5/3/2023

Maroš Bednár

Slovenská technická univerzita v bratislave fakulta informatiky a informačných technológií

Endpointy knižničnej databázy

Dokumentácia zadania 5

Obsah

[Popis zadania 3](#_Toc134360838)

[Diagramy 4](#_Toc134360839)

[ER diagram 4](#_Toc134360840)

[Relačný model 4](#_Toc134360841)

[Všeobecný opis kódu 5](#_Toc134360842)

[Použité SQL dopyty 6](#_Toc134360843)

[Users 6](#_Toc134360844)

[/users/:userId 6](#_Toc134360845)

[/users 6](#_Toc134360846)

[/users/:userId 7](#_Toc134360847)

[/users/:userId 7](#_Toc134360848)

[Publications 8](#_Toc134360849)

[/publications/:publicationId 8](#_Toc134360850)

[/publications 9](#_Toc134360851)

[/publications/:publicationId 10](#_Toc134360852)

[/publications/:publicationId 10](#_Toc134360853)

[Categories 11](#_Toc134360854)

[/categories/:categoryId 11](#_Toc134360855)

[/categories 11](#_Toc134360856)

[/categories/:categoryId 12](#_Toc134360857)

[/categories/:categoryId 12](#_Toc134360858)

[Cards 13](#_Toc134360859)

[/cards/: cardId 13](#_Toc134360860)

[/ cards 13](#_Toc134360861)

[/ cards /: cardId 14](#_Toc134360862)

[/ cards /: cardId 14](#_Toc134360863)

[Instances 14](#_Toc134360864)

[/instances/: instanceId 14](#_Toc134360865)

[/ instances 15](#_Toc134360866)

[/ instances /: instanceId 15](#_Toc134360867)

[/ instances /: instanceId 16](#_Toc134360868)

[Rentals 16](#_Toc134360869)

[/rentals/: rentalId 16](#_Toc134360870)

[/ rentals 17](#_Toc134360871)

[/ rentals /: rentalId 17](#_Toc134360872)

[/ rentals /: rentalId 18](#_Toc134360873)

[Reservations 18](#_Toc134360874)

[/reservations/: reservationId 18](#_Toc134360875)

[/ reservations 19](#_Toc134360876)

[/ reservations /: reservationId 19](#_Toc134360877)

[/ reservations /: reservationId 20](#_Toc134360878)

[Authors 20](#_Toc134360879)

[/authors/: authorId 20](#_Toc134360880)

[/ authors 21](#_Toc134360881)

[/ authors /: authorId 21](#_Toc134360882)

[Zmeny v návrhu DB oproti zadaniu 4 21](#_Toc134360883)

[Záver 22](#_Toc134360884)

# Popis zadania

V rámci nového repozitára je potrebné implementovať API pre navrhnutú databázu zo zadania 4. Definovanie API sa nachádza v rámci yml súboru dostupného na https://github.com/FIIT-Databases/api-assignment. Tento súbor je možné otvoriť napr. https://stoplight.io. V rámci API nie je riešená žiadna autentifikácia požiadaviek.

V rámci jednotlivých endpointov je potrebné kontrolovať formát vstupov podľa definovania v API a v prípade zistenia nesprávneho formátu je potrebné vrátiť odpoveď s HTTP response code 400. V rámci implementácie je potrebné uskutočniť automatické migrácie tj. v prípade, že neexistuje štruktúra (relačná schéma) Vašej databázy v rámci servera, tak je vytvorená. Treba kontrolovať pri tom ako sa spúšťa Váš aplikačný server. Pri každom odovzdaní do testera sa Vám vytvára nová databáza na serveri.

Spustenie tejto migrácie je nevyhnutné pre možnosti overenia a vyhodnotenia funkčnosti Vášho riešenia. V prípade, že Vaše riešenie nedokáže uskutočniť túto začiatočnú migráciu, tak je Vaše riešenie považované za neakceptované.

