

# FINAL PROJECT

## No.1



Author: Markéta Hořáková  
Date: 27.10.2024

ASSIGNMENT	1
TEST SCENARIOS	2
GET	2
Positive scenario	2
Search existing student	2
Negative scenario	3
Search non-existing student	3
POST	4
Positive scenario	4
Boundary testing	4
Minimum first name and last name	4
Medium firstName and lastName	5
Maximum firstName and lastName	6
Minimum Age	7
Medium Age	8
Maximum Age	9
Valid email format	10
Negative scenario	11
Below minimum firstName and last name	11
Above maximum firstName and last name	12
Bellow minimum Age	13
Above maximum Age	14
Email without "@" symbol	15
Email without domain	16
Email without 2LD	17
Duplicate record	18
Empty attributes	19
Lowercase conversion	21
Special characters in names	22

Additional information	23
Invalid JSON format	24
DELETE	25
Delete existing student	25
Delete non existing student	26
Delete existing student with invalid ID format - string	27
Delete existing student with invalid ID format - special character	28
Delete a record without providing an ID	29
EXECUTION OF TESTS	30
Summary Table of Executions	30
Test case ID: 20024001	33
Test case ID: 20024002	34
Test case ID: 20024003	35
Test case ID: 20024004	36
Test case ID: 20024005	37
Test case ID: 20024006	38
Test case ID: 20024007	39
Test case ID: 20024008	40
Test case ID: 20024009	41
Test case ID: 20024010	42
Test case ID: 20024011	43
Test case ID: 20024012	44
Test case ID: 20024013	45
Test case ID: 20024014	46
Test case ID: 20024015	47
Test case ID: 20024016	48
Test case ID: 20024017	49
Test case ID: 20024018	51
Test case ID: 20024019	55

Test case ID: 20024020	56
Test case ID: 20024021	58
Test case ID: 20024022	59
Test case ID: 20024023	60
Test case ID: 20024024	61
Test case ID: 20024025	62
Test case ID: 20024026	63
Test case ID: 20024027	64
<b>BUG REPORT</b>	<b>65</b>
Bug ID: B0012024	66
Bug ID: B0022024	67
Bug ID: B0032024	68
Bug ID: B0042024	69
Bug ID: B0052024	70
Bug ID: B0062024	71

# ASSIGNMENT

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..

- Všechny atributy jsou povinné (name, lastname, age, email)
- Údaje se ukládají v LowerCase()
- first\_name, last\_name obsahují hodnoty v rozmezí 3-50 znaků
- Věk je v rozsahu 1-150 let
- Formát e-mailu: \*@\*.\*

## Přístupové údaje:

Databáze	database: qa_demo Host: aws.connect.psdb.cloud Username: sgqcu2878p6q2zzw5tw4 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.

# TEST SCENARIOS

*Based on the mentioned test scenarios, I verified the functionality of the application.*

## GET

### Positive scenario

#### Search existing student

**ID: 20024001**

#### Abstract :

Verification that the API correctly returns the details of a specific student based on their ID.

#### Data preparation:

Ensure there is existing record with the following attributes in database:

SELECT \*FROM student WHERE id=357;

<i>id</i>	<i>age</i>	<i>email</i>	<i>first_name</i>	<i>last_name</i>
357	18	john.doe@gmail.com	John	Doe

#### Steps:

1. Open app Postman.
2. Set up method GET.
3. Insert URL: *http://108.143.193.45:8080/api/v1/students/357*
4. Push button SEND.
5. Check expecting results.

#### Expecting results:

HTTP Status code 200 (OK)

Responds:

```
{  
    "id": 357,  
    "firstName": "John",  
    "lastName": "Doe",  
    "email": "john.doe@gmail.com",  
    "age": 18  
}
```

## **Negative scenario**

### ***Search non-existing student***

**ID : 20024002**

#### **Abstract:**

Verification that the API correctly returns the error when attempting to get a student record that **does not exist** in the database.

#### **Data preparation**

Ensure there is no existing record with the following attributes in database:

SELECT \*FROM student WHERE id=9999;

*Note: If exist it's necessary to delete.*

#### **Steps:**

1. Open app Postman.
2. Set up method GET.
3. Insert URL: *http://108.143.193.45:8080/api/v1/students/9999*
4. Push button SEND.
5. Check expecting results.

#### **Expecting results:**

HTTP Status code 404 (Not Found)

The response body may contain a message about the non existing student.

# **POST**

## **Positive scenario**

### **Boundary testing**

***Minimum first name and last name***

**ID: 20024003**

#### **Abstract:**

Verification of boundary values for the attributes **firstName** and **lastName** (minimum length - 3 characters)

#### **Data preparation:**

Ensure there is no existing record with the following attributes in database:

SELECT \* FROM student WHERE

```
first_name= "Jan" AND  
last_name="Kim" AND  
email=" " AND  
age= 35 ;
```

If exist is necessary to delete.

#### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
  "firstName": "Jan",  
  "lastName": "Kim",  
  "email": "jan.kim@gmail.com",  
  "age": 35  
}
```

7. Click on the SEND button
8. Check expecting results.

#### **Expecting results:**

HTTP Status code 201 (Created)

Format: JSON

Responds body contains:

```
{  
  "id": <unique_id>,  
  "firstName": "Jan",  
  "lastName": "Kim",  
  "email": "jan.kim@gmail.com",  
  "age": 35  
}
```

## **Medium *firstName* and *lastName***

### **ID: 20024004**

**Abstract:** Verification of boundary values for the attributes **firstName** and **lastName** (medium length - 25 characters)

#### **Data preparation:**

Ensure there is no existing record with the following attributes in database:

```
SELECT * FROM student WHERE
```

```
    first_name= "Bohumilovanárodnostnílová" AND  
    last_name= "Skočdopolemiloslavoválová" AND  
    email = "bohunka.hop@gmail.com" AND  
    age= 35      ;
```

If exist is necessary to delete.

#### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
  "firstName": "Bohumilovanárodnostnílová",  
  "lastName": "Skočdopolemiloslavoválová",  
  "email": "bohunka.hop@gmail.com",  
  "age": 35  
}
```

7. Click on the SEND button
8. Check expecting results.

#### **Expecting results:**

HTTP Status code 201

Format: JSON

Responds body contains:

```
{  
  "id": <unique_id>,  
  "firstName": "Bohumilovanárodnostnílová",  
  "lastName": "Skočdopolemiloslavoválová",  
  "email": "bohunka.hop@gmail.com",  
  "age": 35  
}
```

## **Maximum *firstName* and *lastName***

**ID:20024005**

**Abstract:** Verification of boundary values for the attributes *firstName* and *lastName*  
**(maximum length - 150 characters)**

### **Data preparation:**

Ensure there is no existing record with the following attributes in database:

SELECT \* FROM student WHERE

```
first_name= "Jaroslavoslavoslavoslavoslavoslavoslavoslavoslavoslavos" AND  
last_name="Skočdopolemiloslavoslavoslavoslavoslavoslavoslavoslavoslavol" AND  
email = "jarda.hop@gmail.com" AND  
age= 35 ;
```

If exist is necessary to delete.

### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
  "firstName": "Jaroslavoslavoslavoslavoslavoslavoslavoslavoslavoslavos",  
  "lastName": "Skočdopolemiloslavoslavoslavoslavoslavoslavoslavoslavoslavol",  
  "email": "jarda.hop@gmail.com",  
  "age": 35  
}
```

7. Click on the SEND button
8. Check expecting results.

### **Expecting results:**

HTTP Status code 201

Format: JSON

Responds body contains:

```
{  
  "id": <unique_id>,  
  "firstName": "Jaroslavoslavoslavoslavoslavoslavoslavoslavoslavoslavos",  
  "lastName": "Skočdopolemiloslavoslavoslavoslavoslavoslavoslavoslavol",  
  "email": "jarda.hop@gmail.com",  
  "age": 35  
}
```

## **Minimum Age**

**ID:20024006**

**Abstract:** Verification of boundary values for the attribute **age** (minimum = 1 year)

### **Data preparation:**

Ensure there is no existing record with the following attributes in database:

SELECT \* FROM student WHERE

```
first_name= "Jan" AND  
last_name="Kim" AND  
email = "jan.kim@gmail.com" AND  
age= 1 ;
```

If exist it's necessary to delete.

### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
  "firstName": "Jan",  
  "lastName": "Kim",  
  "email": "jan.kim@gmail.com",  
  "age": 1  
}
```

7. Click on the SEND button
8. Check expecting results.

### **Expecting results:**

HTTP Status code 201

Format: JSON

Responds body contains:

```
{  
  "id": <unique_id>,  
  "firstName": "Jan",  
  "lastName": "Kim",  
  "email": "jan.kim@gmail.com",  
  "age": 1  
}
```

## **Medium Age**

**ID:20024007**

**Abstract:** Verification of boundary values for the attribute **age** (medium = 75 year)

### **Data preparation:**

Ensure there is no existing record with the following attributes in database:

SELECT \* FROM student WHERE

```
first_name= "Jan" AND  
last_name="Kim" AND  
email = "jan.kim@gmail.com" AND  
age= 75 ;
```

If exist it's necessary to delete.

### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
  "firstName": "Jan",  
  "lastName": "Kim",  
  "email": "jan.kim@gmail.com",  
  "age": 75  
}
```

7. Click on the SEND button
8. Check expecting results.

### **Expecting results:**

HTTP Status code 201

Format: JSON

Responds body contains:

```
{  
  "id": <unique_id>,  
  "firstName": "Jan",  
  "lastName": "Kim",  
  "email": "jan.kim@gmail.com",  
  "age": 75  
}
```

## **Maximum Age**

**ID:20024008**

**Abstract:** Verification of boundary values for the attribute **age** (maximum = 150 year)

### **Data preparation:**

Ensure there is no existing record with the following attributes in database:

SELECT \* FROM student WHERE

```
first_name= "Jan" AND  
last_name="Kim" AND  
email = "jan.kim@gmail.com" AND  
age= 150 ;
```

If exist it's necessary to delete.

### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
  "firstName": "Jan",  
  "lastName": "Kim",  
  "email": "jan.kim@gmail.com",  
  "age": 150  
}
```

7. Click on the SEND button
8. Check expecting results.

### **Expecting results:**

HTTP Status code 201

Format: JSON

Responds body contains:

```
{  
  "id": <unique_id>,  
  "firstName": "Jan",  
  "lastName": "Kim",  
  "email": "jan.kim@gmail.com",  
  "age": 150  
}
```

# Valid email format

ID:20024009

**Abstract:** Verification of valid attribute **email**.

## Data preparation:

Ensure there is no existing record with the following attributes in database:

SELECT \* FROM student WHERE

```
first_name= "Jan" AND  
last_name="Novak" AND  
email = "jan.novak@gmail.com" AND  
age= 54 ;
```

If exist it's necessary to delete.

## Steps:

1. Open app Postman.
2. Set up method POST.
3. Insert URL: *http://108.143.193.45:8080/api/v1/students/*
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
  "firstName": "Jan",  
  "lastName": "Novak",  
  "email": "jan.novak@gmail.com",  
  "age": 54  
}
```

7. Click on the SEND button
8. Check expecting results.

## Expecting results:

HTTP Status code 201

Format: JSON

Responds body contains:

```
{  
  "id": <unique_id>,  
  "firstName": "Jan",  
  "lastName": "Novak",  
  "email": "jan.novak@gmail.com",  
  "age": 54  
}
```

# Negative scenario

***Below minimum firstName and last name***

**ID:20024010**

**Abstract:** Verification of boundary values for the attributes **firstName** and **lastName** (below minimum length - 2 characters)

## Data preparation:

Ensure there is no existing record with the following attributes in database:

```
SELECT * FROM student WHERE
    first_name= "Oz" AND
    last_name="Li" AND
    email = "jan.kim@gmail.com" AND
    age= 35      ;
```

If exist it's necessary to delete.

## Steps:

1. Open app Postman.
2. Set up method POST.
3. Insert URL: *http://108.143.193.45:8080/api/v1/students/*
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{
  "firstName": "Oz",
  "lastName": "Li",
  "email": "jan.kim@gmail.com",
  "age": 35
}
```

7. Click on the SEND button
8. Check expecting results.

## Expecting results:

HTTP Status code 400

The response body may contain a message about the incorrect data.

