jamk

Useista lähteistä koostuvan kartta-aineiston resurssioptimoitu hyödyntäminen erilliskäytössä

Justus Hänninen

Opinnäytetyö, AMK
Joulukuu, 2024
Tieto- ja viestintätekniikka

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Hänninen, Justus

Useista lähteistä koostuvan kartta-aineiston resurssioptimoitu hyödyntäminen erilliskäytössä

Jyväskylä: Jyväskylän ammattikorkeakoulu. Joulukuu 2024, 27 sivua

Tieto- ja viestintätekniikan tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: Kyllä

Tiivistelmä

Tutkimus käsitteli karttapalvelimien ja käyttöliittymäteknologioiden soveltuvuutta itsenäisen paikkatietojärjestelmän toteuttamiseen. Tutkimuksen tavoitteena oli tunnistaa avoimen lähdekoodin ratkaisuja, jotka täyttävät toimeksiantajan asettamat tekniset ja toiminnalliset vaatimukset. Tutkimuksissa keskityttiin palvelimiin, jotka kykenevät toimimaan ilman internet-yhteyttä ja tukevat yleisiä karttastandardeja, kuten MBTiles ja GeoTIFF.

Tutkimus toteutettiin testaamalla valittuja teknologioita iteratiivisesti. Näihin kuului muun muassa mbtileserver ja Tileserver-GL, joita arvioitiin suorituskyvyn, yhteensopivuuden ja helppokäyttöisyyden perusteella. QGIS:ää käytettiin tukityökaluna rasterimateriaalin tuottamiseen vertailutestejä varten. Mittaustulokset ja vaatimusten täyttymistä koskeva analyysi esiteltiin säännöllisesti toimeksiantajalle.

Valittujen teknologioiden testaamisen ohella tutkimuksessa kiinnitettiin huomiota myös konttiteknologioiden, kuten Dockerin, rooliin itsenäisten ohjelmistojen kehittämisessä. Docker mahdollisti testausympäristöjen nopean pystytyksen ja toimi todisteena ratkaisujen toistettavuuteen.

Tutkimuksen perusteella valittiin jatkokehitystä varten Tileserver-GL, joka täytti vaatimukset laajimmin ja tarjosi lisäominaisuuksia, jotka osoittautuivat hyödylliseksi. Vaikka tutkimuksen aikataulu oli tiukka, tavoitteet saavutettiin ja toimeksiantaja oli tyytyväinen tuloksiin. Tutkimuksen tulokset, sekä kaikki kirjoitettu koodi ja tehdyt konfiguraatiot julkaistiin toimeksiantajalle sisäisesti jatkokehitystä varten.

Avainsanat (asiasanat)

Paikkatietojärjestelmä, ohjelmointi, ohjelmistoarkkitehtuuri, ohjelmistoelinkaari, ohjelmistotuotantoprojekti, itsenäinen ohjelmisto

Muut tiedot (salassa pidettävät liitteet)

-

jamk

Description

Hänninen, Justus

Resource-optimised use of multi-source map data for stand-alone use

Jyväskylä: JAMK University of Applied Sciences, December 2024, 27 pages

Degree Programme in Information and communication technology Engineering. Bachelor´s thesis.

Permission for open access publication: Yes

Language of publication: Finnish

Abstract

The study explored the suitability of map server and user interface technologies for implementing an autonomous geographic information system (GIS). The objective was to identify open-source solutions that meet the technical and functional requirements set by the client. The focus was on servers capable of operating without an internet connection and supporting standard formats such as MBTiles and GeoTIFF.

The research was conducted iteratively by testing selected technologies, including mbtileserver and Tileserver-GL. These were evaluated based on performance, compatibility, and usability. QGIS was used as a supplementary tool to generate raster material for comparative testing. Performance metrics and requirement compliance analyses were regularly presented to the client.

In addition to evaluating selected technologies, the study emphasized the role of container technologies, such as Docker, in developing independent offline software. Docker enabled rapid deployment of test environments and proved the reproducibility of the solution.

Based on the findings, Tileserver-GL was chosen for further development due to its broad compliance with requirements and additional features that proved useful. Despite a tight schedule, the goals were achieved, and the client expressed satisfaction with the results. Results of the research as well as all code and configurations were released internally to the client for use further in development.

Keywords/tags (subjects)

Geographical information system, programming, software architecture, software development cycle, software project, offline software

Miscellaneous (Confidential information)

-

Sisältö

Taulukot

1 Johdanto

Karttapalveluita on tarjolla monia, mutta harva niistä tarjoaa täysin itsenäistä toiminnallisuutta. Itsenäinen toiminallisuus tarkoittaa, että palvelun tai ohjelman toiminnallisuus ei tarvitse ulkoisia prosesseja tai palveluita toimiakseen. Usein ohjelmistot hakevat tietoa netistä jatkojalostukseen, mutta itsenäisessä ympäristössä tämä ei ole mahdollista. Tämä merkitsee myös sitä, että käyttäjän tietokone suorittaa toimintoja, jotka tyypillisessä ympäristössä tehtäisiin tehokkaalla palvelimella pilvipalvelussa. Kyseisillä rajoituksilla toteutetaan paikkatietojärjestelmä, ja tutkitaan erilaisia kokoonpanoja avoimen lähdekoodin kirjastoista ja ohjelmista.

Opinnäytetyön toimeksiantaja oli Combitech Oy, joka on pohjoismainen teknologia- ja konsultointiyritys. Yrityksellä on tarve tilaus tuotteelle, joka tulee käyttöön edellä kuvatun kaltaisessa ympäristössä. Tämän opinnäytetyön rooli yrityksen toimeksiannossa on tutkimustyö, joka korvaa yrityksen tiimin normaalisti tekemän esiselvityksen. Tutkimustyön tarkoituksena on kartoittaa olemassa olevia teknologioita, joiden pohjalta lopullinen ratkaisu rakennetaan.

Vaatimusmäärittelyt on tehty etukäteen Combitech:n asiakkaan kanssa ennen opinnäytetyön aloittamista, ja niihin perehdytään tarkemmin siihen sopivassa kohdassa. Valmiit vaatimukset auttoivat tutkimusta, sillä tutkimustyölle varattu aika oli suhteellisen rajallinen toimeksiannon kiireellisyyden vuoksi.

Combitech Oy toimii Suomessa puolustustoimialan ja yhteiskunnan turvallisuuteen liittyvien kehittyneiden teknisten ratkaisujen ja palveluiden parissa. Yrityksellä on Suomessa yli 120 työntekijää, jotka on jaettu eri paikkakunnille Helsinkiin, Tampereelle, Jyväskylään ja Säkylään. (Tietoa meistä n.d.)

1.1 Tutkimusasetelma

Tutkimus keskittyy Combitechin ehdotettuihin teknologioihin, mbtileserver ja Tileserver-GL. Näitä verrataan keskenään yksinkertaisessa käyttöliittymässä. Oheinen Taulukko 1 sisältää tutkimusta tukevan vaatimuslistan, jonka Combitech oli opinnäytetyötä varten tehnyt. Vaatimukset ovat merkattu prioriteeteillä, joista kriittiset vaatimukset on pakko toteutua tavalla tai toisin. Jos teknologia ei täytä kriittistä vaatimusta, eikä sitä voi paikata muulla tapaa, sen voi automaattisesti hylätä.

Taulukko 1. Projektin vaatimukset

Vaatimus	Prioriteetti	Kuvaus
Tekninen-01	Kriittinen	Palvelin tukee yleisiä aineisto standardeja kuten mbtiles ja GeoTIFF.
Tekninen-02	Kriittinen	Palvelimella olevaa aineistoa pystyy lisäämään ja poistamaan.
Tekninen-03	Tärkeä	Palvelimella olevaa aineistoa on helppo hakea käyttöliittymästä riippumatta.
Tekninen-04	Tärkeä	Aineistojen välillä on helppo vaihtaa. Voi toteutua käyttöliittymässä.
Tekninen-05	Kriittinen	Karttapalvelimen tiili rajapinta tukee yleisiä standardeja.
Tekninen-06	Tärkeä	Kevyt käyttää, ei vaadi tehokasta konetta.
Tekninen-07	Tärkeä	Helppo jakaa käyttäjille.
Tekninen-08	Kriittinen	Toimii itsenäisessä ympäristössä.
Tekninen-09	Tärkeä	Esitettävä aineisto pystytään yhdistämään useista eri lähtöaineistoista.
Tekninen-10	Hyvä olla	Karttapalvelin tukee usean samanaikaisen aineiston esittämistä eri karttakerroksilla.
Tekninen-11	Kriittinen	Palvelin ja käyttöliittymä tukevat rasteriaineistoa.
Tekninen-12	Tärkeä	Palvelin ja käyttöliittymä tukevat vektoriaineistoa.
Tekninen-13	Tärkeä	Esitettävä aineisto pystytään yhdistämään sekä rasteri- että vektorimuodossa olevista lähtöaineistoista.
Yleinen-01	Kriittinen	Käytettävien teknologioiden lisenssien täytyy sallia kaupallinen käyttö ja käyttö suljetussa lähdekoodissa.
Yleinen-02	Hyvä olla	Hyvä maineiset ohjelmistokirjastot. Epämääräinen tilasto, mutta sille on muutamia hyviä mittoja, kuten kehittäjien arvostelut.

