# LAM-Visualisointi

## Liikenteen automaattiset mittaustiedot

Aleksi Pynnönen

Opinnäytetyö, AMK
Kuukausi 2023
Tieto- ja viestintätekniikka

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

**Kuvailulehti**

**Pynnönen, Aleksi**

**LAM visualisointi. Liikenteen automaattiset mittaustiedot.**

Jyväskylä: Jyväskylän ammattikorkeakoulu. **Huhtikuu 2023**, 48 sivua

Tieto- ja viestintätekniikan tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: kyllä

**Tiivistelmä**

Opinnäytetyön tehtävänä oli visualisoida Fintrafficin Liikenteen automaattisten mittaustietojen eli LAM-asemien keräämää tietoa tieliikenteestä. Näistä tiedoista tavoitteena oli visualisoida liikennemäärät ja ajoneuvojen nopeudet. Lisäksi tavoitteena oli kehittää sovellus visualisoimaan LAM-asemista saatavia tietoja. Opinnäytetyön toimeksiantajana toimi Jyväskylän ammattikorkeakoulun IT-Instituutti ja Tieto tuottamaan-hanke.

Sovelluksen kehitys tietojen visualisointiin toteutettiin JavaScriptin React-kehikolla. Sovelluksen päänäkymään lisättiin Suomen kartta react-simple-maps komponenttien avulla. LAM-asemat lisättiin karttaan karttamerkeillä. Visualisointia varten karttanäkymään tehtiin ponnahdusikkunatoimintoa hyödyntäen avattava ja suljettava näkymä. Visualisointinäkymän avaaminen toteutettiin muuttamalla karttamerkit painikkeiksi. Visualisointinäkymään lisättiin React Google charts pylväsdiagrammi visualisoimaan LAM-asemien tiedot. LAM-asemaan liittyvän karttamerkin valinnan yhteyteen lisättiin kyseisen LAM-aseman tietojen automaattinen haku Digitrafficin rajapinnan kautta. Pylväsdiagrammiin lisättiin toiminto nopeuksien valitsemiseksi, joka mahdollisti sen havainnollistamisen, mihin kellonaikoihin kyseisiä nopeuksia on ajettu.

Opinnäytetyön tavoitteet täyttyivät ja tuloksina syntyi vaatimusten mukaan toimiva sovellus visualisoimaan LAM-asemien keräämää liikenteeseen liittyvää tietoa. Sovellusta voidaan hyödyntää ajoneuvojen nopeuksien ja ylinopeuksien tunnistamista LAM-asemien kohdalla. Sovelluksen avulla voidaan lisäksi tarkastella, mihin aikaan nopeuksia on ajettu.

**Avainsanat (asiasanat)**

Visualisointi, Sovelluskehitys, Frontend, JavaScript, React

**Muut tiedot (salassa pidettävät liitteet)**

jamk

**Description**

**Pynnönen, Aleksi**

**LAM visualization**

Jyväskylä: JAMK University of Applied Sciences, April 2023, 48 pages

Information and Communication Technologies. Bachelor's Degree Programme in Information and Communication Technology. Bachelor's Thesis

Permission for open access publication: Yes

Language of publication: Finnish

**Abstract**

The purpose of the Thesis was to visualize road traffic data collected by Fintraffic's LAM-datapoints. The target was to visualize the traffic volumes and speeds of road traffic and develop an application to visualize the LAM station road traffic data. The client of the Thesis was JAMK University of Applied Sciences and Tieto tuottamaan project.

Visualization application was developed with JavaScript's React framework. Finland's map was added to the applications main view with react-simple-maps components and LAM-datapoint markers were added to the map. Visualization view needed to be opened and closed so that function was added to map markers. A column chart was added to visualization view with React Google charts to visualize LAM-datapoint data. LAM-datapoint selection was added to map markers along with automatic Digitraffic API call for data. Column chart speed column selection was added to visualize selected speed with that driven speeds time of day.

The targets of the Thesis were fulfilled. Compliant with requirements the application was created to functionally visualize LAM stations road traffic data. The application enables observing driven speeds and speeding of the LAM stations. It is also possible to check when the speeds were driven.

**Keywords/tags (subjects)**

Visualization, Application development, Frontend, JavaScript, React

**Miscellaneous (Confidential information)**

**Sisältö**

**1 Johdanto ... 7**
1.1 Opinnäytetyön tausta ... 7
1.2 Tavoitteet ja tehtävä ... 7
1.3 Menetelmät ... 9
**2 Teoreettiset lähtökohdat ... 9**
2.1 Datavisualisointi ... 9
2.1.1 Johdatusta visualisointiin ... 9
2.1.2 Mitä on Datavisualisointi ... 10
2.1.3 Datavisualisoinnin hyvät ja huonot puolet ... 10
2.2 Sovelluskehityksen elinkaari ... 11
2.3 Käyttöliittymäsuunnittelu sovelluskehityksessä ... 13
2.3.1 Käyttölittymäsuunnittelun merkityksestä ... 13
2.3.2 Käyttöliittymä suunnittelutyökalut ... 13
2.4 Kehitysvaihe sovelluskehityksessä ... 15
2.5 Figma työkalu käyttöliittymäsuunnittelussa ... 16
2.5.1 Mikä on Figma? ... 16
2.5.2 Suunnittelutyökalun toimintaperiaate ... 17
2.5.3 Suunnittelutyökalun hyödyt ... 17
2.6 Tieliikenne ... 19
2.6.1 Fintrafficin tieliikennetiedot ... 19
2.6.2 Digitraffic LAM-data ... 21
**3 Prototyypin Toteutus ... 22**
3.1 Suunnitteluvaihe ... 22
3.2 Prototyyppivaihe ... 24
3.3 Kehitysvaihe ... 25
3.3.1 Kehitysvaiheen aloitus ... 25
3.3.2 Datapisteet kartalle ... 28
3.3.3 CSV-datan käsittely React-kirjastolla ... 32
3.4 Visualisointi React-kehikoilla ... 34
3.4.1 Datan hakeminen ... 39

**4 Tulokset** ..... **41**

**5 Pohdinta** ..... **42**

**Lähteet** ..... **46**

**Kuviot**

Kuva 1. UX tools 2022 kysely suunnittelutyökalujen käytöstä ..... 14
Kuva 2. Figma reaaliaikaiset tapahtumat ..... 18
Kuva 3. Suunnittelu prosessi ennen ja jälkeen Figman ..... 19
Kuva 4. Ensimmäiset Figma mallit ..... 22
Kuva 5. Figma malli karttanäkymästä. ..... 23
Kuva 6. Figma malli visualisoinnista. ..... 24
Kuva 7. Prototyypit Figma malleista ..... 25
Kuva 8. Sovelluksen sekvenssikaavio. ..... 26
Kuva 9. Npm komento create-react-app React sovelluksen perusrakenteen asentamiseen. ... 26
Kuva 10. Npm komento prop-types asennukseen. ..... 27
Kuva 11. Popup koodi osa. ..... 28
Kuva 12. react-simple-maps esimerkki muokatusta karttamerkistä ..... 29
Kuva 13. Npm komento react-simple-maps asennukseen. ..... 29
Kuva 14. React-simple-maps koodi osa. ..... 30
Kuva 15. Esimerkki Karttanäkymästä LAM-pisteillä. ..... 31
Kuva 16. Karttanäkymä 514 LAM-pisteellä. ..... 32
Kuva 17. Komento asentamaan react-papaparse ..... 33
Kuva 18. For-loop koodi CSV-datan läpikäyntiin. ..... 33
Kuva 19. Npm komento React ApexCharts asennukseen ..... 34
Kuva 20. Nopeudet visualisoituna ApexCharts avulla. ..... 35
Kuva 21. Npm komento React Google charts asennukseen. ..... 35
Kuva 22. Nopeudet visualisoitu Google Charts avulla. ..... 36
Kuva 23. Nopeudet visualisoituina pylväsdiagrammissa. ..... 37
Kuva 24. Nopeuden kellonaikojen näyttäminen. ..... 38
Kuva 25. Pylväsdiagrammisssa nopeupeuden valinna ilmaiseminen värimuutoksen avulla. .... 38
Kuva 26. Välityspalvelimen koodin rakenne ExpressJS. ..... 39

**Taulukot**

Taulukko 1. Vaatimukset sovellukselle ja tietojen visualisoinnille. ..... 8

**Sanasto**

| | |
|---|---|
| ApexCharts | kehikko luomaan interaktiivisia visualisointeja |
| API | Sovellusohjelmointirajapinta |
| CSV | (Comma-separated values) Pilkulla erotetut arvot |
| ExpressJS | NodeJS verkkosovellus kehikko palvelin ohjelmointiin |
| Figma | Käyttöliittymä suunnittelutyökalu |
| Front-end | Web sovelluksen nähtävä puoli |
| FunctionalReq | (Functional requirement) Toiminnallinen vaatimus |
| Github | Versionhalllinta ja yhteistyö alusta koodeille |
| GeoJSON | Muoto maantieteellisten tietorakenteiden koodaamiseen |
| Google Charts | Googlen luoman interaktiivisia kaaviota ja tietotyökaluja |
| LAM | Liikenteen automaattiset mittaustiedot |
| Npm | Node pakettien hallinta |
| NodeJS | JavaScriptin paketti, joka tarjoaa JavaScriptin ajonaikaisen ympäristön |
| nonFunctionalReq | (non Functional requirement) Ei toiminnallinen vaatimus |
| React | JavaScriptin kehikko Front-end ohjemointiin |
| Topojson | GeoJSON laajennus geospatiaalista topologiaa |
| RQ | (Research Question) Tutkimus kysymys |

# 1 Johdanto

## 1.1 Opinnäytetyön tausta

Liikenteestä kerätään paljon tietoa. Suomessa tiedon kerääjänä toimii Fintraffic, joka kerää tietoa Suomen tie-, rautatie- ja vesiliikenteestä. Fintraffic tarjoama avoin ja ajantasainen tieto mahdollistaa sovelluskehityksen ja uusien palveluiden kehittämisen. Liikennejärjestelmän, liikennepalveluiden ja -ratkaisujen kehittäminen tarvitsee koko liikennealan yhteistyötä, joka toteutetaan työryhmissä ja verkostoissa. Uusien palveluiden kehittämistä varten Fintraffic tarjoaa avointa liikennetietoa rajanpintojen kautta. (Liikenteen dataekosysteemi lyhyesti n.d.)

Fintrafficin tieliikenteen kerätyistä tiedoista on kehitetty sovelluksia visualisoimaan liikennemääriä, kuten esimerkiksi karttanäkymä kuvaamaan liikennemääriä palkkien korkeudella ja pylväsdiagrammi havainnollistamaan liikennemääriä suhteessa eri kellonaikoihin. Ennen tämän opinnäytetyön tekemistä tietoa liikennemääristä suhteessa eri ajonopeuksiin ei ole käsitelty ja visualisoitu. Visualisoituina nopeuksista voidaan tunnistaa liikennemäärät ajettujen nopeuksien ja nopeusrajoituksien yhteydessä. Visualisoinnin seurauksena nopeuksista on mahdollista havaita ajetut ylinopeudet. Tulevat käyttäjät, kuten liikenteenvalvonnasta vastaava organisaatio voi käyttää nopeuksien tarkastelua liikenteen valvontaan. Opinnäytetyön toimeksiantajana toimi Jyväskylän ammattikorkeakoulun IT-Instituutti, Tieto tuottamaan- hanke.

Tieto tuottamaan -hankkeen tarkoituksena on tukea Keski-Suomen alueella toimivia yrityksiä digitalisoitumisessa. Hanke auttaa yrityksiä kyberturvallisuuden ja tekoälyn hyödyntämisessä. Tekoälyä tarvitaan enemmissä määrin, kun datamäärät kasvavat ja tarvitaan suuren datamäärän käsittelyä aikaisessa vaiheessa tai tehdä tuotantolinjoilla nopeita päätöksiä. Tieto tuottamaan projektin on rahoittanut Euroopan Unionin aluekehitysrahasto. (Karjalainen & Rantonen 2021.)

## 1.2 Tavoitteet ja tehtävä

Opinnäytetyössä käytetään Fintrafficin keräämää tieliikenteen dataa, joka saadaan Digitrafin palvelusta. Digitrafin data saadaan liikenteen automaattisten mittaustietojen asemista ja se on käsittelemätöntä tietoa. Opinnäytetyön tavoitteena oli visualisoida liikenteen automaattiset mittaustiedot. Opinnäytetyössä tuli selvittää,

- RQ1 – Miten visualisoida liikennemäärät ja nopeudet?
- RQ2 – Miten näyttää LAM-pisteet kartalla?
- RQ3 – Miten visualisoida aika viimeisen 1/3/7 vuorokauden ajalta?

Opinnäytetyön tehtävänä oli kehittää sovellus visualisoimaan liikenteen automaattiset mittaustiedot. Vaatimukset sovelluksen toiminnallisuudelle ja tietojen visualisoinnille esitetään taulukossa 1.

Taulukko 1. Vaatimukset sovellukselle ja tietojen visualisoinnille.

| **Vaatimuksen ID** | **Vaatimuksen kuvaus** |
|---|---|
| FunctionalReqId001 | Digitrafin palvelusta on voitava hakea CSV-raakadata viimeiseltä 1–7 päivältä |
| FunctionalReqId002 | Käyttäjän on voitava vaihtaa visualisointi näkymä näyttämään tiedot 1/3/7 vuorokausilta |
| FunctionalReqId003 | CSV-raakadata on pystyttävä muuttamaan React-kehikolla JSON-formaattiin |
| FunctionalReqId004 | Pystyttävä visualisoimaan käsitelty JSON-data JavaScriptin React-kehikolla |
| FunctionalReqId005 | Sovelluksen pystyttävä käyttämään karttaa |
| FunctionalReqId006 | Datapisteet pystyttävä mallintamaan sovelluksen kartalle |
| FunctionalReqId007 | Datapisteiden pystyttävä valitsemaan sovelluksen kartasta |
| FunctionalReqId008 | Sovelluksen karttaa on pystyttävä zoomaamaan |
| FunctionalReqId009 | Sovelluksen karttaa on pystyttävä liikuttamaan hiirellä vedettäessä |
| FunctionalReqId010 | Datapisteen valitsemalla kartasta pystyy siirtyä näkemään sen datapisteen visualisoitu data |
| FunctionalReqId011 | Visualisoinnilla on pystyttävä näkemään ajoneuvomäärät ja nopeudet |
| FunctionalReqId012 | Pystyttävä valitsemaan nopeus ja nähtävä ajetun nopeuden päivänajat |
| FunctionalReqId013 | Pystyttävä lataamaan kuva ajetun nopeuden päivänajat näkymästä |
| FunctionalReqId014 | CSV-raakadata voitava hakea automaattisesti Digitrafin palvelusta |
| nonFunctionalReqId015 | Visualisoinnista pitäisi nähdä ylinopeudet ja ajoneuvomäärät |
| nonFunctionalReqId016 | Sovellus ei jumittuisi useamman nopeuden valinnassa kellonaikojen tarkastelussa |

## 1.3 Menetelmät

Opinnäytetyö toteutettiin prototyypin kehityksenä. Kehitysprosessissa keskeistä on, että prototyyppi kehityksen eri vaiheissa tuloksena saatuja prototyyppiversioita voidaan parantaa saadun palautteen pohjalta. Prototyyppi kehitystä hyödynnettiin opinnäytetyössä esittämällä prototyypin eri versioita toimeksiantajalle ja ottamalla huomioon toimeksiantajan esittämät ideat ja toiveet kehityksen suunnasta.

Prototyypin kehitys voidaan jakaa kuuteen vaiheeseen. Ensimmäinen vaihe sisälsi ideoiden keräämisen toimeksiantajalta ja sovelluksen vaatimusten määrittämisen. Toisessa vaiheessa suunnittelin suuntaa antavan version prototyypistä ja kolmannessa vaiheessa suunnittelin toimintaa mallintavan prototyypin suunnittelutyökalua käyttäen. Neljännessä ja viidennessä vaiheessa esittelin prototyypin toimeksiantajalle ja palautteen pohjalta prototyyppi viimeisteltiin. Kuudes vaihe sisälsi sovelluksen kehityksen prototyypin ja vaatimusten pohjalta.

