TAMPEREEN
AMMATTIKORKEAKOULU

eTruck-mobiilisovellus

Saku Lehtinen

Opinnäytetyö
Joulukuu 2017
Tietotekniikan koulutusohjelma
Sulautetut järjestelmät ja elektroniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Sulautetut järjestelmät ja elektroniikka

LEHTINEN, SAKU:
eTruck-mobiilisovellus

Opinnäytetyö 25 sivua
Joulukuu 2017

Tämän opinnäytetyön tarkoituksena oli kehittää sähköajoneuvojen etäseurantaan tarkoitettu älypuhelinsovellus, jolla voi seurata siihen liitettyjen ajoneuvojen tilaa reaaliaikaisesti. Sovellus näyttää seurattavien kohteiden sijainnin kartalla ja kertoo muun muassa niiden tämänhetkisen nopeuden ja akuston varaustilan. Lisäksi sovelluksessa on mahdollista tarkistella koko päivän aikana kuljettua reittiä ja koko päivän energiankulutusta.

Opinnäytetyönä tehty sovellus on toteutettu osana Tampereen ammattikorkeakoulun koordinoimaa ja Euroopan aluekehitysrahasto EAKR:n rahoittamaa eTruck-hanketta, jonka tavoitteena on luoda innovaatioalusta sähkökäyttöisen kuljetuskaluston soveltuvuuden ja taloudellisuuden selvittämiseen Suomen olosuhteissa.

Sovellus kehitettiin käyttäen Apache Cordova -sovelluskehystä, joka mahdollistaa sovelluskehityksen älypuhelimille käyttäen web-tekniikoita: HTML, CSS ja JavaScript. Cordova oli sovelluksen kehittämiseen erittäin toimiva ratkaisu.

Lopputuloksena saatiin helppokäyttöinen, toimiva ja hyödyllinen sovellus, joka kertoo yhdellä vilkaisulla seurattavien ajoneuvojen sijainnin ja mahdollistaa helpon reittihistorian tarkastelemisen.

Asiasanat: mobiilisovellus, Cordova, JavaScript, esineiden internet

ABSTRACT

Tampere University of Applied Sciences
Degree Programme in ICT Engineering
Embedded Systems and Electronics

LEHTINEN, SAKU:
eTruck Mobile Application

Bachelor's thesis 25 pages
December 2017

The purpose of this thesis was to develop a smartphone application for remote monitoring of electric vehicles, which can monitor the state of the vehicles attached to them in real time. The application shows the location of the objects to be tracked on the map, explaining, among other things, their current speed and battery charge. In addition, the application allows you to check the route traveled throughout the day and the energy consumed during the day.

Mobile application was developed as a part of eTruck project coordinated by Tampere University of Applied Sciences. The project aims to create an innovation platform for assessing the suitability and economy of electric vehicles in Finnish conditions. The project is funded by European Regional Development Fund ERDF.

The application was developed using the Apache Cordova application framework that enables application development for smartphones using web technologies: HTML, CSS, and JavaScript. Cordova was a very effective solution for developing a mobile application.

The result was an easy-to-use, functional and useful application that shows the location of the vehicles to be monitored at one glance and enables easy viewing of the route history.

Key words: mobile application, Cordova, JavaScript, Internet of things

SISÄLLYS

LYHENTEET JA TERMIT

CAN-väylä	Controller Area Network, automaatioväylä jota käytetään ajoneuvoissa ja teollisuuskoneissa.
GPS	Global Position System, satelliittipaikannusjärjestelmä.
Sovelluskehys	Framework, ohjelmisto, joka muodostaa pohjan kehitettävälle sovellukselle.
JSON	JavaScript Object Notation, yksinkertainen avoimen standardin tiedostomuoto tiedonvälitykseen
Modbus	Teollisuudessa käytetty sarjaliikenneprotokolla
AJAX	Asynchronous JavaScript And XML, web-tekniikka asynkronisten pyyntöjen lähettämiseen
API	Ohjelmointirajapinta, jonka avulla eri ohjelmat tai ohjelman osat voivat vaihtaa tietoa keskenään.

1 JOHDANTO

Tämä opinnäytetyön tarkoituksena oli toteuttaa sähköajoneuvojen etäseurantaan tarkoitettu mobiilisovellus, joka seuraa siihen liitettyjen ajoneuvojen ja latauspisteiden tilaa reaaliaikaisesti ja visualisoi kerättyä dataa kuvaajin ja karttamerkinnöin. Seurattavia asioita ovat muun muassa sijainti, nopeus, energiankulutus, kuljettu matka ja regeneroitu energia.

Sovellus tehtiin osana Tampereen ammattikorkeakoulun koordinoimaa ja Euroopan aluekehitysrahasto EAKR:n rajoittamaa eTruck-hanketta, jonka tavoitteena on luoda innovaatioalusta sähkökäyttöisen kuljetuskaluston soveltuvuuden ja taloudellisuuden selvittämiseen Suomen olosuhteissa.

Seurattaviin kohteisiin on asennettu etävalvonnan mahdollistama laite, joka lukee kohteen CAN- tai modbus-väylältä tulevia viestejä ja välittää niistä halutut tiedot palvelimelle, josta tiedot saadaan sovelluksen käytettäväksi. Lisäksi laitteessa on GPS-paikannin, joka mahdollistaa sijaintidatan keräämisen.