**Above maximum *firstName* and *lastName***

**ID:20024011**

**Abstract:** Verification of boundary values for the attributes **firstName** and **lastName** (above maximum length - 56 characters)

**Data preparation:**

Ensure there is no existing record with the following attributes in database:

SELECT \* FROM student WHERE

```
first_name= "Bohumilomiroslavoslavoslavoslavoslavoslavoslavoslavoslavoslavoslav"  
last_name="Skočdopolemiloslavoslavoslavoslavoslavoslavoslavoslavoslavoslavoslavoslav  
email = "bohunka.hop@gmail.com" AND  
age= 35 ;
```

If exist it's necessary to delete.

**Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
  "firstName": "Bohumilomiroslavoslavoslavoslavoslavoslavoslavoslavoslavoslavoslav",  
  "lastName": "Skočdopolemiloslavoslavoslavoslavoslavoslavoslavoslavoslavoslavoslavoslav",  
  "email": "bohunka.hop@gmail.com",  
  "age": 35  
}
```

7. Click on the SEND button
8. Check expecting results.

**Expecting results:**

HTTP Status code 400

The response body may contain a message about the incorrect data.

## **Bellow minimum Age**

**ID:20024012**

**Abstract:** Verification of boundary values for the attribute **age** (bellow minimum = -60 year)

### **Data preparation:**

Ensure there is no existing record with the following attributes in database:

SELECT \* FROM student WHERE

```
first_name= "Jan" AND  
last_name="Kim" AND  
email = "jan.kim@gmail.com" AND  
age= -60;
```

If exist it's necessary to delete.

### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
  "firstName": "Jan",  
  "lastName": "Kim",  
  "email": "jan.kim@gmail.com",  
  "age": -60  
}
```

7. Click on the SEND button
8. Check expecting results.

### **Expecting results:**

HTTP Status code 400

The response body may contain a message about the incorrect data.

## **Above maximum Age**

**ID:20024013**

**Abstract:** Verification of boundary values for the attribute **age** (above maximum = 200 year)

### **Data preparation:**

Ensure there is no existing record with the following attributes in database:

SELECT \* FROM student WHERE

```
first_name= "Jan" AND  
last_name="Kim" AND  
email = "jan.kim@gmail.com" AND  
age= 200;
```

If exist it's necessary to delete.

### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
  "firstName": "Jan",  
  "lastName": "Kim",  
  "email": "jan.kim@gmail.com",  
  "age": 200  
}
```

7. Click on the SEND button
8. Check expecting results.

### **Expecting results:**

HTTP Status code 400

The response body may contain a message about the incorrect data.

## **Email without "@" symbol**

**ID:20024014**

**Abstract:** Verification of valid format of attribute **email** (without "@" symbol)

### **Data preparation:**

Ensure there is no existing record with the following attributes in database:

SELECT \* FROM student WHERE

```
first_name= "Jan" AND  
last_name="Novak" AND  
email = "jan.novak@gmail.com" AND  
age=54;
```

If exist it's necessary to delete.

### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
  "firstName": "Jan",  
  "lastName": "Novak",  
  "email": "jan.novak@gmail.com",  
  "age": 54  
}
```

7. Click on the SEND button
8. Check expecting results.

### **Expecting results:**

HTTP Status code 400

The response body may contain a message about the incorrect data.

## **Email without domain**

**ID:20024015**

**Abstract:** Verification of valid format of attribute **email** (without domain)

### **Data preparation:**

Ensure there is no existing record with the following attributes in database:

SELECT \* FROM student WHERE

```
first_name= "Jan" AND  
last_name="Novak" AND  
email = "jan.novak@.com" AND  
age= 54;
```

If exist it's necessary to delete.

### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
  "firstName": "Jan",  
  "lastName": "Novak",  
  "email": "jan.novak@.com",  
  "age": 54  
}
```

7. Click on the SEND button
8. Check expecting results.

### **Expecting results:**

HTTP Status code 400

The response body may contain a message about the incorrect data.

## **Email without 2LD**

**ID:20024016**

**Abstract:** Verification of valid format of attribute **email** (without 2LD)

### **Data preparation:**

Ensure there is no existing record with the following attributes in database:

SELECT \* FROM student WHERE

```
first_name= "Jan" AND  
last_name="Novak" AND  
email : "jan.novak@gmail." AND  
age: 54;
```

If exist it's necessary to delete.

### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
  "firstName": "Jan",  
  "lastName": "Novak",  
  "email": "jan.novak@gmail.",  
  "age": 54  
}
```

7. Click on the SEND button
8. Check expecting results.

### **Expecting results:**

HTTP Status code 400

The response body may contain a message about the incorrect data.

## **Duplicate record**

**ID:20024017**

**Abstract:** Verification that the API returns an error when attempting to create a duplicate record

### **Data preparation:**

Ensure there is no existing record with the following attributes in database:

SELECT \* FROM student WHERE

```
first_name= "Albert" AND  
last_name="Einstein" AND  
email = "albi.Ein@Emc2.com" AND  
age=76;
```

*Note: If exist skip the steps 8-9. Or you can delete the existing record and follow the whole scenario steps.*

### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
    "firstName": "Albert",  
    "lastName": "Einstein",  
    "email": "",  
    "age": 76  
}
```

7. Click on the SEND button
8. Check expecting HTTP status is 201 (Created) and response body contains the newly created record details.
9. Send a second POST request with the same data. Repeat the steps 3-7 and then continue with following step.
10. Check expecting HTTP status code is 409 ( Conflict)

### **Expecting results:**

#### First inserting:

HTTP status code 201

```
{  
    "id": <unique_id>,  
    "firstName": "Albert",  
    "lastName": "Einstein",  
    "email": "",  
    "age": 76  
}
```

#### Second (duplicated) inserting:

HTTP Status code 409

The response body may contain a message about the already existing data.

## **Empty attributes**

**ID:20024018**

**Abstract:** Verification that the API returns an error when attempting to create a record with empty attribute.

### **Data preparation:**

Ensure there is no existing record with the following attributes in database:

SELECT \* FROM student WHERE

```
first_name= "Leonardo" AND  
last_name="da Vinci" AND  
email = "leo.da.vinci@monalisa.com" AND  
age= 67;
```

If exist it's necessary to delete.

### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. For each test case, insert the following data into the "Body" tab and replace the text in the square brackets by appropriate attributes from the below table for each given test case: At least one attribute will always be missing.

```
{  
    "firstName": "[Name]",  
    "lastName": "[Last Name]",  
    "age": [Age],  
    "email": "[E-mail]"  
}
```

7. Click on the SEND button

Test ID	Name	Last Name	Age	E-mail	Description	Expected HTTP Status	Expected result
2002401 8/1	Leonardo	Da Vinci	76		Missing e-mail	400	The response body may contain a message about the presence of missing attribute.
2002401 8/2	Leonardo	Da Vinci		leo.da.vinci@monalisa.com	Missing age	400	The response body may contain a message about the presence of missing attribute.
2002401 8/3	Leonardo		76	leo.da.vinci@monalisa.com	Missing last name	400	The response body may contain a message about the presence of missing attribute.

2002401 8/4		Da Vinci	76	leo.da.vinci@monalisa.com	Missing first name	400	The response body may contain a message about the presence of missing attribut.
2002401 8/5	Leonardo		76		Missing last name and e-mail	400	The response body may contain a message about the presence of missing attribut.
2002401 8/6		Da Vinci		leo.da.vinci@monalisa.com	Missing first name and age	400	The response body may contain a message about the presence of missing attribut.
2002401 8/7		Da Vinci			missing first name, age, e-mail	400	The response body may contain a message about the presence of missing attribut.
2002401 8/8					Missing all fields	400	The response body may contain a message about the presence of missing attribut.

8. Compare the result according to the assigned ID with the record in the database and with the expected result and HTTP status.

## **Lowercase conversion**

**ID: 20024019**

**Abstract:** Verification that the API correctly stores data in lowercase format.

### **Data preparation:**

Ensure there is no existing record with the following attributes in database:

SELECT \* FROM student WHERE

```
first_name= "Elvis" AND  
last_name="Presley" AND  
email = "Elvis.Presley@rnrking.com" AND  
age= 42;
```

If exist it's necessary to delete.

### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
  "firstName": "ElViS",  
  "lastName": "PrEsLeY",  
  "age": 42,  
  "email": "elvis.Presley@RnRKing.com"  
}
```

7. Click on the SEND button

### **Expecting results:**

HTTP Status code 201

Format: JSON

Responds body contains:

```
{  
  "firstName": "Elvis",  
  "lastName": "Presley",  
  "age": 42,  
  "email": "elvis.presley@rnrking.com"  
}
```

## ***Special characters in names***

**ID: 20024020**

**Abstract:** Verification that the API does not allow creating a record with special characters in **first\_name** or **last\_name**.

### **Data preparation:**

Ensure there is no existing record with the following attributes in database:

SELECT \* FROM student WHERE

```
first_name= "Julius" AND  
last_name="Caesar" AND  
email = "ave.caesar@venividivici.com" AND  
age= 57;
```

If exist it's necessary to delete.

### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. For each test case, insert the following data into the "Body" tab and replace the text in the square brackets by appropriate attributes from the bellow table for each given test case:

```
{  
    "firstName": "[Name]",  
    "lastName": "[Last Name]",  
    "age": [Age],  
    "email": ",,[E-mail]"}
```

7. Click on the SEND button
8. Compare the result according to the assigned ID with the record in the database and with the expected result in table.

Test ID	Name	Last Name	Age	E-mail	Description	Expected HTTP Status	Expected result
200240 20/1	J@lius	Caesar	57	ave.caesar@ve nividivici.com	Special character in firstName	400	The response body may contain a message about the presence of invalid or unexpected characters.
200240 20/2	Julius	C@esar	57	ave.caesar@ve nividivici.com	Special character in lastName	400	The response body may contain a message about the presence of invalid or unexpected characters.
200240 20/3	J@lius	C@esar	57	ave.caesar@ve nividivici.com	Special character in firstName and lastName	400	The response body may contain a message about the presence of invalid or unexpected characters.

## ***Additional information***

**ID:20024021**

**Abstract:** Verification that the API allows you to create a record with additional information

### **Data preparation:**

Ensure there is no existing record with the following attributes in database:

SELECT \* FROM student WHERE

```
first_name= "William" AND  
last_name="Shakespeare" AND  
email = "will.shakespeare@hamlet.net" AND  
age= 52;
```

If exist it's necessary to delete.

### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
  "firstName": "William",  
  "lastName": "Shakespeare",  
  "email": "will.Shakespeare@hamlet.net",  
  "age": 52,  
  "book" : "Rome and Juliet"  
}
```

7. Click on the SEND button
8. Check expecting results.

### **Expecting results:**

HTTP Status code 400

The response body may contain a message about the unexpected data.

## ***Invalid JSON format***

**ID:20024022**

**Abstract:** Verification that the API returns an error if a request is sent with an invalid JSON format

### **Data preparation:**

Ensure there is no existing record with the following attributes in database:

SELECT \* FROM student WHERE

```
first_name= "Lukáš" AND  
last_name="Dostal" AND  
email ="Lukas.dostal@mistr.net" AND  
age= 24;
```

If exist it's necessary to delete.

### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
    first_name: Lukáš,  
    last_name: Dostál,  
    age: 24,  
    email: lukas.dostal@mistr.net“  
}
```

7. Click on the SEND button

8. Check expecting results.

### **Expecting results:**

HTTP Status code 400

The response body may contain a message about the invalid format.

