



Vilho Voutilainen

GeoJSON-pohjainen kartta satamaterminaaliin

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

5.5.2023

Tiivistelmä

Tekijä: Vilho Voutilainen
Otsikko: GeoJSON-pohjainen kartta satamaterminaaliin
Sivumäärä: 32 sivua
Aika: 5.5.2023

Tutkinto: Insinööri (AMK)
Tutkinto-ohjelma: Tieto- ja viestintätekniikka
Ammatillinen pääaine: Mediatekniikka
Ohjaaja: Lehtori Toni Spännäri

Insinöörityönä tehtiin kuljetusyrityksen terminaalin merikonttien paikoitusjärjestelmän kehittämistä ja parantamista sekä terminaalisovelluksen karttanäkymän uudistamista. Lisäksi työssä kehitettiin tarvittavat työkalut tämän tavoitteen saavuttamiseksi.

Insinöörityön tavoitteena oli luoda merikonttien paikoittamiseen entistä toimivampi ja tarkempi karttanäkymä, joka vastaa reaaliajassa terminaalin konttien tilannetta. Tämä uudistettu karttanäkymä mahdollistaa paitsi tehokkaamman työskentelyn, myös paremman valmiuden siirtyä kohti terminaalin paikoituksen automatisointia tulevaisuudessa.

Satamaterminaalissa on erittäin tärkeää pystyä paikoittamaan ja seuraamaan suurta määrää merikontteja tehokkaasti ja tarkasti. Terminaalissa on parhaimmillaan yli tuhat merikonttia kerrallaan, mikä asettaa haasteita paikoituksen ja logistiikan hallintaan. Tämän vuoksi konttikurottajan kuljettajalle on kehitetty apuvälineeksi visuaalinen karttanäkymä, joka helpottaa työtä ja auttaa hahmottamaan terminaalin tilanteen.

Työn lopputuloksena saatiin luotua kehittynyt karttanäkymä, joka vastaa täysin terminaalin reaaliaikaista tilannetta ja tarjoaa kattavat tiedot konttien sijainneista. Tämä paranneltu järjestelmä asettaa vahvan pohjan terminaalin paikoituksen automatisoinnin kehittämiseksi ja yrityksen toiminnan tehostamiseksi entisestään.

Avainsanat: GeoJSON, GPS, satelliittipaikannus, satama, terminaali

Abstract

Author:	Vilho Voutilainen
Title:	GeoJSON-based map for port terminal
Number of Pages:	32 pages
Date:	5 May 2023
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Media Technology
Supervisor:	Toni Spännäri, Senior Lecturer

In this engineering thesis we developed and improved the container positioning system at a logistics terminal, as well as renewed the terminal application's map view. Additionally, we developed the necessary tools to achieve this.

In a port terminal, it is essential to efficiently and accurately locate and track a large number of shipping containers. At its peak, the terminal can hold over a thousand containers at once, which poses challenges for position management and logistics. For this reason, a visual map view has been developed as an aid for the reach stacker operator to facilitate their work and help them understand the terminal's situation.

The objective of the engineering thesis is to create a more functional and precise map view for container positioning, which reflects the real-time status of the containers at the terminal. This updated map view not only allows for more efficient work but also provides a better preparedness to move towards the automation of container positioning in the future.

As a final result, an advanced map view was created that fully reflects the real-time situation at the terminal and provides comprehensive information on container locations. This improved system lays a strong foundation for the development of container positioning automation and further enhances the efficiency of the company's operations.

Keywords: GeoJSON, GPS, Satellite Positioning, Port, Terminal

Sisällys

Lyhenteet

1	Johdanto	1
2	Satelliittipaikannus	3
2.1	GPS-paikannusjärjestelmä	3
2.2	Reaaliaikainen kinemaattinen mittaus	5
2.3	GeoJSON-standardi	7
3	Moderni web-kehitys	10
3.1	ReactJS-kirjasto	10
3.2	Responsiivisuus	12
3.3	Saavutettavuus	13
4	Terminaalisovelluksen kartta	14
4.1	Paikoittaminen terminalissa	14
4.2	Vanha kartta	16
4.3	Kartan päivittäminen	17
4.3.1	Kartan pohjana GeoJSON	17
4.3.2	Koordinaattien mittaus	18
4.3.3	Pistemittaustyökalu	19
4.3.4	GeoJSON-työkalu	21
4.3.5	Paikkatiedon versionhallinta	26
4.3.6	Sovelluksen karttanäkymä	26
4.4	Kartan päivitys	30
4.5	Uusi kartta	31
5	Yhteenveto	32
	Lähteet	33

Lyhenteet

GeoJSON:	Geo Javascript Object notation on paikoitustietoa sisältävä Javascript-objektirakenne.
GNSS:	Global Navigation Satellite System tarkoittaa yleisesti kaikkia satelliittipaikannusjärjestelmiä.
GPS:	Global Positioning System on Yhdysvaltain armeijan kehittämä satelliittipaikotusjärjestelmä.
RTK:	Real Time Kinematic. Tunnetussa pisteessä sijaitseva referenssi-asema lähettää korjaustietoa RTK-vastaanottimelle senttimetriluokan tarkkuuden paikannuksen saavuttamiseksi.
IETF:	Internet Engineering Task Force on vapaaehtoinen, kansainvälinen järjestö, joka kehittää ja ylläpitää internetin teknisiä standardeja.
SVG:	Scalable vector graphics on skaalautuva vektorigrafiikkaformaatti.
CSS:	Cascading Style Sheet on merkintäkieli verkkosivujen ulkoasun määrittämiseen.
HTML	Hype Text Markup Language on merkintäkieli, jota käytetään verkkosivujen rakenteen ja sisällön luomiseen.
JSX	Javascript XML on Reactin käyttämä Javascript-laajennus, joka mahdollistaa HTML-rakenteen lisäämisen Javascript-tiedostoon.
DOM	Document Object Model on ohjelmointirajapinta, joka esittää HTML- ja XML-dokumentit puurakenteena ja mahdollistaa niiden muokkaamisen.
ARIA	Accessible Rich Internet Application on saavutettavuusteknologia.

1 Johdanto

Bertschi AG on sveitsiläisperusteinen logistiikkakonserni, joka on keskittynyt kemikaali- ja bulkkitavaran kuljetuksiin. Se on Euroopan markkinajohtaja kemikaalien intermodaalisessa kuljetuksessa, eli kuljetuksessa, joka tapahtuu niin laiva-, raide- kuin tieliikennettä käyttäen. Tämä kaikki on mahdollista intermodaalisessa kuljetuksessa käytettävien kooltaan standardoitujen merikonttien avulla.

Bertschillä on logistiikkaterminalleja ympäri maailmaa, jopa 30 eri maassa, ja niiden välillä yrityksen merikontit liikkuvat maailmanlaajuisten raide-, laiva- ja tiereittien välillä. Kokonaisuudessaan konsernilla on kemikaali- ja bulkkikontit yhteenlaskettuna lähes 50 000 konttia. [1.]

Bertschi Finland Oy on Bertschi AG:n tytäryhtiö, joka sijaitsee Vuosaaren satamassa Helsingissä. Bertschi Finlandin terminaalissa on ajankohdasta riippuen noin viidestäsadasta yli tuhanteen merikonttia kerrallaan. Niiden säilytys noin 10 hehtaarin kokoisessa terminaalissa vaatii tarkkaa paikoitusta, jotta kontit löytyvät isosta massasta.

Tässä opinnäytetyössä tutustutaan Bertschi Finlandin terminaalien konttien paikoitukseen ja luodaan GPS-paikkatiedon avulla tarkka GeoJSON-pohjainen kartta konttien sijainneista terminaalissa. Olemassa olevaan toiminnanohjausjärjestelmään luodaan tarpeelliset lisätyökalut kaiken tämän toteuttamiseksi.

Työ toteutetaan työntekijänä yritykselle, jossa olen ollut jo useamman vuoden töissä. Ohjelmistokehitys toteutetaan käyttäen React- ja Meteor-Javascript-kirjastoja. Tietokantana sovelluksessa toimii MongoDB-objektitietokanta. Paikannukseen käytetään ArduSimple-yrityksen valmistamia RTK-mittalaitteita.

Työn tavoitteena on luoda tarkkaan sijaintitietoon perustuva GeoJSON-pohjainen kartta, joka mahdollistaa konttien paikoituksen tehokkaasti ja tarkasti. Kartan tulee olla helposti päivitettävä, jotta sitä voidaan muokata vaivattomasti terminaalien järjestyksen muuttuessa.

Kartan ulkoasuun ei tule tehdä muutoksia, vaan se tulee säilyttää niin lähellä alkuperäistä kuin mahdollista, jotta muutos käyttäjälle säilyisi mahdollisimman pienenä.

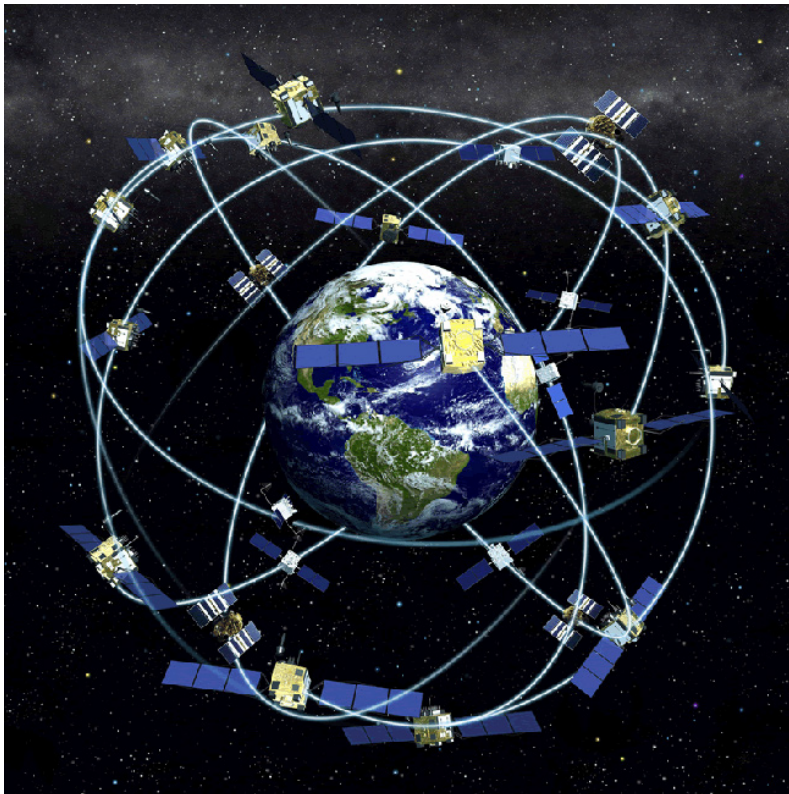
2 Satelliittipaikannus

Uuden, todelliseen sijaintitietoon perustuvan GeoJSON-kartan luomiseksi insinööriyössä käytettiin hyväksi satelliittipaikannusta terminaalin alueiden sijaintien mittaamiseksi. Mahdollisimman tarkan mittaustuloksen saavuttamiseksi työssä käytettiin apuna RTK-mittaustekniikkaa.

