jamk

Microsoft Graph-rajapinnan käyttöönotto WordPress-sivustolle

Jenni Tahvanainen

Opinnäytetyö, AMK
Toukokuu 2024
Tieto- ja viestintätekniikka

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Tahvanainen, Jenni

Microsoft Graph-rajapinnan käyttöönotto WordPress-sivustolle

Jyväskylä: Jyväskylän ammattikorkeakoulu. Toukokuu 2024, 31 sivua.

Tieto- ja viestintätekniikan tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: kyllä

Tiivistelmä

Asiakkaalla oli tarve esittää henkilöiden yhteystietoja WordPress-sivustolla. Sisällönsyötön helpottamista varten nämä tiedot päätettiin hakea Microsoftin palvelusta, jossa asiakas aktiivisesti ylläpiti henkilöiden tietoja. Sivuston perustamisessa hyödynnettiin Digitoimisto Duden Dudestack-työkalua, joka automatisoi uuden WordPress-projektin luomisen. Lisäksi käytettiin Air-light-teemaa ja Air-helper-lisäosaa verkkosivuston rakentamiseen. Olennaisessa osassa oli myös Advanced Custom Fields-lisäosa tietojen muokkaamiseen.

WordPress-projektiin luotiin uusi sisältötyyppi henkilöille, johon tiedot täydennettiin Microsoft Graph -rajapinnan kautta. Henkilöiden sisältötyyppiin lisättiin kenttiä, kuten etunimi, sukunimi ja sähköpostiosoite. Lisäksi luotiin kaksi taksonomiaa henkilöiden järjestämiseksi: yksikkö ja koulutusala.

Integroitaessa Microsoft Graph -rajapintaa WordPressiin, toteutettiin API-yhteys, joka mahdollisti tietojen hakemisen Microsoft 365-palvelusta. Tiedot haettiin REST API -kutsujen avulla, ja access token saatiin autentikointipyyntöjen avulla. Lisäksi toteutettiin tietojen tallennus WordPress-tietokantaan ja kuvien tallennus erilliseen kansioon.

Lopuksi kehitettiin ajastin automaattiselle API-kutsulle, joka mahdollisti tietojen synkronoinnin WordPress-sivustolle. WP-CLI-komennolla oli mahdollista suorittaa synkronointi manuaalisesti tarvittaessa. Tuloksena saatiin toimiva yhteys rajapintaan, josta tiedot saatiin onnistetusti haettua.

Avainsanat (asiasanat)

toimintatutkimus, ohjelmointirajapinnat, web-kehitys, WordPress

Muut tiedot (salassa pidettävät liitteet)

jamk

Description

Tahvanainen Jenni

Deployment of Microsoft Graph API to a WordPress Site

Jyväskylä: JAMK University of Applied Sciences, May 2024, 31 pages.

Degree Programme in Information and Communications Technology. Bachelor´s thesis.

Permission for open access publication: Yes

Language of publication: Finnish

Abstract

The client needed to display contact information for individuals on their WordPress website. It was decided to fetch this information from Microsoft's service, where the client actively maintained individuals' data. The creation of the website utilized the Digitoimisto Dude’s Dudestack tool, which automated the creation of a new WordPress project. Additionally, the Air-light theme and Air-helper plugin were used for website construction. An essential component was also the Advanced Custom Fields plugin for data editing.

A new content type for individuals was added to the WordPress project, with the data retrieved via the Microsoft Graph API. Fields such as first name, last name, and email address were added to the individuals' content type. Additionally, two taxonomies were defined for organizing individuals: department and field of study.

In the integration of the Microsoft Graph API with WordPress, an API connection was created to retrieve data from Microsoft's cloud service. Data retrieval occurred through REST API calls, with access tokens obtained through authentication requests. Furthermore, a mechanism was developed for storing data in the WordPress database and images in a separate directory.

Finally, a timer was created for automatic API calls, enabling data synchronization between the WordPress site and Microsoft's cloud service. Synchronization could also be manually performed using the WP-CLI command. As a result, a functional connection to the API was established, successfully fetching the data.

Keywords/tags (subjects)

research-based development, API, web development, WordPress

Miscellaneous (Confidential information)

Sisältö

Kuviot

Taulukot

1 Johdanto

WordPress on suosittu sisällönhallintajärjestelmä, jolla pystyy tekemään verkkosivuja, blogeja ja sovelluksia. WordPressiä kuvaillaan yksinkertaiseksi, joustavaksi, helpoksi ja sen muita etuja ovat esimerkiksi vapaus muokata ja tehdä lisäyksiä. WordPressille on tehty jopa tuhansia eri lisäosia, joilla pystytään laajentamaan sen toimintaa. (Features N.d.)

Opinnäytetyön toimeksiantaja, Digitoimisto Dude, on suomalainen yritys, joka on erikoistunut WordPress-sivustojen tekemiseen ja ollut aktiivisesti mukana WordPressiin liittyvässä toiminnassa. Digitoimisto Duden asiakkaan verkkosivustolle haluttiin lisätä henkilökunnan yhteystietoja, mutta tietojen ylläpito voi joskus olla työlästä tai päivittäminen unohtua. Tämä saattaa johtaa siihen, että sivustolla näkyy vanhentunutta tietoa henkilöistä tai henkilöitä, jotka eivät ole enää töissä.

Ongelman ratkaisemiseksi asiakkaalle haluttiin tehdä sivuston ylläpito helpommaksi tuomalla henkilökunnan tiedot automaattisesti toisesta palvelusta. Työn toteutuksessa käytettiin Microsoftin palveluita sekä rajapintaa, koska asiakkaalla oli jo entuudestaan käytössä Microsoftin palvelu Microsoft 365, jonne henkilökunnan jäseniä lisättiin. Microsoftilta löytyi tehtävään sopiva ohjelmointirajapinta nimeltä Microsoft Graph, jonka avulla pystyttiin hakemaan Microsoft 365-palvelusta kaikki tarvittavat tiedot.

Työn toteutettiin WordPressillä, koska Digitoimisto Dudella on paljon kokemusta WordPressin käytöstä. WordPress ja Microsoft ovat myös alalla hyvin suosittuja. WordPress on sisällönhallintajärjestelmistä suosituin ja sen suosio on kasvanut nopeammin kuin yhdenkään toisen sisällönhallintajärjestelmän. Myös markkinaosuus on kasvanut jatkuvasti. (Dobrilova 2024.) Microsoft puolestaan on maailman 9. suurin yritys ja M365 on yksi sen tunnetuimpia palveluita (Patrizio 2023; Murphy & Tucker 2023). Sekä Microsoftia, että WordPressiä voi pitää merkittävänä niiden suuruuden puolesta ja ne valikoituvat tähän työhön hyvien tulevaisuudennäkymien ansiosta.

Opinnäytetyön tarkoitus on olla opas Microsoft Graph-ohjelmointirajapinnan integroimiseen ja käyttöön WordPress-ympäristössä. Koska ohjelmointirajapinnan käyttöönotto ja hyödyntäminen WordPress-ympäristössä ovat työn keskiössä, rajattiin WordPressin käytännön toteutuksen esit-

tely vain niihin osiin, jotka ovat olennaisia rajapinnan lisäosan kannalta. WordPressin käyttöönotosta löytyy useita kirjoja, oppaita, sekä muita opinnäytetöitä, joten aihetta on jo tutkittu paljon entuudestaan. Sen laajempi esitteleminen ei toisi tutkimuksen näkökulmasta uutta tietoa.

Ohjelmoinnissa apuna on käytetty GitHubin tarjoamaa Copilot-tekoälyä. Copilot on ohjelmoimiseen tarkoitettu tekoäly, joka osaa koodia kirjoittaessa ehdottaa, mitä seuraavalle riville voisi esimerkiksi tulla. Se tarkkailee aiemmin kirjoitettua koodia ja osaa mukautua käyttäjän tapaan kirjoittaa koodia. (Friedman 2021.) Tässä opinnäytetyössä Copilotista on lähinnä ollut apua koodin kirjoittamisen nopeuttamisessa, kun Copilot on osannut ennakoida vähintään osittain, mitä seuraaville riveille halutaan lisätä.

2 Tutkimusasetelma

2.1 Tutkimuskysymykset

Työtä ennen oli jo valmiiksi määritelty ratkaisu, miten asiakkaan sivuston käyttöönottoa ja sisällönsyöttöä voisi parantaa. Tekniikoiksi valittiin WordPress ja rajapinnaksi Microsoft Graph API. Tämän työn tutkimuskysymyksiä ovat:

1. Miten Microsoft Graph-ohjelmointirajapinta otetaan käyttöön WordPress-ympäristössä?
2. Miten rajapinnalla saadaan haettua käyttäjien tiedot ja tallennettua sivustolle?
3. Miten rajapintayhteydelle tehdään oma WordPress-lisäosa?

2.2 Kehittämistutkimus

Opinnäytetyön toteutustavaksi valittiin kehittämistutkimus, koska sillä pystyttiin parhaiten selvittämään tutkimuskysymyksille vastauksia. Kehittämistutkimuksen avulla pyritään tuottamaan tietoa kolmeen keskeiseen kysymykseen: miten kehittämisessä edetään, millaisia tarpeita ja mahdollisuuksia kehittämisellä on ja millaiseen tuotokseen kehittäminen johtaa. Kehittämispäätökset jaetaan kolmeen kategoriaan: kehittämisprosessi, ongelma-analyysi ja kehittämistuotos, joista jokainen tuottaa erityyppistä teoriaa ja tietoa. (Pernaa 2013, 6–7.)

Pernaan (2023) mukaan Edelson (2002 ja 2006) on määritellyt kehittämistutkimuksen yhdistävän kehittämisen ja tutkimuksen sykliseen prosessiin. Pernaa esittää myös Wangin ja Hannafin (2005) näkökulman, jonka mukaan kehittämistutkimuksen tavoitteina on todellisten tilanteiden systemaattista, joustavaa ja iteratiivista kehittämistä. Olennaisia asioita ovat jatkuva arviointi ja asiantuntijoiden hyödyntäminen kehittämisessä. (Pernaa 2013, 4–5.)

Kehittämistutkimus syntyi tarpeesta kehittää opetusta tutkimuspohjaisesti todellisista opetustilanteista 90-luvulla. 2000-luvulla kehittämistutkimuksen tunnettavuus ja osaaminen alkoivat kasvaa. Tämä näkyy esimerkiksi julkaistujen tutkimusten määrässä. Se on kasvanut tasaisesti 2000-luvun aikana, kun aiemmalla vuosikymmenellä kehittämistutkimuksia julkaistiin vain muutamia kymmeniä. (Pernaa 2013, 3–4.)

