

FINÁLNÍ PROJEKT

č.1



ENGETO

Autor: Vojtěch Machytka
Datum: 12.12.2024

OBSAH

ZADÁNÍ	4
Přístupové údaje:	4
TESTOVACÍ SCÉNÁŘE	5
1. FUNKCE GET	5
1. Získat data o existujícím studentovi	5
2. Získat data o neexistujícím studentovi	5
2. FUNKCE POST	6
1. Založit úspěšně nového studenta	6
2. Založit studenta staršího než 150 let	6
3. Založit studenta s neplatným formátem emailu	7
4. Založit studenta pouze s křestním jménem	7
5. Založit studenta s číslem místo příjmení	7
6. Založit studenta bez uvedení věku	8
3. FUNKCE DELETE	9
1. Smazat existujícího studenta	9
2. Smazat neexistujícího studenta	9
EXEKUCE TESTŮ	10
1. FUNKCE GET	10
1. Získat data o existujícím studentovi	10
2. Získat data o neexistujícím studentovi	10
2. FUNKCE POST	11
1. Založit úspěšně nového studenta	11
2. Založit studenta staršího než 150 let	11
3. Založit studenta s neplatným formátem emailu	12
4. Založit studenta pouze s křestním jménem	12
5. Založit studenta s číslem místo příjmení	13
6. Založit studenta bez uvedení věku	13
3. FUNKCE DELETE	14

1. Smazat existujícího studenta	14
2. Smazat neexistujícího studenta	14
BUG REPORT	15

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/

TESTOVACÍ SCÉNÁŘE

1. FUNKCE GET

1. Získat data o existujícím studentovi

Testovací data:

ID studenta = 1127

V MySQLWorkbench:

```
SELECT * FROM student where id=1127;
```

```
'1127','19','jc34@yourmail.com','John','SCOTT'
```

Steps:

1.) GET <http://108.143.193.45:8080/api/v1/students/1127>

2.) Press Send button

Expected result:

1.) status code 200 OK

2.) Response

```
{  
  "id": 1127,  
  "firstName": "John",  
  "lastName": "SCOTT",  
  "email": "jc34@yourmail.com",  
  "age": 19  
}
```

2. Získat data o neexistujícím studentovi

Testovací data:

ID studenta = 446

V MySQLWorkbench:

```
SELECT * FROM student where id=446;
```

0 rows returned

Steps:

1.) GET <http://108.143.193.45:8080/api/v1/students/446>

2.) Press Send button

Expected result:

1.) status code 404 Not found

2. FUNKCE POST

1. Založit úspěšně nového studenta

Steps:

Zvolím POST

Request body JSON:

```
{  
  "firstName": "Carl",  
  "lastName": "Bananaman",  
  "email": "cbnman@yourmail.com",  
  "age": 42  
}
```

Expected results:

1.) status code 201 created

2.) Response:

```
{  
  "id": 1450,  
  "firstName": "Carl",  
  "lastName": "Bananaman",  
  "email": "cbnman@yourmail.com",  
  "age": 42  
}
```

2. Založit studenta staršího než 150 let

Steps:

Zvolím POST

Request body JSON:

```
{  
  "firstName": "Frankie",  
  "lastName": "Oldman",  
  "email": "oldfrankie@aol.com",  
  "age": 238  
}
```

Expected results:

1.) status code 400 BAD REQUEST

3. Založit studenta s neplatným formátem emailu

Steps:

Zvolím POST

Request body JSON:

```
{  
  "firstName": "John",  
  "lastName": "Bolito",  
  "email": "johnjohn.aol.com",  
  "age": 23  
}
```

Expected results:

1.) status code 400 BAD REQUEST

4. Založit studenta pouze s křestním jménem

Steps:

Zvolím POST

Request body JSON:

```
{  
  "firstName": "Wong",  
  "lastName": "",  
  "email": "mrwong@gmail.com",  
  "age": 19  
}
```

Expected results:

1.) status code 400 BAD REQUEST

5. Založit studenta s číslem místo příjmení

Steps:

Zvolím POST

Request body JSON:

```
{  
  "firstName": "Karel",  
  "lastName": "4",  
  "email": "otecvlasti@praha.cz",  
  "age": 52  
}
```

Expected results:

1.) status code 400 BAD REQUEST

6. Založit studenta bez uvedení věku

Steps:

Zvolím POST

Request body JSON:

```
{  
  "firstName": "Ruda",  
  "lastName": "Rudy",  
  "email": "rudrud@nope.se",  
  "age":  
}
```

Expected results:

1.) status code 400 BAD REQUEST

3. FUNKCE DELETE

1. Smazat existujícího studenta

Steps:

1.) DELETE <http://108.143.193.45:8080/api/v1/students/1450>

2.) Press Send button

Expected results:

1.) status code 200 OK

2. Smazat neexistujícího studenta

Steps:

1.) DELETE <http://108.143.193.45:8080/api/v1/students/1450>

2.) Press Send button

Expected results:

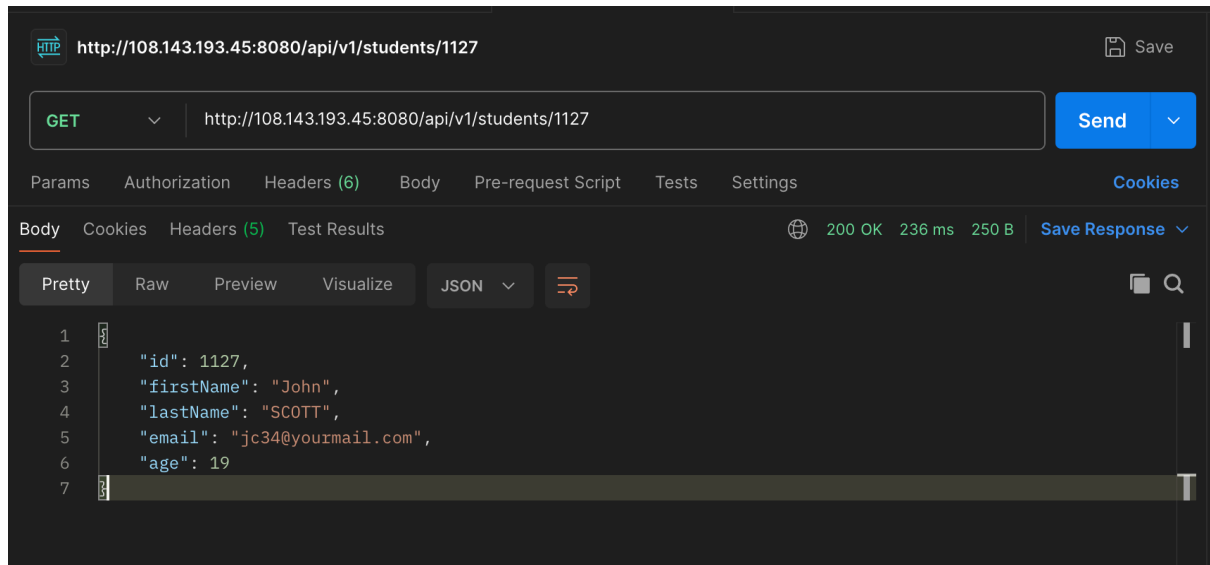
1.) status code 404 Not found

EXEKUCE TESTŮ

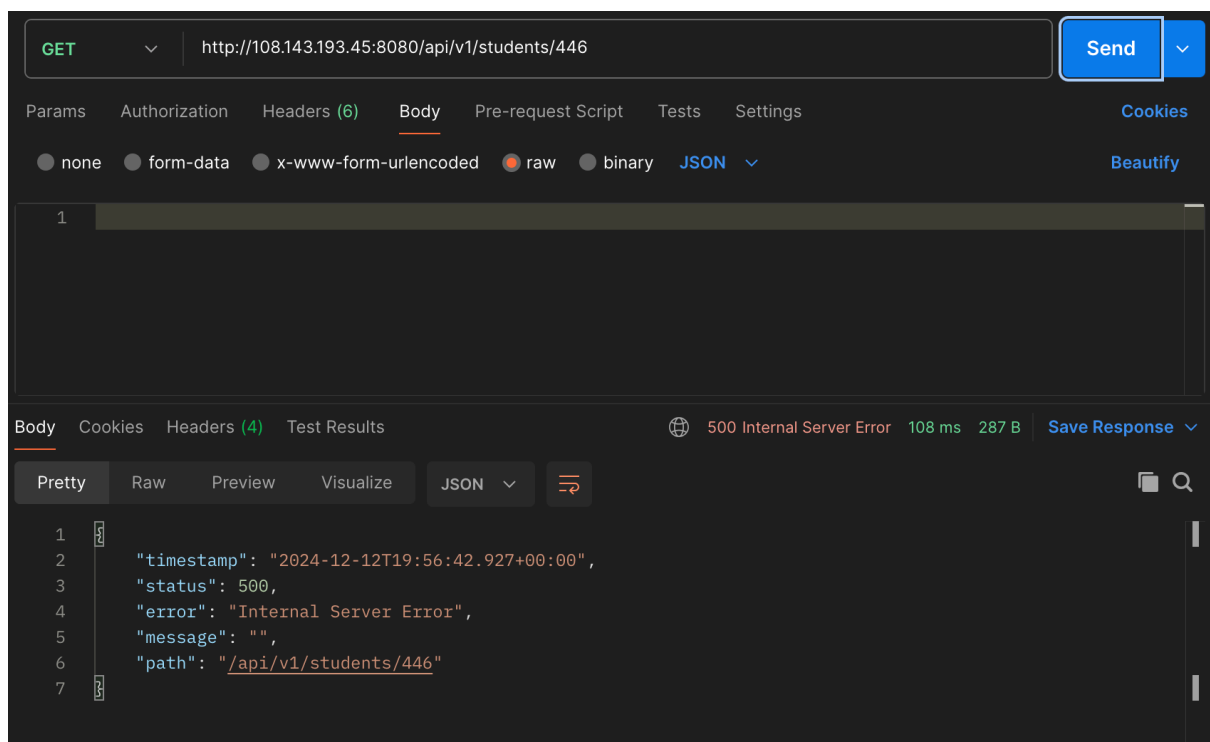
Testovací scénáře jsem provedl, přikládám výsledky testů.

1. FUNKCE GET

1. Získat data o existujícím studentovi



2. Získat data o neexistujícím studentovi



1. FUNKCE POST

1. Založit úspěšně nového studenta

The screenshot shows the Postman interface for a POST request to `http://108.143.193.45:8080/api/v1/students/`. The request body is a JSON object representing a student: `{ "firstName": "Carl", "lastName": "Bananaman", "email": "cbnman@yourmail.com", "age": 42 }`. The response is a 200 OK status with a response time of 183 ms and a size of 256 B. The response body is a JSON object: `{ "id": 1450, "firstName": "Carl", "lastName": "BANANAMAN", "email": "cbnman@yourmail.com", "age": 42 }`.

```
1 {
2   "firstName": "Carl",
3   "lastName": "Bananaman",
4   "email": "cbnman@yourmail.com",
5   "age": 42
6 }
```

