

FINÁLNÍ PROJEKT

č.1



ENGETO

Autor: Veronika Fronková
Datum: 22.07.2024

OBSAH

ZADÁNÍ	3
Přístupové údaje	
Poznámky	
TESTOVACÍ SCÉNÁŘE	4
EXEKUCE TESTŮ	8
BUG REPORT	9

ZADÁNÍ

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

Přístupové údaje:

Databáze	database: qa_demo Host: aws.connect.psdb.cloud
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 ověřila funkčnost aplikace.

Metoda GET

1. Získání dat existujícího studenta

Popis: Test ověří, zda API vrátí data existujícího studenta.

Kroky:

1. Otevření MySQL Workbench:
 - Připojení databáze s tabulkou "student"
 - Ověření existence záznamu studenta s id = "400"
 - `SELECT * FROM student WHERE id=400;`
2. Otevřete Postman a vybereme metodu GET.
3. Nastavíme URL endpointu na:
`http://108.143.193.45:8080/api/v1/students/400`
4. Klikněte send.

```
Očekávaný výstup: {  "id" : 400
                      "firstName" : "Janko"
                      "lastName" : "TEST"
                      "email" : "janko@yourmail.com"
                      "age" : 99
                      }
```

Stavový kód: 200 OK.

Kroky:

1. Otevření MySQL Workbench:
 - Připojení databáze s tabulkou "student"
 - Ověření existence záznamu studenta s id = "361"
 - `SELECT * FROM student WHERE id=361;`
2. Otevřete Postman a vybereme metodu GET.
3. Nastavíme URL endpointu na:
`http://108.143.193.45:8080/api/v1/students/361`
4. Klikněte send.

```
Očekávaný výstup: {  "id" : 361
                      "firstName" : "Petra"
                      "lastName" : "SOVA"
                      "email" : "petraA@sova.cz"
                      "age" : 17
                      }
```

Stavový kód: 200 OK.

Kroky:

1. Otevření MySQL Workbench:
 - Připojení databáze s tabulkou "student"
 - Ověření existence záznamu studenta s id = "359"
 - `SELECT * FROM student WHERE id=359;`
2. Otevřete Postman a vybereme metodu GET.
3. Nastavíme URL endpointu na:
`http://108.143.193.45:8080/api/v1/students/359`

4. Klinkněte send.

Očekávaný výstup: { "id" : 359
 "firstName" : "John"
 "lastName" : "DOE"
 "email" : "john.doe@gmail.com"
 "age" : 969
 }
Stavový kód: 200 OK.

2. Získání dat neexistujícího studenta

Popis: Test ověří, zda API vrátí chybu při pokusu o získání dat neexistujícího studenta.

Kroky:

1. Otevření MySQL Workbench:
 - Připojení databáze s tabulkou "student"
 - Ověření existence záznamu studenta s id = "0"
 - SELECT * FROM student WHERE id=0;
2. Otevřete Postman a vybereme metodu GET.
3. Nastavíme URL endpointu na:
`http://108.143.193.45:8080/api/v1/students/0`
4. Klinkněte send.

Očekávaný výstup:

Stavový kód: 404 Not Found. Chybové hlášení, které informuje, že student s daným id neexistuje.

Kroky:

1. Otevření MySQL Workbench:
 - Připojení databáze s tabulkou "student"
 - Ověření existence záznamu studenta s id = "1987"
 - SELECT * FROM student WHERE id=1987;
2. Otevřete Postman a vybereme metodu GET.
3. Nastavíme URL endpointu na:
`http://108.143.193.45:8080/api/v1/students/1987`
4. Klinkněte send.

Očekávaný výstup:

Stavový kód: 404 Not Found. Chybové hlášení, které informuje, že student s daným id neexistuje.

Kroky:

1. Otevření MySQL Workbench:
 - Připojení databáze s tabulkou "student"
 - Ověření existence záznamu studenta s id = "124"
 - SELECT * FROM student WHERE id=124;
2. Otevřete Postman a vybereme metodu GET.
3. Nastavíme URL endpointu na:
`http://108.143.193.45:8080/api/v1/students/124`
4. Klinkněte send.

Očekávaný výstup:

Stavový kód: 404 Not Found. Chybové hlášení, které informuje, že student s daným id neexistuje.

Metoda POST

3. Vytvoření nového záznamu studenta

- Popis: Test ověří, zda API umožňuje vytvoření nového studenta. Vše zadané se má v aplikaci a databázi zobrazit jako lower case.