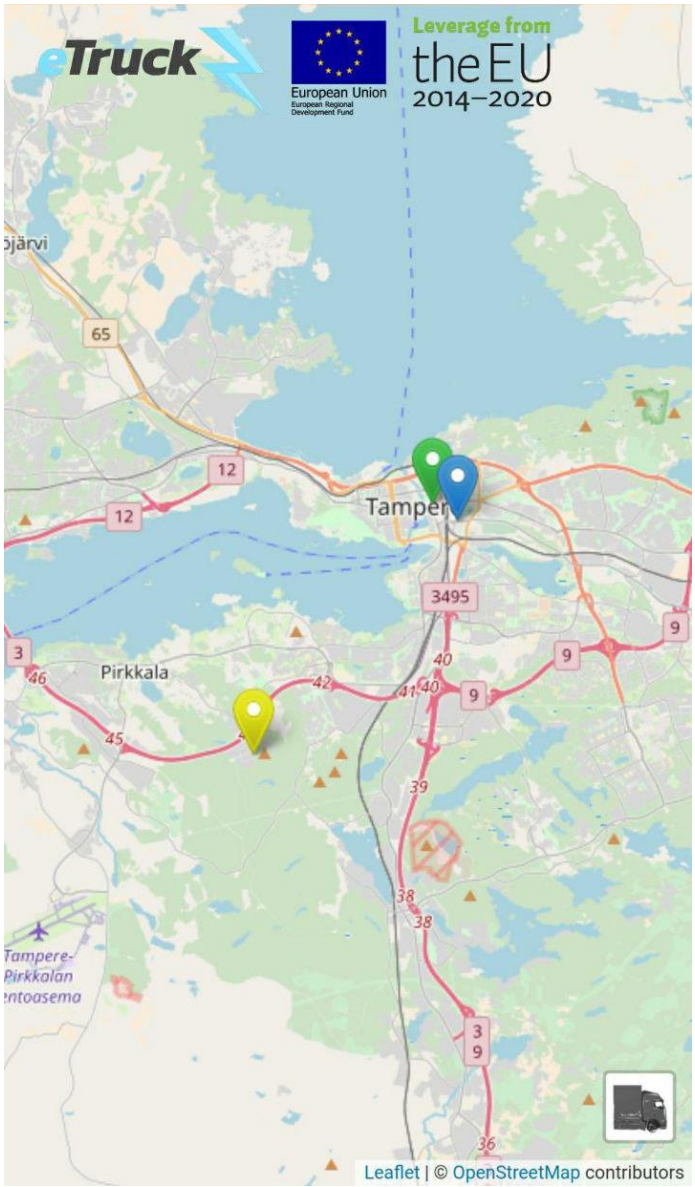
Sovelluksen suunnittelussa vaatimuksena oli helppokäyttöisyys, ja sovellus toteutettiin niin, että käyttäjän ei tarvitse missään vaiheessa säätää asetuksia valikoista tai esimerkiksi valita päivämäärää.

Sovellus on rakennettu Apache Cordova -sovelluskehyksen päälle ja työkaluina sovelluskehityksessä käytettiin Atom-tekstieditoria ja git-versionhallintaa. Sovellusta kehittämässä opinnäytetyön tekijän lisäksi on ollut toinen tietotekniikan opiskelija Erno Mäkelä.

2 SOVELLUS

2.1 Päänäkymä

Päänäkymä on karttaruutu, jossa näkyy kaikki tällä hetkellä saatavilla olevat kohteet. Erityyppiset kohteet on merkattu erilaisilla väreillä. Kohteen valitseminen kartalla avaa lisätietopalkin, jossa on tietoa laitteen tämänhetkisestä nopeudesta ja tehonkulutuksesta, sekä kohteesta riippuen erilaisia lisävalintoja. Kuvassa 1 näkyy päänäkymä, jossa on nähtävissä kolme eriväristä kohdetta. Kaikki merkit toimivat myös painikkeina, ja kohteen valitseminen aukaisee lisätietoja sisältävän ponnahdusikkunan. Kuvan oikeassa alakulmassa oleva painike avaa listan seurattavista kohteista. Karttanäkymä päivittyy 10 sekunnin välein.



KUVA 1. Sovelluksen päänäkymä

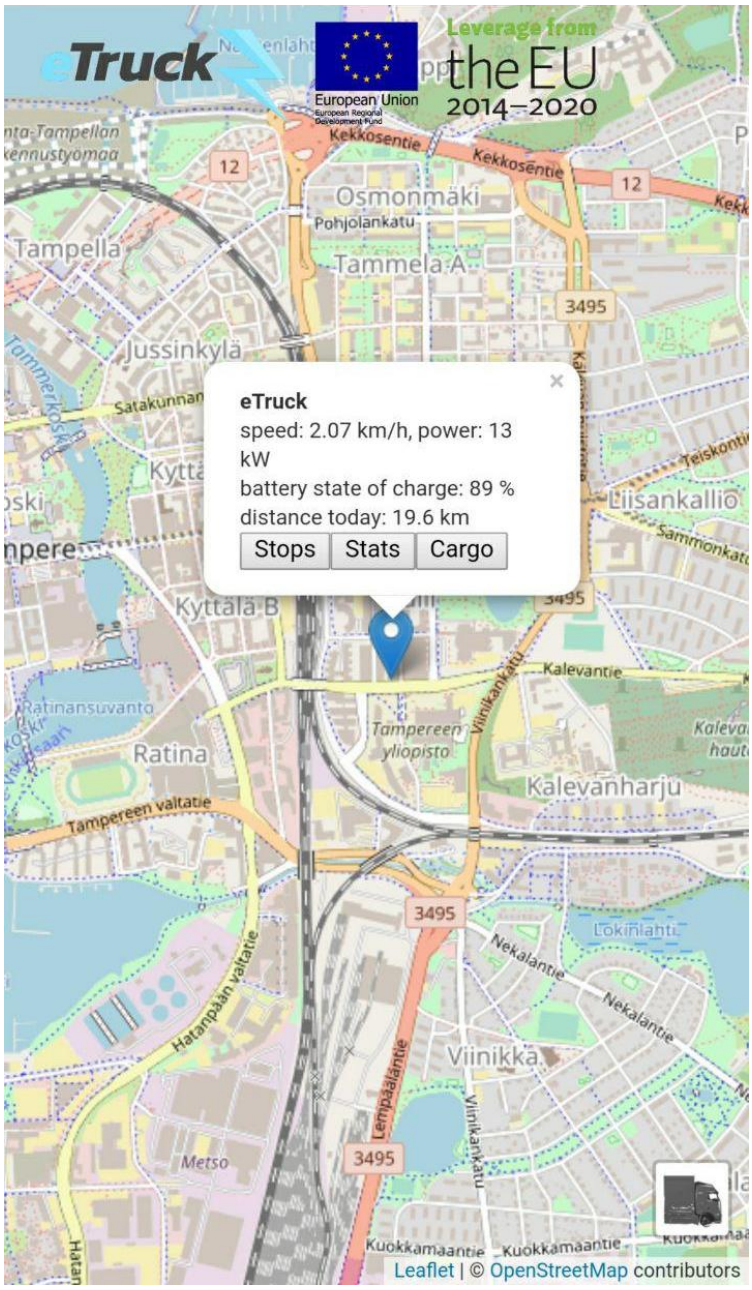
Sovelluksen käytöstä haluttiin tehdä mahdollisimman helppoa ja koska seurattavia kohteita sovelluksessa on vain muutama, päädyttiin esimerkiksi jonkinlaisen valikon tai listan sijaan tehdä sovelluksen päänäkymästä suoraan kartta, joka näyttäisi kaikki seurattavat kohteet. Haluttiin, että käyttäjä pääsisi tarkastelemaan ajoneuvojen sijaintia ilman, että pitäisi esimerkiksi valita jokin ajoneuvo listalta tai painaa useaa painiketta.