3 Käytetyt tekniikat

3.1 WordPress

WordPress on sisällönhallintajärjestelmä, joka kattaa suuren osan tehdyistä web-sivustoista. Vuonna 2024 tehtyjen tutkimusten mukaan noin 43,3 % sivustoista käyttää WordPressiä. (W3Techs N.d.) Alun perin WordPress oli tarkoitettu blogien tekemiseen, mutta vuosien saatossa se on kehittynyt ja nykyään sillä pystyy tekemään muitakin toteutuksia, kuten verkkokauppoja tai yrityksien verkkosivuja. Sitä käyttävät sekä yritykset että yksityishenkilöt. WordPress on avointa lähdekoodia GPLv2-lisenssillä eli kuka tahansa pystyy tarkastelemaan sen koodia, käyttämään ja muokkaamaan sitä haluamallaan tavalla. (What Is WordPress? Explained for Beginners 2023.)

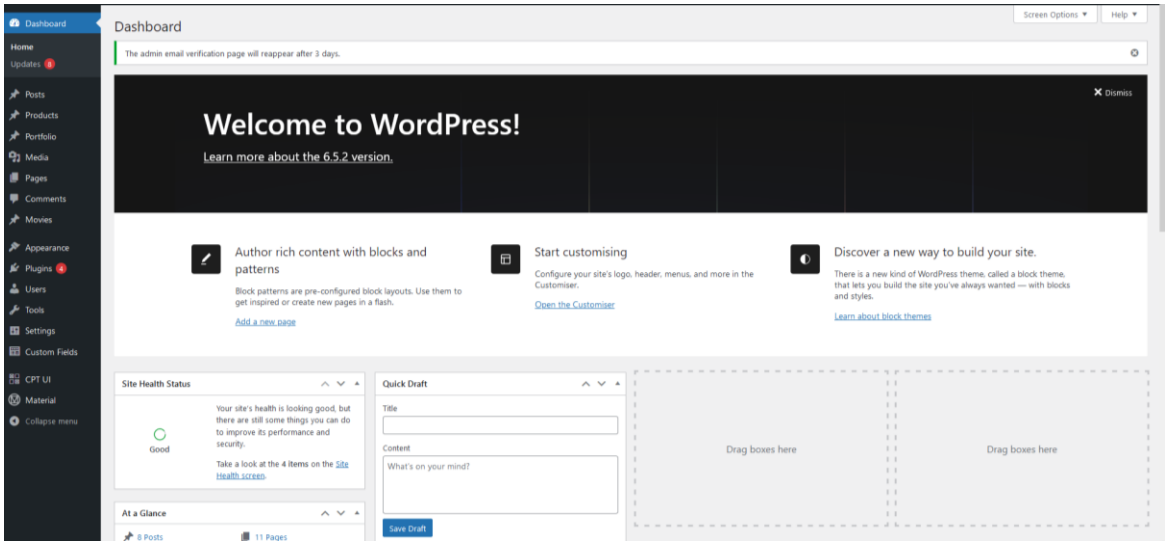
Sisällönhallintajärjestelmä tarkoittaa ohjelmistoa, jolla pystyy luomaan ja hallitsemaan sisältöä ilman teknistä tietämystä. Tämä tarkoittaa sitä, että verkkosivun pystyy luomaan kirjoittamatta koodia alusta asti itse. Sisällön hallitsemiseen on WordPressissä käyttöliittymä, jonka kautta sivuja pystyy luomaan ja niille lisäämään sisältöä, kuten tekstiä ja kuvia. (What Is a Content Management System (CMS)? 2023.)

3.2 WordPressin ominaisuudet

WordPressin olennaisia osia ovat teemat ja lisäosat. Teemoilla toteutetaan sivujen visuaalinen ulkonäkö ja asettelu. Kuten editorit, myös teemat jaetaan kahteen kategoriaan, klassisiin ja lohkopohjaisiin teemoihin. Klassiset teemat tarjoavat vähemmän visuaalista muokkausta ja erilaisia komponentteja, eikä sivujen rakennetta pysty usein muokkaamaan hallinnasta. Lohkoteemat sen sijaan tarjoavat usein enemmän hallintaa ja muokkausvaihtoehtoja. (Understanding Themes 2024.)

Lisäosien tarkoitus on tuoda uusia toiminnallisuuksia ja ominaisuuksia WordPress-sivustolle. Lisäosat voivat tarjota suuria tai pieniä muutoksia verkkosivuun. Esimerkiksi monet WordPressillä tehdyt verkkokaupat käyttävät verkkokaupan luomiseen lisäosaa, joka valmiiksi tarjoaa tuotteiden myymiseen tarvittavat ominaisuudet. (Newcomer 2023.)

WordPress-sivustoille sisältöä pystyy syöttämään graafisen käyttöliittymän kautta hallintapaneelista. Kuviossa 1 näkyy esimerkki pääkäyttäjän näkymästä. WordPress-sivustoille on olemassa kaksi erilaista editoria, klassinen editori ja lohkoeditori. Klassinen editori on vanhempi. Se sisältää yksittäisen tekstikentän, johon sisältöä voi lisätä. Lohkoeditori eli Gutenberg-editori on uudempi ja se lisättiin oletuseditoriksi WordPressiin vuonna 2018. Lohkoeditorissa sisältö rakennetaan lisäämällä ja järjestelemällä erilaisia lohkoja. Jokaisella lohkolla on oma käyttötarkoituksensa, kuten otsikon, kuvan tai tekstin lisääminen. (Hasib 2024.)



Kuvio 1. WordPressin ohjauspaneeli

3.3 WordPressissä käytetyt kielet

WordPress on suurimmaksi osaksi koodattu PHP:lla. PHP:n lisäksi WordPressissä on käytetty JavaScript-, HTML-ja CSS-kieliä. Tietokanta on rakennettu MySQL:llä. (WordPress N.d.)

PHP

PHP on avoimen lähdekoodin palvelinpuolen ohjelmointikieli, joka soveltuu erityisesti web-kehitykseen (What is PHP? N.d.). Sen kehitti vuonna 1994 Rasmus Lerdorf alun perin seuraamaan vierailumääriä hänen ansioluettelollaan ja PHP oli silloin lyhenne sanoille Personal Home Page. PHP nimettiin tämän jälkeen useamman kerran uudestaan ja versiossa 3.0 sen koko nimeksi tuli ”PHP: Hypertext Preprocessor”, jota käytetään yhä tänä päivänä. (History of PHP N.d.)

JavaScript

JavaScript on asiakaspuolen kieli, jota käytetään usein dynaamisten osien tekemiseen web-sivuilla, kuten animoituihin grafiikoihin (What is JavaScript? N.d.). Ensimmäinen versio JavaScriptistä julkaistiin vuonna 1995. Sen kehitti NetScape yhdessä Sun Microsystemsin kanssa. Myöhemmin Microsoft julkaisi kielestä oman versionsa nimeltä JScript ja kielelle päätettiin kehittää standardi, jotta syntaksi ja ominaisuudet olisivat yhdenmukaisia. (Friesbie 2024, luku 1.)

HTML

HTML eli HyperText Markup Language on kieli, jota käytetään web-sivun rakenteen luomiseen elementeillä (HTML N.d.). Se kehitettiin CERNissä ja julkaistiin virallisesti vuonna 1993. HTML:n pääkehittäjänä toimi Tim Berners-Lee, joka kehitti myös World Wide Webin vuonna 1989. (Bellis 2019.)

CSS

CSS eli Cascading Style Sheets on tarkoitettu HTML:n tyylittelyyn (Learn to style HTML using CSS N.d.). Ennen CSS:ää tyylitiedostojen tekemiseen ei ollut yhtenäistä kieltä. Vuonna 1994 norjalainen Håkon Wium Lie teki esityksen CSS-kielestä ja alkoi kehittämään sitä yhdessä Bert Bosin kanssa. Nykyään CSS on laajin käytetty kieli HTML:n tyylien tekemiseen. (A brief history of CSS until 2016 N.d.)

SCSS

CSS:n kirjoittamiseen on myös kehitetty SCSS eli Sassy Cascading Style Sheets. Se on kieli, jonka avulla CSS:ää pystyy kirjoittamaan funktioiden, muuttujien ja muiden ominaisuuksien avulla. Ominaisuudet helpottavat koodin ylläpitämistä, järjestelemistä ja luettavuutta. Jotta SCSS voi toimia, se tarvitsee kääntää CSS-kieleksi. Tähän tarkoitukseen on olemassa erillisiä työkaluja ja sellainen tulee lisätä myös WordPress-ympäristöön, jos SCSS-kieltä aikoo käyttää. (How To Get Started With SCSS: A Beginner Guide 2024; Chobanyan 2020.)

MySQL

MySQL on suosittu avoimen lähdekoodin relaatiotietokantajärjestelmä. SQL on lyhenne sanoista Structured Query Language. Relaatiotietokanta taas tarkoittaa sitä, että tiedot tallennetaan useaan eri taulukkoon ja taulukoiden välille luodaan suhteita, jotka sanelevat taulukon kenttien väliset säännöt. (What is MySQL? N.d.)

3.4 Työkalut

Työ toteutettiin Macbook Pro-tietokoneella. Työssä käytettiin versionhallintajärjestelmää nimeltä Git ja se tallennettiin GitHub-palveluun. Versionhallintajärjestelmä on järjestelmä, jolla pystyy tekemään eri versioita projektin koodista ja yhdistämään niitä helposti. Se on hyödyllinen erityisesti silloin, kun projektilla on useampi tekijä. GitHub on pilvipalvelu, joka on rakennettu Gitin päälle. Sen avulla projekteja ja erityisesti koodia voi esitellä ja jakaa muille. (About GitHub and Git N.d.)

Rajapinnan testaamista varten otettiin käyttöön Postman-niminen sovellus. Postman on tarkoitettu rajapintojen tekemiseen ja käyttämiseen ja sisältä työkaluja, joilla rajapintoja on mahdollista testata. Sovellus on ilmainen, mutta Postman tarjoaa myös maksullisia vaihtoehtoja. (What is Postman? n.d.)

