Dani Kuikka

OAuth 2.0 ja delegoitu pääsy mikro-palveluna

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

3.12.2021

Tiivistelmä

Tekijä:	Dani Kuikka
Otsikko:	OAuth 2.0 ja delegoitu pääsy mikropalveluna
Sivumäärä:	34 sivua
Aika:	3.12.2021
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Ohjelmistotuotanto
Ohjaajat:	Lehtori Vesa Ollikainen Tuotepäällikkö Juho Salmi

Tässä työssä tutkitaan, kuinka OAuth 2.0 -standardia voidaan hyödyntää pääsynhallinnassa käyttäen Microsoftin Identiteettipalvelua ja Msal-kirjastoa. Työssä tutustutaan OAuth-yhteyskäytännön tärkeimpiin piirteisiin ja ominaisuuksiin. Työ toteutetaan mikropalveluna ja integroidaan osaksi valmista mikropalveluarkkitehtuuria, jonka jälkeen se julkaistaan AWS-pilvipalveluympäristössä. Työssä arvioidaan toteutetun työn lisäksi myös toteutusprosessia.

Lopputuloksena on mikropalvelu, joka toteuttaa OAuth 2.0 -standardia, ja on helposti laajennettavissa ja integroitavissa myös toisten palveluiden kanssa.

Avainsanat: OAuth, pääsynhallinta, pääsyn delegointi, Msal

Abstract

Author:	Dani Kuikka
Title:	OAuth 2.0 and Delegated Access as a Microservice
Number of Pages:	34 pages
Date:	3 December 2021
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communications Technology
Professional Major:	Software Engineering
Supervisors:	Vesa Ollikainen, Senior Lecturer
	Juho Salmi, Product Manager

The thesis discusses how OAuth 2.0 standard can be utilized in access management using Microsoft Identity Platform and Msal library. The most important aspects and features of the OAuth 2.0 standard are introduced. In the study, a microservice integrated with an existing microservice architecture was developed and deployed to AWS. The paper reviews the resulting service, as well as the development process.

The result is a microservice which implements the OAuth 2.0 standard and may easily be expanded and integrated with other services as well.

Keywords: OAuth, access management, access delegation, Msal

Sisällys

1 Johdanto

Tämän insinöörityön tavoitteena on perehtyä OAuth 2.0 -standardiin ja tutkia, kuinka sitä voidaan toteuttaa Microsoft Azure -ympäristössä, sekä toteuttaa palvelu, jolle käyttäjä valtuuttaa pääsyn suojatun REST-rajapinnan takana sijaitseviin resursseihinsa. Tämä niin kutsuttu delegoitu pääsy toteutetaan OAuth 2.0 -standardin mukaisesti. Tämän insinöörityön tavoitteena on lisäksi arvioida lopullista toteutusta sekä itse toteutusprosessia.

Projektissa kehitettävä sovellus toteutetaan mikropalveluna ja se liitetään osaksi jo olemassa olevaa mikropalvelusovellusinfrastruktuuria, jonka tarkoitus on muun muassa tuottaa raportteja keräämällä tietoa eri tietolähteistä ja koostaa tästä tiedosta yhteenvetoja ja analyyseja käytettäväksi yrityksen eri tasoille, muun muassa johdon ja hallinnon avuksi. Palvelukokonaisuus pyrkii myös automatisoimaan tiedonkeruuta ja raportointia sekä helpottamaan käyttäjien erilaisten rutiininomaisten työtehtävien suorittamista mahdollistamalla sen kautta suoritettavan erilaisia dataa manipuloivia toimenpiteitä, kuten esimerkiksi työtuntien kirjaamisen työajanhallintajärjestelmään. Microsoftin rajapinnoissa on saatavilla suuri määrä tietoa AD-käyttäjätileistä, jota tähän tietoon valtuutetun palvelun olisi mahdollista hyödyntää. Valtuuttamalla palvelulle OAuth 2.0 -standardin mukaisesti sopivia oikeuksia, olisi palvelun mahdollista myös suorittaa käyttäjän puolesta erilaisia toimenpiteitä Microsoft-ympäristössä, kuten esimerkiksi tehdä kalenterivarauksia tai lähettää sähköpostia.

Projektissa toteutettava mikropalvelu toteutetaan Node.js-projektina ja otetaan käyttöön AWS-pilviympäristössä käyttäen hyödyksi AWS:n tarjoamia palveluita ja teknologioita, kuten Amazon CDK:ta mikropalvelun tarvitseman infrastruktuurin määrittelemiseen. Paikallisessa mikropalvelukehityksessä käytetään Dockeria, joka on ohjelmistokehys konttien (engl. containers) rakentamiseen, ajamiseen ja hallinnoimiseen. Azuren pilvipalvelut ovat myös olennaisessa roolissa mikropalvelun käyttöönotossa, sillä OAuth-standardia tullaan toteuttamaan hyö-

dyntämällä Microsoft Identity Platform -palvelua, jonka välityksellä käyttäjät voivat valtuuttaa palvelullemme pääsyn heidän resursseihinsa. Microsoft Identity Platform on Microsoftin autentikointipalvelu, joka autentikoi myös palvelumme ja myöntää sen käyttöön käyttöoikeustietueita, joita käyttämällä palvelumme pääsee käsiksi käyttäjän suojattuihin resursseihin. Projekti kehitetään Node.js-projektina Microsoftin valtuuttamista ja todentamista varten kehitettyä MSAL-kirjastoa (Microsoft Authentication Library) hyödyntäen.

Työ toteutetaan Gofore Oyj:lle. Gofore on suomalainen, Tampereelta lähtöisin oleva konsulttiyritys, jonka liiketoimintaa ovat digitaalisten palveluiden suunnittelu ja kehittäminen, ja se on ollut mukana toteuttamassa useita suuria julkishallinnon kehitysprojekteja. Goforen liikevaihto vuonna 2020 oli 78 milj. euroa, ja se on vahvasti kasvava yhtiö, joka listautui Helsingin pörssiin vuonna 2017. Gofore on viime vuosina tehnyt lukuisia yritysostoja ja sen tavoitteena on laajentaa toimintaansa edelleen myös ulkomailla. Goforen henkilöstön määrä on viimeisten yrityskauppojen jälkeen yli 800 ja sillä on kuuden kotimaisen toimipisteen lisäksi toimipisteet Tallinnassa, Münchenissä ja Braunschweigissä sekä Madridissa.

2 OAuth 2.0 valtuutusstandardin lähtökohdat

Tässä luvussa tutustutaan tarkemmin siihen, mikä OAuth 2.0 -standardi on ja kuinka sitä olisi palvelussamme mahdollista hyödyntää. Tutustumme standardin sisäisiin yhteyskäytäntöihin ja tavoittelemme ymmärrystä standardin keskeisimmistä tekijöistä voidaksemme hyödyntää tätä tietoa sitä käyttävän toteutuksen kehityksessä sekä tämän toteutusprosessin arvioinnissa.

2.1 Väärinkäsitysten hälventämiseksi

OAuth-standardista on olemassa kaksi versiota: OAuth 1.0 ja OAuth 2.0, jotka eroavat merkittävästi toisistaan. OAuth 1.0 ei ole enää kovin yleisessä käytössä, ja tässäkin työssä käsitellään ainoastaan OAuth 2.0 -standardia, joten jatkossa tässä työssä käytettäessä termiä OAuth viitataan aina nimenomaan

OAuth 2.0 -standardiin. Tämä luku pohjautuu kokonaisuudessaan suurelta osin OAuth 2.0 -standardin viralliseen määrittelyyn lähteenään. [1.]

OAuth-nimen osalta saatetaan helposti sekoittaa kaksi englanninkielistä termiä ”authorization” ja ”authentication”, joista ensimmäinen tarkoittaa suomeksi valtuuttamista ja toinen todentamista. OAuthissa on kyse valtuuttamisesta, eli yksinkertaistetusti siitä, kuinka pääsy määrättyihin käyttäjän suojattuihin resursseihin, esimerkiksi suojatun REST-rajapinnan takana sijaitseviin käyttäjätietoihin voidaan delegoida kolmannen osapuolen ohjelmalle. OAuth itse ei ota kantaa siihen, kuinka käyttäjän todentaminen suoritetaan, vaikka käyttäjä toki joudutaan todentamaan siinä vaiheessa, kun hän valtuuttaa sovellukselle pääsyn resursseihinsa. OAuthissa ei siis ole kyse todentamisesta, mutta sen päälle on kuitenkin kehitetty laajennus nimeltä OpenID Connect, joka mahdollistaa loppukäyttäjän tunnistamisen valtuutettavalle ohjelmalle. OpenID Connectia ei kuitenkaan tässä työssä käsitellä tämän enempää. [2.]

2.2 Valtuutetun pääsyn avoin standardi

OAuth on avoin standardi, jolla on pyritty vastaamaan ongelmaan delegoidusta pääsystä. Toisin sanoen, se määrittelee yhteyskäytännön siitä, kuinka jokin ohjelma voidaan turvallisesti valtuuttaa käyttämään käyttäjän kolmannen osapuolen palvelimella sijaitsevia suojattua tietoja hänen puolestaan. Yksi naiivi tapa toteuttaa tämä olisi, että käyttäjä antaisi ohjelmalle käyttöönsä käyttäjätunnuksensa ja salasanansa, mutta tämä tapa ei olisi kovinkaan turvallista. Tällaisia toteutuksia kuitenkin tehtiin paremman toimintatavan puuttuessa, kunnes pyrkimyksestä ratkaista delegoidun pääsyn ongelma syntyi OAuth. [3.] Nykyään suuri osa isoista teknologiayrityksistä, kuten Amazon, Google, Facebook, Microsoft ja Twitter tukevat OAuth-standardia toteutuksissaan.

