

Joel Helkala

Synt: 19.6.1995

joel.helkala@gmail.com

Erikoistyöraportti

TIES504 Tietotekniikan erikoistyö, 20.1.2022

1. Raportti

Tämä raportti on toimii dokumentaationa TIES504 -kurssin suoritukseen. Raportissa käydään läpi erikoistyöohjelman perustelut, tekniset tiedot, tietorakenteet ja tekijän oma analyysi suoriutumisesta. Laitan tämän raportin yhteydessä linkin Gitiin, jossa minun projektini on sekä linkin Youtubeen, jossa on esittelyvideo minun projektista.

2. Perustelut projektille

Suoritan Jyväskylän yliopistolla tietotekniikan maisterin tutkintoa ja syventäväksi moduuliksi olen valinnut sensoriverkot. Halusin erikoistyöni liittyvän sensoriverkkoihin jotenkin, joten päädyin IoT-monitorointi työpöytäsovellukseen. Toteuttamallani sovelluksella voin monitoroida mökkini lämpötilaa ja kosteutta kotoa käsin. Sovelluksen nimesin Watcheriksi ja sillä pystyy monitoroimaan sensorien lähettämää informaatiota ympäristöstä mm. lämpötilaa, kosteutta ja valoisuutta.

3. Sovelluksen tekniset tiedot

Sovelluksen palvelin ja käyttöliittymä on suoritettu Java ohjelmointikielellä. Aluksi oli tarkoitus tehdä tämä C#:lla, mutta minun Windows tietokoneeni hajosi ennen työn aloittamista. Palvelin on toteutettu Java Spring Boot -kehyksellä ja käyttöliittymä on toteutettu Java Swingillä. Palvelin on toteutettu

REST-arkkitehtuurityylillä. Käyttöliittymä lähettää HTTP-protokollan yli pyyntöjä palvelimelle, joihin palvelin vastaa JSON-muodossa. Tietokantaan käytin PostgreSQLää. Ohjelmassani on myös sähköpostin lähetys, johon käytin kehittäjien toteutusvaiheen sähköposti testaukseen luotua MailDevä. Kehitykseen käytin käyttöliittymän puolella Eclipse IDE:tä ja palvelimella IntelliJä.

Ohjelmaa suunnitellessani tiesin, että haluan käyttäjäkirjautumisen sovellukseen. Kesätöistäni tiesin, että tähän käytetään sessio-tokeneja, mutta en tiennyt miten. Mitä enemmän tätä asiaa mietin, sitä monimutkaisemmalla se tuntui. Alkaessani ohjelmoimaan käyttöliittymää ja palvelinta, ei minulla ollut tietoa siitä, miten ne kannattaisi toteuttaa. En ollut koskaan aikaisemmin tehnyt REST-palvelinta Javalla tai käyttöliittymää Swingillä. Tämän takia aluksi meni muutama päivä, kunnes löysin hyvät tavat toteuttaa ohjelma.

Käyttöliittymälle ja palvelimelle en kerennyt luomaan Unit-testejä, mutta palvelimelle loin testejä Insomnialla. Insomnialla voidaan lähettää HTTP-pyyntöjä, jolla pystyin testaamaan palvelimen toimintaa ilman käyttöliittymää. Tiedän testien olevan todella tärkeä osa tietojärjestelmää, mutta perustelin tämän sillä, että halusin saada toimivan tietojärjestelmän viiteen opintopisteeseen ja testit eivät tätä edesauttaisi.

Java projektit toteutin Maven-kehyksellä, jolla pystyi helposti lisäämään ulkoisia kirjastoja, eikä tarvinnut ladata erikseen näiden *.jar* tiedostoja.

4. Vaikeudet projektissa

Vaikeaa projektissa oli rekisteröitymisen luominen, kirjautuminen ja authorisaatio palvelimen päätepisteille hyödyntäen Spring Boot Securityä. Oletinkin näiden asioiden olevan projektissa vaikeita ja siksi olin motivoitunut keskittymään niiden tekemiseen huolella. Projektissa melkein kaikki oli jotain mitä en ollut tehnyt aiemmin, mutta muut osa-alueet olivat omasta mielestä kuitenkin erittäin suoraviivaisia.

