JavaScript Työ

Tommi Jokinen

Miikael Lehtimäki

Antti Leppänen

Syyskuu 2016

Ohjelmistotekniikka

SISÄLLYS

[1 OHJELMAN VAATIMUKSET 3](#_Toc463425947)

[2 OHJELMAN ARKKITEHTUURIRATKAISU 4](#_Toc463425948)

[3 OHJELMAN SUUNNITTELURATKAISU 5](#_Toc463425949)

[4 HAASTEET 6](#_Toc463425950)

[5 TOIMIVUUS 7](#_Toc463425951)

[6 OHJELMAN KÄYTTÖÖNOTTO 8](#_Toc463425952)

[7 OHJELMAN TESTAUS 9](#_Toc463425953)

[8 TYÖAIKA 10](#_Toc463425954)

[9 YHTEENVETO 11](#_Toc463425955)

# OHJELMAN VAATIMUKSET

Suora lainaus tehtävänannosta:

*Tehtävänäsi on tehdä yksinkertainen navigaattoriohjelma, joka laskee pienimmän etäisyyden kahden suomalaisen kaupungin välillä maanteitse. Vaatimuksia ohjelmalle:*

* *oltava selainpohjainen*
* *hyödynnettävä JavaScript:iä ja sen päälle rakennettuja kirjastoja*
* *suositellut kirjastot: MEAN-pino:*
  + *MongoDB-tietokanta tietovarastona*
  + *Express.js-kirjasto kutsurajapintana Node.js -palvelimeen päin*
  + *AngularJS-kirjasto apuna selainpään toimintojen tekemisessä ja MVC-mallin hyödyntämisessä*
  + *Node.js-kirjasto palvelinpään toimintojen alustana*
* *Muita tutustuttavia teknologioita (ks. alla oleva kuva): REST-rajapinnat*
* *Muitakin kirjastoja voi käyttää mutta olisi suositeltavaa, että ne olisivat valtavirtateknologioita. Perustele selostuksessasi tällaiset valintasi. Käyttöliittymän pitää toimia selaimessa, palvelinpäässä on oltava sovelluksen älykkyys ja käytetyt pohjatiedot on oltava tietokannassa. Lisäksi on sovellettava JavaScript:iä mahdollisimman paljon.*
* *Tämä työ on etupäässä tutustumista uusiin teknologioihin; ei niinkään työ valmiin tuotteen tekemiseksi. Työn eteneminen, aikaansaaminen ja ongelmien ratkaiseminen ovat tärkeämpiä tavoitteita kuin täysin oikein toimiva stilisoitu loppukäyttäjän sovellus. Ja etenemisen yhteydessä tärkeintä on siinä saatava oppiminen. Muistetaan myös ryhmätyön voima eli kaveria autetaan ongelmatilanteissa puolin ja toisin sekä työnjakoa harrastetaan. Etenemisen vaiheet ja lopputuotos kuvataan työstä tehtävässä raportissa.*

# OHJELMAN ARKKITEHTUURIRATKAISU

Käytettiin Meteor.js kirjastoa ja sen mukaista arkkitehtuuria. Valittiin Meteor.js koska se on nopea oppia ja yhteydet clientin, serverin ja databasen väliin tulevat valmiina, samoin kuin testi ympäristö (paikallinen palvelin) ja database.

Ohjelman eri tasot, ja niiden toteutuskielet:

UI: HTML

Client logic: JS

Server logic: JS

Database: MongoDB

Tasojen välinen kommunikointi:

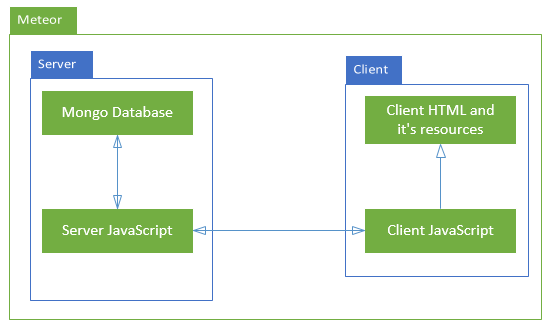
UI – Client logic: Spacebars (Meteor.js)

Client logic – Server logic: Meteor.js (Node.js)

