FINAL PROJECT
n.1

Author: Daniel Pecka
Date: 10.11.2024

LIST OF CONTAINS

PROJECT REQUIREMENTS

The goal of the final project is to test the functionality of the application, which is used to manipulate data about students. The application has a REST-API interface that allows creating, deleting and retrieving data..

Access credentials:

Database	database: qa_demo Host: aws.connect.psdb.cloud Username: k9ik3tg4x7hdxk68nlqy Password: pscale_pw_8Nc3bfP1t3ECEJm8cGPze47MegBUWQzgH7J8XsLghBi
REST-API	http://108.143.193.45:8080/api/v1/students/

Comments:

Remember that in IT, data has to be stored somewhere and then retrieved. Therefore, verify that the data is correctly stored and retrieved from the database.
Don't forget to include test data, expected results including response body and status codes in test scenarios.

TEST CASES (TC)

I have tested the functionality of the application based on the listed test cases.

TC 1: Retrieve Student Data Using Path Parameter (Standard RESTful)

Title: Verify GET request retrieves student data correctly using path parameter.

Precondition: Student with ID 564 exists in the database.

Steps:

1. Open a workspace on Postman (https://web.postman.co/).
2. Insert a GET request http://108.143.193.45:8080/api/v1/students/564 and send it.
3. Verify the response status code.
4. Open MySQL Workbench.
5. Insert the following SQL query:

```
SELECT *
FROM student
WHERE id = 564
;
```

6. Compare the output of the GET request with output of the SQL query.

Expected Result:

- Status code: *200 OK*
- Postman response body:

```
{
   "id": 564,
   "firstName": "Petra",
   "lastName": "RADILOVA",
   "email": "radilova@yourmail.com",
   "age": 19
}
```

- Database output matches the Postman response.

TC 2: Retrieve Student Data Using Query Parameter (Potential Bug Check)

Title: Verify GET request retrieves data correctly using query parameter (id=564).

Precondition: Student with ID 564 exists in the database, while the parameter search being available is an intended design.

Steps:

1. Open a workspace on Postman (https://web.postman.co/).
2. Insert a GET request http://108.143.193.45:8080/api/v1/students?id=564 and send it.
3. Verify the response status code.
4. Open MySQL Workbench.
5. Insert the following SQL query:

```
SELECT *
FROM student
WHERE id = 564
;
```

6. Verify the response status.
7. Compare the output of the GET request with output of the SQL query.

Expected Result:

- Status code: *200 OK*
- Postman response body:


```
{
   "id": 564,
   "firstName": "Petra",
   "lastName": "RADILOVA",
   "email": "radilova@yourmail.com",
   "age": 19
}
```


- Database output matches the Postman response.

TC 3: Handle Invalid Student Data (GET Method for Non-Existent Student)

Title: Verify GET request for a non-existent student returns a 404 Not Found status using path parameter.

Precondition: No student exists with ID 9999. There is a pre-designed error message.

Steps:

1. Open a workspace on Postman (https://web.postman.co/).
2. Insert a GET request http://108.143.193.45:8080/api/v1/students/9999 and send it.
3. Verify the response status code and body.

Expected Result:

- Status code: 404 Not Found
- Response body:

```
{
  "error": "Student not found"
}
```

TC 4: Create a New Student Record (POST Method with Valid Data)

Title: Verify POST request creates a new student record successfully.

Precondition: Ensure a student with ID `1000` does not exist.

Steps:

1. Open a workspace on Postman (https://web.postman.co/).
2. Insert a POST request to http://108.143.193.45:8080/api/v1/students/.
3. Set the request body as:

```
{
  "id": 1000,
  "firstName": "Petr",
  "lastName": "Novák",
  "email": "petr.novak@example.com",
  "age": 30
}
```

4. Send the request.
5. Verify the response code and body.
6. Check the database to ensure the new record was added:

```
SELECT *
FROM student
WHERE id = 1000
;
```

Expected Result:

- Status code: *200 OK*
- Postman response body:

```
{
  "id": 1000,
  "firstName": "Petr",
```

```
  "lastName": "Novák",
  "email": "petr.novak@example.com",
  "age": 30

}
```


- Database output matches the Postman response.

TC 5: Create Student with Invalid Data (POST Method with Long Strings)

Title: Verify POST request fails with long string inputs.

Precondition: Ensure API is supposed to have validation for input data length according documentation. There is a pre-designed error message. No student exists with ID 1002.

Steps:

Steps:

1. Open a workspace on Postman (https://web.postman.co/).
2. Insert a POST request to http://108.143.193.45:8080/api/v1/students/.
3. Set the Pre-request script as:

```
const longString = 'A'.repeat(300); pm.environment.set("longString", longString);
```

4. Set the request body as:

```
{

"id": 1005,
"firstName": "{{longString}}",
"lastName": "{{longString}}",
"email": "mail@mail.com",
"age": 20

}
```

5. Send the request.
6. Verify the response code and body.
7. Check the database to ensure no record was added:

```
SELECT *
FROM student
WHERE id = 1002
;
```

Expected Result:

- Status code: *400 Bad Request*
- Postman response body:

```
{
"error": "Input data too long or invalid format"
}
```

- There is no record with ID 1002 in the database..

TC 6: Create Student with Invalid Data (POST Method with invalid email adress)

Title: Verify API handles invalid email format correctly.

Precondition: Ensure API should have validation for email input according to documentation. There is a pre-designed error message. No student exists with ID 1005.

Steps:

1. Open a workspace on Postman (https://web.postman.co/).
2. Insert a POST request to http://108.143.193.45:8080/api/v1/students/.
3. Set the request body as:

```
{

"id": 1005,
"firstName": "Petr",
"lastName": "Novák",
"email": "petrnovakexample.com",
"age": 30

}
```

4. Send the request.
5. Verify the response code and body.
6. Check the database to ensure no record was added:

```
SELECT *
FROM student
WHERE id = 1005
;
```

Expected Result:

- Status code: *400 Bad Request*
- Postman response body:

```
{
```

```
"error": "Invalid email format"

}
```

- There is no record with ID 1002 in the database..

TC 7: Delete Existing Student (DELETE Method)

Title: Verify DELETE request removes a student record.

Precondition: Student exists with ID 2241 in the database.

Steps:

1. Open a workspace on Postman (https://web.postman.co/).
2. Insert a POST request to http://108.143.193.45:8080/api/v1/students/2241.
3. Send the request.
4. Verify the response code.
5. Check the database to ensure the record was deleted:

```
SELECT *
FROM student
WHERE id = 2241
;
```

Expected Result:

- Status code: *200 OK*
- There is no record with ID 2241 in the database..

