

FINÁLNÍ PROJEKT

č.1



ENGETO

Autor: Iveta Eliášková
Datum: 21.09.2024

OBSAH

ZADÁNÍ	3
TESTOVACÍ SCÉNÁŘE	4
EXEKUCE TESTŮ	5
BUG REPORT	5

ZADÁNÍ

Cílem finálního projektu je otestovat funkčnost aplikace, která slouží k manipulaci s daty o studentech. Aplikace má rozhraní REST-API, které umožňuje vytvoření, smazání a získání dat..

Přístupové údaje:

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

Poznámky:

Nezapomeňte, že v IT se data musí někde uložit a poté získat. Proto ověřte, že data jsou správně uložena a získávána z databáze.

Nezapomeňte do testovacích scénářů uvést testovací data, očekávaný výsledek včetně těla odpovědi a stavových kódů.

TESTOVACÍ SCÉNÁŘE

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

1) Abstract: Get data about existing student - Test the GET method

Test data preparation:

Existing student id = 340

Insert in MySQL: `SELECT * FROM student where id=340;`

Steps:

- 1) GET: `http://108.143.193.45:8080/api/v1/students/340`
- 2) Press 'Send' button

Expected result:

- 1) Status code: 200 OK
- 2) Response:

```
{
  "Id": 340,
  "firstName": "john",
  "lastName": "DOE",
  "email": "john.doe@gmail.com",
  "age": 34
}
```

2) Abstract: Get data about NOT-existing student - Test the GET method

Test data preparation:

NOT-existing student id = 1

Insert in MySQL: `SELECT * FROM student where id=1;`

Steps:

- 1) GET: <http://108.143.193.45:8080/api/v1/students/1>
- 2) Pres 'Send' button

Expected result:

- 1) Status code: 404 not found

3) Abstract: delete all data about an existing student - Test the DELETE method

Test data preparation:

Existing student = 290

Insert in MySQL: SELECT * FROM student where id=290; (check, that he exists)

Delete the student 290 in Postman.

Steps:

- 1) GET: <http://108.143.193.45:8080/api/v1/students/290>
- 2) Change from GET to DELETE method and press "Send" button

Expected:

- 1) All data about particular student are removed
- 2) Status result: (Deleted) 200 OK

4) Abstract: An attempt to delete the deleted student - Test the DELETE method

Test data preparation:

Not-existing student = 290

A test, what the system does, when trying to delete the deleted student again.

Steps:

- 1) DELETE: <http://108.143.193.45:8080/api/v1/students/290>
- 2) Press the 'Send' button

Expected:

- 1) Status code: 404

5) Abstract: An attempt to create a new student, who meets mandatory requirements - Test the POST method

Test data preparation:

There is a prepared dataset for a new student.

Steps:

- 1) Set up the POST method in Postman.
- 2) Delete the number of a student: <http://108.143.193.45:8080/api/v1/students/>
- 3) Set up the Postman: the Body of the request - option 'Raw', set up JSON format
- 4) Insert the prepared dataset:

```
{ "firstName": "Ivetas",  
  "lastName": "Elis",  
  "email": "elis@gmail.com",  
  "age": 37 }
```
- 5) Press 'Send' button
- 6) Checked the new student in the database, if the data in database and Postman are the same.

Expected:

- 1) Status code: 201 created
- 2) The answer from Postman - we see a created student and a newly created ID.
- 3) The dataset of the new student is the same in the Postman and the Database, check it based on a new id.

```
{ "id":  
  "firstName": "Ivetas",  
  "lastName": "Elis",  
  "email": "elis@gmail.com",  
  "age": 37 }
```

6) Abstract: An attempt to create a new student, who does not meet the mandatory requirements - Test the POST method

Test data preparation:

There is a prepared dataset for a new student.

