Haaga-Helia
ammattikorkeakoulu Oy

Pilvipalveluna tarjottavan dokumenttivaraston hyödyntäminen web-sovelluksessa

Marianne Sirén

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2018

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä

5.5.2018

Tekijä(t)
Marianne Sirén

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Pilvipalveluna tarjottavan dokumenttivaraston hyödyntäminen web-sovelluksessa

Sivu- ja liitesivumäärä
40

Opinnäytetyön otsikko englanniksi
Exploiting cloud-based document storage in web application

Opinnäytetyön tavoitteena oli kehittää web-sovellus, jossa käyttäjät voivat muodostaa ryhmiä, ja jakaa ryhmän sisäisesti eri formaatissa olevia dokumentteja. Dokumentit ladataan selaimesta suoraan dokumenttivarastoon. Ryhmään kuuluvat käyttäjät voivat ladata dokumentit itselleen.

Teoriaosuudessa perehdytään pilvipalveluihin. Aluksi käydään läpi pilvipalveluita yleisellä tasolla, jonka jälkeen tarkastellaan miten dokumentteja hallinnoidaan sovelluksissa, mitä tulee huomioida kun valitaan itselle sopivaa pilvipalvelun tarjoajaa, sekä tutustutaan yleisimpiin tietoturva ja tietosuoja haasteisiin.

Toiminnallisessa osuudessa perustellaan valitut teknologiat, sekä käydään yksityiskohtaisesti läpi ohjelmistokehityksen vaiheet. Osio sisältää tärkeimpiä koodeja, sekä kuvia web-sovelluksen näkymistä. Osuuden lopussa tarkastellaan sovelluksen tietoturvaa.

Opinnäytetyöstä rajattiin pois sovelluksen julkaiseminen ja käyttäjien henkilökohtaiset dokumenttivarastot.

Pohdinta on viimeinen osio, ja siellä käydään läpi projektin tavoitteisiin pääsemistä, mitä haasteita matkan varrella tuli ja mitä opittiin. Lopuksi käsitellään jatkokehitysideoita.

Asiasanat
pilvipalvelut, dokumentit, ohjelmistokehitys, web-ohjelmointi

Sisällys

1 Johdanto

Opinnäytetyön tavoitteena oli toteuttaa web-sovellus, jonne käyttäjät voivat rekisteröityä millä tahansa sähköposti-osoitteella ja luoda käyttäjien kanssa ryhmiä. Ryhmän jäsenet voivat jakaa dokumentteja keskenään. Opiskelujeni aikana ja ryhmätöitä tehneenä olen huomannut, että tällaiselle web-sovellukselle olisi ollut tarvetta.

Aihe on kiinnostava ja ajankohtainen, dokumenttien hallinnointi on osa modernia web-kehitystä. Dokumenttien hallinta järjestelmässä vaatii tehokkuutta ja mahdollisuutta skaalautua. Siksi dokumenttien tallentamisen osalta hyödynnetään pilvipalvelua, johon dokumentit ladataan suoraan selaimesta, raahaa ja pudota toiminnon kautta. Teoriaosuudessa perehdytään pilvipalveluiden käyttämiseen apuna dokumenttien hallinnoinnissa ja tallentamisessa, sekä minkälaisia tietoturva- ja tietosuoja-seikkoja tulee huomioida.

Web-sovellus toteutettiin Ruby on Rails -ohjelmistokehystä ja Amazon S3 -pilvipalvelua käyttäen. Sovellus on tarkoitettu käytettäväksi päätelaitteella, sillä yleensä dokumenttien muokkaaminen tehdään tietokoneella, jossa tarvittavat lisenssit dokumenttien muokkaamiseen ovat. Web-sovellus skaalautuu myös mobiililaitteelle ja ryhmän dokumentit voi ladata niin puhelimeen kuin tietokoneelle.

Opinnäytetyön tutkimusongelmina oli selvittää, miten pilvipalveluna toimivaa dokumenttivarastoa voidaan hyödyntää dokumenttien tallentamiseen, sekä mitä tietoturva- ja tietosuoja-haasteita dokumenttivarastolla voi olla.

Tästä työstä oli rajattu pois sovelluksen julkaiseminen, sekä käytetyn ohjelmointikehyksen ja ohjelmointikielen tarkemman toiminnallisuuden avaaminen. Opinnäytetyössä oli keskitytty organisaation näkemykseen dokumenttivarastoista, henkilökohtaiset dokumenttivarastot olivat rajattu pois.

2 Pilvipalvelut apuna dokumenttien hallinnoinnissa

Aluksi osiossa käsitellään, mitä pilvipalveluilla yleisesti ottaen tarkoitetaan, jonka jälkeen paneudutaan dokumenttien hallinnoimiseen järjestelmässä ja pilvipalveluiden tarjoamiin dokumenttivarastoihin. Lopuksi käydään läpi, miten valita pilvipalvelun tarjoaja dokumenttien hallinnoimiseen, sekä mitä tietoturva ja tietosuoja asioita tulee huomioida.

2.1 Pilvipalveluista yleisesti

Pilvipalvelulla tarkoitetaan yleiskielessä puhuttuna tietokonekapasiteettia, sovelluksia tai muita palvelusuoritteita, jotka on hankittu internetistä. Ammattilaisten kesken ilmiössä on kyse siitä, että voidaan luopua fyysisistä konesaleista. (Heino 2010, 32.)

Kun tarkastellaan pilvipalveluiden historiaa, eri lähteiden mukaan ensimmäiset ajatukset pilvipalveluiden kaltaisesta toiminnasta ovat peräisin 1960-luvulta. Heino (2010, 33) mainitsee kirjassaan Douglas Parkhillin, joka esitteli lähes kaikki pilvitoiminnan ominaisuudet, kuten provisioinnin ja rajoittamattomuuden, esitelmässään Challenge of the Computer Utility vuonna 1966. Provisioinnilla tarkoitetaan laitteiden ja ohjelmistojen valmistelua etukäteen niin, että ne saadaan käyttöön automatisoiduilla rutiineilla (Techopedia). ComputerWeekly (2009) mainitsee artikkelissaan JCR Lickliderin, joka vastasi Advanced Research Projects Agency Network -tietoverkon kehittämisestä vuonna 1969. JCR Lickliderin visiona oli tuoda ohjelmat ja data kaikkien ihmisten saataville eri puolille maailmaa. (ComputerWeekly 2009.) Sekä Heino (2010, 33), että ComputerWeekly (2009) mainitsevat lisäksi 1960-luvulta tekoälyn tutkijan ja LISP-ohjelmointikielen kehittäjänä tunnetun John McCarthyn. McCarthyn tavoitteena oli, että jonain päivänä tietokonekapasiteettia olisi saatavilla kuten vettä tai sähköä. Kuitenkin vasta 1990-luvulta alkaen voidaan alkaa puhua pilvipalveluiden tarjoamisesta suurille massoille. Pilvipalveluita tarjoava Salesforce.com saapui markkinoille vuonna 1999, toimien alan pioneerina. Heidän konseptinsa oli tarjota yritysten ohjelmistoja web-sivujen kautta. Vuonna 2002 alalle saapui seuraava suuri palveluntarjoaja, Amazon Web Services, lyhyemmin AWS. AWS tarjosi erilaisia pilvipalveluita, kuten varastointia ja laskentaa. Suuri askel tapahtui 2009 kun Google ja muut yritykset alkoivat tarjota web-selaimella käytettäviä sovelluksia. (Heino 2010, 33; ComputerWeekly 2009.)

Pilvipalvelut on jaettu kolmeen päätyyppiin teknisestä toteutustavasta riippuen. Toteutustavalla määritetään, miten koneistoon liitytään, sekä minkälaisia tietojenkäsittelytehtäviä pilvipalvelusta saadaan. Päätyypit ovat IaaS (Infrastructure as a Service), PaaS (Platform

as a Service), sekä SaaS (Software as a Service). (Heino 2010, 51-54; SearchCloudComputing 2014.)

IaaS -mallissa pilvipalvelun tarjoaja tarjoaa virtuaalisen konesalin tai konesaleja, eli palvelimen ja varaston. Asiakas asentaa itse tarvitsemansa käyttöjärjestelmän ja sovellukset pilvipalveluun. IaaS tarjoaa rajapinnan, jonka kautta pilvipalvelua käytetään. Loppukäyttäjä käyttää sovellusta yleensä selaimen kautta. Amazon Web Services on tunnetuin IaaS -palveluntarjoaja. (Heino 2010, 51-54; SearchCloudComputing 2014.) Yhtenä esimerkkinä IaaS -palvelun käytöstä on tämän opinnäytetyön tuotos, jossa IaaS -palvelua käytetään järjestelmän dokumenttien tallentamiseen.

PaaS -mallissa palvelun tarjoaja tarjoaa kehitystyökaluja tai kapasiteettia virtuaalisessa palvelinympäristössä. Työkaluihin on pääsy rajapinnan, portaalin tai gateway-ohjelmiston kautta. Asiakkaan on itse tehtävä tai teetettävä koneistoa hyödyntävät sovellukset, joita loppukäyttäjä käyttää selaimen kautta. Esimerkiksi Google ja Microsoft tarjoavat PaaS-toteutuksia. (Heino 2010, 51-54; SearchCloudComputing 2014.) Tyypillinen esimerkki PaaS -mallista on räätälöidyt ohjelmistot, joita teetetään esimerkiksi organisaation yksilöllisiin tarpeisiin. Ohjelmistotalo toteuttaa ohjelmiston ja ohjelmisto toimii PaaS -palvelussa, jonka kautta ohjelmiston käyttäjät pääsevät ohjelmistoa käyttämään.

SaaS on jakelumalli, jossa asiakas hankkii itselleen pelkän sovelluksen ja sovellus jaetaan internetin välityksellä eteenpäin. SaaS -palveluntarjoaja huolehtii kaikesta muusta. SaaS toimittaa ohjelmistoja internetin välityksellä. SaaS -ohjelmistoja voidaan käyttää mistä tahansa laitteesta, jossa on internetyhteys. SaaS -palveluntarjoajia on useita, esimeriksi Salesforce.com. Esimerkkinä SaaS-palvelusta on Microsoft 365-ohjelmistot, jotka tarjotaan SaaS -palvelun kautta. (Heino 2010, 51-54; SearchCloudComputing 2014.)

Erilaisten päätyyppien lisäksi vaikuttaa se, halutaanko käyttää julkista (public cloud), yksityistä (private cloud) vai hybridipilveä (hybrid cloud). Julkinen pilvi tarkoittaa pilvipalvelukoneiston käyttöä internetyhteyden kautta. Pilvipalveluntarjoaja vastaa koneistossa olevista laitteista ja omistamisen kustannuksista, sekä järjestää asiakkaan tarvitsemat osoite- ja nimipalveluresurssit. Julkisessa pilvessä maksetaan yleensä vain sen käytöstä, esimerkiksi tallennustilasta ja kaistanleveydestä joita asiakas kuluttaa. Amazon Web Services, Microsoft Azure ja Google Cloud Platform ovat esimerkkejä julkisesta pilvestä. Yksityiset pilvipalvelut toimitetaan asiakkaan datakeskuksesta organisaation sisäisille käyttäjille. Yksityinen pilvi on asiakkaan oman lähiverkon, tai muun luotetun verkon kautta käytettävä pilvipalvelukoneisto. Tällöin asiakas vastaa itse pilvipalvelukoneiston järjestämisestä ja ylläpidosta, sekä vastaa omistamisen kustannuksista. Tämä mahdollistaa datakeskuksen

paikallisen hallinnan, valvonnan ja tietoturvan, tarjoten samalla pilvipalvelun monipuolisuuden. Hybridipilvi on sekoitus julkista- ja yksityistä pilveä. Hybridipilvi mahdollistaa julkisen pilvipalvelun infrastruktuurin hyödyntämisen, mutta pitää huolen kriittisten tietojen hallinnan säilymisestä organisaatiolla. (Heino 2010, 54-56; SearchCloudComputing 2014.)

2.2 Suurimmat palveluntarjoajat

Pilvipalveluiden tarjoajia on tällä hetkellä useita. Palveluntarjoajien käyttöehdot, palveluiden tarjonta, palvelimien sijainnit ja hinnat voivat poiketa toisistaan. Suurimpien palveluntarjoajien lisäksi alalta löytyy useita pienempiä palveluntarjoajia. (Datamation 2017a.)

Julkisista pilvipalveluista puhuttaessa suurimmat yritykset ovat suhteellisen selkeät. Fortune (2017) on listannut Gartnerin vuotuiseen listaan perustuen kolme suurinta pilvipalveluun tarjoajaa:

1. Amazon Web Services
2. Microsoft
3. Google.

Datamation (2017a) on listannut 50 johtavaa julkisen tai yksityisen pilvipalvelun tarjoajaa, laittamatta niitä tarkkaan järjestykseen. Datamation kuitenkin huomauttaa, että listan ensimmäiset sijat on jaettu karkeasti markkinaosuuden mukaan. Listan neljä ensimmäistä, ovat:

1. Amazon Web Services
2. Microsoft
3. Google
4. IBM.

Maailmanlaajuisesti julkisten pilvipalveluiden markkinoiden odotetaan kasvavan. Vuonna 2017 julkisen pilvipalveluiden markkinoiden liikevaihto oli 153,5 miljardia euroa, ja vuodelle 2018 odotetaan 21,4 % kasvua, aina 186,4 miljardiin asti. Nopeimmin kasvaa tällä hetkellä IaaS -palvelut, joiden odotetaan kasvavan 35,9 % vuonna 2018. (Gartner 2018.) Yksityisten pilvipalveluiden markkinat eivät ole yhtä suuret kuin julkisten pilvipalveluiden markkinat (Datamation 2017b). Yhtä tarkkoja liikevaihdon lukuja yksityisten pilvipalveluiden osalta ei ollut saatavilla. Kun tarkastellaan yksityisen pilvipalvelun tarjoajia, Wikibonin (2016) mukaan vuoden 2015 markkinoiden neljä suurinta pilvipalveluiden tarjoajaa ovat:

- Hewlett Packard Enterprise
- Oracle
- VMWare
- EMC.