# 2 Teoreettiset lähtökohdat

## 2.1 Datavisualisointi

### 2.1.1 Johdatusta visualisointiin

Visualisointi tarkoittaa tiedon jäsentämistä, viestintää tai esittämistä. Data on käsittelemätöntä tietoa ja tarvitsee visualisointia, jotta sitä voidaan ymmärtää paremmin. Visualisoinnin avulla voidaan nähdä asioita, joita ei pystyisi havaitsemaan yhtä helposti ilman datan visualisointia. (DATAVISUALISOINTIOPAS-VISUALISOINTI n.d.)

Visualisointi parantaa mahdollisuuksia analysoida ja tulkita dataa. Se voi avata uusia näkökulmia ja korostaa datan sisältämiä trendejä tai muita ominaisuuksia. Visualisointi voi selkeyttää datan sisältämää tietoa ja tehdä siitä vertailukelpoisemman tai havainnollisemman. (DATAVISUALISOINTIOPAS-VISUALISOINTI n.d.)

### 2.1.2 Mitä on Datavisualisointi

Käsitteellä datavisualisointi tarkoitetaan datan ja informaation esittämistä visuaalisessa muodossa, kuten kaavioina, diagrammeina tai karttoina. Näillä kuvioilla pystytään esittämään monimutkaista dataa selkeämmin. Datan esittäminen kaavioina helpottaa isojen datamäärien, kuten koneoppimisen toiminnan ymmärtämistä. Myös esimerkiksi kun väestön kasvua kuvaava data esitetään kaaviossa tai asuinpaikkoihin liittyvä tieto kartassa, pystyy tavallinen käyttäjä ymmärtämään informaatiota nopeammin kuin tiedon ollessa listana asukkaiden määriä tai paikkojen nimiä. (Brush & Burns 2022.; What is Data Visualization? Definition, Examples, and learning resources n.d.; What is data visualization? n.d.)

Dataa voidaan visualisoida monella eri tavalla riippuen datan tyypistä ja siitä, millä tavalla dataa halutaan esittää. Visualisoinnin eri tapoja ovat esimerkiksi taulukot, sektoridiagrammi, pylväsdiagrammi, viivakaavio, aluekaavio, histogrammi, hajakuvaaja, lämpökartta ja puukartta. Yleisimmin käytettyjä datan visualisoinnin muotoja ovat pylväsdiagrammit. Pylväsdiagrammissa esimerkiksi x-akselilla kuvataan ajoneuvojen nopeudet ja y-akselilla liikennemäärät. Pylväiden korkeudet kuvaavat kyseisellä nopeudella liikkuneiden ajoneuvojen määrää. (Brush & Burns 2022.; What is Data Visualization? Definition, Examples, and learning resources n.d.; What is data visualization? n.d.)

### 2.1.3 Datavisualisoinnin hyvät ja huonot puolet

Datavisualisoinnissa on monia hyviä puolia, mutta myös huonoja puolia. Datavisualisoinnin hyviä puolia ovat havaittavuus ja luettavuus. Oikean kaavion tai diagrammin valinta auttaa datan luettavuutta ja ymmärtämistä paljon. Datan näyttäminen hyvin valituilla väreillä tukee sen havainnollistamista. Kuvaajassa käytetty mitta-asteikko saattaa vaikuttaa kuvaajasta tehtyyn tulkintaan. Jos esimerkiksi todellinen mitta-asteikko alkaa nollasta, mutta kuviossa näytettävä mitta-asteikko jostain muusta luvusta, saattavat nämä kuviot näyttämään erilaisilta. Lisäksi kuvioissa voidaan esittää datasisällön suhteellisia osuuksia. Kaaviossa esitettävän datan tulkintaa tukevan informaation lisääminen kaavioon voi auttaa ymmärtämään sen sisältöä paremmin. (Gupta 2022; What is Data Visualization? Definition, Examples, and learning resources n.d.; What is data visualization? n.d.)

Datavisualisoinnin huonoja puolia ovat mahdollisesti saatavat väärät käsitykset näytetystä datasta. Väärin näytetty data voi olla puolueellista tai epätarkkaa. Esimerkiksi kaavion aloittaminen muusta

luvusta kuin nollasta tai datan näyttäminen vain osittain saa erot näyttämään suuremmilta kuin ne todellisuudessa ovat. Jos visualisoija tekee näin tarkoituksellisesti, hänen voidaan katsoa toimivan puolueellisesti. Toisaalta väärän kaavion tai diagrammin valinta saa datan näyttämään vaikeasti luettavalta tai katsoja ei ymmärrä esitettyä dataa. Myös liian monen värin käyttäminen datan havainnollistamisessa voi vaikeuttaa tiedon luettavuutta. Vääriä värien käyttämisen vuoksi lukija joutuu käyttämään paljon aikaa ymmärtääkseen esitetyn datan tarkoituksen. Jos kuvioon on sisällytetty hyvin paljon dataa, voi lukijalla mennä paljon aikaa tiedon tulkitsemiseen. (Gupta 2022; What is Data Visualization? Definition, Examples, and learning resources n.d.; What is data visualization? n.d.)

Datavisualisoinnin hyvien ja huonojen puolten tiedostaminen ja ymmärtäminen on tärkeää, jotta datavisualisoinnissa päästään paremmin hyvään lopputulokseen. Datan mahdollisimman selkeä esittäminen edistää informaation välittämistä lukijalle selkeämmin ja helpommin.

## 2.2 Sovelluskehityksen elinkaari

Sovelluskehityksen elinkaarta käytetään sovellusten kehittämisessä. Sovelluskehityksen elinkaari jaetaan useaan vaiheeseen. Sovelluskehityksen elinkaari koostuu seuraavista vaiheista: vaatimukset, suunnitelma, suunnittelu, prototyyppien tekeminen, kehitys, testaus, toimitus ja huolto. (Software Development Life Cycle (SDLC) Overview 2022.)

Sovelluksen vaatimusten kartoituksen tarkoituksena on määrittää halutut toiminnallisuudet ja ominaisuudet sovellukselle sekä vaatimukset, joilla halutut toiminnallisuudet voidaan mahdollistaa. Vaatimuksien määrittämisen myötä kehittäjät voivat havaita ongelmia sovelluskehityksen alkuvaiheessa. (Software Development Life Cycle (SDLC) Overview 2022.)

Suunnitelmavaiheen tarkoituksena on sovelluskehityksen suunnitelman tekeminen analysoimalla tarvittavat resurssit, riskit ja kustannukset. Suunnitelmavaiheessa tiimi käy läpi käytettävät teknologiat, ominaisuudet, budjetin ja aikataulun. Tässä sovelluskehityksen vaiheessa kehittäjät voivat ja heidän myös kannattaa tehdä yksinkertainen mutta elinkelpoinen tuote sovelluksesta. (Software Development Life Cycle (SDLC) Overview 2022.)

Suunnitteluvaiheessa keskitytään sovelluksen osien suunnitteluun vaatimuksien perusteella. Suunnittelussa sovellukselle mietitään arkkitehtuuri, käyttöliittymä, turvallisuus ja ohjelmointikielet. Suunnittelu määrittää, miten sovelluksen front-end tulee näyttämään ja toimimaan. (Software Development Life Cycle (SDLC) Overview 2022.)

Prototyyppien tekemisen vaiheen tarkoituksena on luoda prototyyppi, jolla testataan suunnittelua ja sovelluksen käytettävyyttä. Vaiheessa saadaan kuvaa kehityksen etenemisestä ja siitä, onko nykyinen suunnitelma toimiva. Prototyyppivaiheessa voidaan tarvittaessa tehdä muutoksia sovelluksen kehitykseen. (Software Development Life Cycle (SDLC) Overview 2022.)

Kehitysvaiheessa tarkoituksena on luoda prototyyppiversio sovelluksesta suunnitelmien ja prototyyppivaiheen pohjalta. Sovelluksen kehityksessä noudatetaan suunnitelman vaatimuksia. Suunnitelmassa kuvataan tiedostonimikkeet, koodaus ja nimeämistyylit. (Software Development Life Cycle (SDLC) Overview 2022.)

Testausvaiheessa testataan sovelluksen prototyyppiversiota. Testauksilla saadaan esiin sovelluksen puutteet, viat ja mahdolliset kehitysvaiheessa huomaamatta jääneet virheet. Tämä antaa mahdollisuuden virheiden korjaamiselle. (Software Development Life Cycle (SDLC) Overview 2022.)

Toimitusvaiheessa sovellus toimitetaan asiakkaan ehdottamaan käyttöönottoympäristöön. Tässä vaiheessa asiakkaan puolelta voi tulla enemmän toimitusvaatimuksia, kuten julkaisua edeltävän version jakaminen testattavaksi sidosryhmille tai julkaisu suoraan sovelluskauppaan. (Software Development Life Cycle (SDLC) Overview 2022.)

Ylläpitovaiheessa korjataan esiin nousseet ongelmat ja virheet. Ylläpitotiimi vastaa teknisistä tai jälkituotannon ongelmista etsien niihin ratkaisut. (Software Development Life Cycle (SDLC) Overview 2022.)

## 2.3 Käyttöliittymäsuunnittelu sovelluskehityksessä

### 2.3.1 Käyttölittymäsuunnittelun merkityksestä

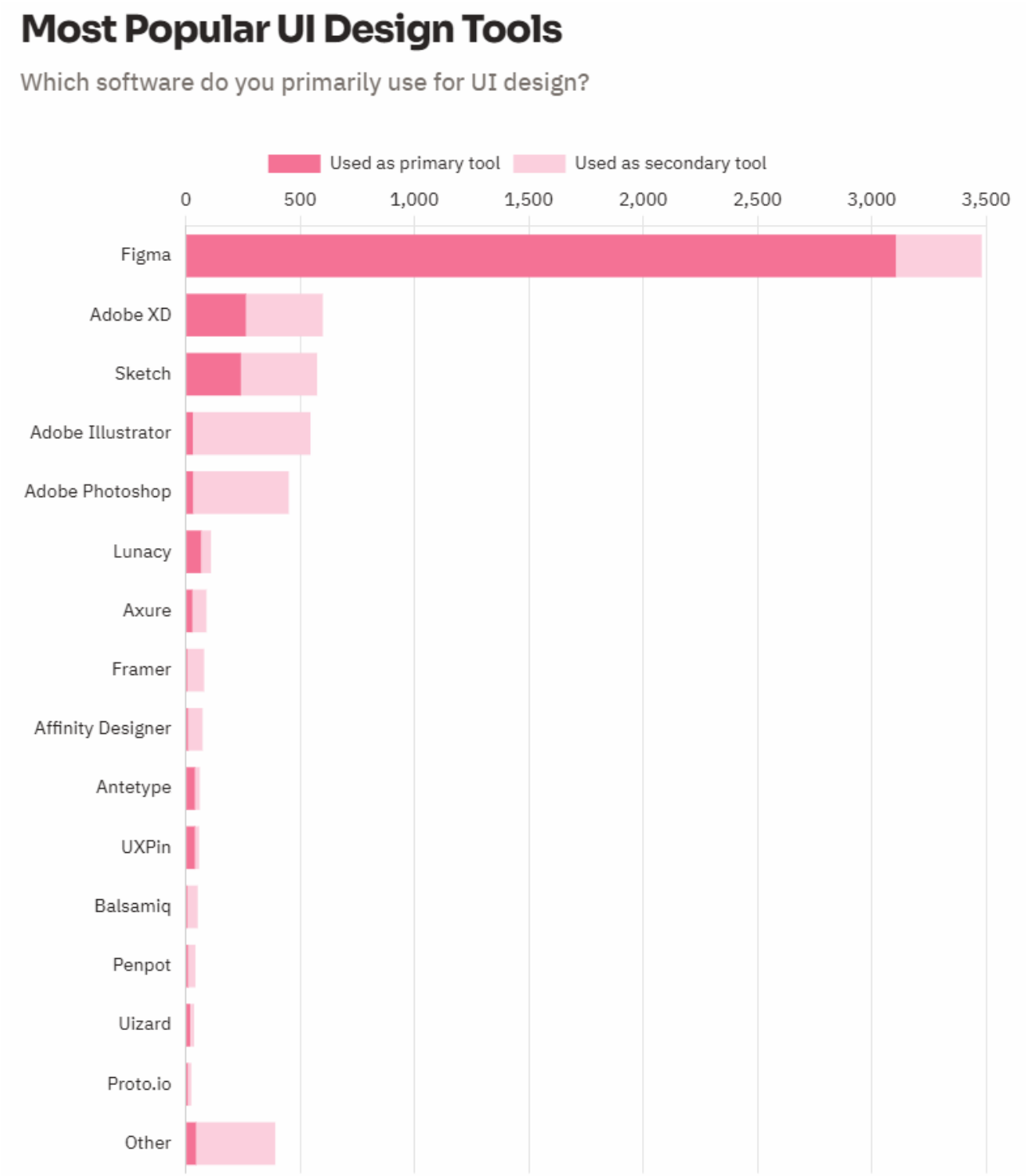
Käyttöliittymäsuunnittelu on rajapintojen rakentamista käyttäjän ja ohjelmiston välille. Käyttöliittymäsuunnittelu pitää sisällään erilaisten käyttöliittymien, kuten graafisten, ääniohjattujen ja elepohjaisten käyttöliittymien, suunnittelua. Lisäämällä käyttöliittymälle visuaalisuutta saadaan se erottumaan muiden joukosta. (User Interface (UI) Design n.d.)

Käyttöliittymäsuunnittelussa on otettava huomioon palvelun käyttäjät monessa asiassa, kuten käyttöliittymän käytettävyydessä, visuaalisuudessa ja monimutkaisuuden vähyydessä. Käyttäjän näkökulmasta toimiva käyttöliittymä on sellainen, jota on helppo, mukava ja vaivatonta käyttää. Tämä lisää käyttäjän halua palata käyttämään palvelua. Käyttöliittymän käyttäjäystävällisyyttä voidaan lisätä tekemällä painikkeet ja muut elementit toimimaan odotetulla tavalla. Käyttöliittymäasettelun tekeminen luettavaksi ja johdonmukaisesti auttaa käyttäjää etenemään käyttöliittymässä. (User Interface (UI) Design n.d.)

### 2.3.2 Käyttöliittymä suunnittelutyökalut

Käyttöliittymän suunnittelutyökalut antavat suunnittelijoille mahdollisuuden tehdä malleja ja prototyyppejä. Suunnittelutyökaluilla voi tuottaa elinkelpoisen tuotteen käyttöliittymän. Käyttöliittymän suunnittelutyökaluilla on eroja toisiinsa verrattuina. (Cardello 2022.)

Vuonna 2022 UX tools kyselyn tulokset on nähtävissä kuviossa 1. Eniten käytetyt suunnittelutyökalut olivat Figma, Adobe XD, Sketch, Adobe Illustrator, Adobe Photoshop ja Lunacy. Kyselyn mukaan Figmaa käytettiin pääasiallisena suunnittelutyökaluna eniten. Toiseksi eniten käytettiin Adobe Illustratoria. (UI Design 2022.)




Kuva 1. UX tools 2022 kysely suunnittelutyökalujen käytöstä

Adobe XD on Adoben kehittämä suunnittelutyökalu, joka toimii Mac- ja Windows käyttöjärjestelmille. Komponenttien uudelleenkäytettävyys ja dynaamisten muutoksien tekeminen on mahdollista. Komponenttien kokoa voi muuttaa responsiivisesti. Suunnittelu tiedostojen osien jakaminen

on mahdollista tiimin kesken jaettavien kehikoiden avulla. Prototyyppien tekeminen animaatioiden kanssa mahdollista. Adobe XD mahdollistaa prototyypin tekemisen äänivuorovaikutuksen kanssa. (Rae 2020.)