Koodin kirjoittamiseen käytettiin sovellusta Visual Studio Code. Visual Studio Code on ilmainen, avoimen lähdekoodin koodieditori, joka tukee useita eri ohjelmointikieliä. Se tarjoaa koodin kirjoittamista helpottavia ominaisuuksia ja siihen on mahdollista tuoda niitä lisää lisäosien avulla. Visual Studio Coden omistaa Microsoft. (Code Editing. Redefined. n.d.)

4 WordPress-sivuston luominen

4.1 Sivuston pystytys, lisäosat ja teemat

Tämän työn WordPress-sivusto pystytettiin Digitoimisto Duden Dudestack-työkalulla. Sen avulla pystyy luomaan helposti uuden WordPress-projektin createproject-skriptillä. Uusi projekti vaatii yleensä lukuisia eri asetuksia ja kyseinen skripti hoitaa ne automaattisesti. Tarkemmat ohjeet löytyvät Dudestackin GitHub-repositoriosta (Dudestack N.d.).

Air-light on Digitoimisto Duden luoma WordPressin aloitusteema. Se on suunnattu kehittäjille ja sen tarkoituksena on olla mahdollisimman kevyt pohja uuden verkkosivuston luomiseen, jonka päälle pystyy luomaan oman teeman. Esimerkiksi WordPress tarjoaa oletuksena paljon ominaisuuksia, joita monet sivustot eivät loppujen lopuksi tarvitse ja Air-lightista nämä ominaisuudet on poistettu käytöstä. (Speed up your WordPress development N.d.)

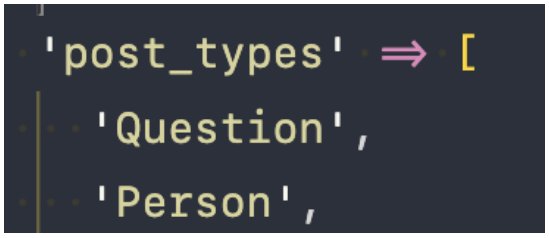
Air-Lightin lisäksi sivustolla on käytössä myös Duden tekemä Air-helper-lisäosa, joka tarjoaa avustavia funktioita ja muutoksia WordPress-projekteihin sekä tukea suosituille lisäosille. Se on tarkoitettu Air lightin kanssa käytettäväksi, mutta toimii myös ilman kyseistä teemaa. (Air helper N.d.)

Tämän työn kannalta olennaisessa osassa oli vielä lisäosa Advanced Custom Fields eli ACF. ACF on lisäosa, jolla pystyy lisäämään täytettäviä kenttiä muokkausnäkymiin (Getting Started with ACF 2023). Tässä työssä sitä käytettiin henkilöiden sisältötyyppiin.

4.2 Henkilöiden sisältötyyppi

WordPressissä sisältöä voidaan luokitella sisältötyyppeihin. Englanniksi käytetään termiä custom post type, lyhennettynä CPT. WordPress sisältää oletuksena joitakin valmiita sisältötyyppejä, kuten sivut ja artikkelit. Yleensä uusi sisältötyyppi on järkevää luoda silloin, kun sisältö ei sovi muihin olemassa oleviin sisältötyyppeihin, se vaatii ominaisuuksia, joita muissa ei ole tai se täytyy esittää eri tavalla kuin artikkelit tai sivut. (Custom Post Types N.d.)

Tässä työssä henkilöille luotiin oma sisältötyyppi, johon tiedot täydennettiin automaattisesti rajapinnan kautta. Sisältötyyppi lisättiin projektiin suoraan teemaan. Air-Light-teemassa sisältötyypit rekisteröidään automaattisesti. Sisältötyypin lisääminen vaatii tekijältä lisäyksen functions.php-tiedostoon post_types-nimisen taulukon sisälle. Kuviossa 2 näkyy esimerkki kyseisestä taulukosta.



Kuvio 2. Sisältötyyppejä listattuna functions.php-tiedostossa

Tämän jälkeen sisältötyypistä luodaan tiedosto inc/post-types-kansion sisään. Air-lightin GitHub-repositoriossa (Air-light - A minimalist WordPress starter theme. N.d.) näkyy kokonaisuudessaan Air-Lightin sisältämä esimerkki sisältötyypin tiedostosta. Tähän sisältötyyppiin lisättiin kuvion 3 mukaiset arvot args-muuttujaan. Args-muuttujassa määritellään sisältötyypin ominaisuuksia. Henkilö-sisältötyypille ei haluttu omia, yksittäisiä näkymiä, joten public-kohdan arvoksi laitettiin

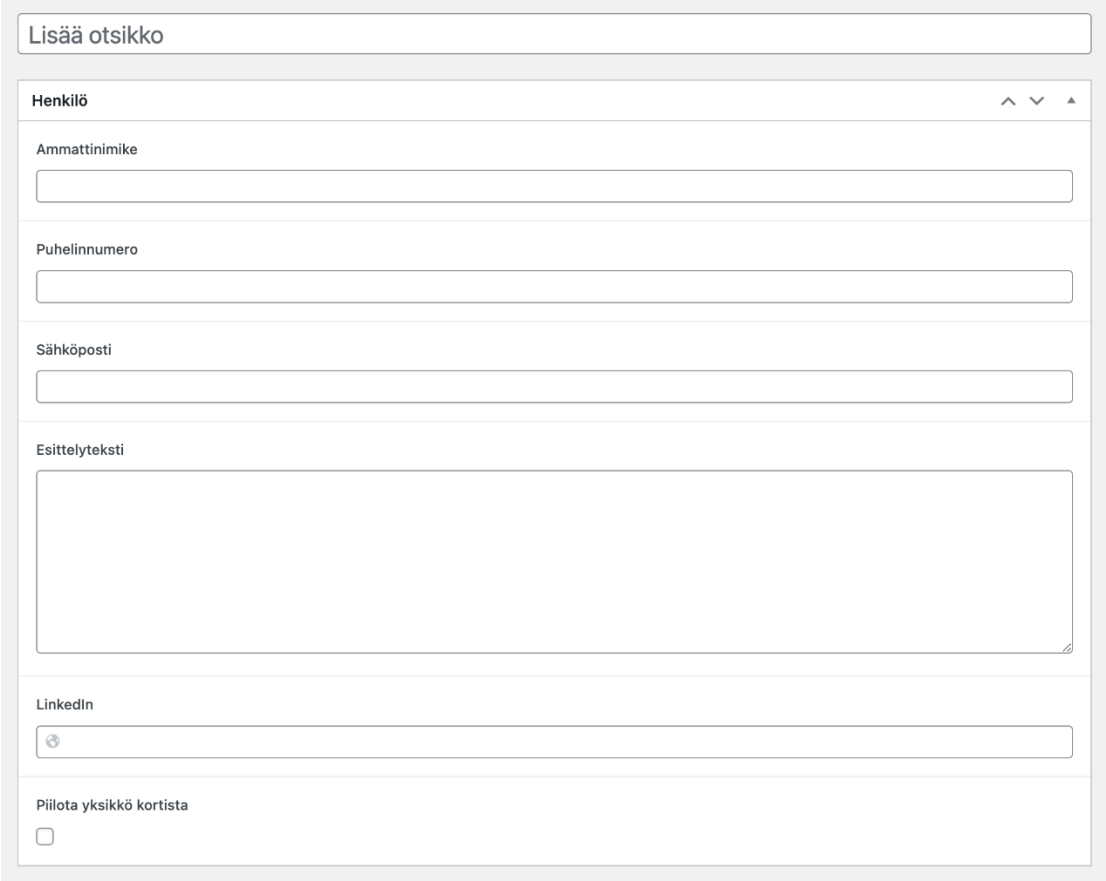
false. Samoin tehtiin has_archive-kohdalle, joka määrittelee, tuleeko sisältötyypille arkistonäkymää. Args-muuttujan arvot eivät vaikuta tämän työn integraation toimivuuteen.

```
    $args = [
        'labels'             => $generated_labels,
        'description'        => '',
        'menu_icon'          => 'dashicons-businessperson',
        'public'             => false,
        'has_archive'        => false,
        'exclude_from_search' => false,
        'show_ui'            => true,
        'show_in_menu'       => true,
        'show_in_rest'       => true,
        'rewrite'            => false,
        'supports'           => [ 'title', 'revisions' ],
        'taxonomies'         => [],
    ];

    $this->register_wp_post_type( $this->slug, $args );
  }
}
```

Kuvio 3. Sisältötyypin asetukset koodissa

Henkilöille lisättiin ACF-lisäosalla kenttiä, joihin tiedot täydennettiin rajapinnan kautta. Kentät näkyvät kuviossa 4. Suurin osa kentistä oli yksinkertaisia tekstikenttiä, paitsi kuvaus ja LinkedIn-kenttä. Kuvaukselle annettiin isompi kenttä, jotta siitä näkisi hyvin useammalle riville menevää sisältöä ja LinkedIn-osoitteelle asetettiin url-tyyppinen kenttä. Kenttien kanssa oli otettava huomioon, että niiden nimiöt vastaisivat tietokantaan tallennettujen sarakkeiden nimiä, jotta rajapinnan kautta tuleva tieto tulisi niitä vastaaviin kenttiin näkyviin.



Kuvio 4. Henkilö-sisältötyypin kentät editorissa

4.3 Henkilö-sisältötyypin taksonomiat

Henkilöiden järjestämistä ja luokittelua varten lisättiin sisältötyypille kaksi taksonomiaa, yksikkö ja koulutusala. Taksonomian rekisteröiminen tapahtui samalla tavalla kuin sisältötyypin mutta pienin eroavaisuuksin. Functions.php-tiedostossa taksonomiat lisättiin taulukkoon ”taxonomies” ja niille määriteltiin sisältötyyppi, jossa niitä haluttiin käyttää. Taulukon koodi on esitetty kuviossa 5. Taksonomian nimi vasemmalla puolella ja sisältötyypin oikealla, hakasulkujen sisällä.

```
172    'taxonomies' => [
173        'Person_Department'          => [ 'person' ],
174        'Field_Of_Study'             => [ 'person' ],
```

Kuvio 5. Taksonomioiden merkintä functions.php-tiedostoon

Kuten sisältötyypille, myös taksonomialle löytyi teemasta jo valmis malli, jonka mukaan taksonomioita pystyi lisäämään. Kumpaankin taksonomiaan luotiin suomenkieliset käännökset ja vain rewrite-kohdan arvoksi vaihdettiin false. Rewritella pystyy muuttamaan taksonomian slugia, mutta tälle ei ollut tarvetta, koska taksonomioille ei haluttu tehdä omia näkymiä. Taksonomian args-muuttujan näkee kuviosta 6.

```
45    $args = [
46        'labels'            => $labels,
47        'public'            => false,
48        'show_in_nav_menus' => true,
49        'show_admin_column' => true,
50        'hierarchical'      => true,
51        'show_tagcloud'     => false,
52        'show_ui'           => true,
53        'query_var'         => false,
54        'rewrite'           => false,
55    ];
```

Kuvio 6. Taksonomian asetukset on laitettava args-muuttujan sisään

5 Lisäosan luominen ohjelmointirajapinnalle

5.1 Micrsoft Graph

Microsoft Graph on REST API, jolla pystyy hakemaan Microsoft 365-palvelusta tietoja (Use the Microsoft Graph API 2023). Microsoftin tunnetuimpia tuotteita ovat muun muassa käyttöjärjestelmä Windows, Edge-selain sekä Microsoft Office, johon kuuluu sovelluksia kuten Word, Excel ja PowerPoint. Vuonna 2017 Microsoft julkaisi Officesta myös pilviversion, joka tunnetaan nykyään nimellä Microsoft 365. (Patrizio 2023.)

