Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2018

Antti Laurila

TUNNISTAUTUMINEN OPENTUNTIIN MICROSOFT GRAPH APIN AVULLA

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikan koulutus

2018 | 31 sivua, 0 liitesivua

Antti Laurila

TUNNISTAUTUMINEN OPENTUNTIIN MICROSOFT GRAPH APIN AVULLA

Tämän opinnäytetyön tarkoituksena oli laatia käyttäjän tunnistautumisjärjestelmä web-sovellus Opentuntiin käyttäen Microsoftin Graph API -rajapintaa. Rajapinnan kautta web-sovellus saa tarvittavat käyttäjän tiedot, ja niillä voidaan suorittaa kirjautuminen tai luoda kokonaan uusi käyttäjä sovellukseen.

Lopputuloksena oli toimiva tunnistautumisjärjestelmä, joka helpotti käyttäjää ja teki tunnistautumisprosessista turvallisemman ja yksinkertaisemman loppukäyttäjälle. Se myös mahdollistaa uusien ominaisuuksien lisäämisen tulevaisuudessa käyttäen Graph API:a ja Azure AD:ta helpommin.

ASIASANAT:

Microsoft, Azure AD, OAuth 2.0, Graph API, Opentunti

BACHELOR´S / MASTER'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Bachelor of Engineering, Information and Communications Technology

2018 | 31 pages, 0 pages in appendices

Antti Laurila

OPENTUNTI AUTHENTICATION WITH MICROSOFT GRAPH API

The purpose of this thesis was to implement a user authentication system into the Opentunti web application using a Microsoft Graph API -interface. The web application acquires user data through the interface to verify the authentication or create a new user.

The final result was a working authentication system which reduced the user interaction and made the authentication process safer and simpler for the end user. It also established a base for working with the Graph API and the Azure AD which can be used in future features in the same work project.

KEYWORDS:

Microsoft, Azure AD, OAuth 2.0, Opentunti

SISÄLTÖ

KUVAT

TAULUKOT

KÄYTETTY SANASTO

AD	Active Directory, Microsoftin käyttäjä- ja hakemistotietokanta.
API	Ohjelmointirajapinta (application programming interface).
Azure	Microsoftin alustapalvelu pilvessä, minkä tarkoituksena on yhdistää kaikki sen palvelut samaan paikkaan.
GET	Yksi HTTP:n pyyntömetodeista. Muita ovat esimerkiksi POST, PATCH ja DELETE.
HTTP	Hypertext Transfer Protocol, protokolla tiedon lähettämiseen selaimen ja palvelimen välillä.
OAuth	Avoin standardi päästää muut palveluiden tietoihin ilman erillistä kirjautumista.
PHP	PHP: Hypertext Preprocessor, ohjelmointikieli web-sivujen ja -sovellusten luontiin.

1 JOHDANTO

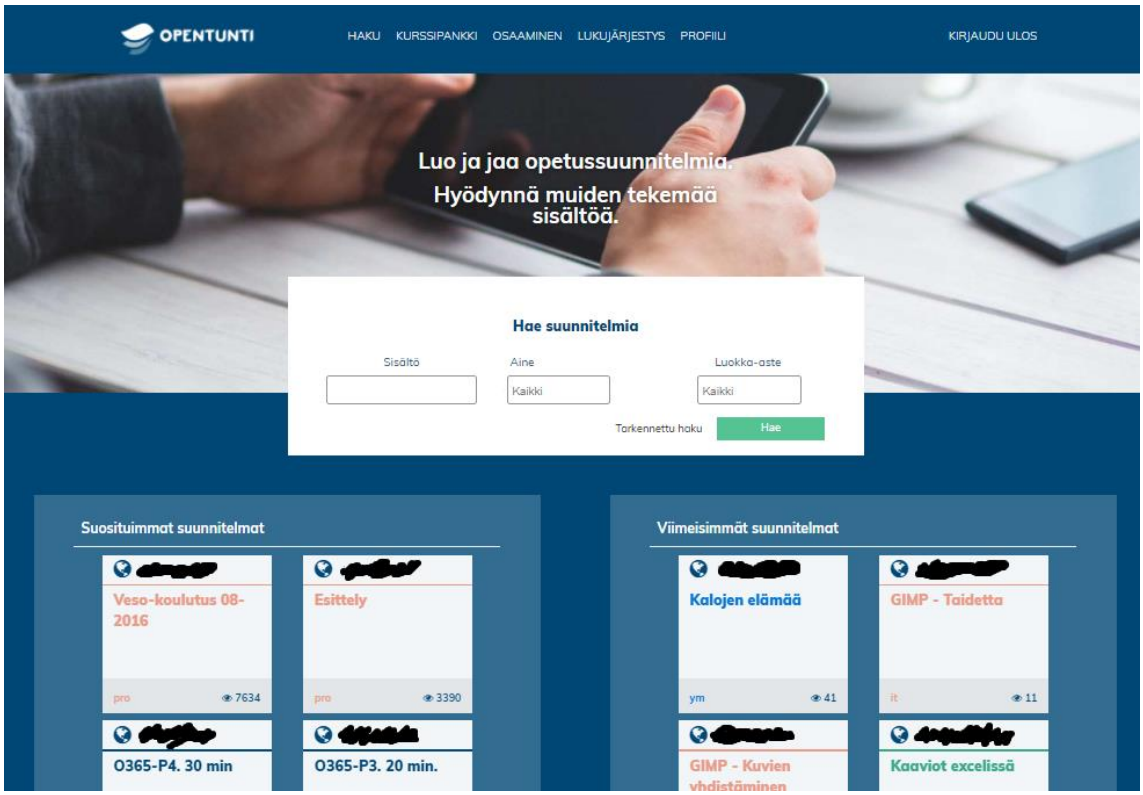
Käyttäjäystävällisyys on tärkeä osa web-sovelluksen kehittämistä ja käyttöä. Tällä tarkoitetaan sitä, että sovelluksen tulisi olla mahdollisimman helppokäyttöinen ja intuitiivinen loppukäyttäjälle. Yksi osa käyttäjäystävällisyyttä on oikeanlainen kirjautuminen tai tunnistautuminen sovellukseen. Liian monimutkainen kirjautuminen aiheuttaa käyttäjille harmaita hiuksia, eikä kukaan pidä ylimääräisistä vaiheista. Liian yksinkertainen kirjautuminen toisaalta aiheuttaa usein ongelmia tietoturvassa ja vaarana on, että käyttäjän tietoja päätyy vieraisiin käsiin.

