

FINÁLNÍ PROJEKT

č.1



ENGETO

Autor: Ing. Pavel Vejvoda

Datum: 18.07.2024

Obsah

| | |
|-------------------------|----|
| ZADÁNÍ..... | 3 |
| Přístupové údaje: | 3 |
| Poznámky: | 3 |
| TESTOVACÍ SCÉNÁŘE | 4 |
| 1. GET Metoda..... | 4 |
| 2. DELETE Metoda | 6 |
| 3. POST Metoda..... | 7 |
| EXEKUCE TESTŮ | 10 |
| 1. GET Metoda..... | 10 |
| 2. DELETE Metoda | 14 |
| 3. POST Metoda..... | 18 |
| BUG REPORT | 26 |
| ZÁVĚR..... | 27 |

ZADÁNÍ

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

Data studentů zahrnují atributy jako ID, věk, jméno, příjmení a email. Projekt zahrnuje testování tří hlavních metod API: GET, DELETE a POST. Důraz je kladen na validaci vstupních dat, správné ukládání do databáze a správné zpracování požadavků a odpovědí.

V projektu pro metody platí, že: metoda GET zobrazí data o existujícím studentovi metoda POST založí záznam o studentovi a metoda DELETE smaže data o studentovi

Přístupové údaje:

| | |
|----------|--|
| Databáze | database: qa_demo Host: aws.connect.psdb.cloud Username: 1mnbj1bcf5pzc5skytir Password: XXX „z důvodu sdílení na gitHub heslo není uvedeno“ |
| REST-API | http://108.143.193.45:8080/api/v1/students/ |

Poznámky:

Pro metodu GET byly použity níže uvedené scénáře.

GET Metoda

1. **Happy-Day Scénáře**
2. **Unhappy-Day Scénáře**

Pro metodu DELETE byly použity níže uvedené scénáře.

DELETE Metoda

1. **Happy-Day Scénáře**
2. **Unhappy-Day Scénáře**

Pro metodu POST byly použity níže uvedené scénáře.

POST Metoda

1. **Happy-Day Scénáře**
2. **Unhappy-Day Scénáře**

Dále byly pro metody GET, DELETE a POST použity níže popsané funkční a nefunkční testy.

Funkční vs. Nefunkční testy

1. **Funkční testy (black box):**
 - Ověřit, zda aplikace splňuje specifikované funkční požadavky.
 - Kontrola, zda je možné přidat nového studenta s validními údaji (POST).
 - Získat existujícího studenta podle ID (GET).
 - Smazat studenta podle ID (DELETE).
2. **Nefunkční testy (white box):**
 - Ověřit, zda aplikace splňuje nefunkční požadavky, jako jsou bezpečnost, výkon, použitelnost.
 - Testování nevalidních emailových adres (POST).
 - Zajištění, že DELETE bez ID nevymaže celou databázi (DELETE).
 - Kontrola, že škodlivý kód se neprovede (POST s malicious daty).

TESTOVACÍ SCÉNÁŘE

1. GET Metoda

Happy-Day Scénáře

Scénář 1: Úspěšné získání existujícího studenta podle ID

- **Předpoklad:** Student s ID 600 existuje v databázi a obsahuje validní data.
- **Kroky:**
 1. Odeslat GET požadavek na
`http://108.143.193.45:8080/api/v1/students/600`
 2. Zkontrolovat, zda odpověď obsahuje správné údaje (ID, age, firstName, lastName, email).
 3. Ověřit, že status kód odpovědi je 200 OK.
- **Očekávaný výsledek:** Odpověď obsahuje data studenta s ID 600, všechny atributy a data dávají logický smysl (např. email obsahující '@')

Scénář 2: Získání všech studentů

- **Předpoklad:** V databázi existuje několik studentů.
- **Kroky:**
 1. Odeslat GET požadavek na `http://108.143.193.45:8080/api/v1/students`
 2. Zkontrolovat, zda odpověď obsahuje seznam všech studentů.
 3. Ověřit, že status kód odpovědi je 200 OK.
- **Očekávaný výsledek:** Odpověď obsahuje seznam všech studentů a status kód je 200 OK.

Unhappy-Day Scénáře

Scénář 3: Získání neexistujícího studenta podle ID

- **Předpoklad:** Student s ID 123456789 neexistuje v databázi.
- **Kroky:**
 1. Odeslat GET požadavek na
`http://108.143.193.45:8080/api/v1/students/123456789`
 2. Zkontrolovat odpověď.

3. Ověřit, že status kód odpovědi je 404 Not Found.
- **Očekávaný výsledek:** Odpověď, že student s ID=123456789 není v databázi nebo podobná hláška.

2. DELETE Metoda

Happy-Day Scénáře

Scénář 1: Úspěšné smazání existujícího studenta

- **Předpoklad:** Student s ID 266 existuje v databázi.
- **Kroky:**
 1. Odeslat DELETE požadavek na
`http://108.143.193.45:8080/api/v1/students/266`
 2. Zkontrolovat, zda odpověď je 200 OK nebo podobná úspěšná zpráva.
 3. Odeslat GET požadavek na
`http://108.143.193.45:8080/api/v1/students/266`, aby se ověřilo,
že student byl skutečně smazán.
 4. Ověřit, že status kód odpovědi je 200 OK.
- **Očekávaný výsledek:** DELETE odpověď je úspěšná, GET požadavek vrátí 404 Not Found nebo chybu 500 Internal Server Error a status kód DELETE odpovědi je 200 OK.

Unhappy-Day Scénáře

Scénář 2: Smazání neexistujícího studenta (nebo již smazaného studenta)

- **Předpoklad:** Student s ID 1 neexistuje v databázi.
- **Kroky:**
 1. Odeslat DELETE požadavek na
`http://108.143.193.45:8080/api/v1/students/1`
 2. Zkontrolovat odpověď.

3. Ověřit, že status kód odpovědi je 404 Not Found.
- **Očekávaný výsledek:** Odpověď je 404 Not Found nebo vhodná chybová zpráva a status kód je 404.

Scénář 3: Smazání studenta bez ID

- **Předpoklad:** Zkontrolovat odpověď při smazání bez specifikace ID.
- **Kroky:**
 1. Odeslat DELETE požadavek na
`http://108.143.193.45:8080/api/v1/students/`
 2. Zkontrolovat odpověď.
 3. Ověřit, že status kód odpovědi je 400 Bad Request nebo 405 Method Not Allowed.
- **Očekávaný výsledek:** Odpověď je 400 Bad Request nebo vhodná chybová zpráva a status kód je 400 nebo 405.

3. POST Metoda

Happy-Day Scénáře

Scénář 1: Úspěšné přidání nového studenta

- **Předpoklad:** Validní údaje nového studenta.
- **Kroky:**

1. Odeslat POST požadavek na /students s tělem obsahujícím validní údaje.
 2. Zkontrolovat odpověď a ověřit, že nový student byl přidán.
 3. Odeslat GET požadavek na /students a zkontrolovat, zda nový student je v seznamu a všechny atributy dávají logický smysl.
 4. Ověřit, že status kód odpovědi je 201 Created.
- **Očekávaný výsledek:** Odpověď je úspěšná, nový student je uložen a status kód je 201 Created.

Unhappy-Day Scénáře

Scénář 2: Přidání studenta s nevalidními údaji

- **Předpoklad:** Nevalidní údaje (např. chybějící povinné pole).
- **Kroky:**
 1. Odeslat POST požadavek na `http://108.143.193.45:8080/api/v1/students/` s tělem obsahujícím nevalidní údaje (např. chybějící `firstName` a `lastName`).
 2. Zkontrolovat odpověď.
 3. Ověřit, že status kód odpovědi je 400 Bad Request.
- **Očekávaný výsledek:** Odpověď je 400 Bad Request nebo vhodná chybová zpráva a status kód je 400.

Scénář 3: Přidání studenta s nevalidní emailovou adresou (bez zavináče):

- **Předpoklad:** Nevalidní emailová adresa (bez zavináče).
- **Kroky:**
 1. Odeslat POST požadavek na /students s tělem obsahujícím nevalidní email (např. `{"firstName": "jan", "lastName": "novak", "email": "jan.novakexample.com"}`).
 2. Zkontrolovat odpověď.
 3. Ověřit, že status kód odpovědi je 400 Bad Request.
- **Očekávaný výsledek:** Odpověď je 400 Bad Request nebo vhodná chybová zpráva a status kód je 400.

Scénář 4: Přidání studenta s nevalidními (malicious) údaji

- **Předpoklad:** Nevalidní a potenciálně škodlivé údaje.