Tässä luvussa esitetään, miten rajapintayhteys testattiin ja toteutettiin WordPress-sivustolle. Rajapinnan yhdistämiseen käytettäviä arvoja tenant, client_id ja client_secret ei paljasteta, koska ne

ovat salassa pidettäviä tietoja. Rajapinnan kautta tuleva data sisältää henkilötietoja ja näin ollen sitä ei työssä näytetä tietosuojan takaamiseksi.

5.2 Rajapintayhteyden testaaminen

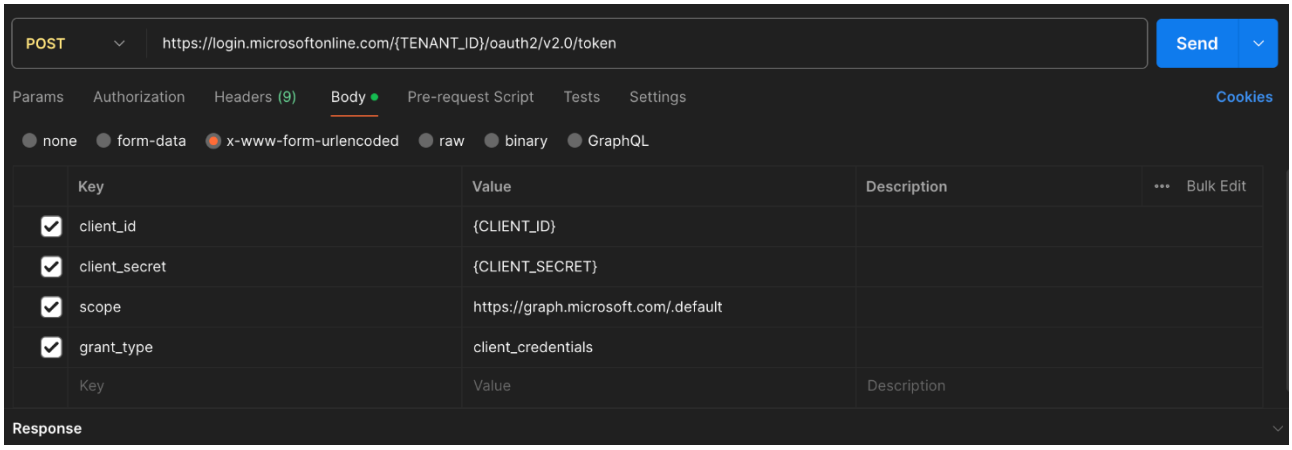
MS Graph-ohjelmointirajapintaan yhdistämistä varten on ensin hankittava autentikointia varten access token (Authentication and authorization basics 2023). Access tokenin saamiseksi on tehtävä HTTP POST-pyyntö. Pyyntöön on lisättävä viisi eri parametria, tenant, client_id, scope, client_secret ja grant_type. Katso taulukosta 1. kuvaukset eri parametreille. (Get access without a user 2024.) Tässä tapauksessa arvot kohdille tenant, client_id ja client_secret saatiin suoraan asiakkaalta.

Taulukko 1. Access tokenin pyynnön parametrit tiivistettynä

Parametri	**Kuvaus**
tenant	Kansio, johon halutaan pyytää lupaa.
client_id	Rajapinnan sovelluksen ID.
scope	Lisää kaikki luvat tokenille, jotka admin on hyväksynyt sovellukselle. Arvoksi laitetaan https://graph.microsoft.com/.default
client_secret	Rajapinnan sovellukselle generoitu client secret, verrattavissa salasanaan.
grant_type	Arvoksi tulee client_credentials.

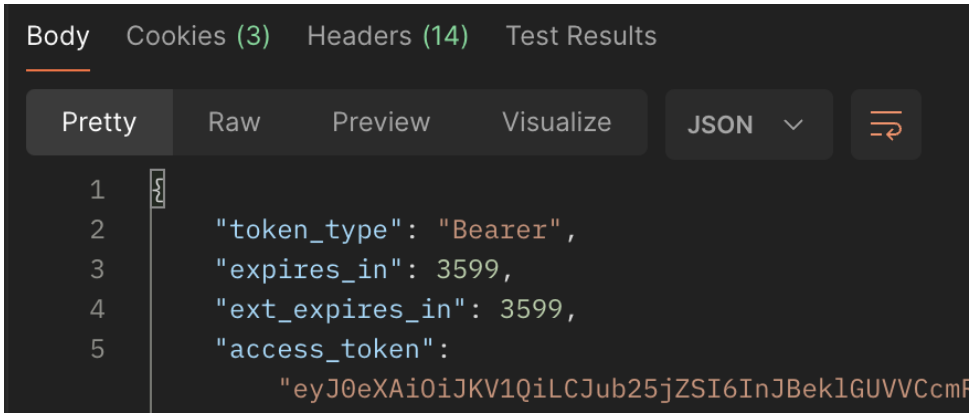
Ennen kuin rajapintaa lähdettiin käyttämään, testattiin sen toimivuus lähettämällä ensimmäinen API-kutsu Postman-sovelluksella. Näin saatiin selville, ovatko rajapinnan client_id ja client_secret toimivia. Postmanilla kutsun tekeminen toimii kuvion 7 mukaisesti. POST-pyynnön alkuun laitetaan palvelin eli https://login.microsoftonline.com. Tämän jälkeen tulee tenantin arvo, oauth2, rajapinnan versionumero ja loppuun token. Body-välilehdelle lisätään Key-sarakkeeseen taulukon

1 parametrit ja viereiseen Value-sarakkeeseen arvot lukuunottamatta tenantia. Yhdelle riville tulee aina yksi parametri sekä parametria vastaava arvo.



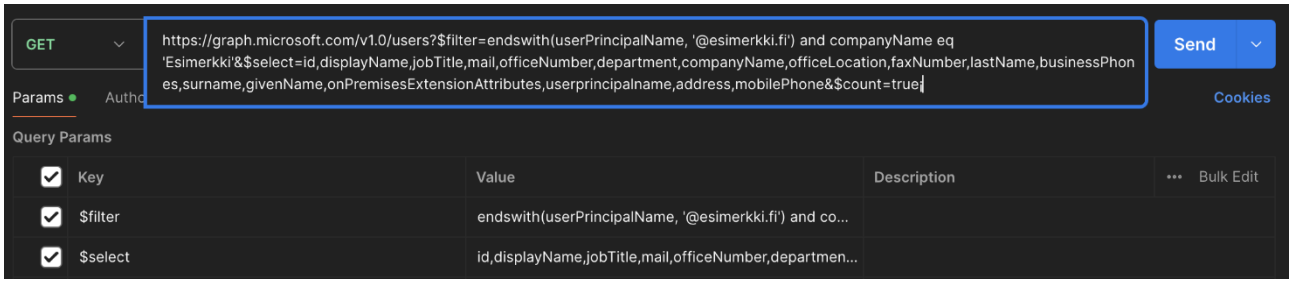
Kuvio 7. POST-pyyntö Postmanin käyttöliittymässä

Jos pyyntö onnistuu, palauttaa pyyntö access tokenin. Tämän lisäksi palaute sisältää tokenin tyypin ja tokenin voimassaoloajan (ks. Kuvio 8).



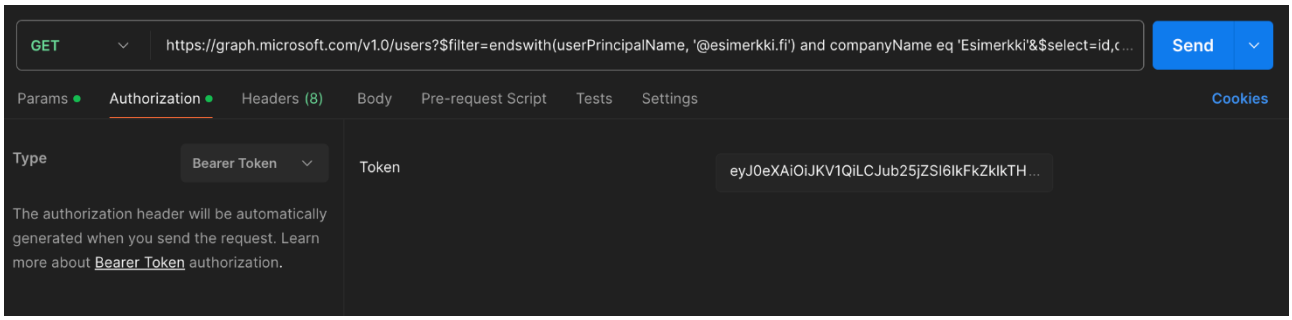
Kuvio 8. POST-pyynnön palaute Postmanissa

Kun access token on saatu, lisätään GET-pyyntö, jolla haetaan rajapinnasta tietoja (ks. Kuvio 9). Pyynnön alkuun tulee https://graph.microsoft.com/v1.0/users. Tämän jälkeen on lisättävä filtterit eli tiedot, millä hakua halutaan rajoittaa. Filtterien jälkeen tulee select-parametri, joka kertoo, mitä tietoja halutaan rajapinnasta palauttaa. Kutsuun voi lisätä myös muita parametreja. Esimerkiksi count palauttaa tiedon siitä, kuinka monta henkilöä rajapintahaku palauttaa.



