Slovenská technická univerzita
Fakulta informatiky a informačných technológii
Ilkovičova 3, 812 19 Bratislava

Databázové systémy

Zadanie č. 1 – Vytvorenie aplikácie komunitného fóra na relačnou a nerelačnou databázou

Marek Kováč

Cvičiaci: Ing. Róbert Móro
Študijný odbor: Informatika
Ročník: 2. Bc
Akademický rok: 2015/2016

1. Znenie zadania

Oficiálne znenie zadania prvej iterácie

Vo vami zvolenom prostredí vytvorte databázovú aplikáciu, **ktorá komplexne rieši minimálne 6 scenárov** vo vami zvolenej doméne. Presný rozsah a konkretizáciu scenárov si dohodnete s Vašim cvičiacim na cvičení. Aplikáciu vytvoríte v dvoch iteráciach. V prvej iterácii, postavenej nad relačnou databázou, musí aplikácia realizovať tieto všeobecné scenáre:

- Vytvorenie nového záznamu,
- Aktualizácia existujúceho záznamu,
- Vymazanie záznamu,
- Zobrazenie prehľadu viacerých záznamov (spolu vybranou základnou štatistikou),
- Zobrazenie konkrétneho záznamu,
- Filtrovanie záznamov spĺňajúcich určité kritériá zadané používateľom.

Spresnenie prvej iterácie

Mojou úlohou bolo vytvoriť komunitné fórum, v ktorom budú môcť používatelia vytvárať, editovať a rušiť svoje kontá ako aj príspevky.

Oficiálne znenie zadania druhej iterácie

V druhej iterácii do aplikácie pridáte min. 1 scenár postavený na nerelačnej databáze Redis alebo Elasticsearch (dohoda s cvičiacim na inom type nerelačnej db je samozrejme možná). Konkrétny scenár si dohodnete s vašim cvičiacim v závislosti od použitej databázy a domény vašej aplikácie (napr. štatistiky o interakciách s jednotlivými záznamami aplikácie v Redise alebo vyhľadavávanie záznamov cez Elasticsearch).

Spresnenie druhej iterácie

V druhej iterácií projektu som implementoval vyhľadávanie textu v príspevkoch na fóre na základe voliteľných parametrov s použitým NoSQL databázy ElasticSearch.

2. Špecifikácia realizovaných scenárov

- ***Pridanie fóra (Scenár: Vytvorenie nového záznamu)***

Fórum ako také je delené na niekoľko hlavných podfór v kt. sa nachádzajú jednotlivé témy a v týchto témach sa dalej nachádzajú príspevku, ktoré sú usporiadané do jednotlivých skupín na zákalde toho či sa jedná o reakcie alebo originálne príspevky na kt. užívateľ reaguje.

Pridanie fóra má za úlohu pridať podfórum do celkového fóra. Takéto podfórum môže byť pridávané len administrátorom, čo je pri pridávaní samotného fóra overené aplikáciou náhľadom do databázy. Pri pridaní podfóra je nevyhnutné vložiť aspoň názov a popis fóra, pričom užívateľ je

automaticky zistení pomocou dát o prihlásení. Dátum vytvorenia fóra je automaticky generovaní vložením záznamu do databázy. Fórum bude dalej možné administrátorom zmazať, prípadne zamknúť nastavením príslsného príznaku.

- **Upravenie konta (Scenár: Aktualizácia existujúceho záznamu)**

Po prihlásení sa do používateľského účtu prostredníctvom príkazu „prihlasenie“ je môžné si pomocou príkazu „upravenie konta“ upraviť používateľské konto, a to napríklad zmeniť heslo alebo kontaktnú emailovú adresu.

- **Zmazanie fóra (Scenár: Vymazanie záznamu)**

Pomocou príkazu „zmazanie fora“ je možné fórum zmazať. Pri zmazávaní fóra je potrebné mať admin. oprávnenia inak sa procedúra zmazania skončí neúspechom.