2.3 OAuth käytännössä

Palvelun käyttöliittymä, jonka osaksi työssä toteutettavan mikropalvelun kehitämme ja jonka kautta kommunikaatio yksittäisen käyttäjän ja palvelun välillä tapahtuu, toimii Slack-viestintäsovelluksessa yrityksen yksityiseen työtilaan luotuna chattibottina. Käyttäjä voisi siis esimerkiksi pyytää chattibottia kirjaamaan viimeisimmät työntuntinsa työajanhallintajärjestelmään, tai tuottamaan hänelle raportin alaistensa käyttöasteesta keskustelemalla Slack-viestintäsovelluksessa toimivan chattibotin kanssa käyttäen luonnollista kieltä tai valitsemalla haluamansa toiminnot chattibotin tarjoaman valintaikkunan kautta. Mikäli kyseessä on toiminto, jonka täyttämiseen tarvittavat resurssit sijaitsevat kolmannen osapuolen palvelussa ja chattibotti on näihin OAuth-standardin mukaisesti valtuutettu, toteuttaa se pyynnön ja kommunikoi tulokset viestintäsovelluksessa takaisin ilman, että tiedot omistavan käyttäjän tarvitsee itse kirjautua kyseiseen palveluun, tai olla muutenkaan aktiivinen.

Pähkinänkuoressa OAuth on siis standardi, jota sovellukset voivat käyttää tarjotakseen toisille sovelluksille turvallisen delegoidun pääsyn rajattuihin suojattuihin tietoihin. OAuth toimii hyödyntämällä HTTP:tä ja sillä voidaan valtuuttaa laitteita, rajapintoja, palvelimia ja ohjelmistoja käyttämällä käyttäjätunnusten sijaan käyttöoikeustietueita (engl. access tokens). Toisin sanoen OAuth-standardi määrittelee yhteyskäytännön pääsyn delegoinnille, jotta internetin käyttäjät pystyvät turvallisesti sallimaan web-sivustoille ja ohjelmistoille pääsyn heidän tietoihinsa, jotka sijaitsevat toisilla web-sivuilla tai palveluilla ilman, että heidän tarvitsee luovuttaa käyttäjätunnuksensa näiden pääsyn saavien ohjelmien tai sivustojen käyttöön.

2.4 Kritiikki

OAuth-standardi on laajasti käytetty ja pääsyn delegoinnin de facto -standardi, mutta sekään ei kuitenkaan ole säilynyt täysin ilman kritiikkiä. Kritiikki OAuth-standardia kohtaan koostuu lähinnä sen monimutkaisuudesta ja vaikeaselkoisuudesta, sekä sen kantaaottamattomuudesta siihen, kuinka jotkin asiat tulisi

tehdä, kuten käyttäjän tunnistaminen [4], sekä sen mahdollisia tietoturvariskejä kohtaan koskien pääasiassa käyttöoikeustietueiden kaappausta. [5.]

2.5 Roolit

OAuth-standardissa on määritelty neljä keskeistä roolia:

1. resurssin omistaja (engl. resource owner)
2. resurssipalvelin (engl. resource server)
3. asiakasohjelma (engl. client)
4. valtuutuspalvelin (engl. authorization server).

Tarkastellaan näitä rooleja seuraavaksi tarkemmin.

2.5.1 Resurssin omistaja

Resurssin omistaja on toimija, jolla on valta myöntää pääsy suojattuun resurssiin, siis resurssiin, johon pyritään OAuth-standardia toteuttamalla delegoida kolmannen osapuolen ohjelman, eli tässä työssä kehitettävän palvelun pääsy. Resurssin omistaja voi olla henkilö, jolloin häneen viitataan tyypillisesti loppukäyttäjänä, tai sitten se voi olla esimerkiksi yritys. Tämän työn kontekstissa resurssinomistaja on Active Directory -tunnukset omaava henkilö. [1.]

2.5.2 Resurssipalvelin

Resurssipalvelin on palvelin, jolla suojatut tiedot sijaitsevat. Se pystyy vastaanottamaan pyyntöjä sekä vastaamaan pyyntöihin, joiden tarkoituksena on päästä käsiksi suojattuihin tietoihin käyttöoikeustietueita käyttäen. Resurssipalvelin ottaa asiakasohjelmalta vastaan käyttöoikeustietueita ja palauttaa niihin liittyvät resurssit validoituaan käyttöoikeustietueet ensin. [1.]

2.5.3 Asiakasohjelma

Asiakasohjelma on sovellus, joka tekee pyynnön päästä käsiksi suojattuihin resursseihin resurssiomistajan puolesta ja valtuuttamana. Se käyttää pääsyyn sille myönnettyjä käyttöoikeustietueita. Tämän työn kontekstissa asiakasohjelmaa edustaa siis työn tuloksena toteutuva palvelu. [1.]

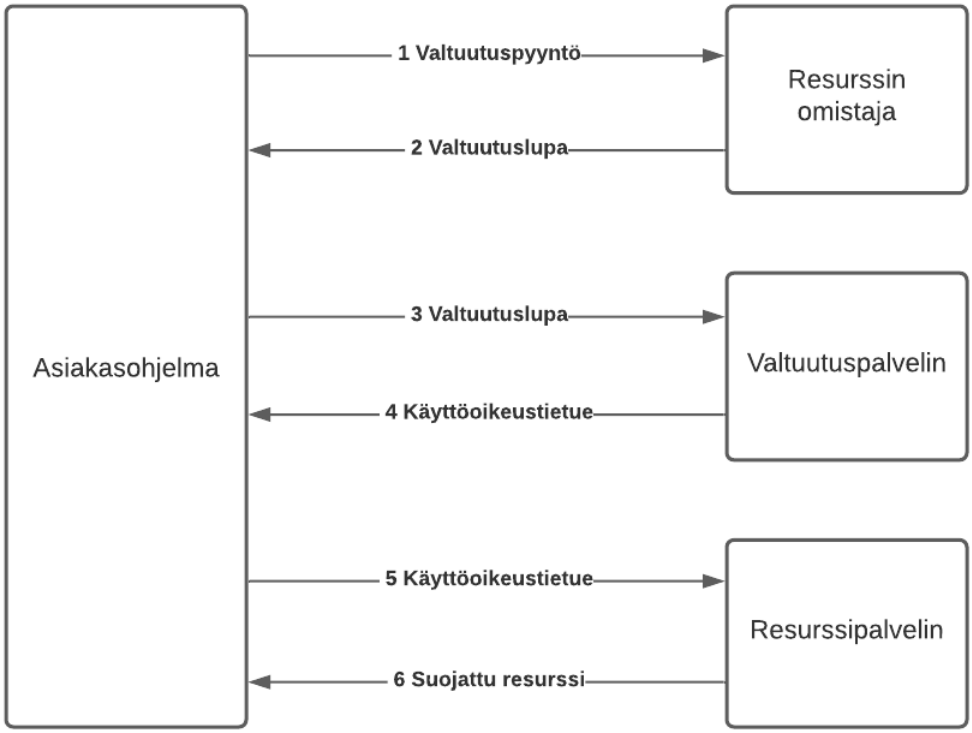
2.5.4 Valtuutuspalvelin

Valtuutuspalvelin on palvelin, joka myöntää asiakasohjelmalle käyttöoikeustietueen sen jälkeen, kun resurssinomistaja on ensin onnistuneesti todennettu ja antanut suostumuksensa. Valtuutuspalvelimella on kaksi päätepistettä (engl. end point): ”/authorize”-päätepiste, jonka kautta käsitellään käyttäjän todennus ja suostumus sekä ”/token”-päätepiste, jonka kautta käyttöoikeustietueet jaetaan. Tämän insinöörityön kontekstissa valtuutuspalvelimen roolia toteuttaa Microsoft Identity Platform. [1.]

OAuth itsessään ei ota kantaa siihen, kuinka valtuutuspalvelin ja resurssipalvelin vuorovaikuttavat keskenään. Valtuutuspalvelin saattaa olla sama palvelin kuin resurssipalvelin, tai siitä erillinen toimija. Yksi valtuutuspalvelin voi myös myöntää käyttöoikeustietueita useille eri resurssipalvelimille. [1.]

2.6 Yhteyskäytännön vaiheet

Tarkastellaan seuraavaksi abstraktilla tasolla, kuinka OAuth-yhteyskäytäntö toimii. Kuva 1 demonstroi, kuinka neljä yllä mainittua roolia vuorovaikuttavat keskenään.



Kuva 1: Roolien vuorovaikutusta havainnollistava kaavio. [1]

Kuvan 1 kaaviossa ovat seuraavat vaiheet:

1. Asiakasohjelma pyytää valtuutusta resurssin omistajalta. Valtuutuspyyntö voidaan tehdä suoraan resurssin omistajalle, mutta suositeltu ja yleisesti käytetty tapa on kuitenkin tehdä se epäsuorasti valtuutuspalvelimen toimiessa välikätenä.

2. Asiakasohjelma vastaanottaa valtuutusluvan (engl. authorization grant), mikä on resurssinomistajan myöntämää valtuutusta edustava pääsytunnus. OAuth-standardissa on määritelty neljä eri lupatyyppiä, joita käsitellään tarkemmin seuraavassa luvussa.

3. Asiakasohjelma pyytää käyttöoikeustietuetta todentamalla itsensä valtuutuspalvelimen kanssa ja esittämällä edellisessä vaiheessa vastaanottamansa valtuutusluvan.

4. Valtuutuspalvelin todentaa asiakasohjelman ja validoi valtuutusluvan. Mikäli valtuutuslupa on validi, valtuutuspalvelin myöntää asiakasohjelmalle käyttöoikeustietueen.

