FINÁLNÍ PROJEKT
č.1

Autor: Walter Pisk
Datum: 5.12.2024

OBSAH

ZADÁNÍ **3**
TESTOVACÍ SCÉNÁŘE **4**
EXEKUCE TESTŮ **12**
BUG REPORT **12**

ZADÁNÍ

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

Přístupové údaje:

Databáze	**Default scheme**: qa_demo **Host**: aws.connect.psdb.cloud **Port**: 3306
REST-API	http://108.143.193.45:8080/api/v1/students/

Poznámky:

Nezapomeňte, že v IT se data musí někde uložit a poté získat. Proto ověřte, že data jsou správně uložena a získávána z databáze.
Nezapomeňte do testovacích scénářů uvést testovací data, očekávaný výsledek včetně těla odpovědi a stavových kódů.

TESTOVACÍ SCÉNÁŘE

Na základě uvedených testovacích scénářů jsem ověřil funkčnost aplikace.

Společný předpoklad pro exekuci všech uvedených testovacích scénářů:
Aplikace pro správu studentů je spuštěna a API je dostupné na adrese http://108.143.193.45:8080/api/v1/students/.

Testovací scénář 1: Ověření záznamu studenta pomocí metody GET (EXISTUJÍCÍ záznam)

Předpoklady

- Existuje minimálně jeden záznam studenta v databázi.
- ID testovaného záznamu studenta je **354** (získáno předem výpisem záznamů všech studentů z databáze v nástroji Postman).

Krok 1: Odeslání GET požadavku v nástroji Postman

1. V okně požadavku nastavit metodu GET a zadat URL pro API endpoint: http://108.143.193.45:8080/api/v1/students/354
2. Odeslat požadavek tlačítkem Send.

Krok 2: Ověření odpovědi

1. Očekávaný stavový kód: 200 OK.
2. Očekávaná odpověď: Odpověď by měla obsahovat objekt se záznamem studenta:

```
{
    "id": 354,
    "firstName": "alfred",
    "lastName": "HITCHCOCK",
    "email": "alfred@mail.com",
    "age": 18
}
```

3. Ověřit, že položky odpovědi (id, firstName, lastName, email, age) odpovídají očekávaným hodnotám - skutečnému záznamu v databázi, a to pomocí

nástroje MySQL Workbench přímým dotazem do databáze (SELECT * FROM student where id=354;)

Krok 3: Zpracování výsledků testu

1. Pokud údaje odpovídají očekáváním, test je úspěšný.
2. Zaznamenat chybu pokud údaje neodpovídají.

Testovací scénář 2: Ověření záznamu studenta pomocí metody GET (NEEXISTUJÍCÍ záznam)

Předpoklady

- ID testovaného záznamu studenta je **99999** (předem ověřeno výpisem v nástroji Postman, že záznam s tímto ID v databázi neexistuje).

Krok 1: Odeslání GET požadavku v nástroji Postman

1. V okně požadavku nastavit metodu GET a zadat URL pro API endpoint: http://108.143.193.45:8080/api/v1/students/99999
2. Odeslat požadavek tlačítkem Send.

Krok 2: Ověření odpovědi

1. Očekávaný stavový kód: 404 Not Found.
2. Očekávaná odpověď: Chybová odpověd o neexistujícím záznamu.
3. Ověřit přímým dotazem do databáze v nástroji MySQL Workbench, že záznam s daným ID opravdu v databázi neexistuje (SELECT * FROM student where id=99999;)

Krok 3: Zpracování výsledků testu

1. Pokud údaje odpovídají očekáváním, test je úspěšný.
2. Zaznamenat chybu pokud údaje neodpovídají.

Testovací scénář 3: Ověření záznamu studenta pomocí metody GET (NEVALIDNÍ formát ID záznamu)

Předpoklady

- ID testovaného záznamu studenta bude **aaa** (není validní formát id).

Krok 1: Odeslání GET požadavku v nástroji Postman

1. V okně požadavku nastavit metodu GET a zadat URL pro API endpoint: http://108.143.193.45:8080/api/v1/students/aaa
2. Odeslat požadavek tlačítkem Send.

Krok 2: Ověření odpovědi

1. Očekávaný stavový kód: 400 Bad Request.
2. Očekávaná odpověď: Chybová odpověď, že zaslaný požadavek nebyl zpracován kvůli nesprávné syntaxi (chyba na straně klienta).

