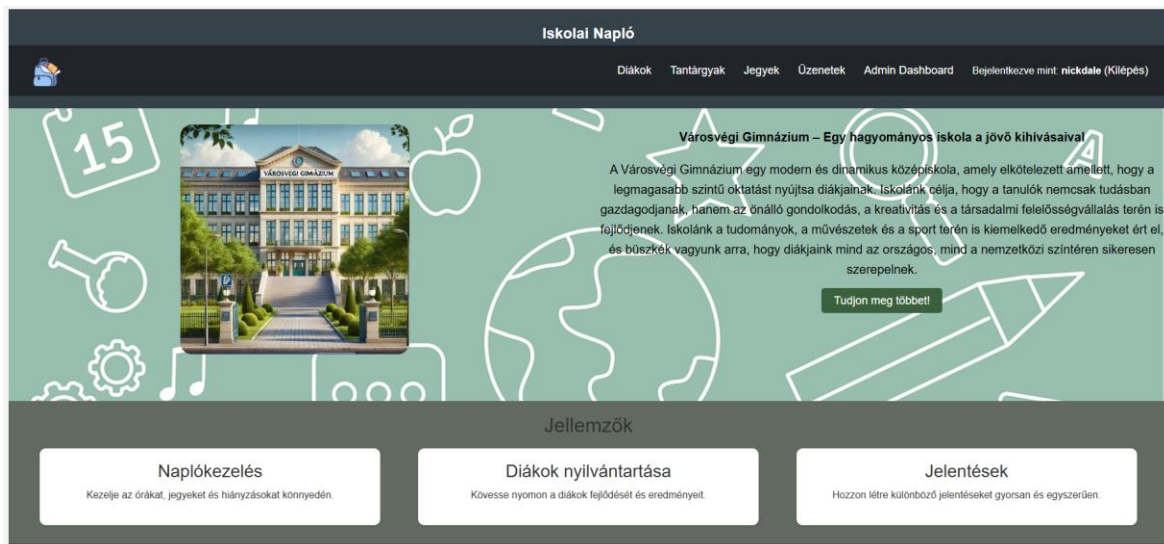


Ha az admin user jeletkezik be, akkor őt a bejelentkezést követően a Spring security a admin felületre továbbítja.



Innen elindulva ő is vissza tud menni a főoldalra, és látja ugyanazokat az adatokat, amit a többi felhasználó.

Amiben különbözik az admin user, hogy ő az üzent küldés menüpont helyett a beérkezett üzenteteket látja.



A különböző menüpontok megjelenését a spring boot security segítségével valósítottuk meg.

A menüt egy tymeleaf fragmentben valósítottuk meg és ezt húztuk be minden html fájlba. Itt lehetőségünk volt megmondani, hogy melyik menüpontok érhetőek el bejelentkezés nélkül és melyek csak bejelentkezéssel. Egyes menüpontokhoz (pl. üzenetek) nem csak azt