5. Asiakasohjelma tekee pyynnön resurssipalvelimella sijaitsevaan suojattuun resurssiin ja autentikoi pyynnön esittämällä käyttöoikeustietueen.

6. Resurssipalvelin validoi käyttöoikeustietueen ja lähettää pyydetyn resurssin vastauksena käyttöoikeustietueen ollessa validi. [1.]

2.7 Valtuutuslupa

Valtuutuslupa (engl. authorization grant) on pääsytunnus, joka sisältää resurssinomistajan valtuutuksen saada pääsy hänen suojattuihin resursseihinsa. Asiakasohjelma tarvitsee sitä vaihtaakseen sen käyttöoikeustietueeseen. OAuth 2.0 -standardissa määritellään neljä eri lupatyyppiä (engl. grant types):

1. valtuutusavain (engl. authorization code)

2. implisiittinen lupatyyppi (engl. implicit grant)

3. resurssin omistajan pääsytiedot (engl. resource owner password credentials)

4. asiakasohjelman pääsytiedot (engl. client credentials).

Näiden tyyppien lisäksi on olemassa myös laajennusmahdollisuus lisätyypeille. Käsitellään seuraavaksi läpi näitä neljää lupatyyppiä yksityiskohtaisemmin. [1.]

2.7.1 Valtuutusavain

Valtuutusavain saadaan käyttämällä valtuutuspalvelinta asiakasohjelman ja resurssinomistajan välillä. Sen sijaan, että asiakasohjelma pyytäisi valtuutusta suoraan resurssinomistajalta, resurssinomistaja ohjataan valtuutuspalvelimelle

selaimen välityksellä, joka puolestaan ohjaa resurssinomistajan takaisin asiakasohjelmalle valtuutusavain mukanaan. [1.]

Ennen kuin valtuutuspalvelin ohjaa käyttäjän takaisin asiakasohjelmalle valtuutusavaimen kanssa, valtuutuspalvelin autentikoi resurssin omistajan ja pyytää häneltä suostumuksen asiakasohjelman valtuutukseen. Resurssinomistajan käyttäjätunnuksia ei näin ollen koskaan jaeta asiakasohjelmalle, kun resurssinomistaja autentikoi itsensä ainoastaan valtuutuspalvelimen kanssa. Tämä on tietoturvaa lisäävä ominaisuus. [1.]

Valtuutusavain-lupatyyppi tarjoaa muitakin tietoturvaa parantavia ominaisuuksia: se pystyy autentikoimaan myös asiakasohjelman ja toimittamaan käyttöoikeustietueen suoraan asiakasohjelmalle välittämättä sitä resurssinomistajan käyttäjäagentin kautta, mikä altistaisi sen muille tahoille (mukaan lukien resurssin omistajalle itselleen). [1.]

Valtuutusavain-lupatyyppi on tämän insinöörityön kannalta neljästä käsiteltävästä lupatyypistä olennaisin, sillä se on lupatyyppi, jota työssä toteutettavassa palvelussa käytetään.

2.7.2 Implisiittinen lupatyyppi

Implisiittinen lupatyyppi on yksinkertaistettu versio valtuutusavain-lupatyypistä, ja se on optimoitu asiakasohjelmille, jotka toimivat suoraan selaimessa ja käyttävät jotakin skriptauskieltä kuten JavaScriptiä. Implisiittisessä lupatyypissä asiakasohjelmalle myönnetään käyttöoikeustietue suoraan sen sijaan, että sille ensin myönnettäisiin valtuutusavain resurssinomistajan valtuutuksen seurauksena. Lupatyyppi on implisiittinen, koska mitään pääsytietoja, kuten valtuutusavainta ei ensin myönnetä käyttöoikeustietueen hankkimiseksi. [1.]

Kun käyttöoikeustietue myönnetään käyttäen implisiittistä lupatyyppiä, valtuutuspalvelin ei autentikoi asiakasohjelmaa, vaikkakin joissakin tapauksissa asiakasohjelman identiteetti on mahdollista varmistaa uudelleenohjaus URI:n kautta, jota käytetään käyttöoikeustietueen toimittamiseen asiakasohjelmalle.

Käyttöoikeustietue saattaa siis prosessissa altistua resurssin omistajalle tai muille sovelluksille, joilla on pääsy resurssin omistajan käyttäjäagenttiin. [1.]

Implisiittinen lupatyyppi parantaa joidenkin asiakasohjelmien, kuten selaimessa toimivien sovellusten, tehokkuutta ja vasteaikaa, koska se vähentää tarvittavien pyyntöjen määrää käyttöoikeustietueen saamiseksi. Implisiittisen lupatyypin käyttöä tulisi kuitenkin harkita tarkoin sen tietoturvaheikkouksien takia. Valtuutusavain-lupatyypin käyttö on suositeltavaa silloin kun se on mahdollista. [1.]

2.7.3 Resurssin omistajan pääsytiedot

Resurssin omistajan pääsytietoja, toisin sanoen käyttäjätunnusta ja salasanaa, voidaan käyttää suoraan valtuutuslupana käyttöoikeustietueen saamiseksi. Tätä tapaa tulisi käyttää ainoastaan silloin, kun resurssin omistajan ja asiakasohjelman välillä vallitsee korkea luottamus. Esimerkiksi sellaisessa tilanteessa, jossa asiakasohjelma on osa laitteen käyttöjärjestelmää, ja kun muita valtuutuksen lupatyyppejä, kuten valtuutusavain-lupatyyppiä, ei voida käyttää. [1.]

Vaikka tämä lupatyyppi tarvitsee asiakasohjelman osalta suoran pääsyn resurssin omistajan pääsytunnuksiin, käytetään resurssin omistajan tunnuksia vain yksittäiseen pyyntöön, jossa ne vaihdetaan käyttöoikeustietueeseen. Tällä tavoin asiakasohjelman ei ole välttämätöntä tallentaa resurssin omistajan pääsytunnuksia myöhempää käyttöä varten, kun tunnukset vaihdetaan pitkäikäiseen käyttöoikeustietueeseen, tai virkistämisoikeustietueeseen (käyttöoikeustietuetta ja virkistämisoikeustietuetta käsitellään tarkemmin luvuissa 2.7 ja 2.8). [1.]

2.7.4 Asiakasohjelman pääsytiedot

Asiakasohjelman pääsytietoja voidaan käyttää valtuutuksen lupatyyppinä silloin, kun valtuutuksen näkyvyysalue (engl. scope) on rajoitettu sellaisiin suojattuihin resursseihin, jotka ovat asiakasohjelman itsensä kontrolloimia, tai sellaisiin suojattuihin resursseihin, joista on entuudestaan sovittu valtuutuspalvelimen

kanssa. Asiakasohjelman pääsytietoja käytetään valtuutuksen lupatyyppinä tyypillisesti silloin, kun asiakasohjelma toimii omasta puolestaan, eli kun asiakasohjelma on itse myös resurssin omistaja. [1.]

2.8 Käyttöoikeustietue

Käyttöoikeustietueet (engl. access tokens) ovat pääsytietoja, joita käytetään saadakseen pääsy suojattuihin resursseihin. Ne ovat merkkijonoja, jotka edustavat asiakasohjelmalle myönnettyä valtuutusta. Merkkijono on yleensä läpinäkymätön asiakasohjelmalle, jolloin sen sisältö ei ole asiakasohjelmalle näkyvissä. Käyttöoikeustietueet edustavat tiettyjä näkyvyysalueita, jolloin niillä on pääsy vain määriteltyihin resursseihin, jotka resurssin omistaja on hyväksynyt ja resurssi- ja valtuutuspalvelin on vahvistanut. Niillä on myös voimassaoloaika, jonka umpeuduttua ne ovat kelvottomia ja asiakasohjelma joutuu hankkimaan valtuutuspalvelimelta uuden käyttösoikeustietueen käyttämällä virkistysoikeustietuetta. [1.]

Käyttöoikeustietue voi myös ilmentää tunnisteen, jota käytetään saadakseen valtuutustietoja, tai se voi itsessään pitää sisällä nämä valtuutustiedot vahvistettavalla tavalla, eli sisältämällä sekä dataa että allekirjoituksen [1]. OAuthin käyttöoikeustietueena usein käytetty Json Web Token on esimerkki tällaisesta itsenäisestä tunnisteesta, joka pitää sisällään sekä valtuutustiedot että allekirjoituksen. Json Web Tokenia käsitellään tarkemmin luvussa 2.10.

Käyttöoikeustietue tarjoaa abrstrahointikerroksen, jonka avulla voidaan korvata erilaisia valtuuttamistapoja, kuten salasanan ja käyttäjätunnuksen, yhdellä käyttöoikeustietueella, jonka sisällön resurssipalvelin ymmärtää. Tämä abstrahointi mahdollistaa käyttöoikeustietueiden myöntämisen rajoitetummin kuin valtuutusluvan, jota sen hankkimiseksi käytetään. Se myös poistaa resurssipalvelimen tarpeen ymmärtää laajan kirjon erilaisia autentikointimenetelmiä. [1.]