Kuvio 9. GET-pyyntö Postmanissa

Tämän lisäksi täytyy lisätä aiemmin haettu access token. Postman-sovelluksessa tokenille löytyy kenttä Authorization-välilehdeltä. Tokenille on myös valittava oikea tyyppi. Tyypin näkee POST-kutsusta kohdasta “type”, eli tässä tapauksessa valitaan Bearer token (ks. Kuvio 10). Headers-välilehdeltä tarkistettiin, että sieltä löytyi parametri ConsistencyLevel, jonka arvona oli eventual.



Kuvio 10. Tokenin lisääminen kutsuun Postmanissa

Tämän jälkeen kutsu on valmis ja se on lähetettävissä. Rajapinnan pitäisi palauttaa ne tiedot, jotka pyyntöön on asetettu. Esimerkki palautteesta näkyy kuviossa 11. Palautteen arvot ovat sumennettuina, koska ne sisältävät henkilötietoja.

Kuvio 11. Rajapinnan palaute

5.3 Lisäosan tekeminen rajapinnalle

Lisäosan pohjana käytettiin Digitoimisto Duden kehittämää Item Sync Plugin Base-lisäosaa. Lisäosasta saatiin työlle valmis rakenne, jonka päälle tehtiin funktioita esimerkiksi tietojen poistamiseen, lisäämiseen ja API-kutsun muodostamiseen. Muut toiminnot toimivat halutusti ilman muokkauksia.

Lisäosa lisättiin projektiin lataamalla GitHubista ZIP-kansio osoitteesta https://github.com/digitoimistodude/item-sync-plugin-base. ZIP-kansio purettiin ja vietiin projektin sisällä olevaan /wp-content/plugins-kansioon. Kun tämä oli tehty, lisäosan nimi vaihdettiin nimeen MS Graph Contact Sync. Näin se kuvasi lisäosan tarkoitusta ja toimintaa paremmin. Lisäosan nimi löytyi neljässä eri muodossa tiedostoista: Item Sync Plugin Base, item-sync-plugin-base, Item_Sync_Plugin_Base ja item_sync_plugin_base. Nämä kaikki korvattiin uudella nimellä. Selkeyden ja koodin toimivuuden

kannalta oli tärkeää pitää huolta nimeä vaihtaessa, että isot kirjaimet ja muut merkit pysyivät yhtäläisenä.

Lisäosan juuresta löytyi vielä tiedosto nimeltä item-sync-plugin-base.php. Myös tämän nimi vaihdettiin vastaamaan lisäosan nimeä eli muotoon ms-graph-contact-sync.php. Tämä oli lisäosan päätiedosto. Päätiedoston pitää sisältää Header-kommentti ja siitä on löydyttävä vähintään lisäosan nimi. Wordpress.org-sivustolla on esitelty kaikki mahdolliset kentät, mitä kommenttiin voi lisätä (ks. Header Requirements 2024).

On suositeltavaa, että lisäosan globaalisti saavutettavassa koodissa käytetään prefiksiä. Globaalisti saavutettava koodi on mahdollista yliajaa missä tahansa muualla koodissa ja se voi aiheuttaa konflikteja muiden lisäosien kanssa, jos niissä sattuu olemaan esimerkiksi samanniminen funktio. (Best Practices 2024) Tätä varten lisäosan päätiedostosta löytyi apufunktio get_prefix(), joka palauttaa prefiksin nimen. Apufunktio, englanniksi helper function, on funktio, joka on tarkoitettu muiden funktioiden käytettäväksi. Kovakoodatut tiedot on hyvä laittaa omiin apufunktioihinsa. Silloin kovakoodatut tiedot voi päivittää vain funktion sisään ja funktiota kutsua kaikissa niissä paikoissa, joissa kovakoodattua tietoa tarvitaan.

Prefiksin nimeksi vaihdettiin msgraph_contact_sync. Tiedostosta löytyi myös apufunktio nimeltä get_cpt_slug ja tässä kohtaa sinne laitettiin sisältötyypin slug, johon tiedot haluttiin lisätä, eli person. Funktiot on esitetty kuviossa 12.

```
30  function get_prefix() {
31    return 'msgraph_contact_sync';
32  } // end get_prefix
33
34  function get_cpt_slug() {
35    return 'person';
36  } // end get_cpt_slug
37
38  function get_tax_mappings() {
39    return [
40      'department' => 'person_department',
41      'field_of_study' => 'field_of_study',
42    ];
43  } // end get_tax_mappings
44
```

Kuvio 12. Lisäosan päätiedoston tärkeitä apufunktioita

5.3.1 Yhteyden muodostaminen

Yhteyden muodostamista varten oli muokattava tiedostoa nimeltä request.php, joka löytyi inc-kansion alta. Tiedostossa ensimmäisenä vastaan tuli call_api-funktio, johon lisättiin kaksi uutta parametria, endpoint ja return_raw. Parametri endpoint sisälsi rajapinnan yksilöllisen client_id:n, jota tarvittiin API-osoitteen muodostamiseen. Funktion alkuosaan haettiin API-avain ja API-osoite muuttujiin api_key ja api_url funktioilla, joita esitellään myöhemmin tässä luvussa. API-osoitteen loppuun lisättiin endpoint-parametri. Osoite toteutettiin näin, jotta se olisi helpompi ylläpitää. Tämän jälkeen varmistettiin, että API-avain ei ole tyhjä, jotta vältyttäisiin ongelmilta myöhemmissä kohdissa palauttamalla tyhjä return, joka keskeyttää funktion suorittamisen.

Mikäli parametri params sisältää jotain, lisätään sen sisältö URLiin. Lopuksi rajapinta vaatii kahta headeria kutsuun, Authorization ja ConsistencyLevel. Nämä lisättiin WordPressin funktiolla wp_parse_args oletusarvoiksi kutsuun. Authorization-kohtaan lisättiin rajapinnan toimimiseen tarvittava Bearer sekä API-avain. ConsistencyLevel-kohtaan eventual, kuten tehtiin luvussa 4.1 rajapintaa testatessa Postmanilla. Kaikki call_api-funktioon tehdyt muutokset ovat nähtävissä kuviossa 13.

```php
function call_api( $endpoint, $params = [], $args = [], $return_raw = false ) {
    $api_key = get_api_key();
    $api_url = get_api_url() . "/{$endpoint}";

    if ( empty( $api_key ) ) {
        return;
    }

    // Add parameters to url
    if ( ! empty( $params ) ) {
        $api_url = add_query_arg( $params, $api_url );
    }

    // Merge headers to args
    $args = wp_parse_args( $args, [
        'headers' => [
            'Authorization'     => "Bearer {$api_key}",
            'ConsistencyLevel'  => 'eventual',
        ],
    ] );
```

Kuvio 13. Muutokset funktioon call_api

Seuraavaksi esitellään, miten API-avain haettiin API-kutsuun. API-avaimen funktio get_api_key (ks. Liite 1) palautti rajapintakutsuun tarvittavan access tokenin ja tätä varten funktioon lisättiin POST-pyyntö, jolla access token saatiin selville. Sitä ennen tarkistettiin, löytyikö tokenia jo välimuistista ja jos löytyi, se haettiin sieltä. Näin POST-pyyntöä ei tarvinnut tehdä turhaan. Token on hyvä tallentaa välimuistiin myös tietoturvan kannalta. Kutsun aikana rajapintaan on mahdollista hyökätä, joten mitä vähemmän pyyntöjä tarvitsee tehdä, sen parempi.

Pyyntöön tarvittiin url, joka sisälsi asiakkaan rajapinnan tenant_id:n, jotta pyyntö pystyttiin suorittamaan oikeaan paikkaan. Tätä varten lisättiin funktio get_api_oauth_url helpers.php-tiedostoon, joka oli tarkoitettu apufunktioille. Kyseistä osoitetta käyttämällä saadaan myös todennettua käyttöoikeudet rajapintaan parametrien avulla. Kuviosta 14 näkee get_api_url- ja get_api_oauth_url-funktiot.

```php
function get_api_url() {
  return 'https://graph.microsoft.com/v1.0';
} // end get_api_url

function get_api_oauth_url() {
  $tenant_id = get_api_tenant();
  if ( empty( $tenant_id ) ) {
    log( 'Tenant ID not configured in environment', 'error' );
    return false;
  }

  return "https://login.microsoftonline.com/{$tenant_id}/oauth2/v2.0/token";
} // get_api_oauth_url
```

Kuvio 14. Funktiot API-osoitteiden hakemiseen

API-osoitteen hakemiseen käytetty get_api_url sijaitsi helpers.php-tiedostossa ja siihen vaihdettiin funktion palautusarvoksi luvussa 4.1 mainittu https://graph.microsoft.com/v1.0, jolla kaikkien kyseisen rajapinnan kutsujen on alettava. Yhteys on määritetty käyttämään https-yhteyttä, koska http on suojaamaton yhteys.

Helpers.php-tiedostoon lisättiin myös apufunktiot tenant id:lle, client id:lle ja client secretille (ks. kuvio 15). Näiden kohtien tiedot on tärkeää pitää hyvin suojattuna ja niitä on kohdeltava samoin kuin salasanoja ja käyttäjätunnuksia. Tästä syystä niiden tiedot lisättiin .env-tiedostoon ja haettiin suoraan sieltä.

.env-tiedosto on projektin ympäristön tietojen tallennukseen tarkoitettu tiedosto. Tässä työssä tiedosto lisättiin projektin alussa sitä luodessa. Kyseiseen tiedostoon tallennetaan usein tietoja esimerkiksi WordPress-projektin asetuksista ja tietokantayhteyden muodostamiseen tarvittavat tiedot. .env-tiedostoa ei tule lisätä esimerkiksi GitHubiin tai julkaista minnekään, mistä ulkopuolinen

taho voisi sen nähdä, jotta tiedot pysyvät salassa. Tiedoston turvallisuudesta on hyvä huolehtia samalla tavalla kuin salasanojen. Jos tiedostosta haluaa tallentaa kopion tai jakaa muille version, kannattaa käyttää hyvin turvattua alustaa, kuten salasananhallintasovellusta.

```
34  function get_api_tenant() {
35      return getenv( 'MS_GRAPH_TENANT_ID' );
36  } // end get_api_tenant
37
38  function get_api_client_id() {
39      return getenv( 'MS_GRAPH_CLIENT_ID' );
40  } // end get_api_client_id
41
42  function get_api_client_secret() {
43      return getenv( 'MS_GRAPH_CLIENT_SECRET' );
44  } // end get_api_client_secret
```

Kuvio 15. Miten salassa pidettävät tiedot haettiin .env-tiedostosta

POST-pyyntöön lisättiin luvussa 4.1 esitellyt tiedot wp_remote_request-funktioon. Se on WordPressin oma funktio, joka suorittaa API-kutsun. Jos kutsu menee läpi ilman error-koodia, haetaan kutsun sisältö funktiolla wp_remote_retrieve_body. Kun saatu data purettiin json-muodosta taulukoksi, sen sisällöstä saatiin access token. Lopussa asetettiin token myös välimuistiin.

