VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Akseli Kataja

ÄLYKKÄÄT KARTTARATKAISUT IOT-ALUSTOILLE

Liiketalouden yksikkö
2022

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma

TIIVISTELMÄ

Tekijä	Akseli Kataja
Opinnäytetyön nimi	Älykkäät karttaratkaisut IoT-alustoille
Vuosi	2022
Kieli	suomi
Sivumäärä	51
Ohjaaja	Antti Mäkitalo

Opinnäytetyön tavoitteena oli tarjota lukijalle tarvittavat tiedot karttaratkaisun valintaan erityisesti IoT-alustoille, jotta tulevia kehityshankkeita varten pystytään valitsemaan optimaalinen ja kustannustehokas ratkaisu. Esiteltävien ratkaisujen tarkoituksena oli mahdollistaa erilaisten visualisointien kuten lämpökarttakerrosten sekä monikulmioalueiden luominen kartalla.

Opinnäytetyön teoriaosuus keskittyi esittelemään tällä hetkellä tarjolla olevia karttaratkaisuvaihtoehtoja. Esiteltäväksi valikoituivat Leaflet-, OpenLayers-, Mapbox- sekä Google Maps -karttasovellukset. Työssä käytiin läpi kyseisten ratkaisujen taustoja, ominaisuuksia sekä suosiota. Opinnäytetyössä korostettiin erityisesti yksittäisten karttaratkaisujen piirteitä sekä hyviä että huonoja puolia.

Tutkimuksen teoriaosuuden tulosten perusteella valittiin optimaalisimmalta vaikuttava karttaratkaisu, Leaflet. Käytännön osuudessa toteutettiin Leaflet-kirjaston avulla verkkopohjainen karttaratkaisu muutamilla toiminnoilla sekä visualisoinneilla.

Opinnäytetyön lopputuloksena on kattava tietopaketti nykyaikaisista IoT-alustoille sopivista karttaratkaisuista, ja sen tarkoitus on auttaa optimaalisen karttaratkaisun valinnassa.

Avainsanat	karttaratkaisut, paikkatietojärjestelmät, iot, web-kehitys

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietojenkäsittelyn koulutusohjelma

ABSTRACT

Author	Akseli Kataja
Title	Intelligent map solutions for IoT platforms
Year	2022
Language	Finnish
Pages	51
Name of Supervisor	Antti Mäkitalo

The aim of the thesis was to provide the reader with the necessary information for choosing a map solution especially for IoT platforms, in order to be able to choose the most optimal and cost-effective solution for future development projects. The objective of the presented solutions was to enable the creation of various visualizations such as thermal map layers and polygon areas on the map.

The theoretical section of the thesis focused on presenting the currently available map solution options. Leaflet, OpenLayers, Mapbox and Google Maps solutions were selected for presentation. The background, features and popularity of these solutions were reviewed. In the thesis, the pros and cons of individual map solutions were especially emphasized.

Based on the results of the theoretical section of the study, the most optimal map solution, Leaflet, was selected. In the practical section, a web-based map solution with a few functions and visualizations was developed with the help of the Leaflet library.

The result of the thesis was a comprehensive information package on modern map solutions suitable for IoT platforms, which is aimed to help in the selection of the optimal map solution.

Keywords map solutions, geographic information systems, iot, web development

SISÄLLYS

KUVA-, TAULUKKO- JA KAAVIOLUETTELO

1 JOHDANTO

Verkkokartoituksen ja paikkatiedon käyttö on kasvanut nopeasti viime vuosikymmeninä internetin kehityksen myötä. Melkein jokainen ihminen maapallolla käyttää jossain määrin karttatietoja joko tietoisesti tai tietämättään. Lähes jokaisessa matkapuhelimessa on nyt paikannuspalvelut, ja jokaisella tapahtumalla ja esineellä maan päällä on sijainti. Suuria määriä paikkatietoa on saatavilla verkossa joka päivä ja sitä käytetään erilaisissa verkkosovelluksissa ja kartoissa esimerkiksi datan analysointiin, mallintamiseen sekä simulointiin.

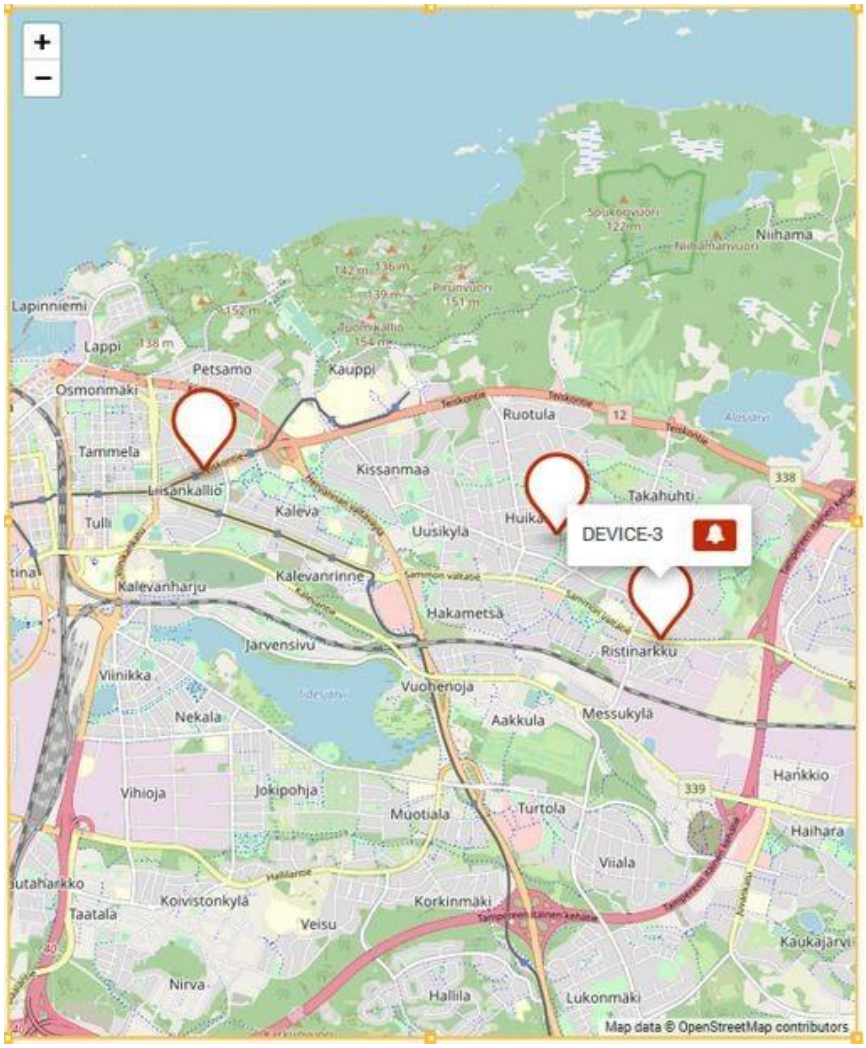
Voidaankin todeta, että nykypäivänä kartat ovat keskeinen osa erilaisia käyttöliittymiä ja sovellusratkaisuja. Nopean kehityksen myötä käyttäjät odottavat yhä edistyneempiä ja interaktiivisempia visualisointeja sekä toimintoja kuvaamaan haluttua dataa esimerkiksi lämpökarttojen avulla. Tietojen visualisointi kartalla tekee datasta luonnollisemman ihmismielen ymmärrettäväksi ja erilaiset toiminnot yleisesti ottaen parantavat käyttäjäkokemusta.

Viime vuosien kehityksen myötä uusia visualisointeja tarjoavia ohjelmistoja sekä kirjastoja on ilmestynyt yhä enemmän helpottamaan kehittäjän työtä karttasovelluksien toteuttamisessa. Sama karttaratkaisu ei välttämättä sovi erilaisiin käyttötarkoituksiin, jonka myötä tutkimus aiheesta on tärkeää optimaalisten karttaratkaisujen löytämiseksi.

Tämä opinnäytetyö tehdään erityisesti avustamaan kehittäjiä karttaratkaisun valitsemisessa erilaisissa käyttötapauksissa. Työ tarjoaa kattavan tietopaketin useiden karttaratkaisujen piirteistä, suosiosta sekä ominaisuuksista. Opinnäytetyö todistaa myös yhden karttaratkaisun kohdalla konseptin toimivuuden verkkopohjaisen karttasovelluksen toteutuksen avulla.

1.1 Tutkimusongelma

Kesällä 2021 Wapice Ltd:lla ilmeni tarve edistyneempään karttakomponenttien käyttöön yrityksen IoT-TICKET tuotteessa. Nykyisessä ratkaisussa yritys on käyttänyt yksinkertaista karttapohjaa, johon on ollut mahdollista visualisoida yksinkertaisia karttamerkkejä ja jäljittää reittiä. Tämä ei kuitenkaan enää riitä, vaan karttapohjaan pitäisi pystyä luomaan erilaisia visualisointeja kuten monikulmioalueita sekä lämpökarttakerroksia. Kuvassa 1 esitetään nykyinen karttaratkaisu käyttöliittymässä.



Kuva 1. IoT-TICKET tuotteen nykyinen karttaratkaisu esitettynä kuvassa. (IoT-TICKET 2022).

Nykyinen karttaratkaisu ei tarjoa riittävää kattavuutta toiminnoista, joita voitaisiin hyödyntää myös uusissa käyttötapauksissa. Tekniikka on vanhentunut eikä se esimerkiksi tarjoa riittävää tukea moderneille sovelluskehyksille kuten Reactille tai Angularille.

Ongelmatilanne tarjosi hyvän mahdollisuuden opinnäytetyölle, jossa tutustutaan nykyaikaisten karttaratkaisujen ominaisuuksiin ja arvioidaan mitkä niistä sopivat yrityksen tarpeisiin sekä IoT-TICKET tuotteessa että muissa tulevissa hankkeissa.

Opinnäytetyön aikana pyritään vastaamaan seuraaviin tutkimuskysymyksiin:

1) Mitä karttaratkaisuja on tällä hetkellä tarjolla?
2) Mitä eroja karttaratkaisuista löytyy?
3) Mitkä karttaratkaisut ovat suosituimpia kehittäjäyhteisön keskuudessa?

1.2 Tutkimuksen tavoitteet

Alkuvaiheessa määriteltiin tutkimuksen tavoitteet sekä toimeksiantajalle, tekijälle sekä lukijalle. Tavoitteiden tarkoituksena on tarjota suuntaviivat opinnäytetyön toteutuksessa.

Toimeksiantajan tavoitteena oli saada selvitys nykyaikaisista paikkatietojärjestelmistä sekä erilaisista karttaratkaisuista. Selvityksen haluttiin keskittyvän erityisesti yksittäisten karttaratkaisujen piirteisiin, sekä hyviin että huonoihin puoliin. Optimaalisen ratkaisun löydyttyä haluttiin myös käytännön esimerkki konseptin toimivuudesta kyseisen karttaratkaisun avulla.

Lopputuloksena valmistuvaa raporttia voitaisiin käyttää tulevaisuudessa auttamaan optimaalisen karttaratkaisun valinnassa useissa erilaisissa käyttötapauksissa. Opinnäytetyöprosessin aikana myös opinnäytetyöntekijällä olisi mahdollisuus kehittää osaamistaan käytetyistä teknologioista, joista opittuja

taitoja voisi tulevaisuudessa hyödyntää myös erilaisissa toimeksiantajan hankkeissa.

Tekijän tavoitteena oli saada kaikki hyöty irti opinnäytetyöprosessista ja kehittää samalla omia taitojaan erityisesti ohjelmistokehityksen parissa. Opinnäytetyö tarjosi mahdollisuuden karttaratkaisujen opiskelemisen lisäksi myös syventää tietotaitoa nykyaikaisten ohjelmointikielten sekä kehysten parissa, kuten TypeScriptin ja Reactin. Teknisen osaamisen kehityksen lisäksi tekijä halusi myös laajentaa osaamistaan projektityössä vastaamalla koko prosessista aina suunnittelusta toteutukseen ja raportointiin.

Opinnäytetyöntekijä halusi lisäksi saada kokonaiskuvan erilaisista nykyaikaisista karttaratkaisuista ja kartuttaa osaamistaan erilaisista paikkatietojärjestelmistä mahdollisimman laajasti. Motivaatiota opinnäytetyön tekemiseen kasvatti myös se, että kyseisen aihealueen osaaminen voisi avata tulevaisuudessa mahdollisuuksia työskennellä erilaisissa karttaratkaisuihin liittyvissä projekteissa sekä hankkeissa.