Krok 3: Zpracování výsledků testu

1. Pokud údaje odpovídají očekáváním, test je úspěšný.
2. Zaznamenat chybu pokud údaje neodpovídají.

Testovací scénář 4: Vytvoření záznamu studenta pomocí metody POST (NOVÝ záznam s VALIDNÍMI ÚDAJI)

Krok 1: Odeslání POST požadavku v nástroji Postman

1. V okně požadavku nastavit metodu POST a zadat URL pro API endpoint: http://108.143.193.45:8080/api/v1/students/
2. V záložce Body vyplnit údaje nového záznamu (formát JSON):

```json
{
    "firstName":"Walter",
    "lastName": "Pisk",
    "email": "walter@pisk.cz",
    "age": 99
}
```

3. Odeslat požadavek tlačítkem Send.

Krok 2: Ověření odpovědi

1. Očekávaný stavový kód: 200 OK.
2. Očekávaná odpověď: Odpověď by měla obsahovat objekt s nově vytvořeným záznamem studenta a přiděleným unikátním ID:

3. Ověřit, že položky odpovědi (id, firstName, lastName, email, age) odpovídají očekávaným hodnotám - nově vytvořenému záznamu v databázi, a to pomocí nástroje MySQL Workbench přímým dotazem do databáze (SELECT * FROM student where id=2585;)

Krok 3: Zpracování výsledků testu

1. Pokud údaje odpovídají očekáváním, test je úspěšný.
2. Zaznamenat chybu pokud údaje neodpovídají.

Testovací scénář 5: Vytvoření záznamu studenta pomocí metody POST (NOVÝ záznam s NEVALIDNÍMI ÚDAJI - zcela chybí jedna z položek záznamu)

Krok 1: Odeslání POST požadavku v nástroji Postman

1. V okně požadavku nastavit metodu POST a zadat URL pro API endpoint: http://108.143.193.45:8080/api/v1/students/
2. V záložce Body vyplnit údaje nového záznamu **bez položky "age"** (formát JSON):

3. Odeslat požadavek tlačítkem Send.

Krok 2: Ověření odpovědi

1. Očekávaný stavový kód: 400 Bad Request.
2. Očekávaná odpověď: Chybová odpověď, že zaslaný požadavek nebyl zpracován kvůli nesprávné syntaxi (chyba na straně klienta).

Krok 3: Zpracování výsledků testu

1. Pokud údaje odpovídají očekáváním, test je úspěšný.
2. Zaznamenat chybu pokud údaje neodpovídají.

Testovací scénář 6: Vytvoření záznamu studenta pomocí metody POST (NOVÝ záznam s NEVALIDNÍMI ÚDAJI - užití znaků, které určité položky standardně neobsahují)

Krok 1: Odeslání POST požadavku v nástroji Postman

1. V okně požadavku nastavit metodu POST a zadat URL pro API endpoint: http://108.143.193.45:8080/api/v1/students/
2. V záložce Body vyplnit údaje nového záznamu; **u položky "firstName" použijeme speciální znaky a její obsah bude "#$~^&"** (formát JSON):

3. Odeslat požadavek tlačítkem Send.

Krok 2: Ověření odpovědi

1. Očekávaný stavový kód: 400 Bad Request.
2. Očekávaná odpověď: Chybová odpověď, že zaslaný požadavek nebyl zpracován kvůli nesprávné syntaxi (resp. chybě na straně klienta).

Krok 3: Zpracování výsledků testu

1. Pokud údaje odpovídají očekáváním, test je úspěšný.
2. Zaznamenat chybu pokud údaje neodpovídají.

Testovací scénář 7: Vytvoření záznamu studenta pomocí metody POST (NOVÝ záznam s NEVALIDNÍMI ÚDAJI - užití nekorektních nebo nesmyslných hodnot)

Krok 1: Odeslání POST požadavku v nástroji Postman

1. V okně požadavku nastavit metodu POST a zadat URL pro API endpoint: http://108.143.193.45:8080/api/v1/students/
2. V záložce Body vyplnit údaje nového záznamu; **u položky "age" použijeme nestandardní (zápornou, navíc pro lidský věk nereálně vysokou) hodnotu a její obsah bude "-6789"** (formát JSON):

3. Odeslat požadavek tlačítkem Send.

Krok 2: Ověření odpovědi

1. Očekávaný stavový kód: 400 Bad Request.
2. Očekávaná odpověď: Chybová odpověď, že zaslaný požadavek nebyl zpracován kvůli nesprávné syntaxi (resp. chybě na straně klienta).

Krok 3: Zpracování výsledků testu

1. Pokud údaje odpovídají očekáváním, test je úspěšný.
2. Zaznamenat chybu pokud údaje neodpovídají.

Testovací scénář 8: Vytvoření záznamu studenta pomocí metody POST (NOVÝ záznam s NEVALIDNÍMI ÚDAJI - užití prázdných hodnot)

Krok 1: Odeslání POST požadavku v nástroji Postman

1. V okně požadavku nastavit metodu POST a zadat URL pro API endpoint: http://108.143.193.45:8080/api/v1/students/
2. V záložce Body vyplnit údaje nového záznamu; **u položky "firstName" a "lastName" nebude vyplněn žádný obsah** (formát JSON):

3. Odeslat požadavek tlačítkem Send.

Krok 2: Ověření odpovědi

1. Očekávaný stavový kód: 400 Bad Request.
2. Očekávaná odpověď: Chybová odpověď, že zaslaný požadavek nebyl zpracován kvůli nesprávné syntaxi (resp. chybě na straně klienta).

Krok 3: Zpracování výsledků testu

1. Pokud údaje odpovídají očekáváním, test je úspěšný.
2. Zaznamenat chybu pokud údaje neodpovídají.

Testovací scénář 9: Smazání záznamu studenta pomocí metody DELETE (EXISTUJÍCÍ záznam)

Předpoklady

- ID testovaného záznamu studenta je **2604** (záznam vytvořen předem pro účely tohoto scénáře v nástroji Postman).