Body Cookies Headers (5) Test Results 200 OK 183 ms 256 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1450,
3   "firstName": "Carl",
4   "lastName": "BANANAMAN",
5   "email": "cbnman@yourmail.com",
6   "age": 42
7 }
```

2. Založit studenta staršího než 150 let

The screenshot shows the Postman interface for a POST request to `http://108.143.193.45:8080/api/v1/students/`. The request body is a JSON object representing a student: `{ "firstName": "Frankie", "lastName": "Oldman", "email": "oldfrankie@aol.com", "age": 238 }`. The response is a 200 OK status with a response time of 182 ms and a size of 256 B. The response body is a JSON object: `{ "id": 2619, "firstName": "Frankie", "lastName": "OLDMAN", "email": "oldfrankie@aol.com", "age": 238 }`.

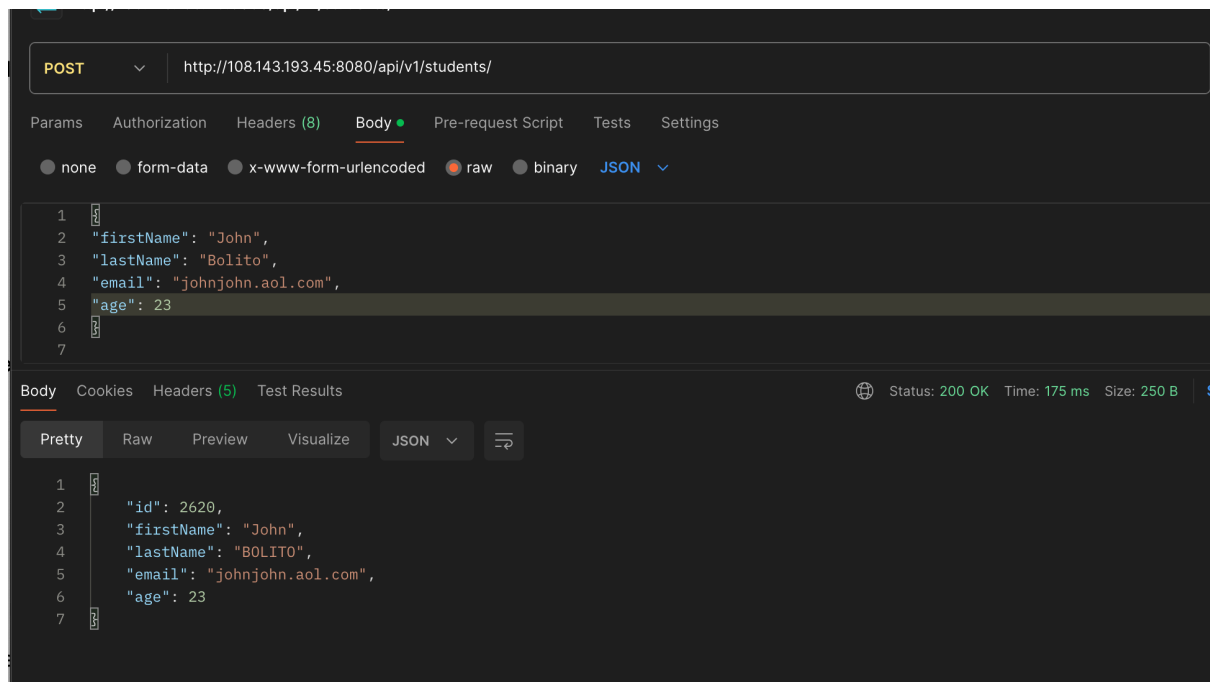
```
1 {
2   "firstName": "Frankie",
3   "lastName": "Oldman",
4   "email": "oldfrankie@aol.com",