Useimmat pilvipalvelun tarjoajat tarjoavat erilaisia pilvipalveluita. Kuvassa 1 on näytetty listana SDXCentralin mukaan joidenkin merkittävimpien pilvipalveluntarjoajien tarjoamia

palveluita. Toisaalta jotkin palveluntarjoajat ovat erikoistuneet tarjoamaan tietyn tyyppisiä palveluita, kuten esimerkiksi tietoturvallisia IaaS-palveluita terveydenhuollon järjestelmille. (SDXCentral).

Cloud Service Provider	IAAS	PAAS	SAAS
Amazon	X	X	
Century Link	X	X	
Google	X	X	X
IBM	X	X	X
Microsoft	X	X	X
Rackspace	X	X	
Salesforce.com		X	X
SAP	X	X	X
Verizon Terremark	X	X	

Kuva 1. Suurimpia palveluntarjoajia ja heidän palveluitaan (SDXCentral).

2.3 Dokumenttien hallinnointi järjestelmässä

The Art of Service -kirjassa (2010, 143-144) on lueteltuna, mitä asioita tulee ottaa huomioon, kun suunnitellaan toimivaa dokumenttien hallintaa järjestelmään. Listassa on otettu huomioon erilaiset skenaariot käyttäjän käyttökokemuksesta tietoturvaan ja tietoturvasta virhetilanteiden hallintaan:

- Fyysinen infrastruktuuri ja vahvuuksien hallinta
- Kapasiteetin hallinta
- Kokoonpanon hallinta
- Datamigraatiot
- Tapahtuman hallinta
- Suorituskyvyn hallinta
- Saatavuuden hallinta
- Turvallisuuden hallinta
- Varmuuskopiointi ja palautus.

Fyysinen infrastruktuuri on jokaisen dokumenttivaraston perusta. Vahvuuksien hallinta huolehtii fyysisestä infrastruktuurista tarjoten prosesseja ja menettelyitä. Fyysinen infrastruktuuri sisältää sekä määrittelyn, että taloudellisen puolen. Määrittelyllä tarkoitetaan esimerkiksi palvelimien fyysistä sijaintia. Taloudellisella puolella tulee ottaa huomioon dokumenttivaraston fyysisen infrastruktuurin kulut. Kapasiteetin hallinta pitää huolen siitä, paljonko kapasiteettiä on käytetty ja kuinka paljon sitä on vielä jäljellä. Yksi suurimmista kapasiteetin hallinnan haasteista on se, että miten huolehtia mahdollisimman kulutehokkaasti riittävästä kapasiteetin saatavuudesta tulevaisuudessa. Jatkuvasti muuttuvat teknologiaympäristöt aiheuttavat toisen suuren haasteen kapasiteettien hallintaan. (The Art of Service 2010, 143-146.)

Kokoonpanon hallinta keskittyy hoitamaan muutokset, joita tehdään esimerkiksi laitteistoon tai ohjelmistoihin. Se luo ja ylläpitää komponentteja, jotka luovat infrastruktuurin. Kokoonpanon hallinta huolehtii komponenttien konfiguraatioista verkossa. Datamigraatiot tarjoavat menettelytapoja datan siirtämiseen palvelinten ja laitteiden välillä. Datamigraatioiden yhtenä huolenaiheena on mahdollisesti muuttuva teknologiaympäristö – esimerkiksi uusia tietotyyppejä tulee. Tapahtuman hallinta huolehtii siitä, mitä tapahtuu järjestelmän virhetilanteissa. Virhetilanteissa tarvitaan oikeanlainen lähestyminen virheeseen, kommunikointia, sekä virheen ratkaisua. Tapahtuman hallinnan tehtävänä onkin pienentää virhetilanteista aiheutuvat riskit mahdollisimman pieniksi. Esimerkiksi päivittäin ajettavat automaattiset varmuuskopiot ovat osa tapahtumanhallintaa. (The Art of Service 2010, 143-144, 146-149.)

Suorituskyvyn hallinnointi huolehtii järjestelmän toiminnan painopisteistä sille annettujen sääntöjen perusteella. Kun kyseessä on dokumenttivarasto, on suorituskyvyn tärkeimpinä tavoitteina kapasiteetti ja saatavuus. Saatavuudenhallinta pitää huolen siitä, että data on saatavilla koko ajan. Suunniteltu tavoitettavuus keskittyy jatkuviin operaatioihin ja järjestelmän normaaliin ylläpitoon. Toinen puoli saatavuudenhallinnasta, suunnittelematon tavoitettavuus keskittyy vikasietoon, tietojen eheyteen sekä virhetilanteista toipumiseen. Turvallisuudenhallinta pitää sisällään datan tietoturvan ja käytäntöjenhallinnan. On huolehdittava siitä, että käyttäjillä on pääsy vain niihin dokumentteihin, joihin heillä on oikeus päästä käsiksi. Lisäksi on varauduttava erilaisiin tietoturvauhkiin. Laajuutensa vuoksi tietoturva-aiheesta on kirjoitettu erillinen kappale 2.5. Varmuuskopiointi ja palautus mahdollistavat palautumisen virhetilanteista ja korruptoidusta datasta. (The Art of Service 2010, 144, 149-153.)

Dokumentit voidaan tallentaa sovelluksen tietokantaan, levylle tiedostojärjestelmään tai dokumenttivarastona toimivaan pilvipalveluun. Thahir (2017) kirjoittaa tietokannan ja tiedostojärjestelmän eroista. Hän mainitsee tietokannan hyviksi puoliksi esimerkiksi paremman tietoturvan kuin tiedostojärjestelmällä ja sen, että dokumentit ja tietokanta ovat aina synkronoidut. Tiedostojärjestelmän hyviä puolia ovat esimerkiksi mahdollisesti parempi suorituskyky verrattuna tietokantaan ja helppous dokumenttien tallentamisessa ja lataamisessa. Dokumenttivarastona toimivista pilvipalveluista kerrotaan kappaleessa 2.4.

2.4 Dokumentit pilvipalveluissa

Pilvipalveluina tarjotut dokumenttivarastot varastoivat ja hallinnoivat dokumentteja ja dataa tietoverkossa. Dokumenttivarastojen tehtävänä on huolehtia esimerkiksi kapasiteetis-

ta, menettelytavoista ja tapahtumien hallinnasta. (The Art of Service 2010, 133). Pilvipalvelussa oleviin dokumentteihin pääsee käsiksi mistä tahansa päin maailmaa tahansa, kunhan on oikeudet palveluun ja laite jossa on toimiva verkkoyhteys. Dokumenttivaraston koko kasvaa sitä mukaa, mitä enemmän tilaa dokumentit pilvipalvelussa vievät. (CloudStorageBest 2012.) Kun pilvipalvelua käytetään dokumenttien tallentamiseen, sitä kutsutaan Storage as a Service -palveluksi, ja se kuuluu IaaS, eli Infrastructure as a Service päätyypin alle (SearchCloudComputing 2017.)

Pilvipalveluiden käyttäminen dokumenttien varastoimisessa tuo omat haasteensa ohjelmistokehitykseen. Esimerkiksi rajapinnan toteuttaminen dokumenttivarastoon, mahdolliset tietoturvan eroavaisuudet sovelluksen ja dokumenttivaraston välillä ja mahdollisuus siihen, että tietokanta ja dokumenttivarasto pitävät sisällään eri tietoja. Mikäli esimerkiksi on mahdollisuus päästä poistamaan dokumentti suoraan dokumenttivarastosta ilman, että tietokantaan tehdään muutoksia, saattaa ilmaantua tietojen epäsynkronisuutta. (PostgresSQL 2016.) Joitakin haasteita ratkaisemaan on kehitetty ohjelmointikirjastoja, joka käytännössä tarkoittaa sitä, että kaikkea kompleksista ohjelmointityötä ei tarvitse tehdä itse (AWS 2018b.)

Arkkitehtuuri pilvipalveluissa voi olla suhteellisen yksinkertainen, vaikka jotkin sen osiot voivat olla erittäin kompleksisia. Arkkitehtuuri koostuu kolmesta eri osiosta. IT-infrastruktuurissa, jossa esimerkiksi käsitellään ja tallennetaan asiakkaan tietoja, asiakkaan ympäristöstä, sekä näiden kahden välissä olevasta pilvestä. Pilvi sisältää esimerkiksi pääsyjen hallinnoinnin sekä tietojen suojaamisen. Kuinka nämä kolme erillistä osioita on rakennettu, riippuu esimerkiksi eri proseduureista ja laitteista, joita on käytetty. (The Art of Service 2010, 132.)

Useimmat pilvipalvelun kautta käytettävät dokumenttivarastot toimivat objekteihin perustuvana objektivarastona (object storage), ne ovat tilattomina ja käytettävissä rajapinnan kautta (ComputerWeekly, 2016.) Objektivarastot ovat strukturoimattomia, eli ne eivät sisällä kansiorakenteita (ComputerWeekly 2017). Objekteihin perustuva tarkoittaa sitä, että dokumentit tallennetaan omina objekteinaan dokumenttivarastoon. Objekti voi siis olla esimerkiksi kuva. Objekteille asetetaan metadatatietoa, eli hyödyllistä lisätietoa datasta. Metadatatietoon voidaan esimerkiksi tallentaa dokumentin muoto perustuen dokumentin sisältöön, dokumentin koko, ja dokumentin yksilöivä id-arvo. (ComputerWeekly 2015.)

Objektivaraston lisäksi dokumenttivarastot voivat toimia joko tiedostovarastoina (file storage) tai lohkovarastoina (block storage) (AWS; IBM). Tiedostovarastot toimivat hierarkkisessa struktuurissa ja data tallennetaan dokumentteina ja kansioina. Dataan päästään

käsiksi eri protokollien kautta, riippuen palvelimen käyttöjärjestelmästä. (SearchStorage 2016.) Myös lohkovarastot toimivat hierarkkisessa struktuurissa (ComputerWeekly 2017). Lohkovarastoissa data tallennetaan lohkoihin. Datalle annetaan tunniste, jolla se tallennetaan ja jolla se voidaan hakea. Lohkot toimivat kuten yksittäinen kovalevy tietokoneessa ja lohkoja hallinnoidaan käyttöjärjestelmällä. (SearchStorage 2017).

Objektivarastolla, tiedostovarastolla ja lohkovarastolla on kaikilla omat hyvät ja huonot puolensa. Objektivaraston etuina ovat sen käyttö yksinkertaisen rajapinnan kautta ja käyttöön perustuva hinnoittelu. Objektivaraston yksittäisiä objekteja ei ole mahdollista muokata, mitä voidaan pitää objektivaraston huonona puolena. Mikäli dokumenttia halutaan muokata, tulee dokumentti ensin ladata ja muokata, ja sen jälkeen tallentaa takaisin objektivarastoon. Tiedostovaraston hyvinä puolina on selkeä ja käyttäjille tuttu struktuuri, sekä helppo käytettävyys. Lisäksi se mahdollistaa dokumenttien muokkauksen. Tiedostovarasto voi kuitenkin käydä hitaaksi ja monimutkaisemmaksi, mitä enemmän kansioita ja dokumentteja siellä on. Lohkovaraston hyvinä puolina on mahdollisuus muokata dokumentteja ja hyvä suorituskyky. Lisäksi lohkovarastossa voidaan tehdä muutoksia kerralla kokonaiseen lohkoon, joka mahdollistaa esimerkiksi hyvän tapahtumahallinnan. Lohkovaraston huonona puolena on sen hinnoittelu, jokainen lohko maksaa, riippumatta siitä, mikä sen käyttöaste on. Dokumenteille ei myöskään voi asettaa samalla tavalla metadatatietoja, kuten tiedostovarastossa tai objektivarastossa. (Red Hat; SearchStorage 2014; ComputerWeekly 2017; Boucheron 2017.)

Dokumenttivarastoja on saatavilla yksityisenä, julkisena ja hybridi pilvipalveluna. Yksityisen pilvipalvelun dokumenttivaraston palvelimet ovat joko asiakkaan omissa tiloissa tai pilvipalvelun tarjoajan tarjoamissa sijainneissa. Yksityisessä pilvessä asiakkaalla on enemmän hallinnoitavaa, mutta pilvipalvelu on muokattavissa asiakkaan tarpeiden mukaan. Julkisena pilvipalveluna tarjottava dokumenttivarasto ei vaadi niin paljon hallinnallista työtä kuin yksityinen pilvi. Julkista pilveä voivat käyttää kaikki, keillä on valtuudet pilveen. Hybridi pilvipalveluna tarjottava dokumenttivarasto on sekoitus yksityistä ja julkista pilveä. Asiakas voi muokata pilvipalvelua haluamansa kaltaiseksi, esimerkiksi osaa dokumenteista voidaan säilyttää yksityisessä pilvessä ja osaa julkisessa pilvessä. (CloudStorageBest 2013).

Kirkconnell (2016) kirjoittaa NoSQL-tietokanta Couchbasen blogissa, miten hänen mukaansa binääritiedostot kannattaa tallentaa sovelluksissa. Binääritiedostot ovat tietokoneen luettavissa olevia dokumentteja, jotka voivat sisältää mitä tahansa tietoa, kuten kuvia, tekstiä tai ääniä (Webopedia). Kirkconnellin (2016) mukaan on järkevintä käyttää aina sellaista alustaa, joka on suunniteltu kyseistä käyttötarkoitusta varten. Binääritiedostojen

tapauksessa kannattaisi suosia staattisille objekteille tarkoitettu dokumenttivarastoa. Argumentteina on käytetty kustannustehokkuutta, sekä palvelimen suorituskykyä. Tietokanta tallennuspaikkana dokumenteille on kalliimpi gigatavulta. Mikäli dokumentit tallennetaan sovelluksen palvelimelle, se kuluttaa palvelimen resursseja, jotka tulisivat olla sovelluksen muussa käytössä. Myös PostgreSQL -tietokannan (2016) Wiki-sivuilla kirjoitetaan, että mikäli dokumentit tallennetaan suoraan tietokantaan, saattaa siitä aiheutua esimerkiksi haasteita suorituskyvyssä, varmuuskopioiden ottaminen käy hitaammaksi ja muistia vaaditaan enemmän. Näiden lisäksi ComputerWeekly (2016) nostaa esiin sen, että pilvipalveluiden käyttäminen dokumenttien varastoimisessa muun muassa vähentää tarvetta ostaa ja ylläpitää laitteistoa, tarjoaa kulutukseen perustuvan hinnoittelun ja mahdollistaa saatavuuden mistä tahansa.

