

# FINÁLNÍ PROJEKT

## č.1



Autor: David Oubrecht  
Datum: 04.11.2024

## OBSAH

<b>ZADÁNÍ</b>	<b>3</b>
<b>TESTOVACÍ SCÉNÁŘE</b>	<b>4</b>
<b>EXEKUCE TESTŮ</b>	<b>14</b>
<b>BUG REPORT</b>	<b>32</b>

# 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 Port: 3306 Username: xxx Password: xxx
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.

# TESTOVACÍ SCÉNÁŘE

Na základě uvedených testovacích scénářů jsem ověřil(a) funkčnost aplikace.

## Název scénáře:

GET metoda

## Cíl testu:

Získání dat o studentovi

**Scénář 1:** Úspěšné načtení existujícího studenta.

- **Předpoklady:** Student s ID 1968 existuje v databázi.
- **Kroky:**
  1. Vytvoř v aplikaci Postman HTTP Request
  2. Zavolej metodu GET  
`http://108.143.193.45:8080/api/v1/students/1968`
  3. Zkontroluj v databázi pomocí nástroje MySQL Workbench že student s ID 1968 je stejný
- **Očekávaný výsledek:** Server vrátí informace o studentovi s ID 1968.

**Scénář 2:** Načtení neexistujícího studenta.

- **Předpoklady:** Student s ID 9999 neexistuje v databázi.
- **Kroky:**
  1. Vytvoř v aplikaci Postman HTTP Request
  2. Zavolej metodu GET  
`http://108.143.193.45:8080/api/v1/students/9999`
  3. Zkontroluj v databázi pomocí nástroje MySQL Workbench že student s ID 9999 neexistuje
- **Očekávaný výsledek:** Server vrátí odpověď s chybou 404 a zprávu "Student nenalezen".

## Název scénáře:

DELETE metoda

## Cíl testu:

Smazání studenta

1. **Scénář 1:** Úspěšné smazání existujícího studenta.
  - **Předpoklady:** Student s ID 1951 existuje v databázi.
  - **Kroky:**
    1. Vytvoř v aplikaci Postman HTTP Request
    2. Zavolej metodu DELETE  
`http://108.143.193.45:8080/api/v1/student/1951`
    3. Zkontroluj v databázi že student byl smazán
  - **Očekávaný výsledek:** Server vrátí stavový kód 200, zprávu "Student úspěšně smazán." a student bude odstraněn z databáze.
  
2. **Scénář 2:** Pokus o smazání neexistujícího studenta.
  - **Předpoklady:** Student s ID 9999 neexistuje v databázi.
  - **Kroky:**
    1. Vytvoř v aplikaci Postman HTTP Request
    2. Zavolej metodu DELETE  
`http://108.143.193.45:8080/api/v1/students/9999`
  - **Očekávaný výsledek:** Server vrátí stavový kód 404 a zprávu "Student nenalezen".

## Název scénáře:

POST metoda

## Cíl testu:

Vytvoření nového studenta v databázi

**Scénář 1:** Úspěšné vytvoření studenta s validními daty

- **Předpoklady:** Záznam studenta se stejnými údaji neexistuje
- **Kroky:**
  1. Vytvoř v aplikaci Postman HTTP Request
  2. v Body vytvoř JSON s těmito daty:

```
1 {  
2   "firstName": "Franta",  
3   "lastName": "Kapr",  
4   "email": "fanda.kapr@engeto.cz",  
5   "age": 30  
6 }
```
  3. Zavolej metodu POST  
http://108.143.193.45:8080/api/v1/students/
  4. Zkontroluj v databázi pomocí nástroje MySQL Workbench že student s přideleným ID byl vytvořen
- **Očekávaný výsledek:** Server vrátí stavový kód 201 (Created) a zobrazí náhled vytvořeného studenta a nový záznam se objeví v databázi.

**Scénář 2:** Zadané povinné pole chybí, jméno

- **Předpoklady:** N/A
- **Kroky:**
  1. Vytvoř v aplikaci Postman HTTP Request
  2. v Body vytvoř JSON s těmito daty:

```
1 {  
2   "lastName": "Biden",  
3   "email": "david.biden@engeto.cz",  
4   "age": "30"  
5 }
```
  3. Zavolej metodu POST  
http://108.143.193.45:8080/api/v1/students/
  4. Zkontroluj v databázi pomocí nástroje MySQL Workbench že student nebyl vytvořen
- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) se zprávou "Jméno je povinný údaj" a v databázi nebude žádný nový záznam.

### Scénář 3: Zadané povinné pole chybí, příjmení

- **Předpoklady:** N/A
- **Kroky:**
  1. Vytvoř v aplikaci Postman HTTP Request
  2. v Body vytvoř JSON v těchto datech:

```
1 {  
2   "firstName": "David",  
3   "email": "david.biden@engeto.cz",  
4   "age": "30"  
5 }
```
  3. Zavolej metodu POST  
http://108.143.193.45:8080/api/v1/students/
  4. Zkontroluj v databázi pomocí nástroje MySQL Workbench že student nebyl vytvořen
- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) se zprávou "Příjmení je povinný údaj" a v databázi nebude žádný nový záznam.

### Scénář 4: Zadané povinné pole chybí, email

- **Předpoklady:** N/A
- **Kroky:**
  1. Vytvoř v aplikaci Postman HTTP Request
  2. v Body vytvoř JSON v těchto datech:

```
1 {  
2   "firstName": "David",  
3   "lastName": "Biden",  
4   "age": "30"  
5 }
```
  3. Zavolej metodu POST  
http://108.143.193.45:8080/api/v1/students/
  4. Zkontroluj v databázi pomocí nástroje MySQL Workbench že student nebyl vytvořen
- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) se zprávou "Email je povinný údaj" a v databázi nebude žádný nový záznam.

### Scénář 5: Zadané povinné pole chybí, věk

- **Předpoklady:** N/A

- **Kroky:**

1. Vytvoř v aplikaci Postman HTTP Request
2. v Body vytvoř JSON v těchto datech:

```
1 {  
2   "firstName": "David",  
3   "lastName": "Biden",  
4   "email": "david.biden@engeto.cz"  
5 }
```

3. Zavolej metodu POST  
http://108.143.193.45:8080/api/v1/students/
4. Zkontroluj v databázi pomocí nástroje MySQL Workbench že student nebyl vytvořen

- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) se zprávou "Věk je povinný údaj" a v databázi nebude žádný nový záznam.