Opinnäytetyön päämääränä on tarjota lukijalle kattava tietopaketti nykyaikaisista karttaratkaisuista ja auttaa optimaalisen ratkaisun valinnassa erilaisissa käyttötapauksissa. Työssä pyritään keräämään sekä esittämään tietoa, mielipiteitä sekä näkökulmia nykyaikaisista karttaratkaisuista että niiden eroista.

1.3 Tutkimuksen menetelmät

Opinnäytetyön aikana tekijän tarkoituksena on kerätä tietoja, jotka pyrkivät ennemminkin kuvailemaan aihetta kuin mittaamaan sitä. Työssä esitetään lisäksi mielipiteitä sekä erilaisia näkökulmia. Opinnäytetyö toteutetaan siis kvalitatiivisena eli laadullisena tutkimuksena.

Aineisto hankitaan muun muassa aihealueeseen liittyvistä artikkeleista sekä karttaratkaisujen dokumentaatiosta. Työssä hyödynnetään tutkimusdokumentteja erilaisten ratkaisujen suosiosta ja käyttäjäkokemuksista

esimerkiksi StackOverFlow- sekä GitHub-sivustojen avulla. Opinnäytetyö sisältää myös käytännön osuuden, joka mahdollistaa havainnoinnin ohjelmistokehittäjän näkökulmasta.

1.4 Toimeksiantaja

Wapice on vuonna 1999 perustettu suomalainen täyden palvelun ohjelmistoyritys, jonka ratkaisut ovat alan johtavien teollisuusyritysten käytössä ympäri maailmaa. (Wapice 2022).

Yritys työllistää yli 330 korkeasti koulutettua ohjelmisto- ja elektroniikka-asiantuntijaa ja se keskittyy tarjoamaan erityisesti teollisuusyrityksille ohjelmisto-osaamista, elektroniikkasuunnittelua, innovaatioita ja parhaiden käytäntöjen konsultointia.

Wapice Ltd:llä on useita omia tuotteita, joita ovat IoT-TICKET, Summium CPQ, Summium Selector, EcoReaction, WRM247+ ja CanRunner. Tuotteista ylivoimaisesti suosituin on kuitenkin ensimmäisenä mainittu IoT-TICKET.

Kyseinen IoT-TICKET -tuote on kattava Internet of Things (IoT) -työkalusarja ja -alusta. Se hyödyntää big data -analytiikkaa ja helppokäyttöisiä työkaluja mahdollistaen web-, mobiili-, pilvi- ja raportointisovellusten luomisen helposti ja nopeasti ilman riviäkään koodia. Wapice on saanut tunnustusta IoT-TICKET -ratkaisusta esimerkiksi Microsoftilta Partner of the Year palkinnon myötä.

2 KARTTARATKAISUT

Opinnäytetyön ensimmäisenä vaiheena oli tutustua mahdollisiin karttaratkaisuvaihtoehtoihin. Tarkoituksena oli löytää karttaratkaisuja, jotka tarjoavat hyvää tukea moderneille sovelluskehyksille, omaavat laajan valikoiman visualisointimahdollisuuksia ja ovat suosittuja kehittäjäyhteisön keskuudessa. Tehdyn kartoituksen perusteella selvisi nopeasti, että Leaflet, OpenLayers, Mapbox sekä Google Maps ovat karttaratkaisujen kentällä suosituimpia vaihtoehtoja.

2.1 Karttaratkaisujen esittely

Tämän kappaleen tarkoituksena on esitellä Leaflet, OpenLayers, Mapbox sekä Google Maps -ratkaisut ja tarjota lisätietoa yksittäisten ratkaisujen ominaisuuksista sekä suosiosta kehittäjäyhteisössä.

2.1.1 LeafletJS

Leaflet on Vladimir Agafonkin kehittämä vuonna 2011 julkaistu avoimen lähdekoodin JavaScript-kirjasto verkkokarttasovellusten rakentamiseen. Kirjaston avulla kehittäjä pystyy luomaan sekä kustomoimaan erilaisia interaktiivisia tasoja, kuten karttamerkkejä ponnahdusikkunoilla.

Kirjasto on tällä hetkellä suosituin ratkaisu ilmaisten avoimen lähdekoodin karttakirjastojen keskuudessa. Se toimii tehokkaasti kaikilla pöytä- ja mobiilialustoilla, sitä voidaan laajentaa monilla laajennusosilla, sillä on helppokäyttöinen ja hyvin dokumentoitu sovellusliittymä ja yksinkertainen, helposti luettava lähdekoodi. (Leaflet 2021).

Leafletin muista ratkaisuista erottava piirre on sen pieni koko. Kirjasto on suunniteltu mahdollisimman kevyeksi ja muihin tarjolla oleviin ratkaisuihin verrattaessa siitä löytyy valmiina vain melko pieni määrä visualisointi- sekä kustomointimahdollisuuksia. Kirjastoon on kuitenkin tarjolla satoja kappaleita

erilaisia laajennusosia, jotka auttavat kehittäjää saamaan kirjastosta kaiken irti. Ilman näitä laajennuksia Leaflet ei tarjoa riittävää kattavuutta ominaisuuksista kehittäessä monimutkaisempia paikkatietojärjestelmäsovelluksia.

Leaflet auttaa käyttäjää olemaan vuorovaikutuksessa karttatietojen kanssa, mutta se ei itse tarjoa valmista karttapohjaa. Kirjasto on riippuvainen kolmannen osapuolen toimittamasta peruskartasta ja sitä voidaan käyttää useiden erilaisten peruskarttakerrosten kanssa. Karttapohjien tarjoajia ovat esimerkiksi Mapbox sekä Esri, mutta yleensä kirjastoa käytetään suositun OpenStreetMaps-karttapohjan yhteydessä.

Kehittäjän näkökulmasta yksi Leafletin hyvistä puolista on laaja tarjonta ratkaisuun liittyvästä materiaalista. Kirjaston oma dokumentaatio on hyvin jäsennelty, kattava ja se tarjoaa myös esimerkkejä. Leafletin kehittäjäyhteisö on suuri ja aktiivinen, joten lisämateriaalia ja apua ongelmanratkaisuun on tarjolla paljon. Esimerkiksi ”Leaflet StackOverFlow” google-hakupyyntö palauttaa noin 354 000 tulosta.

Leaflet ei suoraan tarjoa tukea nykyaikaisille sovelluskehyksille kuten Reactille, Angularille tai Vuelle. Saatavilla on kuitenkin useita kolmannen osapuolen tarjoamia lisäosia, jotka helpottavat kyseisten sovelluskehysten käyttöä Leafletin kanssa. Lisäosista suosituin on ”React-Leaflet”, joka yhdistää Reactin ja Leafletin ominaisuuksia. Se ei korvaa Leaflet-kirjastoa, mutta hyödyntää sitä abstraktien Leaflet-kerrosten React-komponentteina. (React-Leaflet 2021). Angularille sekä Vuelle kehitetyt lisäosat ovat huomattavasti pienempiä eivätkä tarjoa yhtä suurta määrää ominaisuuksia. React on ylivoimaisesti suosituin sovelluskehys, jota käytetään Leaflet kirjaston yhteydessä.

2.1.2 OpenLayers

OpenLayers on kypsempi ja rikkaampi avoimen lähdekoodin JavaScript-kirjasto web-karttojen rakentamiseen, mutta muuten hyvin samanlainen kuin Leaflet.

(Dorman 2018). OpenLayers on kuitenkin monimutkaisempi, kooltaan suurempi ja kehittäjälle vaikeampi kirjasto opetella. Esimerkiksi Leaflet-kirjastoa voidaan pitää kevyempänä ja keskittyneempänä vaihtoehtona OpenLayersille.

OpenLayers on kehitetty edistämään kaikenlaisen maantieteellisen tiedon käyttöä. Se on täysin ilmainen, avoimen lähdekoodin JavaScript-kirjasto, joka on julkaistu ”2-clause BSD” lisenssillä. Se tarjoaa rikkaan API:n (Application Programming Interface) eli rajapinnan, joka mahdollistaa yksinkertaisten ja erittäin monimutkaisten karttasovellusten luomisen ja tarjoaa enemmän toimintoja kuin Leaflet.

Kirjastoa käytetään pääasiassa monimutkaisten paikkatietojärjestelmä sovellusten kehittämiseen, jonka myötä kehittäjäyhteisö on huomattavasti pienempi kuin Leaflet-kirjastolla. OpenLayersin API-dokumentaatio on kuitenkin kattava ja hyvin jäsennelty.

OpenLayersin käyttämisestä Reactin ja Angularin kaltaisten sovelluskehysten kanssa on myös saatavilla materiaalia, mutta ei kuitenkaan samalla tasolla kuin esimerkiksi Leaflet-kirjastolla. Se ei suoraan tue sovelluskehyksiä, jonka myötä suuri osa materiaalista on tarkoitettu vain JavaScript-ohjelmointikielelle.

OpenLayers ei tarjoa valmista karttapohjaa, jonka myötä se on Leafletin tavoin riippuvainen kolmannen osapuolen toimittamasta peruskartasta. Käyttäjällä on mahdollisuus valita haluamansa karttapohja useista eri lähteistä, kuten OpenStreetMaps, Bing, Mapbox tai Stamen.

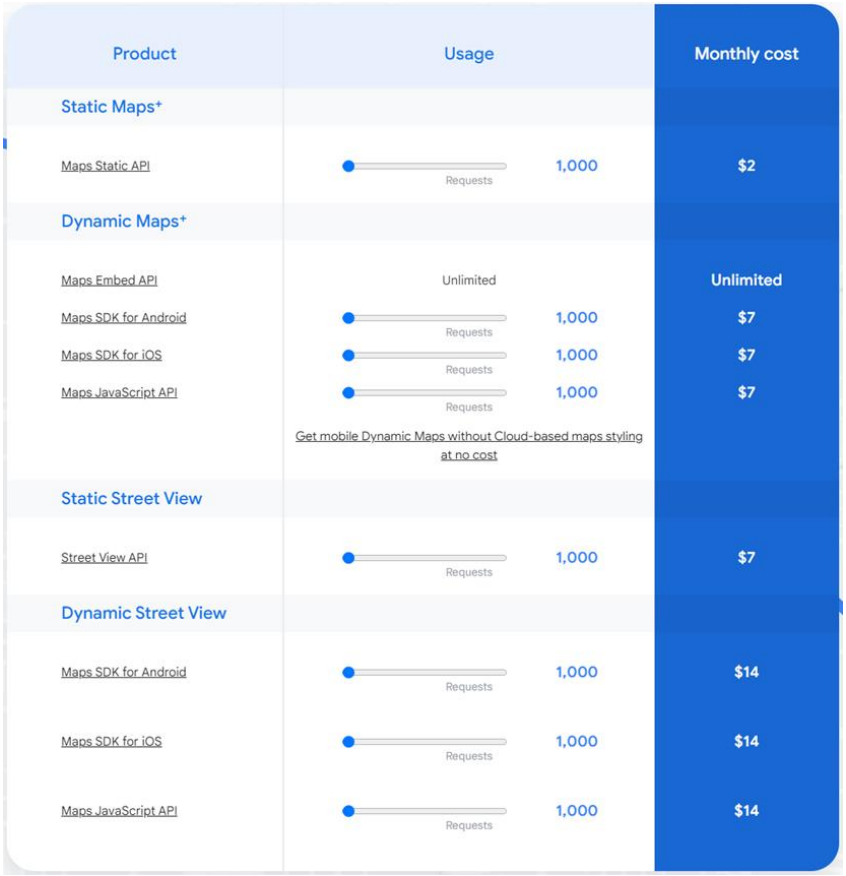
2.1.3 Google Maps

Google Maps on johtava verkkokarttapalvelu, jonka Google on kehittänyt ja ylläpitänyt. Tämä alusta on joukko SDK:ita (Software Development Kit) ja API:ita, joiden avulla voidaan luoda mukautettuja karttoja verkkosivustoille ja mobiilisovelluksille. (Piotrovskyi 2021).

Karttaratkaisuna Google Maps on turvallinen valinta kenelle tahansa. Se on yksi vanhimmista karttaratkaisuista ja vakiinnuttanut asemansa karttateollisuuden johtajana. Yksi houkuttelevimmista piirteistä on helppo integroitavuus mihin tahansa sovellukseen, verkkosivustoon tai palveluun. Tämä onnistuu JavaScript API:n avulla.

Google Maps tarjoaa suuren määrän erilaisia ominaisuuksia. Kuitenkin verrattaessa muihin karttaratkaisuihin, se ei ole yhtä joustava kustomoinnin ja visualisointien suhteen. Ratkaisua on mahdollista käyttää myös sovelluskehysten kuten Angular ja React yhteydessä.