# **DELETE**

**Delete existing student**

**ID:20024023**

**Abstract:** Verification that the API successfully delete data of an existing student.

## **Data preparation:**

Ensure there is existing record with the following attributes in database:

**SELECT \*FROM student WHERE**

```
first_name= "William" AND  
last_name="Shakespeare" AND  
email : "Will.Shakespeare@hamlet.net" AND  
age: 52;
```

*Note: Record the ID of the student from the response. This ID will be use in the next phase of the test scenario for deleting the student.*

If such a record does not exist, create it using a POST request:

1. Set up method POST.
2. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
3. Go to the "Body" tab.
4. Select "raw" and "JSON" as the input type.
5. Insert the following data into the body:

```
{  
  "firstName": "William",  
  "lastName": "Shakespeare",  
  "email": "will.Shakespeare@hamlet.net",  
  "age": 52  
}
```

7. Click on the SEND button

*Note: Record the ID of the created student from the response. This ID will be use in the next phase of the test scenario for deleting the student.*

## **Steps for delete existing student record:**

1. Open app Postman.
2. Set up method DELETE.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/{studentID}>  
\*(Replace {studentID} with the ID recorded from the data preparation step).
7. Click on the SEND button
8. Check expecting results.

## **Expecting results:**

HTTP Status code 200

The response body may contain a message about successfully student record delete.

**Delete non existing student**

**ID:20024024**

**Abstract:** Verification that the API return error message when attempting to delete a student record that does not exist in the database.

**Data preparation:**

Ensure there is no existing record with the following attributes in database:

SELECT \*FROM student WHERE id=9999;

If exist, necessary to delete.

**Steps for delete non existing student record:**

1. Open app Postman.
2. Set up method DELETE.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/9999>
4. Click on the SEND button
5. Check expecting results.

**Expecting results:**

HTTP status code 404

The response body may contain a message about the non existing student.

***Delete existing student with invalid ID format - string***

**ID:20024025**

**Abstract:** Verification that the API prevents deletion with invalid student ID format - **string instead of integer.**

**Data preparation:**

No specific data preparation is required for this scenario.

**Steps:**

1. Open app Postman.
2. Set up method DELETE.
3. Insert URL: *http://108.143.193.45:8080/api/v1/students/abcd*
4. Click on the SEND button
5. Check expecting results.

**Expecting results:**

HTTP Status code 400

The response body may contain a message about the invalid data.

**Delete existing student with invalid ID format - special character  
ID:20024026**

**Abstract:** Verification that the API prevents deletion with invalid student ID format - **special character**

**Data preparation:**

No specific data preparation is required for this scenario.

**Steps:**

1. Open app Postman.
2. Set up method DELETE.
3. Insert URL: *http://108.143.193.45:8080/api/v1/students/inv@lidID*
4. Click on the SEND button
5. Check expecting results.

**Expecting results:**

HTTP Status code 400

The response body may contain a message about the invalid data.

***Delete a record without providing an ID***

**ID:20024027**

**Abstract:** Verification that the API prevents deletion with invalid student ID format - missing ID

**Data preparation:**

No specific data preparation is required for this scenario.

**Steps:**

1. Open app Postman.
2. Set up method DELETE.
3. Insert URL: <http://108.143.193.45:8080/api/v1/students/>
4. Click on the SEND button
5. Check expecting results.

**Expecting results:**

HTTP Status code 400

The response body may contain a message about the invalid data.

# EXECUTION OF TESTS

*I carried out the test scenarios, I am attaching the test results.*

## ***Summary Table of Executions***

**Executed by:** Markéta Hořáková

**Date:** 22.9.2024

**Test Environment:** Postman 11.6.1, API Server: <http://108.143.193.45:8080/api/v1/students/>

**Request Method:** GET

Test Case ID	Test Title	Execution Date	Executed by	Result	Comments
2024001	Verify successful retrieval of existing student.	2024-09-22	Markéta Hořáková	Passed	No issues encountered.
2024002	Search non-existing student	2024-09-22	Markéta Hořáková	Failed	Received 500 error instead of 404.
2024003	Boundary testing first and last name	2024-09-22	Markéta Hořáková	Failed	Received 200 error instead of 201.
2024004	Boundary testing first and last name	2024-09-22	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0012024;B0 032024
2024005	Boundary testing first and last name	2024-10-10	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0012024;B0 032024
2024006	Boundary testing age	2024-10-10	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0012024;B0 032024
2024007	Boundary testing age	2024-10-10	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0012024;B0 032024

2024008	Boundary testing age	2024-10-10	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0012024;B0 032024
2024009	Verification of valid attribute email.	2024-10-10	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0012024;B0 032024
20024010	Negative boundary testing first and last name	2024-10-10	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0042024
20024011	Negative boundary testing first and last name	2024-10-10	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0042024
20024012	Negative Boundary testing age	2024-10-10	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0042024
20024013	Negative Boundary testing age	2024-10-10	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0042024
20024014	Negative testing of attribute e-mail	2024-10-10	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0042024
20024015	Negative testing of attribute e-mail	2024-10-10	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0042024
20024016	Negative testing of attribute e-mail	2024-10-10	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0042024
20024017	Duplicated records	2024-10-27	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0042024
20024018	Empty attributes	2024-10-27	Markéta Hořáková	Failed	
20024019	Lowercase conversion	2024-10-27	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0032024 and B0012024)
20024020	Special characters in names	2024-10-27	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0042024 and B0012024

20024021	Additional information	2024-10-27	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0042024 and B0012024
20024022	Invalid JSON format	2024-10-27	Markéta Hořáková	Passed	
20024023	Delete existing student	2024-10-27	Markéta Hořáková	Passed	
20024024	Delete non existing student	2024-10-27	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0022024
20024025	Delete existing student with invalid ID format - string	2024-10-27	Markéta Hořáková	Passed	
20024026	Delete existing student with invalid ID format - special character	2024-10-27	Markéta Hořáková	Passed	
20024027	Delete a record without providing an ID	2024-10-27	Markéta Hořáková	Passed	

## **Test case ID: 20024001**

### **Data preparation:**

The screenshot shows a database interface titled "SQL File 3\*". At the top, there are various icons for file operations like Open, Save, Print, and a magnifying glass for search. A toolbar includes "Limit to 1000 rows" and other standard database tools. Below the toolbar, a query window contains the SQL command: "SELECT \*FROM student WHERE id=357;". The results are displayed in a "Result Grid" table. The columns are labeled "id", "age", "email", "first\_name", and "last\_name". There is one row of data: id is 357, age is 18, email is john.doe@gmail.com, first\_name is John, and last\_name is DOE. All other fields in the grid are marked as "NULL".

### **Expecting results:**

HTTP Status code: 200 (OK)

Responds:

```
{  
  "id": 357,  
  "firstName": "John",  
  "lastName": "DOE",  
  "email": "john.doe@gmail.com",  
  "age": 18  
}
```

### **Actual Results:**

HTTP Status Code: 200 (OK )

Response Body:

The screenshot shows the Postman application interface. At the top, it says "http://108.143.193.45:8080/api/v1/students/357". Below this, there are tabs for "GET", "Params", "Authorization", "Headers (8)", "Body", "Pre-request Script", "Tests", and "Settings". The "Body" tab is selected. Under "Body", there is a "Query Params" section with a table. The table has two rows: one for "Key" and one for "Value". Both rows have a single entry: "Key". To the right of the table is a "Bulk Edit" button. Above the table, there are "Save" and "Send" buttons.

The screenshot shows the Postman interface after sending the request. At the top, it says "Status: 200 OK Time: 188 ms Size: 248 B". Below this, there are tabs for "Body", "Cookies", "Headers (5)", and "Test Results". The "Body" tab is selected. The response body is displayed in "Pretty" format as a JSON object:

```
1 {  
2   "id": 357,  
3   "firstName": "John",  
4   "lastName": "DOE",  
5   "email": "john.doe@gmail.com",  
6   "age": 18  
7 }
```

**Test results: FAILED**

## **Test case ID: 20024002**

### **Data preparation:**

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a query editor window containing the SQL command: `SELECT *FROM student WHERE id=9999;`. The results grid below shows one row with all columns set to NULL. At the bottom, the timeline shows a single action: a SELECT query at 19:23:58 that returned 0 rows in 0.038 seconds.

id	age	email	first_name	last_name
NULL	NULL	NULL	NULL	NULL

Action Output

Time	Action	Response	Duration / Fetch Time
19:23:58	SELECT *FROM student WHERE id=...	0 row(s) returned	0.038 sec / 0.0012 s

### **Expecting results:**

HTTP Status code 404 (Not Found)

The response body may contain a message about the non existing student.

### **Actual Results:**

The screenshot shows a Postman request configuration. The method is GET, and the URL is `http://108.143.193.45:8080/api/v1/students/9999`. The Headers tab shows 8 items. The Body tab is selected, showing a JSON object with a single key 'Key' and a value 'Value'.

The screenshot shows the Postman response view. The status is 500 Internal Server Error, with a timestamp of 2024-09-22T17:25:32.607+00:00, status 500, error Internal Server Error, message "", and path /api/v1/students/9999. The response body is displayed in JSON format.

```
1 "timestamp": "2024-09-22T17:25:32.607+00:00",
2 "status": 500,
3 "error": "Internal Server Error",
4 "message": "",
5 "path": "/api/v1/students/9999"
```

### **Test results: FAILED**

## Test case ID: 20024003

### Data preparation:

The screenshot shows the MySQL Workbench interface. At the top, a query is written in SQL:

```
1  SELECT * FROM student WHERE
2      first_name= "Jan" AND
3      last_name="Kim" AND
4      email= "jan.kim@gmail.com" AND
5      age= 35 ;
6
```

The results grid shows one row of data:

id	age	email	first_name	last_name
NULL	NULL	NULL	NULL	NULL

Below the results, the status bar shows "student 5". Under "Action Output", there is one entry:

Time	Action	Response	Duration / Fetch Time
18:38:59	SELECT *FROM student WHERE id=35...	0 row(s) returned	0.025 sec / 0.000024...

### Expecting results:

HTTP Status code 201

Format: JSON

Responds body contains:

```
{ "id": <unique_id>,
  "firstName": "Jan",
  "lastName": "Kim",
  "email": "jan.kim@gmail.com",
  "age": 35 }
```

### Actual Results:

The screenshot shows the Postman application interface. The URL is set to `http://108.143.193.45:8080/api/v1/students/`. The method is set to `POST`. The "Body" tab is selected, showing the following JSON payload:

```
1 {
2     "firstName": "Jan",
3     "lastName": "Kim",
4     "email": "jan.kim@gmail.com",
5     "age": 35
6 }
```

The response section shows a successful `200 OK` status with a size of `247 B`. The response body is displayed in "Pretty" format:

```
1 {
2     "id": 1847,
3     "firstName": "Jan",
4     "lastName": "KIM",
5     "email": "jan.kim@gmail.com",
6     "age": 35
7 }
```

### Test results: FAILED

## Test case ID: 20024004

### Data preparation:

```
1 •  SELECT * FROM student WHERE
2      first_name= "Bohumilovanárodnostnílová" AND
3      last_name="Skočdopolemiloslavoválová" AND
4      email= "bohunka.hop@gmail.com" AND
5      age= 35 ;
```

The screenshot shows the MySQL Workbench interface. At the top, a code editor displays the SQL query. Below it is the 'Result Grid' pane, which is currently empty. A status bar at the bottom indicates the query was executed at 18:38:59 and returned 0 rows. Action output shows one entry with a duration of 0.025 sec.

### Expecting results:

HTTP Status code 201

Format: JSON

Responds body contains:

```
{ "id": <unique_id>,
  "firstName": "Bohumilovanárodnostnílová",
  "lastName": "Skočdopolemiloslavoválová",
  "email": "bohunka.hop@gmail.com",
  "age": 35 }
```

### Actual Results:

The screenshot shows a Postman collection. A POST request is made to `http://108.143.193.45:8080/api/v1/students/`. The 'Body' tab is selected, showing a JSON payload with fields: id, firstName, lastName, email, and age. The response status is 200 OK, with a response body identical to the expected result.

### Test results: FAILED see notes

**Notes:** Test failed due to the known bug ( report No, B0012024 and B0032024) . Need to retest once the bug is solved.

## Test case ID: 20024005

### Data preparation:

The screenshot shows a MySQL Workbench interface. At the top, a query window displays the following SQL code:

```
1 •  SELECT * FROM student WHERE
2      first_name= "Jaroslavoslavoslavoslavoslavoslavoslavoslavoslavoslavos" AND
3      last_name="Skočdopolemiloslavoslavoslavoslavoslavoslavoslavoslavol" AND
4      email = "jarda.hop@gmail.com" AND
5      age= 35 ;
6
```

The results grid below shows one row of data:

id	age	email	first_name	last_name
NULL	NULL	NULL	NULL	NULL

At the bottom, the action output shows the execution details:

Action	Time	Response	Duration / Fetch Time
17 10:32:42 SELECT * FROM student WHERE first_name...		0 row(s) returned	0.054 sec / 0.000027...

