Analysaattoridatan etähallinta WebApp

KEHITTÄMISOPAS

Teemu Litmanen, Joni Räsänen, Matti Auvinen ja Tomi Rönkkö
UNIVERSITY OF EASTERN FINLAND | PROJEKTITYÖ

Sisällys

1 Tuotteen kuvaus

Selainpohjainen järjestelmä, jolla käyttäjä voi kätevästi tallentaa mittausdatan ja tarkastella dataa visuaalisesti. Käyttäjä voi syöttää mittalaitteesta saadun tiedoston järjestelmään ja järjestelmä tallentaa mittausdatan tietokantaan. Mittauspaikkojen sijainnit esitetään kartalla ja metallipitoisuudet kuvaajassa. Kuvaajasta käyttäjä näkee eri metallien pitoisuuksien kehityksen aikajanalla.

2 Kehittämis- ja toimintaympäristö

2.1 Kehittämisympäristö

Visual Studio Code

2.2 Versionhallinta

GitHub

2.3 Toimintaympäristö

Sivuston ja tietokannan hostingin hoitaa Heroku.

3 Ohjelmiston rakenne ja toiminnallisuus

Ohjelmisto koostuu React.js pohjaisesta front-endistä, Node.js pohjaisesta back-endistä ja Postgresql tietokannasta.

3.1 client(front-end)

3.1.1 src

3.1.1.1 App.js

Ohjelman pääkomponentti tai main-komponentti.

3.1.1.2 components

Sisältää ohjelman eri ominaisuudet lisäävät komponentit: Kartan piirtävän MapComponentin, kuvaajan piirtävän GraphComponentin ja tiedoston lukemisen hoitavan UploadComponentin.

3.1.1.2.1 GraphComponent.js

Piirtää kuvaajan, jossa X-akselilla mittaustuloksen päivämäärä ja Y-akselilla valitun metallin pitoisuus. Kuvaajan piirtoon on toteutettu Recharts kirjaston avulla. Kirjaston dokumentaatio.

Muuttujat:

data [] : Sisältää kaiken tietokannasta haetun mittausdatan json muodossa.

databyid [] : Sisältää valitussa mittauspaikassa otetun mittausdatan.

this.props.currentId : Valittuun mittauspaikkaan liittyvän mittausdatan test_id.

Funktiot:

getDataFromDb : Lähettää pyynnön backendille hakea tietokannasta mittausdata. Haettu mittausdata määritetään data[] muuttujaan.

data.map : Käy läpi mittausdatan ja valitsee sieltä valittuun mittauspaikkaan liittyvät mittaukset. Valitut mittaukset lisätään databyid[] taulukkoon.

Elementit:

<LineChart/> : Määrittää kuvaajan tyypin (LineChart), kuvaajan leveyden (width), korkeuden (height) ja kuvaajan piirtoon käytettävän data (data).

<Line/>: Määrittää kuvaajan visuaalisen tyylin (type, stroke), minkä mittausdatan sisältämän kentän mukaan datapisteet piiretään (dataKey).

<CartesianGrid/>: Määrittää kuvaajan taustalle piirrettävän ruudukon.

<Tooltip/>: Määrittää kuvaajassa hiiren “hoveroinnilla” esiin tulevan infotekstin.

<X/YAxis/>: Määrittää mitä muuttujaa kyseisellä akselilla esitetään (dataKey).

3.1.1.2.2 MapComponent.js

Piirtää kartan ja esittää siinä mittauspaikkojen sijainnit sijaintimerkeillä. Kartta käyttää Google Maps Apia ja google-maps-react kirjastoa. Kirjaston dokumentaatio.

Muuttujat:

data [] : Sisältää kaiken tietokannasta haetun mittausdatan json muodossa.

showingInfoWindow : Kertoo näkyykö info-ruutu (true/false).

activeMarker : Kertoo mikä mittauspaikka on valittuna kartalla.

selectedPlace : Sisältää kartalla valittuna olevan mittauspaikka merkin tiedot.

