

FINÁLNÍ PROJEKT

č.1



ENGETO

Autor: Václav Schwarz
Datum: 26.11.2024

OBSAH

ZADÁNÍ.....	3
TESTOVACÍ SCÉNÁŘE	4
Testování metody GET.....	4
Testování metody POST	6
Testování metody DELETE	10
EXEKUCE TESTŮ	11
Exekuce metody GET.....	11
Exekuce metody POST	15
Exekuce metody DELETE	22
BUG REPORT	25
Bug report metody GET.....	25
Bug report metody POST	26

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	Default scheme: qa_demo Host: aws.connect.psdb.cloud Port: 3306
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.

Testování metody GET

Testovací scénář 1. – metoda GET zobrazí data o všech existujících studentech

Kroky:

1. Otevřu si program Postman
2. Do příkazového řádku zadám adresu: <http://108.143.193.45:8080/api/v1/students/>
3. Vyberu metodu GET a dám odeslat

Očekávaný výsledek:

1. Aplikace mi vypíše všechny existující studenty
2. Status kód bude 200 OK

Testovací scénář 2. - metoda GET zobrazí konkrétního existujícího studenta podle ID

Kroky:

1. Otevřu si program Postman
2. Do příkazového řádku zadám adresu: <http://108.143.193.45:8080/api/v1/students/366>
3. Vyberu metodu GET a dám odeslat

Očekávaný výsledek:

1. Aplikace mi vypíše jméno konkrétního studenta podle ID
2. Status kód bude 200 OK

Testovací scénář 3. - metoda GET zadání neexistujícího studenta podle ID

Kroky:

1. Otevřu si program Postman
2. Do příkazového řádku zadám adresu: <http://108.143.193.45:8080/api/v1/students/3680>
3. Vyberu metodu GET a dám odeslat

Očekávaný výsledek:

1. Aplikace mi nevypíše jméno konkrétního studenta podle ID
2. Status kód bude 404 Not Found

Testovací scénář 4. - metoda GET chybné zadání existujícího studenta podle ID

Kroky:

1. Otevřu si program Postman
2. Do příkazového řádku zadám adresu: <http://108.143.193.45:8080/api/v1/students/396n>
3. Vyberu metodu GET a dám odeslat

Očekávaný výsledek:

1. Aplikace mi nevypíše jméno konkrétního studenta podle ID
2. Status kód bude 400 Bad Request

Testování metody POST

Testovací scénář 1. – metoda POST vytvoří nového studenta a uloží ho do databáze

Kroky:

1. Otevřu si program Postman
2. Do příkazového řádku zadám adresu: <http://108.143.193.45:8080/api/v1/students/>
3. Vyberu metodu POST
4. Kliknu na tlačítko BODY, zaškrtnu raw a TEXT změním na JSON
5. Vytvořím nového studenta s těmito parametry

```
{
  "firstName": "Jan",
  "lastName": "KOLOVRATEK",
  "email": "kolovrat@gmail.com",
  "age": 35
}
```

Očekávaný výsledek:

1. Aplikace uloží nového studenta a přiřadí mu vlastní id
2. Status kód bude 200 OK

Testovací scénář 2. – metoda POST - kontrola vytvoření nového studenta v databázi

Kroky:

1. Otevřu si program MySQL Workbench
2. Zadám přihlašovací údaje do databáze:
Hostname: aws.connect.psdb.cloud
Port: [3306](#)
Username: [k9ik3tg4x7hdxk68nlqy](#)
Password: [pscale_pw_H9L99PYMlpZMkfuCbuZDu24irPmupXQvZEtg5pt5vtG](#)
Default scheme: [qa_demo](#)
3. Do příkazového pole zadáme následující příkaz:
SELECT * FROM student WHERE id = 2414;
4. Klikneme na ikonku blesku

Očekávaný výsledek:

1. Aplikace nám zobrazí námi hledaného studenta

Testovací scénář 3. – metoda POST neúplné zadání všech parametrů

Kroky:

1. Otevřu si program Postman
2. Do příkazového řádku zadám adresu: <http://108.143.193.45:8080/api/v1/students/>
3. Vyberu metodu POST
4. Kliknu na tlačítko BODY, zaškrtnu raw a TEXT změním na JSON
5. Vytvořím nového studenta s těmito parametry

```
{  
  "firstName": "Tereza",  
  "lastName": "KOPOVÁ",  
  "email": "kopova@gmail.com",  
  "age":  
}
```

Očekávaný výsledek:

1. Aplikace zahlásí chybu
2. Status kód bude 400 Bad Request

Testovací scénář 4. – metoda POST zadání číslic do parametru firstName

Kroky:

1. Otevřu si program Postman
2. Do příkazového řádku zadám adresu: <http://108.143.193.45:8080/api/v1/students/>
3. Vyberu metodu POST
4. Kliknu na tlačítko BODY, zaškrtnu raw a TEXT změním na JSON
5. Vytvořím nového studenta s těmito parametry

```
{  
  "firstName": "1234",  
  "lastName": "KOPOVÁ",  
  "email": "kopova@gmail.com",  
  "age": 28  
}
```

Očekávaný výsledek:

1. Aplikace zahlásí chybu
2. Status kód bude 400 Bad Request

Testovací scénář 5. – metoda POST - kontrola vytvoření nového studenta v databázi, který má ve firstName číslice

Kroky:

1. Otevřu si program MySQL Workbench
2. Zadám přihlašovací údaje do databáze:
Hostname: aws.connect.psdb.cloud
Port: 3306
Username: [k9ik3tg4x7hdxk68nlqy](#)
Password:
[pscale_pw_H9L99PYMlpZMkfuCbuZDu24irPmupXQvZEtg5pt5vtG](#)
Default scheme: [qa_demo](#)
3. Do příkazového pole zadáme následující příkaz:
SELECT * FROM student WHERE id = 2415;
4. Klikneme na ikonku blesku

Očekávaný výsledek:

1. Aplikace nám zobrazí námi hledaného studenta

Testovací scénář 6. – metoda POST zadání speciálních znaků do parametru lastName

Kroky:

