

# FINÁLNÍ PROJEKT

## č.1



# ENGETO

Autor: Václav Osladil  
Datum: 7.4.2024

## OBSAH

<b>ZADÁNÍ</b>	<b>3</b>
<b>TESTOVACÍ SCÉNÁŘE</b>	<b>4</b>
<b>EXEKUCE TESTŮ</b>	<b>5</b>
<b>BUG REPORT</b>	<b>5</b>

# ZADÁNÍ

Čílem 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: 68q0llcbuvyijdt5mzq2 Password: pscale_pw_zxfenAXfSYx9loe9PCS1snuGKr Fzw2J84BMIVZI96o
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.*

## GET

1. Testovací scénář GET API status
2. Testovací scénář GET Ověření všech polí v záznamu studenta
3. Testovací scénář GET Ověření unikátních ID
4. Testovací scénář GET Vybrat studenta dle platného ID
5. Testovací scénář GET Vybrat studenta dle neplatného ID formát čísla
6. Testovací scénář GET Vybrat studenta dle neplatného ID formát písmena abecedy
7. Testovací scénář GET Filtrovat záznamy dle věku
8. Testovací scénář GET Ověření hraničních hodnot věku
9. Testovací scénář GET Ověření formátu email
10. Testovací scénář GET Ověření unikátních emailů
11. Testovací scénář GET Ověření formátu firstName, lastName

## POST

12. Testovací scénář POST Vytvořit záznam s relevantními daty
13. Testovací scénář POST Vytvořit záznam, nevyplněná pole firstName, lastName
14. Testovací scénář POST Vytvořit záznam, nerelevantní data v poli age
15. Testovací scénář POST Vytvořit záznam, duplikát ID
16. Testovací scénář POST Vytvořit záznam, duplikát studenta - stejné údaje
17. Testovací scénář POST Vytvořit záznam, vynechat v záznamu lastName pole
18. Testovací scénář POST Vytvořit záznam, velikost vstupních dat <900 znaků

## DELETE

19. Testovací scénář DELETE Smazat záznam dle platného ID
20. Testovací scénář DELETE Smazat několik záznamů dle platného ID najednou
21. Testovací scénář DELETE Smazat záznam dle neplatného ID formát čísla
22. Testovací scénář DELETE Smazat záznam dle neplatného ID formát písmena abecedy

# EXEKUCE TESTŮ

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

## 1. Testovací scénář GET API status

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci GET pojmenovanou API status
- 3) Do pole GET zadat odkaz na databázi spárovanou v rámci workbench a postman  
`http://108.143.193.45:8080/api/v1/students/`
- 4) Vybrat funkci Tests, v nabídce snippets použít kód Status code: Code is 200

```
pm.test("Status code is 200", function () {  
    pm.response.to.have.status(200);  
});
```

- 5) Kliknout na tlačítko SEND

### Skutečný výsledek

V okně Body 2165 řádků se všemi záznamy studentů, v okně Tests results 1/1 zobrazeno PASS status code is 200.

Status: 200 OK

### Očekávaný výsledek

Očekávaný výsledek se shoduje se skutečným.

## 2. Testovací scénář GET Ověření všech polí v záznamu studenta

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci GET pojmenovanou Ověření všech polí v záznamu studenta
- 3) Do pole GET zadat odkaz na databázi spárovanou v rámci workbench a postman  
`http://108.143.193.45:8080/api/v1/students/`
- 4) Vybrat funkci Tests, použít kód

```
pm.test("Each student has id, firstName, lastName, email, and age fields", function () {  
    var jsonData = pm.response.json();  
  
    jsonData.forEach(function(student) {  
        pm.expect(student).to.have.property("id");  
        pm.expect(student).to.have.property("firstName");  
        pm.expect(student).to.have.property("lastName");  
        pm.expect(student).to.have.property("email");  
        pm.expect(student).to.have.property("age");  
    });  
});
```

- 5) Kliknout na tlačítko SEND

### Skutečný výsledek

V okně Body 2165 řádků se všemi záznamy studentů. V okně Tests results 1/1 zobrazeno PASS Each student has id, firstName, lastName, email, and age fields.

Status: 200 OK

### Očekávaný výsledek

Očekávaný výsledek se shoduje se skutečným.

### 3. Testovací scénář GET Ověření unikátních ID

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci GET pojmenovanou Ověření unikátních ID
- 3) Do pole GET zadat odkaz na databázi spárovanou v rámci workbench a postman <http://108.143.193.45:8080/api/v1/students/>
- 4) Vybrat funkci Tests, použít kód

```
pm.test("IDs are unique", function () {  
    var jsonData = pm.response.json();  
    var ids = jsonData.map(student => student.id);  
  
    pm.expect(ids).toHaveLength(new Set(ids).size);  
});
```

- 5) Kliknout na tlačítko SEND

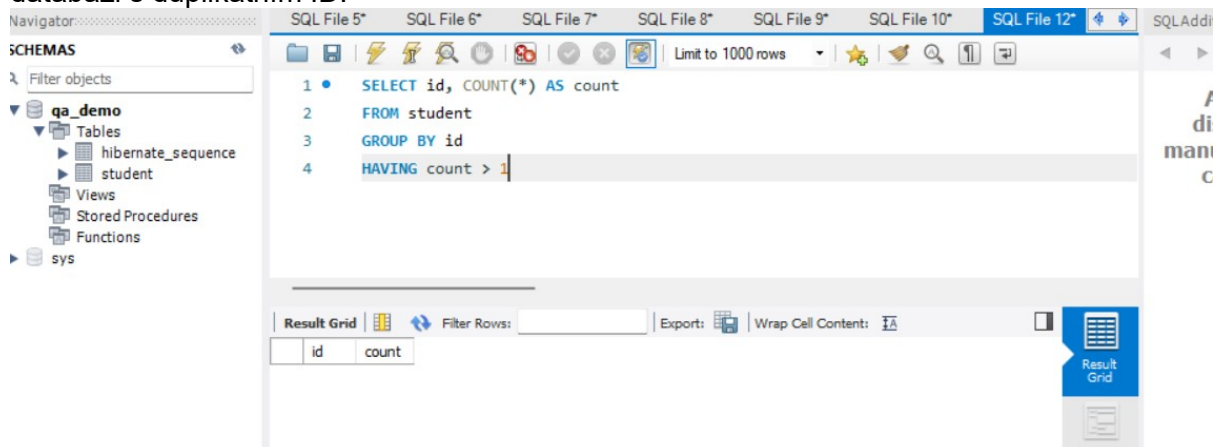
### Skutečný výsledek

V okně Body 2165 řádků se všemi záznamy studentů. V okně Tests results 1/1 zobrazeno PASS IDs are unique.

Status: 200 OK

### Očekávaný výsledek

Očekávaný výsledek se shoduje se skutečným. Dotaz na duplikát ID v SQL. 0 záznamů v databázi s duplikátním ID.



### 4. Testovací scénář GET Vybrat studenta dle platného ID

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci GET pojmenovanou Vybrat studenta dle platného ID
- 3) Do pole GET zadat odkaz na databázi spárovanou v rámci workbench a postman <http://108.143.193.45:8080/api/v1/students/>
- 4) Vyplnit Path Variables do pole KEY zapsat: id, do pole Value zapsat: 175
- 5) Vybrat funkci Tests, použít kód

```
pm.test("Verify specific student details by ID", function () {
```

```

var jsonData = pm.response.json();

var expectedId = 175;

pm.expect(jsonData.id).to.eql(expectedId);
pm.expect(jsonData).to.have.property("firstName");
pm.expect(jsonData).to.have.property("lastName");
pm.expect(jsonData).to.have.property("email");
pm.expect(jsonData).to.have.property("age");
});

```

6) Kliknout na tlačítko SEND

### Skutečný výsledek

V okně Body

```

{
  "id": 175,
  "firstName": "",
  "lastName": "SENNA",
  "email": "ayrton@senna.com",
  "age": 34
},

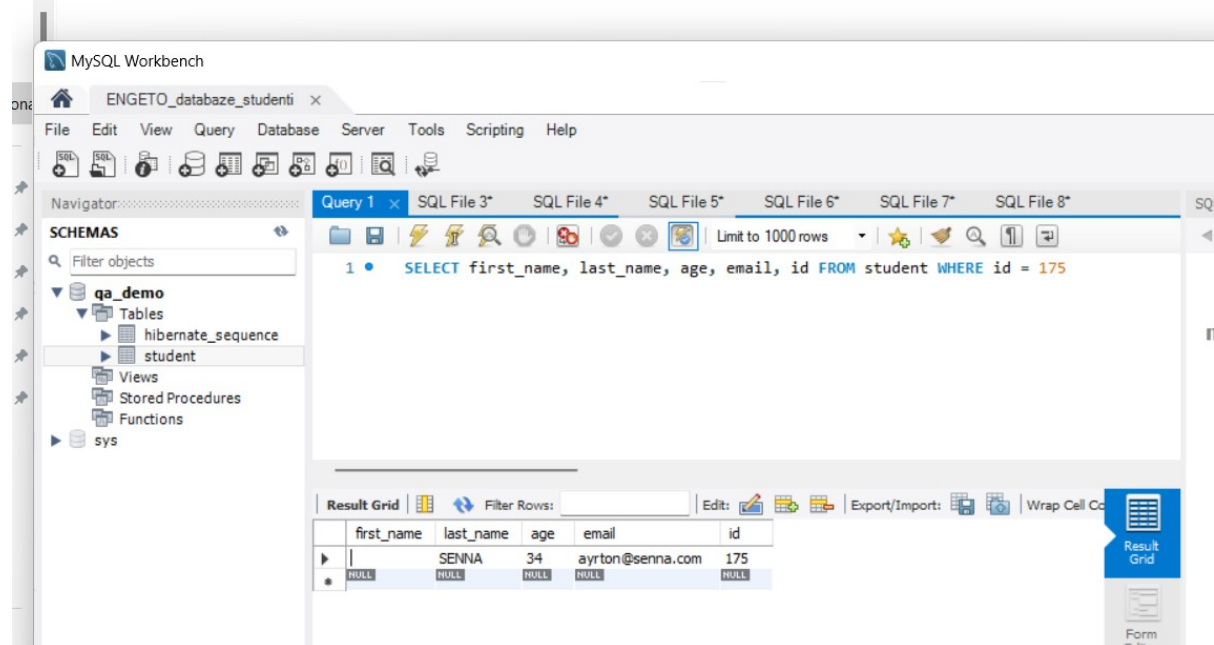
```

V okně Tests results 1/1 zobrazeno PASS Verify specific student details by ID.

Status: 200 OK

### Očekávaný výsledek

Očekávaný výsledek se shoduje se skutečným. Dotaz na ID 175 v SQL databázi.