2.5 Miten valita pilvipalvelun tarjoaja dokumenttivarastolle

Palveluntarjoajan valitseminen voi olla erittäin haastavaa useiden eri palveluntarjoajien takia, tarjolla on lähes identtisiä palveluita. Teknisiä uutisia tarjoavat eWeek ja ComputerWeekly ovat kirjoittaneet artikkelit, mitä tulee huomioida sopivaa palveluntarjoajaa valittaessa.

Pilvipalvelut eivät ole ihmelääke datan tallentamisen tuomiin haasteisiin. Aluksi kannattaa selvittää, minkälaisia organisaation haasteita pilvipalveluilla halutaan ratkaista. Tämän jälkeen voidaan miettiä, onko pilvipalveluilla mahdollista ratkaista kyseiset haasteet. (ComputerWeekly 2015b.) Tämä on mielestäni hyvä huomio sen lisäksi, että etenkin käyttöönottovaiheessa pilvipalvelut teettävät lisätyötä organisaatiolle. Vaikka data siirrettäisiin pilvipalveluun, sitä täytyy edelleen hallinnoida. Mikäli organisaation IT-osastolla ei ole entuudestaan osaamista datan hallinnoimisesta pilvipalveluissa, osaamista joudutaan joko rekrytoimaan, tai mahdollisuuksien mukaan opettelemaan. (ComputerWeekly 2015b.)

Minkälaista palveluntarjoajaa halutaan käyttää ja vastaavatko sen tarjoamat palvelut organisaation kohtaamiin haasteisiin? Palveluntarjoajalta tulee vaatia läpinäkyvyyttä siihen, mitä heidän pilvipalveluissaan tapahtuu. Kannattaa pitää mielessä, että pilvipalvelun tarjoajatkin ovat myyjiä. Tulee siis varmistaa, että ymmärretään minkälaisia palveluita tarjotaan, ja minkälaisiin haasteisiin pilvipalveluntarjoajan tuotteet ja palvelut tarjoavat ratkaisun. (eWeek 2018; ComputerWeekly 2015b.)

Mikäli datan saatavuus jää alle 99,98%, on saatavuudeltaan parempia palveluita tarjolla. Datan saatavuuden lisäksi kannattaa selvittää palveluntarjoajan viimeaikaiset käyttökatkot palvelussa. Toimittajan lukitsemista olisi suotava välttää ja varmistua siitä, että data saa-

daan tarvittaessa siirrettyä pois kyseiseltä palveluntarjoajalta, mikäli myöhemmässä vaiheessa halutaan vaihtaa palveluntarjoajaa. Etukäteen kannattaa myös pitää huoli siitä, että pilvipalvelut sopivat jo olemassa oleviin järjestelmiin suorituskyvyltään ja toimivuudeltaan. Kannattaa selvittää myös nykyisten laitteistojen ja pilvipalvelun yhteensopivuus, tällä varmistutaan data siirrettävyydestä pilvipalveluun. (eWeek 2018; ComputerWeekly, 2015b.)

Kannattaa huomioida myös mahdolliset piilokulut erilaisille skenaarioille, joita pilvipalveluissa voi tapahtua. Piilokulujen välttämiseksi on tärkeä haastatella laajasti kaikkia organisaation tulevia pilvipalvelun käyttäjiä, ei vain IT-osasto ja johtoa. Tämä sen takia, että saadaan mahdollisimman realistinen kuva siitä, minkälaisia toimintoja käyttäjät todella tarvitsevat tulevaan pilvipalveluun. Pilvipalvelun kulut voivat nousta korkeammiksi kuin paikallinen datan säilöminen, mutta pilvipalvelut tarjoavat sellaista joustavuutta, skaalautuvuutta ja universaalia yhdistettävyyttä, jota taas dokumenttien paikallinen säilöminen ei voi tarjota. (eWeek 2018.)

Huomioitavaa on myös, että vaikka data olisi pilvessä, se on kuitenkin organisaation vastuulla – ei pilvipalvelun tarjoajan. Datan salaaminen ja käyttäjien tunnistaminen pilvipalvelussa ovat myös organisaation tehtäviä. Erityisen tärkeää on kiinnittää huomiota pilvipalveluntarjoajan palvelimien sijaintiin. Palvelimet olisi syytä pitää Euroopan Unionin sisällä, sillä Yhdysvalloissa pätevät eri lait tietosuojalle. (eWeek 2018; ComputerWeekly 2015b.) Tietosuojasta jatketaan lisää osiossa 2.6.

On tärkeää olla tietoinen siitä, mitä palvelutasoa koskeva sopimus (SLA) pitää sisällään. Organisaation palvelutasosopimukset voivat vaihdella riippuen ohjelmiston tai datan kriittisyydestä. (eWeek 2018; ComputerWeekly, 2015b.) Palvelutasosopimus on lupaus palvelun tasosta. Sopimuksessa määritellään esimerkiksi huoltokatkojen maksimipituudet ja mahdollisissa vikatilanteissa korjauksiin vievä maksimiaika. (Heino 2010, 35-25.)

On myös olemassa ratkaisuita, joiden avulla voidaan käyttää eri pilvipalveluita erilaisiin tarpeisiin. Tällöin organisaatio saa parhaan mahdollisen hyödyn ympäristöjen käytöstä niille optimoituun tarkoitukseen. Kannattaakin harkita, voisiko jakaa eri pilviin sovellukset ja dokumenttivarastot. Paikallisille laitteistolle kehitetyt varmuuskopioinnit, ryhmittelyt ja kloonaukset ovat myös siirtymässä pilveen. Joillakin julkisilla pilvipalveluilla on olemassa palvelut myös näille laitteistoille kehitetyille ominaisuuksille. (eWeek 2018.)

Vielä viimeisempänä on huomioitava kaistanleveyden riittävyys kumpaankin suuntaan, dokumenttien lisäämiseen pilveen ja dokumenttien lataamiseen pilvestä. Tulee siis tarkis-

taa sekä organisaatiosi, että pilvipalveluntarjoajan yhteyksien nopeudet ja se, että kaista riittää myös palvelun ruuhkaisimpina aikoina. (ComputerWeekly 2015b.)

Onnistunut pilvipalveluun siirtyminen onnistuu siten, että siihen valmistaudutaan huolella. Pilvipalveluun siirtyvän organisaation tulee miettiä etukäteen, mitä dataa siirretään, missä muodossa tiedot ovat, miten niitä käytetään, kuka dataa hallinnoi organisaation sisällä ja miten data on priorisoitu. Tämän lisäksi pilvipalvelun tarjoaja ja valitut tuotteet ja palvelut on mietittävä huolella. Lisäksi on päätettävä, mitä dataa pidetään organisaation sisällä, ja miten sekä pilven, että organisaation sisäistä dataa hallinnoidaan.

2.6 Tietoturva ja riskit pilvipalveluissa

Jim O'Reilly ja New Jerseyn teknillinen korkeakoulu New Jersey Institute of Technology kirjoittavat artikkeleissaan, että yleisesti ottaen pilvipalvelu on turvallinen paikka datalle. Tietyt tietoturvaseikat on syytä huomioida, kun käytetään pilvipalvelua dokumenttivarastona. Tietoturvaratkaisut riippuvat eri toteutustavoista ja valituista palveluista, esimerkiksi siitä, onko käytössä julkinen vai yksityinen pilvipalvelu, sekä säilöttävien dokumenttien tyypistä. (O'Reilly 2017; New Jersey Institute of Technology.)

Jim O'Reilly, New Jersey Institute of Technology ja pilvipalveluiden turvallisuuteen erikoistunut Cloud Security Alliance ovat listanneet tietoturvauhkia pilvipalveluista. Listat pitävät sisällään paljon samanlaisia tietoturvauhkia, kuten tietomurrot, riittämättömät tunnistautumistiedot, huolimattomuus pilvipalvelun käytössä ja datan katoaminen. Lisäksi Cloud Security Alliance mainitsee listallaan pilvipalvelua käyttävät rajapinnat ja sovellukset, mahdolliset sisäpiirin väärinkäytöt ja edistyneet pitkäkestoiset uhat. (O'Reilly 2017; New Jersey Institute of Technology; Cloud Security Alliance.)

Tietomurrot pilvipalveluihin ovat yksi suurimmista riskeistä. Koska dokumenttivarastoon voidaan säilöä hyvin monenlaista dataa yksityishenkilöistä ja yrityksistä maailmanlaajuisesti, ne ovat hyviä kohteita hakkereille. Pilvipalveluntarjoajat huolehtivat pilvipalvelun perusturvallisuudesta, mutta siitä huolimatta, pilvipalvelun käyttäjät ovat vastuussa omien dokumenttiensa suojaamisesta. Tästä voi koitua erittäin vakavia ongelmia tietoturvan kanssa, mikäli asiakasorganisaatiolla ei ole riittävä osaamista suojata dokumenttivarastoaan. (New Jersey Institute of Technology; Cloud Security Alliance.) Esimerkiksi FedExiltä on vuotanut Amazonin S3 dokumenttivarastosta henkilötietoja, jossa on ollut mukana satojen suomalaisten skannattuja passeja ja henkilökortteja. Nämä tietovuodot ovat johtuneet siitä, että asiakasorganisaatio on jättänyt kansiot avoimeksi verkkoon. Toisin sanoen, palvelua ei ole osattu konfiguroida oikein. (Tivi 2018.) Sensitiiviseen dataan suositellaan

esimerkiksi salausta ja tunnistautumista, joista lisää seuraavissa kappaleissa. (New Jersey Institute of Technology.)

Toinen yleinen tietoturvariski on murretut tunnistautumistiedot palveluun. Käyttäjien salasanat voivat olla liian helppoja murtaa. Kyberrikolliset ovat myös voineet saada käsiinsä tunnistetietoja, jotka eivät ole enää käytössä, mutta joilla palveluun päästään vielä kirjautumaan. Paras vaihtoehto olisikin asettaa palveluun useamman vaiheen kirjautuminen. Esimerkiksi kun palveluun yritetään kirjautua, lähetetään käyttäjälle henkilökohtainen tekstiviesti tai puhelu automaatista. Puhelussa tai viestissä on tarvittava tieto lisätunnistaumiseen, yleensä jonkinlainen koodi numeroista ja/tai kirjaimista. Tällöin kirjautumiseen tarvitaan normaalin käyttäjätunnuksen ja salasanan lisäksi henkilökohtainen puhelin, jolloin henkilön tunnistaminen on varmempaa. (New Jersey Institute of Technology; O'Reilly 2017.)

On myös syytä varautua mahdolliseen tietojen katoamiseen varmuuskopiointimenetelmillä ja levykuvilla. Myös varmuuskopiot tulee suojata tietojen monistamisessa, eli replikaatiossa. Vaikka kyseessä on pilvipalvelu, on taustalla fyysiset palvelimet, jotka voivat kohdata esimerkiksi luonnonkatastrofeja tai muita vaaroja, samalla tavalla kuin kaikki muutkin fyysiset rakennelmat. Myös kyberrikolliset voivat aiheuttaa tietojen katoamisia, mikäli he pääsevät käsiksi pilvipalvelussa olevaan dataan. Mahdollisten luonnonkatastrofien ja kyberrikollisten lisäksi myös käyttäjät voivat itse aiheuttaa sen, ettei tietoihin päästä enää käsiksi. Mikäli tiedot on suojattu salauksen avulla, on huolehdittava salausavaimen turvallisuudesta. Salausavain on ainut tapa purkaa salattu data pilvipalvelussa. (New Jersey Institute of Technology; O'Reilly 2017.) Jim O'Reillyn (2017) artikkelin mukaan turvallisin tapa on salata data lähde palvelimella ja hallinnoida avaimia itse.

Pilvipalveluiden käyttöön liittyy paljon muutakin kuin varsinainen pilvipalvelu itse, esimerkiksi käyttäjät, käyttäjien laitteet, rajapinnat ja sovellukset. Eri lähteiden perusteella pilvipalvelut itsessään ovat turvallisia, mutta sen sijaan esimerkiksi käyttäjät ja käyttäjien laitteet eivät aina välttämättä ole. Lisäksi huolimatta nimestään, pilvipalvelun laitteet sijaitsevat maan pinnalla ja voivat kohdata mitä tahansa luonnonkatastrofeja ja onnettomuuksia. Tärkeintä olisikin siis huolehtia siitä, että organisaatiolla on riittävästi osaamista pilvipalveluiden käyttöönottoon, käyttäjät ovat koulutettu tietoturvalliseen pilvipalvelun käyttämiseen ja riittävistä varmuuskopiointi menetelmistä.

2.7 Tietosuoja

Toukokuussa 2018 voimaan astuva Euroopan Unionin uusi tietosuoja-asetus on huomioitava kaikessa sovelluskehityksessä, jossa käsitellään henkilötietoja. Kaikki henkilöä yksilöivä tieto muodostaa henkilörekisterin. Koska esimerkiksi sähköpostit ovat yksilöllisiä, sähköpostiosoitteiden tallentaminen järjestelmän tietokantaan muodostaa henkilörekisterin. Henkilörekisterin tulee noudattaa toukokuusta 2018 alkaen voimaan astuvaa Euroopan Unionin laajuista tietosuoja-asetusta, General Data Protection Regulation, eli GDPR:ää. GPRD:n piiriin kuuluvat kaikki yritykset, jotka käsittelevät Euroopan Union kansalaisten henkilötietoja. (Tietosuojamalli.)

GDPR määrittää rekisterinpitäjän velvollisuudet ja rekisteröidyn oikeudet. Velvollisuuksia ovat muun muassa tietosuojaselosteen tekeminen, sekä velvollisuus pystyä osoittamaan käyttäjän antama suostumus henkilötietojen keräämiseen. Tietosuojaselosteen tulee sisältää tieto, mihin kolmansiin osapuoliin (third party) henkilötietoja siirretään, mitä tietoja kolmas osapuoli saa, ja mihin tarkoitukseen niitä käytetään. (Tietosuojamalli.)

Käyttäjien oikeuksiin taas kuuluu esimerkiksi kaikkien omien tietojen saaminen rekisteristä koneen luettavassa muodossa (esimerkiksi JSON tai XML), sekä oikeus tulla unohdetuksi. (Tietosuojamalli.)

Suurista yrityksistä esimerkiksi PayPal on listannut sivuillaan kaikki kolmannet osapuolet, joihin henkilötietoja mahdollisesti siirretään ja lista on huomattavan pitkä. Listalla on lueteltu ne kolmannet osapuolet, jotka eivät ole PayPalin asiakkaita. (PayPal 2018.) Listalla on reilusti yli 350 kolmatta osapuolta opinnäytetyön tekoaikaan huhtikuussa 2018.