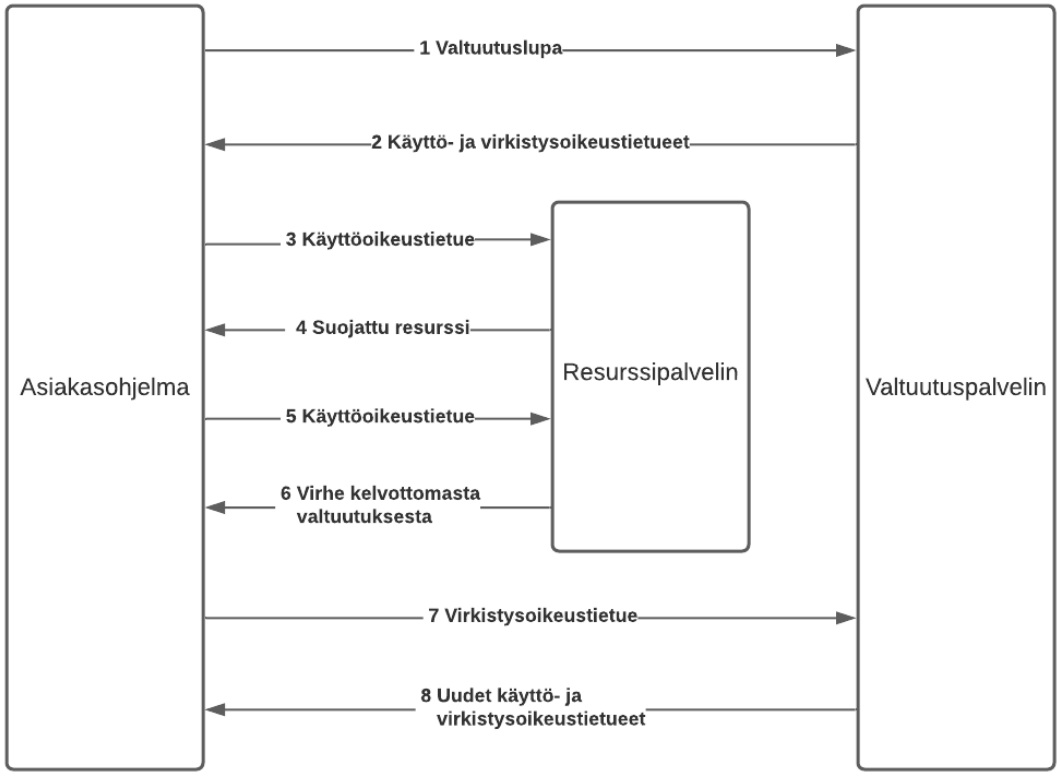
Käyttöoikeustietueilla voi olla erilaisia formaatteja, struktuureja ja käyttötapoja, esimerkiksi kryptografisia ominaisuuksia perustuen resurssipalvelimen tietoturvavaatimuksille. [1.]

2.9 Virkistysoikeustietue

Virkistysoikeustietueet (engl. refresh tokens) ovat pääsytietoja, joita käytetään käyttöoikeustietueiden saamiseen. Valtuutuspalvelin myöntää niitä asiakasohjelmalle ja niitä käytetään uusien käyttöoikeustietueiden hankkimiseen, kun käytössä olevat käyttöoikeustietueet vanhenevat tai muuttuvat mitätöidyiksi. Uusien virkistysoikeustietueiden myöntäminen ei ole valtuutuspalvelimelle pakollinen ominaisuus, mutta mikäli valtuutuspalvelin myöntää niitä, myönnetään ne asiakasohjelmalle samassa vastauksessa käyttöoikeustietueen kanssa. [1.]

Kuten käyttöoikeustietue, myös virkistysoikeustietue on merkkijono, joka edustaa resurssin omistajan asiakasohjelmalle myöntämää valtuutusta. Toisin kuin käyttöoikeustietueet, virkistysoikeustietueet on tarkoitettu käytettäväksi vain valtuutuspalvelimen kanssa, eikä niitä koskaan lähetetä resurssipalvelimelle. [1.]

Kuvassa 2 havainnoidaan vanhentuneen käyttöoikeustietueen uusimista käyttäen virkistysoikeustietuetta.



Kuva 2: Vanhentuneen käyttöoikeustietueen virkistäminen. [1]

Kuvan kaksi kaaviossa ovat seuraavat vaiheet:

1. Asiakasohjelma pyytää käyttöoikeustietuetta autentikoimalla itsensä valtuutuspalvelimella ja esittämällä valtuutuslupansa.

2. Valtuutuspalvelin autentikoi asiakasohjelman ja validoi valtuutusluvan. Mikäli valtuutuslupa on validi, valtuutuspalvelin myöntää käyttö- ja virkistysoikeustietueet.

3. Asiakasohjelma tekee pyynnön suojattuun resurssiin resurssipalvelimella esittämällä käyttöoikeustietueen.

4. Resurssipalvelin validoi käyttöoikeustietueen, ja mikäli valtuutus on validi, toimittaa resurssin.

5. Vaiheet 3 ja 4 toistetaan uudelleen, kunnes käyttöoikeustietue vanhenee. Mikäli asiakasohjelma tietää käyttöoikeustietueen vanhentuneen, siirtyy se suoraan vaiheeseen 7, mutta muuten se tekee vielä uuden pyynnön suojattuun resurssiin.

6. Tässä vaiheessa käyttöoikeustietue on vanhentunut, jolloin se ei enää ole validi ja resurssipalvelin lähettää takaisin virheen kelvottomasta valtuutuksesta.

7. Asiakasohjelma pyytää uutta käyttöoikeustietuetta autentikoimalla valtuutuspalvelimella ja esittämällä tällä kertaa virkistysoikeustietueen. Asiakasohjelman autentikointivaatimukset perustuvat asiakasohjelman tyypille sekä valtuutuspalvelimen käytäntöihin.

8. Valtuutuspalvelin autentikoi asiakasohjelman ja validoi virkistysoikeustietueen. Mikäli virkistysoikeustietue on validi, myöntää valtuutuspalvelin uuden käyttöoikeustietueen ja valinnaisesti myös uuden virkistysoikeustietueen. [1.]

2.10 Json Web Token

Käyttöoikeustietueissa käytetään usein hyväksi Json Web Tokenia. Json Web Token, lyhyemmin JWT, on avoin standardi, joka määrittelee kompaktin ja omavaraisen tavan siirtää tietoa turvallisesti kahden tahon välillä JSON-oliona. Se on omavarainen, koska se sisältää sekä datan että allekirjoituksen ja tiedon siitä, millä algoritmilla allekirjoitus on luotu. Json Web Tokenin sisältämä tieto on luotettavaa ja vahvistettavissa, koska se on digitaalisesti allekirjoitettu käyttäen salaista avainta (HMAC-algoritmia) tai RSA:lla tai ECDSA:lla suojattua julkinen/yksityinen-avainparia. [6.]

Json Web Token pitää sisällään kolme pisteellä eroteltua osaa:

1. header-osa, joka tyypillisesti pitää sisällään tiedon siitä, että kyseessä on JWT sekä allekirjoitukseen käytetyn algoritmin tyypin.

2. payload-osa, joka pitää sisällään viestin väitteet, eli OAuthin tapauksessa käyttöoikeustietueen väittämät.

3. signature- eli allekirjoitusosa.

Näistä kaksi ensimmäistä on Base64url-koodattu ja niitä käytetään algoritmin parametreina allekirjoitusta luotaessa, mikä takaa sen, ettei niitä voida peukaloida kommunikaation aikana. Kuvassa 3 nähdään valmis base64-koodattu JWT. [6.]

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

Kuva 3: Esimerkki JWT:stä. [6]

Kuten kuvassa 3 on värejä käyttäen havainnollistettu, JWT koostuu kolmesta erillisestä Base64-koodatusta ja pistein erotellusta osasta.

Json Web Token ei ole osa OAuth-standardia, mutta useiden valtuutuspalvelimen myöntämät käyttöoikeustietueet ovat Json Web Tokeneita. Microsoft Identity Platform [7], joka tämän insinöörityön kontekstissa toteuttaa valtuutuspalvelimen roolia, myöntää käyttöoikeustietueina Json Web Tokeneita.

3 Palvelun toteutus

Nyt kun olemme luvussa 2 saaneet yleiskäsityksen OAuth-standardista ja ymmärtäneet sen perusteet, siirrymme tässä luvussa tarkastelemaan, kuinka OAuth-standardia voidaan toteuttaa käytännössä Microsoft Azure -ympäristössä. Tässä luvussa tarkastellaan ensin lyhyesti mikropalveluita yleisellä tasolla ja kuvataan sitten asiakasohjelman rekisteröimistä valtuutuspalvelimelle

Microsoft Azure -pilvipalvelussa sekä tämän prosessin olennaisimpia osia. Tämän jälkeen tehdään katsaus projektissa käytettyihin teknologioihin ja käydään toteutusta läpi kooditasolla. Lopuksi arvioidaan koko prosessia kehittäjän näkökulmasta.

Tässä projektissa OAuth-toteutus tehdään käyttäen Microsoftin palveluympäristöä, mutta on mainitsemisen arvoista, että suuri määrä niin suuria kuin pieniäkin teknologiayrityksiä tukee OAuth-yhteyskäytännön toteuttamista. Näihin yrityksiin lukeutuvat muun muassa: Amazon, Apple, Facebook, Dropbox, Discord, Google, Flickr, GitHub, LinkedIn, Netflix ja Twitter.

3.1 Mikropalvelut

Mikropalveluarkkitehtuuri on arkkitehtuurimalli, jossa sovellusohjelma rakennetaan toisistaan erillisistä pienemmistä palveluista, mikropalveluista, jotka kommunikoivat keskenään (tyypillisesti REST-rajapintojen välityksellä) ja joista kukin palvelee jotakin tiettyä tarkoitusta tai toiminnallisuutta. Mikropalveluarkkitehtuurin käyttäminen vähentää ohjelman sisäisiä riippuvuuksia ja tekee sovelluskehittämisestä ketterämpää verraten perinteiseen monoliittiseen arkkitehtuurimalliin, jossa kaikki ohjelman toiminallisuus on yhden palvelun sisällä.

Mikropalveluarkkitehtuuri mahdollistaa yksittäisten palveluiden skaalaamisen, kehittämisen ja julkaisun irrallaan muista mikropalveluista sekä eri teknologioiden ja työkalujen käyttämisen eri mikropalveluissa. Tämä tekee palveluiden kehittämisestä tehokkaampaa ja parantaa myös kehittäjäkokemusta.