1. Otevřu si program Postman
2. Do příkazového řádku zadám adresu: <http://108.143.193.45:8080/api/v1/students/>
3. Vyberu metodu POST
4. Kliknu na tlačítko BODY, zaškrtnu raw a TEXT změním na JSON
5. Vytvořím nového studenta s těmito parametry
{
 "firstName": "Ivona",
 "lastName": "#&@*/+",
 "email": "makalova@gmail.com",
 "age": 37
}

Očekávaný výsledek:

1. Aplikace zahlásí chybu
2. Status kód bude 400 Bad Request

Testovací scénář 7. – metoda POST - kontrola vytvoření nového studenta v databázi, který má v lastName speciální číslice

Kroky:

1. Otevřu si program MySQL Workbench
2. Zadám přihlašovací údaje do databáze:
 Hostname: aws.connect.psdb.cloud
 Port: [3306](#)
 Username: [k9ik3tg4x7hdxk68nlqy](#)
 Password:
 [pscale_pw_H9L99PYMlpZMkfuCbuZDu24irPmupXQvZEtg5pt5vtG](#)
 Default scheme: [qa_demo](#)
3. Do příkazového pole zadáme následující příkaz:
 SELECT * FROM student WHERE id = 2417;
4. Klikneme na ikonku blesku

Očekávaný výsledek:

2. Aplikace nám zobrazí námi hledaného studenta

Testování metody DELETE

Testovací scénář 1. – metoda DELETE – odstranění studenta s ID = 2417

Kroky:

1. Otevřu si program Postman
2. Do příkazového řádku zadám adresu: <http://108.143.193.45:8080/api/v1/students/>
3. Vyberu metodu DELETE
4. Kliknu na tlačítko BODY, zaškrtnu none
5. Do příkazového řádku za poslední lomítko přidám id studenta:
<http://108.143.193.45:8080/api/v1/students/2417>

Očekávaný výsledek:

1. Aplikace smaže studenta s id = 2417
2. Status kód bude 200 OK

Testovací scénář 2. – metoda DELETE - kontrola odstranění studenta z databáze s id = 2417

Kroky:

1. Otevřu si program MySQL Workbench
2. Zadám přihlašovací údaje do databáze:
Hostname: aws.connect.psdb.cloud
Port: 3306
Username: [k9ik3tg4x7hdxk68nlqy](#)
Password:
[pscale_pw_H9L99PYMlpZMkfuCbuZDu24irPmupXQvZEtg5pt5vtG](#)
Default scheme: [qa_demo](#)
3. Do příkazového pole zadáme následující příkaz:
SELECT * FROM student WHERE id = 2417;
4. Klikneme na ikonku blesku

Očekávaný výsledek:

3. Aplikace nenajde námi požadovaného studenta

Testovací scénář 3. – metoda DELETE – odstranění studenta s ID = 2415

Kroky:

1. Otevřu si program Postman
2. Do příkazového řádku zadám adresu: <http://108.143.193.45:8080/api/v1/students/>
3. Vyberu metodu DELETE
4. Kliknu na tlačítko BODY, zaškrtnu none
5. Do příkazového řádku za poslední lomítko přidám id studenta:
<http://108.143.193.45:8080/api/v1/students/id = 2415>

Očekávaný výsledek:

1. Aplikace zhlási chybu

2. Status kód bude 400 Bad Request

EXEKUCE TESTŮ

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

Exekuce metody GET

Testovací scénář 1. – metoda GET zobrazí data o všech existujících studentech

Kroky:

1. Otevřel jsem si program Postman
2. Do příkazového řádku zadal adresu: <http://108.143.193.45:8080/api/v1/students/>
3. Vybral metodu GET a odeslal

Očekávaný výsledek:

1. Aplikace mi vypíše všechny existující studenty
2. Status kód bude 200 OK

Aktuální výsledek:

1. Aplikace zobrazila všechny existující studenty
2. Status kód byl 200 OK

The screenshot shows the Postman interface. At the top, the URL bar contains `http://108.143.193.45:8080/api/v1/students/`. The method is set to `GET`. Below the URL bar, the `Params` tab is selected, showing an empty table with columns `Key`, `Value`, and `Bulk Edit`. The `Body` tab is also visible. At the bottom, the `Test Results` tab is selected, showing a status of `200 OK`, a response time of `312 ms`, and a size of `112.79 KB`. The response body is displayed in `JSON` format, showing a list of student objects. The first object is:

```
{
  "id": 352,
  "firstName": "Jana",
  "lastName": "NOVAKOVA",
  "email": "janovak@gmail.com",
  "age": 19
}
```

Testovací scénář 2. - metoda GET zobrazí konkrétního existujícího studenta podle ID

Kroky:

1. Otevřel jsem si program Postman
2. Do příkazového řádku zadal adresu: <http://108.143.193.45:8080/api/v1/students/366>
3. Vyberal metodu GET a odeslal

Očekávaný výsledek:

3. Aplikace mi vypíše existujícího studenta podle ID
4. Status kód bude 200 OK

Aktuální výsledek:

3. Aplikace zobrazila existujícího studenta
4. Status kód byl 200 OK

The screenshot shows the Postman interface. At the top, the URL bar displays `http://108.143.193.45:8080/api/v1/students/` with a 'Save' button. Below it, the request method is set to 'GET' and the full URL is `http://108.143.193.45:8080/api/v1/students/366`. The 'Send' button is visible. The 'Params' tab is selected, showing a table for Query Params with columns 'Key', 'Value', and 'Bulk Edit'. Below the table, the 'Body' tab is selected, showing the response status '200 OK', response time '173 ms', and response size '243 B'. The response body is displayed in JSON format:

```
1 {
2   "id": 366,
3   "firstName": "Petra",
4   "lastName": "25",
5   "email": "petra@sova.cz",
6   "age": 25
7 }
```

Testovací scénář 3. - metoda GET zadání neexistujícího studenta podle ID

Kroky:

1. Otevřel jsem si program Postman
2. Do příkazového řádku zadal adresu: <http://108.143.193.45:8080/api/v1/students/3680>
3. Vybral metodu GET a odeslal