The screenshot shows the MySQL Workbench interface. The 'Navigator' pane on the left displays the database schema 'qa\_demo' with tables 'hibernate\_sequence' and 'student'. The 'Query Editor' pane shows a SQL query: `SELECT first_name, last_name, age, email, id FROM student WHERE id = 175`. The 'Result Grid' pane at the bottom displays the query results in a table format.

first_name	last_name	age	email	id
	SENNA	34	ayrton@senna.com	175

## 5. Testovací scénář GET Vybrat studenta dle neplatného ID formát čísla

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci GET pojmenovanou Vybrat studenta dle neplatného ID formát čísla
- 3) Do pole GET zadat odkaz na databázi spárovanou v rámci workbench a postman  
`http://108.143.193.45:8080/api/v1/students/`
- 4 ) Vyplnit Path Variables do pole KEY zapsat: id, do pole Value zapsat: 1000
- 5) Vybrat funkci Tests, použít kód

```
pm.test("Verify specific student details by ID", function () {  
    var jsonData = pm.response.json();  
  
    var expectedId = 1000;  
  
    pm.expect(jsonData.id).to.eql(expectedId);  
    pm.expect(jsonData).to.have.property("firstName");  
    pm.expect(jsonData).to.have.property("lastName");  
    pm.expect(jsonData).to.have.property("email");  
    pm.expect(jsonData).to.have.property("age");  
});
```

- 6) Kliknout na tlačítko SEND

### Skutečný výsledek

V okně Body

```
{  
  "timestamp": "2024-04-24T21:52:29.927+00:00",  
  "status": 500,  
  "error": "Internal Server Error",  
  "message": "",  
  "path": "/api/v1/students/1000"  
}
```

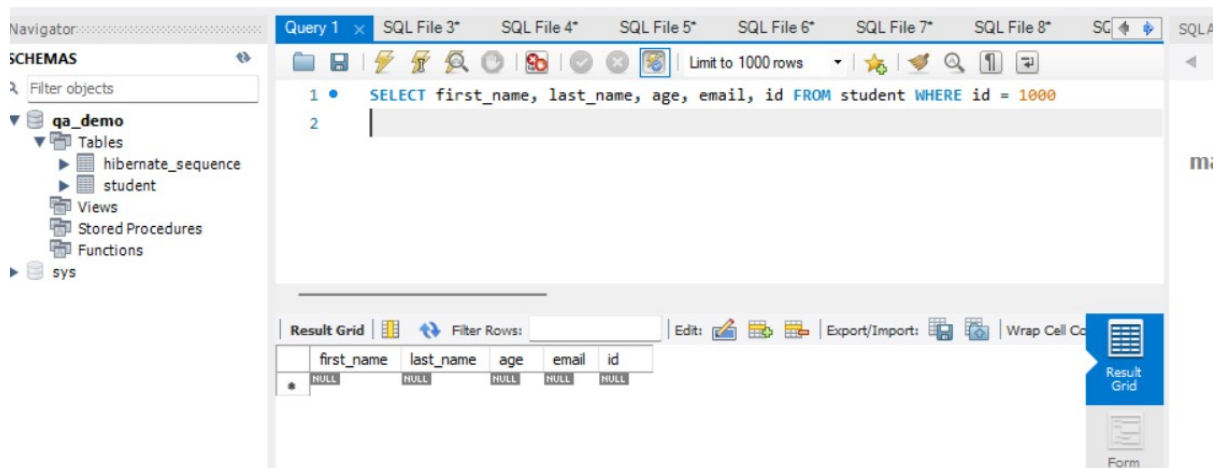
V okně Tests results 1/1 zobrazeno FAIL Verify specific student details by ID AssertionError: expected undefined to deeply equal 1000.

Status: 500 Internal Server Error

### Očekávaný výsledek

Očekávaný výsledek se shoduje se skutečným. Dotaz na ID 1000 v SQL databázi.





## 6. Testovací scénář GET Vybrat studenta dle neplatného ID formát písmena abecedy

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci GET pojmenovanou Vybrat studenta dle neplatného ID formát písmena abecedy
- 3) Do pole GET zadat odkaz na databázi spárovanou v rámci workbench a postman `http://108.143.193.45:8080/api/v1/students/`
- 4) Vyplnit Path Variables do pole KEY zapsat: id, do pole Value zapsat: abrakadabra
- 5) Vybrat funkci Tests, použít kód

```
pm.test("Verify specific student details by ID", function () {
    var jsonData = pm.response.json();

    var expectedId = abrakadabra;

    pm.expect(jsonData.id).to.eql(expectedId);
    pm.expect(jsonData).to.have.property("firstName");
    pm.expect(jsonData).to.have.property("lastName");
    pm.expect(jsonData).to.have.property("email");
    pm.expect(jsonData).to.have.property("age");
});
```

- 6) Kliknout na tlačítko SEND

### Skutečný výsledek

V okně Body

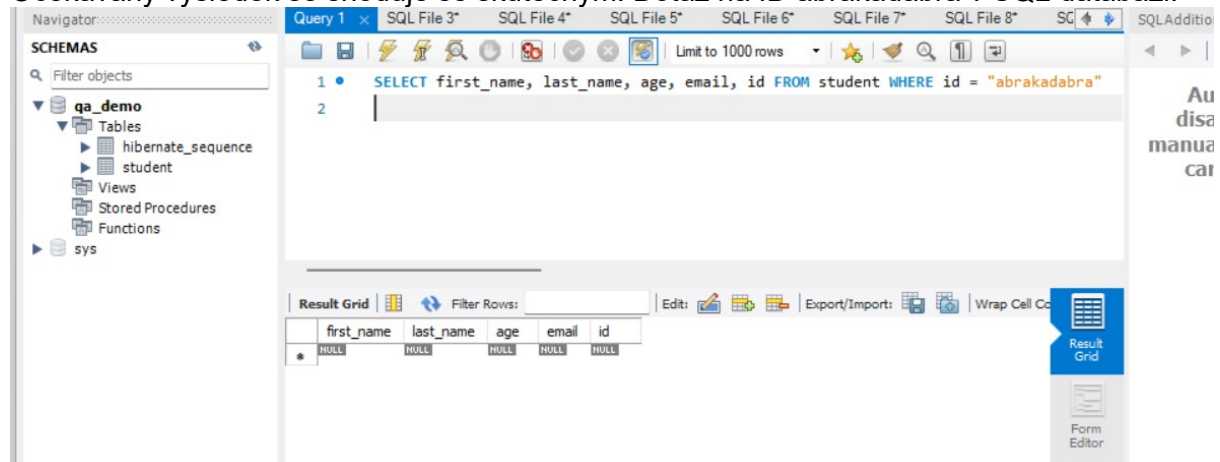
```
{
  "timestamp": "2024-04-24T23:15:45.149+00:00",
  "status": 400,
  "error": "Bad Request",
  "message": "",
  "path": "/api/v1/students/abrakadabra"
}
```

V okně Tests results 1/1 zobrazeno FAIL Verify specific student details by ID | ReferenceError: abrakadabra is not defined.

Status: 400 Bad Request

### Očekávaný výsledek

Očekávaný výsledek se shoduje se skutečným. Dotaz na ID abrakadabra v SQL databázi.



## 7. Testovací scénář GET Filtrovat záznamy dle věku

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci GET pojmenovanou Filtrovat záznamy dle věku
- 3) Do pole GET zadat odkaz na databázi spárovanou v rámci workbench a postman <http://108.143.193.45:8080/api/v1/students/>
- 4) Vyplnit Path Variables do pole KEY zapsat: age, do pole Value zapsat: 22
- 5) Kliknout na tlačítko SEND

### Skutečný výsledek

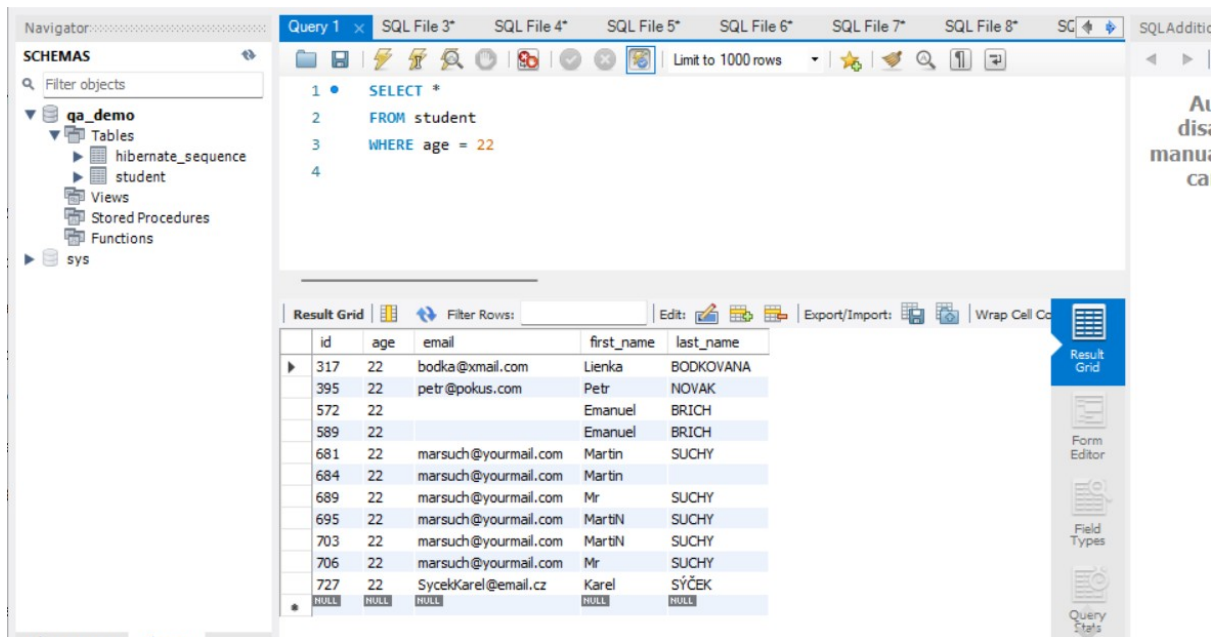
V okně Body

```
{
  "timestamp": "2024-04-24T22:02:04.906+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "message": "",
  "path": "/api/v1/students/22"
}
```

Status: 500 Internal Server Error

### Očekávaný výsledek

V okně body zobrazit pouze studenty s uvedeným věkem 22 let, aplikace nerozpozná Path variable age. Dotaz na filtrování záznamů dle věku v SQL databázi.



