FINÁLNÍ PROJEKT   
 č.1



Autor:

Datum:

OBSAH

[**ZADÁNÍ**](#_heading=h.30j0zll) **3**

[**TESTOVACÍ SCÉNÁŘE**](#_heading=h.2et92p0) **4**

[**EXEKUCE TESTŮ**](#_heading=h.tyjcwt) **5**

[**BUG REPORT**](#_heading=h.3dy6vkm) **5**

# 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:

## 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.*

## TS-001 - získávání dat uživatelů)

**Popis**: Ověření, že můžeme získat data uživatelů

**Kroky testu**:

1. Odešlete **GET** request na /api/v1/students/

2. Ověřte, že se zobrazují data uživatelů

**Očekávaný výsledek**: API vrátí odpověď s kódem 200 a zobrazí data o uživatelích.

**Skutečný výsledek**: API vrátí odpověď s kódem 200 a zobrazí data o uživatelích.

**Stav**: Pass

## TS-002 – získání data neexistujícího uživatele za pomocí čísla)

**Popis**: Ověření, že API vrátí správný response

**Kroky testu**:

1. Odešlete **GET** request na /api/v1/students/**55555**

2. Ověřte, že se zobrazuje správná response

**Očekávaný výsledek**: API vrátí odpověď s kódem **404** a nezobrazí data o uživateli.

**Skutečný výsledek**: API vrátí odpověď s kódem **500** (**Internal server error**) a nezobrazí data o uživateli.

**Stav**: Fail

## TS-003 – získání data neexistujícího uživatele za pomocí speciálních znaků)

**Popis**: Ověření, že API vrátí správný response

**Kroky testu**:

1. Odešlete **GET** request na /api/v1/students/**!!**

2. Ověřte, že se zobrazuje správná response

**Očekávaný výsledek**: API vrátí odpověď s kódem **400** a nezobrazí data o uživateli.

**Skutečný výsledek**: API vrátí odpověď s kódem **400** a nezobrazí data o uživateli.

**Stav**: Pass

## TS-004 – získání data neexistujícího uživatele za pomocí mezery)

**Popis**: Ověření, že API vrátí správný response

**Kroky testu**:

1. Odešlete **GET** request na /api/v1/students/**%20 !(mezera)!**

2. Ověřte, že se zobrazuje správná response

**Očekávaný výsledek**: API vrátí odpověď s kódem **404** nebo **400** a nezobrazí data o uživateli.

**Skutečný výsledek**: API vrátí odpověď s kódem **500** (**Internal server error**) a nezobrazí data o uživateli.

**Stav**: Fail

## TS-005 – přidání uživatele bez (firstName, lastName, email), ale s věkem)

**Popis**: Ověření, že nemůžete přidat uživatele, který bude mít jen věk

**Kroky testu**:

1. Odešlete **POST**  request na /api/v1/students/

2. Použijte tento json

{

    "firstName": "",

    "lastName": "",

    "email": "",

    "age": 52

}

3. Ověřte, že se uživatel nepřidá

**Očekávaný výsledek**: API vrátí odpověď, že uživatel nelze přidat a uživatel se opravdu nepřidá.

**Skutečný výsledek**: API přidá uživatele.

**Stav**: Fail

## TS-006 – přeskočení ID po přidání uživatele s internal errorem)

**Popis**: Ověření, že vždy po přidání neúplného uživatele, který zapřičiní chybovou hlášku 500 tak se stejnak přidá mezi studenty a zabere tím jedno ID

**Kroky testu**:

1. Odešlete **POST**  request na /api/v1/students/

**2.** **Přidejte uživatele pomocí tohoto jsonu**  
{  
 "firstName": "",  
 "lastName": "",  
 "email": "",  
 "age": ""  
}

**3.** **Přidejte teď uživatele pomocí tohoto jsonu**  
{  
 "firstName": "",  
 "lastName": "",  
 "email": "",  
 "age": 52  
}

4. Odešlete **GET** request na /api/v1/students/

5. Ověřte, že poslední uživatel má správné ID (pokud uživatel před ním má ID 505 tak tento uživatel by měl mít ID 506)

**Očekávaný výsledek**: Nově přidaný uživatel bude mít ID hned jdoucí po předchozím uživatelem např. (předchozí uživatel má ID 505 tak tento uživatel bude mít ID 506).