izsgáltuk, hogy be van-e jelentkezve az adott felhasználó, hanem, hogy rendelkezik-e a megfelelő jogosultsággal.

```
<div th:fragment="header">
  <div class="logo">Iskolai Napló</div>
  <nav id="navbar" class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container-fluid">
      <a class="navbar-brand" th:href="@{/home}">
        </a>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav ms-auto">
          <li sec:authorize="isAnonymous()">
            <a th:href="@{/users/register}">Regisztráció</a>
          </li>
          <li sec:authorize="isAnonymous()">
            <a th:href="@{/login}">Bejelentkezés</a>
          </li>

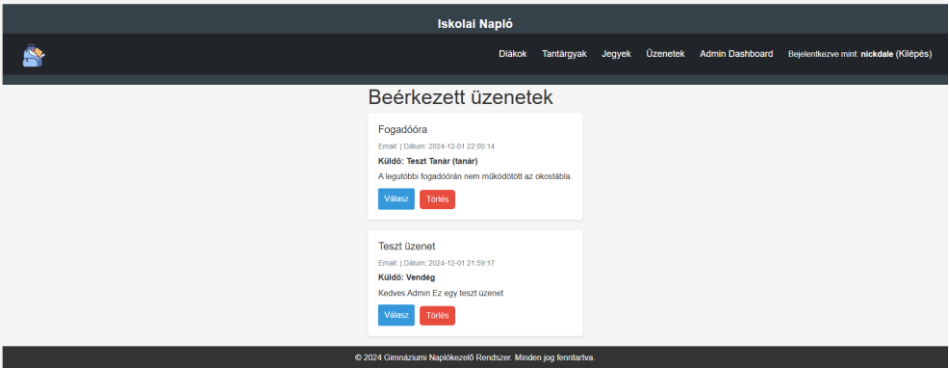
          <li sec:authorize="hasRole('ROLE_TEACHER') or hasRole('ROLE_ADMIN')">
            <a th:href="@{/students}">Diákok</a>
          </li>
          <li sec:authorize="isAuthenticated()">
            <a th:href="@{/subjects}">Tantárgyak</a>
          </li>
          <li sec:authorize="isAuthenticated()">
            <a th:href="@{/grades}">Jegyek</a>
          </li>
          <li sec:authorize="isAnonymous() or (isAuthenticated() and !hasRole('ROLE_ADMIN'))">
            <a th:href="@{/contact}">Írj Üzenetet!</a>
          </li>
          <li sec:authorize="isAuthenticated() and hasRole('ROLE_ADMIN')">
            <a th:href="@{/admins/messages}">Üzenetek</a>
          </li>
          <li sec:authorize="isAuthenticated() and hasRole('ROLE_ADMIN')">
            <a th:href="@{/admins/dashboard}">Admin Dashboard</a>
          </li>

          <li sec:authorize="isAuthenticated()">
            Bejelentkezve mint:
            <strong>
              <span sec:authentication="name"></span>
            </strong>
            <a th:href="@{/logout}">(Kilépés)</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>
</div>
```

Természetesen azt is levédtük, hogy ha a felhasználó kézzel írja be az url-t, és úgy szeretne elérni egy olyan oldalt, amihez nincs jogosultsága. Erre a @PreAuthorized(...) annotációt használtuk.

```
15
16 @PreAuthorize("hasRole('ADMIN')")  NickDale +2
17 @Controller
18 @RequestMapping(path = @"/admins")
19 @RequiredArgsConstructor
20 public class AdminController {
21
22     private final ContactMessageRepository contactMessageRepository;
23
24     @GetMapping(@"/dashboard")  NickDale
25     public String dashboard() {
26         return "admin/dashboard";
27     }
28
29     @GetMapping(@"/messages")  NickDale
30     public String viewMessages(Model model) {
31         List<ContactMessage> messages = contactMessageRepository.findAll(Sort.by(Sort.Direction.DESC, "sentAt"));
32         model.addAttribute("messages", messages);
33         return "admin/messages";
34     }
35
36     @GetMapping(@"/delete-message/{messageId}")  NickDale
37     public String viewMessages(@PathVariable int messageId, Model model) {
38         contactMessageRepository.deleteById(messageId);
39         return "redirect:/admins/messages";
40     }
41
42 }
43
```

4.5. Üzenetek



4.6. Jegyek listázása

Diák neve	Osztálya	Tantárgy neve	Kapott jegy	Dátum	Típus	Művelet
Ajtösi Patrik	9/B	angol nyelv (nyelv)	5	2010-09-21	órai munka	Módosítás Töröl
Ajtösi Patrik	9/B	matematika (matematika)	4	2010-09-23	röpdolgozat	Módosítás Töröl
Ajtösi Patrik	9/B	foldunk és környezetünk (real)	4	2010-09-28	röpdolgozat	Módosítás Töröl
Ajtösi Patrik	9/B	magyar nyelv (human)	3	2010-09-29	szövebeli felelet	Módosítás Töröl
Ajtösi Patrik	9/B	kémia (real)	5	2010-09-30	röpdolgozat	Módosítás Töröl
Ajtösi Patrik	9/B	történelem (human)	2	2010-10-03	röpdolgozat	Módosítás Töröl
Ajtösi Patrik	9/B	informatika (informatika)	5	2010-10-17	órai munka	Módosítás Töröl
Ajtösi Patrik	9/B	angol nyelv (nyelv)	4	2010-10-18	röpdolgozat	Módosítás Töröl
Ajtösi Patrik	9/B	foldunk és környezetünk (real)	3	2010-10-20	házi feladat	Módosítás Töröl

5. RESTful API

Az alkalmazásban diák táblára valósítottuk meg, az alapvető CRUD műveleteket a RESTful elveknek megfelelően.