2.1 GPS-paikannusjärjestelmä

Satelliittipaikannus on mullistanut tavat navigoida, mitata sijainteja ja tehdä karttoja. Se perustuu Maata kiertävien satelliittien lähettämän signaalin käyttöön, jonka avulla satelliittivastaanotin voi laskea laitteen sijainnin. Tunnetuin ja laajimmin käytössä olevan satelliittipaikannusjärjestelmä on Yhdysvaltain armeijan 1970-luvulla kehittämä globaali paikannusjärjestelmä, GPS. [2.]

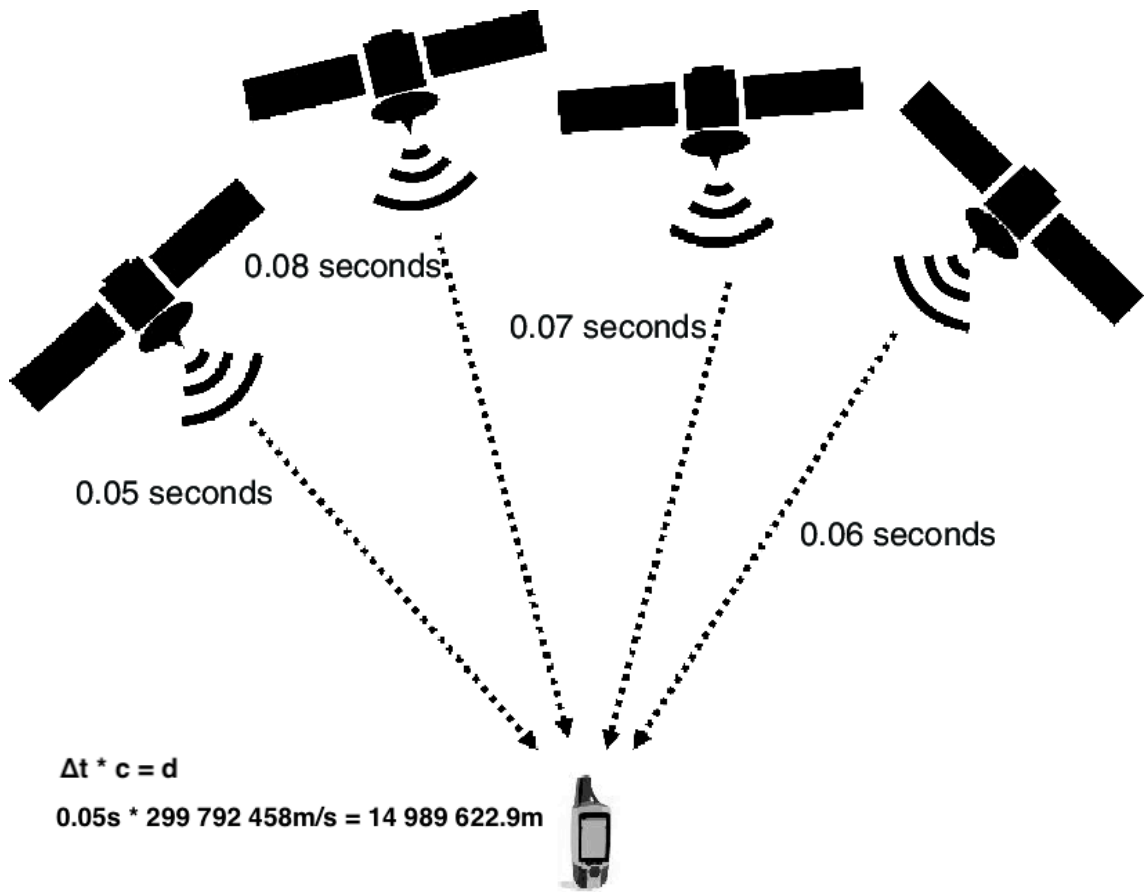
GPS-järjestelmä koostuu nykyisin 31 satelliitin verkosta, jonka satelliitit kiertävät Maata noin 20 000 kilometrin korkeudessa kuudella eri kiertoradalla (kuva 1).



Kuva 1. GPS-satelliitit Maata kiertävällä radalla [3].

Jokainen satelliitti sisältää atomikellon, jonka avulla satelliitti pystyy laskemaan erittäin tarkan aikaleiman. Satelliitti lähettää tätä aikaleimaa ja omaa sijaintiaan radiosignaalina maahan, jolloin maassa olevat satelliittivastaanottimet pystyvät laskemaan signaalin lähetysaikaleiman ja oman vastaanottoaikaleiman avulla erotuksen eli ajan, joka signaalilla kesti kulkea satelliitista vastaanottimeen. Tästä saadaan laskettua satelliitin etäisyys kerrotaessa erotus signaalin nopeudella eli valonnopeudella (kuva 2). [4.]

Normaali trilateraatio eli kolmiomittaus vaatii vähintään 3 tunnettua pistettä ja etäisyyttä niihin paikkatiedon laskemiseksi. Vastaanottimien kello ei ole yhtä tarkka kuin satelliittien atomikello, mikä aiheuttaa virhettä GPS-paikan määrittämisessä. Satelliittipaikannus vaatii täten, että vastaanotin on yhteydessä vähintään neljään satelliittiin samanaikaisesti, jolloin kellojen mittavirhettä voidaan korjata ja vastaanotin voi laskea oman sijaintinsa. [5; 6.]



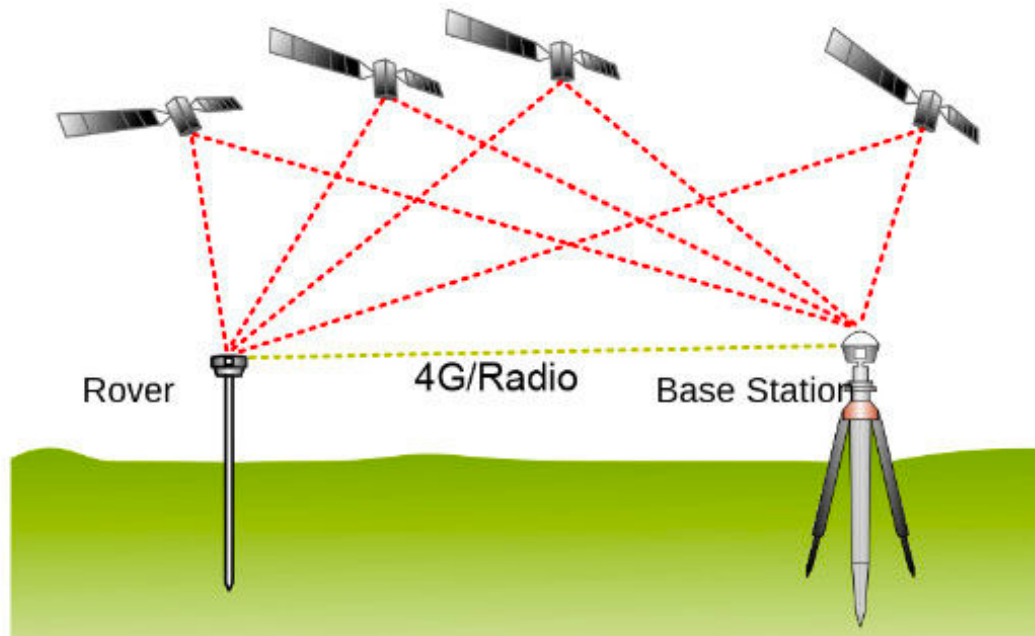
Kuva 2. GPS-vastaanotin ja etäisyyden määrittäminen [lähteen 6 tietojen pohjalta].

Epätarkkuutta GPS-mittaukseen tuo myös signaalin kulkeminen maapallon ilmakehän ionosfääriin ja toposfääriin lävitse, jossa sähkömagneettiset häiriöt heijentävät ja taivuttavat signaalia. Myös signaalin heijastuminen maan pinnalla olevista kohteista, kuten rakennuksista, pidentää signaalin kulkemaa matkaa ja aiheuttaa epätarkkuutta sijaintia laskiessa. Normaaliin sääolosuhteiden kanssa GPS:n tarkkuus on keskimäärin noin 7 metrin luokkaa. [4.]

2.2 Reaaliaikainen kinemaattinen mittaus

RTK eli reaaliaikainen kinemaattinen mittaus on mittaustapa, jossa tavallisen GNSS-signaalin tarkkuutta parannetaan käyttämällä toista GNSS-vastaanotinta luomaan korjausdataa parantamaan tarkkuutta (kuva 3). Tätä paikannustekniikkaa käytetään esimerkiksi maanmittauksessa, rakentamisessa ja muissa

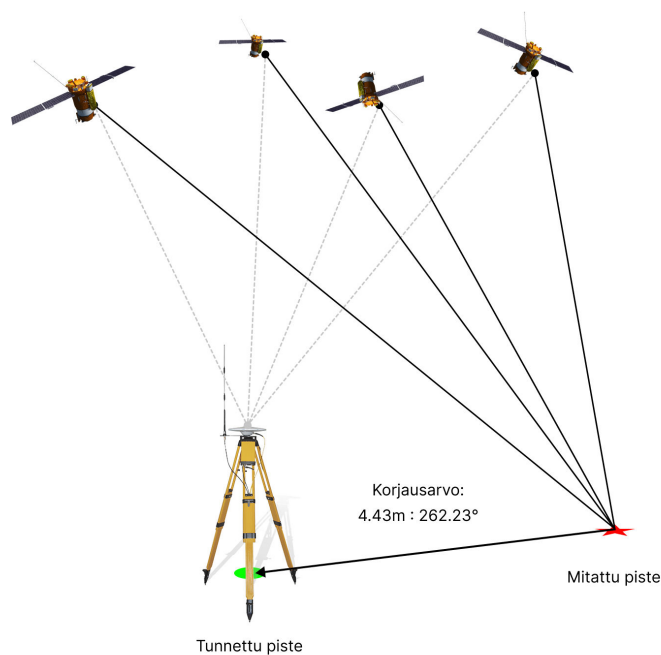
sovellutuksissa, joissa tavallisen satelliittipaikannuksen metrien tarkkuus ei ole riittävä. [7.]