Pre overenie funkčnosti Vášho implementovaného API je tiež nevyhnutné, aby Vám fungovali endpointy typu POST a GET. Pre možnosť otestovania zadania je možné použiť https://testerdbs.fiit.stuba.sk. V prípade metódy POST sa nachádza v rámci popisu aj ID záznamu. Toto ID záznamu sa v normálnych prípadoch negeneruje na strane klienta, ale na strane servera. V rámci zadania je ho potrebné generovať na strane klienta kvôli možnostiam testovania správnosti Vášho API. V praxi je ho nutné generovať na strane servera.

V rámci zadania 5 je možné počas implementácie používať aj **ORM**.

Okrem implementovanie samotného API je potrebné vyhotoviť dokumentáciu k Vášmu riešeniu, ktoré bude obsahovať:

* použité SQL dopyty pre jednotlivé endpointy s ich popisom.
* zmeny v návrhu DB oproti zadaniu 4.

Dokumentácia môže byť realizovaná ako PDF alebo markdown dokumentácia s tým, že sa bude nachádzať v AIS odovzdaní a aj v samotnom github repozitári.

# Diagramy

## ER diagram

Diagram

Description automatically generated

Obrázok 1 Entitno-relačný diagram

## Relačný model

Diagram

Description automatically generated

Obrázok 2 Relačný dátový model

# Všeobecný opis kódu

#### Nasledujúci popis platí pre všetky použité endpointy.

V zadaní bol per priehľadnosť použitý pattern **MVC**. Vstupným bodom do ku endpointu je router.js. V tomto mieste sa aplikácia rozhodne, ktorú URL má spracovať. Následne, keď zistí, akú URL dopytujeme, pošle logiku do patričného kontroléra. **Kontrolér** je miesto, kde sa spracuje konkrétna služba „**service**“ a odpoveď z nej sa v tomto kontroléri zabalí a odošle naspäť na klienta.

Konkrétna logika služby je teda riadená v súbore končiacom „service.js“. Sem sa rozbalí celý request body, overia sa dáta, ktoré prišli z klienta a ak sú validné, spracuje samotnu logiku na databáze. Na databázu sa napojí skrze súbory končiace „DAO.js“

**DAO** je skratka pre Data Access Object. V tomto mieste prebieha samotný zápis do databázy. Zápis prebieha pomocou **ORM** služby poskytovanej v balíku **Knex.js**. Zápis je väčšinou jednoduchý, pretože väčšina logiky sa vykoná už v service časti a preto sa tu uskutočňujú iba operácie ako insert, delete, get a update.

# Použité SQL dopyty

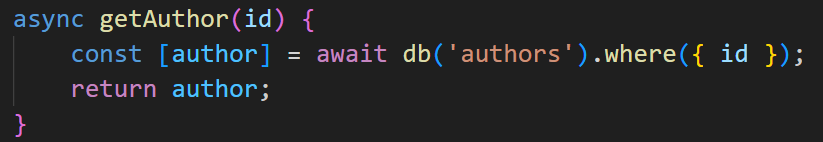
## Users

### /users/:userId

**Metóda**: GET

**Výsledok**: Vráti užívateľa.

**Popis**:



* Vráti jedného používateľa, podľa zadaného ID.

### /users

**Metóda**: POST

**Výsledok**: Vytvorí nového používateľa.

**Popis**:

A picture containing text, screenshot, font

Description automatically generated

* Vytvorí používateľa s parametrami name, surname a id a následne vráti jeho objekt.