Sketch on Mac-käyttöjärjestelmälle kehitetty suunnittelutyökalu. Sketch on vektoripohjainen suunnittelutyökalu mahdollistaen monipuolisen vektorien muokkauksen. Suunnittelutaidetaulujen muuttaminen malleiksi ja mallien uudelleenkäyttö on mahdollista. Tehtyä komponenttia voi käyttää useissa paikoissa ja tiettyyn komponenttiin tehdyt muutokset siirtyvät saman komponentin uudelleen käytettyihin versioihin. Tyylit on mahdollista määrittää muuttujiin ja uudelleen käyttää tiedostossa. Prototyypit on mahdollista tehdä niitä varten olevilla työkaluilla. Yhteistyön tekeminen samassa tiedostossa on mahdollista. Sketch suunnittelutiedostot ovat yhteensopivia monien muiden työkalujen kanssa. (Make something great n.d.)

Adobe Illustrator on Adoben kehittämä vektoripohjainen piirtämistyökalu. Illustratorin ominaisuuksia ovat piirtotyökalut, edistyneet värivaihtoehdot, kerroksittainen työskentely, ruudukkoominaisuus, tiedostojen tallentaminen pilvipalvelimille, monipuolinen fonttikirjasto ja työkalut geometrian tekemiseen ja muokkaamiseen. Illustratoria käytetään moneen eri tehtävään, kuten logojen, ikonien, web grafiikan, digitaalisen taiteen ja käyttöliittymäsuunnittelun tekemiseen. Illustratorin monipuolisuus tekee siitä paljon käytetyn työkalun. (Mitchell n.d.)

Käyttöliittymäsuunnittelussa käytettäviä työkaluja on monenlaisia ja useilla työkaluilla pääsee samankaltaiseen lopputulokseen. Opinnäytetyön käyttöliittymä suunnittelutyökaluksi valittiin Figma. Figman valintaan vaikutti työkalun monipuolinen käytettävyys ja sen useat toiminnot. Lisäksi minulla oli aiempaa kokemusta Figman käytöstä. Figmasta kerrotaan tarkemmin luvussa 2.5.

## 2.4 Kehitysvaihe sovelluskehityksessä

Sovelluskehityksessä kehitysvaihe on sovelluksen kehittämistä suunnitteluvaiheen ja prototyyppivaiheen pohjalta. Kehitysvaiheessa hyödynnetään suunnitteluvaiheen ja prototyppivaiheen malleja sovelluksen kehittämiseen. Web sovelluskehityksessä voidaan hyödyntää samoja sovelluskehityksen vaiheita.

Web sovelluskehityksessä sovelluksen kehittäminen tehdään web ohjelmointikielillä, joita ovat esimerkiksi HTML, CSS ja JavaScript ohjelmointikielet. Front-end eli web sovelluksen nähtävä osio pohjautuu suunnitteluvaiheen malleihin. Front-end ohjelmoinnille on erilaisia JavaScript kehikkoja, kuten React ja Vue. (Johnston 2020.)

React on suosittu ja tehokas Front-end JavaScriptin kehikko (Johnston 2020). React sovellukset tehdään komponenteista (Create user interfaces from components. n.d.). Komponentti on osa käyttöliittymää, joilla on oma logiikka ja ulkonäkö. Komponentti voi olla pelkkä nappi tai kokonainen sivu. React komponentit ovat JavaScript toimintoja. (Quick Start. n.d.) React komponenteille voi antaa dataa ja komponentit päivittyvät näyttämään annetun datan (Create user interfaces from components. n.d.).

Vue on JavaScriptin kehikko käyttöliittymän rakentamiseen. Vue antaa komponenttipohjaisen ohjelmointi mallin tehokkaaseen käyttöliittymän tekemiseen. Vue antaa mahdollisuuden kuvata HTML tulosteen deklaratiivisesti JavaScriptin tilan perusteella. Vue päivittää automaattisesti tapahtumat, kun havaitsee muutoksia JavaScriptin tilassa. (Introduction n.d.)

Front-end ohjelmoinnille on monia JavaScriptin kehikkoja. Opinnäytetyön toteutuksen kehitykseen valittiin React aiemman käytön ja kokemuksen perusteella. Reactin käyttämistä kuvataan tarkemmin luvussa 3.3.

## 2.5 Figma työkalu käyttöliittymäsuunnittelussa

### 2.5.1 Mikä on Figma?

Figma on selainpohjainen suunnittelualusta, jossa voidaan tehdä interaktiivisia prototyyppejä web-sivuista, web-sovelluksista, mobiilisovelluksista tai monesta muusta sovelluksesta. Figman kehityksen tarkoituksena on ollut saada aikaan tehokas suunnittelutyökalu, jossa on mahdollista työskennellä samassa suunnittelutiedostossa paikasta riippumatta. (Creative tools meet the internet n.d.) Hurwitz (2022) mukaan Figman omistajuus vaihtui, kun Adobe osti Figman vuoden 2022 lopussa (Hurwitz 2022).

Figman staattiset suunnittelutiedostot voidaan muokata interaktiivisiksi prototyypeiksi mallintamaan toiminnallisuutta. Suunnittelutiedossa voidaan helposti yhdistää käyttöliittymäelementtejä ja valita niiden välisiä tapahtumia ja/tai animaatioita. Tapahtumia voivat olla esimerkiksi hiiren vieminen elementin päälle, hiiren klikkaus tai näppäimen painallus. Figma yhdistää suunnittelun, prototyypin, jakamisen ja palautteen keräämisen yhteen paikkaan. (Prototyping features n.d.) Figman toimii selaimessa pilvipalvelun avulla mahdollistaen työskentelyn monella eri käyttöjärjestelmällä (UI design tool n.d.).

### 2.5.2 Suunnittelutyökalun toimintaperiaate

Clementen (2022) mukaan Figma on yksi suosituimmista ja käytetyimmistä suunnittelutyökaluista web-sivujen prototyyppien tekemiseen. Clemente korostaa, että Figma on hyvä prototyyppi työkalu, koska se on helppokäyttöinen ja siinä on paljon ominaisuuksia, jotka tekevät prosessista nopean ja yksinkertaisen. (Clemente 2022.)

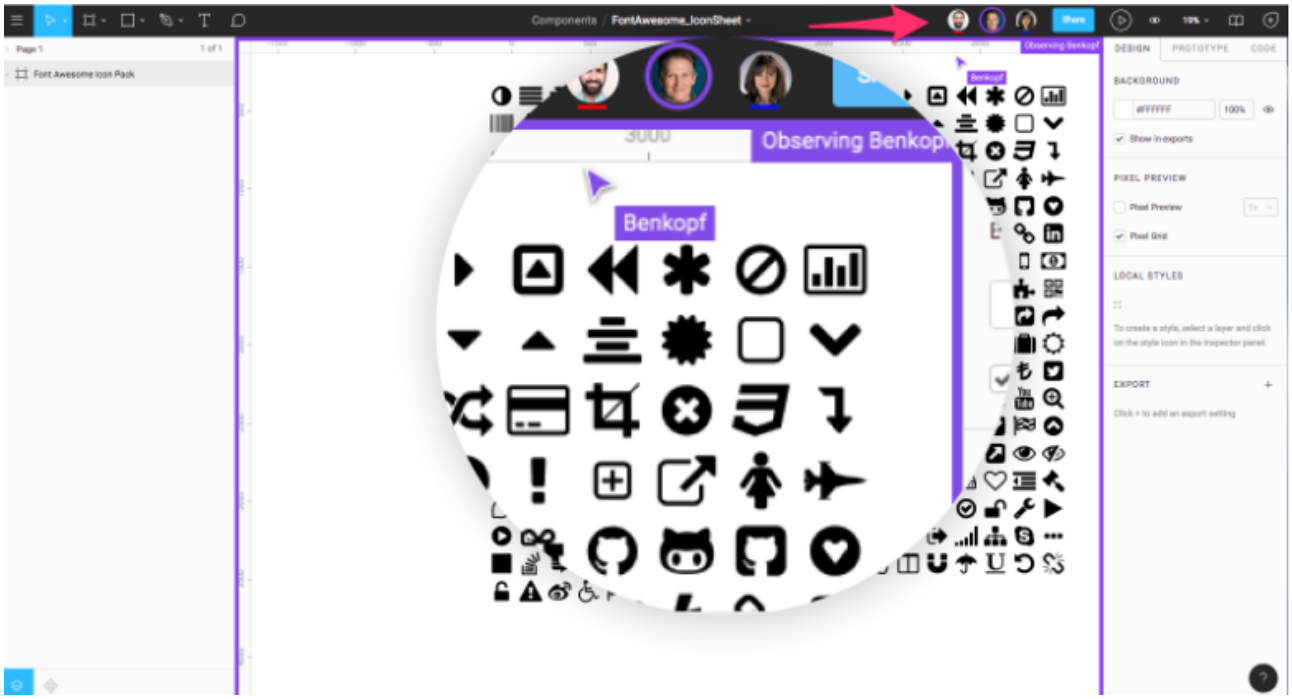
Clemente (2022) toteaa, että ”protoyypisivuston tekeminen Figmalla on helppoa ja vaatii vain kuusi vaihetta: uuden tiedoston luominen; kehysten lisääminen; sisällön lisääminen; vuorovaikutteisuuden lisääminen; testaus; ja julkaiseminen”. Hän ohjeistaa ensimmäiseksi luomaan uuden Figma tiedoston valitsemalla valmiin mallipohjan tai aloittamalla tyhjästä pohjasta. Toisessa vaiheessa Figma tiedostoon lisätään kehys, joka on sisällön säiliö, ja tämän jälkeen kolmantena lisätään kehyksen sisään sisältöä esimerkiksi otsikko ja otsikolle tekstiä. Clemente lisää, että jälkeenpäin voidaan vaihtaa tekstin fonttia, kokoa ja väriä. Neljäntenä vaiheena on interaktiivisuuden lisääminen, kun kehyksessä on tarpeeksi sisältöä. Vuorovaikusasetuksien lisääminen ja muokkaaminen mahdollistaa esimerkiksi elementtien klikkaamisen ja sivujen välisen navigoinnin. Viides vaihe on web-sivun prototyypin testaaminen, joka tapahtuu klikkaamalla esikatselukuvaketta, ja kuudes prototyypin julkaiseminen katselulinkin avulla. (Clemente 2022.)

### 2.5.3 Suunnittelutyökalun hyödyt

Figman hyöty on sekä Figman UI design tool sivun (n.d.) että Kopfin (n.d.) mukaan se, että Figma toimii kaikilla käyttöjärjestelmillä verkkoselaimessa (Kopf n.d.; UI design tool n.d.). Kopf tarkentaa, että Figma on tyypiltään yksi harvoista suunnittelutyökaluista, jota useat käyttäjät pystyvät käyttämään samaan aikaan riippumatta käyttöjärjestelmästä. Yhteistyön tekee helpoksi sen, että Figma

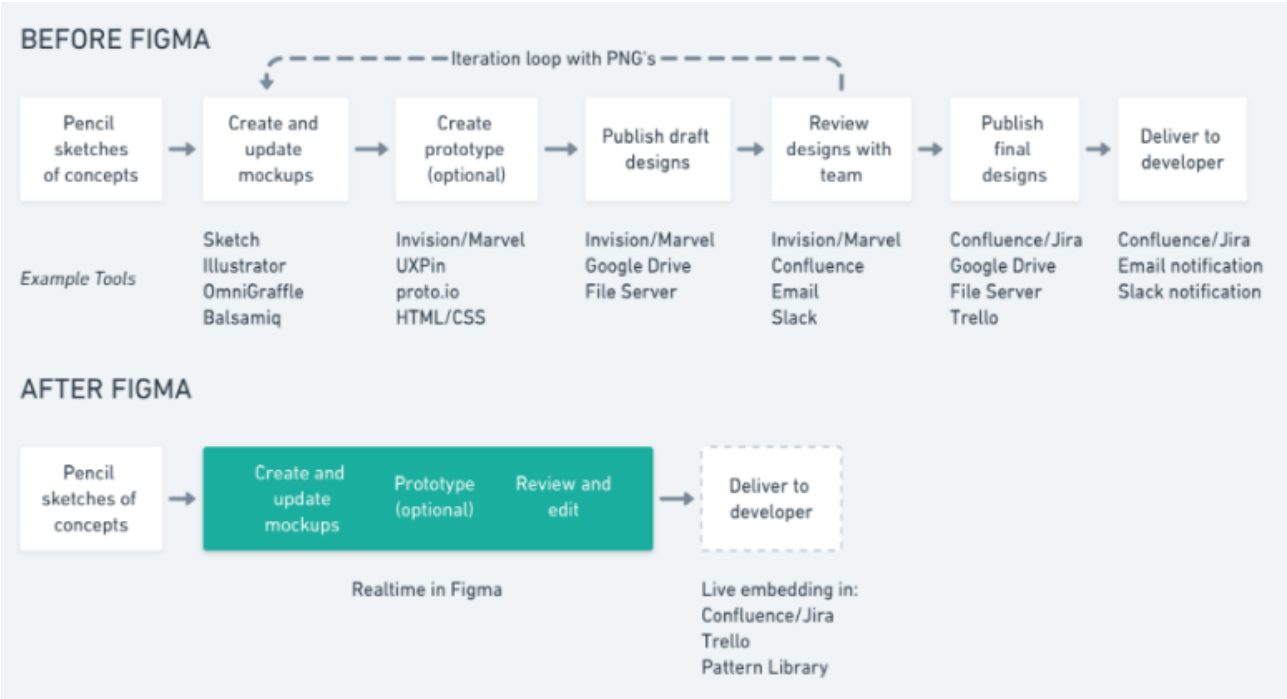
on selainpohjainen. (Kopf n.d.). Tietoja ei tarvitse erikseen tallentaa, koska tiedostot tallennetaan automaattisesti versiohistorian kanssa Figman pilvipalvelimille (Kopf n.d.; UI design tool n.d.).

Kuviossa 2 nähdään esimerkki Figman reaaliaikaisesta tapahtumasta. Yläpalkissa on nähtävissä, ketkä ovat reaaliaikaisesti työstämässä suunnittelutiedostoa. Hiiren liike ja paikka ilmaisee, mitä osaa suunnittelutiedostosta toinen henkilö työstää parhaillaan. Yläpalkissa näkyvää henkilön kuvaa klikkaamalla kuvakulma siirtyy kyseisen henkilön hiiren paikkaan. Kopf (Kopf n.d.) mainitsee, että tämä reaaliaikainen toisten toimintojen näkeminen auttaa tiimejä vähentämään suunnittelun ajautumista väärään suuntaa, koska toimintaa pystyy tarkistamaan ja korjaamaan. (Kopf n.d.)




Kuva 2. Figma reaaliaikaiset tapahtumat

Kuviossa 3 havainnollistetaan suunnitteluprosessia ennen ja jälkeen Figman käyttöä. Kopf kertoo väärään suuntaan ajautumisen tapahtuvan helpommin suunnittelutyökaluilla, joissa työskentely on ei-reaaliaikaista. Tällöin suunnittelujohto ei pysty yhtä helposti tarkistamaan suunnittelun suuntaa. (Kopf n.d.)




Kuva 3. Suunnittelu prosessi ennen ja jälkeen Figman

Figma antaa mahdollisuuden käyttöoikeuksiin perustuvan tiedostojen, sivujen ja kehyksien linkkien jakamisen. Linkkiä klikkaamalla avautuu Figman selainversio määritetyn linkin tiedoston näkymästä. Figma mahdollistaa reaaliaikaisen upotuskoodinpätkän liittämisen kolmannen osapuolen työkaluihin. Kopf tarkentaa, että Figma tiedostoon tehdyt muokkaukset näkyvät upotetussa Figma tiedostossa. (Kopf n.d.)

Kopf (n.d.) mainitsee palautteen antamisen olevan helppoa, sillä Figmassa on sisäänrakennettuna kommentointimahdollisuus sekä suunnittelu- että prototyyppitiloihin. Nämä kommentit voidaan viedä Slackiin tai sähköpostiin. Kopf mainitsee, että suunnittelutarkastuksen aikana tiimi voi keskustella työstä, lukea kommentteja ja korjata epäkohtia tehden kaiken Figmassa. (Kopf n.d.)

