jamk.fi

Opinnäytetöiden seurantajärjestelmä

Opinnäytetöiden seurannan kehittäminen taulukoista palveluksi

Jani Kiikka

Opinnäytetyö
Lokakuu 2017
Tekniikan ja liikenteen ala
Insinööri (AMK), Tietotekniikan koulutusohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Kiikka, Jani	Opinnäytetyö, AMK	Lokakuu 2017
	Sivumäärä 49	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Opinnäytetöiden seurantajärjestelmä

Tutkinto-ohjelma
Insinööri (AMK), Tietotekniikan koulutusohjelma

Työn ohjaaja(t)
Rantonen Mika, Häkkinen Antti

Toimeksiantaja(t)
Jyväskylän Ammattikorkeakoulu, IT-instituutti

Tiivistelmä

Jyväskylän Ammattikorkeakoulun IT-instituutin käytössä on ollut Excel-tiedosto opinnätetöiden seurantaan, jonka ylläpitäminen on ollut vaikeaa. Tiedoston käyttämisen kanssa on havauttu ongelmia erityisesti silloin, jos useampi käyttäjä on yrittänyt avata kyseistä tiedostoa yhtäaikaisesti.

Tarkoituksena on rakentaa opinnäytetöiden seurantaan palvelu, joka auttaisi Jyväskylän Ammattikorkeakoulun IT-instituutiota paremmin seuraamaan opiskelijoiden opinnäytetöitä. Palvelun tarkoituksena on korvata Excel-tiedosto, joka sijaitsee jaetulla verkkolevyllä.

Tämän Excel-tiedoston korvaamiseksi on toteutettu tietokanta, jonka rakenteen tulisi kattaa kaikki tarvittavat tietueet joita kyseiseen Excel-tiedostoon on tallennettu. Palvelulle on myös ohjelmoitu backend-, sekä frontend-sovellukset, jotka antavat käyttäjille mahdollisuuden tallentaa tietoja tietokantaan. Lisäksi palvelimen yhteydet on salattu.

Opinnäytetyön lopputuloksena on palvelu, joka täyttää kaikki vaaditut ominaisuudet, kuten esimerkiksi tietojen muokkaamisen ja tallentamisen koskien opiskelijoiden opinnäytetöitä. Käyttötavaltaan palvelu on erilainen verrattaessa Excelin ruudukkonäkymään. Jokainen opinnäytetyö on eriytetty omaan kokonaisuuteensa, ja näiden töiden hallinnoimiseen tarjotaan sitä tarvitseville käyttäjille näkymä, josta kaikki seurattavat opinnäytetyöt löytyvät. Lopputuloksena syntynyt palvelu toimii ja täyttää tämän vaaditun spesifikaation.

Avainsanat (asiasanat)
Opinnäytetöiden seurantajärjestelmä

Muut tiedot
Muutokset ja korjaukset projektiin löytyvät osoitteesta https://github.com/janikiikka

jamk.fi

Description

Author(s)	Type of publication	Date
Kiikka, Jani	Bachelor's thesis	October 2017
		Language of publication: Finnish
	Number of pages 49	Permission for web publication: x

Title of publication
Thesis Management System

Degree programme
Insinööri (AMK), Tietotekniikan koulutusohjelma

Supervisor(s)
Rantonen Mika, Häkkinen Antti

Assigned by
Jyväskylä University of Applied Scie nces, IT-institute

Abstract

Jyväskylä University of Applied Sciences, IT-institute has been using an Excel spreadsheet to store and track information about students' theses, which has been less than stellar for varying reasons. For example, the current solution has no way of supporting multiple users trying to edit the file at the same time.

The objective is to replace this Excel spreadsheet with a service that allows concurrent users to access and modify said data. It should also come with useability perks that the existing spreadsheet does not have.

To replace this spreadsheet for good, this new implementation should use a dedicated database in addition to a separate frontend and backend programs. The service should also allow for concurrent users by design, unlike the current spreadsheet file on a shared network disk. In addition, the service should be expandable with new features.

The product created in this thesis is a service, that fulfils all specified features such as tracking and modification of the theses data. Usage of the service is quite different from that of Excel's grid system, as it offers each thesis as a separate view instead of a spreadsheet with a list of theses. It does, however, contain a list view for those users that require capability to track multiple theses at the same time. The end product of this thesis should be a functioning service that conforms to the given specifications.

Keywords/tags (subjects)
Thesis Management System

Miscellaneous
Changes and additions to the project can be found at https://github.com/janikiikka

Sisältö

Lyhenteet **5**

1 Johdanto **6**

1.1 Alkusanat 6

1.2 Työn tilaaja 7

1.3 Työn lähtökohdat 7

1.4 Tavoitteet opinnäytetöiden seurantapalvelulle 8

2 Toteutukseen valitut teknologiat **8**

2.1 Linux (Ubuntu 16.04 Xenial Xerus, LTS) 9

2.2 Angular 2 ja TypeScript 10

2.3 PostgreSQL 11

2.4 Microsoft .NET Core 11

3 Palvelun käyttäminen **12**

3.1 Käyttäjän tunnistaminen 12

3.2 Yleisnäkymä 13

3.3 Opinnäytetyönäkymä 13

3.4 Tiedostonäkymä 13

3.5 Kalenterinäkymä 14

4 Nginx ja tietoliikenne **14**

4.1 Palomuurin asetukset 14

4.2 Nginx-palvelin 15

5 PostgreSQL ja tietokannan toteutus **21**

5.1 Tietokannan rakenne 22

5.2 Tietokannan relaatioiden kuvaukset 24

5.2.1 Credentials-relaatio 24

5.2.2 Users-relaatio 24

5.2.3 Documents-relaatio 25

5.2.4 Downloads-relaatio 25

5.2.5 Events-relaatio 25

5.3 Theses-relaatio 25

5.4 Käyttäjän tiedot 26

5.5 Opinnäytetöiden tiedot 27

5.6 Riippumattomat relaatiot 28

5.7 PostgreSQL:n käyttäjätunnukset 29

5.8 PostgreSQL:n asentaminen 30

6 Front-end, käyttöliittymä 32

6.1 Angular CLI:n asentaminen ja käyttö 33

6.2 TypeScriptin asentaminen ja käyttö 34

6.3 Angular 2:lla kehittäminen 36

6.4 Angular 2 sovelluksen siirtäminen tuotantoon 38

7 Backend 39

7.1.1 Käyttäjän todentaminen 39

7.1.2 Käyttäjän tunnistaminen 41

7.1.3 Ohjaimien toiminta yleisesti 42

7.1.4 POST metodien käyttäminen 43

7.1.5 PATCH metodien käyttäminen 45

8 Kehitysehdotukset 46

9 Pohdinta 47

Lähteet 49

Liitteet 50

Kuviot

KUVIO 1 Azure ja terminaali näkymä virtuaalikoneella 10
KUVIO 2 Nginx:n asetukset 16
KUVIO 3 Salaamaton verkkoliikenne 17
KUVIO 4 TLS-salattu verkkoliikenne 18
KUVIO 5 Internetin ja palveluiden välinen liikenne 20
KUVIO 6 Web-sivujen tarjoaminen 21
KUVIO 7 Tietokannan malli 23
KUVIO 8 Esimerkki opinnäytetyön perustiedoista 27
KUVIO 9 Salasanan asettaminen käyttäjälle 29
KUVIO 10 PostgreSQL:n asentaminen 30
KUVIO 11 PostgreSQL virhe, "Roolia ei ole olemassa" 30
KUVIO 12 Uuden käyttäjäroolin lisääminen PostgreSQL:ssa 30
KUVIO 13 Lisätyllä käyttäjällä/roolilla kirjautuminen tietokantapalvelimelle 31
KUVIO 14 Relaatiot tietokannassa 31
KUVIO 15 Angular CLI:n asentaminen NPM:lla 33
KUVIO 16 TypeScript-koodin esittelyä 34
KUVIO 17 Käännetyn JavaScript koodin esittely 34
KUVIO 18 TypeScript tyyppi tarkistusten esittely koodina 35
KUVIO 19 TypeScript-kääntäjän virheilmoitus 35
KUVIO 20 Angular 2:n komponentti luokan esimerkki 36
KUVIO 21 Ominaisuusarvojen kiinnittäminen HTML-malliin. 37
KUVIO 22 Seurantajärjestelmän runkosivu (index.html) 38
KUVIO 23 Tuotantoon vietävä Angular 2-paketti 38
KUVIO 24 Malli kirjautumisen POST-viestistä 39
KUVIO 25 Onnistuneen kirjautumisen paluusanoma 40
KUVIO 26 Tokenit ja niiden aikaleimat tietokannassa 40
KUVIO 27 Hash-tiivisteen luominen 40
KUVIO 28 Esimerkki “Authorization”-otsakkeen käyttämisestä 41
KUVIO 29 Sovelluksen putki 42
KUVIO 30 Esimerkki attribuuttien käytöstä 43

KUVIO 31 Uuden kommentin lisääminen 44
KUVIO 32 Uusi kommentti käyttöliittymällä 44
KUVIO 33 PATCH metodin esimerkki 45

Lyhenteet

AD	Active Directory
UFW	Uncomplicated Firewall
JAMK	Jyväskylän Ammattikorkeakoulu
SPA	Single-page Application
XHR	XMLHttpRequest
LTS	Long Term Support
DOM	Document Object Model
UFW	Uncomplicated Firewall
SSL	Secure Sockets Layer
TLS	Transport Layer Security
IP-address	Internet Protocol address
XHR	XMLHttpRequest
JRE	Java Runtime Environment
NPM	Node Package Manager

1 Johdanto

1.1 Alkusanat

Tämä kyseinen projekti on kohdannut muutamia vastoinkäymisiä sekä viivästymisiä ajan saatossa. Alun perin tavoitteena ollut viiden hengen ryhmä palvelun kehitykseen supistui jo alkumetreillä kahden hengen ryhmään, jonka jälkeen ryhmän kommunikaatio vaikeudet aiheuttivat teiden eriytymisen omiksi projekteikseen. Tämän lisäksi projektia varten on kirjoitettu useampikin vajaa toiminteinen prototyyppi ennen teknologioiden lukkoon lyömistä. On kuitenkin mahdollista, että kaikki nyt tehty työ sekä saatu kokemus saattavat tulevaisuudessa olla pohjana jollekin suuremmalle.

Tämän opinnäytetyön tarkoitus on esitellä sekä tutkia järjestelmää, joka on kehitetty tämän samaisen opinnäytetyön yhteydessä Jyväskylän Ammattikorkeakoulun IT-instituutille opinnäytetöiden seurantaa varten. Tärkeimpinä tavoitteina on pidetty yksinkertaista, helposti käytettävää järjestelmää jonka avulla on riittävän selkeää käydä opinnäytetyöprosessi läpi oppilaskohtaisesti. Tämä suoraan asettaa tiettyjä kriteerejä henkilön todennukselle sekä käyttöliittymän loogisuudelle.

Opinnäytetyön tarkoituksena ei ole tarkasti kuvailla jokaisen toiminnon teknistä toteutusta, vaan keskittyä enemmän selkeään yleiskuvaan toimintojen toteutuksesta. Esimerkiksi palvelun kaikkea koodia ei lähdetä selvittämään auki lukijalle, vaan tarvittaessa keskitytään muutamaan oleelliseen kohtaan sekä esitellään yleisiä käytäntöjä ohjelman rakenteista. Lähes kaikki lähdekoodit löytyvät liitteistä muutamaa poikkeusta lukuun ottamatta. Mahdollisesti käytettyjä kolmannen osapuolen toteutuksia ei listata lähdekoodeissa.

1.2 Työn tilaaja

Jyväskylän Ammattikorkeakoulu (lyhyesti myös JAMK) on kansainvälinen, useammasta erillisestä yksiköstä (ammatillinen opettajakorkeakoulu, hyvinvointiyksikkö, liiketoimintayksikkö sekä teknologiayksikkö) koostuva korkeakoulu. Toimipisteitä on eripuolilla Jyväskylää sekä lisäksi yksi toimipiste löytyy Saarijärven Tarvaalasta. (Tutustu ja menesty, 2015)

Jyväskylän Ammattikorkeakoulussa opiskelijoita on vuonna 2015 noin 8500. (m.t.)

1.3 Työn lähtökohdat

Jyväskylän Ammattikorkeakoulun IT-instituutilla on tarve palvelulle, joka korvaisi hankalasti käytettävän Excel-taulukon. Kyseinen Excel-taulukko käytännössä sisältää kenttiä opinnäytetyö prosessin eri vaiheista, jotka tulisi korvata helposti käytettävällä web-palvelulla. Palvelun käyttäjäkunta olisi myös huomattavasti laajempi kuin alkuperäisen Excel-taulukon, sillä kyseinen Excel-tiedosto sijaitsi jaetulla verkkolevyllä siten, että vain yksi henkilö kerrallaan pystyi sitä muokkaamaan. Lisäksi tälle verkkolevylle pääsi ainoastaan koulun henkilökunta, eikä esimerkiksi oppilaat voineet jakaa opinnäytetöitään yhdestä keskitetystä paikasta. Opinnäytetyön jakamiseen liittyen myös prosessiin kuuluvien muiden dokumenttien tallentaminen ja lataaminen yhdestä keskitetystä paikasta käyttäjän oikeuksien niin salliessa tulisi hoitua mahdollisimman suoraviivaisesti. Kyseiset toimenpiteet kun on aikaisemmin hoidettu käytännössä sähköposti viesteillä.

1.4 Tavoitteet opinnäytetöiden seurantapalvelulle

Jyväskylän Ammattikorkeakoululla on käytössä Microsoftin Active Directory hakemistopalvelu, jonka oleellisin toiminnallisuus on tämän projektin osalta sen käyttäjien todentaminen. Koska jokaisella Jyväskylän Ammattikorkeakoulun oppilaalla sekä useimmilla henkilökunnan jäsenillä on Active Directory tunnukset, tulisi tätä palvelua voida käyttää myös opinnäytetöiden seurantapalvelun todentamisprosessissa.

Palvelun tulisi olla riittävän yksinkertainen, jotta kuka tahansa voi vain kirjautua sisään ja saada asiansa hoidettua ilman suurempia ongelmia. Tarvittaessa voidaan käyttää avustavia tekstejä ja muita seuraavan loogisen toimenpiteen osoittavia elementtejä, mutta luonnollisesti tällaisten esiintyminen palvelussa saattavatta olla viittaus ongelmasta sovelluksen logiikassa.

2 Toteutukseen valitut teknologiat

Toteutus voidaan ajatella kahdeksi erilliseksi osaksi. Käyttäjille palvelusta näkyy websovellus, eli käytännössä verkkosivu, jonka avulla käyttäjät käskyttävät varsinaista palvelua sen tarjoaman rajapinnan kautta. Tällaisen palvelun toteuttamiselle on monta tapaa, mutta usein on tapana eriyttää palvelun käyttäjälle näkyvä osa palvelun taustajärjestelmistä (sisältäen rajapinnat). Tämä mahdollistaa huomattavasti joustavamman kehitysprosessin lisäksi laajemmat mahdollisuudet palveluiden tarjoamiseen, sillä rajapintoja voidaan tarjota myös ulkoisille toimijoille esimerkiksi palveluintegraatioiden toteuttamiseksi. Nämä ajatukset ovat olleet lähtökohtina projektissa käytettävien teknologioiden valintaan.

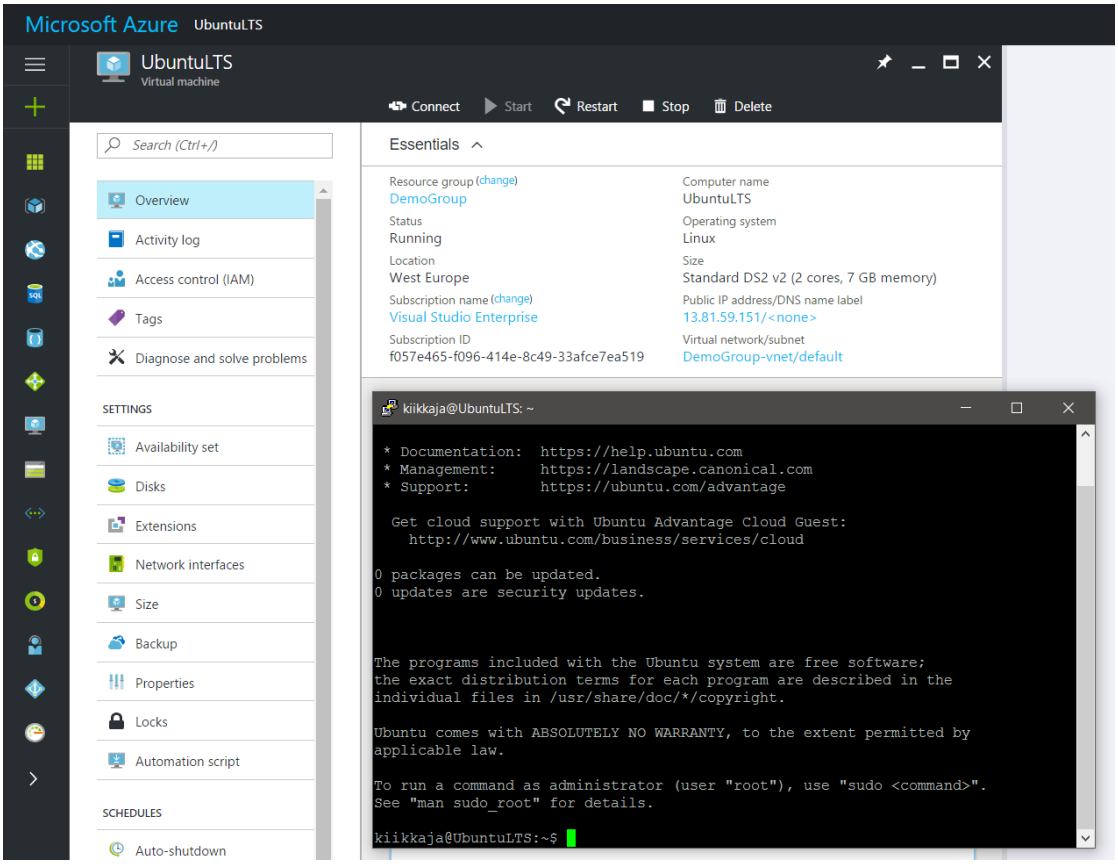
2.1 Linux (Ubuntu 16.04 Xenial Xerus, LTS)

Ubuntun 16.04 version on valittu sen työkalujen sekä LTS-statuksen (Long-term support), joka takaa sille usean vuoden tuen tietoturvapäivityksien osalta. Lisäksi se toimii hyvin yhteen muiden valittujen teknologioiden kanssa ja on suurimman käyttäjäkunnan omaava Linux-distribuutio taaten valtavat resurssit ja dokumentaation lähes jokaiseen tilanteeseen. Toisin kuin usean muun distribuution kohdalla, Ubuntua kehittää Canonical Ltd. (joka on kaupallinen yritys) taaten sille tarvittaessa myös viraллista tukea.

Myös muita distribuutioita harkittiin projektin toteutusta varten (esimerkiksi CentOS), mutta lopulta päädyttiin valitsemaan yleisin käytössä oleva Ubuntu. Muut jakelut usein tuntuivat Ubuntuun verrattuna tarjoavan pieniä parannuksia joillain osa-alueella ja kompromissilta monilla muilla. Joka tapauksessa Linuxin ytimen tuli olla riittävän uusi, jotta jotkin muut sovellukset toimisivat sujuvasti ja oikein.

Ubuntun tilalle aluksi kokeiltiin jo mainittua CentOS-jakelua, tarkemmin 6.7 versiota, mutta käytännössä tämän käyttäminen osoittautui mahdottomaksi liian vanhan ytimen vuoksi. Jotkin ongelmat onnistuttiin kiertämään kääntämällä käytetyt sovellukset erikseen tälle versiolle, mutta jotkin kirjastot vaativat erityisesti uudemman kääntäjän. Lisäksi kyseisen jakelun päivittäminen uudempaan versioon oli käytännön mahdotonta ilman täydellistä uudelleen asennusta.

Toteutuksen testaamiseen on käytetty Microsoftin Azure-pilvipalvelussa ajettavaa virtuaalitietokonetta. Palvelun avulla on helppoa ja nopeaa luoda uusia virtuaalikoneita esimerkiksi tasaamaan kuormaa, mutta asennuksen testaamista varten tulisi riittää yksi virtuaalikone. Kuviossa 1 kuvankaappaus Azuren käyttöliittymästä sekä ensimmäisestä kirjautumisesta virtuaalikoneelle. Kirjautumista varten on konfiguroitu RSA-avainpari, ja ilman avaimen salaista osaa palvelimelle ei voi kirjautua.



KUVIO 1 Azure ja terminaali näkymä virtuaalikoneella

2.2 Angular 2 ja TypeScript

Angular 2 on erittäin suositun AngularJS-sovelluskehyksen seuraava versio. Koska Angularin käyttäminen käytännössä vaatii myös sen metodologian omaksumisen, on sen yleistyminen ollut huomattavasti hitaampaa kuin sen edeltävän version johtuen lähinnä sen lukuisista muutoksista tapaan toteuttaa sovelluksia. Lopulta kuitenkin on Angularilla tarjota hyvin selkeä tapa rakentaa sovelluksia, mikä on erityisen hyvä ominaisuus useampien kehittäjien kanssa toimimiselle: sovelluskehyksen asettamat suuntaviivat pitävät projektin helpommin hallittavana pakettina.

Angular-sovelluskehystä käyttäessä on luontevaa käyttää TypeScriptia, joka laajentaa tavallista JavaScriptia (opinnäytetyön kirjoitushetkellä yleisin käytössä ollut JavaScript versio oli edelleen ECMA-262-standardiin, yleisemmin tunnettu ECMAScript 5:na, perustuva JavaScript) esimerkiksi vahvalla tyypityksellä. Käytännössä kirjoitettu ohjelmakoodi käännetään ennen käyttöä tavalliseksi, toimivaksi JavaScriptiksi eikä TypeScript-koodia sellaisenaan voi käyttää suoraan. Käännön aikana myös tehdään muun

muassa tyyppitarkistukset, ja kääntäjä ilmoittaa virheellä, mikäli käytetyt tyypit eivät vastaa haluttuja tyyppejä.

2.3 PostgreSQL

PostgreSQL (johon jatkossa viitataan nimellä Postgres, joka on hyvin yleisesti käytössä oleva nimi PostgreSQL:lle) on alun perin BSD lisensoitu, nykyisin PostgreSQL lisenssin alainen tietokantojen hallinta järjestelmä (PostgreSQL lisenssi on hyvin samankaltainen MIT/BSD lisenssien kanssa). Tämä käytännössä tarkoittaa sitä, että Postgresin käyttäminen ei tuo projekteille ylimääräisiä kuluja sekä sen lähdekoodeja voi tarvittaessa muokata. (The PostgreSQL Global Development Group, License)

Postgresille vaihtoehtoina samankaltaisina tuotteina olivat myös MySQL sekä Microsoft SQL Server, joista jälkimmäisellä rajoitteiksi tulisivat sen suljettu lähdekoodi sekä lisensointi: yli 10 gigatavun tietokannat vaativat maksullisen lisenssin. Tämän lisäksi SQL Serverin Linux varianttia ei vielä opinnäytetyötä kirjoittaessa ollut julkaistu, joskin tällainen on tulossa.

MySQL:n ja Postgresin välillä ei useimmissa käyttötarkoituksissa ole kovin paljon eroja, ja molemmat olisivat varmasti toimineet palvelun tietojen säilömiseen sekä käsittelyyn erittäin hyvin. Lopulta valinta Postgresin eduksi kuitenkin MySQL:n liberaalimman SQL standardin tulkitsemisesta. Tämä tarkoittaa käytännössä sitä, että esimerkiksi konversio Postgresista Oraclen tietokantajärjestelmiin on joissakin tapauksissa huomattavasti mutkattomampi prosessi.

Lisäksi Postgres-tietokantaosaajat ovat myös harvinaisempia sekä keskimäärin paremmin palkattuja kuin MySQL vastaavat (joskin MySQL on huomattavasti suositumpi ja yleisempi valinta näistä kahdesta), jolla on myös saattanut olla vaikutusta kantapalvelimen valinnassa.

2.4 Microsoft .NET Core

Microsoftin toimitusjohtaja muutoksen jäljiltä Microsoft on kunnostautunut avoimen lähdekoodin sovellusten tuotannossa, ja .NET Core on ehkä yksi kiinnostavammista

tapauksista. Kyseessä on käytännössä Windows maailmassa hyvin tutun .NET Frameworkin avoimen lähdekoodin versio Linux, OSX sekä tietysti Windows alustoille. Lukuun ottamatta muutamia käyttöjärjestelmäkohtaisia toteutuksia, kaikki alustat jakavat saman koodin tarjoten hyvät jakelu mahdollisuudet uusille .NET Core sovelluksille.

.NET Core sovellusten ohjelmointi sujuu luontevimmin C#-kielellä. Lisäksi funktionaaliseen ohjelmointiin tarkoitetun F#-kielen käyttäminen onnistuu.

Tässä projektissa on pääasiassa käytetty C#-kieltä.

Koska .NET Core:n päälle toteutettu palvelu on hyvin keskeinen osa projektia, on hyvä mainita kaksi tällä käytettyä kolmannen osapuolen kirjastoa: Npgsql (Postgres yhteyksiä varten) sekä .NET Corelle käännetty Novell.LDAP-kirjasto, jolla voidaan autentikoida käyttäjä koulun käyttämää AD hakemistoa vasten.

3 Palvelun käyttäminen

Palvelulla on oltava jonkinlainen toimintalogiikka, jonka puitteissa palvelua käytetään ja mitä palvelu antaa käyttäjien tehdä. Oikein toteutettuna palvelu on myös deterministinen, jolloin jokainen arvo tai toiminto tietyssä tilanteessa palauttaa aina saman odotetun lopputuloksen. Näin käyttäjän avatessa esimerkiksi oman opinnäytetyönsä tiedot, hän saa aina juurikin oman työnsä tiedot tietokannasta käyttöliittymän esitettäväksi eikä jonkun toisen tietoja.

Tässä osiossa on kuvailtu pääasiassa käyttäjän sekä ohjelmalogiikan välistä toimintaa. Esimerkki tilanteena voidaan ottaa vaikkapa tiedostojen lataaminen: mitä tapahtuu sellaisessa tilanteessa, jossa käyttäjä haluaa ladata uuden tiedoston, esimerkiksi opinnäytetyöhön liittyvän dokumentin, järjestelmään talteen? Voiko käyttäjä näin tehdä, ja tulisiko tämän olla jokaisessa tilanteessa mahdollista?

3.1 Käyttäjän tunnistaminen

Pääsääntöisesti käyttäjä kirjautuu palveluun käyttämällä jo olemassa olevia AD tunnisteitaan. Näin voidaan varmistaa, että käyttäjä on aktiivinen opiskelija sekä tehdä palvelun käyttäminen mahdollisimman helpoksi. Kun käyttäjä ensimmäisen kirjautuu

palveluun, luodaan hänelle hänen AD-tunnuksiaan vastaava profiili sekä siihen linkitetty opinnäytetyö projekti. Käyttäjä voi kirjautumisen jälkeen täyttää profiilitietoihinsa esimerkiksi oikean nimensä sekä puhelinnumeronsa, sillä oletuksena käyttäjästä voidaan tietää vain käyttäjätunnus. Käyttäjäprofiileihin voidaan myös lisätä oleellista tietoa käyttäjän opintojen aikataulutuksesta, joita voidaan käyttää myöhemmin statistiikan esittämiseen oppilaiden valmistumisesta. Oppilas ei voi itse muokata päivämääriä profiilissaan.

Palveluun on mahdollista myös kirjautua käyttäen niin kutsuttuja paikallisia tunnuksia, mutta nämä tunnukset ovat varattuja ylläpidon tunnuksille sekä järjestelmän testaamiselle. Sisäisen kirjautumispalvelu ei tarjoa minkäänlaisia mahdollisuuksia uusien tunnuksien rekisteröimiseen, vaan nämä tunnukset on luotava suoraan tietokantaan.

3.2 Yleisnäkymä

Yleisnäkymä tarjoaa nopean katsauksen käyttäjän huomiota kaipaaviin töihin. Pääasiassa tämä näkymä on tarkoitettu opinnäytetöiden ohjaajille, jotka voivat nähdä kerralla kaikki heidän ohjauksessaan olevat opinnäytetyöt kerralla. Tämän lisäksi näkymä näyttää valvojalle oletuksena kaikki opinnäytetyöt, jotta tämä voi asettaa uusille aloitetuille töille ohjaajat.

Sama näkymä myös näyttää vertaisarvioijalle arvioitavan opinnäytetyön.

3.3 Opinnäytetyönäkymä

Opinnäytetyönäkymä on opinnäytetöiden tietojen muokkaamiseen tarkoitettu näkymä. Tämä näkymä sisältää käytännössä kaiken opinnäytetyöhön liittyvän, lukuun ottamatta opinnäytetyön tiedostoja sekä seurantapalaverien kalenterimerkintöjä.

Näkymään voi tehdä muokkauksia ainoastaan kyseiselle opinnäytetyölle asetetut ohjaajat.

3.4 Tiedostonäkymä

Opinnäytetöiden tiedostot on hyvä sijoittaa yhteen paikkaan, jotta vältytään turhalta sähköpostien liitetiedostojen sekä jaettujen kansioiden lähettämiseltä.