### /users/:userId

**Metóda**: PATCH

**Výsledok**: Updatnutý používateľ.

**Popis**:

A computer code on a black background

Description automatically generated with low confidence

* Nájde používateľa podľa zadaného id a updatne mu určitú hodnotu, ktorú nastavíme v parametri data

### /users/:userId

**Metóda**: DELETE

**Výsledok**: Zmazaný používateľ aj s jeho rezerváciami a pôžičkami.

**Popis**:

A picture containing text, font, screenshot, line

Description automatically generated

* Zmaže používateľa podľa zadaného id a kaskádovite premaže aj rezervácie a pôžičky

## Publications

### /publications/:publicationId

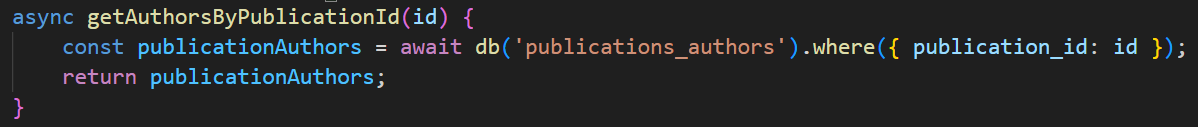
**Metóda**: GET

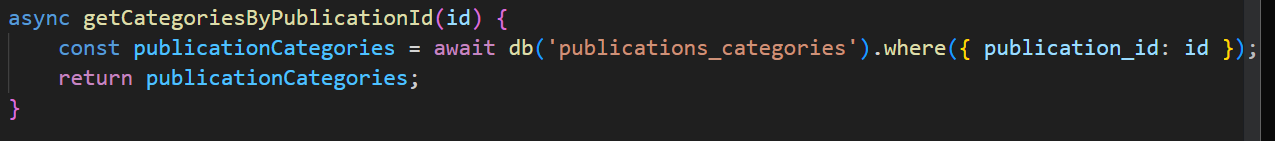
**Výsledok**: Vráti publikáciu s jej autormi a kategóriami

**Popis**:

A picture containing text, screenshot, font, line

Description automatically generated





* Vráti všetky údaje publikácie aj s jej autormi a publikáciami.

### /publications

**Metóda**: POST

**Výsledok**: Vytvorí publikáciu s jej autormi a kategóriami

**Popis**:

A screen shot of a computer code

Description automatically generated with low confidence

A picture containing text, screenshot, software, display

Description automatically generated

A picture containing text, screenshot, software, font

Description automatically generated

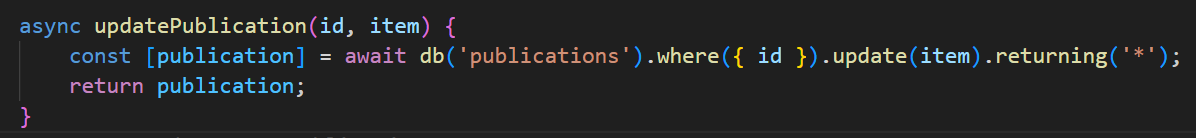
* Vytvorí publikáciu s jej parametrami id a title
* Vytvorí spojenie medzi autormi a publikáciami
* Vytvorí spojenie medzi kategóriami a publikáciami

### /publications/:publicationId

**Metóda**: PATCH

**Výsledok**: Updatne publikáciu

**Popis**:



A screen shot of a computer code

Description automatically generated with low confidence

* Updatne dáta publikácie ako sú titul, autori alebo kategórie.

### /publications/:publicationId

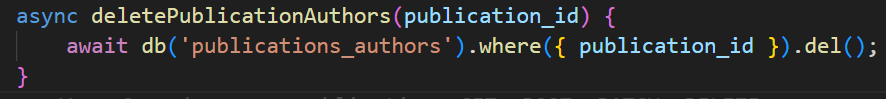
**Metóda**: DELETE

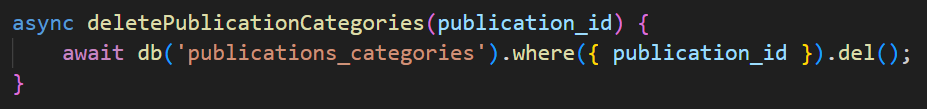
**Výsledok**: Zmaže publikáciu

**Popis**:

A picture containing text, screenshot, font, line

Description automatically generated





* Zmaže publikáciu aj so spojeniami autorov a kategórií

## Categories

### /categories/:categoryId

**Metóda**: GET

**Výsledok**: Vráti kategóriu

**Popis**:

A picture containing text, screenshot, font

Description automatically generated

* Vráti kategóriu podľa zadaného id.

