|  |
| --- |
| Fakulta informatiky a informačných technológií |
| Databázové systémy  - Zadanie 5 |
| Slovenská technická univerzita v Bratislave |

|  |
| --- |
| Michal Kilian  7.5.2023 |

OBSAH

1. ÚVOD – ZADANIE
2. END-POINTY
   1. POST
   2. GET
   3. PATCH
   4. DELETE
3. ZMENY V NÁVRHU DATABÁZY OPROTI ZADANIU 4
4. ZÁVER
   1. ÚVOD - ZADANIE

V tomto v poradí piatom a zároveň poslednom zadaní z predmetu Databázové systémy bolo mojou úlohou vytvoriť nový aplikačný server a implementovať v ňom návrh databázy zo zadania 4, spolu s 29 zadanými end-pointami. Išlo o vytvorenie dotazov na databázu, ktoré vrátia konkrétne požadované informácie v JSON formáte. Toto zadanie som realizoval prostredníctvom Python ORM a vypracoval som ho v programovacom jazyku Python 3.10 v prostredí PyCharm 2022.2.2.

* 1. END-POINTY

Keďže zadaných end-pointov bolo 29, nebudeme sa podrobne pozerať a vysvetľovať si každý z nich, ale namiesto toho si ich rozdelíme do skupín podľa router metódy, t.j. POST, GET, PATCH, DELETE, keďže end-pointy s rovnakou metódou majú veľmi podobné riešenie.

* 1. POST

Ako príklad na metódu POST si zoberieme funkciu *create\_publication()*. Slúži na vytvorenie nového záznamu v korešpondujúcej tabuľke.

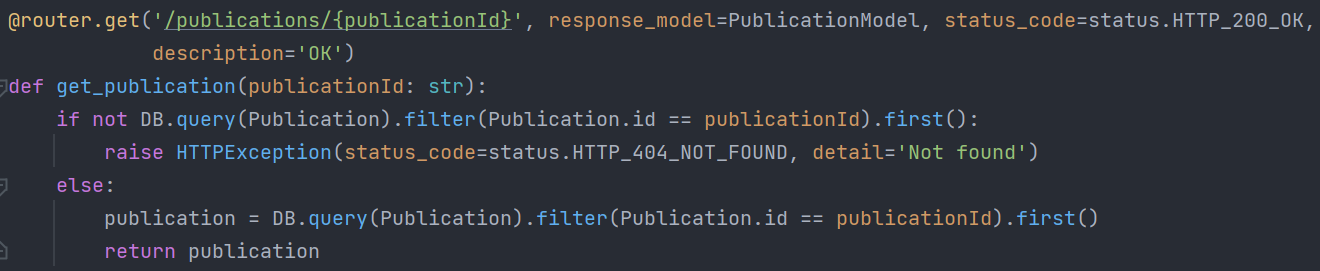
Obrázok, na ktorom je text

Automaticky generovaný popis

Táto funkcia berie *request* objekt ako argument, v ktorom sú uložené informácie o vytvorení novej publikácie. Na začiatku sa ošetria hlavné problémy, ktorými sú nedostatočné množstvo zadaných informácií, ošetrenie pri zadaní existujúceho UUID a nakoniec zistenie, či zadaní autori a kategórie existujú. Tieto *if-*statementy vyhadzujú errory v podobe žiadanej *HTTPException* podľa zadania. Následne, po prejdení všetkých *if-*ov sa dostávame k vytvoreniu novej publikácie. Zadáme jej potrebné atribúty a funkciami *DB.add()* a *DB.commit()* sa vytvorí v tabuľke *publications* nový riadok s pridanými hodnotami.

* 1. GET

Nasleduje metóda GET, pri ktorej si zoberieme príkladom funkciu *get\_publication()*. Slúži na vytiahnutie všetkých detailov o jednom zázname v databáze.



Ide len o jednoduchý dopyt na databázu, v ktorom hľadáme v tabuľke publikácií, filtrujeme podľa *publicationID* zadaným v argumente a vyberáme. Ešte predtým je samozrejme ošetrené, ak sa konkrétna publikácia v tabuľke nenachádza. Takto isto alebo aspoň veľmi podobne sú riešené všetky GET end-pointy.

* 1. PATCH

Následne sa pozrieme na metódu PATCH, ktorá slúži na upravovanie už existujúcich riadkov databázy. Príkladom nám pôjde funkcia *update\_publication()*.

Obrázok, na ktorom je text

Automaticky generovaný popis

Po ošetrení základných nezrovnalostí sa dostávame k upravovaniu konkrétneho záznamu. Najskôr prebehne dopyt na databázu, ktorý nám vyberie chcený riadok. Následne do neho vložíme hodnoty z *request* objektu a použijeme funkciu *DB.commit()*, ktorá tieto zmeny naspäť uloží.

* 1. DELETE

Nasleduje posledná metóda, v ktorej si ukážeme ako program postupuje pri vymazávaní záznamu z tabuľky. Pre vizualizáciu nám bude slúžiť funkcia *delete\_publication()*.

Obrázok, na ktorom je text

Automaticky generovaný popis

Táto metóda využíva postup veľmi podobný GET metóde, hlavne kvôli prvotnému zisťovaniu existencie konkrétneho záznamu, avšak na rozdiel od nej tu tento riadok nevraciame, pretože žiadaný výstup je v tvare *204 No Content*. Namiesto toho záznam vymažeme pomocou *DB.delete()* a *DB.commit()*. Ide o jednoduchý dopyt s filtrovaním podľa zadaného *publicationId*.

* 1. ZMENY V NÁVRHU DATABÁZY OPROTI ZADANIU 4

V tomto zadaní som musel vykonať určité zmeny v návrhu databázy zo zadania 4, pretože nereflektoval všetkými potrebnými funkcionalitami. Medzi hlavné zmeny patrí najmä pridanie spojovacích tabuliek medzi *publications* a *authors*, rovnako aj *publications* a *categories*. To bolo nutné, z dôvodu optimálnosti riešenia. Ďalej išlo samozrejme o dodanie potrebných atribútov do viacerých tabuliek, aby boli kompatibilné s end-pointami v zadaní. Ďalšími zmenami bolo premenovanie viacerých tabuliek, napríklad tabuľka *copies* sa zmenila na tabuľku *instances*, alebo tabuľka *loans* na tabuľku *rentals*. Tabuľky *wishlist, reviews, extensions* a *bad\_returns* ostali bez zmeny, hlavne kvôli tomu, že neboli využívané v žiadnom z endpointov. Celkovo si myslím, že sa môj návrh databázy zo zadania 4 v tomto zadaní až tak drasticky nezmenil, naopak len sa vylepšila jeho funkcionalita.

* 1. ZÁVER

Toto zadanie hodnotím ako mierne náročné, čo sa týka zložitosti, avšak bolo príliš časovo náročné. 61 častí testovacieho nástroja, pričom každá má svoje definované požiadavky neboli jednoduchou úlohou. Ale naopak, toto zadanie mi prišlo asi najbližšie k praxi a k databázovým systémom, s ktorými sa budeme stretávať v reálnom živote, pretože dobre reflektovalo všetky potrebné funkcionality.