2.2 Erilaiset kohteet

Sovelluksen kartalla on käytössä kolmea erityyppistä kohdetta. Sininen merkki kartalla merkkaa sähkökuorma-autoa, vihreällä merkitään sähkölinja-autoa ja keltaisella kiinteää latauspistettä. Kaikki kartalla olevat merkit toimivat myös painikkeina, jotka avaavat ponnahdusikkunan, jossa näkyy hieman erilaisia tietoja, riippuen siitä minkälainen kohde on kyseessä.

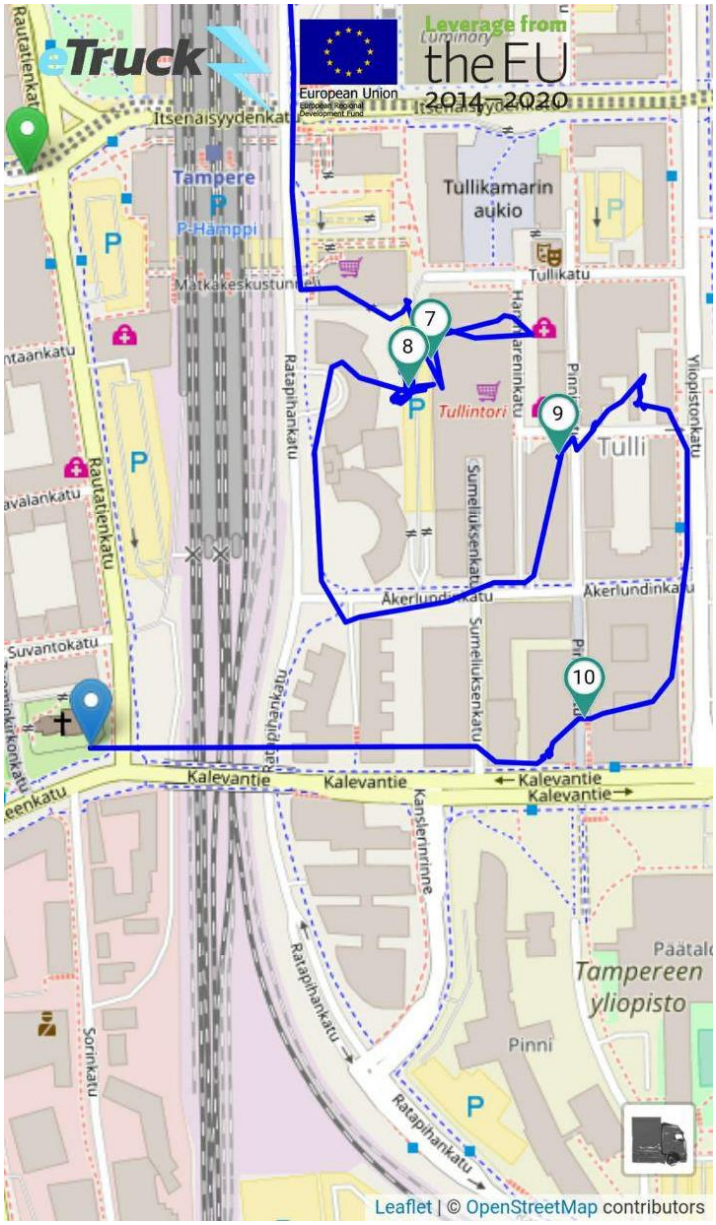
2.2.1 Sähkökuorma-autot

Sinisen merkin, eli kuorma-auton valitsemalla ponnahdusikkuna näyttää sen tämänhetkisen nopeuden, moottorin tehon, akun varaustilan ja päivän aikana kuljetun matkan. Lisäksi ponnahdusikkunassa löytyy “stops”, “stats” ja “cargo” -lisävalinnat. Kuvassa 2 on valittuna sähkökuorma-auto.

