jamk.fi

Analysaattorituotteen pilveyttäminen

Joonas Siltala

Opinnäytetyö
Joulukuu 2015
Tekniikan ja liikenteen ala
Tietotekniikan koulutusohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Siltala, Joonas	Opinnäytetyö	02.12.2015
	Sivumäärä 55	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Analysaattorituotteen pilveyttäminen

Tutkinto-ohjelma
Tietotekniikan koulutusohjelma

Työn ohjaaja(t)
Mika Rantonen, Antti Häkkinen

Toimeksiantaja(t)
Environics Oy, Osmo Anttalainen

Tiivistelmä

Opinnäytetyön toimeksiantajana toimi CBRN-tuotteita valmistava Environics Oy. Opinnäytetyön tavoitteena oli selvittää, mitä mahdollisuuksia pilvipalvelut tarjoavat ja mitä hyötyä niiden käytöstä voisi olla toimeksiantajalle ja heidän analysaattorituotteelleen. Lisäksi tavoitteena oli suunnitella ja toteuttaa toimeksiantajan analysaattorilaitteiden rinnalle uusi järjestelmä niiden tuottaman mittausdatan analysointia varten.

Työssä tutkittiin ja vertailtiin erilaisia pilvipalvelumalleja ja paikalliseen palvelinfarmiin perustuvia ratkaisuja. Tämän tiedon perusteella valittiin toimeksiantajan kanssa heille sopivin ratkaisu ja toteutuksessa käytettävät teknologiat. Tietoturvasyistä päädyttiin paikalliseen palvelinfarmiin perustuvaan ratkaisuun, ja sen perusteella suunniteltiin konsepti tuotantokäyttöön soveltuvasta palvelinfarmista. Lopuksi valittujen teknologioiden avulla toteutettiin toimiva järjestelmä yhden fyysisen palvelinkoneen pilottiympäristössä.

Analysaattorilaitteiden tuottaman mittausdatan suuren määrän takia levytilan kulutus oli yksi tärkeimpiä huomioon otettavia asioita. Muita vaatimuksia olivat tietokantamoottorilla suoritettavat monimutkaisemmat analysointitoiminnot sekä web-pohjainen käyttöliittymä. Vaatimusten perusteella käytettäviksi teknologioiksi valittiin Node.js-pohjainen web-sovellusympäristö ja MongoDB-tietokanta.

Työn tuloksena saatiin koottua toimeksiantajalle kattava tietopaketti pilvipalveluista ja niiden tarjoamista mahdollisuuksista. Lisäksi saatiin onnistuneesti suunniteltua ja toteutettua vaatimusten mukainen yksinkertainen järjestelmä analysaattorilaitteiden tuottaman mittausdatan analysointiin. Pilottiympäristössä toteutettu järjestelmä tarjoaa toimeksiantajalle pohjan varsinaisen tuotannossa käytettävän järjestelmän kehittämiseen.

Avainsanat (asiasanat)

pilvipalvelut, mittausdata, tietokanta, web-sovellus, NoSQL, MongoDB, Node.js

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Siltala, Joonas	Bachelor's thesis	02.12.2015
		Language of publication: Finnish
	Number of pages: 55	Permission for web publication: x

Title of publication

Cloudization of an analyzer product

Degree programme

Information Technology

Supervisor(s)

Mika Rantonen, Antti Häkkinen

Assigned by

Environics Oy, Osmo Anttalainen

Abstract

The Bachelor's thesis was assigned by Environics Oy, a supplier of CBRN products. The purpose of the thesis was to find out what kind of possibilities cloud computing has to offer and how utilizing them could benefit the company and their analyzer products. The goal was to design and implement a new system for analyzing the sensor data produced by their analyzer products.

Different cloud computing models and server farm solutions were compared with each other. Based on the acquired information, the most suitable solution and the technologies used for implementation were selected. Due to security reasons, a server farm based solution was selected and a concept of production deployment was designed. Finally, a working system was implemented using the selected technologies in a one server pilot environment.

Due to the large amount of sensor data produced by the analyzers, the usage of disk space was one of the most important factors to consider. Other requirements included the ability to perform more complex operations using the database engine and a web-based user interface. Based on the requirements, Node.js web application environment and MongoDB database were used for the actual implementation.

As a result of the thesis, a comprehensive information package about cloud computing and the potential they provide was presented to the assigner. Additionally, a basic system for analyzing the sensor data produced by the company's analyzer products was successfully designed and implemented. The system implemented in the pilot environment provides the assigner with a basis on which to build and develop the actual system for production deployment.

Keywords/tags (subjects)

cloud computing, sensor data, database, web application, NoSQL, MongoDB, Node.js

Miscellaneous

Sisältö

Kuviot

Taulukot

Lyhenteet

ACL	Access Control List
API	Application Programming Interface
AWS	Amazon Web Services
BSON	Binary JSON
CBRN	Chemical, Biological, Radiological, Nuclear
CSS	Cascading Style Sheets
DMZ	Demilitarized Zone
DNS	Domain Name System
EBS	Elastic Block Store
EC2	Elastic Compute Cloud
ELB	Elastic Load Balancing
EMR	Elastic MapReduce
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
I/O	Input/Output
IP	Internet Protocol
IPsec	Internet Protocol Security
JSON	JavaScript Object Notation
KATAKRI	Kansallinen turvallisuusauditointikriteeristö
MVC	Model-View-Controller
MVVM	Model View ViewModel
NoSQL	Non-SQL, Not only SQL
NPM	Node Package Manager
PaaS	Platform as a Service

RDS	Relational Database Service
S3	Simple Storage Service
SaaS	Software as a Service
SQL	Structured Query Language
SSH	Secure Shell
URL	Uniform Resource Locator
VPC	Virtual Private Cloud
VPN	Virtual Private Network
VPS	Virtual Private Server
XHR	XMLHttpRequest

1 Työn lähtökohdat

1.1 Environics Oy

Environics Oy toimittaa CBRN-tuotteita (Chemical, Biological, Radiological, Nuclear) eli kemikaalisten, biologisten, ydin- ja muun säteilyn havaitsemiseen tarkoitettuja laitteita ja järjestelmiä. Tuotteita käyttävät niin turvallisuusvirastot kuin asevoimatkin aina henkilökohtaisesta suojauksesta kansalliseen turvallisuuteen asti. Tämän lisäksi Environics tarjoaa myös teollisuudessa kaasujen havaitsemiseen tarkoitettuja järjestelmiä. Yritys työllistää noin 60 henkilöä päätoimipaikassaan Mikkelissä ja toimii jälleenmyyjäverkostonsa kautta yli 40 maassa kaikilla mantereilla. (Company n.d.)

1.2 Toimeksianto ja tavoitteet

Opinnäytetyön tavoitteena oli selvittää, onko Environics Oy:n analysaattorituotteelle löydettävissä teknisesti toimiva ja taloudellisesti kannattava, pilvipohjainen tai paikalliseen palvelinfarmiin perustuva ratkaisu, jossa on riittävällä tavalla otettu huomioon yrityksen ja asiakkaiden tarpeet tietoliikenneratkaisujen, tietoturvan ja käyttöliittymän toiminnallisuuksien osalta.

Teollisuudessa on siirrytty yhä enemmän pilvipohjaisten ratkaisujen käyttöön niin tiedon hallinnan ja analysoinnin, kunnossapidon kuin perustoimintojenkin osalta. Pilvipohjainen ratkaisu voisi tarjota nykyaikaisen mallin, jolla yrityksen kannalta laitteiden ja ohjelmistojen ylläpito ja kehitys helpottuisi ja asiakkaalle pystyttäisiin tarjoamaan uudenlaisia lisäpalveluja ja -ominaisuuksia.

Toisaalta paikallinen palvelinfarmi taas voisi tarjota yritykselle perinteisemmän, täysin heidän hallinnassaan olevan, muista osapuolista riippumattoman ratkaisun. Paikallisessa ratkaisussa yritys hoitaa itse koko järjestelmän infrastruktuurin hallitsemisen omissa tiloissaan vaatimatta minkäänlaista yhteyttä sen ulkopuolelle ja näin lisäten sen luotettavuutta ja turvallisuutta.

Opinnäytetyössä selvitettiin ensin yrityksen ja sen asiakkaiden tarpeet järjestelmän ylläpitoon ja tiedon analysointiin liittyen pääosin yrityksessä jo olevan tiedon perusteella. Tämän perusteella arvioitiin, mitä etuja pilvipalvelun tai paikallisen palvelinfarmin käyttämisellä olisi nyt ja tulevaisuudessa yrityksen ja sen asiakkaiden kannalta.

Varsinaisessa toteutusvaiheessa selvitettiin ensin, mitä erilaisia toteuttamistapoja ratkaisulle on olemassa ottaen huomioon yrityksen ja asiakkaiden IT-infrastruktuurin asettamat haasteet sekä yrityksen ja asiakkaiden tietoturvavaatimukset ja toteutukseen liittyvät tietoturvariskit. Lopuksi valittiin näistä tarkoitukseen parhaiten sopiva ratkaisu ja toteutettiin valitun ratkaisun pohjalta pilottiversio järjestelmästä ja testattiin sen toimivuus pilottivaiheessa yrityksen analysaattorilaitteiden ja toteutetun järjestelmän välillä.

2 Tietoperusta

2.1 Pilvipalvelut

Yksinkertaisuudessaan pilvipalveluilla tarkoitetaan palveluita, joita tarjotaan verkkoyhteyden yli. Toisin sanoen paikallisten laitteiden ja ohjelmistojen sijasta käytetään jaettuja fyysisiä ja virtuaalisia resursseja verkon välityksellä. (What is Cloud Computing? n.d.)

Pilvipalvelujen tarkoituksena on tarjota suuri määrä laskentatehoa, datan säilytystilaa tietoliikenneinfrastruktuuria ja sovelluksia täysin virtualisoidulla tavalla yhdistelemällä resursseja ja tarjoamalla asiakkaalle yhdistetyn kuvan koko järjestelmästä. Yleensä tähän liittyy myös resurssien tarjoaminen sitä mukaan, kun asiakas niitä tarvitsee, ja laskutus kyseisten resurssien käytön perusteella. (Buyya, Broberg & Goscinski 2011.)

Pilvipalvelu-termin pilvellä tarkoitetaan mallia, jossa koko IT-infrastruktuuria voidaan ajatella pilvenä eli joukkona resursseja. Tämän pilven asiakkaat voivat sitten tarpeen mukaan päästä käsiksi kyseisiin resursseihin verkon välityksellä fyysisestä sijainnista riippumatta. Termin palvelu-sanan taustalla on nimenomaan ajatus siitä, että resursseja kuten laskentatehoa ja datan säilytystilaa tarjotaan palveluna asiakkaalle. (Mt.)

Oleellisiin konsepteihin kuuluu myös se, että palvelujen taustalla oleva laitteisto ja arkkitehtuuri ovat täysin abstraktoituja ja asiakkaalle tarjotaan vain virtualisoinnin mahdollistama rajapinta (Mt).

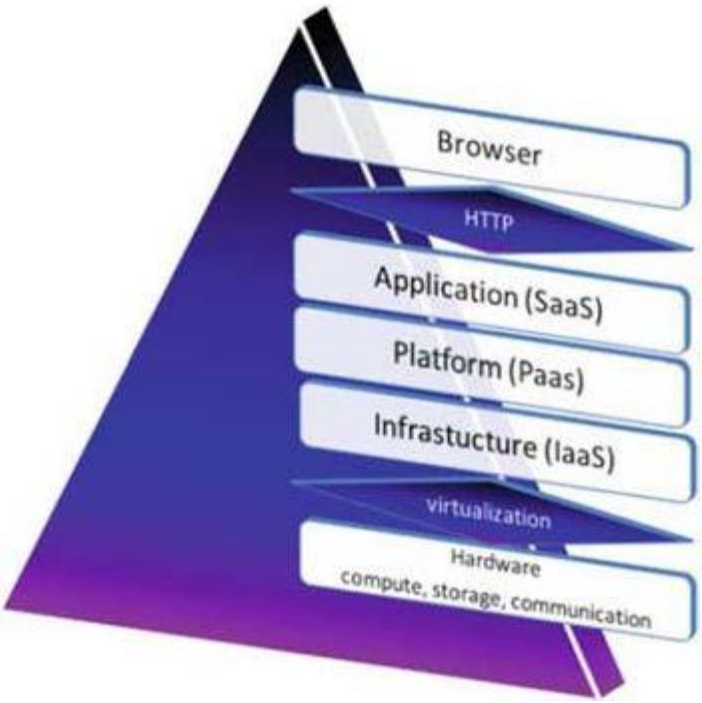
Pilvipalvelut tarjoavat asiakkaalle näennäisesti äärettömän määrän käytettävissä olevia resursseja. Elastisuudella tarkoitetaan sitä, että asiakas voi halutessaan lisätä tai poistaa käytössä olevia resursseja, mahdollisesti jopa automaattisesti kuorman kasvaessa tai vähentyessä. (Mt.)

2.2 Palvelumallit

2.2.1 Yleistä

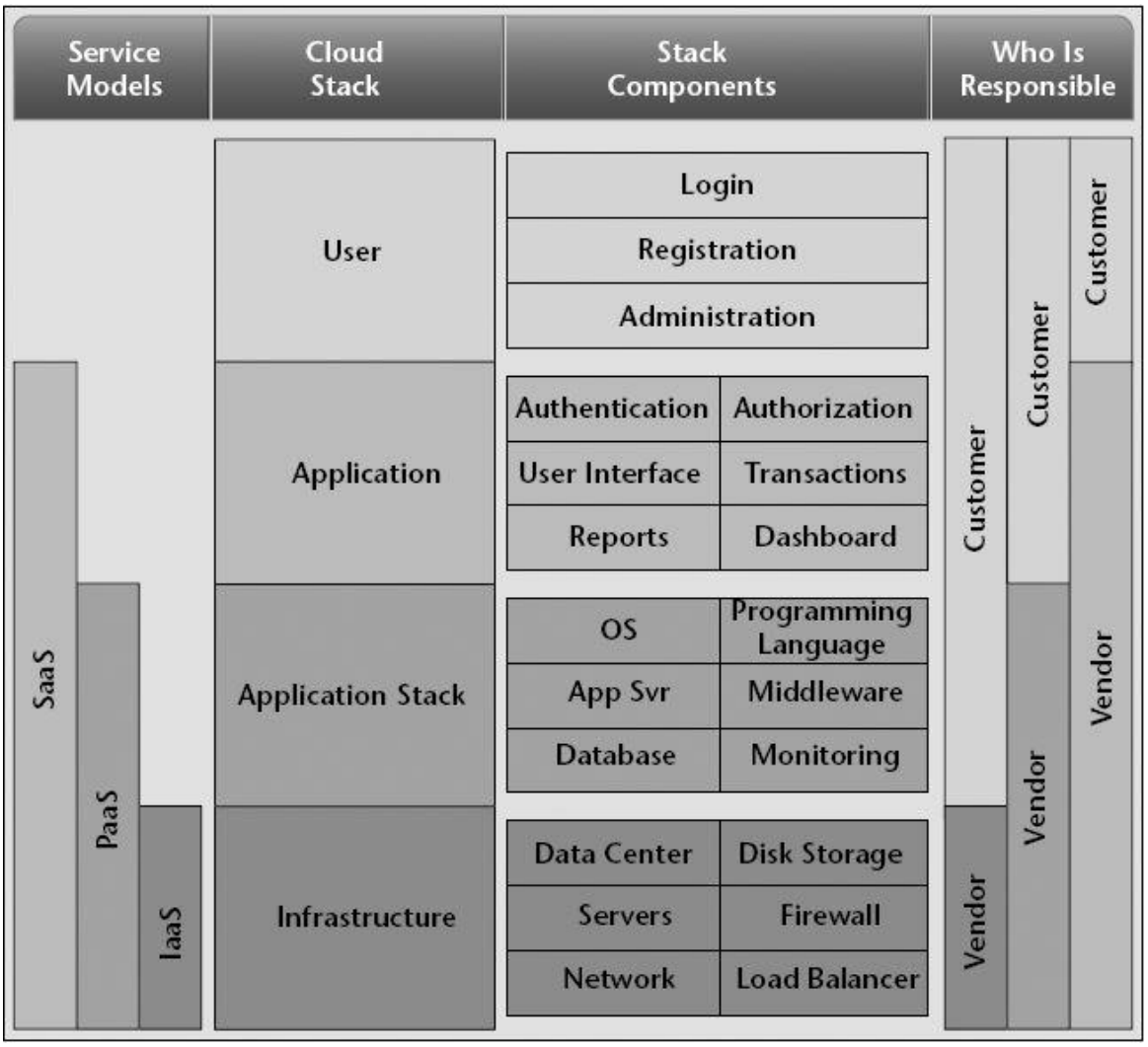
Pilvipalvelut voidaan karkeasti jakaa kolmekerroksiseen malliin, jossa jokainen näistä pilvipalvelumalleista vastaa eri abstraktiotasoa. Jokainen näistä abstraktiotasoista piilottaa ja automatisoi asiakkaan tarpeisiin nähden tarpeettomat osa-alueet ja toiminnot vähentäen järjestelmän rakentamiseen ja käyttöönottoon tarvittavia resursseja. Tämän ansiosta asiakas voi keskittyä tehokkaammin ja käyttää enemmän resursseja omien liiketoiminnallisten ongelmiensa ratkaisemiseen kuin infrastruktuurin hallinnoimiseen. (Kavis 2014.)