Lisäksi GDPR määrittelee henkilötietojen tallentamisesta Euroopan Unionin ulkopuolelle. EU:n ulkopuoliset maat eivät ole kiellettyjä, kunhan EU:n komissio on hyväksynyt kyseisen maan tietosuojan tason. Lista hyväksytyistä maista julkaistaan muun muassa Euroopan Union verkkosivuilla. Mikäli komissio ei ole hyväksynyt riittävää tietosuojan tasoa, tarvitaan tietojen siirtämiseen Euroopan Union ulkopuolelle varmistus siitä, että riittävät suojatoimet on toteutettu ja rekisteröityneiden käyttäjien saatavilla tulee olla tehokkaita oikeussuojakeinoja. (Tietosuojamalli.)

Dokumenttivarasto ei lähtökohtaisesti ole henkilörekisteri vaikka dokumentit voivat sisältää henkilötietoja. Se, että pitävätkö dokumentit sisällään henkilötietoja, riippuu dokumenttien luonteesta ja dokumenttien tekijöistä. Dokumentithan voivat pitää sisällään myös erittäin arkaluonteista tietoa henkilöistä, kuten esimerkiksi henkilötunnuksia ja terveystietoja.

Dokumenttien mahdollisuus arkaluontoisiin tietoihin tulisi huomioida kun ollaan valitsemassa dokumenttivarastoa. Kannattaa valita sellainen palveluntarjoaja, jonka palvelimien sijainnin voi valita Euroopan Unionin sisäpuolelta.

3 Dokumenttivaraston käyttö web-ohjelmoinnissa

Tässä osiossa toteutetaan dokumenttivarasto käyttäen Ruby on Rails ohjelmointikehystä ja Amazon Simple Storage Serviceä. Aluksi perustellaan valitut teknologiat ja käydään läpi web-sovelluksen rakenne, lisäksi katsotaan lyhyesti, mitä ohjelmistoprojektissa on toteutettu ennen integraatiota pilvipalveluun. Näiden jälkeen toteutetaan varsinainen integraatio S3:seen ja varmistetaan, että dokumentit ovat ladattavissa käyttäjien laitteille. Lopuksi kuvaillaan miten käyttöliittymä on toteutettu selaimeen ja varmistetaan sovelluksen tietoturva.

Ruby on Rails on MCV-kehys, eli noudattaa Model-Controller-View rakennetta. Lukijalta odotetaan MCV-kehyksen perusteiden ymmärrystä, eli niitä ei tässä opinnäytetyössä käydä läpi.

3.1 Perustelut valinnoille

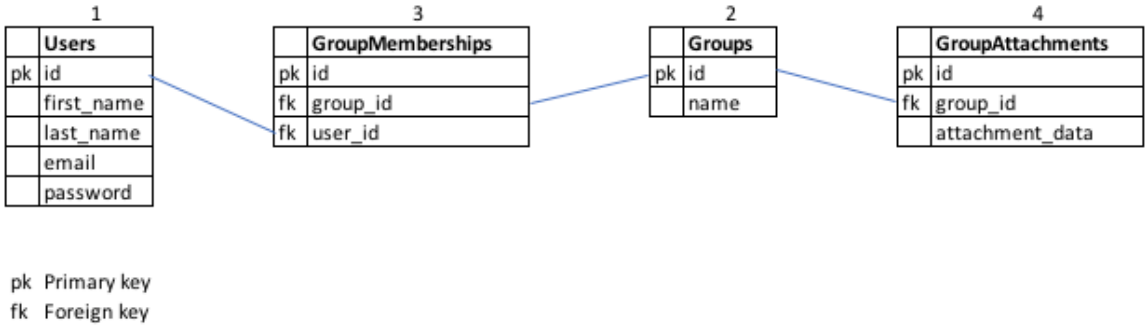
Ruby on Rails, lyhyemmin pelkästään Rails, on valittu alustaksi sen takia, että se on nopea käyttää, ilmainen ja helppo oppia. Rails kehyksessä käytetään ohjelmointikielenä Rubya, dynaamisesti tyypitettyä ja tulkattavaa ohjelmointikieltä. Rails on myös ohjelmoitu Rubylla. Ruby on syntaksiltaan lähes kuin englantia, eli suhteellisen helposti luettavaa. Tämä vähentää esimerkiksi dokumentoinnin ja koodin kommentoinnin tarvetta, koodi kommentoi itse itseään. Ruby on suosittu ohjelmointikieli web-ohjelmoinnissa ja monet suuret yritykset kuten AirBnb, Twitch ja GitHub käyttävät sitä sovelluksissaan. Vuonna 2017 Ruby ohjelmointikielellä tehtiin neljänneksi eniten Pull Requesteja GitHubiin. Pull Request on pyyntö muuttaa olemassa olevaa koodia GitHubissa. Ruby on Railsin avulla web-ohjelmistojen pystyttäminen on nopeaa, sanotaan, että jopa nopeampaa kuin muilla ohjelmistokehyksillä. (Stackify 2018; GitHub 2018.)

PostgreSQL tietokanta on valittu käyttöön sen vuoksi, että Railsin kanssa oletuksena olevaa SQLite3:sta ei ole suunniteltu siten, että useampi prosessi voisi käyttää sitä yhtäaikaisesti. Jos on tarkoituksena tehdä tietokannan useamman prosessin yhtäaikaiseen käyttöön sovelluksia, kannattaa jo alusta asti valita jokin sellainen tietokanta, joka sen mahdollistaa. DB-Engines, joka listaa tietokantoja niiden suosion perusteella, on valinnut PostgreSQL:n vuoden 2017 tietokannaksi, sen jatkuvasti nousevan suosion takia. (DB-Engines 2018b.) Lisäksi DB Engines Ranking on listannut PostgreSQL:n olevan neljänneksi suosituin tietokantajärjestelmistä. (DB-Engines 2018.) PostgreSQL on tällöin erinomainen vaihtoehto, toinen yleisesti käytetty ja hyvä vaihtoehto olisi ollut MySQL. (DigitalOcean 2014.)

Amazon Web Services, jäljempänä AWS, on maailman suurin pilvipalveluiden tarjoaja. AWS:llä on tarjolla hyvin erilaisia tuotteita, joista valita. Kaikissa tuotteissa on standardisoisudut tietoturvaratkaisut, sekä mahdollisuus päättää palvelimien maantieteellinen sijainti 16 eri maantieteellisestä alueesta. Lukuisat isot organisaatiot, kuten Rovio ja Netflix, käyttävät AWS:n pilvipalveluita. AWS:n monipuolisesta tuotevalikoimasta valitaan käyttöön Simple Storage Service objektivarasto, lyhyemmin S3. S3:ssa voit säilöä ja ladata dataa niin paljon kuin tarvitset. Amazon myös itse käyttää omilla globaaleilla verkkosivuillaan. S3 tuotteen saa ilmaiseksi käyttöön 12 kuukaudeksi rekisteröitymisestä 5 GB:n kokoisella tilalla. S3 tuotteelle luvataan 99.999999999% datan säilyvyys. (CloudRanger 2017; AWS 2018a; AWS 2018c; AWS 2018d; AWS2018e.)

3.2 Tietokannan rakenne

Alla kuvattuna sovelluksen tietokannan rakenne. Tietokannan rakenteessa käytetään relaatiomallia. Kuvassa 2 olevien taulujen päällä on järjestysnumero, joka määrittää, missä järjestyksessä kyseiset taulut toteutetaan järjestelmään.



Kuva 2. Sovelluksen tietokannan rakenne mallinnettuna

Ensimmäisenä on Users, eli käyttäjät -taulut. Tauluun tallennetaan pääavaimena käyttäjien id:t. Muut kolumnit ovat etunimi, sukunimi ja salattu salasana. Groups, eli ryhmät -taulu toteutetaan seuraavana. Ryhmät-tauluun tallennetaan pääavaimena ryhmän id ja sen lisäksi ryhmän nimi. Kun käyttäjät ja ryhmät on luotu, voidaan luoda niiden välinen välitaulu, GroupMemberships, eli käyttäjien jäsenyys ryhmään. GroupMemberships-taulua ei kannata luoda ennen kuin sekä käyttäjät-, että ryhmät-taulut ovat luotu, sillä niistä tulee viiteavaimet GroupMemberships-tauluun. GroupMemberships-tauluun tallennetaan pääavaimena oma id ja sen lisäksi viiteavaimena user_id ja group_id. Viimeisenä luodaan GroupAttachments, eli ryhmien dokumentit-taulu. Dokumenteille tarvitaan oma taulu, sillä yhdellä ryhmällä voi olla useampi eri dokumentteja. Mikäli tieto dokumenteista laitettaisiin suoraan ryhmä tauluun, ryhmällä voisi olla vain yksi dokumentti kerrallaan. Ryhmien dokumentti tauluun tallennetaan pääavaimena id, viiteavaimena group_id ja lisäksi taulussa on attachment_data niminen kolumni. Kyseiseen kolumniin tallennetaan JSONB-

muodossa tieto siitä, missä dokumentti, eli attachment, sijaitsee dokumenttivarastossa. JSONB on JSON (JavaScript Object Notation) datatyyppi, joka tallennetaan tietokantaan hajotetussa binäärimuodossa, lisäksi JSONB mahdollistaa indeksoinnin (PostegreSql).

3.3 Ohjelmoinnin vaiheet

Ohjelmointityö aloitetaan luomalla uusi Rails projekti komennolla `rails new opinnaytetyo –database=postgresql`. Oletustietokanta on SQLite3, jonka takia määritellään komennossa erikseen haluttu tietokanta. Kun projekti on luotu, tehdään projektille tietokanta komennolla `rails db:create`. (Rails Guides 2018b.) Seuraavaksi projektiin lisätään Slim- ja Simple Form -kirjastot. Slim-kirjaston avulla voidaan kirjoittaa näkymä-tiedostoihin HTML:ää ja Rubya hyvin kevyellä syntaksilla, esimerkiksi HTML:stä tuttuja lopetustageja ei tarvitse ollenkaan. (Ruby Doc 2018a.) Simple form -kirjasto sen sijaan helpottaa lomakkeiden tekemistä tarjoamalla erilaisia apuvälineitä lomakkeisiin. (Ruby Doc 2018b.) Tulemme tarvitsemaan lomaketta käyttäjien luomiseen, sekä ryhmien luomiseen ja muokkaamiseen.

Ensimmäisenä luodaan aloitussivu, joka nimetään Home-sivuksi. Aloitussivua varten tarvitaan kontrolleri, nimetään se HomesControlleriksi. Kontrolleriin tehdään vain show funktio aloitussivun näyttämiselle. Routes-tiedostoon määritellään polut, joita sovelluksella on. Lisätään routes-tiedostoon rivi `resource :home, only: [:show]`. Kun polku ja kontrolleri on tehty, voidaan lisätä näkymä-tiedosto projektiin, johon voidaan esimerkiksi kirjoittaa sovelluksen tarkoitus ja käyttöohjeita.

Seuraavaksi luodaan järjestelmän käyttäjät, sekä sisäänkirjautuminen. Käyttäjiin ja sisäänkirjautumiseen käytetään Devise-kirjastoa. Devise on kirjasto tunnistautumiseen. Käyttäjät luodaan komennolla `rails generate devise User`. Tämä komento luo migraatiotiedoston, jolla luodaan User-malli, sekä User-tietokantataulu. Migraatiot ajetaan komennolla `rails db:migrate`. Migraation ajaminen luo ohjelmistoprojektiin User mallin, routes-tiedostoon osoitteen käyttäjille, ja lisäksi kirjasto sisältää useita eri kontrollereita esimerkiksi rekisteröitymiseen ja kirjautumiseen. Luodaan käyttäjille näkymä-tiedostot komennolla `rails generate devise:views`. (Devise 2018.)

Kun kirjautuminen on saatu valmiiksi, tehdään seuraavaksi ryhmät. Tietokantataulu ja malli luodaan komennolla `rails generate model Group`, jolloin ryhmä nimetään Groupiksi. Komento luo migraatio-tiedoston, joka ajetaan aikaisemmin mainitulla komennolla. Ryhmät tarvitsevat myös oman kontrollerin, osoitteen, sekä näkymä-tiedostot. Kontrolleri luodaan projektiin käsiin, ja se tulee sisältämään funktiot ryhmien luomiseen, näyttämiseen (yksi tai monta), muokkaamiseen ja poistamiseen. Kontrollerin indeksi funktioon määritel-

lään, että haetaan vain kirjautuneen käyttäjän ryhmät, ei siis kaikkia olemassa olevia ryhmiä. Kuka tahansa voi lisätä ryhmiä ja liittää niihin haluamansa henkilöt. Halutaankin varmistaa, että vain ryhmään kuuluvat henkilöt näkevät ryhmän. Näkymä-tiedostot luodaan myös käsin projektiin, ja jokaiselle eri sivulle tehdään oma tiedosto. Tässä tapauksessa teemme siis tiedostot luomiselle, näyttämiselle ja muokkaamiselle.

Koska yksi käyttäjä voi kuulua useampaan ryhmään ja yhteen ryhmään kuuluu useampi kuin yksi käyttäjä, on kyseessä monen suhde moneen-relaatio, johon tarvitaan välitaulurakennetta. (Mooc 2018.) Seuraavaksi tehdään edellä mainittu välitaulurakenne käyttäjien ja ryhmien välille. Välitaulurakenteeseen tarvitaan välitaulu tietokantaan, sekä oma malli. Tauluun tulee viiteavaimena sekä käyttäjä, että ryhmä ja ne voidaan syöttää valmiiksi jo migraatio-tiedoston luontivaiheessa. Nimetään välitaulu ja malli GroupMembership:ksi. Migraatio luodaan komennolla `rails generate migration GroupMembership user:references group:references`. Tämän jälkeen ajetaan migraatio. Tässä tapauksessa emme tarvitse kontrolleria tai näkymä-tiedostoja.

