FINAL PROJECT
No.1

Author: Markéta Hořáková
Date: 27.10.2024

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;

id	*age*	*email*	*first_name*	*last_name*
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= "Jaroslavoslavoslavoslavoslavoslavoslavoslavos" AND
last_name="Skočdopolemiloslavoslavoslavoslavoslavoslavoslavol" 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": "Jaroslavoslavoslavoslavoslavoslavoslavoslavos",
   "lastName": "Skočdopolemiloslavoslavoslavoslavoslavoslavol",
   "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": "Jaroslavoslavoslavoslavoslavoslavoslavoslavos",
"lastName": "Skočdopolemiloslavoslavoslavoslavoslavoslavol",
"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** (bellow 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 last name
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= "Bohumilomiroslavoslavoslavoslavoslavoslavoslavoslavoslav" AND
last_name="Skočdopolemiloslavoslavoslavoslavoslavoslavoslavoslavoslavosl" AND
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": "Bohumilomiroslavoslavoslavoslavoslavoslavoslavoslavoslav",
   "lastName": "Skočdopolemiloslavoslavoslavoslavoslavoslavoslavoslavoslavosl",
   "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.novakgmail.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.novakgmail.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 atribut.

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 bellow 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
20024018/1	Leonardo	Da Vinci	76		Missing e-mail	400	The response body may contain a message about the presence of missing atribut.
20024018/2	Leonardo	Da Vinci		leo.da.vinci@monalisa.com	Missing age	400	The response body may contain a message about the presence of missing atribut.
20024018/3	Leonardo		76	leo.da.vinci@monalisa.com	Missing last name	400	The response body may contain a message about the presence of missing atribut.

20024018/4		Da Vinci	76	leo.da.vinci@monalisa.com	Missing first name	400	The response body may contain a message about the presence of missing atribut.
20024018/5	Leonardo		76		Missing last name and e-mail	400	The response body may contain a message about the presence of missing atribut.
20024018/6		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 atribut.
20024018/7		Da Vinci			missing first name, age, e-mail	400	The response body may contain a message about the presence of missing atribut.
20024018/8					Missing all fields	400	The response body may contain a message about the presence of missing atribut.

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
20024020/1	J@lius	Caesar	57	ave.caesar@venividivici.com	Special character in firstName	400	The response body may contain a message about the presence of invalid or unexpected characters.
20024020/2	Julius	C@esar	57	ave.caesar@venividivici.com	Special character in lastName	400	The response body may contain a message about the presence of invalid or unexpected characters.
20024020/3	J@lius	C@esar	57	ave.caesar@venividivici.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

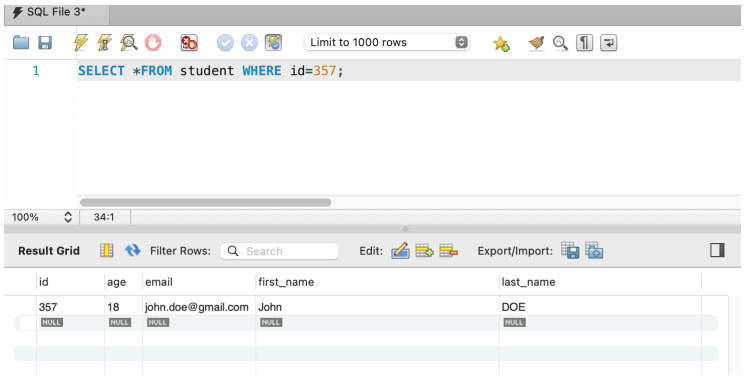
Test Case ID	Test Title	Execution Date	Executed by	Result	Comments
2024001	Verify successful retrieval of existing student.	2024-09-22	Markéta Hořáková	Passed	No issues encountered.
2024002	Search non-existing student	2024-09-22	Markéta Hořáková	Failed	Received 500 error instead of 404.
2024003	Boundary 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;B0032024
2024005	Boundary testing first and last name	2024-10-10	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0012024;B0032024
2024006	Boundary testing age	2024-10-10	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0012024;B0032024
2024007	Boundary testing age	2024-10-10	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0012024;B0032024

2024008	Boundary testing age	2024-10-10	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0012024;B0032024
2024009	Verification of valid attribute email.	2024-10-10	Markéta Hořáková	Failed	Known issue, related to Bug ID: B0012024;B0032024
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

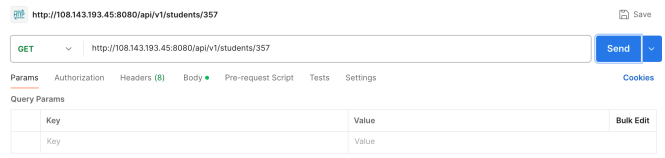
Data preparation:

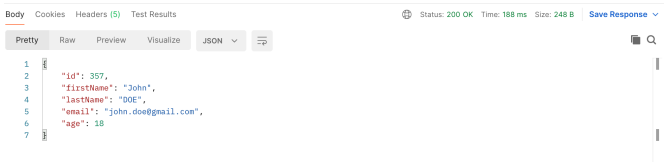


Expecting results:
HTTP Status code: 200 (OK)
Responds:

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

Actual Results:
HTTP Status Code: 200 (OK)
Response Body:

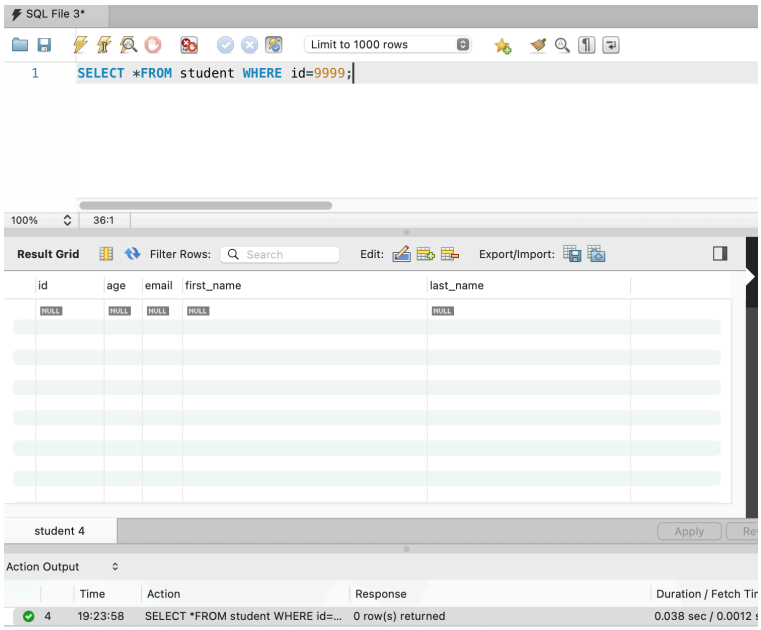




Test results: FAILED

Test case ID: 20024002

Data preparation:

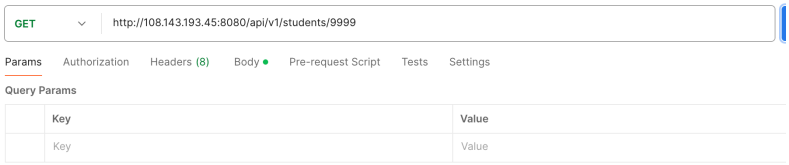


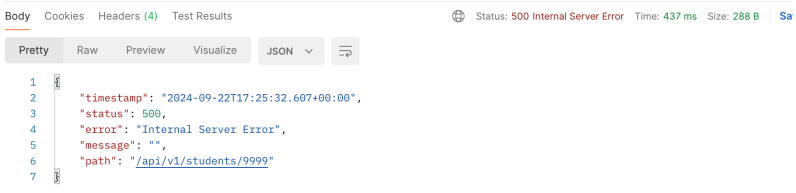
Expecting results:

HTTP Status code 404 (Not Found)

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

Actual Results:

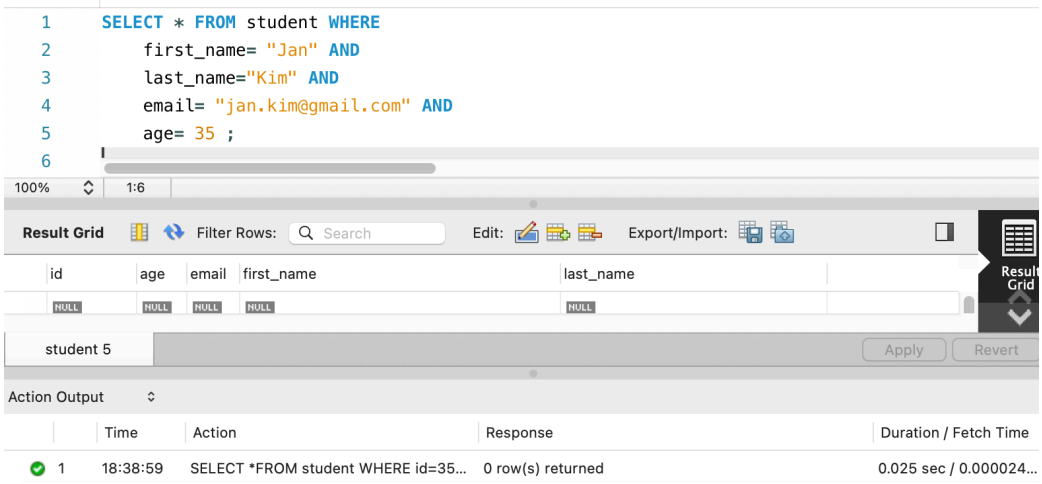




Test results: FAILED

Test case ID: 20024003

Data preparation:



Expecting results:

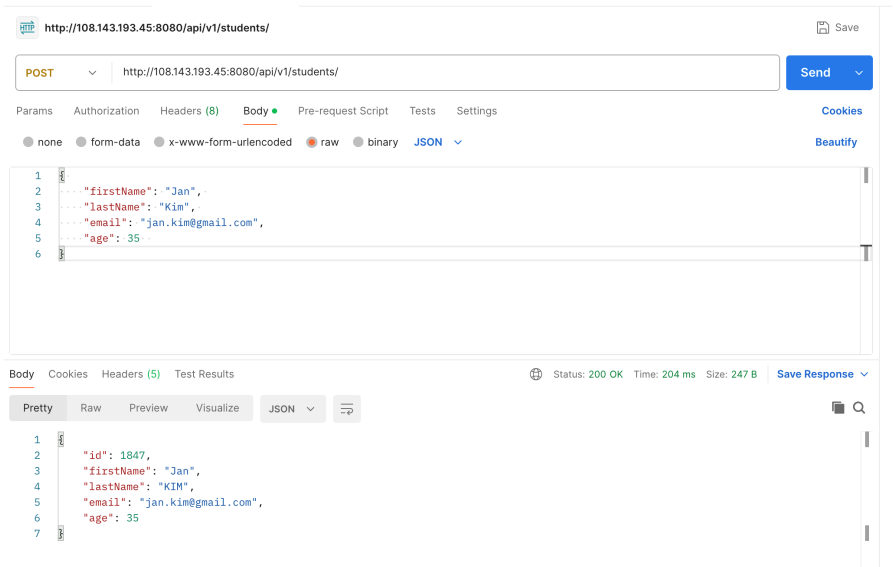
HTTP Status code 201

Format: JSON

Responds body contains:

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

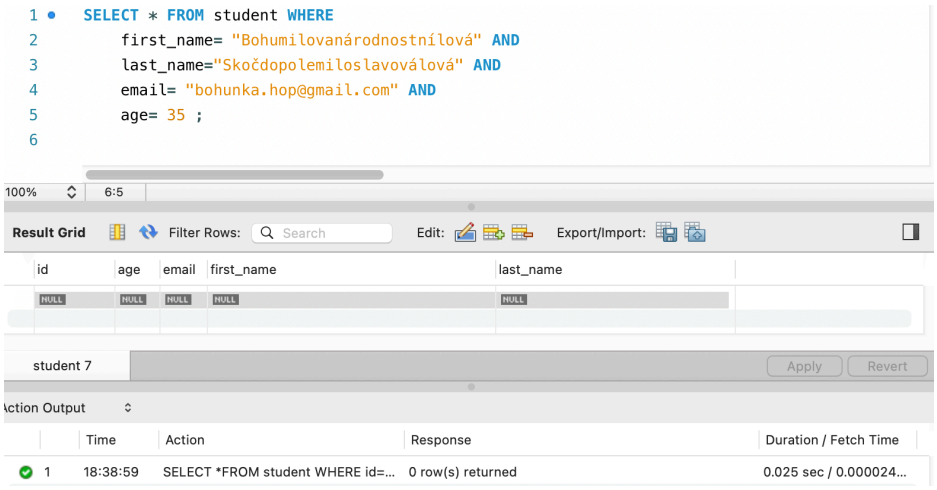
Actual Results:



Test results: FAILED

Test case ID: 20024004

Data preparation:



Expecting results:

HTTP Status code 201

Format: JSON

Responds body contains:

{ "id": <unique_id>,

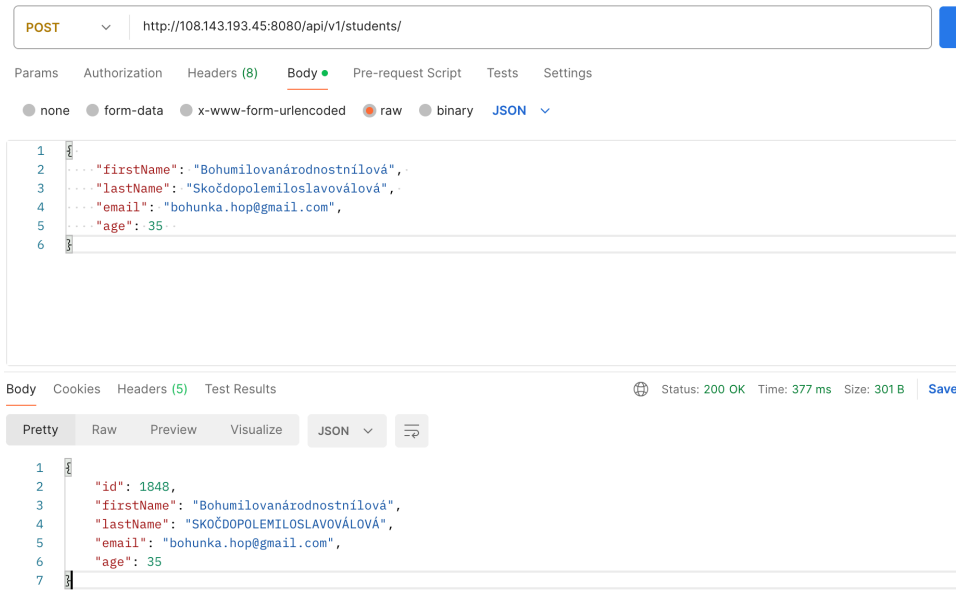
"firstName": "Bohumilovanárodnostnílová",

"lastName": "Skočdopolemiloslavoválová",

"email": "bohunka.hop@gmail.com",

"age": 35 }

Actual Results:

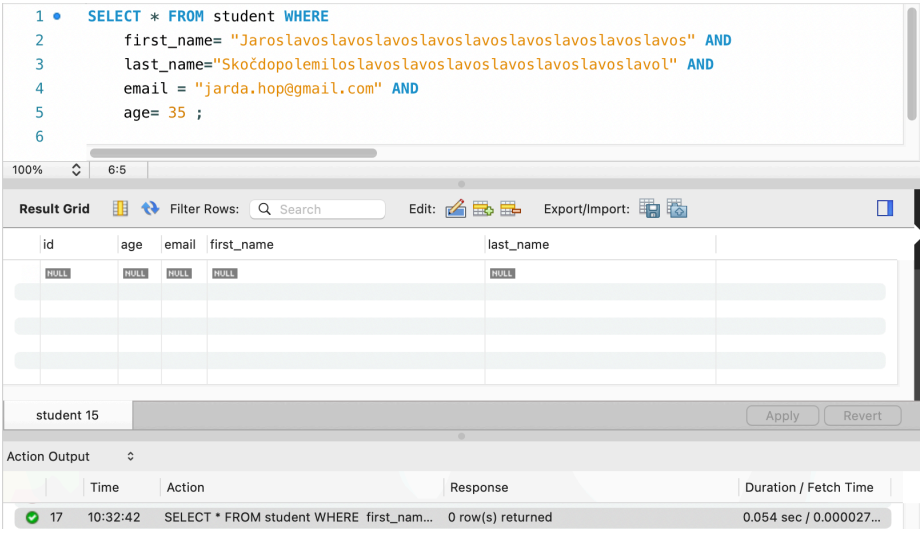


Test results: FAILED see notes

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:



Expecting results:

HTTP Status code 201

Format: JSON

Responds body contains:

{

"id": <unique_id>,

"firstName": "Jaroslavoslavoslavoslavoslavoslavoslavoslavoslavos",

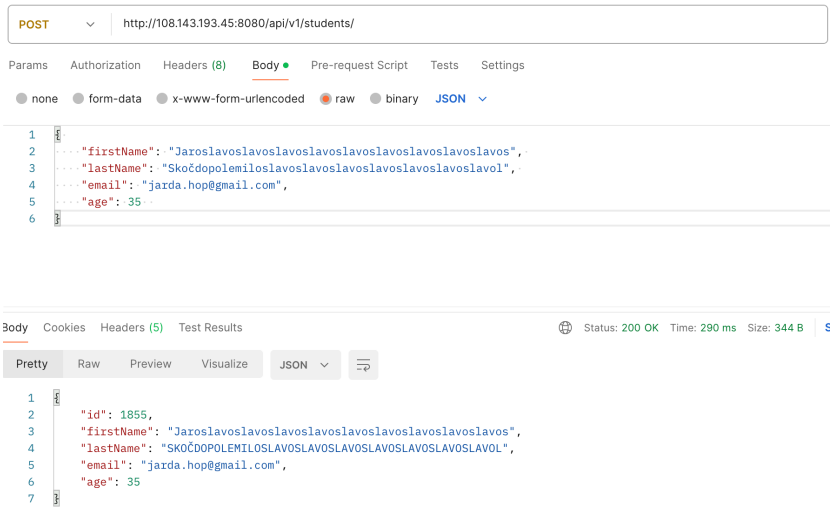
"lastName": "Skočdopolemiloslavoslavoslavoslavoslavoslavoslavol",

"email": "jarda.hop@gmail.com",

"age": 35

}

Actual Results:

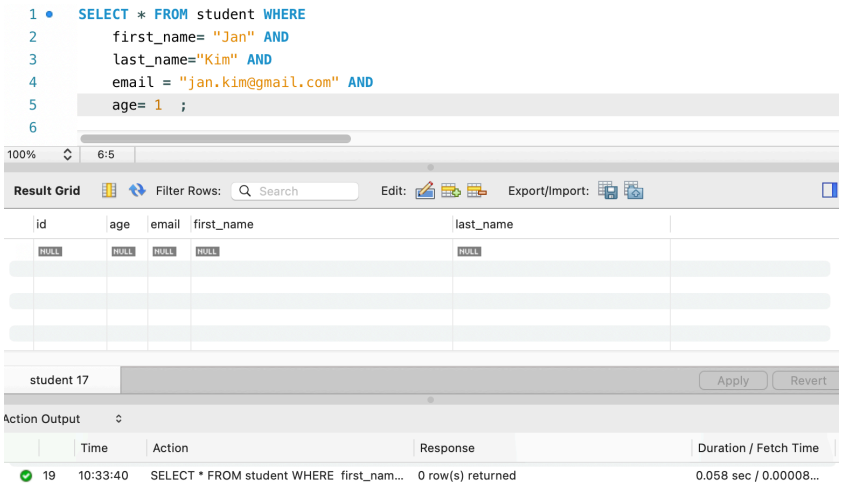


Test results: FAILED see notes

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",

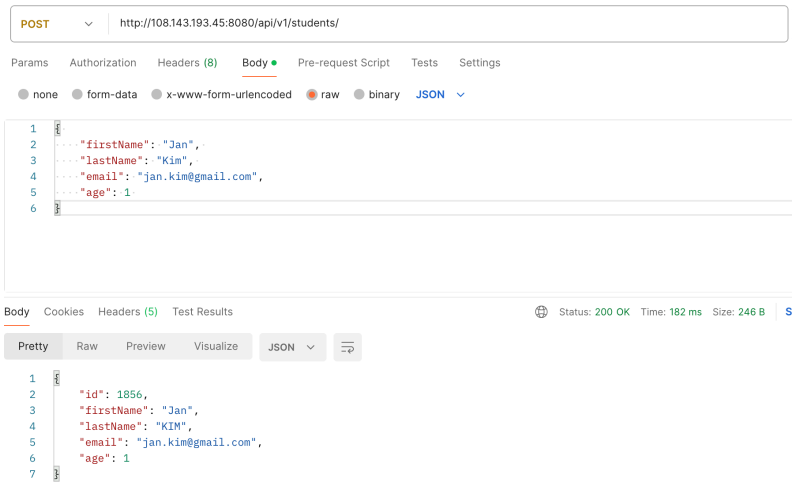
"lastName": "Kim",

"email": "jan.kim@gmail.com",

"age": 1

}

Actual Results:

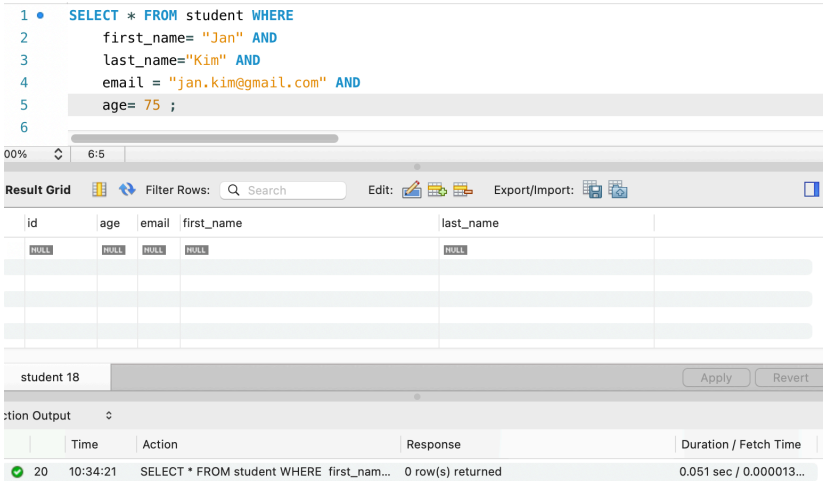


Test results: FAILED see notes

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:



Expecting results:

HTTP Status code 201

Format: JSON

Responds body contains:

{

"id": <unique_id>,

"firstName": "Jan",

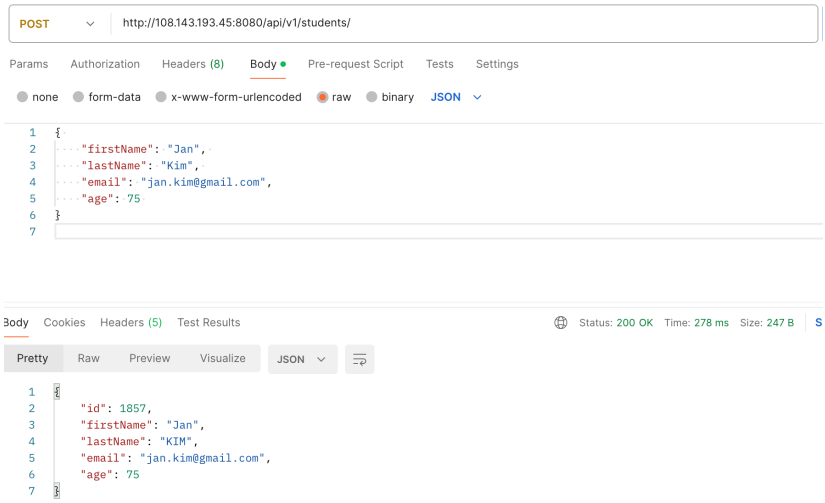
"lastName": "Kim",

"email": "jan.kim@gmail.com",

"age": 75

}

Actual Results:

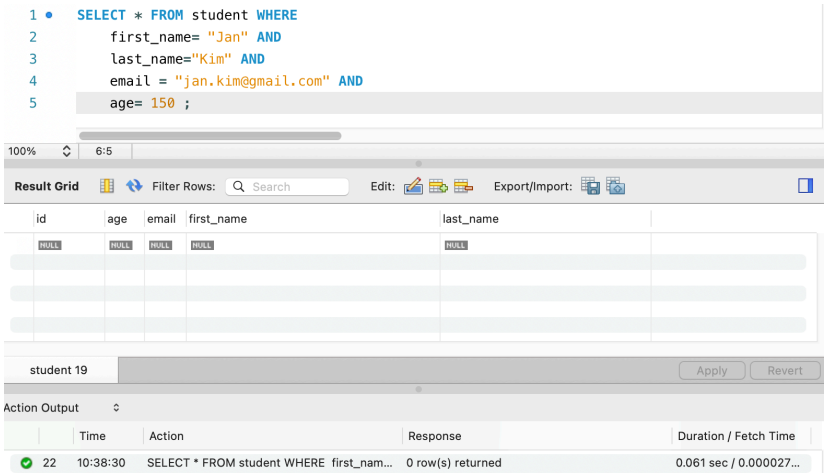


Test results: FAILED see notes

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:



Expecting results:

HTTP Status code 201

Format: JSON

Responds body contains:

{

"id": <unique_id>,

"firstName": "Jan",

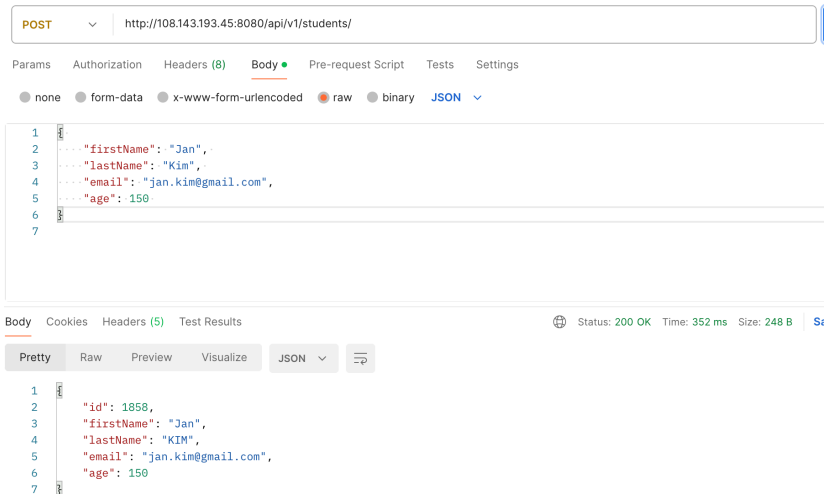
"lastName": "Kim",

"email": "jan.kim@gmail.com",

"age": 150

}

Actual Results:

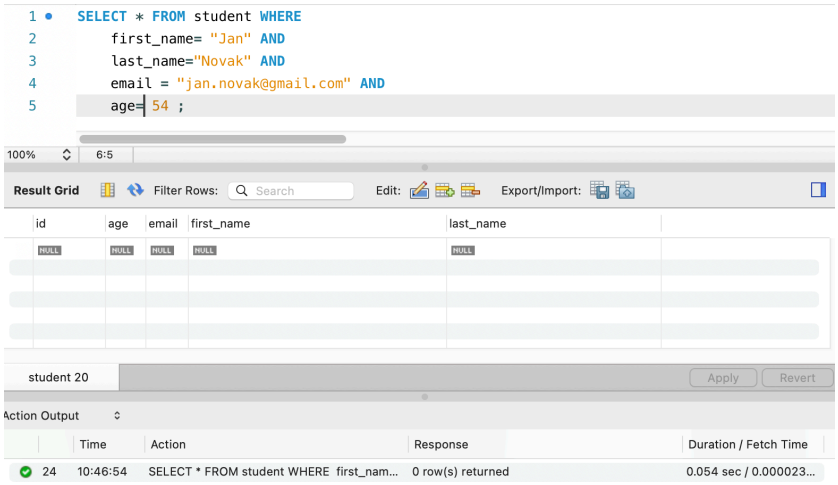


Test results: FAILED see notes

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:



Expecting results:

HTTP Status code 201

Format: JSON

Responds body contains:

{

"id": <unique_id>,

"firstName": "Jan",

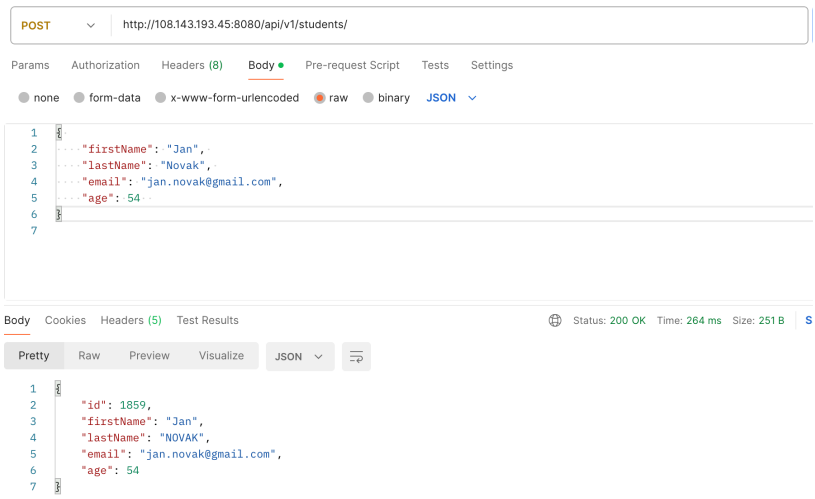
"lastName": "Novak",

"email": "jan.novak@gmail.com",

"age": 54

}

Actual Results:

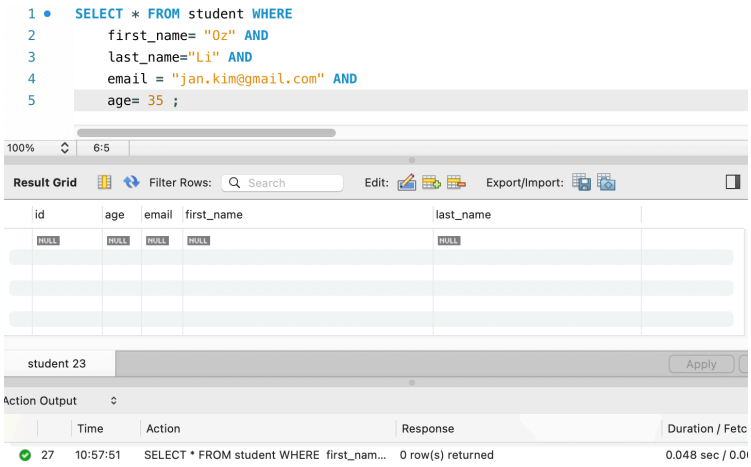


Test results: FAILED see notes

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:

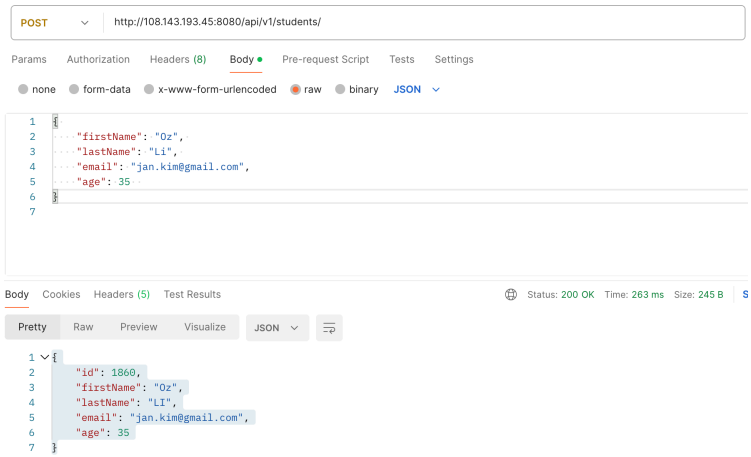


Expecting results:

HTTP Status code 400

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

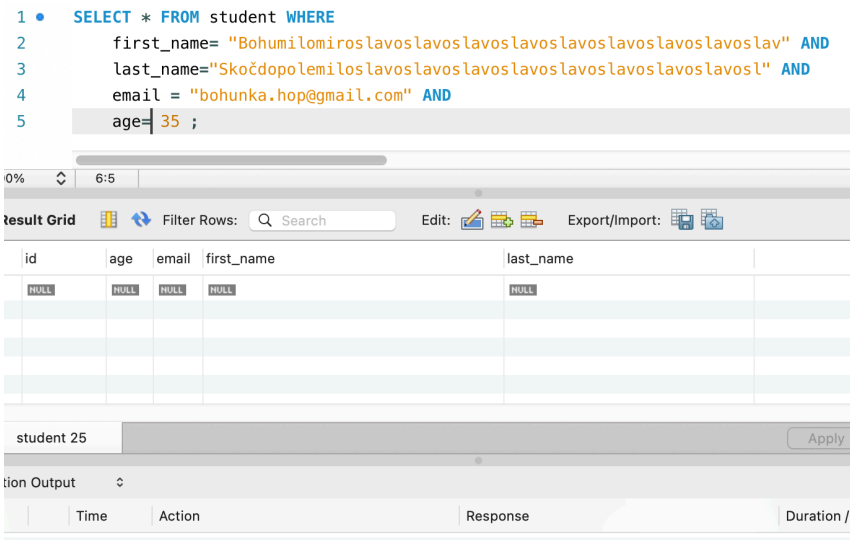
Actual Results:



Test results: FAILD

Test case ID: 20024011

Data preparation:

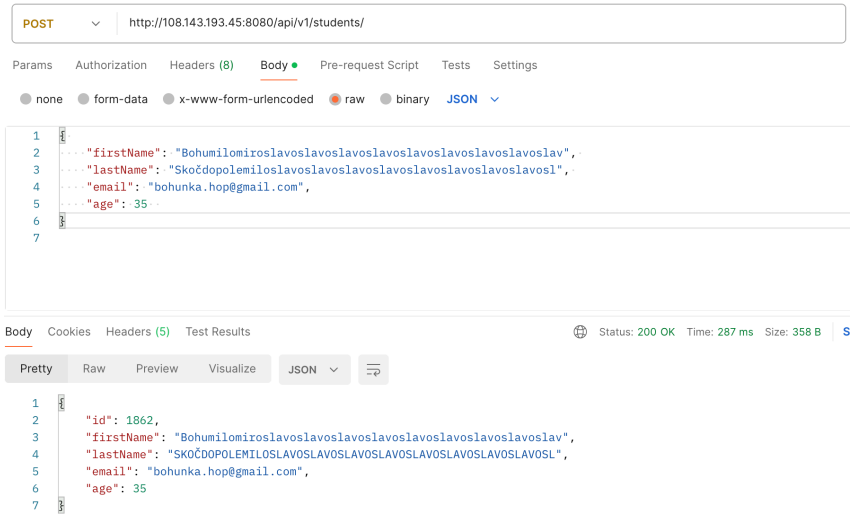


Expecting results:

HTTP Status code 400

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

Actual Results:

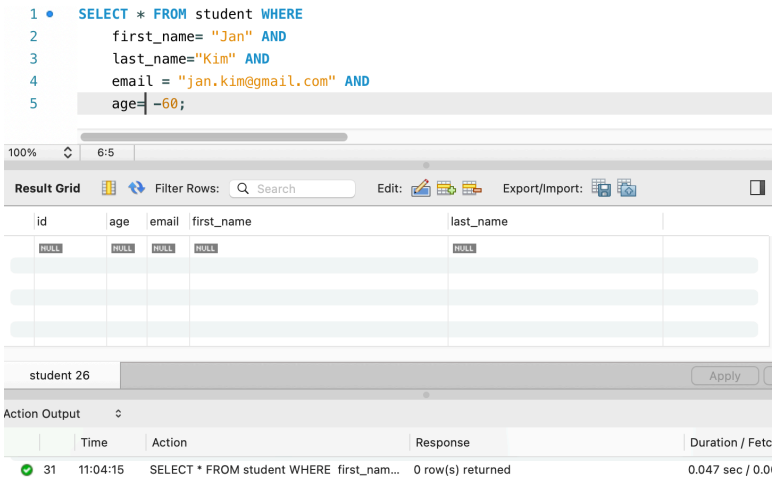


Test results: FAILED see notes

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:

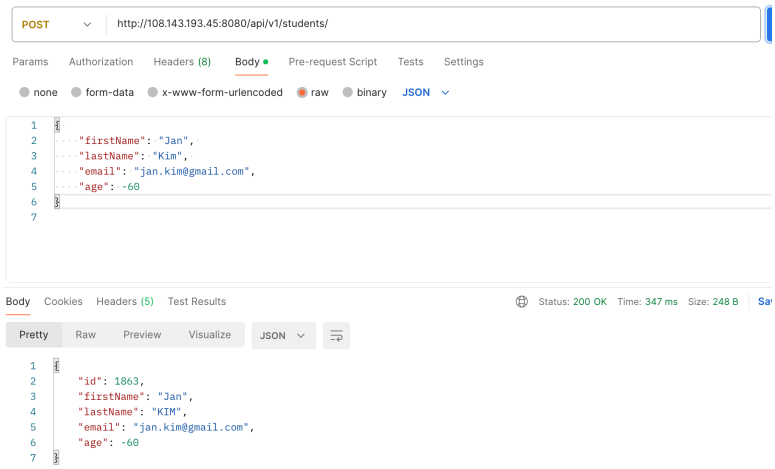


Expecting results:

HTTP Status code 400

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

Actual Results:

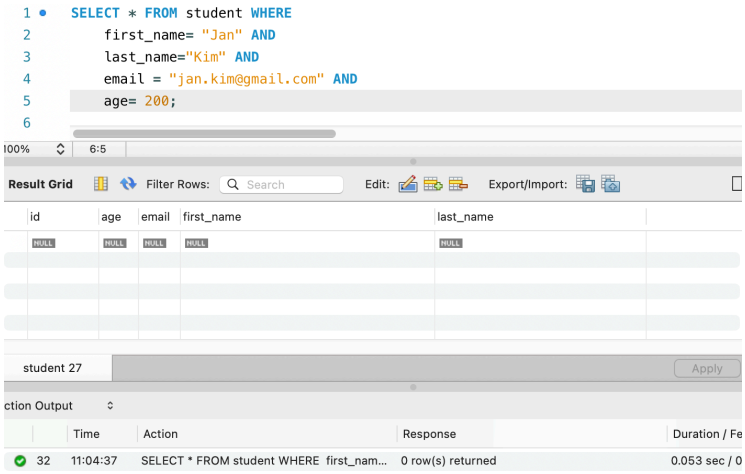


Test results: FAILED see notes

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

Test case ID: 20024013

Data preparation:

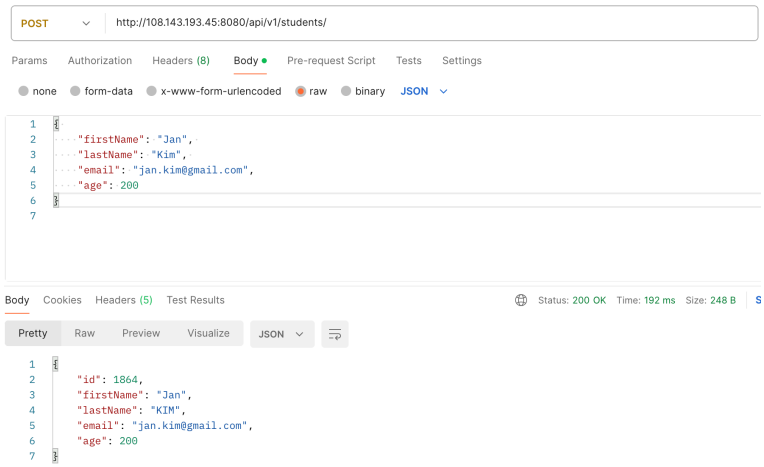


Expecting results:

HTTP Status code 400

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

Actual Results:

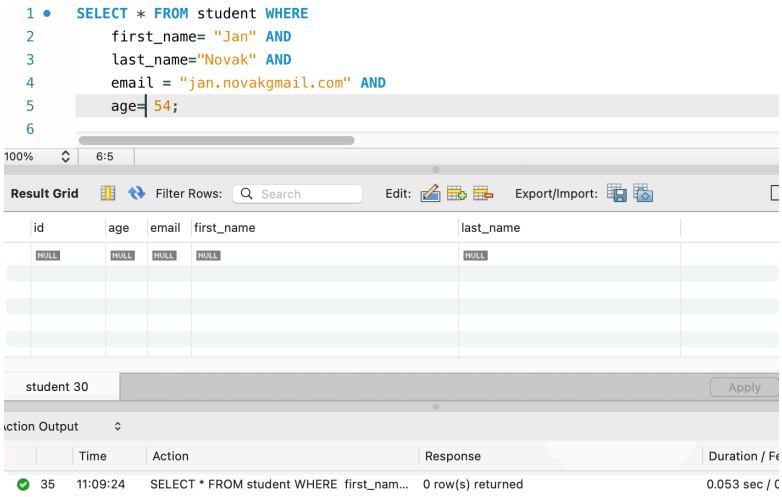


Test results: FAILED see notes

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

Test case ID: 20024014

Data preparation:

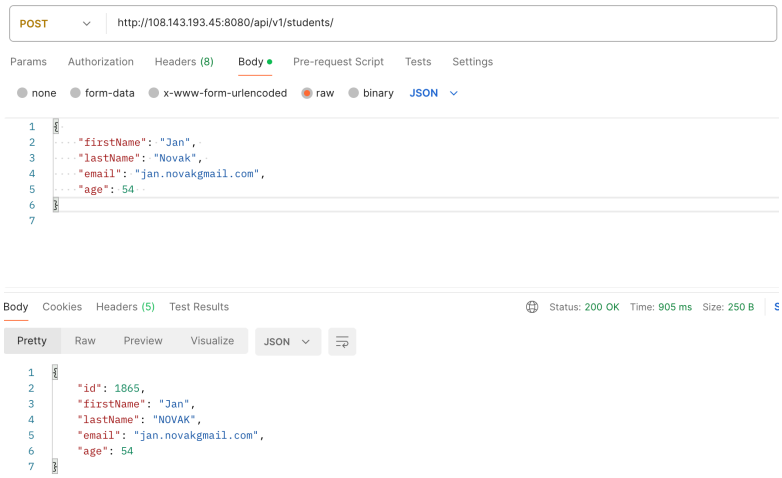


Expecting result:

HTTP Status code 400

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

Actual results:

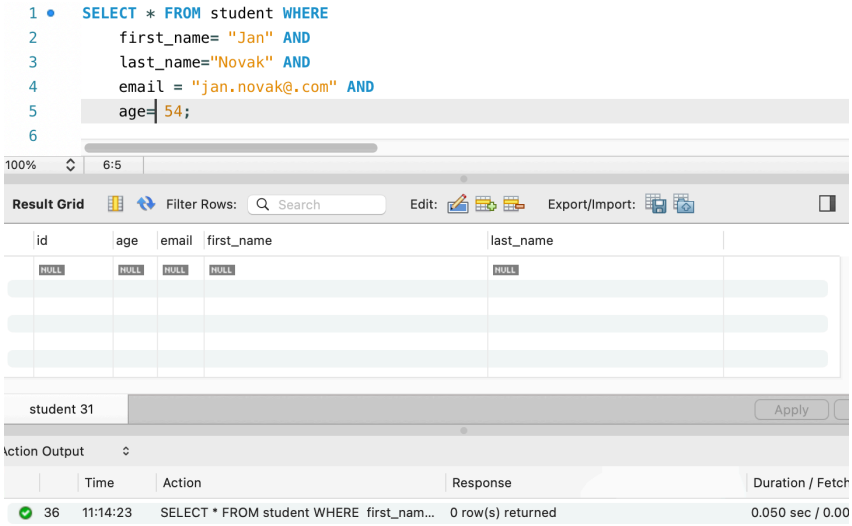


Test results: FAILED see notes

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

Test case ID: 20024015

Data preparation:

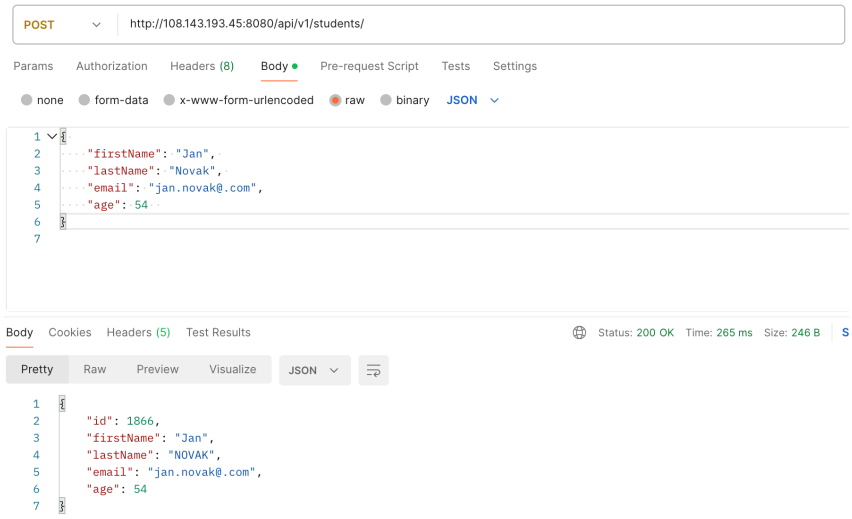


Expecting result:

HTTP Status code 400

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

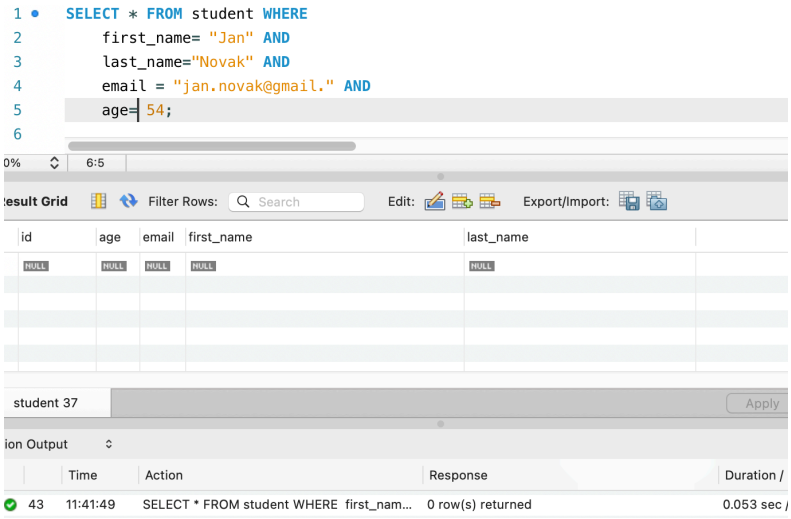
Actual results:



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.

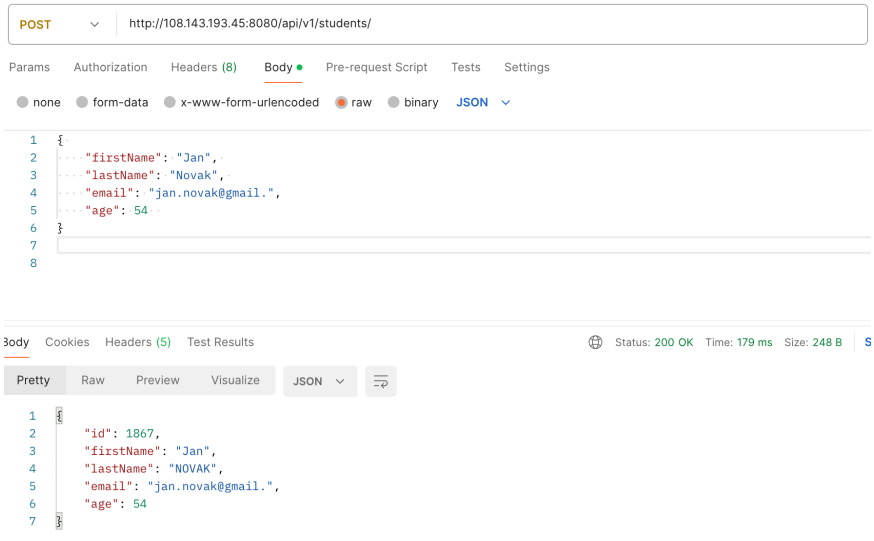
Test case ID: 20024016
Data preparation:



Expecting result:

HTTP Status code 400
The response body may contain a message about the incorrect data.

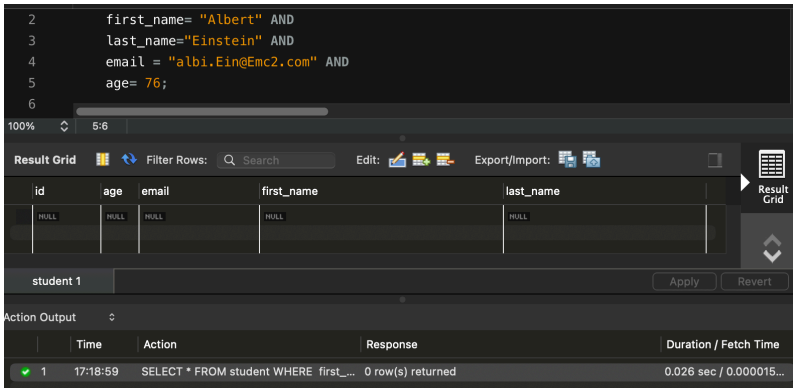
Actual results:



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: 20024017

Data preparation:



Expecting result:

First inserting:

HTTP status code 201

{

"id": <unique_id>,

"firstName": "Albert",

"lastName": "Einstein",

"email": " ",

"age": 76

}

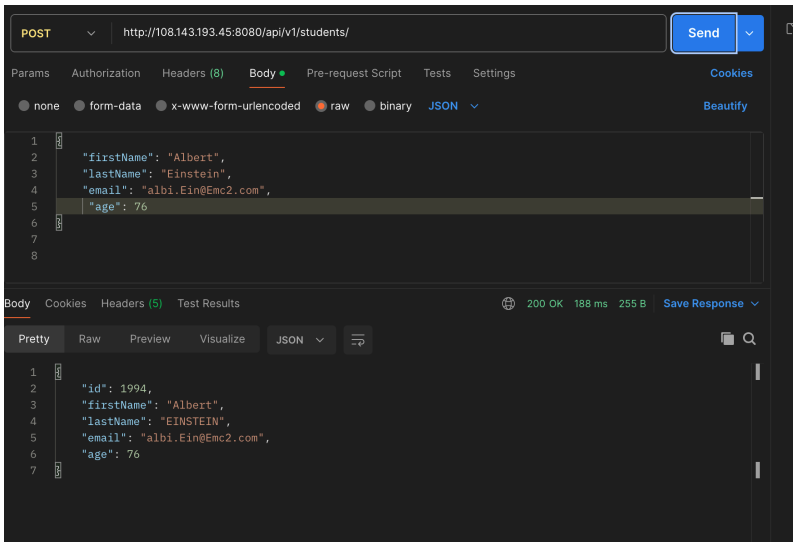
Second (duplicated) inserting:

HTTP Status code 409

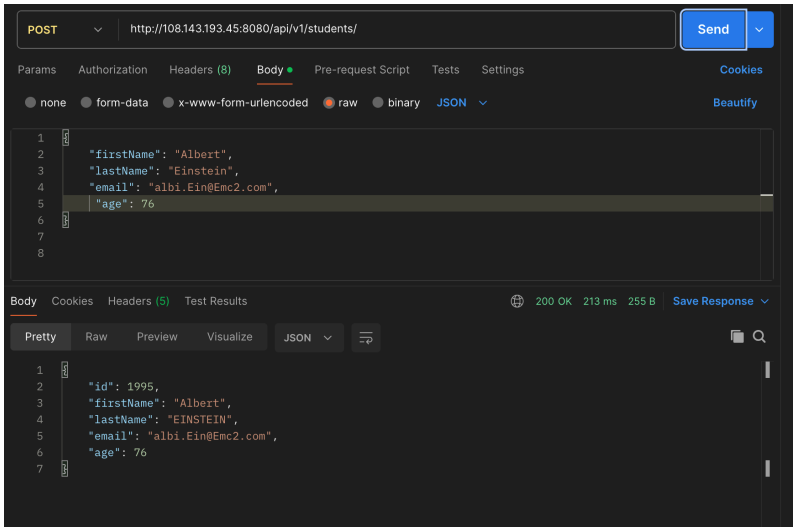
The response body may contain a message about the already existing data.

Actual results:

First inserting:



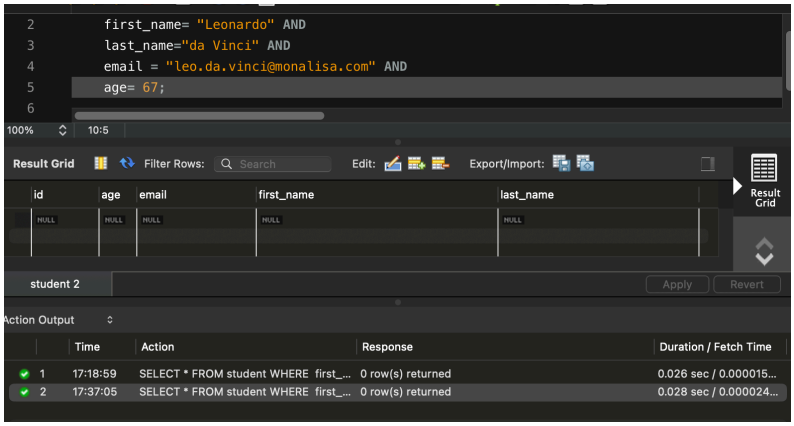
Second (duplicated) inserting:



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:

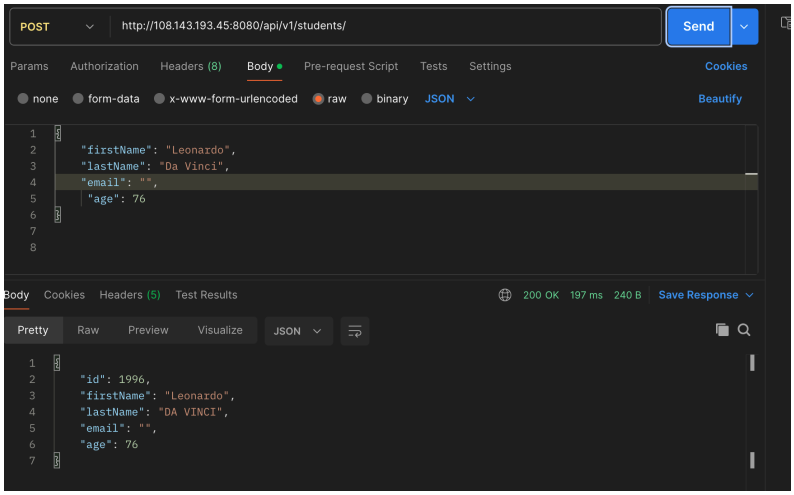


Expecting result:

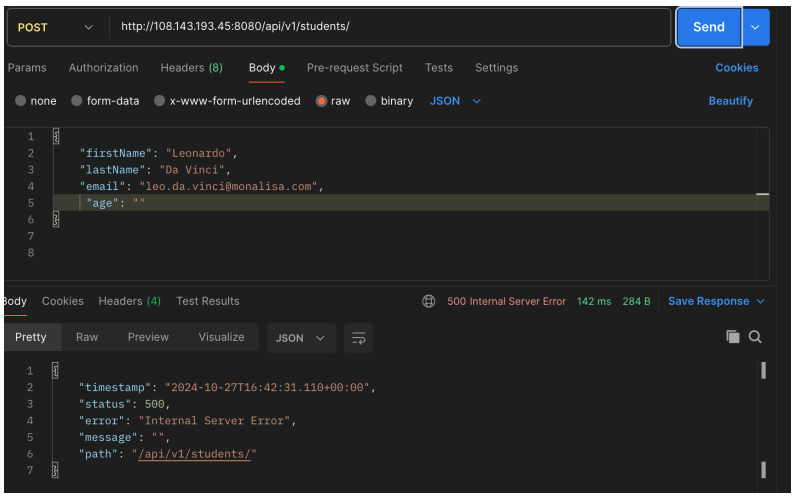
HTTP Status code 400
The response body may contain a message about the incorrect data.