Näkymä käytännössä toimii siten, että opinnäytetyötä tekevä oppilas voi lisätä tiedostoja ”Tekijän tiedostot”-kohdan alle. Nämä tiedostot ovat saatavilla kaikille, sekä ohjaajille että vertaisarvioijalle. Lisäksi ohjaajat voivat lisätä tiedostoja kohdan ”Ohjauksen tiedostot”, jotka ovat saatavilla vain muille ohjaajille sekä oppilaalle itselleen. Näitä tiedostoja voisivat olla esimerkiksi korjausehdotuksien jakaminen dokumentteihin.

Viimeisenä kohtana on ”vertaisarviointi”, joka nimensä mukaisesti on vain vertaisarvioijan käytettävissä, ja kyseisen kohdan tiedostot ovat kaikkien nähtävillä.

3.5 Kalenterinäkymä

Kalenterinäkymän pääasiallisena tarkoituksena on opinnäytetyöhön liittyvien tapaamisen merkitseminen. Kalenterimerkinnät ovat yksilöllisiä opinnäytetyölle, ja opinnäytetyön tekijä sekä molemmat ohjaajat näkevät ne.

Kalenterinäkymän käyttö toimii käytännössä siten, että työn ohjaaja lisää uuden merkinnän ja ilmoittaa tässä päivämäärän, kellonajan sekä kuvauksen tapahtumasta.

4 Nginx ja tietoliikenne

Tämä palvelun toteutukseen keskittyvä osuus koostuu seuraavista neljästä asiasta: verkkorajapintana toimivasta Nginx-palvelimesta, tietokantapalvelimena toimivasta PostgreSQL:sta sekä ohjelmarajapintana toimivasta .NET Core -sovelluksesta että näiden konfiguroimisesta. Neljäntenä kohtana on vielä näitä palvelimia tukevat käyttöjärjestelmä tason asetukset ja ohjelmat. Tätä osiota suositellaan myös käytettävän mahdollisissa palvelun asennuksissa lähdemateriaalina myös jatkossa.

4.1 Palomuurin asetukset

Ubuntun oletuspalomuuri sovellus on Uncomplicated Firewall, lyhyesti ufw. Kuten useimmat muut Linux-palomuuri sovellukset, ufw käyttää Linuxin *netfilter* alijärjestelmää verkkoliikenteen manipuloimiseen. Tämä käytännössä tarkoittaa sitä, että *netfilter* alijärjestelmällä voidaan sallia liikennettä vain muutamaan palveluiden kannalta

oleelliseen porttiin. Ufw tarjoaa *netfilterin* asetuksien säätämiseen käyttöä varten hyvin selkokielisiä ja yksinkertaisia komentoja. (Ubuntu, Ubuntu Server Guide)

Ensin varmistetaan, että ufw on käytössä. Koska sovellus on oletuksena osa Ubuntun asennuspakettia, ei sitä tarvitse erikseen asentaa:

sudo ufw enable

Tämän jälkeen vielä varmistetaan, että kaikki liikenne palvelinta kohti estetään oletuksena:

sudo ufw default deny incoming

Tämä varmistaa sen, ettei ulkoapäin vastaanoteta mitään liikennettä portteihin, joita ei ole erikseen määritelty sallituiksi. Tämän tarkoituksena on pienentää mahdollisia palvelinta vastaan kohdistuvien hyökkäyksien hyökkäyspinta-alaa mahdollisimman pieneksi.

Seuraavaksi avataan portti SSH-liikenteelle:

sudo ufw allow 22/tcp

Tästä edellä mainitusta komennosta huomataan, että ufw:n käyttö on hyvinkin yksinkertaista. Netfilter päästää sallii nyt SSH-liikenteen palvelimen ja asiakasohjelmien välillä. Komennon lopussa oleva ”/tcp” parametri vielä erikseen määrittelee vain TCP-protokollan sallituksi liikenteen muodoksi.

Palomuuriin lisätään vielä sääntöjä Nginx:n asennuksen yhteydessä.

4.2 Nginx-palvelin

Nginx:n asentaminen Ubuntu-palvelimelle on yksinkertaista. Terminaali näkymässä seuraava komento:

sudo apt-get install nginx

Asentaa Nginx:n sekä tämän vaatimat riippuvuudet olettaen, että järjestelmävalvoja vielä komennon kirjoittamisen jälkeen hyväksyy asennuksen.

Esimerkki asennuksessa käytössä on myös Let's Encryptin SSL-sertifikaatti (SSL, Secure Sockets Layer, on protokolla verkkoliikenteen salaamiseen) mutta koska sertifikaatteja voi saada myös muilta tahoilta sekä olemassa on myös mahdollisuus luoda oma, itse allekirjoitettu sertifikaatti, ei Let's Encryptin sertifikaatin hankinta prosessia käydä sen tarkemmin läpi. Ainoa käytännön vaatimus Let's Encryptin sertifikaatille kuitenkin on, että käytössä on oma domain. Pelkkä IP-osoite (laitteen yksilöllinen osoite verkossa) ei riitä.

Oletusasennuksen jälkeen avataan */etc/nginx/sites-enabled/default* asetustiedosto sopivalla tekstieditorilla. Tässä tapauksessa on käytetty "nano" nimistä tekstieditoria, joka tulee *Ubuntun* asennuspaketin mukana.

Nginx:n asennustiedoston käyttäminen on pienen totuttelun jälkeen melko yksinkertaista. Asetuksia voi ajatella omina olioinaan, joiden ominaisuudet määritellään aaltosulkujen sisällä. Kuvion 2 esimerkistä saa mallia uusien blokkien kirjoittamiseen.

```
root@Production: ~
  GNU nano 2.5.3      File: /etc/nginx/sites-enabled/default          Modified

server {
        listen 80 default_server;
        listen [::]:80 default_server;
        server_name jiisoni.com www.jiisoni.com;
        return 301 https://$server_name$request_uri;
       }

server {
        listen 443 ssl;
        listen [::]:443 ssl;
        include snippets/ssl-jiisoni.com.conf;
        include snippets/ssl-params.conf;
        server_name api.jiisoni.com;

        location /api {
                proxy_pass http://localhost:5000/api;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header X-Forwarded-Proto $scheme;

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

KUVIO 2 Nginx:n asetukset

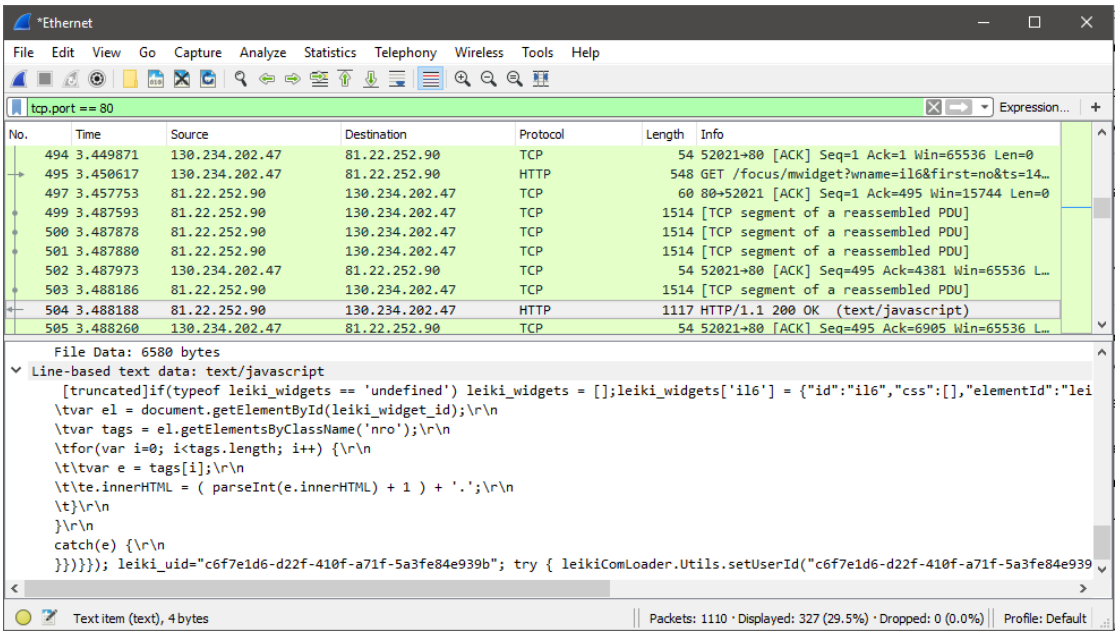
Kuvion 2 esimerkissä huomattava, että osoite "jiisoni.com" on palvelun kehityksen aikana käytössä ollut osoite, joka tulisi tuotanto käytössä korvata omalla osoitteella.

Kuviossa 2 nähdään yksi kokonainen asetusblokki, sekä osittain toista blokkia. Ensimmäisessä blokissa Nginx:lle kerrotaan, että sen tulisi kuunnella porttia 80 (oletusportti salaamattomalle www-palvelulle). Portista 80 ei kuitenkaan haluta tarjota mitään varsinaista palvelua asiakkaalle päin, joten samassa blokissa ohjeistetaan Nginx lähettämään asiakkaalle uudelleenohjaus viesti *server_name* muuttujalle annettuun osoitteeseen, sisällyttäen *https*-skeema osoitteeseen.

Huomio: Demoympäristön konfiguraatiot löytyvät kokonaisuudessaan liitteestä 1.

Asiakasohjelman tulisi siis lopulta yhdistää *https*-etuliitteellä palvelimeen uudestaan, jolloin oletuksena käytetään TLS-salattua (Kyseessä on vain SSL:n uudempi versio, ei kokonaan uusi protokolla) 443 porttia.

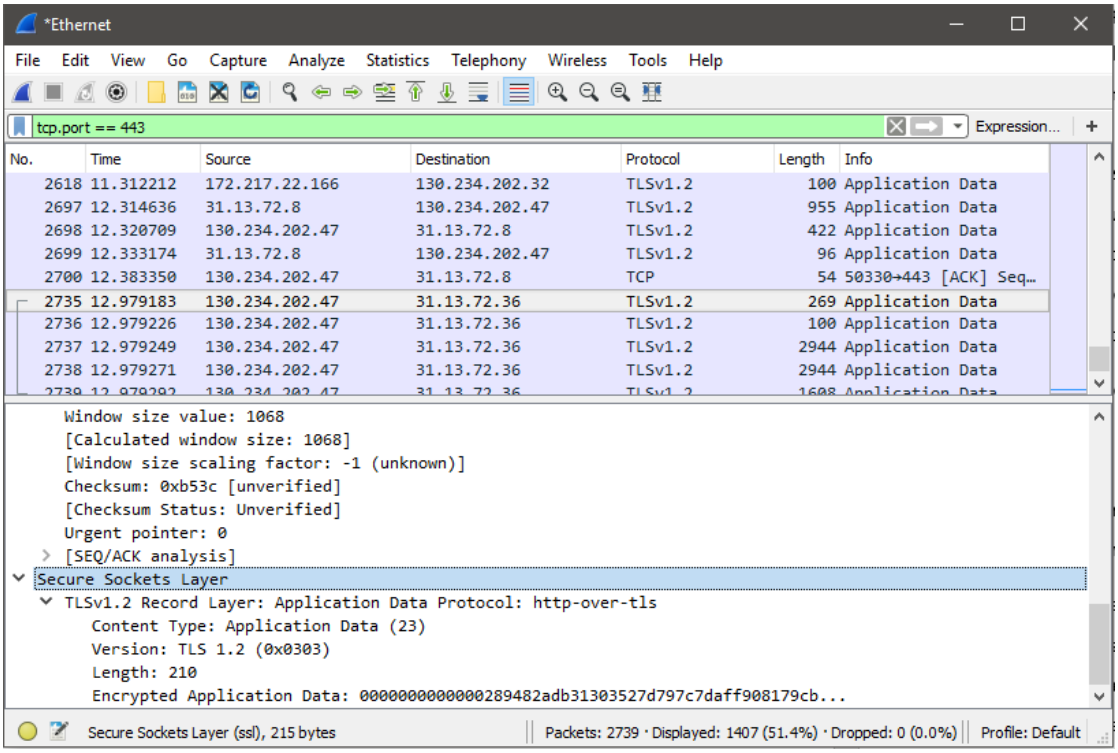
443 porttia käytettäessä asiakasohjelman ja palvelimen välille muodostetaan salattu yhteys, joka mahdollistaa tiedon siirtämisen ilman, että kukaan tai mikään välissä oleva reititin tai muu verkkolaite ei voi suoraan lukea siirrettävää tietoa tietämättä salaukseen käytettyjä avaimia. Seuraavassa kuviossa 3 on esimerkki WireShark-ohjelmalla kaapatusta verkkoliikenteestä, jossa satunnaisesti valitun verkkosivun JavaScript-koodia salaamattomana.



KUVIO 3 Salaamaton verkkoliikenne

Kuten kuviosta 3 on nähtävissä, on koodi täysin selkokielistä luettavaa. Kaikki salaamaton verkkoliikenne on yhtähelposti luettavissa otsaketietoja myöten. Seuraavassa kuviossa 4 on sen sijaan esimerkki toteutetun palvelun salatusta verkkoliikenteestä,

ja tätä liikennettä ei pysty mikään muu kuin asiakasohjelma (tässä tapauksessa verkkoselain) tulkitsemaan selkokielisenä koodina. Kuviossa 4 palvelun salatusta datasta voidaan nähdä pieniosa kohdassa ”Encrypted Application Data”.



KUVIO 4 TLS-salattu verkkoliikenne

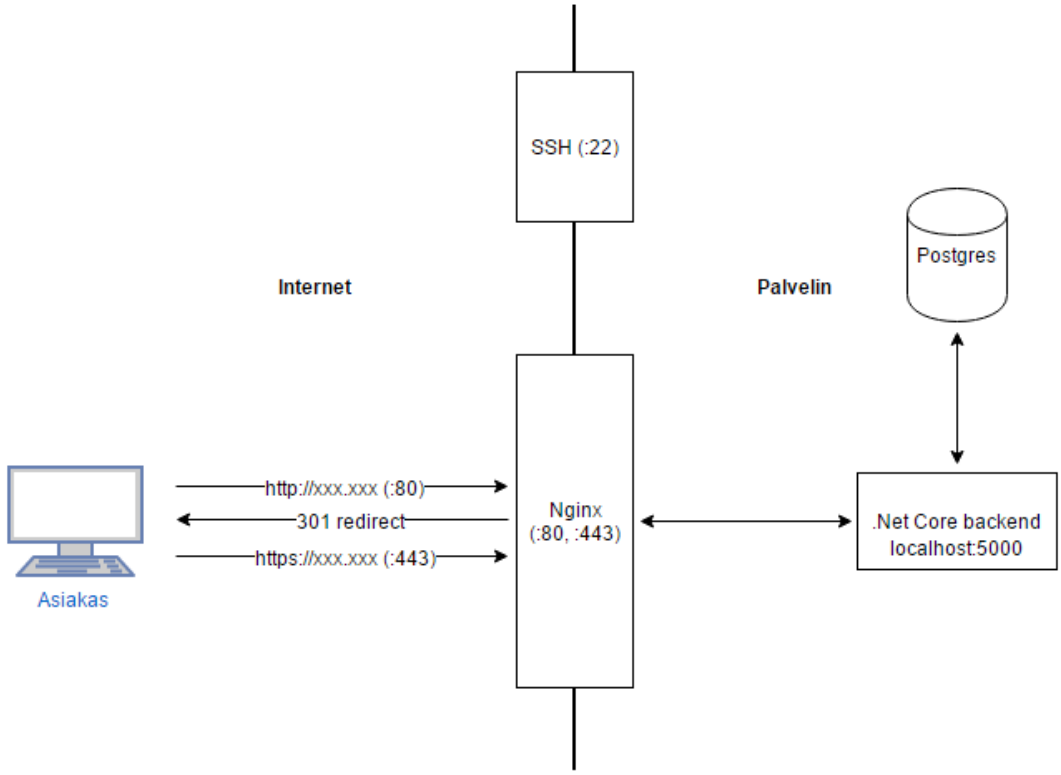
Esimerkkeinä käytetyistä yhteyksistä selviää, salaamaton yhteys kulkee asiakasohjelman sekä palvelimen välillä selkokielisinä pätkinä, kuten kuviosta kolme on nähtävissä (esimerkki on Iltalehti.fi sivustolta, ja esimerkissä on nähtävissä osa kyseisen sivuston käyttämästä JavaScript-koodista selkokielisenä). Kuviossa 4 sen sijaan on yhdistetty Jiisoni.com sivustolle (joka on toiminut tämän projektin demoympäristönä), ja esimerkin sisältämä datapaketti on nähtävissä vain epämääräisenä rivinä erilaisia merkkejä. Jos oletetaan, että tällaista esimerkin kaltaista dataa kaapattaisiin jollain verkkolaitteella joka ei kuulu palveluntarjoajalle, olisi käyttäjän tiedot erittäin suuressa vaarassa joutua väärin käytetyiksi ensimmäisen esimerkin tapauksessa. Salatun yhteyden tapauksessa mahdollisen rikollisen toimijan tulisi myös tietää salaukseen käytetty yksityinen avain sekä julkinen avain (avaimien sopimiseen käytettävä protokolla ei varsinaisesti kuulu tämän työn aiheisiin, joten sitä ei käsitellä sen tarkemmin. Tämän osalta kiinnostuneille IETF:n RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2 -dokumentti tarjoaa kaiken tarvittavan tiedon). (The Transport Layer Security (TLS) Protocol Version 1.2, 2008)

Koska salaus on tämän projektin tapauksessa toteutettu suoraan Nginx:n ulkomaailmaan näkyvään porttiin, ei palvelimen sisällä olevaa dataa tarvitse erikseen suoraan salata. Kaikki WWW-liikenne palvelimelle salataan automaattisesti, sillä 80-portin kautta ei mihinkään palvelimen palveluun Nginx:n uudelleenohjaus sääntöä lukuun ottamatta voi yhdistää. Lisäksi kaikki muut palvelut on asetettu kuuntelemaan vain palvelimen sisäistä liikennettä ja palvelimen palomuuri estää kaiken muun liikenteen lukuun ottamatta yhteydenottoja portteihin 22, 80 sekä 443.

Nginx:n konfiguraatiossa kerrotaan palvelimelle toisen konfiguraatiotiedoston nimi, joka on *ssl-params.conf*. Tämä asetustiedosto kertoo Nginx:lle, mitä ja kuinka TLS-sertifikaatteja tulisi käyttää (asetustiedostossa käytetään SSL:aa parametrien nimissä, vaikka kyseessä on kuitenkin protokollan uudempi versio, TLS).

Lisäksi Nginx:n konfiguraatiossa on määritelty erikseen polulle *"/api"* proxy-asetukset. Tämä tarkoittaa sitä, että Nginx ohjaa polkuun */api* tulevat viestit toisen palvelimen prosessoitaviksi. Asiakkaat eivät siis suoraan ole yhteydessä varsinaiseen palveluun, vaan välissä oleva proxy-palvelin käsittelee tietoa ennen sen ohjaamista oikeaan osoitteeseen. Tässä tapauksessa Nginx lähettää asiakasohjelmilta tulevat viestit paikallisen verkon porttiin 5000. Lisäksi otsaketietoja joudutaan muokkaamaan, jotta vastaanottava palvelin tietää, minne tiedot tulee lähettää takaisin kyselyjen prosessoinnin jälkeen, sekä varmistetaan, ettei tiettyjä tärkeitä otsakkeita tiputeta/muuteta ohjauksen yhteydessä. Nginx:n ja paikallisen verkon palvelinten välinen liikenne ei ole salattu.

Kuviossa 6 näkyvillä yksinkertaistettu, havainnollistava esitys palvelimen yhteysrakenteista.



KUVIO 5 Internetin ja palveluiden välinen liikenne

Lopuksi viimeinen asetusblokki Nginx:n konfiguraatiotiedostossa sisältää käyttöliittymän tarjoamisen asiakkaille. Käyttöliittymä on käytännössä kokonaan toteutettu HTML:llä sekä JavaScriptilla, joten palvelimen puolella ei tarvitse prosessoida mitään erikseen, vaan kaiken voi lähettää asiakkaalle pyyntöjen perusteella. Nginx pystyy hoitamaan tällaisen yksinkertaisen tiedostojen jakamisen ilman erillistä prosessointia palvelinta. Pienenä erikoisuutena, joka juontuu Angular 2:n reitityksen toteutuksesta, käsketään Nginx:aa ohjaamaan asiakas index.html sivulle jo sivua ei löydy. Tämä johtuu siitä, että Angular normaalisti käyttää asynkronisia kutsuja rajapintaa vasten, eikä tarvitse koko sivun uudelleen lataamista näkymän muuttamiseksi. Mutta jos esimerkiksi käyttäjä komentaa selainta lataamaan sivun kokonaan uusiksi samalla kun hän on jollakin alisivulla (esimerkiksi opinnäytetöiden liitteet-sivu), palvelin palauttaisi normaalisti 404 virheen (sivua ei löytynyt) jos erillistä uudelleenohjausta ei ole määritelty. Kuviossa 5 on esillä kuvankaappaus web-sivujen tarjoamisesta Nginx:lla.

```
server {
        # SSL configuration
        listen 443 ssl http2 default_server;
        listen [::]:443 ssl http2 default_server;
        include snippets/ssl-jiisoni.com.conf;
        include snippets/ssl-params.conf;

        root /var/www/html;

        index index.html;

        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                try_files $uri /index.html =404;
        }

