FINÁLNÍ PROJEKT
č.1

Autor: Lucie Mercálková
Datum:

OBSAH

ZADÁNÍ

Cílem finálního projektu je otestovat funkčnost aplikace, která slouží k manipulaci s daty o studentech, vše zadané se má v databázi a aplikaci zobrazit jako lower case. 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.

1. TESTOVACÍ SCÉNÁŘ: ZÍSKÁNÍ EXISTUJÍCÍCH DAT METODOU GET

Popis: Test ověří, zda metoda GET správně získává data existujícího studenta na základě zadaného identifikátoru.

Kroky:
1. MySQL Workbench
   - Připojení k databázi obsahující tabulku “student”
   - Ověření existence záznamu studenta s identifikátorem “570”
   - Příkaz: SELECT * FROM student WHERE id = 570;

2. Otevření aplikace Postman
3. Vytvoření nového požadavku typu GET pro získání dat studenta
   - Zadání URL rozhraní pro získání dat studenta s id “570”
   - http://108.143.193.45:8080/api/v1/students/570
   - Volba metody GET
4. Odeslání GET požadavku pro získání dat studenta
5. Přijetí odpovědi od REST rozhraní
6. Zobrazení stavového kódu odpovědi

Očekávaný výsledek:
- Status kód odpovědi 200 OK
- Informace o studentovi:

"id": 570,
"firstName": "Klára",
"lastName": "Dogová",
"email": "post5@posta.cz",
"age": 25

2. TESTOVACÍ SCÉNÁŘ: ZÍSKÁNÍ NEEXISTUJÍCÍCH DAT METODOU GET

Popis: Test ověří, zda metoda GET správně reaguje na požadavek pro získání dat neexistujícího studenta.

Kroky:
1. MySQL Workbench
   - Připojení k databázi obsahující tabulku “student”
   - Ověření existence záznamu studenta s identifikátorem “1”
   - Příkaz: SELECT * FROM student WHERE id = 1;

2. Otevření aplikace Postman
3. Vytvoření nového požadavku typu GET pro získání dat studenta
   - Zadání URL rozhraní pro získání dat studenta s id “1”
     http://108.143.193.45:8080/api/v1/students/1
   - Volba metody GET
4. Odeslání GET požadavku pro získání dat studenta
5. Přijetí odpovědi od REST rozhraní

6. Zobrazení stavového kódu odpovědi

Očekávaný výsledek:
- Status kód odpovědi 404 Not Found
- Odpovídající chybové hlášení, které informuje uživatele o tom, že student s daným identifikátorem neexistuje.

3. TESTOVACÍ SCÉNÁŘ: VYTVOŘENÍ NOVÉHO ZÁZNAMU METODOU POST

Popis: Test ověří základní funkčnost metody POST vytvořením nového záznamu pomocí POST požadavku a ověří, zda je vytvoření záznamu úspěšné. Vše zadané se má v databázi a aplikaci zobrazit jako lower case.

Kroky:
1. Otevření aplikace Postman
2. Sestavení povinných údajů pro záznam studenta - “age”, “email”, “first_name”, “last_name”
3. Vstup:

```
{
"firstName": "jana",
"lastName": "kratochvílová",
"age":46,
"email": "email@email.cz"
}
```

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

4. Vytvoření nového záznamu v databázi metodou POST
5. New(HTTP)-POST-Body(raw,JSON)
6. Kliknout na “Send”

Očekávaný výsledek:
- Status kód odpovědi 200 OK / 201 Created
- Vytvořený záznam je přítomen v databázi a obsahuje správná data

id	age	email	first_name	last_name
595	46	email@email.cz	jana	kratochvílová

4. TESTOVACÍ SCÉNÁŘ: TEST SPRÁVNOSTI VSTUPNÍCH DAT

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

Kroky:
1. Otevření aplikace Postman
2. Sestavení povinných údajů pro záznam studenta - “age”, “email”, “first_name”, “last_name”

3. Vstup:

```
{
"firstName": "12345",
"lastName": “@123",
"age":46,
"email": "email@email.cz"
}
```

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

4. Vytvoření nového záznamu v databázi metodou POST
5. New(HTTP)-POST-Body(raw,JSON)
6. Kliknout na “Send”

Očekávaný výsledek: Objeví se odpovídající chybové kódy a chybová hláška na nesprávný formát dat u vstupu.