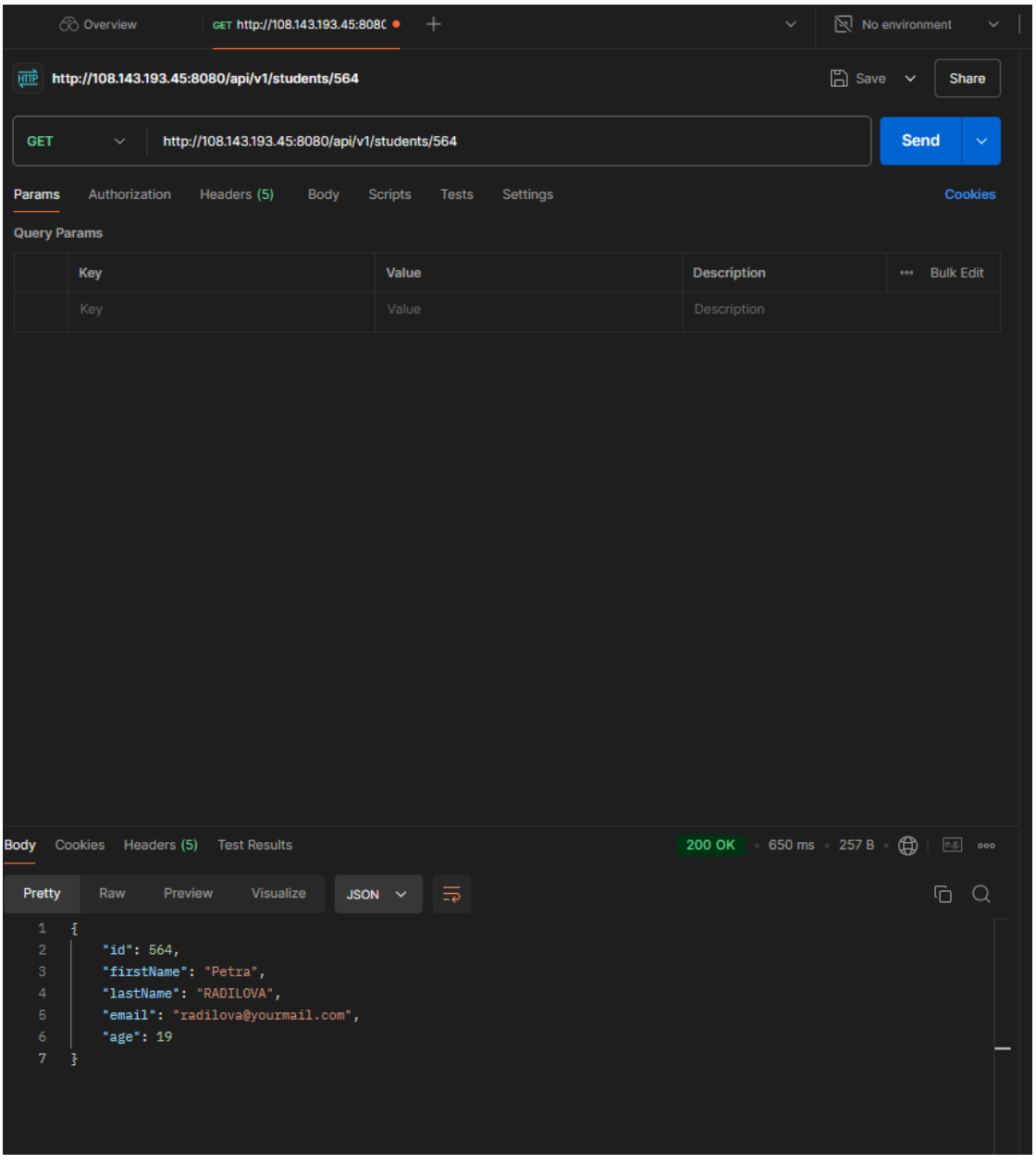
TESTS EXECUTION

I have completed the test cases, I list the test results in this section.

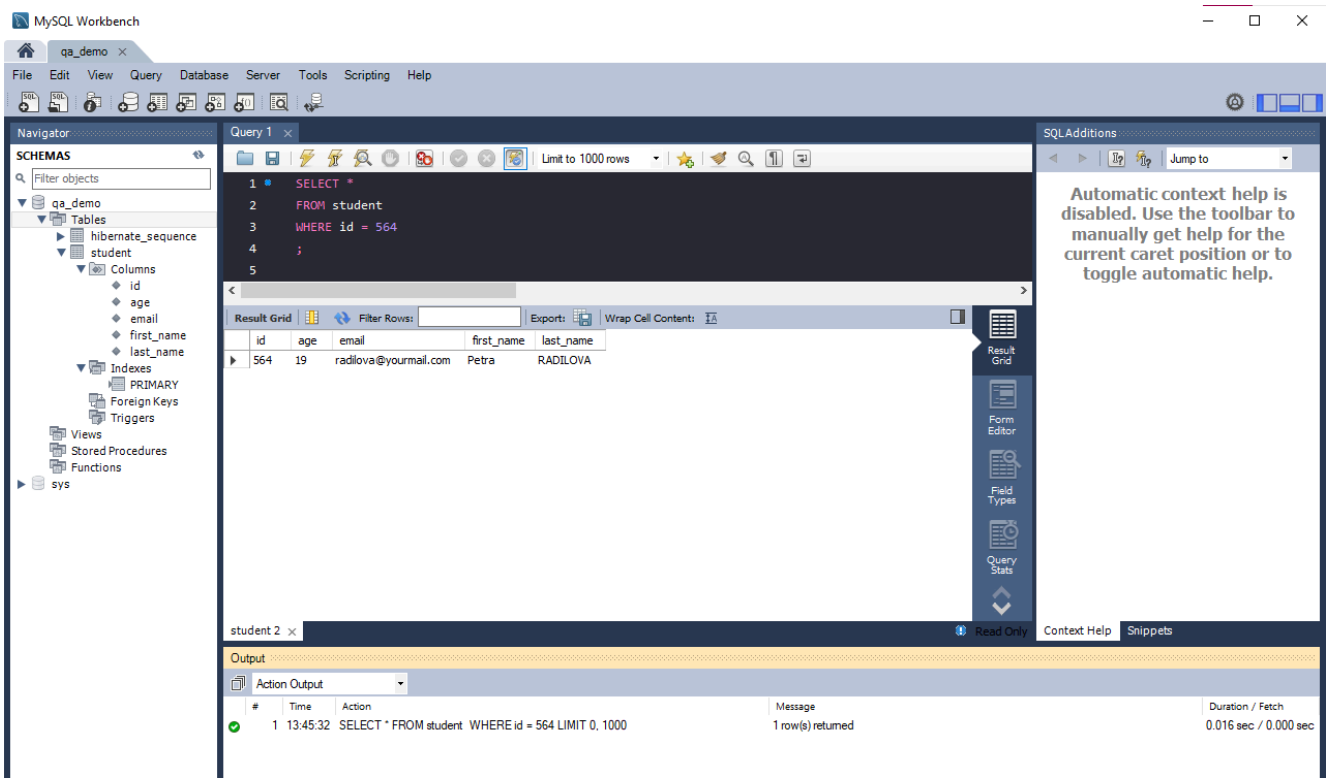
TC 1: Retrieve Student Data Using Path Parameter (Standard RESTful)

Following the scenario in the previous section I have achieved the results in the supplements below:

Supplement 1.1: Postman results



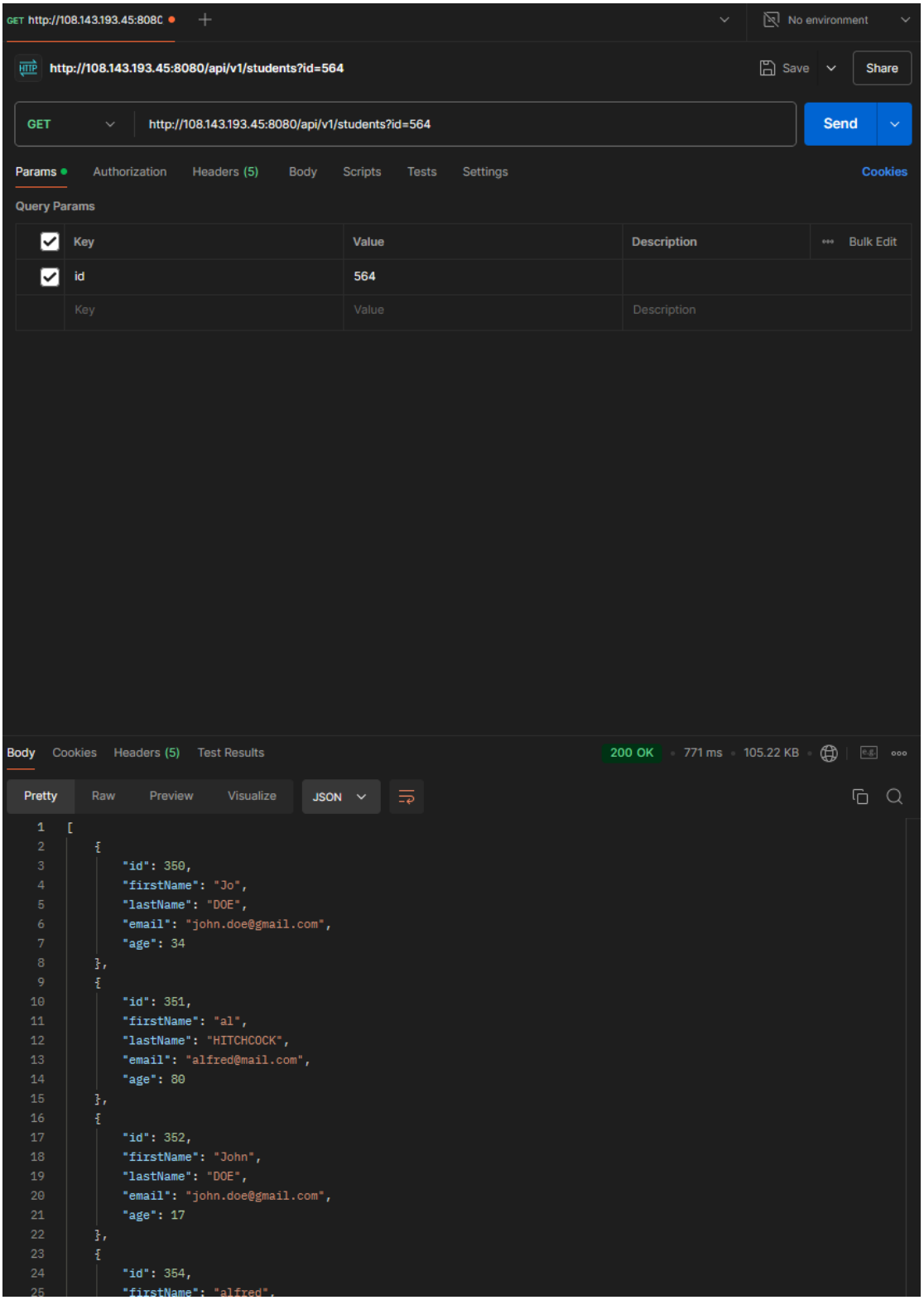
Supplement 1.2: Database results



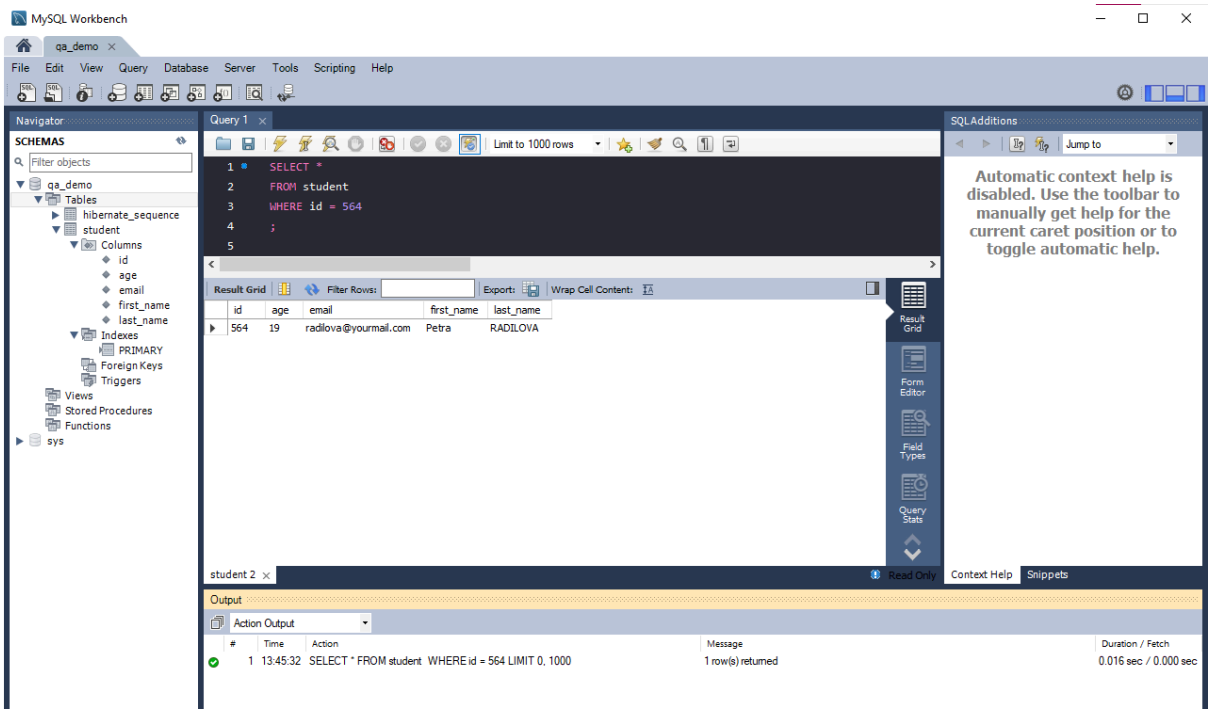
TC 2: Retrieve Student Data Using Query Parameter (Potential Bug Check)

Following the scenario in the previous section I have achieved the results in the supplements below:

Supplement 2.1: Postman results



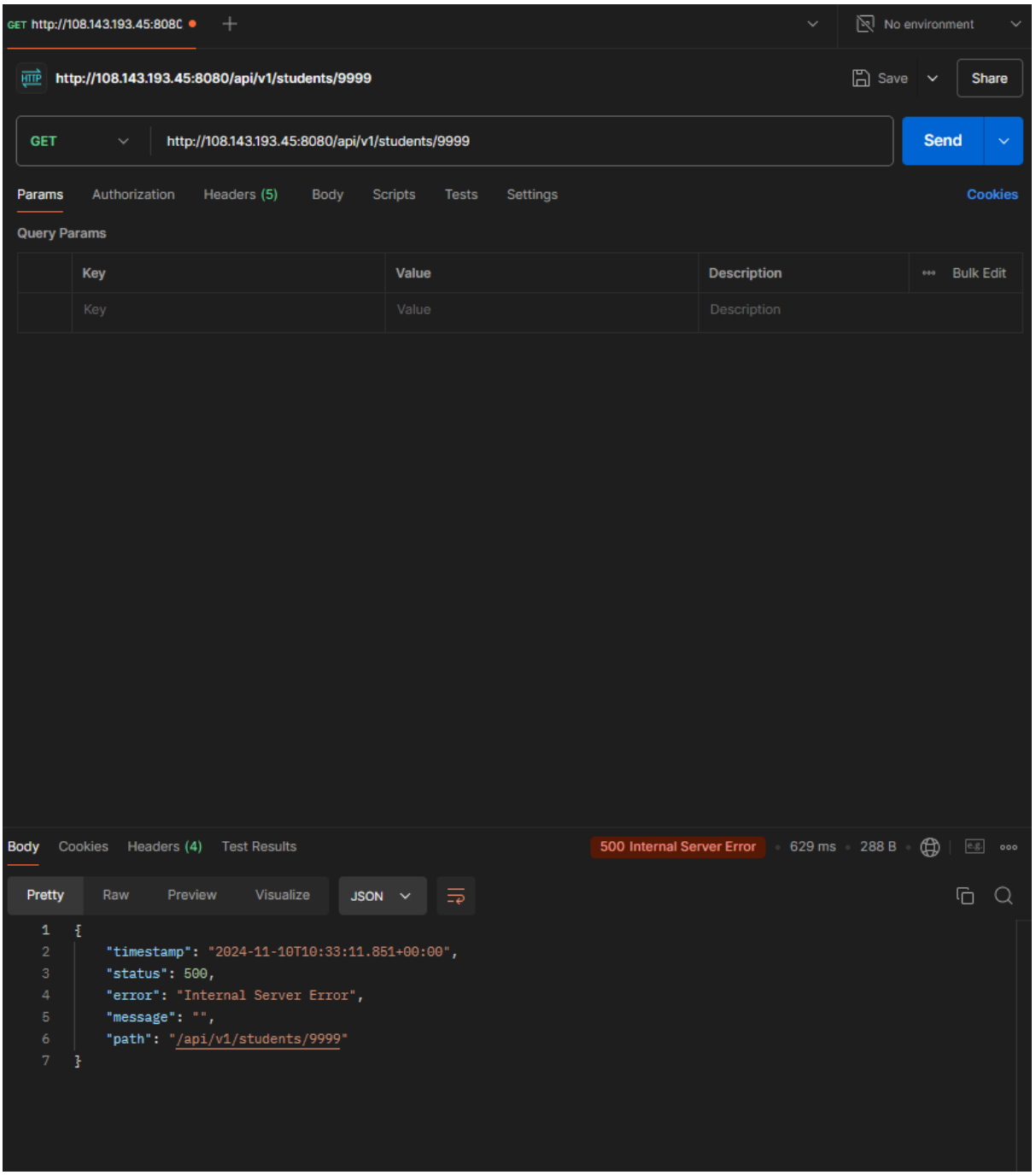
Supplement 2.2: Database results



TC 3: Handle Invalid Student Data (GET Method for Non-Existent Student)