### Scénář 6: Prázdné všechny pole

- **Předpoklady:** N/A

- **Kroky:**

1. Vytvoř v aplikaci Postman HTTP Request
2. v Body vytvoř JSON v těchto datech:

```
1 {  
2   "firstName": "",  
3   "lastName": "",  
4   "email": "",  
5   "age": ""  
6 }
```

3. Zavolej metodu POST  
http://108.143.193.45:8080/api/v1/students/
4. Zkontroluj v databázi pomocí nástroje MySQL Workbench že student nebyl vytvořen

- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) se zprávou "Vyplňte všechny údaje o studentovi" a v databázi nebude žádný nový záznam.



### Scénář 7: Zadání neplatných datových typů, číslo v poli jméno

- **Předpoklady:** N/A

- **Kroky:**

1. Vytvoř v aplikaci Postman HTTP Request
2. v Body vytvoř JSON v těchto datech:

```
1 {  
2   "firstName": "Ondřej2",  
3   "lastName": "Novák",  
4   "email": "ondrej.novak@email.cz",  
5   "age": "20"  
6 }
```

3. Zavolej metodu POST  
http://108.143.193.45:8080/api/v1/students/
4. Zkontroluj v databázi pomocí nástroje MySQL Workbench že student nebyl vytvořen

- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) se zprávou "Jméno nesmí obsahovat číslice." a v databázi nebude žádný nový záznam.

### Scénář 8: Zadání neplatných datových typů, číslo v poli příjmení

- **Předpoklady:** N/A

- **Kroky:**

1. Vytvoř v aplikaci Postman HTTP Request
2. v Body vytvoř JSON v těchto datech:

```
1 {  
2   "firstName": "Ondřej",  
3   "lastName": "Novák2",  
4   "email": "ondrej.novak@email.cz",  
5   "age": "20"  
6 }
```

3. Zavolej metodu POST  
http://108.143.193.45:8080/api/v1/students/
4. Zkontroluj v databázi pomocí nástroje MySQL Workbench že student nebyl vytvořen

- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) se zprávou "Příjmení nesmí obsahovat číslice." a v databázi nebude žádný nový záznam.

### Scénář 9: Zadání neplatných datových typů, věk jako text

- **Předpoklady:** N/A
- **Kroky:**
  1. Vytvoř v aplikaci Postman HTTP Request
  2. v Body vytvoř JSON v těchto datech:

```
1 {
2   "firstName": "Franta",
3   "lastName": "Kapr",
4   "email": "fanda.kapr@engeto.cz",
5   "age": "třicet"
6 }
```
  3. Zavolej metodu POST  
http://108.143.193.45:8080/api/v1/students/
  4. Zkontroluj v databázi pomocí nástroje MySQL Workbench že student nebyl vytvořen
- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) se zprávou "Zadejte platný věk." a v databázi nebude žádný nový záznam.

### Scénář 10: Duplicitní vytvoření studenta

- **Předpoklady:** Student se jménem "Franta Kapr", emailem "fanda.kapr@engeto.cz" a věkem 30 již existuje.
- **Kroky:**
  1. Vytvoř v aplikaci Postman HTTP Request
  2. v Body vytvoř JSON v těchto datech:

```
1 {
2   "firstName": "Franta",
3   "lastName": "Kapr",
4   "email": "fanda.kapr@engeto.cz",
5   "age": 30
6 }
```
  3. Zavolej metodu POST  
http://108.143.193.45:8080/api/v1/students/
  4. Zkontroluj v databázi pomocí nástroje MySQL Workbench že duplicitní student nebyl vytvořen
- **Očekávaný výsledek:** Server vrátí stavový kód 409 (Conflict) s informací o duplicitě. V databázi zůstane jen původní záznam.

### Scénář 11: Zadání hodnot, které přesahují limit polí, příliš dlouhé jméno a příjmení

- **Předpoklady:** N/A

- **Kroky:**

1. Vytvoř v aplikaci Postman HTTP Request
2. v Body vytvoř JSON v těchto datech:

```
1 {  
2   "firstName": "Franta ten nejlepší student na této škole",  
3   "lastName": "Kapr který vše zvládne",  
4   "email": "fanda.kapr@engeto.cz",  
5   "age": "30"  
6 }
```

3. Zavolej metodu POST  
http://108.143.193.45:8080/api/v1/students/
4. Zkontroluj v databázi pomocí nástroje MySQL Workbench že student nebyl vytvořen

- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) a zprávu o překročení limitu pro jméno nebo příjmení. V databázi se žádný nový záznam neobjeví.

### Scénář 12: Spodní limit pole věk

- **Předpoklady:** N/A

- **Kroky:**

1. Vytvoř v aplikaci Postman HTTP Request
2. v Body vytvoř JSON v těchto datech:

```
1 {  
2   "firstName": "Karlik",  
3   "lastName": "Novak",  
4   "email": "novak.karlik@egg.cz",  
5   "age": "-5"  
6 }
```

3. Zavolej metodu POST  
http://108.143.193.45:8080/api/v1/students/
4. Zkontroluj v databázi pomocí nástroje MySQL Workbench že student nebyl vytvořen

- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) a zprávu "Zadejte reálný věk." V databázi se žádný nový záznam neobjeví.

### Scénář 13: Horní limit pole věk

- **Předpoklady:** N/A

- **Kroky:**

1. Vytvoř v aplikaci Postman HTTP Request
2. v Body vytvoř JSON v těchto datech:

```
1 {  
2   "firstName": "Karel",  
3   "lastName": "Novak",  
4   "email": "novak.karel@eg.cz",  
5   "age": "222"  
6 }
```

3. Zavolej metodu POST  
http://108.143.193.45:8080/api/v1/students/
  4. Zkontroluj v databázi pomocí nástroje MySQL Workbench že student nebyl vytvořen
- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) a zprávu "Zadejte reálný věk." V databázi se žádný nový záznam neobjeví.

### Scénář 14: Validace emailu

- **Předpoklady:** N/A

- **Kroky:**

1. Vytvoř v aplikaci Postman HTTP Request
2. v Body vytvoř JSON v těchto datech:

```
1 {  
2   "firstName": "Franta",  
3   "lastName": "Kapr",  
4   "email": "franta.kapr.engeto.cz",  
5   "age": "22"  
6 }
```

3. Zavolej metodu POST  
http://108.143.193.45:8080/api/v1/students/
  4. Zkontroluj v databázi pomocí nástroje MySQL Workbench že student nebyl vytvořen
- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) a zprávu "Email není platný." V databázi se žádný nový záznam neobjeví.

