Tampereen yliopisto

Mikael Korpimaa

KÄYTTÄJÄHALLINTA MONIASIAKAS-YMPÄRISTÖSSÄ

Diplomityö
Informaatioteknologian ja viestinnän tiedekunta
Kari Systä
Marko Helenius
Kesäkuu 2021

TIIVISTELMÄ

Mikael Korpimaa: Käyttäjähallinta moniasiakasympäristössä
Diplomityö
Tampereen yliopisto
Informaatioteknologian ja viestinnän tiedekunta
Kesäkuu 2021

Software as a service -palvelumalli on yleistynyt ohjelmistokehityksen keskuudessa. Palveluiden siirtyessä yksityisistä asennuksista software as a service -palvelumalliin, täytyy useita komponentteja muuttaa palvelemaan useampaa asiakasta. Yksityisiä asennuksia käyttäessä palvelut ovat toisistaan riippumattomia. Myydyn tuotteen asiakaskunnan kasvaessa myös tarvittava ylläpidon määrä yksityisten asennusten kanssa lisääntyy. Tekemällä komponenteista asennusten välille yhteisiä ja siirtymällä kohti software as a service -palvelumallia, voidaan vähentää tarvittavan ylläpidon määrää.

Moniasiakaskäyttäjähallintaa lähdetään kehittämään hyödyntäen Microsoftin Azure AD B2C -käyttäjähallintaa. Valittu teknologia ei suoraan täytä roolipohjaiselle moniasiakaskäyttäjähallinnalle asetettuja vaatimuksia, mutta se tarjoaa hyvin laajan muokattavuuden sen käytön suhteen. Työssä selvitetään kuinka, valitulla teknologialla voidaan toteuttaa roolipohjainen käyttöoikeuksien tarkastelu ja miten se saadaan palvelemaan useampaa asiakasta.

Työtä varten Azure AD B2C:sta muokattiin konseptitason toteutus käyttäjähallista, joka mahdollistaa roolipohjaisen käyttöoikeuksien hallinnan. Toteutus mahdollistaa eritasoisten roolien jakamisen sen käyttäjille ja käyttäjien pääsy saadaan rajattua vain asiakasympäristöihin, joihin heillä on oikeudet. Käyttäjähallinnan käyttöönotto kuitenkin vaatii vielä jatkokehitystyötä integraation suhteen.

Avainsanat: Azure AD B2C, Azure Functions, käyttäjähallinta, moniasiakkuus, SaaS

ABSTRACT

Mikael Korpimaa: User management in a multi-tenant environment
Master's Thesis
Tampere University
Faculty of Information Technology and Communication Sciences
June 2021

Software as a service model has become common amongst software development. As services move from private installations to software as a service model, many components need to change to serve multiple tenants. When using private installations, the services are independent of each other. As the customer base of the product grows, the amount of maintenance required with private installations also increases. By making the components common between installations and moving towards a software as a service model, the amount of maintenance required can be reduced.

Multi-tenant user management will be developed utilizing Microsoft's Azure AD B2C user management. The technology chosen does not directly meet the requirements for multi-tenant role-based user management, but it offers a very wide range of customizability in terms of its use. It is researched how the selected technology can be used to achieve role-based access control and how it can be made to serve multiple tenants.

A proof-of-concept implementation of a role-based multi-client user management was created by customizing Azure AD B2C user management. The implementation allows for the allocation of different levels of roles to its users and allows users to restrict access only to the client environments to which they have the access rights. However, the introduction of user management still requires further development work in terms of integration to a full working system.

Keywords: Azure AD B2C, Azure Functions, user management, multi-tenant, SaaS

ALKUSANAT

Kiitän työnantajaani, Solita Oy:tä diplomityön aiheesta ja mahdollisuudesta suorittaa opinnäytteeni osana jokapäiväistä työtäni. Suuret kiitokset kuuluvat Harri Kalliolle ja Marko Kattelukselle, joiden kanssa suunnittelimme työn toteutusta. Sain täysin vapaat kädet työn toteutusta tehdessä. He myös tukivat tarvittaessa luomieni ratkaisujen ongelmakohtien ratkaisussa. Kiitokset myös muille kollegoilleni, joiden kanssa olen päässyt kehittämään Solitan Agile Data Engineä.

Kiitokset myös Kari Syställe työn ohjauksesta ja Marko Heleniukselle työn tarkastamisesta. Työn kirjoitusvaiheet venyivät diplomityön loppuvaiheille ja heille jäi paljon kerralla tarkastettavaa. Kiitän teitä hyvistä vihjeistä ja työn korjausehdotuksista.

Haluan kiittää myös isääni, DI Heikki Korpimaata, jonka vaikutuksesta päädyin suorittamaan diplomityötutkinnon. Hän on kannustanut minua koko opintojeni ajan ja Ilman häntä en olisi näin pitkällä.

Tampereella, 23.6.2021

Mikael Korpimaa

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

AD	Active Directory
ADE	Agile Data Engine
B2B	Business to Business
B2C	Business to Consumer
SaaS	Software as a Service
FaaS	Function as a Service
JWT	JSON Web Token

1. JOHDANTO

Ohjelmistotuotteen siirtyessä yksityisistä asiakkaiden tuotantoympäristöjen asennuksista software as a service (SaaS) palvelumalliin, monia ohjelmistokomponentteja muutetaan yhteisiksi komponenteiksi, jotka palvelevat useampaa asiakasta. Näistä komponenteista moniasiakaskäyttäjähallinta on olennainen osa SaaS toimintamallia. Sen avulla voidaan keskittyä hallinnoimaan yksittäistä käyttäjähallintaa sen sijaan, että jokaista asiakkuutta varten pitäisi ylläpitää omaa erillistä käyttäjähallintaa. Useamman asiakkuuden hallinta helpottuu, kun saadaan vähennettyä asiakaskohtaisten komponenttien määrää järjestelmässä.

Tietoturvallisen käyttäjähallinnan kehittäminen tyhjästä on pitkä prosessi ja jatkuvasti esiintyvien turvallisuusriskien korjaaminen vaatii paljon ylläpitoa. Käyttäjähallintaa valittaessa on tarjolla monia luotettaviksi todettuja ratkaisuja, jotka saavat jatkuvasti tietoturvapäivityksiä. Olemassa olevia käyttäjähallintoja hyödyntäessä ulkoistetaan turvallisuusriskien korjaaminen sen omille kehittäjille ja voidaan hyödyntää sen valmiita turvallisiksi todettuja toimintoja.

Työssä hyödynnetään jo olemassa olevaa käyttäjähallintaa, jota muokataan tarjoamaan roolipohjainen käyttöoikeuksien hallinta ja palvelemaan useampaa sovellusta tai asiakasta. Roolipohjaisen käyttöoikeuksien hallinnan myötä voidaan jakaa oikeuksia antamalla käyttäjille rooleja. Roolien avulla käyttäjien oikeuksia voidaan rajata asiakkuuksien sisäisesti ja mahdollistetaan käyttäjien oikeuksien rajaaminen vain valittuihin asiakkuuksiin.

Työn lopputuloksena on konseptitason toteutus, joka täyttää nämä vaatimukset. Se toimii lähtökohtana jatkokehitykselle, jossa korvataan aiemmat asiakaskohtaiset käyttäjähallinnat työssä luodulla roolipohjaisella moniasiakaskäyttäjähallinnalla.

2. TYÖN LÄHTÖKOHTA JA TAVOITE

Työ tehdään osana Solitan Agile Data Engine (ADE) kehitystä. ADE:ssa halutaan käyttää valmista identiteetin tarjoajaa, koska se tarjoaa suoraan turvallisen kirjautumisen, salasanan vaihtamisen, monivaiheisen tunnistautumisen ja monia muita turvalliseen käyttäjähallintaan liittyviä ominaisuuksia. Valmiin ja tunnetun käyttäjähallinnan hyödyntäminen vähentää työn määrää verrattuna täysin uuden ratkaisun luomiseen. Käyttäjähallinnaksi on valittu Microsoftin Azure Active Directory business to consumer (Azure AD B2C) sen muokattavuuden takia. Azure AD B2C:n tapauksessa vastuu turvallisuuspäivityksistä voidaan jättää Microsoftille. Asiakkaan halutessa käyttää heidän organisaationsa jo olemassa olevia Azure Active Directory tilejä, voidaan työssä luotavan käyttäjähallinnan sijaan konfiguroida sovelluksessa käytetty identiteetintarjoaja käyttämään asiakkaan omaa käyttäjähakemistoa.

2.1 Asiakasympäristöt

ADE on Data DevOps kehitys ja operointiympäristö pilvipohjaisten tietovarastojen rakentamiseen ja käyttöön, jonka toiminnallisuus on kokonaan pilvessä. Kun asiakkaalle myydään ADE tuotteena, heille asennetaan oma yksityinen asennus, jota voidaan kutsua asiakasympäristöksi. Näissä asiakasympäristöissä kaikki ohjelmistokomponentit ja tietokannat ovat heidän yksityisessä käytössänsä.

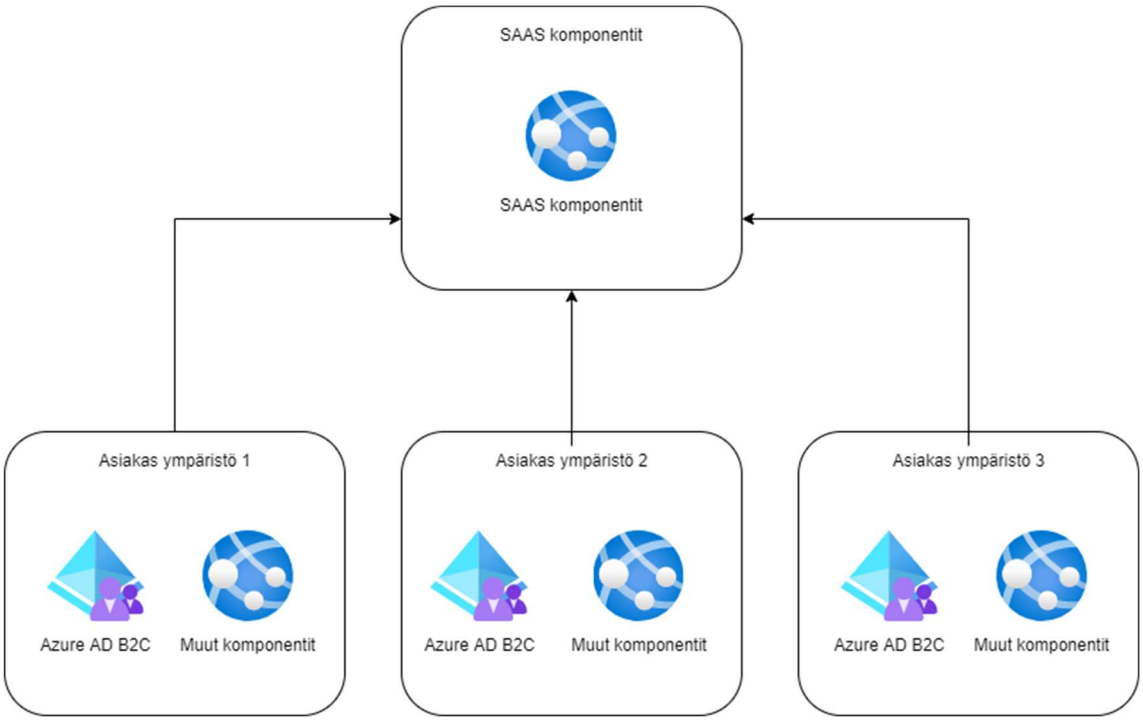
Asiakasympäristöjen välillä on hyvin vähän eroja. Joistakin asiakasympäristöistä voi puuttua joitain komponentteja, jos asiakas ei tarvitse kaikkia ADE:n toiminnallisuuksia. Pääasiassa asiakasympäristöt ovat hyvin samanlaisia ja niissä käytetyt komponentit ovat ohjelmallisesti asennettuja.

2.2 Ratkaistava ongelma

Tähän mennessä ADE:ssa käyttäjähallinta on toteutettu tilapäisellä ratkaisulla, jossa jokaisen asiakasympäristön asennusta kohden on pystytetty oma Azure AD B2C -käyttäjähallinta. Nykyisessä ADE:n Azure AD:n B2C -toteutuksessa käyttäjät ovat oikeuksiltaan tasavertaisia. Azure AD B2C ei suoraan tarjoa roolipohjaista käyttäjähallintaa.

Tämä työ on yksittäinen osa ADE:n siirtymistä yksityisistä asennuksista käyttämään Software as a Service -palvelumallia. ADE:n siirtyessä SaaS-palvelumalliin, mahdollisimman monesta komponentista halutaan tehdä riippumaton asiakkaan ympäristöstä. Kuvassa

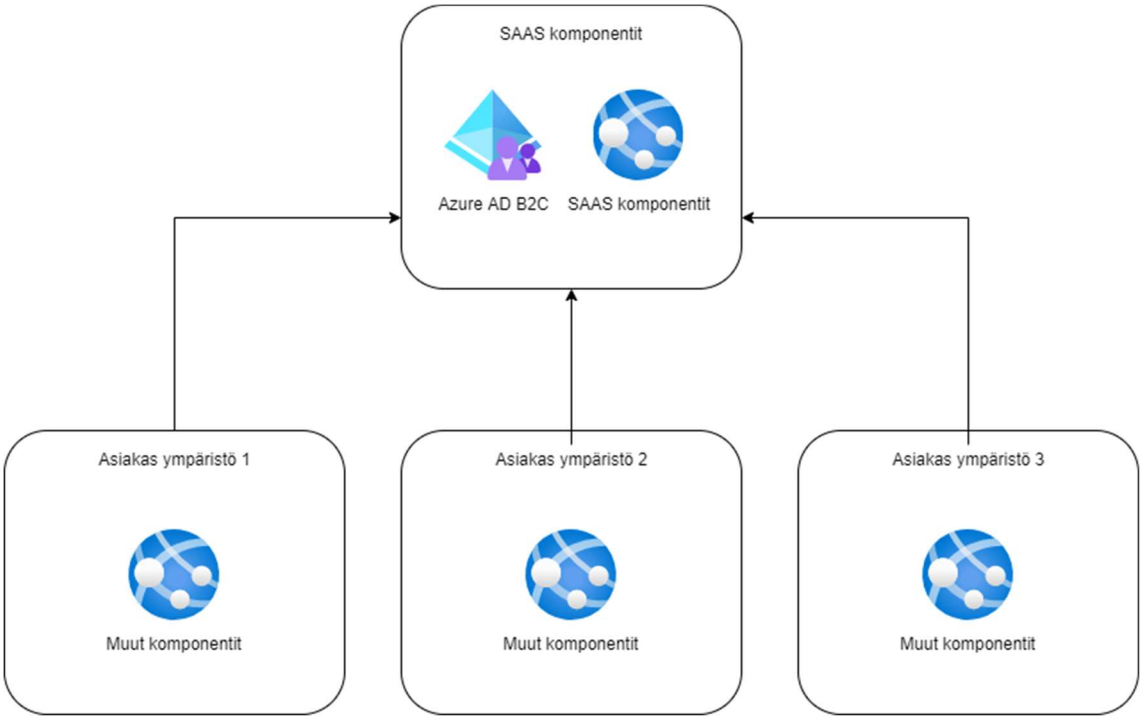
1 on kuvastettu lähtötilanne, jossa jokaisella asiakkaan ympäristöllä on oma Azure AD B2C -palvelu käytössä.