Tämän projektin tarkoituksena oli löytää juuri se helpoin ja käyttäjäystävällisin, mutta toisaalta myös turvallinen kirjautumistapa web-sovellus Opentuntiin. Koska Opentuntia käyttää lähes pelkästään opettajat sekä oppilaat, yhdeksi sellaiseksi osoittautui Microsoft Graph API, jota hyödyntämällä käyttäjästä saadaan tarvittavat tunnistustiedot. Tämä taas vaatii Microsoftin Azure AD:n (Active Directory) OAuth 2.0 -auktorisointiprotokollan käyttöä, joka antaa Opentunnille luvan hakea tietoja Graph API:n kautta. Näin Opentunti tietää, kuka käyttäjä on, ja päästää hänet kirjautumaan sovellukseen. Molemmista tekniikoista on olemassa useita toteutustapoja ja versiota, mutta tässä työssä keskitytään lähinnä authorization code grant -tyyppiin, jonka käyttö on kuvattu vaihe vaiheelta luvussa 4.2 sekvenssiokaavion avulla.

Lopputuloksessa huomioitavia asioita ovat muun muassa skaalautuvuus ja uusien käyttäjien lisäys eli asentamisen helppous.

2 OPENTUNTI

Opentunti on opettajille sekä oppilaille tarkoitettu työkalu, jolla opettajat voivat suunnitella yksittäisiä oppitunteja tai kokonaisia kursseja (Kuva 1). Eroten muista tämänkaltaisista työkaluista, kuten Optima tai Moodle, Opentunti keskittyy tiedon jakamiseen opettajien välillä, olivat he sitten samassa koulussa tai samalla paikkakunnalla tai eivät. Opentunti mahdollistaa muiden käyttäjien sisältöjen hakemisen ja niiden kopioimisen osaksi omia sisältöjä, jolloin opettajien tuntien suunnittelun työmäärä keventyy ja opettajat voivat paremmin keskittyä itse opettamiseen.



Kuva 1. Opentunnin etusivu (Opentunti 2018).

2.1 Historia ja tarkoitus

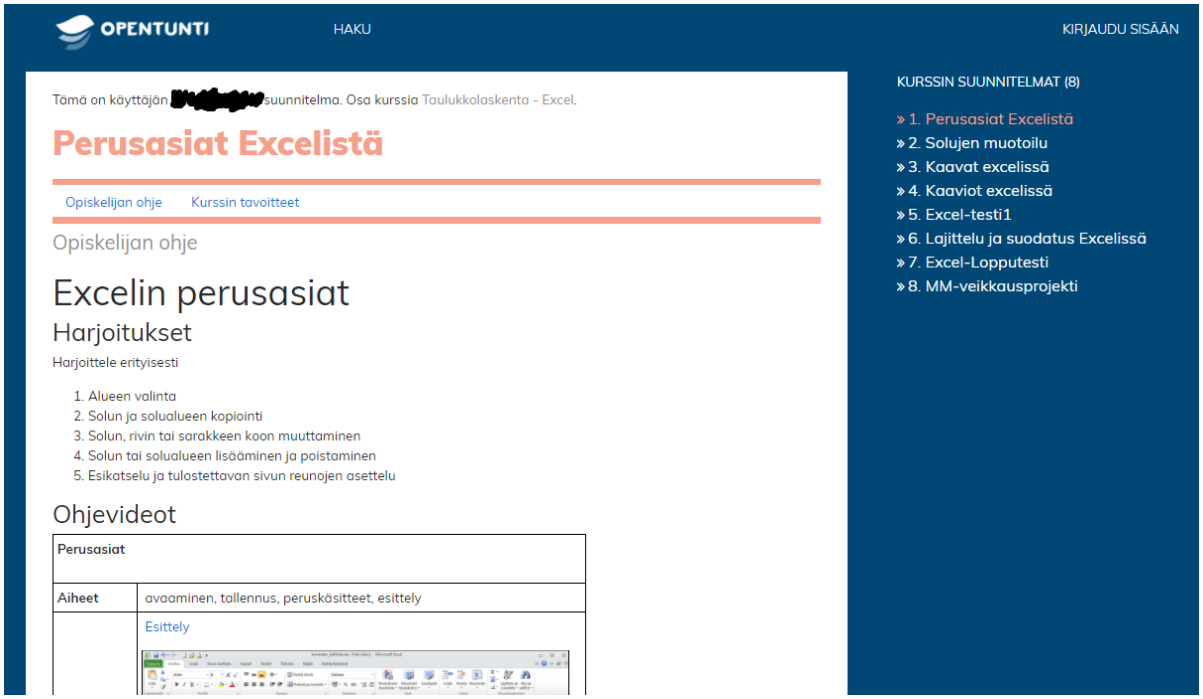
Opentunnin kehitys aloitettiin vuonna 2014. Tarkoitus oli silloin sama kuin nytkin: vähentää opettajien tekemää turhaa ja samaa työtä eli tuntien suunnittelua.

> Pelkästään perusopetuksessa on noin 43 000 opettajaa ja noin 1 000 000 viikoittaista oppituntia. Jos oppituntien suunnitteluun ja kirjaamiseen varataan n. 5 minuuttia, käytetään työhön yhteensä n. 80 000 tuntia viikossa. Opentunti luo opettajille mahdollisuuden kerätä suunnitelmia yhteiseen helposti haettavaan kanavaan suoraan omasta arjesta (Opentuntiry 2018.)

Opentuntiin alettiin lisäämään tuntisuunnittelun lisäksi uusia ominaisuuksia, jotka kaikki hyödyttävät opettajia. Lisäksi oppilaat päästettiin myös Opentuntiin ja heillekin tuli hyödyllisiä ominaisuuksia, kuten lukujärjestys.

2.2 Ominaisuudet

Opentunnin tärkein ominaisuus on kurssikokonaisuuksien ja yksittäisten tuntisuunnitelmien suunnittelu (Kuva 2) sekä niiden jakaminen muille. Tämän lisäksi Opentunti sisältää muun muassa integroinnin Wilmaan, jonka lukujärjestykseen voi liittää omia tuntisuunnitelmia ja osaamismerkit, joilla voidaan selvittää tietyn asian tai aihealueen osaamisen taso koulun tai paikkakunnan sisällä.



Kuva 2. Esimerkki Opentunnin tuntisuunnitelmasta (Opentunti 2018).

3 KIRJAUTUMINEN JA TUNNISTAUTUMINEN VERKOSSA

Suurin osa web-sovelluksista käyttää jonkinlaista käyttäjän tunnistautumisjärjestelmää, jotta tiedetään, kuka käyttäjä on ja mitä oikeuksia hänellä sovelluksessa on. Esimerkiksi Facebook, kuten suurin osa muistakin web-sovelluksista, käyttää yleisintä tunnistautumistapaa eli käyttäjätunnusta sekä salasanaa. Käyttäjätunnus on yleensä joko käyttäjän sähköpostiosoite, nimi tai osa nimeä tai itse valittu tunniste. Salasana on käyttäjän henkilökohtainen salainen merkkijono, johon yleensä suositellaan käytettävän sekä pieniä että isoja kirjaimia, numeroita sekä erikoismerkkejä. Käyttäjä syöttää käyttäjätunnuksen ja salasanan sovellukseen, ne vahvistetaan sovelluksen puolella ja käyttäjä päästetään sisään sovellukseen oikeilla tiedoilla.