Kuva 3. RTK-tukiasema ja liikuteltava vastaanotin [8].

RTK-mittausjärjestelmä koostuu kahdesta osasta: tukiasemasta (Base Station) ja liikuteltavasta vastaanottimesta (Rover). Tukiasema on tunnettuun sijaintiin sijoitettu GNSS-vastaanotin, joka toimii järjestelmän vertailukohtana. Rover on liikuteltava vastaanotin, jota käytetään haluttujen mittapisteiden paikantamiseen. Molemmat laitteet vastaanottavat signaaleja samoista satelliiteista, mikä mahdollistaa ilmakehän vaikutuksen kompensoimisen molemmille laitteille.

Kun sekä tukiasema että Rover vastaanottavat signaaleja samoista satelliiteista, niiden signaaleihin kohdistuvat samankaltaiset ilmakehän aiheuttamat häiriöt. Koska tukiasema sijaitsee tunnetussa pisteessä, se pystyy laskemaan mitatun signaalin etäisyyden ja suunnan tunnetusta pisteestä (kuva 4) ja saa näin korjausarvon, jolla ilmakehän luomaa häiriötä voidaan kompensoida molemmissa signaaleissa. [9.]



Kuva 4. RTK-korjausdatan laskeminen [lähteen 9 tietojen pohjalta].

Tukiasema lähettää korjausarvon Roverille joko radioverkon tai internetin välityksellä. Kun Rover vastaanottaa korjausarvon, se soveltaa sen omiin GNSS-signaaleihinsa. Tämä menetelmä poistaa suuren osan ilmakehän aiheuttamasta epätarkkuudesta ja mahdollistaa paikannuksen settimetrimin tarkkuudella.

RTK-mittauksen tehokkuus riippuu laitteiden välisestä etäisyydestä. Yleensä tukiaseman ja Roverin maksimaalinen toimiva etäisyys on 20–50 kilometriä, sillä pidemmillä etäisyyksillä ilmakehän vaikutukset eroavat merkittävästi laitteiden välillä ja heikentävät tarkkuutta. [10.]

2.3 GeoJSON-standardi

GeoJSON on avoin standardimuotoinen tiedostomuoto, jota käytetään paikkatietoinformaation esittämiseen. Se perustuu JSONiin ja on laajasti käytössä web-pohjaisissa karttasovelluksissa. [11.]

Standardin ensimmäinen versio julkaistiin vuonna 2008 muutamien kehittäjien toimesta [12]. Vuonna 2015 alan standardisoimisjärjestö IETF loi työryhmän, joka jatkoi GeoJSON standardin kehitystä eteenpäin. Elokuussa 2018 IETF

julkaisi GeoJSONista uuden version, joka korvasi vanhan, vuoden 2008 standardin. [13.]

GeoJSONiin voidaan tallentaa geografisia kohteita, geokomponentteja, kuten pisteitä, viivoja ja monikulmioita, sekä tallentaa niihin liittyviä attribuutteja standardoidussa ja koneluettavassa muodossa (esimerkkikoodi 1). Se on suunniteltu ihmisen luettavaksi ja helposti jäsennettäväksi, mikä tekee siitä tehokkaan datamuodon web-pohjaisiin karttasovelluksiin. GeoJSON tiedostomuotona on samanlaista kuin JSON, joten sen luominen ja muokkaaminen onnistuu millä vain tekstieditorilla.

GeoJSON on yksi JSON-objekti, jonka tyyppi on määritelty joko FeatureCollection eli lista yksittäisiä geokomponentteja tai Feature, jolloin se sisältää vai yhden geokomponentin. Nämä geokomponentit sisältävät tiedon komponentin tyypistä, sen attribuutit ja geometriatiedon sisältäen komponentin koordinaatit tai koordinaattilistan. Koordinaattilista on lista esimerkiksi monikulmion kulmapisteiden koordinaateista. [14.]

```

{ "type": "FeatureCollection",
  "features": [
    { "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [102.0, 0.5]
      },
      "properties": {
        "prop0": "value0"
      }
    },
    { "type": "Feature",
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [102.0, 0.0], [103.0, 1.0], [104.0, 0.0], [105.0, 1.0]
        ]
      },
      "properties": {
        "prop0": "value0",
      }
    },
    { "type": "Feature",
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0],
            [100.0, 1.0], [100.0, 0.0] ]
        ]
      },
      "properties": {
        "prop0": "value0",
        "prop1": {"this": "that"}
      }
    }
  ]
}

```

Esimerkkikoodi 1. Esimerkki GeoJSON-datasta [11].

3 Moderni web-kehitys

Modernilla web-kehityksellä tarkoitetaan verkkosivujen ja -sovellusten kehitystä käyttäen uusimpia tekniikoita ja menetelmiä. Ne koostuvat monista erilaisista teknologioista, kirjastoista ja ohjelmistokehyksistä, jotka on suunniteltu helpottamaan web-sovellusten kehitystä ja parantamaan niiden suorituskykyä.

Kirjastot ja ohjelmistokehykset ovat valmiita ratkaisuja, jotka tarjoavat kehittäjille useita toiminnallisuuksia ja ominaisuuksia. ReactJS, Angular ja Vue ovat esimerkkejä suosituista ohjelmistokehyksistä ja kirjastoista, jotka on suunniteltu helpottamaan ja nopeuttamaan kehittäjien työtä. Niiden avulla kehittäjät voivat keskittyä sovelluksen toiminnallisuuksiin ja käyttöliittymän suunnitteluun, ilman että heidän tarvitsee luoda sovelluksen perustoiminnallisuuksia itse. [15.]

3.1 ReactJS-kirjasto

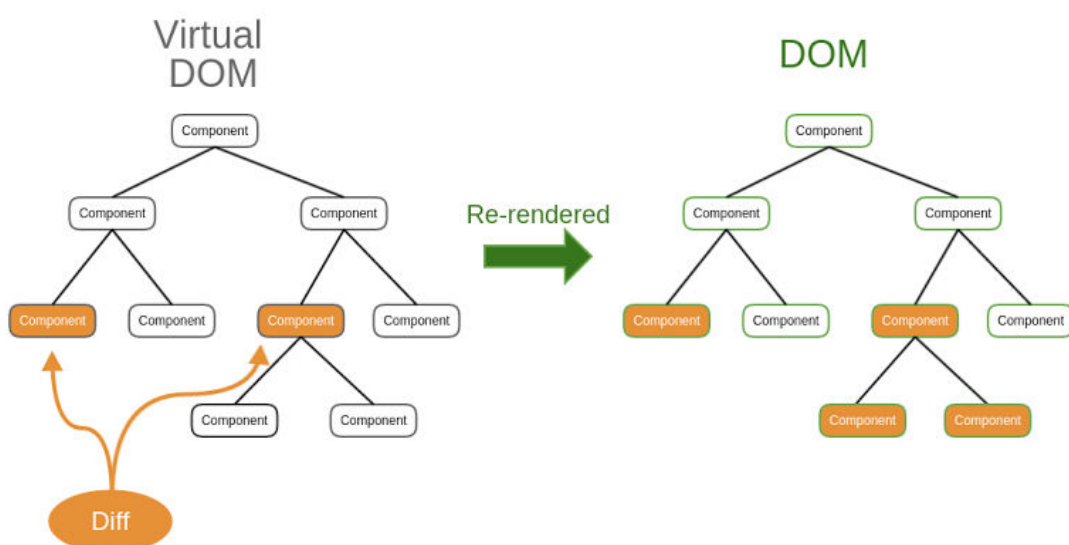
ReactJS on Facebookin kehittämä avoimen lähdekoodin Javascript-kirjasto, joka on suunniteltu erityisesti käyttöliittymien luomiseen ja hallintaan. Kehitystyö alkoi vuonna 2013, ja siitä lähtien React on noussut yhdeksi suosituimmista Javascript-kirjastoista modernissa web-kehityksessä [16]. Sen keskeisenä tavoitteena on tarjota kehittäjille tehokas ja helppokäyttöinen työkalu dynaamisten käyttöliittymien luomiseen.

Reactin perustana on komponenttipohjainen rakenne, joka mahdollistaa sovelluksen jakamisen erillisiksi, toisistaan riippumattomiksi osiksi eli komponenteiksi. Nämä yksittäiset komponentit voidaan suunnitella uudelleenkäytettäviksi sovelluksen eri osissa, mikä vähentää toistuvan koodin määrää ja tehostaa kehitystä. React-komponentit sisältävät sekä visuaalisen rakenteen (HTML) että toiminnallisuuden (Javascript), mikä tekee niistä modulaarisia ja helposti ylläpidettäviä. [17.]

React hyödyntää JSX-kieltä (Javascript XML), joka on Javascriptin laajennus. JSX mahdollistaa HTML:n kaltaisen koodin kirjoittamisen suoraan Javascript-

tiedostoihin, mikä tekee koodista helpommin luettavaa ja ymmärrettävää. React muuntaa JSX-koodin takaisin puhtaaksi Javascriptiksi ennen sen suorittamista selaimessa. Vaikka JSX:n käyttö ei ole pakollista Reactin kanssa, se on suositeltu tapa luoda komponentteja ja käyttöliittymiä.

React käyttää virtuaalista DOM-ohjelmointirajapintaa, joka on kevyt Javascript-objekti, joka toimii oikean DOMin abstraktiona. Virtuaalisen DOMin avulla React seuraa komponenttien tilan muutoksia ja päivittää vain ne osat, jotka ovat muuttuneet, sen sijaan, että koko verkkosivu päivitetäisiin, kuten kuvan 5 kaavio esittää. Tämä optimoi sovelluksen suorituskykyä ja parantaa käyttäjäkokemusta. React-nimi juontaaakin juurensa kirjaston perusajatuksista: reagoida tilan muutoksiin. [17.]



Kuva 5. Virtuaalisen DOM-ohjelmointirajapinnan toiminta [18].