Kuva 1. *SaaS lähtötilanne yksinkertaistettuna.*

Asennuksissa on paljon komponentteja, jotka ovat monistettu jokaiselle asennukselle. Azure AD B2C halutaan poistaa asiakasympäristöistä ja muuntaa yhteiseksi SaaS komponentiksi. SaaS komponenttina se voidaan konfiguroida palvelemaan useampaa asiakasympäristöä. Samanlainen siirtymä tulisi myös tehdä monelle muulle komponentille, mutta tässä työssä keskitytään vain käyttäjähallintaan.

Kuvassa 2 on hyvin yksinkertaisesti esitettynä työn tavoiteltu tila, jossa asiakasympäristöt hyödyntävät vain yhtä yhteistä käyttäjähallintaa. Käyttäjähallinnan muuttaminen SaaS komponentiksi vähentäisi tarvittavan ylläpidon määrää, kun käytössä olisi vain yksi ylläpidettävä käyttäjähallinta. Azure AD B2C:ssä ei kuitenkaan ole valmista toiminnallisuutta erotella käyttäjiä omiin asiakkuuksiin ja rajata käyttäjien oikeuksia tiettyihin asiakasympäristöihin. Käyttäjähallintaan pitää konfiguroida logiikkaa, jolla käyttäjät pystytään rajaamaan omiin asiakkuuksiinsa ja määrittämään käyttäjille rooleja, joilla voidaan asiakasympäristöissä rajata käyttäjien oikeuksia.



Kuva 2. *SaaS tavoitetila.*

Roolien ja käyttäjien hallinta ja uusien asiakkaiden asennusten lisäys käyttäjähallintaan tulisi tapahtua ohjelmallisesti. Tämä mahdollistaisi erillisen hallintapaneelin luomisen ja uusien asennusten automatisoinnin. Työssä käydään läpi, miten kaikki edellä mainitut asiat voidaan tehdä ohjelmallisesti, mutta lopullinen toteutus jää työn ulkopuolelle.

2.3 Vaatimukset käyttäjähallinnalle

Käyttäjähallinnassa käyttäjien tulee löytyä yksittäisestä käyttäjähakemistosta tai tietokannasta. Jos hakemistoja ja tietokantoja lisätään käyttäjien ryhmittelyä varten, kasvaa tarvittavan ylläpidon määrä. Käyttäjien jaottelu eri sovellusten käyttäjiksi tulee toteuttaa jollakin muulla tavalla.

Työssä muokattavan käyttäjähallinnan täytyy palvella useampaa asiakasympäristöä. Käyttäjien kirjautuminen tulee rajata asiakasympäristöihin, joihin heillä on oikeudet. Asiakasympäristöihin pääsyn lisäksi käyttäjille pitää pystyä antamaan eritasoisia oikeuksia, joiden avulla käyttäjien toimintaa voidaan rajata asiakasympäristöjen sisällä.

Käyttäjähallinnan tulee olla turvallinen. Kirjautuminen pitää vahvistaa monivaiheisella tunnistautumisella ja käyttäjillä pitää olla turvallinen tapa palauttaa oma salasanansa. Salasanoja ei saa tallentaa muualle kuin käyttäjähallinnan omaan tietokantaan.

Käyttäjähallinnan hallinnointi tulee pystyä tekemään ohjelmallisesti. Ohjelmallinen hallinnointi mahdollistaa automaation uusia asiakasympäristöjä asennettaessa. Ne mahdollistavat myös uusien rajapintojen ja käyttöliittymien luonnin käyttäjiä ja kehittäjiä varten.

Kirjautumisen teema halutaan muokata vastaamaan ADE:n ulkoasua. Teemoja muokkaamalla käyttäjäkokemuksesta saataisiin sulavampi. Yhtenäisten teemojen avulla käyttäjä ei pystyisi erottamaan kirjautumisen tapahtuvan erillisen palvelun kautta.

3. YLEISTÄ TAUSTATIETOA

Tässä kappaleessa käydään läpi yleistä taustatietoa, joka on tärkeä tietää työtä ymmärtääkseen. Myöhemmin työssä tullaan mainitsemaan näitä termejä ja oletetaan lukijan tietävän mitä ne tarkoittavat.

3.1 Software as a Service

Software as a Service (SaaS) on palveluiden tarjoamisessa käytetty lähestymistapa. SaaS-toimintamallissa palveluiden tarjoajat voivat jakaa yksittäistä palvelua monille asiakkaille. Jokaiselle asiakkaalle ei tarvitse asentaa omaa tuotetta. SaaS-toimintamallista hyötyvät sekä palvelun tarjoajat että asiakkaat. Asiakkailla on mahdollisuus käyttää palveluja, jotka ovat tehokkaita, skaalautuvia ja luotettavia ilman että heidän tarvitsee pystyttää tai hallita palveluita itse. SaaS-toimintamallin avulla palvelun ylläpitäjien työtä saadaan myös vähennettyä. Ylläpitäjien täytyy vain ylläpitää yksittäistä SaaS-palvelua. [1]

SaaS-toimintamallissa kustannukset ovat huomattavasti pienempiä kuin yksityisissä asennuksissa. Jos asiakkaalla käyttötaso on alhainen, SaaS-toimintamallissa heitä voidaan laskuttaa vain käytön perusteella ja kustannuksia saadaan laskettua ajoilta, jolloin palvelua ei käytetä. [1]

3.2 Tunnistautuminen ja valtuutus

Tunnistautumisessa (Authentication) varmennetaan käyttäjän olevan kuka hän väittää olevansa. Valtuutuksessa (Authorization) varmistetaan, onko käyttäjällä oikeuksia käyttää jotakin resurssia tai suorittaa jokin toiminto. Valtuutuksen edellytyksenä yleensä on käyttäjän tunnistautuminen. Valtuutus voidaan myös delegoida toiselle osapuolelle. Delegoidussa valtuutuksessa annetaan oikeus toiselle henkilölle tai sovellukselle toimia sinun puolestasi. [2][3]

Tunnistautumisprosessissa käyttäjä kertoo ohjelmalle, kuka hän väittää olevansa. Varmenteena käyttäjän identiteetille käytetään usein käyttäjätunnusta ja salasanaa. Jos käyttäjän salasana vastaa hänen väittämänsä identiteetin salasanaa, käyttäjä todennäköisesti on, kuka hän väittää olevansa. Käyttäjältä voidaan vaatia myös lisätoimenpiteitä kuten monivaiheista tunnistautumista, jossa hänelle lähetetään sähköpostilla tai tekstiviestillä kerran käytettävä salasana varmempaa tunnistautumista varten. [2][4]

Valtuutuksessa määritetään, mihin resursseihin käyttäjällä on oikeus päästä käsiksi. Valtuutus voidaan toteuttaa esimerkiksi rooli pohjaisella käyttäjähallinnalla (RBAC). Tällöin rooleille voidaan asettaa joukko oikeuksia, joiden avulla voidaan rajata pääsyä erinäisiin resursseihin. Käyttäjien oikeudet tulkitaan roolien perusteella. [4]

3.3 Identiteetin tarjoajat

Identiteetin tarjoajat ovat sovelluksista ulkoistettuja, joiden kautta käyttäjät voivat tunnistautua sovelluksiin. Tämä mahdollistaa käyttäjien pääsyn moniin itsenäisiin palveluihin käyttäen samaa identiteettiä. Käyttäjien tiedot ja käyttöoikeuksien hallinta voidaan keskittää identiteetin tarjoajien hallintaan. Identiteettien federaatioiden avulla voidaan laajentaa käyttäjien pääsyä palveluihin, jotka ovat heidän organisaatiorakenteensa ulkopuolella. Federaatiolla tarkoitetaan palveluiden ja identiteetin tarjoajien välistä luottamusta, joka on saavutettu sopimalla yhteiset protokollat tunnistautumista varten. Yksi tunnetuin federointiin käytetty protokolla on SAML. [5]

Azure AD:n tapauksessa federaatio tapahtuu käyttämällä Microsoftin omaa Active Directory Federation Service -komponenttia (ADFS). Sen avulla Azure AD:n käyttäjät voivat turvallisesti käyttää monia palveluita yksittäisillä federoiduilla identiteeteillä. [6]

3.4 Asiakkuusmallit

Asiakkuusmallit voidaan jakaa kahteen eri luokkaan: Yksittäiseen asiakkuuteen (Single Tenant) ja moniasiakkuuteen (Multi-Tenant). Kun puhutaan asiakkuuksista, on tärkeä huomioida, minkä palvelun näkökulmasta asiakkuutta tarkastellaan. Yksittäis- tai moniasiakkuus kuvastavat kuinka monta asiakasta jokin palvelu palvelee.

3.4.1 Yksittäinen asiakas

Yksittäisessä asiakkuudessa sovellus palvelee vain yhtä asiakasta. Tällöin jokaisella asiakkaalla on omat tietokannat ja asennukset sovelluksista. Yksittäisen asiakkuuden etuina on sen turvallisuus, resurssien saatavuus ja muokattavuus. Käyttäjähallinta on turvallisempi, koska toisten asiakkuuksien käyttäjät eivät ole samassa käyttäjähakemistossa. Palvelimien resurssit ovat asiakaskohtaisia, jolloin yksittäisten asiakkuuksien käyttö ei hidasta muiden asiakkuuksien käyttöä. Yksittäinen asennus myös mahdollistaa laajemman sovelluksen muokattavuuden jokaista asiakkuutta kohden. [7]

Mahdollisia haittoja yksittäisissä asiakkuuksissa on ylläpitotehtävien moninkertaistuminen, työläämpi asennus ja korkeammat kustannukset. Kun jokaisella asiakkuudella on

oma käyttäjähallinta, täytyy jokaista ylläpitää erikseen. Uusia asiakkuuksia luodessa täytyy aina pystyttää uusi käyttäjähallinta, joka tekee asiakkuuksien lisäämisestä työläämpää. Kun käyttäjähallintoja on useampia, kustannukset kasvavat myös suuremmiksi. [7]

3.4.2 Moniasiakkuus

Moniasiakkuudessa yksi sovellus palvelee montaa asiakasta. Tällöin sovelluksessa asiakkaat jakavat tietokannat ja asennetun sovelluksen. Moniasiakkuuden etuina on pienemmät kustannukset, vähemmän työläs ylläpito ja helpompi uusien asiakkuuksien lisääminen. Sen sijaan että maksettaisiin jokaisen asiakkuuden kohdalla erikseen käyttäjähallinnasta, voidaan kuluja vähentää käyttämällä vain yhtä käyttäjähallintaa. Ylläpitotehtävät kohdistuvat vain yksittäiseen käyttäjähallintaan, jolloin ylläpitotehtäviin käytettävä aika vähenee huomattavasti. Uusien asiakkuuksien lisäys on yksinkertaisempaa, koska jokaiselle asiakkaalle ei tarvitse asentaa omaa käyttäjähallintaansa. [7]

Moniasiakkuudessa tulee kuitenkin vastaan mahdollisia haasteita ja riskejä. Käyttäjähallinnan tulee olla yhtenäinen kaikkien asiakkaiden välillä, jolloin itse käyttäjähallintaa ei voi mukauttaa jokaiselle asiakkuudelle omanlaiseksi. Moniasiakkuuden myötä kasvaa myös riski, jossa eri asiakkuuksien käyttäjät voivat mahdollisesti päästä käsiksi toisen asiakkuuden resursseihin. Moniasiakkuudessa täytyy kiinnittää huomattavasti enemmän huomiota käyttäjien oikeuksiin ja pitää pystyä varmistamaan, etteivät käyttäjät pääse käsiksi tietoihin, joihin heillä ei ole asiaa. Kun käyttäjähallintaan tulee muutoksia, jotka myös vaativat muutoksia asiakkaiden puolella, täytyy varmistaa, että kaikki asiakasympäristöt päivitetään mahdollisimman nopeasti. [7]

4. YLEISESTI KÄYTETYT TUNNISTAUTUMIS- JA VALTUUTUSTAVAT

Käyttäjien tunnistautumista ja valtuutusta varten on jo tehty paljon tutkimuksia ja niitä varten on jo luotu useita erilaisia standardeja. Tässä luvussa käydään läpi muutamia protokollia ja teknologioita, joita yleisesti käytetään käyttäjien tunnistautumisessa ja valtuutuksessa.