- **Zobrazenie prehľadu kont (Scenár: Zobrazenie prehľadu viacerých záznamov (+štatistika))**

Zobrazenie prehľadu kont pomocou príkazu „zobrazenie kont“ zobrazí vš. užívateľov v databáze spoločne s ich email adresami, id a počtom príspevkov. Tiež zobrazí celkový počet registrovaných používateľov.

- **Vypísanie info o akt. prihlas používateľovi (Scenár: Zobrazenie konkrétneho záznamu)**

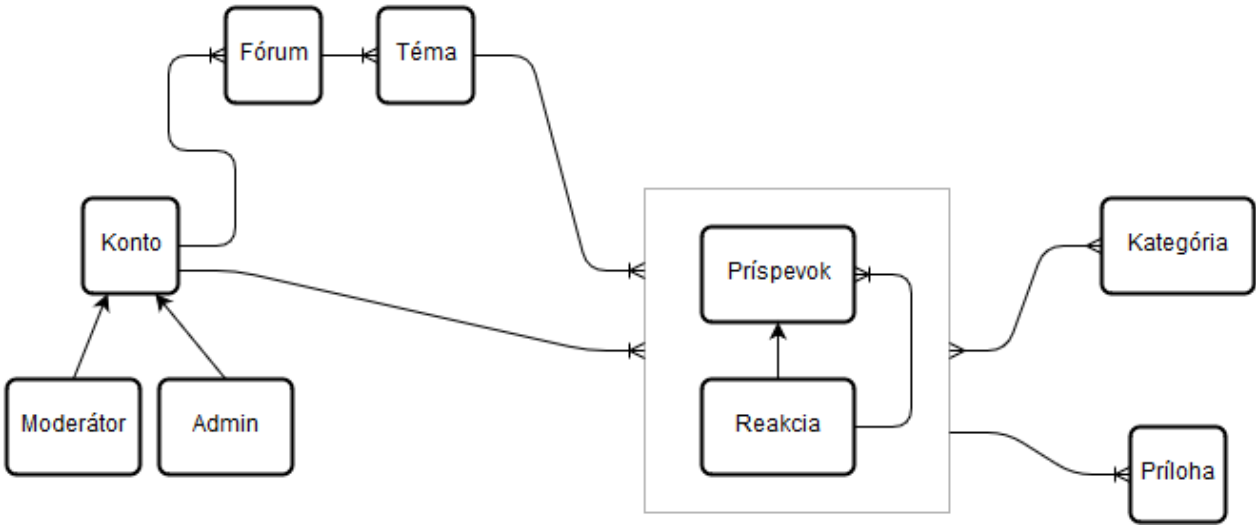
Pomocou príkazu „ktosom“ vypíšeme informácie o akt. Prihlásenom používateľovi. Aby sme sa mohli prihlásiť použije sa príkaz „prihlásenie“, kt. nás prihlási do systému.

- **Zobrazenie kont na základe prednastavených špecifikácií (Scenár: Filtrovanie záznamov spĺňajúcich určité kritériá zadané používateľom)**

Pomocou príkazu „filtrovanie kont“ môžeme filtrovať konta na základe časového rozsahu ich vytvorenia, ako aj iných parametrov ako napr. min. počtu príspevkov.

3. Diagram logického a dátového modelu

3.1 Logický model



Obr 1. Logický model

- Entita **Konto** – databáza obsahuju viacero kônt, pričom na každé jedno konto môže ale nemusí obsahovať aspoň jeden príspevok. Konto reprezentuje užívateľa, inak povedané jedná sa o užívateľský účet. Tento užívateľský účet sa dá na základe príznaku generalizovať na konto moderátora, konto administrátora alebo obyčajné používateľské konto. Tieto tri typy kont sa neskôr v aplikácií rozlišujú na základe nastevených príznakov.