Yksi Google Mapsin huonoimmista puolista on sen monimutkainen ja kallis hinnoittelu. Se käyttää jako-hinnoittelumallia ja hinta vaihtelee tapauskohtaisesti asiakkaan käyttämien palveluiden mukaan. Google on laatinut verkkosivustolleen hinnoittelutaulukon, jotta käyttäjät saavat paremman käsityksen hinnoista. Taulukko esitetään kuvassa 2.



Product	Usage	Monthly cost
Static Maps+		
Maps Static API	1,000 Requests	$2
Dynamic Maps+		
Maps Embed API	Unlimited	Unlimited
Maps SDK for Android	1,000 Requests	$7
Maps SDK for iOS	1,000 Requests	$7
Maps JavaScript API	1,000 Requests	$7
	Get mobile Dynamic Maps without Cloud-based maps styling at no cost	
Static Street View		
Street View API	1,000 Requests	$7
Dynamic Street View		
Maps SDK for Android	1,000 Requests	$14
Maps SDK for iOS	1,000 Requests	$14
Maps JavaScript API	1,000 Requests	$14

Kuva 2. Google Maps hinnoittelutaulukko. (Google Maps 2022).

2.1.4 Mapbox

Mapbox on yksi edistyneimmistä karttapalveluista ja yksi parhaista vaihtoehdoista Google Map:sille tällä hetkellä. Mapbox on avoimen lähdekoodin alusta, jonka kehittäjät ovat luoneet kehittäjille. (Piotrovskyi 2021).

Ratkaisu tarjoaa työkalut mukautettujen dynaamisten sekä staattisten karttojen luomiseen sekä mobiilisovelluksille, että verkkosivustoille. Käyttäjät saavat täyden hallinnan karttojen tyyliin. On syytä huomata, että alusta tarjoaa myös oikeuden käyttää suurta määrää erilaisia tekstuureja, piirroksia, mukautettuja merkkejä, staattisia karttoja ja paljon muuta.

Suurin osa Mapboxin käyttämästä datasta on avoimesti saatavilla. Ne tarjoavat tuoreita päivityksiä ja pysyvät täsmällisinä nopeasti kehittyvien tietojen avulla. Mapbox käyttää lähteinä myös muita tunnettuja palveluita, kuten

OpenStreetMaps, Landsat, United States Geological Survey, Natural Earth ja OpenAddresses.

Mapbox tarjoaa kattavan joukon ominaisuuksia ja työkaluja karttapalveluiden integroimiseksi mihin tahansa verkkosivustoon tai mobiilisovellukseen. Se sisältää myös laajennuksia suosittuihin sovelluskehyksiin, kuten Angular ja React.

Mapbox ei ole täysin ilmainen ja sen hinnoittelu on tehty hieman hankalaksi. Ratkaisu on ilmainen 25 000 mobiilikäyttäjälle ja 50 000 verkkolataukselle, jonka jälkeen Mapbox veloittaa palvelustaan.

2.2 Karttaratkaisujen suosio

Yksi tärkeä karttaratkaisun valintaan vaikuttava asia on ratkaisun suosio muiden kehittäjien ja yritysten keskuudessa. Seuraavaksi tarkoituksena on esittää erilaisten karttaratkaisujen suosio kehittäjäyhteisön keskuudessa erilaisten taulukoiden ja kaavioiden avulla. Havainnollistaviin taulukoihin ja kaavioihin on kerätty dataa Stack Overflow-, Github-, ja NPM Trends sivustoilta.

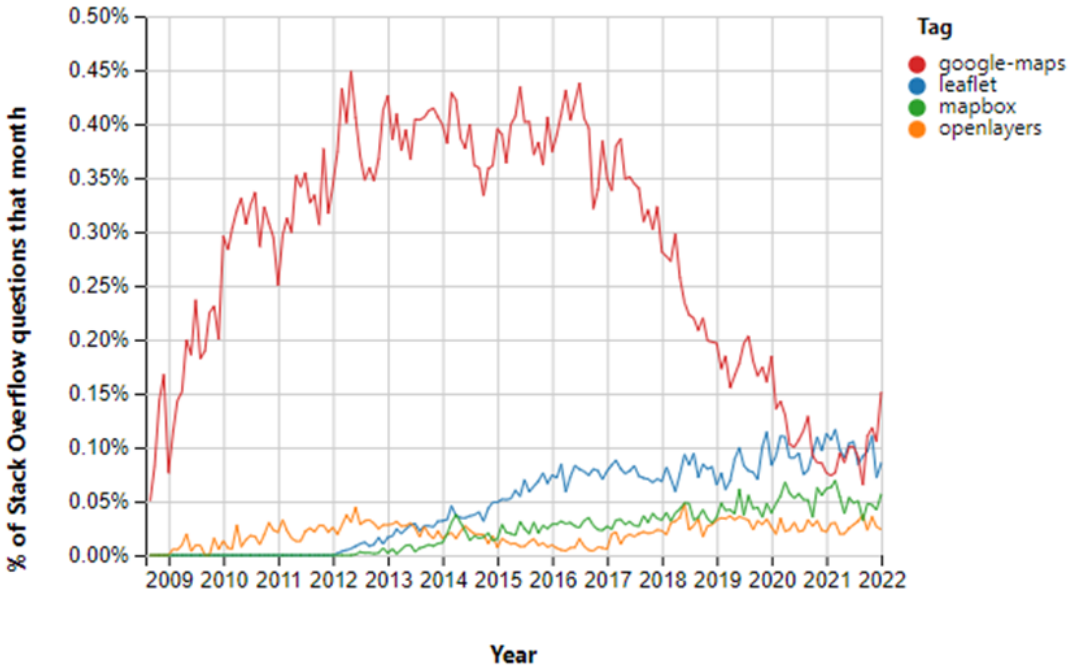
2.2.1 Stack Overflow

Stack Overflow on kysymys- ja vastaussivusto ammattilaisille ja innokkaille ohjelmoijille. Se on Jeff Atwoodin ja Joel Spolskyn vuonna 2008 luoman Stack Exchange Networkin lippulaivasivusto. Sivusto sisältää kysymyksiä ja vastauksia monista tietokoneohjelmoinnin aiheista. Sivusto luotiin alun perin avoimemmaksi vaihtoehdoksi aikaisemmille kysymys- ja vastaussivustoille, kuten Experts-Exchangelle. (Wikipedia 2022).

Stack Overflow on myös erinomainen sivusto tutkia esimerkiksi erilaisten ohjelmointikielien, alustojen sekä kirjastojen kiinnostavuutta ohjelmointiyhteisössä. Se tarjoaa luotettavan datan lähteen mittaamalla ratkaisujen suosiota sivustolle lähetettävien kyselyjen määrän avulla. Kyselyt voivat olla mitä tahansa kyseiseen tekniikkaan liittyvää, jossa kehittäjä hakee apua

ohjelmointiyhteisöltä ongelman ratkaisuun. Kyselyjen määrä kertoo omalla tavallaan esimerkiksi ratkaisun suosiosta kehittäjäyhteisössä.

Kaaviossa 1 on vuosien 2009 – 2022 ajalta kysymysten määrä aiemmin esitellyistä karttaratkaisuista. Kaaviosta voidaan päätellä, että Google Maps sekä Leaflet ovat johtavia ratkaisuja ohjelmointiyhteisön keskuudessa. Google Maps on yksi markkinoiden vanhimmista karttaratkaisuista ja sen myötä yksi tunnetuimmista sekä suosituimmista. Äkillisen laskun ratkaisun suosiossa toi Google Mapsin muuttuminen maksulliseksi palveluksi vuonna 2018.



Kaavio 1. Kysymysten määrä Stack Overflow -sivustolla. (Stack Overflow 2022).

2.2.2 GitHub

GitHub on suosittu ohjelmointiresurssi, jota käytetään koodin jakamiseen. Se on ohjelmoijien sosiaalisen verkostoitumisen sivusto, jota monet yritykset ja organisaatiot käyttävät helpottaakseen projektinhallintaa ja yhteistyötä. Lokakuussa 2020 kerättyjen tilastojen mukaan se on merkittävin lähdekoodien

varasto, sillä vuonna 2020 luotiin yli 60 miljoonaa uuttaa tietovarastoa ja yhteensä yli 56 miljoonaa kehittäjää. (Gaba 2022).

Stack Overflown tavoin myös GitHub tarjoaa käyttäjille tilastoja esimerkiksi erilaisten kirjastojen sekä ratkaisujen suosiosta kehittäjäyhteisössä. Sivuston tarjoama data voidaan jakaa kolmeen pääkohtaan: tähtiin, haaroihin sekä ongelmiin, joihin pystyvät vaikuttamaan kaikki sivuston käyttäjät omalla toiminnallaan.

Käyttäjä pystyy merkitsemään arkiston tähdellä, joka toimii eräänlaisena kirjanmerkkinä käyttäjälle sekä arvostuksen osoituksena tekijälle. Useimmissa tapauksissa tähtien määrä kertoo arkiston laadusta. Mitä enemmän tähtiä, sitä pidetympi arkisto on ollut kehittäjäyhteisön keskuudessa. Ongelmat taas kertovat virhetilanteista ja arkiston huonoista puolista.

Haara on kopio arkistosta. Sen avulla käyttäjä voi vapaasti kokeilla tehdä muutoksia vaikuttamatta alkuperäiseen projektiin. Yleisimmin haaroja käytetään joko ehdottamaan muutoksia jonkun toisen projektiin tai käyttämään jonkun muun projektia lähtökohtana omalle idealle. (GitHub Docs 2021).

Taulukossa 1 on esiteltynä karttaratkaisujen statistiikkatiedot. Sivuston tarjoamasta datasta voidaan päätellä Leaflet karttakirjaston olevan suuressa suosiossa kehittäjäyhteisössä. Taulukko ei sisällä Google Maps-ratkaisua, sillä GitHub ei voi jakaa sen lähdekoodia.

Taulukko 1. Karttaratkaisujen tähdet, haarat sekä ongelmat GitHubissa. (GitHub 2022).

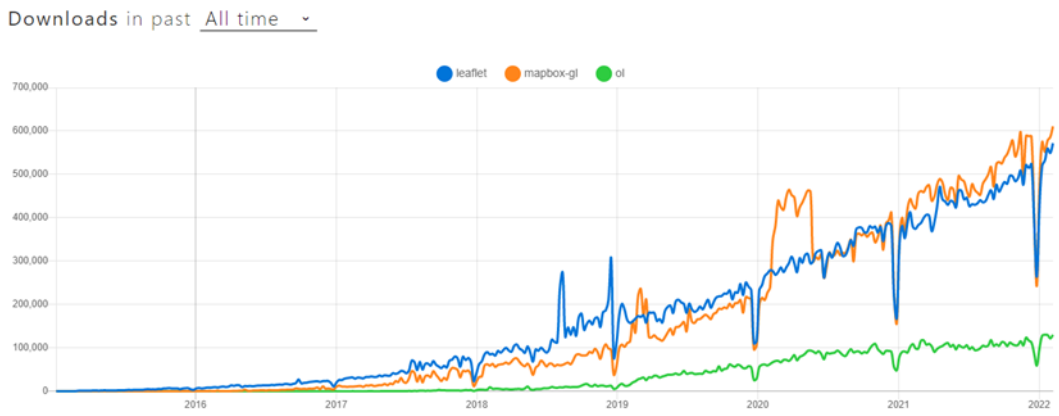
	Tähdet	Haarat	Ongelmat
Leaflet	32.9k	5.1k	383
OpenLayers	8.8k	2.7k	246

Mapbox	8.3k	1.9k	865

2.2.3 NPM Trends

NPM on lyhenne sanoista Node Package Manager, ja se on maailman suurin ohjelmistorekisteri Javascript-ohjelmointikielelle. NPM on online-varasto avoimenlähdekoodin Node.js-projektien julkaisemiseen sekä komentorivityökalu vuorovaikutukseen mainitun arkiston kanssa, joka auttaa pakettien asennuksessa, versionhallinnassa ja riippuvuuksien hallinnassa. (Node.js 2011).

NPM Trends tarjoaa dataa sivustolta esimerkiksi erilaisten kirjastojen suosiosta. Kaaviossa 2 näytetään dataa yksittäisten karttaratkaisujen latauskerroista NPM:n avulla. Kaaviosta huomataan, että latauskerroissa Mapbox päihittää sekä Leaflet-, että OpenLayers karttaratkaisut.



Kaavio 2. Karttaratkaisujen latauksien määrä Npm Trends -sivustolla. (Npm Trends 2022).

2.3 Karttaratkaisujen ominaisuudet

Karttaratkaisujen tarjoamat toiminnot sekä visualisoinnit ovat oleellinen osa valintaa. Taulukon 2 avulla voidaan todeta, että kaikki esitellyistä ratkaisuista tarjoavat hyvän määrän erilaisia ominaisuuksia.