        location ~ /.well-known {
                allow all;
        }
}
```

KUVIO 6 Web-sivujen tarjoaminen

Kuvion 6 esittämässä konfiguraatiossa parametrille *root* kerrotaan, mitä kansiota käytetään juurihakemistona. Tässä tapauksessa on valittu /var/www/html-kansio, jonka alta löytyy index.html tiedosto, joka tarjotaan asiakasohjelmalle, jos mitään muuta polkua ei ole määritelty pyynnössä. Koska käyttöliittymä on rakennettu SPA (Single-page application) periaatteella on index.html käytännössä ainoa HTML-tiedosto jota koskaan palvelimelta tarjotaan.

5 PostgreSQL ja tietokannan toteutus

Tietokannan toteutuksessa on pyritty pitämään kannan looginen malli mahdollisimman yksinkertaisena. Tietokannan toteutuksessa jokainen relaatio (joita vaihtoehtoisesti voidaan kutsua tauluiksi) esittää yhtä toteutuksen kannalta oleellista entiteettiä, kuten vaikkapa käyttäjää tai kommenttia. Ne siis kuvaavat asioita joilla on ominaisuuksia.

Tietokannan toteutuksessa on tärkeää ottaa huomioon viite-eheys. Relationaaliset tietokannat yleisesti mahdollistavat toisiinsa liittyvien tietojen linkittämisen toisiinsa viittaamalla esimerkiksi yhden relaation tietystä attribuutista toisen relaation samanmuotoiseen attribuuttiin (attribuuttien tyypin täytyy olla sama). Voidaan esimerkiksi määritellä, että opinnäytetyön tunniste attribuutti löytyy relaatioista käyttäjä sekä opinäytetyö, ja nämä yhdistämällä saadaan esimerkiksi viitattua halutulta käyttäjältä oikealle opinnäytetyölle. Lisäksi relationaalisessa tietokannassa

voidaan erikseen määritellä attribuutti ”ulkoiseksi avaimeksi”, jolloin attribuutti oletusarvoisesti viittaa toisen relaation haluttuun attribuuttiin. Attribuuteille voidaan myös määrittää ehtoja, esimerkiksi ”attribuutti ei saa olla tyhjä” tai että ”attribuutin täytyy olla uniikki”.

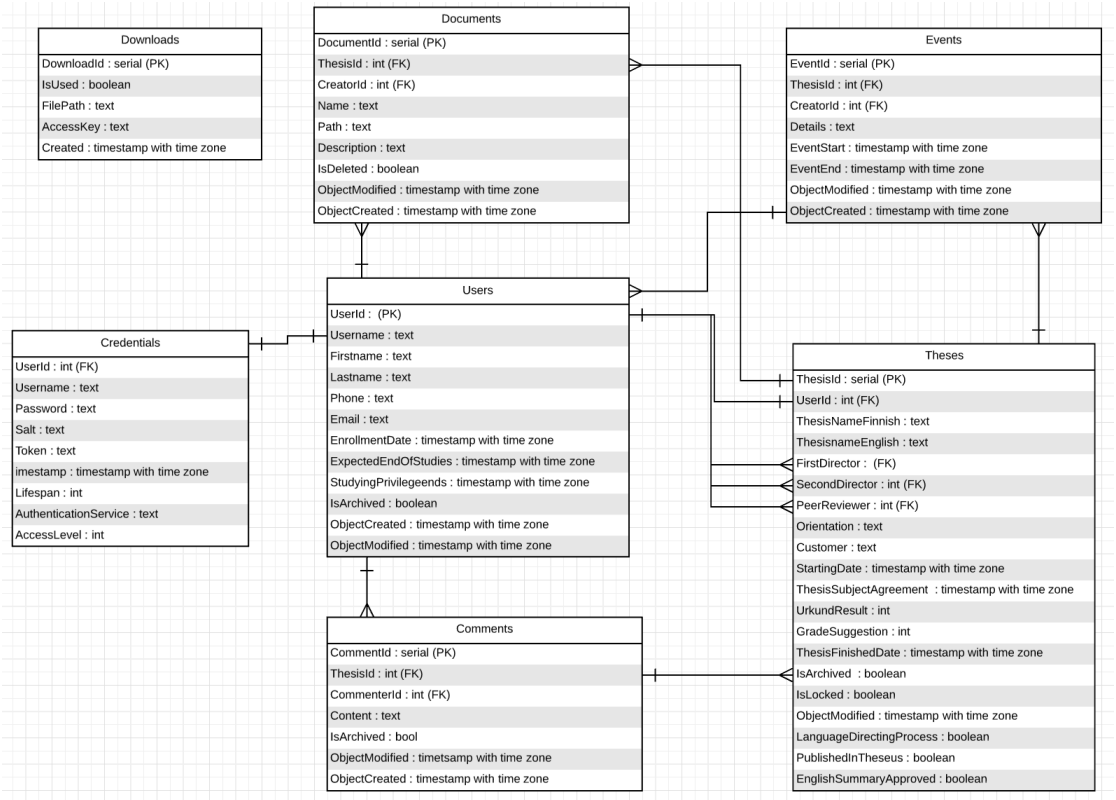
Kun viitataan ulkoisella avaimella toiseen relaatioon, voidaan määrittää myös tapahtumat viitatun relaation tietueen kadotessa.

5.1 Tietokannan rakenne

Tietokannan rakenne ja käyttötapa on hyvä suunnitella ennen toteutusta, sen sijaan että tehtäisiin lennosta uusia relaatioita jatkuvalla syötöllä. Lisäksi projektin tavoitteet ovat jo selvillä, joten suunnittelemalla tietokanta speksien pohjalta pitää fokuksen sen alkuperäisen tavoitteen saavuttamisessa.

Esimerkiksi tiedetään jo pohjasuunnittelun vaatimuksien pohjalta, että opinnäytetyöt tulevat olemaan koko projektin keskiössä. On siis loogisinta lähestyä tietokannan rakennetta opinnäytetöiden kautta; muut toiminnot voidaan rakentaa opinnäytetöiden ympärille.

Opinnäytetöiden lisäksi tietokantaan tulee hallittavaksi toinen erittäin tärkeä joukko entiteettejä, nimittäin käyttäjät. Käyttäjätiedot eivät voi olla suoraan sidoksissa opinnäytetöihin, sillä käyttäjillä saattaa olla esimerkiksi katseluoikeus useampaan opinnäytetyöhön samaan aikaan. Täten on järkevää suunnitella käyttäjät riippumattomiksi opinnäytetöistä, kuvion 7 esittämällä tavalla.



KUVIO 7 Tietokannan malli

Jokaisella entiteetillä on oma identiteettiavain, jolloin kaikki viittaukset hoidetaan ulkoisilla avaintiedoilla. Tämä ei toki olisi tarpeen esimerkiksi Users (1) – Credentials (1) relaatiossa, sillä käyttäjän perustietojen pääavain toimi oikein mainiosti myös saman käyttäjän oikeuksia kuvaavan relaation avaimena. Huomioitavaa toki on, että todellisuudessa pääavaimilla ja ulkoisilla avaimilla ei ole mitään toiminnallista tai edes teoreettista eroa (pois lukien tietokantapalvelimien mahdollinen automaattinen pääavaimen indeksoiminen). Kyseinen erottelu on olemassa vain selkeyden vuoksi.

Tässä kohdassa on ilmoitettu relaatioiden suhde muodossa *Relaatio (viittauksien määrä)*. Relaatioiden perässä, suluissa kerrottu arvo ilmoittaa, kuinka monta viittausta relaation attribuuteista tehdään toisiin relaatioihin. Aiemmin mainitussa tapauksessa (Users (1) – Credentials (1)) molemmat relaatiot sisältävät yhden attribuutin, joka viittaa toisen relaation attribuuttiin. Relaatiolla on myös mahdollista olla viittauksia monesta eri attribuutista useisiin eri relaatioihin, jolloin on tavanomaista ilmaista tällaista suhdetta muihin relaatioihin N:lla. Tällainen tilanne voitaisiin esittää

esimerkiksi Users (1) – Theses (N) relaatioiden tapauksessa, sillä Theses-relaatio sisältää useita attribuutteja, joilla viitataan Users-relaatioon.

5.2 Tietokannan relaatioiden kuvaukset

5.2.1 Credentials-relaatio

Credentials-relaation tarkoituksena on toimia varmistuksena käyttäjän session olemassa ololle. Käytännössä relaatiossa säilötään sessio kohtainen avainmerkkijono, jota vertaamalla asiakasohjelman lähettämään merkkijonoon voidaan käyttäjä tunnistaa yksilöllisesti jokaisen kyselyn yhteydessä. Lisäksi relaatio sisältää atribuutin avaimen eliniälle, joka kerrotaan sekunneissa, sekä viittaukset muihin relaatioihin joilla käyttäjän oikeudet voidaan varmentaa.

Hieman erikoisesti, ja ehkä jopa vähemmän optimaalisesti toteutettu Credentials-relaation tiedoilla ei ole pääavainta. Normaalisti pääavaimella yksilöllistettäisiin relaation tietueet, mutta toteutuksessa on sen sijaan päädytty käyttämän uniikkia viiteavainta. Jokaiselle käyttäjälle kirjoitetaan käyttäjätiliä luodessa uusi tietue Credentials-relaatioon siten, että ”UserId”-kentän arvoksi asetetaan uuden käyttäjän tunnistava lukuarvo. Vastaava lukuarvo tulee löytyä myös Thesis-, sekä Users-relaatioista.

Tämä tietysti tarkoittaa sitä, ettei järjestelmä tällaisenaan tue useaa sessiota yhdelle käyttäjälle. Jos sama käyttäjä kirjautuu toisella laitteella sisään palveluun, edellisen session päälle kirjoitetaan uusi tunnisteavain ja tämä taas vastaavasti lopettaa vanhemman session. Järjestelmän voisi pienehköllä vaivalla muuttaa tukemaan useampaa sessiota siten, että ”UserId” attribuutti muutettaisiin tyypiltään ei-uniikiksi ja päivittämällä palvelimen käyttäjän autentikoimiseen tarkoitetun väliohjaimen toimintalogiikkaa hieman.

5.2.2 Users-relaatio

Users-relaatio nimensä mukaisesti sisältää käyttäjän tietoja. Relaation attribuutit ovat melkoyksiselitteisiä. Relaation pääavaimeen (UserId) käytetään monessa muussa relaatiossa viittaamaan tiettyyn käyttäjään, ja tätä arvoa hyödynnetään sovelluksessa mm. oikeuksia tarkastaessa.

5.2.3 Documents-relaatio

Documents-relaatio kertoo järjestelmään lisättyjen dokumenttien (liitetiedostojen) perustiedot sekä metatietoja, esim. liitteen sijainnin levyllä. Käyttäjän tarkastellessa opinnäytetyön liitesivua näytetään kyseisen opinnäytetyön tunnisteen omaavat (ThesisId-attribuutin arvona) omaavat tiedostot käyttäjälle. Koska järjestelmässä on tiettyjä sääntöjä liitetiedostojen näkymiselle, tarkistetaan myös käyttäjän suhde opinnäytetyöhön. Näistä tiedoista voidaan esimerkiksi päätellä, milloin tiedostoa ei esitetä opinnäytetyön tekijälle, kun liitteen lisääjä on opinnäytetyön vertaisarvioija.

5.2.4 Downloads-relaatio

Downloads-relaatiolla ei ole viittauksia muihin tietokannan tietoihin. Relaation ainoana tehtävänä on toimia välitietona tiedostoja ladatessa: kun käyttäjä haluaa ladata tiedoston, palvelin varmistaa käyttäjän oikeudet kyseiseen tiedostoon ja tekee uuden tietueen Downloads-relaatioon, joka sisältää satunnaiseti generoidun avaimen (AccessKey). Tämän jälkeen palvelin lähettää asiakasohjelmalle avaimen tiedoston lataamiseksi. Asiakasohjelma tämän jälkeen lataa tiedoston käyttämällä latauksille tarkoitetun osoitteen parametrina luotua avainta. Osoitteen avaamisen yhteydessa palvelin kirjoittaa vastikään luodun tietueen ”IsUsed”-attribuutin arvoksi ”true”. Tämä tarkoittaa sitä, että generoitu latauslinkki ei enää palauta tiedostoa palvelimelta avattaessa.

5.2.5 Events-relaatio

Events-relaation tehtävänä on taltioida tapaamisia (kuten seurantapalaverit). Tapaamisista kirjoitetaan tärkeimmät tiedot, kuten alkamis-, ja loppumisajankohdat, sekä tapahtuman kuvaus (”details”-attribuutti”). Tapaamiset näytetään opinnäytetyön ”tapaamiset” -välilehdellä. Relaation ”UserId”-attribuutti kertoo tapaamisen tekijän, ja ”ThesisId” opinnäytetyön, jolla tieto tapaamisesta esitetään.

5.3 Theses-relaatio

Theses-relaatio sisältää kaikki opinnäytetyön seurantaan tarvittavista tiedoista. Muiden relaatioiden kannalta oleellisia tietoja on opinnäytetyön tunniste (ThesisId), sekä

käyttäjät, joita opinnäytetyöprosessi koskettaa. Relaation ”UserId” kertoo, kuka opinnäytetyön tekijä on. ”FirstDirector”-, sekä ”SecondDirector”-attribuutit kertovat, ketkä toimivat opinnäytetyön ohjaajina. Heillä on oikeus muuttaa opinnäytetyön seurantatietoja. Lisäki on ”PeerReviewer”-attribuutti, jolla määritellään työn vertaisarvioija. Tämä henkilö ei voi nähdä opinnäytetyöltä mitään tietoja, mutta voi lisätä arviointinsa liittesivulle. Muut relaation attribuutit ovat käytännössä nimensä puolesta yksiselitteisiä päivämääriä tai binääriarvoja (joskin tilana voi olla useimmissa tapauksissa myös NULL, eli eksplisiitti ”ei arvoa”-tila), jotka liittyvän seurannan prosessiin.

5.4 Käyttäjän tiedot

Tietokannan käyttäjistä pidetään yllä seuraavia (ehdottomia) tietoja: Käyttäjän tunniste (kokonaisluku), käyttäjän sähköposti (merkkijono). Nämä ovat minimi vaatimukset toimivan käyttäjätunnuksen olemassa oloon. Lisäksi aktiivisella käyttäjällä (käyttäjällä, joka on kirjautunut palveluun) tulee olla voimassa oleva sessio, jota ylläpidetään Credentials-taulussa. Tietoisen päätöksen (toteutuksen yksinkertaisena pitämisen) vuoksi palvelu ei salli montaa yhtäaikaista sessiota samalta käyttäjältä, vaan esimerkiksi toisella laitteella kirjautuessaan käyttäjä menettää edellisen laitteen session uudelle laitteelle. Kirjautuminen tapahtuu hyödyntämällä Active Directory-palvelimen tunnistautumis tietoja, sillä palveluun ei suoraan tallenneta salasanoja, ainoastaan satunnaisesti generoitua sessio-avainta. Lisäksi Credentials-taulussa pidetään käyttäjän käyttöoikeuksia kuvaavaa ”AccessLevel”-kokonaislukua. Oletuksena jokaiselle käyttäjälle annetaan arvo 1, joka kuvaa opiskelijaa. Vastaavasti arvo 2 olisi ohjaaja, ja 3 sihteeri (sihteerillä tarkoitetaan tässä tapauksessa henkilöä, jolla on oikeus nähdä kaikki opinnäytetyöt ja asettaa niille ohjaajat sekä vertaisarvioijat). Lopuksi on vielä arvo 4, joka kuvaa järjestelmän ylläpitoa. Kahdella viimeisellä roolilla on lähinnä nimellinen ero.

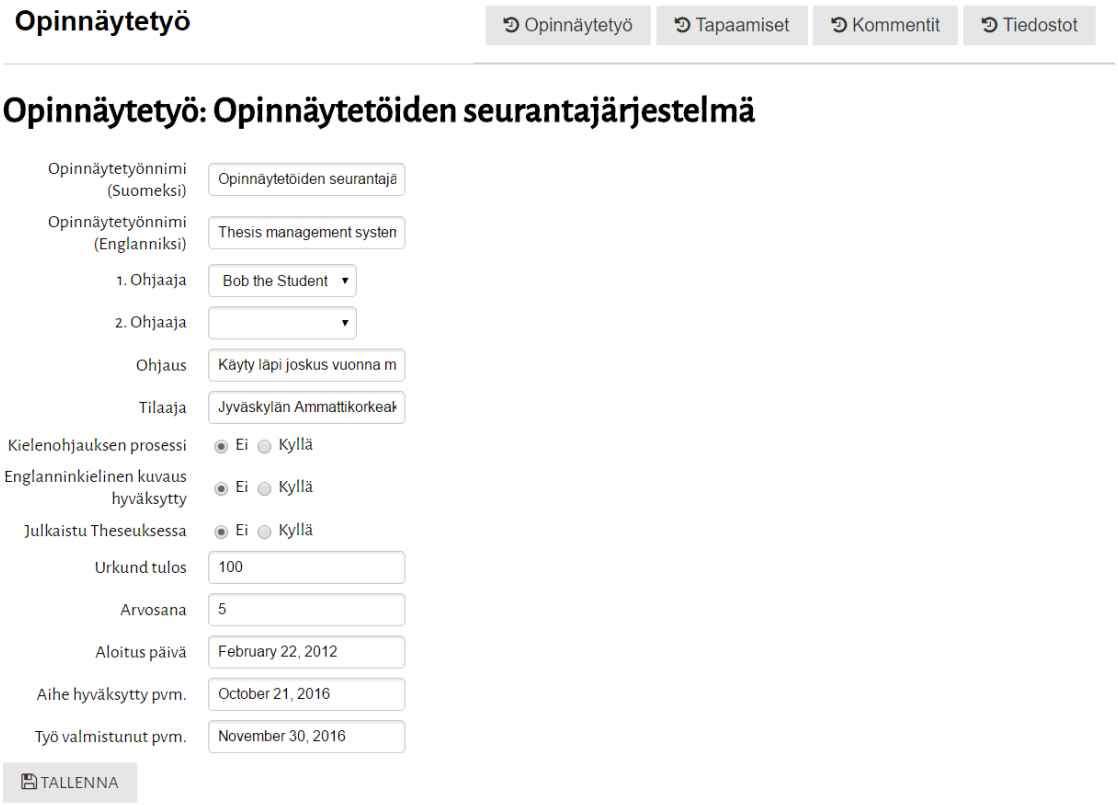
Olettamus projektia tehdessä on ollut, ettei palvelulla tule olemaan kovin suurta ruuhkaa missään vaiheessa sen elinkaarta. On toki mahdollista, että jatkokehityksen kautta palvelulle tulisi huomattavasti suurempi yhtäaikaisten käyttäjien kunta kuin on alun perin odotettu, mutta useimpiin tarpeisiin on odotettu tietokanta pohjaisen, melko yksinkertaisen sessiohallinnan ajavan asiansa.

Session hallinnassa oleellisimpia tietoja ovat kirjautumisen yhteydessä satunnaisesti luotu merkkijono, joka lähetetään käyttäjälle kirjautumisen yhteydessä ja jota käyttäjä tämän jälkeen käyttää uusien kyselyjen tekemiseen. Esimerkiksi tietojen lukeminen tai päivittäminen vaatii kyseisen merkkijonon lähettämistä pyynnön otsake-tiedoissa.

Lisäksi käyttäjän kirjautuessa palveluun tallennetaan aikaleima kirjautumisen hetkeltä, jota vertaamalla palvelimeen aikaan voidaan esimerkiksi päätellä, onko sessio vanhentunut. Oletuksena sessiot ovat voimassa tunnin kirjautumisesta. Pienenä erikoisuutena on myös ”Lifespan”-tieto, jota säätämällä voidaan muuttaa session maksimi pituutta. Oletuksena tämä arvo on kuitenkin 3600 sekuntia, eli tunti.

5.5 Opinnäytetöiden tiedot

Opinnäytetöiden osalta tietokannan skeema on hieman monimutkaisempi kuin käyttäjien osalta, mutta toisaalta myös hyvin yksinkertainen. Jokaiselle opinnäytetyölle asetetaan automaattisesti oma uniikki avaintieto sekä paljon yleistä opinnäytetyöhön liittyvää tietoa. Kuviossa 8 on esimerkki opinnäytetyön tietojen perusnäkymästä.



KUVIO 8 Esimerkki opinnäytetyön perustiedoista

Kaikki kuviossa 8 esitellyt kentät (jotka saattavat muuttua tulevaisuudessa) tallennetaan yhteen tietokannan tauluun (esillä kuviossa 7). Tässä tapauksessa huomattavaa on, että ohjaaja tietoina tallennetaan kyseisten henkilöiden yksilöllinen tunniste tekstin sijaan, joka mahdollistaa esimerkiksi nimen muuttamisen siten, että muuttunut nimi voidaan esittää kaikkialla järjestelmässä oikein. Henkilöiden nimet ohjaaja-valintoihin haetaan erillisellä rajapintakyselyllä.

Ohjaajatiedot ovat opinnäytetyön seurannan kannalta kriittisiä. Koska opinnäytetyön tietoihin pääsee käsiksi ainoastaan opinnäytetyön tekijä, ohjaajat tai järjestelmää hallitseva taho, täytyy jälkimmäisen asettaa työlle ohjaajat ennen kuin tietoja voidaan järjestelmään kirjata (opiskelijalla itsellään ei ole oikeutta muuttaa opinnäytetyön tietoja, eikä ohjaajat voi nähdä opinnäytetyötä ilman oikeuksia). Tästä on myös suoraan johdettu järjestelmän pääasiallinen oikeuksien tarkistus tietojen muokkaamista varten: koska koko palvelun tarkoitus on ylläpitää opinnäytetöiden seurantatietoja, voidaan opinnäytetöiden kirjaamiseen käytettäviä relaatioita hyödyntää käyttäjäoikeuksien tarkistamisessa. Näin siis esimerkiksi ohjaajatietoihin asetetut arvot suoraan kertovat, kuka on työn ohjaaja ja että tällä ohjaajalla on oikeus avata työ. Poikkeus sääntöön on käyttäjät, joille on asetettu järjestelmänvalvojan oikeudet.

Opinnäytetöihin liittyy myös uusien tapaamisien merkkaamiseen luotu taulunäkymä, josta voi tarkistaa milloin seuraava opinnäytetyö projektia koskeva palaveri on sovittu pidettäväksi. Nämä tiedot sijaitsevat omassa relaatiossaan, ja tiedot viitataan tiettyyn opinnäytetyöhön opinnäytetyön tunnisteen perusteella. Opinnäytetyöllä voi olla useampia tapaamisia, mutta tapaamiset voi sopia vain yhdelle opinnäytetyölle kerrallaan. Tästä muodostuu siis 1:N (opinnäytetyöt - tapahtumat) suhde relaatioiden välille.

Loput oleelliset opinnäytetöihin liittyvät entiteetit toimivat jotakuinkin samoin kuin tapahtumat. Sekä kommentit ja tiedostot sisältävät taulut ovat 1:N-suhteita opinnäytetöiden näkökulmasta.

5.6 Riippumattomat relaatiot

Tiedostojen lataamiseen käytettävä Downloads-relaatio tarvitsee erityismaininnan. Kyseisellä relaatiolla ei ole mitään todellisia riippuvuuksia muihin relaatioihin, vaan

relaatio sisältää avaimen (hash-koodi), jota vasten käyttäjä saa ladata tiedoston. Jokainen avain on kertakäyttöinen, joten latauksen alkaessa avain merkitään käytetyksi. Lisäksi rajapintatoteutuksessa avaimet vanhenevat tunnin sisään, joten avaintietueille talletetaan myös luontiaika, jota vasten palvelimen aikaa verrataan.

5.7 PostgreSQL:n käyttäjätunnukset

Projektissa voidaan sikäli käyttää melko yksinkertaista PostgreSQL-asennusta, sillä oletuksena Postgres näkyy vain muille samalla koneella toimiville sovelluksille – yhteyksien hyväksymistä ulkoverkosta ei siis oletuksena ole.

Postgresin käyttöä varten luodaan uusi Linux-käyttäjä. Tämä siksi, että oletuksena Postgres käyttää Linuxin käyttäjätietoja tietokantojen oikeuksien ylläpitämiseen. Tämä oikeusasetus myös antaa suoraan oikeudet tietyn nimiseltä käyttäjältä samannimiseen tietokantaan (huom.! tietokanta ei ole sama asia kuin tietokantapalvelin/hallintajärjestelmä) siten, että esimerkiksi käyttäjällä ”thesis” on suoraan oikeudet käyttää ja tehdä muutoksia tietokantaan nimeltä ”thesis”. Linuxissa root-käyttäjällä on automaattisesti oikeudet kaikkiin tietokantoihin sekä oikeus tehdä niihin myös muutoksia.

Luodaan ensin uusi Linux-käyttäjä seuraavalla komennolla:

useradd thesis

Käyttäjälle tulee myös luomisen jälkeen asettaa salasana seuraavalla komennolla, koska muutoin käyttäjätunnus on olemassa vain lukitussa tilassa eikä sille voi kirjautua:

passwd thesis

Josta kuviossa 9 on esimerkki.

```
kiikkaja@UbuntuLTS:~$ su passwd thesis
No passwd entry for user 'passwd'
kiikkaja@UbuntuLTS:~$ sudo passwd thesis
Enter new UNIX password:
```

KUVIO 9 Salasanan asettaminen käyttäjälle

Kun käyttäjä on luotu, voidaan siirtyä Postgresin asentamiseen.

5.8 PostgreSQL:n asentaminen

PotgreSQL:n asentaminen on helppoa Ubuntun mukana tulevalla paketinhallinnalla. Komennolla ”*sudo apt-get install postgresql”* saadaan uusin, vakaa versio PostgreSQL:sta asennettua kuvion 10 mukaisesti.

```
kiikkaja@UbuntuLTS:~$ sudo apt-get install postgresql
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libpq5 libsensors4 postgresql-9.5 postgresql-client-9.5
  postgresql-client-common postgresql-common postgresql-contrib-9.5 ssl-cert
  sysstat
Suggested packages:
  lm-sensors postgresql-doc locales-all postgresql-doc-9.5 libdbd-pg-perl
  openssl-blacklist isag
The following NEW packages will be installed:
  libpq5 libsensors4 postgresql postgresql-9.5 postgresql-client-9.5
  postgresql-client-common postgresql-common postgresql-contrib-9.5 ssl-cert
  sysstat
0 upgraded, 10 newly installed, 0 to remove and 46 not upgraded.
Need to get 4,861 kB of archives.
After this operation, 19.6 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

KUVIO 10 PostgreSQL:n asentaminen

Asennuksen jälkeen PostgreSQL:aan tulee kirjautua ”postgres”-käyttäjän tunnuksilla. Muussa tapauksessa Postgres palauttaa kuvion 11 esittämän virheen. Kyseinen käyttäjätunnus luodaan asennuksen yhteydessä.

```
kiikkaja@UbuntuLTS:~$ psql
psql: FATAL:  role "kiikkaja" does not exist
```

KUVIO 11 PostgreSQL virhe, "Roolia ei ole olemassa"

Postgres käyttäjälle voidaan vaihtaa komennolla ”sudo -i -u postgres”. Koska ”postgres” käyttäjällä on kaikki oikeudet tietokantapalvelimelle, on järkevää rajoittaa oikeuksia luomalla uusi rooli käyttäjälle ”thesis” (joka on luotu aikaisemmin), asettaa tälle mahdollisimman oikeuksia tietokantapalvelimelle kuvion 12 mukaisesti.

```
postgres@UbuntuLTS:~$ createuser --interactive
Enter name of role to add: thesis
Shall the new role be a superuser? (y/n) n
Shall the new role be allowed to create databases? (y/n) n
Shall the new role be allowed to create more new roles? (y/n) n
postgres@UbuntuLTS:~$
```

KUVIO 12 Uuden käyttäjäroolin lisääminen PostgreSQL:ssa

Kyseiselle käyttäjälle tulee myös luoda tietokanta. Oletuksena käyttäjillä on oikeus vain oman nimiseen tietokantaa. Uusi tietokanta luodaan tässä tapauksessa komen-

nolla ”createdb thesis”. Tämän jälkeen voidaan käyttäjällä kirjautua tietokantapalvelimelle. Kuviosta 13, jossa esitellään tietokantapalvelimelle kirjautumista, huomataan myös, ettei käyttäjällä vielä näy yhtään relaatiota.

```
postgres@UbuntuLTS:~$ createdb thesis
postgres@UbuntuLTS:~$ su thesis
Password:
thesis@UbuntuLTS:/var/lib/postgresql$ psql
psql (9.5.5)
Type "help" for help.

thesis=> \dt
No relations found.
thesis=>
```

KUVIO 13 Lisätyllä käyttäjällä/roolilla kirjautuminen tietokantapalvelimelle

Tietokannan rakenteet (relaatiot) voidaan lisätä kopioimalla liite 1:n skripti tähän näkymään. Tämän jälkeen voidaan ”*\dt*” komennolla nähdä juuri luodut relaatiot (kuvio 14).

```
thesis=> \dt
           List of relations
 Schema |     Name      | Type  | Owner
--------+---------------+-------+--------
 public | Comments      | table | thesis
 public | Credentials   | table | thesis
 public | Documents     | table | thesis
 public | Downloads     | table | thesis
 public | Events        | table | thesis
 public | Theses        | table | thesis
 public | Users         | table | thesis
(7 rows)
```

KUVIO 14 Relaatiot tietokannassa

Kun tietokanta ja relaatiot on luotu, voidaan PostgreSQL:aan yhdistää paikallisesti (localhost) käyttämällä porttia 5432. Oletuksena Postgres kuuntelee vain samasta verkosta tulevia viestejä, eli ulkoverkosta palvelimelle ei voi ottaa yhteyttä.

6 Front-end, käyttöliittymä

Käyttöliittymä on Angular 2-kehyksen päälle rakennettu Single-Page Application (SPA), joka tarjoaa todella paljon valmista toiminnallisuutta modernin JavaScript-pohjaisen front-end käyttöliittymän luomiseksi. Esimerkiksi hyvin yleinen tarve on hakea tietoa rajapinnan kautta (kuten tässäkin projektissa), ja XHR-pyynnöt (XMLHttpRequest) ovat melko työläitä kirjoittaa aina uudestaan eri käyttötarkoituksia varten. Tällaisiin tapauksiin Angular (ja sen hyvin tukema TypeScript-kieli) tarjoaa valmiiksi suhteellisen helposti ymmärrettävät ja käytettävät luokat ja metodit.

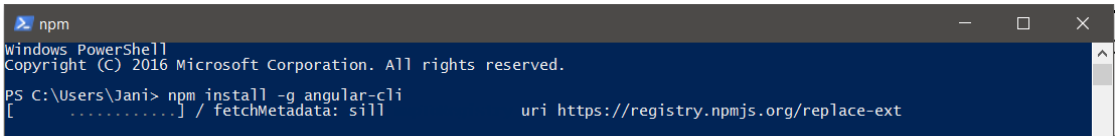
Angular 2:n käyttämisen aloittaminen onnistuu helpoiten Angular CLI:tä käyttämällä. Angular CLI vaatii Node.js:n, joten Node.js täytyy kehitysympäristöstä löytyä. Node.JS:aa tai Angular CLI:tä ei tosin tarvita enää palvelimella, vaan tuotanto valmis Angular-sovellus toimii hyvin tavallisen html-tiedostoja tarjoavan palvelimen päällä.

Vaikka Node.JS on sikäli läsnä kaikissa front-end kehityksen vaiheissa, ei sitä tarvitse suoraan käyttää missään kehityksen vaiheessa. Node.JS:lla voi toki tarvittaessa toteuttaa vaikkapa koko palvelimen. Tällaisessa front-end kehityksessä se kuitenkin toimii vain aputyökalujen ajamiseen tarkoitettuna alustana (vrt. esimerkiksi .NET sekä JRE, eli Java Runtime Environment ympäristöihin), ja esimerkiksi Angular CLI yksinkertaistaa Node.JS:n käyttämisen Angular 2:n osalta vain muutamaan komentoon.

Node.JS:n voi asentaa tietokoneelleen nodejs.org sivustolta. Tämän toteutuksen kehittämisessä on käytetty kirjoittamisen hetkellä uusinta v6.9.4 LTS (Long Term Support) versiota.

6.1 Angular CLI:n asentaminen ja käyttö

Angular CLI:n saa helpoimmin asennettua Node.JS:n pakettien hallinnalla, NPM:lla. NPM:lla, kuten pakettienhallinta sovelluksilla/palveluilla yleensä, voidaan asentaa projektien vaadittavat riippuvuudet keskitetystä verkkovarastosta. Asentaminen tapahtuu komennolla (Windows ympäristöissä siis käytetään joko komentokehoitetta tai PowerShellia) kuvion 15 esittämällä tavalla:



KUVIO 15 Angular CLI:n asentaminen NPM:lla

Komento *"npm install -g angular-cli"* siis asentaa Angular CLI:n käyttäen Node.JS:n pakettienhallintaa. Paketin nimen edessä oleva "-g"-vipu tarkoittaa globaalia Node.JS:n varastoa, jonne useimmiten asennetaan työkalut kuten Angular CLI. (Installing npm packages globally, 2017)

Angular CLI:n asennettua uusi projekti kirjoittamalla komento "*ng new sovelluksennimi*". Kirjaimet "ng" ovat lyhenne Angularista, ja "new sovelluksennimi" tarkoittaa uutta sovellusta (projektia), jolle annetaan haluttu nimi.

Kun uusi projekti on luotu, avataan Angular CLI:n luoma projektikansio halutussa tekstieditorissa. Sovellusta voi testata jo tässä vaiheessa komennolla (ensin navigoimalla sovelluksen kansioon) "ng serve", mutta yhtään muokkaamattomana saadaan lähinnä "se toimii!" tasoinen viesti selaimen näkymään. Komento siis käynnistää lokaalin testipalvelimen, joka pyörii Node.JS:n päällä testaamista varten.

Kun projekti on jossain vaiheessa valmis julkaistavaksi, voidaan paketti kasata "ng build -prod" komennolla. Tämä komento toimii olennaisesti eri tavalla "ng serve" komennosta siinä, että se tekee mm. "TreeShake"-operaation projektin tiedostoille (ns. "pudottaa" turhat koodit pois valmiista paketista, joihin ei projektissa viitata koskaan) sekä pakkaa koodin mahdollisimman pieneksi paketiksi mm. vaihtamalla muuttujien nimet mahdollisimman lyhyiksi ja poistamalla ylimääräisiä välilyöntejä/kommentteja/merkkejä.

6.2 TypeScriptin asentaminen ja käyttö

TypeScript asennetaan samalla tavalla npm:lla kuin Angular CLI: komennolla ”*npm install -g typescript*” npm asentaa TypeScriptin tarvitsemat työkalut Nodella käytettäväksi kaikkialla järjestelmässä.

TypeScript on siis kieli, joka käännetään puhtaaksi JavaScriptiksi käyttöä varten. TypeScript tiedostot tunnistaa päätteestä ”.ts”, ja kääntötyökaluja käyttämällä Node.JS luo tiedostosta ”.js” vastaavan. Kuviossa 16 nähdään esimerkki TypeScript koodista, ja kuviossa 17 vastaavasti esimerkki käännetystä JavaScript koodista:

```
1  class Demo {
2      private number: number = 10;
3      IncreaseNumberByOne() {
4          this.number = this.number + 1;
5      }
6      GetNumber() {
7          return this.number;
8      }
9  }
```

KUVIO 16 TypeScript-koodin esittelyä

```
1  var Demo = (function () {
2      function Demo() {
3          this.number = 10;
4      }
5      Demo.prototype.IncreaseNumberByOne = function () {
6          this.number = this.number + 1;
7      };
8      Demo.prototype.GetNumber = function () {
9          return this.number;
10     };
11     return Demo;
12 }());
```

KUVIO 17 Käännetyn JavaScript koodin esittely

Kuten esimerkeistä on nähtävillä, perinteisempään olio-ohjelmointiin tottuneet ohjelmoijat todennäköisesti pitävät TypeScriptin tavasta esittää mm. luokkia (JavaScript on todellisuudessa kieli jossa ei ole luokkia, mutta ominaisuudet, kuten periminen, ovat käytännössä samanlaiset muiden olio-ohjelmointi kielien kanssa).

TypeScriptin yksi tärkeimmistä erikoisuuksista tavallisen JavaScriptin yli on kuitenkin kielen nimenkin vihjaama tyyppi tarkistus. Koska JavaScript on hyvin dynaaminen kieli, jokainen muuttuja on myös dynaaminen olio. Olion arvoksi voi asettaa käytännössä mitä tahansa luvusta merkkijonoon lennosta. Jopa perinteinen ”==” operaattori (yhtä kuin) tarkoittaa JavaScriptissa jotain ”ehkä” ja ”sinnepäin” väliltä. Sen sijaan JavaScriptillä tulee käyttää kolmea yhtä kuin merkkiä tyyppi turvallisiin arvotarkistuksiin.

TypeScript sen sijaan antaa ohjelmoijan määritellä arvolle tyypin, ja kuten kuviosta 18 on nähtävillä, muuttujan ”number” tyyppi lukuarvo. Näin ollen, jos kuvion 18 esittämässä metodissa nimeltä ”IncreaseNumberByOne” yritetäänkin summata lukuarvoon merkkijono kuvion 18 esittämällä tavalla, saadaan aikaan virhe kääntäjässä:

```
class Demo {
    private number: number = 10;
    IncreaseNumberByOne() {
        this.number = this.number + "one";
    }
```

KUVIO 18 TypeScript tyyppi tarkistusten esittely koodina

Kuviossa 18 näkyvä ”punakynä” muuttujan ja luokan nimen alla on myös tyyppitarkistukseen liittyvä ominaisuus, mutta tässä tapauksessa kyseessä on editorin (Visual Studio Code) lisäosan ominaisuus. Sen sijaan kääntäjän antama virhe on nähtävillä kuviossa 19:

```
PS C:\Users\Jani\Documents> tsc .\demo.ts
demo.ts(4,9): error TS2322: Type 'string' is not assignable to type 'number'.
```

KUVIO 19 TypeScript-kääntäjän virheilmoitus

Tällaisen tyyppi tarkistuksen hyödyntäminen erityisesti tiimissä toimiessa on hyödyllistä: voidaan olla varmoja, että metodien parametreina syötetään oikeita tyyppejä.