5. TESTOVACÍ SCÉNÁŘ: TEST SPRÁVNOSTI VSTUPNÍCH DAT II

Popis: Test ověří, že API správně validuje data zadáním nesprávného formátu dat u “age”.
Kroky:
1. Otevření aplikace Postman
2. Sestavení povinných údajů pro záznam studenta - “age”, “email”, “first_name”, “last_name”
3. Vstup:

```
{
"firstName": "ilona",
"lastName": “marek",
"age":lump,
"email": “posta@post.cz"
}
```

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

4. Vytvoření nového záznamu v databázi metodou POST
5. New(HTTP)-POST-Body(raw,JSON)
6. Kliknout na “Send”

Očekávaný výsledek: Objeví se odpovídající chybové kódy a chybová hláška na nesprávný formát dat u vstupu.

6. TESTOVACÍ SCÉNÁŘ: TEST UPOZORNĚNÍ NA CHYBĚJÍCÍ POVINNÝ ÚDAJ

Popis: Test ověří, že při zadávání nového záznamu musí být vyplněny všechny povinné údaje: “age", “email”, “first_name”, “last_name”.
Kroky:
1. Otevření aplikace Postman
2. Sestavení povinných údajů pro záznam studenta - “age”, “email”, “first_name”, “last_name”
3. Při zadávání vstupů vynechat povinný údaj “last_name”
4. Vstup:

{
"firstName": "petr",
"lastName": “ ",
"age":26,
"email": “posta@post.cz"
}

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

4. Vytvoření nového záznamu v databázi metodou POST
5. New(HTTP)-POST-Body(raw,JSON)
6. Kliknout na “Send”
7. Test opakovat vynecháním dalších povinných údajů

Očekávaný výsledek: Objeví se odpovídající chybové kódy a chybová hláška na chybějící povinný údaj.

7. TESTOVACÍ SCÉNÁŘ: TEST SPRÁVNÉHO FORMÁTU EMAIL
Popis: Test ověří, že API správně validuje data zadáním nesprávného formátu dat u “email”.
Kroky:
1. Otevření aplikace Postman
2. Sestavení povinných údajů pro záznam studenta - “age”, “email”, “first_name”, “last_name”
3. Při zadávání vstupů zadat nesprávný formát u vstupu “email”
4. Vstup:

{
"firstName": "josef",
"lastName": “konopa ",
"age":31,
"email": “postapost.cz"
}

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

4. Vytvoření nového záznamu v databázi metodou POST
5. New(HTTP)-POST-Body(raw,JSON)
6. Kliknout na “Send”

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

8. TESTOVACÍ SCÉNÁŘ: POKUS O VLOŽENÍ PRÁZDNÉHO ZÁZNAMU
Popis: Test ověří, že při zadávání nového záznamu musí být vyplněny všechny povinné údaje: “age", “email”, “first_name”, “last_name”.
Kroky:
1. Otevření aplikace Postman
2. Sestavení povinných údajů pro záznam studenta - “age”, “email”, “first_name”, “last_name”
3. Při zadávání vstupů vynechat všechny údaje
4. Vstup:

{
"firstName": "",
"lastName": “ ",
"age":,
"email": “"
}

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

4. Vytvoření nového záznamu v databázi metodou POST
5. New(HTTP)-POST-Body(raw,JSON)
6. Kliknout na “Send”

Očekávaný výsledek: Objeví se odpovídající chybové kódy a chybová hláška na chybějící povinný údaj.

9. TESTOVACÍ SCÉNÁŘ: TEST POVINNÉHO ÚDAJE AGE

Popis: Test ověří, že API správně validuje data vynecháním povinného údaje “age”.