### Expecting results:

HTTP Status code 201

Format: JSON

Responds body contains:

```
{  
  "id": <unique_id>,  
  "firstName": "Jaroslavoslavoslavoslavoslavoslavoslavoslavoslavoslavos",  
  "lastName": "Skočdopolemiloslavoslavoslavoslavoslavoslavoslavoslavol",  
  "email": "jarda.hop@gmail.com",  
  "age": 35  
}
```

### Actual Results:

The screenshot shows a Postman request configuration. The method is POST, and the URL is `http://108.143.193.45:8080/api/v1/students/`. The Body tab is selected, and the content type is set to JSON. The raw JSON body is identical to the expected results:

```
{  
  "id": 1855,  
  "firstName": "Jaroslavoslavoslavoslavoslavoslavoslavoslavoslavoslavos",  
  "lastName": "SKOČDOPOLEMILOSLAVOSLAVOSLAVOSLAVOSLAVOSLAVOSLAVOL",  
  "email": "jarda.hop@gmail.com",  
  "age": 35  
}
```

The screenshot shows the Postman response details. The status is 200 OK, and the response body is identical to the raw JSON sent in the request:

```
{  
  "id": 1855,  
  "firstName": "Jaroslavoslavoslavoslavoslavoslavoslavoslavoslavoslavos",  
  "lastName": "SKOČDOPOLEMILOSLAVOSLAVOSLAVOSLAVOSLAVOSLAVOSLAVOL",  
  "email": "jarda.hop@gmail.com",  
  "age": 35  
}
```

### Test results: FAILED see notes

**Notes:** Test failed due to the known bug ( report No, B0012024 and B0032024). Need to retest once the bug is solved.

**Test case ID: 20024006**

## Data preparation:

## Expecting results:

HTTP Status code 201

## Format: JSON

Responds body contains:

{

```
  "id": <unique_id>,  
  "firstName": "Jan",  
  "lastName": "Kim",  
  "email": "jan.kim@gmail.com",  
  "age": 1  
}
```

### Actual Results:

The screenshot shows the Postman interface with the following details:

- Method: POST
- URL: http://108.143.193.45:8080/api/v1/students/
- Body tab selected, showing JSON input:

```
1 {"firstName": "Jan",
2 "lastName": "Kim",
3 "email": "jan.kim@gmail.com",
4 "age": 1}
```
- Response status: 200 OK
- Response time: 182 ms
- Response size: 246 B
- Body tab selected, showing JSON response:

```
1 {"id": 1856,
2 "firstName": "Jan",
3 "lastName": "KIM",
4 "email": "jan.kim@gmail.com",
5 "age": 1}
```

## **Test results: FAILED see notes**

**Notes:** Test failed due to the known bug ( report No, B0012024 and B0032024). Need to retest once the bug is solved.

## Test case ID: 20024007

### Data preparation:

The screenshot shows the MySQL Workbench interface. At the top, a query is displayed:

```
1 •  SELECT * FROM student WHERE
2      first_name= "Jan" AND
3      last_name="Kim" AND
4      email = "jan.kim@gmail.com" AND
5      age= 75 ;
6
```

The result grid below shows no rows found:

id	age	email	first_name	last_name
NULL	NULL	NULL	NULL	NULL

The status bar at the bottom indicates "student 18" and "Duration / Fetch Time: 0.051 sec / 0.000013...".

### Expecting results:

HTTP Status code 201

Format: JSON

Responds body contains:

```
{  
  "id": <unique_id>,  
  "firstName": "Jan",  
  "lastName": "Kim",  
  "email": "jan.kim@gmail.com",  
  "age": 75  
}
```

### Actual Results:

The screenshot shows a POST request in Postman to the URL <http://108.143.193.45:8080/api/v1/students/>. The request body is:

```
1 {  
2   ... "firstName": "Jan",  
3   ... "lastName": "Kim",  
4   ... "email": "jan.kim@gmail.com",  
5   ... "age": 75  
6 }
```

The response status is 200 OK, with a duration of 278 ms and a size of 247 B. The response body is:

```
1 {  
2   "id": 1857,  
3   "firstName": "Jan",  
4   "lastName": "KIM",  
5   "email": "jan.kim@gmail.com",  
6   "age": 75  
7 }
```

### Test results: FAILED see notes

**Notes:** Test failed due to the known bug ( report No, B0012024 and B0032024). Need to retest once the bug is solved.

## Test case ID: 20024008

### Data preparation:

```
1 •  SELECT * FROM student WHERE
2      first_name= "Jan" AND
3      last_name="Kim" AND
4      email = "jan.kim@gmail.com" AND
5      age= 150 ;
```

Result Grid Filter Rows:  Search Edit: Export/Import:

id	age	email	first_name	last_name
HULL	HULL	HULL	HULL	HULL

student 19 Action Output Time Action Response Duration / Fetch Time

22 10:38:30	SELECT * FROM student WHERE first_nam...	0 row(s) returned	0.061 sec / 0.000027...
-------------	--	-------------------	-------------------------

### Expecting results:

HTTP Status code 201

Format: JSON

Responds body contains:

```
{  
  "id": <unique_id>,  
  "firstName": "Jan",  
  "lastName": "Kim",  
  "email": "jan.kim@gmail.com",  
  "age": 150  
}
```

### Actual Results:

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

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary JSON

```
1 ...
2   ...
3   ...
4   ...
5   ...
6   ...
7
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 352 ms Size: 248 B

Pretty Raw Preview Visualize JSON

```
1 ...
2   ...
3   ...
4   ...
5   ...
6   ...
7
```

### Test results: FAILED see notes

Notes: Test failed due to the known bug ( report No, B0012024 and B0032024). Need to retest once the bug is solved.

## Test case ID: 20024009

### Data preparation:

```
1 •  SELECT * FROM student WHERE
2      first_name= "Jan" AND
3      last_name="Novak" AND
4      email = "jan.novak@gmail.com" AND
5      age= 54 ;
```

The screenshot shows a database interface. At the top, there's a code editor with the above SQL query. Below it is a 'Result Grid' table with columns: id, age, email, first\_name, and last\_name. All rows in the grid are currently empty (NULL). At the bottom, there's an 'Action Output' table showing one successful action (24) from 10:46:54, which is a SELECT query returning 0 rows.

Action	Time	Response	Duration / Fetch Time
24 10:46:54 SELECT * FROM student WHERE first_name= "Jan" AND last_name="Novak" AND email = "jan.novak@gmail.com" AND age= 54 ;		0 row(s) returned	0.054 sec / 0.000023...

### Expecting results:

HTTP Status code 201

Format: JSON

Responds body contains:

```
{  
  "id": <unique_id>,  
  "firstName": "Jan",  
  "lastName": "Novak",  
  "email": "jan.novak@gmail.com",  
  "age": 54  
}
```

### Actual Results:

The screenshot shows a Postman test run. The request URL is http://108.143.193.45:8080/api/v1/students/. The 'Body' tab is selected, showing a JSON payload with fields: id, firstName, lastName, email, and age. The response status is 200 OK, with a response time of 264 ms and a size of 251 B. The response body is displayed in the 'Pretty' tab, showing the same JSON structure with an additional id field (1859).

Body	Cookies	Headers (5)	Test Results
<pre>1 { 2   "id": 1859, 3   "firstName": "Jan", 4   "lastName": "NOVAK", 5   "email": "jan.novak@gmail.com", 6   "age": 54 7 }</pre>			Status: 200 OK Time: 264 ms Size: 251 B

### Test results: FAILED see notes

**Notes:** Test failed due to the known bug ( report No, B0012024 and B0032024). Need to retest once the bug is solved.

**Test case ID: 20024010**

### Data preparation:

```
1 •  SELECT * FROM student WHERE  
2         first_name= "Oz" AND  
3         last_name="Li" AND  
4         email = "jan.kim@gmail.com" AND  
5         age= 35 ;
```

The screenshot shows the MySQL Workbench interface with the following details:

- Top Bar:** Shows "100%" and "6:5".
- Result Grid:** Titled "Result Grid" with columns: id, age, email, first\_name, and last\_name. The data row has all values set to "NULL".
- Filter Rows:** Includes a search bar and icons for filter and search.
- Edit:** Includes icons for edit, insert, delete, and refresh.
- Export/Import:** Includes icons for export and import.
- Action Bar:** Contains the text "student 23" and an "Apply" button.
- Action Output:** A table with columns: Time, Action, Response, and Duration / Feto. It shows a single row: "27 10:57:51 SELECT \* FROM student WHERE first\_name... 0 row(s) returned 0.048 sec / 0.0".

## Expecting results:

## HTTP Status code 400

The response body may contain a message about the incorrect data.

### **Actual Results:**

The screenshot shows the Postman application interface. At the top, the URL `http://108.143.193.45:8080/api/v1/students/` is entered. Below the URL, the "Body" tab is selected, showing a raw JSON payload:

```
1 {  
2   "id": 1860,  
3   "firstName": "Oz",  
4   "lastName": "Li",  
5   "email": "jan.kim@gmail.com",  
6   "age": 35  
7 }
```

Below the body, the status bar indicates: Status: 200 OK, Time: 263 ms, Size: 245 B.

## Test results: FAILD

**Test case ID: 20024011**

## Data preparation:

0% 6:5

result Grid Filter Rows: Search Edit: Export/Import:

id age email first_name				last_name
HULL	HULL	HULL	HULL	HULL

student 25 Apply

tion Output

Time	Action	Response	Duration /

## Expecting results:

## HTTP Status code 400

The response body may contain a message about the incorrect data.

### **Actual Results:**

## **Test results: FAILED see notes**

**Notes:** Test failed due to the known bug ( report No, B0042024 and B0012024). Need to retest once the bug is solved.

## **Test case ID: 20024012**

### **Data preparation:**

```
1 •  SELECT * FROM student WHERE
2      first_name= "Jan" AND
3      last_name="Kim" AND
4      email = "jan.kim@gmail.com" AND
5      age=-60;
```

### **Expecting results:**

HTTP Status code 400

The response body may contain a message about the incorrect data.

### **Actual Results:**

```
1
2   ...
3   ...
4   ...
5   ...
6   ...
7
```

```
1
2   ...
3   ...
4   ...
5   ...
6   ...
7
```

Body	Cookies	Headers (5)	Test Results
<pre>1 2   ... 3   ... 4   ... 5   ... 6   ... 7</pre>			Status: 200 OK Time: 347 ms Size: 248 B

Pretty Raw Preview Visualize JSON

### **Test results: FAILED see notes**

**Notes:** Test failed due to the known bug ( report No, B0042024 and B0012024). Need to retest once the bug is solved.

## **Test case ID: 20024013**

### **Data preparation:**

```

1 •  SELECT * FROM student WHERE
2      first_name= "Jan" AND
3      last_name="Kim" AND
4      email = "jan.kim@gmail.com" AND
5      age= 200;
6

```

### **Expecting results:**

HTTP Status code 400

The response body may contain a message about the incorrect data.

### **Actual Results:**

Body	Cookies	Headers (5)	Test Results
<pre> 1 2   "firstName": "Jan", 3   "lastName": "Kim", 4   "email": "jan.kim@gmail.com", 5   "age": 200 6 7 </pre>			Status: 200 OK   Time: 192 ms   Size: 248 B   S
<pre> 1 2   "id": 1864, 3   "firstName": "Jan", 4   "lastName": "KIM", 5   "email": "jan.kim@gmail.com", 6   "age": 200 7 </pre>			

### **Test results: FAILED see notes**

**Notes:** Test failed due to the known bug ( report No, B0042024 and B0012024). Need to retest once the bug is solved.

**Test case ID: 20024014**

## Data preparation:

```
1 • SELECT * FROM student WHERE
2     first_name= "Jan" AND
3     last_name="Novak" AND
4     email = "jan.novak@gmail.com" AND
5     age= 54;
6
```

100% 6:5

Result Grid Filter Rows: Search Edit: Export/Import:

id	age	email	first_name	last_name
NULL	NULL	NULL	NULL	NULL

student 30

Action Output

Time	Action	Response	Duration / F
35 11:09:24	SELECT * FROM student WHERE first_nam...	0 row(s) returned	0.053 sec / C

## Expecting result:

HTTP Status code 400

The response body may contain a message about the incorrect data.

### **Actual results:**

The screenshot shows the Postman application interface. At the top, there's a header bar with 'POST' and the URL 'http://108.143.193.45:8080/api/v1/students/'. Below this is a navigation bar with tabs: 'Params', 'Authorization', 'Headers (8)', 'Body' (which is green), 'Pre-request Script', 'Tests', and 'Settings'. Under the 'Body' tab, there are four options: 'none', 'form-data', 'x-www-form-urlencoded', and 'raw'. The 'raw' option is selected and has a dropdown menu showing 'JSON'. The main area contains a code editor with the following JSON payload:

```
1 {  
2   "id": 1865,  
3   "firstName": "Jan",  
4   "lastName": "Novak",  
5   "email": "jan.novak@gmail.com",  
6   "age": 54  
7 }
```

Below the code editor, there are tabs for 'Body', 'Cookies', 'Headers (5)', and 'Test Results'. The 'Body' tab is active. To the right, there's a status bar showing 'Status: 200 OK', 'Time: 905 ms', and 'Size: 250 B'. At the bottom, there are buttons for 'Pretty', 'Raw', 'Preview', 'Visualize', and 'JSON' (with a dropdown menu). The 'Pretty' button is currently selected.

**Test results:** FAILED see notes

**Notes:** Test failed due to the known bug ( report No, B0042024 and B0012024). Need to retest once the bug is solved.

## Test case ID: 20024015

### Data preparation:

The screenshot shows the MySQL Workbench interface. In the SQL editor, a query is being run:

```
1 •  SELECT * FROM student WHERE
2      first_name= "Jan" AND
3      last_name="Novak" AND
4      email = "jan.novak@.com" AND
5      age = 54;
```

The result grid shows one row of data:

id	age	email	first_name	last_name
HULL	HULL	HULL	HULL	HULL

The status bar at the bottom indicates the query took 0.050 sec / 0.00.

### Expecting result:

HTTP Status code 400

The response body may contain a message about the incorrect data.

### Actual results:

The screenshot shows a POST request in Postman. The URL is `http://108.143.193.45:8080/api/v1/students/`. The request body is JSON:

```
1 {"firstName": "Jan",
2  "lastName": "Novak",
3  "email": "jan.novak@.com",
4  "age": 54}
```

The response status is 200 OK, with a size of 246 B.

### Test results: FAILED see notes

**Notes:** Test failed due to the known bug ( report No, B0042024 and B0012024). Need to retest once the bug is solved.

**Test case ID: 20024016**

### Data preparation:

```
1 •  SELECT * FROM student WHERE
2         first_name= "Jan" AND
3         last_name="Novak" AND
4         email = "jan.novak@gmail." AND
5         age= 54;
6

0% 6:5

Result Grid  Filter Rows: Search Edit: Export/Import:

id age email first_name last_name
NULL NULL NULL NULL NULL

student 37 Apply

ion Output ◊

Time Action Response Duration /
43 11:41:49 SELECT * FROM student WHERE first_name... 0 row(s) returned 0.053 sec
```

## Expecting result:

## HTTP Status code 400

The response body may contain a message about the incorrect data.

### **Actual results:**

POST <http://108.143.193.45:8080/api/v1/students/>

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary **JSON** ▾

```
1 {  
2   ... "firstName": "Jan",  
3   ... "lastName": "Novak",  
4   ... "email": "jan.novak@gmail.",  
5   ... "age": 54  
6 }  
7  
8
```

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 179 ms Size: 248 B

Pretty Raw Preview Visualize **JSON** ▾ ⌂

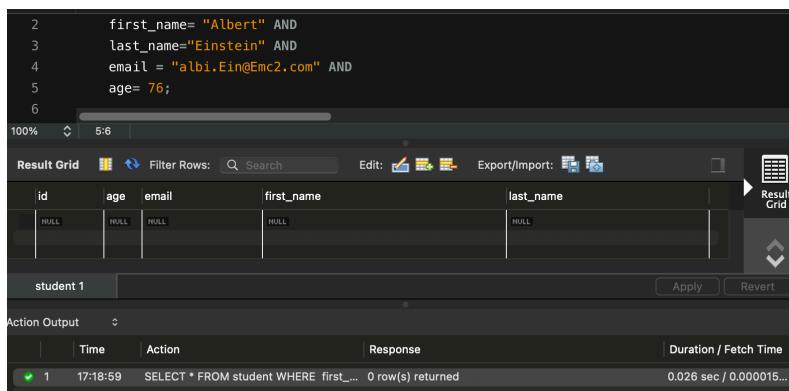
```
1 {"id": 1867,  
2   "firstName": "Jan",  
3   "lastName": "NOVAK",  
4   "email": "jan.novak@gmail.",  
5   "age": 54  
6 }
```

## **Test results: FAILED see notes**

**Notes:** Test failed due to the known bug ( report No, B0042024 and B0012024). Need to retest once the bug is solved.

## **Test case ID: 20024017**

### **Data preparation:**



The screenshot shows the MySQL Workbench interface. At the top, there is a SQL editor window containing the following query:

```
2     first_name= "Albert" AND
3     last_name="Einstein" AND
4     email = "albi.Ein@Emc2.com" AND
5     age= 76;
6
```

Below the query is a Result Grid table with one row of data:

id	age	email	first_name	last_name
HULL	HULL	HULL	HULL	HULL

At the bottom of the interface, the Action Output pane shows the following information:

Time	Action	Response	Duration / Fetch Time
17:18:59	SELECT * FROM student WHERE first_...	0 row(s) returned	0.026 sec / 0.000015...

### **Expecting result:**

#### First inserting:

HTTP status code 201

```
{  
    "id": <unique_id>,  
    "firstName": "Albert",  
    "lastName": "Einstein",  
    "email": "",  
    "age": 76  
}
```

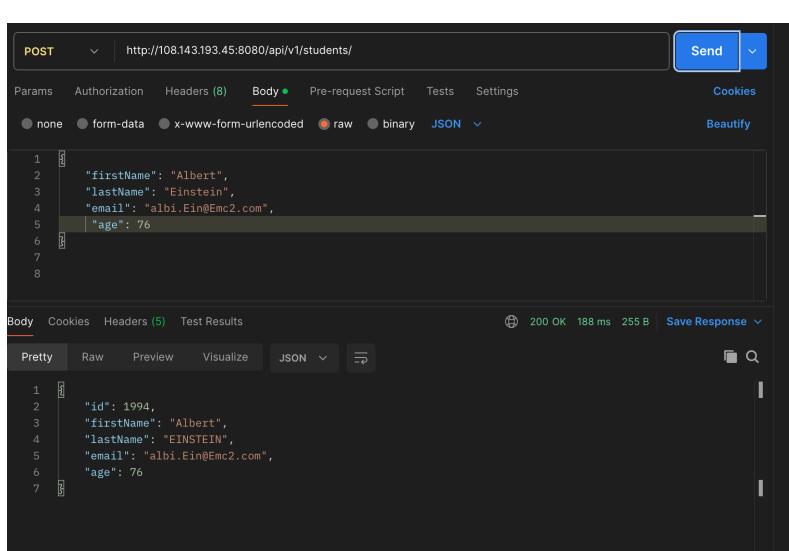
#### Second (duplicated) inserting:

HTTP Status code 409

The response body may contain a message about the already existing data.

### **Actual results:**

#### First inserting:



The screenshot shows the Postman application interface. At the top, it displays a POST request to the URL `http://108.143.193.45:8080/api/v1/students/`.

The Body tab is selected, showing the JSON payload:

```
1  
2     "firstName": "Albert",  
3     "lastName": "Einstein",  
4     "email": "albi.Ein@Emc2.com",  
5     "age": 76  
6  
7
```

Below the request, the Test Results section shows the response details:

Body	Cookies	Headers (5)	Test Results
Pretty	Raw	Preview	Visualize

The response body is displayed as:

```
1  
2     "id": 1994,  
3     "firstName": "Albert",  
4     "lastName": "EINSTEIN",  
5     "email": "albi.Ein@Emc2.com",  
6     "age": 76  
7
```

## Second (duplicated) inserting:

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** http://108.143.193.45:8080/api/v1/students/
- Body (JSON):**

```
1
2   "firstName": "Albert",
3   "lastName": "Einstein",
4   "email": "albi.Ein@Emc2.com",
5   "age": 76
6
7
```
- Response Headers:** 200 OK, 213 ms, 255 B
- Response Body (Pretty):**

```
1
2   "id": 1995,
3   "firstName": "Albert",
4   "lastName": "EINSTEIN",
5   "email": "albi.Ein@Emc2.com",
6   "age": 76
7
```

**Test results:** FAILED see notes

**Notes:** Test failed due to the known bug ( report No, B0032024 and B0012024). Need to retest once the bug is solved.

## Test case ID: 20024018

### Data preparation:

The screenshot shows a database interface with the following query:

```
2     first_name= "Leonardo" AND
3     last_name="da Vinci" AND
4     email = "leo.da.vinci@monalisa.com" AND
5     age= 67;
```

The Result Grid shows the following data:

id	age	email	first_name	last_name
HULL	HULL	NULL	NULL	NULL

Action Output table:

Time	Action	Response	Duration / Fetch Time
17:18:59	SELECT * FROM student WHERE first...	0 row(s) returned	0.026 sec / 0.000015...
17:37:05	SELECT * FROM student WHERE first...	0 row(s) returned	0.028 sec / 0.000024...

### Expecting result:

HTTP Status code 400

The response body may contain a message about the incorrect data.

### Actual results:

20024018/1

The screenshot shows a POST request to `http://108.143.193.45:8080/api/v1/students/`. The Body tab contains the following JSON:

```
1 {
2   "firstName": "Leonardo",
3   "lastName": "Da Vinci",
4   "email": "",
5   "age": 76
}
```

The response status is 200 OK with a timestamp of 2024-10-27T16:42:31.110+00:00, a status of 1996, and a message "Success".

20024018/2

The screenshot shows a POST request to `http://108.143.193.45:8080/api/v1/students/`. The Body tab contains the same JSON as the previous request:

```
1 {
2   "firstName": "Leonardo",
3   "lastName": "Da Vinci",
4   "email": "leo.da.vinci@monalisa.com",
5   "age": ""
}
```

The response status is 500 Internal Server Error with a timestamp of 2024-10-27T16:42:31.110+00:00, a status of 500, and an error message "Internal Server Error".

20024018/3

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

Body (JSON)

```
1
2   "firstName": "Leonardo",
3   "lastName": "",
4   "email": "leo.da.vinci@monalisa.com",
5   "age": "76"
6
7
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

200 OK 268 ms 257 B Save Response

```
1
2   "id": 1999,
3   "firstName": "Leonardo",
4   "lastName": "",
5   "email": "leo.da.vinci@monalisa.com",
6   "age": 76
7
```

20024018/4

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

Body (JSON)

```
1
2   "firstName": "",
3   "lastName": "Da Vinci",
4   "email": "leo.da.vinci@monalisa.com",
5   "age": "76"
6
7
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

200 OK 229 ms 257 B Save Response

```
1
2   "id": 2000,
3   "firstName": "",
4   "lastName": "DA VINCI",
5   "email": "leo.da.vinci@monalisa.com",
6   "age": 76
7
```

20024018/5

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

Body (JSON)

```
1
2   "firstName": "Leonardo",
3   "lastName": "",
4   "email": "",
5   "age": "76"
6
7
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

200 OK 220 ms 232 B Save Response

```
1
2   "id": 2001,
3   "firstName": "Leonardo",
4   "lastName": "",
5   "email": "",
6   "age": 76
7
```