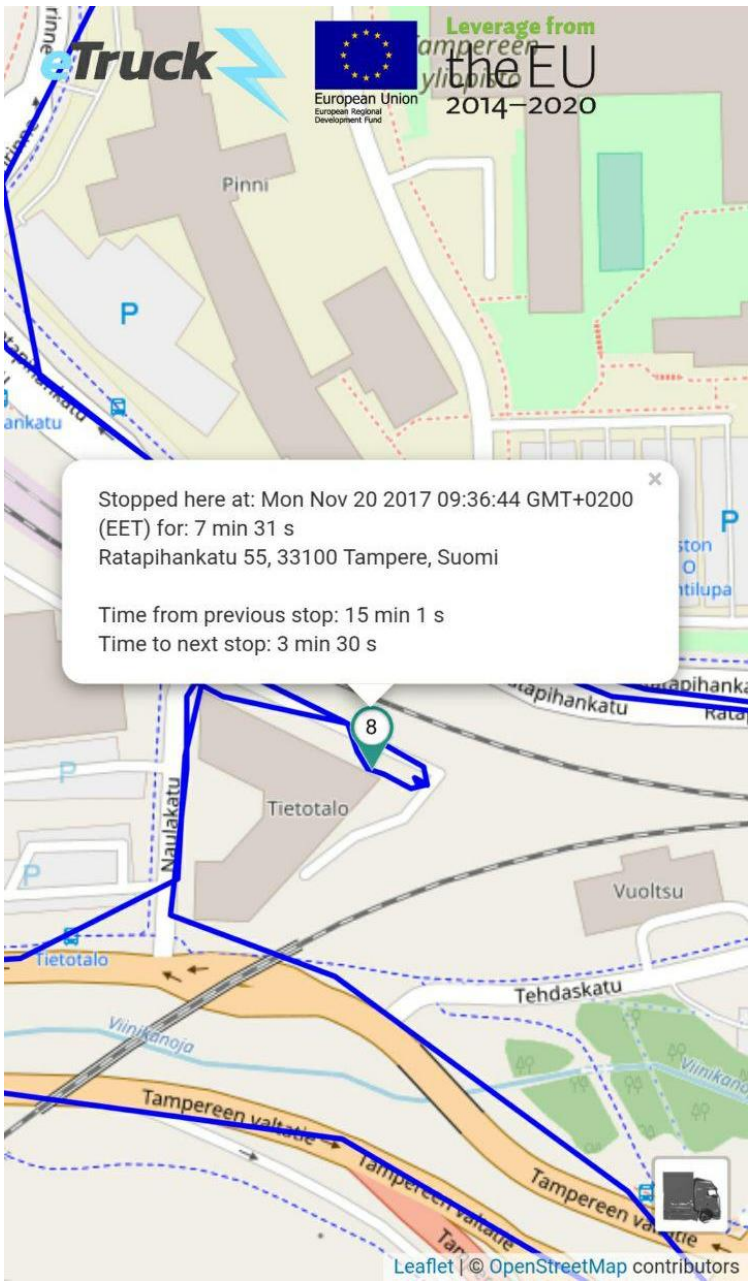


KUVA 2. Sähkökuorma-auto valittuna

“Stops”, eli pysähdykset -valinta piirtää kartalle auton tänään ajetun reitin ja tänään tehdyt pysähdykset. Kuvassa 3 on piirretty sähkökuorma-auton ajamaa reittiä. Kuvassa näkyvät numeroidut karttamerkit kertovat, että auto on pysähtynyt niillä paikoilla. Pysähdysmerkin valitsemalla saa taas lisätietoja pysähdyksestä itsestään. Kuvassa 4 on valittu yksi pysähdysmerkeistä ja lisätietona nähdään koska auto on pysähtynyt, kuinka kauan se on ollut pysähtyneenä, mikä on pysähdyspaikkaa lähinnä oleva osoite, kuinka kauan pysähdyspaikkaan kesti ajaa edelliseltä pysäkiltä ja kuinka kauan matka seuraavalle pysäkille on kestänyt.

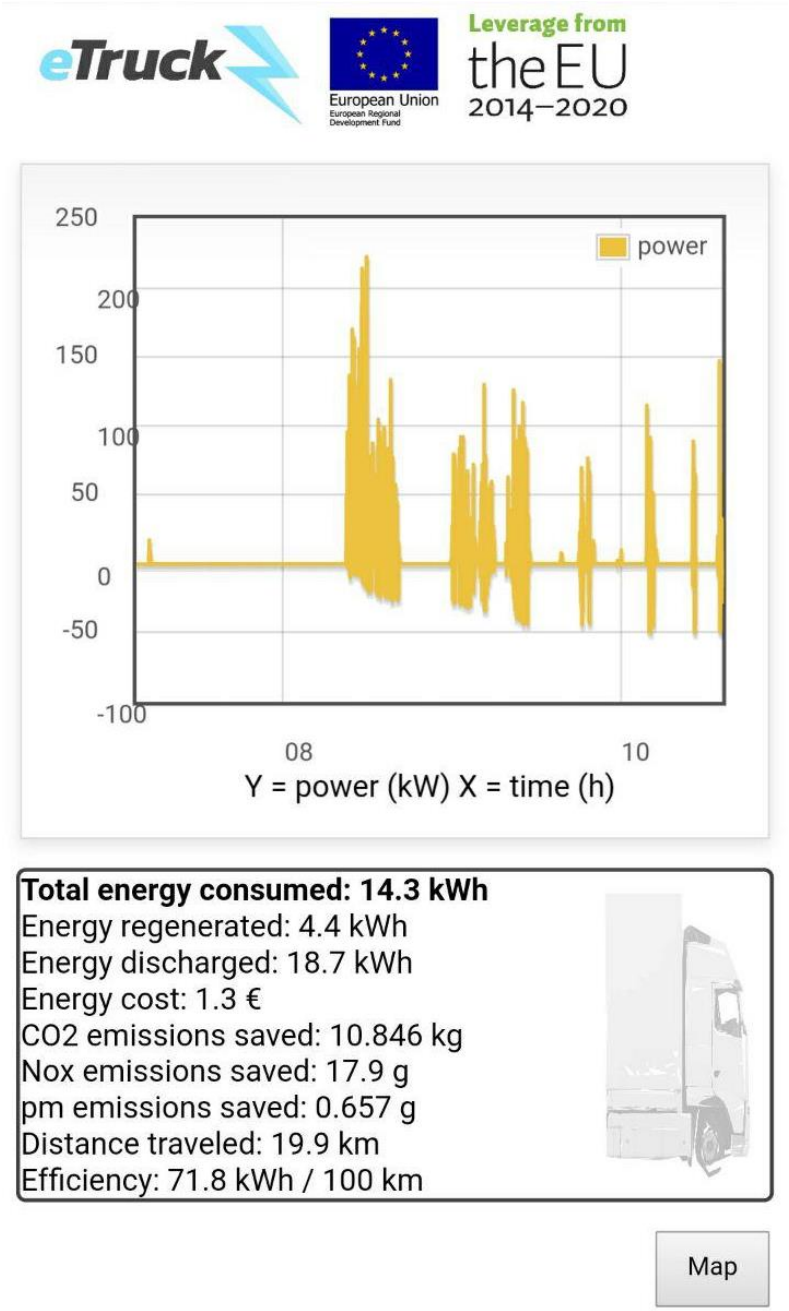


KUVA 3. Sähkökuorma-auton ajamaa reittiä ja pysähdyksiä



KUVA 4. Pysähdyksen ponnahdusikkuna

“Stats” eli tilastot valitsemalla nähdään tarkempaa tietoa ajoneuvon tämän päivän energiankulutuksesta. Sovellus piirtää päivän tehonkäytöstä teho–aika-kuvaajan ja näyttää käytetyn energian määrän, regeneroidun energian määrän, käytetyn kokonaisenergian määrän, energian hinnan, arvioita säästetyistä päästömääristä, kuljetun matkan ja energiankulutuksen sadalta kilometriltä. Kuvaajassa näkyvät negatiiviset tehot tarkoittavat käytännössä energian regenerointia. Kuvassa 5 on näkyvillä sähkökuorma-auton tilastosivu