### /categories

**Metóda**: POST

**Výsledok**: Vytvorí novú kategóriu

**Popis**:

A picture containing text, screenshot, font

Description automatically generated

* Vytvorí novú kategóriu podľa zadaných parametrov id a name a nakoniec vráti jej objekt.

### /categories/:categoryId

**Metóda**: PATCH

**Výsledok**: Updatne kategóriu

**Popis**:

A screen shot of a computer code

Description automatically generated with low confidence

* Updatne meno kategórie podľa parametra name a ako výsledok vráti nový objekt.

### /categories/:categoryId

**Metóda**: DELETE

**Výsledok**: Zmaže kategóriu

**Popis**:

A picture containing text, font, screenshot, line

Description automatically generated

* Zmaže kategóriu podľa zadaného id.

## Cards

### /cards/: cardId

**Metóda**: GET

**Výsledok**: Vráti kartu

**Popis**:

A picture containing text, font, screenshot, line

Description automatically generated

* Vráti kartu podľa hľadaného id.

### / cards

**Metóda**: POST

**Výsledok**: Vytvorí kartu

**Popis**:

A screen shot of a computer code

Description automatically generated with low confidence

* Vytvorí novú kartu podľa parametrov id, user\_id, magstripe a status a ako výsledok vráti vytvorený objekt

### / cards /: cardId

**Metóda**: PATCH

**Výsledok**: Updatne kartu

**Popis**: A picture containing screenshot, text, font, line

Description automatically generated

* Updatne dáta karty podľa zadanej hodnoty item a ako výsledok vráti upravený objekt.

### / cards /: cardId

**Metóda**: DELETE

**Výsledok**: Zmaže kartu

**Popis**:

A picture containing text, font, screenshot, line

Description automatically generated

* Zmaže kartu podľa daného id.

## Instances

### /instances/: instanceId

**Metóda**: GET

**Výsledok**: Vráti kópiu

**Popis**:

A picture containing text, font, screenshot, line

Description automatically generated

* Vráti kópiu knihy podľa zadaného id.

### / instances

**Metóda**: POST

**Výsledok**: Vytvorí kópiu

**Popis**:

A picture containing text, screenshot, software, font

Description automatically generated

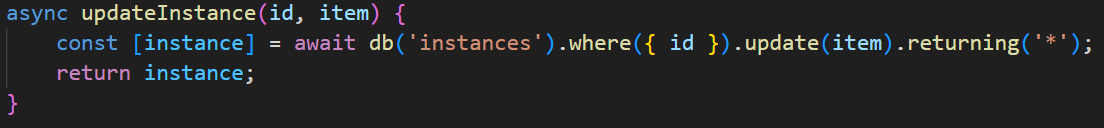
* Vytvorí nový objekt kópie podľa parametrov id, type, publisher, year, status, publication\_id.

### / instances /: instanceId

**Metóda**: PATCH

**Výsledok**: Updatne kópiu

**Popis**:



* Updatne dáta zadanej kópie podľa zadaného parametra item.

### / instances /: instanceId

**Metóda**: DELETE

**Výsledok**: Zmaže kópiu

**Popis**: A picture containing text, font, screenshot, line

Description automatically generated

* Zmaže kópiu podľa zadaného id.

## Rentals

### /rentals/: rentalId

**Metóda**: GET

**Výsledok**: Vráti pôžičku

**Popis**:

A picture containing text, font, screenshot, line

Description automatically generated

* Vráti pôžičku podľa zadaného id.