Ongelma tässä tunnistautumistavassa on se, että käyttäjä joutuu olemaan vuorovaikutuksessa sovelluksen tunnistautumisen kanssa eli muistamaan oman käyttäjätunnus-salasana -yhdistelmän sekä suorittamaan kirjautumisen. Tämä voidaan kokea turhana ja vaivalloisena vaiheena, varsinkin jos käyttäjä joutuu kirjautumaan moneen paikkaan ja sovellukseen. Toisin sanoen se ei ole kauhean käyttäjäystävällistä.

Tässä työssä oleva web-sovellus, Opentunti, on juuri tällainen. Käyttäjäkunta koostuu lähes ainoastaan kuntien ja kaupunkien opettajista ja oppilaista. Tavallisessa kouluympäristössä, kuten juuri vaikkapa Turun ammattikorkeakoulussa, käyttäjä kirjautuu paikkakunnan tai koulun AD-tunnuksilla tietokoneelle sisään. Tämä riittää jo käyttäjän tunnistamiseen, joten kaikista ylimääräisistä vaiheista olisi suotavaa päästä eroon.

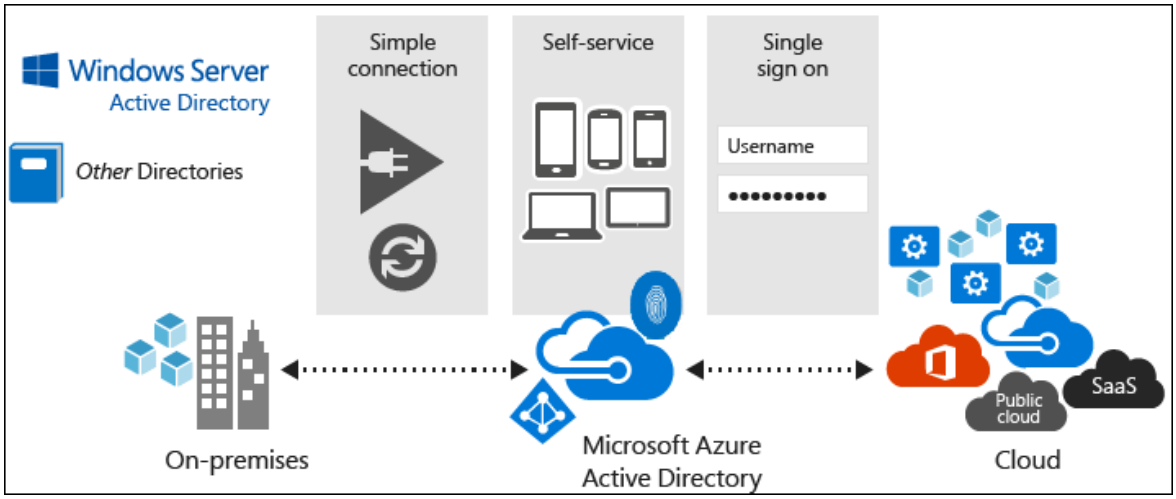
Suurimmalla osalla Suomen paikkakunnista ja kouluista on tietotekniikan osalta taustalla Microsoft, jonka AD-tilejä käyttäjät käyttävät. Tämä mahdollistaa OAuth 2.0 -auktorisoinnin eli luvan antamisen Opentunnille käyttää käyttäjän AD-tilin tietoja. Tämä on oikein rakennettuna kokonaan tai lähes kokonaan automaattinen tapa saada käyttäjältä vaadittavat tiedot tunnistautumiseen, joten se sopii täydellisesti Opentuntiin.

4 AZURE AD OAUTH 2.0 -AUKTOROINTIPROTOKOLLA

Azure AD Oauth 2.0 on Microsoftin kehittämä auktorointiprotokolla, joka pohjautuu avoimeen OAuth 2.0 -protokollastandardiin. OAuthista on useita toteutustapoja, mutta tässä työssä keskitytään vain käytössä olevaan authorization code grant -toteutukseen.

4.1 Azure AD -hakemistoalusta

Azure AD eli Azure Active Directory on Microsoftin kehittämä hakemistoalusta pilvessä, johon organisaatiot voivat tallentaa käyttäjien sekä muiden resurssien tietoja. Azure AD on korvannut lähes kokonaan paikallisen AD:n vieden sen kokonaan pilveen/verkkoon. Azure AD on useimmiten käytössä yritysten ja paikkakuntien IT-infrastruktuurin ytimenä (Kuva 3) halliten käyttäjiä sekä niiden oikeuksia eri sovelluksiin keskitettyjen sääntöjen ja toimintatapojen (policy) avulla (Microsoft Docs, 2018).