## 2.6 Tieliikenne

### 2.6.1 Fintrafficin tieliikennetiedot

Tieliikennetietoa kerätään Fintrafficin hallinnoimissa tieliikenteen ohjaus- ja mittausjärjestelmissä. Tieliikenteestä on tarjolla avointa tietoa useassa eri muodossa, kuten kelikamerakuvat, tiesääase-

matiedot, tiejaksojen keliennusteet, liikenteen automaattiset mittaustiedot (LAM), liikennetiedotteet, TMC/ALERT-C paikannuspisteistö, muuttuvien liikennemerkkien tiedot, maanteiden kunnossapitotiedot, jalankulun ja pyöräilyn mittaustiedot ja metatiedot. Tieliikenteen avoimia tietoja mahdollista hakea ja käyttää Digitrafficin rajapintojen kautta API kutsuilla. (Tieliikenne n.d.)

Keli- ja liikennekameroiden kuvista saadaan tietoa tienpinnan tilasta ja liikennetilanteesta. Rajapinnan kautta on mahdollista hakea kaikkien julkisten kelikameroiden tietoja ja osoitelinkit, mistä kelikameroiden kuvat löytyvät. Fintrafficin sivujen mukaan kelikameroita on käytössä noin 470 kappaletta. (Tieliikenne n.d.)

Tiesääasemat mittaavat säätietoja, kuten ilman lämpötilan, suhteellisen kosteuden, kastepistelämpötilan, sateen ja tuulitiedot. Tietoa tienpinnan tilasta mitataan myös erityisten tienpinta-anturien avulla. Rajapinnan kautta on mahdollista hakea tiesääasemien avoimia mittaustietoja, jotka päivittyvät minuutin välein. Fintrafficin sivujen mukaan Suomen maanteillä on yli 350 tiesääasemapistettä, jotka sijaitsevat pääteiden varsilla. Teiden osalta saadaan myös Tiejaksojen keliennusteet, jotka päivittyvät viiden minuutin välein. (Tieliikenne n.d.)

Liikenteen automaattiset mittaustiedot (LAM) mittaavat liikennemääriä ja nopeuksia ajoneuvoluokittain. LAM-asema toimii upotetun induktiosilmukan avulla. Asemista saadaan tietoa rajapinnan avulla. (Tieliikenne n.d.) Liikenteen automaattisista mittaustiedoista kerrotaan tarkemmin luvussa 2.6.2.

Liikennetiedotteita voi hakea rajapinnan kautta kahdella eri formaatilla. Formaatit ovat DATEX II -standardi ja simppeli JSON. Liikennetiedotteiden tyyppejä ovat erikoiskuljetus, tietyö, liikennetiedote ja painorajoitus. Lisäksi liikennetiedotteet käyttävät TMC/ALERT-C paikannuspisteistöä häiriöiden maantieteellisen sijainnin ilmoittamisessa. (Tieliikenne n.d.)

Muuttuvien liikennemerkkien tiedot voi hakea rajapinnan kautta ja rajapinnan viesti ilmoittaa muuttuvan liikennemerkin viimeisimmän tilan. Muuttuvat liikennemerkit ovat nopeusrajoituksia sekä varoitusmerkkejä. (Tieliikenne n.d.)

Maanteiden kunnossapitotiedot voi hakea rajapinnan kautta. Väylävirasto vastaanottaa kunnossapitotiedot kunnossapitourakoitsijoilta ja myös Digitraffic vastaanottaa samat tiedot. Kunnossapitotiedot ovat ajoneuvojen tehtävätyypit ja ajoneuvojen seurantadata. Ajantasaiset kunnossapitotiedot saadaan Digitrafficin rajapinnan kautta, mutta vuotokautta vanhemmat tiedot saadaan väyläviraston avoimen datan palveluista. (Tieliikenne n.d.)

Jalankulun ja pyöräilyn mittaustietoa voi hakea vain Oulun alueelta rajapinnan kautta. Mittauspisteiden kaikki tieto on GeoJSON-muodossa ja dataa on mahdollista saada JSON- ja CSV-muodossa. (Tieliikenne n.d.)

### 2.6.2 Digitraffic LAM-data

Digitraffic on Fintrafficin palvelu, josta on saatavissa ajantasaista liikennetietoa Suomen tieverkolta, ratatieliikenteestä ja meriliikenteestä (Palvelun esittely n.d.). ”Liikennetiedot ovat avointa dataa, jota jaetaan avointen rajapintojen kautta” (Palvelun esittely n.d.). Avointa dataa on LAM-raakadata ja LAM-raportit. LAM tarkoittaa liikenteen automaattisia mittaustietoja (LAM-tiedot n.d.). LAM-raportit ovat Excel-tiedosto muodossa käsiteltynä tietona ja sisältävät käyttöliittymästä valitut tiedot ja tarkastelutason. LAM-raakadata on puolestaan käsittelemätöntä tietoa ja saatavissa CSV-tiedostoina valitun LAM-pisteen ja päivän mukaan. (Digitraffic avoin LAM-data n.d.)

Mittausasemat keräävät tietoa tieliikenteestä ympäri Suomea. LAM-datan avulla voidaan seurata esimerkiksi liikennemääriä ja ajonopeuksia. LAM-dataa on Digitrafficin sivuilla tarjolla mittauspisteiden reaaliaika- ja historiadata muodossa. Historiadataa on saatavissa LAM-raportteina ja LAM-raakadatana. LAM-raportit sisältävät tarkimmillaan tuntitason summatietoja ja tiedot ovat tarkastettuja ja tarvittaessa korjattuja. LAM-raakadata sisältää yksittäisistä ajoneuvo-ohituksista kerättyä dataa, joka on purettu CSV-muotoiseksi mutta ei ole käsitelty muuten. Raakadata sisältää LAM-pistetunnuksen, vuoden, päivän järjestysnumeron, tunnin, minuutin, sekuntin, sadasosasekuntin, pituus metreinä, kaistan, suunnan, ajoneuvoluokan, nopeuden kilometreinä tunnissa, tiedon virheellisyyden, kokonaisajan, aikavälin ja jononalun. (LAM-tiedot n.d.)

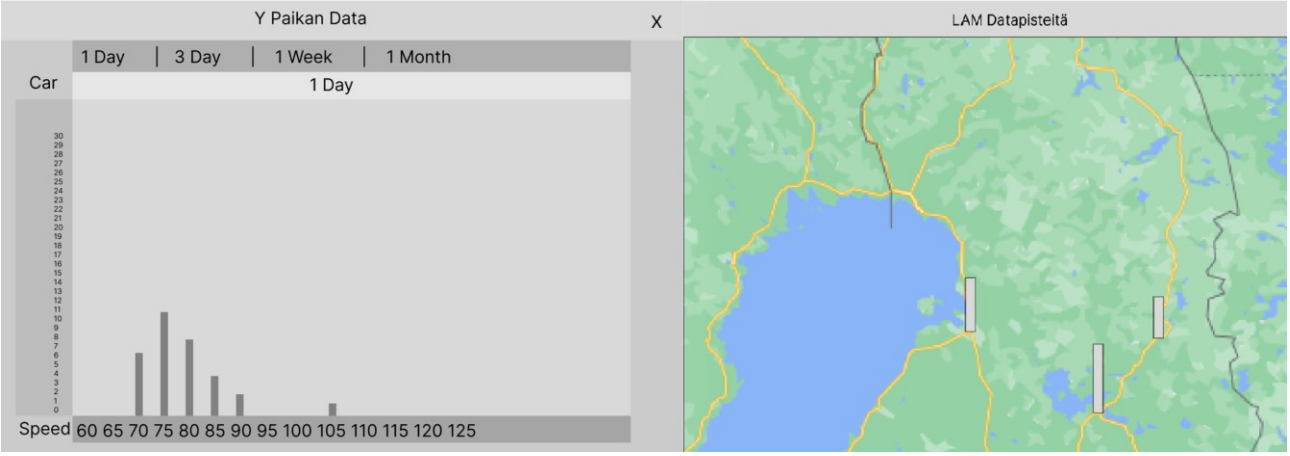
LAM-datan kerääminen tapahtuu LAM-laitteen sähkömagneettisen induktion avulla. Auton metallinen massa aiheuttaa magneettikentän muutoksen päällysteen sisälle upotetussa silmukassa. Sil-

mukoita LAM-pisteillä on kahdessa kohdassa, jotta toisesta pystytään ottamaan aloitusaika ja toisesta lopetusaika. Kummallakin kaistalla on omat induktiosilmukkansa ja tiedonkeruuyksikkönsä. Tiedonkeruu tapahtuu, kun ”LAM-laite rekisteröi pisteen ylittävät ajoneuvot, jolloin jokaisesta ajoneuvosta saadaan ohituksen kellonaika, ajosuunta, ajokaista, ajonopeus, ajoneuvon pituus, peräkkäisten ajoneuvojen aikaero ja ajoneuvoluokka”. (LAM-tiedot n.d.)

# 3 Prototyypin Toteutus

## 3.1 Suunnitteluvaihe

Prototyyppikehitys alkoi ideoiden keräämisenä toimeksiantajalta ja sovelluksen vaatimuksien määrittämisellä. Vaatimuksien pohjalta kartoitettiin työkalut ja ohjelmointikielet sovelluksen tekemiseen. Kartoituksen jälkeen sovelluskehitys jatkui suunnitteluvaiheeseen. Suunnitteluvaihe alkoi annettujen vaatimusten ja tavoitteiden pohjalta tekemällä ensimmäiset Figma mallien esimerkit (kuvio 4).

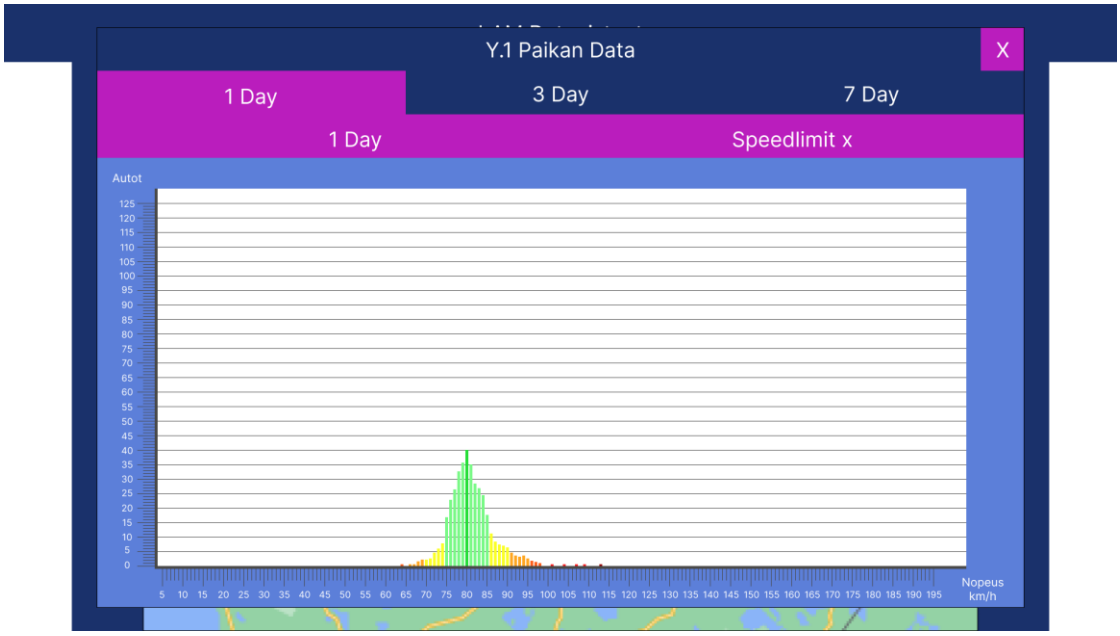



Kuva 4. Ensimmäiset Figma mallit

Figma mallien esimerkit näytettiin toimeksiantajalle ja toimeksiantajan kommenttien pohjalta parannettiin Figma malleja. Vaatimuksena oli, että sovellukseen pitäisi lisätä kartta (FunctionalReqId005) ja datapisteitä lisättävä sovelluksen kartalle (FunctionalReqId006). Näiden mukaan tehtiin Figmalla parannettu aloituskarttanäkymä LAM-pisteistä kartalla. Kuviossa 5 esitetään Figmassa suunniteltu aloitusnäkymä kartasta.

Kuva 5. Figma malli karttanäkymästä.

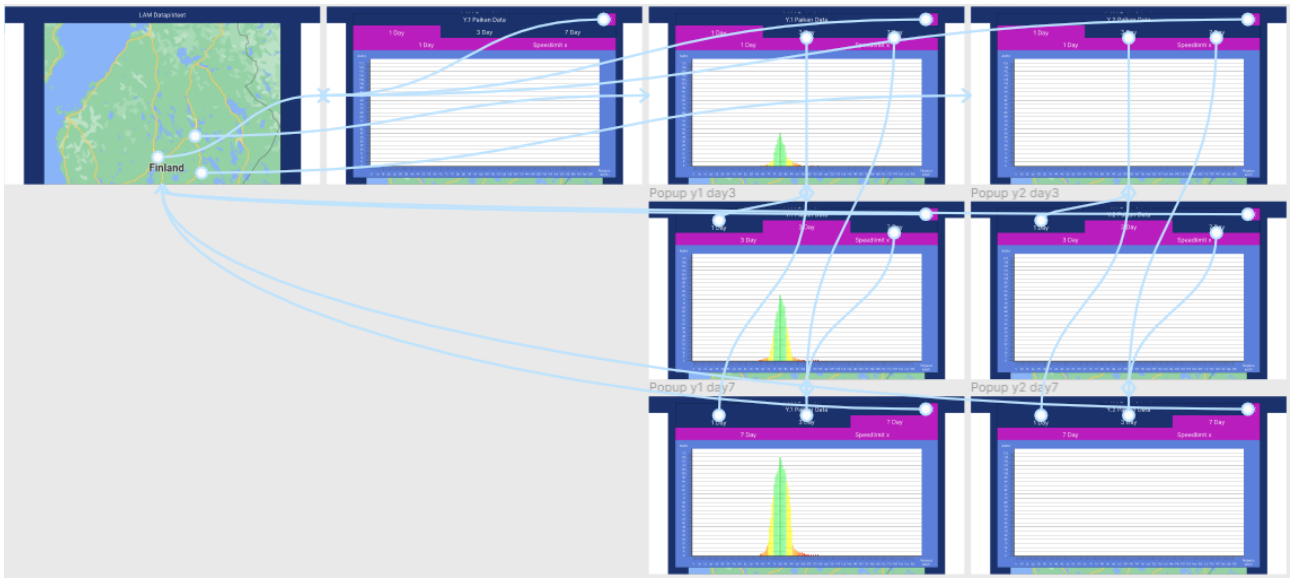
Karttanäkymän jälkeen vaatimuksena oli, että käyttäjä pystyy vaihtamaan visualisointinäkymää näyttämään tietoja yhden, kolmen ja seitsemän vuorokauden ajalta (FunctionalReqId002). Lisäksi visualisoinnilla oli tarkoitus saada näkymään ajoneuvomäärät ja nopeudet (FunctionalReqId011). Näiden mukaan tehtiin Figmassa nopeuksien ja liikennemäärien visualisointinäkymä pylväsdiagrammin muodossa (kuvio 6).




Kuva 6. Figma malli visualisoinnista.

## 3.2 Prototyyppivaihe

Figma mallien tekemisen jälkeen siirryttiin prototyyppivaiheeseen. Figma mallit muutettiin suunnittelutyökalussa prototyypeiksi, joissa testattiin näkymien vaihtumista ja päivien valintaa. Karttamerkkipainikkeita klikkaamalla karttanäkymästä päästään siirtymään visualisointinäkymään (kuviossa 7). ’Päivä’ painikkeilla siirrytään päivien välillä ja visualisointinäkymän ruksi-painikkeella pääsee palaamaan takaisin karttanäkymään. Prototyypit näytettiin toimeksiantajalle ja toimeksiantajan kommenttien pohjalta siirryttiin kehitysvaiheeseen tekemään visualisointi sovellus React-kehikoilla.