A megvalósított REST API-k kaptam egy /api/.. előtagot, ezzel jól elkülöníthetők a MVC Controllerektől.

5.1. GET

Lekérdezésekhez használatos HTTP módszer

A feladatban 2 db GET lekérdezést valósítottunk meg.

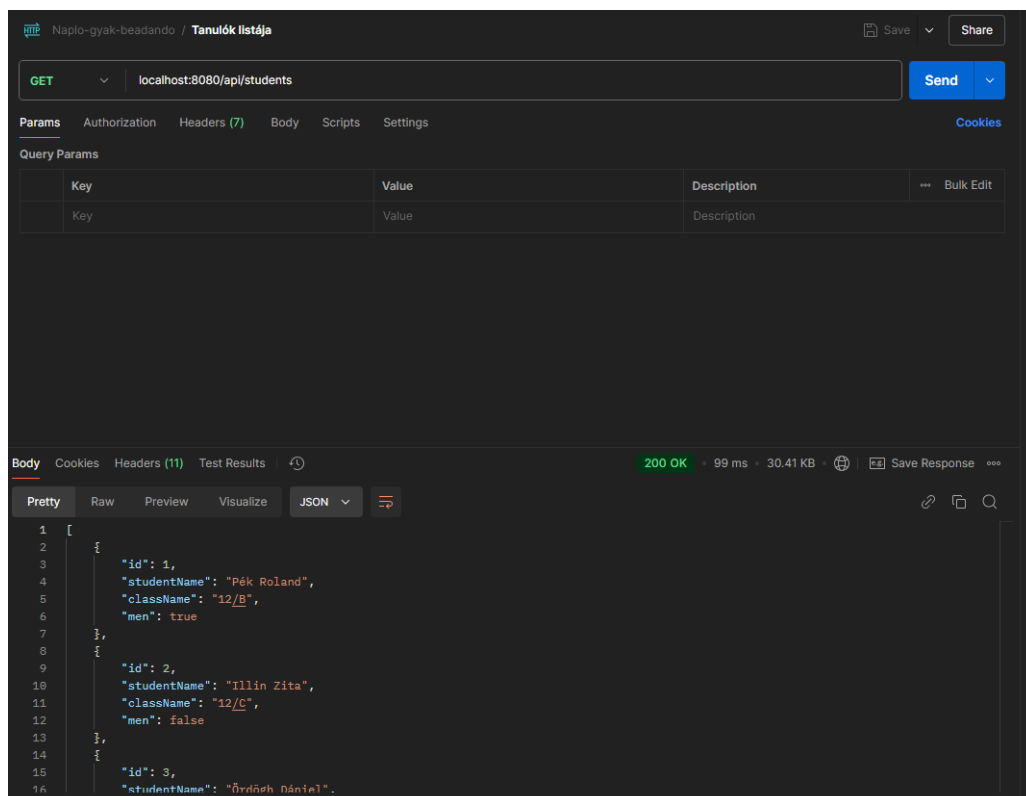
5.1.1 Összes tanuló lekérdezése

HTTP módszer : GET

URL : /api/students/

Válasz: Listában visszaadja az adatbázisban szereplő diákok adatait.