Server logic – Database: Meteor.js

# OHJELMAN SUUNNITTELURATKAISU

Projektin työstäminen Meteorin avulla jaotteli työskentelevät palaset yksinkertaisesti. Meteorin palvelinpuolella on MongoDB, jossa projektin tietokantaa säilytettiin sekä palvelinpuolen JavaScript koodi, jossa tietokantaa käsiteltiin ja luotiin metodit käyttäjäpuolen keskustelua varten. Käyttäjäpuolella on html -pohjat, css -tyylitiedostot, käyttäjänäkymiin tarvittavat resurssit sekä JavaScript koodi, jota kyseisillä sivuilla ajetaan ja käytetään keskustelemaan palvelinpuolella tehtyjen metodien kanssa.



KUVIO 1. Miten Meteor keskustelee eri komponenttien kanssa keskenään.

Kun käyttäjä avaa sivuston, kutsuu käyttäjäpuolen javascript koodi automaattisesti palvelinpuolen metodia, joka palauttaa listan kaupungeista käyttäjäpuolelle, joka taas asettaa sen esille nettisivulle käyttäjälle valittavaksi. Saman tyyppinen toiminta suoritetaan, kun käyttäjä kysyy kahden valitun kaupungin etäisyyttä.

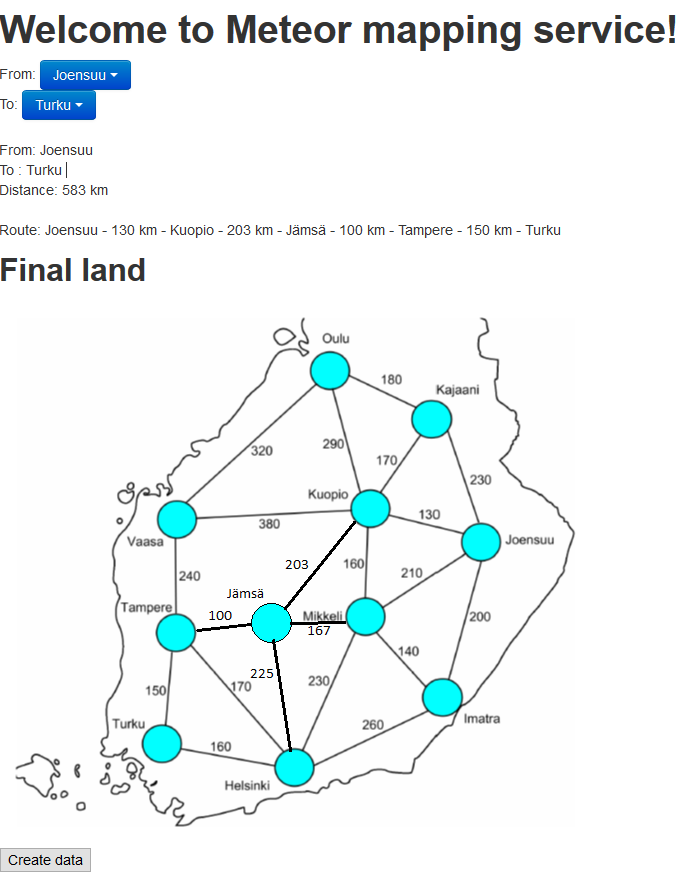
# HAASTEET

Työ oli hyvin haasteellinen malli esimerkkien puutteiden vuoksi. Ohjelman hajautettu kehittäminen toimi siihen asti kunnes useampi henkilö työskenteli saman tiedoston kanssa, joka aiheutti synkronointi ongelmia GitHubin kanssa.

Meteorin valinnan vuoksi joutui opiskelemaan sen toimintaa, jossa suurimpia haasteita olivat mongo tietokannan rajapinta toimintojen epäselkeys sekä meteorin palvelin- ja käyttäjäpuolen rajapinta.

# TOIMIVUUS

Meteor mapping servicen toiminta ja käyttö on yksinkertaistettu yhdelle sivulle. Käyttäjä valitsee sinisistä laatikoista (Kuva 1) kaupungit. Kaupunkien valittua ohjelma automaattisesti lähettää palvelimelle kyselyn näiden kahden kaupungin välisestä lyhimmästä reitistä. Vastaus esitetään yhteismatkan määränä sekä minkälaisesta reitistä kyseinen lyhin reitti koostuu, näyttäen jokaisen läpimentävän kaupungin etäisyyden toisistaan.