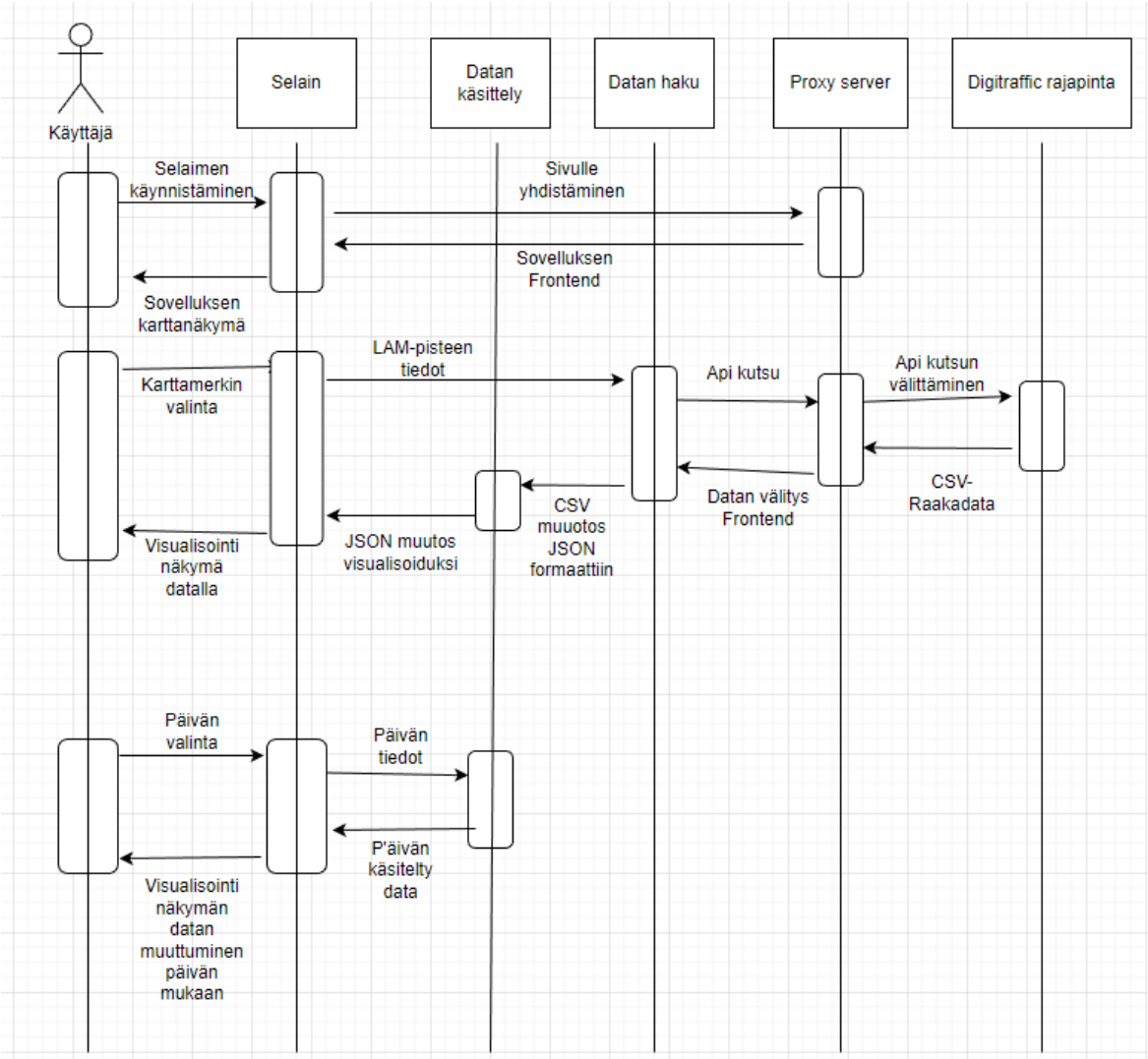



Kuva 7. Prototyypit Figma malleista

## 3.3 Kehitysvaihe

### 3.3.1 Kehitysvaiheen aloitus

Kehitysvaihe alkoi jatkaen suunnitteluvaiheen, prototyyppien ja kuvion 8 sekvenssikaavion pohjalta. Sekvenssikaavio kuvaa sovelluksen toimintaa pääpiirteissään.




Kuva 8. Sovelluksen sekvenssikaavio.

Kehityksen alkuvaiheessa tarvittiin työkaluja web sovelluskehityksen aloittamiseen. Työkaluja ja sovelluksia ovat Windowsin komentokehote, npm eli node pakettien hallinta, React-kehikko ja Visual studio code. Web-sovelluksen tekeminen alkoi komentokehotteessa create-react-app komentoa käyttämällä luoden pohjan React web-sivulle. Komento kuviossa 9 luo tarvittavat tiedostot ja tiedosto rakenteet React ohjelmointia varten.

```
npx create-react-app lam-visualization
```

Kuva 9. Npm komento create-react-app React sovelluksen perusrakenteen asentamiseen.

Tämän jälkeen tein CSS tiedostoon luokkia Figma näkymien perusteella. Seuraavaksi tein Reactia käyttäen web-sivun rakenteen pohjan karttanäkymälle. Näkymä pylväsdiagrammia varten alkoi asentamalla prop-types React paketin npm avulla kuvion 10 mukaan komentokehotteessa. Prop-types auttoi pylväsdiagrammi ponnahdusikkuna näkymän toiminnon tekemisen.

```
npm install prop-types --save
```

Kuva 10. Npm komento prop-types asennukseen.

React:a ja prop-types komponenttia käyttäen tein ponnahdusikkunan omaan tiedostoonsa, josta sitä voidaan kutsua. Ponnahdusikkuna komponentti tehtiin kuvion 11 näyttämällä tavalla.

```
import PropTypes from "prop-types"; //prop-types import
const CustomPopup = (props) => {
    const [show, setShow] = useState(false); //State declaration for popup show

    //function to close popup
    const closeHandler = (e) => {
        setShow(false);
        props.onClose(false);
    }

    useEffect(() => {
        setShow(props.show);
    }, [props.show]);

    return (
        <div
            style={{
                visibility: show ? "visible" : "hidden", //boolean toggle for visible state
                opacity: show ? "1" : "0" //boolean toggle for opacity
            }}
            className="Overlay" //set class for div as Overlay
        >
                <div className="Popup" //set class for div as Popup
                >
                <div className="Popup-header">
                    <h1>{props.title}</h1>
                    <span className="Close" onClick={closeHandler}>
                        &times;
                    </span>
                </div>
                <div className="Popup-header">
                    <h2>Day 1</h2>
                </div>
                <div className="Popup-main" /* Main content area */ >{props.children}</div>
            </div>
        </div>
    );
};

CustomPopup.propTypes = {
    title: PropTypes.string.isRequired,
    show: PropTypes.bool.isRequired,
    onClose: PropTypes.func.isRequired
};
```

Kuva 11. Popup koodi osa.

### 3.3.2 Datapisteet kartalle

Kartan mallintamisen tapoja on monenlaisia, kuten Google Maps hyödyntämistä ja vektorigrafiikkaan eli SVG pohjautuvia karttanäkymiä. Vektorigrafiikkaan pohjautuvista react-simple-maps on tämänlainen karttanäkymä. Simple maps hyödyntää toposjon tai geosjon vektorigrafiikka karttatiedostoja ja niiden pohjalta mallintaa kartan verkkosivulle. (Getting started n.d.) Simple maps kartat eivät ole rajoittuneet tiettyihin karttatiedostoihin ja simple maps käyttääkin SVG karttatiedostoja, joita on saatavilla Githubista (Map files n.d.). Kartta projektoidaan SVG-tiedostosta ja kartalle voidaan antaa erilaisia tyylejä (Projections n.d.; Styling n.d.). Datapisteen näyttämisen kartalle saa-

daan simple maps Marker komponentin avulla (Marker n.d.). Kartta merkille annetaan koordinaatit ja sen ulkonäön pystyy muokkaamaan haluamakseen kuviossa 12 (Marker n.d.). Simple maps tarjoaa vielä zoomaus ominaisuuden kartalle (Zoomable group n.d.).

Kuva 12. react-simple-maps esimerkki muokatusta karttamerkistä

Vaatimuksen sovelluksen pystyttävä käyttää karttaa (FunctionalReqid005) mukaan tehtiin sovellukseen kartta react-simple-maps avulla. Npm avulla kuvio 13 komentokehotteessa asennettiin react-simple-maps React paketti.

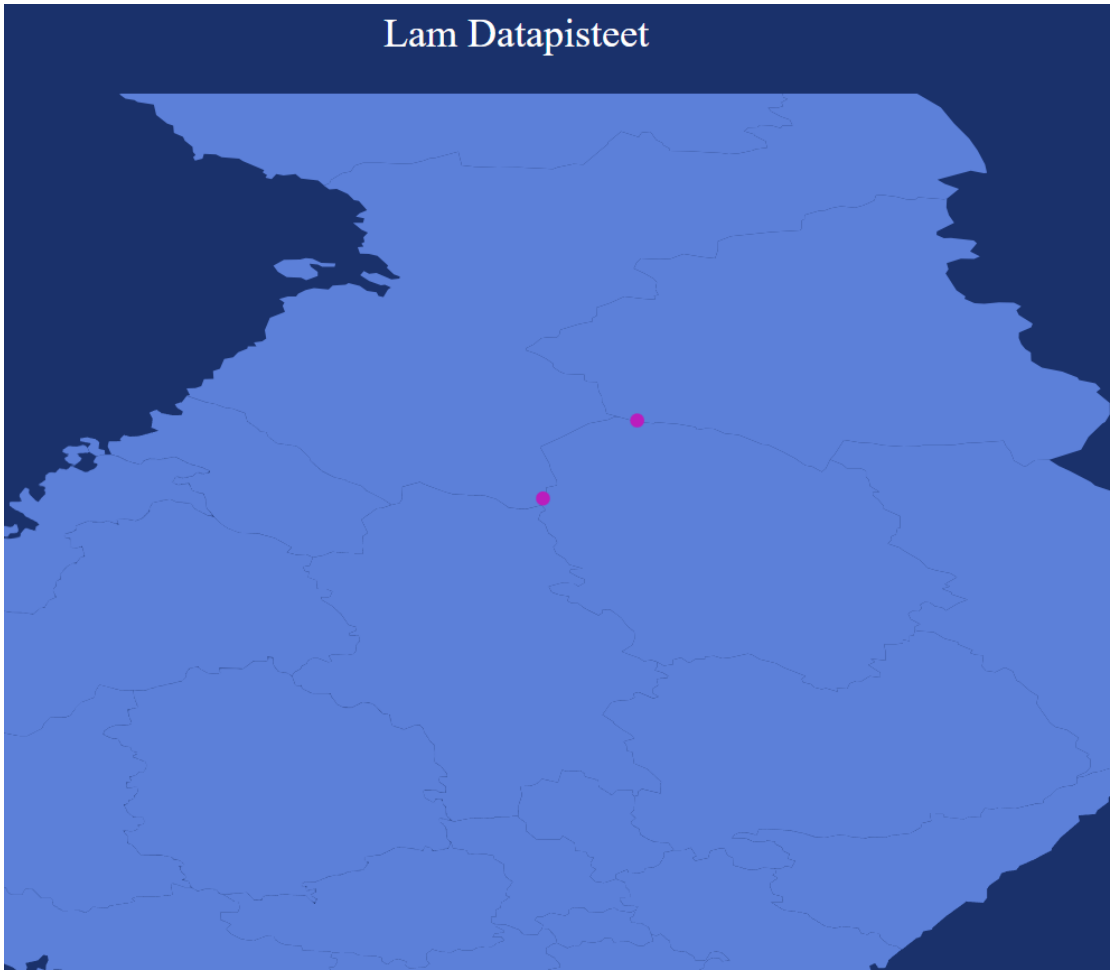
```
npm install --save react-simple-maps
```

Kuva 13. Npm komento react-simple-maps asennukseen.

Karttanäkymää varten etsin kartan GitHubista. GitHubissa on monia eri karttoja jaossa, ja opinnäytetyötä varten etsin Suomi kartan SVG-tiedostomuodossa. Kartan voi lisätä URL muodossa tai ladattuna tiedostona. Kartan näyttämistä varten tarvitaan react-simple-maps komponentin osia projektoimaan SVG kartta. Vaatimuksena oli, että sovelluksen karttaa on pystyttävä zoomaamaan (FunctionalReqid008), jonka mukaan kartalle lisättiin zoomausominaisuus ja tämän jälkeen kartan sijainnin pystyy helpommin keskittämään ruudulle, muuttamaan sen kokoa ja lisäämään väriä. Vaatimuksena oli, että sovelluksen karttaa on pystyttävä liikuttamaan hiirellä vedettäessä (FunctionalReqid009), jonka määrittämä toiminto on react-simple-maps komponentissa valmiina toimintona, joten sitä ei tarvitse erikseen lisätä. Vaatimuksena oli, että datapisteet pystyttävä mallintamaan sovelluksen kartalle (FunctionalReqid006), jonka mukaan lisätään merkkejä käyttöliittymän kartalle näyttämään LAM-pisteet. Kartan koodin rakenne esitetään kuviossa 14 ja esimerkki kartasta LAM-pisteillä kuviossa 15.

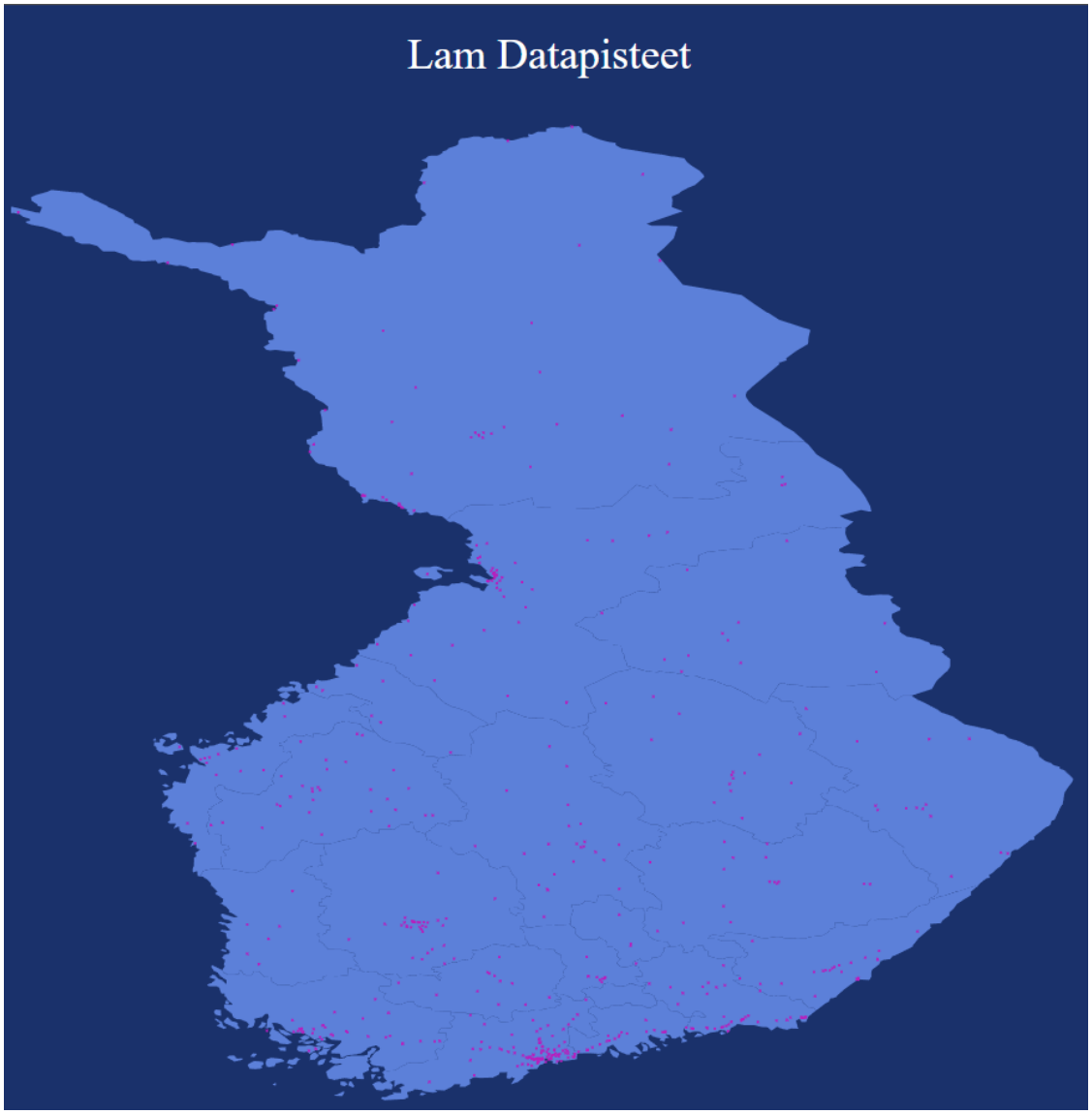
```
<ComposableMap // Component to render SVG map
width="24" //map width
height="23" //map height
>
  <ZoomableGroup // Zoom component for map
    zoom={position.zoom}
    center={position.coordinates} // Coordinates to center map
    onMoveEnd={handleMoveEnd}
  >
  <Geographies geography={geoUrl}>
    {({ geographies }) =>
      geographies.map((geo) => { //function to read and project SVG map file
       return (
          <Geography key={geo.rsmKey} geography={geo}
            fill="#5C80D9"
          />
        )
      })
    }
  </Geographies>
    <Marker coordinates={[26.16, 63.5]} /* Marker component */>
      <circle r={0.1} fill="#BA1DBD" />
    </Marker>
    <Marker coordinates={[27., 64.01]}>
      <circle r={0.1} fill="#BA1DBD" />
    </Marker>
  </ZoomableGroup>
</ComposableMap>
```

Kuva 14. React-simple-maps koodi osa.