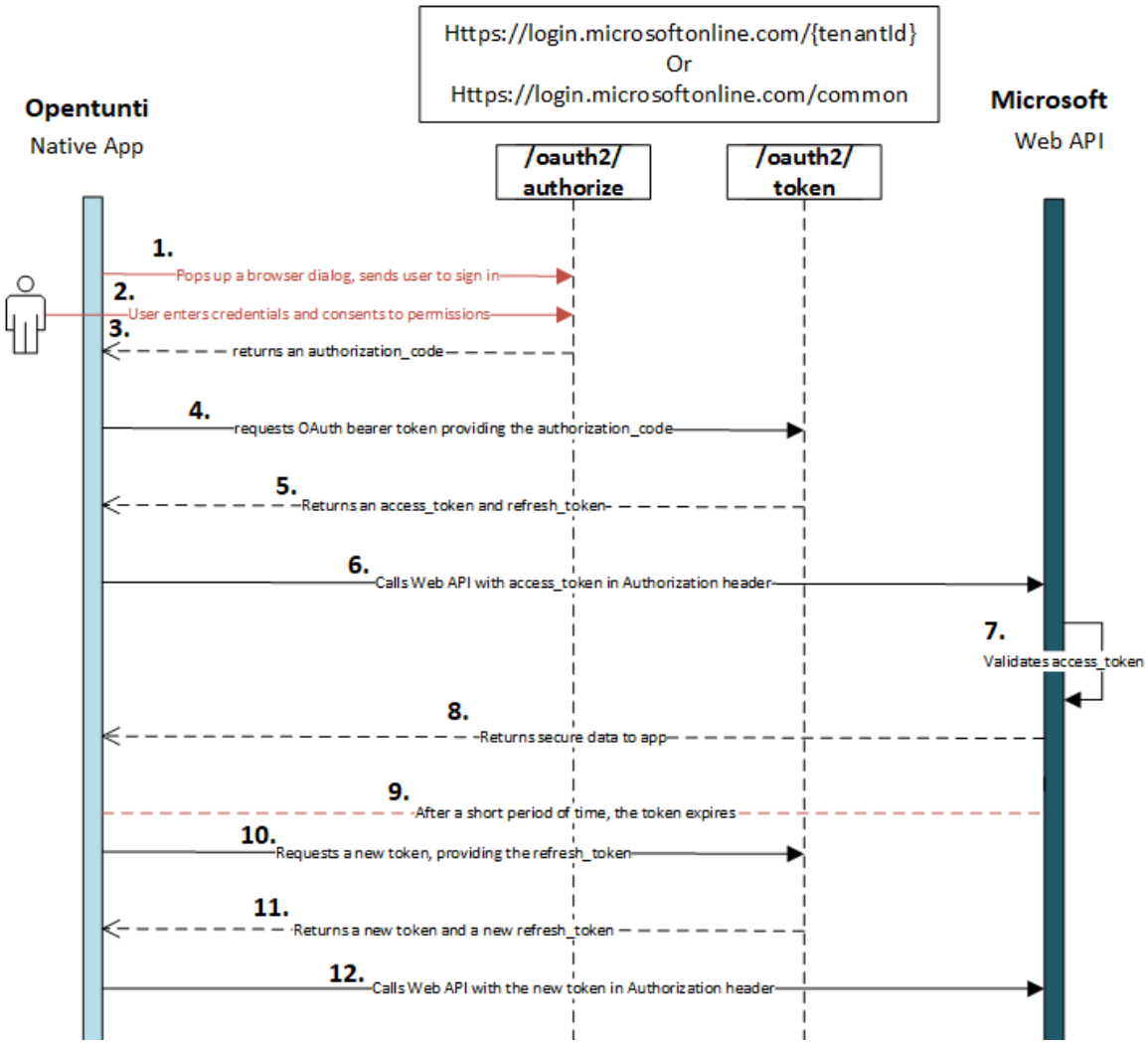


Kuva 3. Azure AD:n toiminta IT-infrastruktuurin ytimenä (Microsoft Docs 2018).

4.2 Authorization code grant -toteutustapa

Authorization code grant on yksi OAuth 2.0 -protokollan toteutustavoista, ja se on myös niistä kaikkein käytetyin esimerkiksi web-sovelluksissa. Muihin grant-tyyppeihin verrattuna authorization code grant on turvallisempi, sillä siinä on yksi ylimääräinen vaihe. Tämä varmistaa sen, ettei kukaan ulkopuolinen pääse sovelluksen ja OAuth-palvelimen väliin sieppaamaan käyttöavainta (access token) (Okta Developer, 2018).

Authorization code grant toimii niin, että OAuth-palvelimelta pyydetään auktorisointikoodi (authorization code, kuten grantin nimestä voi päätellä), joka taas vaihdetaan palvelimella erikseen käyttöavaimeen. Käyttöavain antaa tietyt oikeudet käyttäjän puolesta sovellukselle, esimerkiksi hakea käyttäjän tietoja (Kuva 4).



Kuva 4. Sekvenssikaavio OAuth 2.0 athorization code grant -tunnistautumisesta ja Graph API -rajapinnan käytöstä (Microsoft Docs 2018).

Vaihe vaiheelta käyttöavaimen hakeminen toimii näin:

1. **Sovellus ohjaa käyttäjän kirjautumisikkunaan OAuth-palvelimelle.** Tässä työssä käytetään Microsoftin palvelinta. Web-sovellusten tapauksessa käyttäjä vain ohjataan kirjautumissivulle, muissa tapauksissa yleensä sovellus avaa kokonaan uuden oman ikkunan.
2. **Käyttäjä syöttää tunnuksensa, kirjautuu OAuth-palvelimelle sisään ja antaa sovellukselle luvan käyttää ennaltamääritettyjä oikeuksia.** Tässä työssä käyttäjä on yleensä jo kirjautunut OAuth-palvelimelle tässä kohtaa, ja käyttäjän paikkakunnan IT-tukihenkilö on käynyt antamassa luvan oikeuksille kaikkien

käyttäjien puolesta osana sovelluksen asentamista, joten tämä vaihe ohitetaan kokonaan.

3. **OAuth-palvelin palauttaa sovellukselle auktorisointikoodin.**
4. **Sovellus pyytää OAuth-palvelimelta käyttöavainta auktorisointikoodilla.** Tämä OAuth-palvelimen päätepiste on eri kuin kohdassa 1.
5. **OAuth-palvelin palauttaa sovellukselle käyttöavaimen** ja tarvittaessa päivitysavaimen (refresh token). Käyttöavaimet ovat usein hyvin lyhytkestoisia (esimerkiksi yksi tunti), joten päivitysavaimella voidaan hakea uusi käyttöavain. Tässä työssä sitä ei kuitenkaan tarvita.

OAuth 2.0 tunnistautuminen on tässä kohtaa valmis. Sekvenssikaavion (Kuva 4) loppuosa on Graph API -rajapinnan (tarkemmin seuraavassa kappaleessa) sekä päivitysavaimen käyttöä:

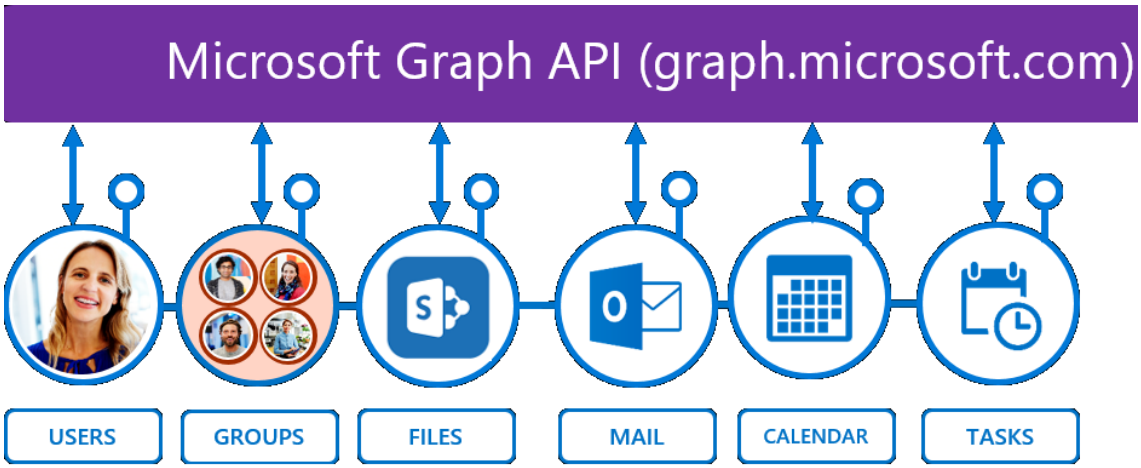
6. **Sovellus kutsuu rajapintaa käyttöavaimella.**
7. **Rajapinta vahvistaa käyttöavaimen.**
8. **Rajapinta palauttaa vastauksen sovellukselle.** Vastaus sisältää esimerkiksi pyydetyt tiedot tai vahvistuksen, että tietoja on lisätty tai muokattu.
9. **Käyttöavain vanhenee.** Yleisin aika on yksi tunti.
10. **Sovellus pyytää OAuth-palvelimelta uuden käyttöavaimen päivitysavaimella.** Päivitysavaimen käyttö nopeuttaa uuden voimassaolevan käyttöavaimen saamista paljon: vaiheet 1-3 voidaan ohittaa kokonaan. Tämä on tietysti mahdollista vain, jos päivitysavain on käytössä.
11. **OAuth-palvelin palauttaa sovellukselle uuden käyttöavaimen ja uuden päivitysavaimen.**
12. **Sovellus kutsuu rajapintaa uudella käyttöavaimella.**

Tämän jälkeen vaiheet toistuvat kuudennesta vaiheesta eteenpäin tarvittaessa.