4.1 JSON Web Token

JSON Web Token (JWT) on yleinen työkalu palvelimien ja käyttäjien välisessä kanssakäymisessä. Se on objekti, joka sisältää tietoa kutsuja tekevästä käyttäjästä. Sen avulla voidaan varmentaa kutsujen tekijä. JWT on merkkijono, joka koostuu kolmesta osiosta: Otsikosta (header), sisällöstä (payload) ja allekirjoituksesta (signature). Otsikossa esitetään allekirjoituksen salauksessa käytetty algoritmi. JWT allekirjoitetaan yksityisillä avaimilla, jotka voidaan varmentaa niiden julkisilla vastakappaleilla. Yksityiset avaimet jäävät vain allekirjoittavan osapuolen tietoon ja julkinen avain jaetaan vastaanottajalle allekirjoituksen varmentamista varten. JWT:n sisältö koostuu kentistä, joita kutsutaan väitteiksi (claim). Ne voivat sisältää tietoa esimerkiksi JWT:n luonnista tai kutsun tekevästä käyttäjästä. [8] [9]

```
{
 "typ": "JWT",
 "alg": "RS256"
}.{
 "exp": 1622019213,
 "nbf": 1622015613,
 "iss": "https://esimerkki.domain.com/taalta-saa-julkisen-avaimen/",
}.[Allekirjoitus]
```

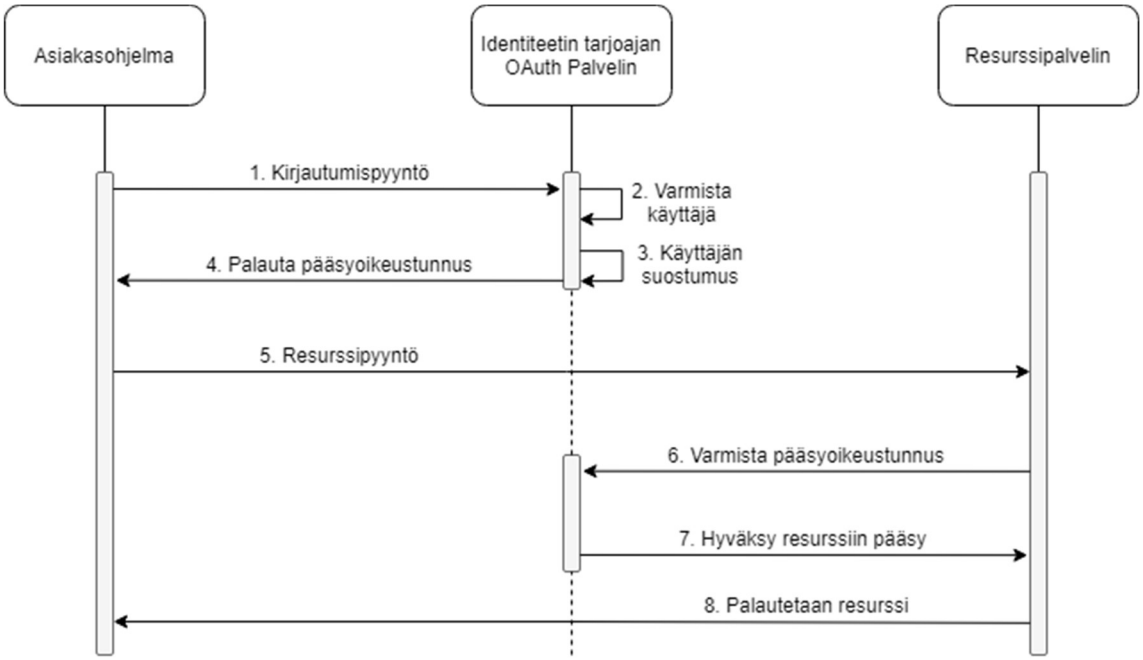
Ohjelma 1. *Esimerkki JWT tokenin sisällöstä.*

Ohjelmassa 1 on esimerkki JWT tokenin sisällöstä. Sen otsikko, sisältö ja allekirjoitus on eroteltu toisistaan pisteen avulla. JWT:n aitouden varmistamista varten täytyy tarkistaa ettei tokenin sisältöä ole muokattu. JWT:n allekirjoituksessa käytettyä yksityistä avainta vastaavan julkisen avaimen saa haettua *iss* väitteestä löytyvästä osoitteesta. Otsikosta löytyvän salauksen algoritmin ja noudetun julkisen avaimen avulla voidaan varmistaa allekirjoituksen vastaavan JWT:n sisältöä. JWT:n sisällöstä löytyy myös *exp* väitteestä ajanhetki tokenin vanhenemiselle ja *nbf* väitteestä voimaantulon alkamisajankohta. Sovelluksen tulisi varmistaa, ettei JWT ole vanhentunut tai sen alkamisajankohta ole tulevaisuudessa. Näiden väitteiden lisäksi JWT:n sisältöön voidaan lisätä mitä tahansa tietoa. [9]

4.2 OAuth tunnistautuminen ja valtuutus

OAuth on käyttäjien ja palveluiden tunnistautumista ja valtuutusta varten kehitetty protokolla. Se sai alkunsa ohjelmointirajapintapohjaisten sovellusten yleistyessä. Kun Google julkaisi ensimmäisen kerran Google-kalenterin ohjelmointirajapinnan, se mahdollisti käyttäjien kalenterien manipuloinnin ohjelmointirajapintojen avulla. Ainoa tapa jakaa kolmannen osapuolen sovelluksille oikeuksia tehdä muokkauksia kalenteriin, vaati käyttäjiä luovuttamaan sovelluksille heidän käyttäjätunnuksensa ja salasanansa. Salasanojen luovutus ulkoisille sovelluksille nähtiin turvallisuusriskinä, jonka ratkaisemiseksi kehitettiin OAuth-protokolla. [2]

OAuth kehitettiin turvallisen tunnistautumisen ja oikeuksien delegoimisen mahdollistamiseksi kolmannen osapuolen sovelluksille. Se tarjoaa mahdollisuuden sovelluksille käyttää käyttäjien tietoja turvallisesti ilman, että käyttäjien täytyy luovuttaa käyttäjätunnuksiaan. OAuth-protokollan mukaisesti käyttäjät ohjataan identiteetin tarjoajan palvelimelle suorittamaan kirjautuminen. Palvelimella varmistetaan käyttäjän syöttämät tunnukset ja varmistetaan käyttäjän suostumus hänen resurssiensa jakoon. Käyttäjän suostumuksen jälkeen kolmannen osapuolen sovellukselle välitetään pääsyoikeustunnus (access token), joka usein on JWT-muodossa. Kuvassa 3 on esitetty käyttötapauskaavio käyttäjän kirjautumisesta kolmannen osapuolen sovellukseen ja käyttäjän resurssien noutamisesta.



Kuva 3. *OAuth-protokollan tunnistautumisen ja valtuutuksen käyttötapauskaavio. [10]*

Kun palvelu on saanut identiteetin tarjoajalta pääsyoikeustunnuksen, se voi noutaa käyttäjän myöntämiä resursseja. Resurssien nouto tehdään erillisellä kutsulla palvelimelle, jonka hallinnassa resurssit ovat. Kutsun yhteydessä välitetään identiteetin tarjoajalta saatu pääsyoikeustunnus. Resurssipalvelin varmistaa saadun pääsyoikeustunnuksen identiteetin tarjoajan OAuth-palvelimelta ja palauttaa pyydetyt resurssit. [11]

4.3 JSON Web Token -pohjainen valtuutus

OAuth-protokollassa tunnistautumisen yhteydessä saatiin pääsyoikeustunnus JWT muodossa. Luvussa 4.2 esitettiin tapa valtuuttaa käyttäjälle oikeudet resursseihin, jotka löytyvät erilliseltä palvelimelta. Jos kolmannen osapuolen sovellus haluaa rajata sovelluksen omien resurssien oikeuksia käyttäjälle, sen täytyy pystyä valtuuttamaan käyttäjä itsenäisesti.

Kutsujen mukana lähetetty JWT voi kuitenkin sisältää tietoa erinäisistä entiteeteistä, käyttäjistä tai rooleista valtuutusta varten. Määrittämällä JWT:hen käyttäjälle kuuluvat roolit, voidaan sovelluksessa rajata oikeuksia tiettyihin resursseihin niiden perusteella. Tällaisessa valtuutuksessa JWT:n aitouden ja alkuperäisyyden tarkistus on kriittistä. [12]

Käyttäjien roolien sisällyttäminen JWT:hen ja varmentamalla roolit sovelluksissa on suorituskyvyltään parempi kuin erillinen resurssien haku identiteetin tarjoajan valtuuttamalta

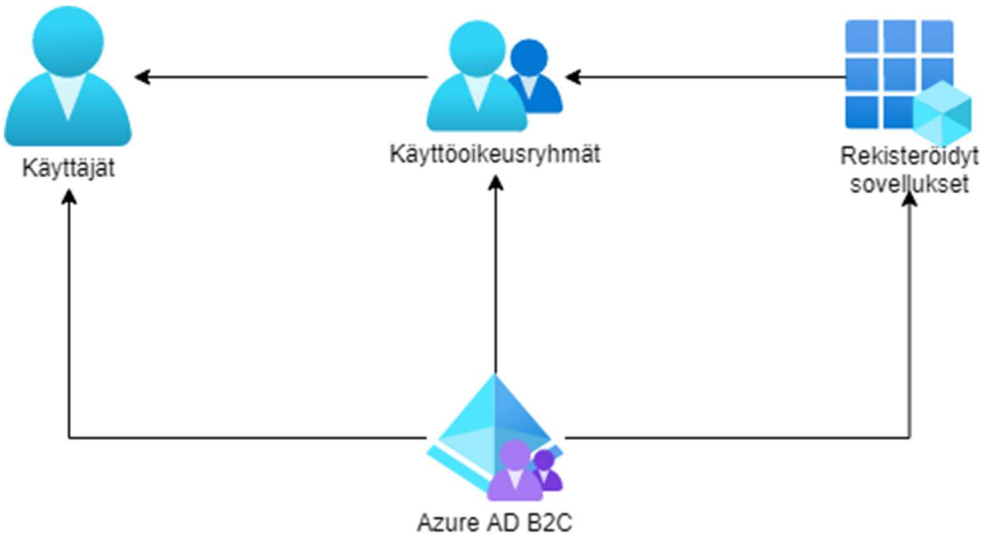
resurssipalvelimelta. Tällöin sovellus tekee vain yhden kutsun tunnistautumista varten, jonka jälkeen sillä on tiedot kaikista käyttäjän oikeuksista. Tämän ratkaisun negatiivisena vaikutus näkyy kuitenkin JWT:n pituudessa. [12]

5. TOTEUTETTAVAN KÄYTTÄJÄHALLINNAN TOIMINTA

Tässä luvussa käydään läpi työssä luotavan käyttäjänhallinnan toimintaa. Lisäksi esitellään käyttäjähallinnassa käytettävät entiteetit ja luotavat prosessit. Roolien käyttöä sovelluksissa. Käyttäjähallinnan toiminnan muokkaus toteutetaan luvussa 8.

5.1 Käyttäjähallinnassa hyödynnettävät entiteetit

Entiteetti on yleinen määritelmä olemassa oleville olioille. Azure AD B2C:ta käyttäessä joudutaan hyödyntämään valmiita entiteettejä. Roolipohjaista käyttäjähallintaa varten tarvitaan seuraavia entiteettejä: käyttäjät, käyttöoikeusryhmät ja rekisteröidyt sovellukset. Nämä entiteetit on hahmotettu kuvassa 4.



Kuva 4. *Käyttäjähallinnassa hyödynnettävät entiteetit.*

Käyttäjäentiteetit ovat käyttäjien identiteettejä. Käyttäjän tunnistautuessa käyttäjätunnukset varmennetaan käyttäjäentiteettiä vasten. Kirjautumisen onnistuessa käyttäjäentiteetistä voidaan hakea käyttäjän tiedot.

Käyttöoikeusryhmiä käytetään käyttäjien oikeuksien jakamiseen. Kun käyttäjiä lisätään käyttöoikeusryhmiin, he saavat ryhmälle kuuluvat oikeudet käyttöönsä. Käyttöoikeusryh-

miin voidaan lisätä useita käyttäjiä ja käyttäjät voivat kuulua useampaan käyttöoikeusryhmään. Käyttöoikeusryhmiä käytetään apuna asiakasympäristöjen sisäisten oikeuksien käsittelyssä.

Rekisteröidyt sovellukset ovat sovelluksia, jotka voivat käyttää Azure AD B2C:n toiminnallisuuksia. Näihin toiminnallisuuksiin voi kuulua esimerkiksi käyttäjähakemiston luku ja kirjoitusoikeudet. Luvussa 2.2 esitetyille asiakasympäristöille rekisteröidään sovellukset, jotka toimivat rajapintana niiden käyttäjien kirjautumiselle. Nämä rajapintasovellukset ovat Azure AD B2C:n näkökulmasta yksittäisiä sovelluksia, joilla on vain käyttäjähakemiston lukuoikeudet. Jos jokin ohjelma tarvitsee oikeuksia käsitellä käyttäjähakemiston resursseja, voidaan sille rekisteröidä oma sovellus ja määrittää sille erilaisia oikeuksia, joiden avulla se pääsee käsiksi erinäisiin resursseihin Azure AD B2C:ssä. [13]

Käyttäjien, käyttöoikeusryhmien ja rekisteröityjen sovellusten hallinnoimista varten joudutaan myös rekisteröimään sovellus. Näiden entiteettien hallinnoimista varten määritetään sovellukselle kattavammat oikeudet.