## 8. Testovací scénář GET Ověření hraničních hodnot věku

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci GET pojmenovanou Ověření hraničních hodnot věku
- 3) Do pole GET zadat odkaz na databázi spárovanou v rámci workbench a postman <http://108.143.193.45:8080/api/v1/students/>
- 4) Vybrat funkci Tests, použít kód

```
pm.test("Age is between 1 and 99", function () {
  var jsonData = pm.response.json();

  jsonData.forEach(function(student) {
    pm.expect(student.age).to.be.within(1,99 );
  });
});
```

- 5) Kliknout na tlačítko SEND

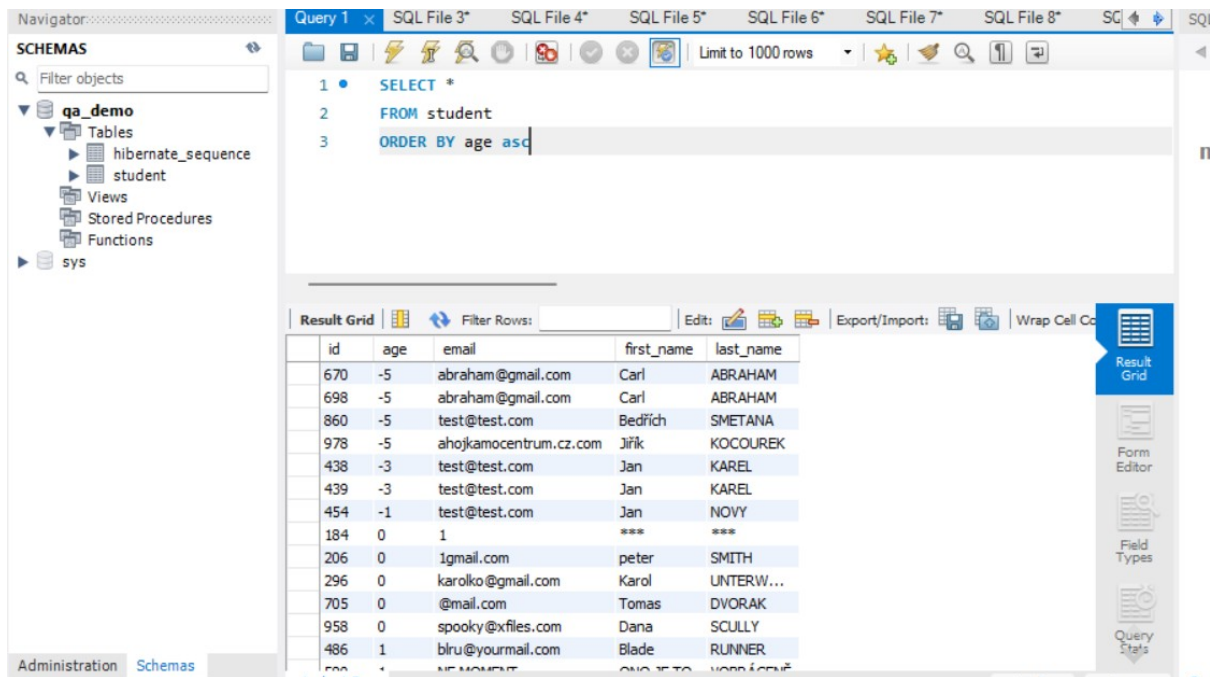
### Skutečný výsledek

V okně Body 2165 řádků se všemi záznamy studentů. V okně Tests results 0/1 zobrazeno FAIL Age is between 1 and 99 | AssertionError: expected +0 to be within 1..99. Do databáze je možné zadat věk v záporném čísle, 0, v kladném čísle bez vymezených hraničních hodnot.

Status: 200 OK

### Očekávaný výsledek

Aplikace by měla upozornit na chybnou hodnotu čísla při vytváření nového záznamu. Vytvořit nový záznam pouze v případě rozptylu např. 6-99.



## 9. Testovací scénář GET Ověření formátu email

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci GET pojmenovanou Ověření formátu email
- 3) Do pole GET zadat odkaz na databázi spárovanou v rámci workbench a postman <http://108.143.193.45:8080/api/v1/students/>
- 4) Vybrat funkci Tests, použít kód

```
pm.test("Emails are valid", function () {
  var jsonData = pm.response.json();

  jsonData.forEach(function(student) {
    pm.expect(student.email).to.match(/^[^s@]+@[^s@]+\.[^s@]+$/);
  });
});
```

- 5) Kliknout na tlačítko SEND

### Skutečný výsledek

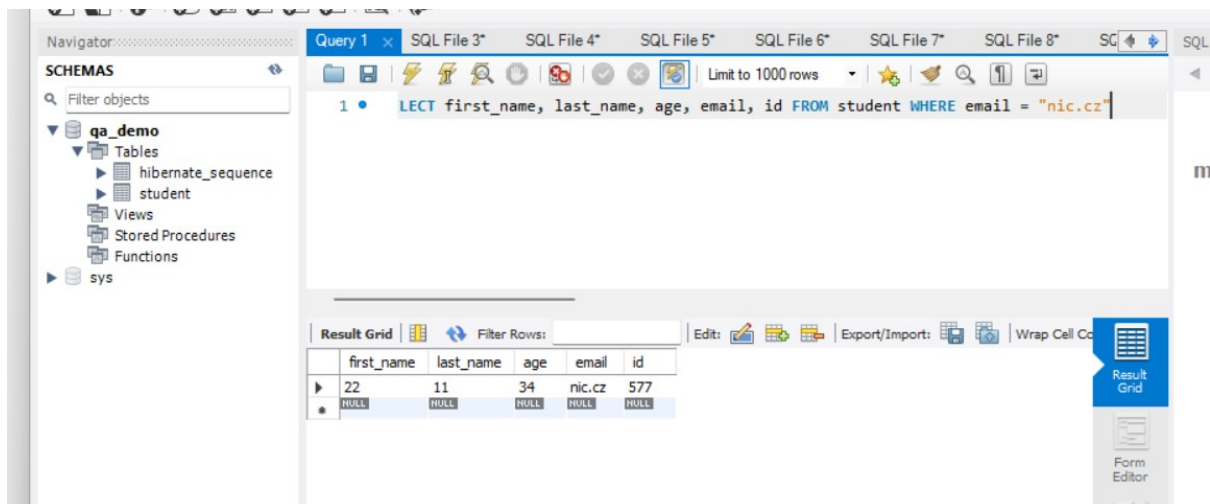
V okně Body 2165 řádků se všemi záznamy studentů. V okně Tests results 0/1 zobrazeno FAIL Emails are valid | AssertionError: expected 'nic.cz' to match /^[^s@]+@[^s@]+\.[^s@]+\$/

Do databáze je možné zadat nesprávný formát emailu.

Status: 200 OK

### Očekávaný výsledek

Aplikace by měla upozornit na chybně zadaný formát emailu při vytváření záznamu. Vytvořit nový záznam pouze v případě vyplněného pole email v uvedeném regulárním výrazu.



## 10. Testovací scénář GET Ověření unikátních emailů

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci GET pojmenovanou Ověření unikátních emailů
- 3) Do pole GET zadat odkaz na databázi spárovanou v rámci workbench a postman <http://108.143.193.45:8080/api/v1/students/>
- 4) Vybrat funkci Tests, použít kód

```
pm.test("Emails are unique", function () {
  const jsonData = pm.response.json();
  const emails = jsonData.map(student => student.email);

  // Check for duplicates
  const uniqueEmails = [...new Set(emails)];

  // Compare the lengths
  pm.expect(uniqueEmails.length).to.equal(emails.length);
});
```

- 5) Kliknout na tlačítko SEND

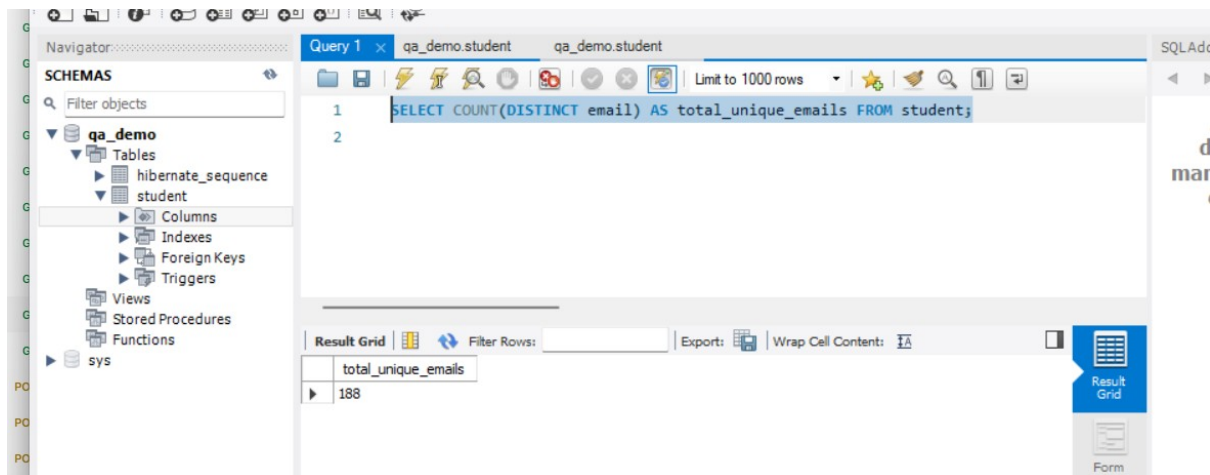
### Skutečný výsledek

V okně Body 2165 řádků se všemi záznamy studentů. V okně Tests results 0/1 zobrazeno FAIL Emails are unique | AssertionError: expected 195 to equal 437. Do databáze je možné zadávat záznamy s duplicitním polem email.

Status: 200 OK

### Očekávaný výsledek

Aplikace by měla upozornit na duplicitu emailu při zakládání nového záznamu. Vytvořit nový záznam pouze s unikátním emailem.



## 11. Testovací scénář GET Ověření formátu firstName, lastName

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci GET pojmenovanou Ověření formátu firstName, lastName
- 3) Do pole GET zadat odkaz na databázi spárovanou v rámci workbench a postman  
`http://108.143.193.45:8080/api/v1/students/`
- 4) Vybrat funkci Tests, použít kód

```
pm.test("First and Last Names are valid", function () {
  var jsonData = pm.response.json();

  jsonData.forEach(function(student) {
    pm.expect(student.firstName).to.match(/^[A-Za-z]+$/);
    pm.expect(student.lastName).to.match(/^[A-Za-z]+$/);
  });
});
```

- 5) Kliknout na tlačítko SEND

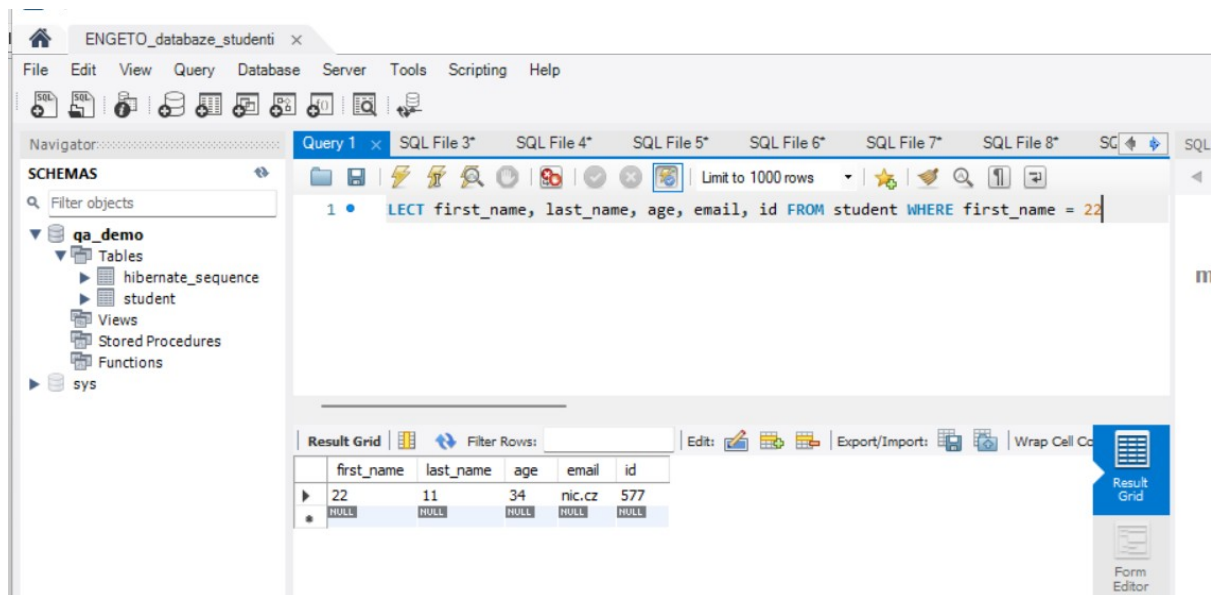
### Skutečný výsledek

V okně Body 2165 řádků se všemi záznamy studentů. V okně Tests results 0/1 zobrazeno FAIL First and Last Names are valid | AssertionError: expected '22' to match /^[A-Za-z]+\$/  
Do databáze je možné zadat jméno ve formátu čísel, znaků, písmen.