Očekávaný výsledek:

1. Aplikace mi nevypíše jméno konkrétního studenta podle ID
2. Status kód bude 404 Not Found

Aktuální výsledek:

1. Aplikace mi nevypsala jméno konkrétního studenta podle ID
2. Status kód byl 500 Internal Server Error

The screenshot shows the Postman interface for a GET request. The URL bar contains `http://108.143.193.45:8080/api/v1/students/3680` and the method is set to GET. The 'Send' button is visible. Below the URL bar, the 'Params' tab is selected, showing a table with one row: 'Key' and 'Value'. The 'Test Results' tab is also visible, showing a '500 Internal Server Error' with a status of 323 ms and 288 B. The response body is displayed in JSON format:

```
1 {
2   "timestamp": "2024-11-19T19:23:03.391+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "message": "",
6   "path": "/api/v1/students/3680"
7 }
```

Testovací scénář 4. - metoda GET chybné zadání existujícího studenta podle ID

Kroky:

1. Otevřel jsem si program Postman
2. Do příkazového řádku zadal adresu: <http://108.143.193.45:8080/api/v1/students/396n>
3. Vybral metodu GET a odeslal

Očekávaný výsledek:

1. Aplikace mi nevypíše jméno konkrétního studenta podle ID
2. Status kód bude 400 Bad Request

Aktuální výsledek:

1. Aplikace mi nevypsala jméno konkrétního studenta podle ID
2. Status kód byl 400 Bad Request

The screenshot shows the Postman interface. At the top, a GET request is configured with the URL `http://108.143.193.45:8080/api/v1/students/396n`. Below the URL bar, the 'Query Params' section is empty. The 'Send' button is visible. The response section shows a status of '400 Bad Request' with a response time of '197 ms' and a size of '271 B'. The response body is displayed in JSON format:

```
1 {
2   "timestamp": "2024-11-19T19:51:27.570+00:00",
3   "status": 400,
4   "error": "Bad Request",
5   "message": "",
6   "path": "/api/v1/students/396n%20"
7 }
```

Exekuce metody POST

Testovací scénář 1. – metoda POST vytvoří nového studenta a uloží ho do databáze

Kroky:

5. Otevřel jsem si program Postman
6. Do příkazového řádku jsem zadal adresu: <http://108.143.193.45:8080/api/v1/students/>
7. Vybral jsem metodu POST
8. Kliknul jsem na tlačítko BODY, zaškrtnul raw a TEXT jsem změnil na JSON
9. Vytvořil nového studenta s těmito parametry

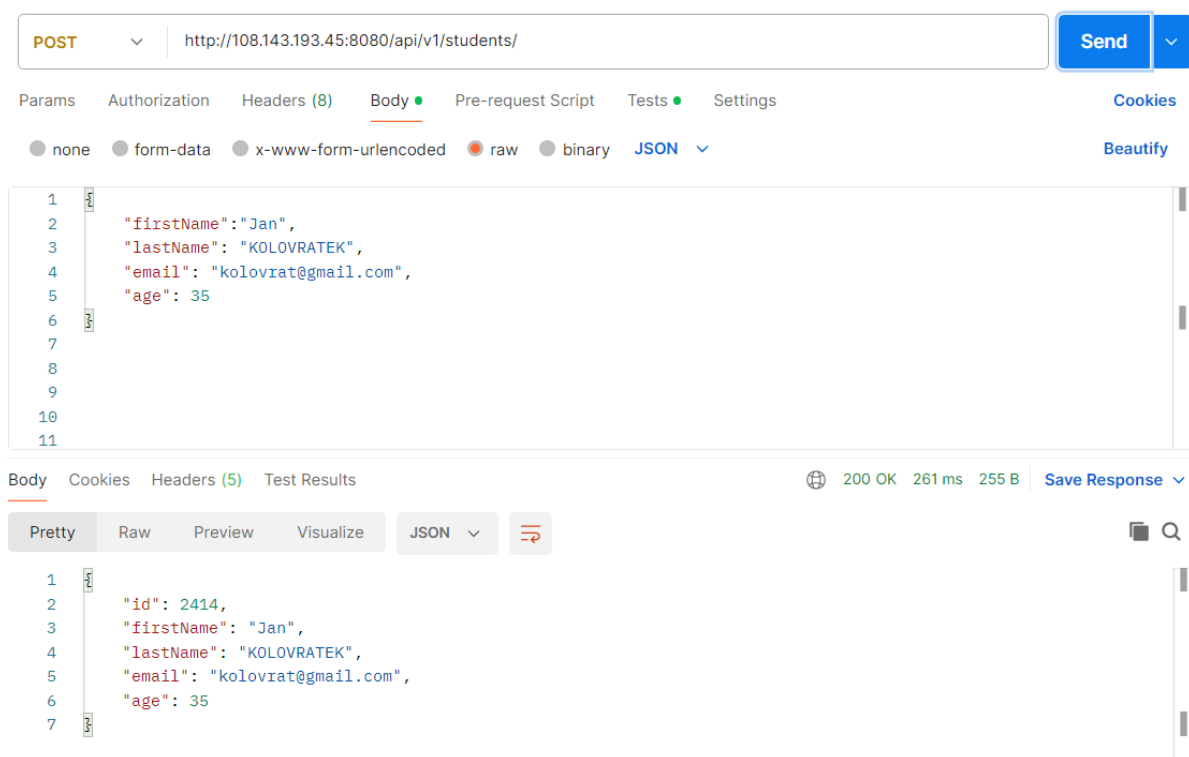
```
{
  "firstName": "Jan",
  "lastName": "KOLOVRATEK",
  "email": "kolovrat@gmail.com",
  "age": 35
}
```

Očekávaný výsledek:

1. Aplikace uloží nového studenta a přiřadí mu vlastní id
2. Status kód bude 200 OK

Aktuální výsledek:

4. Aplikace uložila nového studenta a přiřadila mu vlastní id 2414
5. Status kód byl 200 OK



Testovací scénář 2. – metoda POST - kontrola vytvoření nového studenta v databázi

