# Tietokantojen perusteet, viikko 4

Neljännen viikon harjoitukset koostuvat kahdesta osasta. Ensin tehdään pohdintatehtävä, jonka jälkeen harjoitellaan ohjelmointia.

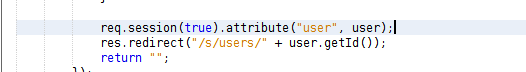
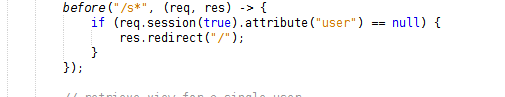
### Osa 1:

1. Oletetaan, että taulussa Varaus on 3000000 riviä, ja taulussa Asiakas on 600 riviä. Tehdään kysely   
     
    SELECT \* FROM Varaus, Asiakas   
    WHERE Varaus.varaaja = Asiakas.id AND   
    Varaus.kohde = 14 AND   
    Asiakas.nimi = ‘Dante’  
     
   Jokainen WHERE-avainsanan jälkeinen ehto tulee suorittaa, mutta ehtojen suoritusjärjestystä ei ole määritelty. Mitä tarkoittaa, että yksittäinen ehto suoritetaan? Missä järjestyksessä ehdot kannattaisi suorittaa? Miksi?

### Osa 2:

Harjoitellaan tällä viikolla läsnä tehtävissä harjoituksissa web-sovellusten käyttöä ja toteuttamista. Jatkokehitetään osoitteessa <https://github.com/tietokantojen-perusteet/tikape-web-snaptsat> olevaa viestipalvelua, jonka tarkoitus on tarjota mullistavaa pikaviestintätoiminnallisuutta sen käyttäjille.

**Huom! Käyttäkää NetBeans with TMC -versiota, jotta käytössä olisi Java 8.**

1. Kloonatkaa osoitteessa <https://github.com/tietokantojen-perusteet/tikape-web-snaptsat> oleva projekti paikalliselle koneelle. Sovelluksen etusivulla on kirjautumislomake, jonka avulla käyttäjä lähettää käyttäjätunnuksensa ja salasanansa palvelimelle. Näiden tarkastamiseen on tarkoitus käyttää UserDao-palveluun toteuttamatta jäänyttä metodia  
     
   *User findByUsernameAndPassword(String username, String password)*Toteuttakaa yllä mainittuun metodiin toiminnallisuus, joka hakee käyttäjää tietokannasta annetulla käyttäjätunnuksella ja salasanalla. Jos käyttäjää ei löydy, metodin tulee palauttaa *null*, muuten palautetaan löydetty käyttäjä.   
     
   Kun testaatte sovellusta tämän jälkeen, huomaatte ettei se toimi. Toteuttakaa tarvittava tietokantataulu sqlite3:n kautta, ja lisätkää tietokantaan ainakin yksi käyttäjä. **Huom. kannattaa katsoa luokasta User mitä sarakkeita taululle tulee määritellä.**  
     
   ---  
     
   Koodissa oleva rivi *req.session(true).attribute(“user”, user)* lisää käyttäjän tiedot palvelimen muistiin, josta ne haetaan aina pyynnön yhteydessä.  
     
     
     
   Jos käyttäjä ei ole kirjautunut, ohjaa sanoilla *before* alkava koodi käyttäjän takaisin etusivulle.  
     
   
2. Osoitetta /s/users/:id kuuntelevassa metodikutsussa halutaan käyttäjän tietojen lisäksi listata myös tietokannassa olevat viestit. Toteuttakaa tätä varten sopiva tietokantataulu -- **luokka Tsat voi toimia pohjana tietokantataulun toteutuksessa.**
3. Toteuttakaa edellä tehtyä tietokantataulua varten luokka TsatDao, joka tarjoaa tietokantatoiminnallisuutta sovellukselle. Ottakaa toteutuksessa mallia UserDao-luokasta. Tehkää TsatDao-luokalle lisäksi metodi *findLastTenMessages()*, joka palauttaa 10 viimeisintä viestiä -- viestien järjestyksen voi todennäköisesti päätellä tietokantataulussa olevasta id-kentästä.
4. Tarkistakaa manuaalisesti, että viestit näkyvät sivulla jos viestejä lisätään manuaalisesti (sqliten avulla) tietokantaan.
5. Käyttäjän sivulla näkyy lomake viestin lisäämistä varten. Toteuttakaa viestin lisäämistoiminnallisuus -- hahmotelmaa löytyy jo metodista, joka alkaa sanoilla *post("/s/tsats" …*
6. Hahmotelkaa sovellukseen lisättävää *tykkäystoiminnallisuutta*, missä viestistä voi tykätä. Nappi eräälle hahmotelmalle löytyy jo viestien listauksesta osana tsats.html-sivua. Tykkäysten lukumäärän tulee myös näkyä jollain tavalla sivulla.
7. Jatkokehittäkää tykkäystoiminnallisuutta siten, että käyttäjä linkittyy jokaiseen tykkäämäänsä viestiin. Tähän tarvitsee todennäköisesti jonkinlaisen liitostaulun, sekä mahdollisesti myös erillisen Dao-luokan.
8. Lisätkää käyttäjän sivulle viestilistauksen lisäksi erillinen listaus, missä näkyy vain tykätyt viestit.