Status: 200 OK

### Očekávaný výsledek

Aplikace by měla upozornit na chybně zadaný formát firstName, lastName při vytváření záznamu. Vytvořit nový záznam pouze v případě vyplněného pole firstName, lastName v uvedeném regulárním výrazu.



## 12. Testovací scénář POST Vytvořit záznam s relevantními daty

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci POST pojmenovanou Vytvořit záznam s relevantními daty
- 3) Do pole POST zadat odkaz na databázi spárovanou v rámci workbench a postman  
<http://108.143.193.45:8080/api/v1/students/>
- 4) Vybrat funkci Body, do pole zapsat

```
{
  "firstName": "John",
  "lastName": "Doe",
  "age": 19,
  "email": "JohnDoe@speed.org"
}
```

- 5) Vybrat funkci Tests, použít kód

```
pm.test("Successful POST request", function () {
  pm.expect(pm.response.code).to.be.oneOf([200, 201]);
});
```

- 6) Kliknout na tlačítko SEND

### Skutečný výsledek

V okně Body

```
{
  "id": 957,
  "firstName": "John",
  "lastName": "DOE",
  "email": "JohnDoe@speed.org",
  "age": 19
},
```

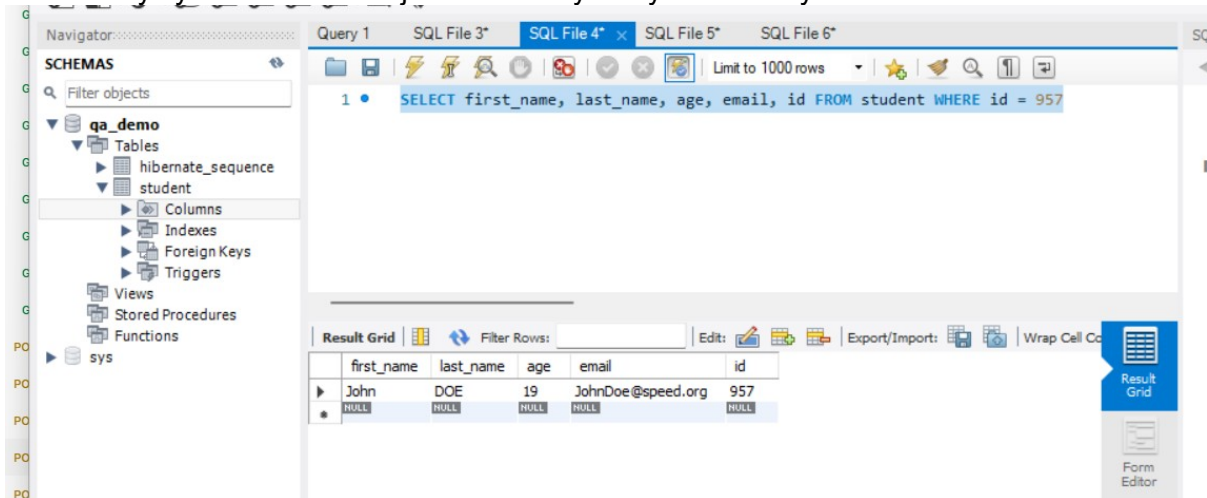
V okně Tests 1/1 PASS Successful POST request.



Status: 200 OK

### Očekávaný výsledek

Očekávaný výsledek se shoduje se skutečným. Vytvořen nový záznam ID 957.



### 13. Testovací scénář POST Vytvořit záznam, nevyplněná pole firstName, lastName

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci POST pojmenovanou Vytvořit záznam, nevyplněná pole firstName, lastName
- 3) Do pole POST zadat odkaz na databázi spárovanou v rámci workbench a postman `http://108.143.193.45:8080/api/v1/students/`
- 4) Vybrat funkci Body, do pole zapsat

```
{
  "firstName": "",
  "lastName": "",
  "age": 19,
  "email": "JohnDoe@speed.or"
}
```

- 5) Vybrat funkci Tests, použít kód

```
pm.test("Successful POST request", function () {
  pm.expect(pm.response.code).to.be.oneOf([200, 201]);
});
```

- 6) Kliknout na tlačítko SEND

### Skutečný výsledek

V okně Body

```
{
  "id": 963,
  "firstName": "",
  "lastName": "",
  "age": 19,
  "email": "JohnDoe@speed.or"
}
```

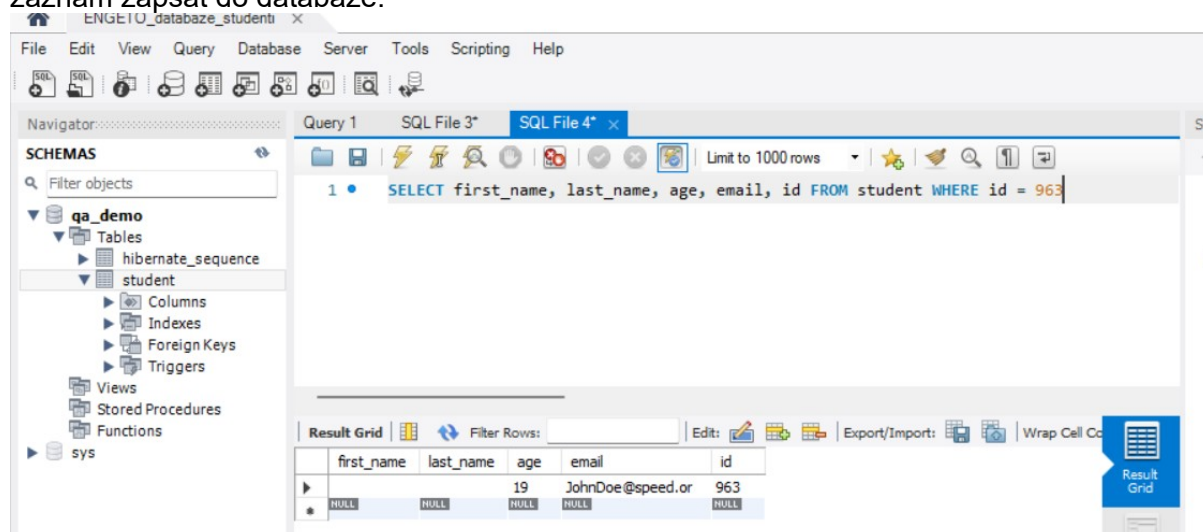


V okně Tests 1/1 PASS Successful POST request.

Status: 200 OK

### Očekávaný výsledek

Aplikace by měla upozornit na NEvyplněná pole firstName, lastName. Neměla by dovolit záznam zapsat do databáze.



## 14. Testovací scénář POST Vytvořit záznam, nerelevantní data v poli age

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci POST pojmenovanou Vytvořit záznam, nerelevantní data v poli age
- 3) Do pole POST zadat odkaz na databázi spárovanou v rámci workbench a postman `http://108.143.193.45:8080/api/v1/students/`
- 4) Vybrat funkci Body, do pole zapsat

```
{
  "firstName": "Jane",
  "lastName": "Doe",
  "age": "aa",
  "email": "JohnDoe@speed.org"
}
```

- 5) Vybrat funkci Tests, použít kód

```
pm.test("Successful POST request", function () {
  pm.expect(pm.response.code).to.be.oneOf([200, 201]);
});
```

- 6) Kliknout na tlačítko SEND

### Skutečný výsledek

V okně Body

```
{
  "timestamp": "2024-05-03T17:50:37.680+00:00",
  "status": 400,
}
```

```
"error": "Bad Request",
"message": "",
"path": "/api/v1/students/"
}
```

V okně Tests 0/1 FAIL Successful POST request | AssertionError: expected 400 to be one of [ 200, 201 ].

Status: 400 Bad Request

### Očekávaný výsledek

Očekávaný výsledek se shoduje se skutečným.

## 15. Testovací scénář POST Vytvořit záznam, duplikát ID

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci POST pojmenovanou Vytvořit záznam, duplikát studenta
- 3) Do pole POST zadat odkaz na databázi spárovanou v rámci workbench a postman  
`http://108.143.193.45:8080/api/v1/students/`
- 4) Vybrat funkci Body, do pole zapsat

```
{
  "id": 957,
  "firstName": "John",
  "lastName": "DOE",
  "email": "JohnDoe@speed.org",
  "age": 19
}
```

- 5) Vybrat funkci Tests, použít kód

```
pm.test("Successful POST request", function () {
  pm.expect(pm.response.code).to.be.oneOf([200, 201]);
});
```

- 6) Kliknout na tlačítko SEND

### Skutečný výsledek

V okně Body

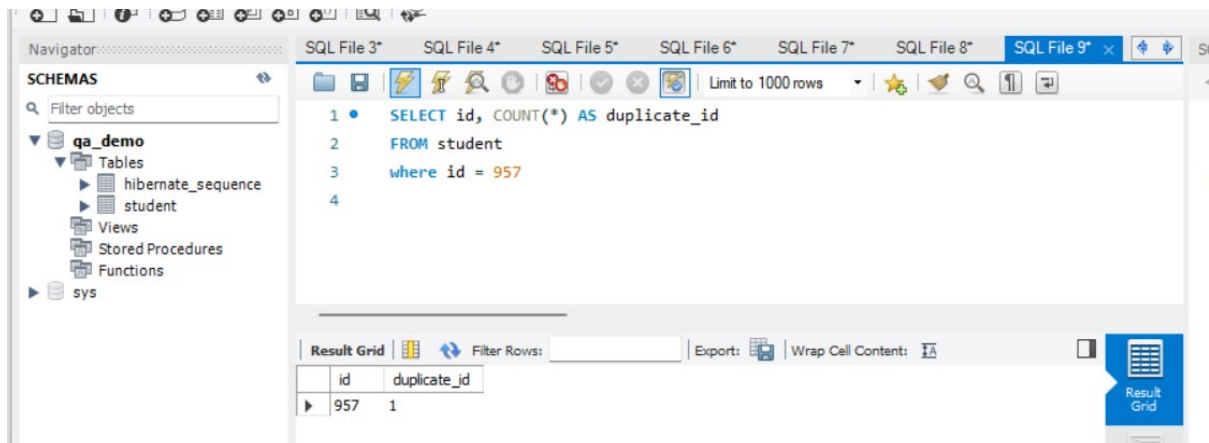
```
{
  "id": 957,
  "firstName": "John",
  "lastName": "DOE",
  "email": "JohnDoe@speed.org",
  "age": 19
},
```

V okně Tests 1/1 PASS Successful POST request.

