FINÁLNÍ PROJEKT
č.1

Autor: Marie Toncarová
Datum: 05.01.2025

Obsah

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 scénaře a exekuce testů

Na základě následujících testovacích scénářů jsem ověřila funkčnost aplikace. Testy byly prováděny na systému Windows 11, verze 23H2, s aplikací Postman verze 11.18.0, architektura x64, platforma win32 10.0.22631. Pro každý testovací scénář je vytvořena přehledná tabulka, která zobrazuje jednotlivé testovací kroky a jejich vykonání. Každý test je podrobně popsán.

1. Otestování metody GET v Postman

a. Prvně otestujeme, zda funguje zobrazení všech údajů z dané tabulky. Jako odkaz k databázi a tabulce využijeme *http://108.143.193.45:8080/api/v1/students/* Tento požadavek by měl vrátit seznam všech studentů v tabulce students.
b. Dále otestujeme, zda funguje zobrazení záznamu uloženého pod určitým ID. Jako odkaz k databázi a tabulce využijeme *http://108.143.193.45:8080/api/v1/students/<id-studenta*>
   - Pozitivní testování: Otestujeme několik platných ID, která existují v tabulce, a ověříme, že server vrátí správné údaje pro každého studenta.
   - Negativní testování: Otestujeme neexistující ID, které v tabulce není, a ověříme, že server správně vrátí odpověď s kódem 404 Not Found, případně vrátí prázdné hodnoty
   - Negativní testování - Testování formátu ID: Otestujeme také neplatný formát ID (např. textové řetězce nebo speciální znaky) a ověříme, zda server správně vrátí chybu o špatném formátu ID, tedy status 400.

Test ID	Testovací scénář	Testovací kroky	Očekávaný výsledek	Skutečný výsledek	Stav
1	Otestování metody GET pro získání dat z tabulky students v databázi	1. Otevřít aplikaci Postman 2. vybrat metodu GET 3. okopírovat url ze zádání projektu odkazující na tabulku students z dané databáze, jež je žádoucí otestovat, do středového pole 4. kliknout na send	v zápatí stránky se objeví všechny údaje z tabulky students	v zápatí stránky se objevily všechny údaje z databáze students	Pass
2	Otestování metody GET pro získání dat vybraného studenta z tabulky students v databázi	1. Otevřít aplikaci Postman 2. vybrat metodu GET 3. okopírovat url ze zádání projektu odkazující na tabulku students z dané databáze a otestovat uživatele, který v databázi je, tedy např. ID 1050 4. kliknout na send	v zápatí stránky se objeví údaje uživatele s ID 1050 z tabulky students	v zápatí stránky se objevIily údaje uživatele s ID 1050 z databáze students	Pass
3	Negativní otestování metody GET pro získání dat neexistujícího studenta z tabulky students v databázi	1. Otevřít aplikaci Postman 2. vybrat metodu GET 3. okopírovat url ze zádání projektu odkazující na tabulku students z dané databáze a otestovat uživatele, který v databázi není, tedy např. ID 50 4. kliknout na send	v zápatí stránky se objeví buďto prázdné pole v JSON zobrazení {}, případně lépe chyba 404 Not Found	v zápatí stránky se objevila chyba 500 "Internal Server Error"	Fail
4	Negativní otestování metody GET s chybným formátem z tabulky students v databázi	1. Otevřít aplikaci Postman 2. vybrat metodu GET 3. okopírovat url ze zádání projektu odkazující na tabulku students z dané databáze a otestovat chybný formát uživatele, např. &50 4. kliknout na send	v zápatí stránky se objeví chyba 400 Bad Request	v zápatí stránky se objevila chyba 400 Bad Request	Pass