- Entita **Fórum** – databáza, resp. celé fórum obsahuje niekoľko podfór (tu označené ako fórum) v kt. sa daľej nachádzajú témy a v nich jednotlivé príspevky utriedené do skupín na základe toho či sa jedná o pôvodný príspevok alebo reakciu na pôvodný príspevok. Jedným užívateľom – administrátorom môže byť vytvorených niekoľko fór, pričom jedno fórum môže obsahovať niekoľko rozličných tém. Jedna konkrétna téma môže sa môže nachádzať iba v jednom podfóre.

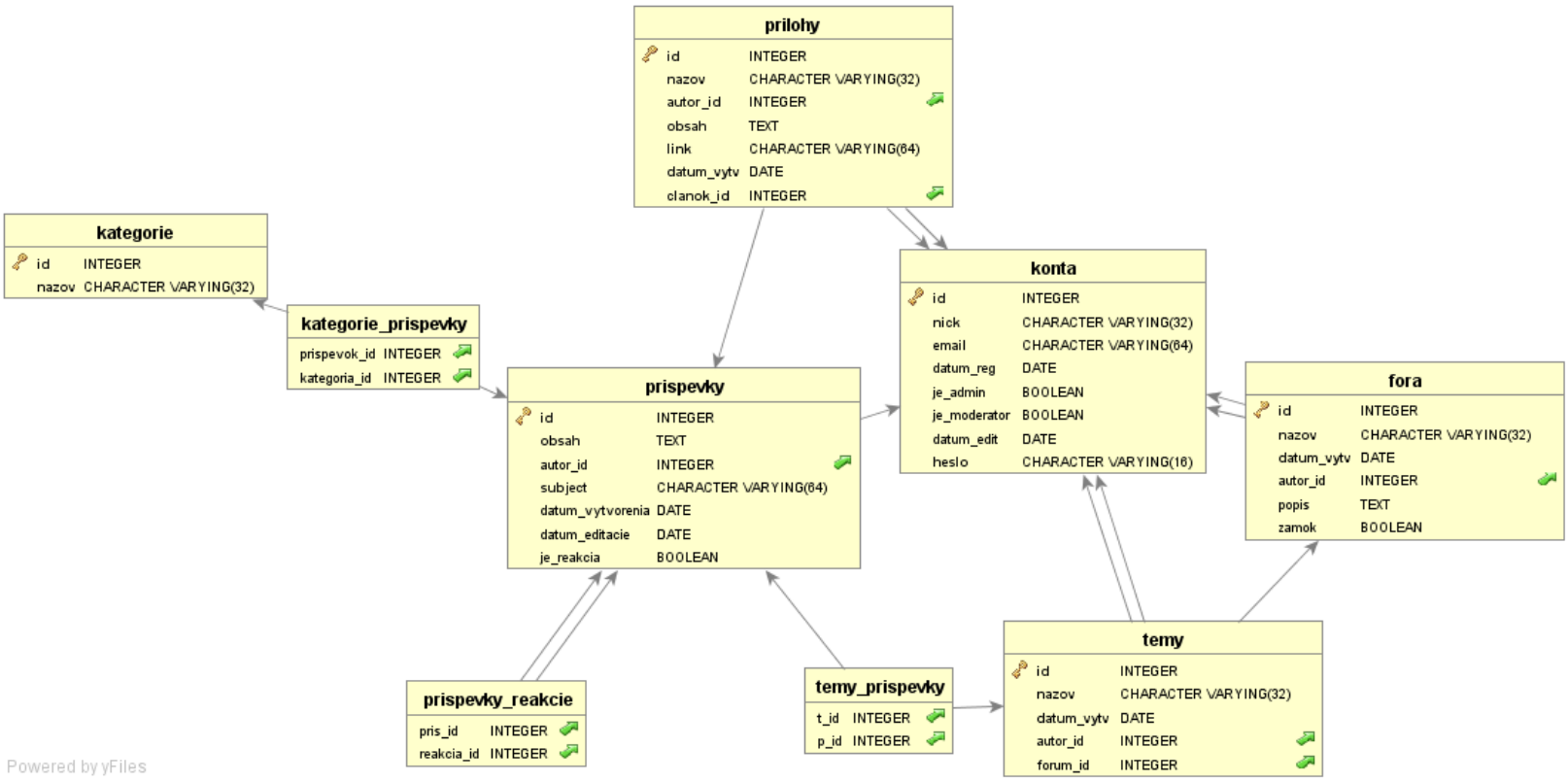
- Entita **Téma** – hierarchický prvok na nižšej úrovni, ktorý obsahuje príspevky. Jedna téma sa môže nachádzať súčasne iba v jednom konkrétnom fóre, ale samotná téma môže obsahovať niekoľko príspevkov resp. žiadny.