5   "age": 238
6 }
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 182 ms Size: 256 B

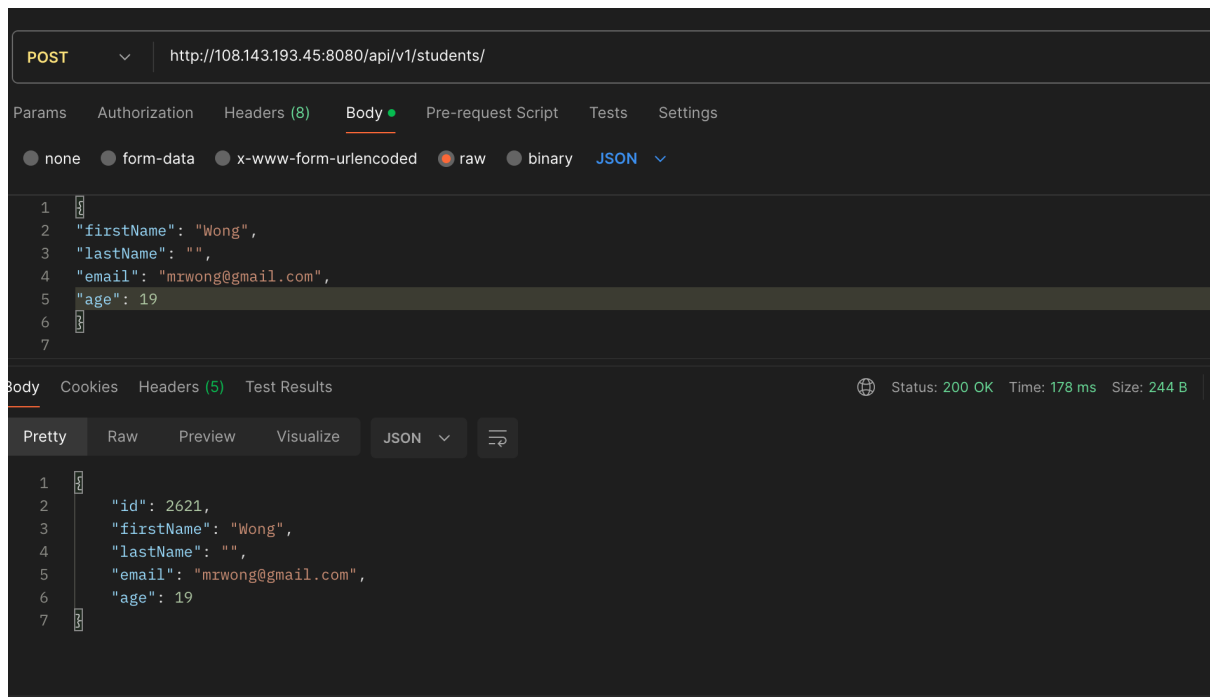
Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 2619,
3   "firstName": "Frankie",
4   "lastName": "OLDMAN",
5   "email": "oldfrankie@aol.com",
6   "age": 238
7 }
```

3. Založit studenta s neplatným formátem emailu



4. Založit studenta pouze s křestním jménem



5. Založit studenta s číslem místo příjmení

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": "Karel", "lastName": "4", "email": "otecvlasti@praha.cz", "age": 52}`. The response status is 200 OK, and the response body is a JSON object: `{"id": 2622, "firstName": "Karel", "lastName": "4", "email": "otecvlasti@praha.cz", "age": 52}`.

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

{
  "firstName": "Karel",
  "lastName": "4",
  "email": "otecvlasti@praha.cz",
  "age": 52
}
```

