

FINÁLNÍ PROJEKT

č.1



Autor: Ing. Lucie Miková
Datum: 10.7.2024

Okomentoval(a): [ML1]: luciemikova

OBSAH

ZADÁNÍ	3
TESTOVACÍ SCÉNÁŘE	5
EXEKUCE TESTŮ	9
BUG REPORT	16

ZADÁNÍ

Cílem finálního projektu je otestovat funkčnost aplikace, která slouží k manipulaci s daty o studentech. Aplikace má rozhraní REST-API, které umožňuje vytvoření, smazání a získání dat.

Přístupové údaje:

Databáze	database: qa_demo Host: aws.connect.psdb.cloud Username: *** Password: ***
REST-API	http://108.143.193.45:8080/api/v1/students/

Poznámky:

Nezapomeňte, že v IT se data musí někde uložit a poté získat. Proto ověřte, že data jsou správně uložena a získávána z databáze.

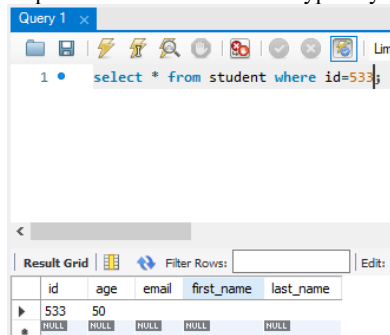
Nezapomeňte do testovacích scénářů uvést testovací data, očekávaný výsledek včetně těla odpovědi a stavových kódů.

Příprava:

Data jsou uložena v databázi „qa_demo“ v tabulce „student“. Ověřila jsem, že je tabulka naplněna údaji o studentech a obsahuje tyto položky: id, age, email, first_name, last_name.

Co jsem si při kontrole tabulky „student“ zjistila je, že u jednotlivých údajů nejsou uloženy všechny položky, tzn. do tabulky lze ukládat údaje bez vyplnění všech položek a to není správně.

Např. student s id=533 nemá vyplněný email, first_name, last_name:



Query 1

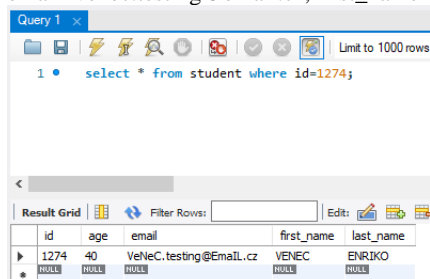
```
1 • select * from student where id=533;
```

Result Grid

	id	age	email	first_name	last_name
▶	533	50			
*	NULL	NULL	NULL	NULL	NULL

Další chybou je, že všechny údaje nejsou ukládány ve formátu „lower case“, což by být měly.

Např. student s id=1274 je uložen id=1274, age=40, email=VeNeC.testing@EmaIL.cz, first_name=VENEC, last_name=ENRIKO, správná podoba by měla být id=1274, age=40, email=venec.testing@email.cz, first_name=venec, last_name=enriko.



Query 1

```
1 • select * from student where id=1274;
```

Result Grid

	id	age	email	first_name	last_name
▶	1274	40	VeNeC.testing@EmaIL.cz	VENEC	ENRIKO
*	NULL	NULL	NULL	NULL	NULL

Další chybou je uložení do položky age s mínusem.

Např. student s id=264 je uložen s věkem -35 let, což je samozřejmě nelogický údaj. Zde musí mít ošetřeno, aby se neukládalo se znaménkem mínus.

Query 1 x

Limit to 1000

```
1 • select * from student where id=264;
```

Result Grid

id	age	email	first_name	last_name
264	-35	jjohn34@yourmail.com	John	JOSHUA
NULL	NULL	NULL	NULL	NULL

Další chybou je uložení do položky email ve formě, která není email.

Např. student s id=407 je uložen s emailem „j2222“. Očekávala bych zde tedy nějakou kontrolu položky na správný formát emailu.

Query 1 x

Limit

```
1 • select * from student where id=407;
```

Result Grid

id	age	email	first_name	last_name
407	99	j2222	Janko	TEST
NULL	NULL	NULL	NULL	NULL

Další nalezená chyba je ukládání různých nesmyslných symbolů a znaků.
Např. studenti s id=306, 547, 603 aj.