Kroky:

1. Otevření aplikace Postman
2. Sestavení povinných údajů pro záznam studenta - “age”, “email”, “first_name”, “last_name”
3. Při zadávání vstupů vynechat povinný údaj “age”
4. Vstup:

{
"firstName": "josefa",
"lastName": “kryšotová ",
“age": ,
"email": “josepha@saint.tr“
}

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

4. Vytvoření nového záznamu v databázi metodou POST
5. New(HTTP)-POST-Body(raw,JSON)
6. Kliknout na “Send”

Očekávaný výsledek: Objeví se odpovídající chybové kódy a chybová hláška na chybějící povinný údaj.

10. TESTOVACÍ SCÉNÁŘ: SMAZÁNÍ EXISTUJÍCÍCH DAT

Popis: Test ověří, zda metoda DELETE správně vymaže data existujícího studenta na základě zadaného identifikátoru.

Kroky:

1. MySQL Workbench
   - Připojení k databázi obsahující tabulku “student”
   - Ověření existence záznamu studenta s identifikátorem “733”
   - Příkaz: SELECT * FROM student WHERE id = 733;

2. Otevření aplikace Postman
3. Vytvoření nového požadavku typu DELETE pro smazání dat studenta
   - Zadání URL rozhraní pro získání dat studenta s id “733”

http://108.143.193.45:8080/api/v1/students/733
- Volba metody DELETE

4. Odeslání DELETE požadavku pro smazání dat studenta
5. Přijetí odpovědi od REST rozhraní
6. Zobrazení stavového kódu odpovědi
7. Ověření smazání dat studenta v databázi

Očekávaný výsledek:
- Status kód odpovědi 200 OK

11. TESTOVACÍ SCÉNÁŘ: SMAZÁNÍ NEEXISTUJÍCÍCH DAT
Popis: Test ověří, zda metoda DELETE správně reaguje na pokus o opakované smazání dat u již neexistujícího studenta.
Kroky:
1. MySQL Workbench
   - Připojení k databázi obsahující tabulku “student”
   - Ověření smazání záznamu studenta s identifikátorem “733”
   - Příkaz: SELECT * FROM student WHERE id = 733;

2. Otevření aplikace Postman
3. Vytvoření nového požadavku typu DELETE pro smazání dat studenta
   - Zadání URL rozhraní pro získání dat studenta s id “733”
     http://108.143.193.45:8080/api/v1/students/733
   - Volba metody DELETE
4. Odeslání DELETE požadavku pro smazání dat studenta
5. Přijetí odpovědi od REST rozhraní
6. Zobrazení stavového kódu odpovědi

Očekávaný výsledek:
- Status kód odpovědi 404 Not Found

EXEKUCE TESTŮ

Testovací scénáře jsem provedla, přikládám výsledky testů.

Testovací scénář	Test proběhl úspěšně ANO/NE
1.Získání existujících dat metodou GET	ANO
2.Získání neexistujících dat metodou GET	NE
3.Vytvoření nového záznamu metodou POST	NE
4.Test správnosti vstupních dat	NE
5.Test správnosti vstupních dat II	ANO
6.Test upozornění na chybějící povinný údaj	NE
7.Test správného formátu email	NE
8.Pokus o vložení prázdného záznamu	ANO

Testovací scénář	Test proběhl úspěšně ANO/NE
9.Test povinného údaje age	ANO
10.Smazání existujících dat	ANO
11.Smazání neexistujících dat	NE

BUG REPORT

Na základě provedených scénářů jsem objevila uvedené chyby aplikace.

1. CHYBA VELIKOSTI PÍSMEN U "LAST_NAME"
Abstract: Aplikace mění u "last_name" velikost písmen na UPPER CASE.

Kroky: 1. Otevření aplikace Postman
2. Vytvoření nového záznamu v databázi metodou POST
3. New(HTTP)-POST-Body(raw,JSON)
4. Vstup:

```
{
"firstName": "Klára",
"lastName": "Dogová",
"age":25,
"email": "post5@posta.cz"
}
```

5. Kliknout na "Send"

Očekávaný výsledek: Aplikace uloží záznam o studentovi dle požadavku lower case.
Skutečný výsledek: Aplikace uloží záznam "last_name" UPPER CASE.