5 MICROSOFT GRAPH API -RAJAPINTA

Microsoft Graph API on Microsoftin julkaisema rajapinta, jolla kehittäjät voivat olla yhteydessä Microsoftin pilvipalveluiden, kuten Exchangen, OneDriven, SharePointin, OneNoten ja Azure AD:n, tietojen kanssa (Kuva 5). Tietoja voidaan hakea, päivittää ja lisätä. Esimerkkejä näiden palveluiden tiedoista ovat käyttäjät, ryhmät, sähköposti, viestit sekä kalenteri. Graph API kokoaa kaiken tämän yhden päätepisteen (endpoint) taakse.

Rajapinnan käyttöön tarvitaan ainoastaan pääsyavain (access token), oikea kysely (query) ja osaan kyselyistä parametrejä.



Kuva 5. Kuvaus Graph API -rajapinnan yhteydestä Microsoftin pilvipalveluihin ja niiden tietoihin (Microsoft Partner Network 2018).

5.1 Rajapinnan historia

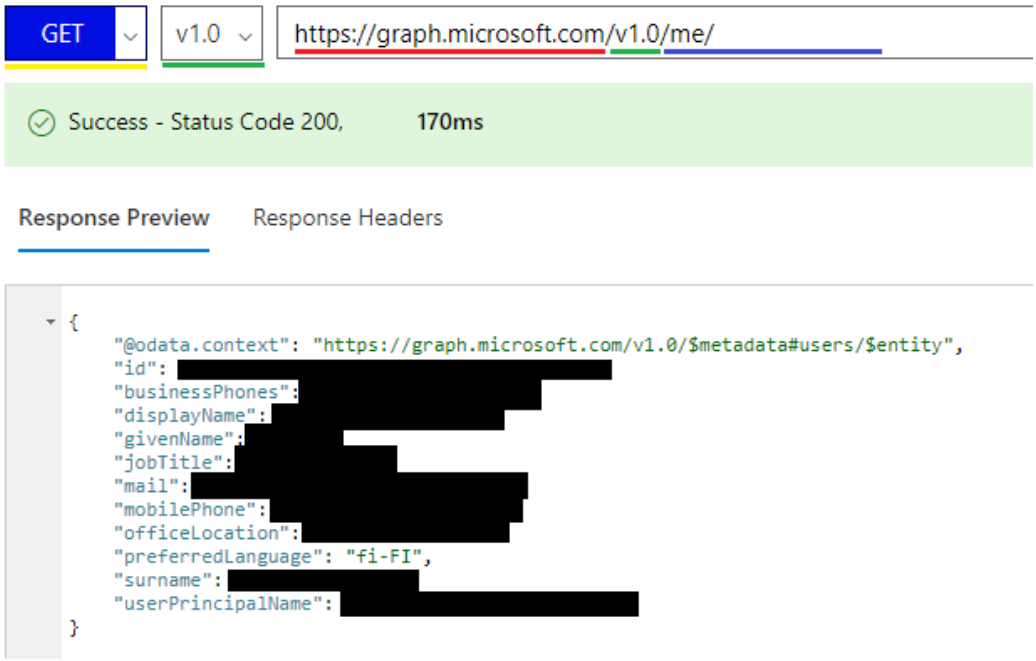
Ennen Graph API -rajapintaa Microsoftilla oli käytössään Office API -rajapinta, joka ajoi suurinpiirtein saman asian. Tämä kuitenkin selkeästi erotteli jokaisen Microsoftin pilvipalvelun (esimerkkejä listattu ylempänä) siten, että jokainen niistä vaati oman rajapinnan, pääsyavaimen sekä päätepisteen. Tämä kuitenkin aiheutti rajapinnan käyttäjille ja Microsoftille sen verran vaivaa, että he päättivät yhdistää kaikki ominaisuudet Graph API:n alle.

5.2 Rajapinnan käyttö

Rajapinnan käyttö on lukittu käyttöavaimen taakse. Kun kysely viimein lähetetään, käyttöavain pitää lisätä osaksi HTTP-pyynnön (HTTP-request) header-kenttää muodossa *Authorization: <käyttöavain>*. Käyttöavain saadaan Azure AD OAuth 2.0 -protokollan kautta, jonka avulla rajapinnan käyttäjälle annetaan valtuudet käyttää Microsoftin pilvipalveluiden tiettyjä tietoja. Tämä tieto on sisällytetty käyttöavaimeen, joten rajapinta suostuu palauttamaan vain nämä tiedot (tai muokkaamaan/lisäämään tietoja takaisin näihin pilvipalveluihin). Käyttöavain on voimassa vain tietyn ajan, yleensä tunnin. Tämän jälkeen rajapintaa varten pitää pyytää uusi käyttöavain.

Koska rajapinnalla on vain yksi päätepiste, pitää sen jotenkin tietää, mitä tietoja rajapinnan käyttäjä haluaa saada, muuttaa tai lisätä. Päätepisteeseen lisätään kysely, joka kertoo rajapinnalle tämän.

Esimerkki 1:ssä (Kuva 6) on kuvattu rajapinnan yksinkertainen käyttö Microsoftin Graph Explorer -palvelun (https://developer.microsoft.com/en-us/graph/graph-explorer) avulla. Tämä on sivu, jolla kehittäjät voivat testata erilaisia kyselyitä ja nähdä, millaista tietoa ne palauttavat.



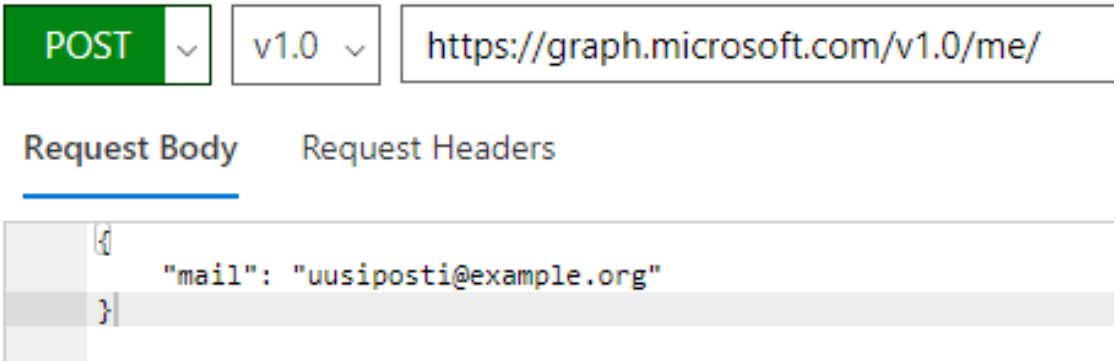
Kuva 6. Esimerkki 1 rajapinnan kyselystä ja sen palauttamista tiedoista (Microsoft Graph Explorer 2018).