Rekisteröitymisessä luodaan käyttäjä jolle asetetaan *boolean* arvo siitä, onko käyttäjä varmistettu. Käyttäjälle lähetetään annettuun sähköpostiin linkki, jolla käyttäjä voidaan varmistaa. Ilman tätä käyttäjä ei voi kirjautua sisään. Käyttäjälle lähetetty linkki on vain pääte piste, jossa on asetettu tokeni URI:iin. Käyttäjä painaa linkistä joka luo GET-pyynnön pääte pisteellä ja tämä aiheuttaa palvelimen asettamaan käyttäjän arvoksi *True*.

Kirjautuessa oikeilla tunnuksilla käyttöliittymälle palautetaan käyttäjän tiedoista luotu sessio tokeni, joka lähetetään palvelimelle jokaisen authorisaatiota vaativan HTTP-pyynnön mukana. Täten käyttöliittymän ei tarvitse lähettää käyttäjätietoja jokaisen pyynnön yhteydessä. Sessio tokenissa on myös viive, jolloin se menee pois käytöstä, jos sitä ei käytetä. Käyttäjän tehdessä HTTP-pyyntö vanhentuneella tokenilla, palvelin palauttaa tästä tiedon ja käyttäjä pakotetaan kirjautumaan ulos.

5. Työn onnistuminen

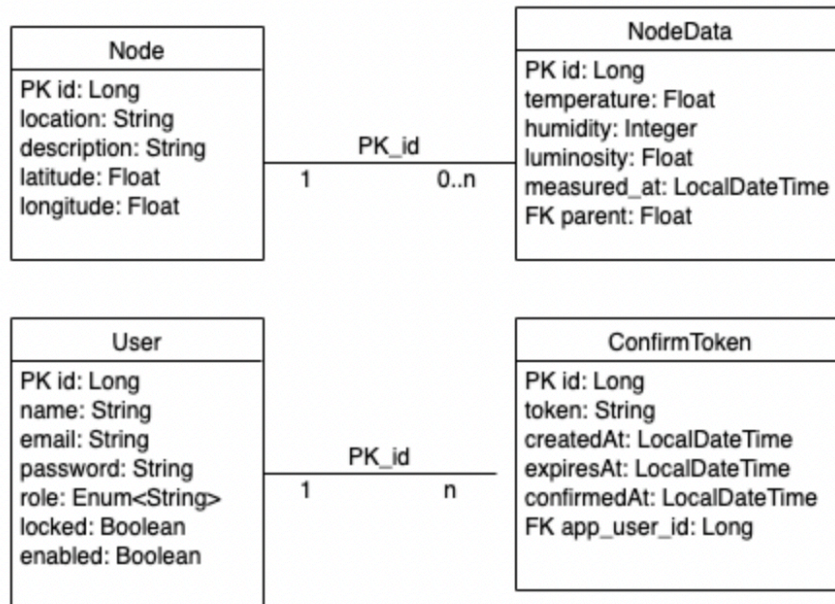
Omasta mielestä erikoistyö onnistui erittäin hyvin. Olen erittäin tyytyväinen lopputulokseen vaikka siinä onkin paljon asioita mitä haluaisin vielä parantaa. Käyttäjien autentikointi ja pääte pisteiden authorisaatio onnistui minusta erittäin hyvin ja tämä oli yksi pääasioista, joita halusin tämän työn ohella oppia. Toivon kuitenkin, että olisin käyttöliittymä kehitykseen valinnut JavaFX -kehiksen, sillä tämä on ilmeisesti hieman yleisemmässä käytössä ja monipuolisempi Swingiin verrattuna.

