Sovelluksen Ohtu-lukuvinkit arkkitehtuuri

Team Team
https://github.com/luupanu/ohtu-lukuvinkit

Sisällysluettelo

Yleiskuvaus arkkitehtuurista

Sovelluksemme *ohtu-lukuvinkit* tai *Reading tips* on lukuvinkkien organisoitiin tarkoitettu ohjelma. Ohjelma on toteutettu web-sovelluksena Javalla käyttäen Spring-frameworkia sekä Thymeleafia näkymien luomisen apuna. Valitsimme nämä teknologiat kompromissina, sillä Java oli käytännössä ainoa kieli jota kaikki tiimimme jäsenet osasivat.

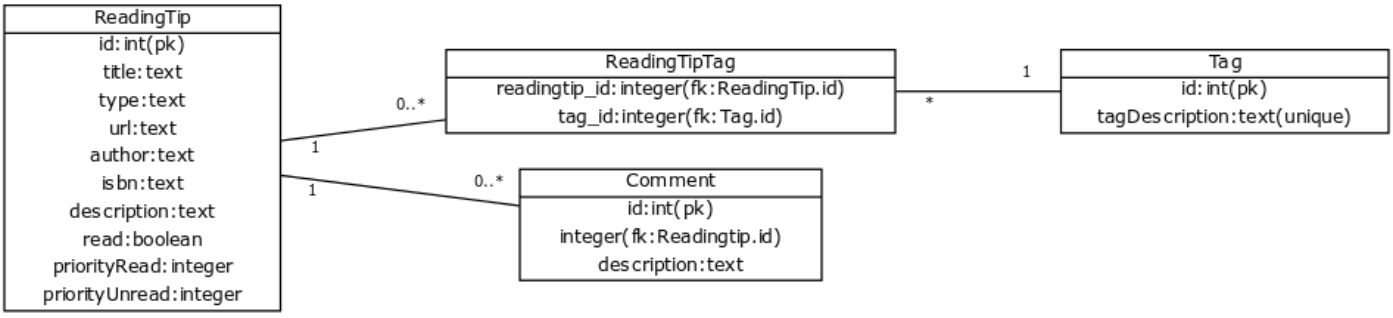
Projektimme on toteutettu niin kutsuttuna *single-page applicationina*, eli siis käyttäjä sovelluksen avattuaan löytää kaiken toiminnallisuuden heti ensimmäiseltä sivulta. Sovelluksemme ei kuitenkaan aivan täysin vältä sivun uudelleenlatauksia, toisin kuin moderneissa javascript-pohjaisissa ohjelmissa. Sovellus käyttää hyödykseen Javascriptiä, ja käyttäjän selaimen tuleekin tukea tätä.

Lukuvinkit tallentuvat paikalliseen relaatiotietokantaan. Valitsimme tietokantajärjestelmäksemme kevyen sqlite3:n.

Projektimme pyrkii mahdollisimman hyvin noudattamaan MVC-mallia. Javan perinteiden mukaan käytämme tietokannan abstrahoimiseen DAO-mallia.

Tietokanta

Tietokantakaavio



Tietokannassa tällä hetkellä ReadingTip kuvaa lukuvinkkiä. Lukuvinkkejä on kolmea eri tyyppiä (Article, Book, Link), jota kuvastaa attribuutti *type*. Artikkeleilla on attribuutti *author*, kirjoilla *author* ja *isbn* sekä linkeillä *url*. Kaikilla kolmella on myös attribuutit *title* ja *description*. Pidämme tällä hetkellä näitä kaikkia kolmea samassa tietokantataulussa, mikä rikkoo periaatetta single responsibility principle. Teimme tämän tietoisena päätöksenä siksi, että tällä hetkellä tietokannan muuttaminen olisi liian kallis operaatio toteutettavaksi. Attribuutti *read* kuvastaa, onko lukuvinkki jo luettu, jolloin se näkyy sovelluksessa erilaisena. Attribuutit *priority_read* ja *priority_unread* kuvastavat lukuvinkkien prioriteettia. Koska luetut vinkit halutaan erottaa lukemattomista, päädyimme toteuttamaan priorisoinnin jakamalla sen kahteen eri attribuuttiin: lukuvinkeille, joita ei ole luettu ja lukuvinkeille, jotka on merkattu luetuiksi.

