TAMPEREEN
AMMATTIKORKEAKOULU

FINDOCS-PILVIPALVELU

Tommi Mantila

Opinnäytetyö
Toukokuu 2016
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

MANTILA TOMMI:
FinDocs-pilvipalvelu

Opinnäytetyö 41 sivua
Toukokuu 2016

Opinnäytetyössä tutustutaan julkisen pilvipalvelun suunnitteluun, toteuttamiseen ja julkaisuun. Työn tarkoitus on ymmärtää julkisen web-palvelun työvaiheet suunnittelusta julkaisemiseen.

Opinnäytetyössä esitellään FinDocs-pilvipalvelun ominaisuudet, tekovaiheet ja käytetyt teknologiat. Lisäksi palvelu julkaistaan Amazon Web Services -palveluntarjoajalta hankitulle virtuaalipalvelimelle.

FinDocs on opinnäytetyöprojektina toteutettu kuluttajalle tarkoitettu pilvipalvelu, joka mahdollistaa rekisteröitymisen jälkeen FinDocs-älypuhelinsovelluksella dokumenttien kuvaamisen, luokittelun, kommentoinnin ja lähettämisen pilvipalveluun. Dokumentteja voi myöhemmin selata ja ladata verkkopalvelun puolelta selaimella tietokoneelle tai muulle laitteelle. Palvelun päätarkoitus on luoda helppo tapa digitoida dokumenttinsa, kuten sopimukset, laskut ja kuitit sekä säilyttää niitä varmassa tallessa keskitetysti luokiteltuna.

Opinnäytetyöprosessin aikana saatiin valmiiksi ensimmäinen julkaisukelpoinen versio, jossa on kaikki opinnäytetyön ydintarkoitukselle tarpeelliset ominaisuudet. Palvelun jatkokehitysmahdollisuudet ovat laajat. Uusien ominaisuuksien kehittäminen käytetyillä teknologioilla on helppoa.

Asiasanat: pilvipalvelut, web-kehittäminen, mobiilikehittäminen

ABSTRACT

ICT Engineering
Software Engineering

MANTILA TOMMI:
FinDocs cloud service

Bachelor's thesis 41 pages
May 2016

Purpose of this thesis is to study planning, implementing and publishing a public cloud service.

Features, development phases and used technologies of FinDocs cloud service are introduced in this thesis. In addition the service is published to a virtual server hosted by Amazon Web Services.

FinDocs is a thesis project thats purpose is to provide users a possibility to take pictures of their essential documents with their smart phones with FinDocs- mobile application. After taking a picture user can tag, describe and finally send their document to the cloud service. Users can later browse and download their documents in FinDocs-web application with their browser. The main purpose of the service is to provide an easy way for users to digitize their important documents like bills, agreements and receipts and store them reliably and categorized in a web service that is available from any browser anywhere.

First functional implementation of FinDocs-service was achieved in the thesis process. Possibilities for further development are enormous. New features are easy to implement with technologies used.

Key words: cloud service, web development, mobile development

SISÄLLYS

ERITYISSANASTO

Pilvipalvelu	Internetissä sijaitseva palvelu, jonka käyttöoikeutta tyypillisesti myydään tietyin aikavälein laskuttaen
HTTP	TCP-yhteyteen perustuva yleinen tiedonsiirtoprotokolla
Backend	Web-sovelluksen palvelinpää
Frontend	Web-sovelluksen selainpää
JavaScript	Selaimessa suoritettava web-ohjelmointikieli
PHP	Palvelimella suoritettava web-ohjelmointikieli
CodeIgniter	PHP-sovelluskehys, joka yksinkertaistaa ohjelmointityötä
Backbone	JavaScript-sovelluskehys, joka yksinkertaistaa ohjelmointityötä
Amazon Web Services	Amazonin tarjoama kokoelma web-kehittäjille suunnattuja pilvipalveluita
Ajax	Asynkroninen tapa tehdä HTTP-kutsuja JavaScript-koodissa
SSH	Salakirjoitettu verkkoprotokolla, jolla voidaan ottaa salatusti yhteyttä toiseen tietokoneeseen
Eväste	Internet sivustoilla käytettävä avain-arvo -pari, joka lähetetään palvelimelle jokaisen HTTP-kutsun yhteydessä esimerkiksi käyttäjän tunnistamiseksi
JSON	Yleisesti käytössä oleva tiedonsiirtoformaatti, joka perustuu avain-arvo -pareihin
SQL	Relaatiotietokantojen käsittelyyn tarkoitettu kieli
SQL-injektio	Verkkohyökkäys, jossa pyritään käyttäjäsyötteen avulla muokkaamaan tietokantahakuja

1 JOHDANTO

Tässä opinnäytetyössä perehdytään pilvipalvelun ja siihen liittyvän mobiilisovelluksen suunnitteluun, toteutukseen ja julkaisuun. Teknologioina käytetään verkkopalvelun osalta palvelinpäässä PHP:tä CodeIgniter-kirjastolla ja MySQL-tietokantaa. Selainpuolella käytetään pääasiassa HTML:ää, CSS:ää ja JavaScriptiä. JavaScript-kirjastoista merkittävimmät ovat jQuery, Backbone ja Marionette. Mobiilisovellus toteutettiin vain Android-älypuhelimelle käyttäen Androidin natiiviohjelmointia, eli käytännössä Javaa ja XML:ää.

Opinnäytetyön tarkoitus on kehittää järjestelmä, joka mahdollistaa yksityishenkilöille tärkeiden dokumenttien digitoimisen älypuhelimella ja siirtämisen luokiteltuna pilvipalveluun. Palvelun tarkoitus on poistaa kadonneisiin sopimuksiin, kuitteihin ja laskuihin liittyvät ongelmat, koska FinDocsia käyttämällä dokumentit löytyvät luokiteltuna, keskitettynä ja varmuuskopioituna aina samasta paikasta.

Toisessa luvussa käsitellään pilvipalvelua käsitteenä ja pilvipalvelulle tyypillisiä ominaisuuksia.

Kolmannessa luvussa esitellään palveluun käytetyt teknologiat, mukaan lukien ohjelmointi, tietokannat, versionhallinta ja julkaisualusta.

Neljännessä luvussa esitellään palvelun rakentamisen työvaiheet ja tärkeimmät ohjelmointiratkaisut.

Viidennessä luvussa pohditaan projektin onnistumista ja palvelun jatkokehitystä mahdollisien uusien ominaisuuksien sekä parannuksien osalta.

2 PILVILASKENTA JA PILVIPALVELUT

Tämä luku käsittelee pilvilaskentaa ja -palveluita. Luvussa käydään läpi pilvipalvelun käsite sekä yleisimmät pilvipalvelutyypit. Lisäksi luvussa tutustutaan FinDocs-palveluun pilvipalvelunäkökulmasta.

2.1 Pilvilaskenta ja pilvipalvelut

Pilvilaskenta ja -palvelut ovat uusi nimitys vanhalle ilmiölle. Määritelmä riippuu osittain näkökulmasta, mutta eräs osuva määritelmä kuvaa pilvilaskentaa ulkoiseksi internetissä tapahtuvaksi IT-toimenpiteeksi, joka tehostaa ja helpottaa käyttäjän tarpeita. Pilvipalvelut ovat palveluita, jotka tarjoavat pilvilaskentaa organisaatioille ja yksityisille käyttäjille. (Hassan, Qusay 2011. Demystifying Cloud Computing.)

Pilvipalvelut voivat olla tietokonekomponentteja, tietoliikennettä, tallennustilaa tai ohjelmistoja. Seuraavaksi esitellään yleisimmät pilvipalvelujen tyypit.

2.1.1 Infrastructure-as-a-Service (IaaS) –pilvipalvelutyyppi

IaaS-palvelut tarjoavat käyttäjälle laitteistoa käytettäväksi internetin välityksellä. Tällaisia laitteistoja ovat esimerkiksi palvelimet sisältäen prosessorilaskentaa, muistia ja tietoliikenneyhteyksiä. Yksi tunnetuimmista IaaS-palvelujen tarjoajista on Amazon Web Services, jota käytetään myös FinDocs-palvelun palvelimen ylläpidossa. (Hassan, Qusay 2011. Demystifying Cloud Computing.)

2.1.2 Platform-as-a-Service (PaaS) –pilvipalvelutyyppi

PaaS-palvelut tarjoavat ohjelmistokehittäjille kehitys- tai julkaisualustan internetin välityksellä. Alustaan kuuluu tyypillisesti esimerkiksi käyttöjärjestelmä, ohjelmointikielen ajoympäristö, web-pavelinohjelmisto ja tietokanta. Ohjelmistokehittäjät voivat kehittää ja ajaa ohjelmiaan pilvessä. (Hassan, Qusay 2011. Demystifying Cloud Computing.)

2.1.3 Software-as-a-Service (SaaS) –pilvipalvelutyyppi

SaaS-palvelut tarjoavat kuluttajalle tai organisaatioille pääsyn internetissä toimivaan sovellukseen, jolloin sovellusta ei tarvitse rakentaa tai ostaa, vaan asiakas tyypillisesti maksaa tasaisin väliajoin palvelun käytöstä. (Hassan, Qusay 2011. Demystifying Cloud Computing.)

2.2 FinDocs-pilvipalvelu

Opinnäytetyönä tehty FinDocs on tyypillinen SaaS-pilvipalvelu. Palvelu tarjoaa käyttäjälle mahdollisuuden säilyttää dokumenttejaan tallessa keskitetysti FinDocsin palvelimella. Lisäksi palveluun sisältyy mobiiliapplikaatio, jolla käyttäjät voivat kirjautua palveluun ja ottaa Android-puhelimensa kameralla kuvia dokumenteista ja lähettää ne pilveen.