Spring Boot oli minusta erittäin hyvä valinta REST-palvelimen toteuttamiseen. Toivon, että olisin keskittynyt alussa hieman enemmän projektin rakenteen suunnitteluun. Kunnan suunnitelman avulla olisin välttänyt monia hieman aikaa vieviä tilanteita ja toteuttaminen olisi ollut vielä suoraviivaisempaa.

Yleisesti ottaen projektin toteutus oli vaivatonta, kun oli käyttöliittymän ja palvelimen toteutustavat saanut päätettyä. Koodin rakenne on omasta mielestä

selkeää. Käyttöliittymän paneelien toteutukset voivat olla ehkä hieman sekavia ja näitä olisi voinut jakaa pienempiin osiin.

6. Tietorakenne



PK tarkoittaa tauluissa Primary keytä ja FK Foreign keytä. Tietokantoihin ei tullut paljoakaan tauluja. Tarkoituksena oli aluksi luoda käyttäjäkohtaisia noodeja, jotta käyttäjiä voi olla useita ja kaikki eivät näkisi samoja noodeja. Tämä kuitenkin jäi kiireiden alta pois. Tämä olisi mahdollista toteuttaa siten, että käyttäjä voi pyytää oikeuksia nähdä jonkin noodin tietoja ja käyttäjä jolla on Admin -oikeudet voisi tämän hyväksyä. Tästä sopimuksesta tehtäisiin oma taulu, jossa noodin *tilaajia* voisi pitää.

Tarkoituksena oli myös luoda käyttäjille mahdollisuus lähettää kaveripyyntöjä keskenään ja tämän avulla jakaa noodien dataa toisilleen. Tämä toimisi samalla tavalla kuten edellinen, ja ystävyyksistä tehtäisiin omat taulut.

Käyttäjän ja varmistustokenin suhde on yhdestä n:ään, koska varmistustokenissa on viive, jolloin se vanhenee. Käyttäjän yrittäessä kirjautua ja

se ei ole varmistettu (*enabled*), käyttäjälle generoidaan uusi varmistustokeni, joka lähetetään sähköpostiin. Noodeilla taas voi olla 0 tai enemmän historia dataa.

7. Itsearvio työn onnistumisesta

Mielestäni työ onnistui todella hyvin, kun ottaa huomioon, että en ole aikaisemmin tehnyt mitään tämänlaista. Halusin asettaa itselleni haasteen ja toteuttaa jotain uutta, jotta saisin tämän kurssin työstä mahdollisimman paljon irti. Tavoitteenani oli suorittaa kurssi viidellä opintopisteellä ja siihen pääsinkin. Työtuntitaulukkoni mukaan tein 127 tuntia töitä ja nämä ovat pelkkää ohjelmointia ja tiedon etsimistä. Tähän voin hyvällä omantunnolla lisätä 10 tuntia suunnittelemista ja pohtimista projektille, jolloin tuntimääräksi tulee 137 tuntia.

Sain toteutettua selkeän ja toimivan ohjelman, josta on myös hyötyä. Ohjelmasta varmasti löytyy bugeja, koska en ole kerennyt luomaan testejä ohjelmalle. En manuaalisella testauksella oli kuitenkaan nyt saanut virheitä aikaiseksi. Tietojärjestelmästä puuttuu vielä toimintoja joita haluaisin lisätä, mutta aika ei vain riittänyt näiden toteuttamiseksi. Koodin luominen oli suoraviivaista ja jos tuli tilanne, jossa en tiennyt mitä tai miten tehdä jokin, osasin helposti lähteä etsimään verkosta ratkaisua. Spring Boot Security:n lisäämiseen jouduin turvautumaan tutoriaali videoihin.

Näiden mainitsemieni asioiden pohjalta antaisin itselleni kurssista viidellä opintopisteellä arvosanan 5. Jos Spring Boot Securityyn liittyvät tutoriaali videot ovat haitaksi, ymmärtäisin tiputtamisen 4 arvosanaan. Oma ymmärrykseni on kuitenkin, että videoiden katsominen asioiden toteuttamisesta on hyväksyttävää ja suotavaa.