- Entity **Príspevok** a **Reakcia** – Samotné príspevky sú obsiahnuté v témach, pričom jeden príspevok môže byť buď originálnym príspevkom alebo reakciou na príspevok. Príspevky ako také sú logicky združené v podobe jedného hlavného príspevku a následne výlučne reakciami na tento príspevok. Jeden príspevok môže byť vytvorení a editovaní iba jedním kontom, pričom to

isté konto môže vytvoriť niekoľko príspevkov. Takisto jeden príspevok sa môže nachádzať iba v jednej téme.

- Entita **Kategória** – Príspevky je možné kategorizovať, pričom jeden príspevok môže mať niekoľko dopredu vytvorených kategórií a jedna kategória môže byť použitá u viacerých príspevkov.
- Entita **Príloha** – K jednotlivým príspevkom je možné pridávať prílohy. Na jeden príspevok je možné mať niekoľko príloh, pričom na jedna konkrétna príloha sa môže nachádzať len v jednom príspevku.

3.2 Fyzický model



Obr 2. Fyzický model

- Tabuľky **kategorie_prispevky, prispevky_reakcie** a **temy_prispevky** su väzobné tabuľky.
- Tabuľka **kategorie** – obsahuje primárný kľúč a názov kategórie, kt. je potrebný pre identifikáciu kategórie. Každá použitá kategória v príspevku by sa mala vložiť do väzobnej tabuľky kategorie_prispevky s potrebým id príspevku.
- Tabuľka **prilohy** – obsahuje PK, cudzí kľúč na ID autora, kt. je tento identifikovaný. Obsah definuje obsah prílohy pokiaľ sa jedná od dlhý text, resp. sa dá použiť link, kde sa uvedie URL zdroja. clanok_id sa odkazuje do tabuľky príspevkov. datum_vytv je generovaný automaticky pri vytvorení nového záznamu.
- Tabuľka **prispevky** – obsahuje text príspevku(obsah), predmet(subject), odkaz na autora(autor_id) a datum_vytv – generovaný automaticky pri vytvorení záznamu, datum_editacie – dátum posl. editácie jeho spravovanie má na starosti aplikácia. Príznak je_reakcia definuje či sa jedná o reakciu na už existujúci príspevok, čo sa využíva pri výpise príspevkov.
- Tabuľka **konta** – obsahuje informácie o regist. užívateľoch. Nick definuje meno autora, email je kontakt, heslo je automaticky generované pri registrácií (vytv. Nového záznamu) a je potrebné ho meniť. Datum_reg je vytvorený pri registrácií a nastavené na súčasný dátum. Príznaky je_moderator a je_admin definujú či sa jedná o moderátora a/alebo administrátora.
- Tabuľka **temy** – obsahuje informácie o témach. tj. názov, autora a dátum vytvorenia, kt. je generovaný pri vložení záznamu.
- Tabuľka **fora** – podobne ako temy. Navyše obsahuje príznak zámku, kt. je štandardne odomknutý a následne môže byť administrátorom uzamknutý.