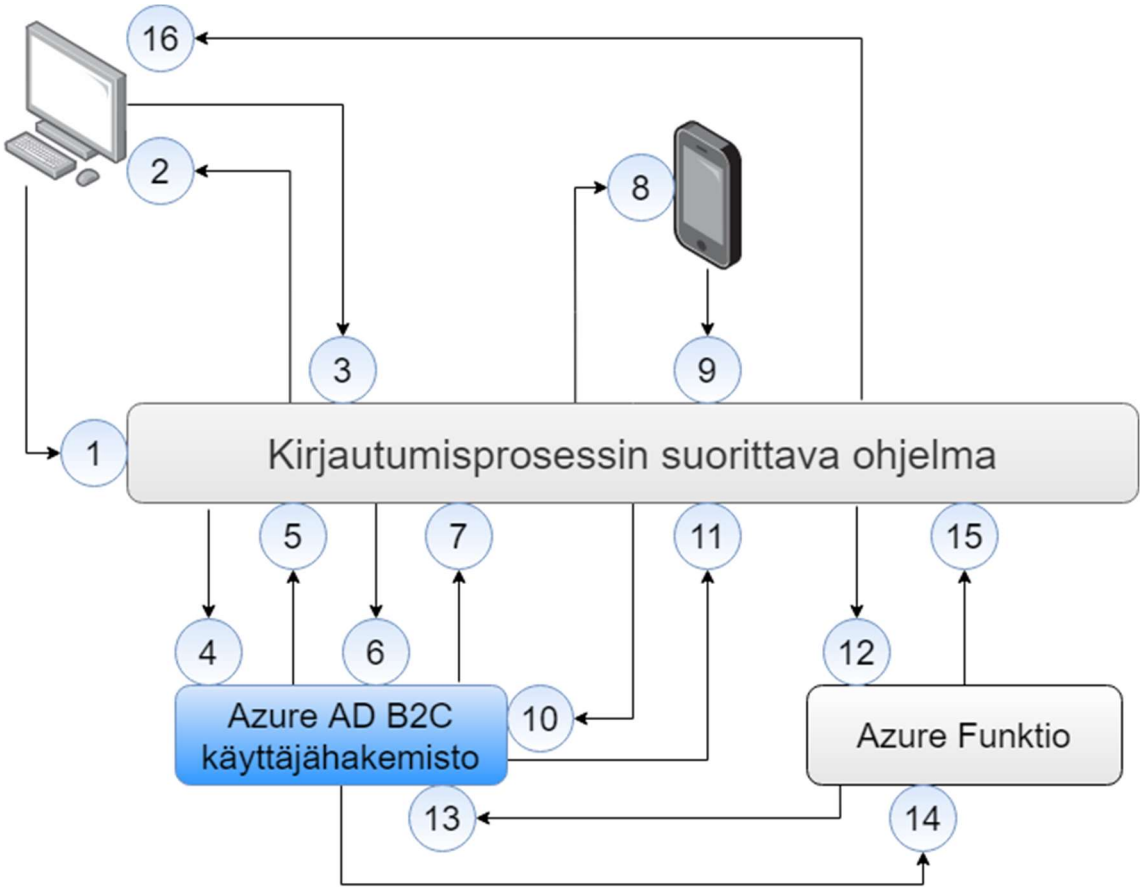
5.2 Luotavat käyttäjäprosessit

Roolipohjaista käyttäjähallintaa varten täytyy pystyä rajaamaan käyttäjien kirjautuminen vain sovelluksiin, joihin heillä on oikeudet. Koska Azure AD B2C ei tarjoa sellaista toiminnallisuutta, täytyy käyttäjäprosesseihin konfiguroida ylimääräisiä askelia käyttäjien roolien noutamiseen ja niiden varmentamiseen. Käyttäjähallintaan määritetään prosessit, jotka hoitavat kirjautumisen ja salasanan palauttamisen. Tämän väliotsikon alla käydään läpi molemmat prosessit askel kerrallaan.

Salasanan palautusprosessia tullaan käyttämään osana uuden käyttäjän luontia. Käyttäjätunnusten salasanoja ei turvallisuussyistä tallenneta erilliseen tietokantaan tai lähetetä tilapäisiä salasanoja sähköpostitse. Kun käyttäjää varten luodaan käyttäjätunnus, salasanaksi generoidaan satunnainen salasana, jota kukaan ei tule näkemään. Käyttäjälle ilmoitetaan hänen käyttäjätunnuksensa luonnista ja hän saa tunnuksensa käyttöön vain salasananpalautusprosessia käyttäen.

5.2.1 Kirjautumisprosessi

Kirjautumisprosessi on esitetty kuvassa 5. Prosessi alkaa käyttäjän mennessä kirjautumissivulle, jolloin Azure AD B2C:n välittää käyttäjälle kirjautumissivun, jonka ulkoasu on generoitu konfiguroidun kirjautumisprosessin mukaisesti (1 ja 2).



Kuva 5. *Kirjautumisprosessin hahmotelma.*

Käyttäjä syöttää käyttäjätunnuksensa ja aloittaa kirjautumisen. Kirjautumiseen käytetyt käyttäjätunnukset välitetään kirjautumisprosessin suorittavalle prosessille (3). Käyttäjän syöttämät käyttäjätunnukset varmennetaan Azure AD B2C -käyttäjähakemiston tietojen perusteella (4). Varmennuksessa palautetaan prosessille vain perustiedot kirjautumiseen liittyen (5). Käyttäjän loput tiedot, jotka halutaan välittää sovellukselle, noudetaan erikseen käyttäjäobjektin ID:n avulla käyttäjähakemistosta (6 ja 7).

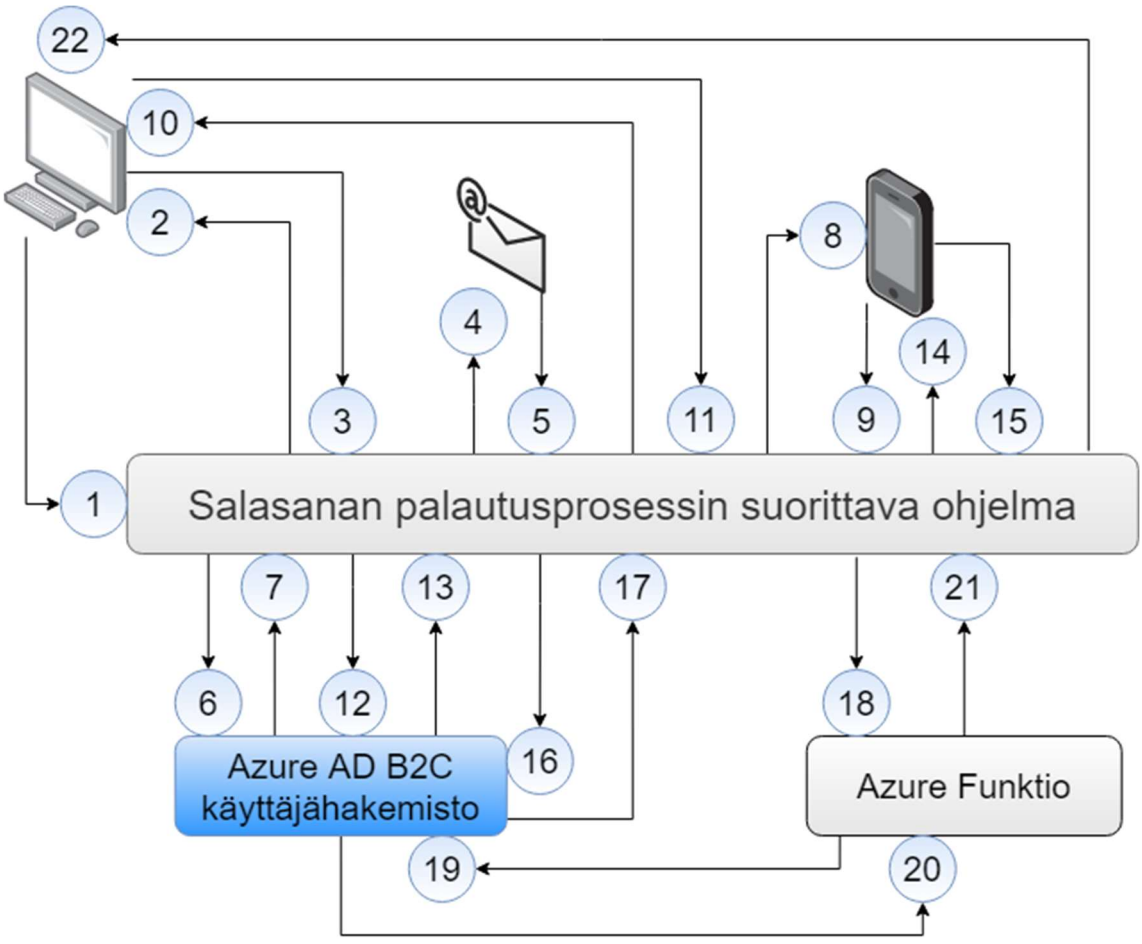
Jos käyttäjällä ei ole monivaiheista tunnistautumista asetettuna, kirjautumisprosessi pyytää käyttäjältä puhelinnumeroa, johon lähetetään tekstiviestillä varmennuskoodi. Muulloin viesti lähetetään jo valmiiksi rekisteröityyn puhelinnumeroon (8). Käyttäjän saadessa puhelimellaan koodin, hänen tulee syöttää saamansa koodi kirjautuakseen sisään (9). Jos uusi puhelinnumero rekisteröitiin käyttäjälle, tallennetaan käyttäjän varmentama puhelinnumero tulevia monivaiheisia tunnistautumisia varten (10 ja 11).

Käyttäjän varmentamisen jälkeen tehdään http-kutsu REST-rajapintaan, josta noudetaan käyttäjän roolit kirjauduttavaan sovellukseen (12). Roolien haku tehdään Microsoft Graph API -ohjelmointirajapinnan avulla (13 ja 14), jonka käyttö esitetään luvussa 9.3.1.

Jos käyttäjällä ei ole ainuttakaan roolia kirjauduttavaan sovellukseen, palautetaan virheilmoitus käyttäjän oikeuksien puutteesta. Tämä toimii ensimmäisenä valtuutustasona, jossa rajataan käyttäjän kirjautumista roolien perusteella. Kun käyttäjältä löytyy rooleja, jotka oikeuttavat käyttämään sovellusta, palautetaan ne kirjautumisprosessille JSON muodossa listana (15). JWT-token muodostetaan kirjautumisessa saatujen tietojen perusteella ja välitetään *Authorization* headerissa palvelulle, jolle kirjautuminen kohdennettiin (16).

5.2.2 Salasanan palautusprosessi

Salasanan palautusprosessissa monet askeleet ovat samoja kuin kirjautumisprosessissa. Salasanan palautusprosessi alkaa kuvan 4 kirjautumisprosessin askeleen 2 jälkeen. Jos käyttäjä kirjautumisen sijaan valitsee linkin salasanan palautusta varten, aloitetaan kuvan 6 mukainen salasanan palautusprosessi.



Kuva 6. *Salasanan palautusprosessin hahmotelma.*

Prosessi alkaa käyttäjän selaimen ilmoittaessa kirjautumisprosessille salasanan palautusprosessin aloituksesta (1). Prosessi ohjaa käyttäjän salasanan palautus sivulle, jossa käyttäjää pyydetään syöttämään hänen käyttäjätunnuksensa sähköpostiosoite (2).

Käyttäjän syöttämä sähköpostiosoite välitetään salasanan palautusprosessille (3). Jos sähköpostiosoitteella löytyy olemassa käyttäjä, sen sähköpostiosoitteeseen lähetetään varmennuskoodi sähköpostilla (4). Käyttäjän pitää syöttää saamansa varmennuskoodi vahvistaakseen sähköpostinsa (5). Sähköpostin varmennuksen jälkeen noudetaan käyttäjätiedon sähköpostiosoitteen avulla (6 ja 7). Turvallisuuden vuoksi käyttäjää pyydetään vahvistamaan puhelinnumeronsa monivaiheisen tunnistautumisen avulla, jos sellainen on käyttäjällä asetettuna (8 ja 9). Käyttäjän tunnistautumisen jälkeen pyydetään käyttäjää syöttämään uusi salasana (10 ja 11), jonka jälkeen se tallennetaan käyttäjälle Azure AD B2C -käyttäjähakemistoon (12 ja 13).

Jos käyttäjällä ei ollut monivaiheista tunnistautumista asetettuna, käyttäjää pyydetään antamaan puhelinnumero ja varmentamaan se (14 ja 15). Uusi varmennettu puhelinnumero tallennetaan käyttäjähakemistoon tulevia monivaiheisia tunnistautumisia varten (16 ja 17). Jos monivaiheinen tunnistautuminen tehtiin jo aiemmin, jätetään askeleet 14–17 tekemättä. Askeleissa 18–22 tehdään täysin samat operaatiot kuin kuvassa 3 esitetyn kirjautumisprosessin askeleissa 12–16. Näiden askeleiden selitykset löytyvät jo kappaleesta 4.2.1.

5.3 Käyttäjien valtuutus sovelluksissa

Microsoftin Azure AD B2C -käyttäjähallinta toteuttaa sen tarjoamassa käyttövalmiissa toteutuksessa vain käyttäjän tunnistautumisen. Valtuutuksen ensimmäinen taso toteutetaan kirjautuessa rajaamalla käyttäjän kirjautuminen sovellukseen, joihin hänellä on oikeuksia. Jos käyttäjällä ei ole oikeuksia sovellukseen, kirjautuminen estetään. Käyttäjien oikeuksien tarkempi valtuutus tehdään sovelluksissa. Kirjautumisen yhteydessä palvelut saavat JSON Web Tokenin, jota välitetään tulevissa kutsuissa valtuutus (Authorization) headerissa. Sovellukset lukevat käyttäjän oikeudet kyseisestä headerista. Työssä päädyttiin hyödyntämään JWT:n sisältöä käyttäjien valtuutuksessa luvun 4.3 tavoin. JWT-tokenin sisällön mukana sovellukset saavat listan rooleja, joiden perusteella sovellukset voivat rajata käyttäjien oikeuksia ennalta määriteltyihin resursseihin. Näiden roolien nimeämisessä käytettyjen nimeämiskäytäntöjen täytyy olla järkeviä, jotta sovelluksissa roolit voidaan tunnistaa systemaattisesti.

6. MICROSOFT AZURE ACTIVE DIRECTORY KÄYTTÄJÄHALLINTANA

Azure Active Directory on Microsoftin hakemistopalvelu, jossa säilytetään ja hallinnoidaan resursseja kuten käyttäjiä, ryhmiä, sovelluksia tai laitteita. Nämä objektit (security principals) on tavallisesti määritelty joko käytettäviksi resursseiksi tai käyttäjiksi, joille voi määrittää oikeuksia. Security principal on yleinen entiteettityyppi, jolle voidaan antaa oikeuksia. Siitä on periytetty kaikki kappaleessa 4.1 esitetyt entiteetit. [14][15]

6.1 Vaihtoehtoiset palvelun toimintamallit

Azure AD B2C (business to consumer) on erillinen Azuren tarjoama palvelu ja B2B (business-to-business) on Azure AD palvelun toiminnallisuus. B2C toimintamalli on tarkoitettu sovellusten paikalliseksi käyttäjähallinnaksi palvelemaan esimerkiksi kuluttajia, jotka voivat rekisteröityä käyttäjiksi hyödyntäen esimerkiksi sosiaalisen median käyttäjätunnuksia.