Kuva 15. Esimerkki Karttanäkymästä LAM-pisteillä.

Vaatimuksena oli, että datapisteet on pystyttävä valitsemaan sovelluksen kartasta (FunctionalReqId007), jonka pohjalta tehtiin karttamerkkeihin klikkaustoiminto, joka avaa ponnahdusikkunan. Vaatimuksena oli, että datapisteen valitsemalla kartasta pystyy siirtyä näkemään sen datapisteen visualisoitu data (FunctionalReqId010), jonka mukaan karttamerkkiä klikkaamalla annetaan karttamerkkiä vastaavan datapisteen tiedot ponnahdusikkunaan. Tämän jälkeen tarkistin datapisteiden määrän, lisäsin karttamerkkeihin datapisteiden pistetunnuksen ja datapisteiden nimiä vastaavat koordinaatit. Datapisteiden määräksi tuli lopulta 514 datapistettä kartalle (kuvio 16).




Kuva 16. Karttanäkymä 514 LAM-pisteellä.

### 3.3.3 CSV-datan käsittely React-kirjastolla

Kartan lisäämisen jälkeen vaatimuksen oli, että CSV-raakadata on pystyttävä muuttamaan React-kehikolla JSON-formaattiin (FunctionalReqId003), jonka mukaan tehtiin toiminto muuttamaan CSV-tiedoston data luettavaksi React-kehikoilla. CSV-tiedoston lukemista varten on tehty React-kehikoilla react-papaparse, joka on tehokas kehikko CSV-tiedoston datan jäsentämiseen. React-papaparse asentaminen tapahtui komentokehotteessa kuvion 17 komennolla.

```
npm install react-papaparse --save
```

Kuva 17. Komento asentamaan react-papaparse

Asentamisen jälkeen lisättiin CSVReader toiminto, jota käytettiin testausvaiheessa lokaalia tiedoston hakua varten mutta joka myöhemmin vaihdettiin verkon yli haettavaan muotoon. CSV-data muutetaan objekti muotoon, jonka sisällä jokainen CSV-tiedoston rivi on omana taulukkonaan. CSV-tiedoston jäsennetty data käydään läpi for-silmukalla ja taulukoista otetaan tarvittava data taulukkoindeksin avulla. Tämä haluttu data lisätään toiseen taulukkoon (kuvio 18).

```
let counter = 0;
//loop for every single speed
for (let k = 0;k < 195;k++) {

  //reset counter after single speed check
  counter = 0;

  //loop for checking how many cars with certain speed

  for(let i = 0; i < props1.data.length; i++) {
    let obj = props1.data[i][11];
    if (speedcounter == obj) {
      counter++;
    }
  }

 let speed = speedcounter.toString();
 let newspeed = speed.concat(" ", text);
 let color = "";
 //give color according to speedlimit
 if (speedcounter > target-10 && speedcounter < target+5) {
  color = "color: #23DE36";
 }
 else if (speedcounter > target-20 && speedcounter < target+10) {
  color = "color: #FBFF26";
 }
 else if (speedcounter > target-40 && speedcounter < target+15) {
  color = "color: #FF5F1B";
 }
 else {
  color = "color: #c90000";
 }
 Handleddata.push([newspeed, counter, color]); //add new speed, car amount and color
 speedcounter++;
}
```

Kuva 18. For-loop koodi CSV-datan läpikäyntiin.

CSV-data käydään läpi for-loopilla tarkistaen jokainen nopeus ja lisätään, montako kertaa nopeus löytyy. CSV-dataa käydään 195 kertaa läpi katsoen jokainen nopeus erikseen ja lopuksi siitä tehdään oma taulukko nopeuden nimellä, nopeuksien määrällä ja värillä. Taulukot lisätään jokaiselta for-loop kierroksella yhteen taulukkoon.

## 3.4 Visualisointi React-kehikoilla

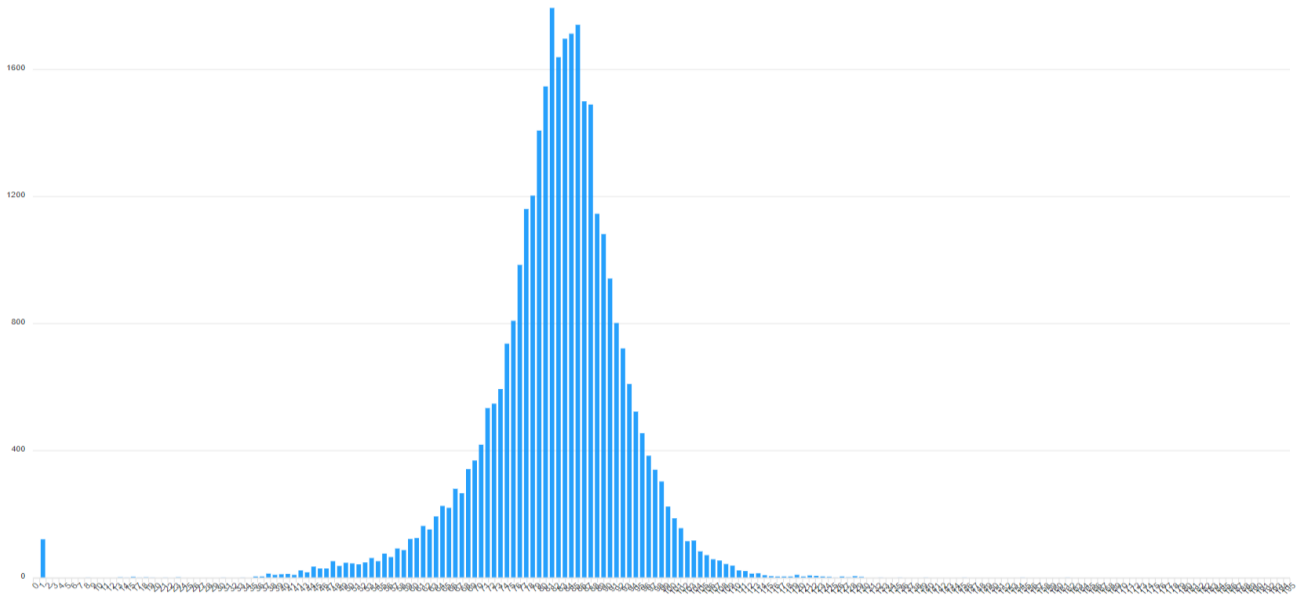
Datan näyttämistä vaatimuksena oli, että pystyttävä visualisoimaan käsitelty JSON-data JavaScriptin React-kehikolla (FunctionalReqId004), joten vertailtiin kahden Reactilla toimivan pylväsdiagrammin visualisoinnin käyttöönottoa ja toimintaa. Pylväsdiagrammin visualisoinnit ovat ApexCharts ja React Google charts sivustoilla olevat React ohjelmoinnilla toteutetut pylväsdiagrammi visualisoinnit.

ApexCharts pylväsdiagrammi visualisointi alkoi asentamalla react-apexcharts kehikko komentokehotteessa kuvion 19 komennolla.

```
npm install -- save react-apexcharts
```

Kuva 19. Npm komento React ApexCharts asennukseen

Asentamisen jälkeen tarvitsi muokata datan käsittely funktiota järjestämään näytettävä data ja x-akselin nopeus luvut ApexCharts mukaiseksi. Tämä tapahtuu antamalla taulukko x-akselin luvuille ja haluttujen datojen määrille. Lopuksi tarvitsi lisätä taulukon näyttäminen ja datan lisääminen. Kuviossa 20 LAM-pisteen datan näyttäminen ApexCharts avulla.



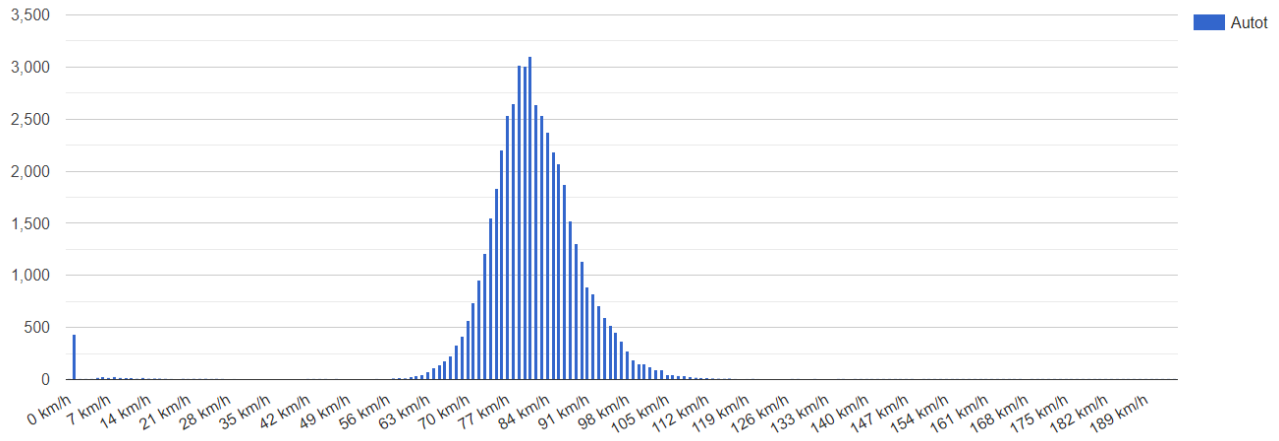

Kuva 20. Nopeudet visualisoituna ApexCharts avulla.

React Google charts visualisointi alkoi asentamalla React Google charts komentokehotteessa kuvion 21 komennolla.

```
npm install --save react-google-charts
```

Kuva 21. Npm komento React Google charts asennukseen.

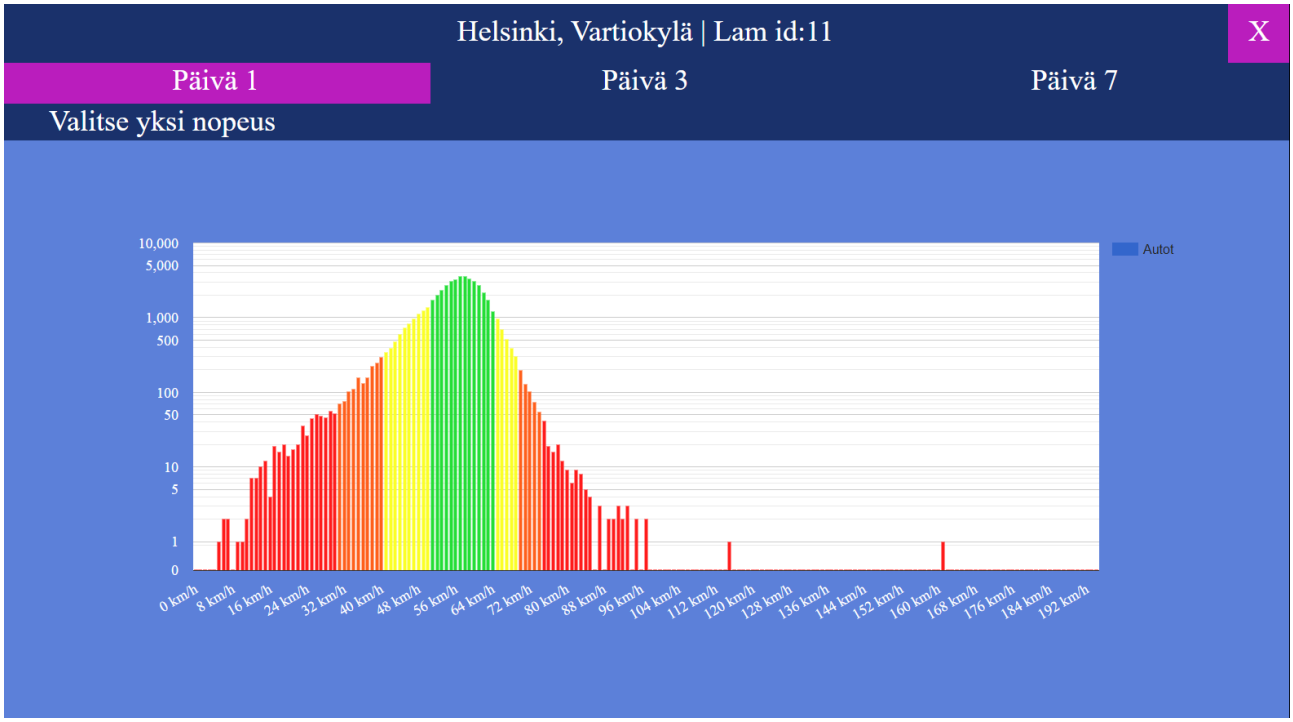
Asentamisen jälkeen muokkasin taulukon rakennetta tarvittavaksi, joka on taulukoita taulukon sisällä eli sisäkkäinen taulukko tavalla ja ensimmäinen taulukko on otsikkorivinä pylväsdiagrammissa. Otsikkorivi muodostuu ensimmäisenä nopeus sarakkeesta ja jälkeenpäin autojen määrä sarakkeesta. Datankäsittely funktiota muokattiin lisäämään nopeus ja autojen määrät jokaiselle riville omana taulukkonaan. Tämän jälkeen lisättiin pylväsdiagrammi ja näytettävä data. Kuviossa 22 LAM-pisteen datan näyttäminen Google charts avulla.




Kuva 22. Nopeudet visualisoitu Google Charts avulla.