- **Kroky:**

1. Odeslat POST požadavek na /students s tělem obsahujícím nevalidní nebo škodlivé údaje.
2. Zkontrolovat odpověď.
3. Ověřit, že status kód odpovědi je 400 Bad Request nebo 422 Unprocessable Entity.
4. Ověřit, že se škodlivý kód neprovede a že data nejsou uložena do databáze.

- **Očekávaný výsledek:** Odpověď je 400 Bad Request nebo 422 Unprocessable Entity, status kód je 400 nebo 422, škodlivý kód se neprovede a data nejsou uložena do databáze.

Tento scénář přidává testování pro situace, kdy se někdo pokusí zneužít API zasláním škodlivých dat. Takové testy jsou důležité pro zajištění bezpečnosti aplikace a databáze.

EXEKUCE TESTŮ

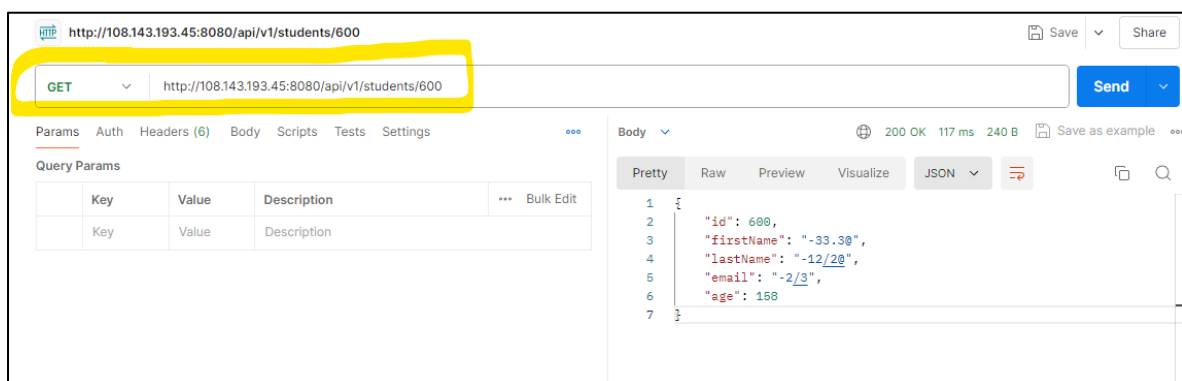
1. GET Metoda

Happy-Day Scénáře

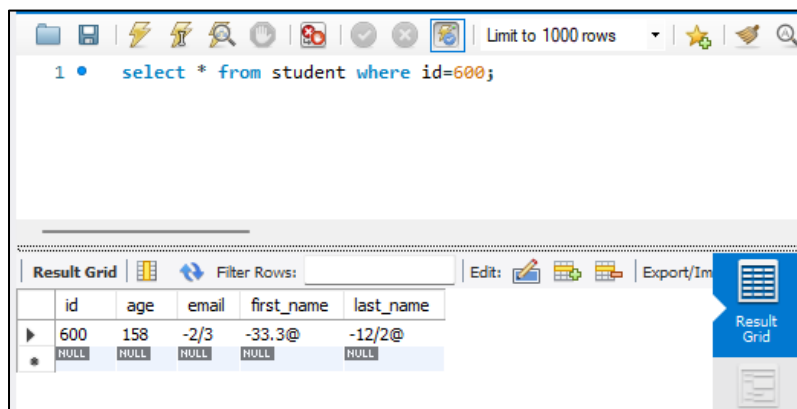
Scénář 1: Úspěšné získání existujícího studenta podle ID 600

- **Předpoklad:** Student s ID 600 existuje v databázi a obsahuje validní data.
- **Kroky:**
 1. Otevřít aplikaci Postman
 2. Otevřít nový požadavek http a zvolit metodu GET
 3. Odeslat GET požadavek na
`http://108.143.193.45:8080/api/v1/students/600`
 4. Zkontrolovat, zda odpověď obsahuje správné údaje (ID, age, firstName, lastName, email) a zda atributy a data dávají smysl.
 5. Ověřit, že status kód odpovědi je 200 OK.

Screen Postman:



Screen Workbench:

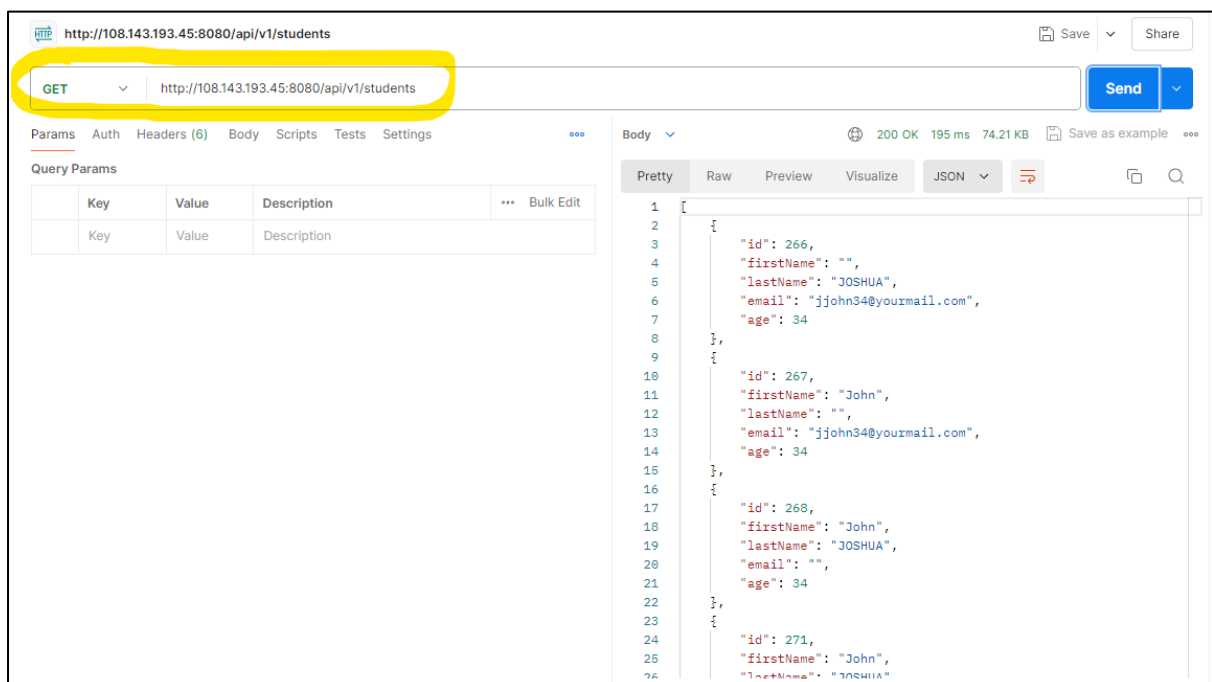


- **Skutečný výsledek:** Odpověď obsahuje data studenta s ID 600, ale ne všechny atributy a data dávají logický smysl: firstName, lastName a email obsahují znaky neodpovídající skutečností.
- **Očekávaný výsledek:** Odpověď obsahuje data studenta s ID 600, všechny atributy a data dávají logický smysl (např. email obsahující '@' a pole name obsahující text)
- **Návrh řešení:** Ověřit u zadavatele akceptaci vhodných znaků v poli a tyto znaky definovat, aby v případě nevhodných znaků aplikace napsala chybovou hlášku.

Scénář 2: Získání všech studentů

- **Předpoklad:** V databázi existuje několik studentů.
- **Kroky:**
 1. Otevřít aplikaci Postman
 2. Otevřít nový požadavek http a zvolit metodu GET
 3. Odeslat GET požadavek na `http://108.143.193.45:8080/api/v1/students`
 4. Zkontrolovat, zda odpověď obsahuje seznam všech studentů.
 5. Ověřit, že status kód odpovědi je 200 OK.

Screen Postman:



Screen Workbench:

The screenshot shows the SQL Workbench interface. At the top, there's a tab labeled 'Query 1'. Below it, a toolbar contains various icons for file operations, execution, and settings. A dropdown menu shows 'Limit to 1000 rows'. The main query editor contains the SQL statement: `1 • select * from student;`. On the right side, there's a panel titled 'SQLAdditions' with a message: 'Automatic disabled. Use manually g current care toggle au'. Below the query editor, there's a 'Result Grid' section. It includes a 'Filter Rows' input field and buttons for 'Edit', 'Export/Import', 'Form Editor', and 'Field Types'. The 'Result Grid' itself displays a table with the following data:

| | id | age | email | first_name |
|---|-----|-----|----------------------|-------------------------------------|
| ▶ | 268 | 34 | | John |
| | 271 | 34 | 548 | John |
| | 283 | 170 | polusa@gmail. | PoPoLUska |
| | 285 | 105 | pnov@sda.com | KrungThepMahanakhonAmonRattanakosin |
| | 287 | 67 | skupina@jedna.cz | Marek |
| | 288 | 67 | skupina@jednacz | LADISLAV |
| | 289 | 999 | skupina?jedna.cz | AI |
| | 290 | 93 | amesnard1@tund1.de | A. |
| | 291 | 11 | adelsvidron@mail.com | Adel |
| | 294 | 20 | 4077@... | Polina Frodla |

At the bottom of the interface, there's a tab labeled 'student 16' and buttons for 'Apply', 'Revert', 'Context Help', and 'Snip'.