Actual results:

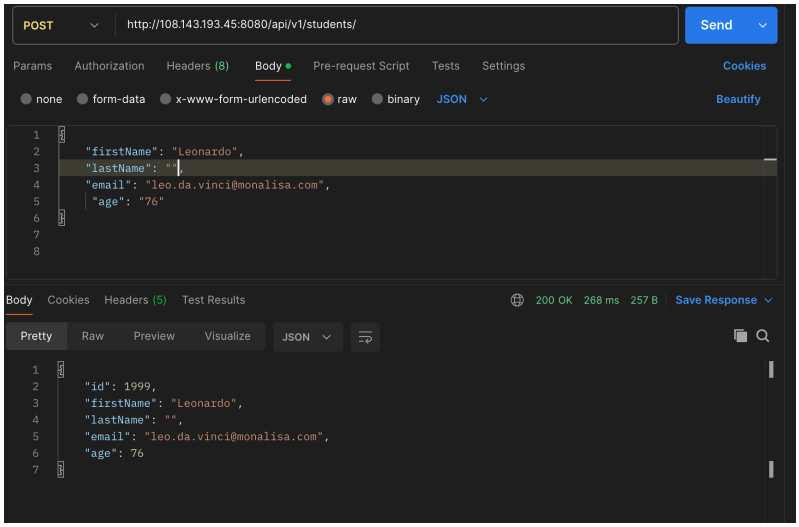
20024018/1



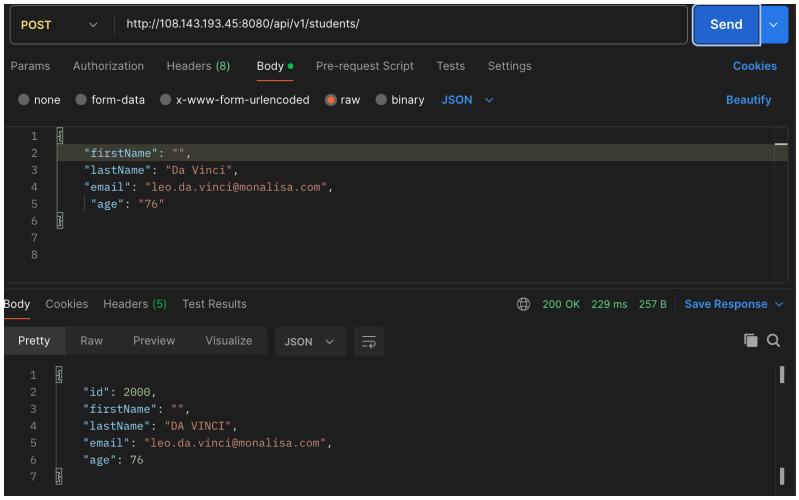
20024018/2



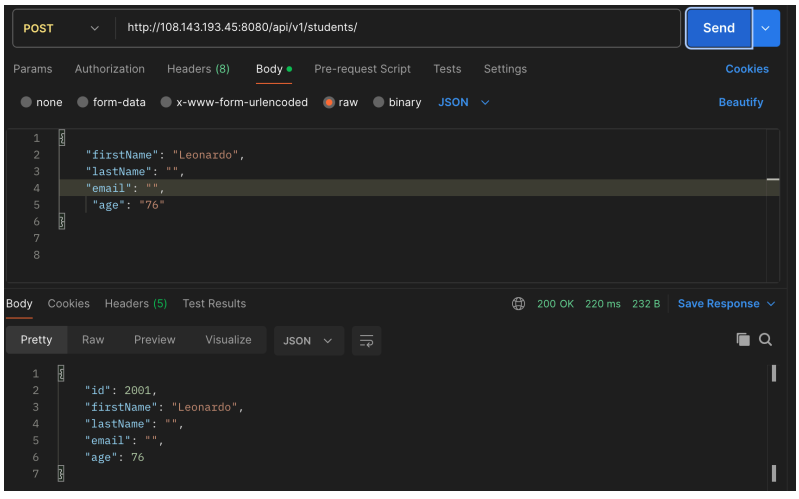
20024018/3



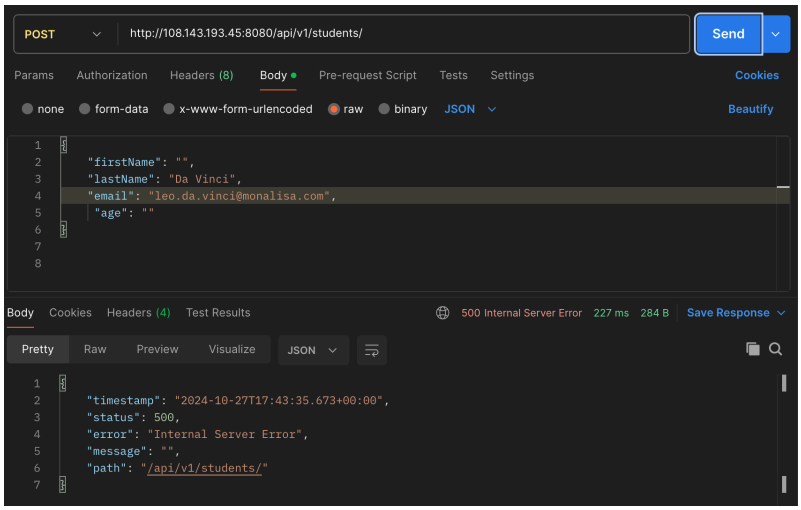
20024018/4



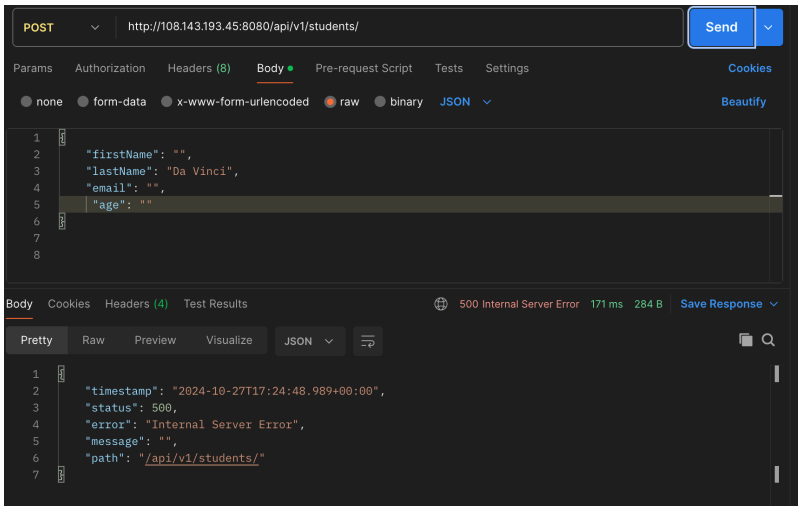
20024018/5



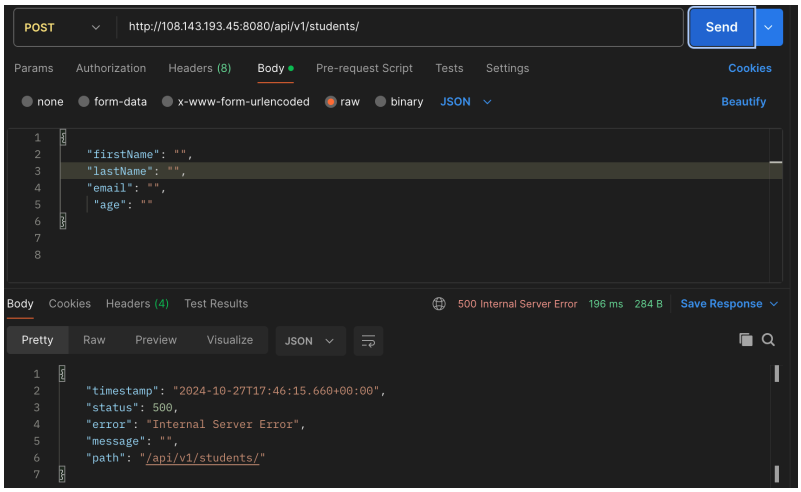
20024018/6



20024018/7



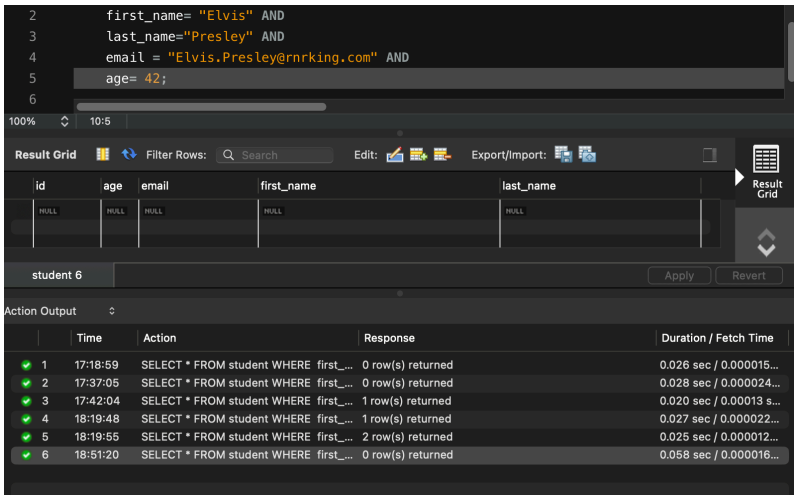
20024018/8



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

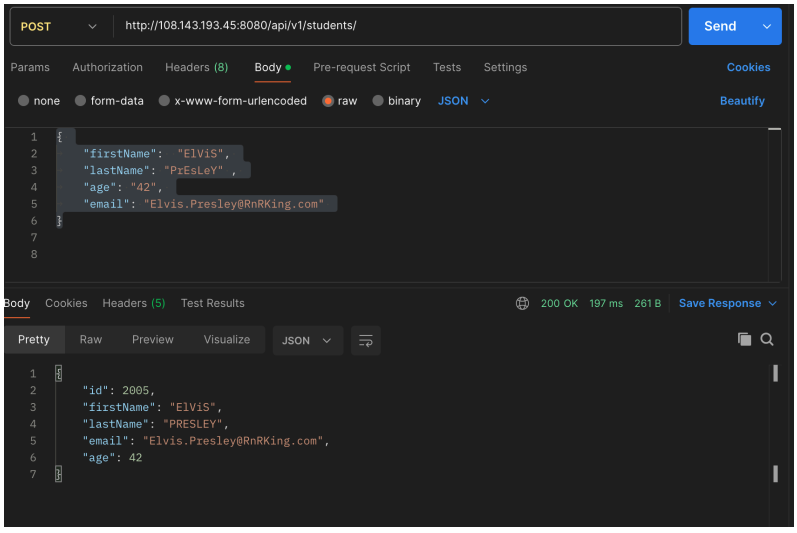
Data preparation:



Expecting result:

HTTP Status code 201
Format: JSON
Responds body contains:
{
"firstName": Elvis“,
"lastName": “Presley“,
"age": 42,
"email": „elvis.presley@rnrking.com“
}

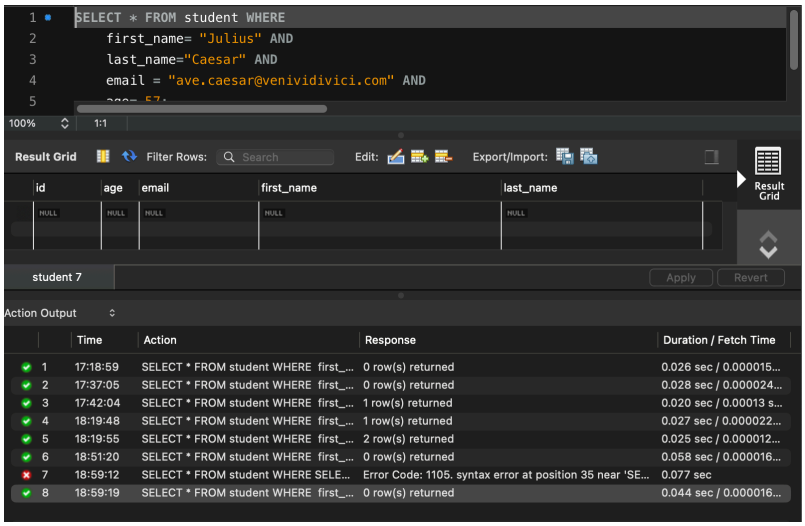
Actual results:



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

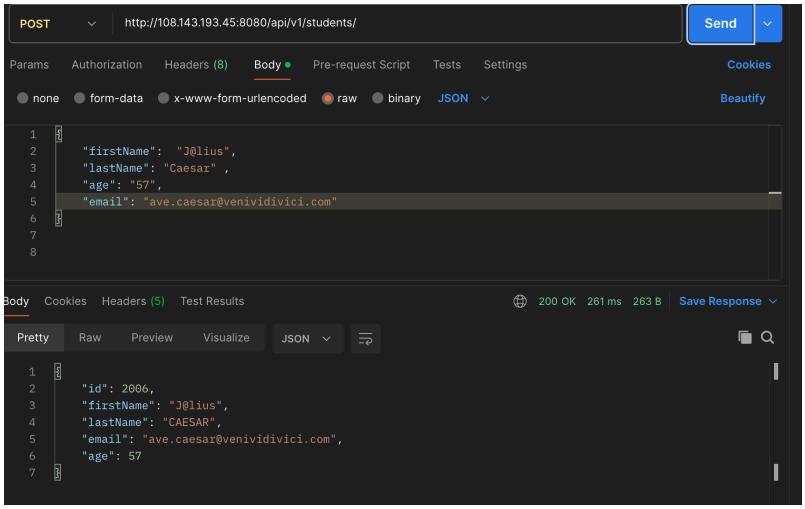
Data preparation:



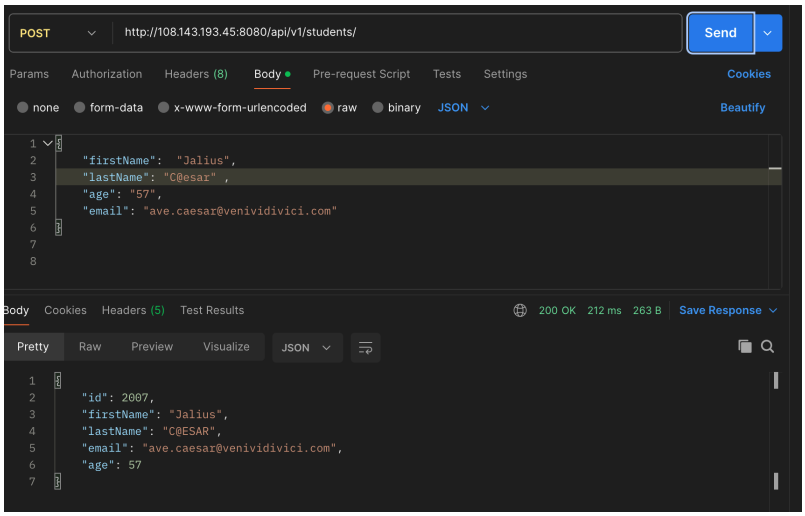
Expecting result:

HTTP Status code 400
The response body may contain a message about the incorrect data.

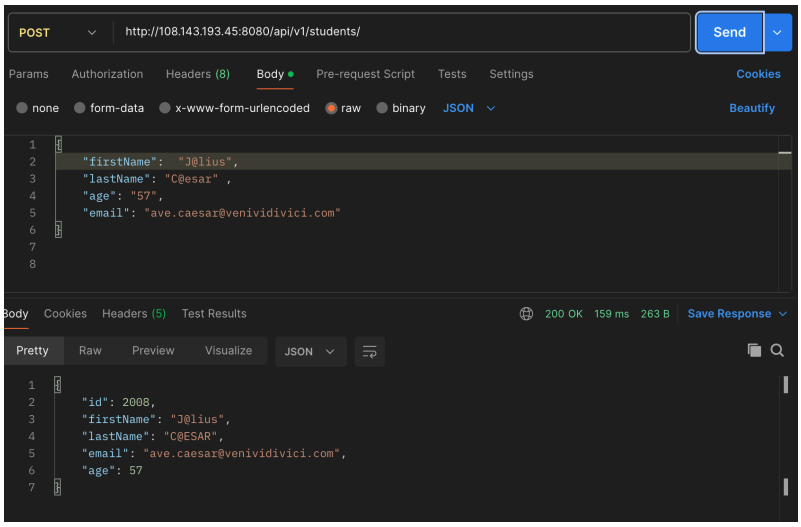
Actual results:
2002420/1



2002420/2



2002420/3

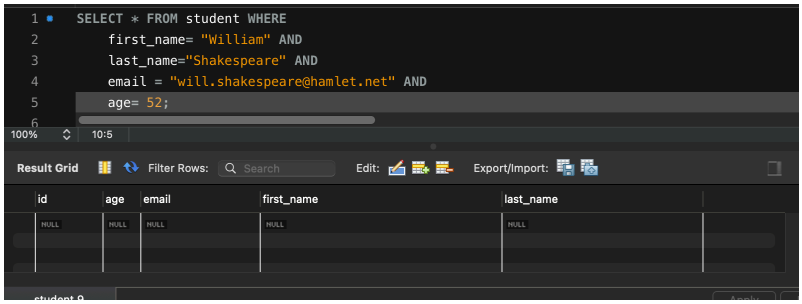


Test results:

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

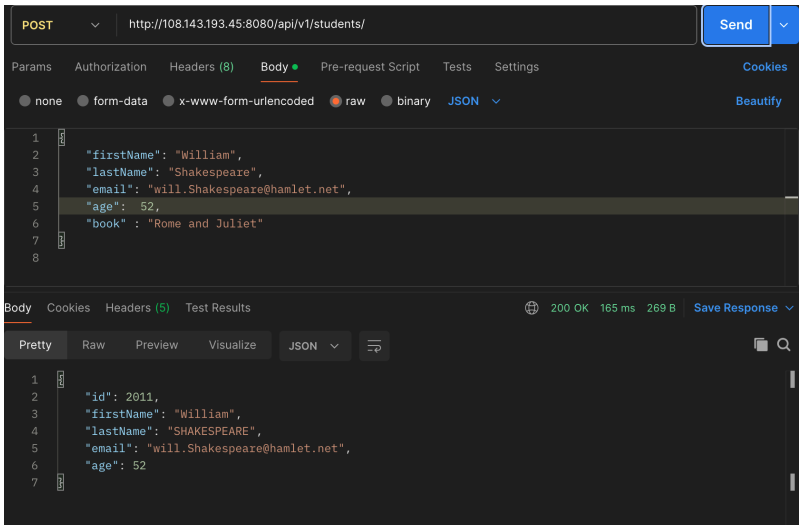
Data preparation:



Expecting result:

HTTP Status code 400
The response body may contain a message about the unexpected data.

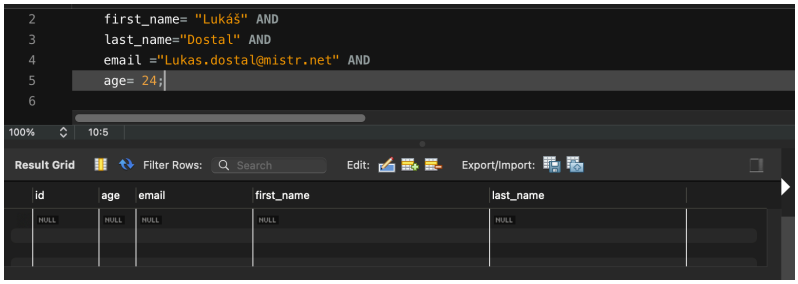
Actual result:



Test result:

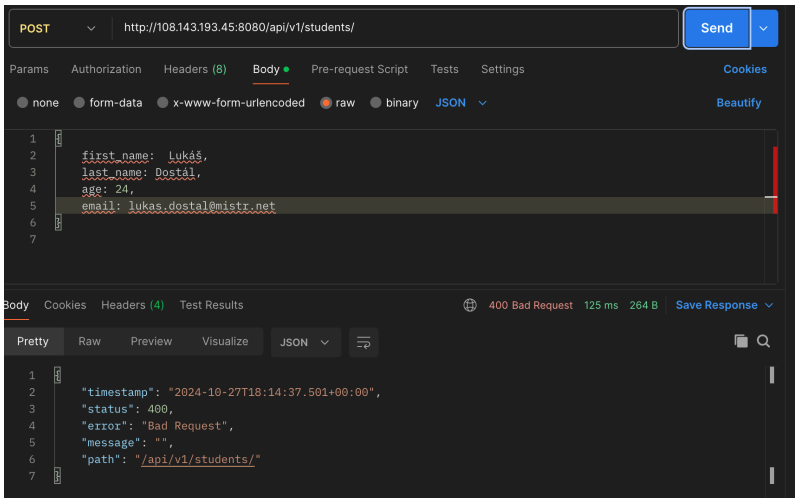
Test results: FAILED see notes
Notes: Test failed due to the known bug (report No, B0042024 and B0012024). Need to retest once the bug is solved.