2. Otestování metody POST v Postman, od které se očekává, že založí záznam o studentovi

a. Otestujeme, zda funguje vložení nových parametrů do tabulky students pomocí metody POST. Jako odkaz k databázi a tabulce využijeme *http://108.143.193.45:8080/api/v1/students/*
Hodnoty budeme vkládat v JSON formátu do těla (body) požadavku:
{
"firstName":"Lord",
"lastName": "Voldemort",
"email": "tenjehozjmenosmimevyslovit@gmail.com",
"age": 60
}

b. Dále otestujeme, zda se uložila opravdu správná, shodná data. Po odeslání požadavku zkontrolujeme, zda se vrátí správná odpověď s příslušnými údaji, jako je ID studenta a vložené hodnoty.

c. pro kontrolu otestujeme, zda jsou data uložená v databázi přes aplikaci my SQL (s heslem uloženým na learning portále ENGETO) a využijeme SQL příkaz *select * from student where id=2676 (Tento příkaz nám ukáže všechny sloupce z tabulky students a zobrazí pouze údaje, které jsou vyplněné pro ID 2676)*

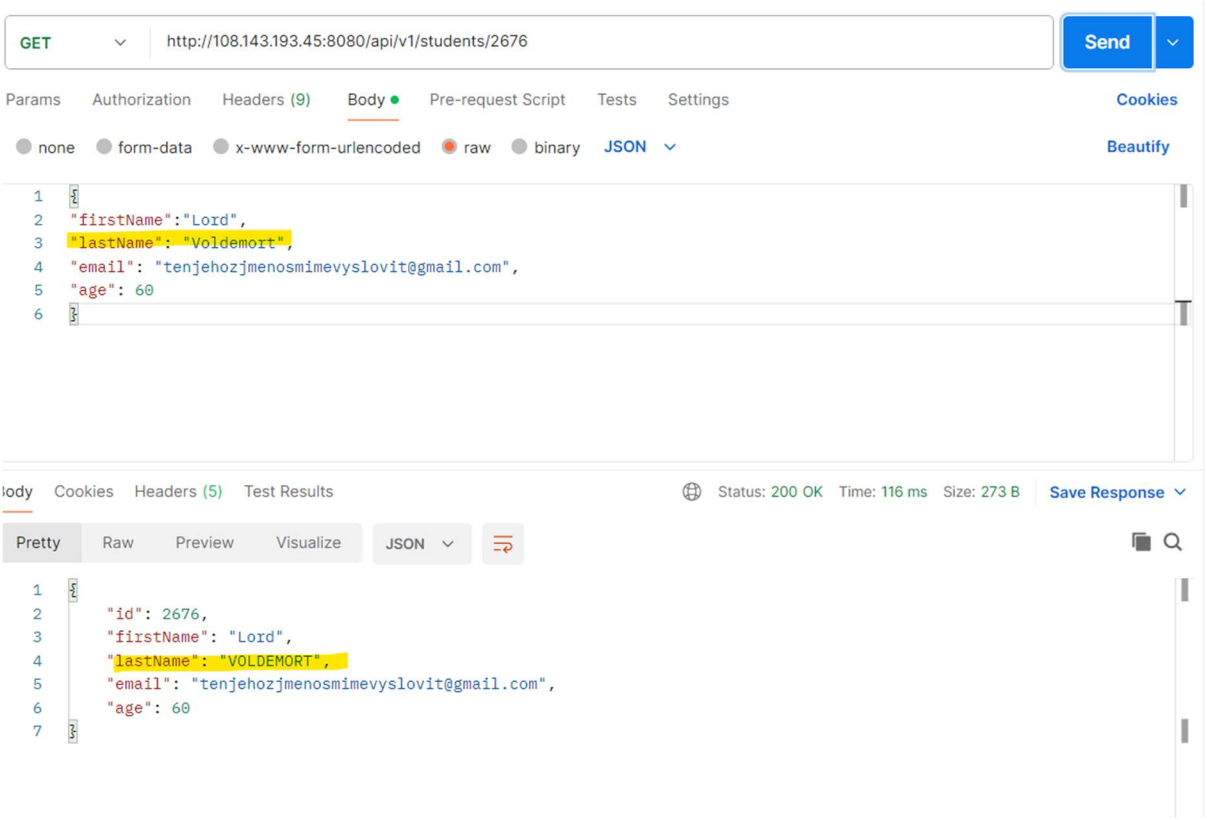
Test ID	Testovací scénář	Testovací kroky	Očekávaný výsledek	Skutečný výsledek	Stav