Krok 1: Odeslání DELETE požadavku v nástroji Postman

1. V okně požadavku nastavit metodu DELETE a zadat URL pro API endpoint: http://108.143.193.45:8080/api/v1/students/2604
2. Odeslat požadavek tlačítkem Send.

Krok 2: Ověření odpovědi

1. Očekávaný stavový kód: 200 OK.
2. Očekávaná odpověď: Odpověď by měla obsahovat prázdný objekt.
3. Ověřit pomocí nástroje MySQL Workbench přímým dotazem do databáze, že záznam byl skutečně smazán (SELECT * FROM student where id=2604;)

Krok 3: Zpracování výsledků testu

1. Pokud údaje odpovídají očekáváním, test je úspěšný.

2. Zaznamenat chybu pokud údaje neodpovídají.

Testovací scénář 10: Smazání záznamu studenta pomocí metody DELETE (NEEXISTUJÍCÍ záznam)

Předpoklady

- ID testovaného záznamu studenta je **99999** (předem ověřeno výpisem v nástroji Postman, že záznam s tímto ID v databázi neexistuje).

Krok 1: Odeslání DELETE požadavku v nástroji Postman

1. V okně požadavku nastavit metodu DELETE a zadat URL pro API endpoint: http://108.143.193.45:8080/api/v1/students/99999
2. Odeslat požadavek tlačítkem Send.

Krok 2: Ověření odpovědi

1. Očekávaný stavový kód: 404 Not Found.
2. Očekávaná odpověď: Chybová odpověd o neexistujícím záznamu.
3. Ověřit přímým dotazem do databáze v nástroji MySQL Workbench, že záznam s daným ID opravdu v databázi neexistuje (SELECT * FROM student where id=99999;)

Krok 3: Zpracování výsledků testu

1. Pokud údaje odpovídají očekáváním, test je úspěšný.
2. Zaznamenat chybu pokud údaje neodpovídají.

EXEKUCE TESTŮ

Testovací scénáře jsem provedl, přikládám výsledky testů.

Testovací scénář č.	Test proběhl bez chyb?	V případě chyby záznam do bug reportu
1	ANO	-
2	NE	ANO
3	ANO	-
4	ANO	-
5	NE	ANO
6	NE	ANO
7	NE	ANO
8	NE	ANO
9	ANO	-
10	NE	ANO

BUG REPORT

Na základě provedených scénářů jsem objevil uvedené chyby aplikace.

bug	probability	severity	test case number	mitigace
Při volání neexistujícího záznamu pomocí metody GET aplikace vrací stavový kód 500 Internal Server Error, (má vracet 404 Not Found).			2	Oprava aplikace tak, aby vracela správný stavový kód pro neexistující záznam.
Při vytváření nového záznamu pomocí metody POST, kdy zcela chybí jedna z			5	Oprava aplikace tak, aby vracela správný stavový kód při špatné syntaxi

položek pro nový záznam (u testu položka “age”), aplikace vrací stavový kód 500 Internal Server Error, (má vracet 400 Bad Request).				požadavku na vytvoření nového záznamu.
Při vytváření nového záznamu pomocí metody POST, kdy je použito pro obsah u položky "firstName" speciálních znaků, které zpravidla jména neobsahují, aplikace záznam v pořádku uloží a vrátí 200 OK (má vracet 400 Bad Request).			6	Aplikace by měla testovat obsah zadávaných údajů pro záznamy a vyhodnocovat, zda jde o korektní obsah, tzn. jména by měla obsahovat písmenné znaky, věk by měl naopak obsahovat striktně čísla.
Při vytváření nového záznamu pomocí metody POST, kdy je použito pro obsah u položky "age" záporné a/nebo nereálně vysoké hodnoty věku, aplikace záznam v pořádku uloží a vrátí 200 OK (má vracet 400 Bad Request).			7	Aplikace by měla testovat obsah zadávaných údajů pro záznamy a vyhodnocovat, zda jde o korektní obsah, tzn. věk by měl být vždy kladné hodnoty a omezen shora hodnotou, které se lidé mohou skutečně dožít (tím vyloučit ukládání chybných či nesmyslných údajů).
Při vytváření nového záznamu pomocí metody POST, kdy je obsah položky "firstName" a “lastName” prázdný, aplikace záznam v pořádku uloží a vrátí 200 OK (má vracet 400 Bad Request).			8	Aplikace by měla testovat obsah zadávaných údajů pro záznamy a vyhodnocovat, zda jde o korektní obsah - předpokládáme, že každá osoba má jméno a příjmení a tudíž by aplikace neměla umožnit uložit data studenta bez obsahu těchto

				položek.
Při pokusu o smazání neexistujícího záznamu pomocí metody DELETE aplikace vrací stavový kód 500 Internal Server Error, (má vracet 404 Not Found).			10	Oprava aplikace tak, aby vracela správný stavový kód pro neexistující záznam.