Steps:

- 1) Set up the POST method in Postman.
- 2) Delete the number of a student: <http://108.143.193.45:8080/api/v1/students/>
- 3) Set up the Postman: the Body of the request - option 'Raw', set up JSON format
- 4) Insert the prepared dataset:

```
{ "firstName": "1",  
  "lastName": "t",  
  "email": "@jde.cas",  
  "age": 0.5}
```
- 5) Press 'Send' button

Expected:

- 1) Status code: 400 bad request "firstName, lastName must be between 3 - 50 characteristic", "email must be in a format: *@*.*", "age must be in a format: 1-150 year"
- 2) The new student will not be created.

7) Abstract: An attempt to create a new student with an empty "age" attribute - Test the POST method

Test data preparation:

There is a prepared dataset for a new student.

Steps:

- 1) Set up the POST method in Postman.
- 2) Delete the number of a student: <http://108.143.193.45:8080/api/v1/students/>
- 3) Set up the Postman: the Body of the request - option 'Raw', set up JSON format
- 4) Insert the prepared dataset:

```
{ "firstName": "Jan",  
  "lastName": "Janicek",  
  "email": "tak@jde.cas",  
  "age": }
```
- 5) Press 'Send' button

Expected:

- 1) Status code: 400 bad request "age must be in a format: 1-150 year".
- 2) The new student will not be created.






EXEKUCE TESTŮ

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

1) Abstract: Get data about existing student - Test the GET method

Test: PASSED

```
1 SELECT * FROM student where id=340
```

Result Grid		 Filter Rows: <input type="text"/>	Edit: 			E
	id	age	email	first_name	last_name	
▶	340	34	john.doe@gmail.com	john	DOE	
*	NULL	NULL	NULL	NULL	NULL	

HTTP `http://108.143.193.45:8080/api/v1/students/340` Save

GET `http://108.143.193.45:8080/api/v1/students/340` Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

Body Cookies Headers (5) Test Results 200 OK 167 ms 248 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 340,
3   "firstName": "john",
4   "lastName": "DOE",
5   "email": "john.doe@gmail.com",
6   "age": 34
}
```

2) Abstract: Get data about NOT-existing student - Test the GET method


Test: FAILED

3) Abstract: delete all data about an existing student - Test the DELETE method

Test: PASSED

```
Query 1 x
1 • select *from student WHERE ID=290;
2
```

	id	age	email	first_name	last_name
▶	290	39	jet@otam.cz	Miru	JETOTAM
*	NULL	NULL	NULL	NULL	NULL

 http://108.143.193.45:8080/api/v1/students/290

 Save

GET  http://108.143.193.45:8080/api/v1/students/290

Send

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

Cookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

Body Cookies Headers (5) Test Results

 200 OK 117 ms 245 B

Save Response 

Pretty

Raw

Preview

Visualize

JSON 



```
1 {
2   "id": 290,
3   "firstName": "Miru",
4   "lastName": "JETOTAM",
5   "email": "jet@otam.cz",
6   "age": 39
7 }
```

HTTP <http://108.143.193.45:8080/api/v1/students/290> Save

DELETE ▼ Send

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

Query Params

	Key	Value	Bulk Edit
	Key	Value	

Body Cookies Headers (4) Test Results 200 OK 343 ms 123 B Save Response ▼

Pretty Raw Preview Visualize Text ▼ 🔍

1

```
1 • select *from student WHERE ID=290;
```

2

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Col

id	age	email	first_name	last_name
NULL	NULL	NULL	NULL	NULL

4) Abstract: An attempt to delete the deleted student - Test the DELETE method
Test result: FAILED

5) Abstract: An attempt to create a new student - Test the POST method
Test result: FAILED

6) Abstract: An attempt to create a new student, who does not meet the mandatory requirements - Test the POST method
Test result: FAILED

7) Abstract: An attempt to create a new student with an empty "age" attribute - Test the POST method
Test result: FAILED

BUG REPORT

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

2) Abstract: Get data about NOT-existing student - Test the GET method

Test result: FAILED

Expected result: Status code: 404

The actual test result: 500 Internal Server Error

The screenshot displays two interfaces. The top interface is a database client showing a SQL query: `1 select *from student where id=1;`. Below the query, a 'Result Grid' shows a single row with all fields (id, age, email, first_name, last_name) set to NULL.

The bottom interface is an API testing tool. The URL bar shows `http://108.143.193.45:8080/api/v1/students/1`. The method is set to GET. The 'Send' button is visible. Below the URL bar, the 'Params' tab is active, showing a table with columns 'Key' and 'Value'.

At the bottom, the 'Body' tab is active, showing the response in JSON format:

```
1 {
2   "timestamp": "2024-07-23T17:21:31.705+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "message": "",
6   "path": "/api/v1/students/1"
7 }
```

The status bar at the bottom indicates a 500 Internal Server Error, 204 ms response time, and 285 B body size.

4) Abstract: An attempt to delete the deleted student - Test the DELETE method
Test result: FAILED

Expected result: Status code: 404
The actual test result: 500 Internal Server Error

HTTP

http://108.143.193.45:8080/api/v1/students/290

Save

DELETE

http://108.143.193.45:8080/api/v1/students/290

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

body

Cookies

Headers (4)

Test Results

500 Internal Server Error

127 ms

287 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

"timestamp": "2024-08-13T17:30:06.999+00:00",

"status": 500,

"error": "Internal Server Error",

"message": "",

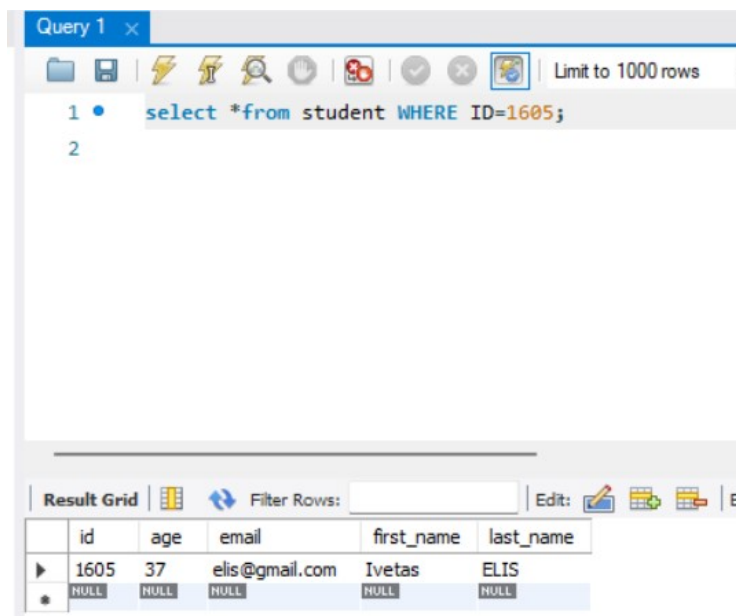
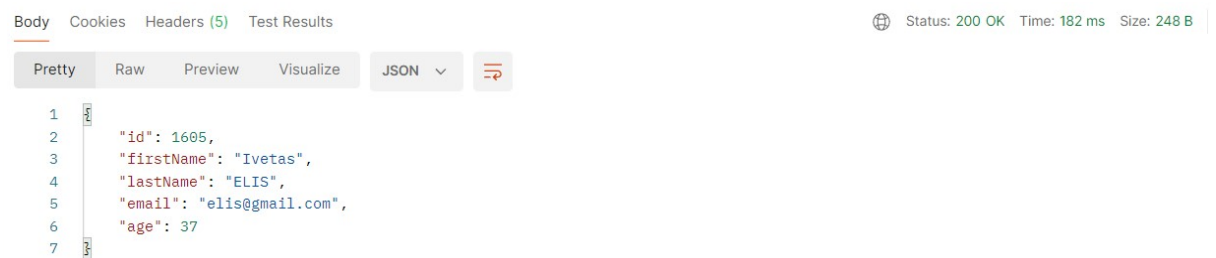
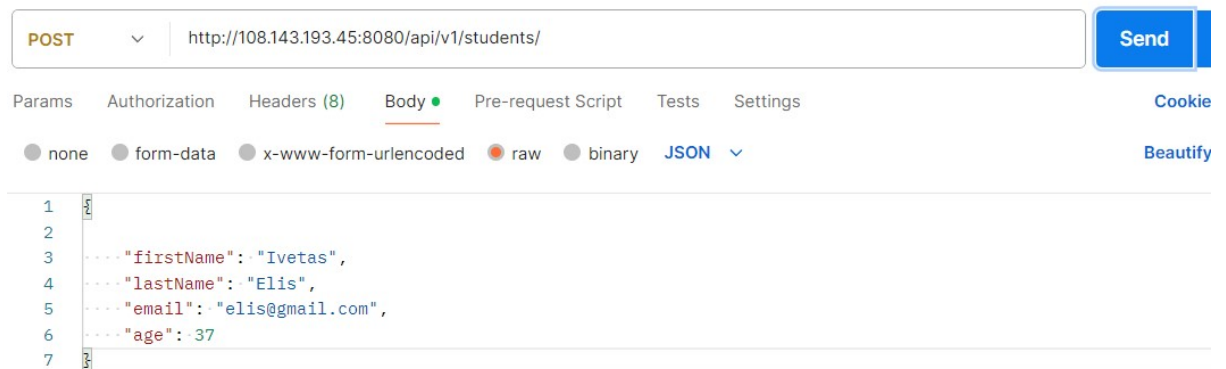
"path": "/api/v1/students/290"

5) Abstract: An attempt to create a new student - Test the POST method

Test result: FAILED

Expected result: Status code 201 created

The actual test result: 200 ok



6) Abstract: An attempt to create a new student, who does not meet the mandatory requirements - Test the POST method

Test result: FAILED

Expected result: 400 bad request "firstName, lastName must be between 3 - 50 characteristic", "email must be in a format: *@*.*", "age must be in a format: 1-150 year". The new student will not be created.

The actual test result: 200 ok, the new student was created.

Note: The numbers should not be accepted in the attributes "firstName" and "lastName".

The screenshot displays a REST client interface with two panels. The top panel shows the request configuration, and the bottom panel shows the response details.

Request Configuration:











- Method: POST
- URL: `http://108.143.193.45:8080/api/v1/students/`
- Body Type: JSON
- Body Content (lines 1-7):