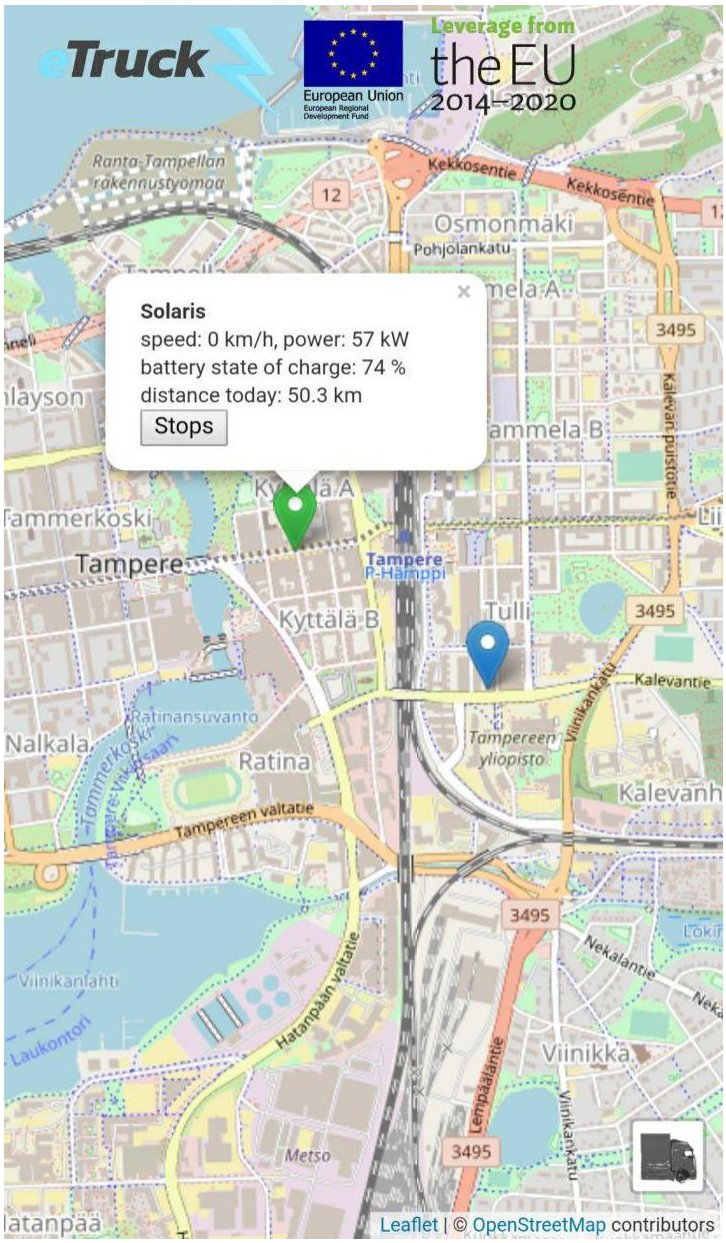


KUVA 5. Sähkökuorma-auton tilastosivu

“Cargo” eli kuormasivu on tehty tulevaisuutta ajatellen ja hakee listan tietokannasta, jossa voisi olla tietoa auton kuormatilassa olevasta tavarasta. Tällä hetkellä seurattavissa ajoneuvoissa ei ole käytännössä tallaista älykästä kuormatilaa, mutta yksi sovelluksen vaadituista ominaisuuksista oli tämä ominaisuus.

2.2.2 Sähköbussit

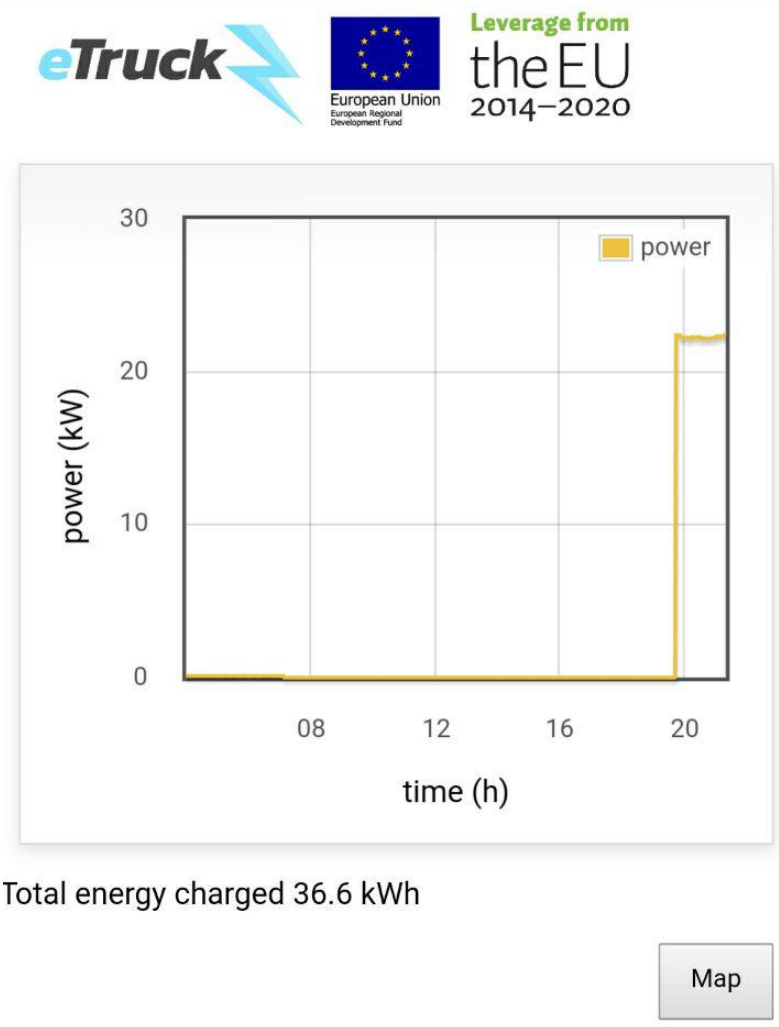
Vihreä merkki kartalla tarkoittaa sähköbussia. Bussilta saadaan ja halutaan näyttää hieman erilaista tietoa kuin kuorma-autoilta. Kuvassa 6 näkyvä bussin ponnahdusikkuna kertoo tämänhetkisen nopeuden, tehonkäytön ja päivän aikana ajetut kilometrit. Bussilla on lisäksi samanlainen “stops” painike kuin kuorma-autoilla ja se toimii täysin samalla tavalla. Bussille ei ole tehty tilastosivua, sillä sitä ei pidetty tarpeellisena.



KUVA 6. Sähköbussin ponnahdusikkuna

2.2.3 Latausasemat

Keltaiset merkit on varattu latausasemille. Valittaessa latausaseman, ponnahdusikkuna kertoo päivän aikana ladatun kokonaisenergian ja tämänhetkisen lataustehon. Lisäksi ponnahdusikkunassa on “stats” lisävalinta, jonka valitsemalla sovellus piirtää kuvaajan päivän lataustehosta ajan funktiona. Kuvassa 7 näkyy latausaseman tehokuvaaja päivältä.