5	Otestování metody POST pro zaslání dat do tabulky students v databázi a přiřazení unikátního kódu	1. Otevřít aplikaci POSTMAN
		2. vybrat metodu POST
		3. okopírovat url ze zádání projektu odkazující na tabulku students z dané databáze a vložit do těla požadavku parametry {
		"firstName":"Lord",
		"lastName": "Voldemort",
		"email":
		"tenjehozjmenosmimevyslovit@gmail.com",
		"age": 60
		}
		4. kliknout na send
			založí se nový záznam s novým ID	založil se nový záznam a přiřadil ID 2676	Pass

6	Otestování metody POST pro zaslání správných dat do tabulky students v databázi	1, ověřit zda data v zápatí stránky jsou stejná jako data zadaná do body	data jsou shodná	parametry nejsou stejné jako jako zadávané parametry do těla, respektive liší se formát parametru → Přijmení používá namísto velkého písmena na počátku, všechna velká písmena	Fail
7	otestování metody POST - ověření nahraných dat	1. Otevřít aplikaci POSTMAN 2. vybrat metodu GET 3. okopírovat url ze zádání projektu odkazující na tabulku students z dané databáze a otestovat uživatele, který v databázi je, tedy např. ID 2676 4. kliknout na send	parametry jsou stejné jako jako zadávané parametry do těla za pomocí metody POST	parametry nejsou stejné jako jako zadávané parametry do těla, respektive liší se formát parametru → Přijmení používá namísto velkého písmena na počátku, všechna velká písmena	Fail
8	otestování metody POST - ověření nahraných dat	1. Otevřít aplikaci MySQL 2. vybrat stejnou databázi 3. otestovat, zda v tabulce students vznikl nový záznam s ID 2676 za pomoci příkazu select * from student where id=2676	objeví se výsledek shodný s parametry zadáváné v Postmanovi	objevil se výsledek s parametry zadáváné v Postmanovi	Pass

9	Otestování metody POST pro zaslání speciální znaků do tabulky students v databázi	1. Otevřít aplikaci POSTMAN
		2. vybrat metodu POST
		3. Ve formátu JSON zadej data do těla požadavku. Zkontroluj, zda lze do pole last name vložit speciální znaky, háčky a čárky. Například{
		"firstName":"Sirius",
		"lastName": "Černý125*.",
		"email": "siriusjepes@gmail.com",
		"age": 40
		}
		4. kliknout na send
			speciální znaky lze vložit do políčka last name	speciální znaky lze vložit do políčka last name, objevil se status 200 ok	Pass

Pro představu přikládám výstřižek chyby u případu Test ID 6/7 – hodnota u klíče „lastName“ je psána velkými písmeny, namísto pouze počátečního písmena.



3. Otestování metody DELETE pro smazání dat o studentovi

a. Otestujeme, zda metoda DELETE správně smaže data o studentovi z databáze.
   Jako odkaz k databázi a tabulce využijeme URL:
   http://108.143.193.45:8080/api/v1/students/2676
   Pro pozitivní testování: Zvolíme platné ID studenta, který existuje v tabulce, a otestujeme, zda metoda DELETE správně odstraní záznam o studentovi. Pokud se úspěšně odstraní měl by se zobrazit status 200.
b. Po provedení požadavku DELETE ověříme, že student již neexistuje v databázi. Použijeme metodu GET k ověření, zda se záznam o studentovi s daným ID skutečně vymazal. Server by měl vrátit odpověď 404 Not Found, protože student již neexistuje. Dále ověřím výmaz i přes MySQL.
c. Pro negativní testování: Zvolíme neplatné ID studenta, který neexistuje v tabulce, a otestujeme, zda metoda DELETE zahlásí status 404 Not Found.
d. Pro negativní testování: Zvolíme ID studenta ve špatném formátu, který neexistuje v tabulce, a otestujeme, zda metoda DELETE zahlásí status 400 – bad request