Samalla on tärkeää tiedostaa, että ominaisuuksien määrä ei välttämättä tarkoita, että karttaratkaisu olisi toista parempi. Se riippuu täysin käyttötapauksesta. Esimerkiksi OpenLayers näyttää taulukon 2 mukaan omaavan isoimman määrän ominaisuuksia. Kehittäjälle voi olla kuitenkin tärkeää, että ratkaisussa on valmis karttapohja. Tässä tapauksessa OpenLayers sekä Leaflet eivät sovi kriteereihin ja vähemmän muita ominaisuuksia omaavat Mapbox ja Google Maps vaikuttavat paremmilta vaihtoehdoilta.

Joissain tapauksissa voi olla tärkeää huomioida myös kirjaston koko. Mitä enemmän ominaisuuksia, sitä suurempi tiedosto. Esimerkiksi Leaflet on melko minimalistinen tuote, jonka koko on vain 140kb. Leaflet on pieni ja mahdollisesti laajennettavissa laajennuksilla, mutta sen tarkoitus ei ole olla täysin täysimittainen verkkopohjainen paikkatietojärjestelmä. OpenLayers taas on täysimittainen karttakirjasto, joka sisältää kaikki vaadittavat ominaisuudet ja on kooltaan jopa 644kb (versio 6). Mapboxin koko taas on 732kb (versio 1.7). (Bac Ha Software 2020).

Taulukko 2. Karttaratkaisujen ominaisuudet taulukossa. (Frerichs 2021).

Name	Leaflet	OpenLayers	Mapbox	Google Maps
Projections	Via Plugin	Integrated	Not available	Non trivial
Dynamic styling	Integrated	Integrated	Integrated	Integrated
Default basemaps	Not available	Not available	Integrated	Integrated
Use your own basemap	Integrated	Integrated	Integrated	Not available
Marker clustering	Via Plugin	Integrated	Integrated	Integrated
Tooltip	Integrated	Integrated	Integrated	Integrated
User drawing	Via Plugin	Integrated	Via Plugin	Not available
Geocoder	Via Plugin	Not available	Integrated	Integrated
Image tiles	Integrated	Integrated	Integrated	Limited
Raster data	Via Plugin	Integrated	Limited	Limited
Raster styling	Via Plugin	Integrated	Not available	Not available
Heatmap	Via Plugin	Integrated	Integrated	Integrated
OGC Services (WMS, WFS)	Via Plugin	Integrated	Limited	Not available
Company support	Third-Party	Third-Party	Creator	Creator

3 TEKNOLOGIAT JA RAJAPINNAT

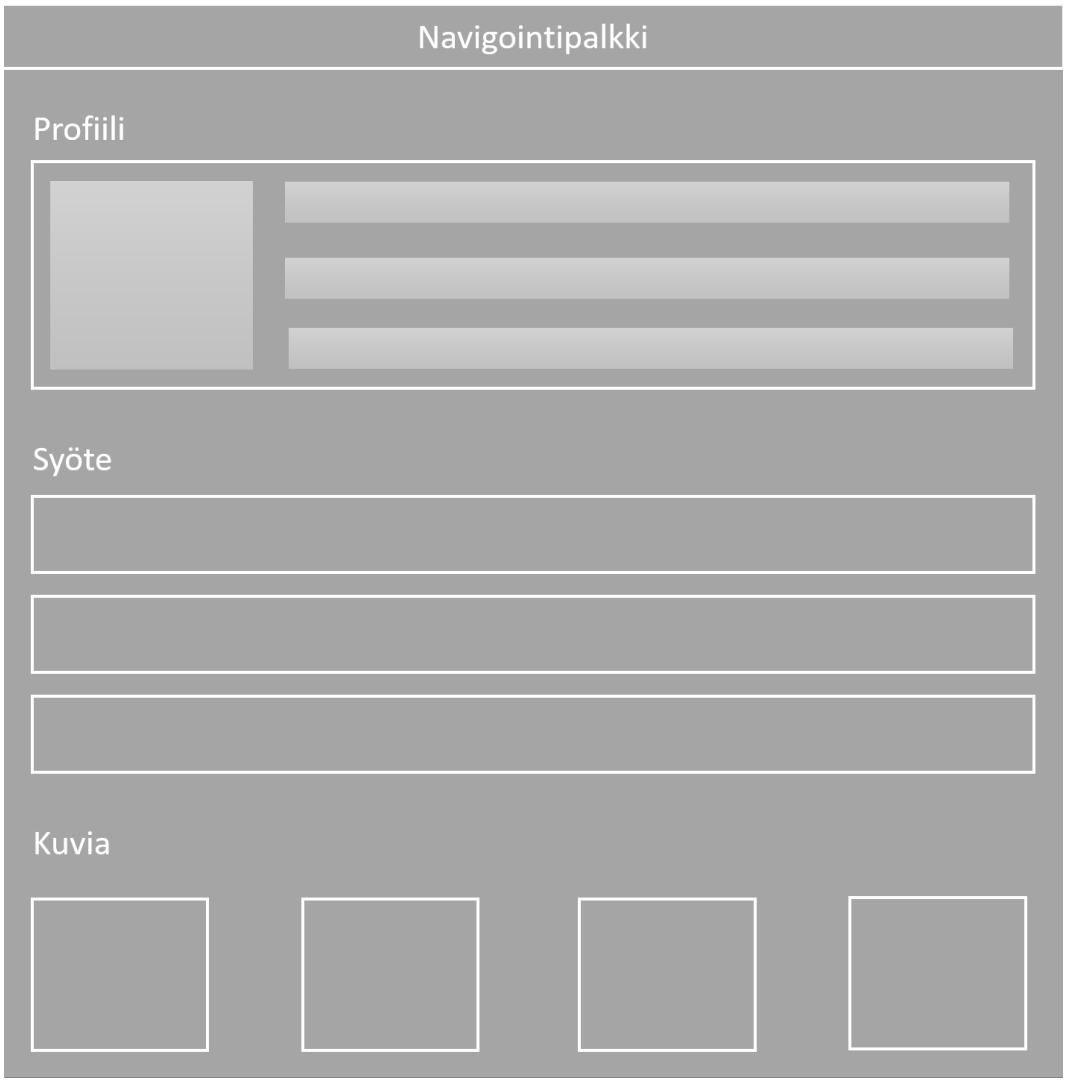
Tässä luvussa esitellään tulevassa käytännön toteutuksessa hyödynnetyt ohjelmointiteknologiat. Karttaratkaisun valitsemisessa on ensisijaisen tärkeää sen toiminta nykyaikaisten ohjelmointikielten sekä ohjelmointikehysten parissa. Tässä luvussa esitellyt teknologiat on valittu erityisesti siksi, että ne ovat hyvin suosittuja nykyaikaisessa ohjelmistokehityksessä.

3.1 React

Nykypäivänä verkkosovellukset ovat yhä suurempia kokonaisuuksia, joiden rakentaminen esimerkiksi pelkästään JavaScript-ohjelmointikielen avulla voi olla kehittäjille erittäin työlästä ja aikaa vievää. Kehityksen tukena hyödynnetäänkin usein erilaisia sovelluskehyksiä, jotka tekevät sovellusten rakentamisesta nopeampaa ja niiden laajentamisesta myös tulevaisuudessa kehittäjälle helpompaa.

React on avoimen lähdekoodin JavaScript-kirjasto, jota käytetään käyttöliittymien rakentamiseen erityisesti yksisivuisille sovelluksille. Ratkaisun avulla käyttöliittymä voidaan jakaa useisiin erilaisiin komponentteihin, jotka ovat itsenäisiä ja uudelleenkäytettäviä koodibittejä. Mahdollisuus uudelleenkäyttää ja jakaa koodia pienempiin osa-alueisiin lyhentää sovelluksen kehitysaikaa huomattavasti ja tekee tulevaisuudessa myös jatkokehittämisestä helpompaa.

Kuvassa 3 havainnollistetaan, miten Reactin avulla verkkosivu voidaan jakaa useampaan eri komponenttiin. React mahdollistaa komponenttien uudelleen käyttämisen, joten esimerkiksi ”Syöte”-kohdassa havainnollistetuissa laatikoissa ei kehittäjän tarvitse kirjoittaa yhä uudelleen samaa koodia, vaan hän voi kutsua ennalta määriteltyä komponenttia yhä uudelleen. Komponentin sisältöä on mahdollista vaihdella tapauskohtaisesti parametrisoimalla kyseinen funktio.



Kuva 3. Käyttöliittymä jaettuna React komponentteihin.

3.2 JavaScript

Vuonna 1995 luotu JavaScript on tekstipohjainen avoimen lähdekoodin komentosarja- ja ohjelmointikieli, jota käytetään tekemään verkkosivustoista interaktiivisempia. Se tunnetaan turvallisuudestaan ja siirrettävyydestään käytännöllisesti katsoen kaikissa käyttöjärjestelmissä. Sen avulla pystytään esimerkiksi lisäämään dynaamista tekstiä suoraan verkkosivun HTML-koodiin ja toimimaan automaattisesti sivun latautuessa. (Baldwin Draper 2021).

Web-kehittäjät käyttävät JavaScriptiä lisätäkseen vuorovaikutteisuutta ja toimintoja verkkosivulle. Tämä voi sisältää esimerkiksi ilmoitusten tai ponnahdusviestien näyttämisen, hiiren napsautuksiin vastaamisen, animaatioiden luomisen tai tietojen vahvistamisen. Sitä voidaan käyttää esimerkiksi verkkosivun sisällön dynaamiseen päivittämiseen sivuston latautumisen jälkeen. JavaScriptiä voidaan käyttää myös taustalla tietojen tallentamiseen ja jakamiseen. (Shokeen 2021).

JavaScript tukee dynaamista kirjoittamista, mikä tarkoittaa että muuttujan tyypit määritellään tallennetun arvon perusteella. Jos esimerkiksi määrität muuttujan x, voit tallentaa joko merkkijonon, numerotyypin arvon, taulukon tai objektin. Tätä kutsutaan dynaamiseksi kirjoittamiseksi. Esimerkiksi Javan kaltaisella kielellä on mainittava nimenomaisesti, että tietty muuttuja tallentaa tietyn tyyppistä dataa, kun taas JavaScriptissä ei tarvitse antaa tietotyyppiä muuttujaa määrittäessä. JavaScriptissä on vain käytettävä avainsanaa ”var” tai ”let” muuttujan ilmoittamiseksi sen tyypistä riippumatta. (Studytonight 2021).

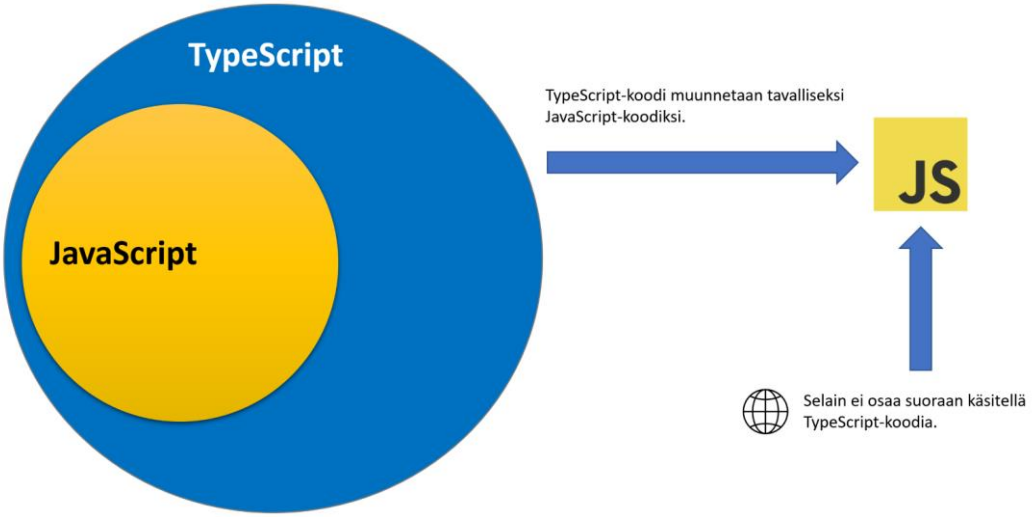
JavaScript on tällä hetkellä yksi maailman suosituimpia ohjelmointikieliä. Syitä menestykseen löytyy useita, kuten JavaScriptin helppo integroitavuus muihin ohjelmointikehyksiin sekä ohjelmointikieliin. Tämän lisäksi avoimen lähdekoodin JavaScriptillä on ollut merkittävä rooli digitaalisessa transformaatiossa luomalla interaktiivisia web-sivuja käyttöliittymäkehyksiä hyödyntäen. Suosioon on vaikuttanut myös se, että kaikki suosituimmat verkkoselaimet kuten Google Chrome, Internet Explorer, Edge, Firefox, Safari sekä Opera tukevat JavaScriptiä.