KUVA 7. Latausaseman tilastosivu

3 KÄYTETYT TEKNOLOGIAT

3.1 Cordova

Apache Cordova on avoimen lähdekoodin, mobiilisovelluksen kehittämiseen tehty sovelluskehys. Käytännössä se mahdollistaa mobiilisovelluksien kehityksen tavallisten webteknologioiden, HTML, CSS ja JavaScript avulla. Cordova mahdollistaa myös kehittämisen eri alustoille, kuten Android, IOS ja Windows.
(Apache Cordova Documentation)

Sovellus siis kehitetään käytännössä web-sivuna ja Cordova kääntää sen laitteella toimivaksi sovellukseksi. Tämä sovellus on tosin tehty vain Android-ympäristöön, mutta teoriassa kääntäminen IOS versioksi onnistuu myös.

3.2 JavaScript

JavaScript (JS) on kevyt tulkattava ohjelmointikieli. Se tunnetaan parhaiten verkkosivujen kehittämiseen käytettynä ohjelmointikielenä, mutta sitä käytetään myös muissa ympäristöissä, kuten node.js, Apache ja Adobe Acrobat
(Mozilla Developer Network)

3.3 HTML

HTML on merkintäkieli, jonka avulla kerrotaan esimerkiksi selaimelle, kuinka näyttää haluttu verkkosivu. HTML koostuu joukosta elementtejä, joita käytetään sisällön eri osien liittämiseen tai pakkaamiseen, jotta se näyttäisi tietyn tapaiselta tai toimisi tietyllä tavalla. Liitettävät tunnisteet voivat tehdä sanan tai kuvan hyperlinkiksi muualle, kursivoida sanoja tai tehdä fontin isommaksi tai pienemmäksi ja niin edelleen.
(Mozilla Developer Network)

HTML kielessä käytettävät elementit koostuvat aloitus- ja lopetustunnisteista ja sisällöstä. Niillä saattaa mahdollisesti olla attribuutteja, jotka ovat kirjoitettuna aloitustunnisteen sisälle. Esimerkki HTML elementistä: <h1>Otsikko</h1>, jossa <h1> on aloitustagi, teksti ”Otsikko” on sisältö ja </h1> on lopetustagi.

3.4 CSS

Cascading Style Sheets (CSS) on tyyliselainkieli, jota käytetään kuvaamaan HTML- tai XML-dokumenttiin kirjoitettua asiakirjaa. CSS-tiedosto kertoo, miten elementit tulisi asetella ruudulle.
(Mozilla Developer Network)

Kuvan 8 esimerkkikoodissa määritellään sovelluksen päänäkymässä olevia kuvia sisällään pitämän containerin ominaisuudet. Container asetetaan pysymään paikoillaan, kartan yläpuolella ja vasemmassa yläkulmassa. Sen leveydeksi asetetaan 100% ruudun leveydestä ja korkeudeksi 10% ruudun korkeudesta. Lisäksi sille annetaan ominaisuus ”pointer-events: none;” eli mahdollistetaan kartan käyttö sen läpi.

```
46  #logoContainer{
47    position: absolute;
48    z-index: 100;
49    top: 0;
50    left: 0;
51    background: rgba(54, 25, 25, 0);
52    pointer-events: none;
53    width: 100%;
54    height: 10%;
55  }
```

KUVA 8. Containerin ominaisuudet määritellään CSS-tyylikielellä

3.5 Käytetyt JavaScript-kirjastot

3.5.1 Leaflet

Leaflet on avoimen lähdekoodin JavaScript-kirjasto mobiiliystävällisille interaktiivisille karttoille. Kooltaan Leaflet on pieni, vain noin 38 kt, mutta ominaisuuksia siinä on tarpeeksi karttasovelluksen kehittämiseen.

Leaflet on yksinkertainen, suorituskyinen ja helposti käytettävä. Se toimii tehokkaasti kaikissa tärkeissä työpöytä- ja mobiililaitteissa, sitä voidaan laajentaa useilla laajennuksilla, sillä on helppokäyttöinen ja hyvin dokumentoitu API sekä yksinkertainen, helppolukuinen lähdekoodi.

Seuraavassa esimerkkikoodissa luodaan map niminen kartta, annetaan sille asetuksina koordinaatit ja zoomaustaso, joita kartta käyttää aloituspisteenään. Näin luodulla "kartalla" ei ole vielä mitään kuvaa, eli se on tyhjä kartta. Lisätään karttaan OpenStreetMapin karttaruudukko lisäämällä siihen tilelayer -olio.

```
var map = L.map('map').setView([51.505, -0.09], 13);

L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a href="http://osm.org/copyright">Open-
StreetMap</a> contributors'
}).addTo(map);
```

Seuraavaksi lisätään luodulle kartalle merkkipiste ja kirjoitetaan siihen tekstiä ja tuloksena syntyy kuvan 9 karttakuva.

```
L.marker([51.5, -0.09]).addTo(map)
    .bindPopup('A pretty CSS3 popup.<br> Easily customizable.')
    .openPopup();
```

Kuva 9: Leaflet-karttaesimerkki (Leaflet dokumentaatio)

3.5.2 JQuery

JQuery on nopea, pieni ja monipuolinen JavaScript-kirjasto. Se helpottaa HTML-asiakirjan manipulointia, tapahtumien käsittelyä, animaatioiden käyttöä ja Ajax-pyyntöjen lähettämistä.

Esimerkiksi datan pyytäminen palvelimelta HTTP GET -pyynnöllä tapahtuu $.get() methodilla. Syntaksiltaan metodi on seuraavanlainen:

```
$.get(URL,callback);
```

URL-parametriksi asetetaan osoite, johon pyyntö lähetetään. Callback-parametriksi voidaan antaa pyynnön suorittamisen jälkeen kutsuttavan funktion nimi. Kuvan 10 esimerkkikoodissa luodaan ensin painike, jota painamalla lähetetään GET-pyyntö ”demo_test.asp” tiedostolle ja sen suorittamisen jälkeen näytetään tiedoston palauttama data ponnahdusikkunassa.

```
$("button").click(function(){
    $.get("demo_test.asp", function(data, status){
        alert("Data: " + data + "\nStatus: " + status);
    });
});
```

KUVA 10. JQuery esimerkkikoodi (W3schools)

3.5.3 FLOT

Flot on jQueryn kanssa käytetettävä JavaScript-kirjasto, joka mahdollistaa erilaisten kuvaajien piirtämisen helposti. Sovellukseen haluttiin löytää helppo tapa piirtää kuvaajia ja Flot-kirjasto sopi tähän tarkoitukseen hyvin.