20024018/6

The screenshot shows a Postman interface with a POST request to `http://108.143.193.45:8080/api/v1/students`. The request body is a JSON object with fields: `firstName`, `lastName`, `email`, and `age`. The response status is 500 Internal Server Error, timestamped at 2024-10-27T17:43:35.673+00:00, with a message indicating an Internal Server Error.

```
POST http://108.143.193.45:8080/api/v1/students/ Send  
Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Beautify  
none form-data x-www-form-urlencoded raw binary JSON  
1 "firstName": "",  
2 "lastName": "Da Vinci",  
3 "email": "leo.da.vinci@monalisa.com",  
4 "age": ""  
5  
6  
7  
8  
Body Cookies Headers (4) Test Results 500 Internal Server Error 227 ms 284 B Save Response  
Pretty Raw Preview Visualize JSON  
1 "timestamp": "2024-10-27T17:43:35.673+00:00",  
2 "status": 500,  
3 "error": "Internal Server Error",  
4 "message": "",  
5 "path": "/api/v1/students/"  
6  
7
```

20024018/7

The screenshot shows a Postman interface with a POST request to `http://108.143.193.45:8080/api/v1/students`. The request body is a JSON object with fields: `firstName`, `lastName`, `email`, and `age`. The response status is 500 Internal Server Error, timestamped at 2024-10-27T17:24:48.989+00:00, with a message indicating an Internal Server Error.

```
POST http://108.143.193.45:8080/api/v1/students/ Send  
Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Beautify  
none form-data x-www-form-urlencoded raw binary JSON  
1 "firstName": "",  
2 "lastName": "Da Vinci",  
3 "email": "",  
4 "age": ""  
5  
6  
7  
8  
Body Cookies Headers (4) Test Results 500 Internal Server Error 171 ms 284 B Save Response  
Pretty Raw Preview Visualize JSON  
1 "timestamp": "2024-10-27T17:24:48.989+00:00",  
2 "status": 500,  
3 "error": "Internal Server Error",  
4 "message": "",  
5 "path": "/api/v1/students/"  
6  
7
```

20024018/8

The screenshot shows a Postman interface with a POST request to `http://108.143.193.45:8080/api/v1/students`. The request body is a JSON object with fields: `firstName`, `lastName`, `email`, and `age`. The response status is 500 Internal Server Error, timestamped at 2024-10-27T17:46:15.660+00:00, with a message indicating an Internal Server Error.

```
POST http://108.143.193.45:8080/api/v1/students/ Send  
Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Beautify  
none form-data x-www-form-urlencoded raw binary JSON  
1 "firstName": "",  
2 "lastName": "",  
3 "email": "",  
4 "age": ""  
5  
6  
7  
8  
Body Cookies Headers (4) Test Results 500 Internal Server Error 196 ms 284 B Save Response  
Pretty Raw Preview Visualize JSON  
1 "timestamp": "2024-10-27T17:46:15.660+00:00",  
2 "status": 500,  
3 "error": "Internal Server Error",  
4 "message": "",  
5 "path": "/api/v1/students/"  
6  
7
```

**Test results:**

20024018/1 FAILED see notes

**Notes:** Test failed due to the known bug ( report No, B0042024 and B0012024). Need to retest once the bug is solved.

20024018/2 FAILED see note

**Notes:** Bug B0062024

20024018/3 FAILED see note

**Notes:** Test failed due to the known bug ( report No, B0042024 and B0012024). Need to retest once the bug is solved.

20024018/4 FAILED see note

**Notes:** Test failed due to the known bug ( report No, B0042024 and B0012024). Need to retest once the bug is solved.

20024018/5 FAILED see note

**Notes:** Test failed due to the known bug ( report No, B0042024 and B0012024). Need to retest once the bug is solved.

20024018/6 FAILED see note

**Notes:** Already noted bug B0062024

20024018/7 FAILED see note

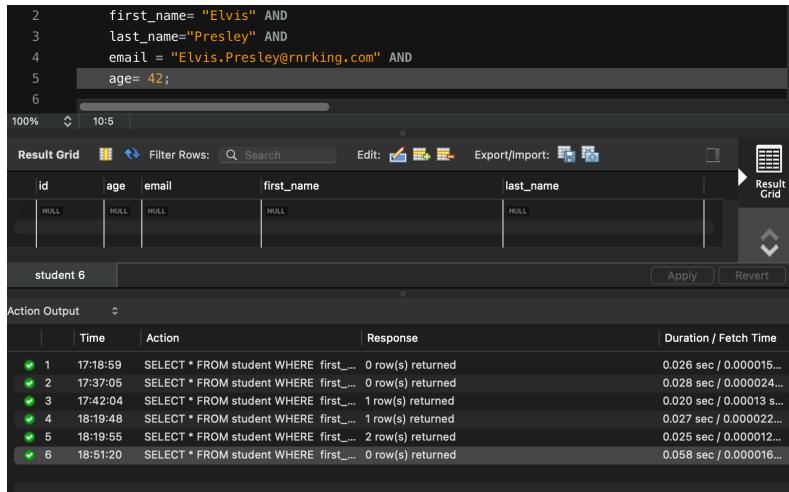
**Notes:** Already noted bug B0062024

20024018/8 FAILED see note

**Notes:** Already noted bug B0062024

## Test case ID: 20024019

### Data preparation:



The screenshot shows a database interface with the following details:

- Query: 

```
2     first_name= "Elvis" AND
3     last_name= "Presley" AND
4     email = "Elvis.Presley@rnrking.com" AND
5     age= 42;
6
```
- Result Grid: A table with columns id, age, email, first\_name, and last\_name. All rows are null.
- Action Output: A table showing the history of actions taken on the student record. It includes columns Time, Action, Response, and Duration / Fetch Time. The actions listed are all successful SELECT operations.

### Expecting result:

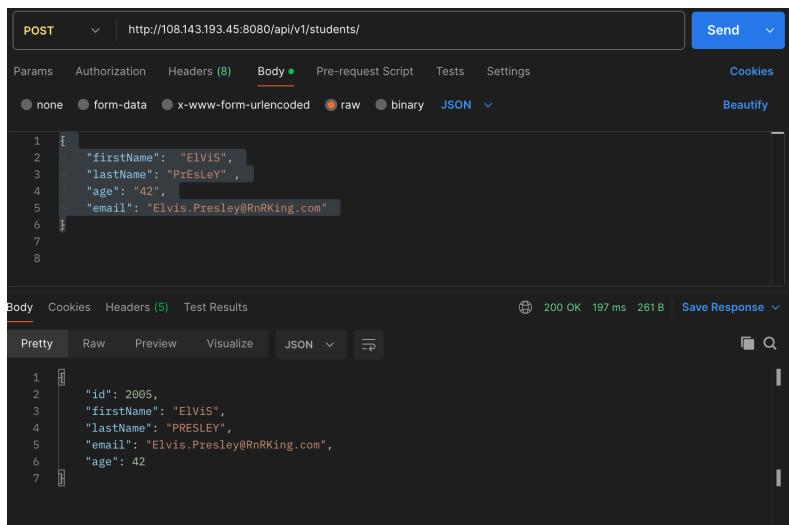
HTTP Status code 201

Format: JSON

Responds body contains:

```
{
  "firstName": "Elvis",
  "lastName": "Presley",
  "age": 42,
  "email": "elvis.presley@rnrking.com"
}
```

### Actual results:



The screenshot shows a Postman test run with the following details:

- Method: POST
- URL: <http://108.143.193.45:8080/api/v1/students>
- Body tab selected, showing a JSON payload:

```
{
  "firstName": "ElViS",
  "lastName": "PrEsLeY",
  "age": "42",
  "email": "Elvis.Presley@RnRKing.com"
```
- Response tab showing:
  - Status: 200 OK
  - Time: 197 ms
  - Size: 261 B

### Test results: FAILED see notes

**Notes:** Test failed due to the known bug ( report No, B0032024 and B0012024). Need to retest once the bug is solved.

## Test case ID: 20024020

### Data preparation:

The screenshot shows the MySQL Workbench interface. At the top, a query is being run:

```
1 • | SELECT * FROM student WHERE
2   first_name= "Julius" AND
3   last_name="Caesar" AND
4   email = "ave.caesar@venividivici.com" AND
5   age = 57.
```

The results grid shows the following data:

id	age	email	first_name	last_name
HULL	HULL	HULL	HULL	HULL

The Action Output section shows the execution history:

Time	Action	Response	Duration / Fetch Time
17:18:59	SELECT * FROM student WHERE first...	0 row(s) returned	0.026 sec / 0.000015...
17:37:05	SELECT * FROM student WHERE first...	0 row(s) returned	0.028 sec / 0.000024...
17:42:04	SELECT * FROM student WHERE first...	1 row(s) returned	0.020 sec / 0.00013 s...
18:19:48	SELECT * FROM student WHERE first...	1 row(s) returned	0.027 sec / 0.000022...
18:19:55	SELECT * FROM student WHERE first...	2 row(s) returned	0.025 sec / 0.000012...
18:51:20	SELECT * FROM student WHERE first...	0 row(s) returned	0.058 sec / 0.000016...
18:59:12	SELECT * FROM student WHERE SELE...	Error Code: 1105. syntax error at position 35 near 'SE...	0.077 sec
18:59:19	SELECT * FROM student WHERE first...	0 row(s) returned	0.044 sec / 0.000016...

### Expecting result:

HTTP Status code 400

The response body may contain a message about the incorrect data.

### Actual results:

2002420/1

The screenshot shows the Postman interface. A POST request is made to `http://108.143.193.45:8080/api/v1/students/`.

The Body tab contains the following JSON payload:

```
1 {
2   "firstName": "J@lius",
3   "lastName": "Caesar",
4   "age": "57",
5   "email": "ave.caesar@venividivici.com"
}
```

The response shows a status of 200 OK with a response time of 261 ms and a response size of 263 B. The response body is:

```
1 {
2   "id": 2006,
3   "firstName": "Julius",
4   "lastName": "CAESAR",
5   "email": "ave.caesar@venividivici.com",
6   "age": 57
}
```

2002420/2

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

Body (8)

```
1 "firstName": "Jalius",
2 "lastName": "@esax",
3 "age": "57",
4 "email": "ave.caesar@venividivici.com"
```

Send Cookies

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary JSON Beautify

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 "id": 2007,
2 "firstName": "Jalius",
3 "lastName": "@ESAR",
4 "email": "ave.caesar@venividivici.com",
5 "age": 57
```

200 OK 212 ms 263 B Save Response

2002420/3

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

Body (8)

```
1 "firstName": "J@lius",
2 "lastName": "C@esar",
3 "age": "57",
4 "email": "ave.caesar@venividivici.com"
```

Send Cookies

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary JSON Beautify

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 "id": 2008,
2 "firstName": "J@lius",
3 "lastName": "@ESAR",
4 "email": "ave.caesar@venividivici.com",
5 "age": 57
```

200 OK 159 ms 263 B Save Response

### Test results:

2002420/1 FAILED bug B0042024 and B0012024

2002420/2 FAILED bug B0042024 and B0012024

2002420/3 FAILED bug B0042024 and B0012024

## **Test case ID: 20024021**

### **Data preparation:**

```
1 ●  SELECT * FROM student WHERE
2      first_name= "William" AND
3      last_name="Shakespeare" AND
4      email = "will.shakespeare@hamlet.net" AND
5      age= 52;
6
100% 10:5
Result Grid Filter Rows: Search Edit: Export/Import: student>0
```

id	age	email	first_name	last_name
HULL	HULL	HULL	HULL	HULL

### **Expecting result:**

HTTP Status code 400

The response body may contain a message about the unexpected data.