Kuvassa on merkattu keltaisella värillä HTTP-pyynnön metodi. Yleisimmät metodit ovat GET ja POST, mutta rajapinnan tapauksessa käytössä on myös harvinaisempia metodeita, kuten PATCH ja DELETE (Taulukko 1).

Taulukko 1. Metodit HTTP-pyynnöissä rajapinnassa.

Metodi	**Käyttö rajapinnassa**
GET	Tietojen hakeminen.
POST	Tietojen lisääminen ja joissain tapauksissa tietojen päivittäminen.
PATCH	Tietojen päivittäminen.
DELETE	Tietojen poistaminen.

Punaisella värillä on merkattu rajapinnan päätepiste. Vihreä väri on rajapinnan versio. Versioita on tällä hetkellä kaksi: v1.0 ja beta. Beta on kehitysvaiheessa ja sisältää paljon uusia ominaisuuksia, kyselyitä ja jopa samat kyselyt vanhan ja beta-version välillä voivat palauttaa eri tietoja. Microsoft suosittelee lähinnä v1.0-version käyttöä, koska beta voi ja tuleekin vielä varmasti muuttumaan, ennen kuin se mahdollisesti nimetään v2.0-versioksi. Sininen väri on kysely. Käyttäjän tiedot saadaan me-kyselyllä eli minä-kyselyllä. Palautettuja tietoja ovat muun muassa nimi, sähköposti ja puhelinnumero. Parametrejä tässä esimerkissä ei ole, sillä niitä ei ole GET-pyynnössä. Esimerkki pyynnöstä, joka sisältää parametrejä, on käyttäjän tietojen päivittäminen. Samaan me-kyselyyn lähetetään POST-pyyntö, ja parametrinä on tiedot, joita halutaan päivittää (Kuva 7). Tiedot on muunnettu JSON-muotoon.



Kuva 7. Esimerkki 2 rajapinnan POST-kyselystä ja sen parametreistä (Microsoft Graph Explorer 2018).

6 TYÖN SUUNNITTELU

Suunnittelun pääpisteinä tässä työssä mielestäni oli kaksi asiaa: vaatimusten määrittäminen ja tarkastelu sekä dokumentaatioon perehtyminen. Microsoftin dokumentointi tuntuu aina välillä vähän sekavalta, ja koska osa ominaisuuksista oli aivan uusia, oli se myös aina välillä puutteellista. Yrityksen ja erehdyksen kautta kuitenkin saatiin hiottua viimeisetkin kohdat.

Tärkeimpänä vaatimuksena oli saada kirjautuminen Opentuntiin hoidettua yksinkertaisesti ilman käyttäjän erityistoimenpiteitä. Lisäksi oli tärkeää, että uudet paikkakunnat saatiin yhdistettyä osaksi tätä kirjautumista helposti ja vaivattomasti.

Suunnitelmana oli tehdä OAuth 2.0 -protokollan toteutus ensin. Hyväksytty OAuth-auktorointi palauttaa palvelimelle käyttöavaimen, jonka avulla on lupa hakea ja muuttaa tietoja AD:ssa. Opentunnin tapauksessa pyydetään lupa saada hakea käyttäjän tietoja. Ainoastaan lukuoikeus (readonly) riittää, koska käyttäjän tietoja ei tarvitse päästä muokkaamaan ollenkaan.

7 TYÖN KEHITYS

7.1 Alusta ja työkalut

Opentunnin palvelimena toimii Windows Server 2016 ja IIS (Internet Information Services), joka on Microsoftin kehittämä palvelinohjelmistokokonaisuus (Wikipedia, 2018).

Yleisesti Windows-pohjaisissa palvelimissa käytetään jotain Microsoftin kehittämää ohjelmointikieltä sekä tietokantaa, kuten ASP-ohjelmointikieltä ja SQL Server -tietokantaa, mutta tästä poiketen Opentunti on koodattu PHP-ohjelmointikielellä ja tietokantana toimii MySQL. Tämä johtuu huonoista kokemuksista yleisesti ASPin sekä SQL Serverin kanssa ja kokemuksesta PHP:n sekä MySQL:n kanssa. Alustalla ei kuitenkaan ole oikeastaan merkitystä tämän työn kannalta.

Kehitystyö eli koodaaminen on suoritettu alusta loppuun käsin, vaikka yleensä tämänkaltaisissa projekteissa aloitetaan asentamalla jokin ohjelmistokehys (engl. framework), joka toimii projektin ytimenä. Opentunnissakin tällainen ohjelmistokehys on, mutta se on kehitetty itse. Ulkopuolisia työkaluja ei siis ole käytössä juuri yhtään.

Alustaa on kuitenkin päivitetty ajan mittaan. Palvelinohjelmiston, PHP:n sekä MySQL:n versiot ovat muuttuneet vuosien varrella, minkä ansiosta tietoturva ja suorituskyky ovat parantuneet lähtöpisteestä.

7.2 MicrosoftGraph -luokka

Kehitys aloitettiin koodaamalla PHP:lla luokka Azure AD Oauth 2.0:aa ja Graph API -rajapintaa varten. Luokka paketoi kaikki tarvittavat pyynnöt, jottei niitä tarvitse toistaa koodissa useaa kertaa. Luokan osia ovat:

- vakiomuuttujat (constants) ja ominaisuudet (properties)
- muodostin (constructor)
- kaksi funktiota tunnistautumista varten
- yksi funktio per HTTP-pyynnön metodi rajapintaa varten.

Vakiomuuttujat sisälsivät päätepisteet Microsoftin kirjautumispalveluun ja rajapintaan, ominaisuudet taas kaikki vaadittavat muuttujat, kuten asiakastunnisteen,

ohjausosoitteen ja pääsyavaimen (Kuva 8). Muodostimessa ominaisuuksiin asetetaan arvot (Kuva 9). Ensimmäinen funktio tunnistautumista varten palauttaa osoitteen Microsoftin kirjautumispalveluun, jota kautta saadaan OAuth 2.0 authorization code grantin auktorointikoodi pääsyavainta varten (Kuva 10) ja toinen funktio hakee pääsyavain käyttämällä tätä koodia (Kuva 11). Rajapintafunktiot joko hakevat tietoa rajapinnasta tietyllä kyselyllä tai sitten lisäävät tai muokkaavat tietoja (Kuva 12).