Kyseiset kolme palvelumallia ovat pienimmästä abstraktiotasosta suurimpaan IaaS (Infrastructure as a Service), PaaS (Platform as a Service) ja SaaS (Software as a Service). Kuviossa 1 on esiteltynä kyseinen kolmekerroksinen malli.



Kuvio 1. Palvelumallit (Hill, Hirsch, Lake & Moshiri 2013)

Vastuu pilvipalveluna toteutetun järjestelmän eri osa-alueista jakautuu eri tavalla riippuen käytetystä palvelumallista. Kuviossa 2 on esiteltynä kummankin osapuolen, asiakkaan ja palveluntarjoajan, vastuut järjestelmän eri osa-alueista kullakin palvelumallin kerroksella.



Kuvio 2. Vastuut palvelumalleissa (Kavis 2014)

2.2.2 IaaS

Fyysisesti pilvipalveluntarjoajien laitteisto koostuu joukosta palvelimia ja verkkoja palvelinkeskuksissa ympäri maailmaa. Palveluntarjoaja on vastuussa kaiken tämän infrastruktuurin hallinnoimisesta, ja se on abstraktoitu virtualisoinnin avulla piiloon asiakkaalta. Alimpana asiakkaalle tarjottavana abstraktiotasona tämän virtualisoinnin päällä on IaaS eli infrastruktuuri palveluna, joka nimensä mukaisesti tarjoaa asiakkaalle pääsyn käsiksi virtualisoituun infrastruktuuriin eli virtuaalilaitteisiin. Näiden virtualisoitujen komponenttien avulla asiakas voi rakentaa itsellensä sopivan skaalautuvan ja kustannustehokkaan alustan IT-ratkaisuilleen ulkoistaen kaiken taustalla olevan laitteiston ja infrastruktuurin hallinnoimisen palveluntarjoajalle. (What is IaaS? n.d.)

Kun fyysisen infrastruktuurin hallinnoiminen on ulkoistettu palveluntarjoajalle, tarjoavat he puolestaan kyseisen infrastruktuurin päälle rakennettuja virtuaalisia komponentteja palveluna asiakkaalle. Asiakas pääsee käsiksi kyseisiin palveluihin joko web-pohjaisen hallintakonsolin tai ohjelmointirajapinnan (API, Application Programming Interface) kautta. Asiakkaalle tarjottaviin palveluihin lukeutuvat esimerkiksi virtuaalikoneet, joissa asiakas voi ajaa haluamiansa käyttöjärjestelmiä ja sovelluksia, datan säilytystila ja verkkolaitteet kuten kuormantasaajat ja palomuurit. Laajimmin käytetty ja kehittynein IaaS-palveluntarjoaja on Amazon Web Services (AWS). Muita mainittavia palveluntarjoajia ovat esimerkiksi Rackspace ja GoGrid. (Kavis 2014.)

2.2.3 PaaS

PaaS eli alusta palveluna tarjoaa nimensä mukaisesti alustan tai ympäristön, joka mahdollistaa sovelluskehittäjien rakentaa sovelluksensa ja palvelunsa verkon yli. Pilvessä tarjottaviin PaaS-palveluihin pääsee yleensä käsiksi web-selaimella. Palveluntarjoaja tarjoaa asiakkaalle työkalut sovellusten rakentamiseen ja ylläpitoon. Palvelut koostuvat valmiiksi konfiguroiduista toiminnoista, joita asiakas voi mielensä mukaan ottaa käyttöön. Asiakas, esimerkiksi web-kehittäjä, voi käyttää PaaS-ympäristöä sovelluskehityksen jokaisella osa-alueella kehittämisestä testaamiseen ja käyttöönottoon välittämättä sen alla olevasta infrastruktuurista. (What is PaaS? n.d.)

PaaS on palvelumallissa yhtä abstraktiotasoa korkeammalla kuin IaaS. PaaS-palvelumallin tapauksessa unohdetaan sovelluksen alla olevasta infrastruktuurista huolehtiminen, ulkoistetaan se palveluntarjoajalle ja keskitytään täysin sovelluksen kehittämiseen. Monet PaaS-ratkaisut tarjoavat toimintoja kuten välimuistiin tallentamisen ja tietokannan skaalaamisen palveluna, jotta sovelluksen kehittäjän ei tarvitse ottaa näihin kantaa ja hän voi keskittyä täysin business-logiikan kehittämiseen. (Kavis 2014.)

IaaS-palvelumallin tapauksessa puhuttiin infrastruktuurikomponenteista kuten virtuaalipalvelimista ja niitä yhdistävistä tietoverkoista. PaaS-palvelumallin tapauksessa näiden sijasta keskitytään sovelluskehityksen ohjelmointikieliin, kirjastoihin, palveluihin ja työkaluihin. Esimerkkejä PaaS-palveluntarjoajista ovat Google App Engine, Microsoft Azure ja Heroku. (Mt.)

2.2.4 SaaS

SaaS eli sovellus palveluna tarkoittaa sellaista pilvipalvelua, missä asiakas pääsee käsiksi sovelluksiin verkon välityksellä. Google, Twitter ja Facebook ovat esimerkkejä tällaisista sovelluksista, joita käyttäjä voi käyttää millä tahansa Internet-yhteyden mahdollistavalla laitteella. Yrityskäytössä esimerkkejä tällaisista sovelluksista ovat esimerkiksi kirjanpito- ja laskutussovellukset sekä webmail sähköpostien lukemiseen ja lähettämiseen. SaaS-sovelluksia voidaan myös ajatella tarpeen vaatiessa käytössä olevina sovelluksina, eli toisin sanoen vuokrataan sovelluksia niiden ostamisen sijaan. Sen sijaan, että ostaisit sovelluksen etukäteen, asentaisit sen tietokoneellesi ja tallentaisit siihen liittyviä tiedostoja omalle kiintolevyllesi, vuokrataan sovellus esimerkiksi kuukaudeksi, käytetään sitä verkon välityksellä ja sen tuottamat tiedostot tallennetaan pilveen. (What is SaaS? n.d.)

SaaS on kolmikerroksisen palvelumallin huipulla, ja sillä tarkoitetaan kokonaista valmista sovellusta, jota tarjotaan palveluna asiakkaalle. Tällä tasolla kaikki sovellusta ylläpitävä infrastruktuuri on ulkoistettu palveluntarjoajalle, ja asiakkaan hoidettaviksi jäävät ainoastaan mahdolliset sovelluskohtaiset asetukset ja käyttäjien hallinta. SaaS-ratkaisut ovat yleisiä yritysten keskuudessa, kun halutaan ulkoistaa jokin yrityksen ydinliiketoimintaan kuulumaton osa-alue. Tällöin yritys voi jättää palkkaamatta henkilökuntaa järjestelmän ylläpitämiseen ja sen sijaan käyttää selainpohjaista palvelua Internetin välityksellä. (Kavis 2014.)

2.2.5 Vertailu ja valinta

Oikean palvelumallin valinnassa tulisi ottaa huomioon useita tekijöitä ja tarkastella kutakin mallia esimerkiksi teknisestä, taloudellisesta, strategisesta, organisaatiollisesta ja riskien näkökulmasta. SaaS-palvelumallissa palveluntarjoaja hallitsee kaiken sovelluksen perustana olevan infrastruktuurin ja asiakkaalle tarjotaan vain valmis sovellus palveluna Internet-yhteyden välityksellä, jota asiakas voi käyttää joko selaimella tai hyödyntäen ohjelmointirajapintaa. Hyvä käyttökohde SaaS-palveluille on yrityksen ydinosaamisen ulkopuolella olevien sovellusten ulkoistaminen. Yleisiä esimerkkejä tällaisista sovelluksista ovat henkilöstö-, palkanlaskenta-, asiakkuudenhallinta-, kirjanpito- ja toiminnanohjausjärjestelmät. Yrityksen ei kannata ostaa tällaisia sovelluksia tai palvelimia ja maksaa näiden ylläpidosta, jos vastaavan SaaS-palvelun

käyttö on kustannustehokkaampaa. SaaS-palvelut eivät kuitenkaan välttämättä tarjoa kaikkia asiakkaan vaatimia toimintoja tai asetuksia, mutta ennen oman sovelluksen rakentamista kannattaa ottaa huomioon kaikki palveluntarjoajan hoitamat osa-alueet turvallisuudesta palvelinsalien infrastruktuuriin ja päivityksistä tietokantoihin ja varmuuskopiointiin. (Kavis 2014.)

PaaS-ratkaisut eivät enää pakota asiakasta mihinkään tiettyyn ohjelmointikieleen tai alustaan, vaan ne tarjoavat mahdollisuuden valita useiden nykyaikaisten vaihtoehtojen väliltä. PaaS-palvelumallin suurimpia etuja on se, että sovellusalustan perustana oleva infrastruktuuri on abstraktoitu kehittäjältä piiloon, jolloin hän voi keskittyä täysin liiketoiminnallisten ongelmien ratkaisemiseen palveluntarjoajan hoitaessa kyseisen alustan mahdollistavan infrastruktuurin ylläpitämisen. PaaS-palveluntarjoajat asettavat rajoituksia asiakkaalle tarjottavien resurssien määrälle, mutta suurimmalle osalle sovelluksista tämä ei ole ongelma, ja vain kaikista suurimmat sovellukset eivät saa PaaS-ratkaisuista haluamaansa suorituskykyä ja skaalautuvuutta. (Mt.)

Jos sovelluksella on sellaisia vaatimuksia suorituskyvyn tai skaalautuvuuden suhteen, jotka vaativat muistinhallintaa, tietokannan tai sovelluspalvelimen konfigurointia, datan sijainnin hallintaa, muutoksia käyttöjärjestelmään tai muuta vastaavaa, kannattaa miettiä IaaS-ratkaisujen hyödyntämistä. Jos tämän kaltaisista asioista huolehtiminen ei ole oleellista, ovat PaaS-ratkaisut luultavasti oikea valinta. (Mt.)

PaaS-ratkaisujen hyödyntäminen voi laskea kustannuksia sovelluksen rakentamiseen tarvittavia resursseja vähentämällä, mutta datamäärien ja suorituskykyvaatimusten kasvaessa ja IaaS-palveluiden halventuessa tämä ei välttämättä ole merkittävä tekijä näiden kahden väliltä valittaessa (Mt).

IaaS-ratkaisujen hyviin puoliin kuuluu palvelukatkosten riskin vähentäminen. PaaS- ja SaaS-ratkaisujen tapauksessa palvelukatkoksen sattuessa asiakas ei voi kuin odottaa, että palveluntarjoaja korjaa vian ja palauttaa palvelut toimintakuntoon. IaaS-ratkaisujen tapauksessa asiakkaalla on mahdollisuus rakentaa useiden palvelinsalien kattavia vikasietoisia palveluita, mikä mahdollistaa yhden palveluntarjoajan palvelukatkoksesta selviämisen ilman käyttäjille näkyvää vaikutusta. (Mt.)

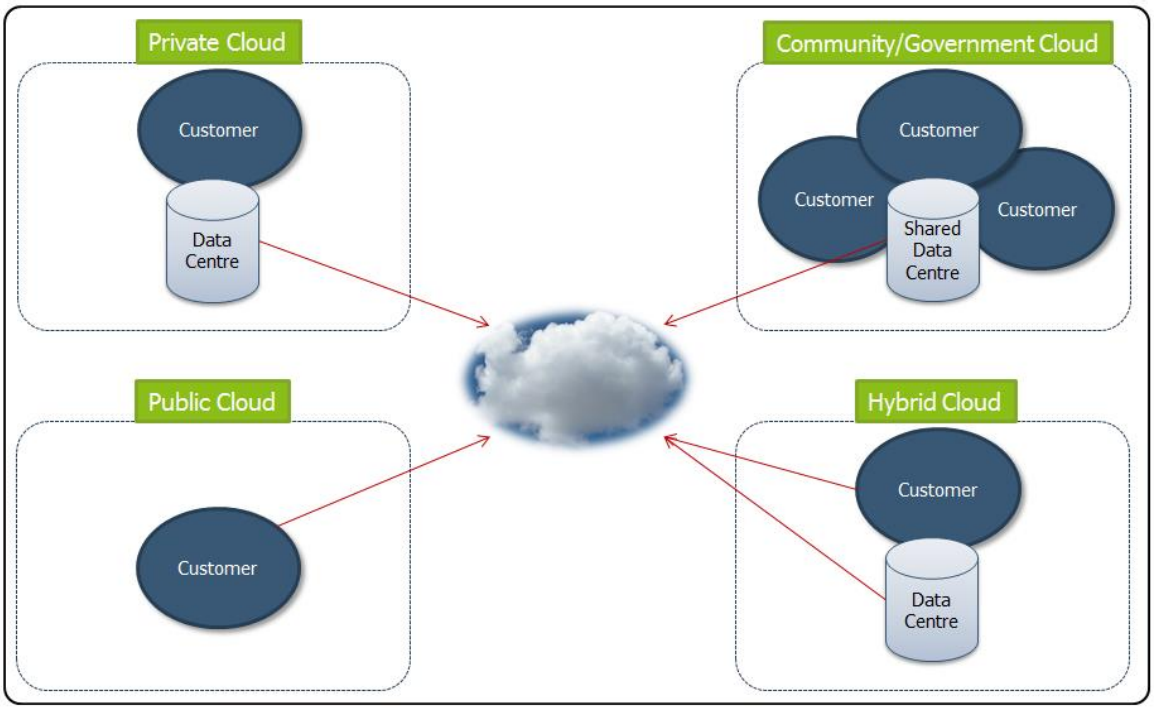
Liikuttaessa abstraktiotasolla ylöspäin kohti SaaS-palveluita tuotteen tuominen markkinoille nopeutuu, ja henkilöstöresurssien tarve ja ylläpitokustannukset vähenevät.

Vastaavasti alaspäin kohti IaaS-palveluita liikuttaessa alla olevan infrastruktuurin hallintamahdollisuudet kasvavat ja mahdollisuudet palvelukatkosten välttämiseen tai niistä toipumiseen paranevat. (Mt.)

2.3 Käyttöönottomallit

2.3.1 Yleistä

Pilvipalveluiden käyttöönottomalleilla tarkoitetaan tietyntyyppisiä pilviympäristöjä, jotka voidaan erotella toisistaan vastaamalla kysymyksiin kuten kuka omistaa pilven, minkä kokoinen se on ja kuka pilveä pääsee käyttämään. Käyttöönottomallit voidaan karkeasti jakaa kahteen kategoriaan: julkisiin pilviin ja yksityisiin pilviin. Lisäksi on olemassa näitä kahta yhdistäviä hybridipilviä ja muita edellä mainittujen mallien erikoistapauksia. Kuviossa 3 on esiteltynä pilvipalveluiden käyttöönottomallit ja niiden yhteydet toisiinsa. (Cloud Deployment Models n.d.)



Kuvio 3. Käyttöönottomallit (Vold 2012)

2.3.2 Julkinen pilvi

Julkisella pilvellä tarkoitetaan kolmannen osapuolen palveluntarjoajan omistamaa julkisesti avointa pilviympäristöä. Julkisen pilven resursseja tarjotaan maksavalle asi-

akkaalle aikaisemmin mainittujen palvelumallien mukaisesti. Palveluntarjoaja on täysin vastuussa julkisen pilven ja sen resurssien luomisesta ja ylläpitämisestä. (Public Clouds n.d.)

Julkisen pilven infrastruktuuri sijaitsee täysin palveluntarjoajan tiloissa, ja sitä tarjotaan avoimesti julkiseen käyttöön. Julkinen pilvi on usean samaa palveluntarjoajaa käyttävän asiakkaan jakama ympäristö, jossa asiakkaat maksavat palveluntarjoajalle kyseisten jaettujen resurssien käyttämisestä. Käyttäjillä ei ole mitään tietoa siitä, missä sovellus fyysisesti sijaitsee, ainakaan tarkemmin kuin tietyn palvelinkeskuksen tarkkuudella. Palveluntarjoaja rakentaa fyysisen laitteiston päälle abstraktiokerroksen ja paljastaa käyttäjälle ainoastaan rajapinnan, jonka avulla käyttäjä voi luoda virtuaalisia resursseja useiden käyttäjien jakamasta resurssivarastosta. (Kavis 2014.)

2.3.3 Yksityinen pilvi

Yksityisellä pilvellä tarkoitetaan yhden organisaation omistamaa pilviympäristöä, joka mahdollistaa organisaation eri osastojen tai eri sijainneista pääsyn käsiksi keskitettyihin resursseihin. Varsinaisen pilviympäristön ylläpitämisen voi hoitaa joko organisaation sisäinen tai ulkoistettu henkilökunta. Yksityisen pilven tapauksessa sama organisaatio on käytännössä sekä asiakas että palveluntarjoaja. Näiden roolien erottamiseksi yleensä jokin erillinen osasto organisaatiossa ottaa palveluntarjoajan roolin, ja vastaavasti yksityisen pilven palveluita tarvitsevat osastot omaksuvat asiakkaan roolin. Yksityisen pilven tapauksessa on hyvä erottaa toisistaan termit ”paikallinen” ja ”pilvipohjainen”. Vaikka yksityinen pilvi sijaitsisikin fyysisesti organisaation tiloissa, sen tarjoamat resurssit ovat silti pilvipohjaisia, jos ne ovat asiakkaan etäkäytettävissä. Asiakkaana toimivien osastojen tarjoamia yksityisen pilven ulkopuolella sijaitsevia resursseja taas pidetään paikallisina verrattuna yksityisiin pilvipohjaisiin resursseihin. (Private Clouds n.d.)