6.1.1 Azure AD

Azure AD on organisaatiotason pilvikäyttäjähakemisto. Se toimii organisaatioille identiteetin tarjoajana, jota voidaan käyttää kaikissa pilvisovelluksissa. Sen käyttö mahdollistaa organisaatioiden käyttäjien kirjautumisen kaikkiin palveluihin yhden käyttäjätunnuksen avulla. Tunnetuin Azure AD:n palveluista on Office 365, josta pääsee käsiksi moniin erilaisiin sovelluksiin. Nämä tunnukset ovat Azure AD tunnuksia. Microsoftin omien palveluiden lisäksi Azure AD tunnuksia voidaan myös käyttää kolmannen osapuolen sovelluksissa. Tunnistautuminen sovelluksiin on mahdollista tehdä useammalla eri protokollalla kuten SAML, WS-Federation ja OAuth. [16]

Azure AD on suunniteltu laajentamaan organisaatioiden identiteetinhallintaa pilvi- ja SaaS-alustoille ja mahdollistamaan käyttäjien työskentely pilviympäristöissä. Kehittäjät voivat suojata sovelluksensa Azure AD:n avulla. Hakemistopalvelu voidaan kehittää palvelemaan yksittäistä organisaatiota, jolloin se palvelee yksittäistä asiakasta (single tenant), tai yleiseksi sovellukseksi, joka palvelee useata Azure AD:ta käyttävää asiakasta (multi-tenant). [17]

Azure AD on federoitu tuhansien yleisten SaaS-palveluiden kanssa. Tämä tarkoittaa, että Azure AD:n ja palveluiden välillä on jo valmiina luottamus ja näiden palveluiden käyttöönotto ja organisaation käyttäjille oikeuksien välittäminen onnistuu helposti. Vaikka federaatiota ei olisi palvelun ja Azure AD:n välillä, Azure AD mahdollistaa silti kyseiseen palveluun kirjautumisen käyttäen SAML- ja OAuth-protokollia. [16]

Azure AD:n hallinta toimii täysin käyttäen internet protokollia. Azure AD:n hallinta tapahtuu REST-ohjelmointirajapinnan kautta, jota kutsutaan Microsoft Graph API:ksi. Tämä mahdollistaa resurssien ja käyttäjien hallinnan ohjelmallisesti. Microsoftin Graph API:n avulla saatavia tietoja voidaan käyttää uusia sovelluksia luodessa.

6.1.2 Azure AD B2B

Azure AD B2B on Azure AD:n toimintamalli, jonka avulla organisaatiot voivat työskennellä keskenään. B2B toiminnallisuutta käyttäessään Azuren pilvialusta luo erillisen Azure AD hakemiston B2B:n kautta jaettaville resursseille. Tämä hakemisto ei kuitenkaan ole tavallinen Azure AD hakemisto. Se sisältää vain jaettavat resurssit ja tiedon käyttäjistä, joille resurssi on jaettu. Kun käyttäjä haluaa päästä käsiksi resurssiin, Azure AD B2B hakemisto osaa ohjata käyttäjän kirjautumisen hänen kotihakemistoonsa. [16]

Organisaatiot työskentelevät usein yhdessä muiden organisaatioiden jäsenten kanssa. Ilman Azure AD B2B:tä (business-to-business) täytyisi organisaation ulkopuoliset lisätä Azure AD hakemistoon omina käyttäjinään. Azure AD B2B:n avulla voidaan kutsua organisaation ulkopuolisia käyttäjiä jakamaan resursseja. Azure AD B2B mahdollistaa organisaatioiden välisen yhteistyön olemassa olevien käyttäjätilien avulla ilman että heidän täytyy kutsua ulkopuolisia heidän oman organisaationsa jäseniksi.

6.1.3 Azure AD B2C

Azure AD B2C -palvelu (business to consumer) on täysin erillinen käyttäjähakemisto, jolla on erilaiset tavoitteet kuin Azure AD:lla. Sitä käytetään useimmiten yksittäisten palveluiden kuluttajien omana käyttäjähakemistona. B2C:ssä voidaan antaa käyttäjän kirjautua käyttäen esimerkiksi sosiaalisen median, kuten Facebookin tai Googlen, käyttäjiä. Erillistä identiteetin tarjoajaa käyttäessä hakemistoon tallennetaan vain tärkeimmät tiedot käyttäjistä, joita sovellus tarvitsee. Käyttäjien hallinta tapahtuu tällöin ulkoisen identiteetin tarjoajan puolella, eikä käyttäjätilien hallintaa tarvitse toteuttaa sovelluksen puolella. Azure AD B2C kuitenkin mahdollistaa myös paikallisten käyttäjien käytön. Tällöin käyttäjien hallinnointi voidaan tehdä rajapintakutsujen kautta. Kirjautumista voidaan

myös vahvistaa lisäämällä esimerkiksi monivaiheinen todennus. Azure AD B2C on tarkoitettu sovelluksia varten, joihin kuluttajat voivat itse rekisteröityä. Tavallisimmissa käyttötapauksissa ei ole tarkoituksena pystyä rajaamaan käyttäjien oikeuksia tietyille resursseille. [16]

Azure AD B2C on teknisesti Azure AD:n pohjalta rakennettu ja jakaa kaikki käyttäjähallintaan liittyvät entiteettityypit sen kanssa. Azure AD B2C on toiminnallisuuksien kannalta karsittu Azure AD. Saatavilla olevien entiteettityyppien ja mukautettujen käytäntöjen avulla voidaan muokata käyttäjähallinta toimimaan halutulla tavalla.

6.2 Käyttäjäprosessit ja mukautetut käytännöt

Mukautetut käytännöt (Custom Policy) ovat XML kielellä kirjoitettuja konfiguraatiotiedostoja, joiden avulla määritetään Azure AD B2C -käyttäjähakemiston toiminnallisuudet. Mukautettujen käytäntöjen avulla voidaan Azure AD B2C:n toiminnallisuuksia muokata vastaamaan mitä tahansa käyttötarvetta. Microsoft on kehittänyt useita valmiita pohjia mukautetuille käytännöille, joita voi käyttää pohjana räätälöidyille ratkaisuille. Niitä muokkaamalla voidaan esimerkiksi kirjautumisprosessiin lisätä ylimääräisiä askelia tai luoda täysin uusia prosesseja. Mukautetuissa käytännöissä määritetään käyttäjäprosessien suorittava logiikka. Niiden avulla voi suorittaa hyvin monimutkaisia prosesseja ja tarvittaessa prosesseihin voidaan lisätä REST-rajapintakutsuja, joiden avulla logiikkaa voidaan ulkoistaa. [18] [19]

Azure AD B2C:stä löytyy useampia valmiita käyttäjäprosesseja (user flow), joiden käyttöä Microsoft suosittelee ensisijaisesti. Valmiit prosessit ovat pohjimmiltaan konfiguroitu samalla tavalla kuin mukautetut käytännöt, mutta niitä ei voi muokata muuten kuin käyttöliittymän kautta rajoitetusti. Valmiita prosesseja löytyy kirjautumiselle, profiilin muokkaukselle, salasanan uusimiselle ja rekisteröitymiselle. Turvallisuutta voidaan lisätä prosesseihin monivaiheisen tunnistautumisen avulla. Valmiilla kirjautumisprosessilla ei kuitenkaan ole mahdollista rajata käyttäjien pääsyä käyttäjähallintaan rekisteröityihin sovelluksiin. [18]

Mukautetuiden käytäntöjen konfiguraatiossa käytetään termiä väite (claim), jotka toimivat konfiguraatioissa muuttujina. Väitteille voidaan määritellä parametrityyppi ja niille annettaville arvoille rajoitteita. Väitteet säilytetään koko käyttäjäprosessin ajan ja niitä voidaan käyttää käyttäjäprosessin päätteeksi JSON Web Tokenin muodostamisessa.

Mukautetuissa käytännöissä määritetään käyttäjäprosessien suorittama käyttäjän matka (user journey). Siinä määritetään logiikka, jolla voidaan suorittaa esimerkiksi käyttäjän

kirjautuminen. Käyttäjän matka koostuu sarjasta orkestraatioaskeleita (orchestration step), joissa määritetään käyttäjäprosessin vaiheet askel kerrallaan. Mukautettujen käytäntöjen avulla askeleita voi muokata, lisätä tai poistaa tarpeen mukaan. Askeleissa voidaan määritellä yksinkertaista logiikkaa, kutsua Microsoftin Azure AD B2C valmiita toimintoja tai ulkoistaa logiikkaa REST-rajapintakutsujen avulla. Askelille voidaan määrittää syötteinä käytettävät väitteet ja tallentaa uusia väitteitä väite säiliöön. [18]

Käyttäjän matkan aikana käyttäjä voidaan ohjata useammalle sivulle, joilla käyttäjä suorittaa esimerkiksi kirjautumisen, monivaiheisen tunnistautumisen tai salasanan palautuksen. Näille sivuille ohjaaminen konfiguroidaan mukautetuissa käytännöissä. Sivustoilla voidaan käyttää Microsoftin valmiita pohjia tai luoda omat räätälöidyt HTML-sivut, joihin Azure AD B2C injektoi konfiguraation mukaisen sisällön. Räätälöidyn HTML-sivun avulla voidaan itse määrittää sisällön tyyli, joka vastaa omaa palvelun teemaa.

6.3 Moniasiakkuus Azure AD:n kontekstissa

Kun puhutaan Azuren käyttämillä termeillä, organisaation AD-hakemistoa kutsutaan asiakkaaksi. Sovelluksen palvellessa useampaa asiakasta, se tarjoaa sovelluksensa usealle organisaatiolle, jolloin yksittäinen sovellus palvelee useampaa Azure AD -asiakasta.

Tämä on päinvastainen ajatusmalli siitä, miten moniasiakkuutta ajatellaan ADE:n näkökulmasta. Sen sijaan että sovellus eli asiakasympäristö palvelee useampaa AD-asiakasta, halutaan yksittäisen AD B2C -palvelun palvelevan useampaa sovellusta. Työssä tavoitellussa toteutuksessa käyttäjähakemistoja tulee olemaan vain yksi, joka palvelee useampaa sovellusta.

6.4 Azure AD B2C:n rajoitteet

Azure AD B2C ja Azure AD omaavat identtiset tietokantamallit ja rajapinnat. Tämän seurauksena kaikki entiteettityypit, jotka esiintyvät Azure AD:ssa, löytyvät myös Azure AD B2C:stä. Monilla entiteettityypeillä ei kuitenkaan ole mitään toiminnallisuuksia Azure AD B2C:ta käyttäessä. Joidenkin resurssien käyttö on jopa estetty Azuren graafisessa käyttöliittymässä. Näitä estoja voidaan kuitenkin kiertää Microsoft Graph API -ohjelmointirajapinnan avulla.

Entiteettityypeistä muun muassa käyttäjäoikeusryhmät (Security Group) ovat vailla minkään näköistä toiminnallisuutta. Käyttäjiä voidaan lisätä ryhmiin, mutta Azure AD B2C:ssä ei ole Microsoftin tarjoamassa toiminnallisuudessa mitään vaikutusta käyttäjähallinnan toimintaan. Jotta Azure AD B2C:ssä voidaan hyödyntää kyseisiä ryhmiä, täytyy

niiden käsittelylogiikka toteuttaa mukautettujen käytäntöjen ja ulkoisen logiikan avulla. [20]

Ohjelmallinen Azure AD B2C -hakemiston resurssien käsittely on täysin riippuvainen Microsoftin luoman ohjelmointi rajapinnan toteutuksesta. Joissakin tapauksissa ulkoista logiikkaa kirjoittaessa joudutaan tekemään outoja ratkaisuja, koska Microsoftin kirjoittamassa ohjelmointirajapinnassa voidaan hakea resursseja vain tietyillä arvoilla, kuten kappaleessa 8.2.3 tullaan huomaamaan. Näissä tapauksissa voidaan joutua hakemaan halutut resurssit useamman rajapintakutsun avulla.