### / rentals

**Metóda**: POST

**Výsledok**: Vytvorí pôžičku

**Popis**:

A picture containing text, screenshot, software, multimedia software

Description automatically generated

* Vytvorí novú pôžičku podľa zadaných parametrov a nakoniec vráti vytvorený objekt.

### / rentals /: rentalId

**Metóda**: PATCH

**Výsledok**: Updatne pôžičku

**Popis**:

A computer code on a black background

Description automatically generated with low confidence

* Updatne pôžičku kde sa zhoduje id. Updatne hodnotu, ktorá je zadaná v parametri item. Ako výsledok vráti updatnutý objekt.

### / rentals /: rentalId

**Metóda**: DELETE

**Výsledok**: Zmaže pôžičku

**Popis**:

A picture containing text, font, screenshot, graphics

Description automatically generated

* Zmaže pôžičku podľa zadaného id.

## Reservations

### /reservations/: reservationId

**Metóda**: GET

**Výsledok**: Vráti rezerváciu

**Popis**: A picture containing text, screenshot, font, line

Description automatically generated

* Vráti objekt rezervácie podľa zadaného id.

### / reservations

**Metóda**: POST

**Výsledok**: Vytvorí rezerváciu

**Popis**:

A screen shot of a computer code

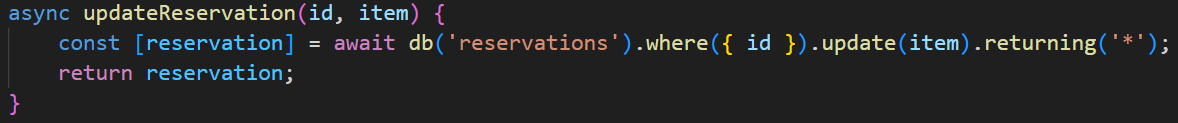
Description automatically generated with low confidence

* Vytvorí novú rezerváciu podľa zadaného id, user\_id a publication\_id. Ako výsledok vráti nový objekt.

### / reservations /: reservationId

**Metóda**: PATCH

**Výsledok**: Updatne rezerváciu

**Popis**: 

* Updatne dáta rezervácie podľa zadaného id a ako výsledok vráti upravený objekt.

### / reservations /: reservationId

**Metóda**: DELETE

**Výsledok**: Zmaže rezerváciu

**Popis**:

A picture containing text, font, screenshot, line

Description automatically generated

* Zmaže rezerváciu podľa zadaného id.

## Authors

### /authors/: authorId

**Metóda**: GET

**Výsledok**: Vráti autora

**Popis**:

A picture containing text, screenshot, font, line

Description automatically generated

* Vráti objekt autora podľa zadaného id.

### / authors

**Metóda**: POST

**Výsledok**: Vytvorí autora

**Popis**:

A picture containing text, screenshot, font

Description automatically generated

* Vytvorí nový objekt autora podľa zadaných parametrov name, surname a id. Ako výsledok vráti nový objekt.

### / authors /: authorId

**Metóda**: DELETE

**Výsledok**: Zmaže autora

**Popis**:

A picture containing text, font, screenshot, line

Description automatically generated

* Zmaže autora podľa zadaného id.

# Zmeny v návrhu DB oproti zadaniu 4

V zadaní 5 a zadaní 4 nie sú žiadne rozdiely v implementácii. Realizácia je rovnaká ako samotný návrh.

# Záver

Dokumentácia slúži ako návrh databázy, ktorú budeme v blízkej doby vytvárať. Slúži na to, aby sme si vedeli predstaviť problematiku a taktiež vedeli, čo všetko nás v ďalšom zadaní čaká. Naviac sme si precvičili tvorbu diagramov. Konkrétne Entitno-relačného diagramu, ktorý slúži na opis vzťahov medzi entitami. A relačný model, ktorý slúži na finálne vytvorenie databázy. Obsahuje presné hodnoty, ktoré budeme používať. Je rozšírený o primárne a cudzie kľúče a rozbíja vzťahy many-many na jednoduchšie.