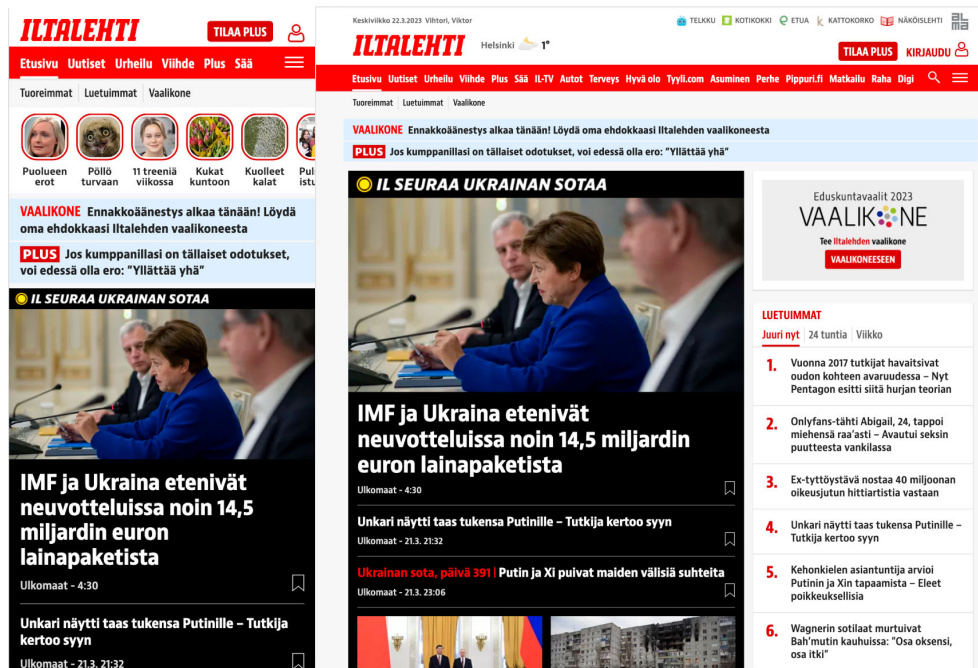
3.2 Responsiivisuus

Responsiivisuus eli verkkosivujen suunnittelu ja toteutus siten, että ne mukautuvat kaikenkokoisille näytöille, on tärkeä osa modernia web-kehitystä. Tämä on tärkeää, sillä käyttäjät käyttävät yhä enemmän verkkosivuja ja web-sovelluksia mobiililaitteilla, jolloin he odottavat niiden toimivan samalla tavalla kuin tietokoneellakin. Vuonna 2022 tehdyn tutkimuksen mukaan 60 % kaikesta internetliikenteestä tuli mobiililaitteilta [19].

Käyttöliittymän tulee mukautua erilaisiin näyttökoko- ja laitetilanteisiin esimerkiksi käyttämällä skaalautuvia fontteja ja kuvia. Sovelluksen tai verkkosivun elementtien asettelua voidaan myös automaattisesti muokata näytön koon muuttuessa. [20.]

Kun käyttäjä lataa verkkosivun, saa verkkosivu laitteelta parametreina laitteen näytön korkeuden ja leveyden pikseleinä. Näiden kokotietojen avulla saadaan selville, onko kyseessä mobiililaitte vai tietokone. Sivustoa kehitettäessä on määriteltäviä tietyt pikselirajat, jotka määrittelevät, miten sivusto esitetään minkä kinkokoisella näytöllä, jotta käyttäjäkokemus säilyisi samana niin mobiilissa kuin tietokoneellakin.

Esimerkkinä responsiivisuudesta näkyy Iltalehden verkkosivut, jossa tietokoneversion kaksi saraketta muuttuvat yhdeksi, kun näytön leveys alittaa 640 pikseliä (kuva 6). Tietokoneversion Luetuimmat-lista siirtyy mobiilissa sivuston ylälaitaan horisontaalisesti selattavaksi valikoksi.



Kuva 6. Ilta-lehden responsiivinen verkkosivu mobiililaitteella ja tietokoneella [21].

3.3 Saavutettavuus

Verkkosivusto tai -sovellus tulee suunnitella ja toteuttaa siten, että se on kaikkien käyttäjien käytettävissä, mukaan lukien niiden, joilla on erilaisia toimintarajoitteita. Kun saavutettavuus otetaan huomioon verkkosivustolla, se mahdollistaa sen käytön myös niille, joilla on esimerkiksi näkö- tai kuulovamma tai jotka käyttävät avustavia teknologioita, kuten ruudunlukijoita. [22.]

Saavutettavuus huomioiden verkkosivu tai -sovellus tulee toteuttaa siten, että se on yhteensopiva näiden avustavien teknologioiden kanssa. Sovelluksen teknisessä toteutuksessa tämä tulee ottaa huomioon lisäämällä sovelluksen eri elementeille kuvaavia ARIA-tunnisteita, jotta esimerkiksi ruudunlukulaite osaa käsitellä elementtiä oikein. [23.]

4 Terminaalisovelluksen kartta

Jotta insinööryönä tehtävä terminaalisovelluksen kartta palvelisi parhaiten käyttäjiänsä, on sen vastattava terminaalin konttien tilannetta tarkasti. Terminaalin jokaisen kontin tulee siis näkyä kartalla juuri siinä sijainnissa, missä se oikeasti sijaitsee.

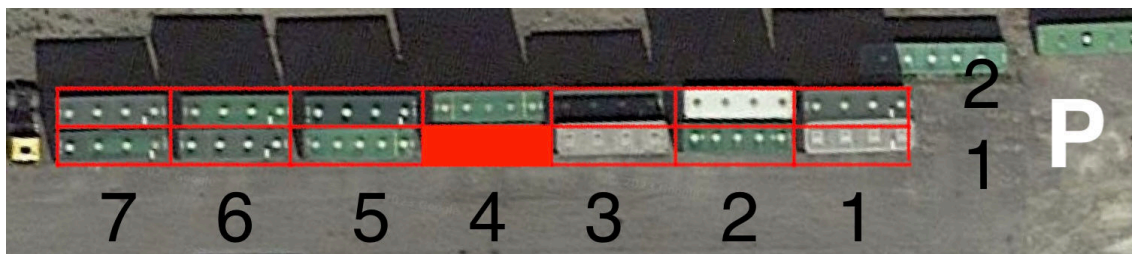
4.1 Paikoittaminen terminalissa

Terminaalin piha on jaettu alueisiin, joita punaiset viivat kuvassa 7 esittävät. Jokainen alue on vaihtelevan kokoinen riippuen siitä, kuinka monta konttia alueella on peräkkäin ja kuinka monta vierekkäin. Alueet on siis jaettu riveihin ja sektoreihin.



Kuva 7. Terminaalin alueita.

Kuvan 8 esimerkissä näkyy, miten alueella P on 7 sektoria ja 2 riviä. Alueen, sektorien ja rivien perusteella saadaan jokaiselle pohjapaikalle luotua terminaalin sisäinen sijainti: alue - sektori - rivi.



Kuva 8. P-alueen sektorit ja rivit.

Kontteja voidaan pinota myös päällekkäin riippuen siitä, onko kontti tyhjä vai onko sillä lastia. Myös tavara, jota kontissa on, määrittelee, voiko kontteja pinota päällekkäin ja jos voi niin kuinka monta. Yleisesti bulkkitavarakontteja voidaan pinota päällekkäin neljä sillä ehdolla, että lastikas kontti ei saa olla tyhjän kontin päällä. Konttien kerrostieto tuo siis viimeisen paikkatiedon kontin sijaintiin. Tällöin kokonaisuudessaan kontin sijaintitieto on alue - sektori - rivi - kerros, eli esimerkiksi P-4-1-1, kuten kuvan 8 esimerkistä näkyy.

Tämä sijaintitieto kertoo kontin sijainnin henkilölle, joka tuntee terminaalin alueen ja tietää järjestelmän. Kontin sijainti on kuitenkin suhteellinen alueeseen nähden eikä tarkka, kuten koordinaatit sijainnin kertovat.

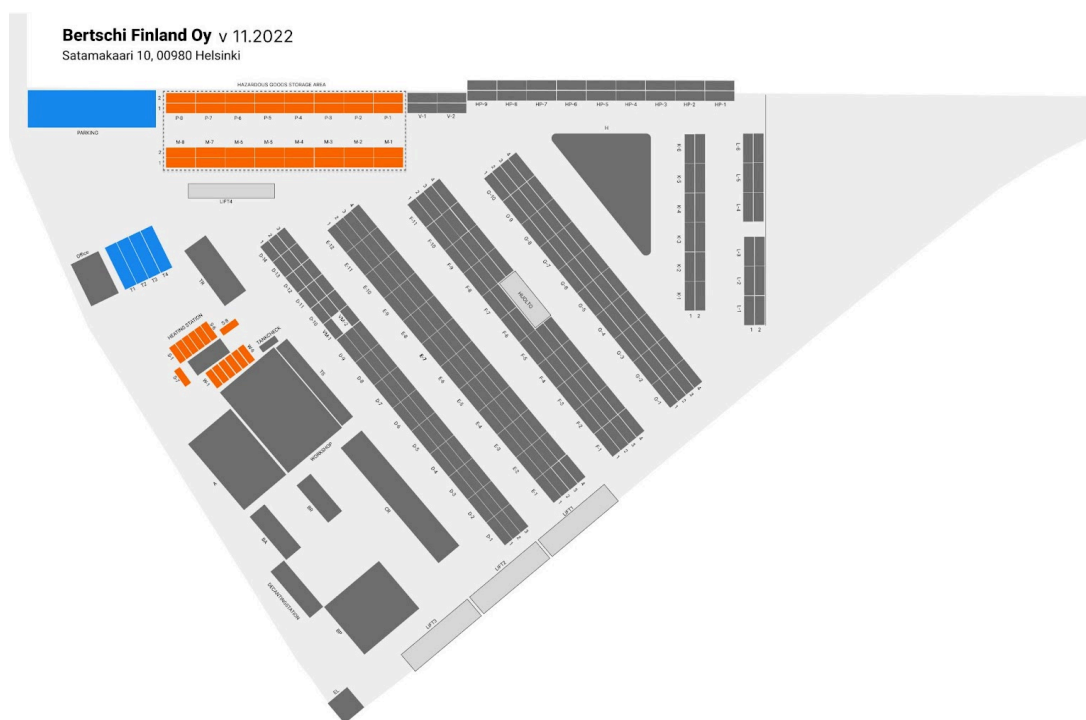
Koordinaatit 60.21577732761796, 25.16795899942841, jotka vastaavat paikan P-4-1 keskipistettä, eivät kuitenkaan ole käytännöllisiä ihmisille. Tästä syystä päädyttiin ratkaisuun, joka käyttää alkuperäistä, ihmiselle selkeää paikkatietoa sekä koordinaattitietoja, jolloin tarkka sijaintitieto ja helposti ymmärrettävä tieto ovat yhdessä.

4.2 Vanha kartta

Terminaalisovelluksen vanha kartta oli Figma-sovelluksella manuaalisesti luotu SVG-kuva (kuva 9).

Ongelmaksi tämän kartan kanssa muodostui sen päivittäminen; kun terminaalin alueiden järjestys muuttuu, luodaan uusi alue, lisätään alueeseen rivi tai tulee jotain muuta muutosta, tarvitsee kartta päivittää. Vanhalla kartalla tätä päivittämistä ei voinut helposti tehdä, sillä kartan päivittäminen vaati aina sovelluksen uuden version julkaisua.