Vaikka yksityisen pilven infrastruktuuri onkin varattu ainoastaan yhden organisaation käyttöä varten, sen voi omistaa ja sitä voi ylläpitää itse organisaation sijasta myös kolmannen osapuolen palveluntarjoaja tai jokin näiden kahden yhdistelmä. Fyysisesti infrastruktuuri voi sijaita organisaation tiloissa tai niiden ulkopuolella kolmannen osapuolen palveluntarjoajan palvelinsalissa. Jos yksityinen pilvi sijaitsee fyysisesti organi-

saation omissa tiloissa, ovat he täysin vastuussa kaikesta, palvelinsalin ylläpitämisestä laitteiden hankintaan ja konfigurointiin. Jos taas yksityisen pilven ulkoistaa kolmannen osapuolen palveluntarjoajalle, on organisaatio riippuvainen palveluntarjoajan tarjoamasta infrastruktuurista. Kummassakin tapauksessa on kuitenkin kyse yhden asiakkaan ympäristöstä, eikä resursseja jaeta toisten asiakkaiden kanssa. (Kavis 2014.)

2.3.4 Hybridipilvi

Hybridipilvellä tarkoitetaan pilviympäristöä, joka koostuu kahdesta tai useammasta eri käyttöönottomallista. Asiakas voi esimerkiksi valita yksityisen pilven arkaluontoista dataa käsitteleviä pilvipalveluita varten ja julkisen pilven muita vähemmän arkaluontoisia palveluita varten. Hybridipilven arkkitehtuuri voi olla monimutkainen ja haastava luoda ja ylläpitää pilviympäristöjen mahdollisten eroavaisuuksien takia. Haasteena on myös pilven ylläpitovastuun jakaminen. Ylläpitovastuut jaetaan yleensä niin, että organisaatio itse hoitaa yksityisen pilven ylläpidon, ja kolmannen osapuolen palveluntarjoaja hoitaa vastaavasti julkisen pilven ylläpidon. (Hybrid Clouds n.d.)

Organisaatio voi käyttää hybridipilveä hyödyntääkseen sekä julkisen että yksityisen pilven parhaita puolia. Parhaana toimintatapana hybridipilven tapauksessa pidetään julkisen ja yksityisen pilven yhdistelmää, jossa julkista pilveä käytetään niin paljon kuin mahdollista, ja yksityistä pilveä käytetään vain kaikista riskialttiimmissa tapauksissa. Näin ollen hybridipilvessä hyödynnetään mahdollisimman paljon kaikkia julkisen pilven tuomia etuja, ja yksityistä pilveä käytetään vain silloin, kun takeet yksityisyydestä, turvallisuudesta tai datan omistajuudesta eivät ole riittäviä julkisessa pilvessä. (Kavis 2014.)

2.3.5 Muita käyttöönottomalleja

Yhteisöpilvi

Yhteisöpilvi (community cloud) vastaa muilta osin julkista pilveä, mutta sen käyttö on rajattu vain tiettyyn asiakasyhteisöön kuuluvia käyttäjiä varten. Yhteisöpilvi voi olla joko yhteisön jäsenten yhteisesti omistama tai kolmannen osapuolen palveluntarjoajan tarjoama rajatun käyttöoikeuden julkinen pilvi. Yhteisöpilven jäsenet jakavat

vastuun pilven määrittelemisestä ja kehittämisestä, mutta jäsenyys ei välttämättä takaa käyttöoikeutta kaikkiin pilven resursseihin. (Community Clouds n.d.)

Virtuaalinen yksityinen pilvi

Virtuaalinen yksityinen pilvi (VPC, Virtual Private Cloud), josta käytetään myös termejä ”dedicated cloud” ja ”hosted cloud”, tarkoittaa itsenäistä tai eristettyä asiakkaalle tarjottavaa julkisen pilvipalveluntarjoajan ylläpitämää pilviympäristöä (Other Deployment Models n.d).

Nimensä mukaisesti VPC toimii yksityisenä pilvenä ja täyttää asiakkaalle tarjottavien virtuaalisten resurssien puolesta samat vaatimukset kuin fyysisesti yksityinen pilvikin, mutta infrastruktuurin eristäminen vain tehdään virtuaalisesti. Tämä mahdollistaa asiakkaalle esimerkiksi virtuaalisen verkkotopologian, IP-osoitteiden (Internet Protocol), reitityksen ja palomuurisääntöjen hallitsemisen. Tyypillisesti VPC-ratkaisuja käytetään organisaation paikallisten resurssien ja palveluntarjoajan tarjoamien pilviresurssien yhdistämiseen käyttäen virtuaalisia erillisverkkoja (VPN, Virtual Private Network). (Furht & Escalante 2010.)

Pilvien yhdistymä

Termeillä InterCloud ja ”cloud federation” tarkoitetaan yleisesti eri palveluntarjoajien hallitsemien pilvien yhdistymää eli niin sanottua pilvien pilveä. Tarkemmin ottaen termillä federaatio viitataan tiettyjen pilvipalveluntarjoajien välisiin yksityisiin sopimuksiin, jotka mahdollistavat toisten palveluntarjoajien resurssien ja palvelujen hyödyntämisen luottamuksellisesti käyttäen yksityisiä rajapintoja. InterCloud-mallin taustalla taas on ajatus maailmanlaajuisesta yhteistyöstä pilvipalveluntarjoajien välillä käyttäen avoimia standardeja. Näin ollen luodaan avoin alusta, jossa sovellusten rakentamiseen voidaan vapaasti käyttää eri palveluntarjoajien tarjoamia palveluja. (Buyya, Vecchiola & Selvi 2013.)

2.4 Käsitteitä

2.4.1 Virtualisointi

Virtualisoinnilla tarkoitetaan fyysisten resurssien simulointia abstraktien eli virtuaalisten resurssien avulla. Virtualisoinnin avulla virtuaalisten resurssien alla oleva fyysinen laitteisto abstraktoidaan piiloon, mikä mahdollistaa dynaamisten ympäristöjen luonnin fyysisen laitteiston päälle. Virtualisointi mahdollistaa pilvipalveluille tärkeitä toimintoja kuten resurssien yhteen keräämisen sekä niiden nopean ja elastisen provisioinnin. Tyypillisissä pilviympäristöissä lähes kaikki mahdollinen on virtualisoitu, palvelinkoneista tallennustilaan ja verkkolaitteisiin. (Hill, Hirsch, Lake & Moshiri 2013.)

Pilvipalveluiden yhteydessä virtuaalikoneella tarkoitetaan ohjelmistollisesti toteutettua itsenäisellä käyttöjärjestelmällä varustettua konetta, joka suorittaa ohjelmia fyysisen koneen tapaan. Virtuaalikoneita voidaan kopioida, siirtää ja tuhota tarpeen mukaan, ja niistä voidaan tallentaa niin sanottuja tilannekuvia, minkä avulla käyttäjä voi palauttaa koneen tallennushetkellä olleeseen tilaan. (Mt.)

Virtualisointi tuo useita hyötyjä palvelinkeskuksiin. Sen ansiosta resurssien käyttö on tehokkaampaa, mikä johtaa virrankäytön, jäähdytyksen ja tilan tarpeen vähenemiseen. Etuna on myös se, että suojattuja täysin erillisiä ympäristöjä voidaan varmuuskopioida, mikä johtaa parempaan joustavuuteen ja saatavuuteen. Yhdessä virtuaalikoneessa tapahtuvien vikojen ei pitäisi vaikuttaa muiden virtuaalikoneiden toimintaan. Sovellukset voivat siirtyä palvelimesta toiseen ilman käyttökatkoksia, ja virtuaalikoneita voidaan käynnistää automaattisesti toisilla palvelimilla. (Mt.)

2.4.2 Autonomiset järjestelmät

Autonomisilla järjestelmillä pyritään tietoteknisten järjestelmien kehittämiseen vähentämällä ihmisten osallisuutta niiden toiminnassa. Tämän tavoitteena on tuottaa järjestelmiä, jotka hallitsevat itse itseään ja mukautuvat ennalta-arvaamattomiin muutoksiin. Autonomisten järjestelmien perusominaisuuksia ovat itsensä konfigurointi, optimointi, korjaus ja suojaus. Näiden ominaisuuksien tuomat hyödyt ovat merkittäviä suurissa palvelinsaleissa, joissa koneviat ovat säännöllisiä tapahtumia. Autonomisissa järjestelmissä käytetään monitorointia ja mittareita erilaisten havaintojen tekemiseen. Näiden havaintojen seurauksena tehdään päätöksiä ja pannaan ne

käytäntöön ennalta määrättyjen käytänteiden perusteella. (Hill, Hirsch, Lake & Moshiri 2013.)

2.4.3 Elastisuus

Pilvipalvelut luovat kuvitelman rajattomasta määrästä tarpeen vaatiessa käytössä olevista resursseista. Toisin sanoen käyttäjät olettavat, että milloin tahansa tarpeen vaatiessa heidän käytössään on tarvittava määrä resursseja. Lisäksi oletuksena on myös automaattinen skaalautuminen eli resurssien automaattinen provisiointi tai vapauttaminen kuorman kasvaessa tai vähentyessä. Tällaista kykyä mukautua automaattisesti kysynnän muutoksiin palveluntarjoajan toimesta kutsutaan elastisuudeksi, joka on yksi suurimmista pilvipalveluiden tuomista hyödyistä. (Buyya, Broberg & Goscinski 2011.)

2.4.4 Tietoturva

Turvallisuusalueiden ja niiden välisten rajojen määrittely on oleellinen osa pilvipalveluiden tietoturvaa. Alueiden välisellä rajalla määritellään organisaatiolle kuuluva luotettu sisäpuoli ja ei-luotettu julkinen ulkopuoli. Perinteisessä mallissa alueiden väliselle rajalle asetetaan palomuuri suojaamaan sisäpuolella sijaitsevia organisaation resursseja ulkopuolelta tulevilta uhilta. Pilvipalveluiden yhteydessä kyseiset rajat on määriteltävä uudestaan riippuen käytössä olevasta pilviarkkitehtuurista. Keskeiset turvallisuushuolet pysyvät samoina kuin perinteisessä mallissakin niiden siirtyessä pilveen kaiken muun toiminnan mukana. Suurimpana haasteena on määritellä ja toteuttaa kyseiset konseptit uudessa pilviympäristössä. (Yeluri & Castro-Leon 2014.)

Käsitykset pilvipalveluiden turvallisuudesta jakautuvat kahteen ääripähän. Toisessa ääripäässä on ajatus siitä, että pilvipalvelut ovat erittäin epäturvallisia eikä julkiseen pilveen tulisi missään tapauksessa sijoittaa minkäänlaista dataa. Toisessa ääripäässä on vastaavasti virheellinen ajatus siitä, että palveluntarjoaja hoitaisi kaiken mahdollisen tietoturvan asiakkaan puolesta. Totuus pilvipalveluiden tietoturvasta on kuitenkin se, että kunnollisen tietoturva-arkkitehtuurin avulla julkinen pilvi voi olla turvallisempi kuin asiakkaan omissa tiloissa sijaitseva palvelinkeskus. Vaikka pilvipalveluntarjoajat voivatkin olla houkuttelevia kohteita kyber-rikollisille, koska ne tarjoavat resursseja suurelle määrälle yrityksiä, tarjoavat heidän palvelinkeskuksensa korkean

paikallisen tietoturvan tason. Näin ollen asiakkaan vastuulle jää itse sovelluksen asianmukainen tietoturvan suunnittelu ja toteutus käyttäen palveluntarjoajan tarjoamia resursseja ja ohjeita. (Kavis 2014.)

2.5 Esimerkkejä pilvipalveluista

Koska yhtenä opinnäytetyön tavoitteena on tutkia mahdollisuuksia pilvipalveluiden hyödyntämiseen, ja Amazon Web Services (AWS) on yksi kehittyneimmistä ja monipuolisimmista palveluntarjoajista, käytetään sen tarjoamia palveluja esimerkkeinä pilvipalveluiden tarjoamien mahdollisuuksien tutkimiseen.

2.5.1 Laskenta

Amazon EC2 (Elastic Compute Cloud) on elastista laskentakapasiteettia pilvessä tarjoava web-palvelu. Käytännössä tällä tarkoitetaan pilvessä sijaitsevia virtuaalisia palvelininstansseja. Web-käyttöliittymän ja ohjelmointirajapinnan kautta hallittavissa oleva palvelu mahdollistaa nopeasti ja automaattisesti käyttöasteen mukaan skaalautuvien ja sen perusteella laskutettavien vikasietoisten järjestelmien suunnittelun. Käyttäjä voi vapaasti valita käytettävän instanssin tyypin ja käyttöjärjestelmän sekä muistin, prosessorien ja säilytystilan määrän. Käyttäjälle annetaan pääkäyttäjän käyttöoikeudet palvelininstanssiin, joten sitä voidaan hallita kuten mitä tahansa muutakin palvelinkonetta. EC2 on suunniteltu toimimaan hyvin yhteen muiden AWS-palveluiden, esimerkiksi VPC-palvelun kanssa. (Mathew 2014.)

Amazon VPC (Virtual Private Cloud) tarjoaa tietoturvaa ja tietoliikennetoimintoja muille AWS-palveluille. Käytännössä se toimii virtuaalisena tietoverkkoympäristönä muille AWS-palveluille, mihin esimerkiksi EC2-palvelininstanssit sijoitetaan. Asiakas voi esimerkiksi luoda aliverkkoja, määrittää haluamansa IP-osoitealueen ympäristölle ja valita mitkä palvelininstanssit ovat Internet-yhteydellä saavutettavissa ja mitkä pysyvät yksityisinä sekä konfiguroida reititystauluja ja yhdyskäytäviä. Tietoturvan osalta lähtevää ja tulevaa tietoliikennettä voidaan hallita turvallisuusryhmien ja ACL-pääsylistojen (Access Control List) avulla. Lisäksi jo olemassa olevia paikallisia resursseja voidaan yhdistää VPC-ympäristöön käyttäen salattuja IPsec (Internet Protocol Security) VPN -yhteyksiä. (Mt.)

Automaattisen skaalautuvuuden avuksi AWS tarjoaa Lambda-palvelun, jonka avulla asiakas voi määrittää haluamansa koodin ajettavaksi vastauksena johonkin tapahtumaan, mahdollistaen näin automaattisen resurssienhallinnan. Elastisuutta tukee myös ELB-palvelu (Elastic Load Balancing) eli elastinen kuormantasaus, jonka avulla saapuvan liikenteen kuorma voidaan jakaa automaattisesti usealle EC2-instanssille, mikä parantaa järjestelmän vikasietoisuutta ja vähentää ylikuormituksen riskiä optimoimalla resurssien käyttöä. (Mt.)

2.5.2 Tietovarastot

Amazon S3 (Simple Storage Service) tarjoaa yksinkertaisen rajapinnan datan varastoimiseen. Tarkoituksena on tarjota kaikkialla ja milloin tahansa saatavilla olevaa säilytystilaa mille määrälle ja minkä tyyppiselle datalle tahansa. S3 tallentaa datan objekteina sangoiksi (bucket) nimettyihin resurssivarastoihin. Yksi sanko voi sisältää rajattoman määrän objekteja, ja yksi objekti voi olla jopa viiden teratavun kokoinen. Käyttäjä voi hallita sankojen käyttöoikeuksia ja tarkastella niihin liittyviä lokitiedostoja. Palvelua voidaan käyttää joko itsekseen tai yhteydessä muiden AWS-palveluiden kanssa esimerkiksi varmuuskopiointiin tai verkkosivustojen staattisen sisällön säilyttämiseen. S3 tarjoaa lisäksi toimintoja kuten versioinnin, salauksen ja haluamansa sijainnin valitsemisen. (Mathew 2014.)

Erityisesti varmuuskopiointia ja muuta pitkäaikaista datan säilytystä varten Amazon tarjoaa Glacier-nimisen palvelun. Koska palvelu on optimoitu vain harvoin tarvittavalle datalle, mahdollistaa se erittäin matalat kustannukset asiakkaalle. Amazon EBS (Elastic Block Store) tarjoaa matalaviiveistä ja nopeasti skaalautuvaa lohkotason levytilaa EC2-instanssien käytettäväksi. Jokainen EBS-levytila replikoidaan automaattisesti, mikä parantaa järjestelmän saatavuutta ja vikasietoisuutta. (Mt.)

AWS Storage Gateway tarjoaa nopean ja tietoturvallisen tavan yrityksen paikallisen IT-ympäristön ja AWS-tietovarastopalveluiden yhdistämiseen. Amazon CloudFront on muiden palveluiden yhteydessä käytetty palvelu sisällön toimittamiseen. Sitä voidaan käyttää esimerkiksi kokonaisen verkkosivuston sisällön toimittamiseen, käyttäen hyväksi sijainteja eri puolilla maailmaa. Käytännössä sisältöä varten tehdyt pyynnöt reititetään lähettäjää lähimpänä olevaan sijaintiin, mikä parantaa suorituskykyä ympäri maailmaa. (Mt.)