Postman test:



cUrl test:

```
curl -v --location "localhost:8080/api/students"
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           % Done      0         0             0       0      0 --:--:-- --:--:-- --:--:-- 0.00
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json
Transfer-Encoding: chunked
Date: Sun, 01 Dec 2024 12:38:45 GMT

[{"id":1,"studentName":"Pék Roland","className":"12/B","man":true},{"id":2,"studentName":"Illin Zita","className":"12/C","man":false}, {"id":3,"studentName":"Ó  
4,"studentName":"Barta Illdikó","className":"12/C","man":false}, {"id":5,"studentName":"Simony Kata","className":"12/B","man":false}, {"id":6,"studentName":"Csé  
7,"studentName":"Nagy Petra","className":"12/B","man":false}, {"id":8,"studentName":"Vörös Dóra","className":"12/C","man":false}, {"id":9,"studentName":"Varga  
0,"studentName":"Szilágyi Réka","className":"12/C","man":false}, {"id":11,"studentName":"Helle Agnes","className":"12/B","man":false}, {"id":12,"studentName":"G  
13,"studentName":"Farkas Áttila","className":"12/B","man":false}, {"id":14,"studentName":"Barta Zsuzsanna","className":"12/C","man":false}, {"id":15,"studentNam  
id":16,"studentName":"Szabó Dária","className":"12/A","man":true}, {"id":17,"studentName":"Nagy Dorottya","className":"12/C","man":false}, {"id":18,"studentNa  
. {"id":19,"studentName":"Csiszár Nikolett","className":"12/C","man":false}, {"id":20,"studentName":"Kond Lili","className":"12/B","man":false}, {"id":21,"stude  
alse}, {"id":22,"studentName":"Haszkó Fanni","className":"12/B","man":false}, {"id":23,"studentName":"Sós Gábor","className":"12/C","man":true}, {"id":24,"stud  
e}, {"id":25,"studentName":"Magyari Tekla","className":"12/A","man":false}, {"id":26,"studentName":"Nagy Enikő","className":"12/A","man":false}, {"id":27,"stude  
alse}, {"id":28,"studentName":"Kollár Zsófia","className":"12/C","man":false}, {"id":29,"studentName":"Petrás Lili","className":"12/B","man":false}, {"id":30,"stud  
se}, {"id":31,"studentName":"Szabad Éva","className":"12/A","man":false}, {"id":32,"studentName":"Iványi Gréta","className":"12/B","man":false}, {"id":33,"stud  
e}, {"id":34,"studentName":"Toldi Viktória","className":"12/B","man":false}, {"id":35,"studentName":"Torók Robert","className":"12/A","man":true}, {"id":36,"stud  
e"}, {"id":37,"studentName":"Kovács Tíme","className":"12/C","man":false}, {"id":38,"studentName":"Kaszárné Henrietta","className":"12/A","man":false}, {"id":39,"s  
true}, {"id":40,"studentName":"Adám Bella","className":"12/C","man":false}, {"id":41,"studentName":"Hevykői Martin","className":"12/A","man":true}, {"id":42,"s  
e"}, {"id":43,"studentName":"Dobai Flórián","className":"12/C","man":false}, {"id":44,"studentName":"Fekete Zoltán","className":"12/B","man":false}, {"id":45,"stud
```

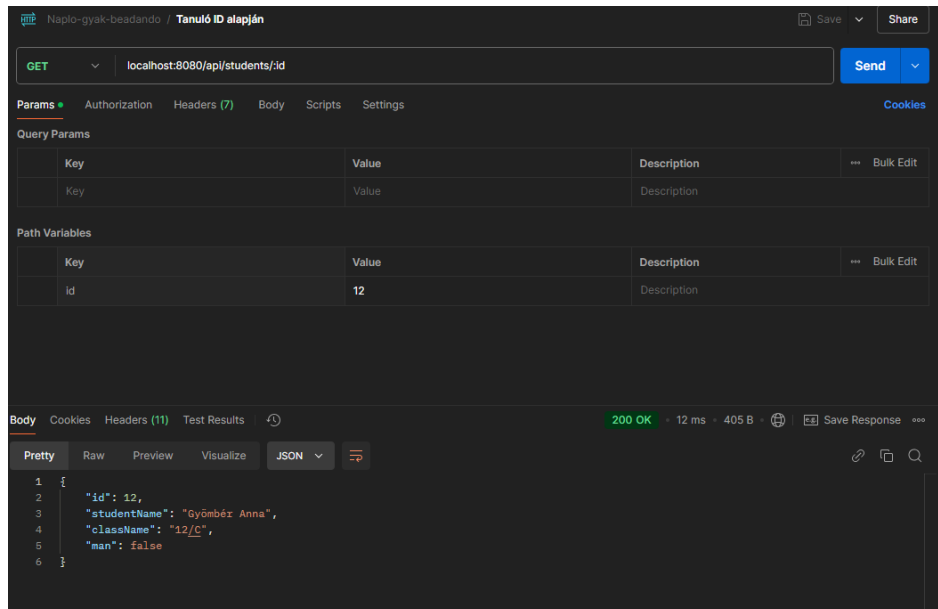
5.1.2 Egy adott tanuló lekérdezése (ID alapján)