FinDocsin luomisessa ilmenevät selvästi yleisimmät pilvipalvelujen tyypit. Palvelun kehittäjä tilaa Amazon Web Servicesiltä (AWS) IaaS-palvelun, joka sisältää virtuaalisen tietokoneen tietyllä laskentateholla. Lisäksi AWS tarjoaa kehittäjälle PaaS-palvelun samassa yhteydessä, kun virtuaalikoneeseen valitaan käyttöjärjestelmä. Kun sovellus on valmis, kehittäjä tarjoaa edellä mainittujen alustojen päällä toimivan SaaS-palvelun loppukäyttäjille. Prosessi helpottaa huomattavasti lopullisen palvelun kehittäjää, verrattaessa siihen, että hänen täytyisi itse huolehtia laitteistosta, verkkoyhteyksistä ja muista palvelun ylläpitämiselle välttämättömistä asioista.

3 TEKNOLOGIAT JA TYÖKALUT

Tässä luvussa käydään läpi FinDocs-pilvipalvelussa käytettävät teknologiat ja työkalut. Ensimmäiseksi kerrotaan palvelinpään ratkaisuista, jonka jälkeen siirrytään käyttäjälle näkyviin teknologioihin.

3.1 Amazon EC2-virtuaalipalvelin

Amazon EC2 on Amazon Web Services -pilvipalvelukirjoon kuuluva IaaS/PaaS-palvelu, jonka tarkoitus on tarjota skaalautuvaa pilvilaskentaa virtuaalipalvelinten muodossa. EC2:sen avulla käyttäjä voi luoda muutamassa minuutissa virtuaalipalvelimen valitsemallaan käyttöjärjestelmällä. Käyttöjärjestelmävaihtoehdot ovat pääasiassa Linux-jakeluita. (Amazon Web Services. Amazon EC2 Product Details.)

FinDocs käyttää EC2-instanssia, johon on asennettu 64-bittinen Ubuntu-käyttöjärjestelmä. Ubuntu valittiin, koska sitä käytetään myös tietokoneilla, joilla palvelua ohjelmoidaan. Näin esimerkiksi ohjelmien konfiguraatiotiedostot löytyvät samoista paikoista. EC2-palvelun käytöstä laskutetaan kuormituksen ja datan siirron mukaan, mutta FinDocsissa käytetään palvelun ilmaisversiota. Kuormituksen kasvaessa palvelu voi mahdollisesti alkaa maksamaan. Nämä maksut hoitaa ainakin alkuvaiheessa FinDocsin ylläpitäjä. Mikäli palvelu saisi tulevaisuudessa laajempaa huomiota, voitaisiin esimerkiksi ottaa käyttöön kuukausimaksut tai alkaa näyttämään mainoksia.

3.2 Palvelinpää

Palvelinpää tarkoittaa web-applikaation osaa, joka tapahtuu piilossa käyttäjältä palveluntarjoajan palvelimella. FinDocsin kaltaisissa SaaS-palveluissa tähän sisältyy tyypillisesti vähintään tietokanta ja jonkin ohjelmointikielen ajoympäristö.

3.2.1 Nginx HTTP -palvelin

Nginx on ilmainen, avoimen lähdekoodin HTTP-palvelin. Nginx on tunnettu hyvästä suorituskyvystä, vakaudesta ja helposta konfiguroinnista. Nginx toimii palvelimena FinDocs-palvelussa, ja kaikki palveluun liittyvä verkkoliikenne kulkee sen kautta. (Nginx. NGINX Wiki's documentation.)

3.2.2 MySQL-tietokanta

MySQL on avoimen lähdekoodin tietokannan hallintajärjestelmä. Hallintajärjestelmän tehtävä on toimia rajapintana käyttäjän tai sovelluksen ja tietokannassa sijaitsevan datan välissä, luoden tiettyjä sääntöjä ja vaatimuksia. MySQL-tietokannat ovat relaatiotietokantoja. MySQL on yksi käytetyimmistä tietokannoista maailmassa. (MySQL 5.7 Reference Manual. What is MySQL?)

FinDocs käyttää MySQL-tietokantaa esimerkiksi tunniste-, dokumentti- ja käyttäjätietojen tallentamiseen. Lisäksi CodeIgniter PHP -sovelluskehys käyttää tietokantaa sessiodatan tallentamiseen.

3.2.3 PHP-ohjelmointi

PHP on avoimen lähdekoodin ohjelmointikieli, jota käytetään pääasiassa web-ohjelmistokehityksessä. PHP-koodia voi kirjoittaa HTML-merkkauskielen rinnalle, mutta PHP ajetaan aina palvelinpäässä. Tyypillisesti palvelin tulkitsee PHP-koodin ja palauttaa sivun selaimelle HTML-muodossa. (PHP. What is PHP?)

FinDocsissa PHP:tä käytetään pääasiassa rajapintana tietokannan hakuihin ja tallennuksiin. Käyttöliittymä luodaan FinDocsissa selainpäässä palvelimelta haettujen tietojen perusteella.

3.2.4 CodeIgniter PHP -sovelluskehys (framework)

CodeIgniter on PHP:n päälle rakennettu sovelluskehys. CodeIgniterin tarkoitus on tehdä PHP-kehittämisestä nopeampaa ja tehokkaampaa. Se sisältää runsaasti kirjastoja erilaisiin toimenpiteisiin kuten tietokannan, sähköpostin ja istuntojen käsittelyyn. (CodeIgniter. CodeIgniter at a Glance.)

CodeIgniter käyttää MVC (Model-View-Controller) -lähestymistapaa. MVC-lähestymistavan tarkoitus on erottaa ohjelmiston logiikka ja ulkoasu selkeästi toisistaan. Lähestymistavassa on eroteltu mallit (model), näkymät (view) ja käsittelijät (controller). Käsittelijät vastaanottavat ulkopuolelta komentoja, joiden perusteella ne käsittelevät tietoja mallien avulla tai ulkoasua näkymien avulla.

FinDocsissa CodeIgniteriä käytetään PHP-ohjelmoinnin tehostamiseen ja selkeyttämiseen. Kehittäjällä on CodeIgniteristä aikaisempaa kokemusta, joten sen valitseminen mahdollistaa ohjelmointityön aloittamisen nopeasti, eikä aikaa kulu turhaan dokumentaation lukemiseen.

3.3 Selainpää

Selainpää tarkoittaa loppukäyttäjää koskevaa osuutta web-applikaatiosta. Selainpäähän sisältyy tyypillisesti ulkoasu ja logiikkaa.

3.3.1 HTML ja CSS

HTML on merkkauskieli, jota käytetään web-sivustojen rakentamiseen. HTML:n tarkoitus on antaa sivustolle rakenne, jota selaimet ymmärtävät. CSS on merkkauskieli, joka kertoo selaimelle, miltä HTML-dokumentin tulee ulkoisesti näyttää.

Yhdessä HTML ja CSS ovat verkkosivujen ydinelementit. Niitä käytetään FinDocsissa verkkosivun pohjan luomiseen ja käyttöliittymän ulkoasun parantamiseen.

3.3.2 JavaScript-ohjelmointi

JavaScript on selaimessa ajettava ohjelmointikieli, jonka tarkoituksena on tehdä HTML-sivustoista interaktiivisia. JavaScript on erittäin suosittu ohjelmointikieli web-sivustoissa ja verkkopalveluissa. (AboutTech. What Is JavaScript?) JavaScript mahdollistaa HTML-sivun manipuloimisen sivun lataamisen jälkeen.

FinDocsissa JavaScriptiä käytetään erilaisten kirjastojen ja sovelluskehysten avulla datan hakemiseen palvelimelta ja käyttöliittymän manipuloimiseen.

3.3.3 jQuery JavaScript-kirjasto

jQuery on JavaScript-kirjasto, joka helpottaa huomattavasti HTML-dokumenttien manipulointia, tapahtumakäsittelyä (event handling), animointia ja asynkronisten verkkokutsujen (Ajax) suorittamista. Kaikki yleisimmät selaimet tukevat jQueryä, ja se on erittäin yleinen interaktiivisten web-sivustojen kehityksessä. (jQuery. What is jQuery.)

FinDocsissa jQueryä käytetään laajasti muiden kirjastojen ohella ulkoasun manipulointiin ja Ajax-kutsuihin.

3.3.4 Backbone Javascript -sovelluskehys

Backbone on JavaScript-sovelluskehys, jonka tarkoitus on luoda web-sovelluksen selainpäähän selvä rakenne. Backbonen tärkeimpiä ominaisuuksia on erotella ulkoasu ja tietorakenteet toisistaan. Backbone tallettaa dataa malleihin ja ulkoasun näkymiin. Kun mallin sisältö muuttuu, Backbonen avulla voi helposti välittää tapahtuman näkymälle, joka muuttaa sivuston ulkoasua mallin uuden sisällön mukaiseksi. (Backbone.js. Getting started.)

3.3.5 Marionette Backbone -sovelluskehys

Marionette on Backbone-sovelluskehys, jonka tarkoitus on yksinkertaistaa Backbonen käyttöä ja lisätä siihen kehitystyötä helpottavia ominaisuuksia. Tällaisia ominaisuuksia ovat esimerkiksi pohjanäkymät (layout views), jotka helpottavat käyttöliittymän rakentamista pienistä palasista sekä radio-ominaisuus, joka helpottaa ohjelman eri komponenttien keskustelua toistensa kanssa. (Marionette. The Backbone framework.)

FinDocsin käyttöliittymän logiikka on rakennettu kokonaisuudessaan Backbone- ja Marionette-yhdistelmällä.

3.4 Mobiilisovellus

FinDocs-mobiilisovellus on natiiviohjelmoitu Android-älypuhelinsovellus, jota käytetään dokumenttien kuvaamiseen ja lähettämiseen palveluun.

3.4.1 Java-ohjelmointikieli

Java on laajasti käytössä oleva oliopohjainen ohjelmointikieli. Java-sovellukset ovat laitteistoriippumattomia, sillä ne ajetaan virtuaalikoneessa Java-ajoympäristössä. (Java. What is Java technology and why do I need it?)