### Scénář 15: Zadání speciálních znaků do polí (např. SQL injection)

- **Předpoklady:** N/A

- **Kroky:**

1. Vytvoř v aplikaci Postman HTTP Request
2. v Body vytvoř JSON v těchto datech:

```
1 {  
2   "firstName": "Franta'); DROP TABLE students;--",  
3   "lastName": "KAPR",  
4   "email": "fanda.kapr@engeto.cz",  
5   "age": "30"  
6 }
```

3. Zavolej metodu POST  
http://108.143.193.45:8080/api/v1/students/
4. Ověř v databázi pomocí MySQL Workbench, že student nebyl vytvořen a že ostatní data v databázi zůstala

- **Očekávaný výsledek:** Server vrátí odpověď 400 (Bad Request) a databáze zůstane nedotčena.

# EXEKUCE TESTŮ

Testovací scénáře jsem provedl(a), přikládám výsledky testů.

## Exekuce testovacího scénáře:

GET metoda

### Scénář 1: Úspěšné načtení existujícího studenta

- **Předpoklady:** Student s ID 1968 existuje v databázi.
- **Kroky:**

1. Zavolej metodu GET

`http://108.143.193.45:8080/api/v1/students/1968`

Výsledek volání: stavový kód 200

Data z odpovědi:

```
1 {  
2   "id": 1968,  
3   "firstName": "Franta",  
4   "lastName": "KAPR",  
5   "email": "fanda.kapr@engeto.cz",  
6   "age": 25  
7 }
```

2. Ověření v databázi, že student existuje

`SELECT * FROM student WHERE id=1968;`

Výsledek dotazu:

	id	age	email	first_name	last_name
▶	1968	25	fanda.kapr@engeto.cz	Franta	KAPR
*	NULL	NULL	NULL	NULL	NULL

- **Očekávaný výsledek:** Odpověď obsahuje informace o studentovi s ID 1968, které odpovídají datům uloženým v databázi.
- **Výsledek testu:** Test proběhl úspěšně.

## Scénář 2: Načtení neexistujícího studenta

- **Předpoklady:** Student s ID 9999 neexistuje v databázi.

- **Kroky:**

1. Zavolej metodu GET

`http://108.143.193.45:8080/api/v1/students/9999`

Výsledek volání: stavový kód 500

Data z odpovědi:

```
1 {  
2   "timestamp": "2024-10-25T16:51:22.555+00:00",  
3   "status": 500,  
4   "error": "Internal Server Error",  
5   "message": "",  
6   "path": "/api/v1/students/9999"  
7 }
```

2. Ověření v databázi, že student neexistuje

`SELECT * FROM student WHERE id=9999;`

Výsledek dotazu: Student s tímto ID v databázi neexistuje.

- **Očekávaný výsledek:** Server vrátí odpověď s chybou 404 a zprávu "Student nenalezen".
- **Výsledek testu:** Test proběhl neúspěšně.

## Exekuce testovacího scénáře:

DELETE metoda

## Scénář 1: Úspěšné smazání existujícího studenta.

- **Předpoklady:** Student s ID 1951 existuje v databázi.

- **Kroky:**

1. Zavolej metodu DELETE

`http://108.143.193.45:8080/api/v1/students/1951`

Výsledek volání: stavový kód 200

Data z odpovědi:

Server nevypsal zprávu "Student úspěšně smazán."

2. Ověření v databázi, že student byl smazán

`SELECT * FROM student WHERE id=1951;`

Výsledek dotazu:

Student byl opravdu smazán a není v databázi.

- **Očekávaný výsledek:** Server vrátí stavový kód 200, zprávu "Student úspěšně smazán." a student bude odstraněn z databáze.
- **Výsledek testu:** Test proběhl neúspěšně.

## Scénář 2: Pokus o smazání neexistujícího studenta.

- **Předpoklady:** Student s ID 9999 neexistuje v databázi.

- **Kroky:**

1. Zavolej metodu DELETE

http://108.143.193.45:8080/api/v1/students/9999

Výsledek volání: stavový kód 500

Data z odpovědi:

```
1 {  
2   "timestamp": "2024-10-25T17:54:50.988+00:00",  
3   "status": 500,  
4   "error": "Internal Server Error",  
5   "message": "",  
6   "path": "/api/v1/students/9999"  
7 }
```

2. Ověření v databázi, že student neexistuje

SELECT \* FROM student WHERE id=9999;

Výsledek dotazu:

Student v databázi s tímto ID neexistuje.

- **Očekávaný výsledek:** Server vrátí stavový kód 404 a zprávu "Student nenalezen".
- **Výsledek testu:** Test proběhl neúspěšně.



## Exekuce testovacího scénáře:

### POST metoda

#### Scénář 1: Úspěšné vytvoření studenta s validními daty

- **Předpoklady:** Záznam studenta se stejnými údaji neexistuje.
- **Kroky:**
  1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s těmito daty:

```
1 {  
2   "firstName": "Franta",  
3   "lastName": "Kapr",  
4   "email": "fanda.kapr@engeto.cz",  
5   "age": 30  
6 }
```

Výsledek volání: stavový kód 200

Data z odpovědi:

```
1 {  
2   "id": 2069,  
3   "firstName": "Franta",  
4   "lastName": "KAPR",  
5   "email": "fanda.kapr@engeto.cz",  
6   "age": 30  
7 }
```

2. Ověření v databázi, že student existuje  
`SELECT * FROM student WHERE id=2069;`  
Výsledek dotazu:

id	age	email	first_name	last_name
2069	30	fanda.kapr@engeto.cz	Franta	KAPR

- **Očekávaný výsledek:** Server vrátí stavový kód 201 (Created) a zobrazí náhled vytvořeného studenta a nový záznam se objeví v databázi.
- **Výsledek testu:** Test proběhl neúspěšně.

## Scénář 2: Zadané povinné pole chybí, jméno

- **Předpoklady:** N/A

- **Kroky:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "lastName": "Biden",  
3   "email": "david.biden@engeto.cz",  
4   "age": "30"  
5 }
```

Výsledek volání: stavový kód 500

Data z odpovědi:

```
1 {  
2   "timestamp": "2024-11-03T09:51:06.992+00:00",  
3   "status": 500,  
4   "error": "Internal Server Error",  
5   "message": "",  
6   "path": "/api/v1/students/"  
7 }
```

2. Ověření v databázi, že student existuje

`SELECT * FROM student WHERE last_name = 'Biden' AND email = 'david.biden@engeto.cz' AND age = '30';`

Výsledek dotazu:

Student neexistuje

- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) se zprávou "Jméno je povinný údaj" a v databázi nebude žádný nový záznam.
- **Výsledek testu:** Test proběhl neúspěšně.

### Scénář 3: Zadané povinné pole chybí, příjmení

- **Předpoklady:** N/A
- **Kroky:**
  1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "David",  
3   "email": "david.biden@engeto.cz",  
4   "age": "30"  
5 }
```

Výsledek volání: stavový kód 500

Data z odpovědi:

```
1 {  
2   "timestamp": "2024-11-03T09:51:06.992+00:00",  
3   "status": 500,  
4   "error": "Internal Server Error",  
5   "message": "",  
6   "path": "/api/v1/students/"  
7 }
```

2. Ověření v databázi, že student existuje

`SELECT * FROM student WHERE first_name = 'David' AND email = 'david.biden@engeto.cz' AND age = '30';`

Výsledek dotazu:

Student neexistuje

- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) se zprávou "Příjmení je povinný údaj" a v databázi nebude žádný nový záznam.
- **Výsledek testu:** Test proběhl neúspěšně.