Query 1 x

Limit to 1000 rows

```
1 • select * from student where id in ("306", "547", "603");
```

Result Grid

id	age	email	first_name	last_name
306	963	yá+ěčzčš1	-86	*+6
547	37		<html><head><title></title></head>	<!DOCTYPE HTML>
603	-15	-2/3	-33.3@	-12/2@
NULL	NULL	NULL	NULL	NULL

TESTOVACÍ SCÉNÁŘE

Na základě těchto uvedených testovacích scénářů jsem ověřila funkčnost aplikace:

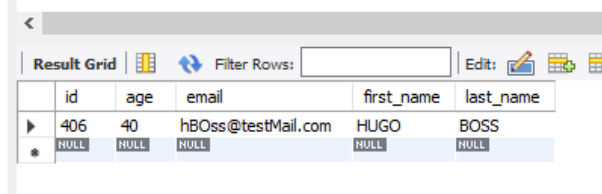
Metoda GET

1. Abstract: hledám data o existujícím studentovi

Konkrétní podoba dat: hledám všechny údaje z databáze „student“ o studentovi, který existuje a má id 406:

```
select * from student where id=406;
```

Ano, v databázi skutečně student s id=406 existuje, zde přikládám výsledek:



The screenshot shows a database interface with a 'Result Grid' tab. The grid contains one row of data for student ID 406. The columns are id, age, email, first_name, and last_name. The values are 406, 40, hBOss@testMail.com, HUGO, and BOSS respectively. There is also a row of NULL values below the first row.

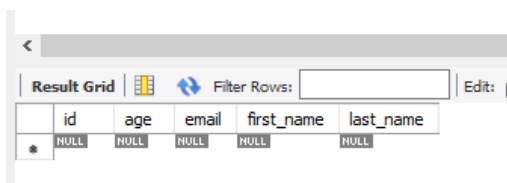
	id	age	email	first_name	last_name
▶	406	40	hBOss@testMail.com	HUGO	BOSS
*	NULL	NULL	NULL	NULL	NULL

2. Abstract: hledám data o neexistujícím studentovi

Konkrétní podoba dat: hledám všechny údaje z databáze „student“ o studentovi, který neexistuje, resp. Hledám, že v databázi žádné údaje o něm nebudou, hledám studenta s id=012:

```
select * from student where id="012";
```

Ano, v databázi skutečně student s id=012 neexistuje, zde dokládám výsledek:



The screenshot shows a database interface with a 'Result Grid' tab. The grid contains one row of NULL values. The columns are id, age, email, first_name, and last_name. The values are NULL, NULL, NULL, NULL, and NULL respectively.

	id	age	email	first_name	last_name
*	NULL	NULL	NULL	NULL	NULL

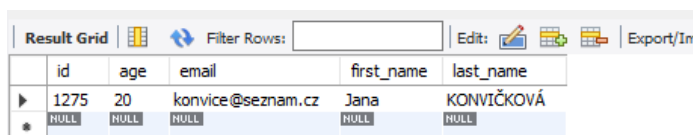
Metoda DELETE

3. Abstract: smažu všechny údaje s existujícím studentem

Konkrétní podoba dat: smažu všechny údaje z databáze se studentem, který v databázi existuje a má id=1275.

Prvně v databázi ověřím, že student s id=1275 existuje:

```
select * from student where id=1275;
```



The screenshot shows a 'Result Grid' interface with a search bar and buttons for 'Filter Rows', 'Edit', and 'Export/Import'. The table has columns: id, age, email, first_name, and last_name. The first row contains the data for student id 1275.

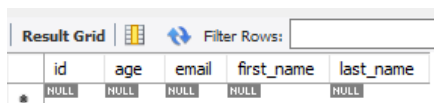
	id	age	email	first_name	last_name
▶	1275	20	konvice@seznam.cz	Jana	KONVIČKOVÁ
*	NULL	NULL	NULL	NULL	NULL

Ano, student existuje a lze jej tedy pomocí metody „delete“ smazat.