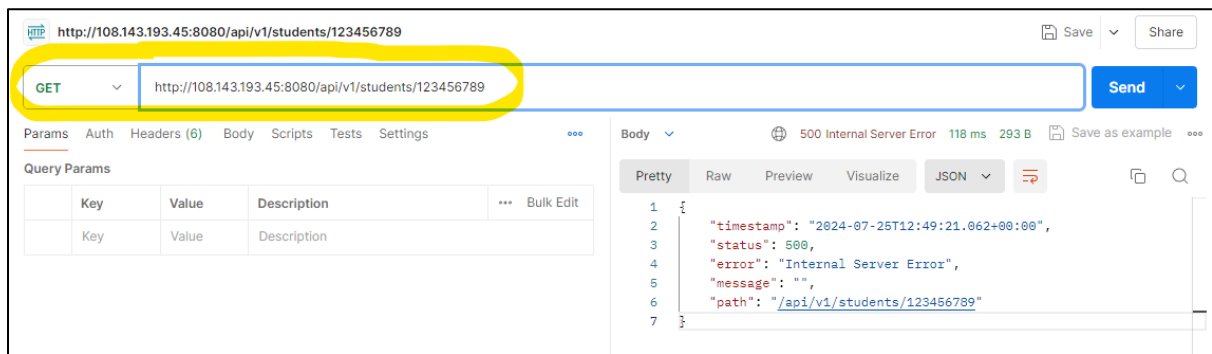
- **Skutečný výsledek:** Odpověď obsahuje seznam všech studentů a status kód je 200 OK.
- **Očekávaný výsledek:** Odpověď obsahuje seznam všech studentů a status kód je 200 OK.
- **Návrh řešení:** Není potřeba návrh řešení. Test pro zobrazení celkového seznamu studentů prošel v pořádku.

Unhappy-Day Scénáře

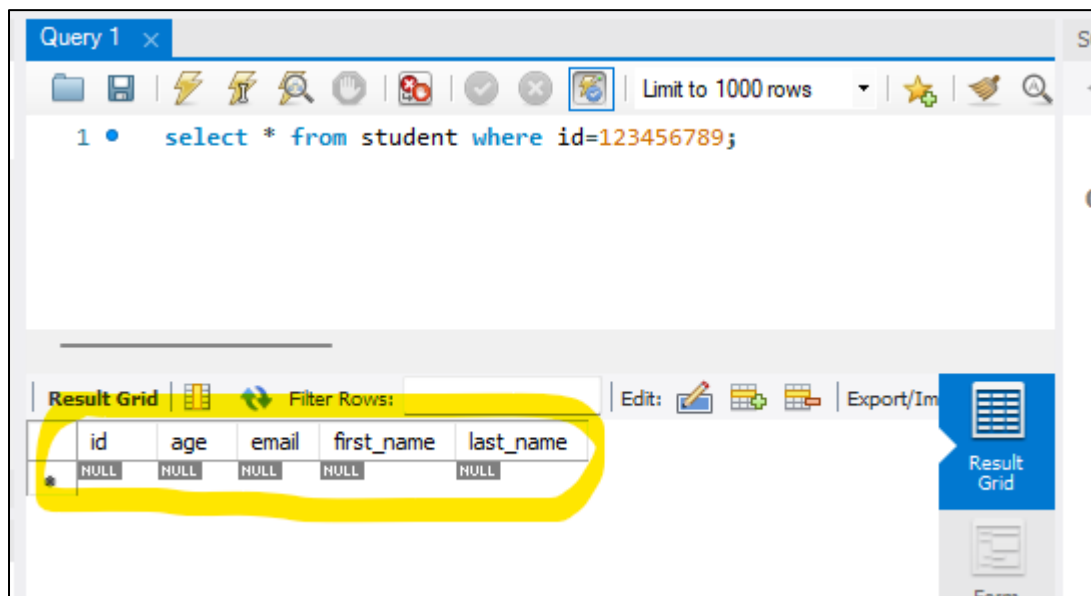
Scénář 3: Získání neexistujícího studenta podle ID

- **Předpoklad:** Student s ID 123456789 neexistuje v databázi.
- **Kroky:**
 1. Otevřít aplikaci Postman
 2. Otevřít nový požadavek http a zvolit metodu GET
 3. Odeslat GET požadavek na
`http://108.143.193.45:8080/api/v1/students/123456789`
 4. Zkontrolovat odpověď.
 5. Ověřit, že status kód odpovědi je 404 Not Found.

Screen Postman:



Screen Workbench:



- **Skutečný výsledek:** V Postman se zobrazí se status code 500. Ve Workbench se zobrazí output „zelena fajfka“, ale bylo by vhodnější zobrazení napsat hlášku „výstup proběhl v pořádku, ale student neexistuje“.
- **Očekávaný výsledek:** Odpověď, že student s ID=123456789 není v databázi nebo podobná hláška. Standartně by se měla zobrazit chyba 404 – neexistující student.
- **Návrh řešení:** Zde na zváženou, zda je status code 500 v pořádku, nebo by se měl zobrazit status code 404. Porada s vývojářem.

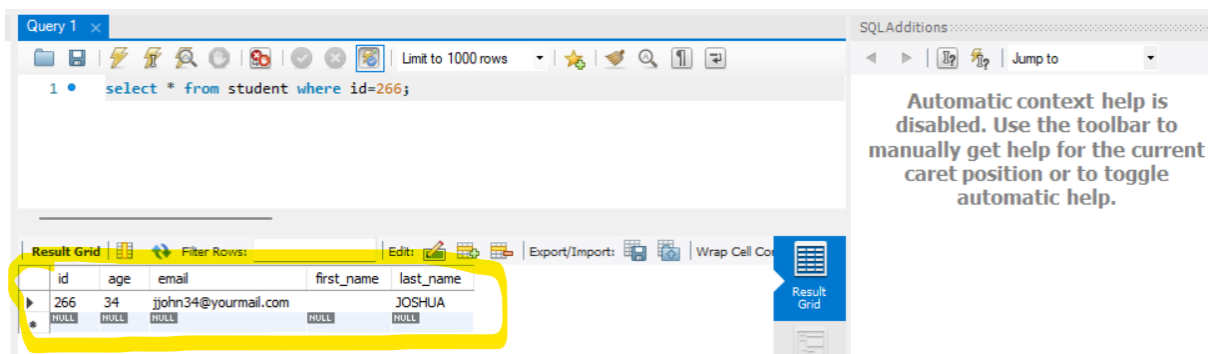
2. DELETE Metoda

Happy-Day Scénáře

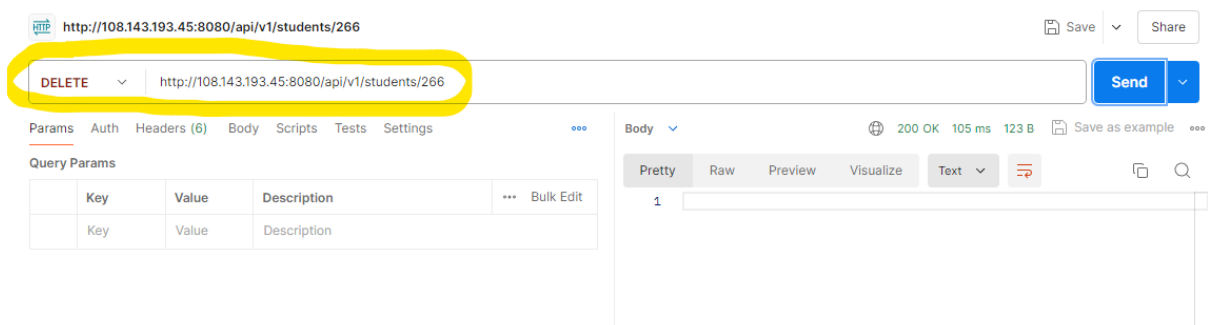
Scénář 1: Úspěšné smazání existujícího studenta

- **Předpoklad:** Student s ID 266 existuje v databázi.
- **Kroky:**
 1. Otevřít aplikaci Postman
 2. Otevřít nový požadavek http a zvolit metodu DELETE
 3. Odeslat DELETE požadavek na
`http://108.143.193.45:8080/api/v1/students/266`
 4. Zkontrolovat, zda odpověď je 200 OK nebo podobná úspěšná zpráva.
 5. Odeslat GET požadavek na
`http://108.143.193.45:8080/api/v1/students/266`, aby se ověřilo,
že student byl skutečně smazán.
 6. Ověřit, že status kód odpovědi je 200 OK.