Kroky:

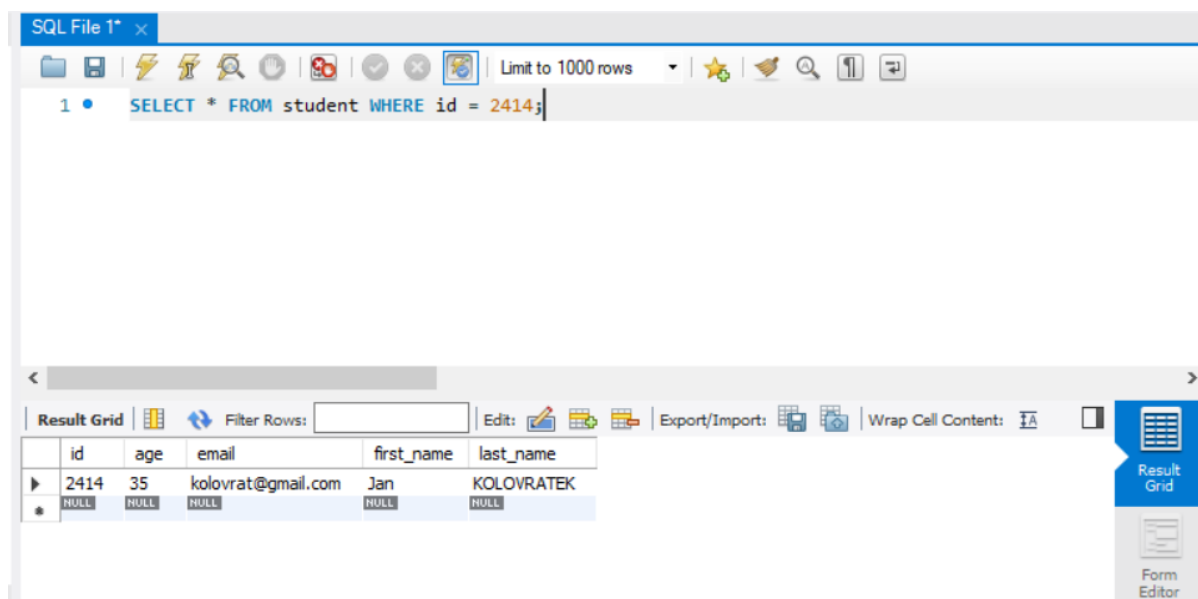
1. Otevřel jsem si program MySQL Workbench
2. Zadal jsem přihlašovací údaje do databáze:
Hostname: [aws.connect.psdb.cloud](#)
Port: [3306](#)
Username: [k9ik3tg4x7hdxk68nlqy](#)
Password:
[pscale_pw_H9L99PYMlpZMkfuCbuZDu24irPmupXQvZEtg5pt5vtG](#)
Default scheme: [qa_demo](#)
3. Do příkazového pole jsem zadal následující příkaz:
`SELECT * FROM student WHERE id = 2414;`
4. Klikl jsem na ikonku blesku

Očekávaný výsledek:

1. Aplikace nám zobrazí námi hledaného studenta

Aktuální výsledek:

2. Aplikace nám zobrazila námi hledaného studenta



The screenshot shows the MySQL Workbench interface. The SQL editor at the top contains the query: `SELECT * FROM student WHERE id = 2414;`. Below the editor, the 'Result Grid' is displayed, showing the results of the query. The grid has columns for 'id', 'age', 'email', 'first_name', and 'last_name'. The first row shows the student with id 2414, age 35, email kolovrat@gmail.com, first name Jan, and last name KOLOVRATEK. Below this row, there are two rows with 'NULL' values for all columns, indicating no more results were found.

	id	age	email	first_name	last_name
▶	2414	35	kolovrat@gmail.com	Jan	KOLOVRATEK
*	NULL	NULL	NULL	NULL	NULL

Testovací scénář 3. – metoda POST neúplné zadání všech parametrů

Kroky:

1. Otevřu si program Postman
2. Do příkazového řádku jsem zadal adresu: <http://108.143.193.45:8080/api/v1/students/>
3. Vybral jsem metodu POST
4. Kliknul jsem na tlačítko BODY, zaškrtnul raw a TEXT změnil na JSON
5. Vytvořil nového studenta s těmito parametry

```
{
  "firstName": "Tereza",
  "lastName": "KOPOVÁ",
  "email": "kopova@gmail.com",
  "age":
}
```

Očekávaný výsledek:

1. Aplikace zhlásí chybu
2. Status kód bude 400 Bad Request

Aktuální výsledek:

1. Aplikace ohlásila chybu
2. Status kód byl 400 Bad Request

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

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

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

```
1
2 "firstName": "Tereza",
3 "lastName": "KOPOVÁ",
4 "email": "kopova@gmail.com",
5 "age":
6
7
8
9
10
11
```

Body Cookies Headers (4) Test Results Status: 400 Bad Request Time: 110 ms Size: 264 B Save Response

Pretty Raw Preview Visualize JSON

```
1
2 "timestamp": "2024-11-24T20:36:53.095+00:00",
3 "status": 400,
4 "error": "Bad Request",
5 "message": "",
6 "path": "/api/v1/students/"
7
```

Testovací scénář 4. – metoda POST zadání číslic do parametru firstName

Kroky:

6. Otevřel jsem si program Postman
7. Do příkazového řádku jsem zadal adresu: <http://108.143.193.45:8080/api/v1/students/>
8. Vyberal jsem metodu POST
9. Kliknul na tlačítko BODY, zaškrtnul raw a TEXT změnil na JSON
10. Vytvořil nového studenta s těmito parametry

```
{
  "firstName": "1234",
  "lastName": "KOPOVÁ",
  "email": "kopova@gmail.com",
  "age": 28
}
```

Očekávaný výsledek:

3. Aplikace zhlásí chybu
4. Status kód bude 400 Bad Request

Aktuální výsledek:

1. Aplikace vytvořila nového studenta a přiřadila mu id 2415
2. Status kód byl 200 OK

The screenshot shows the Postman interface. At the top, the method is set to POST and the URL is <http://108.143.193.45:8080/api/v1/students/>. The 'Body' tab is selected, and the format is set to JSON. The request body contains the following JSON:

```
{
  "firstName": "1234",
  "lastName": "KOPOVÁ",
  "email": "kopova@gmail.com",
  "age": 28
}
```

Below the request, the response is shown. The status is 200 OK, the time is 355 ms, and the size is 251 B. The response body is displayed in the 'Pretty' tab, showing the following JSON:

```
{
  "id": 2415,
  "firstName": "1234",
  "lastName": "KOPOVÁ",
  "email": "kopova@gmail.com",
  "age": 28
}
```

Testovací scénář 5. – metoda POST- kontrola vytvoření nového studenta v databázi, který má ve firstName číslice

Kroky:

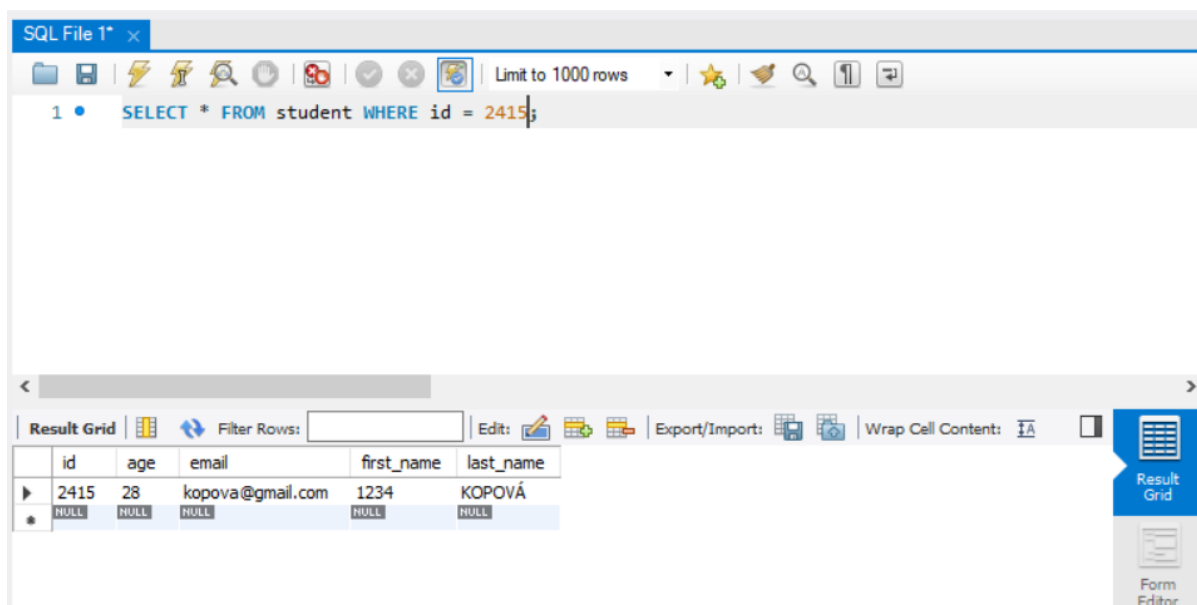
1. Otevřel jsem si program MySQL Workbench
2. Zadal jsem přihlašovací údaje do databáze:
Hostname: [aws.connect.psdb.cloud](#)
Port: [3306](#)
Username: [k9ik3tg4x7hdxk68nlqy](#)
Password:
[pscale_pw_H9L99PYMlpZMkfuCbuZDu24irPmupXQvZEtg5pt5vtG](#)
Default scheme: [qa_demo](#)
3. Do příkazového pole jsem zadal následující příkaz:
`SELECT * FROM student WHERE id = 2415;`
4. Klikl na ikonku blesku

Očekávaný výsledek:

6. Aplikace nám zobrazí námi hledaného studenta

Aktuální výsledek:

1. Aplikace nám zobrazila námi hledaného studenta



The screenshot shows the MySQL Workbench interface. At the top, the SQL editor contains the query: `SELECT * FROM student WHERE id = 2415;`. Below the editor, the 'Result Grid' is displayed, showing a single row of data for the student with id 2415. The columns are id, age, email, first_name, and last_name. The data row shows id: 2415, age: 28, email: kopova@gmail.com, first_name: 1234, and last_name: KOPOVÁ. There is also a row of NULL values below the data row. The interface includes various toolbars and a sidebar on the right with 'Result Grid' and 'Form Editor' buttons.

id	age	email	first_name	last_name
2415	28	kopova@gmail.com	1234	KOPOVÁ
NULL	NULL	NULL	NULL	NULL

Testovací scénář 6. – metoda POST zadání speciálních znaků do parametru lastName

Kroky:

1. Otevřel jsem si program Postman
2. Do příkazového řádku jsem zadal adresu: <http://108.143.193.45:8080/api/v1/students/>
3. Vyberal metodu POST
4. Kliknul jsem na tlačítko BODY, zaškrtnul raw a TEXT změnil na JSON
5. Vytvořil jsem nového studenta s těmito parametry