Tutkimuksessa kartoitettiin annettujen teknologioiden sopivuutta Combitech:n asettamien vaatimusten mukaisesti. Aihe ei ole uusi, ja ratkaisuja eri osiin on monia. Esitetty käyttöympäristö on

myös hyvin rajallinen suhteessa tavalliseen ohjelmistokehityksessä tuotettavaan ohjelmaan. Keskeiset kysymykset tämän perusteella, jotka pidetään mielessä tutkimuksen aikana ovat:

1. Soveltuvatko annetut teknologiat käyttöön työn vaatimusten rajoissa.
2. Onko kolmannen osa puolen ohjelmia, jotka korjaavat mahdollisia puutteita

Nämä kysymykset ohjaavat päätöksiä ja ovat oheisena teknisien- ja toiminallisienvaatimuksien lisäksi tukemassa tutkimussuunnitelmaa, jonka noudattaminen takaa ratkaisun löytämisen

1.2 Eettinen pohja

Tutkimus on objektiivinen perustuen tutkijan rajalliseen päätösvaltaan, sekä siihen että tutkittavat kirjastot ovat vieraita, joten entistä mielipidettä ei ole aiheeseen. Kaupallista vaikutusta valinnalla ei ole, joten toimeksiantaja voi objektiivisesti myös tehdä lopullisen päätöksen tutkimuksen päätteeksi tulosten perusteella.

1.2.1 Tekoälyn hyödyntäminen tutkimuksessa

Opinnäytetyössä on käytetty ChatGPT4 tekoälyä tiedonhaun tukena. Kaikki lähteet, joita tekoäly tarjosi, tutkittiin ja kritisoitiin ennen käyttöönottoa. Tekoälyn tarjoamaa tekstiä ja tietoa ei hyödynnetty lainkaan tai jätettiin huomioimatta kokonaan, eikä tekoälylle annettu muuta tietoa kuin hakutermejä. Tekoälyn rooli tutkimuksessa on siis olla kehittynyt hakukone. ”Tyhmiä” hakukoneita ja tunnettuja tietopankkeja silti suosittiin, ja tekoälyyn turvauduttiin vasta parempien tuloksien puutteessa.

2 Tietoperusta

2.1 Avoin lähdekoodi modernissa ohjelmistokehityksessä

Innovaatiossa pyritään välttämään pyörän uudelleen keksimistä. Ohjelmistokehityksessä tämä näkyy niin, että suosimme avoimenlähdekoodin ohjelmistoja, joita käytämme osana omaa suurempaa kokonaisuutta. Avoimen lähdekoodin ohjelma tarkoittaa sellaista ohjelmaa, jonka koodi on julkisesti nähtävissä. Tällaisten ohjelmien käyttöoikeudet usein myös sallivat uudelleenkäytön, muokkaamisen, ja edelleen myynnin, jolloin uusi ohjelma on kuin valmistunut palapeli koostuen

näistä avoimen lähdekoodin ohjelmista. Oma koodi on liima, joka pitää palapelin kasassa. (Avoin lähdekoodi - mitä se on ja mitkä ovat sen edut ja haitat? 2023.)

Tällainen lähestymistapa ohjelmistokehitykseen mahdollistaa myös nopean prototyypityksen, sillä kehittäjät eivät rakenna kaikkea alusta lähtien, vaan valitsevat olemassa olevia komponentteja ja keskittyvät näiden integrointiin sekä tarvittavien lisäominaisuuksien kehittämiseen. Avoimen lähdekoodin käytöllä onkin merkittävä rooli ohjelmistokehityksen kustannustehokkuuden ja nopeuden edistämisessä.

Avoimen lähdekoodin ohjelmistojen suurimmat edut ovat niiden kustannustehokkuus, yhteisön tarjoama jatkuva kehitys ja skaalautuvuus. Globaalit kehittäjäyhteisöt osallistuvat aktiivisesti avoimen lähdekoodin ohjelmien kehittämiseen, mikä parantaa niiden turvallisuutta ja toimivuutta, sillä virheitä korjataan nopeasti. Lisäksi ohjelmistojen räätälöitävyys ja joustavuus tekevät niistä houkuttelevia erityisesti monimutkaisissa projekteissa. Haittapuolena on kuitenkin usein dokumentaation puutteellisuus ja vaihteleva laatu, mikä voi hankaloittaa kehitystyötä. Tuen puute kaupallisiin ohjelmistoihin verrattuna voi myös aiheuttaa ongelmia erityisesti kriittisissä järjestelmissä, joissa tarvitaan nopeaa reagointia ja tukea. (Mt.)

2.2 Ohjelmistokehityksen elinkaari

Ohjelmistokehityksen elinkaari kuvaa ne vaiheet, joiden kautta ohjelmisto kehitetään ja saatetaan käyttöön. Tutkimustyössä erityisesti korostuu elinkaaren alkuvaiheet, kuten vaatimusmäärittely ja suunnittelu.

Vaatimusmäärittely on yksi kriittisimmistä vaiheista ohjelmistokehityksessä, sillä se asettaa selkeät tavoitteet projektille. Tässä vaiheessa määritellään, mitä ohjelmiston tulee tehdä, jotta se vastaa käyttäjien ja asiakkaiden tarpeita. Työssä tämä tarkoittaa asiakkaan esittämien tarpeiden ja teknisten vaatimusten kartoittamista ja niiden pohjalta tehtyä analyysiä. Oikein suoritettu vaatimusmäärittely minimoi virheet myöhemmissä vaiheissa ja varmistaa, että kehitystyö vastaa alun perin asetettuja tavoitteita. (Software Development Life Cycle (SDLC) 2024.)

Suunnitteluvaiheessa määritellään ohjelmiston arkkitehtuuri ja valitaan tarvittavat teknologiat. Tässä työssä suunnitteluun kuuluu erityisesti eri teknologioiden arviointi ja niiden yhteensopivuuden varmistaminen. Tarkoitus on luoda kokonaiskuva siitä, miten eri komponentit – tässä tapauksessa karttapalvelimet ja käyttöliittymä – liittyvät toisiinsa ja miten ne vastaavat asiakkaan vaatimuksia. Huolellinen suunnittelu mahdollistaa sen, että valitut teknologiat integroituvat saumattomasti yhteen ja täyttävät suorituskykyvaatimukset. (Mt.)

2.3 Ohjelmistotuotantoprojektin alku

Ohjelmistotuotantoprojekti seuraa samoja yleispiirteitä kuin minkä tahansa muunkin alan projektit, mutta yksityiskohdat toki vaihtelevat. Asiakkaan vaatimuksia kuunnellaan, näiden pohjalta rakennetaan vaatimusmäärittelyt, jotka toimivat selkärankana kaikelle tulevalle suunnittelulle ja tekemiselle. Näiden selkeys ja laajuus ovat tärkeitä projektin sujuvalle etenemiselle. (Bell 2023.)

Kun vaatimukset ovat selvitetty ja niiden perusteella tehty linjaukset, voidaan kääntyä internetin puoleen ja tutkia valmiita ratkaisuja. Ohjelmistoalalla on hyvin tavallista löytää ratkaisu ongelmaan – tai edes pieneen osaan ongelmasta – julkisesti jaettua koodia. Pyörän uudelleen keksiminen on kuitenkin turhaa, ja nykypäivänä kirjoitetaan yhä vähemmän koodia pohjasta kuin ennen. Tämä trendi on myös ennustettu kasvavan, joka tietenkin helpottaa ohjelmistokehittäjien työtehtäviä. (Paul 2022.) Vasta kun olemme tunnistaneet olemassa olevat ratkaisut, ja selvittäneen mitä itse joutuu tekemään, voidaan alkaa suunnittelemaan itse tuotteen lopullista arkkitehtuuria.