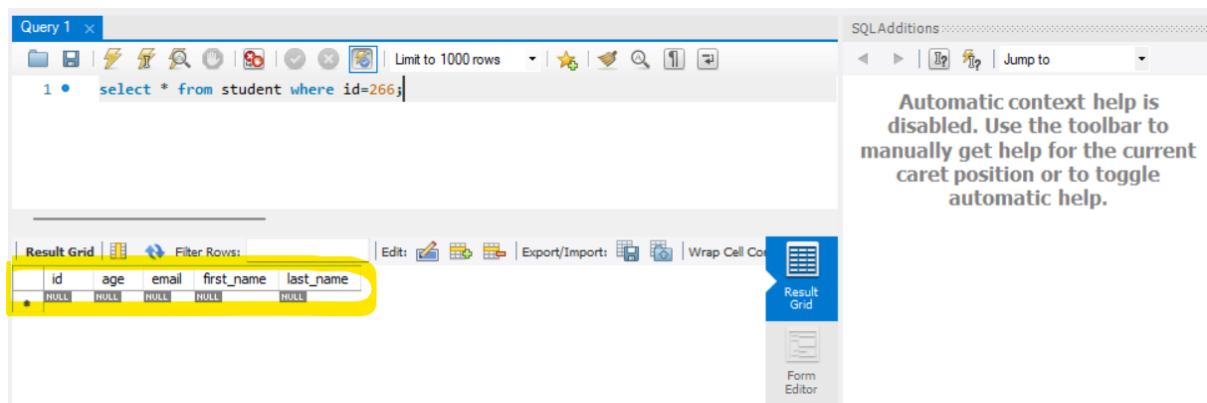
Databáze Workbench před smazáním:



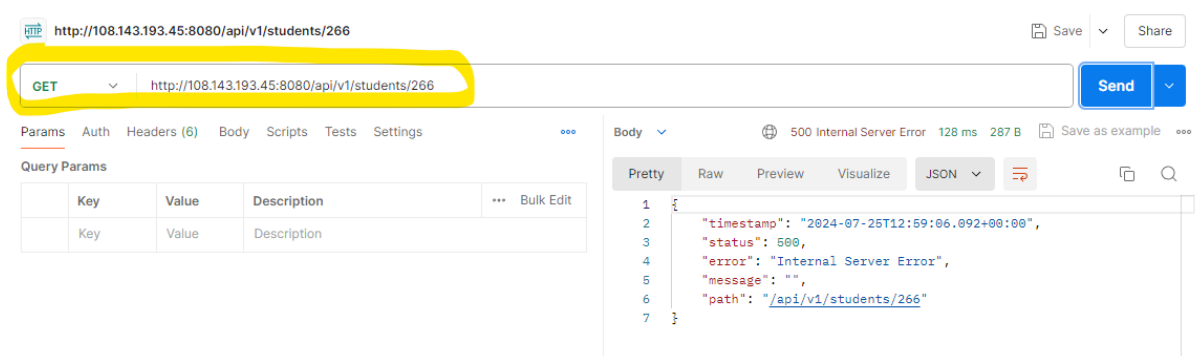
Příkaz DELETE v Postmanovi ke smazání studenta s ID266:



Databáze Workbench po smazání:



Ověření, zda je student smazán pomocí metody GET v Postmanovi:



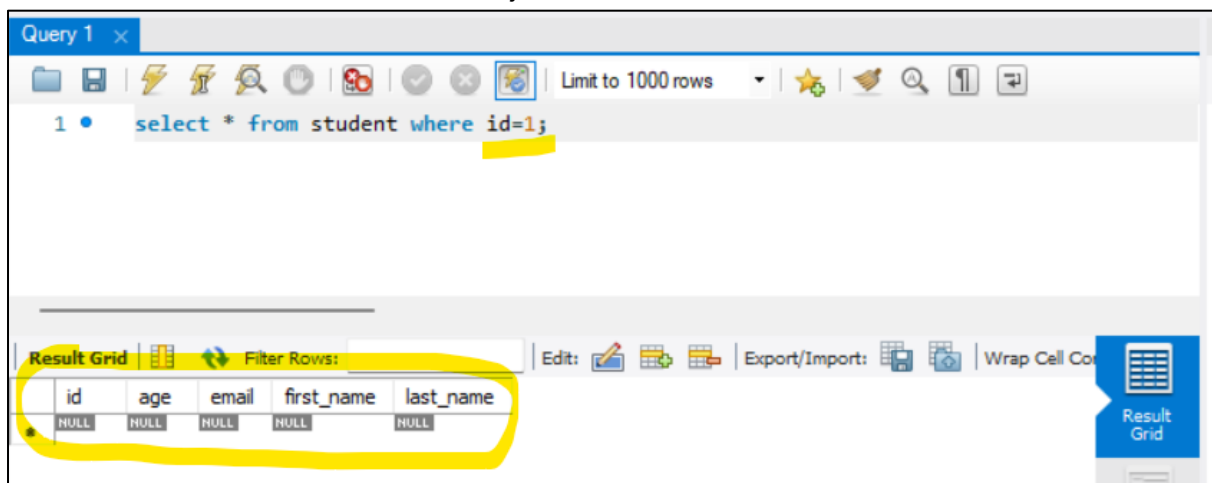
- **Skutečný výsledek:** Student se úspěšně smazal.
- **Očekávaný výsledek:** DELETE odpověď je úspěšná, GET požadavek vrací 404 Not Found nebo chybu 500 Internal Server Error, což je v pořádku, protože se student smazal a nyní ho metoda GET nezíská, a status kód DELETE odpovědi je 200 OK.
- **Návrh řešení:** Zde není žádný návrh na možné řešení. Zdá se, že metoda funguje bez větších obtíží a nejasností.

Unhappy-Day Scénáře

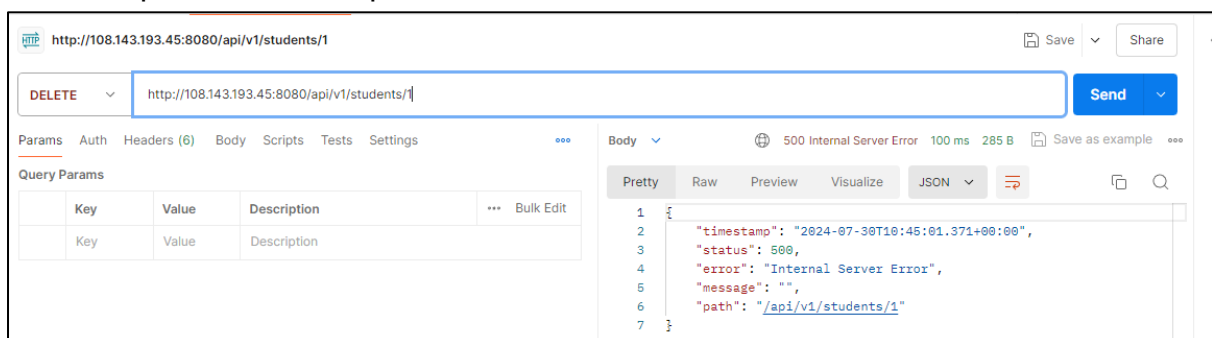
Scénář 2: Smazání neexistujícího (nebo již smazaného) studenta

- **Předpoklad:** Student s ID 1 neexistuje v databázi.
- **Kroky:**
 1. Otevřít aplikaci Postman
 2. Otevřít nový požadavek http a zvolit metodu DELETE
 3. Odeslat DELETE požadavek na `http://108.143.193.45:8080/api/v1/students/1`
 4. Zkontrolovat odpověď.
 5. Ověřit, že status kód odpovědi je 404 Not Found.