KUVA 1. Meteor kartta palvelun käyttäjä näkymä.

Create data nappi (Kuva 1) on (toistaiseksi väliaikaisesti) jätetty nappi, mikäli kaupunki tiedot eivät ole luotuna palvelimella valmiina. Täysin toiminnallisessa ja viimeistellyssä versiossa kaupunki tiedot hoidetaan täysin palvelinpuolella.

# OHJELMAN KÄYTTÖÖNOTTO

* Asenna Meteor.js koneelle:
  + <https://www.meteor.com/install>
  + Noudata linkissä annettuja ohjeita.
* Luo projekti:
  + Avaa komentorivi
  + Navigoi kansioon, johon projekti luodaan
  + Luo uusi projekti komennolla ”meteor create {0}”, jossa {0} on projektille annettu (vapaavalintainen) nimi
  + Kopioi kaikki testattavan ohjelman tiedostot tähän kansioon
* Aja luotu projekti
  + Aja komentorivillä tässä kansiossa komento ”meteor run”
  + Avaa verkkoselain
  + Kirjoita osoiteriville ”localhost:3000”
  + Tadaa!

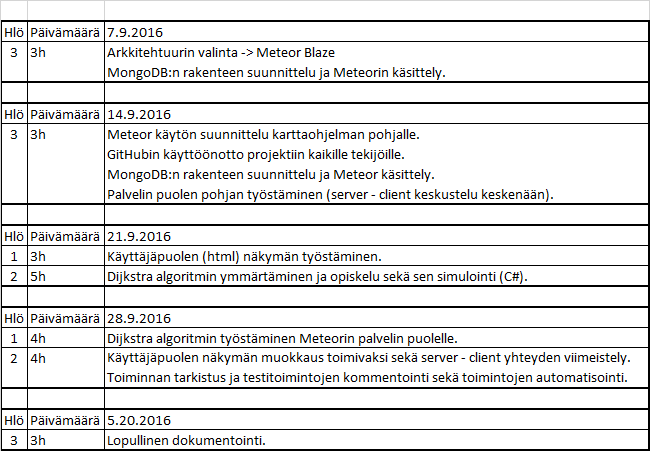
# OHJELMAN TESTAUS

Asenna ohjelma kohdassa 7 annettujen ohjeiden mukaisesti. Valitse pudotusvalikoista lähtö ja määränpää kaupungit (”from” ja ”to”). Tarkista ohjelman antama tulos vertaamalla sitä kuvassa oleviin matkoihin.

Ensimmäisellä ajokerralla pudotusvalikot saattavat olla tyhjiä. Mikäli näin on, paina sivun alalaidassa olevaa painiketta ”Create data”, joka populoi kantaan tarvittavat tiedot.

# TYÖAIKA

Työaikaa kului suhteellisen paljon Meteorin käytön opetteluun. Ongelmia myös koostui tietokoneiden muokattujen erikoisasetusten kanssa, joka alustavasti esti Meteorin käyttöönottamista.



TAULUKKO 1. Projektiin kulutettu aika, kuinka monta henkilöä tehtävää on tehnyt.

Projektissa auttoi työnsiirtoa tekijöiden kesken GitHub, jossa projektia säilytettiin. Tämän avulla työt saatiin jaoteltua eri henkilöille, yksi tekee palvelimen koodia, yksi käyttäjän html näkymää ja yksi käyttäjän koodia.

# YHTEENVETO

Työn pohjalta opimme Meteorin toimintaa ja kuinka sitä pystyttiin käyttämään yksinkertaistamaan rajapintoja palvelimen ja käyttäjänäkymän välillä. GitHubin käyttöä tuli myös opittua.

Karttaohjelmaan voisi lisätä visuaalista näkymää kartasta, joka vaikka näyttäisi millaiselta lyhin reitti näyttää sekä esittää vaihtoehtoisen toiseksi lyhimmän reitin määränpäähän.

Tehtävän annettu esimerkki antoi väärän kuvan tehtävästä, suurin osa työstä oli dijkstra-algoritmin selvittämistä sekä sen integrointia toimivaksi koodiksi omaan ohjelmaan.