Android-sovelluksien natiivikieli on Java, ja sitä käytetään myös FinDocs-mobiilisovelluksen logiikassa.

3.4.2 XML

XML on alustariippumaton merkkauskieli, jolla voidaan esittää dataa.

Android-sovellus käyttää XML:ää ulkoasun luomisen työkaluna. Sitä käytetään myös FinDocs-mobiilisovelluksessa.

3.5 Git-versionhallintajärjestelmä ja Bitbucket-etärepositoriopalvelu

Versionhallintaan projektissa käytettiin Git-versionhallintajärjestelmää. Versionhallinta on järjestelmä, joka tallentaa tiedostossa tai tiedostojoukossa tapahtuvat muutokset niin, että käyttäjä voi palata aiemmin tallennettuihin versioihin myöhemmin. (Git. Alkusanat versionhallinnasta.)

Git käyttää projektien eri versioiden tallentamiseen repositorioita, joiden sijainti voi olla paikallinen tai pilvessä. Pilvessä sijaitsevat repositoriot mahdollistavat useamman kehittäjän osallistumisen projektiin, kun projektiin osallistuvat henkilöt voivat työntää (push) paikallisia versioita pilveen ja vetää (pull) pilvessä sijaitsevia versioita paikalliseen repositorioonsa.

FinDocs-palvelussa käytetään Bitbucket-pilvipalvelua Git-repositorioiden tallentamiseen. Vaikka FinDocs on yhden ihmisen tekemä projekti, pilvessä toimiva versionhallinta helpottaa esimerkiksi useassa eri sijainnissa työskentelyä ja toimii samalla myös varmuuskopiona projektista. Bitbucketia käytetään, koska se on ollut kehittäjällä käytössä myös työelämässä ja se on todettu toimivaksi sekä helppokäyttöiseksi.

4 TYÖVAIHEET JA OHJELMOINTIRATKAISUT

Tässä luvussa käydään läpi FinDocs-palvelun työvaiheet. Työvaiheiden esittely aloitetaan matalimmalta tasolta eli palvelimen pystytyksestä ja konfiguroinnista. Tämän jälkeen kerrotaan palvelinpuolen ohjelmointiratkaisuista. Viimeiseksi kerrotaan selain- ja mobiilipuolen ohjelmointiratkaisuista.

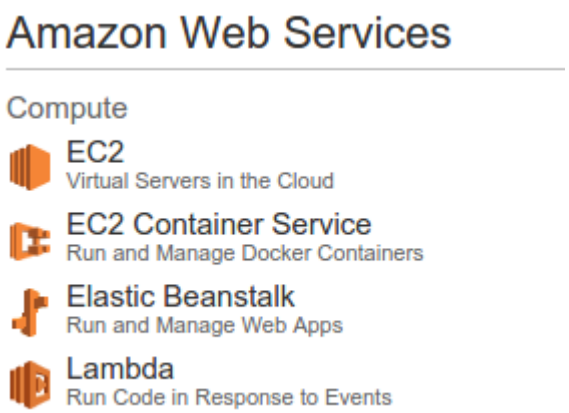
4.1 Palvelimen pystytys ja konfigurointi

Palvelimen pystytykseen ja konfigurointiin liittyy itse virtuaalipalvelimen pystyttäminen Amazon Web Services –pilvipalveluympäristössä, yhteyden muodostaminen palveluun SSH-tekniikan avulla ja tarvittavien palvelin- ja tietokantaohjelmistojen asennus sekä konfigurointi.

4.1.1 AWS (Amazon Web Services) EC2 -virtuaalipalvelimen pystytys

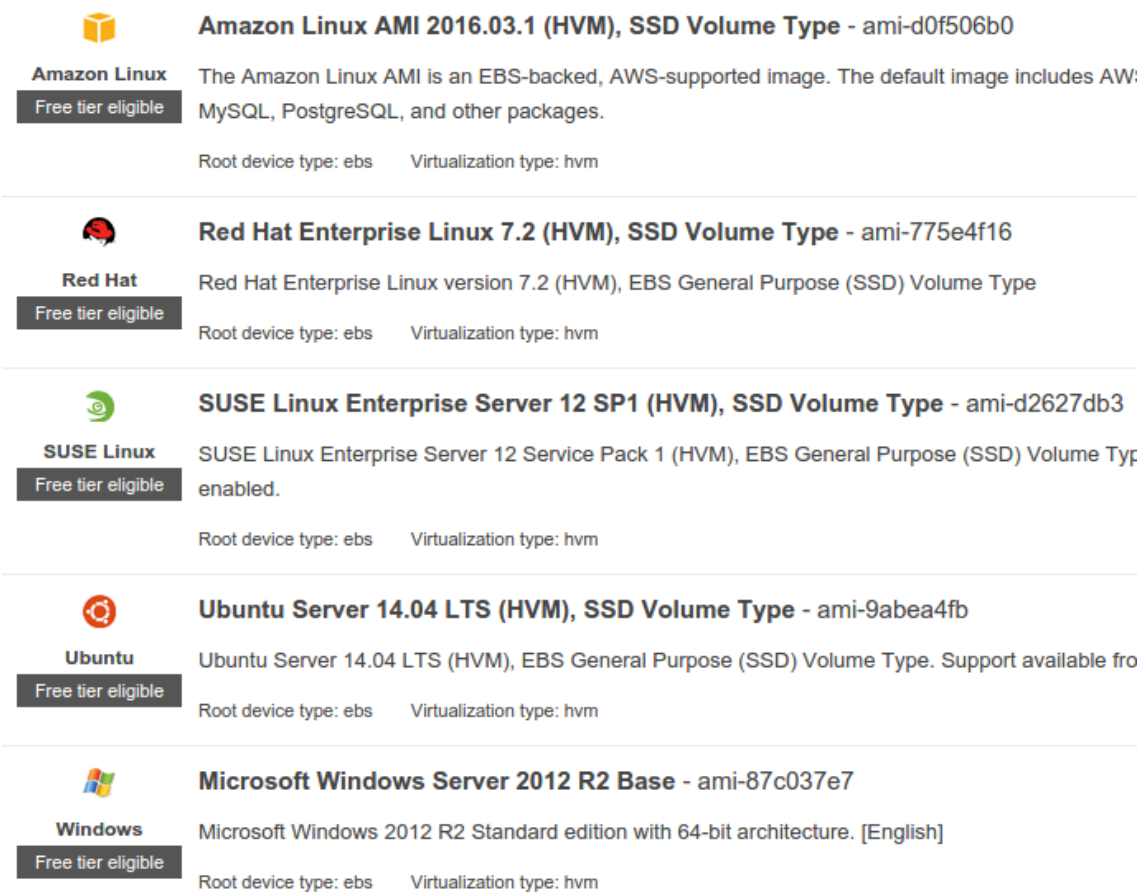
FinDocs-pilvipalvelun perustana toimii Linux-pohjainen virtuaalipalvelin. Palvelimen tehtävä on itse käyttäjälle näytettävän verkkopalvelun säilyttämisen lisäksi käyttäjä-, dokumentti- ja tunnistusdatan säilöminen tietokantaan, lähetettyjen dokumenttien tallentaminen palvelimen kovalevylle sekä rajapintana toimiminen edellämainitun datan tallentamiseen ja hakemiseen.

Amazon EC2-palveluun päädyttiin sen helpon käyttöönoton ja ylläpidettävyyden vuoksi. AWS:ään rekisteröitymisen jälkeen oman virtuaalipalvelimen käynnistäminen on parhaimmillaan muutaman klikkauksen päässä. AWS:ään rekisteröitymistä ja maksutietojen asettamista ei käsitellä tässä opinnäytetyössä.



KUVA 1. Osa AWS:n tarjoamista pilvilaskentapalveluista

Virtuaalipalvelimen käynnistämiseksi valitaan vaihtoehto EC2 kuvasta 1. Tämän jälkeen siirrytään EC2-yleisnäkymään, jossa voi esimerkiksi luoda uusia ja hallita olemassa olevia virtuaalipalvelininstansseja. Sivun oikeasta yläreunasta pääsee valitsemaan, missä konesalissa virtuaalipalvelin tulee sijaitsemaan. Yleisnäkymässä klikataan *Launch instance*-painiketta, jonka jälkeen siirrytään valitsemaan virtuaalipalvelintyyppiä.



KUVA 2. Pieni osajoukko EC2-virtuaalipalvelinalustojen vaihtoehdoista

Kuvan 2 mukaisesti EC2-palvelussa on mahdollista valita monista erilaisista virtuaalipalvelinalustoista. Alustoista käytetään kuvausta Amazon Machine Image (AMI), joka tarkoittaa valmiiksi asennettua ja konfiguroitua kuvausta halutusta järjestelmästä. EC2:ssa on tarjolla yli 5000 erilaista AMI:a. Osa niistä sisältää virtuaalikoneen lisäksi pelkän käyttöjärjestelmän ja osa sisältää valmiiksi konfiguroituja julkaisujärjestelmiä ja kaiken niiden käyttöön tarvittavan.

FinDocs palvelussa käytetään kuvassa 2 näkyvää Ubuntu-palvelinta. Alusta sisältää pelkän virtuaalikoneen ja käyttöjärjestelmän.