Following the scenario in the previous section I have achieved the result in the supplement below:

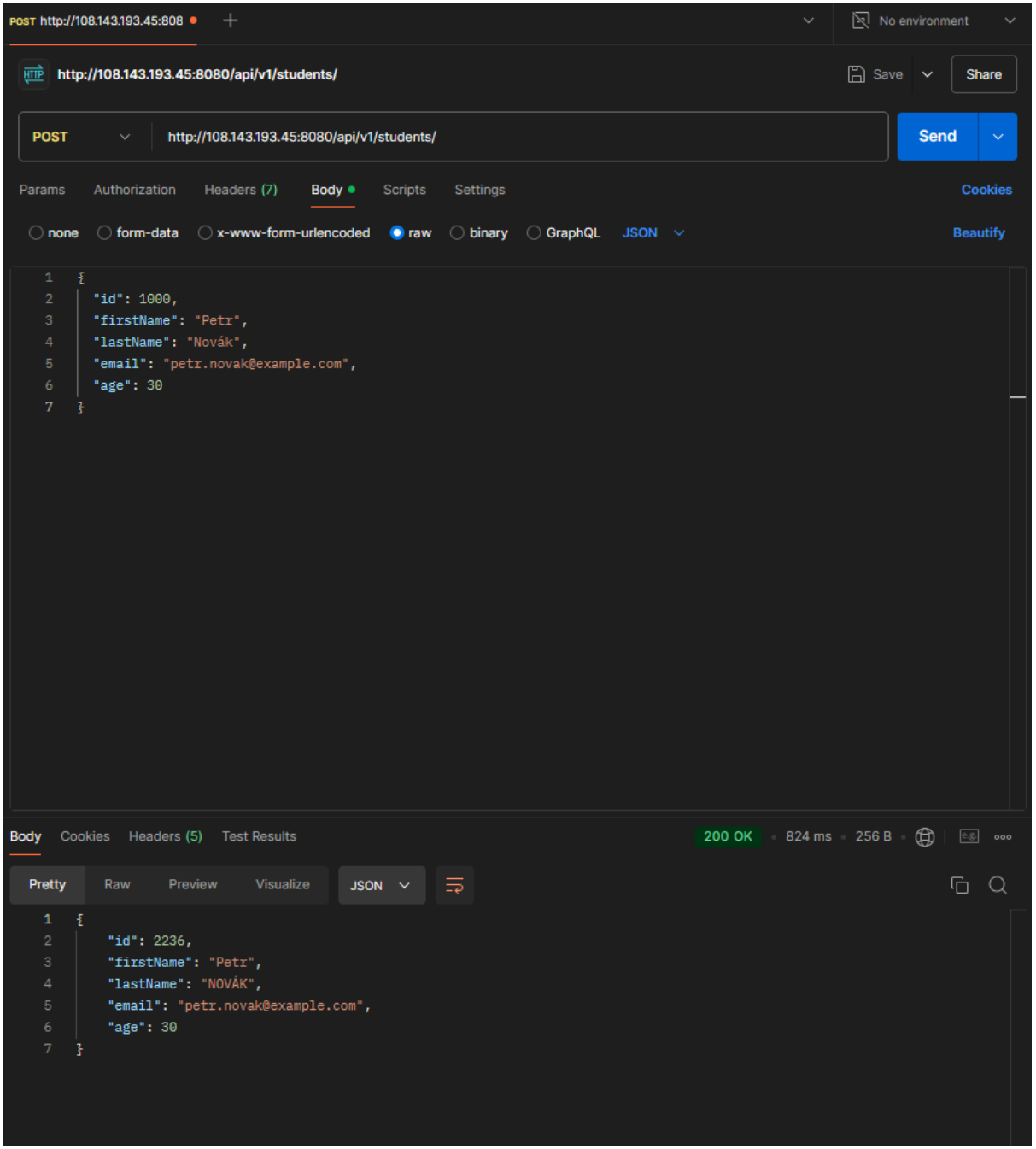
Supplement 3.1: Postman results



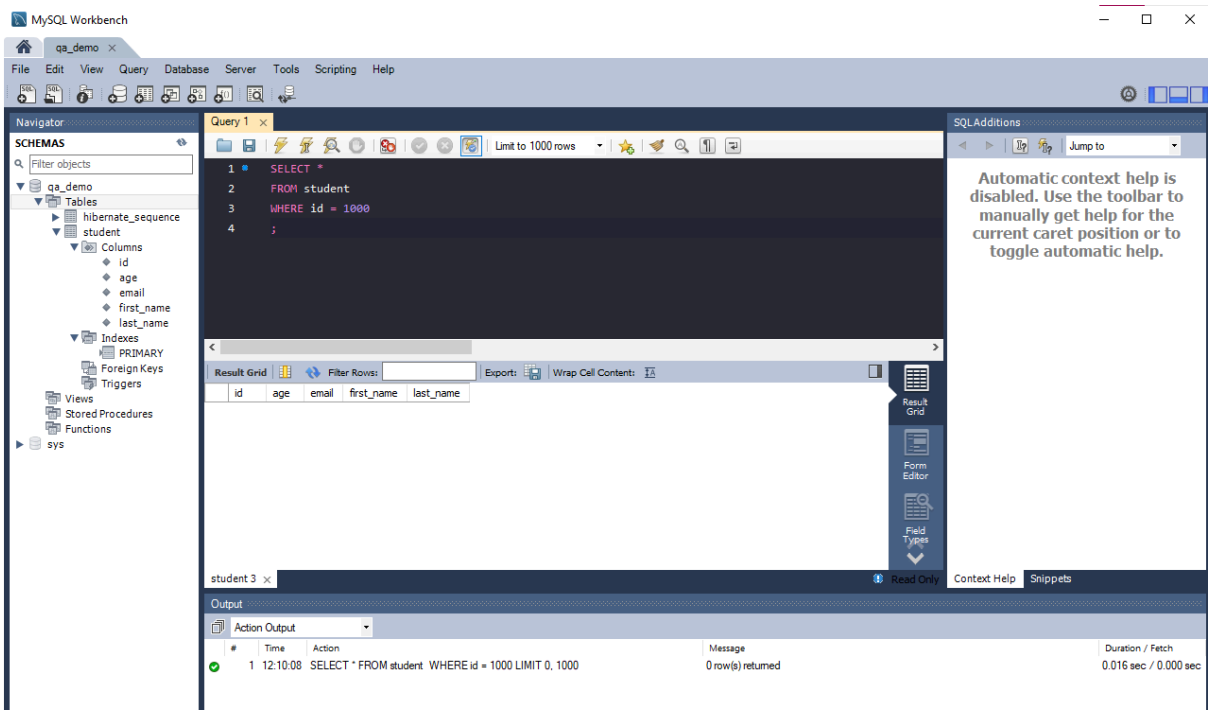
TC 4: Create a New Student Record (POST Method with Valid Data)

Following the scenario in the previous section I have achieved the result in the supplement below:

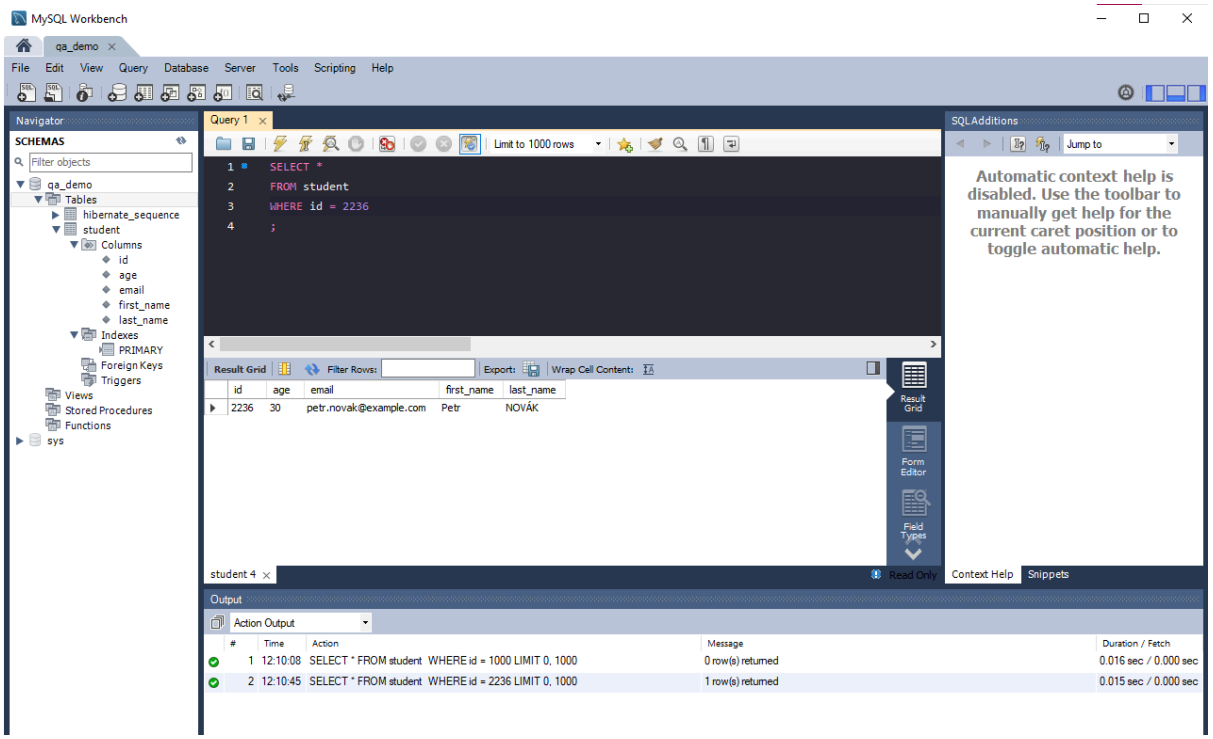
Supplement 4.1: Postman results



Supplement 4.2: Database result for id = 1000



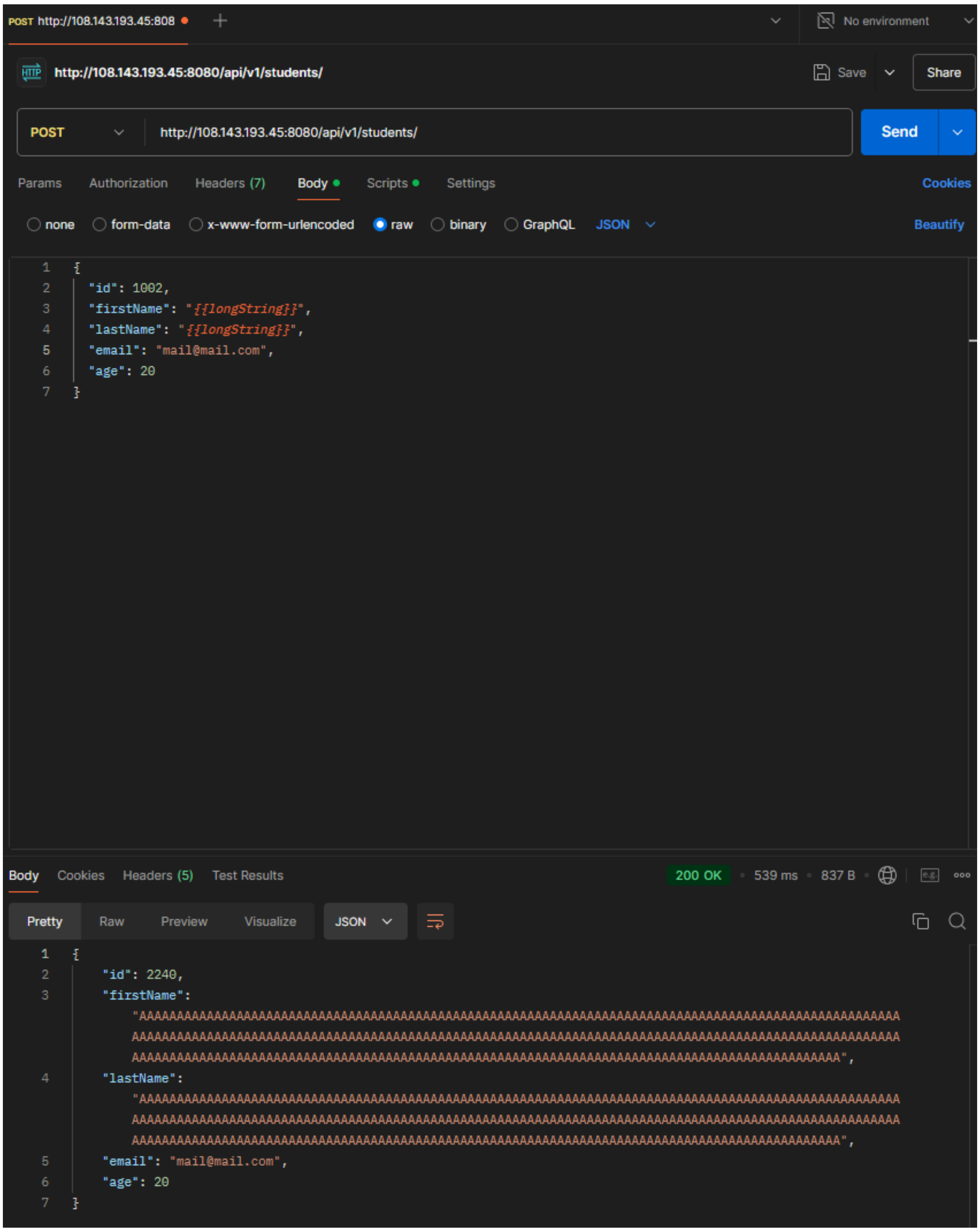
Supplement 4.2: Database result for id = 2236