**Skutečný výsledek**: Nově přidaný uživatel nebude mít ID hned jdoucí po předchozím uživatelem. Např. (předchozí uživatel má ID 505 tak tento uživatel bude mít ID 507), protože i přesto, že přidání uživatele zahlásí response **500** tak se přidá, ale není vidět.

**Stav**: Fail

## TS-007 – přidání uživatele s neplatným emailem)

**Popis**: Ověření, že vždy po přidání uživatele s neplatným emailem, tak se vypíše, že uživatel nelze přidat

**Kroky testu**:

1. Odešlete **POST**  request na /api/v1/students/

**2.** **Přidejte uživatele pomocí tohoto jsonu**  
{

    "firstName": "ok",

    "lastName": "sos",

    "email": "man",

    "age": 54

}  
3. Ověřte, že se uživatel nepřidá a API vrátí, že email není platný

**Očekávaný výsledek**: Uživatel se nepřidá a API vrátí, že email není platný.

**Skutečný výsledek**: Uživatel se přidá.

**Stav**: Fail

## TS-008 – přidání uživatele se všema hodnotama)

**Popis**: Ověření, že uživatel se přidá pokud pošleme všechny platný údaje

**Kroky testu**:

1. Odešlete **POST**  request na /api/v1/students/

**2.** **Přidejte uživatele pomocí tohoto jsonu**  
{

    "firstName": "Martin",

    "lastName": "Konečný",

    "email": "martin.konecny@seznam.cz",

    "age": 60

}  
3. Ověřte, že se uživatel přidá

**Očekávaný výsledek**: Uživatel se přidá.

**Skutečný výsledek**: Uživatel se přidá.

**Stav**: Pass

## TS-009 – přidání uživatele – jeho příjmení)

**Popis**: Ověření, že uživatelovo přijmení bude mít první písmeno velké, ale zbytek zůstane v LowerCase

**Kroky testu**:

1. Odešlete **POST**  request na /api/v1/students/

**2.** **Přidejte uživatele pomocí tohoto jsonu**  
{

    "firstName": "Martin",

    "lastName": "Konečný",

    "email": "martin.konecny@seznam.cz",

    "age": 60

}  
3. Ověřte, že uživatelovo příjmení zůstane ve stejným formátu jako v jsonu

**Očekávaný výsledek**: Uživatel se přidá a jeho příjmení bude mít první písmeno velké a zbytek zůstane malým.

**Skutečný výsledek**: Uživatel se přidá a jeho příjmení má všechna písmena velká

**Stav**: Fail

## TS-010 – smazání studenta bez ID)

**Popis**: Ověření, že se nesmažou všichni studenti pokud nezadáme ID studenta k smazání

**Kroky testu**:

1. Odešlete **DELETE**  request na /api/v1/students/  
2. Ověřte, že API vrátí správný response, že toto není možné

**Očekávaný výsledek**: Uživatel se nesmaže. API vrátí response **405** (**Method Not Allowed**).

**Skutečný výsledek**: Uživatel se nesmaže. API vrátí response **405** (**Method Not Allowed**).

**Stav**: Pass

## TS-011 – smazání uživatele za pomocí mezery)

**Popis**: Ověření, že se nesmaže žádný uživatel a API vrátí správný response

**Kroky testu**:

1. Odešlete **DELETE** request na /api/v1/students/**%20 !(mezera)!**

2. Ověřte, že se vrátí správná response a že se žádný uživatel nesmaže

**Očekávaný výsledek**: API vrátí odpověď s kódem **404** nebo **400** a nesmaže žádného uživatele

**Skutečný výsledek**: API vrátí odpověď s kódem **500** (**Internal server error**) a nesmaže žádného uživatele.

**Stav**: Fail

## TS-012 – smazání uživatele za pomocí mezery a ID)

**Popis**: Ověření, že se nesmaže žádný uživatel a API vrátí správný response

**Kroky testu**:

1. Odešlete **DELETE** request na /api/v1/students/ **2745 !(mezera)!+(existující ID)**

2. Ověřte, že se vrátí správná response a že se žádný uživatel nesmaže

**Očekávaný výsledek**: API vrátí odpověď s kódem **404** nebo **400** a nesmaže žádného uživatele.

**Skutečný výsledek**: API vrátí odpověď s kódem **200** a smaže uživatele s daným ID.

**Stav**: Fail

## TS-013 – smazání uživatele pomocí jeho ID)

**Popis**: Ověření, že se smaže uživatel z databáze

**Kroky testu**:

1. Odešlete **DELETE** request na /api/v1/students/2744