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

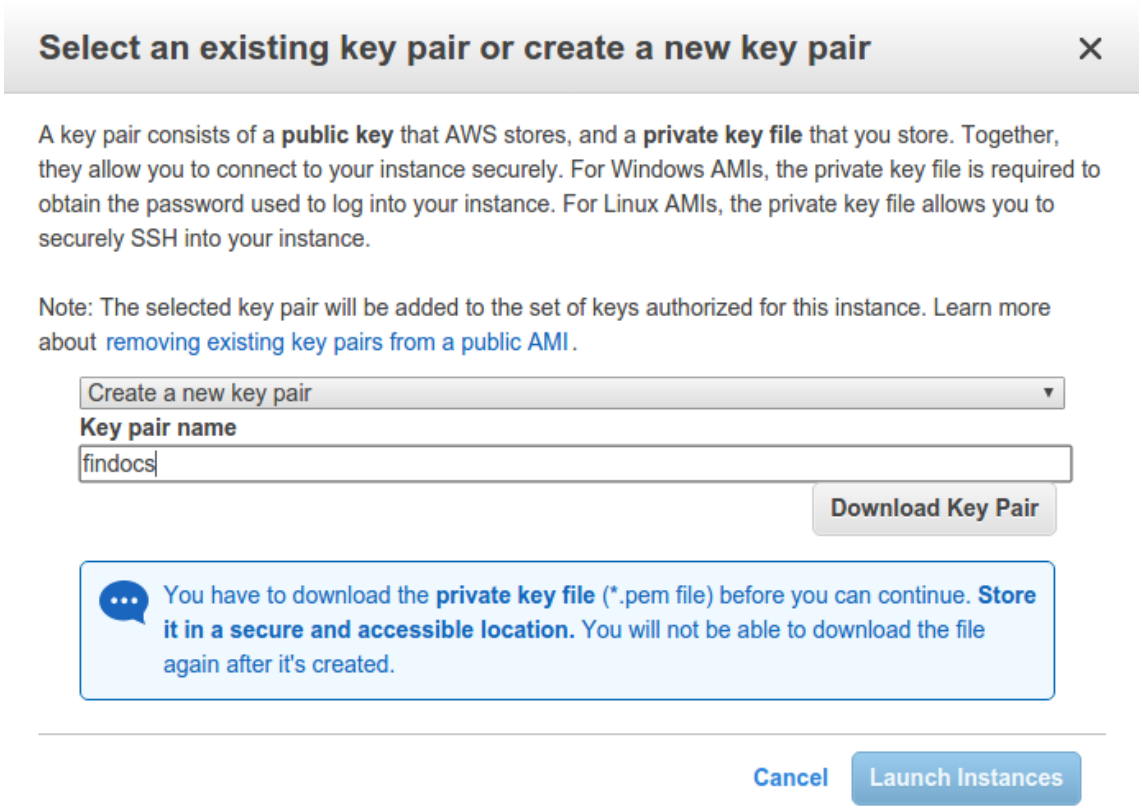
	Family	Type	vCPUs	Memory (GiB)
☐	General purpose	t2.nano	1	0.5
☑	General purpose	t2.micro Free tier eligible	1	1
☐	General purpose	t2.small	1	2
☐	General purpose	t2.medium	2	4
☐	General purpose	t2.large	2	8
☐	General purpose	m4.large	2	8
☐	General purpose	m4.xlarge	4	16
☐	General purpose	m4.2xlarge	8	32
☐	General purpose	m4.4xlarge	16	64
☐	General purpose	m4.10xlarge	40	160

KUVA 3. EC2-palvelimen instanssivaihtoehdot

Seuraavassa vaiheessa käyttäjän täytyy valita sopiva virtuaalikoneinstanssi. Kuvan 3 mukaisesti AWS tarjoaa laskentatehoa hyvin laajalta vaihteluväliltä. Virtuaalipalvelimia on saatavilla pienimmillään yhdellä suorittimella ja 500 megatavun keskusmuistilla varustettuna ja suurimmillaan kymmenillä suorittimilla ja sadoilla gigatavuilla keskusmuistia. Tehokkaammat palvelut ovat luonnollisesti kalliimpia. FinDocs käyttää *t2.micro*-tyyppistä instanssia, koska se on ainut vaihtoehto, joka on saatavilla ilmaiseksi. EC2 on siitä loistava palvelu, että asiakkaan tarpeiden ja käyttäjämäärien kasvaessa myös laskentatehoa ja liikennöintikaistaa on helppo nostaa.

EC2-instanssin luomisvaiheessa on mahdollista konfiguroida instanssia vielä tarkemmin esimerkiksi verkkojen, tallennustilan ja turvallisuuden osalta. FinDocsissa käytetään kuitenkin edellämainittujen vakioasetuksia, jotka ovat moneen tarkoitukseen riittävät.

Seuraavaksi klikataan *Review and Launch*-painiketta ja siirrytään tarkastelemaan instanssin yhteenvetoa. Kun yhteenvedon tiedot on tarkastettu, siirrytään avaintiedoston luomiseen SSH-yhteyttä varten *Launch*-painikkeella.



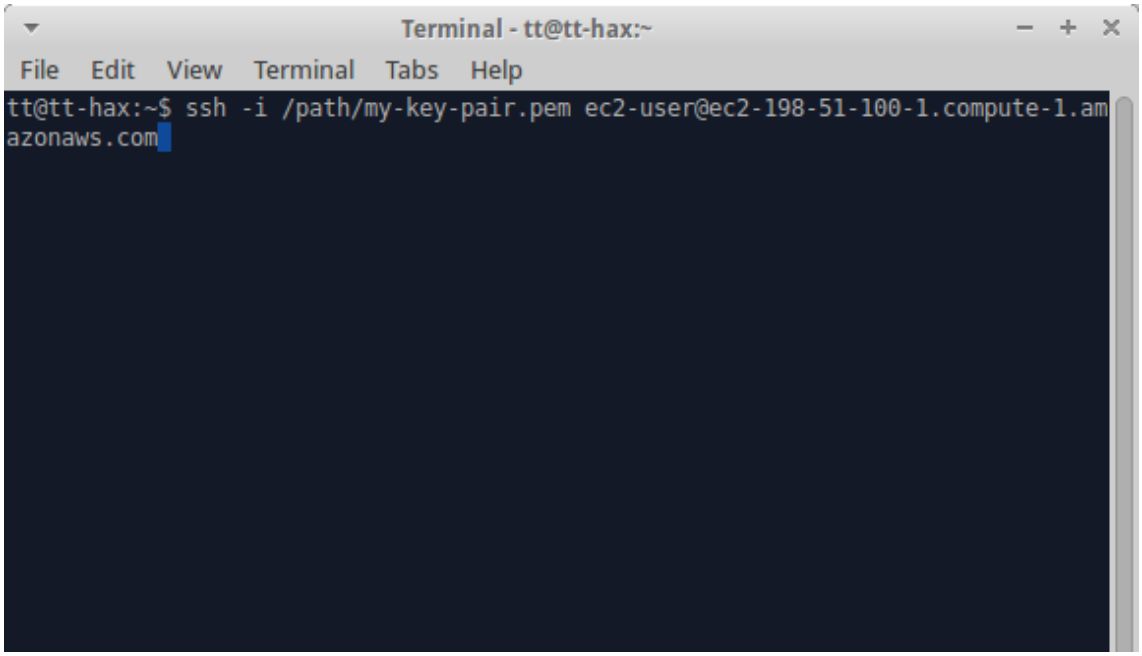
KUVA 4. SSH-yhteyteen tarvittavan PEM-avaintiedoston luonti

Avaintiedoston luomisen jälkeen instanssi käynnistetään. Virtuaalipalvelin on käyttövalmiina yleensä alle minuutissa. Palvelimen tietoihin ja asetuksiin pääsee edellämainitusta EC2-yleisnäkymästä.

4.1.2 SSH-yhteys virtuaalipalvelimeen

Virtuaalipalvelimen käynnistyksen jälkeen tarvitaan SSH-yhteys palvelimeen sen tehokkaan käsittelyn mahdollistamiseksi. SSH-yhteys luodaan palvelimeen aiemmin luodun PEM-avaimen avulla. Suurin osa Linux-käyttöjärjestelmistä sisältää jo valmiiksi

SSH-asiakasohjelman, joten yhteyden luominen on helppoa. Yhteyden luomiseksi tarvitaan vain PEM-avaintiedosto, käyttäjätunnus ja palvelimen osoite.



KUVA 5. Esimerkki komennosta, jolla muodostetaan SSH-yhteys palvelimeen

SSH-yhteys muodostetaan kuvan 5 mukaisella komentorivikomennolla. Komennon lisäparametri *-i* tarkoittaa identiteettitiedostoa, joka on polku AWS:stä ladattuun PEM-avaintiedostoon. Tiedostopolun jälkeen tulee käyttäjätunnus ja palvelimen julkinen DNS- tai IP-osoite @-merkillä erotettuna.

Tämän jälkeen SSH-asiakasohjelma kysyy, halutaanko palvelin lisätä tunnettujen isäntien joukkoon. Myöntävän vastauksen jälkeen päästään kirjautumaan palvelimelle sisään kaikilla käyttöoikeuksilla SSH-yhteyden välityksellä.

4.1.3 Nginx HTTP -palvelimen, PHP-ajoympäristön ja MySQL-tietokannan asennus ja konfigurointi

Nginx HTTP -palvelin valittiin FinDocs-palveluun sen helpon käyttöönoton ja konfiguroinnin vuoksi. PHP-ohjelmointikieli ja MySQL-tietokanta valittiin, koska palvelun kehittäjällä on niistä aiempaa kokemusta pilvipalveluiden kehittämisessä. Nginx, PHP ja MySQL löytyvät valmiiksi Ubuntu-käyttöjärjelmän virallisista

repositorioista ja ne voidaan asentaa Ubuntussa esiasennetulla APT-paketinhallintatyökalulla.

Pakettien asennus tapahtuu automaattisesti, lukuunottamatta MySQL:ää, joka pyytää käyttäjää nimeämään MySQL-palvelimen pääkäyttäjän ja antamaan käyttäjälle salasanan.

Asennuksen jälkeen Nginx-palvelin, MySQL-tietokanta ja PHP-ajoympäristö käynnistyvät automaattisesti. Mikäli asennus tapahtui virheettömästi, selaimella palvelimen osoitteeseen siirryttäessä käyttäjälle pitäisi näkyä Nginxin tervetuloa-sivu. Nginxin konfiguroiminen käyttämään PHP:tä ja osoittamaan FinDocs-palvelun kansioon vaatii vielä hieman konfigurointia.