Funktiot:

getDataFromDb : Lähettää pyynnön backendille hakea tietokannasta mittausdata. Haettu mittausdata määritetään data[] muuttujaan.

data.map : Käy läpi mittausdatan ja määrittää <Marker/> elementin jokaista mittauspaikkaa kohden.

onMarkerClick : Päivittää showingInfoWindow, activeMarker ja selectedPlace muuttujat valitun mittauspaikan mukaisiksi. Lähettää pääkomponentille (App.js) valitun mittauspaikkaan liittyvän mittausdatan test_id:n (, joka sitten lähettää test_id:n GraphComponentille, joka päivittää kuvaajan datan valitun mittauspaikan mukaiseksi).

onClose : Päivittää tiedot showingInfoWindow ja activeMarker muuttulijje, että inforuutu on suljettu.

Elementit:

<Map/> : Määrittää piirettävän kartan, sen tyylin (style) ja aluksi esitettävän sijainnin (initialCenter). Sijainti määritetään leveys- (lat) ja pituusasteina (lng).

<Marker/> : Määrittää piirettävän sijantimerkin. Saa muutujina sijainnin (position), nimen (name), mittauspaikkaan liittyvän test_id:n. Merkkiä painettaessa suoritetaan funktio onMarkerClick.

<InfoWindow/> : Määritä piirettävän inforuudun.

3.1.1.2.3 UploadComponent.js

UploadComponent pitää sisällään mahdollisuuden laajentaa jatkossa toiminnallisuutta, mutta koostuu tällä hetkellä “child componentista” DropzoneComponent.

Funktiot:

render (): Renderöi alicomponentit, eli React.Fragment komponentin sisällä olevat komponentit.

Elementit:

<React.Fragment>: Mahdollistaa komponenttien näkymisen allekkain. Tällä hetkellä sisältää vain yhden comomponentin.

3.1.1.2.4 DropzoneComponent.js

Tämä komponentti ottaa vastaan CSV –tiedoston ja parseroi sen tiedot ja syöttää ne POST komennolla apille. CSV –tiedoston muodon on vastattava määrittelyjä, jotta API:a varten tiedot ovat halutun mukaiset ja tiedoston ja API:n yhteys toimii oikein.

Tiedosto luetaan ensin CSV:stä Arrayksi, jonka jälkeen itse arrayta voidaan lukea ja käyttää muuttujana ‘data’ muuttujassa ‘constHandleforce” jossa lähetetään axios –kirjaston avulla apile POST-kutsu CSV:stä saaduilla ja parsetetuilla muuttujilla.

Funktiot:

Const reader = (): Käyttyyä selaimen omaa tiedostonlataus ikkunaa, ja tallentaa ladatun tiedoston, kun se on kokonaan ladattu, muuttujaksi ‘data’.

Elementit:

‘react-csv-reader': Kirjasto, joka osaa lukea CSV tiedoston Array-muotoon, jotta sitä voidaan jatkokäsitellä muissa funktioissa.

3.2 backend

3.2.1 server.js

On yhteydessä tietokantaan ja käsittelee front-endiltä saapuvia pyyntöjä. Toimii ns. välikätenä tietokannan ja front-endin välillä. Tietokantahakuihin käytetään apuna pg-promise kirjastoa. Front-endin väliseen kommunikaatioon ja pyyntöjen käsittelyyn käytetään express kirjastoa. Pg-promise kirjaston dokumentaatio. Express kirjaston dokumentaatio.

Muuttujat:

API_PORT : Määrittää mitä porttia backend kuuntelee.

dbRoute : Määrittää tietokannan osoitteen. Osoite sisältää palvelimen nimen, käyttäjätunnuksen ja salasanan.

db : Luo yhteyden tietokantaan pg-promise kirjaston avulla. Tämän kautta tehdään hakuja tietokantaan. Haut tehdään pitkälti SQL-lauseilla. Tarkemmat tiedot hakujen määrittelystä löytyy pg-promisen dokumentaatiosta.

app : Hoitaa front-endiltä käsiteltävät pyynnöt. Express-palvelin instanssi.

Funktiot:

app.get("/api/getData") : Käsittelee kyseisellä osoitteella front-endiltä saapuvan get-pyynnön. Hakee tietokannasta kaiken test-taulussa olevan datan ja palauttaa sen front-endille.

app.post("/api/putData") : Käsittelee kyseisellä osoitteella front-endiltä saapuvan post-pyynnön. Saa front-endiltä tietokantaan tallennettavaa dataa ja tallentaa sen tietokantaan test-tauluun. Tietokantakutsujen määrittelystä ja notaatiosta enemmän pg-promisen dokumentaatiossa.