4. Návrh a implementácia

Projekt bol vytvorení v programovacom jazyku JAVA s použitým IDE Eclipse Mars 2 na platforme Microsoft Windows 8. V projekte som na pripojenie sa k databáze využil ovládač JDBC. Pri tvorbe aplikácie som používal databázu PostgreSQL.

Implementácia jednotlivých scenárov

Všeobnecný opis

Jednotlivé scenáre sú implementované s použitým transakcií, ktoré sú nastavené pomocou príznaku `spojenie.setAutoCommit(false);`, ktorý zabezpečí zapnutie transakcií. Po každom vykonaní transakcie sa požadovaný príkaz, ktorým sa vykonal dotaz do databázy ako aj samotné spojenie ukončia.

Aplikácia obsahuje niekoľko pomocných tried v balíku (databaza) obsahujúcich potrebné statické metódy poskytujúce článkovú funkcionalitu pre jednotlivé scenáre.

Napríklad scenár „registrácia používateľa“ najprv overí či dané konto pre kt. boli vložené parametre už neexistuje a to pomocou príslušnej metódy existujeKonto v triede KontaManager, ktorá vráti objekt používateľa v prípade existencie konta, následne sa vykoná metóda vytvorKonto tej istej triedy, kt. nadviaže opätovne kontakt z databázou a vložením dopredu užívateľom zadaních údajov zaregistruje užívateľa do databázy.

Popis implementácie vybraných scenárov

- ***Pridanie fóra (Scenár: Vytvorenie nového záznamu)***

  Scenár pridania fóra sa inicializuje vložením príkazu „`pridanie fora`“ na príkazovom riadku. Tento príkaz je následne spracovaní v triede Spracovanie z balíka aplikacia. Najskôr dôjde k overeniu toho či je užívateľ, ktorý vložil tento príkaz riadne zaregistrovaný pomocou pomocného objektu Prihlásenie uchovávajúceho akt. stav o prihlasení. Pokiaľ sa meno prihláseného konta rovná reťazcovej konštante „guest“ jedná sa o neprihláseného používateľa a celý scenár končí. V opačnom prípade sa prostredníctvom volania databázy cez metódu existujeKonto zistia informácie o práve prihlásenom používateľovi ako je napr. príznak admin, kedže len adminy majú povolenie vytvárať nové fóra. Po úspešnom overení systém vyzve používateľa aby zadal informácie obsahujúce popis nového fóra, následne aplikácia pokračuje vložením týchto údajov do databázy volaním metódy vytvorFórum.

- **Upravenie konta (Scenár: Aktualizácia existujúceho záznamu)**

  Scenár upravenia konta sa začína príkazom „upravenie konta“ na príkazovom riadku. Tento príkaz je následne spracovaní v triede Spracovanie z balíka aplikacia. Následne dôjde pomocou pomocného objektu triedy Prihlasenie k overeniu či je uživateľ vôbec prihlásený a to pomocou porovnania mena akt. používateľa s retazcom „guest“. V prípade ak sa rovnajú užívateľ nie je prihlásený a teda nie je čo upravovať. V inom prípade je používateľ vyzvaný vložiť typ editácie a táto je následne vykonaná prostredníctvom metódy upravKonto, ktorá pozmení danú položku v databáze.

- **Zmazanie fóra (Scenár: Vymazanie záznamu)**

  Zmazanie fóra sa inicializuje príkazom „zmazanie fora“ na príkazovom riadku. Najskôr užívateľ zadá id fóra, ktoré chce zmazať – toto môže získať pomocou príkazu „zobrazenie fora“. Zmazanie dalej prebieha overením či je konto administrátorske s pomocou objektu Prihlasanie. Ak áno, tak sa pomocou metódy zmaž záznam fórum zmaže volaním do databázy. V príapade, že sa snažíme fórum zmazať z neadministrátorského konta, sme o tom upozornený a scenár končí.