3.3 TypeScript

JavaScript kehitettiin alun perin asiakaspuolen ohjelmointikieleksi, mutta pian kehittäjät huomasivat, että sitä voidaan käyttää myös palvelinpuolen ohjelmointikielenä. Kun JavaScript kasvoi ajan myötä, siitä tuli huomattavasti monimutkaisempi eikä se kyennyt saavuttamaan täyttä potentiaaliaan menestyä yrityspuolella palvelinpuolen teknologiana. Tämän seurauksena kehitettiin

JavaScriptin supersetti nimeltään TypeScript, jonka tarkoitus on poistaa koodin monimutkaisuus suurissa projekteissa. (Pang, 2021).

TypeScript on avoimen lähdekoodin puhdas oliopohjainen ohjelmointikieli. Se on vahvasti tyypitetty JavaScriptin supersetti, joka käännetään tavalliseksi JavaScriptiksi. (Javatpoint 2021). Kuvassa 4 havainnollistetaan ero TypeScriptin ja JavaScriptin välillä.



Kuva 4. TypeScript kääntäminen JavaScript-koodiksi. (Raval 2022).

Kuten aiemmin todettu, JavaScript tukee dynaamista kirjoitusta. Tämän vuoksi on vaikea tietää, kuinka käsitellä tietyn muuttujan sisällä olevia asioita. Lisäksi se ei tarjoa staattista kirjoitusta. Staattinen kirjoittaminen tarkoittaa, että kehittäjä ilmoittaa datatyypin, joka muuttujalla voi olla. Jos esimerkiksi ’x’ on ilmoitettu osoittavan vain kokonaislukuja, kääntäjä antaa virheilmoituksen, kun siihen asettaa esimerkiksi merkkijonon. Toisin kuin JavaScript, TypeScript on vahvasti tyypitetty ja mahdollistaa sekä staattisen, että dynaamisen kirjoittamisen.

Staattinen kirjoittaminen on yksi TypeScriptin käyttämisen tärkeimmistä eduista. Sen avulla kehittäjä voi tarkistaa tyypin tarkkuuden käännösaikana. Esimerkiksi JavaScript tarjoaa kieliprimitiivit, kuten merkkijonon ja numeron, mutta se ei tarkista, että kehittäjä on määrittänyt ne johdonmukaisesti. TypeScript taas tekee myös tämän. (Berga & Figueiredo 2021).

Stack Overflow:n vuonna 2021 tekemän kyselyn mukaan TypeScript on seitsemänneksi suosituin ohjelmointikieli noin 83 000 kehittäjän arvioiden mukaan. Kyseinen tutkimus suoritetaan vuosittain ja se on kehittäjäyhteisössä yksi luotettavimmista ja odotetuimmista kehittäjätutkimuksista.

4 KÄYTÄNNÖN TOTEUTUS

Yksi tutkimuksen tavoitteista oli todistaa konseptin toimivuus luomalla verkkopohjainen karttasovellus. Tässä sovelluksessa tarkoituksena oli käyttää apuna yhtä aiemmin tutkimuksessa esiteltyä karttaratkaisua, joka vaikuttaa optimaaliselta vaihtoehdolta tulevaisuuden käyttötapauksiin. Karttaratkaisuksi valikoitui Leaflet, jossa oli teoriaosuuden perusteella useita hyviä piirteitä. Näitä olivat esimerkiksi ratkaisun suuri suosio kehittäjäyhteisössä, visualisointimahdollisuuksien määrä, kirjaston pieni koko sekä erinomainen dokumentaatio.

Tässä luvussa tarkoituksena on esitellä karttasovelluksen toteutusprosessi. Sovellukselle määriteltiin ennalta seuraavat vähimmäiskriteerit toiminnallisuuden osalta:

1) Kartta osaa jäljittää ja esittää sijaintia merkin avulla.
2) Kartta osaa esittää monikulmioalueita.
3) Kartta osaa esittää lämpökarttatyylisesti aktiivisimmat kohdat.

4.1 Kehitysympäristön alustaminen

Opinnäytetyön käytännön toteutuksen ensimmäisenä vaiheena oli alustaa sopiva kehitysympäristö, jonka avulla voidaan arvioida Leafletin toimintaa nykyaikaisten ohjelmointikielien ja sovelluskehysten parissa. Ensimmäisenä vuorossa Reactin ja TypeScriptin asentaminen.

4.1.1 React ja TypeScript

Kehitysympäristön luominen aloitetaan React-peruspohjasta, joka saadaan luotua alla näkyvällä komennolla.

npx create-react-app <sovelluksen nimi tähän> –template typescript

Komennossa esiintyvä npx, eli Node Package Execute on npm-pakettien ajaja, joka voi suorittaa minkä tahansa paketin npm-rekisteristä asentamatta sitä. Komennossa on myös erikseen määritelty TypeScript-ohjelmointikieli, sillä ilman erillistä määrittämistä ja asentamista React-peruspohjan oletuksena käytetään JavaScript-ohjelmointikieltä. Komennossa määritellään myös sovelluksen nimi.

4.1.2 JSON Server

Kehitysympäristöön päätettiin lisätä JSON Server simuloimaan backend REST -palvelua, joka toimittaa tietoja JSON-muodossa sovellukseen ja varmistaa, että kaikki toimii odotetulla tavalla.

JSON on lyhenne sanoista ”JavaScript Object Notation”. Se on avoin standarditiedostomuoto ja tiedonvaihtomuoto. Se käyttää luettavaa tekstiä dataobjektien tallentamiseen ja siirtämiseen. JSON koostuu yleensä kahdesta luettavasta attribuutista eli arvopareista ja taulukoista. (Javatpoint 2021).

JSON Server on taas NodeJS-moduuli, jonka avulla voidaan luoda REST JSON-palveluita muutamassa minuutissa. JSON Serverin valinta perustui erityisesti nopeaan käyttöönottoon ja käytön helppouteen. JSON Server saatiin luotua seuraavalla komennolla:

npm install -g json-server

Asennuksen jälkeen luotiin “db.json” niminen tiedosto. Tämä tiedosto sisältää dataa, jotka REST API:n tulee esittää. Kuvassa 5 esitetään luotu tiedosto ”mock-server” kansion sisällä.

Kuva 5. JSON Server -tiedosto.

4.1.3 Leaflet & React-Leaflet

Seuraavana vaiheena oli asentaa kehitysympäristöön Leaflet-karttakirjasto. Hyvän dokumentaation myötä asentaminen oli helppoa, eikä ongelmia ilmennyt. Ympäristöön asennettiin myös lisäosa React-Leaflet, joka yhdistää Reactin ja Leafletin ominaisuuksia ja tarjoaa valmiita React-komponentteja. Asentaminen onnistui seuraavilla komennoilla:

Komento 1, jolla saadaan asennettua Leaflet:

npm install leaflet

Komento 2, jolla saadaan asennettua React-Leaflet:

npm install react-leaflet

Komento 3, jolla saadaan asennettua React-Dom Leaflet:

npm install react react-dom leaflet

Komento 4, jolla saadaan asennettua TypeScript tuki:

npm install -D @types/leaflet

Asennuksien jälkeen oli tärkeää määrittää karttakomponentin korkeus sekä leveys CSS-tiedostossa. Kuvassa 6 annetaan tiedostossa tarvittavat määritykset. Ilman näitä määrityksiä karttakomponentti ei huomaa sille määriteltyä aluetta käyttöliittymässä, eikä se välttämättä näy ollenkaan tai aiheuttaa virhetilanteita.

```
.leaflet-container {
  height: 100vh;
  width: 100vw;
}
```

Kuva 6. Leaflet css -määritykset.

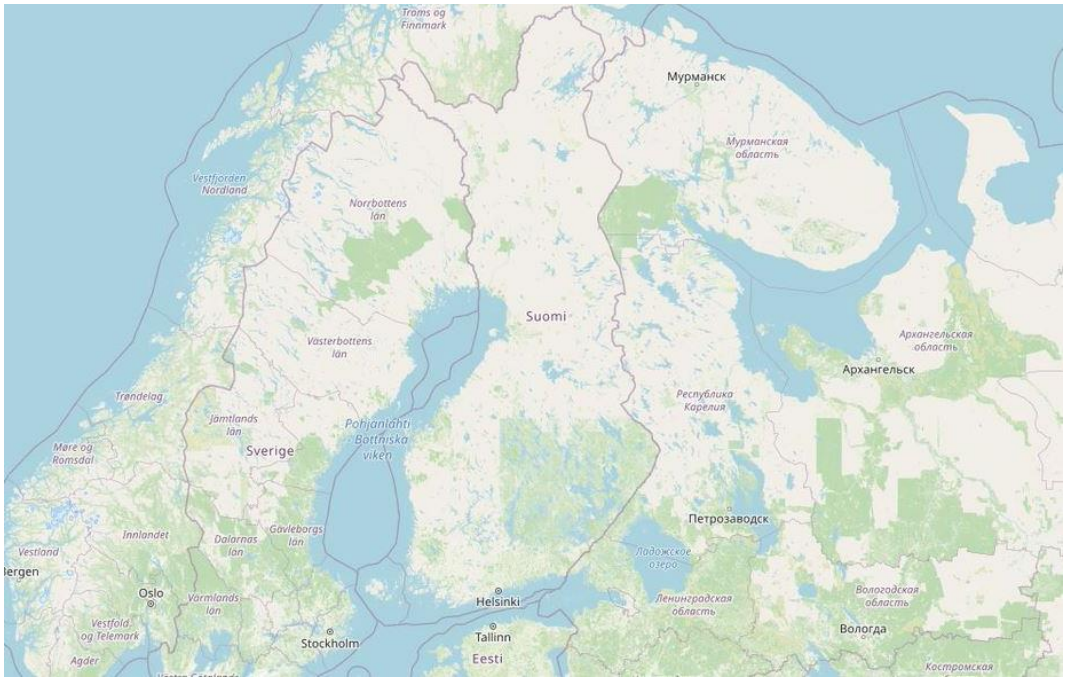
Viimeisenä vaiheena kehitysympäristön pystytyksessä oli tehdä React-komponentti, joka esittää kartan käyttöliittymässä. Komponentin tekemiseen esimerkki löytyi React-Leaflet -lisäosan dokumentaatiosta. Tehdyssä komponentissa oli tärkeää määritellä karttanäkymän zoom-taso, joka rajaa näkymän tiettyyn alueeseen. Toinen tärkeä asia oli määrittää kartan keskipiste, joka alla olevassa kuvassa ”centralizedLocation”-muuttujassa.

Työn karttapohjana päätettiin käyttää OpenStreetMapsia, joka on ilmainen, muokattavissa oleva kartta koko maailmasta, joka on julkaistu avoimen sisällön lisenssillä. Kuvassa 7 määritelty TileLayer lataa kartan kuvat MapContainer-näkymään ja siinä käytetyt karttapohjan kuvat ovat suoraan OpenStreetMapsin sivustolta.

```
import "./BaseMap.css";
import { LatLngTuple } from "leaflet";
import { MapContainer, ZoomControl, TileLayer } from "react-leaflet";

const BaseMap: React.FC = () => {
  // Basic properties.
  const centralizedLocation: LatLngTuple = [61.92411, 25.748152];
  const zoomLevel: number = 5;
  const scrollWheelZoom: boolean = true;

  return (
    <>
      <MapContainer
        zoomControl={false}
        center={centralizedLocation}
        zoom={zoomLevel}
        scrollWheelZoom={scrollWheelZoom}
      >
        <TileLayer
          attribution='&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
          url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
        />
        <ZoomControl position="bottomleft" />
      </MapContainer>
    </>
  );
};

export default BaseMap;
```

Kuva 7. Kartan renderöinti React-Leaflet -komponentteja hyödyntäen.

Seuraamalla tarkasti React-Leafletin dokumentaatiota karttasovelluksen aloittaminen oli helppoa ja melko nopeaa. Kuvassa 8 näkyy karttanäkymä käyttöliittymässä.



Kuva 8. Karttanäkymä käyttöliittymässä.

4.2 Karttamerkit ponnahdusikkunoilla

Karttamerkeillä pystymme esittämään helposti käyttäjälle dataa, joka perustuu sijaintitietoihin. Lisäämällä esimerkiksi ponnahdusikkunoita, voidaan esittää yksittäisistä kohteista lisätietoa. Kuvassa 9 esitetään koodi, jonka avulla karttamerkkeihin pystyttiin lisäämään ponnahdusikkunoita. Ponnahdusikkunat aukeavat käyttäjän klikatessa karttamerkkiä ja näyttävät karttamerkkiin sidottua dataa, tässä tapauksessa Wapicen toimiston sijainnin sekä kuvan.