2.5.3 Tietokannat

Amazon RDS (Relational Database Service) tarjoaa skaalautuvan relaatiotietokannan palveluna. Muista relaatiotietokantajärjestelmistä tuttujen toimintojen lisäksi se mahdollistaa esimerkiksi automaattiset päivitykset, varmuuskopioinnin ja synkronisesti replikoituvat usean tietokannan vikasietoiset korkean saatavuuden järjestelmät. Amazon DynamoDB on vastaavasti nopea NoSQL-tietokantapalvelu, joka tarjoaa dokumentti- ja avain-arvoparipohjaiset tietorakenteet asiakkaan käyttöön. Amazon Redshift on tarkoitettu erittäin suurien tietomäärien varastointiin, missä data tallennetaan sarakemuotoisena useiden solmujen klustereihin, ja kyselyt ajetaan rinnakkaisina suorituskyvyn parantamiseksi. Amazon ElastiCache on muistinsisäistä välimuistia tarjoava palvelu, joka tarjoaa asiakkaan valittavaksi kaksi suosittua moottoria: Memcached ja Redis. Sen tarkoituksena on parantaa web-sovellusten suorituskykyä mahdollistamalla datan hakeminen nopeasta välimuistista hitaampien kiintolevypohjaisten tietokantojen sijaan. Tämän lisäksi palvelu havaitsee ja korvaa vikaantuneet solmut automaattisesti, parantaen suorituskykyä ja vikasietoisuutta. (Mathew 2014.)

2.5.4 Tietoliikenne

AWS Direct Connect mahdollistaa dedikoidun yksityisen yhteyden muodostamisen esimerkiksi yrityksen paikallisen verkkoympäristön ja AWS-palveluiden välille. Yhteys voidaan jakaa useaan virtuaaliseen rajapintaan käyttäen IEEE 802.1Q -standardissa määriteltyjä virtuaalisia lähiverkkoja. Tämä mahdollistaa saman yhteyden käyttämisen sekä julkisten että yksityisten resurssien hyödyntämiseen, verkon silti ollessa loogisesti eroteltuna julkiseen ja yksityiseen alueeseen. Amazon Route 53 on korkean saatavuuden skaalautuva DNS-palvelu (Domain Name System). Tavallisten DNS-toimintojen lisäksi se voidaan esimerkiksi konfiguroida tekemään päätepisteiden toiminnan monitorointiin perustuvaa reititystä. Myös muita DNS-pohjaisia reititys- ja vikasietoisuustoimintoja on saatavilla. (Mathew 2014.)

2.5.5 Analytiikka

Amazon EMR (Elastic MapReduce) on palvelu erittäin suurten datamäärien nopeaan prosessointiin. Se käyttää Hadoop-sovelluskehystä datan hajauttamiseen ja prosessointiin skaalautuvassa EC2-instanssien klusterissa. Amazon Kinesis on palvelu reaaliaikaisten datavirtojen prosessointiin. Se voi kerätä ja prosessoida satoja teratavuja

dataa sadoista tuhansista lähteistä tunnissa, mahdollistaen monenlaisten reaaliaikaisten sovellusten toteuttamisen. AWS Data Pipeline on palvelu, jota käytetään datan luotettavaan siirtämiseen ja prosessointiin määräajoin eri AWS-palveluiden ja paikallisten datalähteiden välillä. Sitä voidaan käyttää esimerkiksi tietyin väliajoin tapahtuvaan datan prosessointiin, missä varastoidulle datalle suoritetaan joitakin toimintoja, minkä jälkeen sen tuottamat tulokset tallennetaan johonkin AWS-palveluun. (Mathew 2014.)

2.5.6 Muut palvelut

Edellä mainittujen oleellisimpien palveluiden lisäksi AWS tarjoaa vielä lukuisan määrän erilaisia palveluita. Sovelluspalveluiden puolelta tarjotaan palveluita viestijonojen, moniosaisten taustaprosessien, datavirtojen, median transkoodauksen, sähköpostien ja hakupalveluiden hallitsemiseen. Web-sovellusten käyttöönottoon ja hallintaan liittyviä palveluita tarjotaan eri ohjelmointikielille ja palvelinalustoille, arkkitehtuurien määrittämiseen ja valmiiden asennuspakettien kokoamiseen, automaattiseen lähdekoodin käyttöönottoon ja versionhallintaan sekä muuhun automaatioon ja jatkuvaan integraatioon. Hallinnan ja turvallisuuden kannalta palveluita on tarjolla käyttäjien autentikaation, pääsyn, roolien ja oikeuksien hallintaan, salauksessa käytettävien avainten hallintaan, lokitietojen keräämiseen, AWS-palveluiden ja sovellusten toiminnan monitorointiin, konfiguraationhallintaan sekä muihin turvallisuuden ja palveluiden hallintaan. Lopuksi saatavilla on myös muutamia mobiilipalveluita ja yrityskäyttöön tarkoitettuja sovelluksia palveluina. (Mathew 2014.)

2.6 Palvelinfarmit

Palvelinfarmilla tarkoitetaan yksinkertaisesti fyysisesti samassa paikassa sijaitsevaa joukkoa palvelinkoneita, jotka toimivat yhdessä tarjotakseen jotakin palvelua. Joskus tästä käytetään myös termiä ”klusteri”, vaikka tarkemmin ottaen sillä tarkoitetaan lähinnä redundanttisuutta ja korkeaa saatavuutta varten yhdistettyjä palvelimia, jotka näkyvät käyttäjälle päin yhtenä palvelimena. Pilvipalveluissa yhdistyvät fyysisen palvelinfarmin ja virtuaalipalvelimien parhaimmat puolet. Tämä luo redundanttisuutta, korkeaa saatavuutta ja optimoitua resurssien käyttöä tarjoavan ympäristön. Palvelinfarmit voidaan jakaa kolmeen eri ryhmään: Internet-, extranet- ja intranet-palvelinfarmeihin (Rouse & Gibilisco 2013.)

2.6.1 Internet-palvelinfarmit

Internet-palvelinfarmeilla tarkoitetaan nimensä mukaisesti Internetin välityksellä käytettävissä olevia palvelinfarmeja. Tyypillisesti näillä mahdollistetaan yritysten tarjoamien palvelujen saatavuus kaikille Internet-käyttäjille. Yleensä myös yrityksen sisäisillä käyttäjillä on pääsy näihin palvelinfarmeihin. Näiden palvelinfarmien tarjoamat palvelut ja niiden käyttäjät nojautuvat selaimella käytettäviin web-käyttöliittymiin, mikä tekee niistä yhtenäisiä Internet-ympäristössä. Internet-palvelinfarmit voidaan vielä jakaa kahteen aliryhmään: dedikoituihin ja DMZ (Demilitarized Zone) Internet-palvelinfarmeihin. (Arregoces & Portolani 2003.)

Dedikoituja Internet-palvelinfarmeja käytetään yleensä suurten Internetin välityksellä tapahtuvien yritysten ydintoimintojen tukemiseen. Tyypillisiä ydintoimintoja ovat esimerkiksi näkyvyys Internetissä ja siellä tapahtuva kaupankäynti. Tietoturva ja skaalautuvuus ovat tämän tyyppisen palvelinfarmin suurimpia huolenaiheita. (Mt.)

DMZ Internet-palvelinfarmeja käytetään tukemaan Internet-sovellusten lisäksi myös Internetiin pääsyä yrityksestä. Tällä tarkoitetaan sitä, että palvelinfarmia tukevaa infrastruktuuria käytetään myös tukemaan yrityksen sisäisten käyttäjien pääsyä Internetiin. Nämä palvelinfarmit sijaitsevat nimensä mukaisesti demilitarisoidulla alueella, koska ne ovat samalla osana yrityksen sisäistä verkkoa, mutta myös käytettävissä Internetin välityksellä. Näitä palvelinfarmeja käytetään tukemaan palveluita kuten verkkokaupankäyntiä ja muita yleisiä sovelluksia, jotka ovat sekä Internet- että intranet-käyttäjien käytettävissä. Tietoturvavaatimukset ovat näissä palvelinfarmeissa erittäin tiukkoja, koska yrityksen sisäinen verkko on pidettävä turvassa ulkopuolisilta käyttäjiltä. (Mt.)

2.6.2 Intranet-palvelinfarmit

Intranet-palvelinfarmit muistuttavat käytännössä muilta osin Internet-palvelinfarmeja, mutta ne ovat ainoastaan yrityksen sisäisten käyttäjien käytettävissä. Yrityksen sisäiset käyttäjät voivat kuitenkin olla näihin palvelinfarmeihin yhteydessä Internetin välityksellä käyttämällä VPN-teknologioita. Nämä palvelinfarmit sisältävät suurimman osan yrityksen liiketoimintaa ja sisäisiä sovelluksia tukevista resursseista. Skaalautuvuus, korkea saatavuus, tietoturva ja hallittavuus ovat näiden palvelinfarmien tärkeimpiä suunnittelukriteerejä. (Arregoces & Portolani 2003.)

2.6.3 Extranet-palvelinfarmit

Extranet-palvelinfarmit asettuvat toiminnaltaan Internet- ja intranet-palvelinfarmien välimaastoon. Niissä käytetään muiden palvelinfarmien tavoin web-pohjaisia sovelluksia, mutta ne ovat ainoastaan rajoitetun käyttäjäryhmän käytettävissä. Nämä palvelinfarmit ovat pääosin liikekumppaneiden eli yrityksen ulkopuolisten, mutta kuitenkin luotettujen käyttäjien käytettävissä. Extranetin pääasiallinen käyttötarkoitus on yritysten välinen kommunikaatio, mikä mahdollistaa nopean informaation jakamisen tietoturvallisessa ympäristössä. Tähän kommunikaatioon voidaan käyttää yritysten välisiä dedikoituja yhteyksiä tai VPN-teknologiaa. Skaalautuvuus, saatavuus ja tietoturva ovat tässäkin tapauksessa tärkeitä huomioon otettavia asioita. Tietoturvan kannalta tulee erityisesti ottaa huomioon, että liikekumppanilla on pääsy tiettyyn osaan yrityksen sovelluksista, kuitenkin pysyen erotettuna muista osista yrityksen sisäistä verkkoa. (Arregoces & Portolani 2003.)

2.7 Valinta ja vertailu

2.7.1 Pilvipalvelut

Sen sijaan, että yritys ottaisi pilvipalveluja käyttöönsä pelkästään teknologian uutuuden takia tai valitsisi vääränlaisen palvelun pelkästään palveluntarjoajan tuttuuden takia, tulisi sovelluksen suunnitteluvaiheessa vastata seuraavanlaisiin kysymyksiin:

- Miksi? Mitä ongelmaa ollaan ratkaisemassa? Mitkä liiketoiminnalliset tavoitteet ajavat tätä muutosta?
- Kuka? Kuka tarvitsee ratkaisun tähän ongelmaan? Mitkä sisäiset ja ulkopuoliset osapuolet ovat osallisena?
- Mitä? Mitä teknisiä tai liiketoiminnallisia vaatimuksia on otettava huomioon? Mitä laillisia tai sääntelyrajoitteita löytyy? Mitä riskejä on olemassa?
- Missä? Missä näitä palveluita tullaan käyttämään? Onko olemassa mitään sijaintiin liittyviä vaatimuksia (sääntely, verot, käytettävyys, kieli, lokalisaatio)?
- Milloin? Milloin näitä palveluita tullaan tarvitsemaan? Kuinka suuri on budjetti? Onko olemassa riippuvuuksia muista projekteista?

- Miten? Miten organisaatio voi tarjota näitä palveluita? Kuinka valmiita organisaatio, arkkitehtuuri ja asiakas ovat muutosta varten?

Näihin kysymyksiin vastaamalla päästään parempaan asemaan sopivia käyttöönotto- ja palvelumalleja valittaessa. Sopivaa palvelumallia valitessa tulee ottaa huomioon tekijöitä kuten aikataulu, budjetti, organisaation valmius sekä tekniset ja liiketoiminnalliset vaatimukset. Huomioon tulee ottaa myös onko kyseessä uusi projekti, ollaanko jo olemassa olevaa järjestelmää siirtämässä pilveen, vai jokin näiden yhdistelmä. Tärkeä huomioon otettava seikka on myös se, minkälaisia käyttäjiä ollaan palvelemassa ja minkälaista dataa ollaan säilyttämässä. Esimerkiksi sosiaalisen median verkkosivustolla on täysin erilaiset vaatimukset verrattuna potilastietoja käsittelevään terveydenhuollon järjestelmään. (Kavis 2014.)

2.7.2 Palvelinfarmit

Hintojen laskiessa nopeasti tulevat pilvipalvelut koko ajan houkuttelevammaksi vaihtoehdoksi yrityksille. Pilvipalvelut eivät kuitenkaan ole ratkaisu kaikkeen. Useissa tapauksissa yksinkertainen yrityksen sisäinen palvelinfarmi voi olla parempi ratkaisu. Näiden ratkaisujen välillä valintaa tekevän yrityksen tulisi verrata molempien hyviä ja huonoja puolia päästäkseen heille sopivampaan tulokseen. (GFI Software 2010.)

Pilvipalveluiden hyvät puolet liittyvät suurilta osin niiden mataliin kustannuksiin, kun taas paikallisiin ratkaisuihin päädytään usein tietoturvan takia. Pilvipalvelut ovat yleisesti ottaen kustannustehokkaita, eikä asiakkaan tarvitse miettiä ohjelmistojen lisensointia tai palveluja tukevan fyysisen infrastruktuurin ylläpitoa. Paikallisessa ratkaisussa koko järjestelmä ja kaikki sen sisältämä data on täysin yrityksen hallinnassa. Yrityksen data säilötään ja käsitellään sisäisesti, ja sitä tukevaa infrastruktuuria hallitsee yrityksen oma henkilökunta. Vaikka aloituskustannukset ovatkin paikallisessa ratkaisussa pilvipalveluita suuremmat, maksaa se itsensä takaisin pitkällä aikavälillä. (Mt.)

Paikallisten ratkaisujen huonoihin puoliin lukeutuvat mahdolliset puutteet tarvittavan infrastruktuurin rakentamiseen ja sen ylläpitämiseen, mahdolliset ohjelmistojen lisensoinnit sekä yrityksen sisäisen henkilökunnan tietotaidon puute. Pilvipalveluihin liittyviin huolenaiheisiin kuuluvat sensitiivisen datan käsittely kolmannen osapuolen

toimesta, redundanttisuus palveluntarjoajan päässä sekä mahdolliset ongelmat kustomoitujen järjestelmien integraatiossa tai palveluntarjoajan vaihdon yhteydessä. (Mt.)

2.7.3 Päätös

Näitä asioita pohdittuaan toimeksiantaja päätyi ratkaisuun, jossa ainakin tätä työtä koskeva pilottiversio toteutetaan yrityksen sisäisessä verkossa toimivana yhden tai useamman palvelinkoneen palvelinfarmina. Suurimpina syinä tähän valintaan päätymiselle olivat käsiteltävän datan sensitiivisyys ja muut tietoturvasyyt. Tietoturvasyiden taustalla on KATAKRI:n (Kansallinen turvallisuusauditointikriteeristö) noudattaminen yrityksen tekemän sopimuksen mukaisesti. Sen mukaan yrityksellä ei ole mahdollisuutta käyttää tähän ulkoisia palveluja, vaan koko järjestelmä tulee toteuttaa yrityksen palomuurien suojaamassa lähiverkossa. Toimeksiantaja pyrki myös ratkaisuun, joka toimii mahdollisimman monessa eri paikassa sitoutumatta mihinkään tiettyyn palveluntarjoajaan. Eli jos tulevaisuudessa on tarve tai mahdollisuus, tulisi paikallisen ratkaisun siirtäminen kolmannen osapuolen palveluntarjoajan pilvipalveluihin olla suhteellisen yksinkertaista.

3 Käytetyt teknologiat

3.1 Tietokannat ja -rakenteet

3.1.1 SQL ja NoSQL

SQL-tietokanta (Structured Query Language), tai relaatiotietokanta, on jo pitkään käytössä ollut malli datan säilyttämiseen tietokannassa. Vaikka se on edelleen käyttökelpoinen tapa perinteisen rakenteellisen datan säilyttämiseen, ei se kuitenkaan sovellu kaiken nykyaikaisten sovellusten mukana tuoman rakenteettoman datan käsittelemiseen. Niinpä sen rinnalle on lähiaikoina noussut vaihtoehdoksi NoSQL (Non-SQL, Not only SQL). (Daniela 2015.)

SQL- ja NoSQL-tietokannat eroavat toisistaan useilla eri tavoilla. Merkittävin ero näiden kahden välillä on se, miten tai missä muodossa dataa säilytetään tietokannassa. Perinteisessä SQL-tietokannassa data järjestellään toisiinsa relaatioiden avulla yhteydessä oleviin tauluihin. Jokaisella taululla on staattinen skeema, jossa määritellään minkälaista dataa sarakkeet voivat sisältää. Data tallennetaan tauluihin riveinä ja sen tulee noudattaa taululle ennalta määrättyä skeemaa. (Mt.)