7. AZURE FUNKTIOT

Azure Funktiot ovat Microsoftin palvelittoman (serverless) tietojenkäsittelyn alusta. Se mahdollistaa skaalautuvien funktioiden luomisen, joista maksetaan ainoastaan käytön mukaan. Funktioissa käytettäviä vaihtoehtoisia kieliä ovat C#, JavaScript, F#, Java, Python, TypeScript, Batch, Bash, PowerShell ja muutamia muita kokeellisia kieliä. [21]

7.1 Palveliton tietojenkäsittely

Palveliton tietojenkäsittely luo useasti kuvan palvelusta, joka ei käytä palvelimia ollenkaan. Tämä ei kuitenkaan ole totta. Palveliton tietojenkäsittely mahdollistaa sovellusten kehityksen ilman, että sen kehittäjien tarvitsee hallinnoida palvelin infrastruktuuria. Palvelittomia tietokonealustoja on esimerkiksi AWS Lambda, Azure Functions ja Google Cloud Functions. [22]

Palveliton tietojenkäsittely tunnetaan myös FaaS-palvelumuotona (function as a service). Siinä kehittäjät joutuvat kehittämään vain lyhyitä koodia suorittavia funktioita ja määrittämään niiden suoritusehdot. FaaS -palvelun tarjoava alusta suorittaa koodin sen suoritusehdon täytyessä ja laskuttaa vain koodin suoritukseen käytetyn ajan perusteella. Palvelimet skaalautuvat automaattisesti tarpeen mukaan. [22]

7.2 Funktioiden vaihtoehtoiset liipaisimet

Azure Funktioissa on neljä erilaista liipaisinta (trigger), joiden perusteella aloitetaan funktion suoritus. Jono (Queue) -liipaisimella funktio lukee arvoja jonosta ja suorittaa niiden avulla logiikkansa. Funktio käynnistyy tällöin aina kun uusi arvo tulee jonoon ja käyttää sitä funktiolle syötteenä. Ajastin (Timer) -liipaisimella funktio suoritetaan aikataulun mukaan tietyn ajan välein. Tapahtumaruudukossa (Event Grid) funktio toimii tilaajana (subscribe) jollekin resurssille. Funktio suoritetaan resurssiin tullessa jokin muutos. Http-kutsu-liipaisimella määritellään http-rajapinta, jota kutsuessa funktio suoritetaan. Funktiolle voidaan määrittää parametreja http-kutsussa. [23] [24]

8. MICROSOFT GRAPH API

Graph API on Microsoftin REST-rajapinta, joka mahdollistaa Office 365 -palveluiden ohjelmallisen käytön. Office 365 -tarjontaan lukeutuu muun muassa Azure Active Directory. Rajapinnan käytön vaatimuksena on ohjelman kyky lähettää http-kutsuja ja käsitellä kutsujen sisältöä JSON muodossa. [25]

8.1 Tunnistautuminen

Rajapinnan käyttämistä varten pitää tunnistautua Graph-palveluun. Työssä käytettäviä rajapintakutsuja varten tunnistautuminen ja valtuuttaminen voidaan tehdä käyttäen Client Credentials -tunnuksia. Client Credentials -tunnistautuminen on OAuth-protokollan tapa palveluille tunnistautua keskenään. Client Credentials -tunnuksia varten täytyy Azure AD B2C:hen rekisteröidä sovellus, määrittää sille salainen avain ja asettaa käyttöoikeudet Graph API:n käyttöä varten. Sovelluksen rekisteröiminen käydään tarkemmin läpi luvussa 7.2. [26]

Työssä käytettyjä rajapintakutsuja varten täytyy Graph Api:a käyttävälle sovellukselle lisätä oikeudet *Application.ReadWrite.All*, *Group.ReadWrite.All, Directory.ReadWrite.All* ja *User.ReadWrite.All*. Nämä mahdollistavat rekisteröityjen sovellusten, käyttöoikeusryhmien ja käyttäjien hallinnan *[27]-[29]*

8.2 Sovellusten rekisteröiminen

Luvussa 5.1 käytiin läpi entiteettejä, joita työssä hyödynnetään. Näistä yksi entiteettityyppi on rekisteröidyt sovellukset. Työssä käytetään kahden tyyppisiä rekisteröityjä sovelluksia: asiakkaiden ympäristöjä vastaavat sovellukset ja käyttäjähakemistoa hallinnoivat sovellukset. Jotta asiakasympäristöt voivat käyttää Azuren kirjautumista, täytyy niiden rekisteröityä Azure AD B2C:n sovelluksiksi. Kirjautuminen vaatii käyttäjien lukuoikeuksia, jotka annetaan jokaiselle sovellukselle oletuksena. Ohjelmallista käyttäjähakemiston hallinnointia varten luodaan oma sovellus, jolle annetaan laajemmat käyttöoikeudet. Hallinnointiin tarkoitettuihin rekisteröityihin sovelluksiin käyttäjät eivät suoraan kirjaudu sisään. Niiden käyttö tapahtuu erillisten ohjelmien kautta, joita käyttäjät voivat kutsua.

Sovelluksen rekisteröimistä varten täytyy hakemistoon luoda kaksi objektia. Jokaisella sovelluksella on objekti, joka on nimeltään *application* ja jokaiselle *application* -objektille

pitää luoda palvelutason objekti *servicePrincipal. Application* -objekti sisältää tietoa rekisteröidystä sovelluksesta, kuten uudelleenohjaus URI, johon JWT lähetetään kirjautumisen jälkeen. *ServicePrincipal* -objektille määritetään sovelluksen oikeudet käyttäjähakemiston sisäisesti ja sille voidaan määrittää relaatioita käyttäjäoikeusryhmiin. [30]

Sovellusten kanssa työskennellessä täytyy olla hyvin tarkkana eri objektien tunnisteiden kanssa. *Application* -objektissa on kaksi eri tunnistetta: *id* ja *appId*. Näistä *id*:tä käytetään yleisenä tunnisteena objektille, jonka avulla esimerkiksi sovellusten tietoja voidaan hakea Graph API ohjelmointirajapinnan kautta. *AppId* toisaalta vastaa tunnistetta, jota käytetään kirjautumisen yhteydessä. Sovelluksen palvelutason objektista löytyy myös viitta*us appId* -tunnisteeseen. Sovelluksen palvelutason objektien tietoja hakiessa käytetään niiden omia *id* -tunnisteita. Esimerkiksi myöhemmin kappaleessa 9.3.1. Azure Funktion toiminnallisuutta läpikäydessä *appRoleAssignments* haun kautta saadaan lista rooleja, jotka viittaavat sovellusten palvelutason objekteihin. Sovelluksen *appId* -tunnistetta ei pysty rajapinnan kautta suoraan hakemaan, vaan se täytyy hakea sovellusten palvelutason objektien avulla.

8.3 Käyttäjien hallinta

Työn toteutuksessa ei haluta kuluttajien pystyvän itsekseen rekisteröityä käyttäjähallinnan käyttäjäksi. Käyttäjät tullaan luomaan ohjelmallisesti, jotta voidaan paremmin rajata ulkopuolisten käyttäjien rekisteröimistä järjestelmään. Käyttäjiä luova järjestelmä tullaan kuitenkin luomaan työn jatkokehityksenä.

Käyttäjähallinnan käyttäjät voidaan luoda Microsoft Graph API ohjelmointirajapinnan avulla. Käyttäjää luodessa salasanaksi täytyy itse generoida jokin arvo. Käyttäjän luonnissa ei ole vaihtoehtoa satunnaisen salasanan luomiselle. Salasanaksi halutaan jokin satunnainen arvo, jota ei tallenneta minnekään muualle kuin Azure AD B2C:n tietokantaan. Tällöin kenenkään ei pitäisi päästä käyttäjään käsiksi ennen kuin käyttäjä on vaihtanut salasanan kappaleessa 9.4 esitettävän salasanan palautusprosessin avulla.

Käyttäjien tietojen hallinta onnistuu Microsoftin Graph API ohjelmointirajapinnan avulla. Oikeiden parametrien löytäminen haluttuun tarpeeseen voi vaatia selvittämistä, mutta Graph API:a käyttäen pystyy tekemään kaikki muutokset, mitä käyttäjille täytyy tehdä.

8.4 Käyttöoikeusryhmien hallinta

Käyttöoikeusryhmät (security group) ovat jäänne Microsoftin aiemmasta paikallisesta Windows Active Directory palvelusta, joiden avulla voidaan rajata pääsyä hakemiston

objekteihin. Niitä ei käytetä paljoa Azure AD:ssa ja niillä ei ole mitään toiminnallisuutta Azure AD B2C:ssä. Ne ovat kuitenkin olennaisena osana työssä toteutettavassa roolipohjaisessa Azure AD B2C käyttäjähallinnassa. Työssä luotavien mukautettujen käytäntöjen määrittämän logiikan avulla käyttöoikeusryhmiä voidaan ajatella käyttäjien rooleina. [31]

Azure AD B2C:ssa käyttöoikeusryhmiä voidaan ajatella geneerisinä ryhminä, joihin voidaan lisätä käyttäjiä ja joita voidaan asettaa sovellusten jäseniksi. Käyttöoikeusryhmän nimi voi olla esimerkiksi *asennus123:sovellus-admin*. Käyttöoikeusryhmien nimeämistä käydään läpi kappaleessa 9.1. Käyttöoikeusryhmien jäseniksi voidaan asettaa mitä tahansa Azure AD B2C -hakemiston objekteja. Oikeuksien hallintaa varten käyttöoikeusryhmät täytyy lisätä kappaleessa *8.2* rekisteröityjen sovellusten palvelutason objektien jäseniksi.

9. ROOLIPOHJAISEN KÄYTTÄJÄHALLINNAN TOTEUTTAMINEN

Tässä luvussa käydään läpi teknistä toteutusta, miten yksittäisen Azuren AD B2C hakemiston avulla voidaan toteuttaa käyttäjähallinta useammalle sovellukselle. Toteutuksessa käytetään vain yksittäistä käyttäjähakemistoa, joka hallitsee kaikkia palvelun käyttäjiä.

9.1 Roolien määrittäminen

Kappaleessa 2.3 esitettyjen vaatimusten mukaan käyttäjähallinnan pitää pystyä määrittämään käyttäjille eri tasoisia oikeuksia, joiden avulla voidaan rajata käyttäjien pääsyä tiettyihin resursseihin ja toimintoihin. Työssä päädyttiin toteuttamaan käyttäjien oikeuksien jako käyttäen RBAC-periaatteella (Role Based Access Control). Roolipohjaisessa käyttäjähallinnassa voidaan luoda käyttöoikeuksiltaan eritasoisia rooleja.

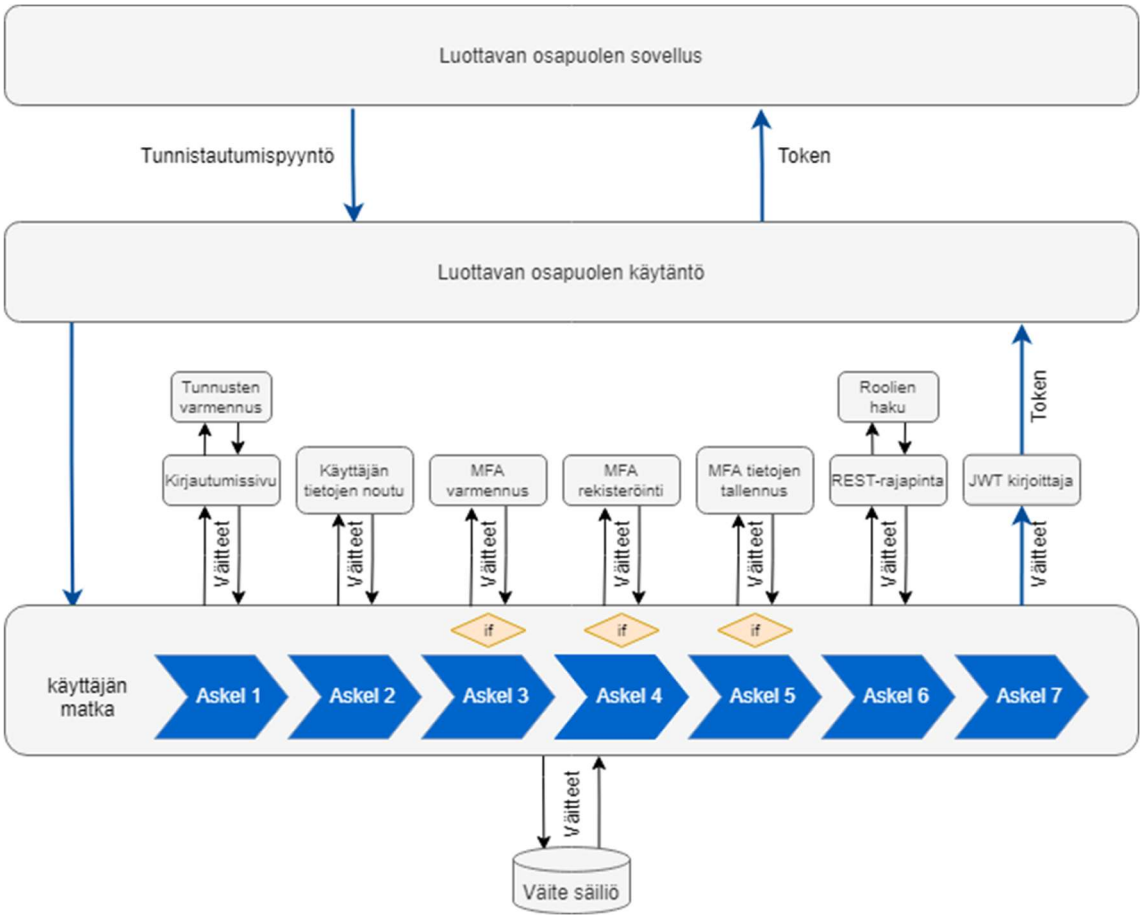
Azure AD B2C hakemistossa käyttöoikeusryhmiin voidaan lisätä käyttäjiä jäseniksi. Tässä toteutuksessa yksittäistä käyttöoikeusryhmää voidaan ajatella yksittäisenä roolina. Kun käyttäjälle annetaan jokin oikeus, hänet lisätään kyseiseen käyttöoikeusryhmään. Azure AD B2C ei tule hallitsemaan konkreettisia resursseja, joihin roolit antavat käyttöoikeuden. Resurssien oikeuksien hallinta toteutetaan sovelluksissa roolien perusteella.

Käyttöoikeusryhmien nimet ovat uniikkeja yksittäisen Azure AD B2C hakemiston sisällä. Tämän rajoituksen avulla sovelluksissa voidaan käsitellä käyttöoikeuksia käyttäen niiden nimiä sen UUID (Universally Unique Identifier) muotoisten tunnisteiden sijaan. Nimien käyttö mahdollistaa systemaattisen nimeämiskäytännön roolien tunnistamisen helpottamiseksi. Työssä on aiemmin esitelty nimeämismuoto *asennus123:sovellus-admin.* Siinä ”asennus123” viittaa asiakasympäristön nimeen, ”sovellus” asiakasympäristön sisäisen sovelluksen nimeen ja ”admin” viittaa oikeustasoon, joka käyttäjälle annetaan.

9.2 Kirjautumisprosessi

Tässä kappaleessa käydään läpi työtä varten räätälöity kirjautumisprosessi. Prosessi kehitettiin hyödyntämällä Microsoftin tarjoamaa mallipohjaa ja karsimalla siitä ylimääräi-

set toiminnallisuudet, kuten ulkoisten identiteetin tarjoajien tuki. Prosessiin lisättiin monivaiheinen tunnistautuminen ja roolien haku REST-rajapinnan kautta. Prosessin askeleet on esitetty kuvassa 7.



Kuva 7. *Kirjautumisprosessin määrittelevä mukautettu käytäntö.*

Kuvassa 7 esitetty luottavan osapuolen sovellus vastaa sovellusta, joka tekee Azure AD B2C:lle kirjautumispyynnön. Sovellukselle on kerrottu käytäntö, jota se voi kutsua kirjautumista varten. Tässä tapauksessa sitä käytäntöä vastaa kuvassa luotettavan osapuolen käytäntö.