TypeScript-kääntäjää voi käyttää komennolla ”*tsc [tiedostonnimi]*”, mutta Angular CLI hoitaa kokonaisen projektin kääntämisen kerralla komennoilla ”*ng serve”* sekä ”*ng build”*.

6.3 Angular 2:lla kehittäminen

Angular 2 sovellukset ovat käytännössä kasa yhteen liitettyjä komponentteja. Komponentit koostuvat HTML-mallista sekä sitä ohjaavasta JavaScript-luokasta (joka tässä tapauksessa käännetään TypeScript-kielestä). Ohjelma voisi esimerkiksi sisältää navigointi komponentin, joissa on linkit tärkeimmille sivuille/toiminnoille, sekä sisältö komponentin, jossa sivun varsinainen leipäteksti sijaitsisi. Kummallakin komponentilla olisi oma rajattu tehtävänsä sivustolla, eivätkä ne välttämättä ole suoraan sidoksissa toisiinsa.

Opinnäytetöiden seurantajärjestelmän tapauksessa on käytetty Angular CLI:n hieman ”mielipiteistettyä” ohjelmarakennetta (eli Angular CLI:n kehittäjät ovat päättäneet, että tämä toimii ja näin suositellaan muidenkin tekevän), joskin tästä on hieman poikettu hyvien toimintatapojen löytämisen vuoksi. Lopulta selkeimmin toimiva projektirakenne on ollut rakentaa jokainen komponentti sekä palvelu omaan hakemistoonsa projektihakemiston alle, jos niitä ei odoteta käytettävän muualla ohjelmassa. Koko projekti ei ole näin toteutettu, vaan jäänteitä ensimmäisistä testeistä saattaa löytyä, kuten esimerkiksi keskitetty palvelu hakemisto. Keskitetty palveluhakemisto olisi muuten erittäin hyvä (ja oikea) ratkaisu, mutta projektin palveluita ei ole suunniteltu käytettäväksi mistä tahansa komponentista vaan ovat hyvin yksilöllisiä tietyille komponenteille.

Angular 2-komponentti voisi näyttää kuvion 20 kaltaiselta:

```
1   import { Component } from '@angular/core';
2
3   @Component({
4     selector: 'view-about',
5     templateUrl: './about.component.html'
6   })
7
8   export class AboutComponent {
9       title = "About";
10  }
```

KUVIO 20 Angular 2:n komponentti luokan esimerkki

Kuvion 20 luokkatiedosto on vielä todella yksinkertainen: se asettaa vain yhden muuttujan arvon (”title”) jota voidaan käyttää HTML-mallissa kuvion 21 esittämällä

tavalla (aaltosulkujen sisällä oleva muuttuja esittää komponentissa määritellyn muuttujan arvon, tässä tapauksessa näyttäen sivuilla ”About” tekstin H1 tagien sisällä). HTML-mallin tiedosto määritellään ”@Component” attribuutti direktiivillä (näiden tarkoitus on määritellä DOM:n käyttäytymistä) ”templateUrl”-ominaisuuden arvona. Koko mallin voisi kirjoittaa ”template”-ominaisuuden sisälle, mutta selkeyden vuoksi on hyvä käyttää erillistä tiedostoa mallille.

```
<h1>{{title}}</h1>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Repellat repudiandae
voluptates soluta ut perspiciatis saepe consectetur odit vitae, officia pariatur
voluptatibus ex inventore error ullam id, tenetur iure suscipit! Dolore alias par

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Blanditiis, error.</
```

KUVIO 21 Ominaisuusarvojen kiinnittäminen HTML-malliin.

Angular 2:n HTML-malleissa käytetään tavallisen HTML-syntaksin lisäksi Angularin omia HTML:n kaltaisia elementtejä. Komponentin voi esimerkiksi sijoittaa toisen komponentin sisälle käyttämällä kirjoittamalla komponentille asetetun ”selector”-ominaisuuden (joka siis asetetaan attribuutti direktiivien sisällä) arvon HTML elementin arvoksi.

Kuten jo mainittua, komponentteja voidaan upottaa HTML-mallien sisään omiksi osioikseen. Mutta Angular 2:lle tarvitsee samaan tapaan myös kertoa, minne ylipäätään sijoittaa itse ohjelma. Koska Angular 2 sovellukset ovat usein SPA-sivustoja (Single-page application, eli yhden sivun sovellus) tarvitaan vain yksi HTML tiedosto, jonka tehtävänä on toimia runkona muulle sovellukselle. Kyseiseen tiedostoon on tämän projektin osalta määritelty sivuston kaksi muuttumatonta osaa sekä itse dynaaminen sovellus. Nämä muuttumattomat osat ovat yläreunaan sijoitettu navigointi sekä sivun alareunassa oleva osio kopiointioikeuksille ja sivuston omistajien yhteystiedoille. Muuttumattomuudella tarkoitetaan tässä tapauksessa sitä, ettei kyseisten osioiden sijainti sivustolla muutu koskaan, vaan nämä näkyvät kaikkialla sivustolla selaillessa. Osioiden sisältö saattaa sen sijaan olla dynaamista, kuten esimerkiksi navigointiosion kirjautumistiedot sekä linkit, jotka näkyvät käyttäjän profiilista riippuen. Kuviossa 22 on esimerkki ohjelman runkotiedostosta (index.html), johon on myös määritelty sivustolla käytettävät muut skriptit sekä tyylitiedostot. Komponentit on kuviossa sijoitettu ”body”-tagien sisälle.

```html
<!doctype html>
<html>
  <head>
    <base href="/">
    <meta charset="utf-8">
    <title>Opinnäytetöiden seurantajärjestelmä</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="styles.css">
    <link rel="stylesheet" href="assets/css/grids-min.css">
    <link rel="stylesheet" href="assets/css/buttons-min.css">
    <link rel="stylesheet" href="assets/css/tables-min.css">
    <link rel="stylesheet" href="assets/css/forms-min.css">
    <link href="https://fonts.googleapis.com/css?family=Alegreya+Sans:400,500" rel="stylesheet">
    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.6.3/css/font-awesome.min.css" rel="stylesheet">
  </head>

  <body>
    <application-header></application-header>
    <application></application>
    <application-footer></application-footer>
  </body>
</html>
```

KUVIO 22 Seurantajärjestelmän runkosivu (index.html)

6.4 Angular 2 sovelluksen siirtäminen tuotantoon

Kun Angular 2 sovellus on valmis tuotantoon vietäväksi, on helpointa käyttää Angular CLI:n ”ng build -prod”-komentoa. Tämä toiminto mm. ”minifioi” sovelluksen JavaScript-koodit viemään mahdollisimman vähän tilaa, sekä pudottaa pois Angularin toimintoja joita ei käytetä.

Kuviossa 23 on esimerkki Angular 2 projektin koosta tilanteessa, jossa se on valmis tuotantoon vietäväksi. Huomion arvoista on, että pelkästään Angular 2:n oma kehityskirjasto vie yksinään yli 12 megatavua tilaa levyltä. Assets-kansio sisältää kaikki lisätyt muut resurssit, kuten kuvat ja ylimääräiset tyylitiedostot/fontit.

assets	17.1.2017 23.42	Tiedostokansio	
favicon	17.1.2017 23.42	Kuvake	6 kt
index	17.1.2017 23.42	Chrome HTML Do...	2 kt
inline.d41d8cd98f00b204e980.bundle.map	17.1.2017 23.42	MAP-tiedosto	14 kt
inline	17.1.2017 23.42	JavaScript Source ...	2 kt
main.7a8c270713e185911160.bundle	17.1.2017 23.42	JavaScript Source ...	928 kt
main.7a8c270713e185911160.bundle.js.gz	17.1.2017 23.42	GZ-tiedosto	210 kt
main.7a8c270713e185911160.bundle.map	17.1.2017 23.42	MAP-tiedosto	6 506 kt
styles.eaab464eda974aa2d537.bundle	17.1.2017 23.42	JavaScript Source ...	5 kt
styles.eaab464eda974aa2d537.bundle.m...	17.1.2017 23.42	MAP-tiedosto	30 kt

KUVIO 23 Tuotantoon vietävä Angular 2-paketti

Valmiit tiedostot voi käytännössä siirtää sellaisenaan web-palvelimelle, kunhan palvelin on konfiguroitu tarjoamaan tavallisia HTML-sivuja.

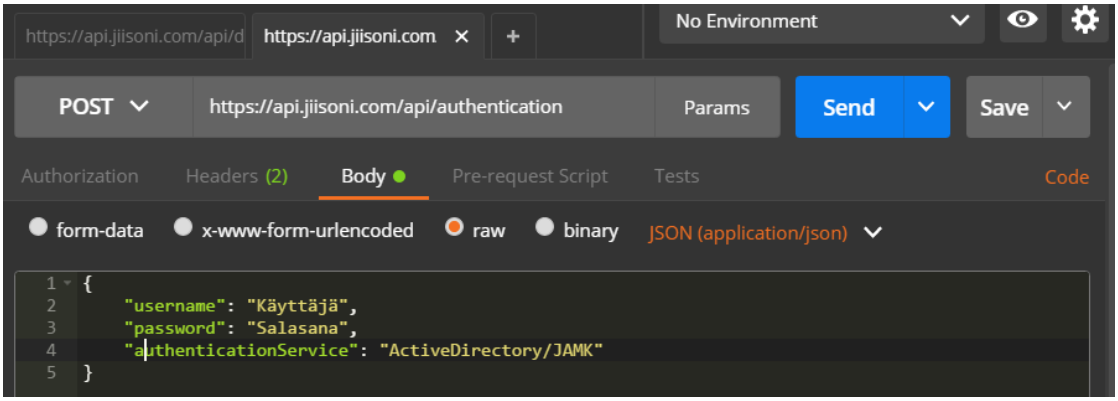
7 Backend

Backend palvelin on toteutettu Microsoftin .NET Core teknologian päälle. Sen kehittämiseen on pääasiassa käytetty Microsoftin tuotteita, kuten Visual Studiota sekä siihen liitettäviä työkaluja. Ohjelman koodi käännetään tavukoodiksi, jota .NET:n ajonaikainen ympäristö tulkkaa konekieleksi. Tämän osion tehtävänä on kuvata ohjelman rakennetta ja teknisiä ratkaisuja sekä käytettyjä työkaluja ja ulkopuolisia kirjastoja.

7.1.1 Käyttäjän todentaminen

Koska palvelun kaikilta käyttäjiltä vaaditaan oletuksena käyttäjätunnukset palveluun, ei anonyymeille käyttäjille tarjota kuin palvelun etusivu, jossa on kirjautumiseen vaadittava lomake.

Käyttäjän kirjautuessa palveluun omilla tunnuksillaan (esim. AD-tunnuksilla), rajapintaan lähetetään POST-sanoma */api/authentication*-endpointille, joka sisältää käyttäjätunnuksen, salasanan sekä kirjautumispalvelun. Kuviossa 24 on esimerkki mahdollisesta viestistä.



KUVIO 24 Malli kirjautumisen POST-viestistä

Kirjautumiseen käytettävä endpoint (endpoint on käytännössä resurssi, josta vastaa rajapinnan ohjaimen metodi) palauttaa tämän jälkeen paluusanomassa uuden hashtiivisteen kuvion 25 mukaisesti (kuviossa on käytetty oikeita tunnuksia sekä autentikointpalveluna oman tietokannan vastaista todentamista).

Body Cookies Headers (8) Tests Status: 200 OK Time: 82 ms

Pretty Raw Preview JSON

```
1 {
2   "userId": 1,
3   "username": "Admin",
4   "token": "1b053514d3682258396aaf15e660889ea482a1d8e5bcd07d1643b45fd78a9f8a",
5   "role": "Admin"
6 }
```

KUVIO 25 Onnistuneen kirjautumisen paluusanoma

Paluusanoman mukana tulevaa tokenia asiakas, eli tässä tapauksessa frontend, käyttää kyseisen käyttäjän muiden toimintojen tekemiseen. Kyseinen merkkijono lähetetään jokaisen kyselyn yhteydessä ”Authorization”-otsakkeessa (kyseessä on siis http-protokollan ”header” tiedot).

Ennen paluusanoman lähetystä palvelu kirjoittaa tokenin tietokantaan kyseisen käyttäjän tunnistetta vastaavalle riville. Koska palvelun käyttäjänhallinnan kirjautumistiedot ovat yksilöllisiä, uudelleen kirjautuminen kirjoittaisi vanhan tokenin ja aikaleiman päälle. Kuviossa 26 on esitys tietokannan kahden olemassa olevan käyttäjän tokeneista:

```
thesis=> select "UserId", "Token", "Timestamp" from "Credentials";
 UserId |                              Token                              |            Timestamp
--------+-----------------------------------------------------------------+-----------------------------
      2 | 39bd23f63b340f6e6a7452b677aa38be114d7f26b42da1ab46182388219e3406 | 2016-11-10 00:06:37.894797+00
      1 | 1b053514d3682258396aaf15e660889ea482a1d8e5bcd07d1643b45fd78a9f8a | 2017-02-01 19:53:37.539252+00
(2 rows)
```

KUVIO 26 Tokenit ja niiden aikaleimat tietokannassa

Kuviossa 27 vastaavasti on tokenin luomiseen käytetty *System.Security.Cryptography*-kirjaston SHA-256 funktio, jolle annetaan käyttäjän yksilöiviä arvoja (palvelu hakee tietokannasta käyttäjän tiedot, jos kirjautumisen yhteydessä annetut tiedot ovat oikeat) sekä kellonaika, jolla saadaan hieman satunnaisuutta generoitaviin tiivisteisiin. Lopuksi tiiviste asetetaan käyttäjän tietoja esittävään olioon ja tallennetaan tietokantaan, jonka jälkeen paluusanoma muodostetaan ja lähetetään asiakkaalle.

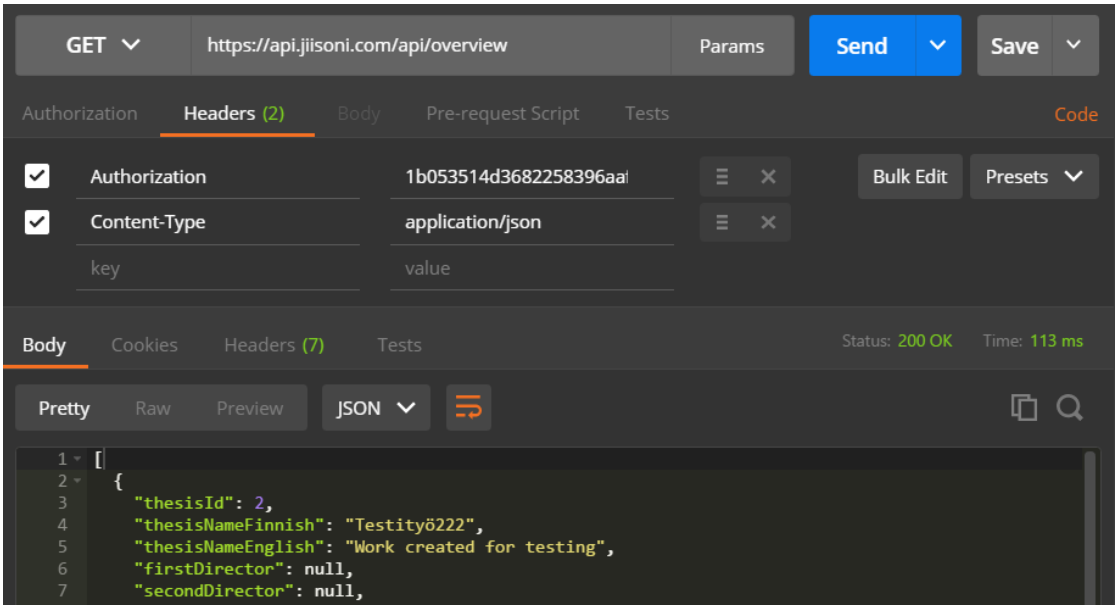
```
var currentTime = DateTime.UtcNow;

// Create token
using (var sha256 = SHA256.Create())
{
    var hashedBytes = sha256.ComputeHash(Encoding.UTF8
        .GetBytes(getUser.Salt + getUser.UserId + currentTime.ToString(CultureInfo.InvariantCulture) + getUser.Username));
    var hash = BitConverter.ToString(hashedBytes).Replace("-", "").ToLower();
    getUser.Token = hash;
}
```

KUVIO 27 Hash-tiivisteen luominen

7.1.2 Käyttäjän tunnistaminen

Kun käyttäjä on todennettu ja käyttäjän asiakasohjelmalle on annettu käyttäjän tunnistamiseen käytettävä merkkijono, asiakas käyttää tätä merkkijonoa jokaisen uuden viestin lähettämiseen. Kuviossa 28 on esimerkki otsakkeen käyttämisestä kyselyiden yhteydessä GET: */api/overview*-endpointtiin, joka palauttaa listan kaikista kyseisellä käyttäjälle näytettävistä opinnäytetöistä.



KUVIO 28 Esimerkki “Authorization”-otsakkeen käyttämisestä

Koska tunnistautumiseen käytettävää tiivistettä käytetään käytännössä jokaisessa rajapinnan ohjaimessa, on tunnistautuminen toteutettu kyselyiden käsittelyn ”putkessa” (pipeline). Sen sijaan että tehtäisiin erillinen luokka, jota kutsuttaisiin jokaisesta ohjaimesta erikseen, on sen sijaan toteutettu erillinen ”väliohjain” (middleware) jonka läpi jokainen rajapintaan tuleva kutsu kulkee. Jos kyseessä on viesti käyttäjän todentamiseen käytettävään endpointtiin, viestin annetaan mennä läpi ilman tunnistamista, muissa tapauksissa käyttäjä yritetään ensin tunnistaa ennen kuin viesti ohjataan halutulle ohjaimelle. Jos käyttäjä on todennettu aikaisemmin ja sessio on olemassa, viesti ohjataan ohjaimelle ja viestin mukana kuljetetaan myös käyttäjän tiedot, jotka on tunnistamisen yhteydessä haettu tietokannasta. Näin ohjaimella ei enää erikseen tarvitse selvittää esimerkiksi käyttäjänimeä tai käyttäjän oikeustasoa. (Middleware, 2016).

Kuviossa 29 on määritelty sovelluksen kyselyputki, jonka lävitse jokainen viesti palvelussa kulkee.

```
// Pipeline configuration
public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory loggerFactory)
{
    app.UseCors("AllowSpecificOrigin");
    app.UseMiddleware<AuthorizationMiddleware>();
    loggerFactory.AddConsole(Configuration.GetSection("Logging"));
    loggerFactory.AddDebug();
    app.UseMvc();
}
```

KUVIO 29 Sovelluksen putki

Jokainen väliohjain (middleware) delegoi viestin seuraavalle ohjaimelle tai vaihtoehtoisesti pysäyttää viestin ja käsittelee sen itse. Jokainen ohjain saa tilaisuuden käsitellä viestiä kahdesti (ohjain 1 -> ohjain 2 -> ohjain 3 -> ohjain 2 -> ohjain 1), joten viesti joka käsitellään jossain väliohjaimessa käytännössä tarkoittaa putken oikosulkua. Esimerkiksi käyttäjän tunnistamiseen käytettävä väliohjain saattaisi todeta, että pyynnön tekijä ei ole kirjautunut käyttäjä ja päättää palauttaa virheilmoituksen sen sijaan, että ohjain siirtäisi kyselyn seuraavalle ohjaimelle.

7.1.3 Ohjaimien toiminta yleisesti

Kaikki loppuohjaimet (jotka siis ohjaavat resursseja joita pääasiassa halutaan käyttää palvelun toimintoja varten) toimivat kaikki pääasiassa samalla tavalla. Ohjaimet pääasiassa sisältävät GET-metodin, joka palauttaa asiakkaalle tämän pyytämän resurssin (esimerkiksi listan opinnäytetöistä tai käyttäjistä). Monet ohjaimista myös omaavat PATCH-, ja POST metodit, jotka vastaavasti päivittävät sekä luovat uusia resursseja.

Ohjaimien metodien käyttöä kontrolloidaan (tässä toteutuksessa) meta attribuuteilla. Tässä tapauksessa käytetyt http-verbi attribuutit kertovat sovelluskehykselle, että tästä ohjaimesta löytyy esimerkiksi GET-metodi. Meta attribuutteja määritellään kuvion 30 kuvaamalla tavalla.

```
[HttpGet]
public ActionResult Get()
{
    var claims = (ClaimsDto) _httpContext.Items["Claims"];

    if (claims.AccessLevel != (int) Role.Director
        && claims.AccessLevel != (int) Role.Manager
        && claims.AccessLevel != (int) Role.Admin)
        return Unauthorized();
```

KUVIO 30 Esimerkki attribuuttien käytöstä

Attribuutteja ei ole välttämätöntä käyttää, vaan ne voidaan korvata myös ohjainkonfiguraatioilla.

Käytännössä tämä tarkoittaa sitä, että jos ohjaimella ”overview” on metodi, jolla on attribuutti ”*HttpGet*”, voidaan sille lähettää kuvion 22 esittämä pyyntö. Attribuutilla voidaan myös määritellä tarkempia polkuja, ja näitä käytetäänkin paljon resurssien kohdentamiseen. Attribuutilla ”*[HttpGet(”{id}”)]”* luetaan osoitteen loppuun lisättävä arvo muuttujana metodille. Tällöin tosin täytyy myös metodin ottaa vastaan parametri ”id” halutulla tietotyypillä.

Seuraava oleellinen osa ohjaimien yleistä logiikkaa on käyttäjän tietojen lukeminen pyynnön kontekstista (katso kuvio 23), jotka tunnistautumiseen tarkoitettu väliohjain lisää pyynnön olioon (jokainen pyyntö käsitellään palvelussa omana tyyppi instanssinaan). Lähes kaikissa ohjaimissa nämä tiedot luetaan ”claims”-oliolle, josta niitä on sen jälkeen helppo lukea. ”Claims” nimitys tulee ajatuksesta, jossa pyynnön tekijän ominaisuuksien perusteella tehdään päätös, onko tällä oikeutta johonkin tiettyyn toimintoon. Yksinkertaistettuna, jos käyttäjä olisi esimerkiksi jonkin tietyn opinnäytetyön tekijä, olisi käyttäjällä oikeus avata kyseisen opinnäytetyön tiedot. Tällaisessa tapauksessa tarkastettaisiin opinnäytetyöltä tämän tekijä, joita sitten verrattaisiin ”väittävän osapuolen” ominaisuuksiin.

Tämän lisäksi voidaan tietysti toteuttaa perinteisempää oikeuksien hallintaa, jossa esimerkiksi järjestelmänvalvojalla voisi olla oikeudet kaikkiin toimintoihin.

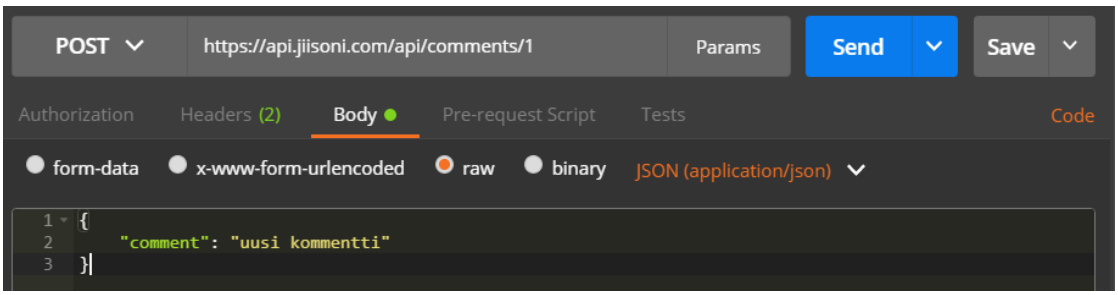
7.1.4 POST metodien käyttäminen

POST ja PATCH metodit eroavat GET metodista siten, että niiden mukana voidaan lähettää myös tietoja pyynnön mukana (toisin kuin GET metodissa, jossa voidaan omia

arvoja ainoastaan rajoittuneissa otsaketiedoissa). Tämä mahdollistaa muun muassa tietojen päivittämisen ja jopa tiedostojen lataamisen palvelimelle.

Pääsääntöisesti POST metodien tehtävänä on luoda uusi resurssi palvelimelle (esimerkiksi uusi opinnäytetyö, tai lisätä uusi tiedosto – siis resurssi jolle ei ole olemassa toimivaa polkua ennestään). On olemassa myös PUT metodi, jonka tehtävä on hieman samankaltainen POST metodin kanssa, mutta PUT metodi asettaisi halutut tiedot ennestään tiedossa olevaan resurssipaikkaan.

Yksinkertainen esimerkki POST metodin käytöstä voisi olla esimerkiksi kommentin lisääminen opinnäytetyölle. Oletetaan, että käyttäjällä on voimassa oleva sessio, ja lähetetään palvelimelle uuden kommentin vaatimat tiedot pyynnön mukana kuvion 31 mukaisesti.



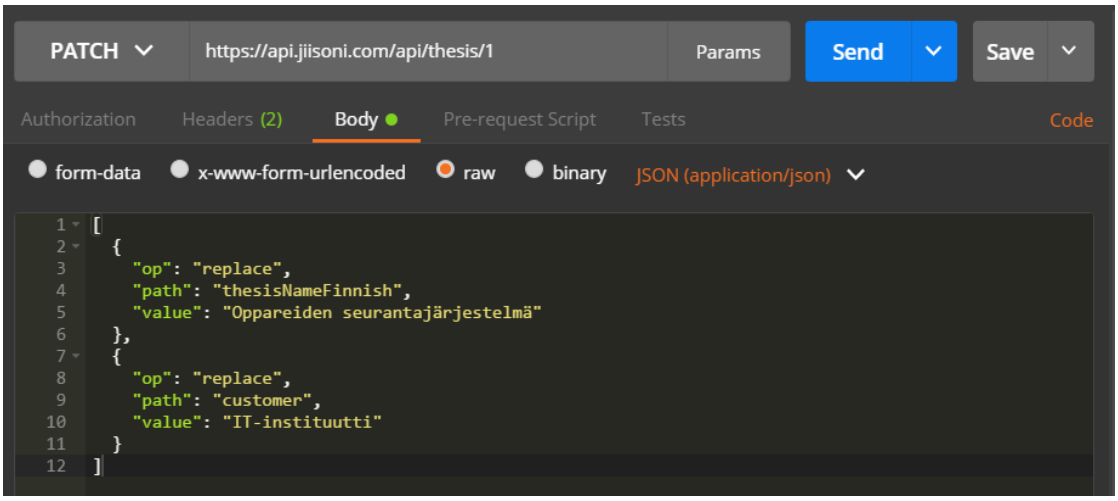
KUVIO 31 Uuden kommentin lisääminen

Huomattavaa on, että opinnäytetyö jolle kommentti lisätään, määritellään sivuston polussa: arvo 1 polun lopussa kuvaa halutun opinnäytetyön tunnistetta. Itse kommentin lisäämiseen tarvotaan vain yksi JSON-muotoisen viestin arvo, ”comment”. Kuviossa 32 voidaan todeta, että kommentti ilmestyi käyttöliittymälle.

KUVIO 32 Uusi kommentti käyttöliittymällä

7.1.5 PATCH metodien käyttäminen

PATCH metodien tarkoitus on päivittää olemassa olevaa resurssia, tai tarkemmin ”paikata” olemassa olevan resurssin arvoja. Tämä tarkoittaa sitä, että esimerkiksi opinnäytetöiden tapauksessa voidaan lähettää viesti, joka vaihtaa vain yhtä opinnäytetyön arvoa, kuten nimeä tai valittua ohjaajaa. Tämän toteutuksen tapauksessa on käytetty ”JSON Patch” formaattia. Tässä formaatissa määritellään jokaiselle arvolle operaation tyyppi, polku ja arvo. Kuviossa 33 on kuvattu viestin sisältö esimerkiksi sellaisena, että se muuttaisi opinnäytetyön nimen sekä tilaajan.



KUVIO 33 PATCH metodin esimerkki

Tässäkin tapauksessa on huomioitavaa, että päivitettävän resurssin tunniste on määritelty osoiterivillä.

8 Kehitysehdotukset

Palvelun testaaminen jäi tässä työssä käytännön olemattomaksi. Esimerkiksi yksikkötestit olisi ollut hyvä kirjoittaa, mutta ajankäytön takia niitä ei ole tässä projektissa kirjoitettu. Lisäksi integraatio testejä ei ole. Nämä voisi olla olemassa, mutta eivät ole – täten ne ovat myös potentiaalisia jatkokehitysideoita. Kunnolla kirjoitetut speksit nykyiselle toteutukselle sekä TDD (test-driven development) metodologian käyttäminen jatkossa varmistaisi ohjelmiston tasaisen laadun.

Koodissa voi myös havaita pieniä epäjohdonmukaisuuksia johtuen uusien asioiden kokeilusta, jota varten koodia voisi hieman yhtenäistää nykyisestä. Lisäksi virheenkäsittely ei ole erityisten loisteliasta nykyisessä toteutuksessa – usein vain heitetään virhekoodi rajapinnalta eikä virhetilanteita esimerkiksi lokiteta.