Toinen ongelma, joka vanhan kartan kanssa oli, oli sen epätarkkuus. Kartta piirrettiin käsin satelliittikuvan päälle, mutta satelliittikuvat eivät olleet koskaan ajan-tasaisia, joten paikat piirrettiin suunnilleen sinne, minne ne kuuluivat.



Kuva 9. Terminaalin vanha SVG-kartta.

4.3 Kartan päivittäminen

Vanhan kartan ylläpito ja epätarkkuus oli ollut ongelma jo parin vuoden ajan aina siitä asti, kun karttanäkymä ensimmäisen kerran otettiin sovelluksessa käyttöön. Vaikka karttanäkymä mullistikin terminaalin paikoituksen ja visualisoi sitä paljon paremmin kuin listanäkymä, oli siinä omat huonot puolensa.

Terminaalisovellukseen haluttiin kartta, joka näyttää samalta kuin aikaisempi kartta, mutta on helppo päivittää ja tarkka (kontit ja mittasuhteet ovat kartassa samalla tavalla kuin oikeasti ovat).

Kartan uudistuksen prosessi kokonaisuudessaan toteutettiin kuvan 10 esittämällä tavalla.



Kuva 10. Kartan uudistamisen prosessikaavio.

4.3.1 Kartan pohjana GeoJSON

Ongelman kriteerit täyttävä ratkaisu löytyi luomalla terminaalin pohjapaikoista GeoJSON-kartta. Nämä monikulmiot ovat samanlaisia objekteja kuin SVG-kartan yksittäiset pohjapaikat sillä erotuksella, että jokaisen pisteen koordinaatit eivät ole suhteellisia SVG-kuvaan nähden vaan aitoja geokoordinaatteja.

Terminaalin alueet ovat aina kooltaan $n \times$ kontin pituus \times $n \times$ kontin leveys. Konttien välissä oleva etäisyys pituussuunnassa on 0,5 m, ja leveyssuunnassa kontit ovat toisissaan kiinni. Tällöin jokaisen alueen 'origo' on siis ensimmäisen sektorin ja ensimmäisen rivin kontin alakulma. Jotta GeoJSON voidaan generoida, tarvitaan tämän kulmapisteen koordinaatit ja rivin suuntima asteina.

Tämän vuoksi tarvittiin siis myös ensimmäisen rivin viimeisen kontin kulmapisteen suuntiman laskemiseksi.

4.3.2 Koordinaattien mittaus

Tarkkojen koordinaattien mittaukseen hankittiin ArduSimple-yritykseltä RTK Base-Rover -paketti. Se sisälsi RTK-tukiaseman ja itse mittaukseen tarvittavan siirrettävän mittatikun. Tukiasema oli tässä paketissa siirrettävä, kolmijalalle asetettava laite, mutta koska tässä tapauksessa laitetta käytetään vain terminaalissa, asennettiin tukiasema kiinteästi toimiston katolle (kuva 11). Maanmittauslaitoksen materiaaleista saatiin tietoon toimiston kulmapisteen tarkat koordinaatit, ja niiden perusteella laskettiin tarkka sijainti tukiasemalle.



Kuva 11. RTK-tukiasema toimiston katolla.

Rover-laite vaihdettiin käyttämään nyt ylimääräiseksi jäänyttä kolmijalkaa, jotta laite olisi tukevasti maassa mittauksia tehdessä (kuva 12).



Kuva 12. RTK Rover kolmijalalla.

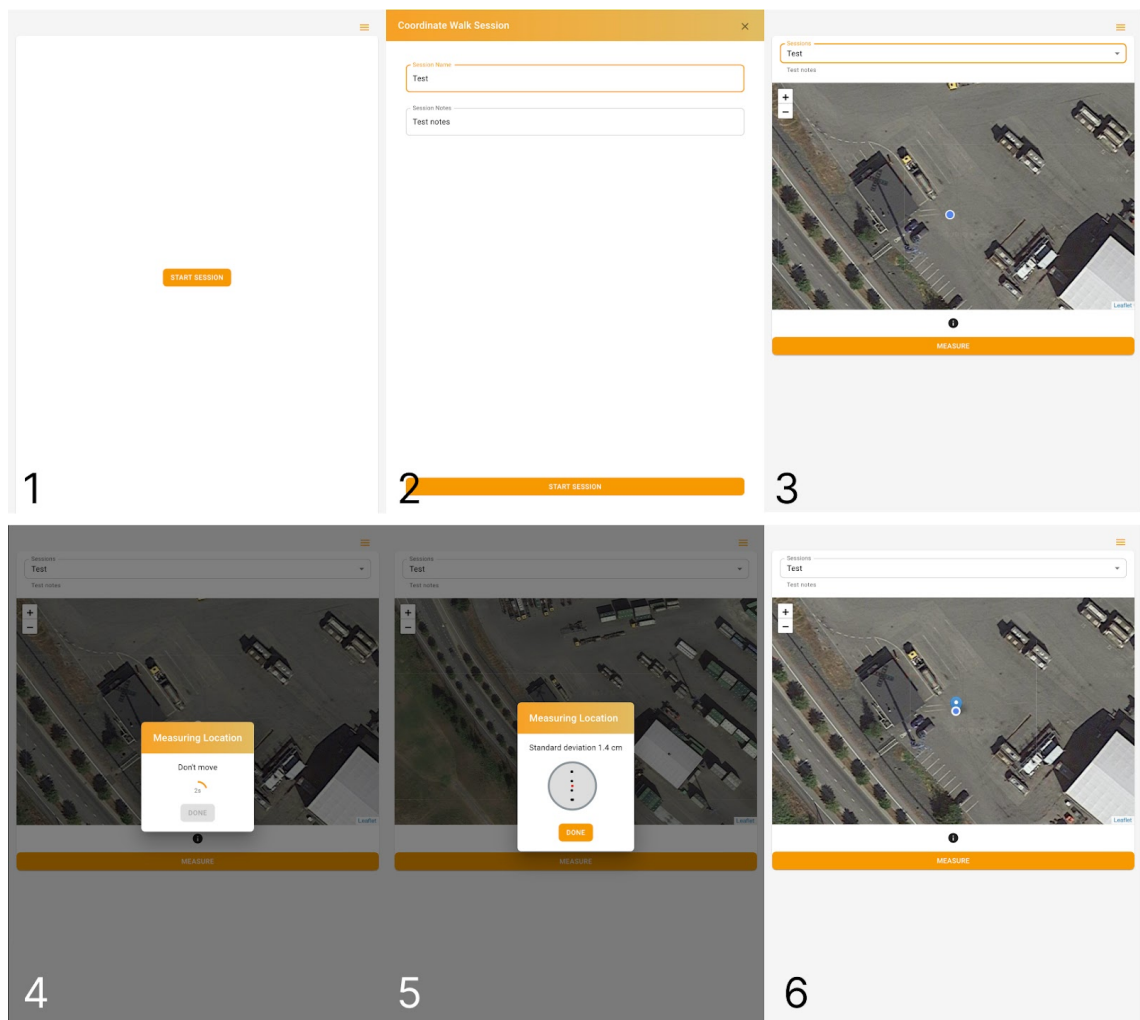
Roverissa on Bluetooth-yhteys, jolla se saadaan yhdistettyä käytössä olevaan mobiililaitteeseen. Lefebure NTRIP Client -sovelluksella saadaan mobiililaitteen sijaintitiedoksi vaihdettua sijaintitieto, joka Roverilta saadaan. Tällöin omaa sovellusta käytettäessä sivun pyytäessä laitteelta sijaintitietoa se saa Roverin tarkan RTK-korjauksella olevan sijaintitiedon.

4.3.3 Pistemittaustyökalu

Jotta GeoJSON terminaalin konttien paikoista voidaan luoda, tarvitaan alueen kulmapisteen, 'orion' koordinaatit. Sitä varten täytyi luoda mittaussovellus, jotta koordinaatit voitiin mitata RTK-mittatikulla.

Mittaussovellukseen tuli luoda sessio, johon kyseisen mittauskerran mittapisteet tallennettiin. Tämän jälkeen mittausseesio-objekti tallennettiin tietokantaan ja tulevat mittapisteet tallentuivat sinne.

Jotta voitiin varmistaa, että mitattu sijainti olisi varmasti mahdollisimman tarkka, täytyi sovelluksen mitata sijaintia sekunnin välein 10 sekunnin ajan ja lopulta laskea keskiarvo näistä koordinaateista. Näin jopa RTK-mittauksen senttimetrin tarkkuudessa päästiin vieläkin tarkempaan lopputulokseen. Kuvan 13 näkymä 5 näyttää mittauksen jälkeen mitatut pisteet ja niiden keskihajonnan eli keskimääräisen etäisyyden keskiarvosta.



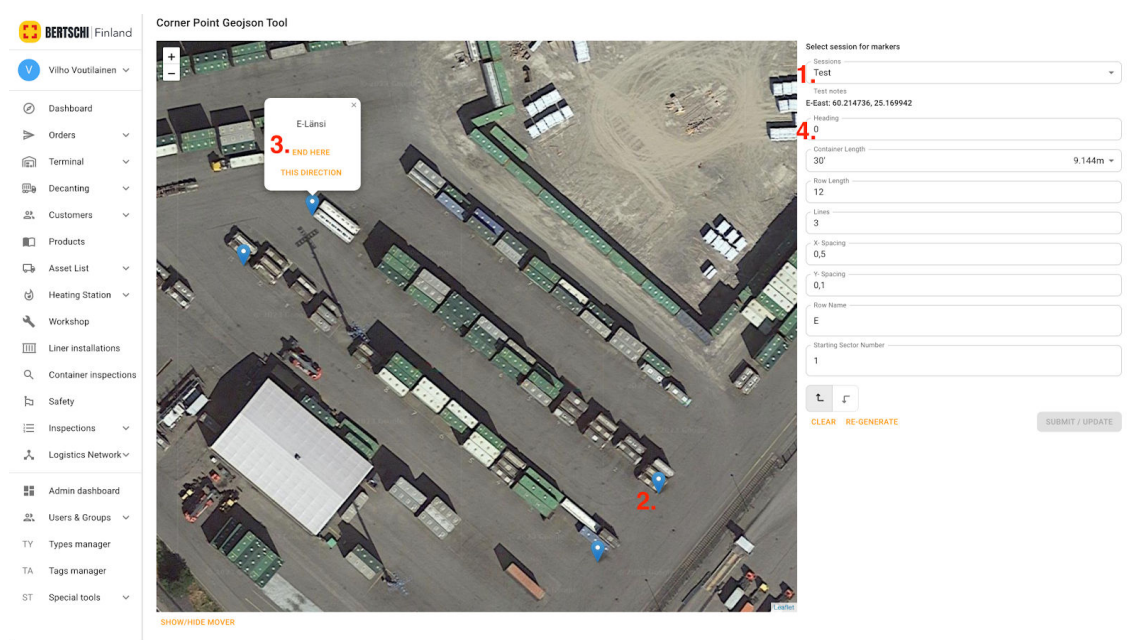
Kuva 13. Mittausseesession näkymät mobiililaitteen ruudulla.