```
1 {
2   "firstName": 1,
3   "lastName": "t",
4   "email": "@jde.cas",
5   "age": 0.5
6 }
```

Response Details:

- Status: 200 OK
- Time: 222 ms
- Size: 233 B
- Save Response: ✓
- Body Type: JSON (Pretty)
- Body Content (lines 1-7):

```
1 {
2   "id": 1846,
3   "firstName": "1",
4   "lastName": "T",
5   "email": "@jde.cas",
6   "age": 0
7 }
```


Limit to 1


1 •


select *from student WHERE ID=1846;

2

Result Grid



 Filter Rows:

Edit: 

	id	age	email	first_name	last_name
▶	1846	0	@jde.cas	1	T
*	NULL	NULL	NULL	NULL	NULL

7) Abstract: An attempt to create a new student with an empty "age" attribute - Test the POST method

Test result: FAILED

Expected result: Status code 400 bad request

The actual test result: 405 Method not allowed

The screenshot displays a REST client interface with two panels. The top panel shows the request configuration, and the bottom panel shows the response details.

Request Configuration:

- Method: POST
- URL: `http://108.143.193.45:8080/api/v1/students/155`
- Body Type: JSON
- Body Content:

```
1 {
2   "firstName": "Jan",
3   "lastName": "Janicek",
4   "email": "tak@jde.cas",
5   "age":
6 }
```

Response Details:

- Status: 405 Method Not Allowed
- Time: 120 ms
- Size: 330 B
- Response Body (Pretty):

```
1 {
2   "timestamp": "2024-09-21T19:50:57.291+00:00",
3   "status": 405,
4   "error": "Method Not Allowed",
5   "message": "",
6   "path": "/api/v1/students/155"
7 }
```