2.4 Ohjelmisto arkkitehtuuri web-teknologioilla

Toimeksiantoon kuuluu tietyt rajoitteet käytettävien teknologioiden kannalta. Karttapalvelun sisältö pitäisi saada ulos muodossa, jota web-teknologiat pystyvät käyttämään. Web-teknologiat tarkoittavat pelkistetysti teknologioita, joilla rakennetaan esim. verkkosivuja. Web-teknologiat ovat siitä hyviä, että ne toimivat miltei millä tahansa alustalla. Nykyään käytössä oleva HTML standardi on ollut olemassa jo 80-luvun loppupuolelta, ja vaikka sitä edelleen laajennetaan, HTML:llä rakennettu verkkosivu hyvällä todennäköisyydellä toimii laitteilla vuosituhannen alusta. On olemassa olevia web-standardeja, joita yleisesti seurataan. Tämä johtaa siihen, että web teknologioilla rakennettu ohjelma toimii hyvällä todennäköisyydellä laitteella kuin laitteella, mikäli laitteella pystyy käyttämään internet selainta. (The web and web standards n.d.)

Paikkatietojärjestelmä ei kuitenkaan tavallisesti ole pelkästään sen käyttöliittymä, vaan myös tieto, jota käyttöliittymä näyttää ja hallitsee. Tarvitsemme vähintään paikan, johon tallentaa tietoa, sekä välikäden joka tuo tiedon tallennetusta paikasta käyttöliittymälle. Tätä voi kuvata nimellä N-tason arkkitehtuuri. Meillä on käyttöliittymä, jota tukee logiikka, joka hakee ja hallitsee tietoa, jota käyttäjän kuuluu nähdä ja määrää yhteydet tiedon muokkaamiselle ja käytölle käyttöliittymän ja tiedonlähteen välillä. (Alexandra 2024.)

2.5 Docker ja konttiteknologiat

Mikropalveluarkkitehtuurin rakentamisessa hyödynnettiin Docker ohjelmaa. Docker on yksi tärkeimpiä ja käytetyimpiä ohjelmia nykypäivänä (2023 Developer Survey 2023). Suuremmat ohjelmistotalotkin kehuvat Dockeria, kuten Atlassian (2024), jonka mukaan ”Docker on täydellinen mikropalveluarkkitehtuurin toimittamiseen”. Se on saanut suosiota kattavana ja helppokäyttöisenä työkaluna, jolla rakentaa ja käyttää kontteja.

Kontit ovat pieniä, suljettuja ja toistettavia ympäristöjä, jonka sisälle kukin mikropalvelu voidaan asettaa pyörimään. Koska kontit ovat täysin omassa ympäristössään ja toistettavia, kontti, joka toimii yhdessä suuremmassa ympäristössä – kuten jonkun käyttäjän tietokoneella – sama kontti toimii harvinaisia poikkeuksia lukuun ottamatta missä tahansa ympäristössä. Kontit kommunikoivat keskenään omassa suljetussa verkossa, ja palveluita voi avata ulkoverkkoon tai käyttäjälle saataville helposti muokattavilla asetuksilla. (Atlassian 2024.)

Docker mahdollistaa nopeamman ja tehokkaamman kehitysprosessin, sillä konttiteknologia varmistaa, että palveluiden eri versiot ja riippuvuudet pysyvät yhdenmukaisina kehittäjätiimin sisällä ja tuotantoympäristössä. Tämä vähentää konflikteja ympäristöjen välillä ja helpottaa virheenkorjausta sekä ohjelmiston skaalautuvuutta, koska palvelut voidaan toistaa useassa ympäristössä ilman suuria asennusprosessin muutoksia. (Mt.)

Dockerin skaalautuvuuden lisäksi se helpottaa resurssien hallintaa, koska kukin palvelu voi toimia erillään ja siten hyödyntää järjestelmäresursseja tehokkaasti. Näin kokonaisuus voi kasvaa tarpeen mukaan, ilman että ympäristön ylläpidosta tulee merkittävä lisähaaste. (Mt.)

2.6 Paikkatietojärjestelmät ja karttapalvelut

Paikkatietojärjestelmä on laaja termi teknologialle, joka vastaanottaa tietoa ja asettaa ne kartalle nähtäväksi. Tällaiset järjestelmät myös tyypillisesti mahdollistavat tiedon muokkaamista käyttäjäystävällä tavalla, kuten siirtämällä merkkiä kartalla, jolloin merkin uusi paikka tallentuu tietokantaan. Hyvin pelkistetysti, paikkatietojärjestelmä tarkoittaa karttaa, joka antaa käyttäjälle tietoa sekä työkaluja käyttää tietoa hänelle sopivalla tavalla. (Jonker 2023.)

Paikkatietojärjestelmät ovat hyvin tärkeitä ja yleisiä. Google Maps on todennäköisesti tuttu lukijalle, mutta erikoisempiinkin käyttöön on tehty järjestelmiä. Paikkatietojärjestelmiä on käytetty kartoittamaan esimerkiksi sairauksien paikallisuutta ja leviämistä. (Mt.)

Toimeksiantajan vaatimuksille ei ole hyvää valmista ratkaisua, josta syystä tutkimus on syntynyt. Valmiita osia kuitenkin löytyy vapaasti käytettävänä, joka mahdollistaa tutkimuksen valmistumisen tiukan aikarajan sisään.

Koska paikkatietojärjestelmät ovat vanhoja, niille on kerennyt kehittyä useampia standardeja, joita tullaan hyödyntämään tutkimuksessa. Standardien noudattaminen ja käyttäminen on tärkeää yritysmaailmassa, joten niitä noudatetaan tutkimuksen aikanakin.

2.6.1 WMS ja WMTS rajapinnat

Web Map Service (WMS) ja Web Map Tile Service (WMTS) ovat kansainvälisen Open Geospatial Consortium standardoimia rajapintoja, joita käytetään paikkatietoaineistojen jakelussa.

WMS tarjoaa karttoja dynaamisesti renderöityinä kuvatiedostoina, jotka muodostetaan käyttäjän antamien parametrien, kuten alueen ja mittakaavan, perusteella. Tämä tekee WMS:stä joustavan vaihtoehdon monimutkaisten ja jatkuvasti päivittyvien karttojen tarjoamiseen. WMS-palvelut tukevat myös tietojen hakemista kartalta, mikä mahdollistaa esimerkiksi klikattavien kohteiden tiedon näyttämisen käyttäjälle. WMS:n joustavuudesta huolimatta sen suorituskyky voi kärsiä, kun kartan renderöinti tapahtuu reaaliajassa palvelimella. (Allen 2024.)

WMTS taas on suunniteltu tarjoamaan karttoja etukäteen luoduista karttatiilistä, jotka on tallennettu palvelimelle eri mittakaavatasoilla. Tämä tekee siitä suorituskykyisemmän ratkaisun erityisesti silloin, kun käyttäjiä on paljon ja kartan sisältö ei muutu dynaamisesti. WMTS-rajapinta mahdollistaa laattapohjaisen kartta-aineiston nopean lataamisen, mikä parantaa käyttäjäkokemusta etenkin suurilla ja yksityiskohtaisilla kartoilla. (Mt.)

2.6.2 XYZ-skeema

XYZ-skeema on yleisesti käytetty karttalaattojen järjestämismenetelmä, joka määrittelee, miten karttatiilet tallennetaan ja haetaan tietojärjestelmästä. Tämä skeema pohjautuu yksinkertaiseen koordinaattijärjestelmään, josta jokainen tiili löytyy kolmen parametrin perusteella:

1. X - koordinaatti, joka määrittää tiilen sijainnin vaaka-akselilla.
2. Y - koordinaatti, joka määrittää tiilen sijainnin pystyakselilla.
3. Z - zoom-taso, joka kertoo kartan tarkkuuden ja tiilen yksityiskohtaisuuden.

Näiden parametrien avulla voidaan viitata tiettyyn karttalaattaan ja ladata se karttapalvelimelta. Esimerkiksi osoite https://example.com/tiles/10/532/385.png viittaa zoomitasolla 10 olevaan laattaan, jonka X-koordinaatti on 532 ja Y-koordinaatti 385.

Tämä on hyvin intuitiivinen tapa hakea kartta-aineiston dataa, mutta kuten kaikella teknologialla, sillä on rajoitteensa. XYZ-skeema vaatii tukea käyttöliittymältä, joka hakee tietoa palvelimelta. Tiilipohjaisuus myös voi olla joskus rajoite verrattuna esim. WMS rajapintaan, josta voi hakea rajoitetun alueen mistä tahansa.