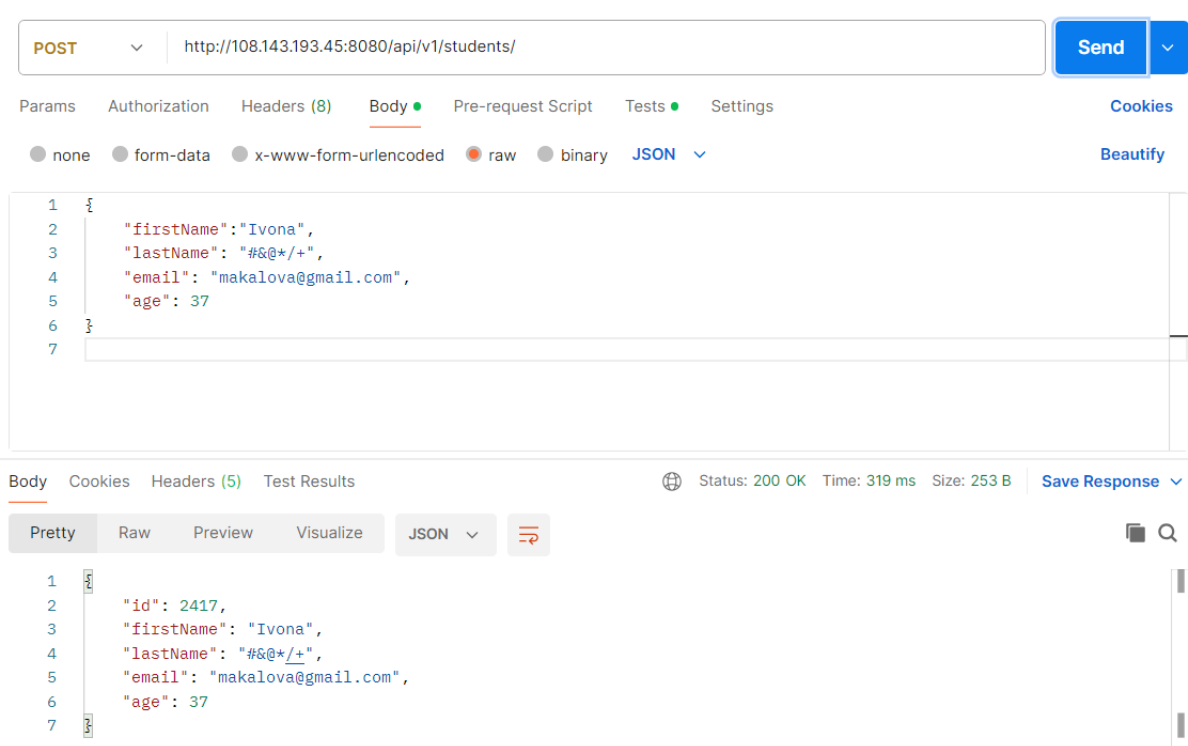
```
{
  "firstName": "Ivona",
  "lastName": "#&@*/+",
  "email": "makalova@gmail.com",
  "age": 37
}
```

Očekávaný výsledek:

1. Aplikace zhlásí chybu
2. Status kód bude 400 Bad Request

Aktuální výsledek:

1. Aplikace vytvořila nového studenta a přiřadila mu id 2417
2. Status kód byl 200 OK



Testovací scénář 7. – metoda POST - kontrola vytvoření nového studenta v databázi, který má v lastName speciální znaky

Kroky:

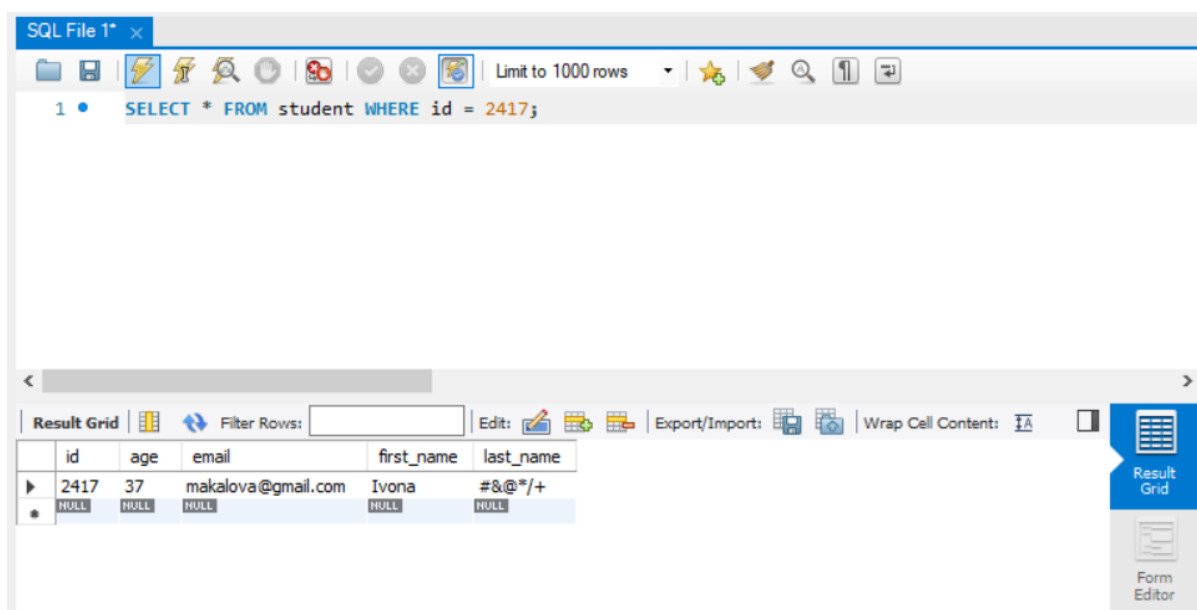
1. Otevřel jsem si program MySQL Workbench
2. Zadal přihlašovací údaje do databáze:
Hostname: [aws.connect.psdb.cloud](#)
Port: [3306](#)
Username: [k9ik3tg4x7hdxk68nlqy](#)
Password:
[pscale_pw_H9L99PYMlpZMkfuCbuZDu24irPmupXQvZEtg5pt5vtG](#)
Default scheme: [qa_demo](#)
3. Do příkazového pole jsem zadal následující příkaz:
`SELECT * FROM student WHERE id = 2417;`
4. Klikl na ikonku blesku

Očekávaný výsledek:

1. Aplikace nám zobrazí námi hledaného studenta

Aktuální výsledek:

1. Aplikace nám zobrazila námi hledaného studenta



Exekuce metody DELETE

Testovací scénář 1. – metoda DELETE – odstranění studenta s ID = 2417

Kroky:

1. Otevřel jsem si program Postman
2. Do příkazového řádku jsem zadal adresu: <http://108.143.193.45:8080/api/v1/students/>
3. Vyberal jsem metodu DELETE
4. Kliknul jsem na tlačítko BODY, zaškrtnul none
5. Do příkazového řádku za poslední lomítko jsem přidal id studenta:
<http://108.143.193.45:8080/api/v1/students/2417>

Očekávaný výsledek:

1. Aplikace smaže studenta s id = 2417
2. Status kód bude 200 OK

Aktuální výsledek:

1. Aplikace smazala studenta s id = 2417
2. Status kód byl 200 OK

The screenshot displays the Postman application interface. At the top, a request is configured with the method 'DELETE' and the URL 'http://108.143.193.45:8080/api/v1/students/2417'. The 'Body' tab is selected, and the content type is set to 'none'. Below the request configuration, the response is shown with a status of '200 OK', a time of '175 ms', and a size of '123 B'. The response is displayed in the 'Body' tab, which is currently empty, indicating a successful deletion. The interface includes various tabs for request configuration (Params, Authorization, Headers, Body, Pre-request Script, Tests, Settings) and response viewing (Body, Cookies, Headers, Test Results).