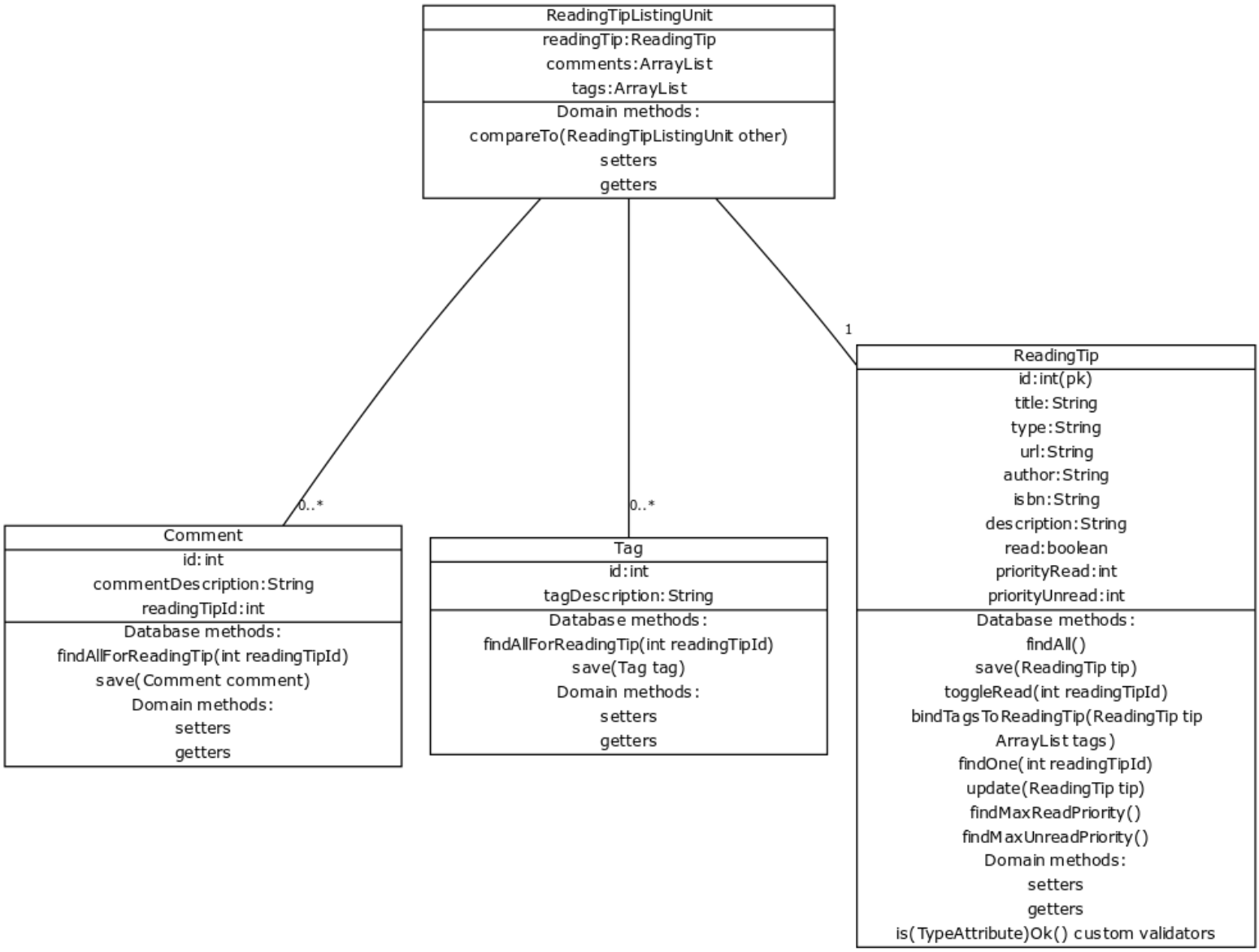
Yhdelle lukuvinkille voi antaa monta kommenttia. Kommentin attribuutista *description* löytyy itse kommentin sisältö.

Lukuvinkeille voi antaa tageja ja vinkkejä voi etsiä tagien perusteella. Lukuvinkeillä ja tageilla on näin ollen monesta-moneen suhde, jonka takia välissä on liitostaulu ReadingTipTag. Taulun Tag attribuutti *tagDescription* kuvaa tagin nimeä.

Tällä hetkellä erillistä testausympäristöä ei ole, joten testien ajaminen luo uusia olioita kehitysympäristön tietokantaan. Database-luokallamme on kuitenkin metodi helpottamaan tietokannan siivoamista testien ajamista varten.

Luokat

Luokkakaavio



Luokkien metodit ovat settereita, gettereita ja ReadingTipin tyyppeihin kuuluvien attribuuttien validaattoreita lukuun ottamatta kaikki metodeja, jotka tekevät kutsuja tietokantaan DAO-luokkien avulla. ReadingTipListingUnit sitoo ReadingTip-oliot ja siihen kuuluvat kommentit sekä tagit yhteen. Se huolehtii controllerissa ja viewissa siitä, että ReadingTipin yhteydessä voidaan näyttää myös siihen liittyvät Tag- ja Comment-oliot.

Comment-oliot ovat suoraan ReadingTip-olioihin sidottuja tietokantakaavion mukaan, kun taas Tag-oliot etsivät siihen kuuluvat ReadingTip-oliot tietokantataulun ReadingTipTags mukaan.

Luokkien attribuutit ovat suoraan verrannollisia tietokantataulun attribuutteihin. Tietokannan tyyppi text on ilmaistu koodissa tyyppinä String.

Tarkemmat metodikuvaukset löytyvät projektimme JavaDocsista. Kaaviossa ei ilmene pakkauksen service metodeja, jotka toimivat tietokannan ja domainin siltana - niistä on kuitenkin JavaDocit. Myöskään luokkien apumetodeja, joiden näkyvyys on private, ei ole merkitty kaavioon.

Bugit ja kehitysehdotukset

ISBN tarkistus

ISBN:stä voisi tarkistaa, että se on oikeassa muodossa.

Testidatan alustaminen

Tällä hetkellä testidata luodaan aina uudestaan tietokantaan testejä ajettaessa. Uusia testitippejä luodaan featureissa jotka eivät testaa juuri tippien luomista. Olisi siis joidenkin featurejen osalta järkevämpää kovakoodata erilaista dataa testitietokantaan ja käyttää tätä dataa testejä ajettaessa. Lisäksi jotkut featureista luottavat siihen, että muut featuret ajetaan ennen niitä. Tätä voisi kehittää järkevämmäksi.