3.3 Tietokanta

Tietokanta on PostgreSQL pohjainen.

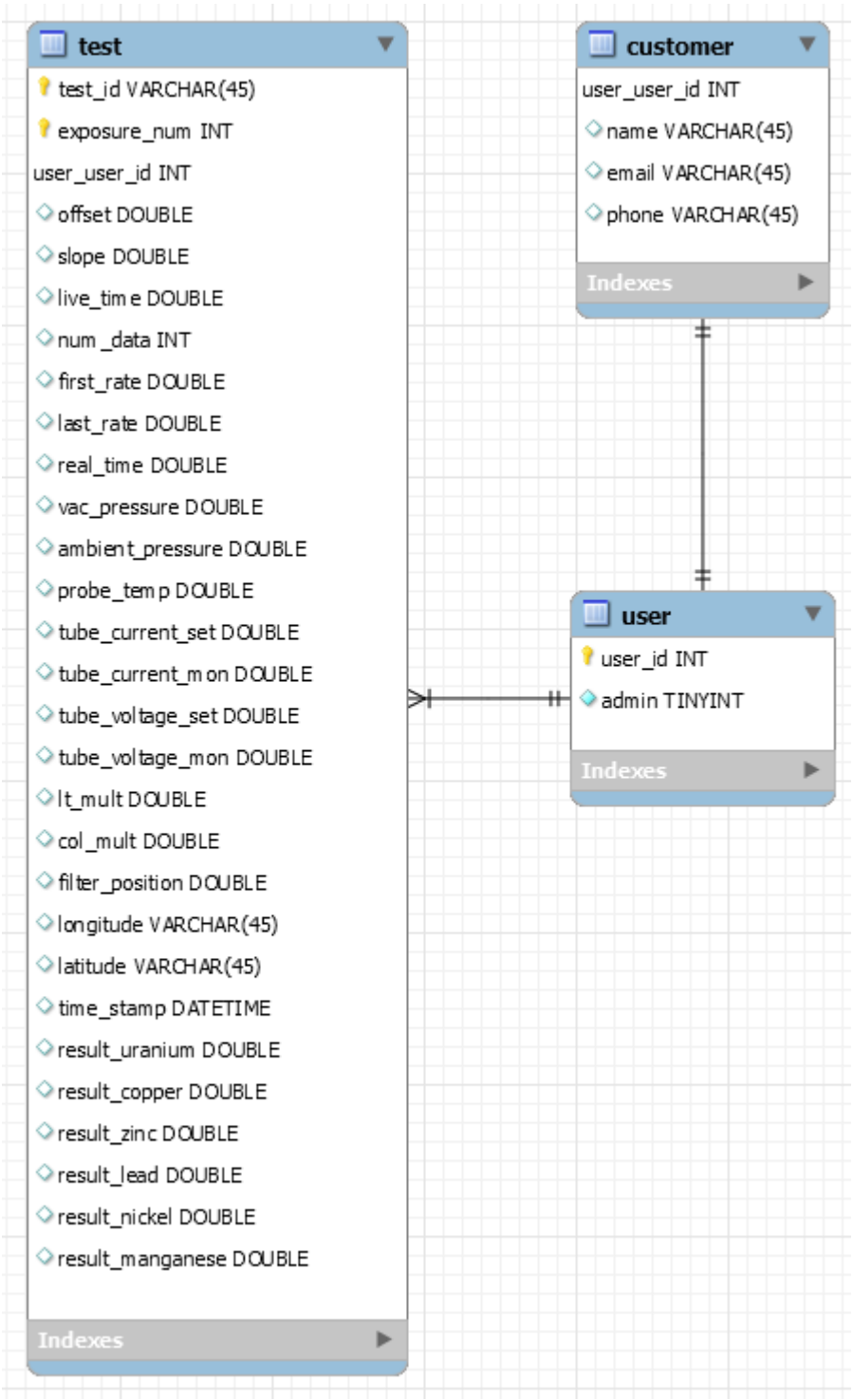
3.3.1 Tietokantamalli

Tietokanta ei täysin toteutunut mallin mukaisesti. Customer ja user tauluja ei ole luotu varsinaiseen tietokantaan ja muutamia muutoksia myös test tauluun on tehty. Kuva tietokantamallista löytyy alta (kuva1)