4.3.4 GeoJSON-työkalu

Kuva 13 näyttää, minkälainen työkalu itse GeoJSONin luomiseen on.

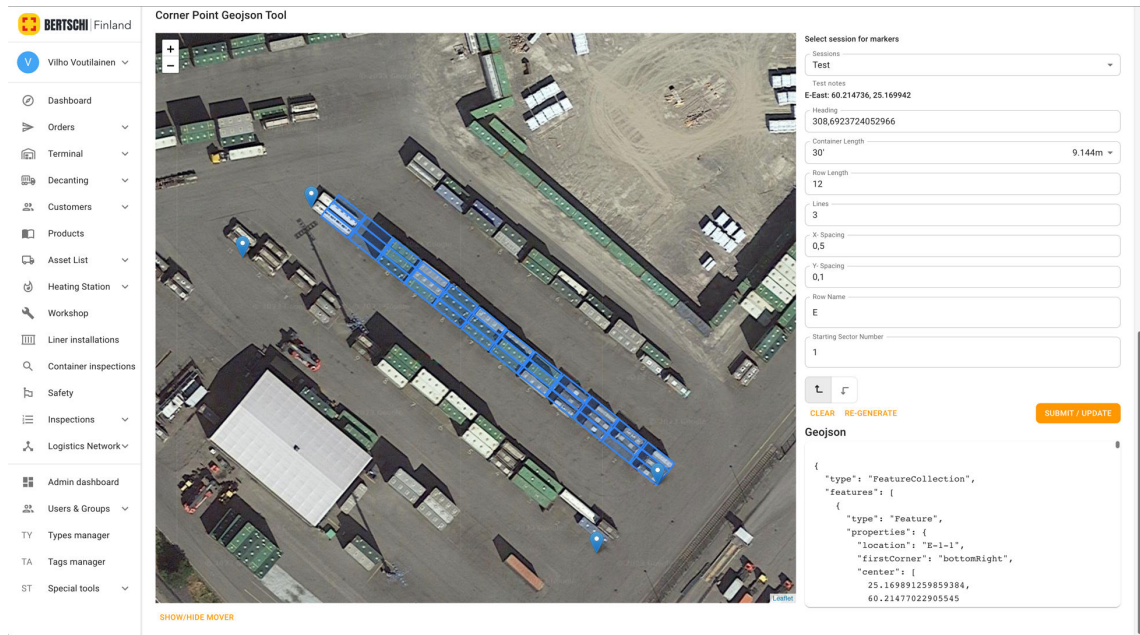
GeoJSONin luominen aloitetaan valitsemalla (kuva 14 kohta 1), minkä mittaus-session koordinaattipisteitä käytetään.

Tämän jälkeen mitatut koordinaattipisteet aukeavat kartalle. Seuraavaksi valitaan kulmapiste, josta GeoJSON lähdetään generoimaan (kuva 14 kohta 2). Jotta GeoJSON generoidaan oikeaan suuntaan rivin mukaisesti, valitaan päätepisteeksi rivin viimeisen paikan mitattu kulmapiste (kuva 14 kohta 3). Päätepisteen kanssa voidaan valita toinen seuraavista: rivin viimeisen kontin kulma tähän pisteeseen, jolloin konttien väliin jäävä etäisyys lasketaan dynaamisesti etäisyyden ja konttimäärän mukaan, tai alueen aloituspisteestä päätepisteen suuntaan, jolloin konttien väliin jäävä etäisyys määräytyy asetusten mukaisesti (kuva 14 kohta 4).



Kuva 14. Alueen kulmapisteet GeoJSON-työkalussa.

Esimerkki E-alueen valmiista generoidusta GeoJSONista näkyy kuvassa 15.



Kuva 15. GeoJSON-työkalulla generoitu GeoJSON kartalla.

Funktio `generateAreaGeojson()`, joka generoi GeoJSONin, ottaa parametreina alkupisteen koordinaatit, alueen suuntiman, yksittäisen kontin pituuden, yksittäisen kontin leveyden, leveyssuunnassa väliin jäävän etäisyyden, pituussuunnassa väliin jäävän etäisyyden, rivin pituuden kontteina, rivien lukumäärän, alueen nimen, generointisuunnan ja ensimmäisen sektorin indeksinumeron (koodiesimerkki 2).

```

/**
 * Create geojson positions for containers
 *
 * @param startCoordinate {array} - [lat, lon]
 * @param heading {number} - heading in degrees
 * @param containerLength {number} - length of container in meters ex
9.144m / 30ft
 * @param containerWidth {number} - width of container in meters
2.4384m / 8ft
 * @param containerSpacingX {number} - spacing between containers in
meters on X-axis
 * @param containerSpacingY {number} - spacing between containers in
meters on Y-axis
 * @param rowLength {number} - length of row in meters X-axis, Sectors
 * @param lines {number} - number of lines in row Y-axis, Depth
 * @param rowName {string} - name of row
 * @param direction {boolean} - true if row is going from left to
right, false if row is going from right to left
 * @param startingSectorNumber {number} - starting sector number
 * @returns {geoJSON} - geojson object
 */
const generateAreaGeojson = (params) => {
  const {
    startCoordinate,
    heading,
    containerLength = 9.144,
    containerWidth = 2.44,
    containerSpacingX = 0.8,
    containerSpacingY = 0.1,
    rowLength,
    lines = 1,
    rowName,
    direction = true,
    startingSectorNumber = 1,
  } = params;

  let startingCorner = startCoordinate;
  let geoJson = [];

  try {
    for (let i = 1; i <= lines; i++) {
      for (let j = 1; j <= rowLength; j++) {

        // loop around container to create gps points

        let rightFront = startingCorner;
        let leftFront = moveTo(rightFront, { heading: heading, dis-
tance: containerLength });
        let leftBack = moveTo(leftFront, {
          heading: direction ? heading + 90 : heading - 90,
          distance: containerWidth,
        });

        let rightBack = moveTo(leftBack, {
          heading: direction ? heading + 180 : heading - 180,
          distance: containerLength,
        });

        const sector = startingSectorNumber !== 1 ? startingSector-
Number + j - 1 : j;
        const feature = {

```

```

        type: 'Feature',
        geometry: {
          type: 'Polygon',
          coordinates: [
            [
              toLonLatTuple(rightFront),
              toLonLatTuple(rightBack),
              toLonLatTuple(leftBack),
              toLonLatTuple(leftFront),
              toLonLatTuple(rightFront),
            ],
          ],
        },
        properties: {
          location: rowName + '-' + sector + '-' + i, // ex. G-1-1
        },
      };

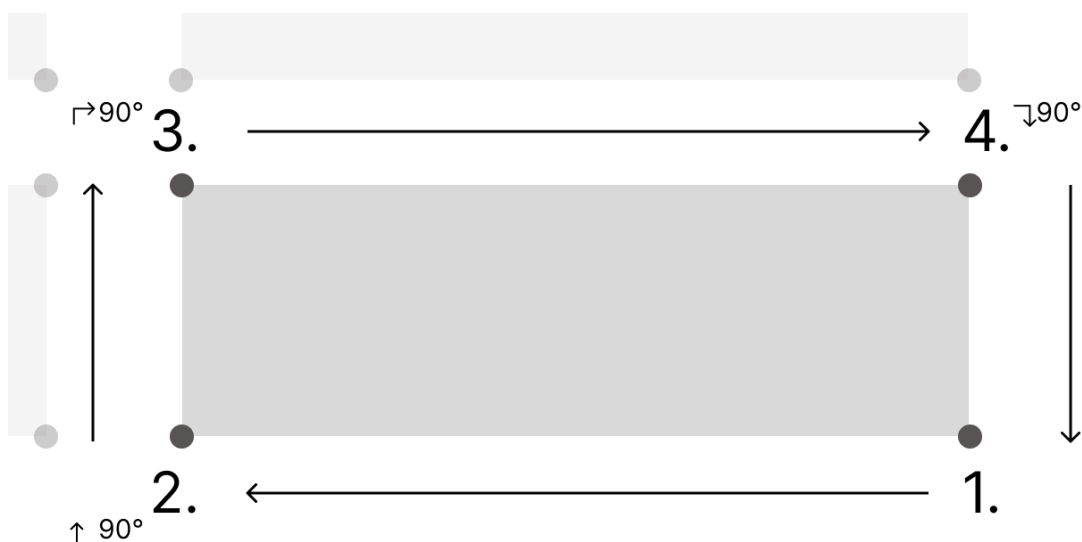
      geoJson.push(feature);
      startingCorner = moveTo(startingCorner, { heading: heading,
distance: containerLength + containerSpacingX });
    }
    startingCorner = moveTo(startCoordinate, {
      heading: direction ? heading + 90 : heading - 90,
      distance: i * (containerWidth + containerSpacingY),
    });
  }
} catch (e) {
  console.log('Error in positionMappingFunction: ', e);
  return e.message;
}
const geoJsonFull = {
  type: 'FeatureCollection',
  features: geoJson,
};

return geoJsonFull;
};

```

Esimerkkikoodi 2. Funktio GeoJSONin luomiseen.

Funktio käy kahden sisäkkäisen for-loopin sisällä läpi jokaisen rivin ja sektorin ja luo aina yhden GeoJSON-monikulmion pohjapaikkaa kohden. Monikulmio luodaan laskemalla jokaisen kulmapisteen koordinaatit. Ensimmäinen kulmapiste on parametreissa määritelty aloituspiste. Seuraavat kulmapisteet ovat kontin pitiuden tai leveyden verran 90 astetta edellistä suuntimaa enemmän (kuva 16).