Kirjautumisprosessissa suoritettava käyttäjän askeleella 1, jossa käytettiin Azure AD B2C:n tarjoamaa kirjautumispalvelua. Kirjautumissivun sisältö määrittyy käytännön konfiguraatioiden avulla. Kirjautumispalvelu ottaa syötteenä käyttäjältä hänen käyttäjätunnuksensa ja salasanansa. Käyttäjä varmennetaan Azure AD B2C:n OAuth-rajapintoja

hyödyntäen, jonka päätteeksi tallennetaan kirjaudutun käyttäjän tunniste väitteisiin. Askeleessa 2 voidaan tunnisteen avulla noutaa käyttäjän kirjautumiseen tarvittavat tiedot käyttäjähakemistosta.

Käyttäjän varmempaa tunnistautumista varten käytetään monivaiheista tunnistautumista. Monivaiheinen tunnistautuminen toteutettiin käyttäen SMS-tunnistautumista. Käyttäjän puhelinnumeron varmennus tehdään askeleessa 3, jos monivaiheinen tunnistautuminen on käyttäjällä aktiivisena. Askeleet 4 ja 5 suoritetaan, jos käyttäjällä ei ole puhelinnumeroa rekisteröitynä monivaiheista tunnistautumista varten. Askeleessa 4 käyttäjä rekisteröi uuden puhelinnumeron ja askeleessa 5 puhelinnumero tallennetaan käyttäjän tietoihin. Käyttäjän täytyy varmentaa puhelinnumero rekisteröinnin ja jokaisen kirjautumisen yhteydessä. Varmennus tapahtuu lähettämällä 6-numeroinen koodi käyttäjän puhelinnumeroon ja käyttäjän pitää syöttää saamansa koodi käyttäjäprosessin aikana.

Askeleessa 6 toteutetaan käyttäjän roolien haku, jonka logiikka ulkoistettiin erillisen REST-rajapinnan taakse. Askeleessa määritetään REST-kutsu, jonka odotetaan palauttavan listan käyttäjän rooleista. Jos käyttäjällä ei ole rooleja kyseiseen sovellukseen, rajapinta palauttaa virheen ja keskeytetään käyttäjän kirjautuminen.

Lopuksi askeleessa 7 välitetään aiemmissa askeleista saadut väitteet JWT:n kirjoittajalla, joka allekirjoittaa JWT:n. JWT välitetään luottavan osapuolen sovellukselle uudelleenohjausosoitteeseen, joka on määritelty sovelluksen rekisteröinnin yhteydessä.

Kirjautumisprosessissa on osittain toteutettu OAuth-protokollaa. Askeleessa 1 tehdyssä tunnusten vahvistamisessa käytetään OAuth-tunnistautumista.

9.3 Oikeutettujen roolien haku kirjautuessa

Tavallisissa Azure AD B2C:n käyttötapauksissa kirjautuminen voidaan tehdä hyödyntäen valmiita käyttäjäprosesseja. Käyttäjän kirjautuessa valmiilla käyttäjäprosessilla, hänelle suoritetaan vain tunnistautuminen. Käyttäjän kirjautumista ei tällöin voi rajata tietylle sovellukselle. Azure AD B2C:n tunnistautuminen perustuu OAuth-protokollaan.

Työssä päädyttiin toteuttamaan roolien hallinta lisäämällä tiedot käyttäjän rooleista JSON Web Tokenin sisältöön luvussa 4.3 esitetyllä tavalla. JWT:hen lisättävät roolit sisältävät ainoastaan roolien nimet, joihin hänellä on oikeus ja jotka liittyvät kirjauduttuun sovellukseen. Samalla JWT:llä käyttäjä ei voi käyttää toista rekisteröityä sovellusta,

vaikka hänellä olisi siihen oikeudet. Vaihtoehtoinen ratkaisu olisi sovellusten hakea käyttäjän oikeudet Azure AD B2C:sta Microsoft Graph API:n kautta sovelluksista käsin. Tämä vaihtoehto kuitenkin hylättiin, koska se vaatisi paljon enemmän liikennettä sovellusten ja Azuren välillä.

9.3.1 Roolien haussa käytetty Azure Funktio

Roolienhakua varten tarvitaan REST-rajapinta, joka on aina käytettävissä. Ulkoistamalla Graph API -kutsut erilliseen ohjelmaan, logiikan kehitys, luettavuus ja ylläpito helpottuu huomattavasti. Kustannussyistä päädyttiin palvelittomaan toteutukseen käyttämällä Azure funktioita.

Funktion kieleksi valittiin C#, koska se on Microsoftin kehittämä kieli ja takaa varman yhteensopivuuden. Liipaisimeksi valittiin http-kutsu, koska kyseessä on REST-rajapinta. C#-koodissa on käytetty Microsoftin luomaa GraphServiceClient -kirjastoa, joka hoitaa kutsujen muodostamisen. Valmiin kirjaston käyttö helpottaa Graph API:n kutsujen tekoa, kun ei tarvitse käsin kirjoittaa http kutsuja.

Käyttäjien roolit lisätään osaksi JSON Web Tokenin sisältöä. Koska JWT:n sisältö on JSON-muodossa, halutaan myös REST-rajapinnan palauttavan listan rooleja JSON-muodossa. REST-rajapinta saa parametreina käyttäjän tunnisteen ja sovelluksen *appId* -tunnisteen. Näiden tietojen avulla rajapinnan pitää pystyä hakemaan käyttäjän roolit Graph API:n avulla ja rajaamaan käyttäjän roolit sovelluksen perusteella. Ohjelmassa 2 on esitetty muoto, jossa roolit tulee palauttaa REST-rajapinnasta.

```
{
        "roles": [
                "asennus123:sovellus-admin",
                "asennus123:sovellus-user"
        ]
}
```

Ohjelma 2. *Esimerkki roolien esitysmuodosta, jossa REST-rajapinnan tulee palauttaa roolit mukautetulle käytännölle.*

Jotta kirjautuessa pystytään yhdistämään rajapintaan, tulee sen olla julkisessa internet-osoitteessa. Rajapinta käyttää palvelintunnuksia (client credentials) Graph API:in, jonka takia rajapinta pitää suojata hyvin.

Azure funktion suojauksessa on useita vaihtoehtoisia valtuutustasoja: *anonymous*, *function* ja *admin.* Valtuutustason ollessa *anonymous,* funktion käyttö ei vaadi mitään avaimia. *Function* valtuutustaso vaatii funktiolle määritetyn salaisen avaimen, jonka voi noutaa esimerkiksi Azuren graafisen käyttöliittymän kautta. Funktiolle voi määrittää oman avaimen tai käyttää oletusavainta, joka generoituu automaattisesti funktioille. *Admin*-valtuutustaso avain vaatisi pääavaimen, joka olisi yhteinen useammalle funktiolle. Näistä valtuutustasoista päädyttiin funktio valtuutukseen, koska työssä käytettäviä funktioita oli vain yksi. Jotta funktiota voi käyttää *function*-valtuutustasolla, täytyy funktioon tehdyssä kutsussa välittää salainen avain headerissa nimeltä *x-functions-key.* [32] [33]

```
1  function http-liipaisin(sovellus_app_id, käyttäjä_id):
2     sovellus_rooli_määritykset = hae_sovellus_rooli_määritykset(käyttäjä_id)
3
4     palvelutason_idt = []
5     for each (sovellus_rooli in sovellus_rooli_määritykset):
6        if (sovellus_rooli.sovellustyyppi == "Group"):
7           palvelutason_idt.add(sovellus_rooli.resurssi_id)
8
9     sovelluksen_palvelutason_objekti = null
10    if (palvelutason_idt != []):
11       palvelutason_objektit = hae_palvelutason_objektit(palvelutason_idt)
12       for each (objekti in palvelutason_objektit):
13          if (objekti.app_id == sovellus_app_id):
14             sovelluksen_palvelutason_objekti = objekti
15
16    roolit = []
17
18    if (sovelluksen_palvelutason_objekti == null):
19       return virhe(401)
20    else:
21       for each (sovellus_rooli in sovellus_rooli_määritykset):
22          if (sovellus_rooli.sovellustyyppi == "Group"
23             & sovellus_rooli.resurssi_id == sovelluksen_palvelutason_objekti.id):
24             roolit.add(sovellus_rooli.nimi)
25       return roolit
```

Ohjelma 3. *Azure funktion toiminta pseudokoodina.*

Roolien hakuun käytettävän funktion logiikka on esitetty ohjelmassa 6 pseudokoodina. Funktio saa parametreina sovelluksen *appId*-tunnuksen ja kirjautuvan käyttäjän tunnisteen. Kuten kappaleessa 8.2 käytiin läpi, rekisteröidyillä sovelluksilla on useampi tunnus eri tarkoituksia varten. Tässä vaiheessa käytettävissä on sovelluksen tunnus, jonka avulla ei voida suoraan hakea sovellusta Microsoft Graph API:n ohjelmointirajapinnasta.

Sovellusten haut toimivat käyttäen sovellusten toista tunnistetta, jota ei tiedetä rajapintaa kutsuessa. Tämän takia käyttäjän oikeus kyseiseen sovellukseen täytyy hakea käyttäjälle kuuluvien roolien avulla.

Ohjelman 3 rivillä 2 haetaan kaikki käyttäjän sovellusten roolien määritykset. Microsoftin Graph API rajapinta tarjoaa *AppRoleAssignment*-haun, joka hakee kaikki roolit, jotka ovat lisätty johonkin sovellukseen. Näistä objekteista selviää sovelluksen palvelutason tunniste, mutta ei sovelluksen appId:tä. Riveillä 4-7 kerätään saaduista rooleista resurssi tunnisteet, jotka vastaavat palvelutason objektien tunnisteita, ja rajataan ainoastaan ne määritykset, joiden tyyppi on Group. Näin varmistamme, että kaikki käsittelemämme määritykset ovat Security Group tyyppiä.

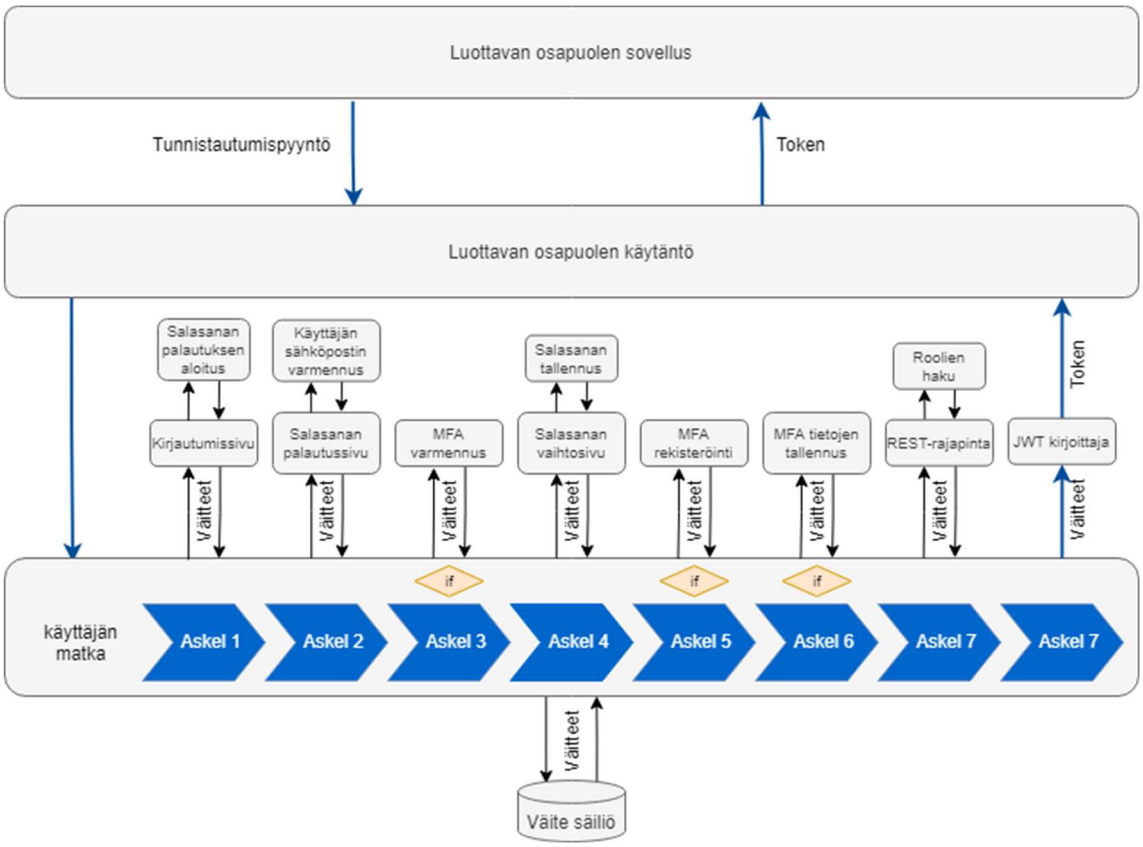
Kuten kappaleessa 8.2 käytiin läpi, sovelluksilla on kuitenkin kahden eri tason objekteja: Palvelutason (Service Principal) ja sovellustason (Application). Ohjelman 3 riveillä 10 ja 11 haetaan aiemmassa vaiheessa haettujen tunnisteiden avulla sovellusten palvelutason objektit, joista päästään käsiksi sovellusten *appId*-tunnisteisiin. Jos käyttäjältä ei löytynyt ainuttakaan sovelluksen palvelutason objektin tunnistetta, jätetään tämä tekemättä. Haku voidaan tehdä Microsoft Graph Apin *DirectoryObject* -haun avulla, jossa haetaan geneerisiä hakemiston objekteja, jotka toimivat monelle entiteetille pohjana. Haku mahdollistaa useamman objektin haun samaan aikaan ottamalla parametrina listan tunnisteita. Riveillä 12-14 käydään läpi kaikki saadut objektit ja varmistetaan löytyykö joltakin niistä funktiolle parametrina annettua sovelluksen appId-tunnistetta. Jos appId löydettiin, otetaan sovelluksen palvelutason objekti talteen. Jos palvelutason objektia ei löytynyt, palautetaan rivillä 19 virhe. Tämä virheen palautus toimii kirjautumisessa ensimmäisenä valtuutustasona, jossa käyttäjän kirjautuminen estetään hänellä ollessa puutteelliset oikeudet.