4 SOVELLUKSEN TOTEUTUS

4.1 Suunnittelu

Sovellusta suunniteltaessa sille asetettiin vaatimukseksi helppokäyttöisyys, kyky näyttää useita ajoneuvoja kartalla samaan aikaan, mahdollisuus tarkastella ajoneuvon sijaintihistoriaa ja mahdollisuus tarkastella ajoneuvon tämänhetkisiä tietoja, kuten nopeutta ja akun varaustilaa. Lisäksi sovellukseen haluttiin mahdollisuus älykkään kuormatilan hyödyntämiseen, eli jos seurattavissa ajoneuvoissa olisi käytössä järjestelmä, joka pitäisi kirjaa kyydissä olevasta tavarasta, niin sovelluksessa pitäisi olla jonkinlainen lista ajoneuvon kyydissä olevasta tavarasta.

Sovelluksen käytön kannalta oleellisinta on nähdä kohteiden tila juuri nykyisellä hetkellä ja aikaisemmin samana työpäivänä. Sovelluksen ei siis tarvitse näyttää kohteiden historiatietoja yhtä päivää kauemmaksi.

4.2 Kartta

Sovelluksen päänäkymänä toimiva kartta toimii alustana, jonka päälle kaikki muut oliot lisätään. Kuvan 11 lähdekoodissa, luodaan leaflet-kirjaston metodeilla karttaolio, joka hakee OpenStreetMap:lta tarvitsemansa karttaruudut.

```
this.eTruckMap = L.map('mapId', {zoomControl:false}); //the map

L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', { //get tiles for the map from OpenStreetMap
  attribution: '&copy; <a href="http://openstreetmap.org">OpenStreetMap</a> contributors'
}).addTo(this.eTruckMap);
```

KUVA 11. Kartan lisääminen sovellukseen

4.2.1 Objektien piirtäminen kartalle

Kartalle voi helposti lisätä kohteita Leafletin .addTo-metodilla. Kuva 12:ssa näkyvässä koodissa annetaan marker-oliolle koordinaatit, tekstiä sen lisätietovalikkoon ja lopuksi lisätään se karttaan .addTo-metodilla.

```
this.marker.setLatLng([this.latitude, this.longitude]);
this.text.innerHTML = "<b>" + this.name + "</b><br>speed: " + this.speed +
    " km/h, power: " + this.power + " kW <br>battery state of charge: " +
    this.soc + " % <br>distance today: " + this.distance + " km<br>";

this.marker.addTo(this.fleet.eTruckMap);
```

KUVA 12. Objektin lisääminen kartalle

4.2.2 Reitin piirtäminen kartalle

Leafletissä on myös omat työkalunsa reitin piirtämiseen kartalle. Tässä sovelluksessa reitin piirtäminen on toteutettu luomalla L.polyline-olio ja lisäämällä tämä tyhjä olio kartalle. Myöhemmin oliolle annetaan koordinaattipareja sisältävä taulukko ja taulukon määrittämä reitti piirtyy kartalle.

Reitin piirtämistä varten päädyttiin käyttämään 10 sekunnin välein tallennettua GPS-sijaintitietoa, vaikka tarjolla olisi ollut jopa 100ms välein tallennettuja GPS-koordinaatteja. Reitin piirtämisessä ja tietojen hakemisessa tosin kesti melko kauan, kun pisteitä oli paljon. Valitulla tarkkuudella piirretyn reitin tarkkuus vaikuttaa hyvältä suorituskykyä häiritsemättä ja suurimmaksi osaksi aikaa auto näyttäisi kartan mukaan pysyvän tiellä.

4.2.3 Pysähdysten havaitseminen

Pysähdysten havaitsemiseen käytettiin alun perin WRM API:lta saatavaa GPS-signaalin perusteella laskettua nopeutta, mutta pian huomattiin, että vaikka ajoneuvo on paikallaan niin GPS:n häiriöt aiheuttavat nopeustietoon liikaa häiriötä ja vaikka ajoneuvo oli täysin paikallaan sen nopeudeksi saattoi tulla vaikka 4 km/h. Alkuperäinen pysähdysten havainnointi toimi siten, että käytiin päivän nopeusdata silmukassa läpi ja piirrettiin pysähdysmerkki kun nopeus oli ollut alle 5km/h kahden minuutin ajan. Algoritmi toimi suhteellisen hyvin, mutta esimerkiksi parkkipaikoilla ja lastausalueilla tai tunneleissa se saattoi piirtää pysähdyksiä vääriin paikkoihin.

Myöhemmin päädyttiin käyttämään nopeustiedon sijaan “Vehicle Motion” tietoa, joka tulee WRM-API:n kautta ajoneuvojen ajotietokoneelta ja on silloin yksi, kun ajoneuvo on ajossa ja nolla ajoneuvon ollessa paikallaan. Pysähdys piirretään, kun vehicle motion on ollut kaksi minuuttia nollatilassa.

4.3 Energiankulutuksen laskeminen

Kuorma-auton ajotietokoneelta saadaan tietona akuston hetkellinen jännite ja virta, näistä tiedoista voidaan laskea hetkellinen teho ja tehosta voidaan laskea energiankulutus. Jännitteen ja virran mittanäytteet on otettu 1 sekunnin välein ja näistä lasketaan ensin yhtälön (1) mukaisesti sekunnin ajan käytetty teho ja siitä sitten sekunnin aikana kulutettu energia yhtälön (2) mukaan.

$$P = UI, \tag{1}$$

$$E = \frac{P}{\Delta t}\ , \tag{2}$$

Mittanäytteiden näytteenottoväliksi on valittu 1 sekunti, sillä suuremmalla näytteenottotaajuudella näytteitä tulee liian paljon, ja energiankulutuksen laskeminen hidastuu huomattavasti. Lopputuloksen kannalta ei ole huomattavaa väliä, otetaanko mittanäytteitä 1000ms vai 100ms välein.

4.4 Kuvaajien piirtäminen

Kuvaajien piirtämiseen käytetään Flot-kirjastoa, jossa on valmiit funktiot kuvaajien piirtämiseen. Kuvaajaa piirtäessä data pitää olla tallennettuna seuraavassa muodossa:

```
[[x1, y1], [x2, y2], ... ]
```

Jossa x-koordinaatit ovat tehoarvoja ja y-koordinaatit tehoarvojen aikaleimoja.