Muiden ehdotusten lisäksi autentikoimiseen käytettävää koodia (ja tietokantarakennetta) voisi parannella. Koska halusin kirjoittaa oman palikan autentikoimista varten (jossa on lisänä Novell:n LDAP-toteutuksen .NET Core käännös), se ei ole ihan täysiverinen moniltakaan toiminnoiltaan. Oma toteutukseni ei esimerkiksi tarjoa tällä hetkellä saman käyttäjän montaa yhtäaikaista sessiota, mutta tämä olisi suhteellisen helppo toteuttaa pienillä muutoksilla ohjelman koodiin ja tietokantaan.

9 Pohdinta

Opinnäytetyön tavoitteena oli rakentaa palvelu, joka mahdollistaa opinnäytetöiden tilanteen seuraamisen yhden, keskitetyn palvelun kautta joka mahdollistaisi myös useamman henkilön yhtäaikaisen käytön. Pääasiassa tähän myös päästiin, ja ainoat mahdolliset puutteet speksatuista ominaisuuksista saattaa löytyä haluttujen tietojen formaatista tai muodosta (jokin kenttä ei välttämättä tietotyypiltään ole halutunlainen, tai tiedon nimi ei välttämättä tarkalleen kuvaa haluttua tietoa). Nämä ovat kuitenkin helposti korjattavia kohtia jälkituotannossa, eikä tällaisille asioille ole välttämättä annettu kovin suurta painoarvoa kehityksen aikana. Ohjelmistot kuitenkin saattavat nykypäivänä kehittyä julkaisun jälkeen huomattavasti erilaisiksi kuin mitä alkuperäiset tavoitteet ovat olleet.

Toteutukseen valitut teknologiat on valittu pitkälti omien kiinnostuksien mukaan, kunhan se on ajanut asiansa. Valinnat ovat kuitenkin osoittautuneet toimiviksi, ja niiden kanssa painiminen on varmasti ollut hyödyllistä kokemusta myöhempiä sovelluksia varten.

Toteutuksen suurimpia ongelmia oli varmaankin tiedostojen lataamisen toteuttaminen. Angular 2:n kanssa tuli ongelmia ensin lähettää tunnistetiedot palvelimelle ja tämän jälkeen ladata tiedosto palvelimelta ilman välikäsittelyjä, sillä tiedosto latautui Angularin kyselyn skopen ”sisään” katseltavaksi. Tähän ei sillä hetkellä käytetty Angularin versio antanut tallennus ratkaisua suoraan, vaan ongelman ratkaisu olisi tarkoittanut Angularin http-moduulin laajentamista käsin. Tämä taas ei tuntunut kovin mielekkäältä ratkaisulta ottaen huomioon, että Angular 2:n kehitykselle ei näkynyt loppua. Tämän johdosta päädyin luomaan rajapintaan kaksi ohjainta jotka liittyvät tiedostojen lataamiseen, sekä yhden relaation lisää tietokantaan. Ajatus on, että käyttäjän klikatessa tiedoston lataukseen tarkoitettua linkkiä Angular lähettäisi tunnistautumistiedot palvelimelle, palauttaisi tiedostokohtaisen avaimen käyttöliittymälle ja käyttöliittymä automaattisesti avaisi uuden ikkunan osoitteella, jossa generoitu avain tiedostolle on parametrina. Tämä toteutus ei ole sikäli optimaalinen, että useimmissa selaimissa on jonkinlainen popup-blokkaus käytössä, ja käyttäjälle pitää erikseen ker-

toa poistamaan kyseinen ominaisuus käytöstä tiedostoja ladatessa, mutta se on kuitenkin parempi kuin että Angularia mahdollisesti tulevaisuudessa päivitettäessä tiedostojen lataaminen hajoaisi kokonaisuudessaan tuntemattomista syistä.

Työkalujen sekä teknologioiden osalta palvelun toteutus muuttui parikin kertaa merkittävästi. Jossain vaiheessa testailin hieman PHP:lla ja PostgreSQL:lla toteutuksen tekoa, ja tämä olisi varmaan ollutkin se nopein tie voittoon. PHP ei kuitenkaan saa minkänlaisia intohimon tuntoja aikaan itsessäni, joten hylkäsin sen vaihtoehtona parin sivun toteutuksen jälkeen. Tämän jälkeen tulikin kokeiltua seksikkäämpää Python-kieltä, jolla ohjelmoin jotakuinkin toimivan demo-toteutuksen melko lyhyessä ajassa. Tämä toteutus sisälsi tärkeimmät ominaisuudet kirjautumisen ja tietojen kirjaamisen kannalta, mutta siitä puuttui vielä käyttäjien esimerkiksi käyttäjien hallinta sekä tiedostojen lataaminen. Pythonin kanssa tuli myös käytettyä ”Flask” nimistä mikro kirjastoa rajapinnan luomiseen, joka lopulta vaihtui .NET:lla ja C#:lla toteutettuun rajapintaan. Frontend puolella oli käytössä AngularJS.

C#:n käyttöönoton lisäksi frontend puoli vaihtui hyvin erilaiseen Angular 2 -kehykseen, vaikka nimestä voisikin jotain muuta päätellä. Myös frontend-puolen kieli muuttui JavaScriptistä TypeScriptiksi, vaikka jälkimmäinen todellisuudessa kääntyykin takaisin JavaScriptiksi. C#:n valinnan taustalla oli myös .NET Core alustan Linux yhteensopivuus, sillä Windows-palvelimet ovat pääsääntöisesti lisenssien vuoksi kalliimpia kuin Linux vastaavat. Koko kehitykseen käytettävä ”pino” olisi siis täysin ilmainen käyttää myöhemmissä projekteissa.

Lähteet

Jyväskylän Ammattikorkeakoulu. 2015. Tutustu ja menesty. Viitattu 21.11.2016. http://www.jamk.fi/fi/Tietoa-JAMKista/Tutustu-JAMKiin/

Ubuntu. Ubuntu Server Guide, Firewall. Viitattu 21.11.2016. https://help.ubuntu.com/lts/serverguide/firewall.html

The Transport Layer Security (TLS) Protocol Version 1.2. Viitattu 3.12.2016. https://tools.ietf.org/html/rfc5246#section-7.4.7

npm, Inc. Installing npm packages globally. Viitattu 24.1.2017. https://docs.npmjs.com/getting-started/installing-npm-packages-globally

Microsoft. Middleware. Viitattu 1.2.2017. https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware

Liitteet

Liite 1. SQL-skripti tietokanta rakenteen luomiseen

```
CREATE ROLE "thesis" WITH
        NOSUPERUSER NOCREATEDB NOCREATEROLE NOINHERIT LOGIN
        ENCRYPTED PASSWORD 'f65e4f04958192d2d25306729faa6fc3b369635bc36bfede0105aef2cb9ec7ee'
        CONNECTION LIMIT 0;

CREATE TABLE "Users" (
        "UserId" SERIAL NOT NULL,
        "Username" text,
        "Firstname" text,
        "Lastname" text,
        "Phone" text,
        "Email" text,
        "EnrollmentDate" timestamp with time zone,
        "ExpectedEndOfStudies" timestamp with time zone,
        "StudyingPrivilegeEnds" timestamp with time zone,
        "IsArchived" bool,
        "ObjectCreated" timestamp with time zone,
        "ObjectModified" timestamp with time zone DEFAULT now(),
        PRIMARY KEY("UserId"),
        CONSTRAINT "Unique_Key_UserId" UNIQUE("UserId")
);

ALTER TABLE "Users" OWNER TO "thesis";

CREATE TABLE "Theses" (
        "ThesisId" SERIAL NOT NULL,
        "UserId" int4 NOT NULL,
        "ThesisNameFinnish" text,
        "ThesisNameEnglish" text,
        "FirstDirector" int4,
        "SecondDirector" int4,
        "PeerReviewer" int4,
        "Orientation" text,
        "Customer" text,
        "StartingDate" timestamp with time zone,
        "ThesisSubjectAgreementDate" timestamp with time zone,
        "LanguageDirectingProcess" int4,
        "EnglishSummaryProgress" int4,
        "UrkundResult" int4,
        "PublishedInTheseus" bool DEFAULT False,
        "GradeSuggestion" int4 DEFAULT 0,
        "ThesisFinishedDate" timestamp with time zone,
```

```
                "IsArchived" bool,
                "IsLocked" bool,
                "ObjectModified" timestamp with time zone,
                "ObjectCreated" timestamp with time zone DEFAULT now(),
                PRIMARY KEY("ThesisId"),
                CONSTRAINT "Unique_Foreign_Key" UNIQUE("UserId"),
                CONSTRAINT "Unique_Key_ThesisId" UNIQUE("ThesisId")
);

ALTER TABLE "Theses" OWNER TO "thesis";

CREATE TABLE "Events" (
                "EventId" SERIAL NOT NULL,
                "ThesisId" int4 NOT NULL,
                "CreatorId" int4 NOT NULL,
                "Details" text,
                "EventStart" timestamp with time zone,
                "EventEnd" timestamp with time zone,
                "ObjectModified" timestamp with time zone,
                "ObjectCreated" timestamp with time zone NOT NULL DEFAULT now(),
                PRIMARY KEY("EventId"),
                CONSTRAINT "EventId" UNIQUE("EventId")
);

ALTER TABLE "Events" OWNER TO "thesis";

CREATE TABLE "Credentials" (
                "UserId" SERIAL NOT NULL,
                "Username" text,
                "Password" text,
                "Salt" text,
                "Token" text,
                "Timestamp" timestamp with time zone NOT NULL DEFAULT transaction_timestamp(),
                "Lifespan" int4,
                "AuthenticationService" text,
                "AccessLevel" int4 NOT NULL DEFAULT 0,
                PRIMARY KEY("UserId"),
                CONSTRAINT "Unique_Identifier" UNIQUE("UserId")
);

ALTER TABLE "Credentials" OWNER TO "thesis";

COMMENT ON CONSTRAINT  ON "Credentials" IS 'False';

CREATE TABLE "Comments" (
                "CommentId" SERIAL NOT NULL,
```

```
                    "ThesisId" int4,
                    "CommenterId" int4,
                    "Content" text,
                    "IsArchived" bool DEFAULT False,
                    "ObjectModified" timestamp with time zone,
                    "ObjectCreatedd" timestamp with time zone,
                    PRIMARY KEY("CommentId"),
                    CONSTRAINT "Unique_Key_CommentId" UNIQUE("CommentId")
);

ALTER TABLE "Comments" OWNER TO "thesis";

CREATE TABLE "Documents" (
                    "DocumentId" SERIAL NOT NULL,
                    "ThesisId" int4 NOT NULL,
                    "CreatorId" int4 NOT NULL,
                    "Name" text,
                    "Path" text,
                    "Description" text,
                    "IsDeleted" bool,
                    "ObjectModified" timestamp with time zone,
                    "ObjectCreated" timestamp with time zone DEFAULT now(),
                    PRIMARY KEY("DocumentId"),
                    CONSTRAINT "Unique_Key_DocumentId" UNIQUE("DocumentId")
);

ALTER TABLE "Documents" OWNER TO "thesis";

ALTER TABLE "Theses" ADD CONSTRAINT "Ref_Theses_to_Users" FOREIGN KEY ("UserId")
                    REFERENCES "Users"("UserId")
                    MATCH SIMPLE
                    ON DELETE NO ACTION
                    ON UPDATE NO ACTION
                    NOT DEFERRABLE;

ALTER TABLE "Events" ADD CONSTRAINT "Ref_Events_to_Theses" FOREIGN KEY ("ThesisId")
                    REFERENCES "Theses"("ThesisId")
                    MATCH SIMPLE
                    ON DELETE NO ACTION
                    ON UPDATE NO ACTION
                    NOT DEFERRABLE;

ALTER TABLE "Credentials" ADD CONSTRAINT "Ref_Credentials_to_Users" FOREIGN KEY ("UserId")
                    REFERENCES "Users"("UserId")
                    MATCH SIMPLE
                    ON DELETE RESTRICT
```

```
                ON UPDATE RESTRICT
                NOT DEFERRABLE;

ALTER TABLE "Comments" ADD CONSTRAINT "Ref_Comments_to_Theses" FOREIGN KEY ("ThesisId")
                REFERENCES "Theses"("ThesisId")
                MATCH SIMPLE
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
                NOT DEFERRABLE;

ALTER TABLE "Comments" ADD CONSTRAINT "Ref_Comments_to_Users" FOREIGN KEY ("CommenterId")
                REFERENCES "Users"("UserId")
                MATCH SIMPLE
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
                NOT DEFERRABLE;

ALTER TABLE "Documents" ADD CONSTRAINT "Ref_Documents_to_Theses" FOREIGN KEY ("ThesisId")
                REFERENCES "Theses"("ThesisId")
                MATCH SIMPLE
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
                NOT DEFERRABLE;

ALTER TABLE "Documents" ADD CONSTRAINT "Ref_Documents_to_Users" FOREIGN KEY ("CreatorId")
                REFERENCES "Users"("UserId")
                MATCH SIMPLE
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
                NOT DEFERRABLE;
```

Liite 2. DbContext.cs

```
using ThesisService.Objects;
using Microsoft.EntityFrameworkCore;

namespace ThesisService.Context
{
    public class AuthenticationServiceDbContext : DbContext
    {

        public AuthenticationServiceDbContext(DbContextOptions options)
            : base(options)
        {
        }

        public AuthenticationServiceDbContext()
        {
        }

        public DbSet<User> Users { get; set; }
        public DbSet<Credential> Credentials { get; set; }
        public DbSet<Thesis> Thesis { get; set; }
        public DbSet<Comment> Comments { get; set; }
        public DbSet<Event> Events { get; set; }
        public DbSet<Document> Documents { get; set; }
        public DbSet<Download> Downloads { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            base.OnConfiguring(optionsBuilder);
            optionsBuilder
                .UseNpgsql("User ID=thesis;" +
                           "Password=salasana;Host=127.0.0.1;" +
                           "Port=5432;" +
                           "Database=thesis;" +
                           "Pooling=true;");
        }

        protected override void OnModelCreating(ModelBuilder builder)
        {
            builder.Entity<User>().HasKey(o => o.UserId);
            builder.Entity<User>().ForNpgsqlToTable("Users");

            builder.Entity<Credential>().HasKey(o => o.UserId);
            builder.Entity<Credential>().ForNpgsqlToTable("Credentials");
```

```
            builder.Entity<Thesis>().HasKey(o => o.ThesisId);
            builder.Entity<Thesis>().ForNpgsqlToTable("Theses");

            builder.Entity<Comment>().HasKey(o => o.CommentId);
            builder.Entity<Comment>().ForNpgsqlToTable("Comments");

            builder.Entity<Event>().HasKey(o => o.EventId);
            builder.Entity<Event>().ForNpgsqlToTable("Events");

            builder.Entity<Document>().HasKey(o => o.DocumentId);
            builder.Entity<Document>().ForNpgsqlToTable("Documents");

            builder.Entity<Download>().HasKey(o => o.DownloadId);
            builder.Entity<Download>().ForNpgsqlToTable("Downloads");

            base.OnModelCreating(builder);
        }
    }
}
```

Liite 3. AccessControlController.cs

```
using System;
using System.Linq;
using ThesisService.Context;
using ThesisService.Models;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.JsonPatch;
using Microsoft.AspNetCore.Mvc;
using ThesisService.Enums;
using ThesisService.Objects;

namespace ThesisService.Controllers
{
  [Route("api/[controller]")]
  public class AccessControlController : Controller
  {
    private readonly AuthenticationServiceDbContext _context;
    private readonly HttpContext _httpContext;

    public AccessControlController(AuthenticationServiceDbContext context, IHttpContextAccessor httpContext)
    {
      _httpContext = httpContext.HttpContext;
      _context = context;
    }

    /// <summary>
    /// Gets a list of users and their access levels
    /// </summary>
    /// <returns></returns>
    [HttpGet]
    public IActionResult Get()
    {
      var claims = (ClaimsDto)_httpContext.Items["Claims"];

      if (claims.AccessLevel != (int) Role.Manager
        && claims.AccessLevel != (int) Role.Admin)
        return Unauthorized();

      using (_context)
      {
        var users = (from credential in _context.Credentials
          join user in _context.Users
          on credential.UserId equals user.UserId
          select new UserCredentialsDto
          {
```

```
            UserId = credential.UserId,
            Username = user.Username,
            Firstname = user.Firstname,
            Lastname = user.Lastname,
            AccessLevel = credential.AccessLevel
        }).ToList();

    return new JsonResult(users);
  }
}

/// <summary>
/// Patches a new access level to an user
/// -> Make sure no other properties are changed except access level!
/// </summary>
/// <param name="update"></param>
/// <param name="id"></param>
/// <returns></returns>
[HttpPatch("{id}")]
public IActionResult Patch([FromBody] JsonPatchDocument update, int id)
{
  var claims = (ClaimsDto)_httpContext.Items["Claims"];

  if (claims.AccessLevel != (int) Role.Manager
    && claims.AccessLevel != (int) Role.Admin)
    return Unauthorized();

  try
  {
    using (_context)
    {
      var userCredentials = _context.Credentials.First(o => o.UserId == id);
      if (userCredentials == null)
      {
        return BadRequest();
      }

      var entityCopy = new Credential();

      update.ApplyTo(entityCopy);

      userCredentials.AccessLevel = entityCopy.AccessLevel;

      if (userCredentials.AccessLevel > (int) Role.Admin)
      {
        return BadRequest();
```

```
            }

            _context.Update(userCredentials);
            _context.SaveChanges();
          }

          return new JsonResult(new GenericMessage
          {
            Type = "Message", Message = "Updated access level!"
          });
        }
        catch (Exception)
        {
          return BadRequest();
        }
      }

    }
}
```

Liite 4. AuthenticationController.cs

```
using System;
using System.Globalization;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using ThesisService.Context;
using ThesisService.Enums;
using ThesisService.Models;
using ThesisService.Objects;

namespace ThesisService.Controllers
{
  [Route("api/[controller]")]
  public class AuthenticationController : Controller
  {
    private readonly AuthenticationServiceDbContext _context;

    public AuthenticationController(AuthenticationServiceDbContext context)
    {
      _context = context;
    }

    // POST: api/authentication
```

```
[HttpPost]
public async Task<ActionResult> Post([FromBody] LoginDto credentials)
{
  var users = _context.Credentials;

  if (credentials.AuthenticationService == "Internal" || credentials.AuthenticationService == null)
  {
    var getUser = (from user in users
      where user.Username == credentials.Username
        && user.Password == credentials.Password
        && user.AuthenticationService == "Internal"
      select new Credential
      {
        UserId = user.UserId,
        Username = user.Username,
        Token = user.Token,
        Lifespan = user.Lifespan,
        Timestamp = user.Timestamp,
        AuthenticationService = user.AuthenticationService,
        Salt = user.Salt,
        Password = user.Password,
        AccessLevel = user.AccessLevel
      }).FirstOrDefault();

    if (getUser == null)
      return BadRequest("Invalid login information.");

    var currentTime = DateTime.UtcNow;

    // Create token
    using (var sha256 = SHA256.Create())
    {
      var hashedBytes = sha256.ComputeHash(Encoding.UTF8
        .GetBytes(getUser.Salt + getUser.UserId + currentTime.ToString(CultureInfo.InvariantCulture) +
getUser.Username));
      var hash = BitConverter.ToString(hashedBytes).Replace("-", "").ToLower();
      getUser.Token = hash;
    }

    getUser.Timestamp = currentTime;
    getUser.Lifespan = 10;

    var userThesisId = (from thesis in _context.Thesis
      where thesis.UserId == getUser.UserId
      select new
      {
```

```
        thesis.ThesisId
    }).Count();

// If thesis doesn't exist, create one
if (userThesisId == 0)
{
    var newThesis = new Thesis
    {
        UserId = getUser.UserId,
        ThesisNameFinnish = "N/A",
        ThesisNameEnglish = "N/A",
        FirstDirector = null,
        SecondDirector = null,
        Orientation = "",
        Customer = "",
        StartingDate = null,
        ThesisSubjectAgreementDate = null,
        LanguageDirectingProcess = false,
        EnglishSummaryApproved = false,
        UrkundResult = null,
        GradeSuggestion = null,
        ThesisFinishedDate = null,
        ObjectCreated = DateTime.UtcNow
    };

    try
    {
        _context.Thesis.Add(newThesis);
        await _context.SaveChangesAsync();
    }
    catch (Exception e)
    {
        return new JsonResult(e);
    }
}

try
{
    _context.Update(getUser);
    await _context.SaveChangesAsync();
}
catch (Exception e)
{
    return new JsonResult(e);
}
```

```
            // Use CredentialDTO to transfer the token to the client
            var transferToken = new CredentialDto
            {
                Token = getUser.Token,
                Username = getUser.Username,
                UserId = getUser.UserId,
                Role = Enum.GetName(typeof(Role), getUser.AccessLevel)
            };

            return new JsonResult(transferToken);
        }

        var message = new GenericMessage();

        if (credentials.AuthenticationService == "ActiveDirectory/JAMK")
        {
            message.Type = "Notification";
            message.Message = "This login service has not been implemented yet.";
        }
        else
        {
            message.Type = "Notification";
            message.Message = "Unkown login method.";
        }
        return new JsonResult(message);
    }
  }
}
```

Liite 5. CommentsController.cs

```
using System;
using System.Linq;
using System.Threading.Tasks;
using ThesisService.Context;
using ThesisService.Models;
using ThesisService.Objects;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace ThesisService.Controllers
{
  [Route("api/[controller]")]
  public class CommentsController : Controller
  {
```

```
private readonly AuthenticationServiceDbContext _context;
private readonly HttpContext _httpContext;


public CommentsController(AuthenticationServiceDbContext context, IHttpContextAccessor httpContext)
{
    _context = context;
    _httpContext = httpContext.HttpContext;
}

/// <summary>
/// Get comments from a thesis
/// </summary>
/// <returns></returns>
/// api/comments/{id}
[HttpGet("{id}")]
public ActionResult Get(int id)
{

    var claims = (ClaimsDto)_httpContext.Items["Claims"];

    if(!CheckUser(id, claims.UserId, claims.AccessLevel)) return Unauthorized();

    var comments = (from comment in _context.Comments
                    join user in _context.Users on comment.CommenterId equals user.UserId
                    where comment.ThesisId == id
                    select new CommentDto
                    {
                        CommentId = comment.CommentId,
                        CommenterId = comment.CommenterId,
                        Comment = comment.Content,
                        CommenterName = user.Firstname + " " + user.Lastname,
                        CommentCreated = comment.ObjectCreated
                    }).ToList();

    return new JsonResult(comments);
}


/// <summary>
/// Posts a new comment and takes ThesisId as parameter
/// </summary>
/// <param name="newComment">New comment object</param>
/// <param name="id">ThesisId</param>
/// <returns></returns>
[HttpPost("{id}")]
```

```
        public async Task<ActionResult> Post([FromBody]CommentDto newComment, int id)
        {
            var claims = (ClaimsDto)_httpContext.Items["Claims"];

            if (!CheckUser(id, claims.UserId, claims.AccessLevel)) return Unauthorized();

            var comment = new Comment
            {
                ThesisId = id,
                CommenterId = claims.UserId,
                Content = newComment.Comment,
                IsArchived = false,
                ObjectCreated = DateTime.UtcNow,
                ObjectModified = null
            };

            try
            {
                _context.Comments.Add(comment);
                await _context.SaveChangesAsync();
                return Ok();
            }
            catch (Exception)
            {
                return BadRequest();
            }
        }

        private bool CheckUser(int thesisId, int userId, int accessLevel)
        {
            int? checkUser = (from thesis in _context.Thesis
                              where thesis.ThesisId == thesisId
                                  && (thesis.UserId == userId
                                      || thesis.FirstDirector == userId
                                      || thesis.SecondDirector == userId
                                      || accessLevel >= 2)
                              select new Thesis()).First().ThesisId;

            return checkUser.HasValue;
        }
    }
}
```

Liite 6. DocumentController.cs

```
using System;
using System.Globalization;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using ThesisService.Context;
using ThesisService.Models;
using ThesisService.Objects;

namespace ThesisService.Controllers
{
    [Route("api/[controller]")]
    public class DocumentController : Controller
    {
        private readonly AuthenticationServiceDbContext _context;
        private readonly HttpContext _httpContext;

        public DocumentController(AuthenticationServiceDbContext context, IHttpContextAccessor httpContext)
        {
            _context = context;
            _httpContext = httpContext.HttpContext;
        }

        /// <summary>
        /// Gets all files associated with a thesis
        /// </summary>
        /// <param name="id">Thesis Id</param>
        /// <returns></returns>
        [HttpGet("{id}")]
        public ActionResult GetList(int id)
        {
            var claims = (ClaimsDto) _httpContext.Items["Claims"];

            try
            {
                var userCheck = (from thesis in _context.Thesis
                    where
                    thesis.ThesisId == id
                    && (claims.UserId == thesis.UserId
                        || claims.UserId == thesis.FirstDirector
                        || claims.UserId == thesis.SecondDirector
                        || claims.UserId == thesis.PeerReviewer)
```

```
            select new Thesis {ThesisId = thesis.ThesisId}).Select(o => o.ThesisId).First();
        }
        catch (Exception)
        {
            return NotFound();
        }

        var documents = (from doc in _context.Documents
            where doc.ThesisId == id
            select new DocumentDto
            {
                DocumentId = doc.DocumentId,
                DocumentName = doc.Name,
                DocumentCreated = doc.ObjectCreated,
                DocumentCreator = (from user in _context.Users
                    where doc.CreatorId == user.UserId
                    select new DocumentCreator
                    {
                        UserId = user.UserId,
                        Name = user.Firstname + " " + user.Lastname,
                        Role = (from creds in _context.Credentials
                            where creds.UserId == doc.CreatorId
                            select new Credential
                            {
                                AccessLevel = creds.AccessLevel
                            }).Select(o => o.AccessLevel).FirstOrDefault()
                    }).FirstOrDefault()
            }).ToList();

        return new JsonResult(documents);
    }

    /// <summary>
    ///     Generates a single use key that let's the user to consume that key and download a file
    /// </summary>
    /// <param name="fileId"></param>
    /// <returns></returns>
    [HttpGet("key/{fileId}")]
    public ActionResult Key(int fileId)
    {
        var claims = (ClaimsDto) _httpContext.Items["Claims"];

        var document = (from doc in _context.Documents
            where doc.DocumentId == fileId
                && doc.IsDeleted == false
            select new Document
```

```
    {
        DocumentId = doc.DocumentId,
        Path = doc.Path,
        Name = doc.Name,
        ThesisId = doc.ThesisId,
        CreatorId = doc.CreatorId
    }).FirstOrDefault();

var thesisUsers = (from thesis in _context.Thesis
    where thesis.ThesisId == document.ThesisId
        && (thesis.UserId == claims.UserId
            || thesis.FirstDirector == claims.UserId
            || thesis.SecondDirector == claims.UserId
            || thesis.PeerReviewer == claims.UserId)
    select new Thesis
    {
        UserId = thesis.UserId,
        FirstDirector = thesis.UserId,
        SecondDirector = thesis.UserId,
        PeerReviewer = thesis.PeerReviewer
    }).FirstOrDefault();

// User should have access to thesis documents only
if (claims.UserId == thesisUsers.UserId && claims.UserId != document.CreatorId)
    return
        new JsonResult(new GenericMessage
        {
            Message = "You don't have the rights to access this file.",
            Type = "Error"
        });

// Peer reviewer has access to thesis and peer review documents
if (claims.UserId == thesisUsers.PeerReviewer &&
    (document.CreatorId == thesisUsers.UserId || document.CreatorId == claims.UserId))
    return
        new JsonResult(new GenericMessage
        {
            Message = "You don't have the rights to access this file.",
            Type = "Error"
        });

// Generate an unique key
var key = GenerateKey(claims.UserId, fileId, document.ThesisId);

// Create new entity
```

```
            var singleUseDownloadKey = new Download
            {
                FilePath = document.Path,
                IsUsed = false,
                AccessKey = key,
                Created = DateTime.UtcNow
            };

            // Save key to database
            try
            {
                _context.Downloads.Add(singleUseDownloadKey);
                _context.SaveChangesAsync();
            }
            catch
            {
                return
                    new JsonResult(new GenericMessage
                    {
                        Message = "Something went wrong while creating a key.",
                        Type = "Error"
                    });
            }

            return new JsonResult(new GenericMessage {Message = singleUseDownloadKey.AccessKey, Type = "AccessKey"});
        }

        private static string GenerateKey(int userId, int fileId, int thesisId)
        {
            var datetimeHash = DateTime.UtcNow.ToString(CultureInfo.InvariantCulture);
            var compilationHash = userId + datetimeHash + fileId + thesisId;

            // Create key string
            using (var sha256 = SHA256.Create())
            {
                var hashedBytes = sha256.ComputeHash(Encoding.UTF8
                    .GetBytes(compilationHash));
                var hash = BitConverter.ToString(hashedBytes).Replace("-", "").ToLower();
                return hash;
            }
        }
    }
}
```