Databáze Workbench, kde neexistuje student s id=1:



Postman příkaz DELETE pro smazání studenta s ID1:

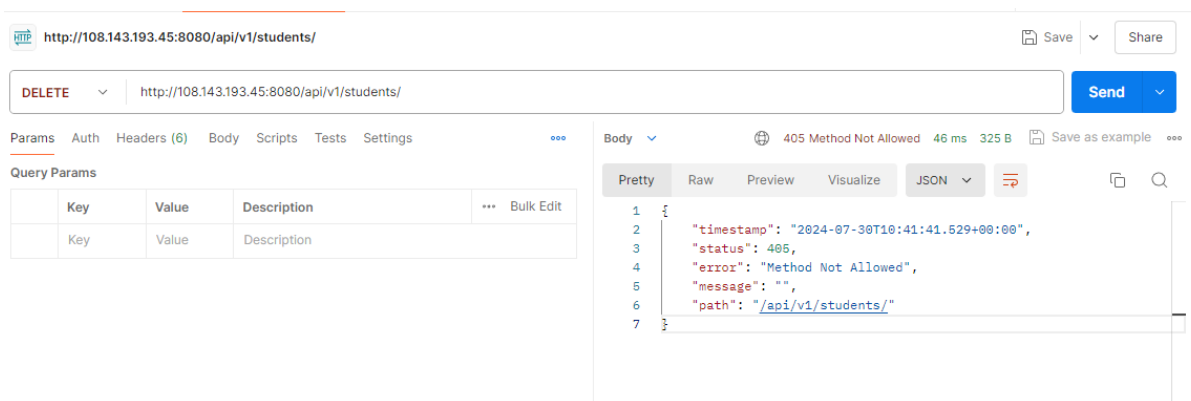


- **Skutečný výsledek:** V databázi Workbench je stav odpovídající. V Postmanovi se zobrazuje status code 500 Internal Server Error.
- **Očekávaný výsledek:** Ve Workbench se tento student nenachází. V Postmanovi by se měl zobrazit status code 404 Not Found.
- **Návrh řešení:** Ideálně by se v Postmanovi měla napsat chybová hláška, že se tento student v databázi nenachází a status code by měl být 404.

Scénář 3: Smazání studenta bez ID

- **Předpoklad:** Zkontrolovat odpověď při smazání bez specifikace ID.
- **Kroky:**
 1. Otevřít aplikaci Postman
 2. Otevřít nový požadavek http a zvolit metodu DELETE
 3. Odeslat DELETE požadavek na
`http://108.143.193.45:8080/api/v1/students/`
 4. Zkontrolovat odpověď.
 5. Ověřit, že status kód odpovědi je 404 Not Found nebo 405 Method Not Allowed.

Screenshot Postman:



- **Skutečný výsledek:** V Postmanovi se zobrazuje status code 405 Internal Server Error.
- **Očekávaný výsledek:** Odpověď bude 404 Not Found nebo vhodná chybová zpráva, že nelze smazat celou databázi a je potřeba zadat ID.
- **Návrh řešení:** Ideálně by se v Postmanovi měla napsat chybová hláška, že nelze smazat celou databázi a je třeba zadat ID.

3. POST Metoda

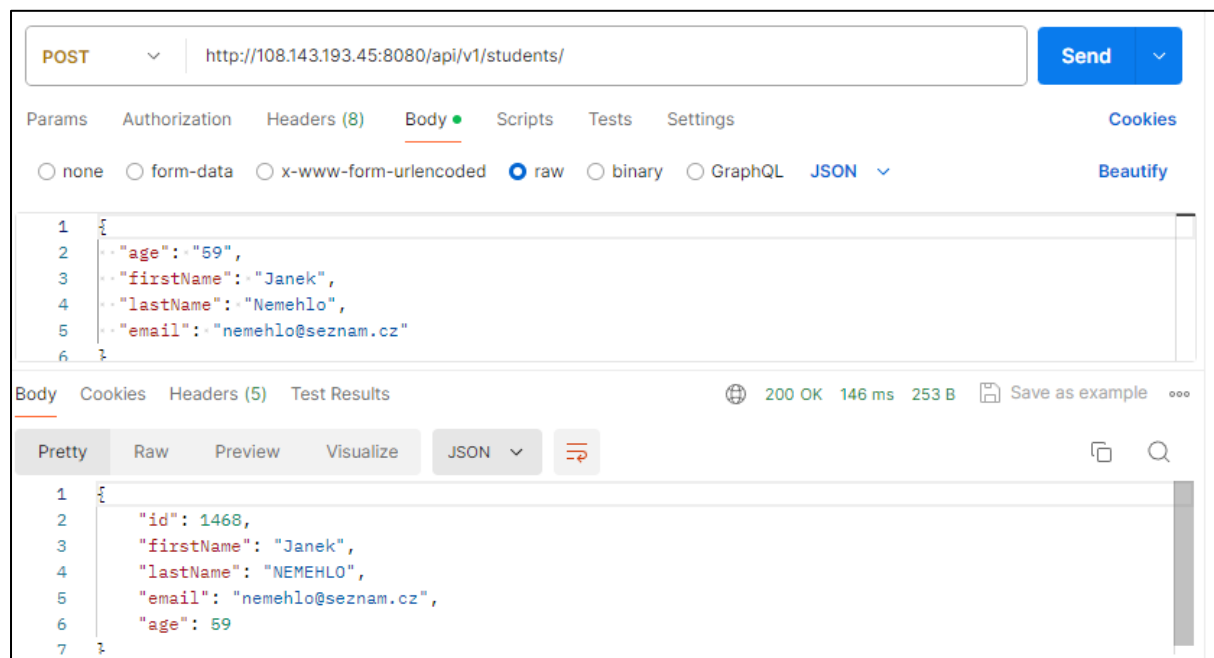
- Poznámka: vše by se měl ukládat jako lowerCase (malými písmeny)

Happy-Day Scénáře

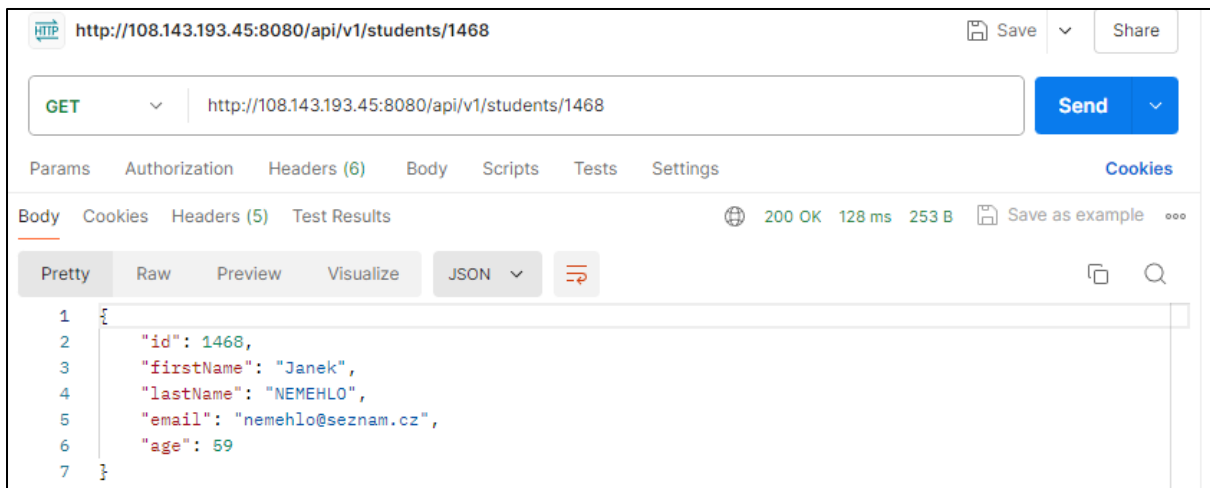
Scénář 1: Úspěšné přidání nového studenta

- **Předpoklad:** Validní údaje nového studenta.
- **Kroky:**
 1. Otevřít aplikaci Postman
 2. Otevřít nový požadavek http a zvolit metodu POST
 3. Odeslat POST požadavek na
http://108.143.193.45:8080/api/v1/students/ s tělem obsahujícím validní údaje.
 4. Zkontrolovat odpověď a ověřit, že nový student byl přidán.
 5. Odeslat GET požadavek na /students/1468 a zkontrolovat, zda nový student je v seznamu.
 6. Ověřit, že status kód odpovědi je 201 Created nebo 200 OK.