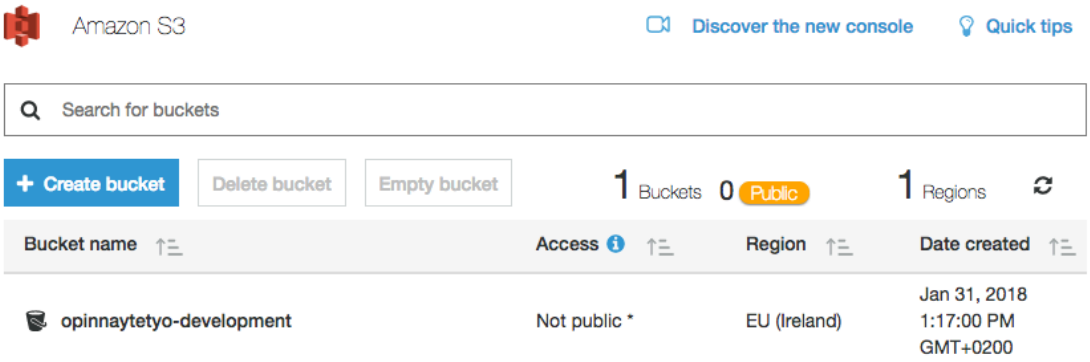
Nyt on tehty valmiiksi seuraavat kohdat

- User, Group ja GroupMembership mallit
- GroupsController kontrolleri
- Määritetty routes-tiedostoon polut kotisivulle, käyttäjille ja ryhmille
- Näkymä-tiedostot ryhmälle ja käyttäjille.

Seuraavassa kappaleessa rakennamme integraation, jolla ryhmän kesken voidaan jakaa dokumentteja, joita säilytetään pilvipalvelussa.

3.4 Integraatio pilvipalveluun

Jotta voidaan käyttää AWS:n pilvipalveluita, on luotava tunnus AWS consoliin. Rekisteröitymiseen tarvitaan henkilötietoja ja luottokortin tiedot, vaikka ei käytettäisikään mitään maksullista tuotetta. Rekisteröitymisen jälkeen valitaan haluttu tuote, tässä tapauksessa S3. Tuotteelle luodaan varastoja (bucket). Koska toistaiseksi työskennellään kehitysvaiheessa, luomme varaston opinnaytetyo-development, kuten kuvassa 3 on näytetty. Varastoa luotaessa tulee valita varaston sijainti. Viimeistään kun sovellus julkaistaisiin, luotaisiin tuotantoversiolle oma varasto, esimerkiksi opinnaytetyo-production.



Kuva 3. Amazon Web Services opinnaytetyo-development dokumenttivarasto

Lisätään ohjelmistoprojektiin kaksi kirjastoa helpottamaan dokumenttien käsittelyä. AWS SDK S3 nopeuttaa tarjoamalla erilaisia Ruby luokkia AWS:n S3 tuotteeseen. Toinen kirjasto jonka lisäämme on Shrine. Shrine sen sijaan helpottaa dokumenttien käsittelyä ohjelmassa. Näistä lisää jäljempänä. (AWS 2018b.)

Koska yhdellä ryhmällä voi olla useita eri dokumentteja, dokumenteille tehdään oma taulu tietokantaan ja nimetään se GroupAttachmentsiksi. Shrine-kirjasto ohjeistaa tekemään tauluun kolumnin, jonka nimen loppuosa on _data, ja tallentamaan sen tyypiksi text tai JSON (Shrine). Tähän kolumniin tallennetaan tieto, mistä kyseinen dokumentti löytyy S3:stä, ja varsinainen dokumentti säilytetään S3:ssä. Käytetään tässä tapauksessa kolumnin nimenä attachment_data joka on tyypiltään JSONB. Lisäksi GroupAttachments tauluun tulee viiteavaimena ryhmä.

Luodaan seuraavaksi FileUploader-luokka, jonne määritellään, minkä tyyppisiä dokumentteja halutaan käsitellä. Railssillä luokkien periminen määritellään < symbolilla, joten kyseinen luokka myös perii Shrine-luokan ominaisuudet ja funktiot.

```
class FileUploader < Shrine
  # plugins and uploading logic comes here
  plugin :determine_mime_type
end
```

GroupAttachment modeliin lisätään virtuaalinen attribuutti attachment dokumenttien käsittelyä varten.

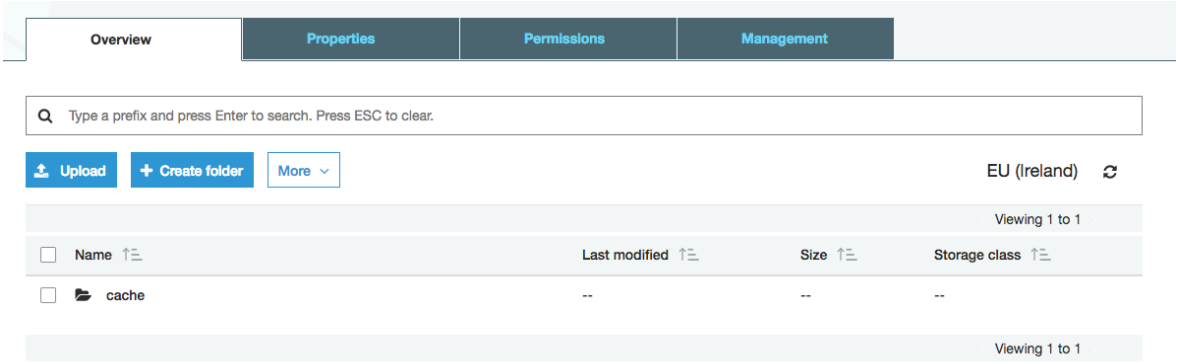
```
class GroupAttachment < ApplicationRecord
  include FileUploader::Attachment.new(:attachment)
  belongs_to :group
end
```

Lisätään projektiin alustajiin Shrine-alustaja. Alustajaan määritellään s3_options muuttujien nimet, mutta varsinaiset tunnukset tallennetaan .env.local-tiedostoon jota ei jaeta julkisesti edes versionhallintaan. Seuraavaksi määritellään dokumenttien väliaikainen ja pysyvä varasto, eli storaget.

Shrinen mukana tulee myös erilaisia lisäosio, plugineita, jotka saa käyttöön kutsumalla lisäosan nimeä halutussa luokassa. Esimerkiksi presign_endpoint -plugin mahdollistaa sen, että dokumentin voi ladata selaimesta suoraan S3 varastoon.

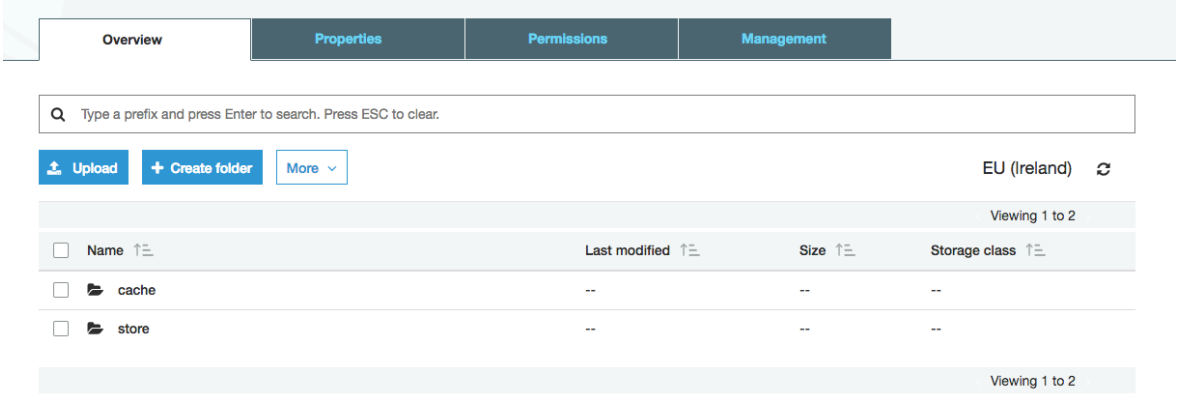
```
require "shrine/storage/s3"

s3_options = {
  access_key_id:          ENV["S3_ACCESS_KEY_ID"],
  secret_access_key:      ENV["S3_SECRET_ACCESS_KEY"],
  region:                 ENV["S3_REGION"],
  bucket:                 ENV["S3_BUCKET"],
}

Shrine.storages = {
  cache: Shrine::Storage::S3.new(prefix: "cache", **s3_options),
  store: Shrine::Storage::S3.new(prefix: "store", **s3_options)
}

Shrine.plugin :presign_endpoint
Shrine.plugin :activerecord
```

Kansiot joihin dokumentteja tallennetaan S3:ssä muodostuvat automaattisesti. Jos käytetään esimerkiksi vain cache-kansiota, näet vain sen S3:ssä. Kuvassa 4 S3:een on muodostunut vain cache-kansio, johon kaikki dokumentit tallennetaan.



Kuva 4. Amazon Web Services cache-kansio

Cache-kansiota käytetään väliaikaisena säilytyspaikkana. Cache nimenä viittaa välimuistiin, eli tässä tapauksessa välimuistiin voidaan ladata nopeasti dokumentteja. (Computer-Hope 2017.) Varsinainen varasto dokumenteille tulee olemaan store-kansio. Store-kansio muodostuu S3:een, kun ohjelmallisesti tallennamme sinne tietoja. Cache-kansiota siis käytetään siihen, että käyttäjän lataamat dokumentit ladataan ensin sinne. Cachesta dokumentit siirretään store-kansioon, jolloin store-kansio muodostuu S3:een. Kuva 5 on kuvakaappaus S3:sta, kun cache ja store-kansiot ovat muodostuneet sinne.



Kuva 5. Amazon Web Services cache ja store-kansiot

Halutaan määritellä, minkälaiset parametrit ovat sallittuja käyttäjiltä kun luodaan tai muokataan ryhmää. Parametreiksi sallitaan ainoastaan ryhmän nimi, jäsenet sekä dokumentit. Määrittely tehdään GroupsControlleriin ja nimetään funktio group_paramsiksi.

```
def group_params
  params.require(:group).permit(
    :name,
    user_ids: [],
    attachments_attributes: [:id, :attachment, :_destroy])
end
```

3.5 Dokumenttien lataus

Mahdollisuus dokumenttien lataamiseen on sovelluksen käyttötarkoituksen kannalta erittäin oleellinen ominaisuus. Eli halutaan, että kaikki ryhmään kuuluvat henkilöt voivat ladata ryhmän minkä tahansa dokumentin itselleen. Tehdään GroupAttachmentsController -niminen kontrolleri, johon tulee vain yksi funktio dokumentin lataamista varten, download funktio. Download funktiossa haetaan GroupAttachment -luokasta dokumentti, ja luetaan sen metadata tietoja.

```
def download
  attachment = GroupAttachment.find(params[:group_attachment_id]).attachment
```

```
    data = open(attachment.url)
    send_data(
      data.read,
      filename: attachment.metadata["filename"],
      type: attachment.metadata["mime_type"]
    )
  end
```

Latausta varten tarvitaan lisäksi oma polku routes-tiedostoon. Koska lataus koskee aina tietyn ryhmän dokumenttien latausta, tulee kyseinen lataus osoite ryhmät-osoiteryhmän sisälle.

```
resources :groups do
  resources :group_attachments, only: [], path: "attachements" do
    get "/download", to: "group_attachments#download"
  end
end
```

Kohdassa get "`/download`", to: "`group_attachments#download`" määritellään, että käytetään GroupAttachmentsControllerin download funktiota.

Tämän jälkeen voidaan käyttöliittymässä mahdollistaa dokumenttien lataus ryhmän jäsenille.

3.6 Käyttöliittymä

Käyttöliittymästä tehdään pelkistetty ja yksinkertainen. Sovelluksella on yksi tarkoitus, pystyä jakamaan dokumentteja ryhmän kesken. Tehdään sovelluksen käyttäminen mahdollisimman helpoksi ja yksinkertaiseksi. Sovellusta tulisi voida käyttää selaimen kautta sekä tietokoneella, että puhelimella.

Käyttöliittymä halutaan toteuttaa siten, että käyttäjä voi raahata ja pudottaa dokumentin dokumenteille tarkoitetun suorakulmion sisälle. Tällaista toimintoa varten tarvitsemme selaimeen JavaScriptiä. DropzoneJs on tunnettu JavaScript kirjasto, jolla voidaan helposti toteuttaa juuri halutun kaltaisia ominaisuuksia. Dropzonen saa käyttöön lisäämällä gemfile-tiedostoon `gem 'dropzonejs-rails'` ja ajamalla komentoriviltä `bundle install`, joka asentaa gemfile-tiedoston määritellyt kirjastot. (GitHub 2017.)

View-tiedostossa käydään läpi kaikki ryhmälle jo lisätyt dokumentit käyttämällä Rubyn each do -funktiota. Dokumenteista näytetään dokumentin nimi, koko ja dokumentin tyyppi.

```
- @group.attachments.each do |a|
   div
     .dz-name= a.attachment.metadata["filename"]
     .dz-size= a.attachment.metadata["size"]
     .dz-type= a.attachment.metadata["mime_type"]
     .dz-key= a.attachment.id
```

Iteraation sisällä kysytään Rubyn persisted-funktiolla, onko dokumentti luotu jo tietokantaan. (ApiDock 2018.)

```
       - if a.persisted?
```

Jos a, (dokumentti), on olemassa, mahdollistetaan, että käyttäjä voi ladata dokumentin. Koska dokumentin lataaminen laitettiin ryhmät-osoiteryhmän sisälle, osoitteeseen tulee ”group_” (viittaa ryhmään) ”group_attachment_” (viittaa ryhmän dokumentteihin) ”download” (viittaa download-funktioon GroupAttachmentsControllerissa). Lisäksi on määritelty parametrit, joita tuo lataus-linkki itselleen saa (@group, a). Parametrit ovat se ryhmä, jonka tiedoissa ollaan, ja se dokumentti, jossa tuo latauslinkki on. Alla näkyvillä koko latauslinkki.

```
.dz-download-path= group_group_attachment_download_path(@group, a)
```

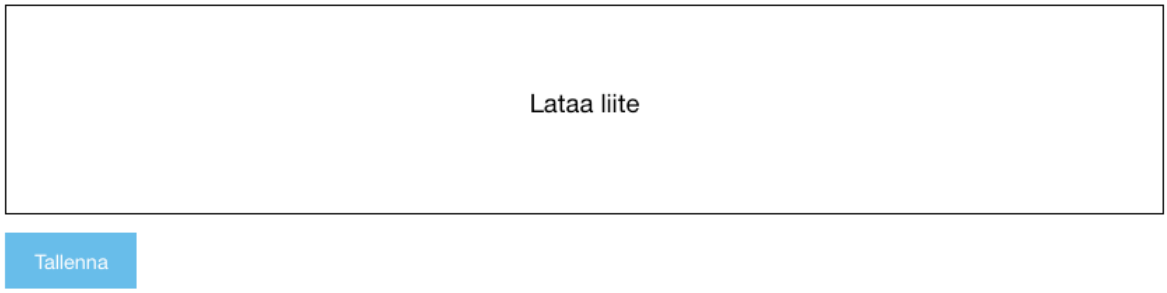
Jos persisted-funktio on palauttanut toden ja käyttäjä painaa tallenna, lähetetään serverille dokumentin id, attachment_data joka pitää sisällään metatietoa dokumentista, sekä tieto, tuleeko dokumentti poistaa.

```
input type="hidden" name="group[attachments_attributes][#{a.id}][id]" value=a.id
input type="hidden" data-role="destroy" data-s3key=a.attachment.id name=
             "group[attachments_attributes][#{a.id}][_destroy]" value="false"
```

Seuraavaksi tehdään suorakulmio, johon käyttäjä voi raahata ja pudottaa dokumentin. Käytetään siinä Dropzonen tarjoamia tyyliluokkia. Koska ryhmälle voi tallentaa useita eri dokumentteja, käytetään luokkaa multiple, eli useita.

```
.dropzone.multiple data-input-name="group[attachments_attributes][id][attachment]"
  .dz-message data-dz-message=""
     span Lataa liite
```

Kuvasta 6 näkee miltä tyhjä suorakulmio näyttää. Suorakulmioon voi joko klikata, tai siihen voi raahata dokumentin.