NoSQL-tietokannoissa käytetään vastaavasti dynaamista skeemaa, eli sitä ei määritetä etukäteen. Tämä mahdollistaa rakenteettoman datan yksinkertaisen käsittelyn. NoSQL-tietokannat voidaan jakaa eri tyyppeihin sen mukaan miten dataa siellä säilytetään. Yksinkertaisin näistä tyypeistä on avain-arvopari-varasto (key-value store). Siinä jokainen tietue tallennetaan tietokantaan avain-arvoparina, jossa avain on jokin attribuutin nimi ja sitä vastaava arvo voi olla jotakin tiettyä tietotyyppiä. Dokumenttivarastossa (document store) jokaista avainta vastaa dokumentiksi kutsuttu monimutkainen tietorakenne. Jokainen dokumentti voi sisältää useita erityyppisiä avain-arvopareja tai jopa sisäkkäisiä dokumentteja. Sarakepohjaiset varastot (wide-column store) ovat suurten datamäärien kyselyjä varten optimoituja tietovarastoja, joissa data säilötään rivien sijasta sarakkeittain. Lisäksi on vielä olemassa graafivarastoja, joissa säilötään informaatiota verkostoista, kuten sosiaalisista yhteyksistä. (NoSQL Database Explained n.d.)

SQL-tietokantoja hallitaan SQL-kielen avulla, kun taas NoSQL-tietokannoilla ei ole käytössä mitään yhteistä kieltä. Yhteisen kielen sijaan NoSQL-kyselyjen syntaksi vaihtelee tietokantojen välillä, minkä vuoksi SQL-tietokantoja suositaan monimutkaisempien kyselyjen tekemiseen. Myös monimutkaisempien transaktioiden suorittaminen on vakaampaa SQL-ympäristössä. NoSQL-tietokantojen vahvuudet ovat skaalautuvuudessa ja ison datan (big data) käsittelyssä. NoSQL-tietokannat skaalautuvat horisontaalisesti, eli uusia palvelimia voidaan lisätä helposti sitä mukaan kun liikenteen tai datan määrä kasvaa. Sen lisäksi NoSQL-tietokannat soveltuvat paremmin käsittelemään suurta määrää rakenteetonta hierarkkista dataa. (Daniela 2015.)

3.1.2 JSON ja BSON

JSON (JavaScript Object Notation) on helposti sekä ihmisen että koneen käsiteltävissä oleva tiedostomuoto tiedonvälitykseen. JSON tukee kaikkia yleisimpiä tietotyyppejä: numeroita, merkkijonoja, boolean-arvoja, taulukoita ja JavaScript-olioita. Sitä käytetään myös dokumenttipohjaisten tietokantojen tietueiden tallentamiseen. Tällaisten tietokantojen palauttamat kyselyjen tulokset ovat helposti suoraan JavaScriptin ja muiden ohjelmointikielien tulkittavissa. (JSON and BSON n.d.)

MongoDB käsittelee JSON-dokumentteja kulissien takana binäärikoodatussa BSON-muodossa (Binary JSON). BSON laajentaa JSON-muotoa lisäten siihen uusia tietotyyppejä sekä mahdollistaen tehokkaamman koodauksen ja dekoodauksen. MongoDB:n BSON-implementaatio tukee JSON-muodon tavoin myös sisäkkäisiä taulukoita ja olioita. Toisin sanoen MongoDB tarjoaa käyttäjälle JSON-muodon helppouden ja joustavuuden yhdessä binäärimuodon keveyden ja nopeuden kanssa. (Mt.)

3.1.3 MongoDB

MongoDB on vapaaseen lähdekoodiin perustuva dokumenttipohjainen NoSQL-tietokanta. MongoDB:ssä tietueet tallennetaan BSON-muotoisina dokumentteina tietokannassa oleviin kokoelmiin. Kokoelma on joukko toisiinsa liittyviä dokumentteja, jotka jakavat yhteiset indeksit toistensa kanssa. MongoDB-kyselyt kohdistetaan aina johonkin tiettyyn kokoelmaan. Kyselyssä voidaan määrittää ehdot, joiden perusteella tuloksia haetaan sekä eräänlainen projektio, tai maski, jolla määritetään mahdollisten tuloksien palautettavat kentät. Lisäksi sen yhteydessä voidaan määrittää esimerkiksi

tuloksien lajittelujärjestykseen vaikuttavia tai niiden määrää rajoittavia toimintoja. Myös dokumenttien lisäämiseen, päivittämiseen ja poistamiseen liittyvät toiminnot toimivat samalla periaatteella. (MongoDB Manual n.d.)

Yksinkertaisten kyselyjen lisäksi MongoDB tarjoaa myös mahdollisuuden aggregaatioiden suorittamiseen. Aggregaatiot ovat operaatioita, jotka prosessoivat tietueita tutkimalla niiden sisältöä, suorittamalla niiden perusteella laskutoimituksia ja lopulta palauttamalla lasketut tulokset. Tavallisten kyselyjen tapaan aggregaatiot kohdistetaan johonkin kokoelmaan, ja tuloksena saadaan yksi tai useampia dokumentteja. MongoDB tarjoaa aggregaatioiden suorittamiseen kaksi eri tapaa: yleisessä käytössä olevan MapReduce-mallin ja MongoDB:n oman aggregaatiokehyksen. (Mt.)

MapReduce-mallin toiminta jaetaan yleensä kahteen vaiheeseen. Map-vaiheessa prosessoidaan jokainen sille syötetty dokumentti, ja jokaisen dokumentin kohdalla lähetetään yksi tai useampi olio seuraavan vaiheen käsiteltäväksi. Reduce-vaiheessa edelliseltä vaiheelta saadut tiedot redusoidaan, tai yhdistetään, palautettavaksi tulokseksi. MongoDB:n aggregaatiokehyksen toimintaa voidaan kuvata monivaiheisena putkistona (pipeline), joka muuntaa sille syötetyt dokumentit aggregoiduksi tulokseksi. Tähän putkistoon voidaan asettaa peräkkäin haluttu määrä erilaisia operaatioita, edellisen operaation tuloksen toimiessa syötteenä seuraavan vaiheen operaatiolle. Tällaisia operaatioita ovat esimerkiksi dokumenttien suodatus, niiden ryhmittely ja lajittelu kenttien perusteella, taulukoiden aggregointi sekä erilaiset laskutoimitukset ja merkkijonotoiminnot. (Mt.)

MongoDB käyttää muiden tietokantajärjestelmien tavoin indeksejä tehokkaampaan kyselyjen suorittamiseen. Ilman indeksejä MongoDB:n täytyy skannata kokoelman jokainen dokumentti löytääkseen kyselyä vastaavat tulokset. Indeksejä käyttämällä saadaan vähennettyä skannattavien dokumenttien määrää. Indeksit ovat tietorakenteita, jotka säilövät pienen osan kokoelman datasta helposti läpikäytävässä muodossa. Tarkemmin ottaen ne säilövät yhden tai useamman kentän arvot kyseisen arvon mukaan lajiteltuna. MongoDB tukee monenlaisia indeksejä, kuten yksittäisiä ja yhdistelmäindeksejä, taulukkoindeksejä, teksti-indeksejä, tiivisteindeksejä sekä uniikkeja ja harvoja indeksejä. MongoDB luo jokaiselle kokoelmalle oletuksena uniikin indeksin _id-kentälle, mikä estää saman _id-arvon sisältävien dokumenttien lisäämisen kokoelmaan. (Mt.)

MongoDB tukee replikointia redundanttisuuden ja datan saatavuuden parantamiseksi. Käytännössä tämä tarkoittaa sitä, että sama data replikoidaan, tai kopioidaan, usealle eri palvelimelle. Näin ollen yksittäisen tietokantapalvelimen hajoaminen tai jokin muu vika ei keskeytä koko tietokannan toimintaa. Replikointia voidaan käyttää myös lukukapasiteetin parantamiseen, koska luku- ja kirjoitusoperaatiot on mahdollista lähettää eri palvelimille. Replikajoukolla tarkoitetaan samaa dataa sisältävien mongod-instanssien ryhmää. Replikajoukko sisältää yhden ensisijaisen instanssin, joka hoitaa oletuksena kaikki luku- ja kirjoitusoperaatiot. Kaikki loput ovat toissijaisia instansseja, jotka pitävät datan synkronoituna ensisijaisen instanssin kirjaamien muutosten perusteella. Jos ensisijainen instanssi lakkaa toimimasta, valitaan toissijaisten instanssien joukosta automaattisesti uusi ensisijainen instanssi. (Mt.)

MongoDB käyttää datan hajauttamista (sharding) useille palvelimille tukeakseen erittäin suuria datamääriä ja korkeaa suorituskykyä vaativia operaatioita. Jos yksittäisen palvelimen kapasiteetti ei ole riittävä, voidaan skaalaus toteuttaa kahdella eri tavalla. Vertikaalisessa skaalauksessa palvelimen resursseja lisätään kapasiteetin kasvattamiseksi, mikä on suhteettoman kallista verrattuna useampaan vähemmän tehokkaaseen palvelimeen. Horisontaalisessa skaalauksessa, tai hajauttamisessa, data taas hajautetaan usealle palvelimelle (shard). Jokainen palvelin on itsenäinen tietokantansa, mutta yhdessä kaikki palvelimet muodostavat yhden loogisen tietokannan. MongoDB:ssä hajauttaminen toteutetaan konfiguroimalla hajautettu klusteri. (Mt.)

Hajautettu klusteri koostuu kolmesta komponentista: varsinaisista tietokantapalvelimista (shard), kyselyjen reitittäjistä (query router) ja konfiguraatiopalvelimista (config server). Tarjotakseen korkeaa saatavuutta ja datan yhdenmukaisuutta, tulisi varsinaisten dataa säilövien tietokantapalvelimien olla tuotantokäytössä replikajoukkoja. Kyselyjen reitittäjät, tai mongos-instanssit, toimivat rajapintana asiakassovelluksille ohjaten kyselyt oikealle tietokantapalvelimelle ja palauttamalla sen tuottaman tuloksen. Niitä voi olla yksi tai useampi hajautetussa klusterissa, mutta asiakassovellus lähettää pyyntöjä niistä vain yhdelle. Konfiguraatiopalvelimissa säilötään hajautetun klusterin metadataa. Kyselyjen reitittäjät käyttävät tätä metadataa reitittäessään kyselyjä oikeille tietokantapalvelimille. Tuotantokäytössä hajautetun klusterin tulisi sisältää kolme konfiguraatiopalvelinta. (Mt.)

MongoDB toteuttaa datan hajauttamisen kokoelmien tasolla. Hajauttamisessa kokoelman data osioidaan käyttäen hajautusavainta. Hajautusavain voi olla joko indeksoitu kenttä tai yhdistelmäindeksi, joka löytyy jokaisesta kokoelman dokumentista. MongoDB jakaa hajautusavaimen arvot paloihin (chunk) ja hajauttaa nämä palat tasaisesti tietokantapalvelimille. Hajautusavaimen arvojen jakamiseen paloihin voidaan käyttää joko arvoalueisiin (range-based partitioning) tai tiivisteisiin perustuvaa osiointia (hash-based partitioning). Arvoalueisiin perustuvassa osioinnissa data jaetaan hajautusavaimen arvojen perusteella arvoaluepaloihin, jotka sisältävät kaikki arvot jostakin tietystä minimiarvosta johonkin tiettyyn maksimiarvoon. Tiivisteisiin perustuvassa osioinnissa kentän arvoista lasketaan tiivisteet, joiden perusteella data jaetaan paloihin. Tiivisteitä käyttämällä kaksi toisiaan lähellä olevaa arvoa kuuluvat epätodennäköisemmin samaan palaan, mikä takaa kokoelman satunnaisemman jakautumisen klusterissa. (Mt.)

3.2 Palvelin

3.2.1 Node.js

Node.js on Google Chromen V8 JavaScript-moottorin päälle rakennettu palvelinpuolen alusta nopeiden ja skaalautuvien verkkosovellusten kehittämiseen. Node.js-sovellukset ohjelmoidaan JavaScript-ohjelmointikielellä, ja se käyttää tapahtumapohjaista, asynkronista I/O-mallia (Input/Output), mikä sopii erityisesti reaaliaikaisten, I/O-intensiivisten sovellusten kehittämiseen. Käytännössä tämä tarkoittaa sitä, että tapahtumia kutsutaan rekisteröimällä callback-funktio tapahtumafunktion palautuksen käsittelyä varten. Tätä palautusta odotettaessa sovelluksen seuraava tapahtuma, tai funktio, voidaan laittaa jonoon sen suorittamista varten. Kun ensimmäinen funktio on valmis, sen callback-tapahtuma suoritetaan ja käsitellään sitä kutsuneen funktion toimesta. (Gackenheimer 2013.)

Useat ohjelmointikielet ja -alustat ovat riippuvaisia kolmannen osapuolen moduuleista niiden käytettävyyden laajentamiseksi. Node.js on yksi tällä periaatteella toimivista alustoista. NPM (Node Package Manager) on Node.js-asennuksen mukana tuleva paketinhallintajärjestelmä ydintoimintoja laajentavien kolmannen osapuolen moduulien hallintaan. (Mt.)

3.2.2 Express.js

Express.js on Node.js http-moduuliin ja Connect-komponentteihin perustuva web-sovelluskehys. Näitä komponentteja kutsutaan termillä ”middleware”, ja ne ovat oleellinen osa sovelluskehyksen konfiguraatioon perustuvaa filosofiaa. Toisin sanoen Express.js on erittäin konfiguroitavissa oleva järjestelmä, joka mahdollistaa kirjastojen vapaan valinnan tarpeen mukaan. Tämän ansiosta Express.js on erittäin joustava ja muunneltavissa oleva sovelluskehys web-sovellusten kehittämiseen. (Mardan 2014.)

Express.js hoitaa useimmat web-sovelluksissa esiintyvät työläät ja itseään toistavat toiminnot, kuten HTTP-pyyntöjen (Hypertext Transfer Protocol) sisällön ja evästeiden tulkinnan, istuntojen hallinnan, reittien tulkitsemisen HTTP-metodin ja URL-osoitteen (Uniform Resource Locator) perusteella sekä vastauksen otsikoiden määrittämisen tietotyyppien mukaan. Lisäksi se tarjoaa MVC-arkkitehtuurin (Model-View-Controller) kaltaisen rakenteen web-sovellusten eri osien toisistaan erottamista varten. (Mt.)

3.3 Käyttöliittymä ja työkalut

3.3.1 Vue.js

Vue.js on MVVM-arkkitehtuuriin (Model View ViewModel) perustuva asiakaspuolen JavaScript-kirjasto web-sovellusten kehittämiseen. Se muistuttaa joiltain osin Googlen yhden sivun web-sovellusten rakentamista varten kehittämää AngularJS-sovelluskehystä. Vue.js on kuitenkin näkymän tasolla toimiva yksinkertainen ja joustava kirjasto, eikä täysimittainen sovelluskehys, joten se sopii erityisesti kevyempiin ja vähemmän toimintoja vaativiin ratkaisuihin. Vue.js-kirjaston keskipisteenä toimii ”ViewModel” eli kaksisuuntainen tiedon sidonta, jonka avulla näkymässä nähtävillä oleva data ja sovelluksen mallin data sidotaan toisiinsa. (Chenkie 2015.)

3.3.2 Gulp.js

Gulp.js on virtoihin (stream) perustuva rakennustyökalu asiakaspuolen web-kehityksessä tarvittavien tehtävien ajamiseen ja automatisointiin. Gulp.js-tehtävää voidaan ajatella erilaisia operaatioita sisältävänä putkena, jonka läpi sille syötetyt tiedostot kulkevat. Käytettävissä olevia operaatioita ovat esimerkiksi staattisten tiedostojen

optimointi eli CSS- (Cascading Style Sheets) ja JavaScript-tiedostojen yhdistäminen ja pienentäminen sekä kuvien ja fonttien optimointi, CSS-esikääntäjäkielten ja erilaisten JavaScript-muotojen muuntaminen selainten ymmärtämään muotoon, automaattinen tehtävänajo ja selaimen päivitys tiedostojen tallennuksen yhteydessä sekä erilaisten testien ajaminen. (Liew 2015.)

3.3.3 Bootstrap

Bootstrap on suosittu HTML (Hypertext Markup Language), CSS ja JavaScript -sovelluskehys responsiivisten web-käyttöliittymien kehittämiseen. Sen tarjoamiin toiminallisuuksiin kuuluvat esimerkiksi 12 sarakkeen responsiivinen ruudukkojärjestelmä (grid system), valmiita tyylejä, komponentteja ja muita resursseja käyttöliittymäkomponentteja varten sekä JavaScript-komponentteja lisäämään toiminnallisuutta käyttöliittymään. Valmiiden komponenttien ja muiden resurssien ansiosta Bootstrap sopii erityisesti nopeaan ja vaivattomaan prototyypittämiseen. Suurin osa sen tarjoamista toiminnallisuuksista saadaan käyttöön lisäämällä CSS-luokkia tai data-attribuutteja HTML-elementteihin. Sen käyttöä voidaan myös optimoida kustomoimalla eli ottamalla käyttöön ainoastaan tarvittavat osat koko sovelluskehyksen tarjonnasta. (Chapman 2014.)