HTTP metódus : GET

URL : /api/students/:studentId

Válasz: Visszaadja az adott ID-val rendelkező diák adatait.

postman test:



cUrl teszt:

```
$ curl -i --location 'localhost:8080/api/students/12'
% Total    % Received % Xferd  Average Speed   Time    Time     Current
           %          %       Dload  Upload  Total  Spent    Left     Speed
100  71    0   71    0    0    8924      0 --:--:-- --:--:-- --:--:-- 10142HTTP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json
Transfer-Encoding: chunked
Date: Sun, 01 Dec 2024 12:41:35 GMT

{"id":12,"studentName":"Gyömbér Anna","className":"12/C","man":false}
```

5.2. POST

A POST metódus használatával egy új diákot tudunk felvenni a rendszerünkbe.

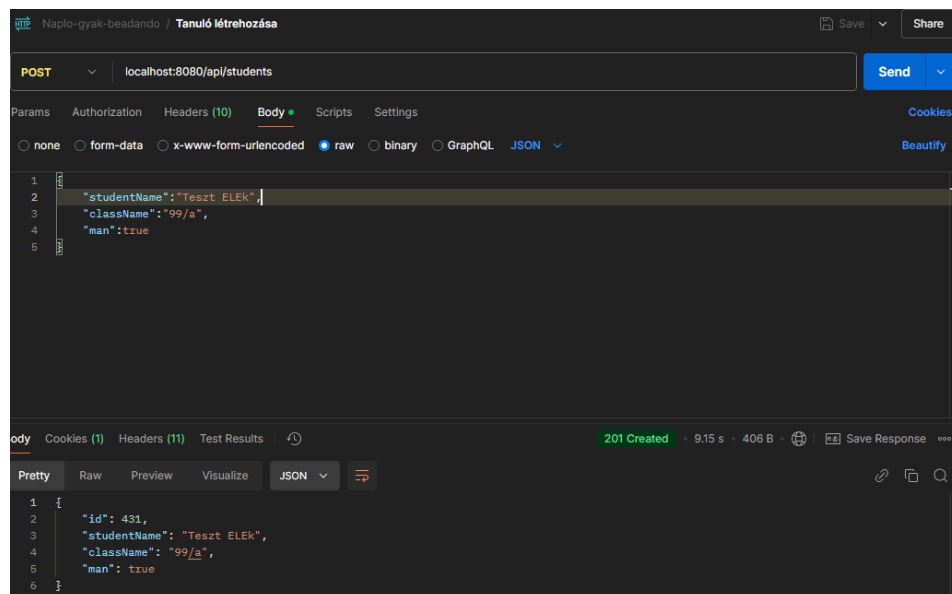
HTTP metódus : POST

URL : /api/students/

Válasz: Visszaadja a felvett diák adatait, beleértve az adatbázis által generált ID-t is.

Ha sikeresen mentésre került a diák, akkor 201-es (CREATED) státuszkóddal tér vissza.

postman teszt:



cUrl teszt:

```
$ curl --location 'localhost:8080/api/students' \
--header 'Content-Type: application/json' \
--header 'Cookie: JSESSIONID=803D196A81768C18E7D4C4AE0C880EE8' \
--data '{
  "studentName": "Teszt ELEK_curl",
  "className": "99/a",
  "man": true
}'
```

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload	Upload	Total	Spent	Left
100	151	0	72	100	79	45	50
					0:00:01	0:00:01	--:--:--

```
96{"id":432,"studentName":"Teszt ELEK_curl","className":"99/a","man":true}
```

5.3. PUT

A PUT metódus használatával egy már meglévő diák adatait tudjuk updatelni.