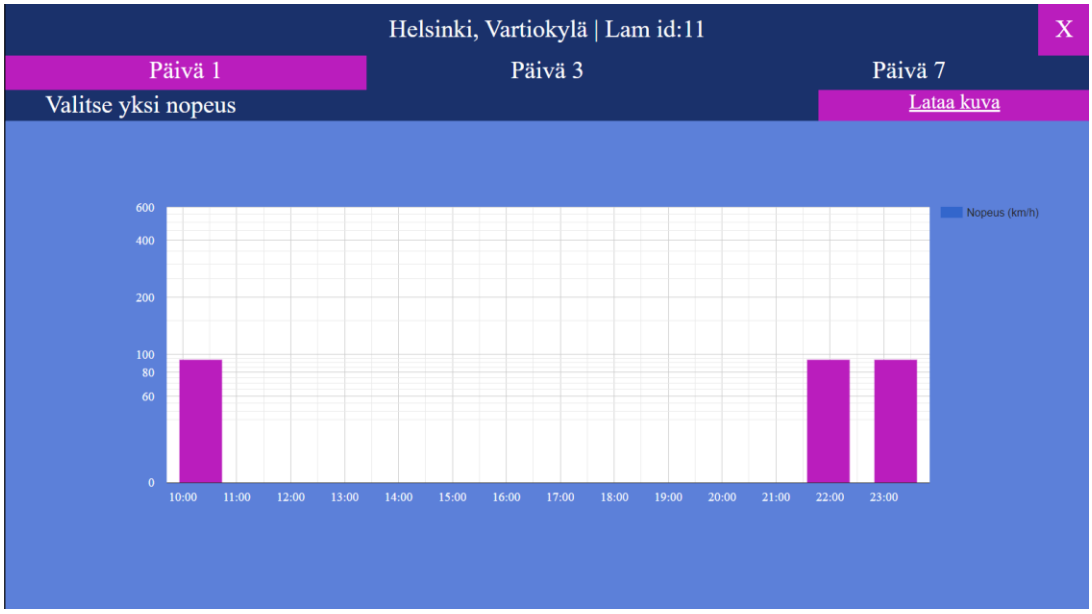
Pylväsdiagrammin toteutuksia vertaillessa React Google charts toteutus päivitti dataa pylväsdiagrammissa, kun data päivittyi ilman erillisen datan päivitys funktion tekemistä. ApexCharts antaa puolestaan vaatii datan erillisen päivittämisen. Vaatimuksena oli, että visualisoinnilla on pystyttävä näkemään ajoneuvomäärät ja nopeudet (FunctionalReqId011), joka toteutui molemmissa ja pylväsdiagrammien ulkonäköä pystyy muokkaamaan. Vertailluista React-kehikoista React Google charts valittiin lopulliseen toteutukseen datan näyttöä varten.

Valitun React-kehikosta katsoin tarkemmin pylväsdiagrammin dokumentointia ja muokkasin pylväsdiagrammin näyttämään lähes samalta, kuin Figma prototyypin pylväsdiagrammin ja muutin y-akselin skaalauksen logaritmiseksi näyttääkseen vähemmän ajetut nopeudet paremmin. Tämän jälkeen vaatimuksena oli, että käyttäjän on voitava vaihtaa visualisointi näkymä näyttämään tiedot 1/3/7 vuorokausilta (FunctionalReqId002), jonka mukaan tein toimintoja näyttämään dataa yhdeltä päivältä, kolmelta päivältä ja seitsemältä päivältä. Tämän jälkeen lisäsin toiminnon, että käyttäjä pystyy vaihtamaan näytetyn datan haluttujen 1/3/7 vuorokausien datan näkymään. Lisäksi vaatimuksena oli, että visualisoinnista pitäisi nähdä ylinopeudet ja ajoneuvomäärät (nonFunctionalReqId015), jonka mukaan muokkasin y-akselin skaalautumisen logaritmiseksi, että vähemmän ajetut nopeudet, kuten ylinopeudet erottuisivat paremmin visualisoinnista kuviossa 23.



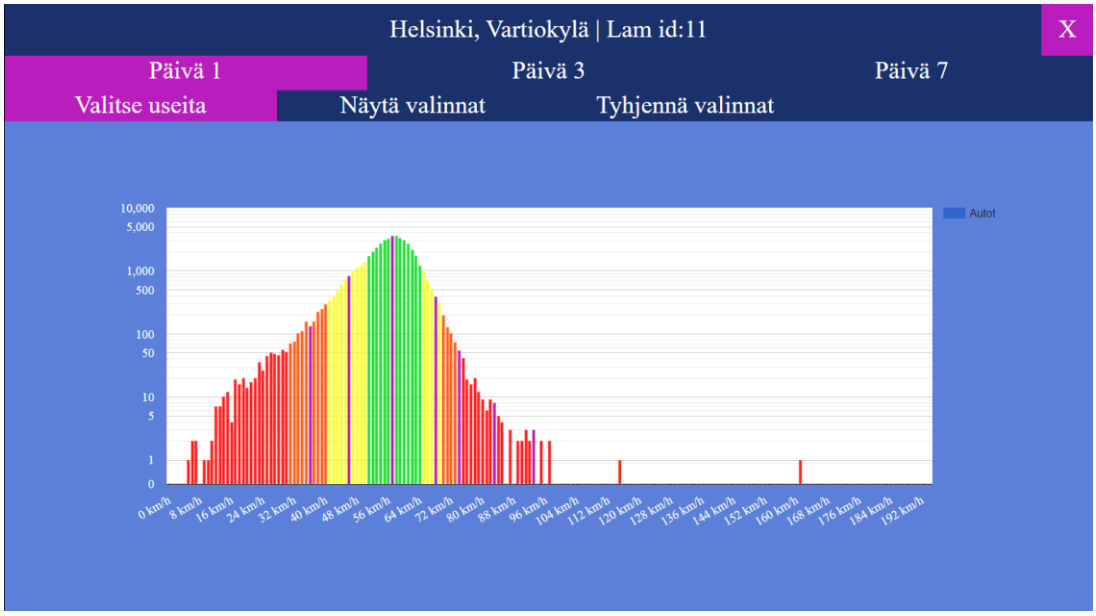

Kuva 23. Nopeudet visualisoituina pylväsdiagrammissa.

Pylväsdiagrammille vaatimuksena oli, että pystyttävä valitsemaan nopeus ja nähtävä ajetun nopeuden päivänajat (FunctionalReqId012), jonka mukaan lisäsin nopeuksia varten klikkaus toiminnon, joka valitsee klikatun nopeuden ja vaihtaa pylväsdiagrammin datan klikatun nopeuden näyttämisen kellonajan, milloin nopeutta on ajettu kuviossa 24. Vaatimuksena oli, että pystyttävä lataamaan kuva ajetun nopeuden päivänajat näkymästä (FunctionalReqId013), jonka mukaan tarkemmin tarkasteltavia nopeuksia varten tein toiminnon, joka ottaa tarkasteltavista nopeuksista kuvan ja kuvan pystyy lataamaan painiketta painamalla.




Kuva 24. Nopeuden kellonaikojen näyttäminen.

Nopeuksien valitsemista varten tein myös toiminnon, että useampia nopeuksia voi klikata ja katsoa niiden tarkempia ajanhetkiä. Useampien nopeuksien katsomista varten lisäsin uusia painikkeita vaihtamaan yksittäisen tai useamman nopeuden valitsemisen välillä, useamman valitun nopeuden näyttäminen ja valittujen nopeuksien pois valitseminen painikkeet. Valituille nopeuksille tein myös värin muutoksen, kun nopeudet ovat valittuina kuviossa 25.




Kuva 25. Pylväsdiagrammisssa nopeupeuden valinna ilmaiseminen värimuutoksen avulla.

Lopuksi vaatimuksena oli, että sovellus ei jumittuisi useamman nopeuden valinnassa kellonaikojen tarkastelussa (nonFunctionalReqId016), jonka mukaan rajoitin useamman nopeuden valitsemista, koska liian monta valittuna on suuri mahdollisuus sovelluksen ja selaimen jumittumiseen.

### 3.4.1 Datan hakeminen

Vaatimuksena oli, että Digitrafin palvelusta on voitava hakea CSV-raakadata viimeiseltä 1–7 päivältä (FunctionalReqId001), jonka datan hakeminen tapahtui Digitrafin web-palvelimilta ja haettava data on CSV muodossa. Datan hakemista varten käytetään Digitrafin omaa API-kutsua, jonka avulla määritetään haettava data. Datan hakeminen suoraan selaimen kautta ei onnistunut, koska selaimen datan haun esto tapahtui, joten jouduttiin tekemään pieni välityspalvelin tai proxy server englanniksi. Välityspalvelimen tekemiseen käytettiin NodeJS-kehikkoa ExpressJS ja valintaan vaikutti esimerkkien löytyminen ja välityspalvelimen helppo tekeminen. ExpressJS kehikolla tehtiin välityspalvelimen toiminnot kuvion 26 mukaisesti. Välityspalvelimen avulla saatiin vietyä API-kutsu oikeaan paikkaan ja selaimen oma vieraan paikan datan haun esto ohitettiin.

```
const express = require('express');
const cors = require('cors');
const { createProxyMiddleware } = require('http-proxy-middleware');
const path = require('path');

const app = express();

const PORT = process.env.PORT || 8000;

const API_SERVICE_URL = "https://tie-test.digitraffic.fi";

app.use(cors());

app.use('/api_call', createProxyMiddleware({
    target: API_SERVICE_URL,
    changeOrigin: true,
    pathRewrite: {
        [`^/api_call`]: '',
    },
}));

app.use(express.static(path.join(__dirname, '/build')));

app.get('*', function(req, res) {
    res.sendFile(path.join(__dirname, '/build', 'index.html'));
});

app.listen(PORT, () => {
    console.log(`Starting Proxy server at ${PORT}`);
});
```

Kuva 26. Välityspalvelimen koodin rakenne ExpressJS.

Datan haku vaatimuksena oli, että CSV-raakadata voitava hakea automaattisesti Digitrafin palvelusta (FunctionalReqId014), jonka mukaan tehtiin ponnahdusikkunan avautumisen jälkeen ja data haetaan automaattisesti. Automaattinen datan hakeminen tapahtui ottamalla datan haun päivämäärä ja muuttamalla päivämäärä API-kutsun mukaiseksi. Tämän jälkeen jokaista edellistä 1–7 päivää varten tehtiin oma API-kutsu osoite ja API-kutsu vietiin välityspalvelimelle. Välityspalvelin uudelleen välitti API-kutsut Digitraffic palvelimille ja vei API-kutsun tiedot takaisin datan haku toiminnolle.

Lopuksi React Front-end osiosta tehtiin tuotanto versio ja tuotanto version Front-end välitetään palvelimen kautta, kun yhdistää palvelimen osoitteeseen. Lopullinen testaaminen jäi vielä palvelimen osalta kehitysversioon, kun opinnäytetyön lopussa sovelluksen kehitys koneen virtuaalikone ei toiminut. Välityspalvelinta pystyi kuitenkin testaamaan, kun NodeJS mahdollisti välityspalvelimen toimivan lokaalisti ja myös hakemaan tarvittavat tiedot Digitraffic palvelimilta. Välityspalvelimelle on mahdollista lisätä jälkeenpäin tarvittaessa datan haun vieminen välityspalvelimen HTTP-välimuistin kautta. Tämä vähentäisi tiedostojen hakua Digitraffic palvelimilta.

HTTP-välimuisti tai HTTP cache englanniksi on rakennettu protokolla web-selaimille ja välityspalvelimille varmistaakseen, että ne voivat tallentaa HTTP-vastauksen välimuistiin tietyltä ajalta. Välimuistiin vastauksen tallentaminen varmistaa, ettei selaimen tai välityspalvelimen tarvitse hakea samaa sisältöä tai dataa ulkoiselta web-palvelimelta useampaan kertaan. Tämä vähentää web-palvelimen kuormitusta. Toiminnasta esimerkkinä web-selain tarvitsee tiedoston, jonka jälkeen lähettää pyynnön välityspalvelimelle. Välityspalvelin tarkistaa onko sen välimuistissa tiedostoa, jonka palauttaa web-selaimelle tai pyynnön kautta hakee tiedoston web-palvelimelta ja palauttaa web-selaimelle. (Grigorik & Posnick 2020.; HTTP caching 2022.; HTTP caching basics n.d.)

HTTP-välimuistin toimintaa pystyy kontrolloimaan lisäämällä pyyntö otsikon ja mahdollisesti vastaus otsikon HTTP-pyyntöihin. Näillä otsikoilla vaikutetaan HTTP-välimuistiin kertomalla vastauksen elinikä, milloin on uudelleen käytettävissä ja milloin se pitää uusia. Vastaukselle voidaan kertoa, milloin se pitää tarkistaa. Vastauksen tallentamiseen voidaan myös kertoa voiko sitä tallentaa välimuistiin ja voiko sen tallentaa vain web-selaimella tai web-selaimella ja välityspalvelimelle. (Grigorik & Posnick 2020.; HTTP caching 2022.; HTTP caching basics n.d.)

HTTP-välimuisti hyödynnetään opinnäytetyön jälkeen, jos on tarvetta välityspalvelimelle tallentaakseen web-palvelimelta haettuja tiedostoja, sillä tiedostoja tullaan hakemaan päivittäin mahdollisesti useampi sata ja tämän avulla vähennetään Digitrafin palvelimen rajapintojen kuormitusta. Web-selaimen tarvittaessa jotain tiedostoa se tarkistetaan välimuistipalvelimella kyseisen tiedoston olemassaolosta ja tarvittaessa hakee tiedoston web-palvelimelta. Näille tiedostoille asetetaan vielä maksimi aika, jonka tiedostot pysyvät välityspalvelimen muistissa.

# 4 Tulokset

Tuloksina kehitettiin vaatimukset täyttävä React ohjelmoinnilla toimiva sovellus visualisoimaan LAM-dataa. Sovelluksen vaatimuksina oli, että Digitrafin palvelusta on voitava hakea CSV-raakadata viimeiseltä 1–7 päivältä (FunctionalReqId001) ja CSV-raakadata on voitava hakea automaattisesti Digitrafin palvelusta (FunctionalReqId014), jotka toteutettiin tekemällä toiminto ottamaan päivämäärät edelliseltä 7 vuorokaudelta ja muuttamalla päivämäärät API-kutsun muotoisiksi, jonka jälkeen päivämäärät muutettiin API-kutsuiksi. API-kutsut vietiin välityspalvelimen kautta Digitrafficin rajapintaan, joka palautti CSV-raakadatan 7 vuorokaudelta. Vaatimuksena oli, että käyttäjän on voitava vaihtaa visualisointi näkymä näyttämään tiedot 1/3/7 vuorokausilta (FunctionalReqId002), joka toteutettiin tekemällä painikkeet visualisointinäkymään vuorokausien vaihtamiseen ja toiminto vaihtamaan visualisoidun datan 1/3/7 vuorokauden mukaiseksi. Vaatimuksena oli, että CSV-raakadata on pystyttävä muuttamaan React-kehikolla JSON-formaattiin (FunctionalReqId003), joka toteutettiin käyttämällä react-papaparse muuttamaan CSV-raakadata JSON-formaattiin. Vaatimuksena oli, että pystyttävä visualisoimaan käsitelty JSON-data JavaScript React-kehikolla (FunctionalReqId004), joka toteutettiin käyttämällä React Google charts visualisoimaan JSON-data pylväsdiagrammiin, kun JSON-datasta otettiin tarpeellinen tieto visualisointia varten.

Vaatimuksina ovat, että sovelluksen on pystyttävä käyttämään karttaa (FunctionalReqId005), datapisteet pystyttävä mallintamaan sovelluksen kartalle (FunctionalReqId006), datapisteiden pystyttävä valitsemaan sovelluksen kartasta (FunctionalReqid007), sovelluksen karttaa on pystyttävä zoomaamaan (FunctionalReqid008) ja sovelluksen karttaa on pystyttävä liikuttamaan hiirellä vedettäessä (FunctionalReqid009), jotka toteutettiin käyttämällä react-simple-maps Suomen kartan

projektoimiseen sovellukseen, lisäämällä karttaan karttamerkit, karttamerkkien valinnan, zoomaus toiminnon ja kartan liikuttamisen hiiren avulla. Vaatimuksena oli, että datapisteen valitsemalla kartasta pystyy siirtyä näkemään sen datapisteen visualisoitu data (FunctionalReqId010), joka toteutettiin yhdistämällä karttamerkin valinta toiminnon avaamaan visualisointinäkymän ja välittämällä karttamerkin LAM-pisteen ID:n API-kutsuun. Visualisointinäkymän avautuessa LAM-pisteen data haetaan automaattiseste, käsitellään ja asetetaan visualisoitu data näkyville. Vaatimuksena oli, että visualisoinnilla on pystyttävä näkemään ajoneuvomäärät ja nopeudet (FunctionalReqId011), joka toteutettiin tekemällä toiminto karsimaan ylimääräinen JSON-data ja välittämällä ajoneuvomäärät ja nopeudet pylväsdiagrammin visualisointiin. Vaatimuksena oli, että pystyttävä valitsemaan nopeus ja nähtävä ajetun nopeuden päivänajat (FunctionalReqId012), joka toteutettiin tekemällä pylväsdiagrammiin nopeuden sarakkeen valinta toiminto, jonka jälkeen tehtiin toiminto vaihtamaan pylväsdiagrammin data näyttämään valitun nopeuden ajetut kellonajat pylväsdiagrammissa. Vaatimuksena oli, että pystyttävä lataamaan kuva ajetun nopeuden päivänajat näkymästä (FunctionalReqId013), joka toteutettiin tekemällä toiminto ottamaan kuva valitun nopeuden kellonajoista ja välittämällä kuvan tiedot latauspainikkeelle, joka lataa kuvan png muodossa.