#### Scénář 4: Zadané povinné pole chybí, email

- **Předpoklady:** N/A
- **Kroky:**
  1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "David",  
3   "lastName": "Biden",  
4   "age": "30"  
5 }
```

Výsledek volání: stavový kód 500

Data z odpovědi:

```
1 {  
2   "timestamp": "2024-11-03T09:51:06.992+00:00",  
3   "status": 500,  
4   "error": "Internal Server Error",  
5   "message": "",  
6   "path": "/api/v1/students/"  
7 }
```

2. Ověření v databázi, že student existuje

`SELECT * FROM student WHERE first_name = 'David' AND last_name = 'Biden' AND age = '30';`

Výsledek dotazu:

Student neexistuje

- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) se zprávou "Email je povinný údaj" a v databázi nebude žádný nový záznam.
- **Výsledek testu:** Test proběhl neúspěšně.

## Scénář 5: Zadané povinné pole chybí, věk

- **Předpoklady:** N/A
- **Kroky:**
  1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 = {  
2   "firstName": "David",  
3   "lastName": "Biden",  
4   "email": "david.biden@engeto.cz"  
5 }
```

Výsledek volání: stavový kód 500

Data z odpovědi:

```
1 = {  
2   "timestamp": "2024-10-31T13:27:56.340+00:00",  
3   "status": 500,  
4   "error": "Internal Server Error",  
5   "message": "",  
6   "path": "/api/v1/students/"  
7 }
```

2. Ověření v databázi, že student neexistuje

`SELECT * FROM student WHERE first_name = 'David' AND last_name = 'Biden' AND email = 'david.biden@engeto.cz';`

Výsledek dotazu:

Student neexistuje

- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) se zprávou "Věk je povinný údaj" a v databázi nebude žádný nový záznam.
- **Výsledek testu:** Test proběhl neúspěšně.

## Scénář 6: Prázdné všechny pole

- **Předpoklady:** N/A
- **Kroky:**
  1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "",  
3   "lastName": "",  
4   "email": "",  
5   "age": ""  
6 }
```

Výsledek volání: stavový kód 500

Data z odpovědi:

```
1 {  
2   "timestamp": "2024-10-31T13:27:56.340+00:00",  
3   "status": 500,  
4   "error": "Internal Server Error",  
5   "message": "",  
6   "path": "/api/v1/students/"  
7 }
```

- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) se zprávou "Vyplňte všechny údaje o studentoj" a v databázi nebude žádný nový záznam.
- **Výsledek testu:** Test proběhl neúspěšně.

## Scénář 7: Zadání neplatných datových typů, číslo v poli jméno

- **Předpoklady:** N/A
- **Kroky:**
  1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 = {  
2   "firstName": "Ondřej2",  
3   "lastName": "Novák",  
4   "email": "ondrej.novak@email.cz",  
5   "age": "20"  
6 }
```

Výsledek volání: stavový kód 200

Data z odpovědi:

```
1 = {  
2   "id": 2126,  
3   "firstName": "Ondřej2",  
4   "lastName": "NOVÁK",  
5   "email": "ondrej.novak@email.cz",  
6   "age": 20  
7 }
```

2. Ověření v databázi, že student nebyl vytvořen

`SELECT * FROM student WHERE first_name = 'Ondřej2' AND last_name = 'Novák';`

Výsledek dotazu:

id	age	email	first_name	last_name
2126	20	ondrej.novak@email.cz	Ondřej2	NOVÁK

- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) se zprávou "Jméno nesmí obsahovat číslice." a v databázi nebude žádný nový záznam.
- **Výsledek testu:** Test proběhl neúspěšně.

## Scénář 8: Zadání neplatných datových typů, číslo v poli příjmení

- **Předpoklady:** N/A
- **Kroky:**
  1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "Ondřej",  
3   "lastName": "Novák2",  
4   "email": "ondrej.novak@email.cz",  
5   "age": "20"  
6 }
```

Výsledek volání: stavový kód 200

Data z odpovědi:

```
1 {  
2   "id": 2125,  
3   "firstName": "Ondřej",  
4   "lastName": "NOVÁK2",  
5   "email": "ondrej.novak@email.cz",  
6   "age": 20  
}
```

2. Ověření v databázi, že student nebyl vytvořen

`SELECT * FROM student WHERE first_name = 'Ondřej' AND last_name = 'Novák2';`

Výsledek dotazu:

id	age	email	first_name	last_name
2125	20	ondrej.novak@email.cz	Ondřej	NOVÁK2

- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) se zprávou "Příjmení nesmí obsahovat číslice." a v databázi nebude žádný nový záznam.
- **Výsledek testu:** Test proběhl neúspěšně.



## Scénář 9: Zadání neplatných datových typů, věk jako text

- **Předpoklady:** N/A

- **Kroky:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "Franta",  
3   "lastName": "Kapr",  
4   "email": "fanda.kapr@engeto.cz",  
5   "age": "třicet"  
6 }
```

Výsledek volání: stavový kód 400

Data z odpovědi:

```
1 {  
2   "timestamp": "2024-10-31T13:39:32.709+00:00",  
3   "status": 400,  
4   "error": "Bad Request",  
5   "message": "",  
6   "path": "/api/v1/students/"  
7 }
```

2. Ověření v databázi, že student neexistuje

`SELECT * FROM student WHERE first_name = 'Franta' AND last_name = 'Kapr' AND email = 'fanda.kapr@engeto.cz' AND age = 'třicet';`

Výsledek dotazu:

Student v databázi s tímto Jménem, příjmením, emailem a věkem neexistuje.

- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) se zprávou "Zadejte platný věk." a v databázi nebude žádný nový záznam.
- **Výsledek testu:** Test proběhl neúspěšně.