2. Ověřte, že se vrátí správná response a že uživatel smaže

**Očekávaný výsledek**: API vrátí odpověď s kódem **200** a smaže daného uživatele.

**Skutečný výsledek**: API vrátí odpověď s kódem **200** a smaže daného uživatele.

**Stav**: Pass

## TS-014 – smazání uživatele pomocí jeho jména)

**Popis**: Ověření, že se smaže uživatel z databáze

**Kroky testu**:

1. Odešlete **DELETE** request na /api/v1/students/**Martin**

2. Ověřte, že se vrátí správná response a že uživatel se nesmaže

**Očekávaný výsledek**: API vrátí odpověď s kódem **400** a nesmaže daného uživatele.

**Skutečný výsledek**: API vrátí odpověď s kódem **400** a nesmaže daného uživatele.

**Stav**: Pass

## TS-015 – smazání uživatele pomocí jeho příjmení)

**Popis**: Ověření, že se smaže uživatel z databáze

**Kroky testu**:

1. Odešlete **DELETE** request na /api/v1/students/**Konečný**

2. Ověřte, že se vrátí správná response a že uživatel se nesmaže

**Očekávaný výsledek**: API vrátí odpověď s kódem **400** a nesmaže daného uživatele.

**Skutečný výsledek**: API vrátí odpověď s kódem **400** a nesmaže daného uživatele.

**Stav**: Pass

## TS-016 – smazání uživatele pomocí jeho emailu)

**Popis**: Ověření, že se smaže uživatel z databáze

**Kroky testu**:

1. Odešlete **DELETE** request na /api/v1/students/**bc@gmail.com**

2. Ověřte, že se vrátí správná response a že uživatel se nesmaže

**Očekávaný výsledek**: API vrátí odpověď s kódem **400** a nesmaže daného uživatele.

**Skutečný výsledek**: API vrátí odpověď s kódem **400** a nesmaže daného uživatele.

**Stav**: Pass

## TS-017 – smazání uživatele pomocí neexistující path ale existujícího ID)

**Popis**: Ověření, že se uživatel nesmaže z databáze

**Kroky testu**:

1. Odešlete **DELETE** request na /api/v1/students/**2485**/**path**

2. Ověřte, že se vrátí správná response a že uživatel se nesmaže

**Očekávaný výsledek**: API vrátí odpověď s kódem **404** a nesmaže daného uživatele.

**Skutečný výsledek**: API vrátí odpověď s kódem **404** a nesmaže daného uživatele.

**Stav**: Pass

## TS-018 – získání data uživatele pomocí neexistující path ale existujícího ID)

**Popis**: Ověření, že API nepošle údaje o uživateli

**Kroky testu**:

1. Odešlete **GET** request na /api/v1/students/**2485**/**path**

2. Ověřte, že se vrátí správná response

**Očekávaný výsledek**: API vrátí odpověď s kódem **404** a neukáže žádné informace.

**Skutečný výsledek**: API vrátí odpověď s kódem **404** a neukáže žádné informace.

**Stav**: Pass

## TS-019 – přidání uživatele a přepsání existujícího uživatele)

**Popis**: Ověření, že uživatele nemůžeme přepsat

**Kroky testu**:

1. Odešlete **POST**  request na /api/v1/students/**2750 (existující ID)**

**2.** **Přidejte uživatele pomocí tohoto jsonu**  
{

    "firstName": "lol",

    "lastName": "se",

    "email": "martin.konecny@seznam.cz",

    "age": 60

}  
3. Ověřte, že se stávájící uživatel nepřepíše a vrátí response **405 (Method Not Allowed)**

**Očekávaný výsledek**: Uživatel se nepřepíše a vrátí response **405 (Method Not Allowed).**

**Skutečný výsledek**: Uživatel se nepřepíše a vrátí response **405 (Method Not Allowed).**

**Stav**: Pass

## TS-020 – přidání uživatele, který má ve jméně speciální znaky)

**Popis**: Ověření, že se uživatel nepřidá, jestli v jeho jméně budou speciální znaky

**Kroky testu**:

1. Odešlete **POST**  request na /api/v1/students/

**2.** **Přidejte uživatele pomocí tohoto jsonu**  
{

    "firstName": "!!!!",

    "lastName": "sos",

    "email": "man",

    "age": 54

}  
3. Ověřte, že se uživatel nepřidá a API vrátí, že jméno nemůže obsahovat speciální znaky