id	age	email	first_name	last_name
570	25	post5@posta.cz	Klára	DOGOVÁ

2. CHYBA VELIKOSTI PÍSMEN U "FIRST_NAME"
Abstract: Aplikace ukládá záznam "first_name" s velkým počátečním písmenem.
Požadavek: vše se má zobrazit jako lower case.
Kroky: 1. Otevření aplikace Postman
2. Vytvoření nového záznamu v databázi metodou POST
3. New(HTTP)-POST-Body(raw,JSON)
4. Vstup:

```
{
"firstName": "Klára",
"lastName": "Dogová",
"age":25,
"email": "post5@posta.cz"
}
```

5. Kliknout na “Send”

Očekávaný výsledek: Aplikace uloží záznam o studentovi dle požadavku lower case.
Skutečný výsledek: Aplikace uloží záznam “first_name” s velkým počátečním písmenem.

3. CHYBA STAVOVÉHO KÓDU 404 METODA GET

Abstract: Při testování metody GET na získání dat o neexistujícím studentovi se objeví stavový kód 500 Internal Server Error.

Kroky:

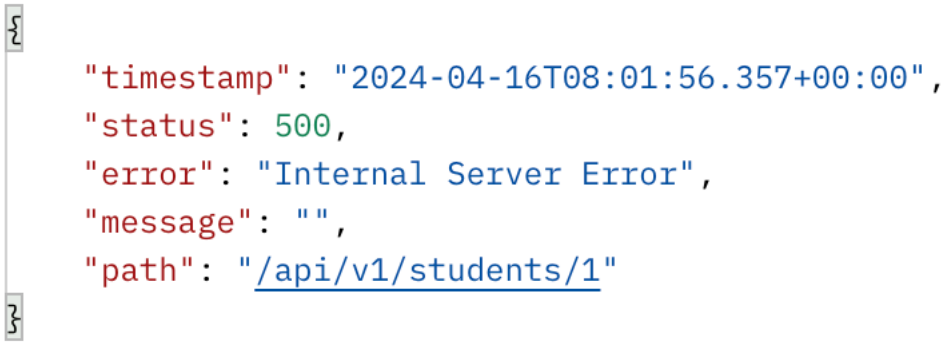
1. MySQL Workbench
- Připojení k databázi obsahující tabulku “student”
- Ověření existence záznamu studenta s identifikátorem “1”
- Příkaz: SELECT * FROM student WHERE id = 1;

2. Otevření aplikace Postman
3. Vytvoření nového požadavku typu GET pro získání dat studenta

- Zadání URL rozhraní pro získání dat studenta s id “1”
  http://108.143.193.45:8080/api/v1/students/1
- Volba metody GET

4. Odeslání GET požadavku pro získání dat studenta
5. Přijetí odpovědi od REST rozhraní

Očekávaný výsledek: Stavový kód odpovědi 404 Not Found a odpovídající chybové hlášení, které informuje uživatele o tom, že student s daným identifikátorem neexistuje.
Skutečný výsledek: Zobrazí se stavový kód 500 Internal Server Error.



4.POVOLENÍ DUPLICITY

Abstract: Při testování metody POST se ukládají data, která v databázi již existují.

Kroky:

1. Otevření aplikace Postman
2. Sestavení povinných údajů pro záznam studenta - “age”, “email”, “first_name”, “last_name”
3. Vstup:

```
{
"firstName": "jana",
"lastName": "kratochvílová",
```

"age":46,
"email": "email@email.cz"
}

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

4. Vytvoření nového záznamu v databázi metodou POST
5. New(HTTP)-POST-Body(raw,JSON)
6. Kliknout na “Send”
7. Postup opakovat se stejnými vstupy

Očekávaný výsledek: Status kód odpovědi 409 Conflict a odpovídající chybové hlášení, že student se stejnými zadanými údaji již existuje.
Skutečný výsledek: Stejná data lze použít několikrát.

id	age	email	first_name	last_name
595	46	email@email.cz	jana	KRATOCHVÍLOVÁ
596	46	email@email.cz	jana	KRATOCHVÍLOVÁ
597	46	email@email.cz	jana	KRATOCHVÍLOVÁ
598	46	email@email.cz	jana	KRATOCHVÍLOVÁ
599	46	email@email.cz	jana	KRATOCHVÍLOVÁ