3.3.2 Tietokannan rakenne

Sisältää test-taulun, jossa on kaikki tarvittavat kentät mittalaitteesta saatavan datan tallennukseen. Alla olevasta taulukosta (taulukko1) löytyy jokaisen kentän nimi ja tietotyyppi. Primary key muodostuu kentistä test_id, user_id ja exposure_num. Eli näiden kenttien yhdistelmän täytyy olla uniikki. Test-taulun luontilauseet löytyy Githubista tiedostosta "vesianalyysidb_test.txt".

TAULUKKO 1


test
test_id VARCHAR(45)
exposure_num INT
user_user_id INT
offset DOUBLE
slope DOUBLE
live_time DOUBLE
num _data INT
first_rate DOUBLE
last_rate DOUBLE
real_time DOUBLE
vac_pressure DOUBLE
ambient_pressure DOUBLE
probe_temp DOUBLE
tube_current_set DOUBLE
tube_current_mon DOUBLE
tube_voltage_set DOUBLE
tube_voltage_mon DOUBLE
lt_mult DOUBLE
col_mult DOUBLE
filter_position DOUBLE
longitude VARCHAR(45)
latitude VARCHAR(45)
time_stamp DATETIME
result_uranium DOUBLE
result_copper DOUBLE
result_zinc DOUBLE
result_lead DOUBLE
result_nickel DOUBLE
result_manganese DOUBLE
Indexes
customer
user_user_id INT
name VARCHAR(45)
email VARCHAR(45)
phone VARCHAR(45)
Indexes
user
user_id INT
admin TINYINT
Indexes

TAULUKKO 2

Nimi	Tietotyyppi	Kuvaus
test_id	varchar(45)	Testille määritetty id
user_id	varchar(45)	Käyttäjälle määritetty id
exposure_num	integer	ExposureNum numerosta katsotaan, mitä metallipitoisuuksia kyseisen sarakkeen arvoilla lasketaan ExposureNum = 1: Uraani ExposureNum = 2: Kupari, Sinkki, Lyijy, Nikkeli ExposureNum = 3: Mangaani
offset	real	
slope	real	
live_time	real	
num_data	integer	
first_rate	real	
last_rate	real	
real_time	real	Mittauksen kesto. Käytetään mittausten normalisointiin mittausajan suhteen.
vac_pressure	real	
ambient_pressure	real	
probe_temp	real	
tube_current_set	real	
tube_current_mon	real	
tube_voltage_set	real	
tube_voltage_mon	real	
lt_mult	real	
colt_mult	real	

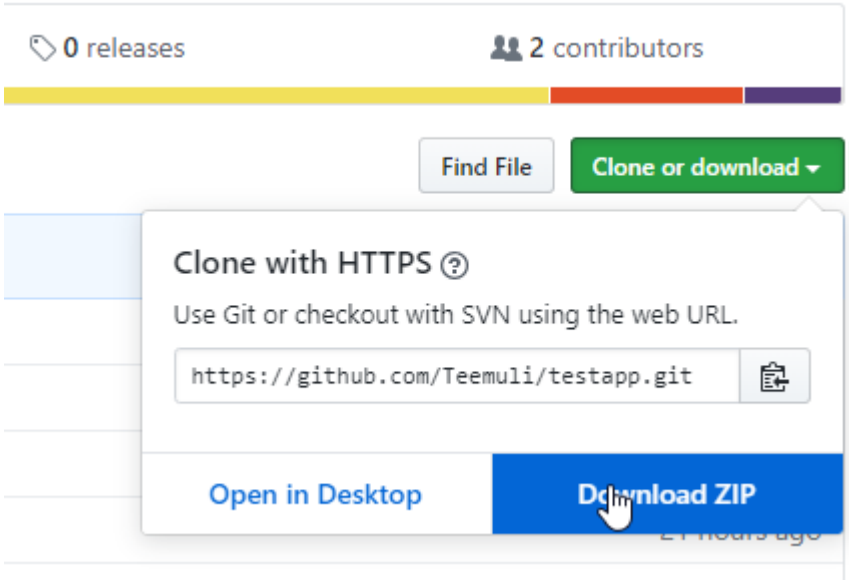
filter_position	real	
time_stamp	timestamp w/o time zone	
result_uranium	real	Mittausdatasta riviltä 703 luettu arvo. Uraanipitoisuus.
result_copper	real	Mittausdatasta riviltä 425 luettu arvo. Kuparipitoisuus.
result_zinc	real	Mittausdatasta riviltä 455 luettu arvo. Sinkkipitoisuus.
result_lead	real	Mittausdatasta riviltä 550 luettu arvo. Lyijypitoisuus.
result_nickel	real	Mittausdatasta riviltä 397 luettu arvo. Nikkelipitoisuus.
result_manganese	real	Mittausdatasta riviltä 318 luettu arvo. Mangaanipitoisuus.
spectrum_data	integer[]	Sisältää mittausdatan sisältävän csv. tiedoston riveltä 23-2070 olevan datan
latitude	real	Mittauspaikan leveysaste
longitude	real	Mittauspaikan pituusaste