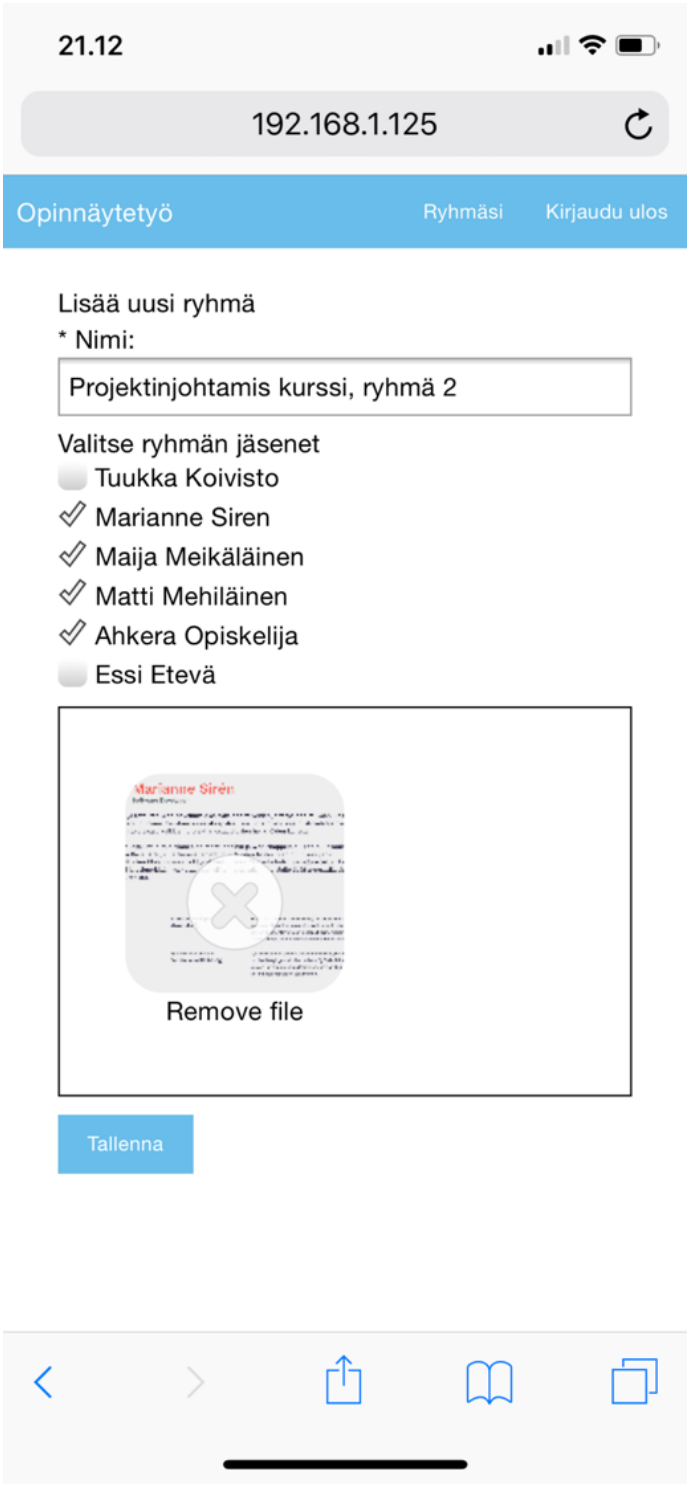


Kuva 6. Tyhjä suorakulmio sovelluksessa

Kun dokumentti on lisätty, se näkyy laatikossa. Voidaan myös lisätä useampia dokumentteja suorakulmioon, kuten kuva 7 osoittaa. Dokumenttien määrää ei ole rajoitettu millään tavalla.

Kuva 7. Suorakulmion alueelle on lisätty dokumentteja

Lisäämällä ”`meta[content="width=device-width, initial-scale=1.0" name="viewport"]`” application.html.slim-tiedostoon pidetään huoli siitä, että web-sivu skaalautuu myös mobiililaitteelle sopivaksi. Kuva 8 on kuvakaappaus puhelimen ruudulta ryhmä luonti vaihteesta. Eli mobiililaitteen tapauksessa suurennetaan sovelluksen kokoa skaalautumaan paremmin mobiilialustalle. Tämä mahdollistaa web-sivun käytettävyyden myös silloin, kun tietokone ei ole saatavissa. On toki mahdollista, ettei mobiililaitteella ole tarvittavia ohjelmia eri dokumenttimuotojen avaamiseen, esimerkiksi Power Point -ohjelmaa.



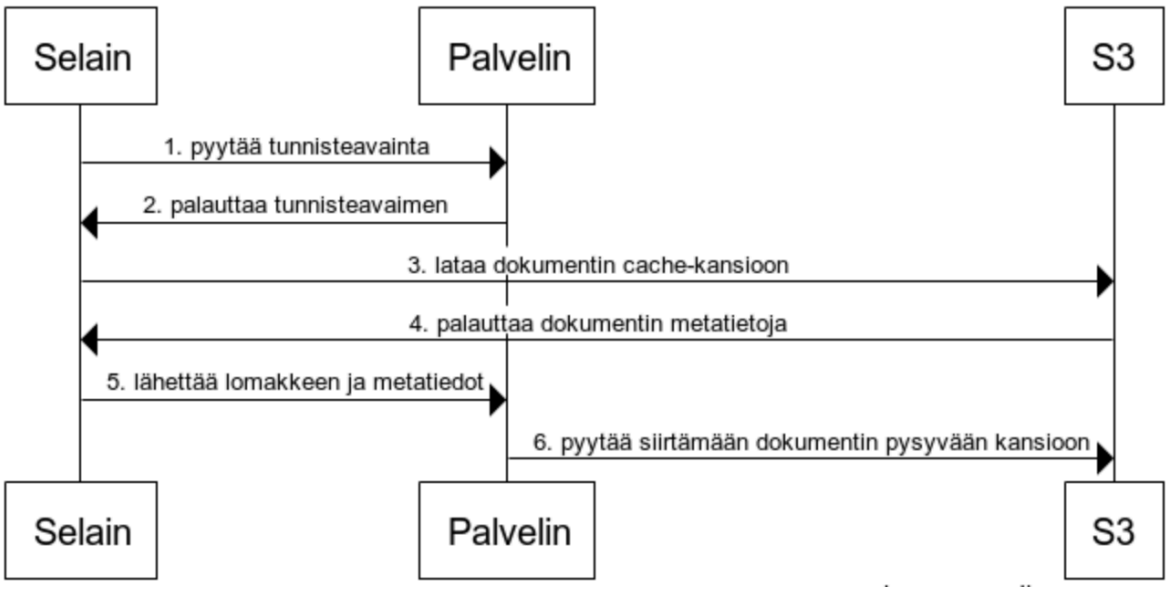
Kuva 8. Mobiililaitteella näkyvä käyttöliittymä ryhmän lisäämisestä

3.7 Prosessi selaimen, palvelimen ja dokumenttivaraston välillä

Kuvassa 9 on piirretty auki mitä tapahtuu, kun käyttäjä tiputtaa selaimen raahaa ja pudota suorakulmioon dokumentin.

Kun käyttäjä on tiputtanut dokumentin, kohdassa yksi selain pyytää palvelinta generoimaan tunnisteavaimen. Tunnisteavain generoidaan dokumentin tietojen ja S3:n tunnusten

avulla, jotka ovat palvelimella tiedossa. Tunnisteavainta tarvitaan myöhemmässä vaiheessa siihen, että S3 vastaanottaa dokumentin. Kohdassa kaksi palvelin lähettää selaimelle takaisin generoidun tunnisteavaimen. Kohta kolme on näistä kaikista hitain, tässä dokumentti ladataan S3:sen cache-kansioon. Dokumentin mukana lähtee palvelimen generoima tunnisteavain ja dokumentin lataus onnistuu vain, mikäli tunnisteavain on oikein. S3 näkee dokumentin tiedot, sekä tietää S3:n tunnukset ja tunnisteavaimen generointitavan ja pystyy siten varmistamaan tunnisteavaimen oikeellisuuden. Kohdassa neljä S3 palauttaa selaimelle metatietoja dokumentista, esimerkiksi dokumentin nimen ja sen yksilöivän id-arvon. Kun neljäs vaihe on suoritettu, käyttäjä näkee selaimessa, että dokumentti on nyt ladattu suorakulmion sisälle. Kohdassa viisi käyttäjä klikkaa tallenna nappia selaimessa ja selain lähettää lomakkeen sisältämät tiedot ja S3:lta saadut dokumentin metatiedot palvelimelle. Palvelimella oleva Shrine-kirjasto vastaanottaa metatiedot, ja kohdassa kuusi Shrine pyytää S3 siirtämään dokumentin cache-kansiosta vakinaiseen säilöön, eli store-kansioon.



Kuva 9. Järjestysdiagrammi dokumenttien lisäämisestä.

Tämä yllä kuvattu toimintamalli mahdollistaa sen, että web-sovelluksen palvelin ei ruuhkaudu vaikka käyttäjä lisäisi erittäin isoa dokumenttia, tai vaikka monta käyttäjää lisäisi dokumentteja samanaikaisesti. Koska dokumentti ladataan kohdassa kolme suoraan S3-palveluun, ei dokumenttien koko tai dokumenttien lataus yhtä aikaisesti vaikuta palvelimen suorituskykyyn.

3.8 Tietoturva

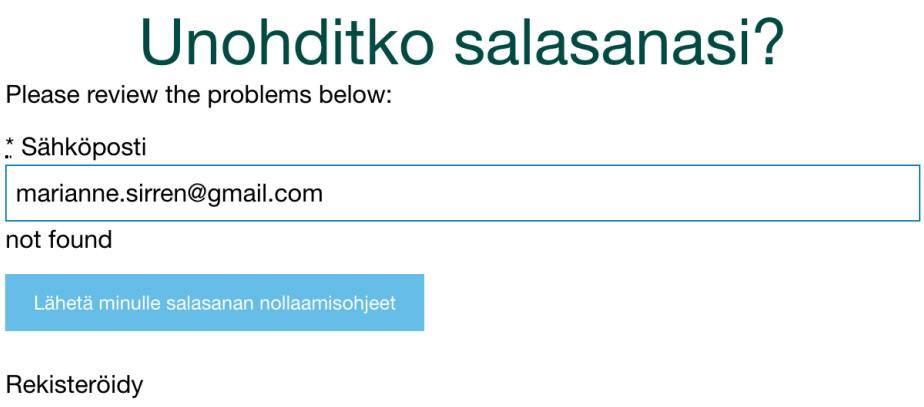
Rails kehyksessä on useita sisäänrakennettuja tietoturvaominaisuuksia, joiden takia se on turvallinen ja hyvä työkalu web-kehitykseen. Rails tarjoaa erilaisia apumetodeja käyttöön tietoturvan lisäämiseen web-sovelluksessa. (Rails Guides 2018b.) Olisi hyvä noudattaa Railsin tapaa toimia, ja mikäli haluaa toimia tästä poikkeavalla tavalla, tulee kehittäjän huolehtia tietoturvasta muulla tavoin.

OWASP Top 10 (2017) tietoturvariskit kuvaa yleisimpiä haavoittuvuuksia web-sovelluksissa:

1. Injektiot
2. Rikkinäinen tunnistautuminen
3. Arkaluontoisen datan altistuminen
4. XML entiteettien haavoittuvuudet
5. Rikkinäinen kulunvalvonta
6. Turvallisuusasioiden virheellinen määritys
7. XSS-haavoittuvuudet
8. Epäluotettava deserialisointi (palauttaminen)
9. Komponentit tunnetuilla haavoittuvuuksilla
10. Riittämätön lokittaminen ja valvonta.

Käydään lyhyesti läpi, miten sovelluksessa on otettu huomioon listan tietoturvariskit.

Ensimmäisenä listalla on injektiot. Rails käyttää ActiveRecord-kirjastoa, joka sanitoi kyselyt SQL-injektioiden varalta. Toisena on rikkinäinen tunnistautuminen. Tunnistautuminen on hoidettu käyttämällä Devise-kirjastoa, joka on yleisesti käytössä oleva kirjasto. Devise-kirjastossa on valmiina paljon ominaisuuksia lisäämään tietoturvaa. Kirjaston tarkoituksena on hoitaa käyttäjän ja käyttäjän istuntoon liittyvät asiat. Devise esimerkiksi hoitaa istuntoavaimen salauksen, tarkistaa, että käyttäjien syöttämät sähköpostit ovat rekisteröityessä oikean muotoiset, validoi lomakkeiden tiedot ja tallentaa salasanan tietokantaan salattuna. Jos käyttäjä esimerkiksi yrittää tilata salasanan sähköpostille, jota ei ole olemassa, kuva 10 näyttää miten Devise oletuksena kertoo ettei kyseisestä sähköpostia löytynyt. Näitä virheilmoituksia on mahdollista muokata Railsin lokalisointi-tiedostoissa. (Devise 2018.) Lokalisointi tarkoittaa käyttöliittymän kääntämistä eri kielille.