3.2 Azure

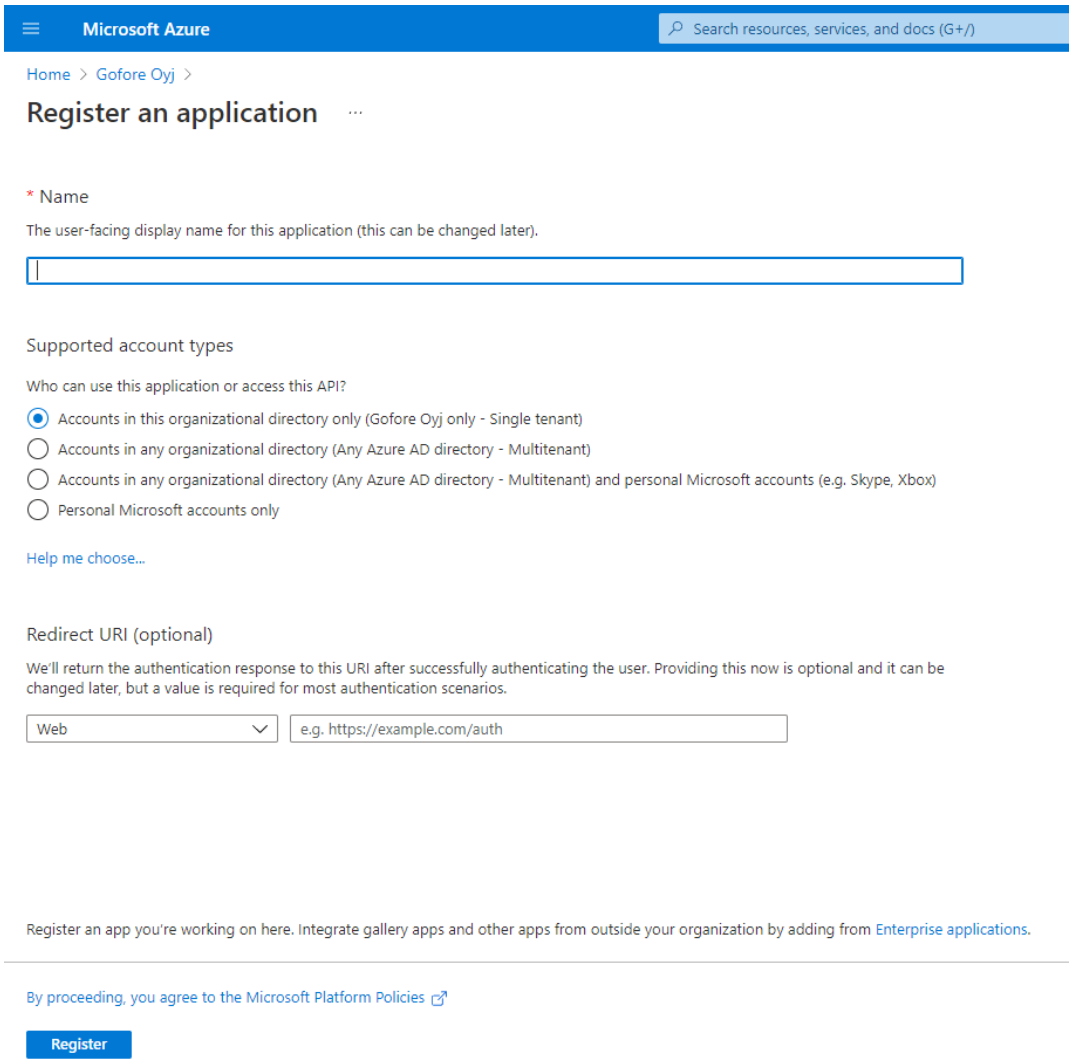
Ennen kuin voimme käyttää OAuth-yhteyskäytäntöä pääsyn delegoimiseen sovelluksessamme, tulee meidän rekisteröidä asiakasohjelmamme valtuutuspalvelimelle. Tämä on olennainen osa yhteyskäytännön toteuttamista. Koska haluamme käyttää Microsoftin AD -tunnuksia palvelumme valtuuttamiseen, rekisteröimme asiakasohjelmamme Microsoftin valtuutuspalvelimelle, eli Microsoft

Identity Platformille. Kun olemme rekisteröineet asiakasohjelmamme, voimme alkaa kehittää mikropalveluamme.

Rekisteröidäksemme asiakasohjelmamme Microsoftin Azure -pilvipalvelun [8] kautta tarvitsemme pääsyn organisaatiomme tiliin Azure Active Directory:ssä [9], eli niin kutsuttuun Azure Active -vuokralaiseen (engl. tenant). Tilillä voidaan rekisteröidä ja hallinnoida sovelluksia sekä määritellä niiden pääsyä dataan Microsoft 365 -palvelussa ja muissa rajapinnoissa. [10.]

3.3 Asiakasohjelman rekisteröinti

Rekisteröimme asiakasohjelmamme Azure Portalissa [11], jotta voimme käyttää Microsoft Identity Platformin tarjoamia autentikointi- ja valtuutuspalveluita, toteuttaaksemme OAuth-yhteyskäytäntöä palvelussamme. Kuvassa 2 näemme kuvankaappauksen uuden asiakasohjelman rekisteröintilomakkeesta Azure Portalissa.



Kuva 3: Uuden asiakasohjelman rekisteröintilomake Azuressa.

Kun olemme rekisteröineet asiakasohjelman, Azure luo sille asiakastunnuksen (engl. client ID), jota käytetään sen tunnistamiseen Microsoft Identity Platformissa. Tätä asiakastunnusta tarvitsemme myös koodissamme, kun ohjelmoimme mikropalveluamme. Tunnusta käytetään osana ohjelmamme valtuutuspalvelimelta vastaanottamien käyttöoikeustietueiden validoimista. [12.]

3.3.1 Uudelleenohjausosoite

Uudelleenohjausosoite (engl. redirect URI) on URI-osoite, eli sijainti, jonne valtuutuspalvelin, eli tämän projektin tapauksessa Micosoft Identity Platform, ohjaa

käyttäjän selaimen ja lähettää käyttöoikeustietueen autentikoinnin ja asiakasohjelman valtuuttamisen jälkeen. Lisäämme Azureen myös asiakasohjelman rekisteröinnin yhteydessä, tai sen jälkeen, tarvitsemamme uudelleenohjausosoitteet. Meidän tapauksessamme nämä ovat localhost paikallista kehitystä varten sekä palvelumme julkaisuosoite AWS:ssä. Kuten asiakasohjelman rekisteröintikin, myös uudelleenohjausosoite on keskeinen osa OAuth-standardin toteuttamista. [12.]

3.3.2 Asiakasohjelman salainen avain

Azuressa on mahdollista toteuttaa asiakasohjelman tunnistaminen käyttämällä joko sertifikaattia tai asiakasohjelman salaista avainta. [12.]

Asiakasohjelman salainen avain on merkkijono-tyyppinen arvo, jota sovelluksemme käyttää autentikoidakseen itsensä valtuutuspalvelimen kanssa. Asiakasohjelman salaisuudet eivät ole aivan yhtä turvallisia, kuin mitä julkisen avaimen salaustekniikkaa käyttävät sertifikaatit ovat, mutta ne ovat helppokäyttöisempiä ja kehittäjäystävällisempiä, sekä tämän projektin tavoitteiden kannalta riittävän turvallisia. On kuitenkin hyvä tiedostaa, että siinä vaiheessa, kun palvelua ollaan viemässä laajalti julkisesti saatavaksi tuotantoympäristöön, tulisi käytettävien pääsytietojen vaihtamista sertifikaattiin harkita. [12.]

Luodaan Azure Portalissa uusi asiakasohjelman salainen avain tekemällemme asiakasohjelman rekisteröinnille. Salainen avain on oletusarvoisesti voimassa 6 kk, jonka jälkeen tulee luoda uusi korvaava salainen avain. Kun olemme luoneet salaisen avaimen, otetaan sen arvo heti talteen käyttöä varten ohjelmamme koodissa, sillä tämän jälkeen salaisuuden arvoa ei enää koskaan näytetä Azuressa uudelleen. [12.]

3.4 Toteutuksessa käytetyt teknologiat

Ennen kuin tarkastelemme ohjelmamme koodia ja sen toimintaa tarkemmin, tehdään lyhyt yleiskatsaus mikropalvelussamme käytettäviin teknologioihin sekä siihen, mikä niiden tarve palvelussamme on.

3.4.1 Node.js ja Express.js

Node.js on suosittu avoimen lähdekoodin järjestelmäriippumaton JavaScript-ajoympäristö. Sitä käyttäen voidaan ohjelmoida JavaScriptillä palvelinpuolella. [13.]

Express.js [13] on Node.js ohjelmistokehys, ja se on kaikista Node.js-moduuleista suosituin. Se on minimaalinen kehys, joka tarjoaa toiminnallisuuksia web- ja mobiilisovellusten kehittämiseen. Se nopeuttaa ja helpottaa Node.js-pohjaisten web-ohjelmistojen kehittämistä. Muutamia Expressin tärkeimmistä ominaisuuksista ovat:

- Se mahdollistaa HTTP-pyyntöihin vastaavien väliohjelmistojen asentamisen.
- Määrittelee reititystaulun, jota käytetään suorittamaan eri toimintoja HTTP-metodin ja URL:n perusteella.
- Mahdollistaa HTML-sivujen renderöinnin dynaamisesti mallille annettujen argumenttien perusteella.
- Express.js perustuu Node.js:n Connect-väliohjelmistomoduulille, joka puolestaan käyttää Node.js:n natiivia HTTP-moduulia. [13.]