Kehitys tapahtui aluksi Microsoftin dokumentaatioita lukemalla ja sitten yrityksen ja erehdyksen kautta. Tämänkaltaisia toimintoja on erittäin hankala saada toimimaan heti ensimmäisellä yrityksellä, koska mukana on kolmannen osapuolen rajapinta ja sen jokseenkin hajallaan oleva ja sekava dokumentaatio.

```php
<?php

class MicrosoftGraph
{
    const BASE_LOGIN_URL = "https://login.microsoftonline.com";
    const BASE_GRAPH_URL = "https://graph.microsoft.com";

    private $clientId;
    private $clientSecret;
    private $redirectUri;
    private $scope;

    private $state;
    private $accessToken;
```

Kuva 8. Luokan alku, vakiomuuttujat sekä ominaisuudet.

```php
    public function __construct(array $params = [])
    {
        $this->clientId = $params["clientId"] ?? null;
        $this->clientSecret = $params["clientSecret"] ?? null;
        $this->redirectUri = $params["redirectUri"] ?? null;
        $this->scope = $params["scope"] ?? null;

        $this->state = $params["state"] ?? base64_encode(openssl_random_pseudo_bytes(16));
        $this->accessToken = $params["accessToken"] ?? null;
    }
```

Kuva 9. Luokan muodostin.

```php
public function getAuthorizationUrl() : string
{
    $queryString = http_build_query(array
    (
        "client_id"     => $this->clientId,
        "response_type" => "code",
        "redirect_uri"  => $this->redirectUri,
        "response_mode" => "query",
        "scope"         => $this->scope,
        "state"         => $this->state,
    ));
    return static::BASE_LOGIN_URL . "/common/oauth2/v2.0/authorize?" . $queryString;
}
```

Kuva 10. Ensimmäinen funktio tunnistautumista varten.

```php
public function getAccessToken(string $code) : string
{
    $params = array
    (
        "client_id"     => $this->clientId,
        "code"          => $code,
        "redirect_uri"  => $this->redirectUri,
        "grant_type"    => "authorization_code",
        "client_secret" => $this->clientSecret,
    );
    $curl = new Curl();
    $result = $curl->post(static::BASE_LOGIN_URL . "/common/oauth2/v2.0/token", $params);
    $curl->close();
    $arr = json_decode($result, true);

    $this->accessToken = sprintf("%s %s", $arr["token_type"], $arr["access_token"]);
    return $this->accessToken;
}
```

Kuva 11. Toinen funktio tunnistautumista varten.

```
62      public function getQuery(string $query)
63      {
64          $url = static::BASE_GRAPH_URL . $query;
65          $headers = array
66          (
67              "Authorization" => $this->accessToken,
68              "Accept"        => "application/json",
69              "Content-Type"  => "application/json",
70              "Prefer"        => "return-content",
71          );
72          $curl = new Curl();
73          $curl->setHeaders($headers);
74          $result = $curl->get($url);
75          $curl->close();
76          $arr = json_decode($result, true);
77
78          return $arr["value"] ?? $arr;
79      }
80  }
```

Kuva 12. Funktio rajapinnan GET-pyyntöä varten.

7.3 Luokan käyttö

Edellä olevaa luokkaa testattiin erittäin yksinkertaisesti ilman virheiden tarkistusta ja tilan (state) vahvistamista (Kuva 13). Tilan vahvistaminen on tärkeä osa tietoturvaa, jottei kukaan ulkopuolinen pääse vaihtamaan koodin käyttämiä GET-parametrejä käsin.

Ensimmäiseksi otetaan luokka käyttöön. Sen jälkeen luodaan osoite Microsoftin kirjautumispalveluun. Käyttäjä ohjataan tähän osoitteeseen, joka tunnistaa käyttäjän jo kirjautuneeksi Microsoftin palveluihin. Kirjautumispalvelu ohjaa käyttäjän takaisin Opentunnin sivulle GET-parametrien kera, joten koodissa päästään eteenpäin. Authorization code on yksi GET-parametreistä, jolla pyydetään käyttöavain. Käyttöavaimella taas haetaan käyttäjän tiedot rajapinnan kautta (Kuva 13).

```php
<?php

require_once("class.MicrosoftGraph.php");

$graph = new MicrosoftGraph(array
(
    "clientId"      => "",
    "clientSecret"  => "",
    "redirectUri"   => "",
    "scope"         => "",
));

if (empty($_GET))
{
    $url = $graph->getAuthorizationUrl();
    redirect($url);
}

$accessToken = $graph->getAccessToken($_GET["code"]);

$user = $graph->getQuery("/beta/me");

$name = $user["displayName"];
$email = $user["mail"];
// ...

```

Kuva 13. Yksinkertaistettu esimerkki luokan käytöstä, jossa haetaan käyttäjän nimi ja sähköposti rajapinnan kautta tunnistautumisen jälkeen.

7.4 Valmis koodi

Valmis koodi toimii samalla tavalla kuin kuvan 13 esimerkki, mutta siihen on vain lisätty paljon tarkistuksia väleihin ja tietojen hakua rajapinnan kautta on jatkettu. Ennen käyttäjän tietoja tarvitaan paikkakunnan eli organisaation tiedot, jotta voidaan löytää oikea asetustiedosto. Palvelimelle on luotu yksi asetustiedosto jokaista paikkakuntaa varten, jotta siitä saadaan tarvittavat tiedot, kuten esimerkiksi missä attribuuteissa käyttäjän tiedot sijaitsevat.

Kun käyttäjän kaikki tiedot on viimein haettu, tarkistetaan, onko Opentunnissa kyseistä käyttäjää olemassa. Jos on, hänet päästetään kirjautuneena sisään. Jos ei, luodaan hänelle uusi käyttäjä Opentuntiin kyseisillä tiedoilla ja vasta sitten kirjaudutaan.

8 TYÖN TESTAUS

Kuten edellisessä luvussa jo todettiin, testaus osoittautui suhteellisen haastavaksi puutteellisen dokumentoinnin takia. Yksinkertaisin esimerkki onnistui nopeasti, mutta työ vaati lisäksi erityisiä ominaisuuksia, joista ei ollut joko ollenkaan dokumentointia tai sitä oli hyvin niukasti ja hajaututesti.

Testausta varten vaadittiin myös tietysti testiympäristö. Koska tuotannossa taustalla on paikkakunnan Azure-ympäristö, tarvittiin myös testaamiseen sellainen. Sinne piti myös luoda samanlaisia käyttäjiä kuin oikeissa ympäristöissä.

Testaus suoritettiin niin, että koodiin lisättiin paljon lokikirjauksia. Lokista näki tarkasti mihin päätepisteeseen käyttäjä ohjattiin ja mistä päätepisteestä käyttäjä palasi ja mitä tietoja sieltä tuli. Lokikirjaukset vähenivät pikkuhiljaa, mutta osa on jätetty tähän päivään saakka tuotantoon, jotta mahdollisissa tulevissa virhetilanteissa olisi helppoa selvittää, mikä on mennyt vikaan.