5.3.2 Testaaminen koodissa

Tässä vaiheessa yhteyttä haluttiin testata koodin puolella ja tämä toteutettiin ajamalla WP-CLI-komento ”wp msgraph-contact-sync --force”, joka ajoi lisäosan rajapintakutsun. WP-CLI-komennot alkavat aina kirjaimilla wp ja tämän jälkeen tulee lisäosan prefix. Prefixin näkee kuviosta 12 get_prefix-funktion sisältä. Viimeisenä komento tarvitsi lisäyksen --force, joka pakottaa komennon ajamisen. Komento on asetettu toimimaan ajastuksella ja ajastetut komennot vaativat pakotuksen, jos niitä ajetaan manuaalisesti.

Koodiin on lisätty tarkistuksia tyhjille ja ei-halutuille arvoille, jotka tulostavat komentoriville viestejä ja keskeyttävät koodin ajamisen. Näin ehkäistään koodin ajaminen silloin, jos arvojen sisältö on virheellistä ja se aiheuttaisi lisää virheitä myöhemmin koodissa. Esimerkiksi access_tokenia kutsuttaessa tarkistetaan, että API-pyyntö antaa palautteena 200, koska muut arvot viittaisisivat virheelliseen tulokseen ja siihen, ettei access tokenia saatu (ks. kuvio 16).

Työtä tehdessä komentoa ajettiin usein muutoksien jälkeen. Lokitekstejä lisättiin väliaikaisesti tarpeen tullen myös lisää log-funktiolla, jos ongelmia ilmeni eikä olemassa olevista ollut tarpeeksi apua vian selvittämiseen. Esimerkiksi muuttujien sisällön tulostaminen auttoi usein hahmottamaan, missä kohtaa koodin tulos aiheutti ongelmia.

5.3.3 Tietojen haku ja tallennus

Tietojen tallentaminen sisältötyyppiin toteutettiin item.php-tiedostossa. Save-muuttujaan lisättiin meta_input-kohdan sisään ne kentät, jotka haluttiin lisätä tietokantaan. Näitä olivat etunimi, sukunimi, sähköpostiosoite, puhelinnumero, osasto ja koulutusala. Joillakin henkilöillä oli myös toissijainen työnimike. Tämä oli tallennettu rajapinnassa kenttään extensionAttribute3. Jos kenttä ei ollut tyhjä, tallennettiin vielä toissijainen työnimike omaan kenttäänsä. Koodi on esitetty kuviossa 16.

```php
function save_item( $item, $force ) {
    // Get title from custom made function
    $post_title = construct_item_title( $item );
    if ( empty( $post_title ) ) {
        $post_title = $item['id'];
    }

    // Values to save into a database. Keys on the left and corresponding values on the right
    $save = [
        'ID'           => $item_post_id,
        'post_type'    => get_cpt_slug(),
        'post_status'  => 'publish',
        'post_title'   => $post_title,
        'meta_input'   => [
            prefix_key( 'sync_id', true )   => $item['id'],
            prefix_key( 'sync_time', true ) => wp_date( 'Y-m-d H:i:s' ),
            $data_hash_key                  => $data_hash,
            'first_name'                    => $item['givenName'],
            'surname'                       => $item['surname'],
            'email'                         => mb_strtolower( $item['userPrincipalName'] ),
            'phone'                         => $item['mobilePhone'],
            'job_title'                     => ucfirst( $item['jobTitle'] ),
            'department'                    => $item['department'],
            'field_of_study'                => $item['onPremisesExtensionAttributes']
            ['extensionAttribute1'],
        ],
    ];

    // Save secondary job title only if it exists      You, 2 months ago • Uncommitted changes
    if ( ! empty( $item['onPremisesExtensionAttributes']['extensionAttribute3'] ) ) {
        $save['meta_input']['job_title_secondary'] = ucfirst( $item
        ['onPremisesExtensionAttributes']['extensionAttribute3'] );
    }
```

Kuvio 16. Tallennettavat tiedot yksittäiselle henkilölle

Jokaiselle henkilölle haluttiin antaa otsikoksi etunimi ja sukunimi. Tätä tarkoitusta varten lisättiin apufunktio, jolla otsikko muodostettiin. Jos jompikumpi nimistä puuttui, palautettiin pelkästään etunimi tai sukunimi. Jos kummatkin puuttuivat, annettiin arvoksi false. Kun save-muuttujaan saatiin kaikki tiedot, mitä haluttiin, lisättiin sen tietojen perusteella uusi henkilö tietokantaan WordPressin funktiolla wp_insert_post. Funktio on nähtävissä kuviossa 17.

```
114
115  function construct_item_title( $item ) {
116    if ( ! empty( $item['givenName'] ) && ! empty( $item['surname'] ) ) {
117      return "{$item['givenName']} {$item['surname']}";
118    }
119
120    if ( ! empty( $item['givenName'] ) ) {
121      return $item['givenName'];
122    }
123
124    if ( ! empty( $item['surname'] ) ) {
125      return $item['surname'];
126    }
127
128    return false;
129  } // end construct_item_title
```

Kuvio 17. Henkilön otsikon muodostaminen etu- ja sukunimen perusteella

Henkilöille lisättiin aiemmin luvussa 3.3 kaksi eri taksonomiaa, yksikkö (department) ja koulutusala (field of study). Tietojen tallentamista varten tehtiin lisäys päätiedoston funktioon get_tax_mappings. Palautettavan taulukon avaimien kohdille asetettiin kenttien nimet, jotka haluttiin taksonomioihin tallentaa. Arvoiksi annettiin kenttiä vastaavien taksonomioiden slugit. Kuviossa 18 on esitetty kyseinen kohta koodista.

```
38  function get_tax_mappings() {
39    return [
40      'department' => 'person_department',
41      'field_of_study' => 'field_of_study',
42    ];
43  } // end get_tax_mappings
44
```

Kuvio 18. Henkilöiden taksonomiat, joihin tietoa halutaan tuoda

Taksonomian termien asettamiseen tehtiin funktio set_item_terms (ks. kuvio 19), joka kävi get_tax_mappings-funktion taulukon arvot. Funktiolle syötettiin parametrina suoraan rajapinnan kautta tulevan yksittäisen henkilön tiedot. Tämän jälkeen tarkistettiin, löytyikö henkilön tiedoista yksikköä tai koulutusalaa. Jos löytyi, tallennettiin löydetty termi kyseille henkilölle. Jos henkilön kenttä esimerkiksi yksikölle oli tyhjä, ajettiin varmuuden vuoksi termien poisto tietokannasta, koska henkilöllä on saattanut aiemmin olla yksikkö, joka on myöhemmin poistettu.

```php
function set_item_terms( $item ) {
  // Get tax mappings from the main plugin file
  $tax_mappings = get_tax_mappings();

  // Process each taxonomy
  foreach ( $tax_mappings as $data_key => $tax ) {
    // Process the department taxonomy
    if ( false !== strpos( $data_key, 'department' ) ) {
      // If a department of a person has empty value, remove department from database
      if ( empty( $item['department'] ) ) {
        maybe_remove_old_terms( $item['wp_post_id'], $data_key );
        continue;
      }

      $value = $item['department'];
    } elseif ( false !== strpos( $data_key, 'field_of_study' ) ) {
      // Remove the field of study from database if API returns empty value
      if ( empty( $item['onPremisesExtensionAttributes']['extensionAttribute1'] ) ) {
        maybe_remove_old_terms( $item['wp_post_id'], $data_key );
        continue;
      }

      $value = $item['onPremisesExtensionAttributes']['extensionAttribute1'];

      // Split string by ; and save values in array
      $value = preg_split( '/;/', $value );
    } else {
      if ( empty( $item[ $data_key ] ) ) {
        continue;
      }

      $value = $item[ $data_key ];
    }

    // Allow handling save via filter & short circuit normal save
    $value = apply_filters( prefix_key( "item/term/{$data_key}" ), $value, $item );

    $terms = [];

    // If terms are in an array, loop through the array. Try to save each term into database
    if ( is_array( $value ) ) {
      foreach ( $value as $k => $v ) {
        $term = maybe_save_tax_term( strval( $v ), $tax );

        if ( ! $term ) {
          continue;
        }

        // Add term id to terms array
        $terms[] = intval( $term );
      }
    } else {
      $term = maybe_save_tax_term( strval( $value ), $tax );

      if ( ! $term ) {
        continue;
      }

      $terms[] = intval( $term );
    }

    // Set terms for person
    $term_set = wp_set_post_terms( $item['wp_post_id'], $terms, $tax );
```

Kuvio 19. set_item_terms-funktio, jolla lisättiin henkilöille termit

Termien tallentamiseen käytettiin kuvion 20 mukaista funktiota. Jos termi oli jo olemassa, käytettiin sitä. Jos termiä ei ollut, lisättiin se WordPressin funktioita käyttäen. Termin tallennukseen oli

myös rakennettu mahdollisuus tallentaa muuta dataa, mutta tässä tapauksessa sitä ei hyödynnetty.

```
75
76  function maybe_save_tax_term( $term = null, $taxonomy = null, $metadata = [] ) {
77    if ( empty( $term ) || empty( $taxonomy ) ) {
78      return false; // Bail early if there's no term.
79    }
80
81    // Check if term exists in wp. If does, return term id.
82    $term_exists = term_exists( $term, $taxonomy );
83    if ( ! empty( $term_exists ) ) {
84      return $term_exists['term_id'];
85    }
86
87    // Term didn't exist, try to insert it into wp.
88    $insert_term = wp_insert_term( $term, $taxonomy );
89    if ( is_wp_error( $insert_term ) ) {
90      // Term insert failed, log it and bail.
91      log( "Failed saving term {$term} to {$taxonomy}", 'debug', $insert_term );
92      return false;
93    }
94
95    if ( ! empty( $metadata ) ) {
96      foreach ( $metadata as $key => $value ) {
97        update_term_meta( $insert_term['term_id'], $key, $value );
98      }
99    }
100
101   return $insert_term['term_id'];
102 } // end maybe_save_tax_term
```

Kuvio 20. Taksonomian termin tallennus

Termien poistamiselle lisättiin oma funktio maybe_remove_old_terms (ks. kuvio 21). Sillä yksinkertaisesti tarkistettiin, löytyikö tietokannasta henkilöltä entuudestaan tietyn taksonomian termejä. Jos ei löytynyt, funktion suorittaminen keskeytettiin. Jos löytyi, ne poistettiin WordPressin funktiolla wp_remove_object_terms.

```
103
104  function maybe_remove_old_terms( $id, $tax_name ) {
105    $old_terms = wp_get_object_terms( $id, $tax_name, [ 'fields' => 'ids' ] );
106
107    if ( empty( $old_terms ) ) {
108      return;
109    }
110
111    wp_remove_object_terms( $id, $old_terms, $tax_name );
112  } // end maybe_remove_old_terms
113
```

Kuvio 21. Vanhojen termien poistaminen

Kuvien tallentamiselle käytettiin Digitoimisto Duden tekemää koodia, joka löytyy liitteestä 2. Koodissa tehdään API-kutsu jokaiselle mahdolliselle kuvakoolle ja selvitetään kuvatiedoston tyyppi. Tämän jälkeen kuva tallennetaan erilliseen kansioon ja tietokantaan tallennetaan henkilöön kuvan osoite.