Test ID	Testovací scénář	Testovací kroky	Očekávaný výsledek	Skutečný výsledek	Stav	Poznámky
10	Otestování DELETE metody pro odstranění uživatele	1. Otevřít aplikaci POSTMAN 2. vybrat metodu DELETE 3. okopírovat url ze zádání projektu odkazující na tabulku students z dané databáze a otestovat uživatele, který v databázi je, tedy např. ID 2676 4. kliknout na send	údaj se smaže a zobrazí status 200.	údaj se smazal a zobrazil se status 200.	Pass	
11	otestování metody DELETE - ověření smazání uživatele	1. Otevřít aplikaci POSTMAN 2. vybrat metodu GET 3. okopírovat url ze zádání projektu odkazující na tabulku students z dané databáze a otestovat uživatele, který v databázi je, tedy např. ID 2676 4. kliknout na send	údaj se nezobrazí a zobrazí status 404	údaj se nezobrazil a ale ukázal status 500 Internal Server Error	Fail	
12	otestování metody DELETE - ověření smazání uživatele	1. Otevřít aplikaci MySQL 2. vybrat stejnou databázi 3. otestovat, zda v tabulce students je údaj opravdu smazán, tedy již není záznam s ID 2676 za pomoci příkazu select *	údaj se nezobrazí	údaj se nezobrazil a vrátil prázdné údaje	Pass	-

		from student where id=2676				
13	negativní otestování metody DELETE - ověření smazání neexistujícího uživatele	1. Otevřít aplikaci POSTMAN 2. vybrat metodu DELETE 3. okopírovat url ze zádání projektu odkazující na tabulku students z dané databáze a otestovat uživatele, který v databázi není, tedy např. již smazané ID 2676 4. kliknout na send	zobrazí se chyba 404, not found	zobrazil se status 500, Internal Server error	Fail	
14	negativní otestování metody DELETE - ověření smazání uživatele s chybným formátem ID	1. Otevřít aplikaci POSTMAN 2. vybrat metodu DELETE 3. okopírovat url ze zádání projektu odkazující na tabulku students z dané databáze a otestovat nějaké ID v chybném formátu, tedy. Např. ID *350 4. kliknout na send	zobrazí se status 400, bad request	zobrazil se status 400, bad request	pass	

BUG REPORT

Na základě provedených scénářů jsem objevila uvedené chyby aplikace.

bug	probability	severity	test ID	mitigace
Chyba 500 při GET pro neexistující ID	High	High	3	Opravit serverovou logiku pro správné zpracování žádostí o neexistující ID, aby vracel status 404 místo 500.
Nesoulad mezi parametry zadanými v POST a uloženými - lastname vrací převádí do velkých písmen	High	Low	6	Zkontrolovat, zda server neprovádí automatické změny (např. převod textu na velká písmena) při ukládání dat.
Nesoulad mezi parametry zadanými v POST a vrácenými při GET	High	Low	7	Zkontrolovat validaci a formátování dat na serveru, aby odpovídaly zadaným parametrům.
Chyba 500 při GET po smazání uživatele	High	High	11	Opravit logiku serveru pro správné zpracování GET požadavků po smazání uživatele (měla by vracet status 404).
Chyba 500 při DELETE pro neexistujícího uživatele	High	High	13	Opravit serverovou logiku pro správné zpracování DELETE požadavků pro neexistující uživatele, měla by vracet status 404 místo 500.