## Scénář 10: Duplicitní vytvoření studenta

- **Předpoklady:** Student se jménem "Franta Kapr", emailem "fanda.kapr@engeto.cz" a věkem 30 již existuje.
- **Kroky:**
  1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "Franta",  
3   "lastName": "Kapr",  
4   "email": "fanda.kapr@engeto.cz",  
5   "age": 30  
6 }
```

Výsledek volání: stavový kód 200

Data z odpovědi:

```
1 {  
2   "id": 2072,  
3   "firstName": "Franta",  
4   "lastName": "KAPR",  
5   "email": "fanda.kapr@engeto.cz",  
6   "age": 30  
7 }
```

2. Ověření v databázi, že student nebyl vytvořen

`SELECT * FROM student WHERE first_name = 'Franta' AND last_name = 'Kapr' AND email = 'fanda.kapr@engeto.cz' AND age = '30';`

Výsledek dotazu:

id	age	email	first_name	last_name
2069	30	fanda.kapr@engeto.cz	Franta	KAPR
2072	30	fanda.kapr@engeto.cz	Franta	KAPR

- **Očekávaný výsledek:** Server vrátí stavový kód 409 (Conflict) s informací o duplicitě. V databázi zůstane jen původní záznam.
- **Výsledek testu:** **Test proběhl neúspěšně.**

**Scénář 11:** Zadání hodnot, které přesahují limit polí, příliš dlouhé jméno a příjmení

- **Předpoklady:** N/A

- **Kroky:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "Franta ten nejlepsi student na teto skole",  
3   "lastName": "Kapr ktery vse zvladne",  
4   "email": "fanda.kapr@engeto.cz",  
5   "age": "30"  
6 }
```

Výsledek volání: stavový kód 200

Data z odpovědi:

```
1 {  
2   "id": 2073,  
3   "firstName": "Franta ten nejlepsi student na teto skole",  
4   "lastName": "KAPR KTERY VSE ZVLADNE",  
5   "email": "fanda.kapr@engeto.cz",  
6   "age": 30  
7 }
```

2. Ověření v databázi, že student nebyl vytvořen

`SELECT * FROM student WHERE first_name = 'Franta ten nejlepsi student na teto skole' AND last_name = 'Kapr ktery vse zvladne';`

Výsledek dotazu:

id	age	email	first_name	last_name
2073	30	fanda.kapr@engeto.cz	Franta ten nejlepsi student na teto skole	KAPR KTERY VSE ZVLADNE

- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) a zprávu o překročení limitu pro jméno a příjmení. V databázi se žádný nový záznam neobjeví.
- **Výsledek testu:** Test proběhl neúspěšně.

## Scénář 12: Spodní limit pole věk

- **Předpoklady:** N/A

- **Kroky:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "Karlík",  
3   "lastName": "Novak",  
4   "email": "novak.karlik@gg.cz",  
5   "age": "-5"  
6 }
```

Výsledek volání: stavový kód 200

Data z odpovědi:

```
1 {  
2   "id": 2086,  
3   "firstName": "Karlík",  
4   "lastName": "NOVAK",  
5   "email": "novak.karlik@gg.cz",  
6   "age": -5  
7 }
```

2. Ověření v databázi, že student nebyl vytvořen

`SELECT * FROM student WHERE id = '2086';`

Výsledek dotazu:

id	age	email	first_name	last_name
2086	-5	novak.karlik@gg.cz	Karlík	NOVAK

- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) a zprávu "Zadejte reálný věk." V databázi se žádný nový záznam neobjeví.
- **Výsledek testu:** Test proběhl neúspěšně.

### Scénář 13: Horní limit pole věk

- **Předpoklady:** N/A

- **Kroky:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "Karel",  
3   "lastName": "Novak",  
4   "email": "novak.karel@gg.cz",  
5   "age": "222"  
6 }
```

Výsledek volání: stavový kód 200

Data z odpovědi:

```
1 {  
2   "id": 2085,  
3   "firstName": "Karel",  
4   "lastName": "NOVAK",  
5   "email": "novak.karel@gg.cz",  
6   "age": 222  
7 }
```

2. Ověření v databázi, že student nebyl vytvořen

`SELECT * FROM student WHERE id = '2085';`

Výsledek dotazu:

id	age	email	first_name	last_name
2085	222	novak.karel@gg.cz	Karel	NOVAK

- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) a zprávu "Zadejte reálný věk." V databázi se žádný nový záznam neobjeví.
- **Výsledek testu:** Test proběhl neúspěšně.

## Scénář 14: Validace emailu

- **Předpoklady:** N/A

- **Kroky:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "Franta",  
3   "lastName": "Kapr",  
4   "email": "franta.kapr.engeto.cz",  
5   "age": "22"  
6 }
```

Výsledek volání: stavový kód 200

Data z odpovědi:

```
1 {  
2   "id": 2084,  
3   "firstName": "Franta",  
4   "lastName": "KAPR",  
5   "email": "franta.kapr.engeto.cz",  
6   "age": 22  
7 }
```

2. Ověření v databázi, že student nebyl vytvořen

`SELECT * FROM student WHERE id = '2084';`

Výsledek dotazu:

id	age	email	first_name	last_name
2084	22	franta.kapr.engeto.cz	Franta	KAPR

- **Očekávaný výsledek:** Server vrátí stavový kód 400 (Bad Request) a zprávu "Email není platný." V databázi se žádný nový záznam neobjeví.
- **Výsledek testu:** Test proběhl neúspěšně.

### Scénář 15: Zadání speciálních znaků do polí (např. SQL injection)

- **Předpoklady:** N/A

- **Kroky:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "Franta'); DROP TABLE students;--",  
3   "lastName": "KAPR",  
4   "email": "fanda.kapr@engeto.cz",  
5   "age": "30"  
6 }
```

Výsledek volání: stavový kód 200

Data z odpovědi:

```
1 {  
2   "id": 2074,  
3   "firstName": "Franta'); DROP TABLE students;--",  
4   "lastName": "KAPR",  
5   "email": "fanda.kapr@engeto.cz",  
6   "age": 30  
7 }
```

2. Ověř v databázi, že student nebyl vytvořen a že ostatní data v databázi zůstala

`SELECT * FROM student WHERE id = 2074;`

Výsledek dotazu:

id	age	email	first_name	last_name
2074	30	fanda.kapr@engeto.cz	Franta'); DROP TABLE students;--	KAPR

`SELECT COUNT(*) FROM student;`

Výsledek dotazu:

count(*)
1010

- **Očekávaný výsledek:** Server vrátí odpověď 400 (Bad Request) a databáze zůstane nedotčena.
- **Výsledek testu:** Test proběhl neúspěšně.

# BUG REPORT

Na základě provedených scénářů jsem objevil(a) uvedené chyby aplikace.