Liite 7. DownloadController.cs

```
using System;
using System.Linq;
using System.Threading.Tasks;
using ThesisService.Context;
using Microsoft.AspNetCore.Mvc;

namespace ThesisService.Controllers
{
    [Route("api/[controller]")]
    public class DownloadController
    {

        private readonly AuthenticationServiceDbContext _context;

        public DownloadController(AuthenticationServiceDbContext context)
        {
            _context = context;
        }

        /// <summary>
        /// Let's user download a file with a key
        /// </summary>
        /// <returns></returns>
        [HttpGet]
        public async Task<FileResult> Download([FromQuery]string key)
        {
            var downloadDetails = (from download in _context.Downloads
                                   where download.AccessKey == key
                                       && download.IsUsed == false
                                   select download).First();

            if (downloadDetails.Created.AddMinutes(5) < DateTime.UtcNow) return null;

            downloadDetails.IsUsed = true;
            _context.Downloads.Update(downloadDetails);
            await _context.SaveChangesAsync();

            var result = new FileContentResult(
                System.IO.File.ReadAllBytes("Documents/" + downloadDetails.FilePath), "application/octet-stream")
            {
                FileDownloadName = downloadDetails.FilePath
            };

            return result;
```

```
        }
    }
}
```

Liite 8. FileController.cs

```
using System;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using ThesisService.Context;
using ThesisService.Models;
using ThesisService.Objects;

namespace ThesisService.Controllers
{
    [Route("api/[controller]")]
    public class FileController : Controller
    {
        private readonly AuthenticationServiceDbContext _context;
        private readonly HttpContext _httpContext;

        public FileController(AuthenticationServiceDbContext context, IHttpContextAccessor httpContext)
        {
            _context = context;
            _httpContext = httpContext.HttpContext;
        }

        /// <summary>
        ///     Allows the user to upload new files
        /// </summary>
        /// <param name="file">File data</param>
        /// <param name="id">ThesisId</param>
        /// <returns></returns>
        [HttpPost("{id}")]
        public async Task<IActionResult> Upload(IFormFile file, int id)
        {
            var claims = (ClaimsDto) _httpContext.Items["Claims"];

            var thesisIds = (from thesis in _context.Thesis
                where thesis.ThesisId == id
                    && (claims.UserId == thesis.UserId
                        || claims.UserId == thesis.FirstDirector
                        || claims.UserId == thesis.SecondDirector
```

```
                || claims.UserId == thesis.PeerReviewer)
            select new Thesis()).FirstOrDefault();

        if (thesisIds == null)
            return null;

        var currentDatetime = DateTime.UtcNow;

        if (file.Length > 0)
            using (
                var fileStream =
                    new FileStream(
                        Path.Combine("Documents/",
                            id + "_" + currentDatetime.ToString("ddMMyyyy_H-mm-ss") + "_" + file.FileName),
                        FileMode.Create))
            {
                await file.CopyToAsync(fileStream);
            }

        var document = new Document
        {
            Name = file.FileName,
            Path = id + "_" + currentDatetime.ToString("ddMMyyyy_H-mm-ss") + "_" + file.FileName,
            CreatorId = claims.UserId,
            ThesisId = id,
            ObjectCreated = DateTime.UtcNow,
            IsDeleted = false
        };

        try
        {
            _context.Documents.Add(document);
            await _context.SaveChangesAsync();
        }
        catch (Exception e)
        {
            Console.Write("Couldn't create new tuple for a document! \n" + e);
        }
        return Ok();
    }
  }
}
```

Liite 9. PlannerController.cs

```
using System;
using System.Linq;
using ThesisService.Context;
using ThesisService.Models;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.JsonPatch;
using Microsoft.AspNetCore.Mvc;
using ThesisService.Enums;
using ThesisService.Objects;

namespace ThesisService.Controllers
{
    [Route("api/[controller]")]
    public class PlannerController : Controller
    {
        private readonly AuthenticationServiceDbContext _context;
        private readonly HttpContext _httpContext;

        public PlannerController(AuthenticationServiceDbContext context, IHttpContextAccessor httpContext)
        {
            _httpContext = httpContext.HttpContext;
            _context = context;
        }

        // Get
        [HttpGet]
        public ActionResult Get(int id)
        {
            var claims = (ClaimsDto)_httpContext.Items["Claims"];

            if (claims.AccessLevel != (int) Role.Manager
                && claims.AccessLevel != (int) Role.Admin)
            {
                var hasClaims = _context.Thesis.First(o => o.ThesisId == id &&
                                                      (
                                                          o.UserId == claims.UserId
                                                          || o.FirstDirector == claims.UserId
                                                          || o.SecondDirector == claims.UserId
                                                      ));

                if (hasClaims == null) return Unauthorized();
            }

            var listOfEvents = (from evt in _context.Events
```

```
        join user in _context.Users on evt.CreatorId equals user.UserId
        where evt.ThesisId == id
        select new PlannerDto
        {
            Creator = "(" + user.Username + ") " + user.Firstname + " " + user.Lastname,
            Details = evt.Details,
            EventStart = evt.EventStart,
            EventEnd = evt.EventEnd,
            ObjectModified = evt.ObjectModified,
            ObjectCreated = evt.ObjectCreated
        }).ToList();

    return new JsonResult(listOfEvents);
}

// Post
[HttpPost("{id}")]
public IActionResult Post(PlannerDto newEntry, int id)
{
    if(!ModelState.IsValid) return BadRequest();

    var claims = (ClaimsDto) _httpContext.Items["Claims"];
    try
    {
        if (claims.AccessLevel != (int) Role.Manager
            && claims.AccessLevel != (int) Role.Admin)
        {
            var thesis = _context.Thesis
                .First(o => o.ThesisId == id
                        || o.UserId == claims.UserId
                        || o.FirstDirector == claims.UserId
                        || o.SecondDirector == claims.UserId);

            if (thesis == null) return Unauthorized();
        }

        var newEvent = new Event
        {
            ThesisId = id,
            CreatorId = claims.UserId,
            Details = newEntry.Details,
            EventStart = newEntry.EventStart,
            EventEnd = newEntry.EventEnd,
            ObjectCreated = DateTime.UtcNow
        };
```

```
            Console.WriteLine(Json(newEvent).ToString());

            _context.Add(newEvent);
            _context.SaveChangesAsync();
        }
        catch (Exception e)
        {
            Console.WriteLine(e);
            BadRequest();
        }

        return Ok();
    }

  }
}
```

Liite 10. OverviewController.cs

```
using System.Linq;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using ThesisService.Context;

using ThesisService.Enums;

using ThesisService.Models;


namespace ThesisService.Controllers

{

  [Route("api/[controller]")]

  public class OverviewController : Controller

  {

    private readonly AuthenticationServiceDbContext _context;

    private readonly HttpContext _httpContext;


    public OverviewController(AuthenticationServiceDbContext context, IHttpContextAccessor httpContext)

    {

      _context = context;

      _httpContext = httpContext.HttpContext;
```

```
}

/// <summary>
///     Gets the user's own thesis
/// </summary>
/// <returns></returns>
[HttpGet]
public ActionResult Get()
{
    var claims = (ClaimsDto) _httpContext.Items["Claims"];

    if (claims.AccessLevel != (int) Role.Director
        && claims.AccessLevel != (int) Role.Manager
        && claims.AccessLevel != (int) Role.Admin)
        return Unauthorized();

    using (_context)
    {
        var allAvailableThesis = (from thesis in _context.Thesis
            where claims.UserId == thesis.UserId
                || claims.UserId == thesis.FirstDirector
                || claims.UserId == thesis.SecondDirector
                || claims.AccessLevel == (int) Role.Manager
                || claims.AccessLevel == (int) Role.Admin

            select new OverviewThesisDto
            {
                ThesisId = thesis.ThesisId,
                ThesisNameFinnish = thesis.ThesisNameFinnish,
                ThesisNameEnglish = thesis.ThesisNameEnglish,
                FirstDirector = (from user in _context.Users
                    where thesis.FirstDirector == user.UserId
```

```
    select new DirectorDto
    {
        DirectorName = user.Firstname + ' ' + user.Lastname,
        DirectorId = user.UserId
    }).FirstOrDefault(),
SecondDirector = (from user in _context.Users
    where thesis.SecondDirector == user.UserId
    select new DirectorDto
    {
        DirectorName = user.Firstname + ' ' + user.Lastname,
        DirectorId = user.UserId
    }).FirstOrDefault(),
Orientation = thesis.Orientation,
Customer = thesis.Customer,
StartingDate = thesis.StartingDate,
ThesisSubjectAgreementDate = thesis.ThesisSubjectAgreementDate,
LanguageDirectingProcess = thesis.LanguageDirectingProcess,
EnglishSummaryApproved = thesis.EnglishSummaryApproved,
UrkundResult = thesis.UrkundResult,
PublishedInTheseus = thesis.PublishedInTheseus,
GradeSuggestion = thesis.GradeSuggestion,
ThesisFinishedDate = thesis.ThesisFinishedDate,
// User details
Author = (from user in _context.Users
    where thesis.UserId == user.UserId
    select new Author
    {
        Username = user.Username,
        Firstname = user.Firstname,
        Lastname = user.Lastname,
        Email = user.Email
    }).FirstOrDefault()
```

```
            }).ToList();

            return new JsonResult(allAvailableThesis);

        }

    }

  }

}
```

Liite 11. ThesisController.cs

```
using System;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.JsonPatch;
using Microsoft.AspNetCore.Mvc;
using ThesisService.Context;
using ThesisService.Enums;
using ThesisService.Models;
using ThesisService.Objects;

namespace ThesisService.Controllers
{
  [Route("api/[controller]")]
  public class ThesisController : Controller
  {
    private readonly AuthenticationServiceDbContext _context;
    private readonly HttpContext _httpContext;

    public ThesisController(AuthenticationServiceDbContext context, IHttpContextAccessor httpContext)
    {
      _context = context;
      _httpContext = httpContext.HttpContext;
    }

    /// <summary>
    ///     Gets the user's own thesis
    /// </summary>
    /// <returns></returns>
    [HttpGet]
    public ActionResult Get()
    {
      var claims = (ClaimsDto) _httpContext.Items["Claims"];

      var getThesis = (from thesis in _context.Thesis
```

```
        where claims.UserId == thesis.UserId
            && thesis.IsArchived != true
        select new ThesisDto
        {
            ThesisNameFinnish = thesis.ThesisNameFinnish,
            ThesisNameEnglish = thesis.ThesisNameEnglish,
            FirstDirector = thesis.FirstDirector,
            SecondDirector = thesis.SecondDirector,
            Orientation = thesis.Orientation,
            Customer = thesis.Customer,
            StartingDate = thesis.StartingDate,
            ThesisSubjectAgreementDate = thesis.ThesisSubjectAgreementDate,
            LanguageDirectingProcess = thesis.LanguageDirectingProcess,
            EnglishSummaryApproved = thesis.EnglishSummaryApproved,
            UrkundResult = thesis.UrkundResult,
            PublishedInTheseus = thesis.PublishedInTheseus,
            GradeSuggestion = thesis.GradeSuggestion,
            ThesisFinishedDate = thesis.ThesisFinishedDate
        }).ToList().FirstOrDefault();

    return new JsonResult(getThesis);
}

/// <summary>
///     Gets a thesis by Id
/// </summary>
/// <param name="id"></param>
/// <returns></returns>
[HttpGet("{id}")]
public ActionResult Get(int id)
{
    var claims = (ClaimsDto) _httpContext.Items["Claims"];

    var getThesis = (from thesis in _context.Thesis
        where thesis.ThesisId == id
            && (thesis.UserId == claims.UserId
                || thesis.FirstDirector == claims.UserId
                || thesis.SecondDirector == claims.UserId
                || claims.AccessLevel == (int) Role.Manager
                || claims.AccessLevel == (int) Role.Admin)
        select new ThesisDto
        {
            ThesisNameFinnish = thesis.ThesisNameFinnish,
            ThesisNameEnglish = thesis.ThesisNameEnglish,
            FirstDirector = thesis.FirstDirector,
            SecondDirector = thesis.SecondDirector,
```

```
            Orientation = thesis.Orientation,
            Customer = thesis.Customer,
            StartingDate = thesis.StartingDate,
            ThesisSubjectAgreementDate = thesis.ThesisSubjectAgreementDate,
            LanguageDirectingProcess = thesis.LanguageDirectingProcess,
            EnglishSummaryApproved = thesis.EnglishSummaryApproved,
            UrkundResult = thesis.UrkundResult,
            PublishedInTheseus = thesis.PublishedInTheseus,
            GradeSuggestion = thesis.GradeSuggestion,
            ThesisFinishedDate = thesis.ThesisFinishedDate
        }).ToList();

    var singleThesis = getThesis.FirstOrDefault();
    return new JsonResult(singleThesis);
}

/// <summary>
///     Create new thesis for the user
/// </summary>
/// <returns></returns>
[HttpPost]
public async Task<ActionResult> Post()
{
    var claims = (ClaimsDto) _httpContext.Items["Claims"];

    var getThesis = (from thesis in _context.Thesis
        where claims.UserId == thesis.UserId
            && thesis.IsArchived != true
        select thesis).FirstOrDefault();

    // If thesis doesn't exist, create one
    if (getThesis != null) return new JsonResult(getThesis);

    var newThesis = new Thesis
    {
        UserId = claims.UserId,
        ThesisNameFinnish = "N/A",
        ThesisNameEnglish = "N/A",
        FirstDirector = null,
        SecondDirector = null,
        Orientation = "",
        Customer = "",
        StartingDate = null,
        ThesisSubjectAgreementDate = null,
        LanguageDirectingProcess = false,
        EnglishSummaryApproved = false,
```

```
        UrkundResult = null,
        PublishedInTheseus = false,
        GradeSuggestion = null,
        ThesisFinishedDate = null,
        ObjectCreated = DateTime.UtcNow
    };

    try
    {
        _context.Thesis.Add(newThesis);
        await _context.SaveChangesAsync();
        return new JsonResult(newThesis);
    }
    catch (Exception e)
    {
        return new JsonResult(e);
    }
}

/// <summary>
///     Patches new properties into a thesis
/// </summary>
/// <param name="update"></param>
/// <param name="id"></param>
/// <returns></returns>
[HttpPatch("{id}")]
public async Task<ActionResult> Patch([FromBody] JsonPatchDocument<ThesisDto> update, int id)
{
    if (!ModelState.IsValid)
        return BadRequest(ModelState);

    var claims = (ClaimsDto) _httpContext.Items["Claims"];

    var oldThesis = (from thesis in _context.Thesis
        where thesis.ThesisId == id
            && (thesis.UserId == claims.UserId
                || thesis.FirstDirector == claims.UserId
                || thesis.SecondDirector == claims.UserId
                || claims.AccessLevel == (int) Role.Manager
                || claims.AccessLevel == (int) Role.Admin)
        select thesis
    ).FirstOrDefault();

    if (oldThesis == null)
        return BadRequest("Thesis not found.");
```

```
// Convert booleans

var intermediaryModel = new ThesisDto
{
   ThesisNameFinnish = oldThesis.ThesisNameFinnish,
   ThesisNameEnglish = oldThesis.ThesisNameEnglish,
   FirstDirector = oldThesis.FirstDirector,
   SecondDirector = oldThesis.SecondDirector,
   Orientation = oldThesis.Orientation,
   Customer = oldThesis.Customer,
   StartingDate = oldThesis.StartingDate,
   ThesisSubjectAgreementDate = oldThesis.ThesisSubjectAgreementDate,
   LanguageDirectingProcess = oldThesis.LanguageDirectingProcess,
   EnglishSummaryApproved = oldThesis.EnglishSummaryApproved,
   UrkundResult = oldThesis.UrkundResult,
   PublishedInTheseus = oldThesis.PublishedInTheseus,
   GradeSuggestion = oldThesis.GradeSuggestion,
   ThesisFinishedDate = oldThesis.ThesisFinishedDate
};

if (oldThesis.IsLocked == true)
   return new JsonResult(new GenericMessage {Message = "Cannot update locked entities.", Type = "Error"});

// Apply changes to intermediary model
update.ApplyTo(intermediaryModel);


// Apply changes to database object
oldThesis.ThesisNameFinnish = intermediaryModel.ThesisNameFinnish;
oldThesis.ThesisNameEnglish = intermediaryModel.ThesisNameEnglish;
oldThesis.FirstDirector = intermediaryModel.FirstDirector;
oldThesis.SecondDirector = intermediaryModel.SecondDirector;
oldThesis.Orientation = intermediaryModel.Orientation;
oldThesis.Customer = intermediaryModel.Customer;
oldThesis.StartingDate = intermediaryModel.StartingDate;
oldThesis.ThesisSubjectAgreementDate = intermediaryModel.ThesisSubjectAgreementDate;
oldThesis.LanguageDirectingProcess = intermediaryModel.LanguageDirectingProcess;
oldThesis.EnglishSummaryApproved = intermediaryModel.EnglishSummaryApproved;
oldThesis.UrkundResult = intermediaryModel.UrkundResult;
oldThesis.PublishedInTheseus = intermediaryModel.PublishedInTheseus;
oldThesis.GradeSuggestion = intermediaryModel.GradeSuggestion;
oldThesis.ThesisFinishedDate = intermediaryModel.ThesisFinishedDate;
oldThesis.ObjectModified = DateTime.UtcNow;

try
{
```

```
            await _context.SaveChangesAsync();
        }
        catch (Exception e)
        {
            return new JsonResult(e);
        }

        return new JsonResult(new GenericMessage {Message = "OK", Type = "Message"});
    }

    /// <summary>
    ///     Gets a list people that can act as a director
    /// </summary>
    /// <returns></returns>
    [HttpGet("directors")]
    public ActionResult GetDirectors()
    {
        var directors = (from director in _context.Users
            join credential in _context.Credentials
            on director.UserId equals credential.UserId
            where credential.AccessLevel == 1
            select new DirectorDto
            {
                DirectorId = director.UserId,
                DirectorName = director.Firstname + " " + director.Lastname
            }).ToList();

        return new JsonResult(directors);
    }
  }
}
```

Liite 12. UserController.cs

```
using System;
using System.Linq;
using Microsoft.AspNetCore.JsonPatch;
using Microsoft.AspNetCore.Mvc;
using ThesisService.Context;
using ThesisService.Enums;
using ThesisService.Models;

namespace ThesisService.Controllers
{
  [Route("api/[controller]")]
  public class UserController : Controller
```

```
{
    private readonly AuthenticationServiceDbContext _dbContext;

    public UserController(AuthenticationServiceDbContext dbContext)
    {
        _dbContext = dbContext;
    }

    /// <summary>
    ///     Gets information of the user that made the request
    /// </summary>
    /// <returns></returns>
    [HttpGet]
    public IActionResult Get()
    {
        var claims = (ClaimsDto) HttpContext.Items["Claims"];

        using (_dbContext)
        {
            var userProfile = (from user in _dbContext.Users
                where user.UserId == claims.UserId
                select new UserDto
                {
                    Username = user.Username,
                    Firstname = user.Firstname,
                    Lastname = user.Lastname,
                    Email = user.Email,
                    Phone = user.Phone,
                    EnrollmentDate = user.EnrollmentDate,
                    ExpectedEndOfStudies = user.ExpectedEndOfStudies,
                    StudyingPrivilegeEnds = user.StudyingPrivilegeEnds
                }).FirstOrDefault();

            if (userProfile == null)
                return NotFound();

            return Json(userProfile);
        }
    }

    /// <summary>
    ///     Returns user by ID if the requesting user has the right privileges
    /// </summary>
    /// <param name="id"></param>
    /// <returns></returns>
    [HttpGet("{id}")]
```

```
public IActionResult Get(int id)
{
    var claims = (ClaimsDto) HttpContext.Items["Claims"];

    if (claims.AccessLevel != (int) Role.Director
        && claims.AccessLevel != (int) Role.Manager
        && claims.AccessLevel != (int) Role.Admin)
        return BadRequest();

    var userProfile = (from user in _dbContext.Users
        where user.UserId == id
        select new UserDto
        {
            Username = user.Username,
            Firstname = user.Firstname,
            Lastname = user.Lastname,
            Email = user.Email,
            Phone = user.Phone,
            EnrollmentDate = user.EnrollmentDate,
            ExpectedEndOfStudies = user.ExpectedEndOfStudies,
            StudyingPrivilegeEnds = user.StudyingPrivilegeEnds
        }).FirstOrDefault();

    if (userProfile == null)
        return NotFound();

    return Json(userProfile);
}

/// <summary>
///     Update user profile
/// </summary>
/// <param name="update"></param>
/// <returns></returns>
[HttpPatch]
public IActionResult Post([FromBody] JsonPatchDocument<UserDto> update)
{
    var claims = (ClaimsDto) HttpContext.Items["Claims"];

    if (!ModelState.IsValid)
        return BadRequest();

    var storedEntity = _dbContext.Users.First(user => user.UserId == claims.UserId);

    if (storedEntity == null)
        return BadRequest();
```

```
            var intermediateEntity = new UserDto
            {
                Firstname = storedEntity.Firstname,
                Lastname = storedEntity.Lastname,
                Email = storedEntity.Email,
                Phone = storedEntity.Phone,
                EnrollmentDate = storedEntity.EnrollmentDate,
                ExpectedEndOfStudies = storedEntity.ExpectedEndOfStudies,
                StudyingPrivilegeEnds = storedEntity.StudyingPrivilegeEnds
            };

            update.ApplyTo(intermediateEntity);

            storedEntity.Firstname = intermediateEntity.Firstname;
            storedEntity.Lastname = intermediateEntity.Lastname;
            storedEntity.Email = intermediateEntity.Email;
            storedEntity.Phone = intermediateEntity.Phone;
            storedEntity.EnrollmentDate = intermediateEntity.EnrollmentDate;
            storedEntity.ExpectedEndOfStudies = intermediateEntity.ExpectedEndOfStudies;
            storedEntity.StudyingPrivilegeEnds = intermediateEntity.StudyingPrivilegeEnds;

            try
            {
                _dbContext.Update(storedEntity);
                _dbContext.SaveChanges();
            }
            catch (Exception e)
            {
                return BadRequest(e);
            }

            return Ok();

        }
    }
}
```

Liite 13. Role.cs

```
namespace ThesisService.Enums
{
    public enum Role
    {
        Student,
        Director,
```

```
        Manager,
        Admin
    }
}
```

Liite 14. Authorization.cs

```
using System;
using System.Linq;
using System.Threading.Tasks;
using ThesisService.Context;
using ThesisService.Models;
using ThesisService.Objects;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Http;

namespace ThesisService.Middleware
{
    /// <summary>
    /// Authorization middleware that determines if a user should be allowed to access the service.
    /// Also passes on an user claim object which contains essential information what the user should be
    /// able to do within the service.
    /// </summary>
    public class AuthorizationMiddleware
    {
        private readonly RequestDelegate _next;

        public AuthorizationMiddleware(RequestDelegate next)
        {
            _next = next;
        }

        public async Task Invoke(HttpContext httpContext, AuthenticationServiceDbContext context)
        {
            var requestPath = httpContext.Request.Path.ToString();

            // On authentication request, bypass authorization
            if (requestPath == "/api/authentication"
                || requestPath == "/api/authentication/"
                || requestPath == "/api/download"
                || requestPath == "/api/download/")
            {
                await _next(httpContext);
                return;
            }
```

```
var authHeader = httpContext.Request.Headers["Authorization"].ToString();

// Don't needlessly query database if there is no 'Authorization' header present
if (string.IsNullOrEmpty(authHeader))
{
   httpContext.Response.StatusCode = 302; // Redirect
   httpContext.Response.Redirect("/api/authentication");
   return;
}

var tokens = context.Credentials;
var users = context.Users;

var getTokenFromDb = (from token in tokens
                 where token.Token == authHeader
                 select new Credential
                 {
                    UserId = token.UserId,
                    Token = token.Token,
                    AccessLevel = token.AccessLevel,
                    Timestamp = token.Timestamp,
                    Lifespan = token.Lifespan
                 }).FirstOrDefault();

if(getTokenFromDb == null)
{
   httpContext.Response.StatusCode = 401; // Redirect
   return;
}

// Check if the token is valid or not
if (string.IsNullOrEmpty(getTokenFromDb.Token)
   || getTokenFromDb.Timestamp.AddHours(1) < DateTime.UtcNow)
{
   httpContext.Response.StatusCode = 401; // Redirect
   return;
}

// Claims are granted on request basis, and are generated on every request
var userClaims = (from user in users
                  where user.UserId == getTokenFromDb.UserId
                  select new ClaimsDto
                  {
                     UserId = user.UserId,
                     Username = user.Username,
                     Firstname = user.Firstname,
```

```
                    Lastname = user.Lastname,
                    AccessLevel = getTokenFromDb.AccessLevel
                  }).FirstOrDefault();

        if (userClaims == null)
        {
          httpContext.Response.StatusCode = 500; // Internal server error
          await httpContext.Response.WriteAsync("Claims handling encountered an unexpected error.");
          return;
        }

        // Add claims object to a dictionary that will be disposed when response is sent back to the browser
        httpContext.Items["Claims"] = userClaims;

        Console.WriteLine("User: " + userClaims.Username + " is accessing the service.");

        // If user credentials are valid, go to the next middleware component
        await _next(httpContext);
      }

    }

    public class AuthorizationMiddlewareExtensions
    {
      public IApplicationBuilder UseAuthorizationMiddleware(IApplicationBuilder builder)
      {
        return builder.UseMiddleware<AuthorizationMiddleware>();
      }
    }
  }
```

Liite 15. ClaimsDTO.cs

```
namespace ThesisService.Models
{
    public class ClaimsDto
    {
        public int UserId { get; set; }
        public string Username { get; set; }
        public string Firstname { get; set; }
        public string Lastname { get; set; }
        public int AccessLevel { get; set; }
    }
}
```

Liite 16. CommentDto.cs

```
using System;

namespace ThesisService.Models
{
    public class CommentDto
    {
        public int? CommentId { get; set; }
        public string Comment { get; set; }
        public int? CommenterId { get; set; }
        public string CommenterName { get; set; }
        public DateTime CommentCreated { get; set; }
    }
}
```

Liite 17. CredentialDto.cs

```
namespace ThesisService.Models
{
    public class CredentialDto
    {
        public int UserId { get; set; }
        public string Username { get; set; }
        public string Token { get; set; }
        public string Role { get; set; }
    }
}
```

Liite 18. DirectorDto.cs

```
namespace ThesisService.Models
{
   public class DirectorDto
   {
      public string DirectorName { get; set; }
      public int DirectorId { get; set; }
   }
}
```

Liite 19. DocumentDto.cs

```
using System;

namespace ThesisService.Models
{
   public class DocumentDto
   {
      public int DocumentId { get; set; }
      public string DocumentName { get; set; }
      public DateTime DocumentCreated { get; set; }
      public DocumentCreator DocumentCreator { get; set; }
   }

   public class DocumentCreator
   {
      public int UserId { get; set; }
      public string Name { get; set; }
      public int Role { get; set; }
   }
}
```

Liite 20. GenericMessage.cs

```
namespace ThesisService.Models
{
   public class GenericMessage
   {
      public string Type { get; set; }
      public string Message { get; set; }
   }
}
```

Liite 21. LoginDto.cs

```
namespace ThesisService.Models
{
   public class LoginDto
   {
      public string Username { get; set; }
      public string Password { get; set; }
      public string AuthenticationService { get; set; }
   }
}
```

Liite 22. OverviewThesisDto.cs

```
using System;

namespace ThesisService.Models
{
   public class OverviewThesisDto
   {
      public int ThesisId { get; set; }
      public string ThesisNameFinnish { get; set; }
      public string ThesisNameEnglish { get; set; }
      public DirectorDto FirstDirector { get; set; }
      public DirectorDto SecondDirector { get; set; }
      public string Orientation { get; set; }
      public string Customer { get; set; }
      public DateTime? StartingDate { get; set; }
      public DateTime? ThesisSubjectAgreementDate { get; set; }
      public bool LanguageDirectingProcess { get; set; }
      public int? EnglishSummaryProgress { get; set; }
      public int? UrkundResult { get; set; }
      public bool? PublishedInTheseus { get; set; }
      public string TheseusLink { get; set; }
      public int? GradeSuggestion { get; set; }
      public DateTime? ThesisFinishedDate { get; set; }
      public Author Author { get; set; }
      public bool EnglishSummaryApproved { get; set; }
   }
   public class Author
   {
      public string Username;
      public string Firstname;
      public string Lastname;
      public string Email;
   }
}
```

Liite 23. PlannerDto.cs

```
using System;

namespace ThesisService.Models
{
    public class PlannerDto
    {
        public int EventId { get; set; }
        public string Details { get; set; }
        public string Creator { get; set; }
        public int CreatorId { get; set; }
        public DateTime? EventStart { get; set; }
        public DateTime? EventEnd { get; set; }
        public DateTime? ObjectModified { get; set; }
        public DateTime ObjectCreated { get; set; }
    }
}
```

Liite 24. ThesisDto.cs

```
using System;

namespace ThesisService.Models
{
    public class ThesisDto
    {
        public string ThesisNameFinnish { get; set; }
        public string ThesisNameEnglish { get; set; }
        public int? FirstDirector { get; set; }
        public int? SecondDirector { get; set; }
        public string Orientation { get; set; }
        public string Customer { get; set; }
        public DateTime? StartingDate { get; set; }
        public DateTime? ThesisSubjectAgreementDate { get; set; }
        public bool LanguageDirectingProcess { get; set; }
        public bool EnglishSummaryApproved { get; set; }
        public int? UrkundResult { get; set; }
        public bool PublishedInTheseus { get; set; }
        public int? GradeSuggestion { get; set; }
        public DateTime? ThesisFinishedDate { get; set; }
    }
}
}
```

Liite 25. UserCredentialsDto.cs

```
namespace ThesisService.Models
{
   public class UserCredentialsDto
   {
      public int UserId { get; set; }
      public string Username { get; set; }
      public string Firstname { get; set; }
      public string Lastname { get; set; }
      public int AccessLevel { get; set; }
   }
}
```