Koodissa käytetään React-Leaflet -lisäosan tarjoamaa ”Marker”-komponenttia karttamerkin luomiseen, mihin arvoiksi annetaan yksilöllinen tunnus jokaiselle merkille sekä koordinaattitiedot. Karttamerkkiin sidotaan ”Popup”-komponentti, joka mahdollistaa ponnahdusikkunan esittämisen käyttäjälle. Ponnahdusikkunaan on sisällytetty kolme tietoa: yrityksen nimi, sijainti sekä kuva toimistosta. Komponentissa käydään läpi jokainen erillinen paikkatieto ”db.json” -tiedostossa ja luodaan annettujen sijaintitietojen perusteella jokaisen toimiston kohdalla

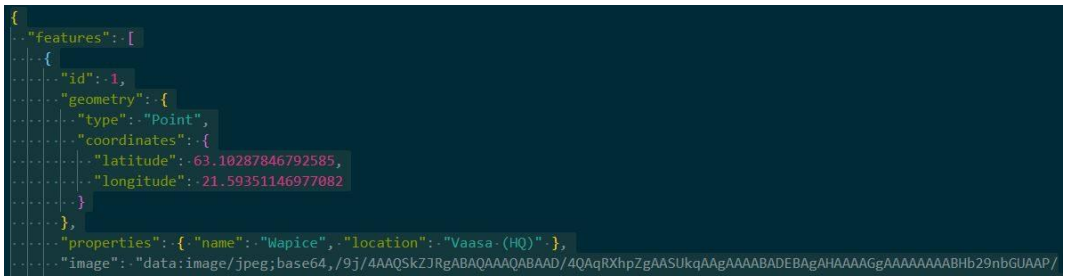
karttamerkki, johon yhdistetään toimistopisteelle yksilöllinen data, eli kuva sekä sijainti. Kuvassa 9 esitetään tähän tarvittava koodi.

```
<LayersControl.Overlay name="Markers">
  <LayerGroup>
    {state &&
      state.map((postDetail: any) => [
        <Marker
          key={postDetail.id}
          position={[
            postDetail.geometry.coordinates.latitude,
            postDetail.geometry.coordinates.longitude,
          ]}
        >
          <Popup>
            <h2>
              {postDetail.properties.name},{" "}
              {postDetail.properties.location}
            </h2>
            <br />
            <img
              src={postDetail.image}
              alt={postDetail.properties.location}
              width="250"
              height="225"
            />
          </Popup>
        </Marker>,
      ])}
  </LayerGroup>
</LayersControl.Overlay>
```

Kuva 9. Karttamerkit ponnahdusikkunoilla koodi.

Komponenttiin data tulee aiemmin luodusta ”db.json” -tiedostosta ja kyseisessä tiedostossa sijaitseva data on luotu GeoJson-formaatilla. GeoJson on formaatti erilaisten maantieteellisten tietorakenteiden koodaamiseen. (GeoJSON 2021). Alun perin GeoJson perustuu Json-muotoon ja sitä käytetään hyvin laajasti

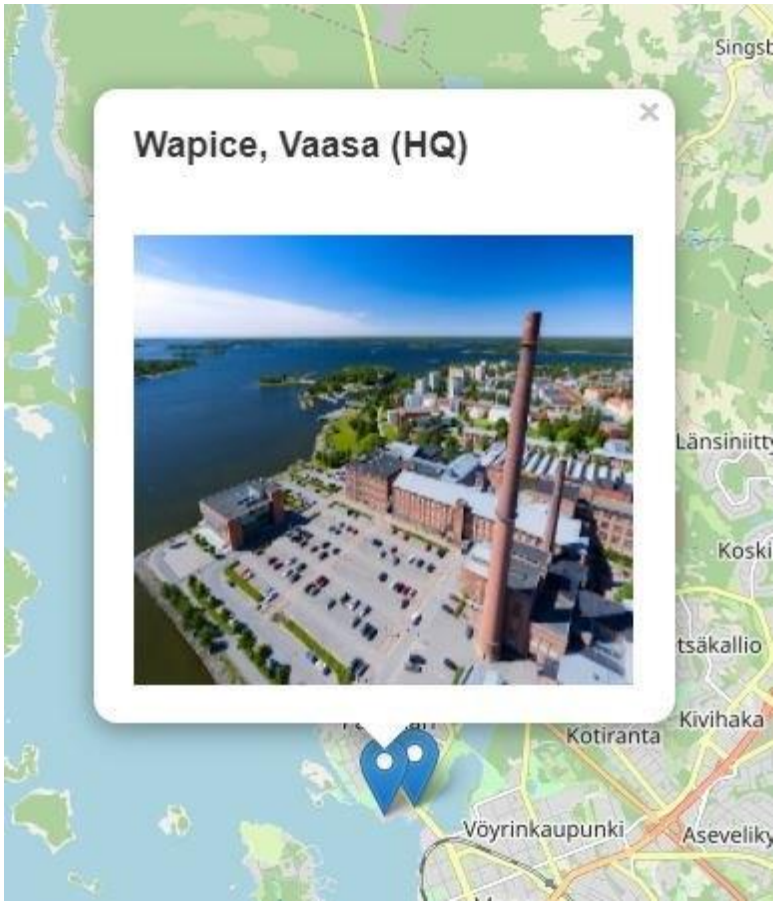
nykypäivän paikkatietojärjestelmissä. Kuvassa 10 esitetään komponenttiin tulevaa dataa GeoJson-muodossa.



```
{
  "features": [
    {
      "id": 1,
      "geometry": {
        "type": "Point",
        "coordinates": {
          "latitude": 63.10287846792585,
          "longitude": 21.59351146977082
        }
      },
      "properties": { "name": "Wapice", "location": "Vaasa (HQ)" },
      "image": "
```


Kuva 10. GeoJson -data.

Loppukäyttäjälle komponentti näkyy karttamerkkeinä, joita klikkaamalla kyseisestä sijainnista esitetään lisätietoa. Kuvassa 11 näkyy lopputuloksena karttamerkki sekä ponnahdusikkuna ja sen yksilölliset tiedot.



Kuva 11. Karttamerkki ponnahdusikkunalla.

4.3 Sijainnin jäljitys ja esittäminen karttamerkin avulla

Useissa karttaratkaisujen käyttötapauksissa on tärkeää saada dataa käyttäjän sijainnista. Sijainnin selvittämiseen ei välttämättä tarvita erillistä karttaratkaisua, sillä selvittämiseen voidaan käyttää useimmissa selaimissa tarjolla olevaa geolocation-rajapintaa.

Kuvassa 12 esitetään sijainnin jäljitykseen toteutettu funktio. Funktiossa kutsutaan ”getCurrentPosition()”-menetelmää, joka käynnistää asynkronisen pyynnön käyttäjän sijainnin tunnistamiseksi ja lähettää kyselyitä paikannuslaitteistolle ajantasaisten tietojen saamiseksi. (MDN Web Docs. 2022).

On tärkeää ottaa huomioon, että käyttäjä pystyy myös estämään sijainnin jakamisen ja kaikki selaimet eivät välttämättä salli sijainninjakoa. Tämän takia

funktiossa on määritelty erikseen onError()-metodi joka esittää virheilmoituksen, jos sijaintitietoa ei ole saatavilla.

```
import { useEffect, useState } from "react";

const UseGeoLocation = () => {
  const [location, setLocation]: any = useState({
    loaded: false,
    coordinates: { lat: "", lng: "" },
  });

  const onSuccess = (location: any) => {
    setLocation({
      loaded: true,
      coordinates: {
        lat: location.coords.latitude,
        lng: location.coords.longitude,
      },
    });
  };

  const onError = (error: any) => {
    setLocation({
      loaded: true,
      error,
    });
  };

  useEffect(() => {
    if (!("geolocation" in navigator)) {
      onError({
        code: 0,
        message: "Geolocation not supported",
      });
    }

    navigator.geolocation.getCurrentPosition(onSuccess, onError);
  }, []);

  return location;
};

export default UseGeoLocation;
```

Kuva 12. Sijainnin jäljitys.

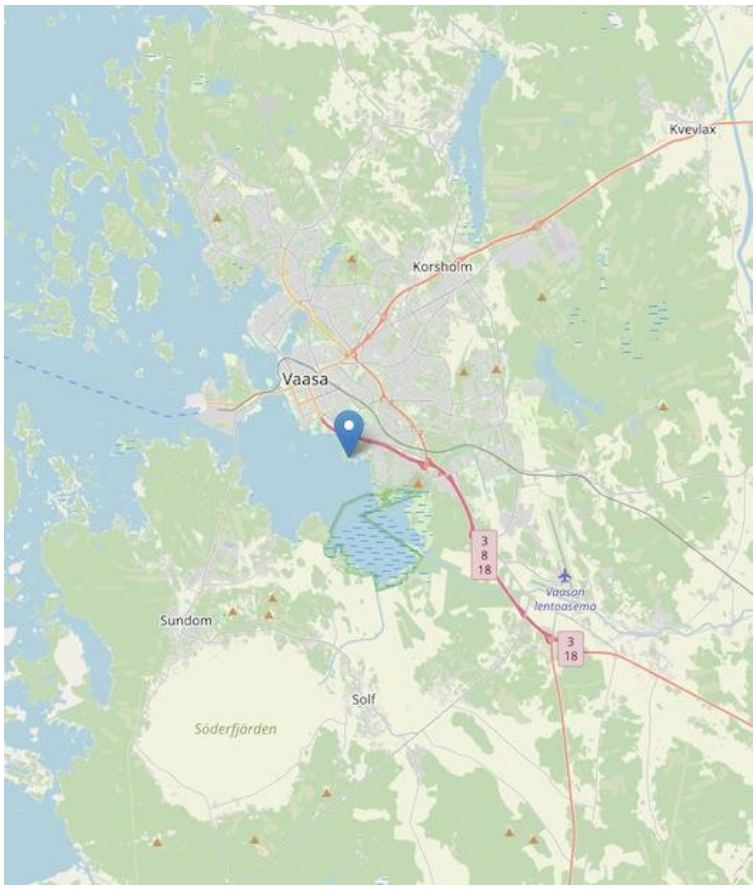
Kuvassa 13 esitetään kuinka UseGeoLocation -funktion avulla voidaan luoda karttamerkki havainnollistamaan käyttäjän sijaintia. Tässä tapauksessa käyttäjälle tarjotaan mahdollisuus käyttöliittymässä painaa valintaruutua, jonka jälkeen sovellus paikantaa käyttäjän sijainnin. Koodissa varmistetaan, että valintaruutua on painettu ja sijaintitieto on saatu. Karttamerkki luodaan React-Leaflet -lisäosan avulla, antaen komponentille kaksi arvoa: leveys-, ja pituusasteen.

```
{checkedLocateMe && location.loaded && !location.error && (
  <Marker
    position={[location.coordinates.lat, location.coordinates.lng]}
  ></Marker>
)}
```

Kuva 13. Sijainnin esittäminen karttamerkillä.

Kuvassa 14 esitetään kartassa näkyvä lopputulos. Leaflet-karttakirjasto tarjoaa mahdollisuuksia myös karttamerkkien kustomointiin, esimerkiksi värin ja leveyden suhteen. Karttamerkkejä on mahdollisuus tehdä myös kokonaan itse ja liittää ne esimerkiksi jpg- tai csv-muodoissa projektiin.

Kuvassa käytettyä karttamerkkiä käytetään oletusarvoisesti Leaflet-karttakirjaston yhteydessä.



Kuva 14. Karttamerkki esitettynä kartassa.

4.4 Monikulmioalueiden ja erilaisten visualisointien toteutus

Karttaratkaisuissa on tärkeää erilaisten alueiden havainnollistaminen kartalla. Alueita voidaan rajata esimerkiksi erilaisten muotojen avulla. Muoto on kartalla oleva objekti, joka on sidottu leveys- ja pituuskoordinaattiin. Leaflet-karttakirjaston avulla pystytään luomaan karttapohjaan esimerkiksi erilaisia viivoja, monikulmioalueita, ympyröitä sekä suorakulmioita.