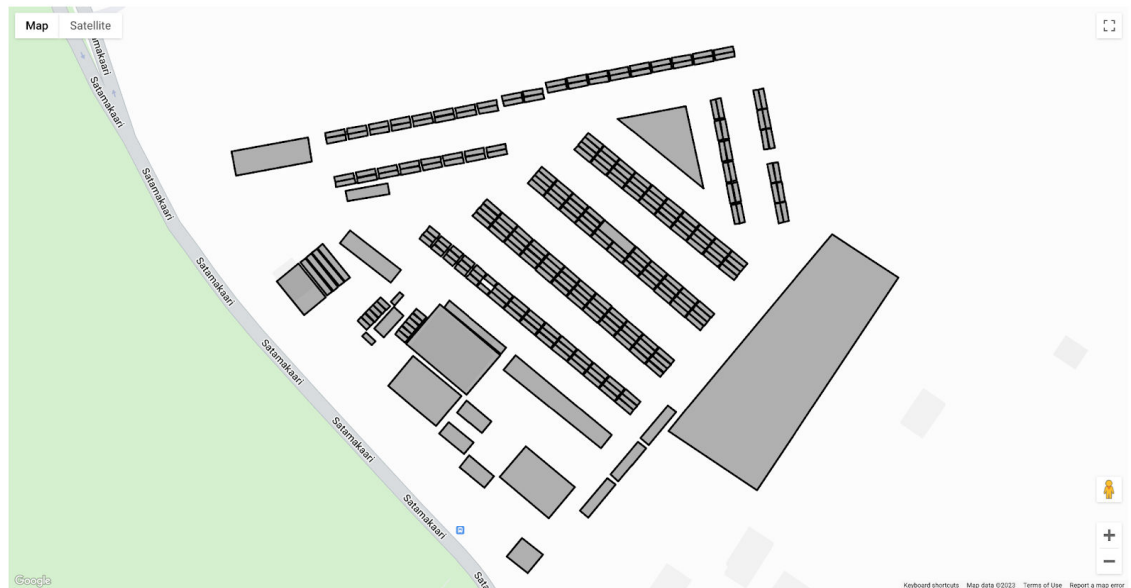
Kuva 16. Monikulmion koordinaattien piirto.

Kulmapisteiden koordinaatit lasketaan TurfJS-kirjaston [24] moveTo-funktion avulla, joka ottaa parametreina lähtöpisteen, suuntiman ja etäisyyden.

GeoJSON-standardi vaatii, että monikulmio päättyy samaa pisteeseen, josta se alkaa, joten viimeinen piste on sama kuin piste, josta lähdettiin liikkeelle. Viimeinen koordinaattisiirtymä on, kun funktio asettaa seuraavalle monikulmion lähtöpisteeksi kontin pituus + konttien väliin jäävä etäisyys, jotta seuraavaa konttia piirrettäessä lähtöpiste on uuden monikulmion ensimmäinen kulma. Jokaisen monikulmion properties-objektiin lisätään pohjapaikkatieto eli esimerkiksi G-1-1.

Tätä jatketaan niin monta kertaa, kuin rivillä on paikkoja. Kun rivin viimeinen paikka on käsitelty, siirretään aloituspiste alkuperäisestä kulmapisteestä kontin leveyden ja väliin jäävän leveyden verran, jotta päästään seuraavan rivin ensimmäiseen paikkaan, jolle tehdään samat toimet.

Kun alueen GeoJSON on luotu, tallennetaan tietokantaan erillinen objekti alueen metadatasta, eli tieto alueen nimestä, rivien määrästä, sektorien määrästä, aloituskoordinaateista ja suuntimasta. Tämän lisäksi tietokantaan tallennetaan jokainen yksittäinen monikulmio omana GeoJSON Featurena, jotta päästään helposti käsiksi yksittäisiin pohjapaikkoihin. Kuva 17 näyttää kartan päällä esitettyä kaikki luodut GeoJSON-pohjapaikat.



Kuva 17. GeoJSON-data Google Maps -kartalla.

4.3.5 Paikkatiedon versionhallinta

Koska GeoJSONia on nykyisillä työkaluilla helppo muokata, täytyi sovellukseen luoda versionhallinta eri versioille. Jokainen tallennettu GeoJSON saa revisionumeron tietokantadokumenttiin, jotta pysytään perässä, mihin revisioon mikäkin GeoJSON kuuluu. Tällöin uusia alueita voidaan suunnitella helposti etukäteen ja kun kartta halutaan päivittää, vaihdetaan vain aktiivinen revisio toiseen.

4.3.6 Sovelluksen karttanäkymä

Sovelluksen karttanäkymää ei haluttu visuaalisesti muuttaa, sillä aikaisemman karttanäkymän esitystapa oli todettu toimivaksi. Uuden GeoJSON-pohjaisen SVG-kartan generointiin käytettiin React Simple Maps -kirjastoa [25] (RSM-kirjastoa).

Kartan pohjaksi täytyy luoda RSM:n ComposableMap-komponentti. Tämä ylimmän tason komponentti määrittelee SVG-kuvan vektorikoordinaatiston vastamaan maantieteellistä koordinaatistoa. Komponentissa määritellään kartan esittämiseen asetuksia, kuten projection eli miten kartta projisoidaan. Tämän

asetuksen avulla voidaan määritellä, miten kolmiulotteinen kartta esitetään 2D-näytöllä. Komponentissa määritellään myös kartan skaalaus, kierto ja itse komponentin koko sovelluksessa (esimerkkikoodi 3). Kaikki seuraavaksi mainitut komponentit sijoitetaan tämän komponentin sisään.

```
<ComposableMap
  projection="geoMercator"
  width={window.innerWidth / 1.5}
  height={window.innerHeight / 1.5}
  projectionConfig={{
    rotate: [-10.9, -52.3, -22.2],
    scale: 15969600,
  }}
/>
```

Esimerkkikoodi 3. React Simple Maps ComposableMap -komponentti.

Jotta karttaa voidaan zoomata ja siirrellä, käytettiin RSM:n ZoomableGroup-komponenttia. Tässä komponentissa määritellään kartan keskipiste ja zoomauksen minimi- ja maksimitasot (esimerkkikoodi 4).

```
<ZoomableGroup center={mapCenter} minZoom={0.3} maxZoom={6} >
```

Esimerkkikoodi 4. React Simple Maps ZoomableGroup -komponentti.

Itse GeoJSON luodaan kartalle Geographies-komponentin avulla. Tämä komponentti ottaa parametrina terminaalien alueista luodun GeoJSONin ja luo jo-kaista yksittäistä Featurea vastaavan Geography-komponentin (esimerkkikoodi 5).


```

<Geographies geography={terminalGeojson}>
  ({ geographies }) => {
    return geographies?.map((geo) => (
      <Geography
        id={geo?.properties?.location}
        className={state?.selected === geo.properties?.location ?
'svgMapsLocationSelected' : ''}
        geography={geo}
        onClick={(e) => {
          e.preventDefault();
          handleClick(geo.properties?.location);
        }}
        style={{
          default: {
            fill: '#ff9800',
            outline: 'none',
          }}
      />
    ));
  }}
</Geographies>

```

Esimerkkikoodi 5. React Simple Maps Geographies -komponentti.

Geography-elementit saavat ID-arvon, joka vastaa monikulmion pohjapaikan sijaintia. Jokaiselle Geography-elementille määritellään oletusarvoiseksi väriksi #ff9800 eli vaalean harmaa. Jos monikulmiolle halutaan jokin muu väri, voidaan sitä muuttaa monikulmion ID:n perusteella kartan CSS-tiedostossa (esimerkkikoodi 6).

```
#M-1-1, #M-1-2, #M-1-3,  
#M-2-1, #M-2-2, #M-2-3,  
#M-3-1, #M-3-2, #M-3-3,  
#M-4-1, #M-4-2, #M-4-3,  
#M-5-1, #M-5-2, #M-5-3,  
#M-6-1, #M-6-2, #M-6-3,  
#M-7-1, #M-7-2, #M-7-3,  
#M-8-1, #M-8-2, #M-8-3,
```

```
#P-1-1, #P-1-2, #P-1-3,  
#P-2-1, #P-2-2, #P-2-3,  
#P-3-1, #P-3-2, #P-3-3,  
#P-4-1, #P-4-2, #P-4-3,  
#P-5-1, #P-5-2, #P-5-3,  
#P-6-1, #P-6-2, #P-6-3,  
#P-7-1, #P-7-2, #P-7-3,  
#P-8-1, #P-8-2, #P-8-3,
```

```
{  
    fill: #ec5403;  
}
```

Esimerkkikoodi 6. CSS-tyylikoodi.

4.4 Kartan päivitys

Kartan päivittämisen prosessi menee kokonaisuudessa kuvan 18 kuvaamalla tavalla. Ensimmäiseksi koordinaattimittaustyökalulla luodaan mittausseessio, johon mitataan alueiden kulmapisteiden koordinaatit käyttäen RTK-mittatikkua. Näiden tarkasti mitattujen koordinaattien mittauksen jälkeen GeoJSON-työkalulla luodaan koordinaattipisteistä uudet GeoJSON-alueet. Alueet tallennetaan sovelluksen tietokantaan uuden revision alle. Kun uusi kartta halutaan ottaa käyttöön, vaihdetaan aktiiviseksi revisioksi uusi luotu revisio, jolloin sovelluksen karttanäkymä vaihtaa käyttöön vanhan GeoJSONin tilalle uuden luodun GeoJSONin ja luo uuden SVG-kartan tämän perusteella. Näin vanha kartta korvaantuu kokonaan uudella ja ajankohtaisella päivitetyllä kartalla.

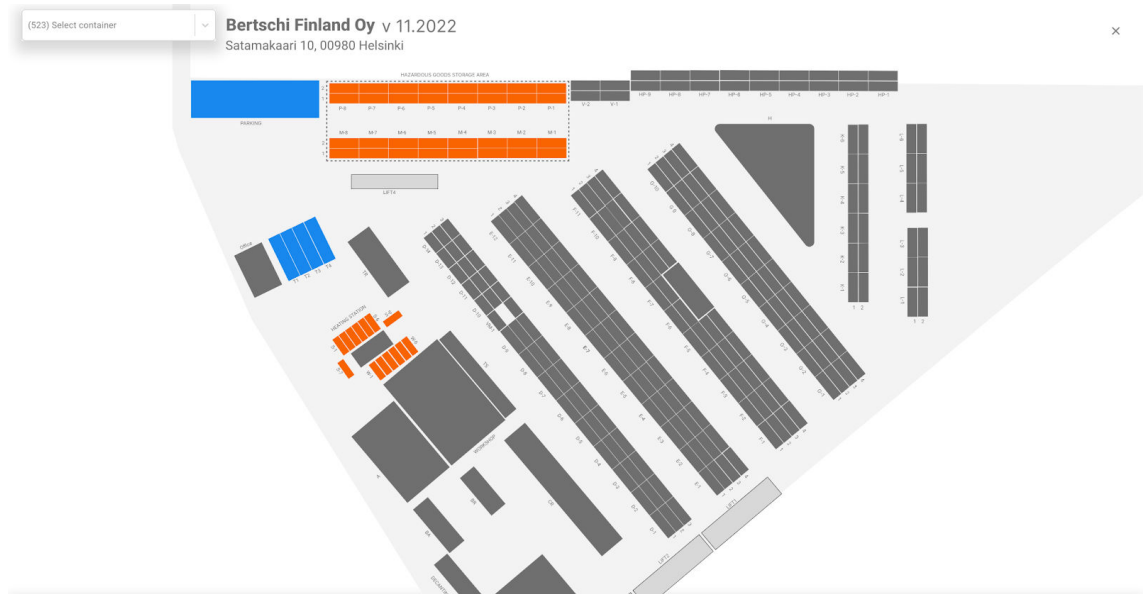


Kuva 18. Kartan päivittämisen prosessi.