**Bug ID:** 01

**Název bugu:** GET metoda nevrací očekávanou chybu 404 pro neexistujícího studenta

**Priorita:** Vysoká

**Stav:** Otevřená

**Datum:** 25.10.2024

**Autor:** David Oubrecht

**Popis bugu:**

Metoda GET pro zobrazení dat o studentovi vrací neočekávaný výsledek, pokud student s daným ID neexistuje. Místo očekávané odpovědi 404 a zprávy "Student nenalezen" vrací metoda status kód 500 a nevypíše žádnou zprávu.

**Kroky k reprodukcí chyby:**

1. Zavolej metodu GET `http://108.143.193.45:8080/api/v1/students/9999` kde je ID neexistujícího studenta.
2. Sleduj výsledek odpovědi.

**Očekávaný výsledek:**

Server vrátí odpověď s chybou 404 a zprávu "Student nenalezen".

**Skutečný výsledek:**

```
1 {  
2   "timestamp": "2024-10-25T16:51:22.555+00:00",  
3   "status": 500,  
4   "error": "Internal Server Error",  
5   "message": "",  
6   "path": "/api/v1/students/9999"  
7 }
```



**Bug ID:** 02

**Název bugu:** DELETE metoda nevypíše zprávu

**Priorita:** Střední

**Stav:** Otevřená

**Datum:** 25.10.2024

**Autor:** David Oubrecht

**Popis bugu:**

Metoda DELETE úspěšně vrátí status kód 200, smaže studenta, ale nevypíše zprávu že byl student smazán.

**Kroky k reprodukcí chyby:**

1. Zavolej metodu DELETE  
`http://108.143.193.45:8080/api/v1/students/1951`
2. Sleduj výsledek odpovědi.

**Očekávaný výsledek:**

Server vrátí stavový kód 200, zprávu "Student úspěšně smazán." a student bude odstraněn z databáze.

**Skutečný výsledek:**

Server vrátí stavový kód 200, smaže studenta z databáze, ale nevypíše zprávu

**Bug ID:** 03

**Název bugu:** DELETE metoda při zadání neexistujícího ID nevypíše chybovou zprávu

**Priorita:** Střední

**Stav:** Otevřená

**Datum:** 25.10.2024

**Autor:** David Oubrecht

**Popis bugu:**

Metoda DELETE úspěšně vrátí status kód 200, smaže studenta, ale nevypíše zprávu že byl student smazán.

**Kroky k reprodukcí chyby:**

1. Zavolej metodu DELETE `http://108.143.193.45:8080/api/v1/students/9999`
2. Sleduj výsledek odpovědi.

**Očekávaný výsledek:**

Server vrátí stavový kód 404 a zprávu "Student nenalezen".

**Skutečný výsledek:**

Server vrátí stavový kód 500 a nevypíše chybovou zprávu

```
1 {  
2   "timestamp": "2024-10-25T17:54:50.988+00:00",  
3   "status": 500,  
4   "error": "Internal Server Error",  
5   "message": "",  
6   "path": "/api/v1/students/9999"  
7 }
```

**Bug ID:** 04

**Název bugu:** POST metoda nevrací očekávaný kód 201

**Priorita:** Nízká

**Stav:** Otevřená

**Datum:** 31.10.2024

**Autor:** David Oubrecht

**Popis bugu:**

Metoda POST vytvoří nového studenta v databázi, pouze se vypisuje jiný statový kód.

**Kroky k reprodukcí chyby:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "Franta",  
3   "lastName": "Kapr",  
4   "email": "fanda.kapr@engeto.cz",  
5   "age": 30  
6 }
```

2. Sleduj výsledek odpovědi.

**Očekávaný výsledek:**

Server vrátí stavový kód 201 (Created) a zobrazí náhled vytvořeného studenta a nový záznam se objeví v databázi.

**Skutečný výsledek:**

Server nevrátí kód 201, ale 200, jinak vše proběhne v pořádku.

**Bug ID:** 05

**Název bugu:** POST metoda s prázdným polem jmeno, nevrací očekávaný kód

**Priorita:** Střední

**Stav:** Otevřená

**Datum:** 31.10.2024

**Autor:** David Oubrecht

**Popis bugu:**

Metoda POST po nevyplnění pole a odeslání se nechová jak má.

**Kroky k reprodukcí chyby:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 = {  
2   "lastName": "Biden",  
3   "email": "david.biden@engeto.cz",  
4   "age": "30"  
5 }
```

2. Sleduj výsledek odpovědi.

**Očekávaný výsledek:**

Server vrátí stavový kód 400 (Bad Request) se zprávou "Jméno je povinný údaj" a v databázi nebude žádný nový záznam.

**Skutečný výsledek:**

Server nevrátí kód 400, ale 500 a nevypíše chybovou hlášku.

```
1 = {  
2   "timestamp": "2024-11-03T09:51:06.992+00:00",  
3   "status": 500,  
4   "error": "Internal Server Error",  
5   "message": "",  
6   "path": "/api/v1/students/"  
7 }
```

**Bug ID:** 06

**Název bugu:** POST metoda s prázdným polem prijmeni, nevrací očekávaný kód

**Priorita:** Střední

**Stav:** Otevřená

**Datum:** 31.10.2024

**Autor:** David Oubrecht

**Popis bugu:**

Metoda POST po nevyplnění pole a odeslání se nechová jak má.

**Kroky k reprodukcí chyby:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "David",  
3   "email": "david.biden@engeto.cz",  
4   "age": "30"  
5 }
```

2. Sleduj výsledek odpovědi.

**Očekávaný výsledek:**

Server vrátí stavový kód 400 (Bad Request) se zprávou "Příjmení je povinný údaj" a v databázi nebude žádný nový záznam.

**Skutečný výsledek:**

Server nevrátí kód 400, ale 500 a nevypíše chybovou hlášku.

```
1 {  
2   "timestamp": "2024-11-03T09:51:06.992+00:00",  
3   "status": 500,  
4   "error": "Internal Server Error",  
5   "message": "",  
6   "path": "/api/v1/students/"  
7 }
```

**Bug ID:** 07

**Název bugu:** POST metoda s prázdným polem email, nevrací očekávaný kód

**Priorita:** Střední

**Stav:** Otevřená

**Datum:** 31.10.2024

**Autor:** David Oubrecht

**Popis bugu:**

Metoda POST po nevyplnění pole a odeslání se nechová jak má.

**Kroky k reprodukcí chyby:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 = {  
2   "firstName": "David",  
3   "lastName": "Biden",  
4   "age": "30"  
5 }
```

2. Sleduj výsledek odpovědi.

**Očekávaný výsledek:**

Server vrátí stavový kód 400 (Bad Request) se zprávou "Email je povinný údaj" a v databázi nebude žádný nový záznam.