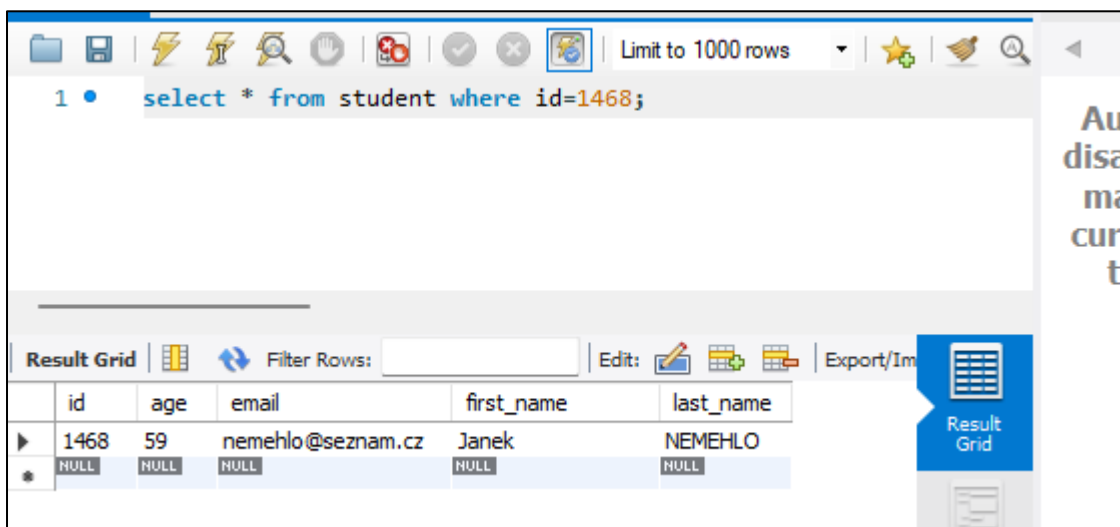
Implementace nového studenta:



Zjištění pomocí metody GET, zda se student zařadil do databáze:



Shlédnutí databáze, zda je student s id=1468 v databázi správně:



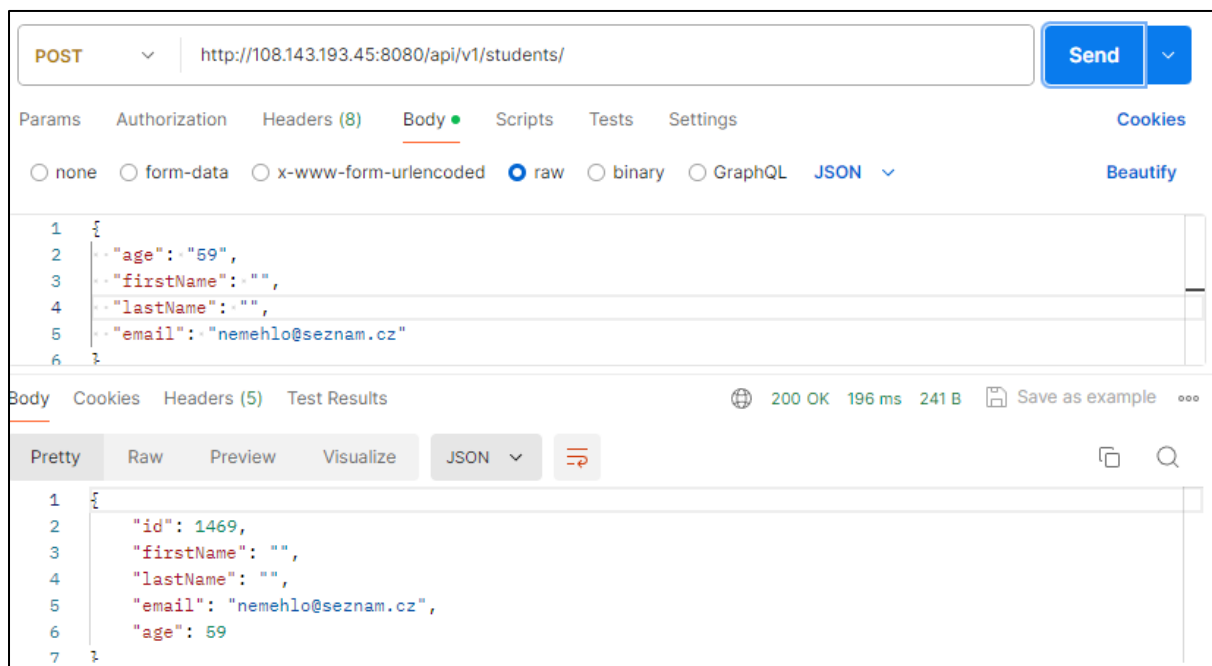
- **Skutečný výsledek:** V Části lastName dojde ke změně velikosti písma na velké, namísto lowerCase.
- **Očekávaný výsledek:** Status code v Postmanovi bude 201 OK a student se založí do databáze Workbench s lowerCase písmeny a přesně zadanými údaji.
- **Návrh řešení:** Chyba v lowerCase pro zapsání databáze. Navrhnout vývojáři opravu při zápisu do databáze.

Unhappy-Day Scénáře

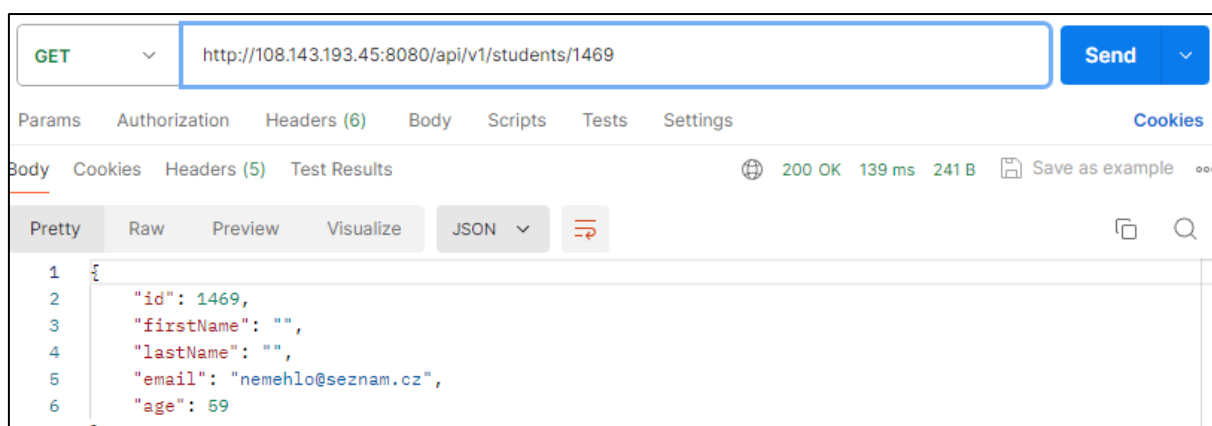
Scénář 2: Přidání studenta s nevalidními údaji

- **Předpoklad:** Nevalidní údaje (např. chybějící povinné pole).
- **Kroky:**
 1. Otevřít aplikaci Postman
 2. Otevřít nový požadavek http a zvolit metodu POST
 3. Odeslat POST požadavek na
`http://108.143.193.45:8080/api/v1/students/` s tělem obsahujícím nevalidní údaje (např. chybějící firstName a lastName).
 4. Zkontrolovat odpověď, kde jsme zjistili, že bylo přiděleno ID=1469.
 5. Ověřit, že status kód odpovědi je 400 Bad Request.

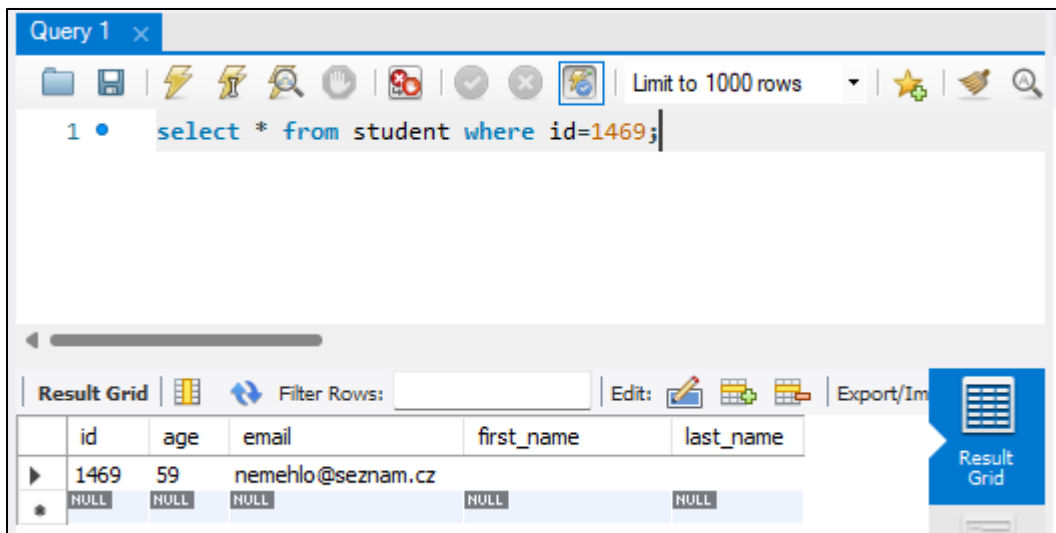
Implementace studenta bez validního jména a příjmení:



Shlédnutí databáze pomocí metody GET a získání studenta s ID=1469:



Shlédnutí ID=1469 v databázi Workbench:

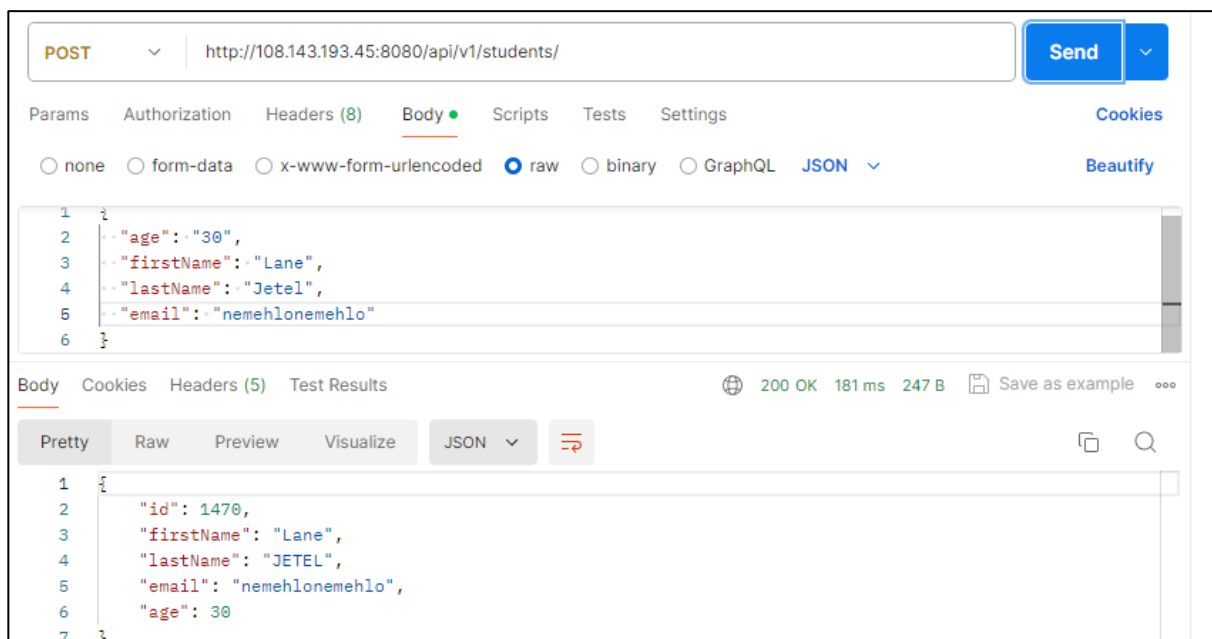


- **Skutečný výsledek:** V Postmanovi se zobrazí status code 200 OK, že je vše v pořádku, ale při prázdných polích name by se mělo zobrazit, že se takový student nemůže uložit. Studentovi bez nadefinovaných polích bylo přiděleno ID a následně byl propsán do Workbench databáze.
- **Očekávaný výsledek:** Předpoklad, že se student s prázdnými poli neuloží a program vypíše výzvu k zadání firstName a lastName. Nezobrazí se student v databázi Workbench, jelikož by se takový student neměl vůbec uložit.
- **Návrh řešení:** Implementace ochrany a hlášky, že nelze uložit studenta bez definovaných polích firstName a lastName.

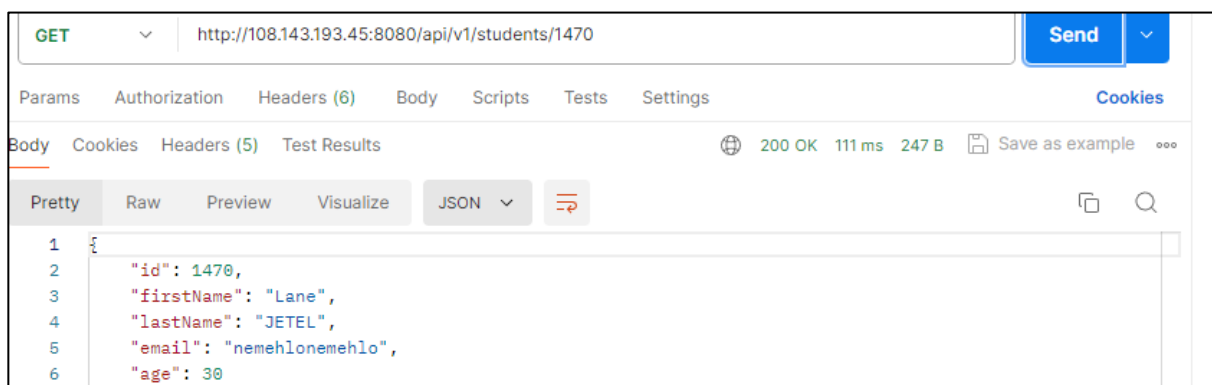
Scénář 3: Přidání studenta s nevalidní emailovou adresou (bez zavináče):

- **Předpoklad:** Nevalidní emailová adresa (bez zavináče).
- **Kroky:**
 1. Otevřít aplikaci Postman
 2. Otevřít nový požadavek http a zvolit metodu POST
 3. Odeslat POST požadavek na
http://108.143.193.45:8080/api/v1/students/ s tělem obsahujícím
nevalidní email
 4. Zkontrolovat odpověď.
 5. Ověřit, že status kód odpovědi je 400 Bad Request a hlášku se špatně
zadaným formátem emailu.

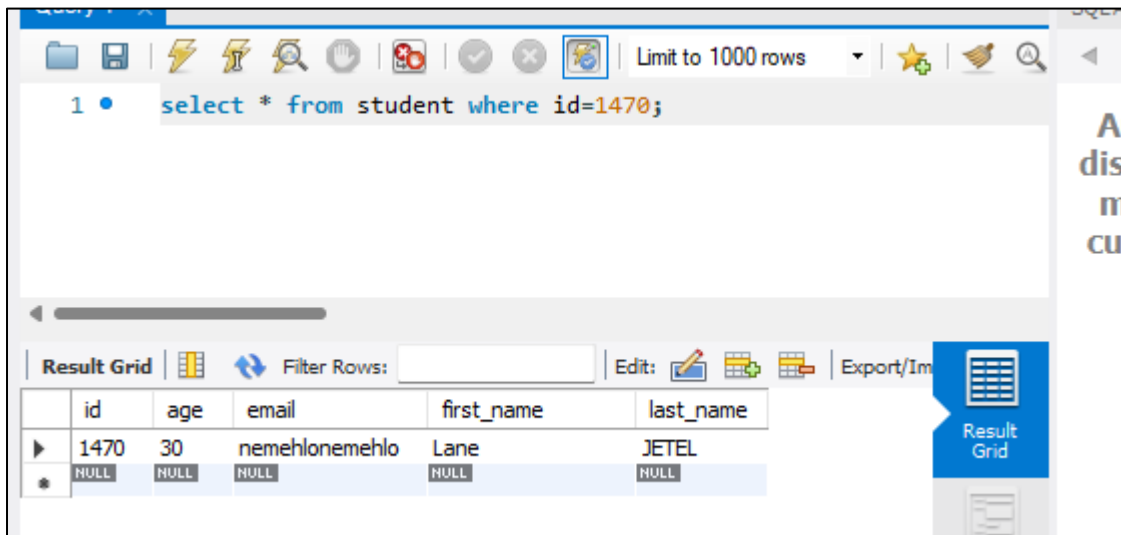
Postman - Implementace studenta bez validního emailu:



Shlédnutí databáze pomocí metody GET v Postmanovi:



Shlédnutí databáze Workbench:



The screenshot shows the MySQL Workbench interface. At the top, a SQL query is entered in the editor: `select * from student where id=1470;`. Below the editor, the 'Result Grid' tab is active, displaying the query results in a table. The table has five columns: 'id', 'age', 'email', 'first_name', and 'last_name'. The first row contains the values 1470, 30, nemehlonemehlo, Lane, and JETEL. The second row contains NULL values for all columns. The interface also shows a toolbar with various icons and a 'Limit to 1000 rows' dropdown.

| | id | age | email | first_name | last_name |
|---|------|------|----------------|------------|-----------|
| ▶ | 1470 | 30 | nemehlonemehlo | Lane | JETEL |
| * | NULL | NULL | NULL | NULL | NULL |

- **Skutečný výsledek:** V Postmanovi se zobrazí status code 200 OK, že je vše v pořádku, ale při zadání nevalidního emailu by se mělo zobrazit, že se takový student nemůže uložit a mělo by dojít k zadání validního emailu. Studentovi s nevalidním emailem bylo přiděleno ID a následně byl propsán do Workbench databáze.
- **Očekávaný výsledek:** Aplikace by měla upozornit uživatele na špatně zadaný email a mělo by dojít k zadání správného emailu, jinak by student neměl být propsán do databáze a nemělo by mu být přiděleno ID.
- **Návrh řešení:** Implementace ochrany a hlášky, že byl špatně zadaný formát email adresy.