Edellä mainittuihin toimenpiteisiin tarvittava konfiguraatiotiedosto löytyy Ubuntu-käyttöjärjestelmässä oletuksena tiedostosta */etc/nginx/sites-available/default*.

```
root /var/www/findocs-web/;
index index.html index.htm index.php;
```

KUVA 6. Nginx-konfiguraatiotiedoston kohta, jossa määritellään web-palvelimen tarjoama sisältö

```
# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
#
#location ~ \.php$ {
#       fastcgi_split_path_info ^(.+\.php)(/.+)$;
#       # NOTE: You should have "cgi.fix_pathinfo = 0;" in php.ini
#
#       # With php5-cgi alone:
#       #fastcgi_pass 127.0.0.1:9000;
#       # With php5-fpm:
#       fastcgi_pass unix:/var/run/php5-fpm.sock;
#       fastcgi_index index.php;
#       include fastcgi_params;
#}
```

KUVA 7. Nginx-konfiguraatiotiedoston kohta, jossa määritellään PHP-asetukset

Kuvien 6 ja 7 osoittamat kohdat ovat olennaisimpia Nginx-konfiguraatiossa palvelun pystyttämisen kannalta.

Kuvan 6 *root*-kohdassa määritellään kansio, johon kaikki palvelimelle vastaanotettavat HTTP-kutsut ohjataan. *Index*-rivi määrittelee tiedostovaihtoehdot, johon kutsut ohjataan. Tässä tapauksessa tarvitaan vain *index.php*, koska kaikki palvelun liikenne ohjataan kyseisen tiedoston kautta.

Kuvassa 7 näkyy konfiguraatiotiedoston kohta, jossa määritellään PHP:n sijainti ja asetukset. Rivit ovat merkitty kommenteiksi #-merkeillä. *Location*-rivi ja sen jälkeen aaltosulkeilla rajattujen rivien kommentointimerkit tulee poistaa. Mikäli PHP on asennettu oikein, PHP toimii suoraa kommentoinnin poistamisen jälkeen.

Konfigurointimuutosten jälkeen käynnistetään PHP ja Nginx uudelleen komennoilla *sudo service php5-fpm restart* ja *sudo service nginx restart*. Tämän jälkeen uudet konfiguroinnit ovat käytössä.

Nginxissä, PHP:ssä ja MySQL:ssä on helposta käyttöönotosta huolimatta lukemattomia mahdollisuuksia konfigurointiin. Tämän opinnäytetyön pääpaino on kuitenkin ohjelmistosuunnittelulla, joten vain vaaditut osat ympäristön pystyttämisestä käydään läpi.

4.2 Tietokantarakenne

FinDocs-palvelun ensimmäisen version, joka opinnäytetyössä esitellään, tietokanta pyritään pitämään mahdollisimman yksinkertaisena. Tietokantaa käytetään ensimmäisessä versiossa käyttäjä-, dokumentti- ja tunnistustietojen tallentamiseen. Näille kaikille luodaan tietokantaan omat taulut.

Käyttäjätauluun tallennetaan käyttäjän tiedot rekisteröitymisen yhteydessä. Kerättävän tiedon määrä pyritään pitämään matalana käyttöönottokynnyksen pienenä pitämiseksi. Tietokantaan tallennetaan käyttäjän id, nimi, sähköpostiosoite, käyttäjätunnus ja salattu salasana.

Dokumenttitauluun kerätään tiedot käyttäjän palveluun tallettamista dokumenteista. Dokumenttitauluun tallennetaan dokumentin id, käyttäjän id, tiedostonimi, talletusaika, luokitus ja kuvaus. Käyttäjän id on vierasavain (foreign key), joka viittaa käyttäjätaulun

id:hen. Tiedostonimen avulla dokumentti haetaan myöhemmin koodissa palvelimen kiintolevyltä.

Tunnistustauluun kerätään tunnistusavaimia, joita luodaan, kun käyttäjä kirjautuu palveluun uudella mobiililaitteella.

4.3 PHP-backend ohjelmointiratkaisut

FinDocs käyttää palvelinpäässä PHP:tä ja sen päällä CodeIgniter-sovelluskehystä. CodeIgniter yksinkertaistaa palvelinpään ohjelmointia merkittävästi. *Index.php*-tiedosto, johon Nginx-palvelin osoittaa, tulee CodeIgniterissä mukana ja CodeIgniter ohjaa HTTP-kutsut osoitteen perusteella oikealle käsittelijälle. FinDocs-palvelu ohjaa pääasiassa käsittelijöille tulevat HTTP-kutsut malleille, jotka käsittelevät tietokantaa.

```
function __construct() {
    parent::__construct();
    $this->load->database();
    $this->load->library('pbkdf2');
    $this->load->library('session');
    $this->load->helper('url');
}

function get_documents($user) {
    $result = $this->db
    ->from("documents")
    ->where("user_id", $user->id)
    ->order_by("uploaded desc")
    ->get()
    ->result();
```

KUVA 8. Koodiesimerkki tietokannan käsittelystä CodeIgniterillä

Tietokannan käsittely CodeIgniterillä on helppoa. CodeIgniter sisältää konfiguraatiotiedoston, johon laitetaan MySQL-palvelimen osoite, käyttäjätunnus, salasana ja tietokannan nimi. Tämän jälkeen tietokantaa voi käyttää malleissa lataamalla ensin tietokannanhallintamoduuli mallin konstruktorissa kuvan 8 mukaisesti. CodeIgniter sisältää myös tavan käsitellä tietokantaa ilman SQL-käskyjen kirjoittamista. Kuvassa 8 on esimerkki kuinka tietokannasta haetaan kaikki käyttäjän dokumentit ja järjestetään ne latausajan mukaisesti.

Palvelussa käytetään myös HTML-pohjaisia CodeIgniter-näkymiä. Näkymiä käytetään ensin sisäänkirjautumis- ja rekisteröitymissivujen näyttämiseen, ja sisäänkirjautumisen jälkeen näkymässä ajetaan Backbone-pohjainen selainpään yhden sivun sovellus.

4.3.1 Rekisteröityminen ja kirjautuminen

KUVA 9. Kuva FinDocsin kirjautumissivusta

Rekisteröityminen ja kirjautuminen tapahtuu FinDocsissa CodeIgniter-näkymän kautta. Näkymä sisältää HTML-lomakkeen (form), jossa kysytään käyttäjätunnus ja salasana. Näkymässä ladataan JavaScript-tiedosto, jossa asetetaan *Kirjaudu sisään* -painikkeelle tapahtumakuuntelija (event handler). Painiketta painettaessa lähetetään Ajax HTTP-kutsu, joka ohjataan sisäänkirjautumisfunktiolle CodeIgniter-käsittelijätiedostossa.

```
function login() {
    $username = $this->input->post("username");
    $password = $this->input->post("password");
    $success = $this->auth_model->login($username, $password);
    $this->output->set_content_type('application/json');
    if($success) {
        $this->output->set_output(json_encode(array("code" => 100, "msg" => "logged in")));
    }
    else {
        $this->output->set_output(json_encode(array("code" => 500, "msg" => "login failed")));
    }
}
```

KUVA 10. Sisäänkirjautumisfunktio

Kuvassa 10 näkyy FinDocsin käsittelijätason sisäänkirjautumisfunktio, joka on tyypillinen toteutus CodeIgniter-käsittelijän funktiosta. Selaimelta lomakedatana lähetetyt käyttäjätunnus ja salasana saadaan käyttöön käyttämällä CodeIgniterin sisäisiä funktioita. Tämän jälkeen tiedot toimitetaan eteenpäin mallille, joka suorittaa tietokanta- ja sessiokäsittelyn. CodeIgniter hoitaa käyttäjäsession käsittelyn erittäin pitkälle kehittäjän puolesta. Kehittäjän täytyy vain ladata mallin konstruktorissa sessio-moduuli kuvan 8 mukaisesti ja käyttää sessio-moduulin *set_userdata*-funktiota. Sessio ja evästeet luodaan automaattisesti. Mallin funktion palautusarvon perusteella palautetaan HTTP-kutsun tekijälle tieto sisäänkirjautumisen onnistumisesta tai epäonnistumisesta JSON-muodossa. Tähän käytetään CodeIgniterin sisäistä *set_output*-funktiota. Mikäli sisäänkirjautuminen onnistuu, käyttäjä ohjataan CodeIgniter-näkymään, joka käynnistää selainpään Backbone-sovelluksen.

Rekisteröitymisessä rakenne on samankaltainen kun sisäänkirjautumisessa. Erona on, että tietokantaan kirjoitetaan käyttäjän antamia tietoja niiden vertailun sijaan. Rekisteröitymisen onnistuessa sisäänkirjautuminen tapahtuu automaattisesti.

CodeIgniter suojaa automaattisesti tietokantakyselyt SQL-injektio hyökkäyksiltä poistamalla erikoismerkit kyselyiden käyttäjän syöttämistä kohdista. FinDocsin salasanan salaukseen käytetään RSA Laboratoriesin kehittämää PBKDF2-tekniikkaa.

4.3.2 Mobiililaitteen tunnistautuminen

Mobiililaitteen sisäänkirjautuminen vaatii hieman erilaista implementaatiota kuin selaimella kirjautuminen, koska evästeet eivät ole käytössä. FinDocsissa mobiililaitteiden tunnistaminen tapahtuu tietokantaan tallennettavien tunnistautumisavaimien avulla.

Kun käyttäjä kirjautuu sisään mobiilisovelluksen avulla, kutsu lähetetään eri käsittelijäfunktioon kuin selainkutsuissa. Kirjautumiseen käytetään samaa mallitason funktiota, mutta onnistuneen sisäänkirjautumisen jälkeen ohjelma luo uniikin SHA-1 -tunnistautumisavaimen, joka muodostetaan useasta eri osasta. Tunnistautumisavain palautetaan mobiililaitteelle. Kun myöhemmin mobiililaitteesta tehdään kutsuja palvelimelle, tätä avainta käytetään käyttäjän tunnistamiseen.