2.6.3 TileJSON rajapinta

Toisin kuin edelliset rajapinnat, TileJSON ei tarjoa kartta kuvia tai tiiliä. Se vain kuvaa mitä palvelimella on, missä muodossa palvelimen aineisto on ja miten palvelimen aineistoon pääsee käsiksi. TileJSON on kevytrakenteinen JSON-formaatti, jota käytetään tiilipohjaisten karttojen metadatan tallentamiseen ja jakamiseen. Se määrittää karttatiilien sijainnin, mittakaavat, koordinaatistot ja muut tiedot, jotka ovat olennaisia karttapalveluiden toiminnalle. TileJSON-rajapinta on erityisen

hyödyllinen, kun halutaan standardoida ja yksinkertaistaa karttapalveluiden käyttöä, sillä JSON-formaatti on erityisen laajasti tuettu formaatti.

TileJSON:n etuna on sen helppokäyttöisyys ja yhteensopivuus modernien karttasovellusten kanssa. Tämä tekee siitä suosittavan vaihtoehdon erityisesti silloin, kun halutaan jakaa tiilidataa useiden eri alustojen välillä yhtenäisesti.

2.6.4 Rasteri- ja vektoritiilet

Kartta-aineistot jaetaan kahteen päätyyppiin: rasteritiiliin ja vektoritiiliin. Nämä teknologiat eroavat toisistaan niin datan esitystavan kuin käyttötarkoituksen osalta.

Rasteritiilet ovat esirenderöityjä kuvia, jotka muodostavat kartan eri mittakaavoissa. Ne ovat helppoja tuottaa ja käyttää, sillä ne eivät vaadi erityistä renderöintiä käyttäjän laitteella. Rasteritiilet soveltuvat erityisesti tilanteisiin, joissa kartan ulkoasu on tarkkaan määritelty, eikä sitä tarvitse muokata käyttäjän päässä. Tällaisia käyttötapauksia ovat esimerkiksi taustakartat ja historialliset kartat. Haittapuolena rasteritiilissä on niiden suuri tiedostokoko, joka voi hidastaa latausaikoja, sekä niiden tarkkuus. Koska rasteri tiilet ovat kuvia, niillä on pikselimääräinen raja, jolloin mitä tarkemmin yrittää katsoa karttaa, sitä epätarkemmaksi se muuttuu. (Martin 2024.)

Vektoritiilet puolestaan koostuvat geometriasta ja attribuuttitiedosta, jotka käyttäjän laite renderöi kartaksi. Tämä mahdollistaa karttojen dynaamisen muokkaamisen, kuten tyylien ja näkyvyyksien muuttamisen reaaliaikaisesti. Vektoritiilillä on huomattavia etuja, kuten pienempi tiedostokoko ja parempi suorituskyky suurilla mittakaavoilla. Ne soveltuvat hyvin interaktiivisiin sovelluksiin, joissa käyttäjä voi esimerkiksi tarkastella eri tietokerroksia tai muokata kartan ulkoasua. Vektoritiilillä on myös se valtava etu, että ne pysyvät tarkkana, vaikka niihin zoomaisi äärettömästi. Koska vektoritiilet renderöidään datan perusteella, kun käyttäjä zoomaa lähemmäs, voidaan ”laskea” uusi tarkka kuva. (Mt.)

Vektoritiilet ovat hyviä tarkkuutensa ja resurssitehokkuutensa vuoksi, mutta rasteritiilissä on se etu, että mitään ei tarvitse laskea. Voidaan vaan suoraan asettaa pikseleitä oikeaan paikkaan, joka tekee niistä nopeampia. Tavanomaisesti loppukäyttäjä ei näkisi tätä vektorien laskemista, mutta

itsenäisessä ympäristö tämä vektorien laskeminen tapahtuu käyttäjän koneella, eikä palvelinsalissa.

2.6.5 MBTiles tiedostoformaatti

MBTiles on tiedostoformaatti, joka mahdollistaa karttalaattojen tallentamisen yhteen, kompaktisti pakattuun tietokantatiedostoon. Se on suunniteltu erityisesti laattapohjaisten karttojen tallentamiseen ja jakamiseen, ja se tukee sekä rasteri- että vektoritiiliä. MBTiles on suosittu ratkaisu tilanteissa, joissa tarvitaan tehokasta ja tapaa käsitellä suuria kartta-aineistoja itsenäisessä ympäristössä.

MBTiles-tiedosto koostuu tietokantatauluista, jotka sisältävät metadataa, tiilikohtaisia tietoja ja tarvittaessa tyylejä. Tämä keskitetty tallennustapa tekee tiedostosta helposti siirrettävän ja käytettävän eri järjestelmissä. MBTiles-tiedostoja voidaan käyttää karttasovelluksissa suoraan, sekä palvella karttoja karttapalvelimen kautta.

MBTiles on erinomainen valinta itsenäisissä karttasovelluksissa, joissa tarvitaan paikallista datan hallintaa. Lisäksi se soveltuu tilanteisiin, joissa halutaan yksinkertaistaa tiilidatan jakamista esimerkiksi pilvipalveluista tai kehitysympäristöistä käyttäjille. Vaikka MBTiles on tehokas ja monikäyttöinen formaatti, sen käyttö vaatii yhteensopivia työkaluja ja kirjastoja, jotka pystyvät lukemaan tiedoston tietokantaa ja renderöimään dataa karttasovelluksissa. Tämä voi asettaa rajoituksia esimerkiksi täysin mukautettujen ratkaisujen kehittämisessä.

Suuremmat ohjelmat eivät yleensä ole monoliittisia, tarkoittaen että olisi vain yksi pala, joka tekee kaiken. On tavanomaisempaa, että ohjelma koostu useasta pienestä osasta, jotka yhdistyvät hienoksi kokonaisuudeksi. Kun katsomme verkkosivua, tai toisin sanoen käyttöliittymää, on melko harvinaista, että tällä sivulla ei olisi jotain, joka ei olisi tullut tietokannasta. Kyseiselle tietokannalle on myös hyvin todennäköisesti jokin mikropalvelu olemassa, joka vastaa siitä, että käyttäjä saa odotettua sisältöä. Tämän mikropalvelun ohessa voi olla muitakin mikropalveluita, jotka vastaavat esimerkiksi pääsyhallinnasta, eli siitä että käyttäjä ei näe mitä hänen ei kuulu nähdä, tai palveluita, jotka hakevat mainoksia perustuen käyttäjän tietoihin. (Yaskiv & Dolynyuk 2021.)

Tällaisessa arkkitehtuurissa on se etu, että voimme rakentaa uudelleen käytettäviä osia, jotka on helppo korvata toisella. Kuvattu arkkitehtuuri sopii myös täydellisesti tutkimustyöhön, joka vertaa eri ohjelmistokokoonpanoja. Voimme helposti vaihtaa karttapohjan tarjoavan palvelun toiseen, tai käyttöliittymän, joka käyttää kyseistä karttapohjaa. Tästä syystä tutkimustyön sujuvaan etenemisen varmistamiseksi, päädyttiin mikropalvelu arkkitehtuuriin.

2.7 Käyttäjäkokemus ja sen tärkeys

Ohjelma kuin ohjelma tarvitsee hyvän käyttäjäkokemuksen, jotta sitä ei ole turhauttava käyttää. Käyttäjäkokemus voi suoraan vaikuttaa konversioihin. Jo se, että verkkosivu latautuu hitaasti, voi käännyttää mahdollisen asiakkaan pois sivuilta (6 Website Load Time Statistics and Why They Matter 2024; How website performance affects conversion rates n.d; Green 2016). Harmain Arif (2023) kuvaa ilmiötä helposti ymmärrettävästi: ”Kun ihmiset pitävät tuotteesta, he käyttävät sitä paljon ja suosittelevat sitä ystävilleen”.

Vaikka käyttäjät ei tule suoraan käyttämään tutkittavia palveluita, palvelut vastaavat käyttäjälle näkyvästä kartta-aineistosta. Siksi niiden suorituskykyä ja nopeutta kritisoidaan. Tutkimustyötä tukevassa käyttöliittymässä kuitenkin kiinnitettiin huomiota käyttäjäkokemukseen, sillä siitä on hyötyä tutkimuksen demonstroinnissa, sekä tulevassa tuotekehityksessä.