Liite 26. UserDto.cs

```
using System;

namespace ThesisService.Models
{
   public class UserDto
   {
      public string Username { get; set; }
      public string Firstname { get; set; }
      public string Lastname { get; set; }
      public string Phone { get; set; }
      public string Email { get; set; }
      public DateTime? EnrollmentDate { get; set; }
      public DateTime? ExpectedEndOfStudies { get; set; }
      public DateTime? StudyingPrivilegeEnds { get; set; }
   }
}
```

Liite 27. Comment.cs

```
using System;

namespace ThesisService.Objects
{
   public class Comment
   {
      [System.ComponentModel.DataAnnotations.Key]
      public int CommentId { get; set; }
      public int ThesisId { get; set; }
      public int CommenterId { get; set; }
      public string Content { get; set; }
      public bool? IsArchived { get; set; }
      public DateTime? ObjectModified { get; set; }
      public DateTime ObjectCreated { get; set; }
   }
}
```

Liite 28. Credential.cs

```
using System;

namespace ThesisService.Objects
{
   public class Credential
   {
      [System.ComponentModel.DataAnnotations.Key]
      public int UserId { get; set; }
      public string Username { get; set; }
      public string Password { get; set; }
      public string Salt { get; set; }
      public string Token { get; set; }
      public DateTime Timestamp { get; set; }
      public int? Lifespan { get; set; }
      public string AuthenticationService { get; set; }
      public int AccessLevel { get; set; }
   } }
```

Liite 29. Document.cs

```
using System;

namespace ThesisService.Objects
{
   public class Document
   {
      [System.ComponentModel.DataAnnotations.Key]
      public int DocumentId { get; set; }
      public int ThesisId { get; set; }
      public int CreatorId { get; set; }
      public string Name { get; set; }
      public string Path { get; set; }
      public string Description { get; set; }
      public bool IsDeleted { get; set; }
      public DateTime? ObjectModified { get; set; }
      public DateTime ObjectCreated { get; set; }
   }
}
```

Liite 30. Download.cs

```
using System;

namespace ThesisService.Objects
{
   public class Download
   {
      [System.ComponentModel.DataAnnotations.Key]
      public int DownloadId { get; set; }
      public bool IsUsed { get; set; }
      public string FilePath { get; set; }
      public string AccessKey { get; set; }
      public DateTime Created { get; set; }
   }
}
```

Liite 31. Event.cs

```
using System;

namespace ThesisService.Objects
{
   public class Event
   {
```

```
        [System.ComponentModel.DataAnnotations.Key]
        public int EventId { get; set; }
        public int ThesisId { get; set; }
        public int CreatorId { get; set; }
        public string Details { get; set; }
        public DateTime? EventStart { get; set; }
        public DateTime? EventEnd { get; set; }
        public DateTime? ObjectModified { get; set; }
        public DateTime ObjectCreated { get; set; }
    }
}
```

Liite 32. Thesis.cs

```
using System;
using System.ComponentModel.DataAnnotations;

namespace ThesisService.Objects
{
    public class Thesis
    {
        [System.ComponentModel.DataAnnotations.Key]
        public int ThesisId { get; set; }
        public int UserId { get; set; }
        public string ThesisNameFinnish { get; set; }
        public string ThesisNameEnglish { get; set; }
        public int? FirstDirector { get; set; }
        public int? SecondDirector { get; set; }
        public int? PeerReviewer { get; set; }
        public string Orientation { get; set; }
        public string Customer { get; set; }

        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:yyyy-MM-dd}")]
        public DateTime? StartingDate { get; set; }

        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:yyyy-MM-dd}")]
        public DateTime? ThesisSubjectAgreementDate { get; set; }

        public bool LanguageDirectingProcess { get; set; }
        public bool EnglishSummaryApproved { get; set; }
        public int? UrkundResult { get; set; }
        public bool PublishedInTheseus { get; set; }
        public int? GradeSuggestion { get; set; }

        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:yyyy-MM-dd}")]
        public DateTime? ThesisFinishedDate { get; set; }
```

```
        public bool? IsArchived { get; set; }
        public bool? IsLocked { get; set; }

        public DateTime? ObjectModified { get; set; }
        public DateTime? ObjectCreated { get; set; }
    }
}
```

Liite 33. User.cs

```
using System;

namespace ThesisService.Objects
{
   public class User
   {
      [System.ComponentModel.DataAnnotations.Key]
      public int UserId { get; set; }
      public string Username { get; set; }
      public string Firstname { get; set; }
      public string Lastname { get; set; }
      public string Phone { get; set; }
      public string Email { get; set; }
      public DateTime? EnrollmentDate { get; set; }
      public DateTime? ExpectedEndOfStudies { get; set; }
      public DateTime? StudyingPrivilegeEnds { get; set; }
   }
}
```

Liite 34. Program.cs

```
using System.IO;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;

namespace ThesisService
{
   public class Program
   {
      public static void Main(string[] args)
      {

         new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .Build();

         var host = new WebHostBuilder()
            .UseKestrel()
            .UseUrls("http://*:5000")
            .UseContentRoot(Directory.GetCurrentDirectory())
            .UseIISIntegration()
            .UseStartup<Startup>()
            .Build();
```

```
            host.Run();
        }
    }
}
```

Liite 35. Project.json

```
{
  "dependencies": {
    "System.Security.Cryptography.Algorithms": "4.3.0",
    "Npgsql.EntityFrameworkCore.PostgreSQL": "1.1.0",
    "Npgsql.EntityFrameworkCore.PostgreSQL.Design": "1.1.0",
    "Microsoft.AspNetCore.Authentication.Cookies": "1.1.0",
    "Microsoft.AspNetCore.Cors": "1.1.0",
    "Microsoft.AspNetCore.Http.Extensions": "1.1.0",
    "Microsoft.AspNetCore.JsonPatch": "1.1.0",
    "Microsoft.AspNetCore.Mvc": "1.1.0",
    "Microsoft.AspNetCore.Mvc.Core": "1.1.0",
    "Microsoft.AspNetCore.Server.IISIntegration": "1.1.0",
    "Microsoft.AspNetCore.Server.Kestrel": "1.1.0",
    "Microsoft.EntityFrameworkCore": "1.1.0",
    "Microsoft.Extensions.Configuration.EnvironmentVariables": "1.1.0",
    "Microsoft.Extensions.Configuration.FileExtensions": "1.1.0",
    "Microsoft.Extensions.Configuration.Json": "1.1.0",
    "Microsoft.Extensions.Configuration.UserSecrets": "1.1.0",
    "Microsoft.Extensions.Logging": "1.1.0",
    "Microsoft.Extensions.Logging.Console": "1.1.0",
    "Microsoft.Extensions.Logging.Debug": "1.1.0",
    "Microsoft.Extensions.Options.ConfigurationExtensions": "1.1.0",
    "Microsoft.NETCore.App": "1.1.0",
    "Novell.Directory.Ldap.NETStandard": "2.3.6"
  },

  "version": "1.0.0-*",
  "buildOptions": {
    "debugType": "portable",
    "emitEntryPoint": true
  },

  "frameworks": {
    "netcoreapp1.1": {
      "dependencies": {
      },
      "imports": "dnxcore50"
    }
  },
```

```
  "runtimes": {
    "win10-x64": {
      "#import": [ "win10", "win81-x64" ]
    },
    "ubuntu-x64": {
      "#import": [ "ubuntu", "debian-x64" ]
    }
  },

  "runtimeOptions": {
    "configProperties": {
      "System.GC.Server": true
    }
  },

  "publishOptions": {
    "include": [
      "wwwroot",
      "Views",
      "Areas/**/Views",
      "appsettings.json",
      "web.config"
    ]
  },

  "scripts": {
    "postpublish": [ "dotnet publish-iis --publish-folder %publish:OutputPath% --framework %publish:FullTargetFramework%" ]
  }
}
```

Liite 36. Startup.cs

```
using System.IO;
using ThesisService.Context;
using ThesisService.Middleware;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;

namespace ThesisService
{
  public class Startup
  {
    public IConfigurationRoot Configuration { get; set; }

    public Startup(IHostingEnvironment env)
    {
      var builder = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory())
                .AddJsonFile("appsettings.json", false, true)
                .AddJsonFile($"appsettings.{env.EnvironmentName}.json", true)
                .AddJsonFile("appsettings.docker.json", true)
                .AddEnvironmentVariables();

      Configuration = builder.Build();
    }

    // This method gets called by the runtime. Use this method to add services to the container.
    public void ConfigureServices(IServiceCollection services)
    {
      // Add framework services.
      services.AddMvc();
      services.AddRouting();

      // CrossOrigin
      services.AddCors(options =>
      {
        options.AddPolicy("AllowSpecificOrigin",
          builder => builder.AllowAnyOrigin()
            .AllowAnyHeader()
            .AllowAnyMethod()
        );
      });
```

```
        services.AddSingleton(Configuration);
        services.AddSingleton<IHttpContextAccessor, HttpContextAccessor>();
        services.AddDbContext<AuthenticationServiceDbContext>();
      }

      // Pipeline configuration
      public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory loggerFactory)
      {
        app.UseCors("AllowSpecificOrigin");
        app.UseMiddleware<AuthorizationMiddleware>();
        loggerFactory.AddConsole(Configuration.GetSection("Logging"));
        loggerFactory.AddDebug();
        app.UseMvc();
      }
    }
}
```

Liite 37. access-control.component.html

```
<div class="pure-g page-sub-header">
  <div class="pure-u-1-5">
    <div class="page-sub-name">
    Käyttäjähallinta
    </div>
  </div>
</div>
<div class="pure-g">
  <div class="pure-u-1 pure-form">
    <table class="pure-table pure-table-striped pure-table-bordered" style="width: 100%">
      <thead>
        <th>Id</th>
        <th>Käyttäjätunnus</th>
        <th>Etunimi</th>
        <th>Sukunimi</th>
        <th>Käyttäjätaso</th>
      </thead>
      <tr *ngFor="let user of users" id="user-{{user.UserId}}">
        <td>{{ user.UserId }}</td>
        <td>{{ user.Username }}</td>
        <td>{{ user.Firstname }}</td>
        <td>{{ user.Lastname }}</td>
        <td>
          <div class="pure-control-group">
            <select (change)="onChange($event.target.id, $event.target.value)"
              id="{{ user.UserId }}"
              name="user-{{ user.UserId }}"
```

```
                [(ngModel)]="user.AccessLevel">
                <option [ngValue]="0">Oppilas</option>
                <option [ngValue]="1">Ohjaaja</option>
                <option [ngValue]="2">Manageri</option>
                <option [ngValue]="3">Järjestelmänvalvoja</option>
              </select>
            </div>
          </td>
        </tr>
      </table>
    </div>
</div>
```

Liite 38. access-control.component.ts

```
import { Component, Input }        from '@angular/core';
import { Router, ActivatedRoute }   from '@angular/router';
import { Validators, FormBuilder }  from '@angular/forms';
import { AccessControlService }     from './access-control.service';

@Component({
  selector: 'accessControl',
  templateUrl: './access-control.component.html'
})

export class AccessControlComponent {

  private users = new Array();
  private user: Object = {};

  constructor(private accessControlService: AccessControlService, private router: Router) {

    // Populate page with all Users and their access levels
    let result = this.accessControlService.getUserCredentials().subscribe((result) => {
      result.forEach(user => {
        this.users.push({
          UserId: user.userId,
          Username: user.username,
          Firstname: user.firstname,
          Lastname: user.lastname,
          AccessLevel: user.accessLevel
        });
      });
    })
```

```
  }

  onChange(userId, accessLevel) {
    this.accessControlService.setUserAccessLevel(parseInt(userId), parseInt(accessLevel))
    .subscribe(result => {
      document.getElementById("user-" + userId.toString()).style.backgroundColor = "#27ae60";
    },
    error => {
      document.getElementById("user-" + userId.toString()).style.backgroundColor = "#e74c3c";
    }
    );
  }

}
```

Liite 39. access-control.service.ts

```
import { Injectable }      from '@angular/core';
import { Http, Headers }   from '@angular/http';
import { Observable }      from 'rxjs/Rx';

@Injectable()
export class AccessControlService {

  private api: string = "https://api.jiisoni.com/api/AccessControl/";
  private token: string;

  constructor(private http: Http) {
    this.token = localStorage.getItem('auth_token');
  }

  // Gets a list of users and their access levels
  public getUserCredentials() {

    let headers = new Headers();
    headers.append('Authorization', this.token);
    headers.append('Access-Control-Allow-Origin', 'GET');

    return this.http.get(
      this.api,
      { headers }
    )
    .map((res: any) => res.json())
    .map((res) => {
```

```
        return res;
      });
    }

    public setUserAccessLevel(userId: number, accesslevel: number) {
      let headers = new Headers();
      headers.append('Authorization', this.token);
      headers.append('Access-Control-Allow-Origin', 'PATCH');

      let patchMessage = [{
        op: 'replace', path: 'accessLevel', value: accesslevel
      }];

      return this.http.patch(
        'https://api.jiisoni.com/api/AccessControl/' + userId.toString(),
        patchMessage,
        { headers }
      );
    }

  }
```

Liite 40. authentication.component.html

```
<div class="login-area">
  <div *ngIf="isLoggedIn">
    <button class="pure-button button-warning" (click)="logout()">
      Kirjaudu ulos
    </button>
  </div>

  <div *ngIf="!isLoggedIn">
    <form [formGroup]="authenticationForm" (ngSubmit)="onSubmit()">
      <input type="text" placeholder="Sähköpostiosoite" formControlName="username" id="username" />
      <input type="password" placeholder="Salasana" formControlName="password" id="password" />
      <select formControlName="authenticationService" id="authenticationService">
        <option value="1">AD/JAMK</option>
        <option selected="selected" value="0">Internal</option>
      </select>
      <button type="submit">Kirjaudu</button>
    </form>
  </div>
</div>
```

Liite 41. authentication.component.ts

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { UserService }  from '../services/user.service';
import { Validators, FormBuilder } from '@angular/forms';

@Component({
  selector: 'application-authentication',
  templateUrl: './authentication.component.html'
})

export class AuthenticationComponent {

  authenticationForm: any;
  private isLoggedIn: boolean = false;
  private username: string;
  private userId: number;

  constructor (private userService: UserService, private router: Router, private formBuilder: FormBuilder) {
    if(userService.isLoggedIn()) {
      this.isLoggedIn = true;
      this.username = localStorage.getItem('username');
    } else {
      this.isLoggedIn = false;
    }
    this.authenticationForm = this.formBuilder.group({
      'username': ['', Validators.required],
      'password': ['', Validators.required],
      'authenticationService': []
    });
  }

  onSubmit(username:string, password:string) {
    this.userService.login(this.authenticationForm.value.username, this.authenticationForm.value.password, parseInt(this.au-
thenticationForm.value.authenticationService))
    .subscribe((result) => {
      if (result === true) {
        this.isLoggedIn = true;
        this.username = localStorage.getItem('username');
        setTimeout(function() {
          this.router.navigate(['/'])
        }, 3000);
      }
    });
  }
```

```
  logout() {
    this.isLoggedIn = false;
    this.userService.logout();
    //location.reload();
  }
}
```

Liite 42. header.component.html

```
<nav>
  <div class="pure-g">
    <div class="pure-u-4-5">
      <button routerLink="/home" class="pure-button" *ngIf="isLoggedIn">Etusivu</button>
      <button routerLink="/overview" class="pure-button" *ngIf="isLoggedIn && !isStudent"><i class="fa fa-bar-chart" aria-
hidden="true"></i> Yleisnäkymä</button>
      <button [routerLink]="['/thesis', userId]" class="pure-button" *ngIf="isLoggedIn"><i class="fa fa-graduation-cap" aria-
hidden="true"></i> Opinnäytetyö</button>
      <button [routerLink]="['/profile', userId]" class="pure-button" *ngIf="isLoggedIn"><i class="fa fa-user" aria-hid-
den="true"></i> Profiili</button>
      <button routerLink="/accesscontrol" class="pure-button" *ngIf="isLoggedIn && !isStudent">Käyttäjähallinta</button>
    </div>
    <div class="pure-u-1-5">
      <application-authentication></application-authentication>
    </div>
  </div>
</nav>
```

Liite 43. header.component.ts

```
import { Component } from '@angular/core';
import { AuthenticationComponent } from '../authentication/authentication.component';
import { UserService } from '../services/user.service';

@Component({
 selector: 'application-header',
 templateUrl: './header.component.html'
})
export class HeaderComponent {
  title = "Jiisoni";

  private isLoggedIn: boolean = false;
  private role: string = 'student';
  private userId: string;

  constructor(private userService: UserService) {
   this.isLoggedIn = userService.isLoggedIn();
   if(localStorage.getItem('userId') !== null) {
```

```
      this.userId = localStorage.getItem('userId');
      this.role = localStorage.getItem('role');
     }
   }
}
```

Liite 44. overview.service.ts

```
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs/Rx';
import { Http, Headers } from '@angular/http';
import 'rxjs/add/operator/map';

@Injectable()
export class OverviewService {

   private token: string = localStorage.getItem('auth_token');

   constructor(private http: Http) {

   }
   // Returns a list of theses
   getTheses() {
     let headers = new Headers();
     headers.append('Authorization', this.token);
     headers.append('Access-Control-Allow-Origin', 'GET');

     return this.http
     .get(
        'https://api.jiisoni.com/api/overview',
        { headers }
     )
     .map((res: any) => res.json())
     .map((res) => {
           return res;
        }
     )
  }

}
```

Liite 45. protect.guard.ts

```
// protect.guard.ts
import { Injectable } from '@angular/core';
import { Router, CanActivate } from '@angular/router';
import { UserService } from './user.service';

@Injectable()
export class ProtectGuard implements CanActivate {

  private loggedIn: boolean = false;

  constructor(private user: UserService) {
    this.loggedIn = user.isLoggedIn();
  }

  canActivate() {
    return this.loggedIn;
  }
}
```

Liite 46. user.service.ts

```
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs/Rx';
import { Http, Headers } from '@angular/http';
import 'rxjs/add/operator/map';

@Injectable()
export class UserService {

  private loggedIn = false;
  private username: string;
  private userid: number;
  private lifespan: number;
  private thesisId: number;

  constructor(private http: Http) {
    this.loggedIn = !!localStorage.getItem('auth_token');
  }

  login(username: any, password: any, authServerId: number) {
    let headers = new Headers();
    headers.append('Content-Type', 'application/json');

    let authenticationService: string;

    if (authServerId === 0) {
      authenticationService = 'Internal';
    } else if (authServerId === 1) {
      authenticationService = 'ActiveDirectory/JAMK';
    } else {
      authenticationService = 'ActiveDirectory/JAMK';
    }

    return this.http
    .post(
      'https://api.jiisoni.com/api/authentication',
      { username, password, authenticationService},
      { headers }
    )
    .map((res: any) => res.json())
    .map((res) => {
      if (res.token) {
        localStorage.setItem('auth_token', res.token);
        localStorage.setItem('username', res.username);
        localStorage.setItem('userId', res.userId);
```

```
            localStorage.setItem('role', res.role);
            this.loggedIn = true;
            let el = document.getElementById("authentication-form");
            el.innerHTML = "<i class=\"fa fa-check\" aria-hidden=\"true\"></i> Success! Welcome back, " + res.username + "!";
            el.style.backgroundColor = "#00B16A";
            setTimeout(function() { el.className = "authentication-form-hidden" }, 3000);
            location.reload();
            return true;
          } else {
            false;
          }
        })
    }

    logout() {
        localStorage.removeItem('auth_token');
        localStorage.removeItem('userId');
        localStorage.removeItem('username');
        localStorage.removeItem('thesisId');
        this.loggedIn = false;
        location.reload();
    }

    isLoggedIn() {
        return this.loggedIn;
    }

}
```

Liite 47. home.component.html

```
<div class="pure-g page-sub-header">
  <div class="pure-u-1-5">
    <div class="page-sub-name">
    Etusivu
    </div>
  </div>
</div>
<div class="content">
  <div class="pure-g">
    <div class="pure-u-1">
      <h1>Opinnäytetöiden seuranta</h1>
    </div>
  </div>
</div>
```

Liite 48. home.component.ts

```
import { Component, Input } from '@angular/core';


@Component({
  selector: 'view-home',
  templateUrl: './home.component.html'
})


export class HomeComponent {
   title = "Home";
}
```

Liite 49. overview.component.html

```
<div class="pure-g page-sub-header">
   <div class="pure-u-1-5">
      <div class="page-sub-name">
      Yleisnäkymä
      </div>
   </div>
</div>
<table class="pure-table pure-table-striped pure-table-bordered" style="width: 100%;">
   <thead>
      <tr>
      <th>Opinnäytetyön nimi (Suomeksi)</th>
      <th>Opinnäytetyön nimi (Englanniksi)</th>
      <th>Tilaaja</th>
      <th>Tekijä</th>
      <th>1. Ohjaaja</th>
      <th>2. Ohjaaja</th>
      <th>Kielen arviointi</th>
      <th>Englanninkielinen kuvaus hyväksytty</th>
      <th>Julkaistu Theseuksessa</th>
      <th>Aloitus pvm.</th>
      <th>Aihe hyväksytty pvm.</th>
      <th>Valmistumis pvm.</th>
   </tr>
   </thead>
   <tr *ngFor="let thesis of theses">
```

```
      <td><a [routerLink]="[thesis.Url]">{{thesis.ThesisNameFinnish}}</a></td>
      <td>{{thesis.ThesisNameEnglish}}</td>
      <td>{{thesis.Customer}}</td>
      <td>{{thesis.Firstname}} {{thesis.Lastname}}</td>
      <td>{{thesis.FirstDirector.directorName}}</td>
      <td>{{thesis.SecondDirector.directorName}}</td>
      <td>{{thesis.LanguageDirectingProcess}}</td>
      <td>{{thesis.EnglishSummaryApproved}}</td>
      <td>{{thesis.PublishedInTheseus}}</td>
      <td>{{thesis.ThesisStartingDate | date:'dd / MM / yyyy'}}</td>
      <td>{{thesis.ThesisSubjectAgreementDate | date:'dd / MM / yyyy'}}</td>
      <td>{{thesis.ThesisFinishedDate | date:'dd / MM / yyyy'}}</td>
    </tr>
</table>
```

Liite 50. overview.component.ts

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { Validators, FormBuilder } from '@angular/forms';
import { OverviewService } from '../common/services/overview.service';

@Component({
  selector: 'view-overview',
  templateUrl: './overview.component.html'
})

export class OverviewComponent {

  private ThesisName: string;
  private ThesisDescription: string;
  private ThesisAuthor: string;
  private ThesisState: string;
  private ThesisRelevantDate: string;
  private ThesisSubjectAgreementDate: Date;
  private ThesisStartingDate: Date;
  private ThesisFinishedDate: Date;

  private theses = new Array();

  thesisForm: any;

  originalValues: any[];
  fieldNames: string[];
```

```
  fieldValues: any[];

  private isPublished: boolean;

  constructor (private overviewService: OverviewService, private router: Router) {
    let firstDirector: string;
    let secondDirector: string;
    let result = this.overviewService.getTheses().subscribe((result) => {
      for (let i = 0; i < result.length; i++) {
        this.theses.push(
          {
            Url: '/thesis/' + result[i].thesisId.toString(),
            ThesisNameFinnish: result[i].thesisNameFinnish,
            ThesisNameEnglish: result[i].thesisNameEnglish,
            FirstDirector: result[i].firstDirector || '',
            SecondDirector: result[i].secondDirector || '',
            LanguageDirectingProcess: result[i].languageDirectingProcess,
            EnglishSummaryApproved: result[i].englishSummaryApproved,
            PublishedInTheseus: result[i].publishedInTheseus,
            Customer: result[i].customer,
            ThesisStartingDate: result[i].startingDate,
            ThesisSubjectAgreementDate: result[i].thesisSubjectAgreementDate,
            Firstname: result[i].author.firstname,
            Lastname: result[i].author.lastname,
            Username: result[i].author.username,
            Email: result[i].author.email,
          }
        )
      }
    })
  }

}
```

Liite 51. profile.component.html

```
<div class="pure-g page-sub-header">
  <div class="pure-u-1-5">
    <div class="page-sub-name">
    Profiili
    </div>
  </div>
</div>

<div class="full-width-container">
  <div class="content-middle">

  <form [formGroup]="profileForm" (ngSubmit)="updateProfile()" class="pure-form pure-form-aligned">

    <div class="pure-control-group">
      <label for="username">Käyttäjätunnus</label>
      <input type="text" value="{{ Username }}" formControlName="username" id="username" disabled/>
    </div>

    <div class="pure-control-group">
      <label for="firstname">Etunimi</label>
      <input type="text" value="{{ Firstname }}" formControlName="firstname" id="firstname" />
    </div>

    <div class="pure-control-group">
      <label for="lastname">Sukunimi</label>
      <input type="text" value="{{ Lastname }}" formControlName="lastname" id="lastname" />
    </div>

    <div class="pure-control-group">
      <label for="email">Sähköposti</label>
      <input type="text" value="{{ Email }}" formControlName="email" id="email" />
    </div>

    <div class="pure-control-group">
      <label for="phone">Puhelinnumero</label>
      <input type="text" value="{{ Phone }}" formControlName="phone" id="phone" />
    </div>

    <div class="pure-control-group">
      <label for="enrollmentDate">Opinnot aloitettu</label>
      <input type="datetime" value="{{ EnrollmentDate | date:'yMMMMd'}}" formControlName="enrollmentDate" id="enroll-
mentDate" />
    </div>
```

```
    <div class="pure-control-group">
      <label for="expectedEndOfStudies">Odotettu valmistuminen</label>
      <input type="datetime" value="{{ ExpectedEndOfStudies | date:'yMMMMd'}}" formControlName="expectedEndOfStud-
ies" id="expectedEndOfStudies" />
    </div>

    <div class="pure-control-group">
      <label for="studyingPrivilegeEnds">Opinto-oikeus päättyy</label>
      <input type="datetime" value="{{ StudyingPrivilegeEnds | date:'yMMMMd'}}" formControlName="studyingPrivi-
legeEnds" id="studyingPrivilegeEnds" />
    </div>

    <div class="form-options">
      <button type="submit" class="pure-button"><i class="fa fa-floppy-o" aria-hidden="true"></i> TALLENNA</button>
    </div>
  </form>
  </div>
</div>
```

Liite 52. profile.component.ts

```
import { Component } from '@angular/core';
import { Router, ActivatedRoute } from '@angular/router';
import { Validators, FormBuilder } from '@angular/forms';
import { ProfileService } from './profile.service';

@Component({
  selector: 'view-Profile',
  templateUrl: './Profile.component.html'
})

export class ProfileComponent {

  private Username: string;
  private Firstname: string;
  private Lastname: string;
  private Phone: string;
  private Email: string;
  private EnrollmentDate: Date;
  private ExpectedEndOfStudies: Date;
  private StudyingPrivilegeEnds: Date;

  profileForm: any;
```

```
  private originalValues = new Array;
  private newValues = new Array;

  private isPublished: boolean;
  private userId: number;

  constructor (private _profileService: ProfileService, private router: Router, private route: ActivatedRoute, private formBuilder:
FormBuilder) {

    // Get ProfileId from requested url
    this.route.params.subscribe(params => {
      this.userId = +params['id'];
    });

    let result = this._profileService.getProfile().subscribe((result) => {

      Object.assign(this.originalValues, result);

      this.Username = result.username;
      this.Firstname = result.firstname;
      this.Lastname = result.lastname;
      this.Phone = result.phone;
      this.Email = result.email;
      this.EnrollmentDate = result.enrollmentDate;
      this.ExpectedEndOfStudies = result.expectedEndOfStudies;
      this.StudyingPrivilegeEnds = result.studyingPrivilegeEnds;
    })

    // Form
    this.profileForm = this.formBuilder.group({
      'username': [this.Username],
      'firstname': [this.Firstname],
      'lastname': [this.Lastname],
      'phone': [this.Phone],
      'email': [this.Email],
      'enrollmentDate': [this.EnrollmentDate],
      'expectedEndOfStudies': [this.ExpectedEndOfStudies],
      'studyingPrivilegeEnds': [this.StudyingPrivilegeEnds]
    });
  }

  updateProfile() {

    Object.assign(this.newValues, this.profileForm.value);
    var updatedProperties = Object.keys(this.newValues);
```

```
      var pushProperties = new Array();
      var pushValues = new Array();

      for (let i = 0; i < updatedProperties.length; i++) {
        if (this.profileForm.value[updatedProperties[i]] !== null) {
          pushProperties.push(updatedProperties[i]);
          console.log(updatedProperties[i]);
          pushValues.push(this.profileForm.value[updatedProperties[i]]);
          console.log(this.profileForm.value[updatedProperties[i]]);
        }
      }

      // Display detected changes in console
      if(pushProperties.length === 0) {
        console.log("No changes detected no patch message sent!");
      } else {
        console.log("Patched following properties:");
        console.log(pushProperties)

        this._profileService.updateProfile(pushProperties, pushValues)
        .subscribe(result => {
          console.log("Saved!");
          }, (err) => {
            console.log(err);
          }
        );
      }

      this.newValues = null;
      this.originalValues = null;
      updatedProperties = null;
      pushProperties = null;
      pushValues = null;
    }
  }
```