Test case ID: 20024022
Data preparation:



Expecting result:
HTTP Status code 400
The response body may contain a message about the invalid format.

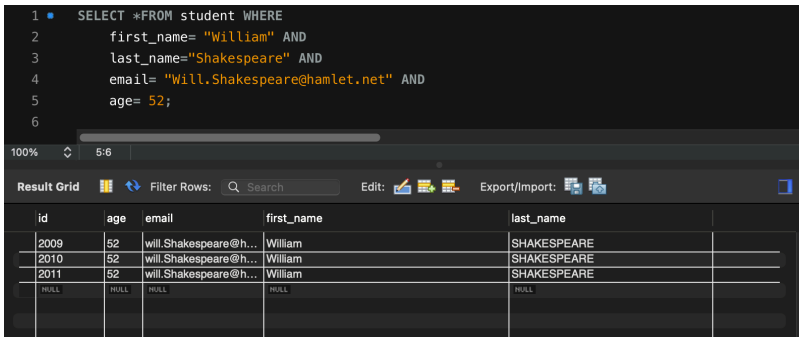
Actual result:



Test result: PASSED

Test case ID: 20024023

Data preparation:

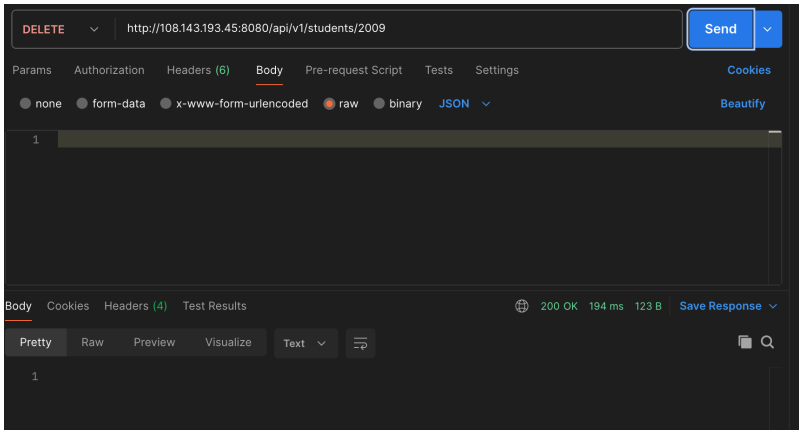


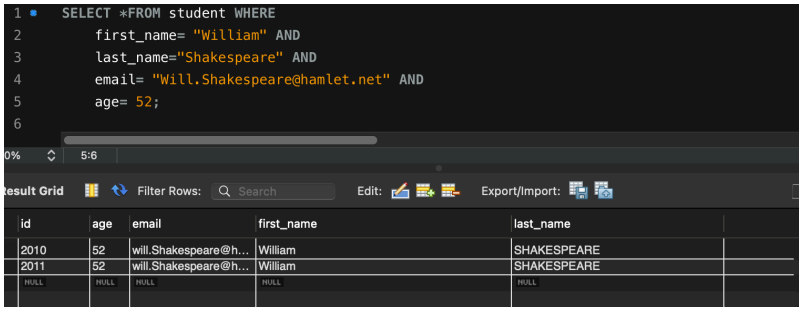
ID number: 2009

Expecting result:

HTTP Status code 200
The response body may contain a message about successfully student record delete.

Actual result:

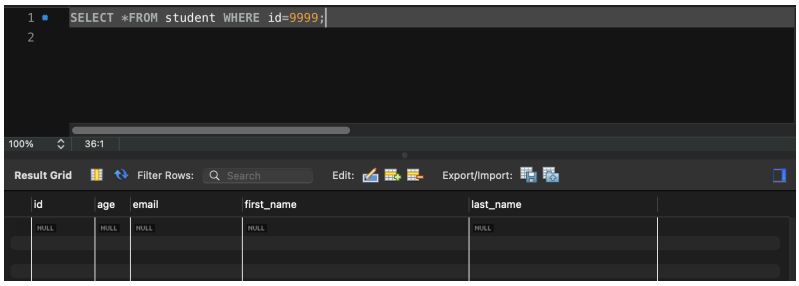




Test result: PASSED

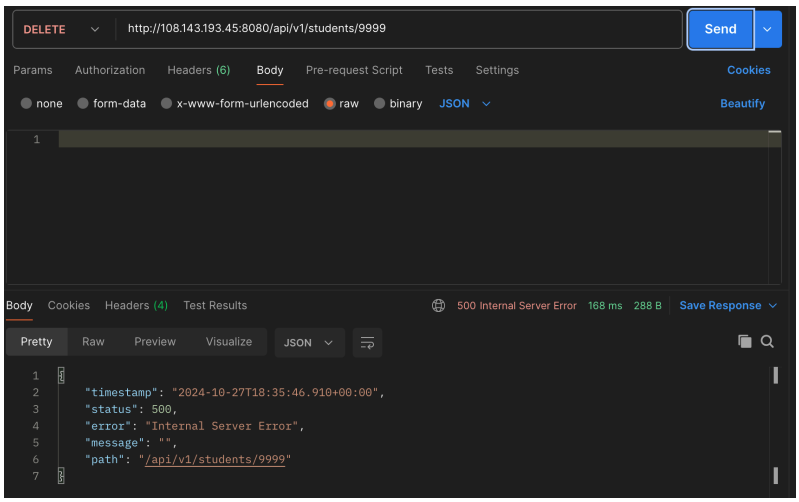
Test case ID: 20024024

Data preparation:



Expecting result:
HTTP status code 404
The response body may contain a message about the non existing student.

Actual result:



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

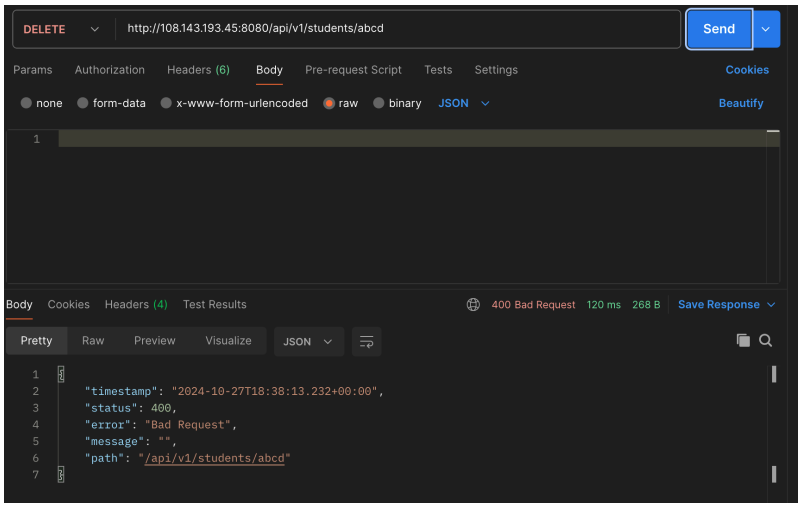
Test case ID: 20024025

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:



Test results: PASSED

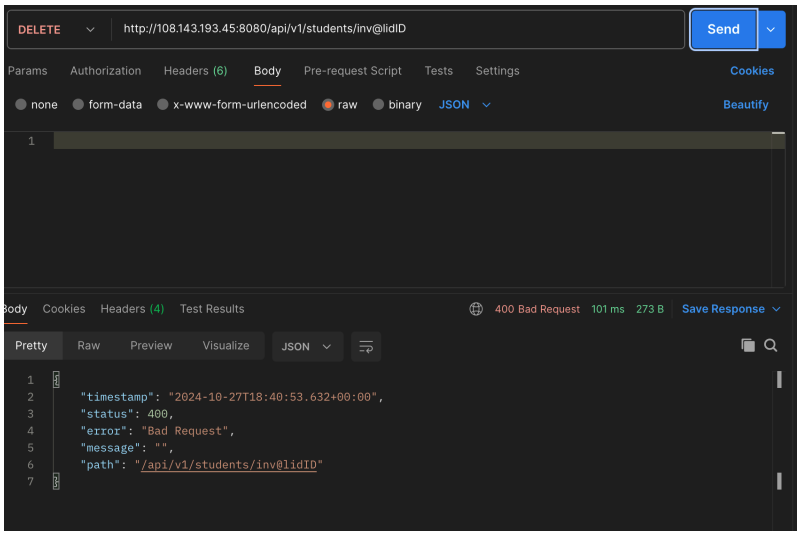
Test case ID: 20024026

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:



Test results: PASSED

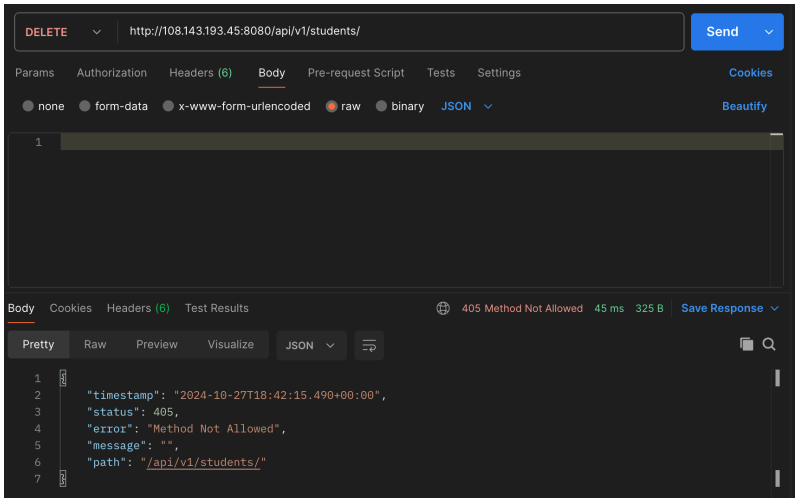
Test case ID: 20024027

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:



Test results: PASSED

BUG REPORT

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

Bug ID	Title	Test Case ID	Priorty	Status	Comments
B0012024	Last name returned in all uppercase letters instead of the correct format	2024001-2024012	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.	2024003-2024008	Lower	In progress	
B0042024	API returns 200 OK instead of 400 for creating invalid record	2024010 ; 2024011;2024012 ;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
Enviroment:
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}

Attachement: 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.
Enviroment:
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

Attachement: 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.
Enviroment:
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

Attachement: 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.
Enviroment:

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

Attachement: 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 ofduplication) API returns 200 OK instead of 409 for creating duplicated record.
Enviroment:
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)

Expecitng results:
HTTP Status code 409
The response body may contain a message about the already existing data.

Actual results:
HTTP Status code 200 OK

Attachement: 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.

Enviroment:

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

Expecitng 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

Attachement: Screenshot of the response in Postman.