Kuvassa 15 esitetään funktio, jonka avulla kartalle saadaan näkymään murtoviivoja, monikulmioita sekä ympyröitä. Ominaisuuksien lisääminen kartalle on tehty kehittäjälle hyvin helpoksi React-Leaflet -lisäosan ansiosta. Lisäosa tuo valmiit React-komponentit, joille käyttäjän tarvitsee antaa vain koordinaattitieto sekä mahdollisesti säde.

```
<LayersControl.Overlay name="Feature group">
  <FeatureGroup>
    <Popup>Popup in FeatureGroup</Popup>
    <Circle center={[59.913868, 10.752245]} radius={29000} />
    <Rectangle bounds={rectangle} />
    <Circle center={center} pathOptions={fillBlueOptions} radius={200} />
    <CircleMarker
      center={[51.51, -0.12]}
      pathOptions={redOptions}
      radius={20}
    >
      <Popup>Popup in CircleMarker</Popup>
    </CircleMarker>
    <Polyline pathOptions={greenOptions} positions={polyline} />
    <Polyline pathOptions={greenOptions} positions={multiPolyline} />
    <Polygon pathOptions={greenOptions} positions={polygon} />
    <Polygon pathOptions={purpleOptions} positions={multiPolygon} />
    <Rectangle bounds={rectangle} pathOptions={blackOptions} />
  </FeatureGroup>
</LayersControl.Overlay>
```

Kuva 15. Visualisointien luominen koodissa.

Visualisointien ominaisuuksia oli helppo muokata. Kuvassa 15 esitetään kuinka esimerkiksi ”Polygon”-komponentille voidaan antaa helposti kaksi arvoa, ”pathOptions”, jonka avulla voidaan muuttaa esimerkiksi väriä ja ”positions”, jonka avulla voidaan määritellä visualisoinnin sijainti kartalla.

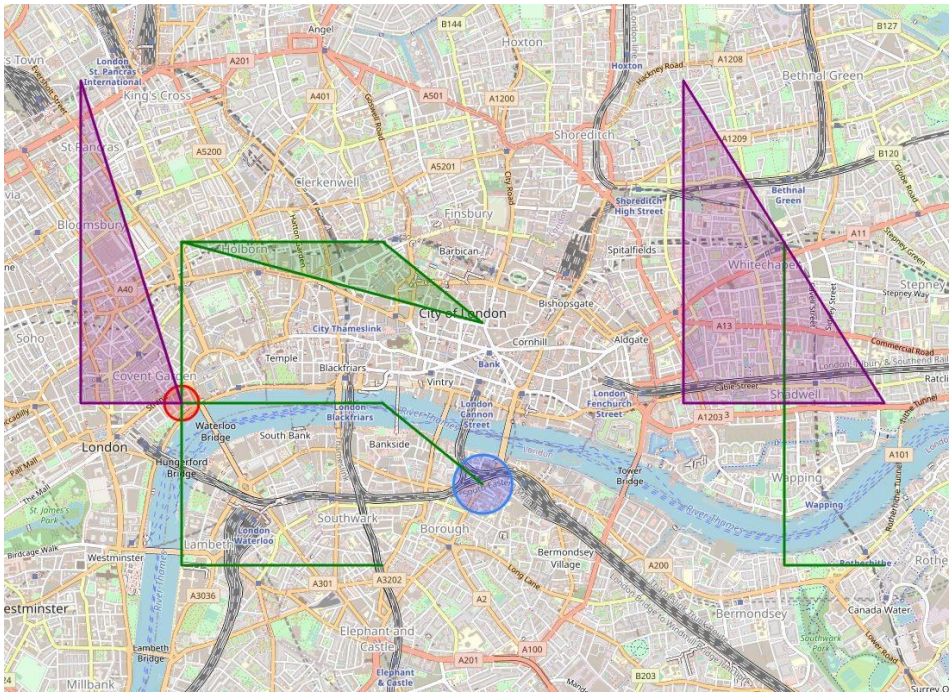
Kuvassa 16 esitetään, kuinka helposti yksinkertaiseen muuttujaan voidaan tallentaa visualisoinnin väri, jota kehittäjä pystyy käyttämään yhä uudestaan koodissaan määrittelemättä sitä joka kerta erikseen. Kuvassa 15 myös sijaintitiedot on tallennettu muuttujiin.

```
// Add styling
const fillBlueOptions: any = { fillColor: "blue" };
const blackOptions: any = { color: "black" };
const greenOptions: any = { color: "green" };
const purpleOptions: any = { color: "purple" };
const redOptions: any = { color: "red" };
```

Kuva 16. Visualisointien värien määrittely.

Kuvassa 17 esitetään lopputulos karttapohjassa. Ominaisuuksien tekeminen oli ohjelmistokehittäjän näkökulmasta hyvin yksinkertaista ja vaati vain vähän koodirivejä. On tärkeää huomata, että lopputulokseen vaikutti merkittävästi React-Leaflet -lisäosa, joka tarjosi melkein valmiita komponentteja käytettäväksi visualisointien esittämiseen.

Kehitettyjä ominaisuuksia voitaisiin myös laajentaa helposti. Yksi vaihtoehto olisi esimerkiksi esittää dataa alueesta ponnahdusikkunoilla siinä vaiheessa, kun käyttäjä klikkaa jotain toteutetuista visualisoinneista.



Kuva 17. Karttapohja erilaisilla visualisoinneilla.

4.5 Lämpökartta

Nykypäivän karttaratkaisuissa on ensisijaisen tärkeää ominaisuudet, joiden avulla pystytään havainnollistamaan erilaista dataa kartalla. Lämpökartat ovat paljon käytetty ominaisuus datan visualisointiin. Lämpökarttojen avulla voidaan esittää kartalla kohteiden käyttömäärää tai kuvastamaan esimerkiksi ihmisjoukkoja tietyllä alueella.

Leaflet tai React-Leaflet eivät tarjoa suoraan mahdollisuutta lämpökarttojen esittämiseen. Leafletillä on kuitenkin lämpökarttojen tekemisiin useita kolmannen osapuolen lisäosia, joiden ominaisuudet eroavat tapauskohtaisesti toisistaan jonkin verran. Seuraavassa komponentissa hyödynnetään Vladimir Agafonkin ylläpitämää ”Leaflet.heat” -laajennusta.

Lisäosa mahdollistaa data syöttämisen esimerkiksi array- eli taulukkomuodossa. Toinen mahdollisuus on syöttää dataa objekteina. Kuvassa 18 esitetään kuinka

lisäosaan syötetään useita koordinaattitietoja, jotka esitetään lopuksi kartalla pisteinä, joista muodostuu lämpökartta.

```
export const addressPoints = [
  [-37.8839, 175.3745188667, "571"],
  [-37.8869090667, 175.3657417333, "486"],
  [-37.8894207167, 175.4015351167, "807"],
  [-37.8927369333, 175.4087452333, "899"],
  [-37.90585105, 175.4453463833, "1273"],
  [-37.9064188833, 175.4441556833, "1258"],
  [-37.90584715, 175.4463564333, "1279"],
  [-37.9033391333, 175.4244005667, "1078"],
  [-37.9061991333, 175.4492620333, "1309"],
  [-37.9058955167, 175.4445613167, "1261"],
  [-37.88888045, 175.39146475, "734"],
  [-37.8950811333, 175.41079175, "928"],
  [-37.88909235, 175.3922956333, "740"],
```

Kuva 18. Lämpökartta data.

Lämpökartan muodostaminen koodissa on melko helppoa, eikä se vaadi paljon koodirivejä. Kuvassa 19 esitetään lämpökarttakomponentin luominen. Komponentti on luotu uudelleen käytettäväksi ja kartalle voidaankin lisätä useita lämpökarttoja eri alueille kirjoittamatta samaa koodia yhä uudelleen.

Koodissa ”useEffect()”-ominaisuus renderöi aiemmin määritellyt koordinaattipisteet kartalle aina sivun latautuessa. Lämpökartta saadaan asetettua kartalle painamalla valintaruutua, jonka myötä koodiin on kirjoitettu myös tarkistus, joka tarkistaa onko lämpökartta jo esitettynä kartassa. Jos lämpökartta on jo esillä, ohjelma osaa itse poistaa vanhan kerroksen ja luoda uuden tilalle, samalla päivittämällä mahdollisesti muuttuvan datan. Vanhan kerroksen poistamiseen käytetään koodissa ”removeLayer”-ominaisuutta, joka poistaa minkä tahansa kartalla viitatun kerroksen, tässä tapauksessa lämpökartan.

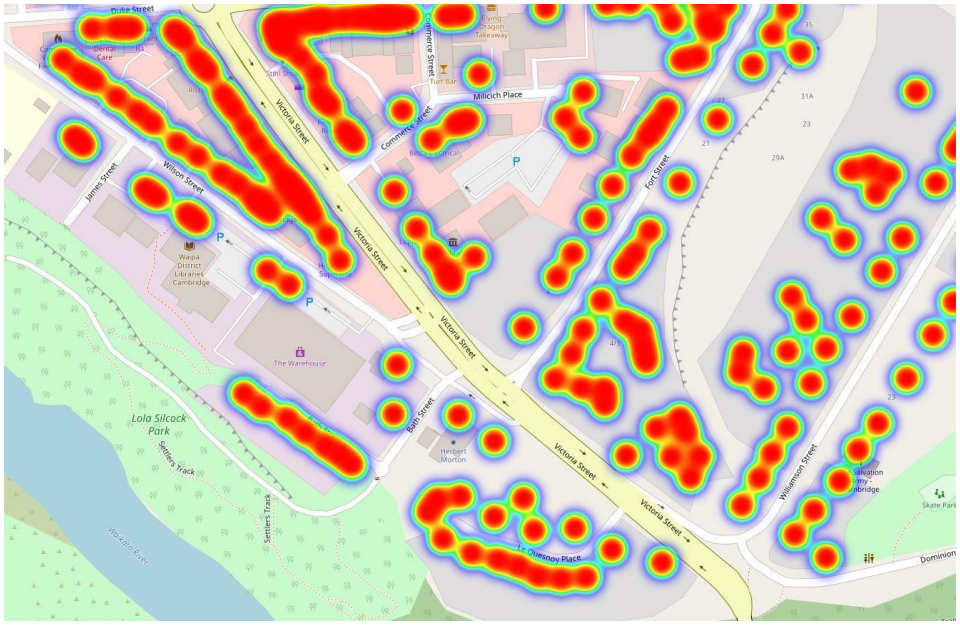
Komponentissa käytetään React-Leafletin tarjoamaa ”useMap()”-ominaisuutta, jonka avulla kehittäjä pystyy tekemään muutoksia karttaan erillisestä komponentista.

```
import { useMap } from "react-leaflet";
import L from "leaflet";
import "leaflet.heat";
import { useEffect, useState } from "react";
import { addressPoints } from "../../../data/addressPoints";

export interface HeatmapProps {
  layerActive: boolean;
}

function HeatMap(props: HeatmapProps) {
  const [layerOne, setLayerOne]: any = useState();
  const map = useMap();

  useEffect(() => {
    const points = addressPoints
      ? addressPoints.map((p) => {
          return [p[0], p[1]]; // lat lng intensity
        })
      : [];
    if (props.layerActive) {
      setLayerOne((L as any).heatLayer(points).addTo(map));
    } else {
      if (layerOne !== undefined) {
        map.removeLayer(layerOne);
      }
    }
  }, [props.layerActive]);
  return null;
}

export default HeatMap;
```

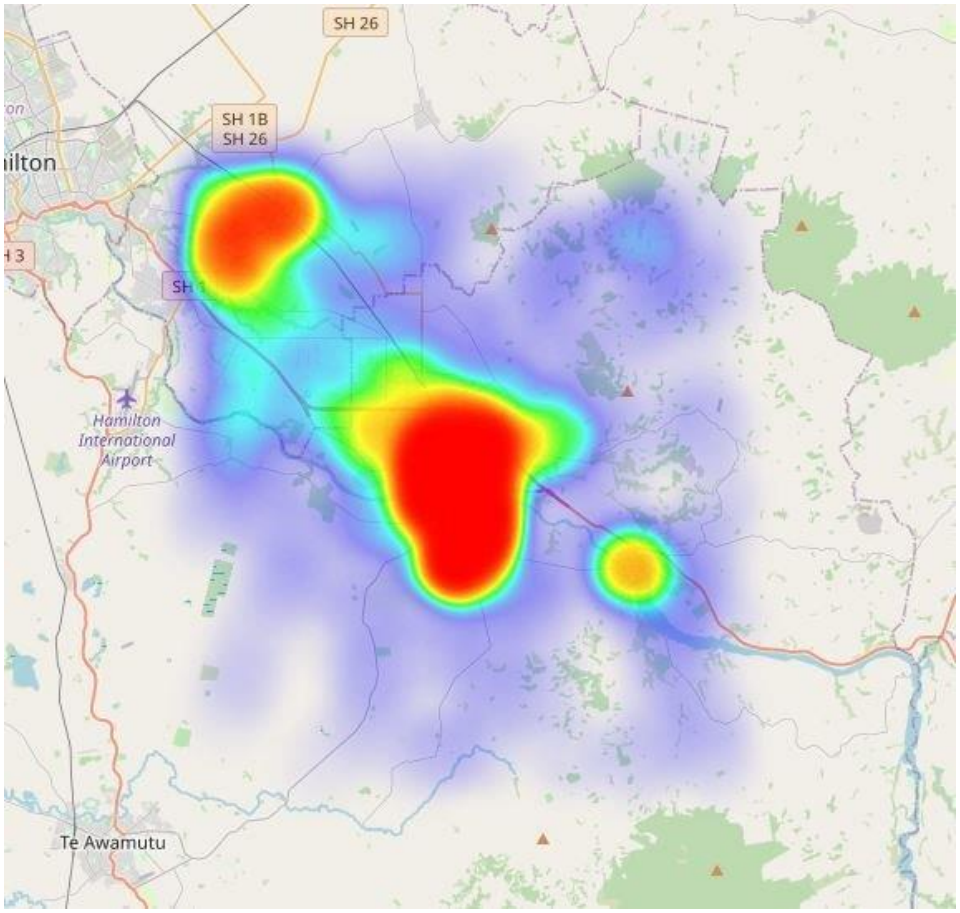
Kuva 19. Lämpökartan luominen.