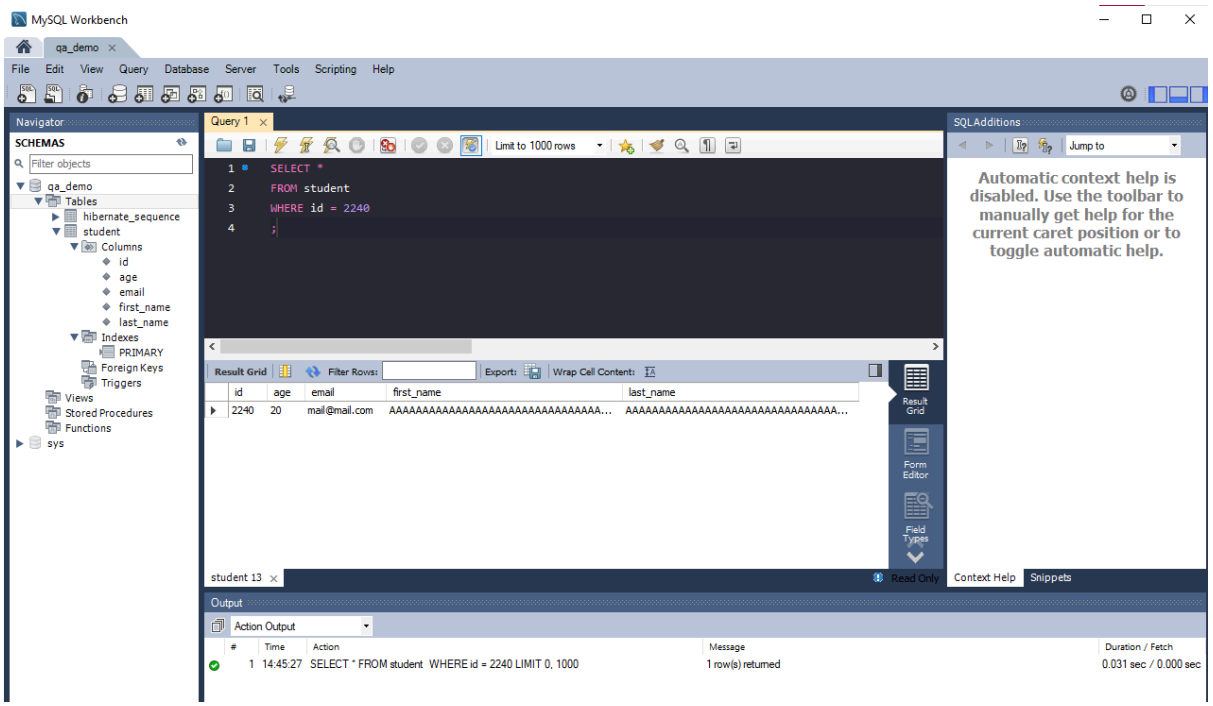
TC 5: Create Student with Invalid Data (POST Method with Long Strings)

Supplement 5.1.1: Postman Pre-request

Supplement 5.1.2: Postman results

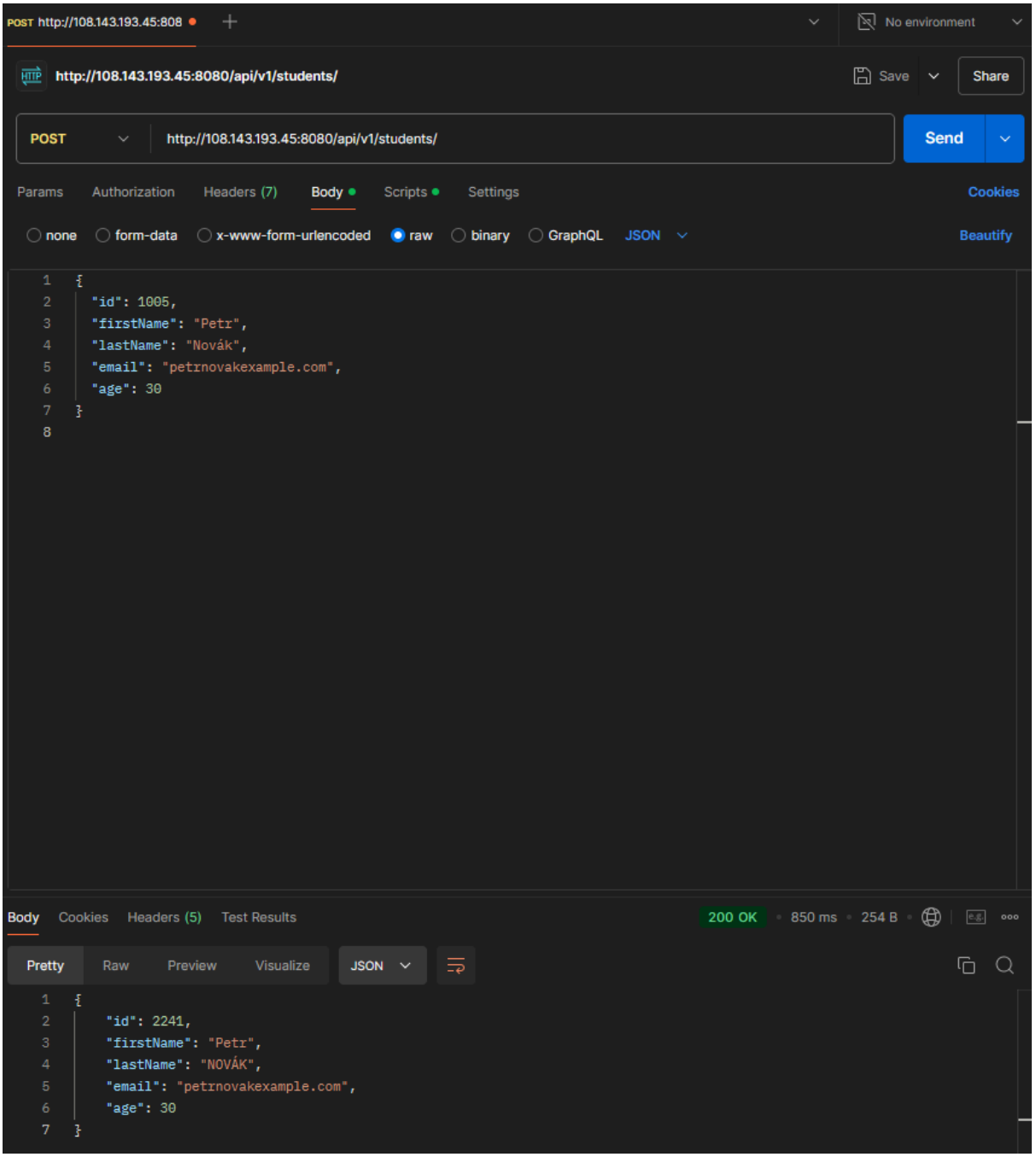


Supplement 5.2: DB output

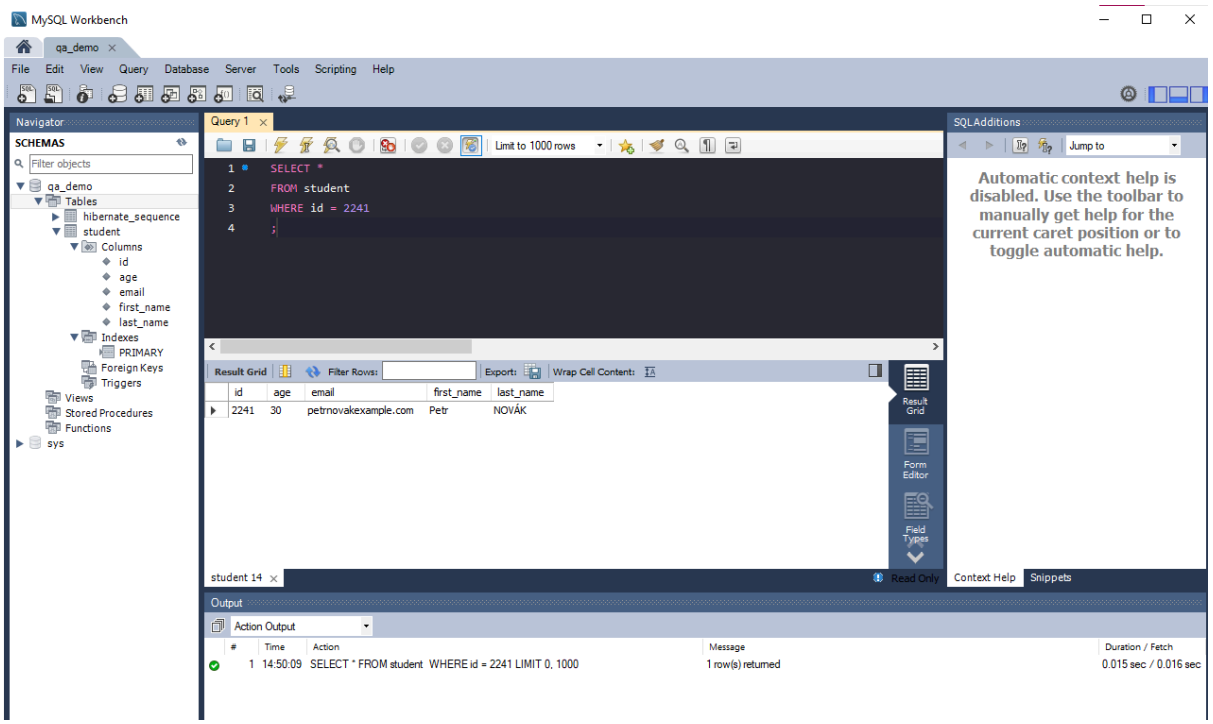


TC 6: Create Student with Invalid Data (POST Method with invalid email adress)