Kuvien tallennuksen koodi lisättiin erilliseen item-image.php-tiedostoon. Jotta tiedosto toimi, sitä täytyi kutsua päätiedostossa kuten kuviossa 22 on esitetty. Kaikille muille tiedostoille tämä oli tehty jo valmiiksi. Päätiedostoon lisättiin uudelle tiedostolle include paikkaan, joka oli tarkoitettu yksittäisen sisällön synkronoimiselle.

```
53  /**
54   * Handlers for singular item sync and cleanup.
55   */
56  include plugin_dir_path( __FILE__ ) . '/handlers/item.php';
57  include plugin_dir_path( __FILE__ ) . '/handlers/item-terms.php';
58  include plugin_dir_path( __FILE__ ) . '/handlers/item-image.php';
59  include plugin_dir_path( __FILE__ ) . '/handlers/cleanup.php';
60
```

Kuvio 22. Miten erilliset tiedostot saatiin toimimaan lisäosassa

5.3.4 Ajastus automaattiselle API-kutsulle

Tietojen synkronoinnin pystyi tekemään manuaalisesti WP-CLI-komennolla, joka löytyi valmiiksi lisäosan pohjasta. WP-CLI (WordPress command-line interface) on WordPressille tarkoitettu komentorivi, jonka avulla pystyy ajamaan komentoja. Siihen on mahdollista lisätä omia mukautettuja komentoja, kuten tähän lisäosaan oli tehty. (WP-CLI N.d)

Tietojen automaattinen päivitys hoidettiin WordPressin WP-Cron-ominaisuudella, joka on tarkoitettu ajastettuihin toimintoihin. Nimi Cron tulee UNIX-järjestelmissä olevasta ajonaikaisesta palveluiden ajoitusjärjestelmästä. Tämä osio oli koodattu projektin pohjaan jo valmiiksi cron.php-tiedostoon.

Ainoa funktio, joka vaati muutoksia kyseisessä tiedostossa, oli sync-funktio. Kyseisen funktion sisällä kutsuttiin toista funktiota nimeltä call_api ja tähän vaihdettiin kutsun parametreiksi ne arvot, joita rajapinnan kautta haluttiin saada. Muuttujalla top määritettiin, kuinka monta tulosta annetaan. Muuttuja count antoi tuloksien lukumäärän.

Tuloksiin haluttiin saada kaikki käyttäjät, joiden tili oli voimassa. Tämä onnistui, kun lisättiin filterkohtaan ”accountEnabled eq true”. Pelkästään tällä tuloksiin tuli käyttäjien lisäksi palveluiden tietoja, joita sivuille ei haluttu tuoda, joten kutsua rajattiin lisää. Yrityksen nimi laitettiin vastaamaan asiakkaan yritystä, työnimike ei saanut olla tyhjä ja käyttäjän userPrincipalName-kohdan piti sisältää tietynlainen sähköpostiosoitteen loppuosa. Näin tuloksista onnistuttiin suodattamaan pois kaikki muut paitsi käyttäjät. Kuviossa 23 näkyy, miten suodatus tarkalleen toteutettiin.

```php
function sync( $force = false ) {
  update_option( prefix_key( 'sync_end' ), null ); // set end to null for extra cleanup prevention if problems
  update_option( prefix_key( 'sync_start' ), wp_date( 'Y-m-d H:i:s' ) );

  $response = call_api( 'users', [
    '$top'    => '999', // # of users returned
    '$count'  => 'true',
    '$filter' => "accountEnabled eq true and endswith(userPrincipalName, '[redacted]') and companyName eq '[redacted]' and jobTitle ne null",
    '$select' => implode( ',', [
      'id',
      'displayName',
      'givenName',
      'surname',
      'userprincipalname', // email
      'mobilePhone',
      'companyName',
      'department',
      'jobTitle',
      'onPremisesExtensionAttributes',
    ] ),
  ] );
  if ( ! $response ) {
    return;
  }

  foreach ( $response['value'] as $item ) {
    save_item( $item, $force );
```

Kuvio 23. API-kutsu, jolla käyttäjiä haettiin

6 Työn tulokset

Lopputuloksena saatiin WordPressiin lisäosa, jolla oli toimiva yhteys Microsoft Graph-rajapintaan. Henkilöiden tiedot haettiin rajapinnan kautta automaattisesti. Vanhentuneita tietoja onnistuttiin muokkaamaan ja poistamaan tietokannasta. WP CLI-komennolla rajapintakutsu saatiin ajettua myös komentorivin kautta manuaalisesti. Komento oli erityisesti hyödyllinen testaillessa, ettei automaattista päivitystä tarvinnut odotella.

Koodin ylläpidettävyydestä pyrittiin pitämään huoli niin, että kovakoodatut arvot laitettiin omiin funktioihinsa. Näin arvot muuttuessa riittää, että arvo muutetaan vain funktion sisällä sen sijaan, että se jouduttaisiin etsimään ja vaihtamaan aina useampaan paikkaan koodissa. Muuttujien ja funktioiden nimistä yritettiin tehdä mahdollisimman hyvin kuvaavia, mutta samalla lyhyitä. Tämä helpottaa koodin ymmärrettävyyttä ja vähensi kommentoimisen tarvetta. Koodia on kommentoitu epäselvemmissä kohdissa ja funktioiden luettavuutta on helpotettu lisäämällä funktion loppuun

kommentti siitä, minkä funktion loppu on kyseessä. Koodia selatessa tämä vähentää edestakaisin rullaamista.

Rajapinnan kautta haettuja tietoja ei tässä luvussa esitellä tarkemmin. Tämä rajaus on tehty siitä syystä, että kyseinen materiaali sisältää paljon henkilötietoja. Kyseisten tietojen esittely rikkoisi henkilöiden tietosuojaa.

7 Pohdinta

Työn tavoitteena oli helpottaa asiakkaan sisällönsyöttöä hakemalla henkilöiden yhteystiedot Microsoftin palvelusta, jossa asiakas pitää tietoja ajan tasalla. Tutkimustehtävällä oli kolme kysymystä: Miten rajapintayhteys saadaan luotua, miten tiedot sen kautta haettua ja tallennettua, ja miten rajapintayhteydelle tehdään oma lisäosa. Kaikkiin kysymyksiin saatiin työssä vastattua.

Työssä saatiin toimiva rajapintayhteys Microsoft Graph-ohjelmointirajapintaan WordPress-ympäristössä. Rajapintayhteydelle ja siihen liittyville toiminnoille onnistuttiin tekemään uusi WordPress-lisäosa, joka räätälöitiin projektin tarpeisiin. Sitä ennen luotiin projektiin uusi sisältötyyppi henkilöille ja kaksi taksonomiaa, joilla henkilöitä voi jakaa koulutusaloihin ja yksiköihin. Lisäosan tekemiseen käytettiin Item Sync Plugin Base-nimistä lisäosaa pohjana. Tämän päälle tehtiin muokkauksia ja lisättiin uusia toimintoja liittyen tietojen tallentamiseen, poistamiseen ja API-yhteyden muokkaamiseen.

Henkilöiden tiedot saatiin tallennettua tietokantaan ja vanhentuneet tiedot poistettua. Automaattisessa tietojen päivityksessä tiedot jäivät välillä päivittymättä. Tämä johtui usein siitä, että tiedot jäivät herkästi välimuistiin talteen eivätkä aina itsestään poistuneet. Muuten MS Graph-rajapinta osoittautui toimivaksi ratkaisuksi sisällön hakemiseen. Työtä tehdessä huomattiin, että sen pystyi hyvin ottamaan käyttöön WordPress-ympäristöön.

Työtä on mahdollista hyödyntää muissa projekteissa, jotka vaativat MS Graph-rajapintayhteyttä. Se soveltuu mallipohjaksi, mutta ei suoraksi kopioksi. Tämän työn koodi sisältää useassa kohtaa tietoja, jotka ovat räätälöity tämän työn tarpeiden mukaan. Esimerkiksi rajapinnan kautta haetta-

vissa tiedoissa ja taksonomioiden tallennuksessa on otettu huomioon vain niihin halutut ominaisuudet. Tietojen tallentaminen on rakennettu vain asiakkaiden toiveiden ympärille eikä muita vaihtoehtoja ole lisätty.

Työn toteutusta ei voi käyttää muualla kuin WordPress-ympäristössä tai se vaatisi suuria muutoksia. Työn avulla voi kuitenkin mahdollisesti saada ymmärrystä siitä, miten rajapinta toimii, jos on tarvetta kehittää vastaava toisella ohjelmointikielellä.