Status: 200 OK

### Očekávaný výsledek

Očekávaný výsledek 400. Kontrola v datábázi pomocí sql dotazu. viz přiložený screenshot. Duplikát záznamu se v databázi nezapsal.



## 16. Testovací scénář POST Vytvořit záznam, duplikát studenta - znovu zadané stejné údaje

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci POST pojmenovanou Vytvořit záznam, duplikát studenta - znovu zadané stejné údaje
- 3) Do pole POST zadat odkaz na databázi spárovanou v rámci workbench a postman <http://108.143.193.45:8080/api/v1/students/>
- 4) Vybrat funkci Body, do pole zapsat

```
{
  "firstName": "John",
  "lastName": "Doe",
  "age": 19,
  "email": "JohnDoe@speed.org"
}
```

- 5) Vybrat funkci Tests, použít kód

```
pm.test("Successful POST request", function () {
  pm.expect(pm.response.code).to.be.oneOf([200, 201]);
});
```

- 6) Kliknout na tlačítko SEND

### Skutečný výsledek

V okně Body

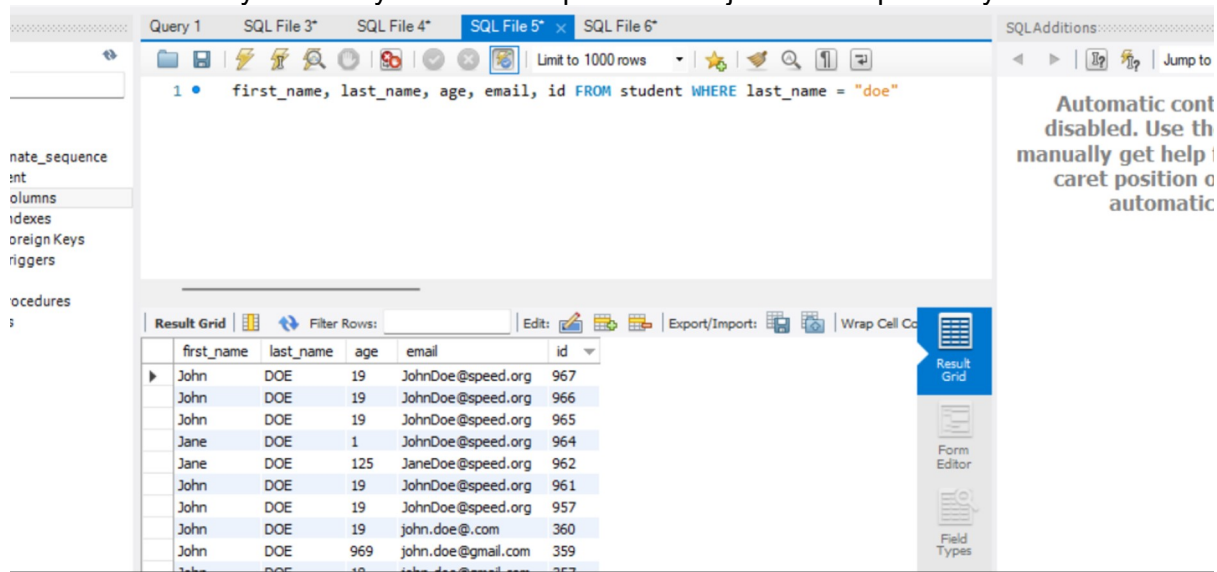
```
{
  "id": 967,
  "firstName": "John",
  "lastName": "DOE",
  "email": "JohnDoe@speed.org",
  "age": 19
}
```

V okně Tests 1/1 PASS Successful POST request.

Status: 200 OK

## Očekávaný výsledek

Aplikace by měla rozpoznat duplikátní záznam na základě stejného emailu. Aplikace by neměla umožnit vytvořit nový záznam s duplicitním údajem emailu pod novým ID.



## 17. Testovací scénář POST Vytvořit záznam, vynechat v záznamu lastName pole

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci POST pojmenovanou Vytvořit záznam, duplikát studenta - znovu zadané stejné údaje
- 3) Do pole POST zadat odkaz na databázi spárovanou v rámci workbench a postman <http://108.143.193.45:8080/api/v1/students/>
- 4) Vybrat funkci Body, do pole zapsat

```
{
  "firstName": "Jane",
  "age": 19,
  "email": "JaneDoe@speed.org"
}
```

- 5) Vybrat funkci Tests, použít kód

```
pm.test("Successful POST request", function () {
  pm.expect(pm.response.code).to.be.oneOf([200, 201]);
});
```

- 6) Kliknout na tlačítko SEND

## Skutečný výsledek

V okně Body

```
{
  "timestamp": "2024-05-03T19:14:13.764+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "message": "",
  "path": "/api/v1/students/"
}
```

Status: 500 Internal Server Error

Očekávaný výsledek se shoduje se skutečným. Aplikace nevytvořila nový záznam.

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci POST pojmenovanou Vytvořit záznam, velikost vstupních dat <900 znaků
- 3) Do pole POST zadat odkaz na databázi spárovanou v rámci workbench a postman <http://108.143.193.45:8080/api/v1/students/>
- 4) Vybrat funkci Body, do pole zapsat

```
{
  "firstName":
    "b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAEEbm9uZQAA7kuuifiiiiiiiii",
    "lastName":
      "Db3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAEEbm9uZQAA7kuuifiiiiiiiiiOE",
        "email":
          "JohnDoeb3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAEEbm9uZQAA7kuuifiiiiiiiii@speed.org",
            "age": 19
}
```

```
pm.test("Successful POST request", function () {
    pm.expect(pm.response.code).to.be.oneOf([200, 201]);
});
```

V okně Body

 $\{$

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci DELETE pojmenovanou Smazat záznam dle platného ID
- 3) Do pole DELETE zadat odkaz na databázi spárovanou v rámci workbench a postman  
`http://108.143.193.45:8080/api/v1/students/`
- 4) Do pole DELETE doplnit 329

celý vstup <http://108.143.193.45:8080/api/v1/students/329>

5) Vybrat funkci Tests, použít kód

```
pm.test("Successful DELETE request", function () {  
    pm.expect(pm.response.code).to.be.oneOf([200, 202, 204]);  
});
```

5) Kliknout na tlačítko SEND

### Skutečný výsledek

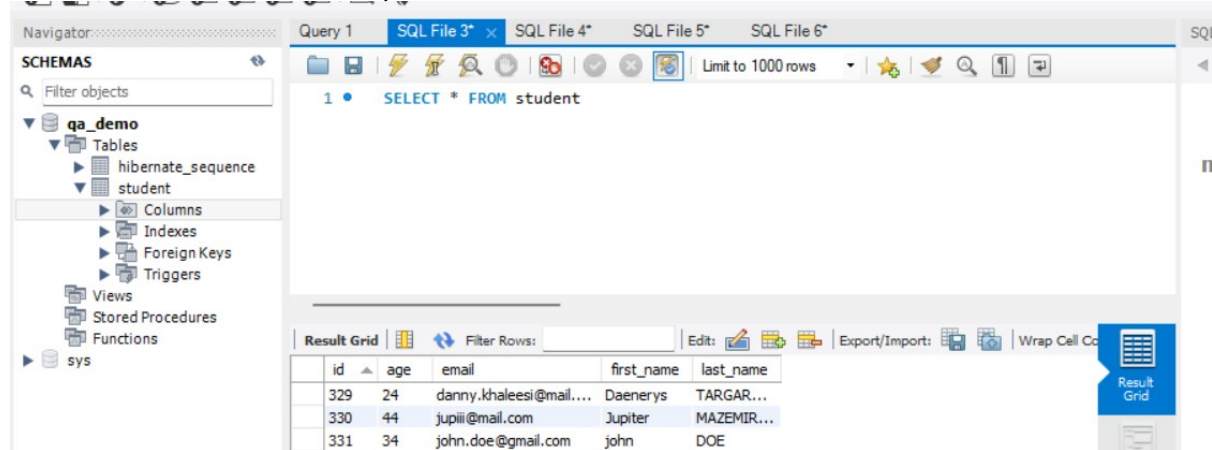
V okně Body prázdný řádek, v okně Tests zobrazeno 1/1 PASS Successful DELETE request.

Status code 200 OK

### Očekávaný výsledek

Očekávaný výsledek se shoduje se skutečným. Aplikace smazala ID 329.

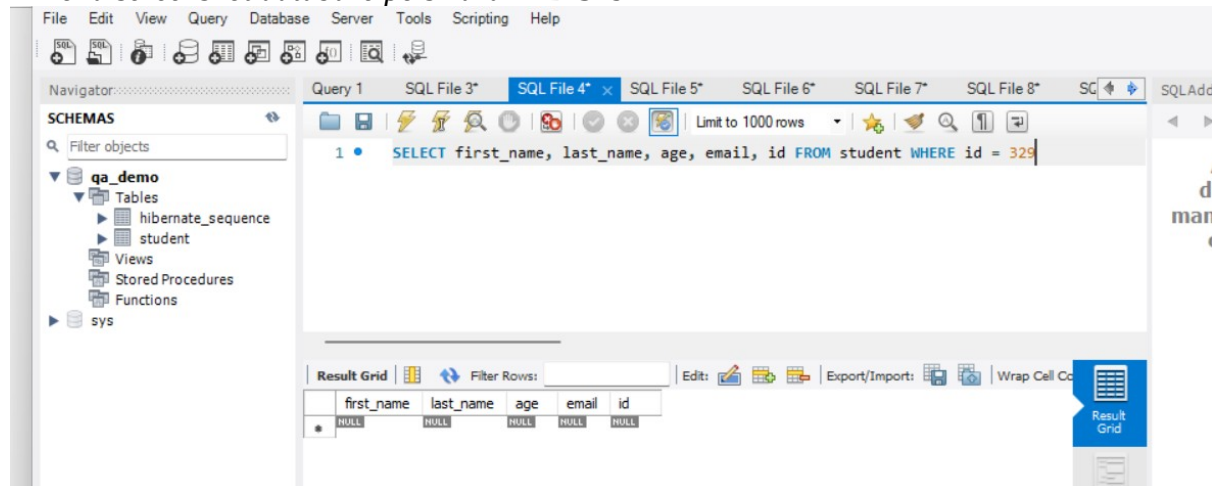
*Příloha screenshot databáze před smazáním ID 329*



The screenshot shows a database client interface with a 'Schemas' pane on the left and a 'Query' pane on the right. The 'student' table is selected in the 'Schemas' pane. The 'Query' pane shows a SQL query: `SELECT * FROM student`. The 'Result Grid' shows the following data:

id	age	email	first_name	last_name
329	24	danny.khaleesi@mail...	Daenerys	TARGAR...
330	44	jupiii@mail.com	Jupiter	MAZEMIR...
331	34	john.doe@gmail.com	john	DOE

*Příloha screenshot databáze po smazání ID 329*



The screenshot shows the same database client interface as the previous one, but the 'Query' pane now shows a SQL query: `SELECT first_name, last_name, age, email, id FROM student WHERE id = 329`. The 'Result Grid' shows the following data:

first_name	last_name	age	email	id
NULL	NULL	NULL	NULL	NULL

## 20. Testovací scénář DELETE Smazat několik záznamů dle platného ID najednou

1) Otevři postman api client

2) Vytvořit v Collections funkci DELETE pojmenovanou Smazat několik záznamů dle platného ID najednou

- 3) Do pole DELETE zadat odkaz na databázi spárovanou v rámci workbench a postman  
`http://108.143.193.45:8080/api/v1/students/`
- 4) Do pole DELETE doplnit `?ids=175,183,184`  
celý vstup `http://108.143.193.45:8080/api/v1/students/?ids=175,183,184`
- 5) Vybrat funkci Tests, použít kód

```
pm.test("Successful DELETE request", function () {  
  pm.expect(pm.response.code).to.be.oneOf([200, 202, 204]);  
});
```

- 5) Kliknout na tlačítko SEND

### Skutečný výsledek

V okně Body

```
{  
  "timestamp": "2024-05-03T22:17:42.369+00:00",  
  "status": 405,  
  "error": "Method Not Allowed",  
  "message": "",  
  "path": "/api/v1/students/"  
}
```

V okně Tests zobrazeno 0/1 FAIL Successful DELETE request | AssertionError: expected 405 to be one of [ 200, 202, 204 ].