3 Projektisuunnitelma

3.1 Työntavoite

Suunnittelu vaiheessa käytiin lävitse tutkittavat teknologiat, sekä Combitech:n vaatimukset projektille (Taulukko 1. Projektin vaatimukset). Vaatimukset toimivat listana, josta voi raksia toteutuneet toiminnallisuudet. Listan perusteella voi sitten helposti verrata teknologioita ja tehdä lopullinen päätös, mitä viedään jatkokehitykseen. Tutkimustyön aikana saattaa tulla haasteita ja ideoita eteen, jotka menevät suunnitelman ulkopuolelle. Aika budjettiin pitää varata tähän aikaa, joka on kuitenkin jo valmiiksi tiukka. Tehty suunnitelma kartoittaa tutkittavat teknologiat, testaus menetelmät, sekä miten todennetaan ja hyväksytään tulokset Combitech:n kanssa.

Tavoite on suorittaa tutkimustyötä, jossa keskitytään teknologioiden kartoittamiseen. Käytännön toteutukset jäävät vähemmälle, mutta mahdollisten ratkaisujen arvioimiseksi on kuitenkin tarpeen rakentaa pieniä demoversioita eri kokoonpanoista. Työn keskeinen tarkoitus on etsiä kaupalliseen käyttöön soveltuvia ohjelmistoja ja kartoittaa niiden ominaisuuksia. Näin voidaan tunnistaa eri ratkaisukokonaisuuksien vahvuudet ja heikkoudet.

Tuloksia arvioidaan säännöllisesti yhteistyössä Combitech:n kanssa, hyödyntäen itse rakennettua minimaalista versiota lopullisesta tuotteesta. Tutkimuksen päättyessä valitaan parhaiksi havaitut teknologiat jatkokehitystä varten.

Lopullinen tuote koostuu kolmesta osasta. Ensimmäinen osa on työkalu, joka kokoaa ja paketoi taustakartan ja sen eri tasot. Tämä osa ei kuulu tutkimuksen piiriin, sillä Combitech:n asiakkaalla on jo olemassa oleva ratkaisu tähän tarpeeseen. Tällaista työkalua kuitenkin hyödynnettiin tutkimuksessa, sillä se auttoi testaamaan toteutusta.

Toinen osa koskee eri kartta-aineistojen pakkaamista siten, että niiden välillä on helppo vaihtaa. Tämä sisältää myös loppukäyttäjien mahdollisuuden tuoda aineistoa ohjelmaan ulkoiselta tallennuslaitteelta. Kolmas ja viimeinen osa koostuu paikallisesta karttapalvelimesta ja siihen liitettävästä käyttöliittymästä. Tämä paikallinen palvelin korvaa palvelut, kuten OpenStreetMap, joista tyypillisesti haetaan karttanäkymän muodostamiseen tarvittavat tiilet. Paikallinen karttapalvelin hakee aineiston edellä mainituista paketeista ja tarjoaa sen käyttöliittymälle. Karttapalvelimen ja käyttöliittymän on myös pystyttävä tukemaan pieniä muutoksia aineistoihin, esimerkiksi tasojen lisäämistä tai poistamista.

3.2 Haasteet

Vaatimusmäärittelyssä on tunnistettu useita haasteita. Ympäristössä, jossa loppukäyttäjät toimivat, ei ole mahdollisuutta internet-yhteyteen, ja heidän laitteensa saattavat olla tehoiltaan rajoitettuja tai vanhentuneita. Nämä rajoitteet asettavat erityisiä vaatimuksia toteutukselle, ja on tärkeää varmistaa, että tekniset vaatimukset pysyvät hallinnassa. Erityisen haastaviksi on tunnistettu kartta-aineiston tuonti kätevästi loppukäyttäjille, sekä itse tuotteen paketointi ja jakelu käyttäjien

laitteille. Lisäksi käyttäjien – jotka eivät välttämättä ole alkuperäisen aineiston kokoajia – on pystyttävä päivittämään aineistoa helposti ja intuitiivisesti, mikä vaati erityistä huomiota käyttäjäkokemuksen suunnittelussa.

Näiden haasteiden lisäksi on ratkaistava asiakkaan antamat tekniset ja toiminnalliset vaatimukset (Taulukko 1. Projektin vaatimukset), jotta lopullinen tuote vastaa odotuksia. Tutkimuksen tarkoitus ei kuitenkaan ole näitä kaikkia ratkaista, mutta selvittää ovatko mahdollisia ratkaista, kuinka helposti, ja mitä käyttäen.

3.3 Tutkimuksen tueksi valittuja teknologioita

Tutkimus sisältää paljon viittauksia tiettyihin ohjelmointikieliin, kieliä käyttäviin kirjastoihin, sekä valmiisiin ohjelmiin. Tutkimuskohteita avataan sopivassa kohdassa, mutta lukemisen avuksi on hyvä tietää pinnallisesti käytetyistä oheisista ohjelmointikielistä ja kirjastoista. Nämä teknologiat tukivat testaamista tuottamalla aineistoa, tai näyttämällä sitä.

3.3.1 JavaScript

JavaScript on ohjelmointikieli, joka on todennäköisesti palvellut eniten käyttäjiä maailmassa. 98.9 % verkkosivuista käyttää JavaScriptiä, joka käytännössä tarkoittaa sitä, että jos käytät internettiä, olet kohdannut Javascriptillä rakennettua sisältöä (Usage statistics of JavaScript as client-side programming language on websites n.d). JavaScript on myös levinnyt verkkosivujen ulkopuolelle suuren suosionsa ansiosta. Tästä syystä se on luonnollinen tai ehkä jopa ainut ratkaisu web-teknologia pohjaisen käyttöliittymän rakentamiseen. JavaScript antaa työkalut, joilla yhdistää karttapalvelun ulos työntämän datan käyttöliittymään, sekä logiikan sen manipulointiin.

3.3.2 React

Nykypäivänä verkkosivuja harvemmin kirjoitetaan puhtaalla JavaScriptillä. Yleisemmin käytetään JavaScript kehyksiä, jotka ratkaisevat yleisiä ongelmia ja nopeuttavat kehitystyötä. Yksi suosituimmista kehyksistä on Facebookin kehittämä React, joka tarjoaa vahvoja valmiita toimintoja käyttöliittymän rakentamiseen komponenttipohjaisesti. Kuin rakentaisi Lego paloilla, mutta itse rakentaa myös palat.

React on tällä hetkellä suosituin kehys käyttöliittymien rakentamiseen, joka myös tarkoittaa sitä, että integraatiot eri kirjastoihin on todennäköisesti joku muu jo kerennyt tehdä (2023 Developer Survey 2023). Tämä valinta on jatketta ympäristön kaikkien osien yksinkertaistamiseen, joka takaa, että eri karttapalveluiden testaaminen on mahdollisen helppoa. Sama rakennettu React käyttöliittymä oli käytössä koko tutkimuksen ajan.

3.3.3 Leaflet

Leaflet on avoimen lähdekoodin kirjasto, jota käytetään interaktiivisten karttanäkymien rakentamiseen. Se on erityisen suosittu, koska se on kevyt ja joustava, mutta tarjoaa kuitenkin yleisimmät karttaohjelman toiminnallisuudet. Leaflet on myös todettu hyväksi ikänsä takia. Teknologiassa yleensä vanha on huonoa, mutta se voi myös tarkoittaa vakautta ja luotettavuutta. Leaflet projekti aloitettiin vuonna 2011, ja silti saa pieniä päivityksiä vuonna 2024, joka antaa hyvän kuvan kirjaston pitkäjänteisyydestä. (Leaflet 2024).

Leaflet yksinkertaistaa karttaohjelman kehitystä, sillä vaikeammat ongelmat kuten mitenkä sijoittaa ja näyttää vastaanotettu paikkatieto, elementtien sijoittaminen kartalle ja eri karttatasojen hallinta ovat jo ratkaistu. React ja Leaflet myös toimivat hyvin yhdessä, sillä tälle yhdistelmälle on kätevä React-Leaflet kirjasto, joka antaa käyttää Leaflettia Reactin komponentti tyylillä.

3.3.4 QGIS

Quantum Geographic Information System (QGIS) on avoimen lähdekoodin paikkatieto-ohjelmisto, joka on erityisen suosittu kartta-aineistojen hallintaan, muokkaamiseen ja analysointiin. Ohjelma tarjoaa laajan valikoiman työkaluja erilaisten paikkatietomuotojen käsittelyyn, ja se tukee sekä rasteri- että vektoridataa.