**Skutečný výsledek:**

Server nevrátí kód 400, ale 500 a nevypíše chybovou hlášku.

```
1 = {  
2   "timestamp": "2024-11-03T09:51:06.992+00:00",  
3   "status": 500,  
4   "error": "Internal Server Error",  
5   "message": "",  
6   "path": "/api/v1/students/"  
7 }
```

**Bug ID:** 08

**Název bugu:** POST metoda s prázdným polem vek, nevrací očekávaný kód

**Priorita:** Střední

**Stav:** Otevřená

**Datum:** 31.10.2024

**Autor:** David Oubrecht

**Popis bugu:**

Metoda POST po nevyplnění pole a odeslání se nechová jak má.

**Kroky k reprodukcí chyby:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 = {  
2   "firstName": "David",  
3   "lastName": "Biden",  
4   "email": "david.biden@engeto.cz"  
5 }
```

2. Sleduj výsledek odpovědi.

**Očekávaný výsledek:**

Server vrátí stavový kód 400 (Bad Request) se zprávou "Věk je povinný údaj" a v databázi nebude žádný nový záznam.

**Skutečný výsledek:**

Server nevrátí kód 400, ale 500 a nevypíše chybovou hlášku.

```
1 = {  
2   "timestamp": "2024-10-31T13:27:56.340+00:00",  
3   "status": 500,  
4   "error": "Internal Server Error",  
5   "message": "",  
6   "path": "/api/v1/students/"  
7 }
```

**Bug ID:** 09

**Název bugu:** POST metoda s vyplněnými poli nevrací očekávaný kód

**Priorita:** Střední

**Stav:** Otevřená

**Datum:** 1.11.2024

**Autor:** David Oubrecht

**Popis bugu:**

Metoda POST po nevyplnění polí a odeslání se nechová podle scénáře

**Kroky k reprodukci chyby:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "",  
3   "lastName": "",  
4   "email": "",  
5   "age": ""  
6 }
```

2. Sleduj výsledek odpovědi.

**Očekávaný výsledek:**

Server vrátí stavový kód 400 (Bad Request) se zprávou "Vyplňte všechny údaje o studentoj" a v databázi nebude žádný nový záznam.

**Skutečný výsledek:**

Server nevrátí kód 400, ale 500 a nevypíše chybovou hlášku.

```
1 {  
2   "timestamp": "2024-10-31T13:27:56.340+00:00",  
3   "status": 500,  
4   "error": "Internal Server Error",  
5   "message": "",  
6   "path": "/api/v1/students/"  
7 }
```



**Bug ID:** 10

**Název bugu:** POST metoda a pole jmeno prijma cisla

**Priorita:** Střední

**Stav:** Otevřená

**Datum:** 1.11.2024

**Autor:** David Oubrecht

**Popis bugu:**

Do pole jmeno zle zapsat cislo a odeslat a uložit do databáze

**Kroky k reprodukcí chyby:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "Ondřej2",  
3   "lastName": "Novák",  
4   "email": "ondrej.novak@email.cz",  
5   "age": "20"  
6 }
```

2. Sleduj výsledek odpovědi.

**Očekávaný výsledek:**

Server vrátí stavový kód 400 (Bad Request) se zprávou "Zadejte platný věk." a v databázi nebude žádný nový záznam.

**Skutečný výsledek:**

Server nevrátí kód 400, ale 200 a studenta zapíše do databáze.

```
1 {  
2   "id": 2126,  
3   "firstName": "Ondřej2",  
4   "lastName": "NOVÁK",  
5   "email": "ondrej.novak@email.cz",  
6   "age": 20  
7 }
```

id	age	email	first_name	last_name
2126	20	ondrej.novak@email.cz	Ondřej2	NOVÁK

**Bug ID:** 11

**Název bugu:** POST metoda a pole prijmeni prijma cisla

**Priorita:** Střední

**Stav:** Otevřená

**Datum:** 1.11.2024

**Autor:** David Oubrecht

**Popis bugu:**

Do pole prijmeni zle zapsat cislo a odeslat a uložit do databáze

**Kroky k reprodukcí chyby:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "Ondřej",  
3   "lastName": "Novák2",  
4   "email": "ondrej.novak@email.cz",  
5   "age": "20"  
6 }
```

2. Sleduj výsledek odpovědi.

**Očekávaný výsledek:**

Server vrátí stavový kód 400 (Bad Request) se zprávou "Příjmení nesmí obsahovat číslice." a v databázi nebude žádný nový záznam.

**Skutečný výsledek:**

Server nevrátí kód 400, ale 200 a studenta zapíše do databáze.

```
1 {  
2   "id": 2125,  
3   "firstName": "Ondřej",  
4   "lastName": "NOVÁK2",  
5   "email": "ondrej.novak@email.cz",  
6   "age": 20
```

id	age	email	first_name	last_name
2125	20	ondrej.novak@email.cz	Ondřej	NOVÁK2

**Bug ID:** 12

**Název bugu:** POST metoda pri zadani spatneho datového typu nevypíše chybovou zprávu

**Priorita:** Střední

**Stav:** Otevřená

**Datum:** 1.11.2024

**Autor:** David Oubrecht

**Popis bugu:**

Po zadání neplatného datového typu do pole server vrátí chybovou hlášku podle předpokladu.

**Kroky k reprodukcii chyby:**

3. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "Franta",  
3   "lastName": "Kapr",  
4   "email": "fanda.kapr@engeto.cz",  
5   "age": "třicet"  
6 }
```

4. Sleduj výsledek odpovědi.

**Očekávaný výsledek:**

Server vrátí stavový kód 400 (Bad Request) se zprávou "Zadejte platný věk." a v databázi nebude žádný nový záznam.

**Skutečný výsledek:**

Server nevrátí kód 400, ale 500 a nevypíše text.

```
1 {  
2   "timestamp": "2024-10-31T13:39:32.709+00:00",  
3   "status": 400,  
4   "error": "Bad Request",  
5   "message": "",  
6   "path": "/api/v1/students/"  
7 }
```

**Bug ID:** 13

**Název bugu:** POST metoda může vytvořit duplicitní záznam do databáze

**Priorita:** Vysoká

**Stav:** Otevřená

**Datum:** 1.11.2024

**Autor:** David Oubrecht

**Popis bugu:**

Lze do databáze vytvořit identického studenta vícekrát.

**Kroky k reprodukci chyby:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "Franta",  
3   "lastName": "Kapr",  
4   "email": "fanda.kapr@engeto.cz",  
5   "age": 30  
6 }
```

2. Zavolej tuto metodu znova.
3. Sleduj výsledek odpovědi.

**Očekávaný výsledek:**

Server vrátí stavový kód 409 (Conflict) s informací o duplicitě. V databázi zůstane jen původní záznam.