Status Code 405 Method Not Allowed

### Očekávaný výsledek

Aplikace by měla být schopná v admin režimu, smazat několik záznamů dle platného ID najednou.

## 21. Testovací scénář DELETE Smazat záznam dle neplatného ID formát čísla

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci DELETE pojmenovanou Smazat záznam dle neplatného ID formát čísla
- 3) Do pole DELETE zadat odkaz na databázi spárovanou v rámci workbench a postman  
`http://108.143.193.45:8080/api/v1/students/`
- 4) Do pole DELETE doplnit 1000  
celý vstup `http://108.143.193.45:8080/api/v1/students/1000`
- 5) Do dialogového okna Tests použít kód

```
pm.test("Successful DELETE request", function () {  
  pm.expect(pm.response.code).to.be.oneOf([200, 202, 204]);  
});
```

- 5) Kliknout na tlačítko SEND

### Skutečný výsledek

v okně Body zobrazeno

```
{  
  "timestamp": "2024-05-04T22:09:27.449+00:00",  
  "status": 500,
```



```
"error": "Internal Server Error",  
"message": "",  
"path": "/api/v1/students/1000"  
}.
```

V okně Tests FAIL Successful DELETE request | AssertionError: expected 500 to be one of [ 200, 202, 204 ].

Status code 500 Internal Server Error

### **Očekávaný výsledek**

Očekávaný výsledek se shoduje se skutečným.

## **22. Testovací scénář DELETE Smazat záznam dle neplatného ID formát písmena abecedy**

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci DELETE pojmenovanou Smazat záznam dle neplatného ID formát písmena abecedy
- 3) Do pole DELETE zadat odkaz na databázi spárovanou v rámci workbench a postman `http://108.143.193.45:8080/api/v1/students/`
- 4) Do pole DELETE doplnit abra  
celý vstup `http://108.143.193.45:8080/api/v1/students/abra`
- 5) Do dialogového okna Tests použít kód

```
pm.test("Successful DELETE request", function () {  
  pm.expect(pm.response.code).to.be.oneOf([200, 202, 204]);  
});
```

- 5) Kliknout na tlačítko SEND

### **Skutečný výsledek**

V okně Body

```
{  
  "timestamp": "2024-05-04T22:13:19.291+00:00",  
  "status": 400,  
  "error": "Bad Request",  
  "message": "",  
  "path": "/api/v1/students/abra"  
}
```

V okně Tests 0/1 FAIL Successful DELETE request | AssertionError: expected 400 to be one of [ 200, 202, 204 ].

Status code 400 Bad Request

### **Očekávaný výsledek**

Očekávaný výsledek se shoduje se skutečným.

# BUG REPORT

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

## 7. Testovací scénář GET Filtrovat záznamy dle věku

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci GET pojmenovanou Filtrovat záznamy dle věku
- 3) Do pole GET zadat odkaz na databázi spárovanou v rámci workbench a postman  
`http://108.143.193.45:8080/api/v1/students/`
- 4) Vyplnit Path Variables do pole KEY zapsat: age, do pole Value zapsat: 22
- 5) Kliknout na tlačítko SEND

### Skutečný výsledek

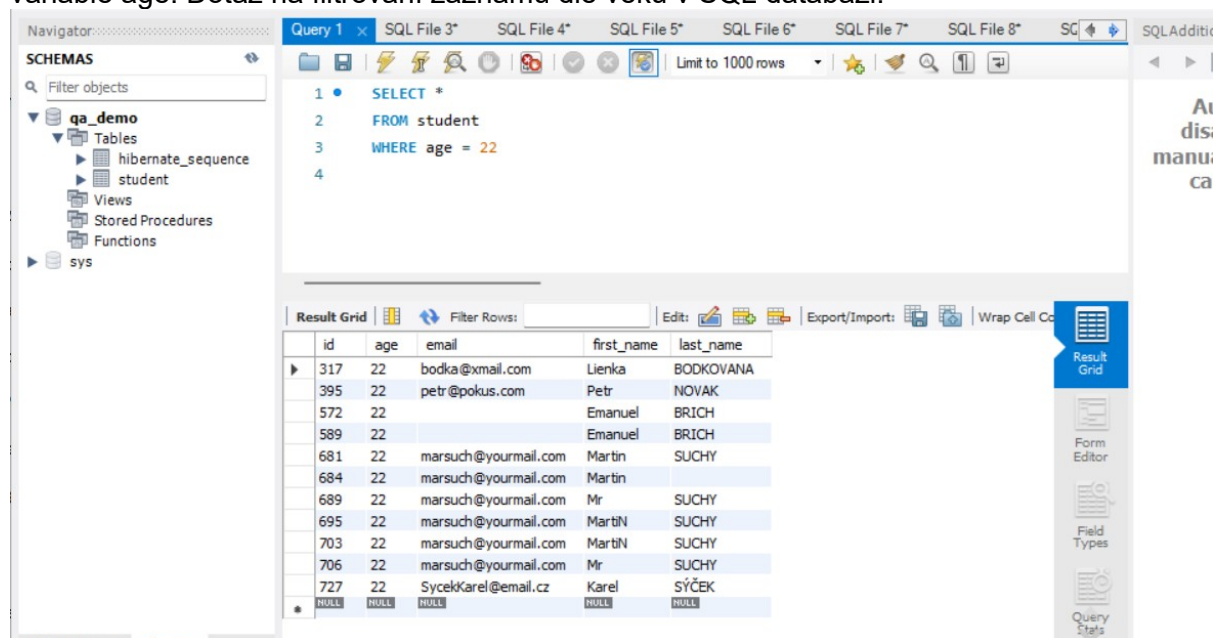
V okně Body

```
{
  "timestamp": "2024-04-24T22:02:04.906+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "message": "",
  "path": "/api/v1/students/22"
}
```

Status: 500 Internal Server Error

### Očekávaný výsledek

V okně body zobrazit pouze studenty s uvedeným věkem 22 let, aplikace nerozpozná Path variable age. Dotaz na filtrování záznamů dle věku v SQL databázi.



## 8. Testovací scénář GET Ověření hraničních hodnot věku

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci GET pojmenovanou Ověření hraničních hodnot věku
- 3) Do pole GET zadat odkaz na databázi spárovanou v rámci workbench a postman

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

4) Vybrat funkci Tests, použít kód

```
pm.test("Age is between 1 and 99", function () {  
    var jsonData = pm.response.json();  
  
    jsonData.forEach(function(student) {  
        pm.expect(student.age).to.be.within(1,99 );  
    });  
});
```

5) Kliknout na tlačítko SEND

### Skutečný výsledek

V okně Body 2165 řádků se všemi záznamy studentů. V okně Tests results 0/1 zobrazeno FAIL Age is between 1 and 99 | AssertionError: expected +0 to be within 1..99.

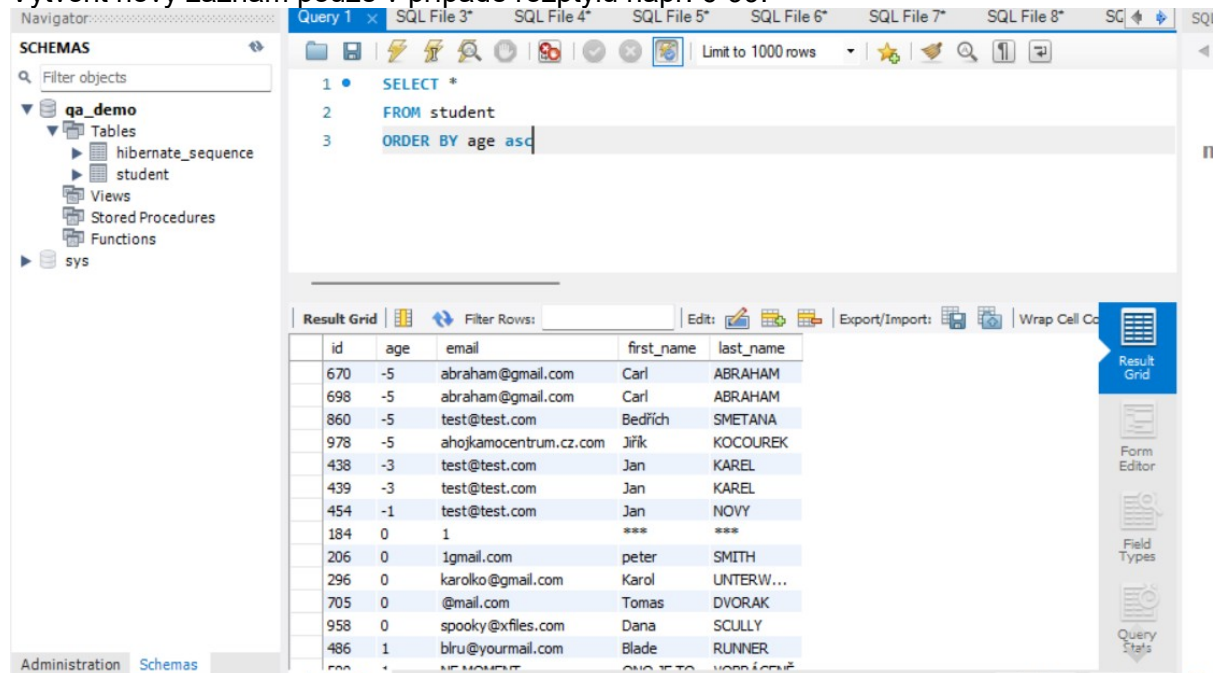
Do databáze je možné zadat věk v záporném čísle, 0, v kladném čísle bez vymezených hraničních hodnot.