Lähteet

About GitHub and Git. N.d. GitHubin dokumentaatiosivu. Viitattu 24.4.2024 https://docs.github.com/en/get-started/start-your-journey/about-github-and-git

A brief history of CSS until 2016. N.d. Artikkeli w3.org-sivustolla. Viitattu 15.4.2024 https://www.w3.org/Style/CSS20/history.html

Air helper. N.d. Air helper-lisäosan Github-repositorion esittelyteksti. Viitattu 30.3.2024 https://github.com/digitoimistodude/air-helper

Air-light - A minimalist WordPress starter theme. N.d. Air-light-lisäosan Github-repositorio. Viitattu 30.4.2024 https://github.com/digitoimistodude/air-light

Authentication and authorization basics. 2023. Artikkeli Microsoftin sivustolla. Viitattu 30.3.2024 https://learn.microsoft.com/en-us/graph/auth/auth-concepts

Bellis, M. 2019. The History of HTML and How It Revolutionized the Internet. Artikkeli thoughtco.com-sivustolla. viitattu 14.4.2024 https://www.thoughtco.com/history-of-html-1991418

Best Practices. 2024. WordPressin dokumentaatiosivu lisäosista. Viitattu 19.4.2024 https://developer.wordpress.org/plugins/plugin-basics/best-practices/

Chobanyan, H. 2020. SASS for WordPress developers: here's everything you MUST know. 10webin artikkeli. Viitattu 16.4.2024 https://10web.io/blog/wordpress-sass/

Code Editing. Redefined. n.d. Visual Studio Coden etusivu. Viitattu 24.4.2024 https://code.visualstudio.com/

Custom Post Types. N.d. Opetusmateriaali learn.wordpress.org-sivustolla. Viitattu 15.4.2024 https://learn.wordpress.org/lesson-plan/custom-post-types/

Dobrilova, T. 2024. 20+ WordPress Statistics to Know in 2024. Scalahosting.com-sivustolla julkaistu blogi. Viitattu 14.5.2024 https://www.scalahosting.com/blog/20-wordpress-statistics-to-know/

Dudestack. N.d. GitHub-repositorion kuvaus. Viitattu 23.4.2024 https://github.com/digitoimistodude/dudestack

Features. N.d. Verkkosivu WordPressin ominaisuuksista. Viitattu 24.3.2024 https://wordpress.org/about/features

Friedman, N. 2021. Introducing GitHub Copilot: your AI pair programmer. Artikkeli Github.blog-sivustolla, jossa esitellään GitHub Copilotia. Viitattu 30.3.2024 https://github.blog/2021-06-29-introducing-github-copilot-ai-pair-programmer/

Friesbie, M. 2024. Professional JavaScript for Web Developers. 5. p. Hoboken, New Jersey: John Wiley & Sons, Inc.

Get access without a user. 2024. Artikkeli Microsoft Graph-rajapinnasta. Viitattu 31.3.2024 https://learn.microsoft.com/en-us/graph/auth-v2-service?tabs=http

Getting Started with ACF. 2023. Dokumentaatiosivu advancedcustomfields.com-sivustolla. Viitattu 20.4.2024 https://www.advancedcustomfields.com/resources/getting-started-with-acf/

Hasib. 2024. WordPress Block Editor vs Classic Editor — A Detailed Comparison. Artikkeli wpmet.com-sivustolla. Viitattu 30.3.2024 https://wpmet.com/block-editor-vs-classic-editor

Header Requirements. 2024. WordPressin dokumentaatiosivu. Viitattu 19.4.2024 https://developer.wordpress.org/plugins/plugin-basics/header-requirements/

History of PHP. N.d. PHP:n dokumentaatiosivu. Viitattu 14.4.2024 https://www.php.net/manual/en/history.php.php

How To Get Started With SCSS: A Beginner Guide. 2024. Artikkeli coders.dev-sivustolla. Viitattu 15.4.2024 https://www.coders.dev/blog/how-to-get-started-with-scss-a-beginner-guide.html

HTML. N.d. Artikkeli HTML-kielestä Mozillan kotisivuilla. Viitattu 7.4.2024 https://developer.mozilla.org/en-US/docs/Glossary/HTML

Learn to style HTML using CSS. N.d. Artikkeli CSS-kielestä Mozillan kotisivuilla. Viitattu 7.4.2024 https://developer.mozilla.org/en-US/docs/Learn/CSS

Newcomer, C. 2023. How to Use WordPress Plugins: The Complete Beginner's Guide. Artikkeli WordPress.com-sivustolla. Viitattu 30.3.2024. https://wordpress.com/go/website-building/how-to-use-wordpress-plugins/

Patrizio, A. 2023. Microsoft. Artikkeli techtarget.com-sivustolla. Viitattu 16.4.2024 https://www.techtarget.com/searchwindowsserver/definition/Microsoft

Pernaa, J. 2013. Kehittämistutkimus tutkimusmenetelmänä. Jyväskylä: PS-kustannus. Viitattu 14.5.2024 https://helda.helsinki.fi/server/api/core/bitstreams/fd4fcd23-2d7c-474a-a426-438c93075ff3/content

Speed up your WordPress development. N.d. Air-Light-teeman kotisivu. Viitattu 30.3.2024 https://airwptheme.com/

Understanding Themes. 2024. Verkkosivu WordPressin teemoista. Viitattu 31.3.2024 https://wordpress.com/learn/courses/getting-started/understanding-themes/

Use the Microsoft Graph API. 26.7.2023. Artikkeli Microsoftin sivustolla. Viitattu 24.3.2024 https://learn.microsoft.com/en-us/graph/use-the-api

Murphy, A & Tucker, H. 2023. The Global 2000. Forbes-talouslehden artikkeli. Viitattu 16.4.2023 https://www.forbes.com/lists/global2000/

W3Techs. N.d. W3Techsin etusivu. Viitattu 30.3.2024 https://w3techs.com

What Is a Content Management System (CMS)? 2023. Artikkeli sisällönhallintajärjestelmistä. Viitattu 6.4. 2024 https://kinsta.com/knowledgebase/content-management-system/

What is JavaScript? N.d. Artikkeli JavaScript-kielestä Mozillan kotisivuilla. Viitattu 7.4.2024 https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

What is MySQL? N.d. Artikkeli Oracle.com-sivustolla. Viitattu 7.4.2024 https://www.oracle.com/mysql/what-is-mysql/

What is PHP? N.d. PHP:n dokumentaatiosivu. Viitattu 7.4.2024 https://www.php.net/manual/en/intro-whatis.php

What is Postman? n.d. Verkkosivu Postmanin kotisivuilla. Viitattu 24.4.2024 https://www.postman.com/product/what-is-postman/

What Is WordPress? Explained for Beginners. 2023. Artikkeli WordPressistä. Viitattu 6.4.2024 https://kinsta.com/knowledgebase/what-is-wordpress/

WordPress. N.d. WordPressin kehitys-repositorio. Viitattu 7.4.2024 https://github.com/WordPress/wordpress-develop

WP-CLI. N.d. wp-cli.orgin etusivu. Viitattu 15.4.2024 https://wp-cli.org/

Liitteet

Liite 1

```php

function get_api_key() {
    // get API Token from cache
    $api_key = get_transient( prefix_key( 'api_key', true ) );
    if ( ! empty( $api_key ) ) {
        log( 'API Token obtained from internal cache', 'debug' );
        return $api_key;
    }

    // get authentication url
    $api_url = get_api_oauth_url();
    if ( empty( $api_url ) ) {
        log( 'Could not do API token call', 'error' );
        return false;
    }

    $args = [
        'method'      => 'POST',
        'body'        => [
            'client_id'     => get_api_client_id(),
            'client_secret' => get_api_client_secret(),
            'scope'         => 'https://graph.microsoft.com/.default',
            'grant_type'    => 'client_credentials',
        ],
    ];

    // Call api and save response to a variable
    $response = wp_remote_request( $api_url, $args );

    // Check if request is successfully completed, otherwise bail
    $response_code = wp_remote_retrieve_response_code( $response );
    if ( 200 !== $response_code ) {
        log( "API key get returned error code {$response_code}", 'error' );
        return false;
    }

    // Get body of the response, it contains access token
    $body = wp_remote_retrieve_body( $response );

    if ( empty( $body ) ) {
        log( 'API key get returned empty body', 'debug' );
        return false;
    }

    // Decode json into a php value
    $data = json_decode( $body, true );

    if ( ! $data || ! isset( $data['access_token'] ) ) {
        log( 'API Token could not be obtained', 'error', $data );
        return false;
    }

    // Add access token to cache
    set_transient( prefix_key( 'api_key', true ), $data['access_token'], $data['expires_in'] );

    return $data['access_token'];
} // end get_api_key

```

Liite 2

```php

defined( 'ABSPATH' ) || exit;

function save_person_image( $item, $force = false ) {
  $image_last_update = get_post_meta( $item['wp_post_id'], prefix_key( 'image_updated', true ), true );

  /**
   * Update image only if has not been done in two days.
   */
  if ( ! $force && strtotime( '-2 days' ) < strtotime( $image_last_update ) ) {
    log( 'Image update skipped', 'debug' );
    return false;
  }

  /**
   * Save the sync time here so we don't end up trying the update repeatably if
   * there's no image or if saving fails.
   */
  update_post_meta( $item['wp_post_id'], prefix_key( 'image_updated', true ), wp_date( 'Y-m-d H:i:s' ) );

  log( 'Image update started', 'debug' );

  $sizes = [
    '648x648',
    '504x504',
    '432x432',
    '360x360',
    '240x240',
    '120x120',
    '96x96',
    '64x64',
    '48x48',
  ];

  foreach ( $sizes as $size ) {
    log( "Trying size {$size}", 'debug' );

    $response = call_api( "users/{$item['id']}/photos/{$size}/" . '$value', [], [], true );
    if ( ! $response ) {
      log( "Image not found for size {$size}", 'debug' );
      continue;
    }

    break;
  }

  if ( ! $response ) {
    return false;
  }

  /**
   * Images might be in different format, so get the righ file extension
   * from content-type headers.
   */
  $content_type = wp_remote_retrieve_header( $response, 'content-type' );
  $file_extension = explode( '/', $content_type )[1];

  /**
   * Put filename and path together.
   */
  $upload_dir = wp_get_upload_dir();
  $filename = sanitize_title( $item['displayName'] ) . '.' . $file_extension;
  $file_path = trailingslashit( $upload_dir['basedir'] ) . 'personnel';

  $save = file_put_contents( trailingslashit( $file_path ) . $filename, wp_remote_retrieve_body( $response ) );

  if ( is_int( $save ) ) {
    log( 'Image updated', 'debug' );

    update_post_meta( $item['wp_post_id'], prefix_key( 'image_file_name', true ), $filename );

    return $filename;
  } else {
    log( 'Image update failed', 'error' );
    return false;
  }
} // end save_person_image

```