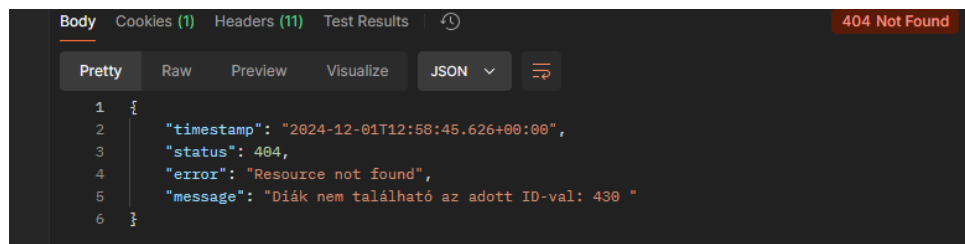
HTTP metódus : PUT

URL : /api/students/:id

Válasz: 200 OK státuszkóddal visszaadja az módosított diák adatait

A teszt során az előzőleg létrehozott Teszt Eleket és a Teszt Elek_curl diákat fogjuk módosítani.

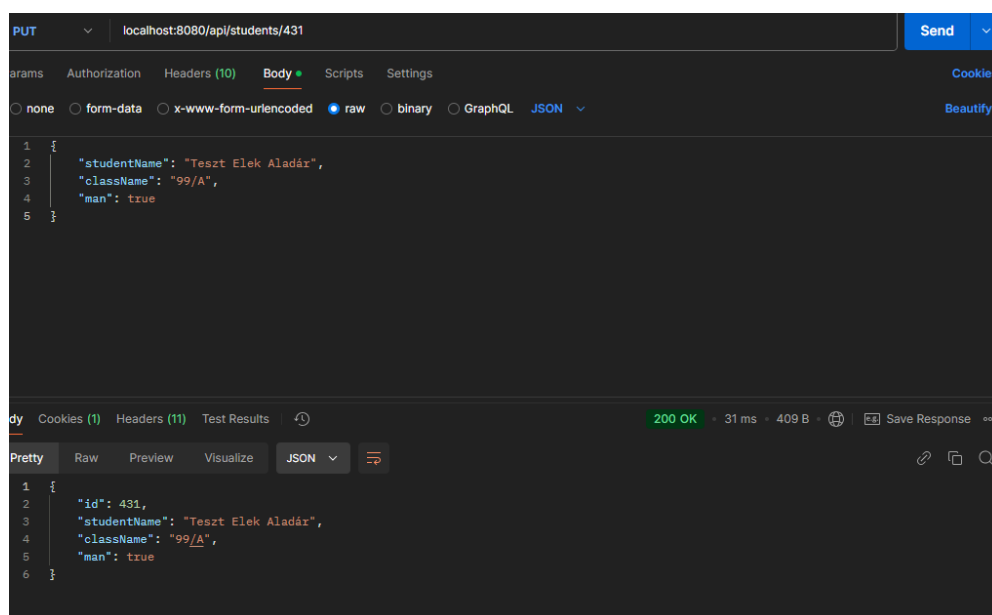
Ha esetleg olyan diák adatait próbáljuk módosítani, akinek nem létezik az Id-ja az adatbázisban, akkor a következő hibát kapjuk.



The screenshot shows a REST client interface with the 'Body' tab selected. The response is a JSON object indicating a 404 error:

```
{
  "timestamp": "2024-12-01T12:58:45.626+00:00",
  "status": 404,
  "error": "Resource not found",
  "message": "Diák nem található az adott ID-val: 430 "
}
```

postman teszt:



The screenshot shows a REST client interface with the 'PUT' method selected for the URL 'localhost:8080/api/students/431'. The request body is a JSON object:

```
{
  "studentName": "Teszt Elek Aladár",
  "className": "99/A",
  "man": true
}
```

 The response is a 200 OK status with a response time of 31 ms and a body size of 409 B. The response body is a JSON object:

```
{
  "id": 431,
  "studentName": "Teszt Elek Aladár",
  "className": "99/A",
  "man": true
}
```

cUrl teszt:

```
curl --location --request PUT 'localhost:8080/api/students/432' \
--header 'Content-Type: application/json' \
--header 'Cookie: JSESSIONID=803D196AB1768C18E7D4C4AE0C880EE8' \
--data '{
  "studentName": "Teszt Elek2",
  "className": "1/A",
  "man": true
}'
```