**Skutečný výsledek:**

Server vytvoří identické studenty.

id	age	email	first_name	last_name
2069	30	fanda.kapr@engeto.cz	Franta	KAPR
2072	30	fanda.kapr@engeto.cz	Franta	KAPR

**Bug ID:** 14

**Název bugu:** pole jméno a příjmení není délkově omezeno

**Priorita:** Střední

**Stav:** Otevřená

**Datum:** 1.11.2024

**Autor:** David Oubrecht

**Popis bugu:**

Pole nejsou délkově omezená a jdou tam zadávat dlouhé texty

**Kroky k reprodukcí chyby:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 = {  
2   "firstName": "Franta ten nejlepsi student na teto skole",  
3   "lastName": "Kapr ktery vse zvladne",  
4   "email": "fanda.kapr@engeto.cz",  
5   "age": "30"  
6 }
```

2. Sleduj výsledek odpovědi.

**Očekávaný výsledek:**

Server vrátí stavový kód 400 (Bad Request) a zprávu o překročení limitu pro jméno a příjmení. V databázi se žádný nový záznam neobjeví.

**Skutečný výsledek:**

Server vytvoří záznam v databázi s tímto dlouhým jménem a příjmením

```
1 = {  
2   "id": 2073,  
3   "firstName": "Franta ten nejlepsi student na teto skole",  
4   "lastName": "KAPR KTERY VSE ZVLADNE",  
5   "email": "fanda.kapr@engeto.cz",  
6   "age": 30  
7 }
```

id	age	email	first_name	last_name
2073	30	fanda.kapr@engeto.cz	Franta ten nejlepsi student na teto skole	KAPR KTERY VSE ZVLADNE

**Bug ID:** 15

**Název bugu:** pole vek nema spodní limit

**Priorita:** Střední

**Stav:** Otevřená

**Datum:** 4.11.2024

**Autor:** David Oubrecht

**Popis bugu:**

Do pole vek lze zapsat záporné číslo

**Kroky k reprodukcí chyby:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 = {  
2   "firstName": "Karlík",  
3   "lastName": "Novak",  
4   "email": "novak.karlik@gg.cz",  
5   "age": "-5"  
6 }
```

2. Sleduj výsledek odpovědi.

**Očekávaný výsledek:**

Server vrátí stavový kód 400 (Bad Request) a zprávu "Zadejte reálný věk." V databázi se žádný nový záznam neobjeví.

**Skutečný výsledek:**

Server vytvoří záznam v databázi s tímto záporným číslem.

```
1 = {  
2   "id": 2086,  
3   "firstName": "Karlík",  
4   "lastName": "NOVAK",  
5   "email": "novak.karlik@gg.cz",  
6   "age": -5  
7 }
```

id	age	email	first_name	last_name
2086	-5	novak.karlik@gg.cz	Karlík	NOVAK

**Bug ID:** 16

**Název bugu:** pole vek nema horní limit

**Priorita:** Střední

**Stav:** Otevřená

**Datum:** 4.11.2024

**Autor:** David Oubrecht

**Popis bugu:**

Do pole vek lze zapsat nereálné velké číslo

**Kroky k reprodukcí chyby:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "Karel",  
3   "lastName": "Novak",  
4   "email": "novak.karel@gg.cz",  
5   "age": "222"  
6 }
```

2. Sleduj výsledek odpovědi.

**Očekávaný výsledek:**

Server vrátí stavový kód 400 (Bad Request) a zprávu "Zadejte reálný věk." V databázi se žádný nový záznam neobjeví.

**Skutečný výsledek:**

Server vytvoří záznam v databázi s tímto nereálným věkem

```
1 {  
2   "id": 2085,  
3   "firstName": "Karel",  
4   "lastName": "NOVAK",  
5   "email": "novak.karel@gg.cz",  
6   "age": 222  
7 }
```

id	age	email	first_name	last_name
2085	222	novak.karel@gg.cz	Karel	NOVAK

**Bug ID:** 17

**Název bugu:** pole email přijima neplatnou emailovou adresu

**Priorita:** Vysoká

**Stav:** Otevřená

**Datum:** 4.11.2024

**Autor:** David Oubrecht

**Popis bugu:**

Do pole email lze zapsat neplatnou adresu a odeslat požadavek

**Kroky k reprodukci chyby:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "Franta",  
3   "lastName": "Kapr",  
4   "email": "franta.kapr.engeto.cz",  
5   "age": "22"  
6 }
```

2. Sleduj výsledek odpovědi.

**Očekávaný výsledek:**

Server vrátí stavový kód 400 (Bad Request) a zprávu "Email není platný." V databázi se žádný nový záznam neobjeví.

**Skutečný výsledek:**

Server vytvoří záznam v databázi s tímto neplatným emailem

```
1 {  
2   "id": 2084,  
3   "firstName": "Franta",  
4   "lastName": "KAPR",  
5   "email": "franta.kapr.engeto.cz",  
6   "age": 22  
7 }
```

id	age	email	first_name	last_name
2084	22	franta.kapr.engeto.cz	Franta	KAPR



**Bug ID:** 18

**Název bugu:** pole přijímá speciální znaky a zapíše je do databáze

**Priorita:** Vysoká

**Stav:** Otevřená

**Datum:** 4.11.2024

**Autor:** David Oubrecht

**Popis bugu:**

Do polí jdou zapsat speciální znaky které se dají zneužít k SQL injection

**Kroky k reprodukcí chyby:**

1. Zavolej metodu POST `http://108.143.193.45:8080/api/v1/students/` s temito daty:

```
1 {  
2   "firstName": "Franta'); DROP TABLE students;--",  
3   "lastName": "KAPR",  
4   "email": "fanda.kapr@engeto.cz",  
5   "age": "30"  
6 }
```

2. Sleduj výsledek odpovědi.

**Očekávaný výsledek:**

Server vrátí odpověď 400 (Bad Request) a databáze zůstane nedotčena.

**Skutečný výsledek:**

Server vše dovolil zapsat do databáze která by mohla být napadena kdyby tam byl napsaný název databáze správně

```
1 {  
2   "id": 2074,  
3   "firstName": "Franta'); DROP TABLE students;--",  
4   "lastName": "KAPR",  
5   "email": "fanda.kapr@engeto.cz",  
6   "age": 30  
7 }
```

id	age	email	first_name	last_name
2074	30	fanda.kapr@engeto.cz	Franta'); DROP TABLE students;--	KAPR