4.3.3 Dokumenttien käsittely

Dokumenttien käsittely palvelinpäässä jakautuu niiden hakemiseen verkkopalvelun käyttöön ja niiden lähettämiseen mobiilisovelluksesta.

Dokumenttien hakeminen palvelimelta tarkoittaa käytännössä verkkopalvelusta lähetettäviin Ajax-kutsuihin vastaamista. Kun verkkopalvelusta tehdään kutsu palvelimelle, ensimmäisenä tutkitaan, onko käyttäjä kirjautunut sisään. Sisäänkirjautumista tutkitaan yrittämällä hakea käyttäjän id CodeIgniterin session-moduulin *userdata*-funktiolla. Mikäli aktiivinen sessio on käynnissä, palauttaa funktio käyttäjän id:n, jonka avulla haetaan tietokannasta muut käyttäjätiedot objektiin. Näitä tietoja käytetään esimerkiksi tietokantahauissa oikeiden tietojen hakemiseksi.

Kun käyttäjä on tunnistettu, haetaan dokumentin metatiedot ensin tietokannasta käyttäjän id:n perusteella. Itse kuvatiedostot haetaan kiintolevyltä toisen käsittelijäfunktion avulla silloin kun niitä tarvitaan selainpäässä. Tietokannasta saatuihin dokumenttiobjekteihin lisätään kenttä, joka sisältää osoitteen, josta itse kuvatiedostot ladataan.

Kuvatiedostojen haku palvelimelta tapahtuu siis PHP-koodin kautta, koska kuvatiedostoja ei säilytetä Nginx-palvelimen tarjoamassa ohjelmiston kansiossa. Näin vähennetään riskiä, että kuvatiedostot paljastuvat ulkopuolisille, koska kuva haetaan palvelimelta PHP:n avulla sijainnista, johon internetin kautta ei ole suoraa pääsyä. Tiedostot haetaan käyttäjän id:n ja tiedostonimen perusteella PHP:n *file_get_contents*-funktion avulla, joka hakee levyllä olevan tiedoston sisällön muuttujaan. Tämä tiedosto palautetaan selaimelle oikeassa muodossaan käyttäen CodeIgniterin output-moduulin sisäisiä funktioita.

Dokumenttien lähettäminen mobiililaitteesta palveluun tapahtuu JSON-muodossa. Käyttäjän tunnistamiseen käytetään HTTP-kutsun otsikkotiedoissa lähetettävää tunnistusavainta, joka on aiemmin luotu mobiililaitteelle sisäänkirjautumisen yhteydessä. JSON muutetaan PHP:n oliomuotoon kielen sisäisellä *json_decode*-funktiolla ja siirretään tunnistusavaimen avulla haettujen käyttäjätietojen kanssa CodeIgniter-mallitiedostolle tietokantaan ja levylle tallennusta varten.

Palvelimelle saapuvat dokumentit ovat JSON-objekteja, jotka sisältävät dokumentin luokituksen, kuvauksen ja itse dokumenttitiedoston Base64-muodossa. Base64 on datan koodaustapa, jolla binääridata voidaan muuttaa ASCII-merkkijonoksi. PHP:ssä on sisäänrakennetut funktiot, joilla Base64-koodaus voidaan toteuttaa tai purkaa.

Mallitasolla dokumenttitiedoston Base64-koodaus puretaan ja tiedosto tallennetaan levylle PHP:n *file_put_contents*-funktiolla. Dokumentin metatiedot ja tiedostonimi tallennetaan tietokantaan. Mobiililaitteelle palautetaan ilmoitus lähettämisen onnistumisesta tai epäonnistumisesta JSON-muodossa.

4.4 Backbone, Marionette ja muut frontend ohjelmointiratkaisut

Backbone ja Marionette JavaScript-sovelluskehykset mahdollistavat tehokkaasti verkkopalvelun ulkoasun ja datan erottamisen toisistaan kuitenkin mahdollistaen niiden pitämisen ajan tasalla keskenään.

4.4.1 Selainpään rakenne

FinDocs-selainpään sovellus käynnistyy PHP:n CodeIgniter-näkymästä, johon käyttäjä ohjataan sisäänkirjautumisen jälkeen. Näkymässä ladataan JavaScript-laajennus RequireJS, jota käytetään JavaScript-tiedostojen lataamiseen moduulimaisesti. RequireJS helpottaa useista eri tiedostoista ja kirjastoista koostuvien JavaScript-ohjelmistojen hallintaa.

RequireJS lataa tarvittavat moduulit käytettäväksi sovelluksessa ja käynnistää lopulta JavaScript-tiedoston, jossa selaimen ikkunaobjektiin kiinnitetään Marionette-sovellus ja ajetaan se.

Selaimen ikkunaan kiinnitetty Marionette-sovellus toimii FinDocsin selainpään pohjana. Tässä tiedostossa määritellään ulkoasun pohjanäkymä, johon kaikki alemman tason näkymät kiinnitetään. Tiedostossa määritellään myös router-objekti, joka ohjaa selaimen oikeaan funktioon riippuen selaimen osoitteesta.

```
define([
    'marionette'
], function(Marionette) {
    var Router = Marionette.AppRouter.extend({
        appRoutes: {
            "": "showDocumentsLayout",
            "info": "showInfo"
        }
    });
    return Router;
});
```

KUVA 11. FinDocsin käyttämä Marionette Router -luokan toteutus

```
        showDocumentsLayout: function() {
            this.findocsLayoutView.contentRegion.show(new DocumentsView({}));
        },

        showInfo: function() {
            this.findocsLayoutView.contentRegion.show(new InfoView({}));
        }
```

KUVA 12. FinDocsin käyttämän Marionette Application -toteutuksen osa

Kuvassa 11 on FinDocsin ensimmäisen version router-implementaatio. Router-luokkaan määritellään *appRoutes*-objekti, jossa attribuuttien nimet vastaavat selaimen osoitteen päätteitä ja niiden arvot kertovat funktion nimen, joka ajetaan applikaatiotiedostossa, kun osoitteeseen saavutaan.

Esimerkiksi mikäli sivuston osoite olisi *www.findocs.fi*, siirryttäessä osoitteeseen *www.findocs.fi/fd#info* ajettaisiin applikaatiotiedoston *showInfo*-funktio. Esimerkin fd-osa on CodeIgniter-käsittelijän nimi, jonka kautta applikaatioon saavutaan. Kuvan 12 *showInfo*-funktiossa näytetään applikaatiotiedostossa luodussa pohjanäkymässä määritellyssä sisältöalueessa uusi näkymä, joka on tässä tapauksessa nimeltään *InfoView*.

4.4.2 Dokumenttikokoelma

Backbonessa ja Marionettessa sovelluksessa käytettävä data säilytetään malli-tyyppisissä tietorakenteissa. Nämä mallit sisältävät palvelimelta haettuja tietoja. Esimerkiksi yksi dokumenttimalli sisältää yhden palveluun lähetetyn dokumentin tiedot.

Tietojen säilyttämisen lisäksi malleihin on mahdollista ohjelmoida ylimääräistä logiikkaa.

Malleja voi esiintyä ohjelmassa erikseenkin, mutta tyypillisesti niitä säilytetään kokoelmissa, jotka sisältävät useita samantyyppisiä malleja. FinDocsissa esimerkiksi on dokumenttikokoelma, joka sisältää kaikki käyttäjän dokumentit malleina, jotka on haettu palvelimen kautta tietokannasta.

```
1  define([
2      'jquery',
3      'backbone'
4      ],
5  function($, Backbone) {
6      var Collection = Backbone.Collection.extend({
7          url: "/fd/documents",
8
9          fetchDocuments: function() {
10             var me = this;
11             return this.fetch()
12             .success(function(data) {
13                 me.trigger("documents:fetched");
14             });
15         }
16     });
17     return Collection;
18 });
19
```

KUVA 13. Dokumenttikokoelman toteutus

Kuvassa 13 on FinDocsin dokumenttikokoelman toteutus. Kokoelmalle määritellään *url*-attribuutti, jonka arvoksi tulee osoite, josta kokoelma hakee HTTP-kutsulla sisältönsä. Tässä tapauksessa se osoittaa FinDocsin palvelinpään FD-nimisen CodeIgniter-käsittelijän *documents*-funktioon. Palvelinpää palauttaa dokumenttien tiedot JSON-muodossa, ja kokoelma luo niistä palvelimelta saatuja tietoja vastaavat mallit kokoelmaan. Dokumenttikokoelmaan voi myös ohjelmoida ylimääräistä logiikkaa, kuten FinDocsissa on *fetchDocuments*-funktion osalta tehty. Funktiossa käytetään kokoelman sisäistä *fetch*-funktiota dokumenttien hakemiseen palvelimelta, ja kutsun palatessa onnistuneena laukaistaan rivillä 13 tapahtuma, jota dokumentteja

listaava näkymä kuuntelee. Tämä mahdollistaa näkymän piirtämisen vasta kun dokumenttimallit on ladattu.

4.4.3 Näkymät

Käyttäjälle näkyvä osuus verkkopalvelun selainpäässä tapahtuu Marionetten näkymissä. Marionettessa on useita erityyppisiä näkymiä. Yhteistä niille on, että näkymä koostuu JavaScript-tiedostosta, joka sisältää näkymään liittyvän toiminnallisuuden ja template-tiedostosta, joka on HTML:ää. Template-tiedostoihin voi kuitenkin upottaa myös JavaScriptiä. Tämä tapahtuu Backbonen käyttämällä UnderscoreJS-lisäosalla.