Käyttäessämme projektissamme Node.js- ja Express.js-moduuleita saamme palvelumme nopeasti toimintakuntoiseksi, jolloin pääsemme nopeasti kehittämään OAuth-standardia toteuttavaa mikropalveluamme.

3.4.2 Axios

Axios [14] on JavaScriptin lupauksiin (engl. JavaScript promise) perustuva HTTP-asiakasohjelma, joka toimii Node.js:llä ja selaimella. Palvelinpuolella se, kuten Express.js-moduulikin, käyttää Node.js:n natiivia HTTP-moduulia. Selainpuolella se toimii käyttäen XMLHttpRequest-pyyntöä. Axios helpottaa asynkronisten HTTP-pyyntöjen lähettämistä REST-rajapintojen päätepisteisiin ja CRUD-operaatioiden suorittamista. REST-rajapintojen päätepisteet ovat URL-osoitteita, joiden kautta resursseja pyydetään ja jaetaan. Muita Axioksen toiminnallisuuksia ovat muun muassa seuraavat: [15.]

- pyynnön ja vastauksen pysäyttäminen
- pyyntö- ja vastausdatan muuntaminen
- pyyntöjen peruuttaminen
- automaattinen muunnos JSON-datalle.

Projektissamme käytämme Axiosta HTTP-pyyntöjen lähettämiseen palvelimelle.

3.4.3 Dotenv

Ympäristömuuttujat ovat muuttujia, jotka määritellään ohjelman ulkopuolella, usein pilvipalveluntarjoajan tai käyttöjärjestelmän puolesta. Nodessa ympäristömuuttujat ovat käytössä env-olion kautta, joka puolestaan on globaalin process-objektin ominaisuus.

Dotenv [16] on moduuli ympäristömuuttujien hallintaan. Se lataa ympäristömuuttujat .env-tiedostosta Node.js:n process.env-objektiin. Ympäristökonfiguraation pitäminen koodista erillään on parhaiden ohjelmistokäytäntöjen mukaista, ja tässä Dotenv on avuksi. Emme halua esimerkiksi viedä asiakasohjel-

mamme salaisuutta versionhallintaan. Lisäksi kehitys- ja tuotantoympäristössämme käytetään eriarvoisia ympäristömuuttujia, ja Dotenv helpottaa näiden hallintaa.

3.4.4 Msal-node

Microsoft Authentication Library (MSAL) on Microsoftin tarjoama ohjelmointikirjasto. Siitä löytyy versio usealle eri ohjelmointikielelle ja alustalle. Tässä projektissa käytämme Msal-node-moduulia, joka on tarkoitettu Node.js:lle. Msal-node-moduuli mahdollistaa sovellusten autentikoida käyttäjiä käyttäen Azure AD työ- ja koulutilejä sekä Microsoftin henkilökohtaisia tilejä ja sosiaalisen median tilejä kuten Facebookia, Googlea ja LinkedInia. Sen avulla voimme toteuttaa OAuth-yhteyskäytäntöä ja hankkia käyttöoikeustietueita sekä autentikoida käyttäjiä ja saada pääsyn suojattuihin rajapintoihin. [17.]

MSAL tukee kaikkia OAuth-standardissa määriteltyjä valtuutuslupatyyppejä, eli valtuutusavain, implisiittinen, resurssinomistajan pääsytiedot sekä asiakasohjelman pääsytiedot -lupatyyppejä, joiden lisäksi se tukee myös joitakin näihin tehtyjä laajennuksia. [18.]

Msal-moduulin hyötyjä on, että se tekee OAuth yhteyskäytännön toteuttamisesta koodissa helpompaa huolehtimalla yhteyskäytäntötason yksityiskohdista, poistaen kehittäjän tarpeen ohjelmoida itse suoraan yhteyskäytäntöä vasten. Sen avulla voidaan hankkia käyttöoikeustietueita käyttäjän tai sovelluksen puolesta ja se säilyttää näitä käyttöoikeustietueita välimuistissa sekä virkistää niitä itsenäisesti, kun ne ovat vanhentumassa, jolloin kehittäjän ei itse tarvitse huolehtia myöskään käyttöoikeustietueiden vanhentumisesta. Tämän lisäksi Msal-moduuli auttaa myös rajaamaan tilejä, joiden kautta käyttäjiä voidaan autentikoida. Esimerkiksi niin, että autentikoidaan ainoastaan yritystilejä tietystä organisaatiosta, eikä muita tilejä. Kehitysprosessin avuksi Msal auttaa myös ongelmanratkaisussa tarjoten käyttökelpoisen poikkeusten käsittelyn ja kattavat loki-

tiedot sekä telemetriaa, mikä auttaa virheiden ja ongelmatapausten selvittämisessä. Tämän lisäksi se on myös melko helppokäyttöinen ja sen käyttöönotto on nopeaa. [17.]

3.5 Ohjelmiston koodi

Ennen kuin tarkastelemme ohjelman koodirivejä yksityiskohtaisemmin, tehdään pikainen kertaus projektimme tavoitteisiin ja vaatimuksiin. Haluamme kehittää ohjelman, joka toteuttaa OAuth-yhteyskäytäntöä. Meidän tulee siis pystyä autentikoimaan käyttäjiä sekä saamaan käyttöömme käyttöoikeustietueita, joiden avulla teemme pyyntöjä suojatun rajapinnan takan sijaitseviin resursseihin OAuth-yhteyskäytännön mukaisesti. OAuth-yhteyskäytännön lupatyyppi, jota tulemme käyttämään on valtuutusavain-lupatyyppi.

3.5.1 Alkutoimet

Projektin toteuttamisen käynnistämiseksi alustettiin ensin uusi Node.js-projekti käyttämällä NPM:ää (Node Package Manager) [19] ja luotiin projektille index.js-tiedosto ohjelmiston käynnistämistä ja koodia varten. Tässä vaiheessa alustettiin myös Git-projekti versionhallintaa varten sekä luotiin .env-tiedosto paikallisia ympäristömuuttujia varten ja lisättiin se .gitignore-tiedostoon pitääksemme sen poissa versionhallinnastamme.

Seuraavaksi aloitettiin index.js -tiedostomme kehittäminen tuomalla ensin projektiin mukaan siinä tarvittavat moduulit, joita ovat aikaisemmin käsittelemämme Express.js, Msal-Node, Axios ja Dotenv. Tämän jälkeen voidaan kutsua dotenvin config()-metodia, joka lataa ympäristömuuttujat ohjelmamme käyttöön joko luomastamme .env-tiedostosta lokaalissa kehitysympäristössämme tai sitten AWS:ään lisäämistämme ympäristömuuttujista ollessamme tuotantoympäristössä.

Määritellään vakio palvelimemme portille ja haetaan ympäristömuuttujista sen arvoksi AWS:ään määrittelemämme portti, ollessamme tuotantoympäristössä.

Käytetään porttia 3000 ollessamme paikallisessa kehitysympäristössä. Luodaan sitten Express-objekti ja pyydetään sitä kuuntelemaan saapuvia pyyntöjä määrittelemäämme porttiin. Tässä vaiheessa olemmekin jo suureksi osaksi ohjelmoineet tarvittavan palvelinosuuden määrittelyn ja voimme siirtyä päätepisteiden ja OAuth-yhteyskäytännön toteuttamisen pariin.

3.5.2 Config-olio

Määritellään Msal-node-moduulin käyttöön config-olio, jolla on kaksi ominaisuutta: auth ja system. Auth on objekti, joka pitää sisällään asiakasohjelmamme autentikointia varten tarvitsemamme ympäristömuuttujat. Näitä ovat asiakasohjelman tunnus ja salainen avain sekä Azure-vuokralainen, eli organisaatiomme tili.

System-ominaisuus pitää sisällään konfiguraatio-objektin Msal-moduulin lokitiedostoa pitävälle oliolle (logger). Määrittelemällä sen logLevel-ominaisuuden tasolle runsas (verbose), kerromme sille, että haluamme saada yksityiskohtaisia lokitietoja. Tämä auttaa meitä kehitysprosessissa sekä mahdollisten ongelmatilanteiden ratkaisussa. Koodiesimerkissä 1 nähdään, kuinka tällainen määrittely voidaan tehdä. [20.]

```
const config = {
    auth: {
        clientId: process.env.CLIENT_ID,
        authority: process.env.AUTHORITY,
        clientSecret: process.env.CLIENT_SECRET
    },
    system: {
        loggerOptions: {
            loggerCallback(loglevel, message, containsPii) {
                console.log(message);
            },
            piiLoggingEnabled: false,
            logLevel: msal.LogLevel.Verbose,
        }
    }
};
```

Koodiesimerkki 1: Config-olion määrittely. [21]

3.5.3 Asiakasohjelman tyyppi

Seuraavaksi luodaan uusi yksityinen ConfidentialClientApplication-olio, joka on siis Msal-olio, ja annetaan sen konstruktorikutsun argumentiksi määrittelemämme config-objekti. Msal tukee kahdenlaista asiakasohjelman tyyppiä: luottamuksellista ja julkista (engl. confidential ja public). Näiden kahden ero on niiden kyvyssä autentikoida turvallisesti valtuutuspalvelimen kanssa sekä säilyttää asiakasohjelman pääsytietojen yksityisyys. Yksityisiä asiakasohjelmia ovat sovellukset, jotka ajetaan palvelimella. Tällaisia ovat esimerkiksi web-sovellukset ja web-rajapinnat, tai ohjelmat, jotka ovat käyttöjärjestelmän taustaprosesseja. Ne voidaan mieltää vaikeasti saavutettaviksi ja sen takia kykeneviksi turvallisesti säilyttämään asiakasohjelman salainen avain. [22.]