Liite 53. profile.service.ts

```
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs/Rx';
import { Http, Headers } from '@angular/http';
import 'rxjs/add/operator/map';

@Injectable()
export class ProfileService {

  private loggedIn = false;
  private token: string = localStorage.getItem('auth_token');
  private userId: number;

  ProfileForm: any;
  private patchMessage = new Array();

  constructor(private http: Http) {
     this.loggedIn = !!localStorage.getItem('auth_token');
  }

  getProfile() {
    let headers = new Headers();
    headers.append('Authorization', this.token);
    headers.append('Access-Control-Allow-Origin', 'GET');

    return this.http
    .get(
      'https://api.jiisoni.com/api/user/',
      { headers }
    )
    .map((res: any) => res.json())
    .map((res) => {
          return res;
       }
    )
  }

  updateProfile(patchPaths: string[], patchValues: any[]) {
    let headers = new Headers();
    headers.append('Content-Type', 'application/json');
    headers.append('Authorization', this.token);
    headers.append('Access-Control-Allow-Origin', 'POST');

    if (patchPaths.length === patchValues.length) {
```

```
        for(let i = 0; i < patchPaths.length; i++) {
          this.patchMessage.push(
            { op: "replace", path: patchPaths[i], value: patchValues[i] }
          );
        }
      }
      return this.http
      .patch(
        'https://api.jiisoni.com/api/user/',
        this.patchMessage,
        { headers }
      )
      .map((res: any) => res.json())
  }
}
```

Liite 54. comment.component.html

```
<div class="pure-g page-sub-header">
  <div class="pure-u-1-5">
    <div class="page-sub-name">
    Kommentit
    </div>
  </div>
  <div class="pure-u-4-5 page-sub-header">
    <div class="page-sub-menu">
      <button [routerLink]="['/thesis', thesisId]" type="button" class="pure-button"><i class="fa fa-history" aria-hid-
den="true"></i> Opinnäytetyö</button>
      <button [routerLink]="['/planner', thesisId]" type="button" class="pure-button"><i class="fa fa-history" aria-hid-
den="true"></i> Tapaamiset</button>
      <button [routerLink]="['/comments', thesisId]" type="button" class="pure-button"><i class="fa fa-history" aria-hid-
den="true"></i> Kommentit</button>
      <button [routerLink]="['/files', thesisId]" type="button" class="pure-button"><i class="fa fa-history" aria-hid-
den="true"></i> Tiedostot</button>
    </div>
  </div>
</div>
<div class="full-width-container">
  <div class="content-middle">


    <div class="table-row "><h2>Kommentit</h2></div>


    <ul id="comments">
      <li *ngIf="!hasComments">Ei kommentteja.</li>
      <li *ngFor="let comment of comments">
        <div class="comment-poster">{{ comment.CommenterName }}:</div>
        {{ comment.Comment }}<br />
```

```
            <div class="comment-timestamp">{{comment.CommentCreated | date:"dd/MM/yyyy - HH:mm" }}</div><br />
          </li>
        </ul>

        <form [formGroup]="commentForm" (ngSubmit)="addComment()">
          <input type="text" name="newComment" placeholder="Lisää kommentti" formControlName="newComment" />
          <div class="form-options">
            <button type="submit">Lähetä kommentti</button>
          </div>
        </form>
      </div>
    </div>
```

Liite 55. comment.component.ts

```
import { Component } from '@angular/core';
import { Router, ActivatedRoute } from '@angular/router';
import { CommentService }  from './comment.service';
import { Validators, FormBuilder } from '@angular/forms';

@Component({
  selector: 'application-comments',
  templateUrl: './comment.component.html'
})

export class CommentComponent {

  commentForm: any;
  private isLoggedIn: boolean = false;
  private thesisId: number;

  // Comment object properties
  private comments = new Array();
  private newComment: string;
  private hasComments: boolean = false;

  constructor (private commentService: CommentService, private router: Router, private route: ActivatedRoute, private form-
Builder: FormBuilder) {

    this.route.params.subscribe(params => {
      this.thesisId = +params['id'];
    });

    let result = this.commentService.getComments(this.thesisId).subscribe((result) => {
```

```
        for(let i = 0; i <  result.length; i++) {
          this.comments.push(
            {
              CommentId: result[i].commentId,
              Comment: result[i].comment,
              CommenterId: result[i].commenterId,
              CommenterName: result[i].commenterName,
              CommentCreated: result[i].commentCreated
            });
        }

        if(this.comments.length > 0) {
          this.hasComments = true;
        } else {
          this.hasComments = false;
        }
      })

      this.commentForm = this.formBuilder.group({
        'newComment': ['', Validators.required]
      });
    }

    addComment(username:string, password:string) {
      this.commentService.addComment(this.thesisId, this.commentForm.value.newComment).subscribe((result) => {
        if (result) {

        }
      });
    }
  }
```

Liite 56. comment.service.ts

```
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs/Rx';
import { Http, Headers } from '@angular/http';
import 'rxjs/add/operator/map';

@Injectable()
export class CommentService {

  private loggedIn = false;
  private token: string = localStorage.getItem('auth_token');

  commentForm: any;

  private patchMessage = new Array();

  constructor(private http: Http) {
    this.loggedIn = !!localStorage.getItem('auth_token');
  }

  getComments(id: number) {
    let headers = new Headers();
    headers.append('Authorization', this.token);
    headers.append('Access-Control-Allow-Origin', 'GET');

    return this.http
    .get(
      'https://api.jiisoni.com/api/comments/' + id.toString(),
      { headers }
    )
    .map((res: any) => res.json())
    .map((res) => {
        return res;
      }
    )
  }

  addComment(id: number, newComment: string) {
    let headers = new Headers();
    headers.append('Content-Type', 'application/json');
    headers.append('Authorization', this.token);
    headers.append('Access-Control-Allow-Origin', 'POST');
```

```
    return this.http
    .post(
      'https://api.jiisoni.com/api/comments/' + id.toString(),
      { comment: newComment },
      { headers }
    )
    .map((res: any) => res.json())
  }
}
```

Liite 57. document.service.ts

```
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs/Rx';
import { Http, Headers, RequestOptions } from '@angular/http';
import 'rxjs/add/operator/map';

@Injectable()
export class DocumentService {

  private token: string;

  constructor(private http: Http) {
    this.token = localStorage.getItem('auth_token');
  }

  getDocuments(thesisId: number) {
    let headers = new Headers();
    headers.append('Authorization', this.token);
    headers.append('Access-Control-Allow-Origin', 'GET');

    let authenticationService: string;

    return this.http
    .get(
      'https://api.jiisoni.com/api/document/' + thesisId.toString(),
      { headers }
    )
    .map((res: any) => res.json())
    .map((res) => {
        return res;
      }
    )
  }
```

```
  downloadFile(fileId: number) {
    let headers = new Headers();
    headers.append('Content-Type', 'application/json');
    headers.append('Authorization', this.token);
    headers.append('Access-Control-Allow-Origin', 'GET');

    return this.http
    .get(
      'https://api.jiisoni.com/api/document/key/' + fileId.toString(),
      { headers },
    ).map((res: any) => res.json())
    .map((res) => {
          return res;
       }
    )
  }
}
```

Liite 58. documents.component.html

```
<div class="pure-g page-sub-header">
  <div class="pure-u-1-5">
    <div class="page-sub-name">
    Tiedostot
    </div>
  </div>
  <div class="pure-u-4-5 page-sub-header">
    <div class="page-sub-menu">
      <button [routerLink]="['/thesis', thesisId]" type="button" class="pure-button"><i class="fa fa-history" aria-hid-
den="true"></i> Opinnäytetyö</button>
      <button [routerLink]="['/planner', thesisId]" type="button" class="pure-button"><i class="fa fa-history" aria-hid-
den="true"></i> Tapaamiset</button>
      <button [routerLink]="['/comments', thesisId]" type="button" class="pure-button"><i class="fa fa-history" aria-hid-
den="true"></i> Kommentit</button>
      <button [routerLink]="['/files', thesisId]" type="button" class="pure-button"><i class="fa fa-history" aria-hid-
den="true"></i> Tiedostot</button>
    </div>
  </div>
</div>
<div class="full-width-container">
  <div class="content-middle">
    <div class="important-notice">
      Tällä hetkellä tiedostojen lataaminen vaatii popup-ikkunoiden hyväksymisen selaimen asetuksista. Usein selain saattaa
oletuksena estää uuden ikkunan aukeamisen.
    </div>
    <form action="https://api.jiisoni.com/api/document/" method="post" enctype="multipart/form-data">
      Valitse tiedosto:
      <input type="file" name="fileToUpload" id="fileToUpload"><br />
      <input type="submit" value="Upload File" name="submit">
    </form>

    <table class="pure-table pure-table-striped">
      <thead>
        <th>Tiedosto</th>
        <th>Lisääjä</th>
        <th>Lisääjän rooli</th>
        <th>Lisätty</th>
        <th>Tiedoston koko</th>
        <th>Lataa</th>
      </thead>
      <tr *ngFor="let document of documents">
        <td>{{document.DocumentName}}</td>
        <td>{{document.DocumentCreatorName}}</td>
        <td>{{document.DocumentCreatorRole}}</td>
```

```
            <td>{{document.DocumentCreated | date:'dd / MM / yyyy - HH:mm'}}</td>
            <td></td>
            <td><span (click)="downloadFile(document.DocumentId)" class="span-link">Lataa</span></td>
          </tr>
        </table>
      </div>
    </div>
```

Liite 59. documents.component.ts

```
import { Component } from '@angular/core';
import { Router, ActivatedRoute } from '@angular/router';
import { Validators, FormBuilder } from '@angular/forms';
import { DocumentService } from './document.service';

@Component({
  selector: 'view-thesisFiles',
  templateUrl: './documents.component.html'
})

export class DocumentsComponent {
  private files = new Array();

  private thesisId: number;
  private file;
  private documents = new Array();

  constructor(private documentService: DocumentService, private router: Router, private route: ActivatedRoute) {
    // Get ThesisId from requested url
    this.route.params.subscribe(params => {
      this.thesisId = +params['id'];
    });

    let result = this.documentService.getDocuments(this.thesisId).subscribe((result) => {
      for (let i = 0; i < result.length; i++) {
        this.documents.push(
          {
            DocumentId: result[i].documentId,
            DocumentName: result[i].documentName,
            DocumentCreated: result[i].documentCreated,
            DocumentCreatorId: result[i].documentCreator.userId,
            DocumentCreatorName: result[i].documentCreator.name,
            DocumentCreatorRole: result[i].documentCreator.role,
          }
```

```
            )
          }
        })
    }

    downloadFile(documentId: number) {
        let result = this.documentService.downloadFile(documentId).subscribe(data => {
            window.open('https://api.jiisoni.com/api/download/?key=' + data.message);
        },
        err => console.log("Couldn't download file!"));
    }

}
```

Liite 60. thesis.component.html

```
<div class="pure-g page-sub-header">
    <div class="pure-u-1-5">
        <div class="page-sub-name">
        Opinnäytetyö
        </div>
    </div>
    <div class="pure-u-4-5 page-sub-header">
        <div class="page-sub-menu">
            <button [routerLink]="['/thesis', thesisId]" type="button" class="pure-button"><i class="fa fa-history" aria-hid-
den="true"></i> Opinnäytetyö</button>
            <button [routerLink]="['/planner', thesisId]" type="button" class="pure-button"><i class="fa fa-history" aria-hid-
den="true"></i> Tapaamiset</button>
            <button [routerLink]="['/comments', thesisId]" type="button" class="pure-button"><i class="fa fa-history" aria-hid-
den="true"></i> Kommentit</button>
            <button [routerLink]="['/files', thesisId]" type="button" class="pure-button"><i class="fa fa-history" aria-hid-
den="true"></i> Tiedostot</button>
        </div>
    </div>
</div>
<div class="full-width-container">
    <div class="content-middle">
    <h1>Opinnäytetyö: {{ ThesisNameFinnish }}</h1>

    <form [formGroup]="thesisForm" (ngSubmit)="updateThesis(thesisId)" class="pure-form pure-form-aligned">

        <div class="pure-control-group">
            <label for="ThesisNameFinnish">Opinnäytetyönnimi (Suomeksi)</label>
            <input type="text" value="{{ ThesisNameFinnish }}" formControlName="thesisNameFinnish" id="thesisNameFinnish"
/><br />
        </div>
```

```
    <div class="pure-control-group">
      <label for="ThesisNameEnglish">Opinnäytetyönnimi (Englanniksi)</label>
      <input type="text" value="{{ ThesisNameEnglish }}" formControlName="thesisNameEnglish" id="thesisNameEnglish"
/><br />
    </div>

    <div class="pure-control-group">
      <label for="FirstDirector">1. Ohjaaja</label>
      <select formControlName="firstDirector">
        <option *ngFor="let director of listOfDirectors"
            [ngValue]="director.directorId" [selected]="director.directorId === FirstDirector">{{ director.directorName }}</op-
tion>
      </select>
    </div>

    <div class="pure-control-group">
      <label for="SecondDirector">2. Ohjaaja</label>
      <select formControlName="secondDirector" [(ngModel)]="SecondDirector">
        <option *ngFor="let director of listOfDirectors"
            [ngValue]="director.directorId" [selected]="director.directorId === SecondDirector">{{ director.directorName
}}</option>
      </select>
    </div>

    <div class="pure-control-group">
      <label for="Orientation">Ohjaus</label>
      <input type="text" value="{{ Orientation }}" formControlName="orientation" id="orientation" /><br />
    </div>

    <div class="pure-control-group">
      <label for="Customer">Tilaaja</label>
        <input type="text" value="{{ Customer }}" formControlName="customer" id="customer" />
    </div>

    <div class="pure-control-group">
      <label for="LanguageDirectingProcess">Kielenohjauksen prosessi</label>
      <input type="radio" [(ngModel)]="LanguageDirectingProcess"
            [checked]="LanguageDirectingProcess == false" [value]="false"
            formControlName="languageDirectingProcess" id="languageDirectingProcess" /> Ei
      <input type="radio" [(ngModel)]="LanguageDirectingProcess"
            [checked]="LanguageDirectingProcess == true" [value]="true"
            formControlName="languageDirectingProcess" id="languageDirectingProcess" /> Kyllä
    </div>

    <div class="pure-control-group">
      <label for="EnglisSummaryApproved">Englanninkielinen kuvaus hyväksytty</label>
```

```
    <input type="radio" [(ngModel)]="EnglishSummaryApproved"
                [checked]="EnglisSummaryApproved == false" [value]="false"
                formControlName="englishSummaryApproved" id="englishSummaryApproved" /> Ei
    <input type="radio" [(ngModel)]="EnglishSummaryApproved"
                [checked]="EnglishSummaryApproved == true" [value]="true"
                formControlName="englishSummaryApproved" id="englishSummaryApproved" /> Kyllä
  </div>

  <div class="pure-control-group">
    <label for="PublishedInTheseus">Julkaistu Theseuksessa</label>
    <input type="radio" [(ngModel)]="PublishedInTheseus"
                [checked]="PublishedInTheseus == false" value="false"
                formControlName="publishedInTheseus" id="publishedInTheseus" /> Ei
    <input type="radio" [(ngModel)]="PublishedInTheseus"
                [checked]="PublishedInTheseus == true" value="true"
                formControlName="publishedInTheseus" id="publishedInTheseus" /> Kyllä
  </div>

  <div class="pure-control-group">
    <label for="UrkundResult">Urkund tulos</label>
    <input type="text" value="{{ UrkundResult }}" formControlName="urkundResult" id="urkundResult" />
  </div>

   <div class="pure-control-group">
  <label for="GradeSuggestion">Arvosana</label>
    <input type="text" value="{{ GradeSuggestion }}" formControlName="gradeSuggestion" id="gradeSuggestion" />
  </div>

  <div class="pure-control-group">
    <label for="StartingDate">Aloitus päivä</label>
    <input type="datetime" value="{{ StartingDate | date:'yMMMMd'}}" formControlName="startingDate" id="startingDate"
/>
  </div>

  <div class="pure-control-group">
    <label for="ThesisSubjectAgreementDate">Aihe hyväksytty pvm.</label>
      <input type="datetime" value="{{ ThesisSubjectAgreementDate | date:'yMMMMd'}}" formControlName="thesisSub-
jectAgreementDate" id="thesisSubjectAgreementDate" />
  </div>

  <div class="pure-control-group">
    <label for="ThesisFinishedDate">Työ valmistunut pvm.</label>
    <input type="datetime" value="{{ ThesisFinishedDate | date:'yMMMMd'}}" formControlName="thesisFinishedDate"
id="thesisFinishedDate" />
  </div>

  <button type="submit" class="pure-button"><i class="fa fa-floppy-o" aria-hidden="true"></i> TALLENNA</button>
```

```
  </form>
  </div>
</div>
```

Liite 61. thesis.component.ts

```
import { Component } from '@angular/core';
import { Router, ActivatedRoute } from '@angular/router';
import { Validators, FormBuilder } from '@angular/forms';
import { ThesisService } from './thesis.service';

@Component({
  selector: 'view-thesis',
  templateUrl: './thesis.component.html'
})

export class ThesisComponent {

  private ThesisNameFinnish: string;
  private ThesisNameEnglish: string;
  private FirstDirector: number;
  private SecondDirector: number;
  private Orientation: string;
  private Customer: string;
  private StartingDate: Date;
  private ThesisSubjectAgreementDate: Date;
  private LanguageDirectingProcess: number;
  private EnglishSummaryApproved: boolean;
  private UrkundResult: number;
  private PublishedInTheseus: boolean;
  private GradeSuggestion: number;
  private ThesisFinishedDate: Date;

  thesisForm: any;

  private listOfDirectors = new Array;

  private originalValues = new Array;
  private newValues = new Array;

  private isPublished: boolean;
  private thesisId: number;

  constructor (private _thesisService: ThesisService, private router: Router, private route: ActivatedRoute, private formBuilder:
FormBuilder) {
```

```
// Get ThesisId from requested url
this.route.params.subscribe(params => {
   this.thesisId = +params['id'];
});



let result = this._thesisService.getThesis(this.thesisId).subscribe((result) => {

   Object.assign(this.originalValues, result);

   this.ThesisNameFinnish = result.thesisNameFinnish;
   this.ThesisNameEnglish = result.thesisNameEnglish;
   this.FirstDirector = result.firstDirector;
   this.SecondDirector = result.secondDirector;
   this.Orientation = result.orientation;
   this.Customer = result.customer;
   this.StartingDate = result.startingDate;
   this.ThesisSubjectAgreementDate = result.thesisSubjectAgreementDate;
   this.LanguageDirectingProcess = result.languageDirectingProcess;
   this.EnglishSummaryApproved = result.englishSummaryApproved;
   this.UrkundResult = result.urkundResult;
   this.PublishedInTheseus = result.publishedInTheseus;
   this.GradeSuggestion = result.gradeSuggestion;
   this.ThesisFinishedDate = result.thesisFinishedDate;
});

let directors = this._thesisService.getDirectors().subscribe((result) => {
   this.listOfDirectors = result;
   this.listOfDirectors.push({
      directorId: 0,
      directorName: "Ei valittu"});
});

if(this.FirstDirector === null) {
   this.FirstDirector = 0;
}
if(this.SecondDirector === null) {
   this.SecondDirector = 0;
}

// Form
this.thesisForm = this.formBuilder.group({
   'thesisNameFinnish': [this.ThesisNameFinnish],
   'thesisNameEnglish': [this.ThesisNameEnglish],
   'firstDirector': [this.FirstDirector],
   'secondDirector': [this.SecondDirector],
```

```
      'orientation': [this.Orientation],
      'customer': [this.StartingDate],
      'startingDate': [this.StartingDate],
      'thesisSubjectAgreementDate': [this.ThesisSubjectAgreementDate],
      'languageDirectingProcess': [this.LanguageDirectingProcess],
      'englishSummaryApproved': [this.EnglishSummaryApproved],
      'urkundResult': [this.UrkundResult],
      'publishedInTheseus': [this.PublishedInTheseus],
      'gradeSuggestion': [this.GradeSuggestion],
      'thesisFinishedDate': [this.ThesisFinishedDate]
    });
  }

  updateThesis(thesisId) {

    Object.assign(this.newValues, this.thesisForm.value);
    var updatedProperties = Object.keys(this.newValues);

    var pushProperties = new Array();
    var pushValues = new Array();

    for (let i = 0; i < updatedProperties.length; i++) {
      if (this.thesisForm.value[updatedProperties[i]] !== null) {
        pushProperties.push(updatedProperties[i]);
        console.log(updatedProperties[i]);
        pushValues.push(this.thesisForm.value[updatedProperties[i]]);
        console.log(this.thesisForm.value[updatedProperties[i]]);
      }
    }

    // Display detected changes in console
    if(pushProperties.length === 0) {
      console.log("No changes detected no patch message sent!");
    } else {
      console.log("Patched following properties:");
      console.log(pushProperties)

      this._thesisService.updateThesis(thesisId, pushProperties, pushValues)
      .subscribe(result => {
        console.log("Saved!");
        }, (err) => {
          console.log(err);
        }
      );
    }
```

```
      this.newValues = null;
      this.originalValues = null;
      updatedProperties = null;
      pushProperties = null;
      pushValues = null;
   }
}
```

Liite 62. thesis.service.ts

```
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs/Rx';
import { Http, Headers } from '@angular/http';
import 'rxjs/add/operator/map';

@Injectable()
export class ThesisService {

   private loggedIn = false;
   private token: string = localStorage.getItem('auth_token');
   private thesisId: number;

   constructor(private http: Http) {
      this.loggedIn = !!localStorage.getItem('auth_token');
   }

   getThesis(thesisId: number) {

      let headers = new Headers();

      headers.append('Authorization', this.token);
      headers.append('Access-Control-Allow-Origin', 'GET');

      return this.http
      .get(
         'https://api.jiisoni.com/api/thesis/' + thesisId.toString(),
         { headers }
      )
      .map((res: any) => res.json())
      .map((res) => {
            return res;
         }
      )
   }
```

```
getDirectors() {

  let headers = new Headers();

  headers.append('Authorization', this.token);
  headers.append('Access-Control-Allow-Origin', 'GET');

  return this.http
  .get(
    'https://api.jiisoni.com/api/thesis/directors',
    { headers }
  )
  .map((res: any) => res.json())
  .map((res) => {
        return res;
     }
  )
}

updateThesis(thesisId: number, patchPaths: string[], patchValues: any[]) {

  let patchMessage = new Array();
  let headers = new Headers();

  headers.append('Content-Type', 'application/json');
  headers.append('Authorization', this.token);
  headers.append('Access-Control-Allow-Origin', 'POST');

  if (patchPaths.length === patchValues.length) {
     for(let i = 0; i < patchPaths.length; i++) {
        patchMessage.push(
          { op: "replace", path: patchPaths[i], value: patchValues[i] }
        );
     }
  }
  return this.http
  .patch(
    'https://api.jiisoni.com/api/thesis/' + thesisId.toString(),
    patchMessage,
    { headers }
  )
  .map((res: any) => res.json())
}

// Lists all documents available to user
```

```
  // Returns an array of documents
  listDocuments(thesisId: number) {
    let headers = new Headers();
    headers.append('Content-Type', 'application/json');
    headers.append('Authorization', this.token);
    headers.append('Access-Control-Allow-Origin', 'GET');

    return this.http
     .get(
       'https://api.jiisoni.com/api/documents/' + thesisId.toString(),
       { headers }
     )
     .map((res: any) => res.json())
     .map((res) => { return res; }
    )
  }

}
```

Liite 63. app.component.html

```
<div class="view">
  <div class="view-content">
    <router-outlet></router-outlet>
  </div>
</div>
```

Liite 64. app.component.ts

```
import { Component } from '@angular/core';
import { ActivatedRoute, Router, NavigationStart, NavigationEnd, NavigationError, NavigationCancel, RoutesRecognized } from
'@angular/router'

@Component({
  selector: 'application',
  templateUrl: './app.component.html'
})
export class AppComponent {
  public currentPage: string;
  constructor(public router: ActivatedRoute) {
    console.log(router.url);
  }

  ngOnInit() {
  this.router.params.subscribe(params => {
    if (params['id']) {
```

```
        this.currentPage = "Opinnäytetyö";
      }
    });
    }
}
```

Liite 65. app.module.ts

```
import { NgModule }      from '@angular/core';
import { BrowserModule }  from '@angular/platform-browser';
import { routing }       from './app.routing';
import { HttpModule }    from '@angular/http';
import { FormsModule, ReactiveFormsModule }   from '@angular/forms';

import { AppComponent }           from './app.component';
import { AuthenticationComponent }  from './common/authentication/authentication.component';
import { FooterComponent }        from './common/footer/footer.component';
import { HeaderComponent }        from './common/header/header.component';
import { CommentComponent }       from './thesis/comments/comment.component';
import { HomeComponent }          from './home/home.component';
import { AboutComponent }         from './about/about.component';
import { ThesisComponent }        from './thesis/thesis.component';
import { DocumentsComponent }     from './thesis/documents/documents.component';
import { PlannerComponent }       from './thesis/planner/planner.component';
import { ProfileComponent }       from './profile/profile.component';
import { OverviewComponent }      from './overview/overview.component';
import { AccessControlComponent }   from './access-control/access-control.component';

import { UserService }            from './common/services/user.service';
import { OverviewService }        from './common/services/overview.service';
import { ThesisService }          from './thesis/thesis.service';
import { ProfileService }         from './profile/profile.service';
import { CommentService }         from './thesis/comments/comment.service';
import { ProtectGuard }           from './common/services/protect.guard';
import { DocumentService }        from './thesis/documents/document.service';
import { AccessControlService }   from './access-control/access-control.service';

@NgModule({
  imports: [
    BrowserModule,
    routing,
    HttpModule,
    FormsModule,
    ReactiveFormsModule
  ],
```

```
  declarations: [
    AppComponent,
    AuthenticationComponent,
    FooterComponent,
    HeaderComponent,
    HomeComponent,
    AboutComponent,
    OverviewComponent,
    ThesisComponent,
    DocumentsComponent,
    PlannerComponent,
    ProfileComponent,
    AccessControlComponent,
    CommentComponent
  ],
  bootstrap: [
    AppComponent,
    FooterComponent,
    HeaderComponent
  ],
  providers: [
    UserService,
    ThesisService,
    ProfileService,
    DocumentService,
    OverviewService,
    AccessControlService,
    CommentService,
    ProtectGuard
  ]
})
export class AppModule { }
```

Liite 66. app.routing.ts

```
import { ModuleWithProviders }  from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { HomeComponent }       from './home/home.component';
import { AboutComponent }      from './about/about.component';
import { ThesisComponent}      from './thesis/thesis.component';
import { DocumentsComponent }   from './thesis/documents/documents.component';
import { ProfileComponent }     from './profile/profile.component';
import { PlannerComponent }     from './thesis/planner/planner.component';
import { CommentComponent }     from './thesis/comments/comment.component';
import { OverviewComponent }    from './overview/overview.component';
import { AccessControlComponent }   from './access-control/access-control.component';

import { UserService }         from './common/services/user.service';
import { ProtectGuard }        from './common/services/protect.guard';

const appRoutes: Routes = [
  {
    path: '',
    redirectTo: '/home',
    pathMatch: 'full'
  },
  {
    path: 'home',
    component: HomeComponent
  },
  {
    path: 'overview',
    component: OverviewComponent,
    canActivate: [ ProtectGuard]
  },
  {
    path: 'thesis/:id',
    component: ThesisComponent,
    canActivate: [ ProtectGuard ]
  },
  {
    path: 'profile/:id',
    component: ProfileComponent,
    canActivate: [ProtectGuard]
  },
  {
    path: 'planner/:id',
```

```
      component: PlannerComponent,
      canActivate: [ ProtectGuard ]
    },
    {
      path: 'comments/:id',
      component: CommentComponent,
      canActivate: [ ProtectGuard ]
    },
    {
      path: 'files/:id',
      component: DocumentsComponent,
      canActivate: [ ProtectGuard ]
    },
    {
      path: 'accesscontrol',
      component: AccessControlComponent,
      canActivate: [ ProtectGuard ]
    },
    {
      path: 'about',
      component: AboutComponent
      //canActivate: [ ProtectGuard ]
    }
];

export const routing: ModuleWithProviders = RouterModule.forRoot(appRoutes);
```

Liite 67. index.html

```
<!doctype html>
<html>
  <head>
    <base href="/">
    <meta charset="utf-8">
    <title>Opinnäytetöiden seurantajärjestelmä</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="styles.css">
    <link rel="stylesheet" href="assets/css/grids-min.css">
    <link rel="stylesheet" href="assets/css/buttons-min.css">
    <link rel="stylesheet" href="assets/css/tables-min.css">
    <link rel="stylesheet" href="assets/css/forms-min.css">
    <link href="https://fonts.googleapis.com/css?family=Alegreya+Sans:400,500" rel="stylesheet">
    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.6.3/css/font-awesome.min.css" rel="stylesheet">
  </head>

  <body>
    <application-header></application-header>
    <application></application>
    <application-footer></application-footer>
  </body>
</html>
```

Liite 68. styles.css

```
body {
  margin: 0;
  font-family: 'Alegreya Sans', Verdana;
}
nav {
  color: #FFF;
  padding: 10px;
  background-color: #2c3e50;
}
.page-sub-name {
  font-weight: 600;
  font-size: 1.5em;
  padding: 15px 10px 15px 10px;
}
.page-sub-menu {
  padding: 15px 10px 15px 10px;
}
.page-sub-header {
  border-width: 0 0 1px 0;
  border-color: #CCC;
  border-style: solid;
}
.login-area {
  text-align: right;
}
.planner-head {
  padding: 10px;
  background-color: #3582ce;
  color: #FFFFFF;
}
```