Supplement 6.1: Postman results

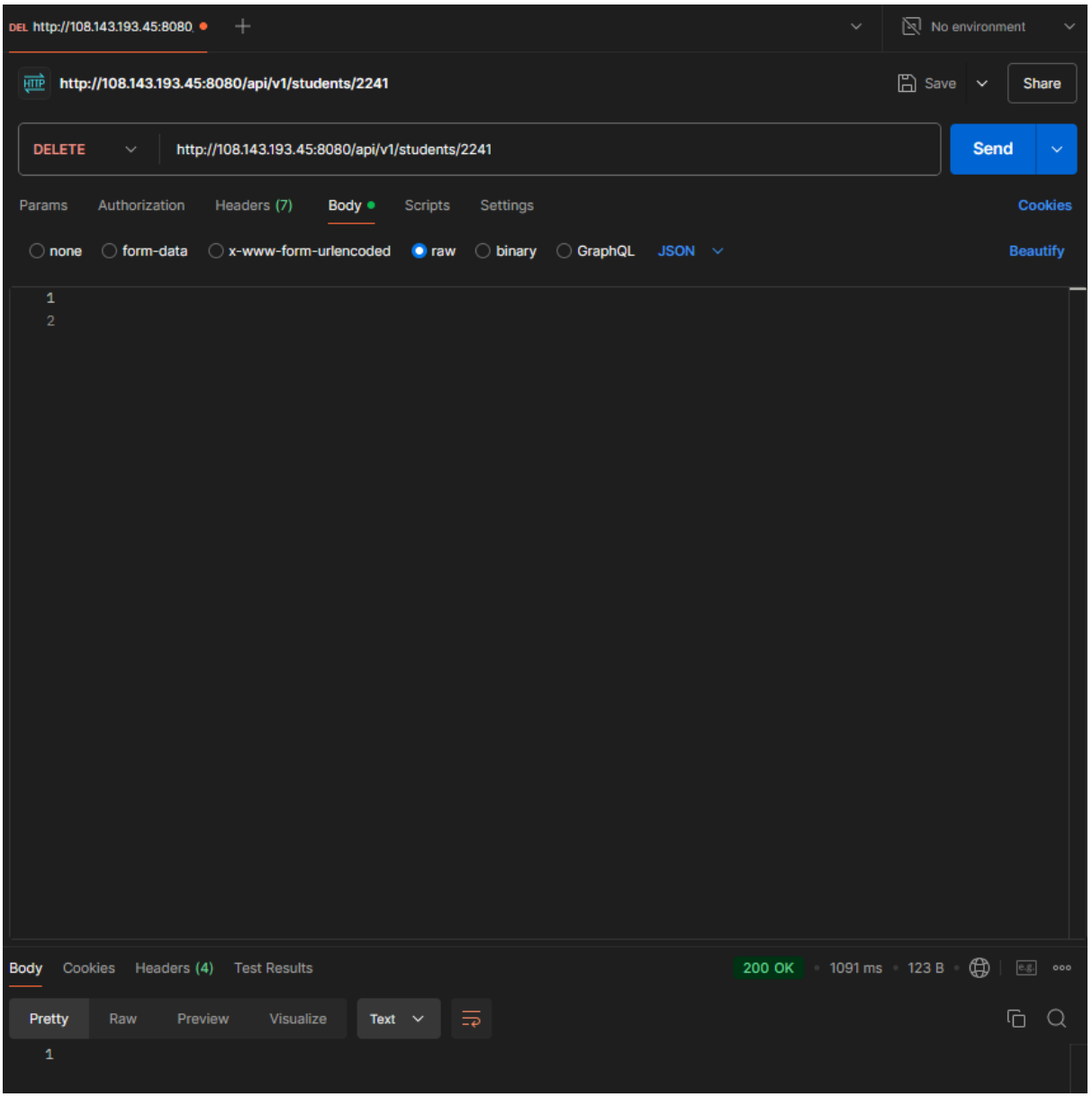


Supplement 6.2: DB output

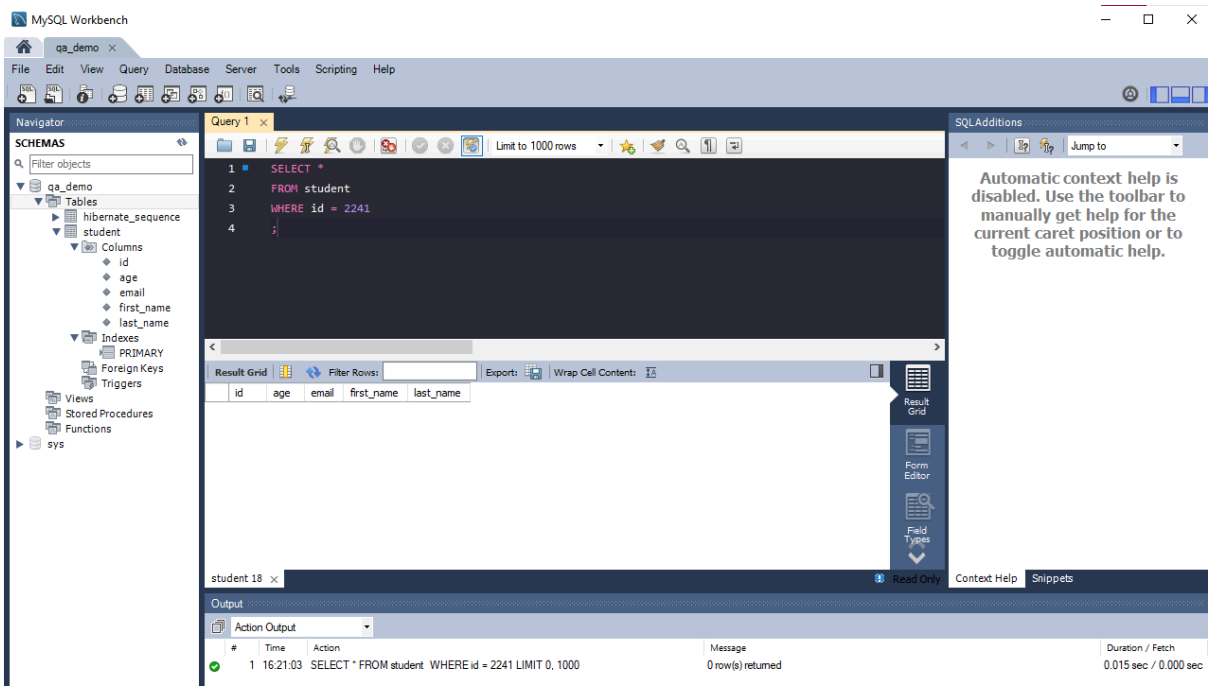


TC 7: Delete Existing Student (DELETE Method)

Supplement 7.1: Postman results



Supplement 7.2: DB output



TESTS OVERVIEW

Below is an overview of all test scenarios I have done, 2 passed and 5 failed. For those failed a bug report with additional information is included in the next section.