Tutkimuksessa QGIS:n rooli oli keskittynyt rasterimateriaalin tuottamiseen. Tämä tarkoitti lähinnä kartta-aineiston esikäsittelyä ja muokkaamista sellaisiksi tiedostoiksi, jotka voidaan helposti integroida muihin järjestelmiin, kuten MBTiles-tiedostoihin. Ohjelmalla luotiin useampi aineisto, jotka toimivat karttatasoina. Tuotettu rasteriaineisto pysyi samana koko tutkimuksen ajan, ja aineistoa hyödynnettiin karttapalvelimien vertaamiseen.

3.4 Kokoonpanon testaus

Minimaalisen käyttöliittymän avulla oli mahdollista manuaalisesti testata karttapalvelun eri toimintoja. Käyttöliittymässä testattiin karttatasojen toimivuutta, niiden välillä vaihtamista, sekä kuinka nopeasti kartta latautui käyttäjälle, ja yleistä resurssien käyttöä. Näistä latausnopeus ja resurssien käyttö olivat helpoimmat testata, sillä ne olivat mitattavia arvoja. Karttapalvelun ominaisten toimintojen testaaminen vaati kuitenkin hiomista käyttöliittymään. Vaikka itse karttapalvelut tarjoavat yleisten standardien mukaista dataa, miten data tarjoiltiin, vaihteli paljon. Tälle on myös standardeja, mutta kaikki karttapalvelut eivät käyttäneet samaa.

Tutkimus olisi voinut olla laajempi, mikäli olisi tyydytty toimintojen vertailuun paperilla, mutta ohjelmistokehityksessä voi helposti tulla yllättäviä haasteita varsinaisessa käyttöönotossa. Nähtiin paremmaksi toteuttaa kokonaisuus oikeasti, jotta jatkokehitys on sujuvaa.

3.5 Tulosten varmentaminen

Tutkimuksen aikana pidettiin katsastustilaisuuksia tuloksista viikoittain. Näissä tilaisuuksissa oli Combitech:n yhteyshenkilö, sekä mahdollisuuksien mukaan muita yrityksen kehittäjiä ja arkkitehtejä. Tilaisuuksissa pystyttiin vaihtamaan ideoita tulevaan toteutukseen, sekä tarkentamaan tutkittavia toimintoja. Vaatimuslistakin laajentui näissä tilaisuuksissa hieman, jotka johtivat Taulukko 1:n lopulliseen muotoon. Tilaisuuksien tueksi tehtiin jotain näytettävää, olipahan se mitattua tilastoa suorituskyvystä, karttapalveluita käyttävä käyttöliittymä esittämään palvelujen toimintaa, tai suoraan koodia siitä kiinnostuneille.

4 Tutkimuksen toteutus

4.1 Valmiudet työlle

Tutkimuksen toteutus rakentui vankalle pohjalle, sillä paikkatietojärjestelmien kanssa työskentelyoli ennalta tuttuja. Tämä mahdollisti keskittymisen tutkimuksen varsinaisiin kysymyksiin ja nopeutti prototyyppien rakentamista. Tutkimustyössä käytettiin iteratiivista lähestymistapaa, jossa kokeiltiin ja arvioitiin erilaisia teknologioita ja niiden soveltuvuutta tutkimuksen tavoitteisiin.

Tutkimusprosessia tuki myös vahva tekninen tausta ja hyvät lähtöasetelmat. Erityisesti Combitechin ennakkoon tunnistamat kirjastot tarjosivat selkeän suunnan tutkimustyölle heti alusta alkaen. Näin vältettiin laajan ja aikaa vievän kartoittavan vaiheen tarve, ja työssä pystyttiin keskittymään valikoituihin teknologioihin niiden ominaisuuksien perusteellisessa arvioinnissa.

4.2 Etukäteen tunnistetut teknologiat

Ennen tutkimuksen aloittamista, Combitech oli jo kartoittanut pari potentiaalista karttapalvelinta, jotka voisivat sopia heidän käyttöönsä. Varsinaista perustetta näille ei annettu, mutta se kuitenkin oli apu tutkimuksen aloittamiselle. Käyttöliittymää mukautettiin hieman hyödyntämään näiden karttapalveluiden ominaisuuksia, jotta demo tilaisuuksissa voitiin nähdä kaikki mitä ne tarjoavat.

4.2.1 Mbtileserver

mbtileserver on kevyt ja helposti käyttöön otettava palvelin, joka on suunniteltu tarjoamaan karttalaattoja MBTiles-formaatista. Tämä teknologia on erityisen hyödyllinen ympäristöissä, joissa tarvitaan nopeasti ja tehokkaasti toimivia karttapalveluita.

Tutkimuksessa mbtileserverin suurimmaksi eduksi nousi sen yksinkertaisuus ja suoraviivaisuus. Palvelimen pystytys onnistui nopeasti, ja se vaati vain vähän resursseja toimiakseen luotettavasti. Mbtileserver tuki sekä rasteri- että vektoritiiliä, joka ei ole taattu pelkästään MBTiles tiedostoformaatin tuella. Erityisesti mbtileserverin tarjoama TileJSON rajapinta oli erittäin mukava käyttää. Mbtileserver tarjosi yksinkertaisen indeksin, joka näytti kaikki sen sisältävän aineiston. Tämä indeksi oli hyvin helppo käydä lävitse käyttöliittymässä iteratiivisesti, jolla pystyi generoimaan karttatasot pienimmällä vaivalla mahdollista. Se oli myös erittäin nopea, todennäköisesti koska se ei tarjoa mitään ylimääräisiä toimintoja.

Ehkä kaikkein paras toiminto oli mbtileserverin itsensä päivittäminen. Mikäli palvelimelle tuotiin uutta aineistoa, mbtileserver osaa automaattisesti lukea sen metadatan, ja lisätä sen palvelimen tarjontaan. Käänteisesti myös poistaminen on automaattista. Tähän ei mikään muu ratkaisu pystynyt oletuksena.

Ottaen kuitenkin huomioon, että tätä projektia varten olennaisimmat ominaisuudet on rajapintojen laaja tuki, mbtileserverin WMS ja WMTS rajapintojen tuen puute oli harmillista. Mbtiletileserveristä puuttui myös rasterointi, eli vektoridatan muuttaminen rasteriksi. Tämä rajoittaa pois joitakin vanhempia käyttöliittymiä, sillä vektoritiilet ovat suhteellisen uusi idea.

4.2.2 Tileserver-GL ja Tileserver-GL-light

Tileserver-GL on selkeästi isoin ohjelma tutkimuskohteista. Mikäli etsii GIS ohjelmiin liittyviä kysymyksiä, on hyvin tavallista, että hakutuloksiin ilmestyy kysymyksiä GIS kehittäjiltä foorumeilla, jotka ovat Tileserver-GL kohtaisia. Ilmiölle on kyllä hyvä perusta, sillä Tileserver-GL on oikeasti hyvä.

Tileserver-GL on todennäköisesti toimintorikkain karttapalvelin olemassa tänä päivänä. Siinäkin on kuitenkin pari puutetta. Tileserver-GL:n isoin myyntivaltti on rasterointi. Rasterointi antoi mahdollisuuden tarjota vektoriaineistoa reaaliajassa rasterina, joka on olennaista vanhemmille ohjelmille. Tämä rasterointi myös tukee eri tyylejä, joka mahdollistaa kartan ulkonäön muuttamisen helposti. Rasteriaineistoilla tämä ei ole yksinkertaisesti mahdollista, ellei rakenneta aineistoa alusta lähtien uudestaan eri parametreilla.

Tileserver-GL myös tuki rajapintoja kaikkein laajimmin, tarjoten WMTS ja XYZ skeemat. TileJSON pohjaista indeksiä ei ole, mutta palvelin tarjoaa oman tyylisensä indeksinsä, joka kertoo mitä aineistoa palvelimella on. Toisin kuin mbtileserver, tämä indeksi ei kerro aineiston metadataa, eikä suoraa rajapinta linkkiä kyseiseen aineistoon. Jälkimmäinen oli vain pieni vika kuitenkin, sillä linkin rakentaminen dynaamisesti on helppoa. Itse aineistoilla on TileJSON standardin mukainen dokumentti, mutta nämä ei ole yhtä kätevä hakea kuin yhtenäisestä indeksistä.

Tileserver-GL:n mukana tuli myös ominaisuuksia, joita ei tarvita valmiissa projektissa, eikä niitä voinut ottaa pois päältä, joka tekisi toimitetusta palvelusta raskaamman ajaa. Tähän kuuluu selattava käyttöliittymä, josta pystyi avaamaan aineistoa esikatseluun. Kyseinen esikatselu voisi toimia erittäin minimaalisena käyttöliittymä ratkaisuna, mutta koska tämä tukee vain yhtä aineistoa kerrallaan, tämä ei täytä vaatimuksia.