Vaatimuksena oli, että visualisoinnista pitäisi nähdä ylinopeudet ja ajoneuvomäärät (nonFunctionalReqId015), joka toteutettiin lisäämällä värejä pylväsdiagrammiin nopeusrajoituksien mukaan ja muuttamalla pylväsdiagrammin y-akselin skaalauksen logaritmiseksi, jotta vähemmän ajetut nopeudet erottuisivat, kuten ylinopeudet. Vaatimuksena oli, että sovellus ei jumittuisi useamman nopeuden valinnassa kellonaikojen tarkastelussa (nonFunctionalReqId016), joka toteutettiin rajoittamalla useamman nopeuden valintaa pylväsdiagrammista sen mukaan, mikä vuorokausi näkymä on valittuna. Tuloksina saatiin kokonaisuudessaan toimiva ja lähes ilman vikoja oleva sovellus LAM-datan liikennemäärien ja nopeuksien visualisointiin.

# 5 Pohdinta

Opinnäytetyön tarkoituksena oli visualisoida LAM-asemien keräämiä liikennetietoja. Visualisoitavat liikennetiedot ovat liikennemäärät ja nopeudet. Tavoitteena oli tehdä sovellus visualisoimaan

kartalle LAM-pisteet, jotka valitsemalla avautuu sen LAM-pisteen liikennemäärien ja nopeuksien visualisointi 1/3/7 vuorokaudelta.

Tuloksina kehitettiin vaatimusten pohjalta React ohjelmoinnilla toimiva sovellus visualisoimaan LAM-dataa. Vaatimuksena oli, että Digitrafin palvelusta on voitava hakea CSV-raakadata viimeiseltä 1–7 päivältä (FunctionalReqId001), jonka toteutuksessa onnistuttiin hyvin ja datan haku toimi. Vaatimuksena oli, että käyttäjän on voitava vaihtaa visualisointi näkymä näyttämään tiedot 1/3/7 vuorokausilta (FunctionalReqId002), jonka toteutuksessa onnistuttiin hyvin ja data vaihtuu päivän mukaan. Vaatimuksena oli, että CSV-raakadata on pystyttävä muuttamaan React-kehikolla JSON-formaattiin (FunctionalReqId003), jonka toteutuksessa onnistuttiin ja datan muutos toimii onnistuneesti. Vaatimuksena oli, että pystyttävä visualisoimaan käsitelty JSON-data JavaScriptin React-kehikolla (FunctionalReqId004), jonka toteutuksessa onnistuttiin. Vaatimuksena oli, että sovelluksen pystyttävä käyttää karttaa (FunctionalReqId005), jonka toteutuksessa onnistuttiin, mutta karttana olisi voinut käyttää muuta karttaa. Vaatimuksena oli, että datapisteet pystyttävä mallintamaan sovelluksen kartalle (FunctionalReqId006), jonka toteutuksessa onnistuttiin, mutta lähellä toisiaan olevat karttapisteiden koon joutui pienentämään ja siirtämään ettei tulisi päällekkäisyyksiä. Vaatimuksena oli, että datapisteiden pystyttävä valitsemaan sovelluksen kartasta (FunctionalReqId007), jonka toteutuksessa onnistuttiin, mutta visualisointinäkymän suljettua ei pysty valitsemaan uutta karttapistettä 2 sekuntiin jostain syystä. Vaatimuksena oli, että sovelluksen karttaa on pystyttävä zoomaamaan (FunctionalReqId008), jonka toteutuksessa onnistuttiin, mutta toiminto ei toimi visualisointinäkymän suljettua 2 sekuntiin. Vaatimuksena oli, että sovelluksen karttaa on pystyttävä liikuttamaan hiirellä vedettäessä (FunctionalReqId009), jonka toteutuksessa onnistuttiin sen ollessa osa kartta komponenttia.

Vaatimuksena oli, että datapisteen valitsemalla kartasta pystyy siirtyä näkemään sen datapisteen visualisoitu data (FunctionalReqId010), jonka toteutuksessa onnistuttiin hyvin. Vaatimuksena oli, että visualisoinnilla on pystyttävä näkemään ajoneuvomäärät ja nopeudet (FunctionalReqId011), jonka toteutuksessa onnistuttiin, mutta y-akselin skaalaus piti muuttaa logaritmiseksi, jotta vähemmän ajetut nopeudet nähtäisiin visualisoinnista. Vaatimuksena oli, että pystyttävä valitsemaan nopeus ja nähtävä ajetun nopeuden päivänajat (FunctionalReqId012), jonka toteutuksessa onnistuttiin, mutta vain yhden päivän kohdalla ajankohdat tarkoittavat kyseistä päivää. Vaatimuk-

sena oli, että pystyttävä lataamaan kuva ajetun nopeuden päivänajat näkymästä (FunctionalReqId013), jonka toteutuksessa onnistuttiin, mutta kuvasta ei näe helposti ajetun nopeuden lukua. Vaatimuksena oli, että CSV-raakadata voitava hakea automaattiseste Digitrafin palvelusta (FunctionalReqId014), jonka toteutuksessa onnistuttiin, mutta datan haku voi epäonnistua, jos datan haku tapahtuu ennen, kuin edellisen päivän data on jaossa Digitrafin palvelussa. Vaatimuksena oli, että visualisoinnista pitäisi nähdä ylinopeudet ja ajoneuvomäärät (nonFunctionalReqId015), jonka toteutuksessa onnistuttiin hyvin. Vaatimuksena oli, että sovellus ei jumittuisi useamman nopeuden valinnassa kellonaikojen tarkastelussa (nonFunctionalReqId016), jonka toteutuksessa onnistuttiin, mutta 7 päivän kohdalla maksimaalinen nopeuksien valinta voi hidastaa selainta.

Kehityshaaste RQ1 toteutui käyttämällä pylväsdiagrammia ja värejä nopeusrajoituksien erottelussa liikennemäärien ja nopeuksien visualisoimisessa. Kehityshaaste RQ2 toteutui näyttämällä LAM-pisteet karttamerkkeinä pisteinä kartalla, mutta karttamerkeille olisi voinut antaa vektorigrafiikkaa. Kehityshaaste RQ3 toteutui tekemällä vuorokausien vaihtamisen ja datan vaihtamisen pylväsdiagrammiin vuorokausien perusteella.

Toteutuksen onnistumisia olivat CSV-datan muuttaminen visualisoituun muotoon, nopeuksien valinta pylväsdiagrammista, kuvan lataaminen nopeuksien kellonaikojen näkymästä ja datan automaattinen haku välityspalvelimen kautta. Toteutuksen aikana ajoneuvoluokkien hyödyntäminen ei onnistunut opinnäytetyössä tehdyllä datakäsittely toiminnolla.

Visualisoitu LAM-data voi näyttää epärealistiselta, jos haetussa datassa on virheellisiä tietoja paljon. Tämä johtui siitä, että LAM-pisteiden määrä oli yli 500 ja jokaisen LAM-pisteen datan visualisointia ei tarkistettu ennen karttamerkin lisäämistä. Näiden LAM-pisteiden kartalta pois jättäminen on kuitenkin helppoa ottamalla karttamerkin pois. Myös datan näyttäminen omituiselta voi johtua eri suuntien omista nopeusrajoituksista. Tämä johtaa visualisoinnin näyttämään oudolta värien kohdalla, kun värit annetaan yhden nopeusrajoituksen mukaan LAM-pisteen nopeuksille.

LAM visualisoinnin tuloksia voi hyödyntää katsomalla LAM-pisteiden ajettuja nopeuksia ja ylinopeuksia. Nopeuksista pystyy tarkastamaan, milloin nopeuksia on ajettu. Nopeuksien kellonaikojen tarkastelusta kuitenkin pystyy tarkasti näkemään vain yhden päivän nopeuksien kellonajat.

Sovellusta voidaan vielä jatkokehittää pidemmälle. Jatkokehityksen kohteita olisi ajoneuvoluokkien lisääminen tarkasteluun nopeuksien yhteydessä ja LAM-pisteen eri suuntien huomioonottaminen visualisoinnissa. Opinnäytetyössä keskityttiin eniten nopeuksiin, joten monet muut LAM-pisteiden keräämät tiedot jäivät hyödyntämättä.

# Lähteet

Brush, K., Burns, E. 2022. What is data visualization? Techtarget. Artikkeli Techtarget -verkkosivulla 12.2022. Viitattu 13.1.2023. https://www.techtarget.com/searchbusinessanalytics/definition/data-visualization.

Cardello, J. 2022. Webflow. Artikkeli Webflow -verkkosivuilla. Viitattu 16.3.2023. https://webflow.com/blog/ui-ux-design-tools.

Clemente, D. 2022. How do you make a prototype Figma website? Websitebuilderinsider. Artikkeli Websitebuilderinsider -verkkosivuilla 28.7.2022. Viitattu 17.1.2023. https://www.websitebuilderinsider.com/how-do-you-make-a-prototype-figma-website/.

Create user interfaces from components. N.d. React. Viitattu 27.3.2023. https://react.dev/.

Creative tools meet the internet. N.d. Figma. Figma -verkkosivuilla. Viitattu 12.1.2023. https://www.figma.com/about/.

DATAVISUALISOINTIOPAS-VISUALISOINTI. N.d. Xamk. Xamk -verkkosivuilla. Viitattu 27.1.2023. https://www.xamk.fi/dataopas-visualisointi/.

Digitraffic avoin LAM-data. N.d. Fintraffic. Fintraffic -verkkosivuilla. Viitattu 12.1.2023. https://www.fintraffic.fi/fi/tie/digitraffic-avoin-lam-data.

Hurwitz, O. 2022. Nasdaq. Artikkeli Nasdaq -verkkosivuilla. Viitattu 5.4.2023. https://www.nasdaq.com/articles/does-the-purchase-of-figma-make-sense-for-adobe.

Getting started. N.d. React-simple-maps. Dokumentointi react-simple-maps -verkkosivuilla. Viitattu 13.1.2023. https://www.react-simple-maps.io/docs/getting-started/.

Grigorik, I., Posnick, J. 2020. Prevent unneccessary network requests with the HTTP Cache. Web.dev. Artikkeli web.dev -verkkosivulla 17.4.2020. Viitattu 16.1.2023. https://web.dev/http-cache/.

Gupta, A. 2022. 10 Good and Bad Examples of Data Visualization. Polymersearch. Blogi polymersearch -verkkosivulla 23.2.2022. Viitattu 20.1.2023. https://www.polymersearch.com/blog/10-good-and-bad-examples-of-data-visualization.

HTTP caching. 2022. Developer mozilla org. Artikkeli Mozilla developer -verkkosivulla 23.8.2022 Viitattu 13.1.2023. https://developer.mozilla.org/en-US/docs/Web/HTTP/Caching.

HTTP caching basics. N.d. Varnish Ohje varnish software -verkkosivulla. Viitattu 16.1.2023. https://www.varnish-software.com/developers/tutorials/http-caching-basics/.

Introduction. N.d. Vue. Dokumentointi Vue -verkkosivuilla. Viitattu 27.3.2023. https://vuejs.org/guide/introduction.html.

Johnston, J. 2020. Budibase. Artikkeli Budibase -verkkosivuilla. Viitattu 27.3.2023. https://budibase.com/blog/web-application-development/.

Karjalainen, M., Rantonen, M. 2021. Tieto tuottamaan - Digitalisaatiosta kaikki hyöty irti. Jamk. Blogi Jamk -verkkosivuilla. Viitattu 6.3.2023. https://blogit.jamk.fi/techtothefuture/2021/12/02/tieto-tuottamaan-digitalisaatiosta-kaikki-hyoty-irti/.

Kopf, B. N.d. The power of Figma as a design tool. Toptal. Artikkeli Toptal -verkkosivulla. Viitattu 13.1.2023. https://www.toptal.com/designers/ui/figma-design-tool.

LAM-tiedot. N.d. Fintraffic -verkkosivuilla. Viitattu 12.1.2023. https://www.digitraffic.fi/tieliikenne/lam/.

Liikenteen dataekosysteemi lyhyesti. N.d. Fintraffic. Artikkeli Fintraffic -verkkosivuilla. Viitattu 3.4.2023. https://www.fintraffic.fi/fi/fintraffic/liikenteen-dataekosysteemi-lyhyesti.

Make something great. N.d. Sketch. Artikkeli Sketch -verkkosivuilla. Viitattu 24.3.2023. https://www.sketch.com/design/.

Map files. N.d. React-simple-maps. Dokumentointi react-simple-maps -verkkosivuilla. Viitattu 13.1.2023. https://www.react-simple-maps.io/docs/map-files/.

Marker. N.d. React-simple-maps. Dokumentointi react-simple-maps -verkkosivuilla. Viitattu 13.1.2023. https://www.react-simple-maps.io/docs/marker/.

Mitchell, D. N.d. Adobe. Artikkeli Adobe -verkkosivuilla. Viitattu 27.3. 2023. https://blog.udemy.com/what-is-adobe-illustrator-used-for/.

Palvelun esittely. N.d. Fintraffic. Fintraffic -verkkosivuilla. Viitattu 12.1.2023. https://www.digitraffic.fi/palvelun-esittely/.

Projections. N.d. React-simple-maps. Dokumentointi react-simple-maps -verkkosivuilla. Viitattu 25.1.2023. https://www.react-simple-maps.io/docs/projections/.

Prototyping features. N.d. Figma. Figma -verkkosivuilla. Viitattu 12.1.2023. https://www.figma.com/prototyping/.

Quick Start. N.d. React. Dokumentointi React -verkkosivuilla. Viitattu 27.3.2023. https://react.dev/learn.

Rae, M. 2020. Adobe. Artikkeli Adobe -verkkosivuilla. Viitattu 24.3.2023. https://www.adobe.com/products/xd/learn/get-started/what-is-adobe-xd-used-for.html.

Software Development Life Cycle (SDLC) Overview. 2022. Aristeksystems. Artikkeli Aristeksestems -verkkosivuilla. Viitattu 13.3.2023. https://aristeksystems.com/blog/sdlc-overview/.

Styling. N.d. React-simple-maps. Dokumentointi react-simple-maps -verkkosivuilla. Viitattu 25.1.2023. https://www.react-simple-maps.io/docs/styling/.

Tieliikenne. N.d. Fintraffic. Fintraffic -verkkosivuilla. Viitattu 16.3.2023. https://www.digitraffic.fi/tieliikenne/.

UI Design. 2022. UX tools. Kysely UX tools -verkkosivuilla. Viitattu 16.3.2023. https://ux-tools.co/survey/2022/ui-design/.

UI design tool. N.d. Figma. Figma -verkkosivuilla. Viitattu 12.1.2023. https://www.figma.com/ui-design-tool/.

User Interface (UI) Design. n.d. Interaction design foundation. Artikkeli Interaction design foundation -verkkosivuilla. Viitattu 13.3.2023. https://www.interaction-design.org/literature/topics/ui-design.

What is data visualization? N.d. IBM. Artikkeli IBM -verkkosivulla. Viitattu 19.1.2023. https://www.ibm.com/topics/data-visualization.

What is Data Visualization? Definition, Examples, and learning resources. N.d. Tableau. Artikkeli Tableau -verkkosivulla. Viitattu 19.1.2023. https://www.tableau.com/learn/articles/data-visualization.

Zoomable group. N.d. React-simple-maps. Dokumentointi react-simple-maps -verkkosivuilla. Viitattu 25.1.2023. https://www.react-simple-maps.io/docs/zoomable-group/.