5. NESPRÁVNÝ FORMÁT DAT
Abstract: Při testování metody POST se data u vstupu “first_name” a “last_name” ukládají v nesprávném formátu.
Kroky:
1. Otevření aplikace Postman
2. Sestavení povinných údajů pro záznam studenta - “age”, “email”, “first_name”, “last_name”
3. Vstup:

{
"firstName": "12345",
"lastName": “@123",
"age":46,
"email": "email@email.cz"
}

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

4. Vytvoření nového záznamu v databázi metodou POST
5. New(HTTP)-POST-Body(raw,JSON)
6. Kliknout na “Send”

Očekávaný výsledek: Objeví se odpovídající chybové kódy a chybová hláška na nesprávný formát dat u vstupu.

Skutečný výsledek: Status kód 200 OK a záznam se uloží do databáze.

id	age	email	first_name	last_name
611	46	email@email.cz	12345	@123

6. UKLÁDÁNÍ ZÁZNAMŮ BEZ POVINNÝCH ÚDAJŮ
Abstract: Při testování metody POST se záznam ukládá bez vyplnění povinných údajů “first_name”, “last_name” a “email”.
Kroky:
1. Otevření aplikace Postman
2. Sestavení povinných údajů pro záznam studenta - “age”, “email”, “first_name”, “last_name”
3. Při zadávání vstupů vynechat povinný údaj “last_name”
4. Vstup:

```
{
"firstName": "petr",
"lastName": “ ",
"age":26,
"email": “posta@post.cz"
}
```

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

4. Vytvoření nového záznamu v databázi metodou POST
5. New(HTTP)-POST-Body(raw,JSON)
6. Kliknout na “Send”

Očekávaný výsledek: Objeví se odpovídající chybové kódy a chybová hláška na chybějící povinný údaj.
Skutečný výsledek: Status kód 200 OK a záznam se uloží do databáze bez povinného údaje.

id	age	email	first_name	last_name
612	26	posta@post.cz	petr	

7. UKLÁDÁNÍ ZÁZNAMŮ VE ŠPATNÉM FORMÁTU
Abstract: Při testování metody POST se záznam ukládá ve špatném formátu “email”.
Kroky:
1. Otevření aplikace Postman
2. Sestavení povinných údajů pro záznam studenta - “age”, “email”, “first_name”, “last_name”
3. Při zadávání vstupů zadat nesprávný formát u vstupu “email”

4. Vstup:

```
{
"firstName": "josef",
"lastName": “konopa ",
"age":31,
"email": “postapost.cz"
}
```

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

4. Vytvoření nového záznamu v databázi metodou POST
5. New(HTTP)-POST-Body(raw,JSON)
6. Kliknout na “Send”

Očekávaný výsledek: Objeví se odpovídající chybové kódy a chybová hláška na nesprávný formát emailu.
Skutečný výsledek: Záznam se uloží i přes nesprávný formát emailu.

id	age	email	first_name	last_name
613	31	postapost.cz	josef	KONOPA

8. CHYBA STAVOVÉHO KÓDU 404 METODA DELETE
Abstract: Při testování metody DELETE pro smazání dat o neexistujícím studentovi se objeví stavový kód 500 Internal Server Error.
Kroky:

1. MySQL Workbench

- Připojení k databázi obsahující tabulku “student”
- Ověření smazání záznamu studenta s identifikátorem “733”
- Příkaz: SELECT * FROM student WHERE id = 733;

2. Otevření aplikace Postman
3. Vytvoření nového požadavku typu DELETE pro smazání dat studenta

- Zadání URL rozhraní pro získání dat studenta s id “733”
  http://108.143.193.45:8080/api/v1/students/733
  Volba metody DELETE

4. Odeslání DELETE požadavku pro smazání dat studenta
5. Přijetí odpovědi od REST rozhraní

Očekávaný výsledek: Stavový kód odpovědi 404 Not Found a odpovídající chybové hlášení, které informuje uživatele o tom, že student s daným identifikátorem neexistuje.
Skutečný výsledek: Zobrazí se stavový kód 500 Internal Server Error.


```
{
    "timestamp": "2024-04-23T09:13:53.502+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "message": "",
    "path": "/api/v1/students/733"
}
```