Kroky:

1. Otevřete Postman a vybereme metodu POST.
3. Vypíšeme povinné údaje pro záznam studenta – “firstName”, “lastName”, “email”, “age”

Vstup -> Url: <http://108.143.193.45:8080/api/v1/students/>

```
{  "firstName": "Alexandr"
  "lastName": "Veliký"
  "email": "thisis@yourmail.com"
  "age": 36
}
```

4. Klinkněte send.
5. Vytvoření nového záznamu v databázi.

Očekávaný výstup -> { "id": 1035
"firstName": "alexandr"
"lastName": "veliký"
"email": "thisis@yourmail.com"
"age": 36
}

	id	age	email	first_name	last_name
	1035	36	thisis@yourmail.com	alexandr	veliký

Stavový kód: 201 Created nebo 200 OK

4. Test správnosti vstupních dat

Popis: Test ověří, zda API správně validuje data zadáním různých formátů dat u “firstName” a “lastName” .

Kroky:

1. Otevřete Postman a vybereme metodu POST.
3. Vypíšeme povinné údaje pro záznam studenta – “firstName”, “lastName”, “email”, “age”

Vstup -> Url: <http://108.143.193.45:8080/api/v1/students/>

```
{  "firstName": "Jan"
  "lastName": "NOVÁK"
  "email": "thisis@yourmail.com"
  "age": 36
}
```

4. Klinkněte send.

5. Vytvoření nového záznamu v databázi.

```
Očekávaný výstup -> { "id": 1034
                        "firstName": "jan"
                        "lastName": "novák"
                        "email": "thisis@yourmail.com"
                        "age": 36
                      }
```

Stavový kód: 201 Created nebo 200 OK

5. Test správnosti vstupních dat II

Popis: Test ověří, zda API správně validuje data zadáním nesprávného formátu "email".

Kroky:

1. Otevřete Postman a vybereme metodu POST.
3. Vypíšeme povinné údaje pro záznam studenta – "firstName", "lastName", "email", "age"

Vstup -> Url: <http://108.143.193.45:8080/api/v1/students/>

```
{
  "firstName": "Jarmila"
  "lastName": "Malinká"
  "email": "thismaybeyourmail.com"
  "age": 15
}
```

4. Klinkněte send.

Očekávaný výstup: Objeví se odpovídající chybové kódy a chybová hláška na nesprávný formát vstupních údajů.

Metoda DELETE

4. Smazání existujícího studenta

Popis: Test ověří, zda API umožňuje smazání existujícího studenta na základě zadaného id.

Kroky:

1. Otevření MySQL Workbench:
 - Připojení databáze s tabulkou "student"
 - Ověření existence záznamu studenta s id = "1000"
 - SELECT * FROM student WHERE id=1000;

2. Otevřete Postman a vybereme metodu DELETE.

3. Nastavíme URL endpointu na:

<http://108.143.193.45:8080/api/v1/students/1000>

4. Odeslání DELETE požadavku.

```
Očekávaný výstup: {
  "timestamp": "2024-05-17T10:15:23.123+00:00",
  "status": 204,
  "message": "Student with ID 1000 successfully deleted",
  "path": "/api/v1/students/1000"
}
```

}

Stavový kód: 204 No Content.

5. Smazání neexistujícího studenta

Popis: Test ověří, zda API správně validuje data při pokusu o smazání neexistujícího studenta.

Kroky:

1. Otevření MySQL Workbench:
 - Připojení databáze s tabulkou "student"
 - Ověření existence záznamu studenta s id = "1000"
 - SELECT * FROM student WHERE id=1000;
2. Otevřete Postman a vybereme metodu DELETE.
3. Nastavíme URL endpointu na:
http://108.143.193.45:8080/api/v1/students/1000
4. Odeslání DELETE požadavku.

Očekávaný výstup: {
"timestamp": "2024-05-17T10:15:23.123+00:00",
"status": 204,
"message": "Student with ID 1000 successfully deleted",
"path": "/api/v1/students/1000"
}

Stavový kód: 204 No Content.

EXEKUCE TESTŮ

Testovací scénáře jsem provedla, přikládám výsledky testů.