Lämpökerroksen luomisen jälkeen kartalla näkyy kokoelma toisiaan lähellä olevia värikoodattuja pisteitä, jotka loppujen lopuksi koostuvat isommaksi

lämpökerroskokonaisuudeksi. Näkymä vaihtelee kartan zoom-tason mukaisesti. Kuvassa 20 on esitetty lämpökartta kerros suurella zoom-tasolla, kun taas kuvassa 21 pienellä tasolla.



Kuva 20. Lämpökarttapisteet.



Kuva 21. Lämpökarttakerros.

5 TULOKSET

Kappaleen tarkoituksena on koota yhteen opinnäytetyön aikana kerätyt tulokset ja havainnot, joiden perusteella on mahdollista valita useisiin erilaisiin käyttötapauksiin optimaalinen karttaratkaisu.

Tuloksena opinnäytetyössä syntyi kattava teoriaosuus erilaisista karttaratkaisuista, niiden piirteistä ja käytännönosuus, jonka avulla todistettiin yhden karttaratkaisun konseptin toimivuus.

5.1 Karttaratkaisut

Opinnäytetyön teoriaosuudessa löydettiin useita erilaisia karttaratkaisuja, joista jokainen esitti hyviä ominaisuuksia. Teoriaosuudessa esiteltiin yksittäiset karttaratkaisut, niiden ominaisuudet sekä suosion lähteet. Opinnäytetyössä onnistuttiin löytämään useita sopivia vaihtoehtoja tulevaisuuden käyttötarkoituksia varten.

Teoriaosuus tarjosi hyvän yleiskuvan Leaflet-, OpenLayers-, Mapbox- sekä Google Maps -ratkaisuista, jonka myötä opinnäytetyön lukijalle tarjottiin riittävä tietopaketti nykypäivän suosituimmista karttaratkaisuista. Karttaratkaisuja onnistuttiin löytämään ja esittelemään useita, minkä ansiosta valinnan tekeminen on helpompaa erilaisiin käyttötarkoituksiin sekä opinnäytetyön tilaajalle että lukijalle.

5.2 Käytännön toteutus

Opinnäytetyön käytännön osuudessa toteutettiin verkkopohjainen karttasovellus. Sovelluksessa hyödynnettiin Leaflet-karttakirjastoa, jonka avulla onnistuttiin toteuttamaan kaikki ennalta-asetetut toiminallisuusvaatimukset.

Toteutuksen aikana huomattiin, että Leaflet-karttakirjastolla on tarjolla paljon hyviä ominaisuuksia. Kirjaston pieni koko ja hyvä kehittäjäyhteisön tuki olivat Leafletin ehdottomia vahvuuksia. Vaikka kirjasto ei suoraan tarjonnut suurta

määrää visualisointi- eikä toiminallisuusmahdollisuuksia, kolmannen osapuolen lisäosia oli tarjolla useita satoja kappaleita. On kuitenkin selvää, että kaikkia näitä lisäosia ei aktiivisesti ylläpidetä ja ongelmia saattaa ilmetä ajan myötä.

Käytännön toteutuksessa pystyttiin onnistuneesti todistamaan karttaratkaisun konseptin toimivuus ja testaamaan ratkaisua käytännössä. Toteutuksen jälkeen voidaan todeta, että Leaflet voisi olla sopiva vaihtoehto sekä IoT-TICKET -tuotteeseen, että tulevaisuudessa erilaisiin käyttötapauksiin.

5.3 Jatkokehitys

Opinnäytetyön aikana tekijä sai hyvän perustason ymmärryksen nykypäivän moderneista karttaratkaisuista. Työn aihe oli kuitenkin hyvin laaja ja mahdollisia jatkokehitys- ja tutkimuskohteita löytyy aihealueelta paljon.

Erityisen hyödyllistä olisi suorittaa samankaltaiset käytännön toteutukset Leafletin lisäksi myös muilla esitetyillä karttaratkaisuilla. Tämä tarjoaisi parempaa ymmärrystä karttaratkaisujen eroista erityisesti ohjelmistokehittäjän näkökulmasta. Lisätutkimus tarjoaisi myös parempaa ymmärrystä yksittäisten ratkaisujen ominaisuuksista, sopivuudesta tiettyihin käyttötapauksiin sekä mahdollisista eroista koodissa.

Lisäarvoa tutkimukselle voisi tuoda myös laajempi listaus jokaisen karttaratkaisun ominaisuuksista. Tämä auttaisi ratkaisun valitsemisessa erilaisissa käyttötapauksissa, jossa tarvitaan esimerkiksi tiettyä visualisointimahdollisuutta, jota ei välttämättä jokaisesta karttaratkaisusta löydy.

6 JOHTOPÄÄTÖKSET JA POHDINTA

Viimeisen kappaleen tarkoituksena on yleisesti pohtia opinnäytetyö prosessia ja sen tuomia hyötyjä sekä toimeksiantajalle, että tekijälle. Kappaleessa esitetään yhteenveto opinnäytetyölle asetetuista tavoitteista ja niiden saavuttamisesta.

6.1 Työn onnistuminen

Opinnäytetyö onnistui odotusten mukaisesti. Työn aikana tekijä löysi useita nykyaikaisia karttaratkaisuja, joita pystyi verrata keskenään esimerkiksi yksittäisten ratkaisujen ominaisuuksien sekä suosion perusteella. Teoriaosuudessa onnistuttiin tuomaan esiin karttaratkaisujen piirteitä sekä esittämään dataa useasta eri lähteestä.

Karttaratkaisuista valittiin teoriaosuuden pohjalta testattavaksi Leaflet, jonka avulla luotiin verkkopohjainen karttasovellus. Sovellukseen pystyttiin tekemään useita erilaisia ominaisuuksia karttaratkaisun avulla. Käytännön osuudessa saatiin selville, että Leaflet tarjoaa kattavan määrän ominaisuuksia ja voisi täten olla sopiva vaihtoehto useisiin käyttötapauksiin tulevaisuudessa.

Ilman aiempaa aihealueen kokemusta erityisesti sopivien karttaratkaisuvaihtoehtojen löytäminen oli haastavaa, sillä viime vuosien kehityksen myötä karttaratkaisuja on tarjolla suuri määrä. Oli haastavaa löytää sopivia artikkeleita ja aiempia tutkimuksia, koska materiaalia on saatavilla hyvin paljon.

Opinnäytetyö tarjosi kokonaisuudessaan kattavan tietopaketin erilaisten karttaratkaisujen piirteistä sekä ominaisuuksista. On kuitenkin selvää, että lisätutkimusta aihealueesta voitaisiin tehdä. Erityisesti muiden esiteltyjen karttaratkaisujen testaaminen käytännössä voisi olla hyödyllistä.

6.2 Työn hyödyt

Toimeksiantajalle opinnäytetyö tarjosi konkreettisen tutkimuksen erilaisista karttaratkaisuista tarjoamaan apua optimaalisen ratkaisun valitsemisessa tulevaisuuden käyttötapauksissa. Tutkimuksen ansiosta myös yleinen osaaminen ja tietotaito karttaratkaisuista kehittyy yrityksen sisällä, sillä opinnäytetyön tekijälle osaamista on kertynyt prosessin aikana ja asiasta kiinnostuneille työntekijöille tarjotaan mahdollisuus tutustua opinnäytetyöhön yrityksen sisäisen viestinnän sivustolla.

Tekijälle opinnäytetyön tekeminen tarjosi mahdollisuuden opetella karttaratkaisuja ja kehittää omaa osaamista verkkopohjaisissa sovelluksissa. Käytännön osuuden ansiosta taidot kehittyivät myös teknologioiden kuten TypeScript ja React parissa. Kokonaisuudessaan projekti opetti tekijälle myös projektinhallintaa, suunnittelua sekä yleisesti ohjelmistokehityshankkeen läpiviemistä.

LÄHTEET

Bac Ha Software. 2020. Comparing Mapbox, OpenLayers and Leaflet. Viitattu 10.7.2021. https://bachasoftware.com/custom-software-development/

Baldwin Draper, E. 2021. What is Javascript Used For? Viitattu 11.8.2021. https://www.bloomtech.com/article/what-is-javascript-used-for

Berga, M. & Figueiredo, R. 2021. TypeScript vs JavaScript: which one is better? Viitattu 10.9.2021. https://www.imaginarycloud.com/blog/typescript-vs-javascript/

Dorman, M. 2018. Introduction to Web Mapping. Viitattu 28.6.2021. https://geobgu.xyz/web-mapping-2018/leaflet.html

Frerichs, M. 2021. Feature Comparison of Online Mapping Libraries. Viitattu 10.8.2021. https://mappingwithd3.com/feature-comparison/

Gaba, I. 2022. What is GitHub and How to Use It? Viitattu 1.3.2022. https://www.simplilearn.com/tutorials/git-tutorial/what-is-github

GeoJSON. 2021. Viitattu 12.9.2021. http://www.geojson.org

GitHub. 2022. Leaflet. Viitattu 4.3.2022. https://github.com/Leaflet/Leaflet

GitHub. 2022. Mapbox. Viitattu 4.3.2022. https://github.com/mapbox/mapbox-gl-js

GitHub. 2022. OpenLayers. Viitattu 4.3.2022. https://github.com/openlayers/openlayers

GitHub Docs. 2021. Fork a Repo. Viitattu 6.7.2021. https://docs.github.com/en/get-started/quickstart/fork-a-repo

Google Maps. 2021. Viitattu 15.10.2021. https://mapsplatform.google.com/pricing/

IoT-TICKET. 2022. Viitattu 2.3.2022. https://www.iot-ticket.com

Javatpoint. 2021. Difference between JavaScript and TypeScript. Viitattu 1.9.2021. https://www.javatpoint.com/javascript-vs-typescript

Javatpoint. 2021. JSON Server. Viitattu 20.9.2021. https://www.javatpoint.com/json-server

Leaflet. 2021 . Viitattu 16.7.2021. https://leafletjs.com/

MDN Web Docs. 2022. Using the Geolocation API. Viitattu 1.3.2022. https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API/Using_the_Geolocation_API

Node.js. 2011. What is npm? Viitattu 10.7.2021. ttps://nodejs.org/en/knowledge/getting-started/npm/what-is-npm/

Npm Trends. 2022. Viitattu 1.3.2022. https://www.npmtrends.com/

Pang, A. 2021. TypeScript vs. JavaScript. Viitattu 1.9.2021. https://medium.com/geekculture/typescript-vs-javascript-e5af7ab5a331

Piotrovskyi, Y. 2021. Mapbox vs Google Maps: Which Web Mapping Service Is Better. Viitattu 6.7.2021. https://yojji.io/blog/mapbox-vs-google-maps

Raval, N. 2022. TypeScript vs JavaScript: The Difference You Should Know. Viitattu 14.3.2022. https://radixweb.com/blog/typescript-vs-javascript

React-Leaflet. 2021. Viitattu 16.7.2021. https://react-leaflet.js.org/

Shokeen, M. 2021. What Is JavaScript? Viitattu 10.8.2021. https://code.tutsplus.com/tutorials/what-is-javascript--cms-26177

Stack Overflow. 2021. 2021 Developer Survey. Viitattu 15.9.2021. https://insights.stackoverflow.com/survey/2021

Studytonight. 2021. Javascript features. Viitattu 15.8.2021. https://www.studytonight.com/javascript/javascript-features

Wapice. 2022. Viitattu 1.3.2022. https://www.wapice.com

Wikipedia. 2022. Stack Overflow. Viitattu 1.3.2022. https://en.wikipedia.org/wiki/Stack_Overflow