Kuva 10. Kuva virheellisestä sähköpostista unohditko salasanasi-sivulla.

Kolmantena on arkaluontoisen datan altistuminen, mutta varsinaista arkaluonteista dataa, kuten luottokorttitietoja, terveystietoja tai muita GDPR:n listaamia arkaluonteisia henkilötietoja, ei kerätä käyttäjiltä. Seuraavana listalla on XML entiteettien haavoittuvuudet, mutta sovelluksessa ei käsitellä XML:ää. Viidentenä listalla on rikkinäinen kulunvalvonta. Kulunvalvontaan kuuluu näyttää käyttäjälle vain ne ryhmät, joihin käyttäjä on lisätty. Sivulle, jossa listataan kaikki ryhmät, näytetään vain kirjautuneen käyttäjän ryhmät. Se huolehditaan ryhmät kontrollerin index-funktiossa kutsumalla `current_user.groups`. Current_user on Devise-kirjaston tarjoama apumetodi. (Devise 2018.)

```
class GroupsController < ApplicationController
  def index
    @groups = current_user.groups.all
  end
```

Kuudentena listalla olevaan turvallisuusasioiden virheelliseen määritykseen kuuluu esimerkiksi kirjastojen ja kehysten pitäminen ajan tasalla, eli kirjastot tulee päivittää aina olemaan ajan tasalla kun uusia versioita tulee. Myös Rails kehys on syytä päivittää, mikäli siitä löydetään haavoittuvuuksia. Turvallisuusasioiden virheelliseen määritykseen kuuluu lisäksi huolehtiminen siitä, että dokumenttivarasto ei ole avoinna verkossa. XSS-haavoittuvuudet ovat listalla seitsemäntenä ja ne ovat myös hoidettu oletuksena Rails kehyksessä. Kun merkkijono näytetään näkymissä, se muunnetaan turvalliseksi (escaped) ennen sen lähettämistä selaimelle. Kahdeksantena on epäluotettava deserialisointi ja se hoidetaan web-sovelluksessa tarkistamalla palvelimella käyttäjän antamat syötteet. Esimerkiksi ryhmän tietoja muuttaessa etsitään syötteen ryhmä id:tä vastaava ryhmä kirjautuneen käyttäjän ryhmistä. Näin varmistetaan, ettei käyttäjä voi muokata muiden ryhmien tietoja.

```
  def group
    @group ||= current_user.groups.find(params[:id])
  end
```

Yhdeksäntenä on komponentit tunnetuilla haavoittuvuuksilla, mutta sellaisia komponentteja ei sovelluksessa ole käytössä. Sovelluksessa on käytössä vain tunnettuja ja yleisesti käytössä olevia kirjastoja ja lisäosia. Viimeisenä listalla on riittämätön valvonta ja lokitta-

minen, jotka saattavat johtaa siihen, ettei tietomurtoja huomata ajoissa, joten on tärkeä seurata sovelluksen käyttöä ja lisätä tarvittaessa lokitusta.

Myös Top 10 listan ulkopuolisia tietoturva-uhkia on hoidettu. Esimerkiksi Cross-Site Request Forgeryltä, eli CSRF:ltä on suojauduttu syöttämällä automaattinen tunnisteavain lomakkeisiin. Avain on serverin generoima ja kun lomake lähetetään, serveri tarkistaa että avain täsmää generoituun tunnisteavaimeen. Ilman avainta, lisäys-kyselyä (POST) ei voida suorittaa. Rails lisää automaattisesti tämän tunnisteavaimen pakollisuuden ApplicationControlleriin, jonka ominaisuudet kaikki muut kontrollerit oletuksena perivät. (Rails Guides 2018b; EngineYard 2017.) Käytännössä tämä tarkoittaa sitä, että dokumentteja ei pystytä lisäämään tai poistamaan muuten kuin varsinaisen lomakkeen kautta. Ja jotta lomake aukeaa, vaaditaan käyttäjältä tunnistautuminen järjestelmään ja kyseiseen ryhmään kuuluminen.

Lisäksi ryhmien kontrollerissa on määritelty, mitä arvoja ryhmän lisäyksessä tai muokkauksessa saadaan muuttaa. Näitä kutsutaan vahvoiksi parametreiksi.

```
private def group_params
  params.require(:group).permit(
    :name,
    user_ids: [],
    attachments_attributes: [:id, :attachment, :_destroy])
end
```

Tässä tapauksessa voidaan muuttaa ryhmän nimeä, ryhmään kuuluvien käyttäjien id:tä, sekä dokumenttien attribuutteja.

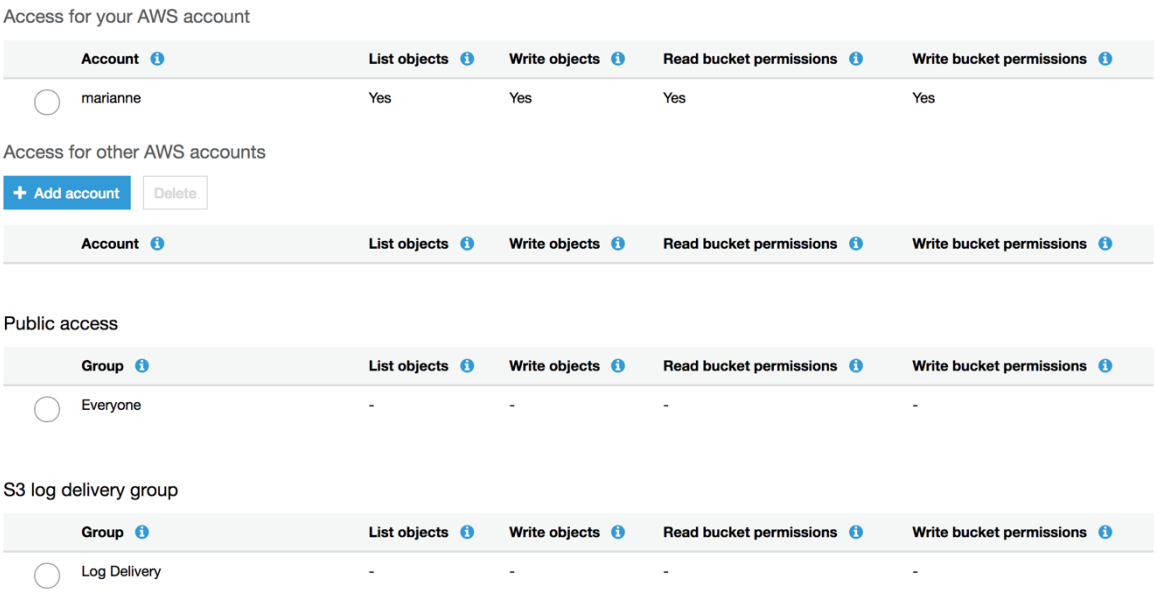
Amazon S3 varaston käyttöön tarvittavat tunnukset ovat tallennettu muuttujiin, joita ei tallenneta edes versionhallintaan. Muuttujat säilytetään .env.local-tiedostossa ja sen pääsy versiohallintaan estetään lisäämällä kohta `.env.local` .gitiginore-tiedostoon. Shrine-kirjaston luoma Shrine-luokka käyttää muuttujiin tallennettuja S3-tunnuksia dokumenttien lisäämisessä, poistamisessa ja siirtämisessä eri varastoihin.

Railsin ja S3 välinen yhteys on suojattu HTTPS:llä. HTTP-yhteys käyttää salaamatonta TCP-protokollaa, mutta HTTPS käyttää selaimen ja palvelimen välillä SSL/TLS-salausprokollaa. Tämä vähentää mahdollisuuksia tiedon salakuunteluun. (Yksityisyydensuoja 2018.)

Amazon S3 varastolle määritellään Permissions-osassa niin sanotut CORS-konfikuraatiot. CORS-konfikuraatiot ovat XML-muotoisia sääntöjä. Sääntöihin on määritelty esimerkiksi, mistä osoitteesta tulevat kutsut ovat sallittuja ja mitkä metodit ovat sallittuja. Nämä säännöt koskevat selaimesta tehtyjä HTTP-pyyntöjä. Alla olevassa esimerkissä sallittuja osoitteita ovat http://localhost:3000 ja http://rails.fi:3000, ja sallittuja metodeja ovat GET (hakee dokumentit), HEAD (hakee metatiedot), POST (lisää uuden dokumentin) ja PUT (päivittä olemassa olevaa dokumenttia).

```
<?xml version="1.0" encoding="UTF-8"?>
<CORSConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<CORSRule>
    <AllowedOrigin>*</AllowedOrigin>
    <AllowedMethod>GET</AllowedMethod>
    <AllowedMethod>HEAD</AllowedMethod>
    <MaxAgeSeconds>3000</MaxAgeSeconds>
    <ExposeHeader>Accept-Ranges</ExposeHeader>
    <ExposeHeader>Content-Encoding</ExposeHeader>
    <ExposeHeader>Content-Length</ExposeHeader>
    <ExposeHeader>Content-Range</ExposeHeader>
    <AllowedHeader>Range</AllowedHeader>
    <AllowedHeader>Authorization</AllowedHeader>
</CORSRule>
<CORSRule>
    <AllowedOrigin>http://localhost:3000</AllowedOrigin>
    <AllowedOrigin>http://rails.fi:3000</AllowedOrigin>
    <AllowedMethod>GET</AllowedMethod>
    <AllowedMethod>POST</AllowedMethod>
    <AllowedMethod>PUT</AllowedMethod>
    <AllowedHeader>*</AllowedHeader>
</CORSRule>
</CORSConfiguration>
```

Lisäksi Permissions-osassa määritellään kuvassa 11 esitellyt pääsyt dokumenttivarastoon. Kuvasta näkee, että vain käyttäjällä marianne on oikeus AWS tilille, eikä varastolle ole julkista pääsyä. Juuri Public access on se kohta, joka on aiheuttanut tietovuotoja, joista empiirisessä osassa mainittiin.



Kuva 11. Amazon S3 Access Control lista.

4 Pohdinta ja jatkokehitys

Seuraavaksi pohdin tavoitteisiin pääsemistä, aikataulua, mitä opittiin ja mitkä tuottivat haasteita. Lisäksi käsitellään jatkokehitysideoita.

4.1 Pohdinta

Opinnäytetyön tavoitteena oli toteuttaa web-sovellus, jonne voi rekisteröityä millä tahansa sähköpostilla, luoda ryhmiä jo rekisteröityneiden käyttäjien kesken ja jakaa ryhmien sisällä dokumentteja. Sovellusta ei ole julkaistu, eikä julkaiseminen tämän työn tavoitteena ollut-kaan. Tämän web-sovelluksen julkaiseminen vaatisi tietosuojan ja käyttöehtojen osalta ulkopuolista konsultointia ja koska tämä työ ei ole toimeksiantona tehty, ei konsultointia toistaiseksi suoriteta.

Toteutetun kaltainen web-sovellus olisi ollut hyödyllinen omissa opinnoissani. Useasti ryhmätöitä tehdessä ryhmä lopulta päätyi lähettämään sähköpostilla eri versioita dokumenteista. Tämä kävi suhteellisen haasteelliseksi. Google Drive tai Dropbox ovat joillain kursseilla olleet käytössä, ja ne ovatkin toimineet erittäin hyvin. Ongelmia tulee silloin, kun osalla ryhmän jäsenistä ei ole Google- tai Dropbox-tiliä. Tämän vuoksi web-sovelluksen yksi tärkeimmistä kriteereistä oli se, että sovellukseen pitää pystyä rekisteröityä millä tahansa sähköpostilla.

Opinnäytetyöprojekti antoi erittäin kattavan kuvan ohjelmistokehityksestä kokonaisuudessaan. Useasti olen ollut mukana vain yhdessä roolissa, joko toteuttamassa, suunnittele-

massa tai testaamassa. Valitut teknologiat toimivat mielestäni tässä tarkoituksessa oikein hyvin, vaikka toivoin, että Amazon pilvipalveluntarjoajana olisi tullut minulle tutummaksi. Nyt pääsin toistaiseksi näkemään vain hyvin pienen osan siitä. Paljon ideoita tuli lisää matkan varrella, kuten jatkokehitys-kappaleen pituudesta huomaa. Tässä olisi auttanut prototyyppi, joka olisi toteutettu ennen ohjelmistokehitystä. Prototyyppiä olisi voinut testata todellisilla käyttäjillä.

Aikataulussa oli suunniteltuna 16 viikkoa ja siinä pysyttiin, tosin suunniteltujen viikkojen aiheet muuttuivat verrattuna toteutuneisiin viikkoaiheisiin. Aikataulu oli suunniteltu siten, että ensin toteutettaisiin empiirinen osa ja sitten kirjoitettaisiin tietoperusta. Todellisuudessa kuitenkin tein sekaisin empiiristä osaa ja tietoperustaa, sillä se toi kaivattua vaihtelua, ja osaltaan tietoperustan selvittäminen ensin helpotti empiirisen osan toteutusta. Aikataulu oli tiukka ja työ toteutettiin päivätyön ohessa. Viikkoaiheet olisi siis voinut suunnitella paremmin.

Jos lähtisin nyt toteuttamaan uutta ohjelmistoprojektia yksin, tekisin joitain asioita toisin. Suunnittelisin enemmän, tekisin prototyypin ja pyytäisin testikäyttäjiä. Koen opinnäytetyön olleen hyvä oppimisprojekti minulle, koska sain tämän projektin aikana paljon selkeämmän kuvan siitä, mitä ohjelmistokehitysprojekti pitää sisällään.

4.2 Jatkokehitys

Kehitetyn web-sovelluksen seuraava tavoite tulisi olla julkaiseminen. Ennen kuin tuohon julkaisu-vaiheeseen voitaisiin päästä, tulisi hoitaa sovelluksen tietosuoja-asiat kuntoon, eli sovelluksen tulisi täyttää GDPR:n vaatimukset. Sovellukseen tulisi ensimmäisenä kirjoittaa tietosuojaseloste, ja varmistaa käyttäjiltä tietosuojaselosteen hyväksyntä. Tämä hyväksyntä tulisi voida osoittaa, ja esimerkiksi lisätä omana kolumninaan tietokantaan Users-tauluun. Aikaisemmissa sovelluksissa olen käyttänyt kolumnia, jonka olen nimennyt ”accepted_at_privacy_policy”, tyypiltaan aikaleima, ja se tallentaa nimensä mukaisesti aikaleiman siitä hetkestä, kun käyttäjä klikkaa tietosuojaselosteen hyväksytyksi.