Kuvan 13 esimerkkikoodissa luodaan #truckPower niminen kuvaaja ja annetaan sille dataksi taulukko, joka sisältää teho-aikaleima -pareja. Y-akselin nimeksi annetaan ”power (kW)” ja x-akselin nimeksi ”time (h)”. Lisäksi taulukolle annetaan muitakin asetuksia, kuten aikaformaatti.

```
69        $.plot("#truckPower", [{ label: "power", data: this.power }], {
70          xaxis: { axisLabel: "Y = power (kW)   X = time (h)",
71          axisLabelPadding: 15,
72          mode: "time", timeformat: "%H", minTickSize: [2, "hour"] },
73          yaxis: { minTickSize: 1 },
74          grid: { labelMargin: 20 }
75        });
```

KUVA 13. Objektin lisääminen kartalle

4.5 Rajapinnat

Sovellus hakee käyttämänsä tiedot erilaisten rajapintojen kautta. Tietoa haetaan lähettämällä pyyntöjä palvelimelle, joka sitten palauttaa pyydetyt tiedot JSON-muodossa.

4.5.1 WRM API

Sovellus hakee kaikki ajoneuvoihin liittyvät tiedot WRM eli Wapice Remote Management API:n kautta.

Seurattaviin kohteisiin, eli sähköautoihin ja latauspisteisiin on asennettu laitteet, jotka keräävät kohteiden CAN- tai Modbus-väylästä saatavaa tietoa ja välittää kerätyn tiedon WRM-palvelimelle. Laitteissa on myös GPS-paikannin, jolla ne keräävät sijaintitietoa. Kerätty tieto on saatavilla WRM API:n kautta.

Tiedot mitä WRM API:n kautta saadaan, ovat ajoneuvon nopeus, akun jännite, akussa kulkeva virta, akusta otettava teho, liiketila ja sijainti. Sovellus hakee tämänhetkiset tiedot 10 sekunnin välein ja sijaintihistorian tai energiankulutuksen laskemiseen tarvittavat tiedot silloin käyttäjän niitä pyytäessä.

4.5.2 Google Maps Geocoding API

Google Maps Geocoding API käytetään pysähdysten yhteydessä hakemaan pysähdyksen koordinaatteja lähinnä oleva osoite ja kuormatilassa osoite, jossa auto oli, kun kuormaa on lastattu tai purettu.

Google Maps Geocoding -sovellusliittymää käytetään HTTP-käyttöliittymän kautta. Sille välitetään parametreina halutun osoitteen koordinaatit ja sovelluksen API-käyttöavain ja vastauksena se palauttaa annetuista koordinaateista JSON-muotoisena datana tietoja, kuten esimerkiksi koordinaatteja lähinnä olevan osoitteen, maan, maakunnan ja kaupungin. Esimerkkipyyntö mitä API:lle lähetetään näyttää tältä:

```
https://maps.googleapis.com/maps/api/geocode/json?latlng=61.50064999999999,23.7589716
6666667&key=INSERT_API_KEY_HERE&format=json
```

API:n ilmaisversiossa on rajoitettu päivässä lähettävien pyyntöjen määrä 2500, joka riittää sovelluksen tarpeisiin enemmän kuin hyvin.

POHDINTA

Projektin tavoitteena oli toteuttaa helppokäyttöinen mobiilisovellus sähköajoneuvojen etäseurantaan ja tuloksena saatu sovellus vastaa asetettuja tavoitteita. Sovellus on osoittautunut helppokäyttöiseksi, toimivaksi ja hyödylliseksi. Se kertoo yhdellä vilkaisulla seuratun ajoneuvon sijainnin, joka on paljon nopeampaa, kuin esimerkiksi kuskille soittaminen ja sijainnin kysyminen. Lisäksi sovelluksen laskemia tunnuslukuja ja piirtämiä kuvaajia on käytetty ongelmatilanteiden selvittämiseen ja havaitsemiseen.

Sovelluskehityksessä käytetty Apache Cordova -sovelluskehys osoittautui järkeväksi ja toimivaksi työkaluksi mobiilisovelluksen kehittämiseen. Muut käytetyt ohjelmat Atom-tekstieditori ja git-versionhallintajärjestelmä sopivat myös tarkoituksiinsa erinomaisesti.

Vaikka toteutettu sovellus toteuttaa kaikki sille asetetut vaatimukset, on siinä silti mahdollisuuksia kehitykseen. Esimerkiksi Flot-kirjaston avulla piirrettyjen kuvaajien akselit eivät aina piirry tyylikkäästi. Ongelman ratkaisemiseksi koko kirjasto pitäisi todennäköisesti korvata jollain toisella vaihtoehdolla. Lisäksi ajoneuvojen lisääminen sovellukseen on tällä hetkellä vaikeaa ja lisäämistä varten voisi kehittää järkevän järjestelmän.

Sovelluksen kehittäjille projekti oli erittäin mielenkiintoinen ja opettavainen. Kummallakaan kehittäjistä ei ollut juurikaan aikaisempaa kokemusta mobiilisovelluksen kehittämisestä ja sovelluksen tekeminen oli jatkuvaa oppimista ja uuden löytämistä. Projektiin saatiin viikoittaista palautetta ja palautteen perusteella tehtiin jatkuvasti pieniä tai isoja muutoksia sovellukseen.

Projektin parissa työskentely sujui kaikkien projektin jäsenten mielestä sujuvasti ja aikaa projektin kehitykseen oli sopivasti. Aikataulut tuntuivat projektia tehdessä tiukoilta, mutta ne osoittautuivat realistisiksi ja sovellus valmistui sovitussa aikataulussa.

LÄHTEET

Mozilla Developer Network 2017: HTML basics. Luettu 21.11.17
https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics

Mozilla Developer Network 2017: JavaScript. Luettu 21.11.2017
https://developer.mozilla.org/en-US/docs/Web/JavaScript

Apache Cordova Documentation. Luettu 21.11.2017
https://cordova.apache.org/docs/en/latest/guide/overview/index.html

Leaflet dokumentaatio. Luettu 21.11.2017.
http://leafletjs.com/reference-1.2.0.html

W3schools. AJAX get() and post() Methods. Luettu 4.12.2017
https://www.w3schools.com/jquery/jquery_ajax_get_post.asp