Metoda GET

1. Získání dat existujícího studenta s ID 400
Výsledek: **Prošel** (Stavový kód: 200 OK)
2. Získání dat existujícího studenta s ID 361
Výsledek: **Prošel** (Stavový kód: 200 OK)
3. Získání dat existujícího studenta s ID 359
Výsledek: **Prošel** (Stavový kód: 200 OK)
4. Získání dat neexistujícího studenta s ID 0
Výsledek: **Neprošel** – chyba 500 Internal Server Error místo 404 Not Found
5. Získání dat neexistujícího studenta s ID 1987
Výsledek: **Neprošel** – chyba 500 Internal Server Error místo 404 Not Found
6. Získání dat neexistujícího studenta s ID 124
Výsledek: **Neprošel** – chyba 500 Internal Server Error místo 404 Not Found

Metoda POST

1. Vytvoření nového záznamu studenta
Výsledek: **Prošel** (Stavový kód: 201 Created nebo 200 OK)

2. Test správnosti vstupních dat

Výsledek: **Neprošel** – záznam se má uložit jako lower case, ale nedělá to

3. Test správnosti vstupních dat II

Výsledek: **Neprošel** – je možné zadat nesprávný formát emailu

Metoda DELETE

1. Smazání existujícího studenta s ID 1000

Výsledek: **Neprošel** - špatný statusový kód a žádná zpráva o úspěšném smazání záznamu - 200 OK

2. Smazání neexistujícího studenta s ID 1000

Výsledek: **Neprošel** – špatný statusový kód a žádná zpráva o úspěšném smazání záznamu – 500 Internal Server Error

BUG REPORT

Na základě provedených scénářů jsem objevila uvedené chyby aplikace.

Metoda GET

1. Chyba stavového kódu 404 Not Found.

Abstract: Při testování získání dat o neexistujícím studentovi se místo očekávaného stavového kódu 404 Not Found objeví chybový stav 500 Internal Server Error.

1. Neexistující student id = 0

Kroky:

1. Otevření MySQL Workbench:
 - Připojení databáze s tabulkou "student"
 - Ověření existence záznamu studenta s id = "0"
 - SELECT * FROM student WHERE id=0;
2. Otevřete Postman a vybereme metodu GET.
3. Nastavíme URL endpointu na:
http://108.143.193.45:8080/api/v1/students/0
4. Klikněte send.

Očekávaný výsledek: Stavový kód 404 Not Found s chybovou zprávou o neexistenci studenta.

Skutečný výsledek: Zobrazí se stavový kód 500 Internal Server Error.

```
{
  "timestamp": "2024-07-02T14:12:46.530+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "message": "",
  "path": "/api/v1/students/0"
}
```

2. Neexistující student id = 1987

Kroky:

1. Otevření MySQL Workbench:
 - Připojení databáze s tabulkou "student"
 - Ověření existence záznamu studenta s id = "1987"
SELECT * FROM student WHERE id=1987;
2. Otevřete Postman a vybereme metodu GET.
3. Nastavíme URL endpointu na:
http://108.143.193.45:8080/api/v1/students/1987
4. Klikněte send.

Očekávaný výsledek: Stavový kód 404 Not Found s chybovou zprávou o neexistenci studenta.

Skutečný výsledek: Zobrazí se stavový kód 500 Internal Server Error.

```
{
  "timestamp": "2024-07-02T14:14:02.013+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "message": "",
  "path": "/api/v1/students/1987"
}
```

3. Neexistující student id = 124

Kroky:

1. Otevření MySQL Workbench:
 - Připojení databáze s tabulkou "student"
 - Ověření existence záznamu studenta s id = "1987"
 - SELECT * FROM student WHERE id=1987;
2. Otevřete Postman a vybereme metodu GET.
3. Nastavíme URL endpointu na:
http://108.143.193.45:8080/api/v1/students/1987
4. Klikněte send.

Očekávaný výsledek: Stavový kód 404 Not Found s chybovou zprávou o neexistenci studenta.

Skutečný výsledek: Zobrazí se stavový kód 500 Internal Server Error.

```
{
  "timestamp": "2024-07-02T14:18:02.239+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "message": "",
  "path": "/api/v1/students/124"
}
```

Metoda POST

2. Chyba velikosti písmen při uložení – nedodržení lower case

Abstract: Aplikace mění u "lastName" velikost písmen na upper case a "firstName" ukládá s počátečním písmenem velkým. Aplikace by měla vše zobrazovat lower case.

Kroky:

1. Otevřete Postman a vybereme metodu POST.
3. Vypíšeme povinné údaje pro záznam studenta – “firstName”, “lastName”, “email”, “age”.

Vstup -> Url: <http://108.143.193.45:8080/api/v1/students/>

```
{  "firstName": "Jan"
  "lastName": "Novak"
  "email": "thisis@yourmail.com"
  "age": 36
}
```

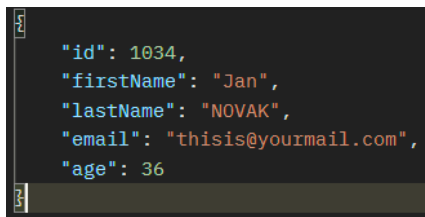
4. Klikněte send a vytvoříte nový záznam v databázi.