Testovací scénář 2. – metoda DELETE - kontrola odstranění studenta z databáze s id = 2417

Kroky:

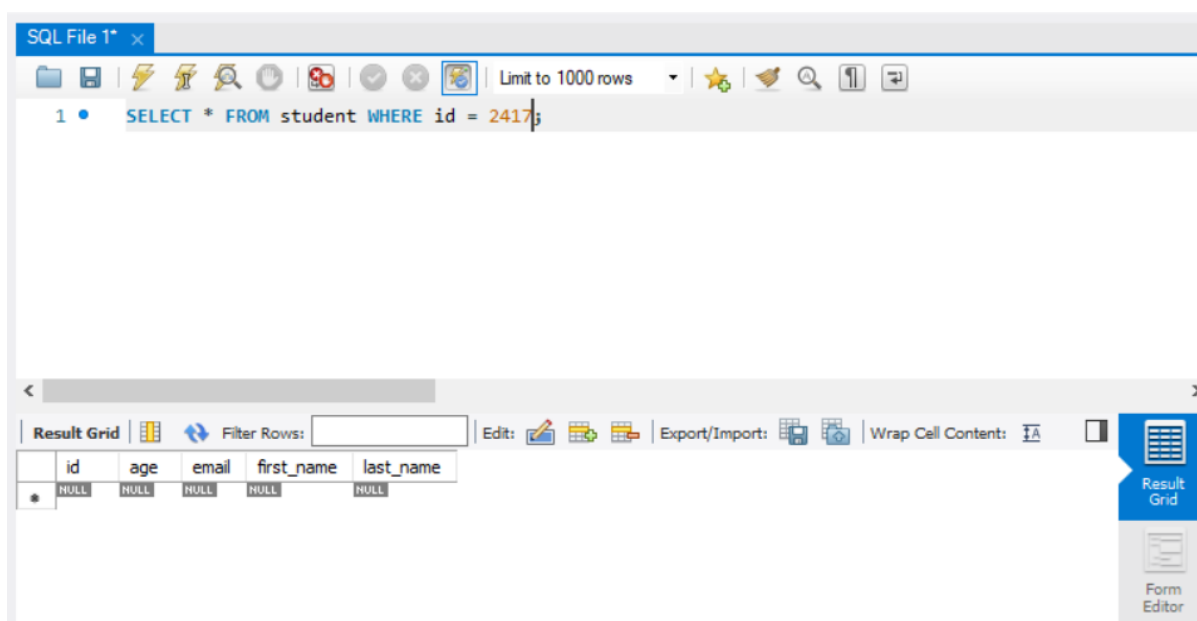
1. Otevřel jsem si program MySQL Workbench
2. Zadal jsem přihlašovací údaje do databáze:
Hostname: aws.connect.psdb.cloud
Port: 3306
Username: [k9ik3tg4x7hdxk68nlqy](#)
Password: [pscale_pw_H9L99PYMIpZMkfuCbuZDu24irPmupXQvZEtg5pt5vtG](#)
Default scheme: [qa_demo](#)
3. Do příkazového pole jsem zadala následující příkaz:
`SELECT * FROM student WHERE id = 2417;`
4. Klikl na ikonku blesku

Očekávaný výsledek:

7. Aplikace nenajde námi požadovaného studenta

Aktuální výsledek:

5. Aplikace nám nenašla námi požadovaného studenta



Testovací scénář 3. – metoda DELETE – odstranění studenta s ID = 2415

Kroky:

1. Otevřel jsem si program Postman
2. Do příkazového řádku jsem zadal adresu: <http://108.143.193.45:8080/api/v1/students/>
3. Vyberal jsem metodu DELETE
4. Kliknul na tlačítko BODY, zaškrtnul none
5. Do příkazového řádku za poslední lomítko jsem přidal id studenta:
<http://108.143.193.45:8080/api/v1/students/id = 2415>

Očekávaný výsledek:

1. Aplikace zhlásila chybu
2. Status kód byl 400 Bad Request

The screenshot shows the Postman interface. At the top, the method is set to **DELETE** and the URL is `http://108.143.193.45:8080/api/v1/students/id = 2415`. The **Send** button is visible. Below the URL bar, the **Params** tab is selected, showing a table with one row:

Key	Value	Bulk Edit
Key	Value	

. The **Body** tab is also visible. At the bottom, the **Body** tab is selected, showing the response in **JSON** format. The response is a JSON object with the following fields:

```
1 {
2   "timestamp": "2024-11-26T19:29:08.151+00:00",
3   "status": 400,
4   "error": "Bad Request",
5   "message": "",
6   "path": "/api/v1/students/id%20=%202415%20"
7 }
```

BUG REPORT

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

Bug report metody GET

Testovací scénář 3. - metoda GET zadání neexistujícího studenta podle ID

Název:

1. V případě zadání neexistujícího studenta nám byl vrácen kód 500 Internal Error místo 404 Not Found

Kroky reprodukce:

1. Otevřel jsem si program Postman
2. Do příkazového řádku zadal adresu: <http://108.143.193.45:8080/api/v1/students/3680>
3. Vybral metodu GET a odeslal

Očekávaný výsledek:

1. Aplikace zahlásí chybu
2. 404 Not Found

Aktuální výsledek:

1. Aplikace zhlásila chybu
2. 500 Internal Server Error

GET

▼

http://108.143.193.45:8080/api/v1/students/3680

Send

▼

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests ●

Settings

Cookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

Body

Cookies

Headers (4)

Test Results

🌐

Status: 500 Internal Server Error

Time: 192 ms

Size: 288 B

Save Response ▼

PrettyRawPreviewVisualizeJSON ▼

1

2

3

4

5

6

7

```
"timestamp": "2024-11-24T19:33:45.030+00:00",
"status": 500,
"error": "Internal Server Error",
"message": "",
"path": "/api/v1/students/3680"
```

Bug report metody POST

Testovací scénář 4. – metoda POST zadání číslíc do parametru firstName

Název:

1. V případě zadání číselné hodnoty do firstName nám nebyla vrácená chyba a status kód 400 Bad Request. Místo toho byl vytvořen nový student a vrátil se nám kód 200 OK

Kroky reprodukce:

1. Otevřel jsem si program Postman
2. Do příkazového řádku jsem zadal adresu: <http://108.143.193.45:8080/api/v1/students/>
3. Vyberal jsem metodu POST
4. Kliknul na tlačítko BODY, zaškrtnul raw a TEXT změnil na JSON
5. Vytvořil nového studenta s těmito parametry

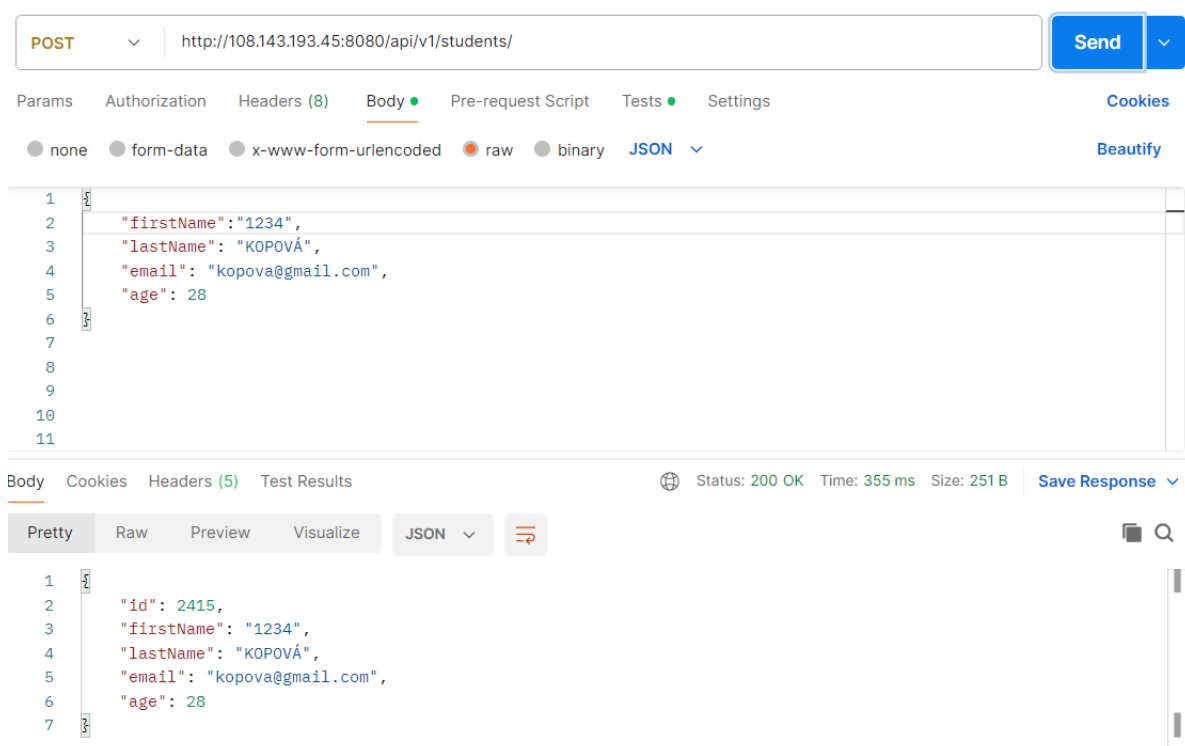
```
{
  "firstName": "1234",
  "lastName": "KOPOVÁ",
  "email": "kopova@gmail.com",
  "age": 28
}
```

Očekávaný výsledek:

1. Aplikace zahlásí chybu
2. Status kód bude 400 Bad Request

Aktuální výsledek:

1. Aplikace vytvořila nového studenta a přiřadila mu id 2415
2. Status kód byl 200 OK



Testovací scénář 6. – metoda POST zadání speciálních znaků do parametru lastName

Název:

1. V případě zadání speciálních znaků do lastName nám nebyla vrácená chyba a status kód 400 Bad Request. Místo toho byl vytvořen nový student a vrátil se nám kód 200 OK

Kroky reprodukce:

1. Otevřel jsem si program Postman
2. Do příkazového řádku jsem zadal adresu: <http://108.143.193.45:8080/api/v1/students/>
3. Vyberal jsem metodu POST
4. Kliknul na tlačítko BODY, zaškrtnul raw a TEXT změnil na JSON
5. Vytvořil nového studenta s těmito parametry

```
{
  "firstName": "Ivona",
  "lastName": "#&@*/+",
  "email": "makalova@gmail.com",
  "age": 37
}
```

Očekávaný výsledek:

1. Aplikace zahlásí chybu
2. Status kód bude 400 Bad Request

Aktuální výsledek:

1. Aplikace vytvořila nového studenta a přiřadila mu id 2417
2. Status kód byl 200 OK

POST

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

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

JSON

Beautify

```
1 {
2   "firstName": "Ivona",
3   "lastName": "#&@*/+",
4   "email": "makalova@gmail.com",
5   "age": 37
6 }
7
```

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Time: 319 ms

Size: 253 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

```
"id": 2417,
"firstName": "Ivona",
"lastName": "#&@*/+",
"email": "makalova@gmail.com",
"age": 37
```