Test Case	Scenario	Expected Result	Actual Result	Status	Execution Timestamp	Environment
TC1	Retrieve Student Data Using Path Parameter (Standard RESTful)	200 OK; DB output matches Postman response	200 OK; DB output matches Postman response	PASS	2024-11-09 13:40:26	MySQL Workbench 8.0.32 build 3737333 CE (64-bit); Firefox Browser 132.0.1 (64-bit); Postman for Web Version 11.19.0-241023-0434; Windows 10 Pro Version 22H2 OS build 19045.4894
TC2	Retrieve Student Data Using Query Parameter (Potential Bug Check)	200 OK; DB output matches Postman response.	200 OK; DB output does not match postman. Postman returns different data	FAIL	2024-11-10 13:45:32	MySQL Workbench 8.0.32 build 3737333 CE (64-bit); Firefox Browser 132.0.1 (64-bit); Postman for Web Version

			than requested: https://drive.google.com/file/d/1X6KOuBi3b0ur39TqlGtpC9AyPwEG4-GA/view?usp=sharing			11.19.0-241023-0434; Windows 10 Pro Version 22H2 OS build 19045.4894

TC3	Handle Invalid Student Data (GET Method for Non-Existent Student)	404 Not Found;
		
		Error message is displayed:
		{
		
		“error": "Student not found"
		
		}
			500 Internal Server Error;
			
			Postman output: https://drive.google.com/file/d/18Yv5sleUaQCVdbutHsgCspex6qdYVtrU/view?usp=sharing
				FAIL	2024-11-10 10:33:11	Firefox Browser 132.0.1 (64-bit); Postman for Web Version 11.19.0-241023-0434; Windows 10 Pro Version 22H2 OS build 19045.4894

TC4	Create a New Student Record (POST Method with Valid Data)	200 OK; Postman response: { "id": 2236, "firstName":	200 OK; Postman output: https://drive.google.com/file/d/1M3ipnKU3kraMvKR74TpUtqAAIBEiJ2Hf/view?usp=sharing	FAIL	2024-11-10 12:10:45	MySQL Workbench 8.0.32 build 3737333 CE (64-bit); Firefox Browser 132.0.1 (64-bit); Postman for Web Version 11.19.0-241023-0

		"Petr", "lastName": "Novák", "email": "petr.novak@exa mple.com", "age": 30 } DB output matches Postman response	The record is created with a different ID and capitalized surname, can be found in DB but does not fulfill the test requirements.			434; Windows 10 Pro Version 22H2 OS build 19045.4894

TC5	Create Student with Invalid Data (POST Method with Long Strings)	400 Bad Request;
		
		Postman response:
		
		{
		
		"error": "Input data too long or invalid format"
		
		}
			200 OK;
			
			Postman output: https://drive.google.com/file/d/1-JjPCvBmRCUt7AJFhCQUXEM2F7mzTE-c/view?usp=sharing
			
			The record is created with invalid data.
				FAIL	2024-11-10
					14:45:27
						MySQL
						Workbench 8.0.32
						build 3737333 CE
						(64-bit);
						Firefox Browser
						132.0.1 (64-bit);
						Postman for Web
						Version
						11.19.0-241023-0
						434;
						Windows 10 Pro
						Version 22H2 OS
						build 19045.4894

		No record is created in the DB.				

TC6	Create Student with Invalid Data (POST Method with invalid email adress)	400 Bad Request;
		
		Postman response:
		
		{
		
		"error": "Invalid email format"
		
		}
		
		No record is created in the DB.
			200 OK;
			
			Postman output: https://drive.google.com/file/d/1JpMh76pm1M-SYYBUZh-tVVaaHpiEII4K/view?usp=sharing
			
			The record is created with invalid data.
				FAIL	2024-11-10 14:50:09	MySQL Workbench 8.0.32 build 3737333 CE (64-bit);
						Firefox Browser 132.0.1 (64-bit);
						Postman for Web Version 11.19.0-241023-0434;
						Windows 10 Pro Version 22H2 OS build 19045.4894

TC7	Delete Existing Student (DELETE Method)	200 OK; DB output will return empty for id 2241.	200 OK; DB output returns empty for id 2241.	PASS	2024-11-10 16:21:03	MySQL Workbench 8.0.32 build 3737333 CE (64-bit); Firefox Browser 132.0.1 (64-bit); Postman for Web Version 11.19.0-241023-0434; Windows 10 Pro Version 22H2 OS

						build 19045.4894

BUG REPORT

Based on the performed scenarios, I discovered the mentioned application errors. I know it is ideal to include individual steps in the bug report, but this format does not make it very readable if I add them. For this purpose I list the "Related Test Case" so the information can be looked up.

Bug ID			**Tracking**			**Bug Details**				**Environment**			
ID Number	**Reported by**	**Submission date**	**Assigned to**	**Severity**	**Priority**	**Related Test Case**	**Scenario**	**Expected Result**	**Actual Result**	**DB Tool**	**Browser**	**Postman**	**Operating system**
001	Daniel Pecka	2024-11-10	-	Major	High	TC2	Retrieve Student Data Using Query Parameter (Potential Bug Check)	200 OK; DB output matches Postman response.	200 OK; DB output does not match postman. Postman returns different data than requested: https://drive.google.com/file/d/1X6KOuBi3b0ur39TqlGtpC9AyPwEG4-GA/view?usp=sharing	MySQL Workbench 8.0.32 build 3737333 CE (64-bit)	Firefox Browser 132.0.1 (64-bit)	Postman for Web Version 11.19.0-241023-0434	Windows 10 Pro Version 22H2 OS build 19045.4894

002	Daniel Pecka	2024-11-10	-	Minor	Low	TC3	Handle Invalid Student Data (GET Method for Non-Existent Student)	404 Not Found; Error message is displayed: { “error": "Student not found" }	500 Internal Server Error; Postman output: https://drive.google.com/file/d/18Yv5sleUaQCVdbutHsgCspex6qdYVtrU/view?usp=sharing	-	Firefox Browser 132.0.1 (64-bit)	Postman for Web Version 11.19.0-241023-0434	Windows 10 Pro Version 22H2 OS build 19045.4894
003	Daniel Pecka	2024-11-10	-	Average	Medium	TC4	Create a New Student Record (POST Method with Valid Data)	200 OK; Postman response: { "id": 2236, "firstName": "Petr", "lastName": "Novák", "email": "petr.novak@example.com", "age": 30 } DB output matches Postman response	200 OK; Postman output: https://drive.google.com/file/d/1M3ipnKU3kraMvKR74TpUtqAAIBEiJ2Hf/view?usp=sharing The record is created with a different ID and capitalized surname, can be found in DB but does not fulfill the test requirements.	MySQL Workbench 8.0.32 build 3737333 CE (64-bit)	Firefox Browser 132.0.1 (64-bit)	Postman for Web Version 11.19.0-241023-0434	Windows 10 Pro Version 22H2 OS build 19045.4894

004	Daniel Pecka	2024-11-10	-	Major	High	TC5	Create Student with Invalid Data (POST Method with Long Strings)	400 Bad Request; Postman response: { "error": "Input data too long or invalid format" } No record is created in the DB.	200 OK; Postman output: https://drive.google.com/file/d/1-JjPCvBmRCUt7AJFhCQUXEM2F7mzTE-c/view?usp=sharing The record is created with invalid data.	MySQL Workbench 8.0.32 build 3737333 CE (64-bit)	Firefox Browser 132.0.1 (64-bit)	Postman for Web Version 11.19.0-241023-0434	Windows 10 Pro Version 22H2 OS build 19045.4894
005	Daniel Pecka	2024-11-10	-	Major	High	TC6	Create Student with Invalid Data (POST Method with invalid email adress)	400 Bad Request; Postman response: { "error": "Invalid email format" } No record is created in the DB.	200 OK; Postman output: https://drive.google.com/file/d/1JpMh76pm1M-SYYBUZh-tVVaaHpiEll4K/view?usp=sharing The record is created with invalid data.	MySQL Workbench 8.0.32 build 3737333 CE (64-bit)	Firefox Browser 132.0.1 (64-bit)	Postman for Web Version 11.19.0-241023-0434	Windows 10 Pro Version 22H2 OS build 19045.4894