- **Zobrazenie prehľadu kont (Scenár: Zobrazenie prehľadu viacerých záznamov (+štatistika))**

  Zobrazenie prehľadu kont sa inicializuje príkazom „zobrazenie kont“. Tento príkaz najprv volaním do databázy cez metódu zistiInfoKonta ziska celkový počet registrovaných užívateľov v databáze a následne pomocou zistiPocty získa prehľadnú tabuľku o celkovom

počte registrovaných používateľov v databáze spoločne s prehľadom ich mien, emailových kontaktov ako aj počtov príspevkov na daného používateľa. Týmito údajmi vyššie menovaná metóda naplní zoznam objektov typu UserLoaded, kt. aj vypíše.

- **Vypísanie info o akt. prihlas používateľovi (Scenár: Zobrazenie konkrétneho záznamu)**

  Príkazom „ktosom“ sú zobrazené informácie o akt. Používateľovi, tento scenár sám osobe nenadväzuje žiadne spojenie z databázou ale pomocou prihlásanie naplní objekt Prihlásenie z databázy údajmi o aktuálne prihlásenom používateľovi.

- **Zobrazenie kont na základe prednastavených špecifikácií (Scenár: Filtrovanie záznamov spĺňajúcich určité kritériá zadané používateľom)**

  Príkazom „filtrovanie kont“ sa vykonáva filtrovanie záznamov kont pomocou parametrov dátumu a min. počtu príspevkov na základe autora. Scenár je riešení dopytom do databázy, kde ako parametre sú použité informácie vložené používateľom do databázy.

5. Vyhľadávanie v príspevkoch pomocou ElasticSearch

Inštalácia

Kedže môj projekt presahuje rozsah stanovený odovzdaním v AIS na 20MB, bol som nútený vyseknúť z môjho projektu knižnice databázy ElasticSearch nevyhnutné pre správne bežanie programu.

Tieto je potrebné nakopírovať nazad do priečinka lib v projekte z podpriečinka lib hlavného priečinka elasticsearch. Jedná sa o všetky súbory.

Scenár – vyhľadávanie v texte

V projekte som implementoval jeden scenár postavený na nerelačnej (NoSQL) databáze ElasticSearch – vyhľadávanie textu na základe stanovených parametrov.

Tento scenár sa spúšta príkazom “vyhladavanie” al. skrátene “vh”.

Príkazom „synch“ je možné zosynchronizovať obsah databázy ElasticSearchu s relačnou databázou.

Priebeh scenára je nasledovný:

Používateľ si môže vybrať, či chce vyhľadávať podľa mien autorov, slov vysktujúcich sa v texte subjektu príspevku, alebo v samotnom obsahu článku. Takisto vyhľadávanie obsahuje aj dodatočné parametre ako vyhľadávanie podľa dátumu vytvorenia príspevku, podľa toho či sa jedná o reakciu na príspevok alebo pôvodný príspevok. Nakoniec si môže zvoliť obmedziť celkový počet zobrazených článkov.

- **Vyhľadávanie podľa autora** – je možné zadať nula, jedno alebo viac mien autorov, ktoré budú pri vyhľadávaní zohľadnené.
- **Vyhľadávanie podľa obsahu predmetu *(subjektu)* článku** – je možné zadať kľúčové slová, kt. sa majú vyskitovať v predmete článku.
- **Vyhľadávanie podľa obsahu v samotnom tele článku** – je možné zadať kľúčové slová, kt. sa mjú vyskitovať v tele článku a pri ich zadavaní je možné použiť regulérne výrazy ElasticSearch-u.
- **Vyhľadávanie podľa dátumu** – je možné zadať dolný aj rozsah dátumu vytvorenia článku. Používateľ nemusí zvoliť žiadny, počiatočný dátum alebo rozsah.
- **Vyhľadávanie podľa reakcií** – je možné filtrovať záznamy na základe toho, či sa jedná o reakciu na príspevok, alebo originálny príspevok. V prípade hľadania reakcii je možné zadať ID článku pre kt. hladáme reakcie.
- **Filtrovanie počtu výsledných záznamov** – je možné obmedziť počet zobrazených výsledkov na maximálnu zadanú hodnotu. Ak si používateľ nezvolí obmedzenie, je táto hodnota prednastavená.