4 Toteutus

4.1 Analysaattorituote

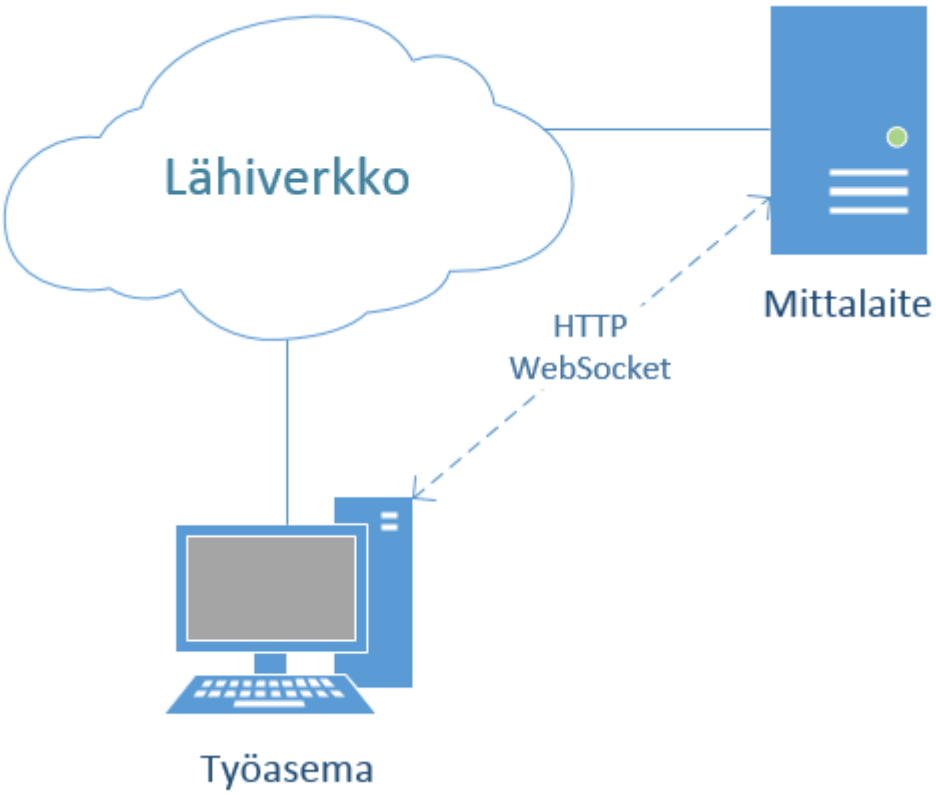
EnviAnalyzer on Environics Oy:n valmistama mittalaite, jonka toiminta perustuu ionien tunnistamiseen niiden liikkuvuusmuutoksen perusteella ja ainepitoisuuden laskemiseen ionien luovuttamien alkeisvarausvirtojen perusteella. Mittalaitteen mittauskennoon kuuluu kaksi osaa: mitattavan näyteilman ionien esierotin, joka aiheuttaa liikkuvuusmuutoksen altistamalla ionit korkealle sähkökentälle sekä komponentti, jossa mitataan ionien varausvirrat annetuilla esierottimen arvoilla. (Julkunen 2015.)

Käytännössä analysaattorissa vaihdellaan AIMS-, VC- ja VRF-sähkökenttien jännitearvoja sekä mitataan kolmea kanavaa: Z, CH ja S. Tämän lisäksi laite pitää kirjaa erilaisista sekasensoreista kuten paineista, lämpötiloista ja virtauksista.

Laitteella tehtyjen mittausten yhteydessä käytetään termejä pyyhkäisy (sweep) ja skannaus (scan). Pyyhkäisyllä tarkoitetaan sitä, kun mittalaite mittaa jotakin tiettyä AIMS-, VC-, VRF-avaruuden aluetta, esimerkiksi vaihdellen yhtä jännitearvoa ja pitäen kaksi muuta arvoa vakiona. Yhdestä pyyhkäisystä saadaan laskettua erilaisia piirteitä (feature) eli hyötydataa sisältäviä lukuarvoja, esimerkiksi tiettyjä huippu- ja keskiarvoja. Skannauksella taas tarkoitetaan yhden tai useamman pyyhkäisyn kokoelmaa. Skannauksen tuloksena sen sisältämien pyyhkäisyjen piirteistä voidaan laskea käyttäjän tarvitsemaa pitoisuustietoa tai muita laitteen toimintaan vaikuttavia asioita.

4.2 Mittalaitejärjestelmä

Korkealta tasolta katsottuna koko mittalaitejärjestelmä näyttää yksinkertaiselta. Käyttäjän näkökulmasta analysaattori on yksi fyysinen lähiverkossa oleva laite, jota käyttäjä ohjaa web-käyttöliittymän avulla toiselta lähiverkossa olevalta työasemalta. Kuviossa 4 on esitettynä mittalaitejärjestelmä käyttäjän näkökulmasta.

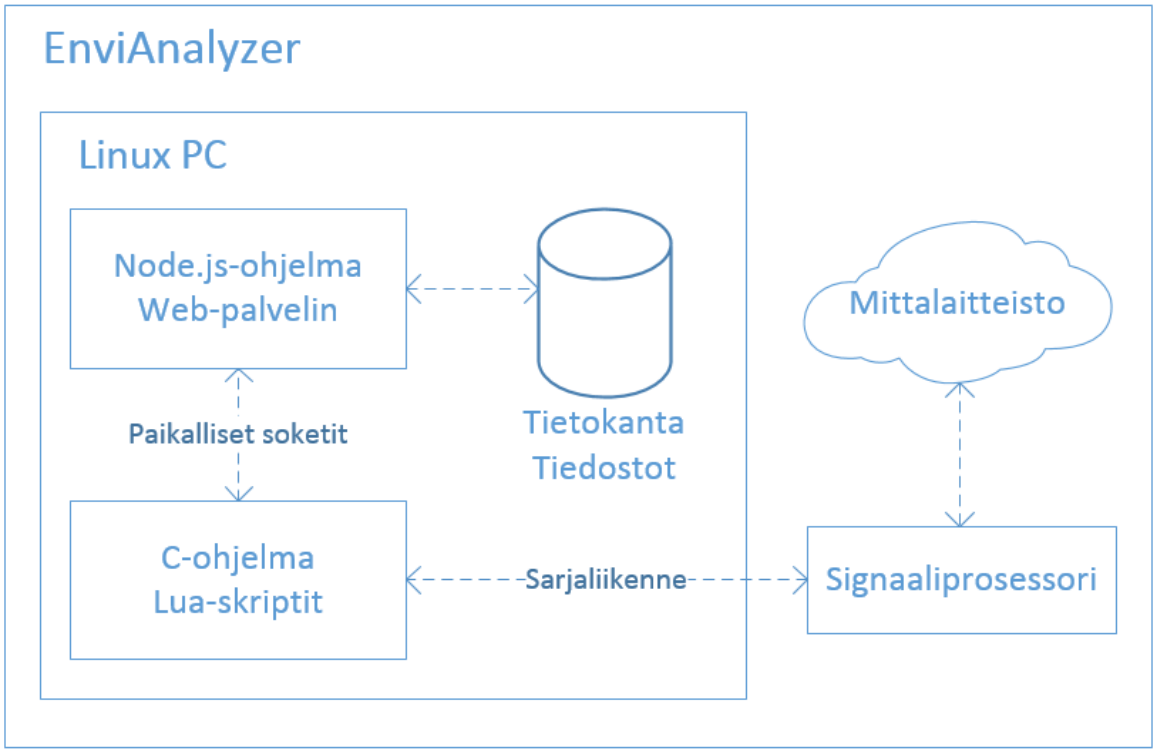


Kuvio 4. Mittalaitejärjestelmä käyttäjän näkökulmasta

EnviAnalyzer-mittalaitejärjestelmään kuuluu kuitenkin varsinaisen mittalaitteen lisäksi kaksi muuta tärkeää osaa: koko sulautettua järjestelmää hallitseva yhden piirilevyn tietokone ja signaaliprosessori varsinaisen mittalaitteen hallintaan. Signaaliprosessori on sarjaliitännän kautta tietokoneeseen yhteydessä oleva laite, jolla varsinaisen mittalaitteen toimintaa ohjataan. Tämä toiminta toteutetaan tietokoneella ajettavien skriptien avulla. Näillä skripteillä mittalaitteen raudalta tilataan skannauksia ja määritellään sallitut vaihteluvälit sekasensorien signaaleille.

Sulautetulla tietokoneella ajettavien ohjelmien alla toimii erittäin pieni ja kevyt Linux-käyttöjärjestelmä. Sen päällä ajettava ohjelmisto voidaan jakaa kahteen osaan: C-kielellä kirjoitettuun ohjelmaan ja Node.js-palvelinsovellukseen. C-ohjelmalla suoritetaan kaikki raskaat laskentaoperaatiot, ja se kommunikoi sekä signaaliprosessorin kanssa sarjaliitännän kautta että Node.js-sovelluksen kanssa käyttöjärjestelmän sisäisten sokettien avulla. C-ohjelma tarjoaa myös rajapinnan Lua-kielisten skriptien ajamiseen, mikä mahdollistaa mittalaitteen ohjaamisen niiden avulla.

Node.js-sovellus toimii web-palvelimena käyttöliittymälle, tarjoaa käyttäjälle tai käyttöliittymälle web-rajapinnan ja vastaa datan varastoinnista tiedostoihin tai tietokantaan. Käyttäjä kommunikoi sovelluksen kanssa HTTP-pyyntöjen ja WebSocket-teknologian välityksellä. Kuviossa 5 on esitettynä mittalaitejärjestelmän sisäinen kommunikaatio.



Kuvio 5. Mittalaitejärjestelmän sisäinen kommunikaatio

4.3 Toteutuksen toimeksianto

Tässä työssä toteutetun uuden järjestelmän tarkoituksena on toimia tietokantapohjaisena haku- ja analysointisovelluksena mittalaitteen tuottamalle sensoridatalle sekä tarjota web-pohjainen käyttöliittymä sen käyttämiseen. Järjestelmän toteutuksessa tuli ottaa huomioon seuraavia asioita:

- Järkevä varastointitapa levynkulutuksen kannalta
- Toiminnot tulisi suorittaa suurimmaksi osaksi kantamoottorilla
- Redundanttisuus ja varmuuskopiointi
- Yksinkertainen tietokantaan tallentaminen ja sieltä ulos vieminen

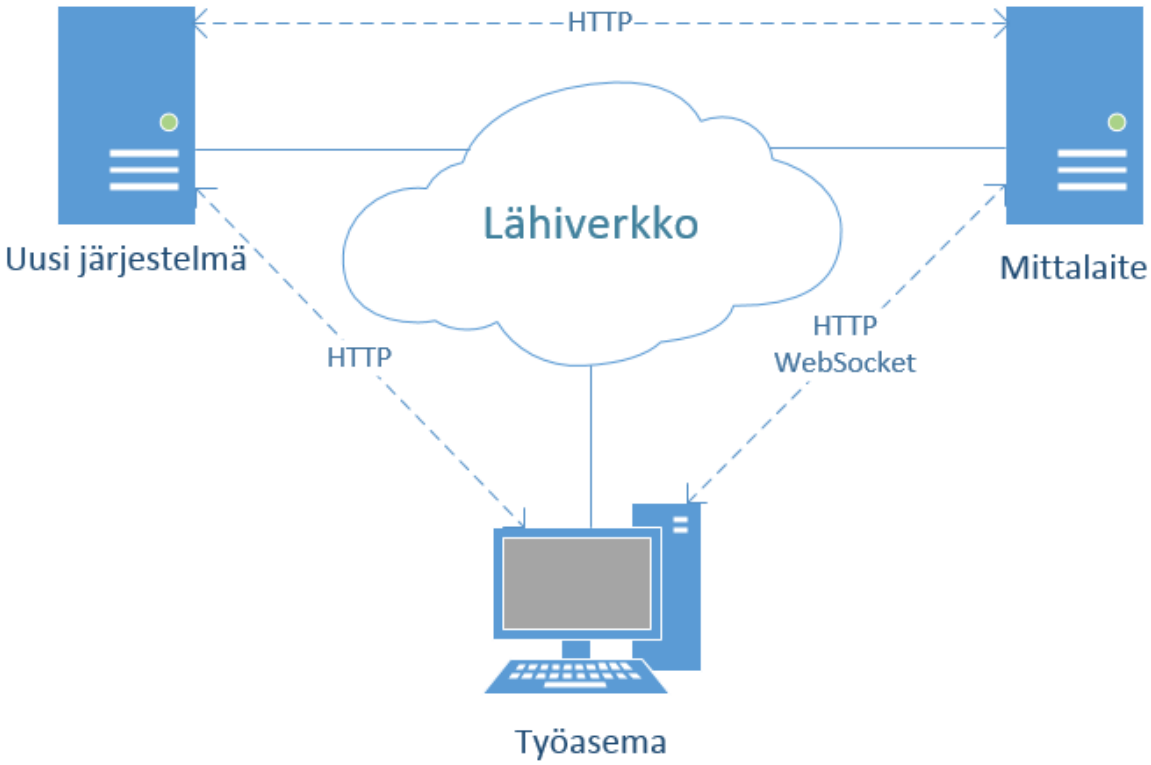
- Yksinkertaiset hakutoiminnot dokumentin kenttien perusteella
- Monimutkaisemmat laskutoimitukset ja muut analysointitoiminnot dokumenttien sisältämän datan perusteella
- Selainpohjainen päätelaiteriippumaton käyttöliittymä

Luvussa 3 esitetyt käytetyt teknologiat, etenkin MongoDB, mahdollistavat kyseisten vaatimusten täyttämisen. NoSQL-tietokanta soveltuu erinomaisesti mittalaitteiden tuottaman sensoridatan tallentamiseen. Mittalaitteiden tuottama data on JSON-muotoista, mikä mahdollistaa yksinkertaisen tietokantaan tallentamisen yhden kutsun avulla. Lisäksi NoSQL-tietokantojen skeemattomuus takaa sen, että tietokanta ei välitä mahdollisista muutoksista sinne tallennettavan mittausdatan rakenteessa.

MongoDB tarjoaa mahdollisuuden käytettävän kantamoottorin valintaan kahdesta vaihtoehdosta. Näistä kantamoottoreista toinen mahdollistaa tietokannan datan zlib-pakkauksen, mikä pienentää sen viemää levytilaa jopa gzip-pakattujen tiedostojen tasolle. MongoDB tarjoaa myös suhteellisen monipuoliset mahdollisuudet datan analysointiin aggregaatiokehyksen muodossa. Tätä käyttämällä kaikki raskaammatkin analysointitoiminnot saadaan suoritettua suoraan kantamoottorilla. MongoDB:n tarjoamiin ominaisuuksiin kuuluvat myös replikajoukot redundanttisuuden takaamiseen, hajautetut palvelinklusterit datan hajauttamiseen sekä useat työkalut datan varmuuskopiointiin, vientiin ja tuontiin.

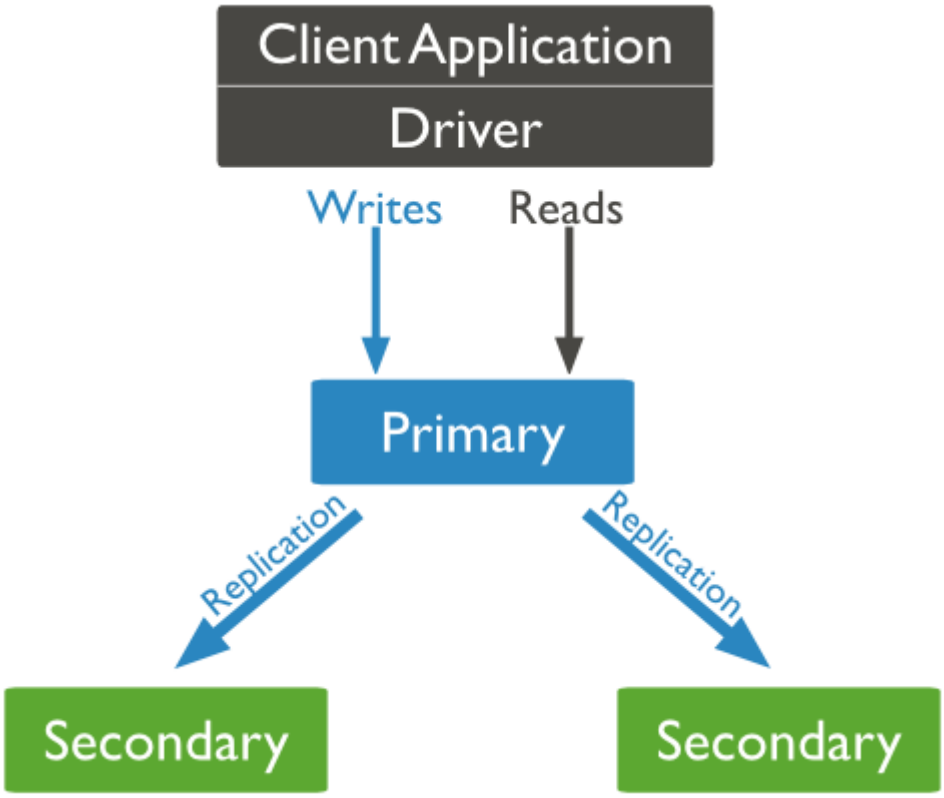
4.4 Palvelinfarmin konsepti

Uusi järjestelmä voidaan yksinkertaisimmillaan toteuttaa yhdellä palvelinkoneella, jossa sama palvelinkone on vastuussa web-palvelimesta ja kaikista tietokantatoiminnoista. Vaihtoehtoisesti redundanttisuutta ja skaalautuvuutta vaadittaessa järjestelmää voidaan laajentaa useamman palvelinkoneen palvelinfarmiksi. Tarkoituksena oli siis luoda kokonaan uusi erillinen järjestelmä, eikä integroida sitä jo olemassa olevaan mittalaitejärjestelmään. Järjestelmien välinen kommunikointi tulee tapahtumaan web-palvelimen tarjoaman rajapinnan kautta. Kuviossa 6 on selvennettynä toteutetun järjestelmän toiminta käyttäjän näkökulmasta.