Očekávaný výsledek :

```
{  "id": 1034
  "firstName": "jan"
  "lastName": "novak"
  "email": "thisis@yourmail.com"
  "age": 36
}
```

Stavový kód: 201 Created nebo 200 OK

Skutečný výsledek : Aplikace uloží data “lastName” upper case a data “firstName” s počátečním písmenem velkým.



```
{  "id": 1034,
  "firstName": "Jan",
  "lastName": "NOVAK",
  "email": "thisis@yourmail.com",
  "age": 36
}
```

id	age	email	first_name	last_name
1034	36	thisis@yourmail.com	Jan	NOVAK

2. Ukládání dat ve špatném formátu “email”

Abstract: Při testování metody post jde vložit “email” ve špatném formátu.

Kroky:

1. Otevřete Postman a vybereme metodu POST.
3. Vypíšeme povinné údaje pro záznam studenta – “firstName”, “lastName”, “email”, “age”

Vstup -> Url: <http://108.143.193.45:8080/api/v1/students/>

```
{  "firstName": "Jarmila"
  "lastName": "Malinká"
  "email": "thismaybeyourmail.com"
  "age": 15
}
```

4. Klikněte send.

Očekávaný výsledek: Objeví se odpovídající chybové kódy a chybová hláška na nesprávný formát vstupních údajů.

Skutečný výsledek: Záznam se uloží i přesto, že vložíme nesprávný formát emailu.

```
1 {  
2   "id": 1345,  
3   "firstName": "Jarmila",  
4   "lastName": "MALINKA",  
5   "email": "thismaybeyourmail.com",  
6   "age": 15  
7 }
```

id	age	email	first_name	last_name
1345	15	thismaybeyourmail.com	Jarmila	MALINKA

Metoda DELETE

4. Chyba při smazání existujícího studenta

Abstract: Při testování metody DELETE u existujícího studenta se objeví pouze stavový kód 200 OK bez dalších informací.

Kroky:

1. Otevření MySQL Workbench:
 - Připojení databáze s tabulkou "student"
 - Ověření existence záznamu studenta s id = "1000"
 - SELECT * FROM student WHERE id=1000;
2. Otevřete Postman a vybereme metodu DELETE.
3. Nastavíme URL endpointu na:
http://108.143.193.45:8080/api/v1/students/1000
4. Odeslání DELETE požadavku.

Očekávaný výsledek:

```
{  
  "timestamp": "2024-05-17T10:15:23.123+00:00",  
  "status": 204,  
  "message": "Student with ID 1000 successfully deleted",  
  "path": "/api/v1/students/1000"  
}
```

Stavový kód: 204 No Content.

Skutečný výsledek: Stavový kód 200 OK a nic víc.

5. Chyba při smazání neexistujícího studenta.

Abstract: Při testování metody DELETE u neexistujícího studenta se objeví stavový kód 500 Internal Server Error namísto 404 Not Found.

Kroky:

1. Otevření MySQL Workbench:
 - Připojení databáze s tabulkou "student"
 - Ověření existence záznamu studenta s id = "1000"
 - SELECT * FROM student WHERE id=1000;

2. Otevřete Postman a vybereme metodu DELETE.
3. Nastavíme URL endpointu na:
`http://108.143.193.45:8080/api/v1/students/1000`
4. Odeslání DELETE požadavku.

Očekávaný výsledek: Stavový kód odpovědi 404 Not Found a odpovídající chybové hlášení, které informuje uživatele o tom, že student s daným id neexistuje.

Skutečný výsledek: Zobrazí se stavový kód 500 Internal Server Error.

```
{
  "timestamp": "2024-07-02T15:29:49.376+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "message": "",
  "path": "/api/v1/students/1000"
}
```

Shrnutí

V testování byly identifikovány následující problémy:

- Metoda GET nevrací očekávaný stavový kód 404 Not Found pro neexistující záznamy, místo toho vrací 500 Internal Server Error.
- Metoda POST nedodržuje požadavek na ukládání hodnot `firstName` a `lastName` jako lowercase.
- Metoda POST a DELETE neposkytují správné potvrzující zprávy a nesprávné stavové kódy po úspěšném provedení operace.

Prosím, opravte tyto chyby, aby byla zajištěna správná funkcionální a očekávané chování aplikace.