Tämän perusteella katsottiin myös Tileserver-GL:n kevyempää ”light” versiota. Tämä ei kuitenkaan ratkaissut mitään, ja ainut ero täyteen versioon on rasteroinnin puuttuminen. Tällä on valitettavasti vakavia sivuvaikutuksia.

Tileserver-GL on tehnyt valitettavan päätöksen, että tietyt rajapinnat, jotka oikeassa maailmassa ei ole rasterointi kohtaisia, ovat riippuvaisia tässä palvelimessa nimenomaan rasteroinnista. Vektoriaineisto ei toimi ollenkaan ilman rasteroinnin olemassaoloa, vaikka täysi versio tukee tätä normaalisti. Myös WMTS rajapinta on lukittu nimenomaan rasteroituun aineistoon, joten ilman rasterointia tietenkään ei voi saada WMTS rajapintaa. Tileserver-GL-light ei ollut lopulta edes vaihtoehto, vaan tyrmättiin kokonaan.

4.3 Toteutuksen aikaiset löydökset

4.3.1 MapProxy

Etsiessä ratkaisua Mbtileserverin vajaalle rajapintojen tuelle, vastaan tuli MapProxy ohjelma. MapProxy ei ole itsenäinen karttapalvelin, vaan kääntää olemassa olevan rajapinnan toiseen muotoon, ja pystyy vielä manipuloimaan karttadataa kehittäjän parametrien mukaan. MapProxy projektissa toimisi laastarina. Jos erityisesti tykätään jostakin palvelimesta, mutta se ei tue WMS tai WMTS rajapintoja, MapProxy voidaan ottaa mukaan arkkitehtuuriin välikädeksi.

Tässä on selviä etuja sekä haittoja. Etuna tietenkin on joustavuus arkkitehtuurissa, sillä enää ei tarvitse välittää paljon mitä karttapalvelin tarjoaa, ja voidaan taata laaja tuki mihin tahansa käyttöön. Haittana tietenkin on ylimääräinen pala arkkitehtuurissa, sillä MapProxy ei voi toimia yksin. MapProxy oli myös hankala konfiguroida toimivaksi, ja siinä oli dokumentoimattomia bugeja, ja outoja interaktioita Tileserver-GL:n kanssa. Eräs ärsyttävä bugi esiintyi Tileserver-GL:llä tuotetussa rasteroidulla aineistolla. MapProxyn tuomana kaikki tasot skaalautuivat isommaksi, joka rasteriaineistolla aiheuttaa pahaa pikselöintiä, joka tekee kartasta hankalaa lukea. MapProxy ei myöskään tukenut vektori materiaalia ollenkaan, joka tarkoittaa, että ainoastaan Tileserver-GL voisi käyttää vektori aineistoa MapProxyssä rasterointinsa ansiosta, mutta siitä ei ole suuresti iloa, kun Tileserver-GL jo tukee kaikkia olennaisia rajapintoja.

MapProxyä ei mielellään oteta käyttöön, sillä haitat ovat suuremmat kuin hyödyt, mutta se on hyvä oljenkorsi, mikäli se osoittautuu tarpeelliseksi. Suurimmat haasteet kuitenkin sen kanssa on jo ratkaistu, ja toimivaksi ja hyväksi todetut konfiguraatiot on julkaistu sisäisesti.

4.3.2 Protomaps ja PMTiles standardi

Tutkimuksen lopussa tuli vastaan vielä yksi tapa tuoda karttatiiliä käyttäjille. Sen sijaan että käytettäisiin palvelinta, joka hallinnoi tiedostoja ja tarjoaa kyselyä vastaan oikeaa dataa käyttöliittymälle, Protomaps tarjoaisi mahdollisuuden ohittaa palvelimen tarpeen kokonaan ja lukea suoraan tiedostoa. PMTiles itsessään on henkistä jatkoa MBTiles standardille, ja ohjelmat, jotka tukevat MBTiles standardia saadaan pienellä vaivalla tukemaan myös PMTiles standardia. PMTiles tiedosto ei juuri kuitenkaan eroa MBTiles tiedostosta kooltaan, joten

Tästä olisi valtava etu arkkitehtuurin yksinkertaistamiseen, sillä enää tarvittaisiin vain yksi ohjelma, itse käyttöliittymä. Tällä hypättäisiin yli askelia, ja lopputulos tällöin on nopeampi. Tämä palaa käytännössä monoliittiseen arkkitehtuuriin, jossa on etuja täysin itsenäisessä ratkaisussa. Kuten voi olettaa, yhden ohjelman jakaminen ilman ulkoisia riippuvuuksia toisiin ohjelmiin on paljon helpompi asentaa kuin täysi mikropalveluarkkitehtuuri mahdollisesti tusinalla eri palveluita.

Teknologia on kuitenkin hyvin nuori verrattuna muihin ratkaisuihin, sekä esittää omat rajoitteensa. Koska web teknologialla rakennettu ohjelma ei voi suoraan lukea tiedostoja käyttäjän tietokoneelta kovakoodattujen turvallisuus määrityksen vuoksi, aineisto ei koskaan voi olla yhtä dynaamista kuin karttapalvelimen tukemana. Kuten MapProxy, Protomaps tuskin tulee näkemään käyttöä rajoitteidensa vuoksi, mutta on hyvä olla tiedossa tulevaisuudessa.

5 Tutkimuksen tulokset

5.1 Vaatimusten toteutuminen

Tutkimuksen päätteeksi koostettiin eri mittojen mukaan kunkin karttapalvelimen vahvuudet. Kaikista tärkeintä kuitenkin on projektivaatimusten täyttyminen. Huomioon otettiin vaatimuksien toteutuminen raksimalla palvelinkohtaisesti listalta toiminto toteutuneeksi. Tulosten perusteella

päätettiin lopullinen palvelu, joka menee jatkokehitykseen. Tietyillä vaatimuksilla on myös suurempi merkitys, jotka olivat Taulukko 1. Projektin vaatimukset taulukossa merkattu kriittisiksi vaatimuksiksi. Oheisessa taulukossa kunkin kirjaston toteuttamista vaatimuksista, kriittiset vaatimukset on merkattu paksunnetulla tekstillä.

Taulukko 2. Vaatimusten toteutuminen palvelu kohtaisesti

Vaatimus	Mbtileserver	Tileserver-GL	Selite
Tekninen-01	✓	✓	GeoTIFF tuki itsessään on sekundaarinen, joita kumpikaan ei tue. MBTiles tuki löytyy molemmista,
Tekninen-02	✓	✓	Tileserver-GL vaatii pienen mikropalvelun aineiston muutoksen tunnistamiseen, mutta tätä ei nähdä ongelmaksi.
Tekninen-03	✓	✓	Molemmista löytyy hyvät rajapinnat aineiston haulle.
Tekninen-04	X	X	Ei kuulu karttapalvelun ominaisuuksiin, käyttöliittymä valitsee olemassa olevan aineiston välillä.
Tekninen-05	X	✓	Vain Tileserver-GL tukee WMS ja WMTS rajapintoja ilman MapProxyn tukea.
Tekninen-06	✓	✓	Mbtileserver on jonkin verran kevyempi, mutta nykypäivän tietokoneilla Tileserver-GL:kään ei ole raskas.
Tekninen-07	✓	✓	Tukevat Dockeria sekä natiivia asennusta käyttöjärjestelmästä riippumatta. Vaihtoehtoja löytyy siis.
Tekninen-08	✓	✓	Pystyvät hakemaan aineistoa paikallisesti, eikä ole riippuvaisia verkkoyhteydestä.
Tekninen-09	X	✓	Vain Tileserver-GL pystyy yhdistämään tasoja yhdeksi tasoksi tyylien avulla.

Tekninen-10	X	✓	Vain Tileserver-GL:n tuottama metadata antaa tarpeeksi tietoa erittämään aineistoa karttakerroksiin.
Tekninen-11	✓	✓	Molemmat tukevat rasteriaineistoa.
Tekninen-12	✓	✓	Molemmat tukevat vektoriaineistoa.
Tekninen-13	✓	✓	Pystyy yhdistämään molemmissa.
Yleinen-01	✓	✓	Lisenssit sallivat kaupallisen käytön.
Yleinen-02	✓	✓	Tileserver-GL on suositumpi, mutta molemmat ovat kehuttuja ohjelmia. Tileserver-GL:n kehittäjät ovat myös aktiivisempia kuin Mbtileserverin

Molemmat kirjastot osuvat useampaan kriittiseen vaatimukseen, mutta vain Tileserver-GL tukee WMTS rajapintaa suoraan hyllystä. Vaikka nämä ei olleet vaatimuksia, Tileserver-GL:n rasterointi toiminto sekä vektorien tyylien määrittäminen on valtava plussa.