Status: 200 OK

### Očekávaný výsledek

Aplikace by měla upozornit na chybnou hodnotu čísla při vytváření nového záznamu.

Vytvořit nový záznam pouze v případě rozptylu např. 6-99.



The screenshot shows a database management interface. On the left is a 'SCHEMAS' tree with 'qa\_demo' expanded, showing 'Tables' (hibernate\_sequence, student), 'Views', 'Stored Procedures', and 'Functions'. The main area displays a SQL query: 'SELECT \* FROM student ORDER BY age asc'. Below the query is a 'Result Grid' showing a table of student data. The table has columns: id, age, email, first\_name, last\_name. The data rows show various students with ages ranging from -5 to 1, and some with age 0. The interface also includes a toolbar with icons for query execution, filters, and other database operations.

id	age	email	first_name	last_name
670	-5	abraham@gmail.com	Carl	ABRAHAM
698	-5	abraham@gmail.com	Carl	ABRAHAM
860	-5	test@test.com	Bedřich	SMETANA
978	-5	ahojkamocentrum.cz.com	Jiřík	KOCOUREK
438	-3	test@test.com	Jan	KAREL
439	-3	test@test.com	Jan	KAREL
454	-1	test@test.com	Jan	NOVY
184	0	1	***	***
206	0	1gmail.com	peter	SMITH
296	0	karolko@gmail.com	Karol	UNTERW...
705	0	@mail.com	Tomas	DVORAK
958	0	spooky@xfiles.com	Dana	SCULLY
486	1	blru@yourmail.com	Blade	RUNNER

## 9. Testovací scénář GET Ověření formátu email

1) Otevři postman api client

2) Vytvořit v Collections funkci GET pojmenovanou Ověření formátu email

3) Do pole GET zadat odkaz na databázi spárovanou v rámci workbench a postman

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

4) Vybrat funkci Tests, použít kód

```
pm.test("Emails are valid", function () {  
    var jsonData = pm.response.json();
```

```

jsonData.forEach(function(student) {
    pm.expect(student.email).to.match(/^[^s@]+@[^s@]+\.[^s@]+$/);
});
});

```

5) Kliknout na tlačítko SEND

### Skutečný výsledek

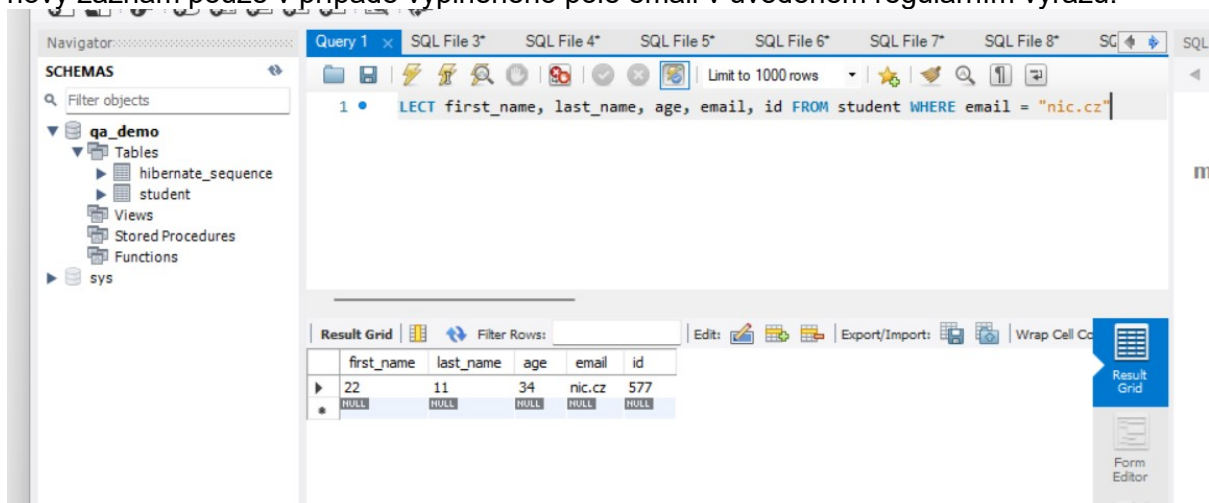
V okně Body 2165 řádků se všemi záznamy studentů. V okně Tests results 0/1 zobrazeno FAIL Emails are valid | AssertionError: expected 'nic.cz' to match /^[^s@]+@[^s@]+\.[^s@]+\$/

Do databáze je možné zadat nesprávný formát emailu.

Status: 200 OK

### Očekávaný výsledek

Aplikace by měla upozornit na chybně zadaný formát emailu při vytváření záznamu. Vytvořit nový záznam pouze v případě vyplněného pole email v uvedeném regulárním výrazu.



## 10. Testovací scénář GET Ověření unikátních emailů

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci GET pojmenovanou Ověření unikátních emailů
- 3) Do pole GET zadat odkaz na databázi spárovanou v rámci workbench a postman <http://108.143.193.45:8080/api/v1/students/>
- 4) Vybrat funkci Tests, použít kód

```

pm.test("Emails are unique", function () {
    const jsonData = pm.response.json();
    const emails = jsonData.map(student => student.email);

    // Check for duplicates
    const uniqueEmails = [...new Set(emails)];

    // Compare the lengths
    pm.expect(uniqueEmails.length).to.equal(emails.length);
});

```

5) Kliknout na tlačítko SEND

### Skutečný výsledek

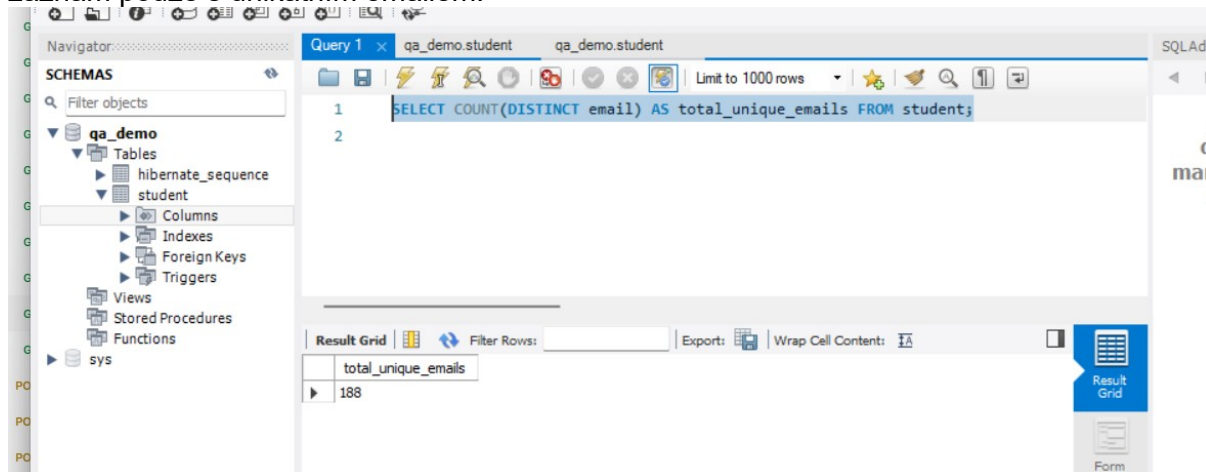
V okně Body 2165 řádků se všemi záznamy studentů. V okně Tests results 0/1 zobrazeno FAIL Emails are unique | AssertionError: expected 195 to equal 437.

Do databáze je možné zadávat záznamy s duplicitním polem email.

Status: 200 OK

### Očekávaný výsledek

Aplikace by měla upozornit na duplicitu emailu při zakládání nového záznamu. Vytvořit nový záznam pouze s unikátním emailem.



## 11. Testovací scénář GET Ověření formátu firstName, lastName

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci GET pojmenovanou Ověření formátu firstName, lastName
- 3) Do pole GET zadat odkaz na databázi spárovanou v rámci workbench a postman <http://108.143.193.45:8080/api/v1/students/>
- 4) Vybrat funkci Tests, použít kód

```
pm.test("First and Last Names are valid", function () {  
    var jsonData = pm.response.json();  
  
    jsonData.forEach(function(student) {  
        pm.expect(student.firstName).to.match(/^[A-Za-z]+$/);  
        pm.expect(student.lastName).to.match(/^[A-Za-z]+$/);  
    });  
});
```

- 5) Kliknout na tlačítko SEND

### Skutečný výsledek

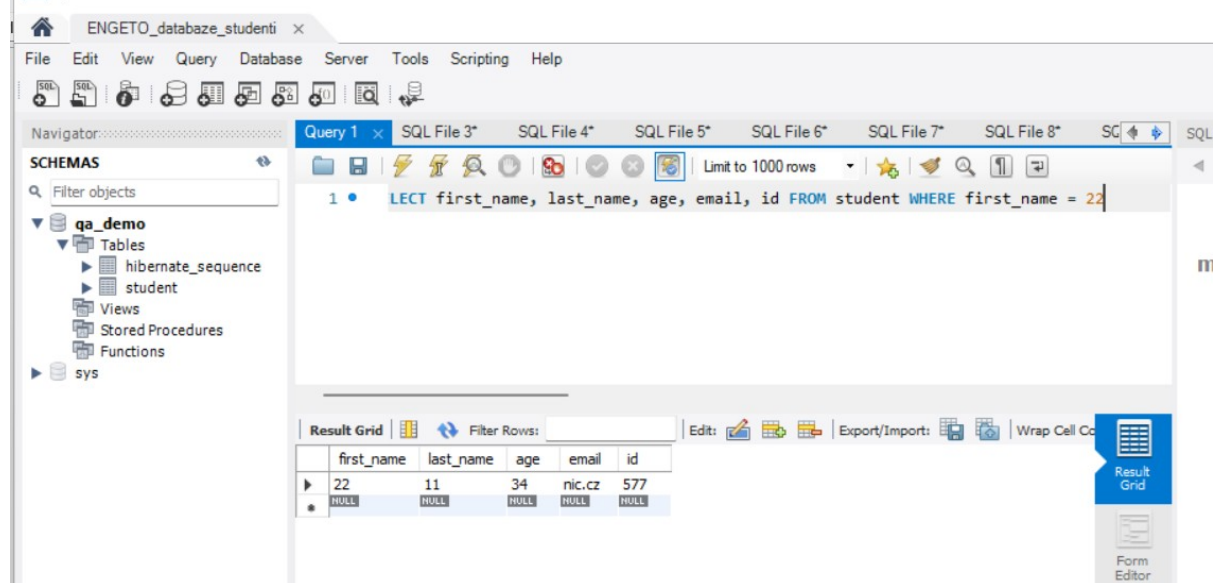
V okně Body 2165 řádků se všemi záznamy studentů. V okně Tests results 0/1 zobrazeno FAIL First and Last Names are valid | AssertionError: expected '22' to match /^[A-Za-z]+\$/  
Do databáze je možné zadat jméno ve formátu čísel, znaků, písmen.

Status: 200 OK

### Očekávaný výsledek

Aplikace by měla upozornit na chybně zadaný formát firstName, lastName při vytváření záznamu. Vytvořit nový záznam pouze v případě vyplněného pole firstName, lastName v

uvedeném regulárním výrazu.