Graph API -rajapinnan osalta virheet oli helppo havaita, koska se palautti aina enemmän tai vähemmä selkeän virheviestin. OAuth-tunnistautumisen suhteen tilanne oli vähän hankalampi, koska virheviesti jäi ainoastaan näkymään käyttäjälle selaimeen. Omassa testiympäristössä ja testaamisessa tämä ei tietysti haitannut, mutta kun oikeat käyttäjät pääsivät testaamaan (ja rikkomaan) asioita ensimmäisen kerran, päätä tuli raavittua muutaman kerran. Testiympäristön kanssa helppoa oli myös se, että tiesi, mitä tietoja testikäyttäjät sisältävät missäkin kentässä/attribuutissa. Oikeiden käyttäjien kanssa minun piti olla pari kertaa yhteydessä paikkakunnan IT-tukihenkilöön, joka vahvisti, mitä tietoja kyseisen paikkakunnan käyttäjillä oikeasti on.

Testiympäristön lisäksi testaamista siis suoritettiin myös aina uuden paikkakunnan liittyessä Opentuntiin. Aiemman kokemuksen mukaan erilaiset ympäristöt aiheuttavat aina ongelmia, eikä tietojen samankaltaisuuteen voi tosiaan luottaa satavarmasti. Näin kävi myös tässä tapauksessa. Ensimmäisten kahden paikkakunnan kanssa piti asetustiedostoihin lisätä kohtia ja koodia päivittää.

9 TYÖN DOKUMENTOINTI

Osana työn vaatimuksia oli dokumentoida työtä (tämän opinnäytetyön ulkopuolella) ja kirjoittaa tarkasti asennuksen eri vaiheet ylös. Jokainen uusi Opentuntiin liittyvä paikkakunta joudutaan lisäämään samalla tavalla (poissulkien koodaus), joten asennusohjeet ovat tärkeät kyseisen paikkakunnan IT-tukihenkilölle tai -ylläpitäjälle.

Loppujen lopuksi asennusvaiheista tuli seuraavat:

1. Paikkakunnalla pitää olla Azure AD käytössä.
2. Opentunnin palvelimelle luodaan uusi asetustiedosto paikkakuntaa varten, jossa on listattu yleisiä tietoja (kuva X).
3. Paikkakunnan IT-tukihenkilö tai -ylläpitäjä, jolla on ylläpitotili Azure AD:hen, käy antamassa Opentunnin sovellukselle luvan hakea käyttäjien tietoja koko organisaatiosta Graph API -rajapinnan kautta.
4. Paikkakunnan IT-tukihenkilö tai -ylläpitäjä kertoo ne henkilöt tai ryhmät, joilla on paikkakunnan osilta erityisoikeuksia tietyissä Opentunnin ominaisuuksissa. Nämä tiedot lisätään asetustiedostoon (Kuva 14).

Tämän jälkeen asennus on valmis. Kun paikkakunnan käyttäjä kirjautuu Opentuntiin, pääsee hän suoraan sisään ilman erillistä kirjautumista.

```php
<?php

return array
(
    "main.enabled"                  => true,
    "main.domain"                   => ".onmicrosoft.com",
    "main.directoryId"              => "",
    "main.domainHint"               => "",

    // List of groups (email) where members are Opentunti admins.
    "opentunti.admin.groupList"     => array
    (
    ),
```

Kuva 14. Esimerkki Opentunnin yksittäisestä asetustiedostosta.

Dokumentointi on myös muuten tärkeää. Jos projektin kehitystiimiin tulee lisäyksiä tai muutoksia, on uusille kehittäjille paljon helpompaa lukea dokumentointia tai edes jonkinlaisia muistiinpanoja, kuin alkaa tuhansista koodiriveistä selvittämään, miten sovellus toimii.

Tämän projektin kannalta dokumentoinnissa tärkeää oli myös se, että Graph API:sta on käytössä beta-versio, koska vanhasta versiosta ei tule ulos tarpeeksi tietoja, joita Opentunti vaatii. Virallisen Microsoftin dokumentoinnin mukaan tämän version ominaisuudet ja toiminnot saattavat muuttua tulevaisuudessa jopa ilman varoitusta. Tällaiset mahdolliset ongelmat tulevaisuudessa on hyvä kirjata, jotta niiden korjaaminen sujuu tarvittaessa nopeasti ja sujuvasti.

10 LOPUKSI

Azure AD OAuth 2.0 -tunnistautumisen ja Graph API -rajapinnan käytön integrointi Opentuntiin onnistuivat lopulta hienosti. Ominaisuus saatiin suoraan käyttöön yhteen paikkakuntaan ja aika pian muihinkin Opentunnissa mukana oleviin paikkakuntiin. Vaatimukset täyttyivät ja käyttäjät pääsivät eroon käyttäjätunnus-salasana-yhdistelmästä. Microsoftin AD-tilille tietysti pitää kirjautua käyttäjätunnus-salasana-yhdistelmällä, mutta tätä ei voi välttää.

Jatkokehitystä varten on mietitty käyttää Graph API -rajapintaa laajemmin osana Opentuntia. Rajapinnan kautta voi saada käyttäjän ryhmät, jotka koulumaailmassa usein tarkoittavat käyttäjän kurssiryhmiä ja/tai luokkia. Näin saadaan lisää tietoa käyttäjistä ja samassa ryhmässä tai luokassa olevia opettajia tai oppilaita voidaan yhdistää toisiinsa. Kuten jo mainittu, rajapinta tukee myös tiedon lähettämisen toiseen suuntaan eli tiedon tallentamisen takaisin pilveen Microsoftin palvelimille. Tämä mahdollistaa sen, että pilveen voidaan viedä tarkempia tietoja esimerkiksi käyttäjän koulusta ja paikkakunnasta. Tämän avulla voidaan luopua osittain tiedon tallentamisesta Opentunnin palvelimelle ja luottaa rajapinnan kautta tulevaan tietoon, joka Azure AD:ssa synkronoituu automaattisesti ja pysyy näin aina ajan tasalla.

Suurin osa tässä työssä käsiteltävistä asioista oli minulle aivan uusia. Tässä siis yhdistyivät kätevästi työprojektin eteneminen, tärkeän ja kaivatun toiminnon lisääminen, opinnäytetyön teko sekä uuden asian oppiminen, joka tulee työelämässä olemaan varmasti arvokasta.

LÄHTEET

Opentunti ry 2018. Etusivu. Viitattu 2.5.2018 http://www.opentuntiry.fi/.

Microsoft docs 2018. What is Azure Active Directory (Azure AD)? Viitattu 2.5.2018 https://docs.microsoft.com/en-us/azure/active-directory/active-directory-whatis.

Okta Developer 2018. What is the OAuth 2.0 Authorization Code Grant Type? Viitattu 2.5.2018 https://developer.okta.com/blog/2018/04/10/oauth-authorization-code-grant-type.

Wikipedia 2018. Internet Information Services. Viitattu 2.5.2018 https://fi.wikipedia.org/wiki/Internet_Information_Services.