Julkisia asiakasohjelmia taas ovat sovellukset, jotka ajetaan laitteilla, pöytäkoneilla tai selaimessa. Ne eivät pysty pitämään asiakasohjelman salaisuutta turvassa, ja niille käytössä olevien lupatyyppien määrä on rajattu. [22.]

3.5.4 Autentikointi-päätepiste

Ohjelmoidaan seuraavaksi ensimmäinen päätepisteemme. Määritellään ohjelmamme kuuntelemaan ”/”-päätepistettä. Tämä päätepiste toimii autentikointipäätepisteenämme, jonne ohjaamme uudet käyttäjät, jotka eivät vielä ole autentikoineet ja antaneet suostumustaan asiakasohjelmallemme, navigoimaan selaimella. Määritellään sitten parametrit valtuutusavain-lupatyypin käyttöä varten. Näitä ovat asiakasohjelman käyttäjältä tarvitsema näkyvyysalue sekä uudelleenohjausosoite pyynnön uudelleenohjausta varten, kun käyttäjä on autentikoinut ja antanut suostumuksensa. Näkyvyysalue user.read antaa luvan lukea käyttäjän Microsoft Graph API:ssa sijaitsevia käyttäjätietoja. [23.]

Kun nämä määritykset on tehty, pyydetään Msal-oliolta url valtuutusavainslupatyypin käyttöä varten ja ohjataan käyttäjä sinne. Url ohjaa käyttäjän selaimen Microsoftin valtuutuspalvelimelle, missä käyttäjän selaimessa aukeaa kehote, jossa pyydetään käyttäjää ensin kirjautumaan sisään AD-tunnuksillaan ja sitten

antamaan suostumuksensa sille, että asiakasohjelmallemme saa pääsyn hänen tietoihinsa määrittelemämme näkyvyysalueen mukaisesti. Koodiesimerkissä 2 nähdään, kuinka tällainen toteutus voidaan tehdä.

```
// Msal-olion luonti
const cca = new msal.ConfidentialClientApplication(config);

// Autentikaatio-päätepiste
app.get('/', (req, res) => {
    const authCodeUrlParameters = {
        scopes: ["user.read", "offline_access"],
        redirectUri: "http://localhost:3000/redirect",
    };
    cca.getAuthCodeUrl(authCodeUrlParameters).then((response) => {
        res.redirect(response);
    }).catch((error) => console.log(JSON.stringify(error)));
});
```

Koodiesimerkki 2: Msal-olion luonti ja ”/”-päätepisteen ohjelmointi. [21]

3.5.5 Uudelleenohjaus-päätepiste

Ohjelmoidaan seuraavaksi ”/redirect”-päätepiste käyttöoikeustietueen vastaanottamista varten. Tähän päätepisteeseen ohjataan myös valtuutusavaimen sisältävä pyyntö edellä tarkastellusta ”/” -päätepisteestä. Luodaan tokenRequest-vakio ja tuodaan sinne valtuutusavain vaihdettavaksi käyttöoikeustietueeseen. Lisätään pyyntöön myös näkyvyysalue, jolle haluamme käyttöoikeustietueen.

Kutsutaan Msal-moduulin acquireTokenByCode-metodia, joka tekee pyynnön resurssipalvelimen /token -päätepisteeseen toimittaen sinne argumenttina antamamme valtuutustunnuksen. Tulostetaan pyynnön onnistuessa vastaus konsoliimme ja kuitataan pyyntö onnistuneeksi lähettämällä vastauksena 200-tilakoodin. Msal tallentaa käyttöoikeustietueen meille myös välimuistiin. Mikäli pyyntö epäonnistuu, tulostetaan virhe ja lähetetään 500-tilakoodi vastauksena pyyntöön. Koodiesimerkissä 3 nähdään, kuinka tällainen toteutus voidaan tehdä.

```
// Uudelleenohjaus-päätepiste
app.get('/redirect', (req, res) => {
    const tokenRequest = {
        code: req.query.code,
        scopes: ["user.read"]
};
    cca.acquireTokenByCode(tokenRequest).then((response) => {
        console.log("\nResponse: \n:", response);
        res.sendStatus(200);
    }).catch((error) => {
        console.log(error);
        res.status(500).send(error);
    });
});
```

Koodiesimerkki 3: “/redirect”-päätepisteen ohjelmointi. [21]

3.5.6 Pyyntö suojattuun resurssiin

Määritellään vielä kolmas päätepiste, jonne tästä mikropalvelusta erillinen Slack-viestintäsovellusta kuunteleva mikropalvelumme voi lähettää pyynnön saadakseen käyttäjän Microsoft Graph API:ssa sijaitsevat käyttäjätiedot. Slackiä kuunteleva mikropalvelumme tuntee käyttäjästämme jo joitakin tietoja, kuten sähköpostiosoitteen. Tämä sähköpostiosoitteen avulla voimme tarkastaa OAuth-standardia toteuttavan mikropalvelumme välimuistista, onko käyttäjä antanut suostumuksensa asiakasohjelmallemme, sekä hakemaan käyttöömme käyttäjän valtuutusta edustavan käyttöoikeustietueen.

Saamme Msal-moduulin välimuistiin tallentamat tiedot käyttöömme kutsumalla msal-olion getTokenCache()-metodia ja välimuistiin tallennetut käyttäjätilit kutsumalla tämän tokenCache-olion getAllAccounts()-metodia. Käyttäjätileistä saamme haettua oikean tilin käyttämällä pyynnön mukana tullutta sähköpostiosoitetta.

Kun meillä on oikea käyttäjätili käytössämme, kutsutaan Msal-moduulin acquireTokenSilent()-metodia ja annetaan sille argumenttina käyttäjätili sekä näkyvyysalue. AcquireTokenSilent()-metodi hakee sille argumenttina annetun käyttäjän

käyttöoikeustietueen välimuistista. Mikäli se on vanhentunut, koettaa metodi hankkia uuden käyttöoikeustietueen käyttämällä virkistämisoikeustietuetta.

Tehdään lopuksi Axios-moduulia käyttäen HTTP-pyyntö Microsoft Graph API:iin (OAuth-terminologiaa käytettäessä puhuisimme resurssipalvelimesta) määrittelemällä pyyntöön ensin Authorization-headerin, jonka arvona on käyttöoikeustietue. Koodiesimerkissä 4 nähdään, kuinka tällainen toteutus voidaan tehdä.

```
// Päätepiste tietojen hakemiselle käyttäen käyttöoikeustietuetta
app.get('/userinfo', async (req, res) => {
    if (!req.query.email) {
        res.status(400).send("Email is a required parameter")
        return
    }
    const msalTokenCache = cca.getTokenCache();
    accounts = await msalTokenCache.getAllAccounts();
    account = accounts.find(element => element.username ==
req.query.email)
    const silentRequest = {
        account: account,
        scopes: ["user.read"],
    };
    cca.acquireTokenSilent(silentRequest).then((response) => {
        axios.default.get('https://graph.microsoft.com/v1.0/me', {
            headers: {
                Authorization: `Bearer ${response.accessToken}`
            }
        })
            .then(function (response) {
                res.status(200).send(response.data);
                console.log(response.data);
            })
            .catch(function (error) {
                console.log(error);
                res.status(403).send(error);
            })
    }).catch((error) => {
        console.log('cannot acquire token')
        console.log(error);
        res.status(403).send(error);
    })
})
```

Koodiesimerkki 4: Suojattujen resurssien hakemien resurssipalvelimelta.

Mikäli pyyntö onnistuu, saamme vastauksena käyttäjämme käyttäjätiedot, jotka lähetämme 200-tilakoodin kera vastauksena eteenpäin Slackiä kuuntelevalle mikropalvelullemme. Mikäli pyyntö ei onnistu, saamme virheen, jonka lähetämme 403-tilakoodin kera vastauksena Slackiä kuuntelevalle mikropalvelullemme, joka puolestaan ilmoittaa virheestä loppukäyttäjälle ja ohjeistaa käyttäjää siirtymään autentikointipäätepisteeseemme ja autentikoitumaan sekä antamaan suostumuksen asiakasohjelmallemme, mikäli hän ei vielä ole niin tehnyt.

3.6 Integraatio

Jotta integraatio ja paikallinen kehitys onnistuisi jo olemassa olevan mikropalveluarkkitehtuurin kanssa, luotiin projektille Docker-tiedosto (engl. Dockerfile) ja lisättiin mikropalvelun Docker-kontti samaan verkkoon mikropalvelun kanssa, josta sitä kutsutaan. Integraatio Slackiä kuuntelevan mikropalvelun kanssa toteutettiin HTTP-pyyntöjen välityksellä. Slackin kanssa keskustelevalle mikropalvelulle luotiin myös oma moduuli noudettavan datan käsittelyä ja sen käyttäjälle esittämistä varten.

3.7 Julkaisu

Amazon CDK [24] on avoimen lähdekoodin ohjelmistokehys, jolla määritellään pilvi-infrastruktuuria koodina. AWS CDK mahdollistaa siis palvelun tarvitseman pilvi-infrastruktuurin määrittelyn yhdessä tiedostossa käyttäen kehittäjälle tuttua ohjelmointikieltä sen sijaan, että infrastruktuuri luotaisiin pala palalta AWS dashboard -käyttöliittymän kautta. Infrastruktuuri koodina -toimintatapa (engl. Infrastructure as Code) mahdollistaa lisäksi infrastruktuurin nopean ja helpon skaalaamisen, muokkaamisen sekä uudelleenkäytettävyyden.

Tämän projektin julkaisua varten luotiin uusi CDK-tiedosto, jossa määriteltiin projektia varten tarvittava infrastruktuuri, ja se julkaistiin käyttämällä AWS CDK Toolkitiä.