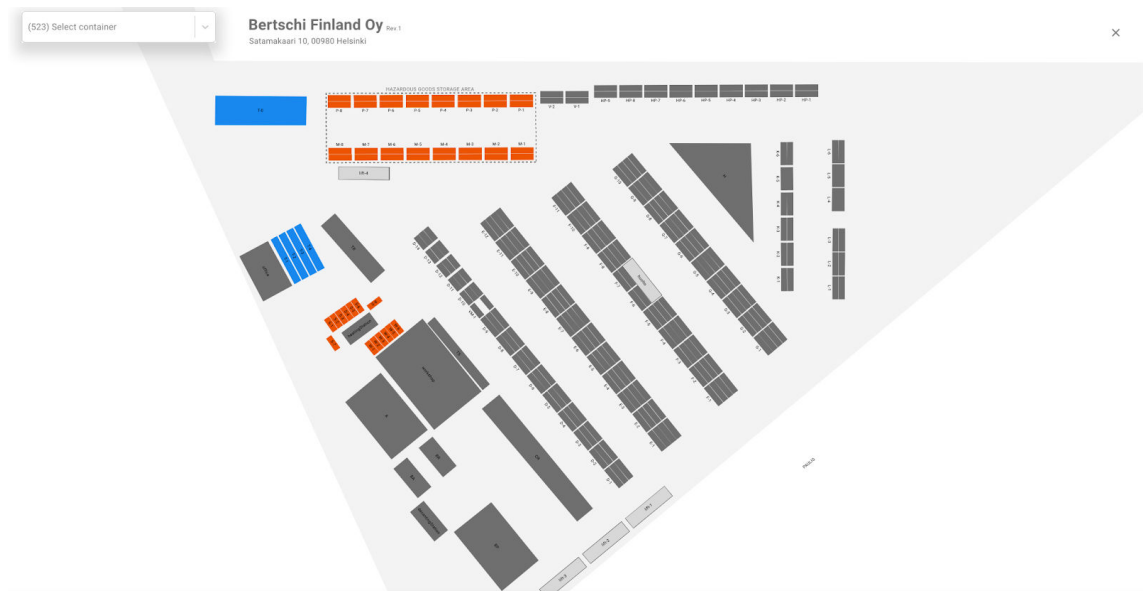
Tulevaisuudessa esimerkiksi tilanteessa, jossa yritys perustaa uuden terminaalin tai laajentaa nykyistä terminaalialia, olisi kartan päivittäminen tai kokonaan uuden kartan luominen helppoa luotujen työkalujen avulla.

4.5 Uusi kartta

Visuaalisesti uuden kartan tavoitteena oli, että muutos kartassa olisi mahdollisimman pieni, vaikka pinnan alla kaikki muuttui (kuvat 19 ja 20). Käyttäjälle tällöin muutos olisi mahdollisimman pieni eivätkä totutut toimintamallit muuttuisi.



Kuva 19. Vanha terminaalin karttanäkymä.



Kuva 20. Uusi terminaalin karttanäkymä.

5 Yhteenveto

Projektin tavoitteena oli uusia terminaalisovelluksen karttanäkymä vastaamaan aidossa mittakaavassa terminaalin tilaa. Tämä uusi kartta toteutettiin luomalla konttien paikoista terminaalissa GeoJSON-kartta, josta lopulta generoitiin uusi SVG-kuva karttanäkymään. Toinen projektin vaatimus oli kartan helppo muokaus jatkossa.

Kaikki projektin tavoitteet täyttyivät ja sain kehitettyä yritykselle helposti muokattavan ja pihan todellista tilannetta vastaavan konttipinokartan.

Tämä maantieteelliseen koordinaatistoon perustuva GeoJSON on tulevaisuudessa pohja, kun terminaalin konttikurottajiin asennetaan GPS-paikannuslaitteet, joilla konttien paikoitus on tarkoitus tehdä automaattisesti. Tulevaisuudessa kun kurottaja nostaa kontin, se tietää, miltä paikalta kontin nosti, ja näin ollen siis tietää, minkä kontin nosti kyytiin. Myös konttia laskiessa kurottaja tietää GPS:n perusteella, minne kontin laski, jolloin kontin uudelleenpaikoitus voidaan toteuttaa automaattisesti.

Tämä automaattinen paikoitus tulee vähentämään ihmisen tekemiä inhimillisiä virheitä, joissa esimerkiksi kontti on paikoitettu väärälle paikalle. Automaattinen paikoitus vähentää myös konttikurottajan kuljettajan tarvetta käyttää terminaalisovelluksen tablettia ajon aikana, mikä lisää työn turvallisuutta ja tehokkuutta.

Haastetta insinööriyössä tuotti sen kokonaislaajuus, joka vaikeutti aiheen rajaamista. Valittu katsontakulma kartan päivityksestä tuntui helpoimmalta näkökulmalta tarkastella kokonaisuutta.

Projektin yksittäisistä osa-alueista, kuten RTK-mittauksesta, olisi voinut yksinään kirjoittaa kokonaisen opinnäytetyön. Jotta työ pysyi kokonaisuutena suhteellisen selkeänä, tuli kaikista asioista käsiteltyä vain pintaraapaisu. Olen kuitenkin tyytyväinen projektin lopputulokseen ja odotan innolla tulevaisuuden jatkokehitystä automaattipaikoituksen osalta.

Lähteet

- 1 Facts & figures. Verkkoaineisto. Bertschi AG.
<<https://www.bertschi.com/en/about/facts-and-figures>> Luettu 25.2.2023.
- 2 Dunbar, Brian. 2012. Global Positioning System history. Verkkoaineisto. NASA <https://www.nasa.gov/directorates/heo/scan/communications/policy/GPS_History.html> Luettu 4.3.2023.
- 3 How does GPS work. 2019. Verkkoaineisto. NASA <<https://spaceplace.nasa.gov/review/gps/constellation.en.jpg>> Luettu 4.3.2023.
- 4 Satelliittipaikannus. Verkkoaineisto. Maanmittauslaitos.
<<https://www.maanmittauslaitos.fi/tutkimus/teematietoa/satelliittipaikannus>> Luettu 4.3.2023.
- 5 The Global Positioning System. Verkkoaineisto. National Oceanic and Atmospheric Administration. <https://oceanservice.noaa.gov/education/tutorial_geodesy/geo09_gps.html> Luettu 4.3.2023.
- 6 Blewitt, G. 1997. Basics of the GPS Technique: Observation Equations. Verkkoaineisto. Department of Geomatics, University of Newcastle
<<https://web.gps.caltech.edu/classes/ge111/Docs/GPSbasics.pdf>> Luettu 4.3.2023.
- 7 Real-Time Kinematic (RTK). 2022. Verkkoaineisto. U-Blox.
<<https://www.u-blox.com/en/technologies/rtk-real-time-kinematic>> Luettu 4.3.2023.
- 8 Eriksson, TS. RTK concept. 2020. Verkkoaineisto. Wikipedia.
<https://en.wikipedia.org/wiki/Real-time_kinematic_positioning#/media/File:Real_time_kinematic.svg> Luettu 4.3.2023
- 9 How RTK works. Verkkoaineisto. Emlid. <<https://docs.emlid.com/reachrs/rtk-quickstart/rtk-introduction/>> Luettu 4.3.2023.
- 10 Centimeter Precision GPS/GNSS – RTK Explained. Verkkoaineisto. ArduSimple. <<https://www.ardusimple.com/rtk-explained/>> Luettu 4.3.2023.
- 11 GeoJSON. Verkkoaineisto. ESRI. <<https://doc.arcgis.com/en/arcgis-online/reference/geojson.htm>> Luettu 6.3.2023.

- 12 GeoJSON, Version 1.0 (2008). 2008. Verkkoaineisto. Library of Congress. <<https://www.loc.gov/preservation/digital/formats/fdd/fdd000382.shtml>> Luettu 6.3.2023.
- 13 GEOJSON. Verkkoaineisto. Geojson.org. <<https://geojson.org/>> Luettu 6.3.2023.
- 14 Butler et al. The GeoJSON Format. 2016. Verkkoaineisto. <<https://www.rfc-editor.org/rfc/rfc7946>> Luettu 6.3.2023.
- 15 Understanding client-side JavaScript frameworks. 2023. Verkkoaineisto. MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks> Luettu 22.3.2023.
- 16 Vailshery, L. S. 2022. *Most used web frameworks among developers 2022*. Verkkoaineisto. Statista. <<https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/>> Luettu 21.3.2023.
- 17 Banks, Alex & Porcello, Eve. 2020. Learning React. 2nd Edition. E-kirja. O'Reilly.
- 18 Virtual Dom. 2017. Verkkoaineisto. Medium. <<https://medium.com/naukri-engineering/naukriengineering-virtual-dom-fa8019c626b>> Luettu 6.4.2023
- 19 Mobile internet usage worldwide – Statistics & Facts. 2023. Verkkoaineisto. Statista. <<https://www.statista.com/topics/779/mobile-internet/>> Luettu 22.3.2023.
- 20 The guide to responsive web design in 2022. 2021. Verkkoaineisto. Webflow. <<https://webflow.com/blog/responsive-web-design>> Luettu 22.3.2023.
- 21 Iltalehti. 2023. Verkkoaineisto. Iltalehti. <<https://iltalehti.fi>> Luettu 23.3.2023.
- 22 What is the European Union's Web Accessibility Directive? 2021. Verkkoaineisto. Bureau of Internet Accessibility. <<https://www.boia.org/blog/what-is-the-european-unions-web-accessibility-directive-1>> Luettu 23.3.2023.
- 23 Accessible Rich Internet Applications (WAI-ARIA) 1.1. 2017. Verkkoaineisto. W3C. <<https://www.w3.org/TR/wai-aria/>> Luettu 23.3.2023.
- 24 TurfJS Documentation. Verkkoaineisto. TurfJS. <<https://turfjs.org/>> Luettu 9.3.2023.

- 25 React Simple Maps Documentation. Verkkoaineisto. React Simple Maps.
<<https://www.react-simple-maps.io/>> Luettu 9.3.2023.