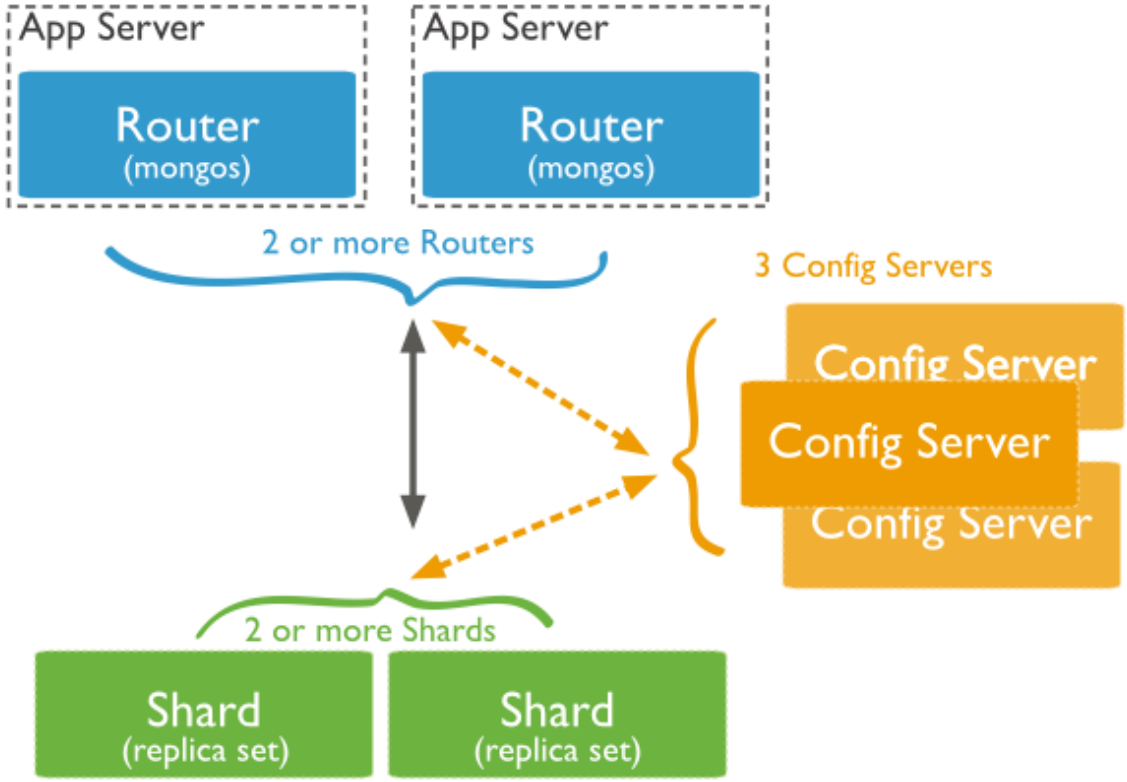
Kuvio 6. Toteutetun järjestelmän toiminta käyttäjän näkökulmasta

Mahdollisen palvelinfarmin toteuttamiseen on olemassa tarpeen mukaan useita eri vaihtoehtoja. Ensimmäisenä askeleena on web-palvelimen ja muun sovelluslogiikan erottaminen varsinaisista tietokantatoiminnoista. Tämän jälkeen esimerkiksi tietokannan redundanttisuutta vaadittaessa voidaan ottaa käyttöön MongoDB:n tarjoama replikajoukko, jossa tietokannan sisältämä data replikoidaan useammalle tietokantapalvelimelle. Kuviossa 7 on kuvattuna replikajoukon toteuttaminen kolmella tietokantapalvelimella.



Kuvio 7. Replikajoukon toteuttaminen kolmella tietokantapalvelimella (MongoDB Manual n.d)

Skaalautuvuutta tavoiteltaessa voidaan ottaa käyttöön MongoDB:n tarjoama hajautettu klusteriarkkitehtuuri, jossa tietokannan sisältämä data hajautetaan useammalle tietokantapalvelimelle. Tämän lisäksi arkkitehtuuriin kuuluu kyselyjen reitittäjiä ja konfiguraatiopalvelimia. Jos tavoitellaan sekä redundanttisuutta että skaalautuvuutta, voidaan hajautetun klusterin yksittäiset tietokantapalvelimet korvata kuviossa 7 kuvatun ratkaisun kaltaisella replikajoukolla. Tuotantokäytössä datan redundanttisuuden takaamiseksi hajautetun klusterin tulisi sisältää ainakin kolme konfiguraatiopalvelinta, kaksi tai useampaa replikajoukkoa hajautettuina tietokantapalvelimina ja yksi tai useampi kyselyjen reitittäjä. Kuviossa 8 on kuvattuna esimerkki tuotantokäyttöön soveltuvasta hajautetun klusterin toteutuksesta.



Kuvio 8. Tuotantokäyttöön soveltuva hajautetun klusterin toteutus (MongoDB Manual n.d)

Kuvion 8 mukainen tuotantokäyttöön soveltuva hajautetun klusterin ratkaisu voitaisiin toteuttaa esimerkiksi 11 erillisellä palvelimella. Kyseiseen toteutukseen kuuluisi kolme konfiguraatiopalvelinta, kaksi kolmen tietokantapalvelimen replikajoukkoa datan hajautukseen sekä kaksi sovelluspalvelimena ja kyselyjen reitittäjänä toimivaa palvelinta. Esimerkin kaltaisen toteutuksen vaatiman palvelinmäärän takia kolmannen osapuolen tarjoamat pilvipalvelut olisivat tähän luultavasti käytännöllisempi ratkaisu, vaikka onnistuu sen toteutus myös paikallisella palvelinfarmillakin.

4.5 Pilottiympäristön kuvaus

Varsinainen käytännön toteutus toteutettiin aluksi testimielessä yksinkertaisessa yhden palvelinkoneen pilottiympäristössä. Pilottiympäristönä toimi toimeksiantajan toimittama fyysinen palvelinkone. Palvelinkoneessa oli toimeksiantajan toimesta asennettuna Linux-käyttöjärjestelmä, ja se sisälsi suuren määrän uuden järjestelmän kehittämiseen ja testaukseen käytettävissä olevaa mittalaitteiden tuottamaa sensoridataa. Taulukossa 1 on listattuna kyseisen palvelinkoneen tekniset tiedot.

Taulukko 1. Palvelinkoneen tekniset tiedot

Prosessori	Intel Core 2 Quad 64-bit 2.40GHz
Muisti	8GB
Levytila	1TB
Käyttöjärjestelmä	Arch Linux 4.1.6-1-ARCH 64-bit

Kaikki kehittämisessä tarvittavat toiminnot suoritettiin kyseisellä palvelinkoneella, eli se toimi web- ja tietokantapalvelimena sekä suoritti kaikki muut sovellustoiminnot ja tehtävien ajot. Palvelinkoneeseen oltiin yhteydessä ja kaikki järjestelmän testaaminen suoritettiin samassa lähiverkossa sijaitsevalta työasemalta.

Sovellusten puolelta pääosassa oli Node.js-pohjainen web-palvelin ja MongoDB-tietokanta. Myös kaikki muu palvelinpuolen sovelluslogiikka oli Node.js-pohjaista ja asiakaspuolen sovelluslogiikasta vastasi Vue.js-sovelluskehys. Näin ollen koko järjestelmän kehityksen yhdistävänä tekijänä toimi JavaScript-ohjelmointikieli.

4.6 Pilottijärjestelmän toteutus

4.6.1 Ympäristön pystytys

Ympäristön pystyttäminen aloitettiin Linux-käyttöjärjestelmän komentoriviltä. Kaikki komennot suoritettiin root-käyttäjän käyttöoikeuksilla. Aluksi varmistettiin se, että pakettivarastot olivat ajan tasalla, ja päivitettiin kaikki asennetut paketit Arch Linuxin pacman-paketinhallintajärjestelmän avulla komennolla

```
pacman -Syyu
```

Seuraavaksi asennettiin paketinhallintajärjestelmän avulla järjestelmän kannalta oleelliset paketit eli Node.js, sen paketinhallintajärjestelmä NPM, MongoDB ja MongoDB-työkaluja komennolla

```
pacman -S nodejs npm mongodb mongodb-tools
```

Lopuksi luotiin hakemistot MongoDB-tietokannan dataa ja Node.js-projektin tiedostoja varten komennolla

```
mkdir mongodata scan
```

4.6.2 Tietokanta

Tietokannan tapauksessa yksi tärkeistä huomioon otettavista asioista oli sen sisältämän datan käyttämä levytila. MongoDB käyttää oletuksena MMAPv1-nimistä kantamoottoria, mutta versiosta 3.0 lähtien 64-bittisissä asennuksissa on valittavana ollut myös WiredTiger-kantamoottori. Uuden kantamoottorin käytön pitäisi tuoda hyötyjä monenlaisiin sovelluksiin, esimerkiksi sensoridatan analysointia ajatellen, mutta sen tärkeimpänä ominaisuutena on kuitenkin mahdollisuus käyttää zlib-pakkausta tietokantaan tallennettujen dokumenttien pakkaamiseen. Aloitettiin käynnistämällä MongoDB-tietokanta käyttäen aikaisemmin luotua hakemistoa sen datan tallentamiseen sekä otettiin käyttöön WiredTiger-kantamoottori ja määriteltiin oletusasetukseksi zlib-pakkauksen käyttö komennolla

```
mongod --dbpath /root/mongodata --storageEngine wiredTiger
--wiredTigerCollectionBlockCompressor zlib
```

MongoDB:n haluttiin kuitenkin käynnistyvän automaattisesti taustaprosessina järjestelmän käynnistyksen yhteydessä. Otettiin kyseinen toiminto käyttöön komennolla

```
systemctl enable mongodb.service
```

Palvelun käynnistyksen yhteydessä käytetään kuitenkin oletuksena ”/usr/lib/systemd/system/mongodb.service” käynnistysskriptin oletusasetuksia. Vaihdettiin palvelun käynnistämiseen käytetty komento vastaamaan haluttuja asetuksia korvaamalla kyseisen tiedoston rivi seuraavalla rivillä

```
ExecStart=/usr/bin/mongod --quiet --dbpath /root/mongodata --storageEngine
wiredTiger --wiredTigerCollectionBlockCompressor zlib
```

Otettiin käynnistysskriptiin tehdyt muutokset käyttöön komennolla

```
systemctl daemon-reload
```

Muutettiin hakemistojen oikeuksia niin, että palvelun käynnistäminen onnistuu käynnistysskriptin avulla seuraavilla komennoilla

```
chown -R mongodb /root/mongodata
chmod -R +x /root/mongodata
chmod +x /root
```

Lopuksi käynnistettiin MongoDB-tietokantapalvelu komennolla

```
systemctl start mongodb.service
```

Tietokannan toimintaa testattiin käynnistämällä MongoDB-komentorivi komennolla

```
mongo
```

Komentoriviä tarkastellessa huomattiin, että MongoDB varoittaa käytössä olevasta muistinhallintajärjestelmästä nimeltä ”Transparent Huge Pages”, joka voi vaikuttaa negatiivisesti tietokannan suorituskykyyn. Otettiin kyseinen järjestelmä pois käytöstä käynnistyksen yhteydessä ajettavan skriptin avulla. Luotiin scan-hakemistoon tähän tarvittavat tiedostot komennolla

```
cd scan && touch thp.sh thp.service
```

Tiedosto ”thp.sh” on varsinainen bash-skripti kyseisen järjestelmän käytöstä poistamiseen, ja ”thp.service” on käynnistysskripti bash-skriptin ajamiseen käynnistyksen yhteydessä. Annettiin bash-skriptille suoritusoikeudet, ja otettiin käynnistysskripti käyttöön komennoilla

```
chmod +x thp.sh
```

```
systemctl enable /root/scan/thp.service
```

4.6.3 Testidata

Mittalaite tuottaa suoraan JSON-muotoista dataa, minkä ansiosta käytettävissä oleva testidata on yksinkertaisesti käsiteltävissä ja MongoDB-tietokantaan tallennettavissa. Yksi JSON-dokumentti vastaa yhtä mittalaitteen tekemää skannausta. Koko kehityksen ja testauksen ajaksi otettiin käsittelyyn yhden mittalaitteen tuottama testidata, joka tarkoitti noin 137 tuhatta JSON-dokumenttia. Kaikki mittalaitteen tuottamat dokumentit olivat tallennettuina gzip-pakattuina JSON-tiedostoina levytilan säästämiseksi. Mittalaitteen tuottamat dokumentit sisältävät useita eri kenttiä, mutta taulukossa 2 on esitelty tämän työn kannalta niistä tärkeimmät.

Taulukko 2. Testidatan tärkeimmät kentät

Kentän nimi	**Kuvaus**
timestart, timestop	Skannauksen alkua ja loppua vastaavat aikaleimat
scriptname	Lua-skriptin nimi, jonka toimesta skannaus toteutettiin
scriptMD5	MD5-tiiviste Lua-skriptistä
serial	Skannauksen tehneen mittalaitteen sarjanumero
sweeps	Taulukko skannauksen sisältämistä pyyhkäisyistä

Ensimmäisenä vaiheena tietokantaan tallentamista varten, gzip-pakatut tiedostot piti purkaa takaisin raakaan JSON-muotoonsa. Luotiin yksinkertainen bash-skripti mittalaitteen tuottamien tiedostojen läpikäymiseen. Skriptin avulla joko purettiin gzip-pakatut tiedostot tai kopioitiin JSON-muotoiset tiedostot uuteen niille varattuun hakemistoon.

Vaikka testidata olikin valmiiksi JSON-muotoista, tuotti dokumenteissa käytetty datan tallennusmuoto vaikeuksia kyseistä dataa analysoitaessa MongoDB:n aggregaatioiden avulla. Ongelma oli yksinkertaistettuna se, että pyyhkäisyjen sisältämä data oli tallennettuna useisiin, indeksien avulla toisiinsa yhteydessä oleviin, yhtä mitattavaa arvoa vastaaviin taulukoihin. Tämä rakenne ei soveltunut hyvin kyseisen datan analysointiin, joten se piti muuttaa sisäkkäisiä dokumentteja hyödyntävään muotoon.

Luotiin Node.js-skripti kyseisen toiminnon suorittamiseen. Skripti kävi läpi kaikki tiedostot JSON-dokumentteja sisältävästä hakemistosta, muuttaen ne uuteen haluttuun muotoon ja tallentaen ne uuteen niille varattuun hakemistoon. Tämän jälkeen testidata voitiin vihdoinkin tallentaa MongoDB-tietokantaan. Samaan skriptiin yhdistettiin toiminto, joka käy läpi kaikki uuteen muotoon tallennetut JSON-dokumentit ja tallentaa ne tietokantaan käyttäen MongoDB-työkalujen sisältämää komentoa. Jokainen tiedostopolkua vastaava JSON-dokumentti tallennettiin test-tietokannan testi-kokoelmaan komennolla

```
mongoimport --host 127.0.0.1 --db test --collection testi --file <tiedostopolku>
```

JSON-dokumenttien muuttaminen uuteen muotoon ja niiden tallentaminen tietokantaan kesti useita tunteja. Näiden prosessien päätyttyä tarkistettiin operaation onnistuminen ajamalla seuraava komento MongoDB-komentoriviltä

```
db.testi.stats()
```

Tuloksena nähtiin, että testi-kokoelma sisälsi 137089 dokumenttia, niiden viemän levytilan ja muuta yleistä tietoa kokoelmasta. MongoDB luo jokaiselle dokumentille automaattisesti uniikin _id-kentän ja sitä vastaavan indeksin. Lopuksi luotiin hakujen nopeuttamiseksi indeksit kaikille tärkeimmille kentille seuraavilla MongoDB-komentorivin komennoilla

```
db.testi.createIndex({ scriptname: 1 })
db.testi.createIndex({ scriptMD5: 1 })
db.testi.createIndex({ serial: 1 })
db.testi.createIndex({ timestart: 1 })
db.testi.createIndex({ timestop: 1 })
```

4.6.4 Node.js-sovellus

Node.js-sovelluksen kehittäminen aloitettiin luomalla scan-hakemistoon package.json-tiedosto komennolla

```
npm init
```

Kyseisen komennon seurauksena käyttäjältä kysytään yleistä tietoa sovelluksesta, kuten sen nimi, versio ja käynnistysskripti. Nämä tiedot tallennetaan package.json-tiedostoon yhdessä moduuliriippuvuuksien kanssa. Asennettiin varsinaiset sovelluksen kannalta tärkeät moduulit ja tallennettiin niitä vastaavat riippuvuudet package.json-tiedostoon seuraavalla komennolla

```
npm install mongodb express body-parser --save
```

Asennetut moduulit olivat MongoDB:n Node.js-ajurit, Express.js web-palvelin ja web-palvelimen lisäosa HTTP-pyyntöjen sisällön tulkintaan. Seuraavaksi asennettiin pelkästään sovelluksen kehittämisvaiheessa tarvittavat moduulit ja tallennettiin niitä vastaavat riippuvuudet erilliseen package.json-tiedoston osioon komennolla

```
npm install bootstrap browserify gulp gulp-uglify vinyl-buffer vinyl-source-stream
vue vue-resource vue-router vueify --save-dev
```

Asennetut moduulit voidaan jakaa kahteen osaan: asiakaspuolen sovelluskehyksiin ja tehtävänajoon liittyviin moduuleihin. Asiakaspuolen sovelluskehykset, kuten Bootstrap, Vue.js ja siihen liittyvät moduulit asennettiin tässä tapauksessa kehitysvaiheen riippuvuuksiin, koska niihin ei viitattu suoraan, vaan ne pienennettiin ja yhdistettiin tehtävänajoprosessien kautta yhdeksi asiakkaalle tarjottavaksi staattiseksi tiedostoksi. Loput asennetut moduulit olivat tehtävien ajoon tarkoitettu Gulp.js ja sen käyttämät lisäosat.