3.8 Ratkaisun arviointi

Olemme toteuttaneet mikropalvelun, joka puolestaan toteuttaa OAuth-yhteyskäytäntöä käyttäen siinä määriteltyä valtuutusavain-lupatyyppiä. Olemme siis toteuttaneet palvelun, joka ohjaa käyttäjän autentikoitavaksi Microsoftin valtuutuspalvelimelle ja pyytää käyttäjältä suostumuksen saadakseen pääsyn hänen suojattuihin resursseihinsa. Palvelu vastaanottaa sitten valtuutuspalvelimelta valtuutusavaimen ja käyttää tätä avainta käyttö- ja virkistämisoikeustietueiden saamiseksi. Lopulta palvelu käyttää näitä käyttöoikeustietueita saadakseen pääsyn käyttäjän suojattuihin resursseihin, jotka lähetetään edelleen mikropalveluun, johon integroimme OAuthia-yhteyskäytäntöä toteuttavan mikropalvelumme ja josta pyyntö yhteyskäytännön toteuttamisen aktivoimiseksi saapuu.

Lisäksi käytettävien teknologioiden osalta tekemiemme valintojen puolesta palvelumme säilyttää käyttöoikeustietueita ja käyttäjätietoja muistissaan. Palvelu myös päivittää automaattisesti vanhentuneita käyttöoikeustietueita on helposti laajennettavissa tekemään uusia pyyntöjä esimerkiksi eri näkyvyysalueilla tai eri rajapintoihin.

Voimme todeta, että olemme täyttäneet insinöörityötä varten asettamamme tavoitteet siltä osin, että olemme perehtyneet OAuth-standardin olennaisimpiin osa-alueisiin ja hyödyntäneet tätä tietoa OAuth-yhteyskäytäntöä onnistuneesti toteuttavan mikropalvelun luomisessa. Voimme työn pohjalta myös todeta, että OAuthia voidaan hyödyntää onnistuneesti yhdessä Microsoftin valtuutuspalvelimen kanssa, eikä mikropalvelun liittämisessä osaksi jo olemassa olevaa mikropalveluarkkitehtuuria esiintynyt esteitä.

Tämä mikropalvelu hoitaa sille asetetun tehtävän erinomaisesti. Se hankkii käyttäjän suostumuksella pääsyn käyttäjän suojattuihin resursseihin OAuth-yhteyskäytännön mukaisesti. Tätä yksittäistä mikropalvelua laajempana jatkokehityskohteena olisikin näiden resurssien merkittävä hyödyntäminen sovelluskokonaisuuden muissa mikropalveluissa. Mainittavan arvoista on myös, että koska palvelu on kehitetty mikropalveluna, se on helposti liitettävissä ja muokattavissa

myös muihin sovelluksiin, jotka tarvitsevat, tai voivat hyötyä OAuth-yhteyskäytäntöä toteuttavasta mikropalvelusta.

4 Yhteenveto

Tutustuttuamme ensin OAuth-standardin keskeisimpiin osa-alueisiin ja yksityiskohtiin sekä katsastettuamme mikropalveluita yleisellä tasolla esittelimme projektissa käytettävät teknologiat ja tarkastelimme toteutettua palvelua yksityiskohtaisesti koodiesimerkkien avulla. Tämän jälkeen integroimme mikropalvelumme jo olemassa olevaan mikropalveluarkkitehtuuriimme ja julkaisimme sen AWS:ssä, jonka jälkeen analysoimme toteutusta ja totesimme sen täyttäneen sille asetetut tavoitteet.

Projektin toteuttamisen kehitysprosessi oli suurimmaksi osaksi sujuvaa, eikä sen aikana esiintynyt esteitä, joiden ylittäminen ei olisi ollut mahdollista. Osa käytetyistä teknologioista vaati kuitenkin hyvinkin yksityiskohtaista perehtymistä, kuten tarvittavan AWS-infrastruktuurin määrittely julkaisua varten käyttäen Amazon CDK:ta. Tältä osin palvelun kehittäminen juuri mikropalveluna ja julkaiseminen pilvipalvelussa lisäsi projektin haastavuutta.

Projektissa käytetty Msal-node-moduuli osoittautui erinomaiseksi kirjastoksi OAuth-yhteyskäytännön toteuttamista varten Microsoft Azure -ympäristössä. Se teki kehittäjäkokemuksesta mielekkään ja toisinaan myös helpon tuntuisen. Msal-node piilottaa kehittäjältä osan OAuth-yhteyskäytännöstä niin, ettei kehittäjän itse tarvitse huolehtia kaikista yhteyskäytäntötason yksityiskohdista. Msal-node-moduulin sisäänrakennetut ominaisuudet tiedon automaattiselle tallentamiselle ja käyttöoikeustietueiden päivittämiselle osoittautuivat myös hyödyllisiksi. Tämän lisäksi Msal-node, kuten myös Microsoftin valtuutuspalvelin ja Azure-pilvipalvelu kokonaisuudessaan, ovat hyvin dokumentoitu ja vastaukset projektin kehityksen aikana esiintyneisiin ongelmatilanteisiin ja kysymyksiin löytyivät yleensä nopeasti.

Päätämme insinöörityön toteamalla, että OAuth-standardin toteuttaminen mikropalveluna ja Microsoft Azure -palveluympäristössä ei ole ainoastaan mahdollista, vaan on parhaassa tapauksessa ja oikeita työkaluja käyttäen myös kehittäjäkokemukseltaan mainiota.

Lähteet

1. The OAuth 2.0 Authorization Framework. 2012. Verkkoaineisto. <https://datatracker.ietf.org/doc/rfc6749/> Päivitetty 21.1.2020. Luettu 3.11.2021.

2. Welcome To OpenID Connect. OpenID Foundation. Verkkoaineisto. <https://openid.net/connect/>. Luettu 3.11.2021.

3. OAuth 2.0 and OpenID Connect. 2018. OktaDev. Verkkoaineisto. <https://www.youtube.com/watch?v=996OiexHze0>. Kuunneltu 3.11.2021.

4. Degges, Randall. Nobody Cares About OAuth or OpenID Connect. 2019. Verkkoaineisto. Okta <https://developer.okta.com/blog/2019/01/23/nobody-cares-about-oauth-or-openid-connect>. Luettu 3.11.2021.

5. Wanpeng Li, Chris J. Mitchell, Thomas Chen. Your code is my code: Exploiting a common weakness in OAuth 2.0 implementations. Julkaisussa: Security Protocols XXVI: 26th International Workshop, Cambridge, UK, March 19-22, 2018. Revised selected papers (pp. 24-41) Sähköisesti saatavilla: <https://www.researchgate.net/publication/329145654_Your_Code_Is_My_Code_Exploiting_a_Common_Weakness_in_OAuth_20_Implementations>.

6. Introduction to JSON Web Tokens. Auth0. Verkkoaineisto. <https://jwt.io/introduction>. Luettu 3.11.2021.

7. Microsoft Identity Platform. Microsoft. Verkkoaineisto. <https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-overview>. Luettu 3.11.2021.

8. Microsoft Azure. Microsoft. Verkkoaineisto. <https://azure.microsoft.com/en-us/>. Luettu 3.11.2021.

9. Azure Active Directory. Microsoft. Verkkoaineisto. <https://azure.microsoft.com/en-us/services/active-directory/>. Luettu 3.11.2021.

10. Microsoft Azure Tenant. Microsoft. Verkkoaineisto. <https://docs.microsoft.com/en-us/azure/active-directory/develop/quickstart-create-new-tenant>. Luettu 3.11.2021.

11. Microsoft Azure Portal. Microsoft. Verkkoaineisto. <https://azure.microsoft.com/en-us/features/azure-portal/>. Luettu 3.11.2021.

12. Quickstart: Register an application with the Microsoft identity platform. Microsoft. Verkkoaineisto. <https://docs.microsoft.com/en-us/azure/active-directory/develop/quickstart-register-app>. Luettu 3.11.2021.

13. Express/Node Introduction. Mozilla. Verkkoaineisto. <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction>. Luettu 3.11.2021.

14. Axios. Verkkoaineisto. <https://axios-http.com/> Luettu 3.11.2021

15. Getting started. Axios. Verkkoaineisto. <https://axios-http.com/docs/intro>. Luettu 3.11.2021.

16. Dotenv. Verkkoaineisto. <https://www.npmjs.com/package/dotenv>. Luettu 3.11.2021.

17. Msal. Microsoft. Verkkoaineisto. <https://docs.microsoft.com/en-us/azure/active-directory/develop/msal-overview>. Luettu 3.11.2021.

18. Authentication Flows. Microsoft. Verkkoaineisto. <https://docs.microsoft.com/en-us/azure/active-directory/develop/msal-authentication-flows>. Luettu 3.11.2021.

19. NPM. Verkkoaineisto. <https://www.npmjs.com/>. Luettu 3.11.2021.

20. Logging in MSAL.js. Microsoft. Verkkoaineisto. <https://docs.microsoft.com/en-us/azure/active-directory/develop/msal-logging-js>. Luettu 3.11.2021.

21. Sign in users in a Node.js & Express web app. Microsoft. Verkkoaineisto. <https://docs.microsoft.com/en-us/azure/active-directory/develop/tutorial-v2-nodejs-webapp-msal>. Luettu 7.11.2021.

22. Public client and confidential client applications. Microsoft. Verkkoaineisto. <https://docs.microsoft.com/en-us/azure/active-directory/develop/msal-client-applications>. Luettu 3.11.2021.

23. Permissions and consent in the Microsoft identity platform. Microsoft. Verkkoaineisto. <https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-permissions-and-consent>. Luettu 3.11.2021.

24. AWS CDK. Amazon. Verkkoaineisto. <https://aws.amazon.com/cdk/>. Luettu 3.11.2021.