4 Asennus ja käyttäminen

Sivusto löytyy osoitteesta https://vesianalyysi.herokuapp.com/. Jos haluat kuitenkin ajaa sivuston localisti, seuraa alla olevaa asennusohjetta.

4.1 Asennus

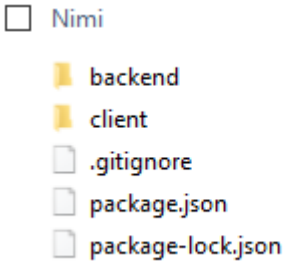
1. Aloita lataamalla projektitiedosto (.zip) githubista.



2. Ladattuasi tiedoston (testapp-master.zip), pura se haluamaasi sijaintiin.

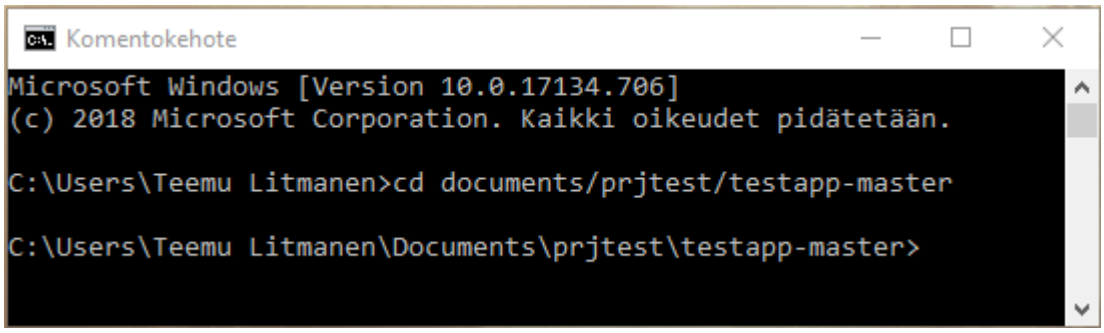
3. Kansion testapp-master sisällä tulisi olla toinen saman niminen kansio. Siirrä se ulos kansiosta.

4. Nyt testapp-master kansion sisältö tulisi näyttää alla olevan kuvan mukaiselta.



5. Asenna node.js.

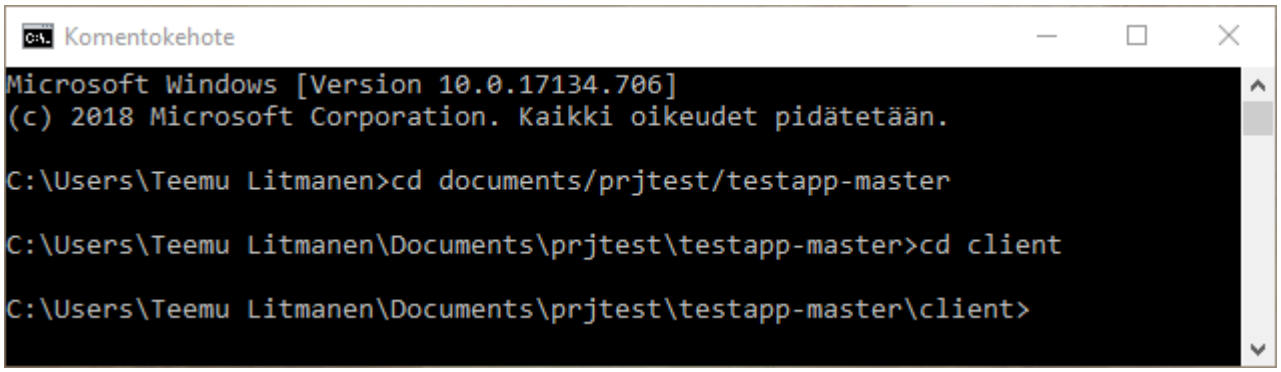
6. Avaa komentokehote ja siirry testapp-master kansioon.