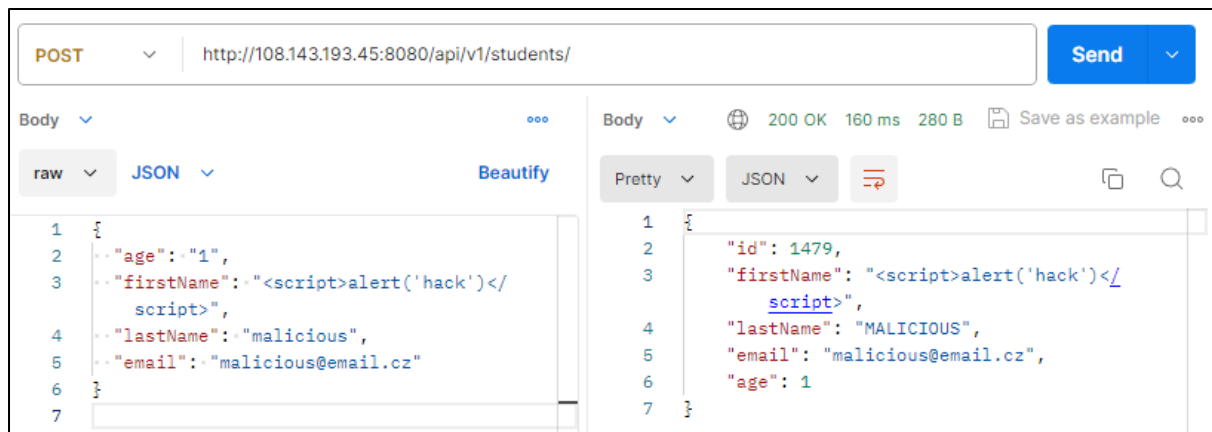
Scénář 4: Přidání studenta s nevalidními (malicious) údaji

- **Předpoklad:** Nevalidní a potenciálně škodlivé údaje.
- **Kroky:**
 1. Otevřít aplikaci Postman
 2. Otevřít nový požadavek http a zvolit metodu POST
 3. Odeslat POST požadavek na /students s tělem obsahujícím nevalidní nebo škodlivé údaje

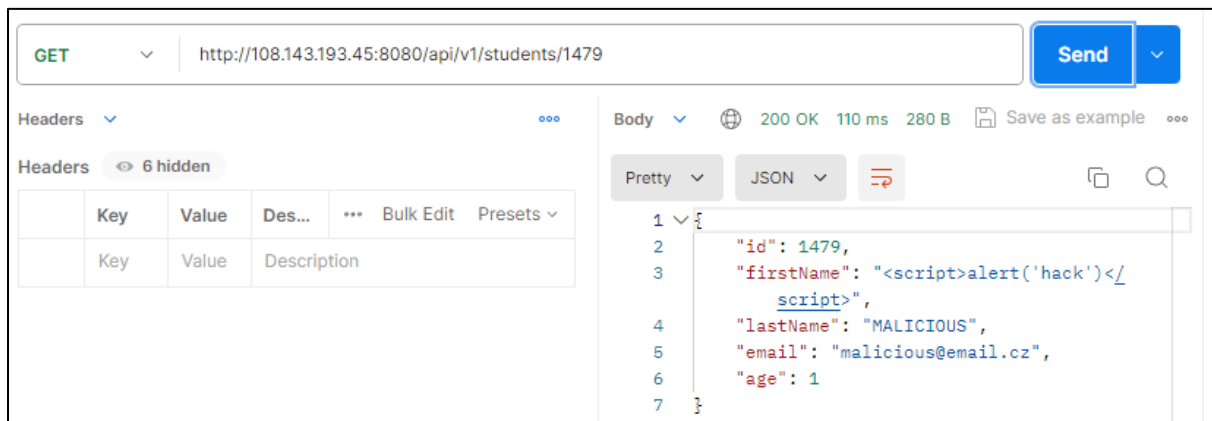
```
{  
  "age": "1",  
  "firstName": "<script>alert('hack')</script>",  
  "lastName": "malicious",  
  "email": "malicious@email.cz"  
}
```

4. Zkontrolovat odpověď.
5. Ověřit, že status kód odpovědi je 400 Bad Request nebo 422 Unprocessable Entity.
6. Ověřit, že se škodlivý kód neprovede a že data nejsou uložena do databáze.

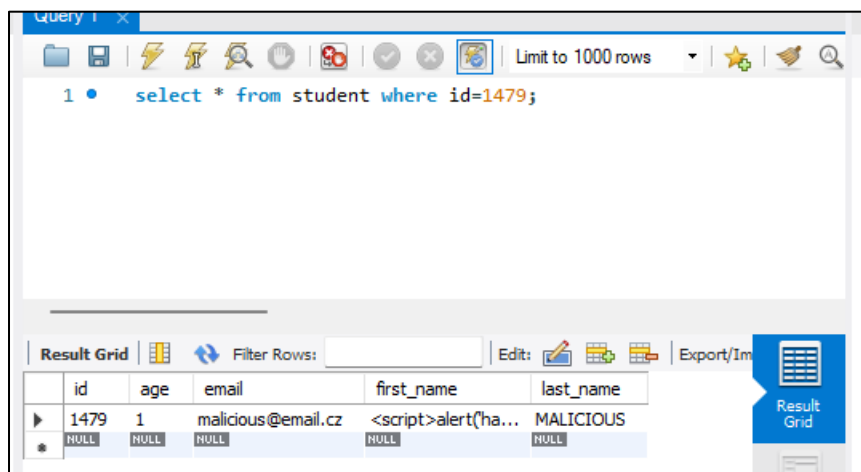
Implementace škodlivého skriptu do pole firstName:



Shlédnutí databáze pomoc metody GET:



Shlédnutí databáze Workbench:



- **Skutečný výsledek:** V Postmanovi se zobrazí status code 200 OK, že je vše v pořádku, ale při zadání malicious by se mělo zobrazit, že se takový student nemůže uložit a že může obsahovat škodlivý obsah. Studentovi se škodlivým malicious bylo přiděleno ID a následně byl propsán do Workbench databáze. Také lastName je uloženo velkými písmeny, nikoliv lowerCase.
- **Očekávaný výsledek:** Odpověď je 400 Bad Request nebo 422 Unprocessable Entity, status kód je 400 nebo 422, škodlivý kód se neprovede a data nejsou uložena do databáze. Dále by lastName mělo být lowerCase.
- **Návrh řešení:** Implementace ochrany a hlášky, že byla zadána potenciálně možná hrozba.

Tento scénář přidává testování pro situace, kdy se někdo pokusí zneužít API zasláním škodlivých dat. Takové testy jsou důležité pro zajištění bezpečnosti aplikace a databáze.

BUG REPORT

Na základě mého testování jsem dospěl k těmto níže zmíněným defectům:

GET:

Scénář 1 – Výsledek obsahuje nelogické znaky (pouze písmena v firstName, lastName a nevhodný email.

Scénář 3 – Status code jako 404 Not Found namísto status code 500.

DELETE:

Scénář 2 – Smazání již smazaného studenta – měl se zobrazit status code 404 - neexistující student resp. neexistující ID. Zobrazí se pouze chybně status code 500.

Scénář 3 – Smazání databáze bez zadání ID – vypsát hlášku 404 Not Found

POST:

Scénář 1 – Založení nového studenta – buď lowerCase.

Scénář 2 – Založení nového studenta bez nadefinovaných polí – nelze, ideálně vypsát hlášku, že jsou potřeba vyplnit všechny atributy

Scénář 3 – Založení nového studenta s nevalidním emailem – Implementace ochrany a hlášky, že byl špatně zadaný formát email adresy

Scénář 4 – Přidání studenta s nevalidními (malicious) údaji – Implementace ochrany

ZÁVĚR

Testování aplikace pro manipulaci s daty o studentech je klíčové pro zajištění její správné funkčnosti a robustnosti. Testy pokrývají širokou škálu scénářů od validních a nevalidních vstupů přes kontrolu formátování dat, bezpečnosti až po exekuci testů a bug reportů.

Důkladné testování pomocí metod GET, DELETE a POST zajistilo, že aplikace bude po opravení vývojářem spolehlivá, bezpečná a bude splňovat všechny požadavky na zpracování a validaci dat. Projekt přispívá k vyšší kvalitě aplikace a měl by zajistit, že všechny operace s daty studentů budou opraveny a následně prováděny korektně a efektivně.