4. Abstract: budu se snažit smazat údaje studenta, který neexistuje

Konkrétní podoba dat: pomocí selectu si najdu studenta, který v databázi není. Takový je například s id=255 a dále jej pomocí metody „delete“ budu chtít smazat. Očekávaným výsledkem je status code 404 – Page Not Found.

```
select * from student where id=255;
```



The screenshot shows a 'Result Grid' interface with a search bar and buttons for 'Filter Rows', 'Edit', and 'Export/Import'. The table has columns: id, age, email, first_name, and last_name. All cells in the data row are NULL.

	id	age	email	first_name	last_name
*	NULL	NULL	NULL	NULL	NULL

Metoda POST

5. Abstrakt: zavedu nového studenta, který v databázi „student“ ještě neexistuje.
Očekávaným výstupem je, že student úspěšně založí se všemi správně naplněnými položkami a status code bude 201 created.

Konkrétní podoba dat: založím nového studenta s first_name=VeNec, last_name=eNRicO, email=VeNEc.Enr@seZNaM.cz, age=20.

6. Abstrakt: budu se snažit založit studenta, který nemá všechny atributy (tzn. age, email, first_name, last_name). Očekávaným výstupem je status code 400 bad request. A student nebude založen.

Konkrétní podoba dat: budu se snažit založit studenta s last_name=Novák, email=test.testing@email.cz, vek=40 (ale bez first_name).


EXEKUCE TESTŮ

Testovací scénáře viz výše jsem provedla a příkládám výsledky testů:

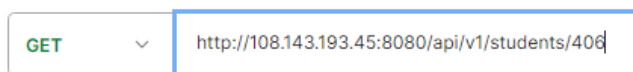
Metoda GET

1. Abstract: hledám data o existujícím studentovi

select * from student where id=406;



	id	age	email	first_name	last_name
▶	406	40	hBOss@testMail.com	HUGO	BOSS
*	NULL	NULL	NULL	NULL	NULL



GET

Body Cookies Headers (5) Test Results

200 OK

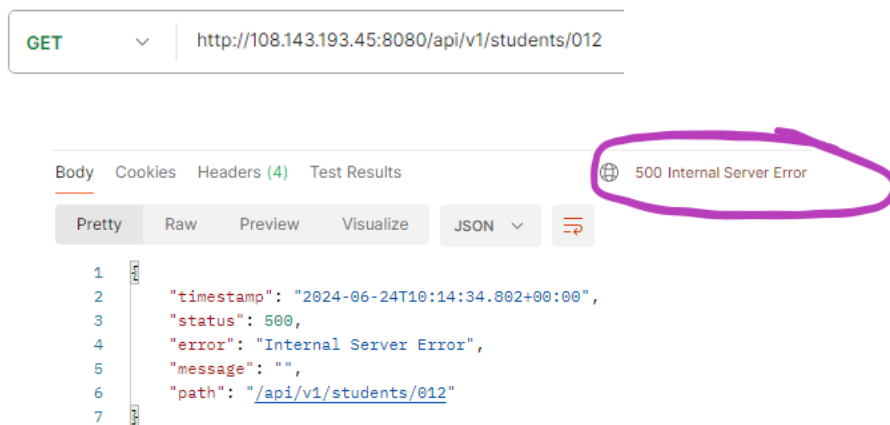
Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 406,
3   "firstName": "HUGO",
4   "lastName": "BOSS",
5   "email": "hBOss@testMail.com",
6   "age": 40
7 }
```

Očekávaný výsledek je v pořádku, zobrazil se mi student se správnými výsledky a správným status code 200 – OK.

Tento testovací scénář prošel.

2. Abstract: hledám data o neexistujícím studentovi. Očekávaným výsledkem je status code 404 – Page Not Found.



Provedením tohoto scénáře jsem odhalila bug – místo očekávaného status code 404 je výsledkem status code 500 – Internal Server Error. Tento scénář tedy neprošel, více o něm v kapitole Bug report (**bug č. 1**).

Metoda DELETE

3. Abstract: smažu všechna data z databáze se studentem, který v databázi existuje a má id=1275

Požadavek byl úspěšný, resp. Status Code=200 OK (success status response code) a všechna data o studentovi s id=1275 jsou smazána.

The screenshot shows a REST client interface. At the top, the URL is `http://108.143.193.45:8080/api/v1/students/012`. Below it, the method is set to **DELETE** and the URL is `http://108.143.193.45:8080/api/v1/students/1275`. The **Body** tab is selected, showing a JSON payload:

```
{  "firstName": "Jana",  "lastName": "Konvičková",  "email": "konvice@seznam.cz",  "age": 20}
```

. The status bar at the bottom indicates a successful response: **200 OK** with a response time of 126 ms and a size of 123 B. The response body is empty.

V dalším kroku tohoto scénáře ověřím v databázi, že je student skutečně smazaný:

```
select * from student where id=1275;
```

Result Grid					
Filter Rows:					
	id	age	email	first_name	last_name
*	NULL	NULL	NULL	NULL	NULL

Ano, select v databázi žádného studenta s id=1275 nenašel.

Tento testovací scénář prošel bez chyby.

4. Abstract: budu se snažit smazat všechny údaje se studentem, který neexistuje a má id=255. Očekávaným výsledkem je status code 404 – Page Not Found.

The screenshot shows a REST client interface. At the top, the URL bar displays `http://108.143.193.45:8080/api/v1/students/255`. Below it, the method is set to `DELETE`. The request body is empty. The response status is `500 Internal Server Error` with a response time of `23 ms` and a size of `287 B`. The response body is displayed in JSON format:

```
1 {
2   "timestamp": "2024-06-22T09:27:02.543+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "message": "",
6   "path": "/api/v1/students/255"
7 }
```

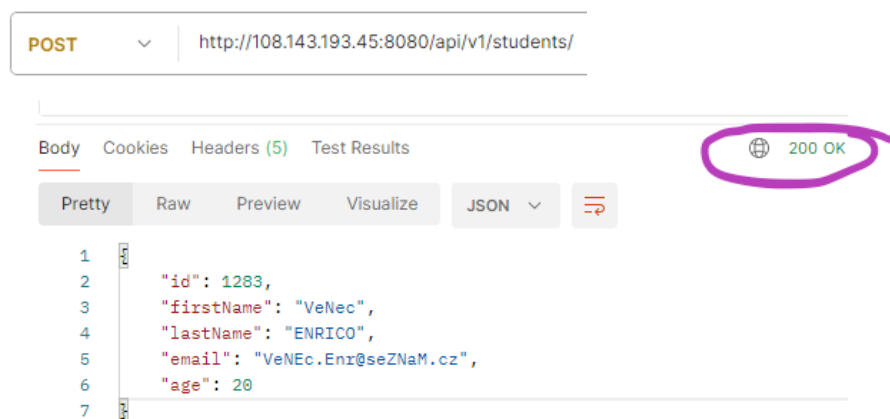
Provedením tohoto scénáře jsem odhalila bug – místo očekávaného status code 404 je výsledkem status code 500 – Internal Server Error. Tento scénář tedy neprošel, více o něm v kapitole Bug report (**bug č. 2**).

Metoda POST

5. Abstrakt: zakládám nového studenta. Očekávaný stav je status code 201 created.

```
{
  "firstName": "VeNec",
  "lastName": "eNRico",
  "email": "VeNEc.Enr@seZNaM.cz",
  "age": 20
}
```

Student byl založen s id 1283:



Výsledkem je status code 200 OK (místo očekávaného 201 created), označuji tedy jako **bug č. 3** (viz kapitola Bug report).

V databázi zkontrolováno, student byl opravdu založen se všemi položkami:

select * from student where id=1283;

	id	age	email	first_name	last_name
▶	1283	20	VeNEc.Enr@seZNaM.cz	VeNec	ENRICO
*	NULL	NULL	NULL	NULL	NULL

Nicméně při testování tohoto scénáře jsem zjistila, jako už při kontrole samotné tabulky „student“ na začátku tohoto projektu, že se student neukládá ve správném formátu „lower case“, že lze uložit různé symboly a znaky a také prázdné položky... Tyto chyby uvádím dále jako **bug č. 4**, více o nich na začátku v kapitole „Příprava“.

6. Abstrakt: zakládám studenta, který nemá všechny atributy

```
{
  "lastName": "Novák",
  "email": "test.testing@email.cz",
  "age": 40
}
```

U vkládaného studenta chybí atribut „firstName“, tudíž očekávaným výsledkem byl status code 400 bad request. Při založení studenta se mi však zobrazil status code 500, tudíž označuji jako **bug č. 5** a více viz kapitola Bug report.

The screenshot shows a REST client interface. At the top, a POST request is shown to the URL `http://108.143.193.45:8080/api/v1/students/012`. Below this, the request details are shown for the endpoint `http://108.143.193.45:8080/api/v1/students/`. The request body is in JSON format:

```
1 {
2   "firstName": "Novák",
3   "email": "test.testing@email.cz",
4   "age": 40
5 }
```

The response status is **500 Internal Server Error**. The response body is shown in JSON format:

```
1 {
2   "timestamp": "2024-07-03T15:12:47.084+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "message": "",
6   "path": "/api/v1/students/"
7 }
```

Další kontrolou bude, že se tento student nezaložil. Jelikož založení studenta proběhlo s chybou status code 500 a ani nemám jeho id, nepředpokládám, že by byl založen, nicméně raději zkontrolováno v databázi „student“, že student opravdu nebyl založen:

The screenshot shows the DBeaver SQL editor interface. At the top, the title bar reads "Query 1". Below it is a toolbar with various icons for file operations, editing, and viewing. The main text area contains the following SQL query:

```
1 select * from student where first_name="Novák" and email= "test.testing@email.cz" and age="40";
```

Below the query editor, there is a "Result Grid" section. It includes a "Filter Rows:" input field, an "Edit:" button, and an "Export/Import:" button. The results are displayed in a table with the following columns: id, age, email, first_name, and last_name. The first row of data shows:

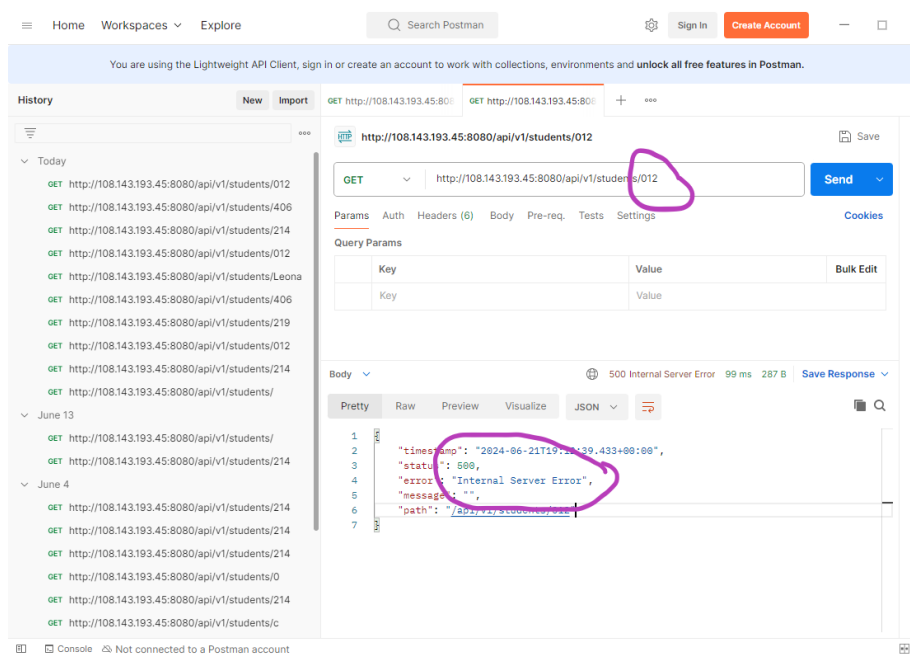
id	age	email	first_name	last_name
1	40	test.testing@email.cz	Novák	

BUG REPORT

Na základě provedených scénářů jsem objevila uvedené chyby aplikace:

Metoda GET: bug č. 1

Scénář č. 2 (hledání studenta, který neexistuje) neprošel a odhalil tuto chybu: metoda GET měla zobrazit výsledek s chybovou hláškou 404 Not found, nicméně zobrazila chybu 500 Internal Server Error, což ukazuje níže doložený screenshot.



Metoda DELETE: bug č. 2

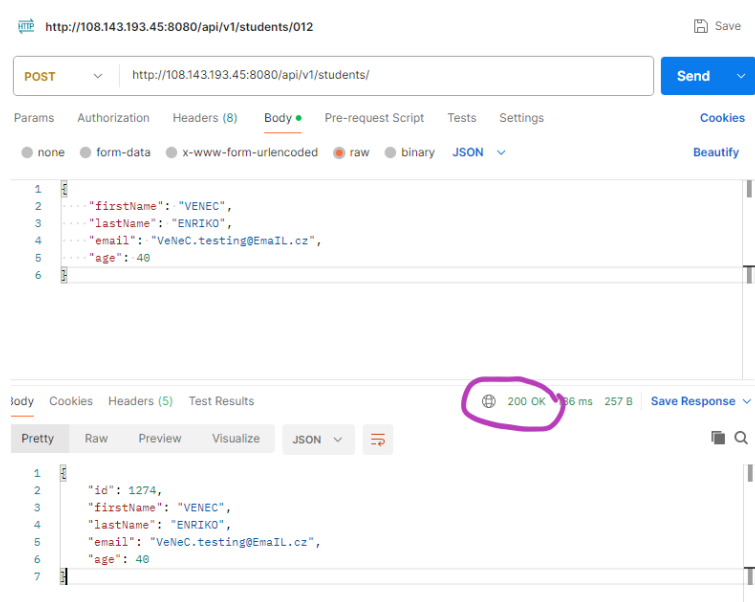
Scénář č. 4 (snažím se smazat studenta, který neexistuje) neprošel a odhalil tuto chybu: metoda DELETE měla zobrazit výsledek s chybovou hláškou 404 Not found, nicméně zobrazila chybu 500 Internal Server Error, stejně jako v případě bugu č. 1 (dokládám screenshotem níže):

The screenshot shows a REST client interface. At the top, the URL bar displays `http://108.143.193.45:8080/api/v1/students/255`. Below it, the method is set to `DELETE` and the same URL is repeated. The interface includes tabs for Params, Authorization, Headers (9), Body, Postman Script, Tests, and Settings. Under the Headers tab, there is a table for Query Params with two columns: Key and Value. The Body tab is selected, showing a JSON response. The response status is `500 Internal Server Error` with a response time of 23 ms and a size of 287 B. The JSON body is as follows:

```
1 {
2   "timestamp": "2024-06-22T09:27:02.543+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "message": "",
6   "path": "/api/v1/students/255"
7 }
```

Metoda POST: bug č. 3

Scénář č. 5 (zakládám nového studenta) odhalil chybu, že místo očekávaného status code 201 created je nesprávný status code 200 OK, což níže dokládám:



The screenshot shows a REST client interface with a POST request to `http://108.143.193.45:8080/api/v1/students/`. The request body is a JSON object: `{ "firstName": "VENEC", "lastName": "ENRIKO", "email": "VeNeC.testing@Email.cz", "age": 40 }`. The response status is `200 OK`, which is circled in purple. The response body is a JSON object: `{ "id": 1274, "firstName": "VENEC", "lastName": "ENRIKO", "email": "VeNeC.testing@Email.cz", "age": 40 }`.

```
POST http://108.143.193.45:8080/api/v1/students/

{
  "firstName": "VENEC",
  "lastName": "ENRIKO",
  "email": "VeNeC.testing@Email.cz",
  "age": 40
}
```

200 OK 16 ms 257 B

```
{
  "id": 1274,
  "firstName": "VENEC",
  "lastName": "ENRIKO",
  "email": "VeNeC.testing@Email.cz",
  "age": 40
}
```

Metoda POST: bug č. 4

Při zakládání studenta bylo odhaleno více chyb – blíže již specifikováno výše v kapitole „Příprava“.

Metoda POST: bug č. 5

Scénář č. 6 (zakládám studenta, který nemá všechny atributy). Zde se mi podařilo opět odhalit chybu, kdy místo očekávaného výsledku status code 400 bad request byl status code 500 Internal Server Error.

The screenshot shows a REST client interface with a POST request to `http://108.143.193.45:8080/api/v1/students/`. The request body is a JSON object: `{ "firstName": "Novák", "email": "test.testing@email.cz", "age": 40 }`. The response is a 500 Internal Server Error with the following JSON body: `{ "timestamp": "2024-07-03T15:12:47.084+00:00", "status": 500, "error": "Internal Server Error", "message": "", "path": "/api/v1/students/" }`. The error status is circled in purple.

```
POST http://108.143.193.45:8080/api/v1/students/

{
  "firstName": "Novák",
  "email": "test.testing@email.cz",
  "age": 40
}
```

Body Cookies Headers (4) Test Results 500 Internal Server Error

```
1
2 {
3   "timestamp": "2024-07-03T15:12:47.084+00:00",
4   "status": 500,
5   "error": "Internal Server Error",
6   "message": "",
7   "path": "/api/v1/students/"
}
```