7. Suorita seuraava komento: npm install body-parser cors express morgan pg pg-promise axios. Tämä asentaa tarvittavat kirjastot backendille.

```
C:\Users\Teemu Litmanen\Documents\prjtest\testapp-master>npm install body-parser
cors express morgan pg pg-promise axios
```

8. Siirry kansioon client.



9. Suorita seuraava komento: npm install. Tämä asentaa tarvittavat kirjastot frontendille.

```
C:\Users\Teemu Litmanen\Documents\prjtest\testapp-master\client>npm install
```

10. Jätä komentokehote päälle ja avaa toinen komentokehote. Backend ja frontend vaativat omat komentokehotteet, jotta saamme ne yhtäaikaa päälle.

11. Siirry ensin toisessa komentokehotteessa kansioon testapp-master ja sitten kansioon backend.

```
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. Kaikki oikeudet pidätetään.

C:\Users\Teemu Litmanen>cd documents/prjtest/testapp-master/backend

C:\Users\Teemu Litmanen\Documents\prjtest\testapp-master\backend>
```

12. Suorita seuraava komento: node server.js. Tämä käynnistää backendin. Backend täytyy aina käynnistää ennen frontendiä. Komentokehotteessa tulisi nyt lukea ”LISTENING ON PORT 3001”.

```
C:\Users\Teemu Litmanen\Documents\prjtest\testapp-master\backend>node server.js
LISTENING ON PORT 3001
```

13. Jätä tämä komentokehote päälle ja siirry toiseen komentokehotteeseen. Siirry kansioon client, jos et siellä vielä ole.

```
C:\Users\Teemu Litmanen\Documents\prjtest\testapp-master>cd client

C:\Users\Teemu Litmanen\Documents\prjtest\testapp-master\client>
```

14. Suorita seuraava komento: npm start. Tämä käynnistää frontendin.

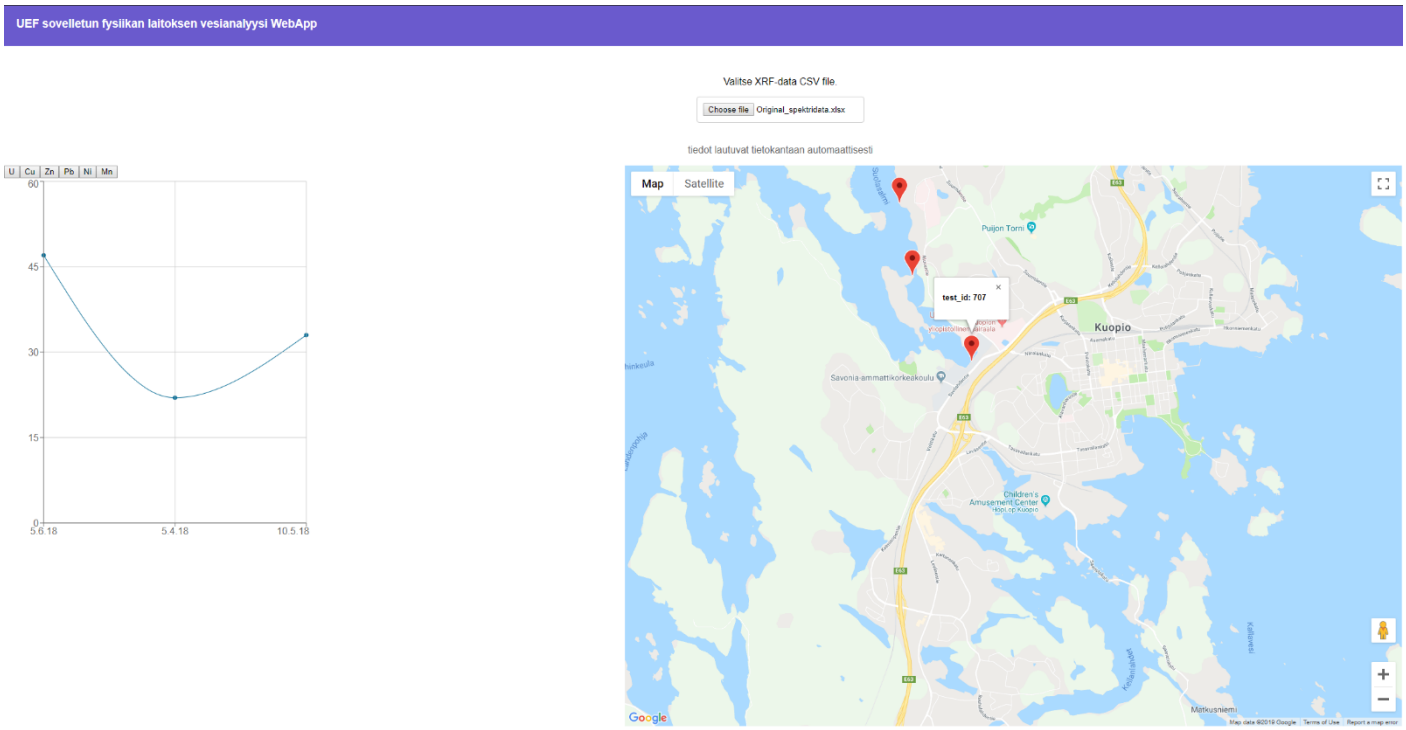
```
C:\Users\Teemu Litmanen\Documents\prjtest\testapp-master\client>npm start
```