Implementácia scenára

Scenár je implementovaný prostredníctvom niekoľkých funkcií v JAVA:

- **hladajVelasticu2(String, String, String, String, String, int, int, int)** – slúži na samotné vyhľadávanie v texte
- **pridajDoelasticu(String, String, PrispevokLoaded, int)** – pridanie článku (dokumentu) do databázy
- **zmazZelasticu(String, String, int)** – zmazanie článku (dokumentu) z databázy
- **massSync2()** - zabezpečuje synchronizáciu dokumentov medzi ElasticSearchom a relačnou databázov.

Nutné zmeny v prvej iterácií projektu:

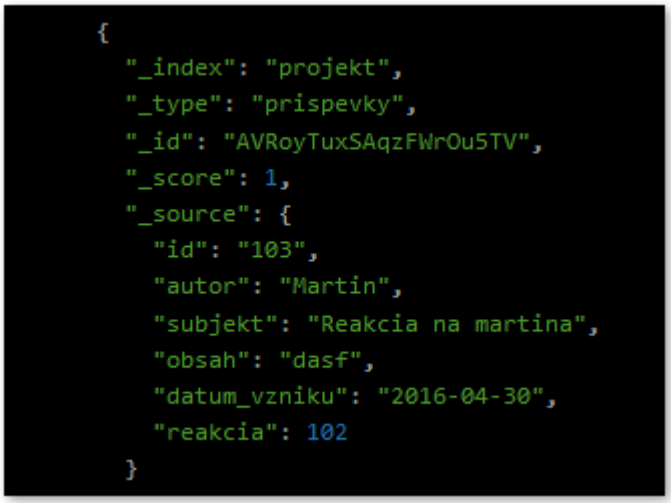
Pre správne fungovanie druhej iterácie bolo nutné upraviť funkcie:

- **Zmazania fóra** – kedže zmazaním celého podfóra zmažeme aj príspevky k nemu prislúchajúce je potrebné aby sa táto zmena prejavila aj v databáze ElasticSearch.
- **Zmazanie témy** – nápodobne zmazaním témy vieme zmazať aj všetky príspevky v téme obsiahnuté a teda je potrebné aby sa zmena prejavila aj v databáze ElasticSearch.
- **Zmazanie príspevku/reakcie** – obdobne treba zmazať aj príspevok z databázy ElasticSearch.

Schéma (Mapping) databázy

Celá schéma pozostáva z jedného indexu „projekt“ a jedného typu „prispevky“.

Formát dokumentu v ElasticSearch:



Obr. Schéma dokumentu

- **id** – id článku korešpondujúce s PK v nerelačnej databáze
  - *typ:* string, not_analyzed
- **autor** – autor článku
  - *typ:* string, analyzed
- **subjekt** – predmet článku
  - *typ:* string, analyzed
- **obsah** – obsah článku
  - *typ:* string, analyzed
- **datum_vzniku** – dátum vytvorenia článku
  - *typ:* date, not analyzed
- **reakcia** – jedná sa o reakciu, ak nie = 0, inak číslo článku reakcie
  - *typ:* integer, not_analyzed

Typy označené ako analyzed sú ďalej analyzátormi ElasticSearchu rozoberané na tokeny, pomocou kt. prebieha full-textové vyhľadávanie v daných poliach. Napr. textový reťazec „Lorem ipsum dolor simet“ je rozobratý na reťazce štandardne na základe bieleho miesta a to: lorem, ipsum, dolor, simet a následne nad týmito reťazcami prebieha vyhľadávanie. Čím viac zhôd nastane tým lepšie – príspevok dostane vyššie ohodnotenie.