Status: 200 OK Time: 148 ms Size: 249 B

```
{
  "id": 2622,
  "firstName": "Karel",
  "lastName": "4",
  "email": "otecvlasti@praha.cz",
  "age": 52
}
```

6. Založit studenta bez uvedení věku

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": "Ruda", "lastName": "Rudy", "email": "rudrud@nope.se", "age": ""}`. The response status is 400 Bad Request, and the response body is a JSON object: `{"timestamp": "2024-12-12T21:04:46.677+00:00", "status": 400, "error": "Bad Request", "message": "", "path": "/api/v1/students/"}`.

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

{
  "firstName": "Ruda",
  "lastName": "Rudy",
  "email": "rudrud@nope.se",
  "age": ""
}
```

Status: 400 Bad Request Time: 45 ms Size: 264 B

```
{
  "timestamp": "2024-12-12T21:04:46.677+00:00",
  "status": 400,
  "error": "Bad Request",
  "message": "",
  "path": "/api/v1/students/"
}
```

3. FUNKCE DELETE

1. Smazat existujícího studenta

The screenshot shows a REST client interface with the URL `http://108.143.193.45:8080/api/v1/students/1450`. The method is set to **DELETE**. The **Body** tab is selected, showing the message "This request does not have a body". The response status is **200 OK** with a response time of **126 ms** and a size of **123 B**. The response body is empty.

2. Smazat neexistujícího studenta

The screenshot shows the same REST client interface, but the response status is **500 Internal Server Error** with a response time of **93 ms** and a size of **288 B**. The response body is displayed in JSON format:

```
1 {
2   "timestamp": "2024-11-22T18:53:06.568+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "message": "",
6   "path": "/api/v1/students/1450"
7 }
```

BUG REPORT

Na základě provedených scénářů jsem objevil uvedené chyby aplikace:

- 1. U založení nového studenta je odpověď status 200 OK místo požadovaného 201 Created.*
- 2. U funkce get pro neexistující studenta je odpověď status 500 místo požadovaného 404 Not found.*
- 3. U založení nového studenta staršího než 150 let status 200 OK místo statusu 400 Bad request.*
- 4. U založení nového studenta s neplatným formátem emailu status 200 OK místo statusu 400 Bad request.*
- 5. U založení nového studenta bez lastName status 200 OK místo statusu 400 Bad request.*
- 6. U založení nového studenta číslem místo příjmení status 200 OK místo statusu 400 Bad request.*
- 7. U pokusu o smazání neexistující studenta je odpověď status 500 místo statusu 404 Not found*