### **Actual result:**

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

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Body (8) x-www-form-urlencoded raw binary JSON

```
1 {<highlight>"id": 2011</highlight>
2   "firstName": "William",
3   "lastName": "Shakespeare",
4   "email": "will.Shakespeare@hamlet.net",
5   "age": 52,
6   "book" : "Rome and Juliet"
7 }
```

Body Cookies Headers (5) Test Results 200 OK 165 ms 269 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {<highlight>"id": 2011</highlight>
2   "firstName": "William",
3   "lastName": "SHAKESPEARE",
4   "email": "will.Shakespeare@hamlet.net",
5   "age": 52
6 }
```

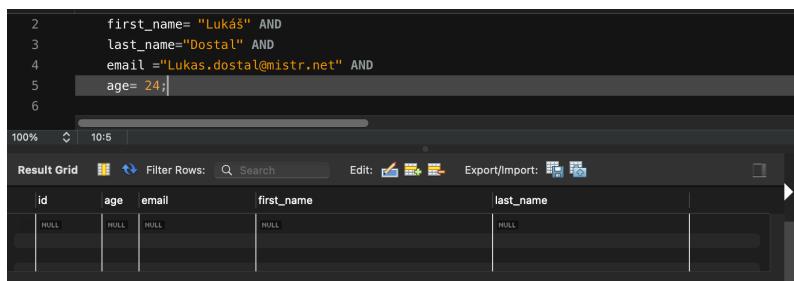
### **Test result:**

**Test results:** FAILED see notes

**Notes:** Test failed due to the known bug ( report No, B0042024 and B0012024). Need to retest once the bug is solved.

## **Test case ID: 20024022**

### **Data preparation:**



```
2     first_name= "Lukáš" AND
3     last_name="Dostál" AND
4     email ="Lukas.dostal@mistr.net" AND
5     age= 24;
6
```

Result Grid | Filter Rows: Search | Edit: Export/Import: |

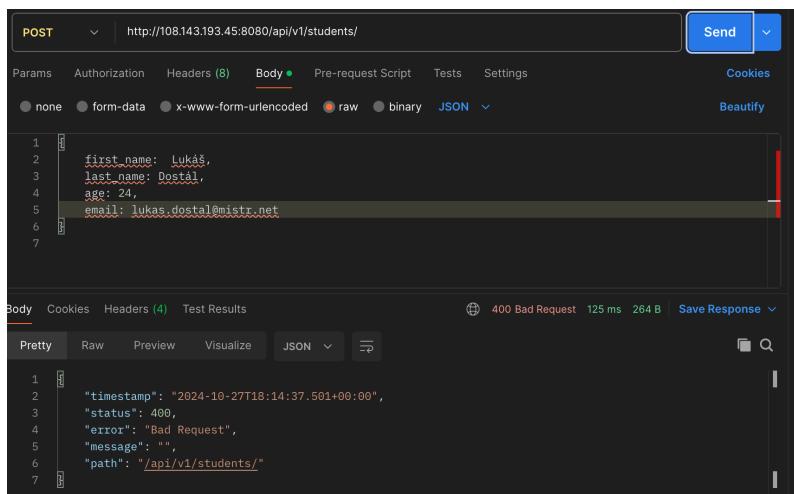
id	age	email	first_name	last_name
HULL	HULL	HULL	HULL	HULL

### **Expecting result:**

HTTP Status code 400

The response body may contain a message about the invalid format.

### **Actual result:**



POST | http://108.143.193.45:8080/api/v1/students/ | Send

Params | Authorization | Headers (8) | **Body** | Pre-request Script | Tests | Settings | Cookies | Beautify

• none • form-data • x-www-form-urlencoded • raw • binary | **JSON** |

```
1
2     first_name: Lukáš,
3     last_name: Dostál,
4     age: 24,
5     email: lukas.dostal@mistr.net
6
7
```

Body | Cookies | Headers (4) | Test Results | 400 Bad Request | 125 ms | 264 B | Save Response |

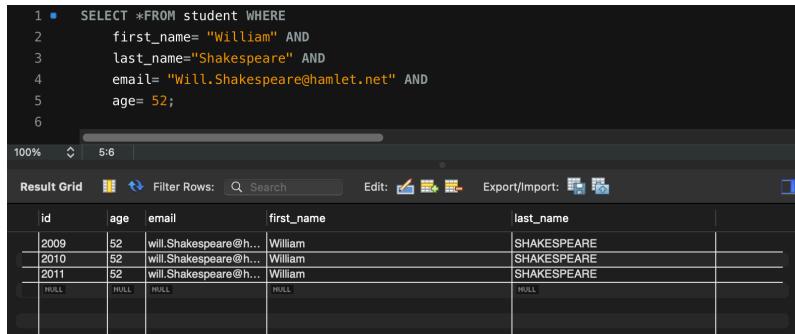
Pretty | Raw | Preview | Visualize | **JSON** |

```
1
2     "timestamp": "2024-10-27T18:14:37.501+00:00",
3     "status": 400,
4     "error": "Bad Request",
5     "message": "",
6     "path": "/api/v1/students/"
```

**Test result: PASSED**

## Test case ID: 20024023

### Data preparation:



A screenshot of a database interface showing a query and its results. The query is:

```
1 •  SELECT *FROM student WHERE
2      first_name= "William" AND
3      last_name="Shakespeare" AND
4      email= "Will.Shakespeare@hamlet.net" AND
5      age= 52;
6
```

The result grid shows three rows of data:

id	age	email	first_name	last_name
2009	52	will.Shakespeare@h...	William	SHAKESPEARE
2010	52	will.Shakespeare@h...	William	SHAKESPEARE
2011	52	will.Shakespeare@h...	William	SHAKESPEARE

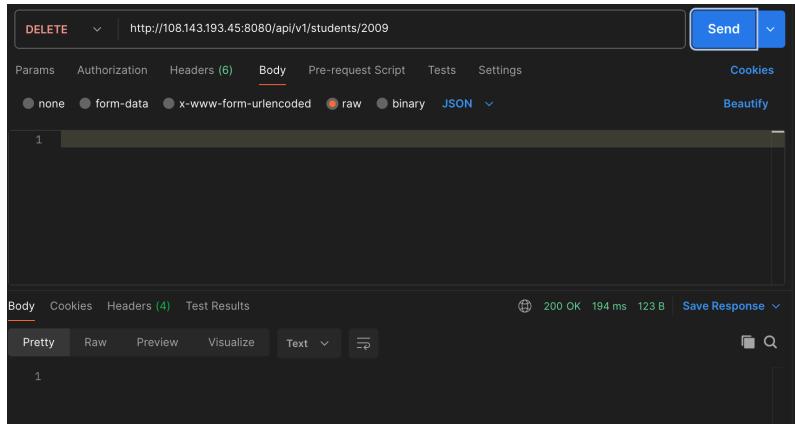
ID number: 2009

### Expecting result:

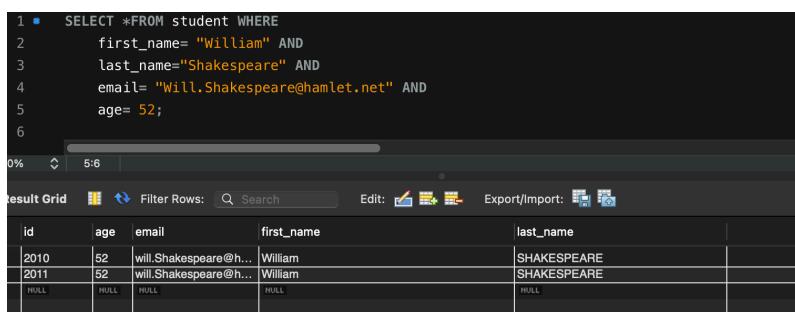
HTTP Status code 200

The response body may contain a message about successfully student record delete.

### Actual result:



A screenshot of the Postman application showing a DELETE request to the endpoint `http://108.143.193.45:8080/api/v1/students/2009`. The request body is empty. The response status is 200 OK with a response time of 194 ms and a body size of 123 B.



A screenshot of a database interface showing a query and its results. The query is:

```
1 •  SELECT *FROM student WHERE
2      first_name= "William" AND
3      last_name="Shakespeare" AND
4      email= "Will.Shakespeare@hamlet.net" AND
5      age= 52;
6
```

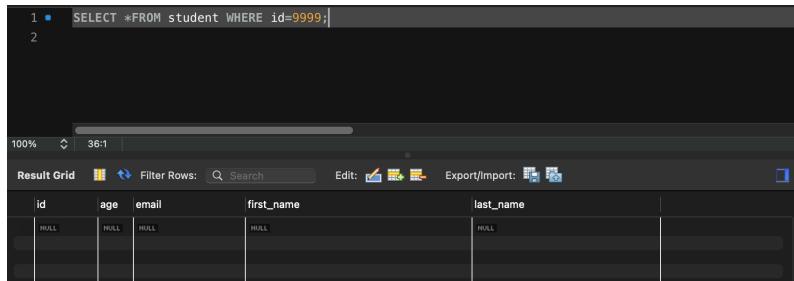
The result grid shows two rows of data:

id	age	email	first_name	last_name
2010	52	will.Shakespeare@h...	William	SHAKESPEARE
2011	52	will.Shakespeare@h...	William	SHAKESPEARE

Test result: PASSED

## **Test case ID: 20024024**

### **Data preparation:**



A screenshot of MySQL Workbench showing a query result grid. The query is:

```
1 • SELECT *FROM student WHERE id=9999;
```

The result grid has columns: id, age, email, first\_name, last\_name. All rows show NULL values.

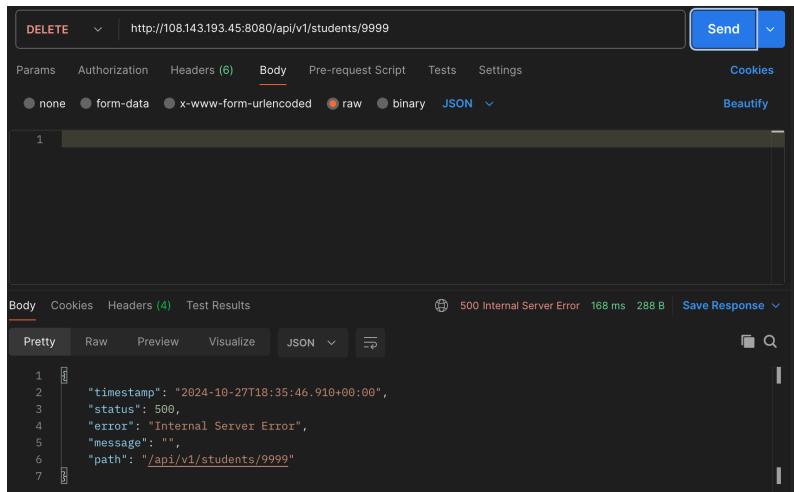
id	age	email	first_name	last_name
NULL	NULL	NULL	NULL	NULL

### **Expecting result:**

HTTP status code 404

The response body may contain a message about the non existing student.

### **Actual result:**



A screenshot of Postman showing a DELETE request to `http://108.143.193.45:8080/api/v1/students/9999`. The response is:

Body

```
1
2
3
4
5
6
7
```

500 Internal Server Error

```
1
2
3
4
5
6
7
```

Pretty Raw Preview Visualize JSON

**Test results:** FAILED see notes

**Notes:** Test failed due to the known bug ( report No, B0022024). Need to retest once the bug is solved.

## **Test case ID: 20024025**

### **Data preparation:**

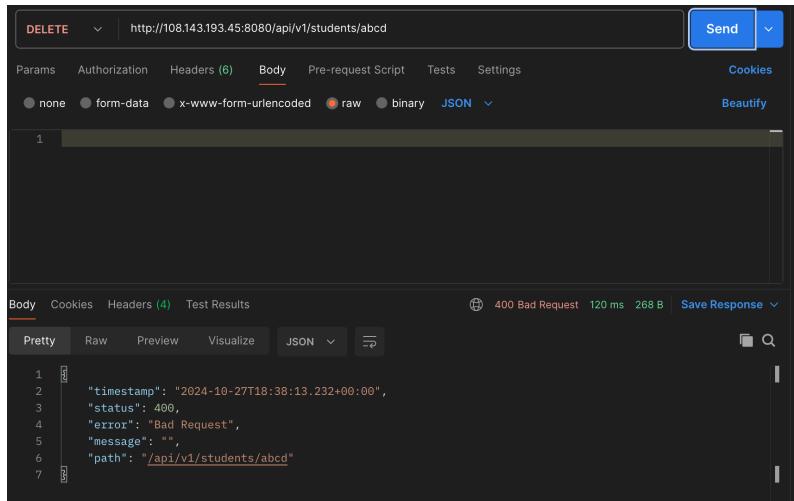
No specific data preparation is required for this scenario.

### **Expecting result:**

HTTP Status code 400

The response body may contain a message about the invalid data.

### **Actual result:**



The screenshot shows the Postman application interface. A DELETE request is being made to the URL `http://108.143.193.45:8080/api/v1/students/abcd`. The 'Body' tab is selected, showing a JSON response with the following content:

```
1
2   "timestamp": "2024-10-27T18:38:13.232+00:00",
3   "status": 400,
4   "error": "Bad Request",
5   "message": "",
6   "path": "/api/v1/students/abcd"
```

The status bar at the bottom indicates a 400 Bad Request, 120 ms response time, and 268 B size. The response is saved under the name 'Response'.

**Test results: PASSED**

## **Test case ID: 20024026**

### **Data preparation:**

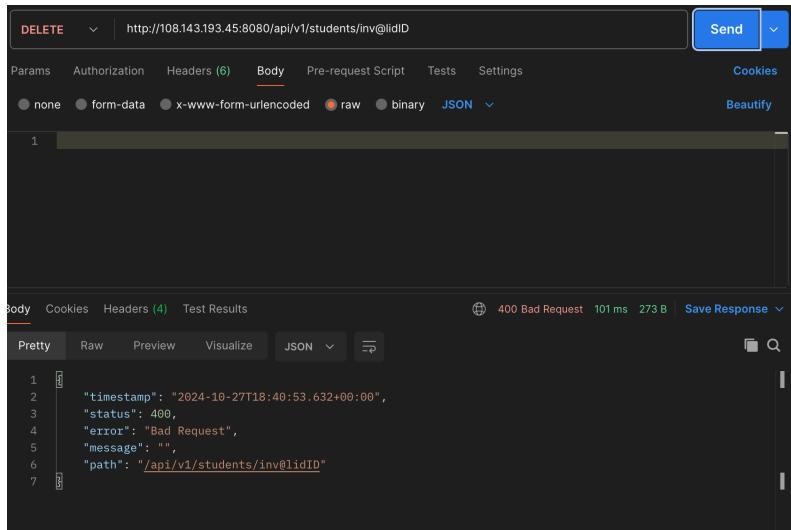
No specific data preparation is required for this scenario.

### **Expecting result:**

HTTP Status code 400

The response body may contain a message about the invalid data.

### **Actual result:**



The screenshot shows the Postman application interface. A DELETE request is being made to the URL `http://108.143.193.45:8080/api/v1/students/inv@lidID`. The request is set to 'Body' type and 'JSON' format. The response status is 400 Bad Request, with a timestamp of 2024-10-27T18:40:53.632+00:00, status 400, error 'Bad Request', message '', and path '/api/v1/students/inv@lidID'.

```
1  "timestamp": "2024-10-27T18:40:53.632+00:00",
2  "status": 400,
3  "error": "Bad Request",
4  "message": "",
5  "path": "/api/v1/students/inv@lidID"
```

### **Test results: PASSED**

## **Test case ID: 20024027**

### **Data preparation:**

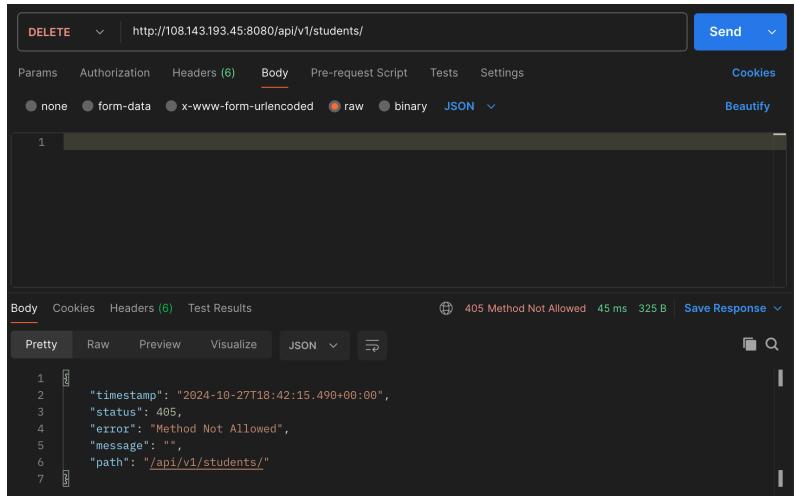
No specific data preparation is required for this scenario.

### **Expecting result:**

HTTP Status code 400

The response body may contain a message about the invalid data.

### **Actual result:**



The screenshot shows the Postman application interface. A DELETE request is being made to the URL `http://108.143.193.45:8080/api/v1/students`. The Headers tab shows six entries. The Body tab is selected, showing the option "raw" is chosen. The response section shows a status of 405 Method Not Allowed, with a timestamp of `2024-10-27T18:42:15.490+00:00`, status 405, error "Method Not Allowed", message "", and path "/api/v1/students/".

```
1
2   "timestamp": "2024-10-27T18:42:15.490+00:00",
3   "status": 405,
4   "error": "Method Not Allowed",
5   "message": "",
6   "path": "/api/v1/students/"
```

**Test results: PASSED**

# BUG REPORT

*Based on the performed scenarios, I discovered the mentioned application errors.*

Bug ID	Title	Test Case ID	Priority	Status	Comments
B0012024	Last name returned in all uppercase letters instead of the correct format	2024001-20 2412	Medium	In progress	
B0022024	API returns 500 Internal Server Error instead of 404 for invalid student ID	2024002	High	In progress	
B0032024	API returns 200 OK instead of 201 for creating new record.	2024003-20 24008	Lower	In progress	
B0042024	API returns 200 OK instead of 400 for creating invalid record	2024010 ; 2024011;20 24012 ;2024018	High	In progress	
B0052024	API returns 200 OK instead of 409 for creating duplicated record	2024017	High	In progress	
B0062024	API returns 500 instead of 400 for inserting missing values	2024018/1 ; 2024018/6	High	In progress	

**Bug ID: B0012024**

**Title:** Last name returned in all uppercase letters instead of the correct format

**Created by:** Market Hořáková

**Date:** 22.9.2024

**Priority:** Medium

**Assignee:** Development dep.

**Description:**

When executing test scenario ID: 2024001 (GET student based on ID), API returns last\_name with all uppercase letters instead of the correct format lowercase

**Environment:**

*URL: http://108.143.193.45:8080/api/v1/students/*

*Postman Version: 11.6.1 (11.6.1)*

*Operating System: macOS 14.5*

**Steps:**

1. Open app Postman.
2. Set up method GET.
3. Insert URL: *http://108.143.193.45:8080/api/v1/students/357*
4. Push button SEND.

**Expecting results:**

HTTP Status code 200 (OK)

**Responds:**

```
{ "id": 357,  
  "firstName": "John",  
  "lastName": "Doe",  
  "email": "john.doe@gmail.com",  
  "age": 18}
```

**Actual results:**

HTTP Status code 200 (OK)

**Responds:**

```
{ "id": 357,  
  "firstName": "John",  
  "lastName": "DOE",  
  "email": "john.doe@gmail.com",  
  "age": 18}
```

**Attachment:** Screenshot of the response in Postman.

**Bug ID: B0022024**

**Title:** API returns 500 Internal Server Error instead of 404 for invalid student ID

**Created by:** Market Hořáková

**Date:** 22.9.2024

**Priority:** High

**Assignee:** Development dep.

**Description:**

When executing test scenario ID: 2024002 (DELETE request with an invalid student ID), API returns a 500 Internal Server Error instead of the expected 404 Not Found.

**Environment:**

*URL: http://108.143.193.45:8080/api/v1/students/*

*Postman Version: 11.6.1 (11.6.1)*

*Operating System: macOS 14.5*

**Steps:**

1. Open app Postman.
2. Set up method GET.
3. Insert URL: *http://108.143.193.45:8080/api/v1/students/9999*
4. Push button SEND.

**Expecting results:**

HTTP Status code 404 (Not Found)

Note: The response body may contain a message about the non existing student.

**Actual results:**

HTTP Status code 500 Internal Server Error

**Attachment:** Screenshot of the response in Postman.

**Bug ID: B0032024**

**Title:** API returns 200 OK instead of 201 for creating new student.

**Created by:** Market Hořáková

**Date:** 22.9.2024

**Priority:** Lower

**Assignee:** Development dep.

**Description:**

When executing test scenario ID: 2024003 (Create new student), API returns 200 OK instead of the expected 201 Created.

**Environment:**

*URL: http://108.143.193.45:8080/api/v1/students/*

*Postman Version: 11.6.1 (11.6.1)*

*Operating System: macOS 14.5*

**Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: *http://108.143.193.45:8080/api/v1/students/*
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
    "firstName": "Jan",  
    "lastName": "Kim",  
    "email": "jan.kim@gmail.com",  
    "age": 35  
}
```

7. Click on the SEND button

**Expecting results:**

HTTP Status code 201 (Created)

**Actual results:**

HTTP Status code 200 OK

**Attachment:** Screenshot of the response in Postman.

## **Bug ID: B0042024**

**Title:** API returns 200 OK instead of 400 for creating invalid record.

**Created by:** Market Hořáková

**Date:** 10.10.2024

**Priority:** High

**Assignee:** Development dep.

### **Description:**

When executing test scenario ID: 20240010 ( Testing boundary values for the attributes **firstName** and **lastName** (bellow minimum length - 2 characters) API returns 200 OK instead of 400 for creating invalid record.

### **Environment:**

*URL: <http://108.143.193.45:8080/api/v1/students/>*

*Postman Version: 11.6.1 (11.6.1)*

*Operating System: macOS 14.5*

### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: *http://108.143.193.45:8080/api/v1/students/*
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
  "firstName": "Oz",  
  "lastName": "Li",  
  "email": "jan.kim@gmail.com",  
  "age": 35  
}
```

7. Click on the SEND button
8. Check expecting results.

### **Expecting results:**

HTTP Status code 400

The response body may contain a message about the incorrect data.

### **Actual results:**

HTTP Status code 200 OK

**Attachment:** Screenshot of the response in Postman.

## **Bug ID: B0052024**

**Title:** API returns 200 OK instead of 409 for creating duplicated record.

**Created by:** Market Hořáková

**Date:** 27.10.2024

**Priority:** High

**Assignee:** Development dep.

### **Description:**

When executing test scenario ID: 20240010 ( Testing creation of duplication) API returns 200 OK instead of 409 for creating duplicated record.

### **Environment:**

*URL: http://108.143.193.45:8080/api/v1/students/*

*Postman Version: 11.6.1 (11.6.1)*

*Operating System: macOS 14.5*

### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: *http://108.143.193.45:8080/api/v1/students/*
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
    "firstName": "Albert",  
    "lastName": "Einstein",  
    "email": "",  
    "age": 76  
}
```

7. Click on the SEND button
8. Check expecting HTTP status code is 409 ( Conflict)

### **Expecting results:**

HTTP Status code 409

The response body may contain a message about the already existing data.

### **Actual results:**

HTTP Status code 200 OK

**Attachment:** Screenshot of the response in Postman.

## **Bug ID: B0062024**

**Title:** API returns 500 OK instead of 400 for creating invalid /missing record.

**Created by:** Market Hořáková

**Date:** 27.10.2024

**Priority:** High

**Assignee:** Development dep.

### **Description:**

When executing test scenario ID: 20240018/2 ( Testing empty attributes) API returns 200 OK instead of 400 for creating invalid / missing record.

### **Environment:**

*URL: http://108.143.193.45:8080/api/v1/students/*

*Postman Version: 11.6.1 (11.6.1)*

*Operating System: macOS 14.5*

### **Steps:**

1. Open app Postman.
2. Set up method POST.
3. Insert URL: *http://108.143.193.45:8080/api/v1/students/*
4. Go to the "Body" tab.
5. Select "raw" and "JSON" as the input type.
6. Insert the following data into the body:

```
{  
  "firstName": "Leonardo",  
  "lastName": "Da Vinci",  
  "email": "leo.da.vinci@monalisa.com",  
  "age": ""  
}
```

7. Click on the SEND button

### **Expecting results:**

HTTP Status code 400

The response body may contain a message about the missing data / values.

### **Actual results:**

HTTP Status code 500 Internal Server Error

**Attachment:** Screenshot of the response in Postman.