FinDocsissa käytetään kahdentyyppisiä näkymiä. Näkymien tyypit ovat layout-näkymä (layout view) ja item-näkymä (item view). Layout-näkymä on tarkoitettu pohjaksi alinäkymille ja siihen voi eritellä alueita (region), joissa alinäkymiä näytetään. FinDocsissa sitä käytetään pohjanäkymässä, joka luodaan applikaatiotiedostossa. Muut näkymät ovat item-tyyppisiä ja niitä käytetään pohjanäkymän alueissa.

Tärkein näkymä palvelussa on dokumenttinäkymä, jossa käyttäjän palveluun lähettämät dokumentit ja niiden tiedot näytetään. Dokumenttinäkymässä käytetään HTML-taulukkoelementtiä, joka täytetään aiemmin esitellyn dokumenttikokoelman sisällön perusteella.

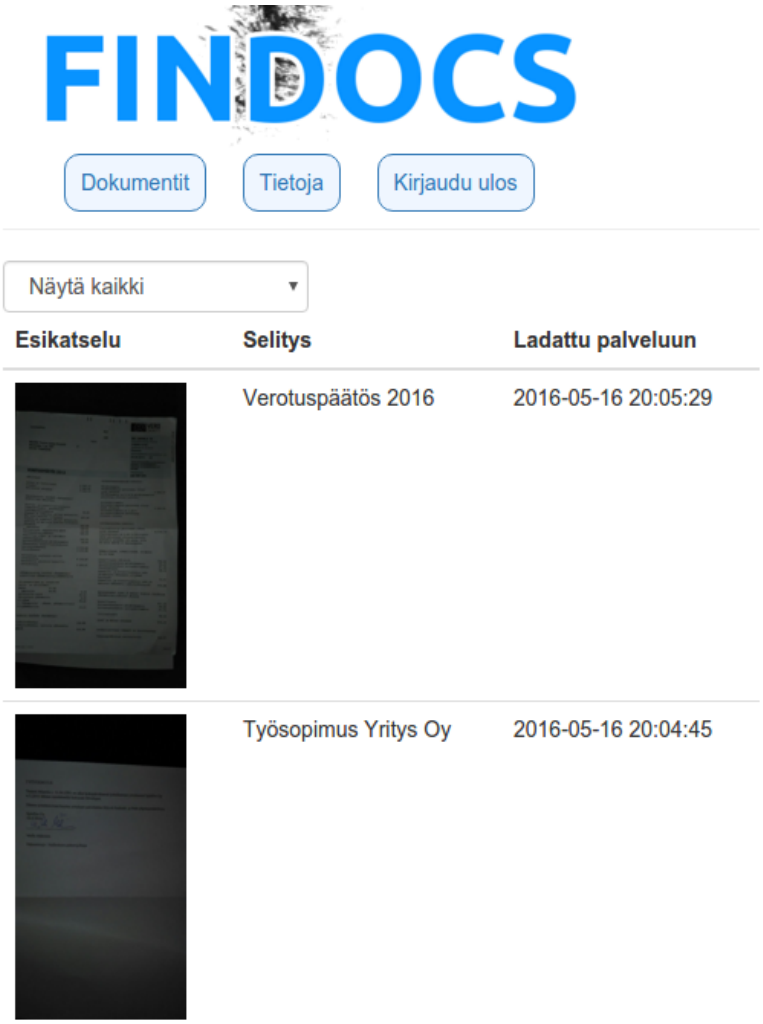
```
_render: function() {

    var variables = {
        documents: FD.documentsCollection.toJSON(),
        filter: this.documentsFilter
    }
    var template = _.template(tmpl, variables);
    this.$el.html(template);

    this.$("#document-filter").val(this.documentsFilter);
}
```

KUVA 14. Dokumenttinäkymän piirtofunktio

```
<tbody>
<%documents.forEach(function(document) { %>
    <%if(filter == "all" || filter.toLowerCase() == document.classification.toLowerCase()) {%>
    <tr>
        <td>
            <a href="{{document.img}}">
                <img class="thumbnail-document" src="{{document.img}}" alt="document">
            </a>
        </td>
        <td>{{document.description}}</td>
        <td>{{document.uploaded}}</td>
    </tr>
<%}})%>
</tbody>
```

KUVA 15. Dokumenttinäkymän template-tiedoston osa

Kuvassa 14 näkyy kuinka template-tiedostoon viedään dataa *variables*-objektissa. *Variables*-objekti ja kuvassa 15 osittain näkyvä HTML-muotoinen template-tiedosto yhdistetään UnderscoreJS:n *template*-funktiolla. Näin JavaScript-puolella sijaitseva dokumenttikokoelma saadaan iteroitua taulukon riveiksi HTML:ään.



KUVA 16. FinDocsin dokumenttinäkymä selaimessa

4.5 Android-mobiilisovellus

FinDocs-palveluun kuuluu olennaisena osana Android-mobiilisovellus, jolla käyttäjän on tarkoitus kuvata ja lähettää dokumentteja palveluun. Sovelluksen logiikka on ohjelmoitu Java-kielellä ja ulkoasu rakennettu XML-merkkauksella, eli kyseessä on natiivisovellus.

4.5.1 Mobiilisovelluksen rakenne

FinDocs-mobiilisovellus on rakennettu Android-aktiviteeteista, yhdestä palvelusta ja muutamasta apuluokasta.

Android-aktiviteetti on sovelluskomponentti, jonka tarkoitus on luoda alusta käyttäjän toiminnoille Android-sovelluksessa. Aktiviteetille luodaan aina ikkuna, johon rakennetaan käyttöliittymä. Ikkuna vie tyypillisesti koko ruudun, mutta myös muita mahdollisuuksia on. (Developers. Activities.)

Aktiviteetteja käytetään FinDocs-mobiilisovelluksessa päävalikkoon, sisäänkirjautumisruutuun, dokumenttien luokittelunäkymään ja käyttäjätietojen esittämisnäkymään. Aktiviteetista siirrytään toiseen käyttäjän toimien perusteella.

Apuluokat on itse luotuja perinteisiä Java-luokkia, joita käytetään apuna erinäisissä toimenpiteissä sovelluksessa. FinDocsissa käytetään apuluokkia esimerkiksi staattisten vakioiden säilyttämiseen, käyttäjän asetusten säilyttämiseen ja palvelimelle lähetettävien dokumenttien lähetystiedostojen käsittelemiseen.

Androidin palvelukomponenttia käytetään dokumenttien lähettämiseen palvelimelle ja levylle kirjoitettavien JSON-tiedostojen luomiseen. Palvelut suoritetaan omassa säikeessään ja näin vältytään käyttöliittymän pysäyttämiseltä dokumenttien lähetyksen ajaksi. Tämä mahdollistaa usean dokumentin lähettämisen palveluun peräkkäin ilman ylimääräistä odottamista.

4.5.2 Sisäänkirjautuminen ja tunnistautuminen

Sisäänkirjautuminen on ensimmäinen toimenpide FinDocs-mobiilisovellusta käynnistettäessä. Sisäänkirjautuminen tapahtuu sovelluksessa vastaavan aktiviteetin kautta.

KUVA 17. Sisäänkirjautumisaktiviteetti ja sen käyttöliittymä

Aktiviteetissa luodaan kuvan 17 mukainen käyttöliittymä, johon käyttäjä kirjoittaa palveluun rekisteröitymisen yhteydessä valitsemansa sähköpostiosoitteen ja salasanan. Aktiviteetin käyttöliittymä luodaan erillisessä XML-tiedostossa, johon määritellään käyttöliittymään haluttavat komponentit, niiden sijoitus ja tyylittely.

```
Button loginButton = (Button) findViewById(R.id.login_button);
mEmailField = (EditText) findViewById(R.id.email);
mPasswordField = (EditText) findViewById(R.id.password);

loginButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        login();
    }
});
```

KUVA 18. Käyttöliittymäkomponenttien käsittelyä Java-koodissa toiminnallisuuden luomisen ja tietojen hakemisen mahdollistamiseksi

Aktiviteetin käyttöliittymäkomponentit saadaan ohjelmakoodin käyttöön kuvan 18 mukaisesti. Kuvassa kirjautumispainike sekä sähköposti ja salasanakenttä asetetaan muuttujiin. Tämän jälkeen kirjautumispainikkeeseen kiinnitetään tapahtumakuuntelija, joka käynnistää *login*-sisäänkirjautumismetodin painiketta painettaessa.

Itse sisäänkirjautuminen tapahtuu *login*-metodissa. Metodissa käyttäjän syöttämät tiedot haetaan käyttöliittymäkomponenteilta ja muutetaan JSON-muotoon. Tämän jälkeen JSON-lähetetään eri säikeessä palvelimelle HTTP-protokollaa käyttäen POST-muodossa OkHttp-kirjaston avulla. Mikäli sisäänkirjautuminen onnistuu, palvelin palauttaa sovellukselle tunnistautumisavaimen, jota käytetään myöhemmin dokumenttien lähettämisessä palvelimelle.

4.5.3 Dokumenttien kuvaaminen ja luokittelu

FinDocs-mobiilisovelluksessa dokumenttien kuvaamiseen käytetään Android-laitteen mukana tullutta kamerasovellusta. Kamerasovelluksen kuvaustoiminto käynnistetään pääpiirteittäin kuten mikä tahansa muukin aktiviteetti.

KUVA 19. FinDocs-mobiilisovelluksen päävalikko

Dokumenttien kuvaukseen pääsee kuvassa 19 näkyvän sovelluksen päävalikon kautta. *Kuvaa dokumentti* -painiketta painettaessa käynnistetään Android-laitteen kamerasovellus. Kamera-aktiviteetille annetaan lisäparametreina tieto, että kuva halutaan tallentaa levylle, sekä haluttu tallennuspolku. Kamerasovellus tallentaa kuvan haluttuun polkuun automaattisesti.