### 13. Testovací scénář POST Vytvořit záznam, nevyplněná pole firstName, lastName

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci POST pojmenovanou Vytvořit záznam, nevyplněná pole firstName, lastName
- 3) Do pole POST zadat odkaz na databázi spárovanou v rámci workbench a postman <http://108.143.193.45:8080/api/v1/students/>
- 4) Vybrat funkci Body, do pole zapsat

```
{
  "firstName": "",
  "lastName": "",
  "age": 19,
  "email": "JohnDoe@speed.or"
}
```

- 5) Vybrat funkci Tests, použít kód

```
pm.test("Successful POST request", function () {
  pm.expect(pm.response.code).to.be.oneOf([200, 201]);
});
```

- 6) Kliknout na tlačítko SEND

#### Skutečný výsledek

V okně Body

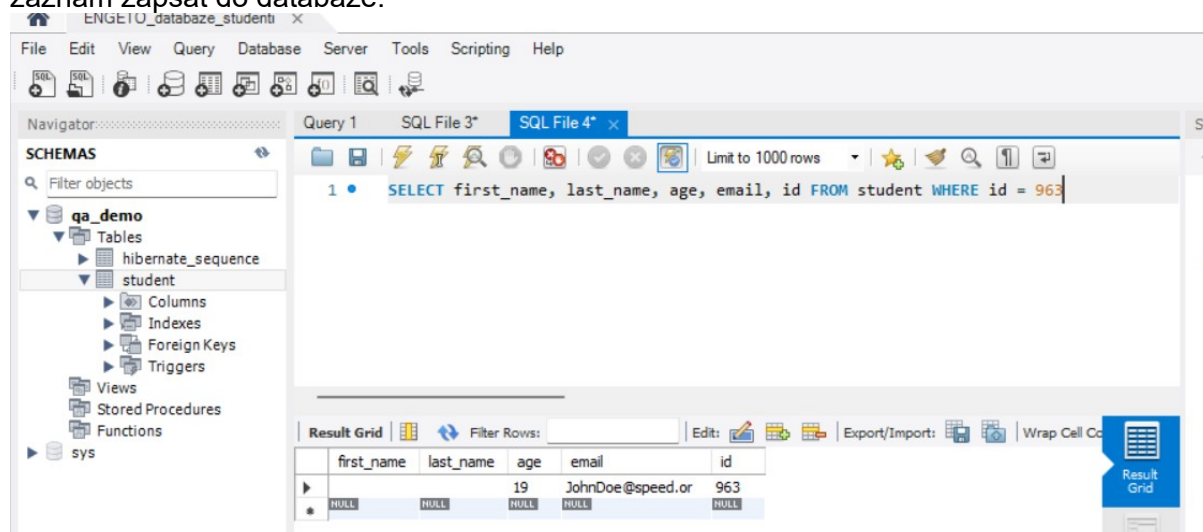
```
{
  "id": 963,
  "firstName": "",
  "lastName": "",
  "age": 19,
  "email": "JohnDoe@speed.or"
}
```

V okně Tests 1/1 PASS Successful POST request.

Status: 200 OK

### Očekávaný výsledek

Aplikace by měla upozornit na NEvyplněná pole firstName, lastName. Neměla by dovolit záznam zapsat do databáze.



## 15. Testovací scénář POST Vytvořit záznam, duplikát ID

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci POST pojmenovanou Vytvořit záznam, duplikát studenta
- 3) Do pole POST zadat odkaz na databázi spárovanou v rámci workbench a postman `http://108.143.193.45:8080/api/v1/students/`
- 4) Vybrat funkci Body, do pole zapsat

```
{
  "id": 957,
  "firstName": "John",
  "lastName": "DOE",
  "email": "JohnDoe@speed.org",
  "age": 19
}
```

- 5) Vybrat funkci Tests, použít kód

```
pm.test("Successful POST request", function () {
  pm.expect(pm.response.code).to.be.oneOf([200, 201]);
});
```

- 6) Kliknout na tlačítko SEND

### Skutečný výsledek

V okně Body

```
{
  "id": 957,
  "firstName": "John",
```



```

    "lastName": "DOE",
    "email": "JohnDoe@speed.org",
    "age": 19
  },

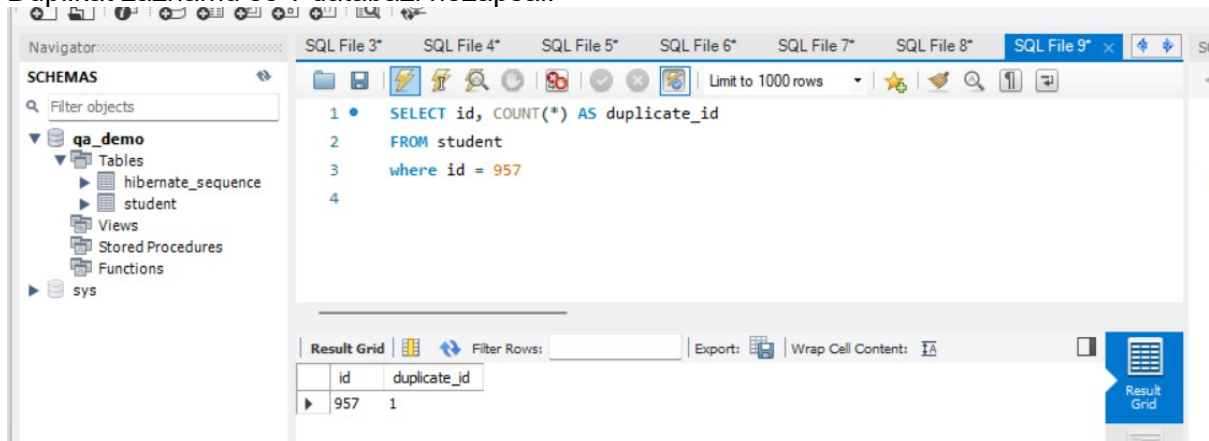
```

V okně Tests 1/1 PASS Successful POST request.

Status: 200 OK

### Očekávaný výsledek

Očekávaný výsledek 400. Kontrola v datábázi pomocí sql dotazu. viz přiložený screenshot. Duplikát záznamu se v databázi nezapsal.



## 16. Testovací scénář POST Vytvořit záznam, duplikát studenta - znovu zadané stejné údaje

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci POST pojmenovanou Vytvořit záznam, duplikát studenta - znovu zadané stejné údaje
- 3) Do pole POST zadat odkaz na databázi spárovanou v rámci workbench a postman <http://108.143.193.45:8080/api/v1/students/>
- 4) Vybrat funkci Body, do pole zapsat

```

{
  "firstName": "John",
  "lastName": "Doe",
  "age": 19,
  "email": "JohnDoe@speed.org"
}

```

- 5) Vybrat funkci Tests, použít kód

```

pm.test("Successful POST request", function () {
  pm.expect(pm.response.code).to.be.oneOf([200, 201]);
});

```

- 6) Kliknout na tlačítko SEND

### Skutečný výsledek

V okně Body



[illegible]

```

}
    "email":
    "JohnDoeb3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAA7kuuiiOe",
    "age": 19
}

```

5) Vybrat funkci Tests, použít kód

```

pm.test("Successful POST request", function () {
    pm.expect(pm.response.code).to.be.oneOf([200, 201]);
});

```

6) Kliknout na tlačítko SEND

### Skutečný výsledek

V okně Body

```

{
  "id": 971,
  "firstName":
  "b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAA7kuuiiOe",
  "lastName":
  "DB3BLBNNZAC1RZXKTDJEAAAAABG5VBMUAAAAEBM9UZQAAAA7KUUIiOe",
  "email":
  "JohnDoeb3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAA7kuuiiOe",
  "age": 19
}

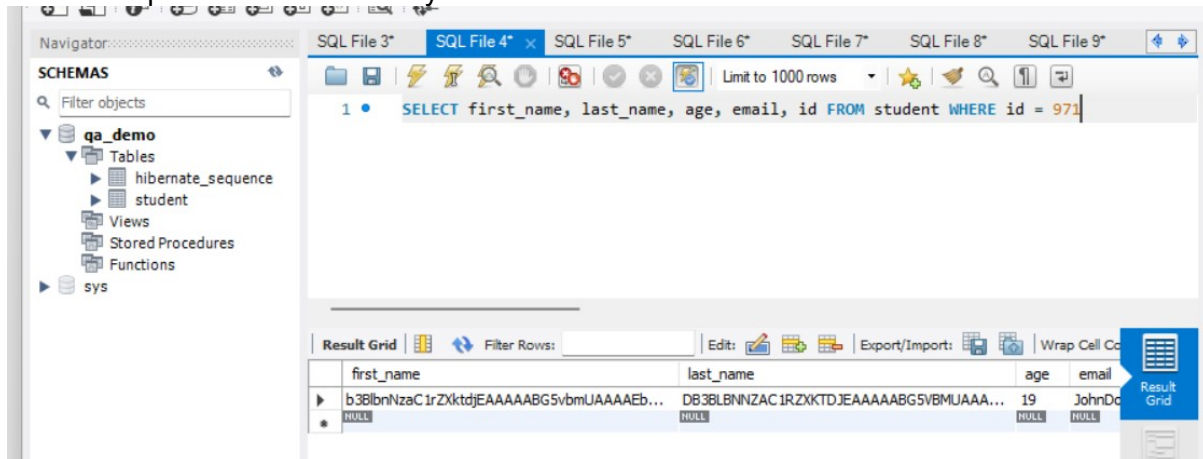
```

V okně Tests 1/1 PASS Successful POST request.

Status: 200 OK

### Očekávaný výsledek

Očekávaný výsledek se shoduje se skutečným. Defekt - aplikace by měla omezit počet znaků vstupu dat na nižší desítky.



## 20. Testovací scénář DELETE Smazat několik záznamů dle platného ID najednou

- 1) Otevři postman api client
- 2) Vytvořit v Collections funkci DELETE pojmenovanou Smazat několik záznamů dle platného ID najednou
- 3) Do pole DELETE zadat odkaz na databázi spárovanou v rámci workbench a postman <http://108.143.193.45:8080/api/v1/students/>
- 4) Do pole DELETE doplnit ?ids=175,183,184  
celý vstup <http://108.143.193.45:8080/api/v1/students/?ids=175,183,184>
- 5) Vybrat funkci Tests, použít kód

```
pm.test("Successful DELETE request", function () {  
    pm.expect(pm.response.code).to.be.oneOf([200, 202, 204]);  
});
```

- 5) Kliknout na tlačítko SEND

### Skutečný výsledek

V okně Body

```
{  
  "timestamp": "2024-05-03T22:17:42.369+00:00",  
  "status": 405,  
  "error": "Method Not Allowed",  
  "message": "",  
  "path": "/api/v1/students/"  
}
```

V okně Tests zobrazeno 0/1 FAIL Successful DELETE request | AssertionError: expected 405 to be one of [ 200, 202, 204 ].

Status Code 405 Method Not Allowed

### Očekávaný výsledek

Aplikace by měla být schopná v admin režimu, smazat několik záznamů dle platného ID najednou.