% Total	% Received	% Xferd	Average	Speed	Time	Time	Time	Current
Dload	Upload	Total	Spent	Left	Speed			
100	144	0	67	100	77	3057	3514	--:--:-- --:--:-- --:--:-- 6857{"id":432,"studentName":"Teszt Elek2","className":"1/A","man":true}

5.4. DELETE

A delete metódust használjuk, ha egy diákot szeretnénk törölni a rendszerből.

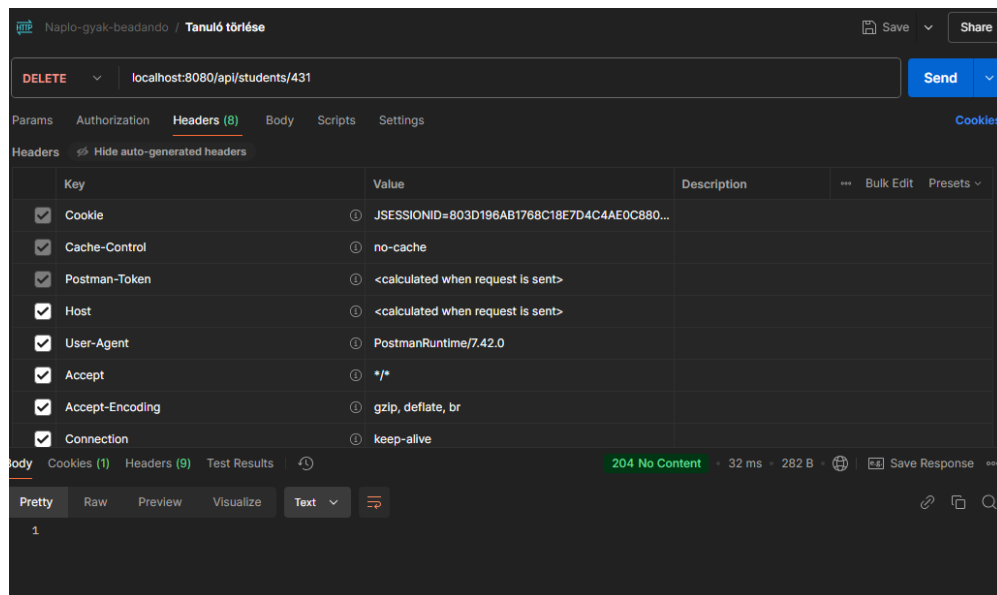
Ilyenkor az adott diák az ID alapján törlésre kerül. Az előző két példával fogjuk tesztelni.

HTTP metódus : DELETE

URL : /api/students/:id

Sikeres törlést követően csak egy 204 (No content) HTTP státuszkóddal rendelkező választ kapunk vissza.

postman teszt:



cUrl:

Természetesen törölni csak egyszer tudjuk, így ha újra megpróbáljuk, akkor szól a rendszer, hogy nem található ilyen diák.

Neumann János Egyetem - Informatika Tanszék
Balogh Norbert - I2I25Q
Torontáli Eszter - J3GE0B
2024

A képen az első a sikeres törlés, a második már a figyelmeztetést mutatja.

```
$ curl --location --request DELETE 'localhost:8080/api/students/432'
% Total    % Received % Xferd  Average Speed   Time    Time     Current
                               Dload  Upload  Total    Spent    Left     Speed
0   0      0    0         0             0          0      0      0      0      0      0
$ herbert@DESKTOP-CMEE75-MINGW64 ~$
$ curl --location --request DELETE 'localhost:8080/api/students/432'
% Total    % Received % Xferd  Average Speed   Time    Time     Current
                               Dload  Upload  Total    Spent    Left     Speed
100  143    0    143      0         0        46    0:00:03    0:00:03    0:00:03    46["timestamp": "2024-12-01T13:06:36.870+00:00", "status": "404", "error": "Resource not found", "message": "Diák nem található az adott ID-v
```