Tietosuoja-asetuksen lisäksi tällaisen kaltaisessa sovelluksessa on tärkeä huomioida, kuka on vastuussa, mikäli jotain epätoivottua tapahtuu, ja ryhmien dokumentit esimerkiksi katoavat. Nämä vastuut ja vastuunvapautukset tulisi kirjata web-sovelluksen käyttöehtoihin, jotka tulisi hyväksyä samaan tapaan kuin tietosuojaseloste. Sekä tietosuojaselosteessa, että käyttöehdoissa olisi järkevää konsultoida alan asiantuntijaa, esimerkiksi asianajotoimistoa, sillä oma osaaminen ei näihin riitä.

Ennen julkaisua tulisi myös parantaa muutama käytettävyyshaaste. Kun ryhmää luodaan, tulisi ryhmään lisättäviä käyttäjiä voida hakea nimellä tai sähköpostilla. Oletuksena on, että käyttäjiä tulisi olemaan kymmeniä tai satoja, joten listan selaaminen ilman hakutoimintoa veisi käyttäjät muihin sovelluksiin. Toinen käytettävyyshaaste jonka huomasin, on kuvaustekstin puuttuminen dokumenteista. Olisi hyvä, mikäli ryhmän jäsen voisi antaa kuvaustekstin lisätessään dokumenttia muille ryhmäläisille nähtäväksi. Tämän voisi toteuttaa esimerkiksi siten, että lisättäisiin Groups-tauluun kolumni description, joka olisi tietotyypiltään text, joka mahdollistaa pidemmän tekstin kirjoittamisen.

Lisäksi tekisin pienen lisäyksen käyttäjien toimintaan. Olisi hyvä, mikäli käyttäjät voisivat kutsua sovellukseen uusia käyttäjiä sähköpostitse. Varsinaisen kutsun ohjelmoiminen ei olisi iso työ, mutta prosessi pitäisi suunnitella kokonaisuutena ensin, ja se vaatisikin isomman työn. Esimerkiksi, miten ohjata käyttäjä ensin hakemaan käyttäjiä ja sitten lähettämään kutsu puuttuvan käyttäjän sähköpostiin. Tulisiko kutsujalle tieto kun kutsuttu rekisteröityy, voitaisiinko hänet automaattisesti liittää ryhmiin joihin häntä on aikaisemmin yritetty etsiä? Kun toimintalogiikka kutsuttaville käyttäjille olisi päätetty, varsinainen ohjelmointi voitaisiin toteuttaa.

Kun tietosuoja päivitys, käyttöehdot ja käytettävyysparannukset ovat tehty, voisi sovelluksen julkaista esimerkiksi Heroku-pilvipalvelussa. Julkaisun jälkeen voisi suorittaa käyttäjäkyselyn käyttäjille, ja kehittää sovellusta tulleiden palautteiden perusteella. Kysely voisi sisältää myös kysymykset käyttäjien kutsumisesta, mikäli kyseistä ominaisuutta ei olisi toteutettu ennen julkaisua.

Lähteet

ApiDock. 2018. Persisted? Luettavissa: https://apidock.com/rails/v4.2.7/ActiveRecord/Persistence/persisted%3F. Luettu: 3.3.2018

The Art of Service. 2010. Cloud Computing Best Practices Specialist Guide for Storage Management and Platform as a service (PaaS): Understanding and Applying PaaS Solutions. Australia.

AWS. 2018e. Amazon S3. Luettavissa: https://aws.amazon.com/s3/. Luettu 3.3.2018

AWS. 2018d. AWS Free Tier. Luettavissa: https://aws.amazon.com/free/. Luettu: 12.2.2018.

AWS. 2018b. AWS SDK for Ruby. Luettavissa: https://aws.amazon.com/sdk-for-ruby/. Luettu: 12.2.2018.

AWS. 2018a. Case Studies. Luettavissa: https://aws.amazon.com/solutions/case-studies/all/. Luettu: 12.2.2018.

AWS. Cloud File Storage. Luettavissa: https://aws.amazon.com/what-is-cloud-file-storage/. Luettu: 1.5.2018.

AWS. 2018c. What is Amazon S3? Luettavissa: https://docs.aws.amazon.com/AmazonS3/latest/dev/Welcome.html. Luettu: 12.2.2018.

Boucheron, B. 2017. Object Storage vs. Block Storage services. Luettavissa: https://www.digitalocean.com/community/tutorials/object-storage-vs-block-storage-services. Luettu: 1.5.2018.

CloudRanger. 2017. 10 Reasons to choose AWS as your cloud platform. Luettavissa: https://cloudranger.com/10-reasons-to-choose-aws/. Luettu: 8.2.2017.

Cloud Security Alliance. Top Threats to Cloud Computing Plus: Industry Insights. Luettavissa: https://cloudsecurityalliance.org/download/top-threats-cloud-computing-plus-industry-insights/. Luettu: 29.3.2018

CloudStorageBest. 2013. Different Cloud Storage Types. Luettavissa: http://www.cloudstoragebest.com/cloud-storage-types/. Luettu: 1.5.2018.

CloudStorageBest. 2012. What is Cloud Storage, How Does it Work, Its Benefits & Uses Explained. Saatavissa: http://www.cloudstoragebest.com/what-is-cloud-storage/. Luettu: 19.4.2018.

ComputerHope. 2017. Cache. Luettavissa: https://www.computerhope.com/jargon/c/cache.htm. Luettu: 3.3.2018.

ComputerWeekly. 2016. Cloud Storage Appliances: What they are and who provides them. Luettavissa: http://www.computerweekly.com/feature/Cloud-storage-appliances-What-they-are-and-who-provides-them. Luettu: 11.3.2018.

ComputerWeekly. 2015b. Cloud Storage: The top five things that go wrong – and how to avoid them. Luettavissa: http://www.computerweekly.com/feature/Cloud-storage-Five-things-that-go-wrong-and-how-to-avoid-them. Luettu: 11.3.2018.

ComputerWeekly. 2015. Cloud storage cloud vs in house. Luettavissa: http://www.computerweekly.com/feature/Object-storage-Cloud-vs-in-house. Luettu: 11.3.2018.

ComputerWeekly. 2009. A history of cloud computing. Luettavissa: http://www.computerweekly.com/feature/A-history-of-cloud-computing. Luettu: 14.2.2018.

ComputerWeekly. 2017. Storage 101: Object storage vs block vs file. Luettavissa: https://www.computerweekly.com/feature/Storage-101-Object-storage-vs-block-vs-file. Luettu: 1.5.2018.

Datamation. 2017a. Cloud Computing Companies. Luettavissa: https://www.datamation.com/cloud-computing/cloud-computing-companies.html. Lettu: 14.2.2018.

Datamation. 2017b. Private Cloud Computing Providers. Luettavissa: https://www.datamation.com/cloud-computing/private-cloud-providers.html. Luettu: 30.4.2018.

DB-Engines. 2018a. DB-Engines Ranking. Luettavissa: https://db-engines.com/en/ranking. Luettu: 7.3.2018.

DB-Engines. 2018b. PostgreSQL is the DBMS of the Year 2017. Luettavissa: https://db-engines.com/en/blog_post/76. Luettu: 7.3.2018.

Devise. 2018. Devise ReadMe. Luettavissa: https://github.com/plataformatec/devise. Luettu: 7.2.2018.

Digital Ocean. 2014. SQLite vs MySQL cs PostgreSQL. Luettavissa: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems. Luettu: 14.12.2018.

Fortune. 2017. Amazon Still Leads Cloud Rankings, But Competition Is Coming On Strong. Luettavissa: http://fortune.com/2017/06/15/gartner-cloud-rankings/. Luettu: 30.4.2018.

EngineYard, 2017. Ruby on Rails Security 17-item checklist. Luettavissa: https://www.engineyard.com/blog/ruby-on-rails-security-checklist. Luettu: 8.4.2018.

eWeek. 2018. How to Select the Right Cloud Storage for Your Compnay. Luettavissa: http://www.eweek.com/cloud/how-to-select-the-right-cloud-storage-for-your-company. Luettu: 11.3.2018.

Gartner. 2018. Gartner Forecasts Worldwide Public Cloud Revenue to Grow 21.4 Percent in 2018. Luettavissa: https://www.gartner.com/newsroom/id/3871416. Luettu: 30.4.2018.

GitHub. 2017. Dropzonejs-rails. Luettavissa: https://github.com/ncuesta/dropzonejs-rails. Luettu: 3.3.2018

GitHub. 2018. The state of the octoverse. Luettavissa: https://octoverse.github.com/. Luettu: 8.2.016.

Heino, Petteri. 2010. Pilvipalvelut. Kariston Kirjapaino Oy. Hämeenlinna.

IBM. What is cloud storage? Luettavissa: https://www.ibm.com/cloud/learn/what-is-cloud-storage. Luettu: 1.5.2018.

Kirkconnell, K. 1.1.2016. It's a Mediocre Idea to Permanently Store Large Objects in a Database. Couchbasen blogi. Luettavissa: https://blog.couchbase.com/large-objects-in-a-database/. Luettu: 3.3.2018.

Mooc. 2018. Tietokantojen perusteet. Luettavissa: https://materiaalit.github.io/tikape-k18/part2/. Luettu: 7.2.2018.

New Jersey Institute of Technology. 3 Security Risks to Cloud Storage. Luettavissa: https://graduatedegrees.online.njit.edu/resources/mscs/mscs-articles/3-security-risks-to-cloud-storage/. Luettu: 29.3.2018.

O'Reilly, J. 2017. 7 Ways to Secure Cloud Storage. Luettavissa: https://www.networkcomputing.com/data-centers/7-ways-secure-cloud-storage/866645128. Luettu: 29.3.2018.

OWASP. 2017. The Ten Most Critical Web Application Security Risks. Luettavissa: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf Luettu: 19.4.2018.

PayPall. 2018. List of Third Parties (other than PayPall Customers) with Whim Pesonal Information May be Shared. Luettavissa: https://www.paypal.com/gi/webapps/mpp/ua/third-parties-list. Luettu: 14.4.2018.

PosgresSql. 2016. Binary Files In Db. Luettavissa: https://wiki.postgresql.org/wiki/BinaryFilesInDB#Storing_the_large_binary.2A_file_aka_unstructured_data_streams_in_a_database. Luettu: 3.3.2018

PostgresSql. JSON types. Luettavissa: https://www.postgresql.org/docs/9.4/static/datatype-json.html. Luettu: 27.4.2018.

Rails Guides. 2018b. The Rails Command Line. Luettavissa: http://guides.rubyonrails.org/command_line.html. Luettu: 7.2.2018.

Rails Guides. 2018a. Ruby on Rails Security Guide. Luettavissa: http://guides.rubyonrails.org/security.html. Luettu: 8.4.2018.

Red Hat. File storage, block storage, or object storage? Luettavissa: https://www.redhat.com/de/topics/data-storage/file-block-object-storage. Luettu: 1.5.2018.

Ruby Doc. 2018b. Simpe Form. Luettavissa: http://www.rubydoc.info/github/plataformatec/simple_form/master/frames. Luettu: 7.2.2018.

Ruby Doc. 2018a. Slim. Luettavissa: http://www.rubydoc.info/gems/slim/frames. Luettu: 7.2.2018.

SearchCloudComputing. 2014. Cloud Computing. Luettavissa: http://searchcloudcomputing.techtarget.com/definition/cloud-computing. Luettu: 14.2.2018.

SearchCludComputing. 2017. Infrastructure as a Service (IaaS). Luettavissa: https://searchcloudcomputing.techtarget.com/definition/Infrastructure-as-a-Service-IaaS. Luettu: 15.4.2018.

SearchStorage. 2017. Block Storage. Luettavissa: https://searchstorage.techtarget.com/definition/block-storage. Luettu: 1.5.2018.

SearchStorage. 2016. File Storage. Luettavissa: https://searchstorage.techtarget.com/definition/file-storage. Luettu: 1.5.2018.

SearchStorage. 2014. How an object store differs from file and block storage. Luettavissa: https://searchstorage.techtarget.com/feature/How-an-object-store-differs-from-file-and-block-storage. Luettu: 1.5.2018.

SDXCentral. What are cloud service providers. Luettavissa: https://www.sdxcentral.com/cloud/definitions/what-are-cloud-service-providers/. Luettu: 25.3.2018.

Shrine. 2018. Luettavissa: http://shrinerb.com/. Luettu: 12.2.2018.

Stackify. 2018. Most popular and influential programming languages of 2018. Luettavissa: https://stackify.com/popular-programming-languages-2018/. Luettu: 8.2.2018.

Tietosuojamalli. Luettavissa: https://fakta.tietosuojamalli.fi/aihe/gdpr. Luettu: 14.4.2018.

Techopedia. Provisioning. Luettavissa: https://www.techopedia.com/definition/4069/provisioning-computing-computing. Luettu: 30.4.2018.

Thair, A. 2017. Which is better? Saving files in Database or in File System. Luettavissa: https://habiletechnologies.com/blog/better-saving-files-database-file-system/. Luettu: 28.4.2018.

Tivi. 2018. Mistä näitä S3-vuotoja tulee? Amazon selittää. Luettavissa: https://www.tivi.fi/Kaikki_uutiset/mista-naita-s3-vuotoja-tulee-amazon-selittaa-6706173. Luettu: 29.3.2018.

Webopedia. Binary File. Luettavissa: https://www.webopedia.com/TERM/B/binary_file.html. Luettu: 30.4.2018.

Wikibon. 2016. Public Cloud IaaS is 3.5x the Size of True Private Cloud Adoption. Luettavissa: https://wikibon.com/public-cloud-iaas-is-3-5x-the-size-of-true-private-cloud-adoption/. Luettu: 30.4.2018.

Yksityisyydensuoja. 2018. Salatut sivut. Luettavissa: https://www.yksityisyydensuoja.fi/salatut-sivut. Luettu: 9.4.2018.