5.2 Suorituskyky

Suorituskyvyn mittaaminen tapahtui lataamalla sama aineisto kuhunkin tiilipalvelimeen, käynnistämällä käyttöliittymä, ja avaamalla palvelimen aineisto. Aineisto sisältää vektoripohjaisen pohjakartan OpenStreetMap palvelusta, sekä satunnaisia rasterikerroksia maanmittaus laitoksen karttapaikka palvelusta. Aineisto koottiin MBTiles tiedostoihin QGIS ohjelman avulla.

Palvelin kirjaa ylös milloin pyyntö tiileen on tullut ja milloin tiili on toimitettu. Nämä tiedot riittävät saamaan yhteensä menneen ajan palvelimella ensimmäisestä tiilestä viimeiseen saakka. Käyttöliittymällä kuluneella ajalla ei ole merkitystä, sillä se ei kuulu tutkimuksen piiriin. Testi toistettiin 10 kertaa kullakin palvelimella vähentääkseen virhemarginaalia. Tileserver-GL mitattiin ilman rasterointia, sekä rasteroinnilla erikseen.

Taulukko 3. Palvelimien suorituskyvyn mittaustuloksia

Tiilipalvelin	Mbtileserver	Tileserver-GL	Tileserver-GL rasteroituna
Yhteensä	97,81 ms	292,77 ms	20 908,25 ms
Maksimi	3,71 ms	8,59 ms	2 559,94 ms
Minimi	0,03 ms	0,43 ms	37,14 ms
Mean	0,72 ms	2,17 ms	464,63 ms
Mediaani	0,25 ms	1,89 ms	113,82 ms

Mbtileserver on karttapalvelimista nopein melko suurella marginaalilla. Tämä osittain on käytetyn kielen ansiota, sillä mbtileserverissä käytetty Go ohjelmointikieli on tilastollisesti nopeampi kuin JavaScript, johon Tileserver-GL perustuu (Mayank 2023). Toinen mahdollinen syy on Mbtileserverin yksinkertaisuus. Siinä ei ole mitään ylimääräistä, joka voisi hidastaa karttatiilien hakemista. Toinen huomio on rasteroinnin hitaus. Rasterointi on kallis prosessi, etenkin prosessorilla suoritettuna. Tämä olisi mahdollista nopeuttaa käyttötarkoituksen mukaisella grafiikka prosessorilla, mutta testausympäristössä tämä ei ollut mahdollista. Ylimääräinen rasterointi prosessi tulee kuitenkin aina olemaan hitaampi verrattuna suoraan vektori materiaalin käyttämiseen.

5.3 Lopullinen päätös

Työn lopuksi esitettiin tutkimuksen löydökset. Palaveriin osallistui Combitechin arkkitehtejä ja kehittäjiä, myös heitä, joilla ei ole omaa panosta projektiin. Palaverissa itsessään ei tehty päätöstä, mutta annetun tiedon perusteella Combitechilla oli perustat tietoisen päätöksen tekemiseen.

Tileserver-GL lopulta voitti, sillä se täytti kaikki kriittiset vaatimukset ilman ylimääräistä palvelua taustalle tukemaan sitä. Sille ominainen rasterointi oli myös toiminnallisuus, joka nähtiin hyödylliseksi.

Jatkokehityksen tueksi, kaikki dokumentaatio, oma koodi ja konfiguraatio julkaistiin Combitechin sisäverkossa, joka on saanut hyvää palautetta kehittäjiltä, joille toimeksianto menee tuotantoon.

6 Pohdinta

Kovasta aikapaineesta huolimatta tutkittavista teknologioista saatiin syvällisesti selvitettyä kaikki oleellinen, jonka perusteella pystyttiin tekemään selkeitä ja perusteltuja ratkaisuja. Tutkimus tuotti konkreettisia tuloksia, joiden pohjalta oli mahdollista käynnistää tuotanto sujuvasti ja varmuudella. Tutkimuksen aikana löytyi mahdollisuuksia lisätoiminnallisuuksille, jotka voidaan tarvittaessa ottaa käyttöön tulevaisuudessa. Combitech:lta saatu palaute oli kiittävää, mikä vahvistaa tutkimuksen onnistuneisuutta ja käytännön merkitystä.

Vaikka olisi ollut kiinnostavaa tutkia muitakin karttapalvelinvaihtoehtoja, lopputuloksiin voi olla tyytyväinen. Tutkimusprosessi itsessään oli myös merkittävä oppimiskokemus. Paikkatietojärjestelmät olivat jo entuudestaan tuttuja, mutta tutkimuksen myötä ymmärrys itsenäisten arkkitehtuurien suunnittelusta ja toteutuksesta syveni huomattavasti. Jatkuva itsenäisen ja modulaarisen arkkitehtuurin ylläpito oli myös haaste, josta tuli opittua paljon.

Lähteet

2023 Developer Survey. 2023. Yleisötutkimus suosituista teknologioista. Viitattu 7.11. https://survey.stackoverflow.co/2023/#section-most-popular-technologies-other-tools.

Alexandra. What is N-Tier Architecture? How It Works, Examples, Tutorials, and More. 2023. Viitattu 12.8.2024. https://stackify.com/n-tier-architecture/.

Allen, M. 2020. OGC Web Services. Viitattu 1.12.2024. https://markallengis.medium.com/ogc-web-services-d3186fdc988f.

Atlassian. What is Docker? A guide to containerization. 2024. Viitattu 7.11.2024. https://www.atlassian.com/microservices/microservices-architecture/docker.

Avoin lähdekoodi - mitä se on ja mitkä ovat sen edut ja haitat?. 2023. Viitattu 7.10.2024. https://www.haltu.fi/blogi/avoin-lahdekoodi-mita-se-on.

Harmain Arif. 2023. Why UX Matters: Unraveling the Secrets of User Experience Design. Viitattu 7.11.2024. https://www.linkedin.com/pulse/foundation-user-experience-ux-design-harmain-arif-5x0df.

How website performance affects conversion rates. N.d. Viitattu 7.11.2024. https://www.cloudflare.com/learning/performance/more/website-performance-conversion-rates/.

Jonker, A. 2023. What is a geographic information system (GIS)?. Viitattu 7.10.2024. https://www.ibm.com/topics/geographic-information-system.

Leaflet. 2024. Leaflet kirjaston lähdekoodi. Viitattu 30.11.2024. https://github.com/Leaflet/Leaflet.

MacWright, T. 2010. MBTiles 1.0. Viitattu 12.8.2024. https://github.com/mapbox/mbtiles-spec/blob/master/1.0/spec.md

Martin, E. 2024. Raster vs Vector Map Tiles: What is the Difference Between the Two Data Types?. Viitattu 1.12.2024. https://documentation.maptiler.com/hc/en-us/articles/4411234458385-Raster-vs-Vector-Map-Tiles-What-is-the-Difference-Between-the-Two-Data-Types.

Mayank, C. 2023. Node.js vs Go: Performance comparison for JWT verify and MySQL query. Viitattu 2.12.2024. https://medium.com/deno-the-complete-reference/node-js-vs-go-performance-comparison-for-jwt-verify-and-mysql-query-98231cd22407

Paul, C. 2022. The State of Enterprise Open Source: A Red Hat report. Viitattu 6.10.2024. https://www.redhat.com/en/resources/state-of-enterprise-open-source-report-2022.

Software Development Life Cycle (SDLC). 2024. Viitattu 7.10.2024. https://www.geeksforgeeks.org/software-development-life-cycle-sdlc/.

The web and web standards. N.d. Viitattu 7.10.2024. https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/The_web_and_web_standards.

Tietoa meistä. N.d. Combitechin yritys esittely sivu. Viitattu 6.10.2024. https://www.combi-tech.fi/tietoja-meista/.

Usage statistics of JavaScript as client-side programming language on websites. N.d. Haettu 26.11.2024. https://w3techs.com/technologies/details/cp-javascript.

Viki Green. 2016. Impact of slow page load time on website performance. Viitattu 7.11.2024. https://medium.com/@vikigreen/impact-of-slow-page-load-time-on-website-performance-40d5c9ce568a.

Yaskiv, A & Dolynyuk T. 2021. Software Architecture Types: Monolith vs Microservices. Viitattu 6.10.2024. https://apiko.com/blog/software-architecture-types-monolith-vs-microservices/.