**Očekávaný výsledek**: Uživatel se nepřidá a API vrátí, že jméno nemůže obsahovat speciální znaky.

**Skutečný výsledek**: Uživatel se přidá.

**Stav**: Fail

# EXEKUCE TESTŮ

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

|  |  |  |
| --- | --- | --- |
| ***Test Scenář číslo*** | ***Výsledek testu*** | ***Komentář*** |
| *TS-001* | Passed | *Data byla správně načtena z databáze* |
| *TS-002* | Fail | API vrací chybný stavový kód, očekáváno **404**, obdrženo **500 Internal Server Error** |
| *TS-003* | Passed | API vrací správný response a nenačte data uživatele |
| *TS-004* | Fail | API vrací chybný stavový kód, očekáváno **404** nebo **400**, obdrženo **500 Internal Server Error** |
| *TS-005* | Fail | API přidá uživatele i když má jen věk |
| *TS-006* | Fail | Uživatel se přidá i po chybové hlášce **500,** ale není vidět |
| *TS-007* | Fail | Uživatel se přidá i když email není platný |
| *TS-008* | Pass | Uživatel se přidá pokud má všechny údaje vyplněný |
| *TS-009* | Fail | Uživatelovo příjmení se přidá s velkýma písmenama |
| *TS-010* | Pass | API vrátí response **405** (**Method Not Allowed**) |
| *TS-011* | Fail | API vrátí response 500 po zadání mezery místo ID uživatele |
| *TS-012* | Fail | API vrátí odpověď s kódem **200** a smaže uživatele s daným ID. |
| *TS-013* | Pass | API vrátí odpověď s kódem **200** a smaže uživatele s daným ID. |
| *TS-014* | Pass | API vrátí odpověď s kódem **400** a nesmaže uživatele s daným jménem. |
| *TS-015* | Pass | API vrátí odpověď s kódem **400** a nesmaže uživatele s daným příjmením. |
| *TS-016* | Pass | API vrátí odpověď s kódem **400** a nesmaže uživatele s daným emailem. |
| *TS-017* | Pass | API vrátí odpověď s kódem **404** a nesmaže uživatele |
| *TS-018* | Pass | API vrátí odpověď s kódem **404** a neukáže žádné informace o uživateli |
| *TS-019* | Pass | Uživatel se nepřepíše |
| *TS-020* | Fail | Uživatel se přidá i přes neexistující jméno |

# BUG REPORT

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **bug** | **probability** | **severity** | **test case number** | **mitigace** |
| API vrací špatný response kód | Vysoká | Vážná | TS-002 | Oprava metody API k vrácení správného response kódu **404** a né **500** |
| API vrací špatný response kód | Střední | Vážná | TS-004 | Oprava metody API k vrácení správného response kódu **404** nebo **400** a né **500** |
| API přidá uživatele pouze s věkem | Vysoká | Kritická | TS-005 | Oprava metody pro přidání uživatele, která zamezí přidat uživatele, který má uvedený pouze věk |
| Uživatel se přidá i přesto, že se vrátí **500** | Vysoká | Kritická | TS-006 | Oprava metody pro přidání uživatele, zamezí se přidat do databáze |
| Uživatel se přidá i když email není platný | Vysoká | Vážná | TS-007 | Oprava metody pro přidání uživatele, aby vrátila response, že email není platný |
| Uživatelovo příjmení se přidá s velkýma písmenama | Vysoká | Vážná | TS-009 | Oprava metody přidání uživatelového příjmení, aby nevracela UpperCase, ale pouze UpperCase u prvního písmena |
| API vrací špatný response kód | Střední | Méně vážná | TS-011 | Oprava metody API k vrácení správného response kódu **404** nebo **400** a né **500** |
| API vrací špatný response kód a smaže uživatele | Střední | Vážná | TS-012 | Oprava metody pro mazání studenta. Aby student nebyl smazaný pokud tam není přímo jen jeho ID |
| Můžeme přidat uživatele, který má ve jméně speciální znaky | Vysoká | Kritická | TS-020 | Oprava metody pro přidání uživatele, aby se nemohl přidat uživatel co má ve jméně speciální znaky |