Lopuksi asennettiin vielä Gulp.js globaalisti, mikä mahdollisti tehtävien ajamisen suoraan komentoriviltä

```
npm install -g gulp
```

4.6.5 Järjestelmän tiedostot

Seuraavana esitellään tärkeimmät valmiin Node.js-sovelluksen sisältämät tiedostot ja hakemistot.

”server.js” on Node.js-sovelluksen alkupiste, joka ajamalla koko sovellus käynnistetään. Se toimii web-palvelimena, sisältää kaikki sen tuntemat reitit, hoitaa yhteyden MongoDB-tietokantaan ja suorittaa reittien perusteella halutut tietokantakyselyt käyttäen scan-moduulin tarjoamaa rajapintaa.

“scan.js” on moduuli, joka tarjoaa sovellusrajapinnan kaikkien sovelluksessa tarvittavien tietokantatoimintojen suorittamiseen.

”index.html” on ainoa varsinainen asiakkaalle tarjoiltava web-sivu, koska sovellus toimii yhden web-sivun sovelluksen periaatteella. Se toimii pohjana asiakaspuolen sovelluskehysten toiminnalle.

”main.js” on asiakaspuolen JavaScript-toimintojen alkupiste, josta tehtävänajossa aloitetaan JavaScript-tiedostojen yhdistys- ja pienennysprosessi. Se alustaa ja käynnistää Vue.js-sovelluksen sekä määrittää sen käyttämät moduulit ja asiakaspuolen reitityksen.

”components” on hakemisto, joka sisältää kaikki Vue.js-sovelluksen käyttämät komponentit. Komponentit yhdistetään muiden JavaScript-tiedostojen kanssa yhdeksi staattiseksi tiedostoksi tehtävänajoprosessin seurauksena.

”public” on hakemisto, joka sisältää kaikki lopulliset asiakkaalle tarjoiltavat staattiset tiedostot, kuten JavaScript- ja CSS-tiedostot.

4.6.6 Järjestelmän rajapinnat

Järjestelmässä on käytössä kaksi erilaista rajapintaa: asiakkaan tai asiakassovelluksen käytössä oleva web-rajapinta ja scan-moduulin tarjoama sovellusrajapinta tietokantakyselyjen suorittamiseen. Web-rajapinnalla tarkoitetaan web-palvelimen ylläpitämiä reittejä, joihin asiakas voi olla yhteydessä HTTP-pyyntöjen avulla. Toteutetussa järjestelmässä asiakaspuolen Vue.js-sovellus käyttää yksinkertaisuuden vuoksi XHR-pyyntöjä (XMLHttpRequest) web-rajapinnan kanssa kommunikoimiseen, mutta tämä voidaan tulevaisuudessa tarpeen mukaan korvata esimerkiksi WebSocket-teknologialla. Taulukossa 3 on listattuna toteutetun web-rajapinnan tarjoamat reitit ja niitä vastaavat toiminnot.

Taulukko 3. Sovelluksen web-rajapinta

Metodi	**Reitti**	**Parametrit**	**Toiminto**
GET	/	-	Tarjoilee asiakkaalle sovelluksen etusivun.
GET	/metadata	-	Palauttaa metadataa sisältävän dokumentin JSON-muodossa.
GET	/scan/:id	id	Palauttaa yhden JSON-dokumentin sen _id-kentän perusteella.
POST	/scan/find	query	Hakee dokumentteja annettujen hakuehtojen perusteella ja palauttaa tulokset JSON-dokumentteja sisältävänä taulukkona.
POST	/scan/filter	query	Suorittaa monimutkaisempaa datan analysointia annettujen haku- ja suodatusehtojen perusteella ja palauttaa tulokset JSON-dokumentteja sisältävänä taulukkona.
POST	/scan/insert	body	Tallentaa annetun JSON-dokumentin tietokantaan ja palauttaa sen _id-kentän arvon.

Scan-moduulin tarjoamalla sovellusrajapinnalla tarkoitetaan scan-moduulin eli scan.js-tiedoston ulkopuolelle paljastamia funktioita. Käytännössä tämä tarkoittaa sitä, että kyseiset funktiot saadaan käyttöön missä tahansa Node.js-tiedostossa sisällyttämällä siihen scan-moduuli. Scan-moduulin funktioiden pääasiallisena tarkoituksena on suorittaa erilaisia tietokantatoimintoja. Toteutetussa järjestelmässä näitä funktioita käytetään sovelluksen web-rajapinnan yhteydessä, missä suurin osa reiteistä vastaa jotakin scan-moduulin tietokantatoimintoa. Taulukossa 4 on listattuna toteutetun scan-moduulin tarjoamat toiminnot.

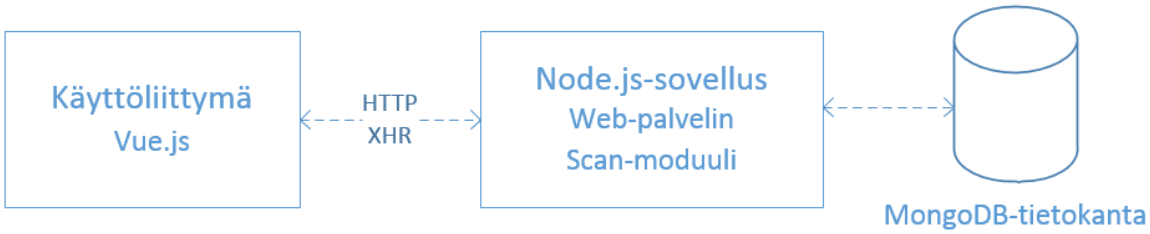
Taulukko 4. Scan-moduulin sovellusrajapinta

Funktio	**Toiminto**
find(db, query, [mask], callback)	Hakee dokumentteja annettujen hakuehtojen perusteella ja suodattaa palautettavat kentät annetun maskin avulla. Palauttaa tulokset dokumentteja sisältävänä taulukkona.
findById(db, id, [mask], callback)	Hakee dokumentin annetun _id-kentän arvon perusteella ja suodattaa palautettavat kentät annetun maskin avulla.
filter(db, query, callback)	Suorittaa monimutkaisempaa datan analysointia annettujen haku- ja suodatusehtojen perusteella. Palauttaa tulokset dokumentteja sisältävänä taulukkona.
metadata(db, callback)	Palauttaa metadataa sisältävän dokumentin.
generateMetadata(db, callback)	Generoi metadataa tietokannasta ja tallentaa tulokset sille varattuun kokoelmaan.
insert(db, doc, callback)	Tallentaa annetun JSON-dokumentin tietokantaan.

Toteutetun järjestelmän toiminta on käyttäjän näkökulmasta yksinkertainen. Käyttäjä ottaa palvelimen kanssa samassa lähiverkossa sijaitsevan työaseman selaimella yhteyden web-palvelimeen, suorittaa web-käyttöliittymän avulla haluamansa toiminnon ja saa sen tuottamat tulokset näkyville käyttöliittymässä. Teknologian näkökulmasta tämän prosessin taustalla on kuitenkin joukko erinäisiä toimintoja.

Käyttäjä lähettää HTTP-pyynnön Node.js-pohjaiselle web-palvelimelle, joka tarjoilee käyttäjälle vastauksena Vue.js-pohjaisen yhden sivun web-sovelluksena toimivan käyttöliittymän. Kun käyttäjä suorittaa jonkin toiminnon käyttöliittymän avulla, Vue.js lähettää XHR-pyynnön toimintoa vastaavalle reitille web-palvelimen tarjoamassa rajapinnassa. Jokaisen tietokantatoiminnon yhteydessä reitti kutsuu jotakin scan-moduulin tarjoaman rajapinnan funktiota, joka taas vuorostaan suorittaa jonkin MongoDB-tietokantatoiminnon. Web-palvelin saa tietokantatoiminnon tuottamat tulokset takaisin callback-funktion kautta ja lähettää ne edelleen JSON-muodossa XHR-pyynnön vastauksena käyttöliittymälle. Käyttöliittymässä Vue.js tulkitsee JSON-

muotoisen vastauksen ja muotoilee sen halutulla tavalla käyttöliittymässä esitettäväksi. Kuviossa 9 on vielä esitettynä yksinkertainen kuvaus toteutetun järjestelmän sisäisestä kommunikaatiosta.



Kuvio 9. Toteutetun järjestelmän sisäinen kommunikaatio

5 Pohdinta

Opinnäytetyöllä oli kaksi varsinaista tavoitetta. Teoriaosuudessa oli tarkoitus tutkia ja selvittää mitä mahdollisuuksia pilvipalvelut voisivat tarjota toimeksiantajalle ja heidän analysaattorituotteelleen. Käytännön osuuden tavoitteena oli kehittää toimeksiantajan analysaattorilaitteiden rinnalle toimiva järjestelmä niiden tuottaman mittausdatan analysointia varten.

Teoriaosuuden tuloksena saatiin tuotettua toimeksiantajalle yleiskuvan antava tietopaketti pilvipalveluista, niihin liittyvistä käsitteistä ja niiden tarjoamista mahdollisuuksista. Teoriaosuudessa esiteltiin yleistietoa pilvipalveluista ja niiden käyttötarkoituksista, vertailtiin pilvipalveluiden käyttöönotto- ja palvelumalleja toisiinsa, selvennettiin pilvipalveluihin liittyviä käsitteitä, annettiin esimerkkejä tarjolla olevista pilvipalveluista ja vertailtiin pilvipalveluja paikalliseen palvelinfarmiin perustuviin ratkaisuihin.

Käytännön osuuden tuloksena saatiin toteutettua onnistuneesti yksinkertainen järjestelmä analysaattorilaitteiden tuottaman mittausdatan analysointiin toimeksiantajan toimittamassa pilottiympäristössä. Käytännön osuudessa esiteltiin toteutuksessa käytetyt teknologiat, kuvattiin varsinaisen analysaattorituotteen ja sen ympärillä toimivan nykyisen järjestelmän toiminta, laadittiin konsepti mahdollisesta paikallisen palvelinfarmin ratkaisusta, käytiin läpi varsinaisen sovelluksen toteutus pilottiympäristössä sekä esiteltiin toteutetun järjestelmän tarjoamat rajapinnat ja niiden välinen kommunikaatio.

Käytännön osuudessa toteutettu järjestelmä tarjoaa runsaasti mahdollisuuksia jatkotoimenpiteitä ajatellen. Yhdellä palvelinkoneella toteutettu järjestelmä voidaan tarpeen mukaan, esimerkiksi redundanttisuutta tai datan hajautusta vaadittaessa, laajentaa useamman palvelinkoneen paikalliseksi palvelinfarmiksi. Järjestelmän siirtäminen kolmannen osapuolen palveluntarjoajan pilvipalveluihin pitäisi olla yksinkertaista, jos se tulee mahdolliseksi tulevaisuudessa. Tietokannan ja sovelluksen suorittamia toimintoja voidaan luultavasti optimoida ja sovellukseen tullaan varmasti vielä lisäämään uusia toimintoja. Käyttöliittymän ulkoasua voidaan kohentaa tämänhetkiseen prototyyppiin verrattuna ja sovelluksen käyttämiä teknologioita voidaan vaihtaa tarpeen vaatiessa.

Lähteet

Arregoces, M. & Portolani, M. 2003. Data Center Fundamentals: Understand Data Center Network Design and Infrastructure Architecture, Including Load Balancing, SSL, and Security.

Buyya, R., Broberg, J. & Goscinski, A. 2011. Cloud Computing: Principles and Paradigms.

Buyya, R., Vecchiola, C. & Selvi, S. 2013. Mastering Cloud Computing: Foundations and Applications Programming.

Chapman, C. 2014. The ultimate guide to Bootstrap. Viitattu 11.11.2015. http://www.webdesignerdepot.com/2014/10/the-ultimate-guide-to-bootstrap/.

Chenkie, R. 2015. Build an App with Vue.js: A Lightweight Alternative to AngularJS. Viitattu 10.11.2015. https://scotch.io/tutorials/build-an-app-with-vue-js-a-lightweight-alternative-to-angularjs.

Cloud Deployment Models. N.d. Arcitura Education Inc. Viitattu 11.6.2015. http://whatiscloud.com/cloud_deployment_models/index.

Community Clouds. N.d. Arcitura Education Inc. Viitattu 16.6.2015. http://whatiscloud.com/cloud_deployment_models/community_clouds.

Company. N.d. Environics Oy. Viitattu 1.6.2015. http://www.environics.fi/company/.

Daniela. 2015. SQL Versus NoSQL: What are the Differences and How Do You Choose? Viitattu 5.11.2015. http://blog.bigstep.com/nosql/sql-versus-nosql-differences-choose/.

Furht, B. & Escalante, A. 2010. Handbook of Cloud Computing.

Gackenheimer, C. 2013. Node.js Recipes: A Problem-Solution Approach.

GFI Software. 2010. On-premise vs. Cloud-based Solutions. https://www.gfi.com/whitepapers/Hybrid_Technology.pdf.

Hill, R., Hirsch, L., Lake, P. & Moshiri, S. 2013. Guide to Cloud Computing: Principles and Practice.

Hybrid Clouds. N.d. Arcitura Education Inc. Viitattu 11.6.2015. http://whatiscloud.com/cloud_deployment_models/hybrid_clouds.

JSON and BSON. N.d. MongoDB Inc. Viitattu 5.11.2015. https://www.mongodb.com/json-and-bson.

Julkunen, V. 2015. Mittaussovelluksen suunnittelu DMS-AIMS2 ioniliikkuvuusspektrometrille. Opinnäytetyö. Mikkelin ammattikorkeakoulu, ympäristöteknologian koulutusohjelma. Viitattu 11.11.2015. http://urn.fi/URN:NBN:fi:amk-201505229691.

Kavis, M. 2014. Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS).

Liew, Z. 2015. Gulp for Beginners. Viitattu 10.11.2015. https://css-tricks.com/gulp-for-beginners/.

Mardan, A. 2014. Practical Node.js: Building Real-World Scalable Web Apps.

Mathew, S. 2014. Overview of Amazon Web Services. http://aws.amazon.com/whitepapers/overview-of-amazon-web-services/.

MongoDB Manual. N.d. MongoDB Inc. Viitattu 5.11.2015. https://docs.mongodb.org/manual/.

NoSQL Database Explained. N.d. MongoDB Inc. Viitattu 5.11.2015. https://www.mongodb.com/nosql-explained.

Other Deployment Models. N.d. Arcitura Education Inc. Viitattu 16.6.2015. http://whatiscloud.com/cloud_deployment_models/other_deployment_models.

Private Clouds. N.d. Arcitura Education Inc. Viitattu 11.6.2015. http://whatiscloud.com/cloud_deployment_models/private_clouds.

Public Clouds. N.d. Arcitura Education Inc. Viitattu 11.6.2015. http://whatiscloud.com/cloud_deployment_models/public_clouds.

Rouse, M. & Gibilisco, S. 2013. Virtual server farm. Viitattu 3.11.2015. http://whatis.techtarget.com/definition/virtual-server-farm.

Vold, N. 2012. Cloud basics – Deployment models. Visma Corporate Blog 12.3.2012. Viitattu 4.8.2015. http://www.visma.com/blog/cloud-basics-deployment-models/.

What is Cloud Computing? N.d. Interoute Communications Ltd. Viitattu 8.6.2015. http://www.interoute.com/cloud-article/what-cloud-computing.

What is IaaS? N.d. Interoute Communications Ltd. Viitattu 8.6.2015. http://www.interoute.com/what-iaas.

What is PaaS? N.d. Interoute Communications Ltd. Viitattu 8.6.2015. http://www.interoute.com/what-paas.

What is SaaS? N.d. Interoute Communications Ltd. Viitattu 8.6.2015. http://www.interoute.com/what-saas.

Yeluri, R. & Castro-Leon, E. 2014. Building the Infrastructure for Cloud Security: A Solutions View.