Komentokehotteen tulisi näyttää tältä:

```
> react-scripts start
Starting the development server...
Compiled successfully!

You can now view client in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://192.168.1.104:3000/

Note that the development build is not optimized.
To create a production build, use npm run build.
```

Tämän tulisi avata automaattisesti projekti selaimessa. Jos sivusto ei kuitenkaan aukea, avaa itse selain ja siirry osoitteeseen http://localhost:3000/. Sivuston pitäisi nyt näkyä selaimessa.

4.2 Käyttäminen



Tietokantaan lisätään uusia mittaustuloksia ylhäällä keskellä olevasta “Choose File” -painikkeesta. Kun haluttu tiedosto on valittu ja selainikkuna ladattu uudelleen, muutokset tulevat näkyviin oikealla olevalle kartalle.

Jokainen punainen merkki sisältää yhden mittauspaikan kaikki mittaustulokset. Mittauspaikka valitaan painamalla halutun paikan merkkiä. Valinnan jälkeen vasemmalla näkyvät eri metallien mitatut pitoisuudet eri päivinä. Kuvaajan yläpuolella olevien painikkeiden avulla voidaan valita näytettäväksi halutun metallin mitatut pitoisuudet, minkä jälkeen kuvaaja päivittyy automaattisesti näyttämään tämän metallin mittaustulokset.

5 Tunnetut ongelmat ja ideat

5.1 Tunnetut ongelmat

- Tiedoston lautauksen jälkeen selainikkuna päivitettävä, jotta data näkyy myös kartalla.
- Onnistuneesta latauksesta ei tule promptia.
- Sivusto ei lue suoraan varsinaista mittauslaitteesta saatua csv-tiedostoa. Pitäisi päivittää backendin puolella olevat insert-lauseet.

5.2 Jatkokehitys ideat

- Lisää prompti onnistuneesta tai epäonnistuneesta latauksesta.
- Lisää autorefresh selaimen ikkunalle (tai Google Map komponentille), jotta uudet ladatut tiedot näkyvät heti, ilman selainikkuan virkistämistä.
- Lisää käyttäjänhallinta. Sivustolle tulisi kirjautua ja jokainen käyttäjä näkisi vain omat mittauksensa.
- Lisää tietokantaan taulu käyttäjien tietojen tallennukseen.
- Lisää pääkäyttäjä. Pääkäyttäjällä olisi pääsy kaikkeen dataan.
- Lisää datan analyysi.
- Päivitä csv-tiedoston luku.
- Päivitä ulkoasua.