Kamera-aktiviteetti palaa tämän jälkeen takaisin pääaktiviteettiin, koska se käynnistettiin aiemmin *startActivityForResult*-metodilla. Kamera-aktiviteetti palauttaa pääaktiviteettiin koodin, joka kertoo onnistuiko aktiviteetti tehtävässään. Mikäli dokumentin kuvaaminen ja tallennus onnistui, siirrytään luokitteluaktiviteettiin. Jos kuvauksessa tapahtui virhe, näytetään ruudulla virheilmoitus.

KUVA 20. Luokittelunäkymä

Kuten kuvassa 20 nähdään, luokitteluaktiviteetissa dokumentti luokitellaan johonkin valmiiksi määritellyistä dokumenttikategorioista ja sille voi antaa vapaamuotoisia lisätietoja. Luokitusvalinta on spinner-typpinen Android-käyttöliittymäelementti, joka on tässä tapauksessa pudotusvalikko.

Tietojen antamisen jälkeen *Lähetä*-painike aloittaa dokumenttien lähetyspalvelun ja lähettää sille dokumentin sijainnin levyllä sekä luokittelunäkymässä annetut metatiedot.

4.5.4 Dokumenttien lähetys palveluun

Lähetyspalvelu käynnistetään luokittelunäkymän jälkeen ja se ajetaan omassa säikeessään. Palvelu ottaa luokittelunäkymältä vastaan tarvittavat tiedot dokumentista.

Ensimmäinen vaihe lähetyspalvelun toiminnassa on JSON-muotoisen dokumenttitiedoston luominen vastaanotettujen tietojen perusteella. Prosessiin sisältyy dokumenttikuvan pienennys, sen muuntaminen Base64-muotoon ja JSON-muotoisen tekstitiedoston luominen sekä Base64-koodatusta kuvasta että dokumentin metatiedoista. Tiedostopohjainen lähestymistapa valittiin Androidin SQLite-toteutuksen monimutkaisuuden ja taipumattomuuden vuoksi. Lähetystiedostoa ei ikinä tarvitse muuttaa. Se kirjoitetaan levylle luomisvaiheessa ja luetaan lähetysvaiheessa. Lisäksi tiedoston rakennetta on helppo muuttaa päivityksien yhteydessä.

Aiemmin kuvattu dokumentti saattaa olla useiden megatavujen kokoinen kuvatiedosto. Tämän vuoksi ensimmäinen vaihe dokumentin ja sen tietojen käsittelyssä lähetystä varten on kuvan koon pienennys. JPEG-kuvan pienentämiseksi se täytyy ensin muuttaa bittikartta-muotoon. Tämä tapahtuu Javan Bitmap-luokkaa käyttäen. Tämän jälkeen tarkastellaan kuvan kokoa. Kuvan koko puolitetaan niin kauan kuin joko korkeus tai leveys on alle 2000 pikseliä. Väärinkäytöksien estämiseksi palvelin estää liian suurien dokumenttien lähettämisen. Kun kuvan koko on muutettu, tallennetaan se takaisin JPEG-muotoon alkuperäisen kuvan päälle.

```
String encodedImage = null;
try {
    encodedImage = de.greenrobot.common.Base64.encodeFromFile(filename);
} catch (IOException e) {
    e.printStackTrace();
    Log.e(TAG, "Encoding image failed");
    return false;
}

Long tsLong = System.currentTimeMillis()/1000;
String ts = tsLong.toString();

JSONObject uploadJson = new JSONObject();
try {
    uploadJson.put("document", encodedImage);
    uploadJson.put("classification", classification);
    uploadJson.put("description", description);
} catch(JSONException e) {
    e.printStackTrace();
    Log.e(TAG, "Creating uploadJson object failed!");
    return false;
}
```

KUVA 21. Koodiesimerkki dokumenttikuvan Base64-koodaamisesta ja kuvan sekä metatietojen JSON-muotoon saattamisesta

Kun kuva on pienennetty, seuraava vaihe on koodata kuva Base64-muotoiseksi merkkijonoksi. Javasta löytyy vakiona Base64-kirjasto, mutta FinDocsissa käytetään Greenrobotin Java Common -kirjaston Base64-työkaluja. Tämä vähentää koodin määrää, koska kirjastosta löytyy metodi, joka ottaa parametrina tiedoston polun ja palauttaa Base64-merkkijonon kuvan 21 mukaisesti.

Seuraavaksi Base64-muotoisesta dokumentista ja sen metatiedoista luodaan JSON-objekti Javan JSON-luokkia käyttäen. JSON-objekti muutetaan tämän jälkeen merkkijonoksi ja se tallennetaan sovelluksen kansioon tekstitiedostona odottamaan lähettämistä palveluun. Tiedostot poistetaan vasta onnistuneen lähetyksen jälkeen.

Itse dokumenttien lähetys tapahtuu siten, että kaikki sovelluksen kansiosta löytyvät lähetystekstitiedostot käydään läpi ja yritetään lähettää palveluun. Normaalissa tilanteessa näitä pitäisi olla vain yksi, mutta jos esimerkiksi internet-yhteys katkeaa kesken lähetysprosessin, tiedosto jää levylle ja se lähetetään seuraavalla kerralla kun lähetyspalvelu käynnistetään.

Dokumentin lähetys palveluun tapahtuu HTTP POST -kutsulla JSON-muodossa. HTTP -kutsuun lisätään myös käyttäjän tunnistautumisavain otsikkotietona. Mikäli lähetys onnistuu, lähetystiedosto poistetaan levyltä ja palvelin hoitaa loput.

4.6 Testaus

Opinnäytetyössä luotava ensimmäinen FinDocs-palvelun versio on ominaisuuksiltaan hyvin niukka. Tämän vuoksi palvelun testaaminen onnistuu helposti kokeilemalla kaikkia ominaisuuksia.

Verkkopalvelun toimivuus, kattaen kaikki käyttötapaukset, testattiin Opera-, Chrome- ja Firefox-selaimilla. Mobiilisovellusta testattiin yhdellä Huawein- ja yhdellä Samsungin Android-laitteella.

Testausta vaativia käyttötapauksia on hyvin vähän. Palvelussa on vain yhdentyyppisiä käyttäjiä, joiden ainoat toimenpiteet ovat rekisteröinti, sisään- ja uloskirjautuminen, dokumenttien lähettäminen palveluun ja dokumenttien tarkastelu palvelussa. Lähes kaikki bugit pystyttiin korjaamaan jo kehitysvaiheessa.

Mahdollisesti tulevaisuudessa palvelun ominaisuuksien lisääntyessä on aiheellista käyttää kehittyneempiä testausmetodeja.

5 POHDINTA

FinDocs-palvelun luominen auttoi ymmärtämään web-palvelun ja siihen liittyvän mobiilisovelluksen kehittämisen eri vaiheet ja niiden haasteet. FinDocs-palvelussa käytetyt teknologiat ja työkalut olivat ennestään työelämästä tuttuja. Tämä helpotti palvelun kehittämistä huomattavasti ja ensimmäinen toimiva versio palvelusta saatiin valmiiksi noin viikossa aloittamisen jälkeen.

Projekti onnistui halutussa aikataulussa ja kaikki perustoiminnallisuus onnistuttiin implementoimaan palveluun. Ohjelmointityö käytetyillä teknologioilla oli tuttua jo ennestään, joten eniten ajatustyötä vaati palvelinympäristön pystyttäminen ja siihen liittyvät toimenpiteet.

Projektin lopputuloksena valmistui toimiva dokumenttien digitointiin ja talletukseen tarkoitettu pilvipalvelu ja siihen liittyvä mobiilisovellus. Palvelun käyttöoikeuksia jaetaan aluksi muutamille henkilöille. Palvelun kehityttyä palautteen perusteella, se on tarkoitus julkaista vapaaseen käyttöön sekä verkkopalveluna julkisen domainin päässä että mobiilisovelluksena Google Play -sovelluskaupassa.

Palvelun jatkokehitysmahdollisuuksien rajana on vain mielikuvitus. Käytettävät teknologiat mahdollistavat jatkokehityksen erittäin laajalta skaalalta. Mahdollisia seuraavia ominaisuuksia voisi olla dokumenttien talletus verkkopalvelussa, dokumenttien yhdistäminen ja lataaminen PDF-muodossa sekä jaetut tilit.

LÄHTEET

Hassan, Qusay (2011). Demystifying Cloud Computing. Luettu 11.5.2016
http://www.crosstalkonline.org/storage/issue-archives/2011/201101/201101-Hassan.pdf

Amazon Web Services. Amazon EC2 Product Details. Luettu 11.5.2016
https://aws.amazon.com/ec2/details/

Nginx. NGINX Wiki's documentation. Luettu 11.5.2016
https://www.nginx.com/resources/wiki/

PHP. What is PHP? Luettu 11.5.2016
http://php.net/manual/en/intro-whatis.php

CodeIgniter. CodeIgniter at a Glance. Luettu 11.5.2016
http://www.codeigniter.com/user_guide/overview/at_a_glance.html

AboutTech. What Is JavaScript?. Luettu 11.5.2016
http://javascript.about.com/od/reference/p/javascript.htm

jQuery. What is jQuery. Luettu 11.5.2016
https://jquery.com/

Backbone.js. Getting started. Luettu 11.5.2016
http://backbonejs.org/

Java. What is Java technology and why do I need it? Luettu 12.5.2016
https://java.com/en/download/faq/whatis_java.xml

Java. Learn about Java technology. Luettu 12.5.2016
https://java.com/en/about/

Git. Alkusanat versionhallinnasta. Luettu 12.5.16
https://git-scm.com/book/fi/v1/Alkusanat-Versionhallinnasta

MySQL 5.7 Reference Manual. What is MySQL? Luettu 18.5.2016
https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

Marionette. The Backbone framework. Luettu 18.5.2016
http://marionettejs.com/

Developers. Activities. Luettu 19.5.2016
https://developer.android.com/guide/components/activities.html