Riveillä 21-25 käydään uudestaan läpi käyttäjien sovellusten roolien määritykset ja kerätään listaan ne roolit, jotka liittyvät aiemmassa vaiheessa löydettyyn sovelluksen palvelutason objektiin. Nämä roolit palautetaan funktiosta JSON-muodossa, jotta ne voidaan kirjautumisprosessin päätteeksi lisätä suoraan JSON Web tokeniin.

9.4 Käyttäjän salasanan palautusprosessi

Käyttäjän salasanan palautusprosessi toteutettiin muuttamalla muutamaa askelta kirjautumisprosessista. Salasanan palautusprosessi aloitetaan kirjautumissivulta, jossa käyttäjällä on vaihtoehtona kirjautua sisään tai palauttaa unohtunut salasana. Toteutuksessa

päädyttiin ratkaisuun, jossa salasanan vaihto tapahtuu osana käyttäjän kirjautumisprosessia. Tämä tarkoittaa sitä, että käyttäjän uusiessa salasanan, hän suoraan kirjautuu sisään palveluun. Toteutuksessa ei koettu tarpeelliseksi ohjata käyttäjää takaisin kirjautumisprosessin alkuun. Salasanan palautusprosessi on esitetty kuvassa 8.



Kuva 8. *Salasanan palautusprosessin määrittämä käytäntö.*

Salasanan palautusprosessi varsinaisesti alkaa askeleessa 2, jolloin käyttäjä ohjataan salasanan palautussivulle. Käyttäjää pyydetään syöttämään käyttäjätunnuksensa sähköpostiosoitteen. Se hakee Azure B2C -hakemistosta käyttäjän sähköpostin perusteella. Käyttäjälle lähetetään sähköpostilla koodi, joka hänen tulee syöttää edetäkseen salasanan palautuksessa. Sähköpostin varmennuksen yhteydessä haetaan käyttäjän kirjautumiseen tarvittavat tiedot.

Monivaiheinen tunnistautuminen lisättiin askeleessa 3 osaksi salasanan palauttamista, koska salasanan palautusprosessi päättyy käyttäjän sisään kirjautumiseen. Sillä saadaan myös lisättyä ylimääräinen turvallisuuskerros ennen salasanan vaihtoa. Jos moni-

vaiheinen tunnistautuminen on käyttäjällä aktivoituneena, kukaan ei voi ilman hänen puhelintaan muuttaa salasanaa. Monivaiheisen tunnistautumisen jälkeen askeleessa 4 ohjataan käyttäjä salasanan palautussivulle, jossa hän voi syöttää uuden salasanan.

Loput askeleista vastaavat luvussa 9.2 läpikäytyä kirjautumisprosessia. Salasanan palautusprosessi on myös tärkeässä osassa käyttäjien rekisteröimistä. Käyttäjillä ei ole mahdollisuutta rekisteröidä itseään käyttäjiksi. Palveluiden ylläpitäjät lisäävät käyttäjät palveluun ja he saavat käyttäjätunnustensa salasanat salasanan palautusprosessin avulla.

10. TOTEUTUKSEN ARVIOINTI

Työssä muokattu käyttäjähallinta ei ole täysin uusi ratkaisu. Tunnistautuminen toimii pohjimmiltaan käyttäen OAuth-protokollaa ja valtuutus hyödyntää JSON Web Tokenien väitteitä. Toteutuksen uutuusarvo löytyy valitusta teknologiasta ja miten OAuth- ja JSON-protokollia on sovellettu käyttäjän tunnistautumisessa ja valtuutuksessa. Vastaavia toteutuksia käyttäen Azuren AD B2C -käyttäjähallinnan muuttamisesta roolipohjaiseksi käyttäjähallinnaksi ei ole löydetty.

10.1 Vaatimusten toteutuminen

Käyttäjähallintaa varten oli esitetty useita vaatimuksia, joista suurin osa saatiin toteutettua. Käyttäjähallinnasta saatiin kehitettyä konseptitason toteutus, jonka perusteella voidaan todeta käyttäjähallinnan sopivan suunniteltuun tarkoitukseen. Vaatimusten toteutuminen on esitetty taulukossa 1.

Taulukko 1. *Vaatimusten toteutuminen.*

Vaatimus	Toteutuminen	Huomioita
Yksittäinen käyttäjähakemisto	Onnistuttiin	
Käyttäjähallinta palvelee useampaa asiakasympäristöä	Onnistuttiin	
Käyttäjien jaottelu eri asiakasympäristöjen käyttäjiksi	Onnistuttiin osittain	Jaottelu käyttäjien hallinnan kannalta vielä kesken
Käyttäjille voidaan määrittää eritasoisia oikeuksia	Onnistuttiin osittain	Roolien valtuutus asiakasympäristöissä vielä toteuttamatta
Kirjautumisessa monivaiheinen tunnistautuminen	Onnistuttiin	Monivaiheinen tunnistautuminen on osana kirjautumis- ja salasanan palauttamisprosessia.
Käyttäjille salasanan palautusprosessi	Onnistuttiin	Salasanan palautusprosessia käytetään myös osana käyttäjien rekisteröimistä
Salasanoja ei tallennetaan vain käyttäjähakemistoon	Onnistuttiin	Käyttäjät saavat aktivoitua salasanansa käyttäen salasanan palautusprosessia.
Käyttäjähallinnan ohjelmallinen hallinnointi	Vielä tekemättä	Ohjelmallinen hallinnointi on mahdollista, mutta toteutus kesken. Automaatio uusia asiakasympäristöjä asentamista varten osana jatkokehitystä.
Käyttäjähallinnalle räätälöitävä teema	onnistuttiin	Toteutettiin ulkoisten HTML- ja CSS-sivujen avulla.

Toteutus tehtiin käyttäen yksittäistä Microsoftin Azure AD B2C käyttäjähakemistoa. Hakemisto sisältää palvelun kaikkien asiakkuuksien käyttäjät. Käyttäjähallinta saatiin palvelemaan useampaa asiakasympäristöä rekisteröimällä Azure AD B2C:hen jokaiselle asiakasympäristölle oma sovellus. Nämä sovellukset toimivat asiakasympäristöille rajapintoina kirjautumiselle. Tällöin kirjautumisessa voidaan rajata käyttäjien pääsyä, kun heidän oikeuksissaan on puutteita.

Käyttäjähallinnan käyttäjien käyttöoikeuksien rajaaminen tehtiin roolien avulla. Käyttäjien kirjautuminen estetään, jos heillä ei ole asiakasympäristöön liittyviä rooleja. Tällöin käyttäjien pääsy voidaan rajata vain tiettyihin ympäristöihin. Roolit välitetään asiakasympäristöille kirjautumisen jälkeen JSON web tokenin mukana. Saatujen roolien avulla asiakasympäristöissä voidaan hoitaa käyttäjien valtuutus.

Käyttäjille voidaan jakaa käyttöoikeuksia useampaan asiakasympäristöön. Tällöin kuitenkin muuttuu epäselväksi, minkä asiakkuuden piiriin käyttäjä alun perin kuuluu. Hallinnoidessa asiakkuuksien käyttäjiä, tarvittaisiin jokin tapa ryhmitellä käyttäjät alkuperäisiin asiakkuuksiinsa. Tulevaisuudessa halutaan mahdollistaa asiakkuuksien omien ylläpitäjien pystyvän hallinnoimaan omia käyttäjiään. Tämä vaatisi erillisen ryhmittelyn, joka jakaa käyttäjät asiakkuuksiin käyttäjien hallinnoimisen näkökulmasta.

Toteutuksessa mahdollistettiin käyttäjien oikeuksien jako roolien avulla. Tämänhetkiset roolit valtuuttavat vain käyttäjän kirjautumaan asiakasympäristöihin. Tarkempi oikeuksien tarkastelu ja käyttöoikeuksien valtuutus täytyy tehdä asiakasympäristöissä. Asiakasympäristöjen tulee tulkita oikeudet JWT:n mukana tulevien roolien nimien perusteella. Roolitoteutus on ajatustasolla todettu toimivaksi, mutta käytännön toteutuksia ei niiden avulla ole vielä tehty.

Kirjautumisesta tehtiin vaatimuksien mukaan turvallinen. Kirjautumisen kriittisissä operaatioissa käytettiin Microsoftin luomia menetelmiä, jotka tulevat myös ajan saatossa saamaan järjestelmäpäivityksiä. Kirjautumisen ja salasanan palautuksen turvallisuutta parannettiin lisäämällä niihin monivaiheinen tunnistautuminen. Käyttäjät voivat kirjautua tai palauttaa salasanansa ainoastaan puhelinnumeron varmistamisen jälkeen.

Käyttäjien rekisteröiminen päädyttiin jättämään ylläpitäjien tehtäväksi. Palvelussa ei ole tarvetta antaa käyttäjien rekisteröidä itseään. Ylläpidon luodessa uudet käyttäjät, heidän salasanansa generoidaan satunnaisesti. Salasanoja ei tallenneta erillisiin kantoihin tai lähetetä käyttäjille. Käyttäjille ilmoitetaan käyttäjätunnusten luonnista ja he voivat asettaa heidän käyttäjätunnuksilleen salasanan hyödyntämällä salasanan palautusprosessia.

Käyttäjähallinnan ohjelmallinen hallinnointi todettiin toimivaksi käyttäen Microsoftin Graph Api -ohjelmointirajapintaa. Kaikki tarvittavat toiminnallisuudet käyttäjien, ryhmien ja sovellusten hallinnoinnin suhteen testattiin työtä kehitettäessä. Varsinaista hallinnointi käyttöliittymää tai rajapintojen ohjelmallista toteutusta ei työssä luotavan Azure Funktion lisäksi tehty.

Käyttäjähallinnalle saatiin määriteltyä räätälöity teema käyttäen omia ulkoisia HTML- ja CSS-sivuja. Sivut säilytettiin Azuren pilvitietovarastoon, jossa on julkiset lukuoikeudet. Mukautettuihin käytäntöihin konfiguroitiin, mistä sivut haetaan käyttäjälle käyttäjäprosessien yhteydessä.

Tehty työ todettiin onnistuneeksi. Työn toteutus mahdollistaa kaikki käyttäjähallinnalle esitetyt vaatimukset. Osa vaatimusten toteutuksista on vielä kesken, mutta työn perusteella voidaan luottaa loppujen vaatimuksien täyttyvän jatkokehityksen yhteydessä.

10.2 Jatkokehitys

Työn jatkokehityksessä aloitetaan käyttäjähallinnan integrointi osaksi Solitan Agile Data Engineä. Käyttäjänhallinnan asentaminen tavoitellaan täysin suoritettavaksi ohjelmallisesti. Ohjelmallisen asennuksen avulla käyttäjähallinnan pystyisi milloin tahansa asentamaan tyhjästä. Käyttäjähallinnan sisällön määrittäminen tiedetään onnistuvan käyttäen Microsofting Graph API:a. Asennettaessa käyttäjähallintaa, täytyy selvittää kuinka sen asennus voidaan tehdä käyttäen Azuren pilviympäristöjen hallintaan tarkoitettuja työkaluja.

Integraation yhteydessä täytyy asiakasympäristöjen sisäisiin sovelluksiin määritellä oikeuksien käsittely. Oikeuksien käsittely on mahdollisesti työläin osa integraatiota, joka riippuu asiakasympäristöjen sovellusten laajuudesta. Tässä vaiheessa tullaan huomaamaan, kuinka hyvin roolien toteutus toimii.

Tulevaisuudessa halutaan myös vähentää käyttäjähallinnan ADE:n ylläpitäjien työtä. Lopullisessa tavoitteessa ylläpitäjät luovat asiakkaiden ympäristöt ja alustavat asiakkuuteen liittyvät resurssit käyttäjähallintaan ohjelmallisesti. Asiakkuuksille luodaan omat ylläpito käyttäjät, jotka voivat luoda käyttäjiä ja hallita niiden oikeuksia omien asiakkuuksiensa sisällä. Näitä toiminnallisuuksia varten tullaan luomaan oma käyttäjähallintapaneeli. Tämä vaatii myös luvussa 10.1 mainitun käyttäjien jaon asiakkuuksiin, jotta asiakkuuksien ylläpitäjien hallintaoikeudet voidaan rajata oman asiakkuuden käyttäjiin.

10.3 Toteutuksen sovellettavuus

Työssä kehitetty käyttäjähallinta toimii parhaiten, kun siihen rekisteröitävät sovellukset ovat omia sovelluksia. Toteutus vaatii käyttäjien oikeuksien valtuutuksen toteuttamista sovelluksissa. Valtuutuksia kehitettäessä täytyy noudattaa käyttäjähallinnassa käytettyä nimeämiskäytäntöä. Kolmannen osapuolen sovellusten rekisteröinti on kuitenkin mahdollista. Niiden toimintakuntoon laittaminen vaatii kehittäjiä luomaan samanlainen käyttöoikeuksien valtuutus.

Toteutus toimii, kun useammalle pilvipalvelulle halutaan luoda yhteinen käyttäjähallinta, jota hallinnoidaan itse. Valittuun ratkaisuun päädyttiin, koska ei haluttu käyttää muiden hallinnoimia käyttäjiä. Omalla käyttäjäkannalla voidaan helpommin ylläpitää asiakkuuksia ja rajata käyttäjien oikeuksia asiakasympäristöihin.

Monissa tapauksissa ulkoisen identiteetin tarjoajan käyttö on parempi ratkaisu kuin työssä toteutettu vaihtoehto. Työssä toteutettu käyttäjähallinta on tarkoitettu rajattuun käyttöön. Käyttäjät lisätään ylläpitäjien toimesta eivätkä he voi itse rekisteröityä käyttäjiksi. Tämä rajoittaa käyttäjähallinnan käyttöä julkisessa käytössä.

Käyttäjähallintaan voisi luoda käyttäjille rekisteröinnin, mutta haasteita esiintyisi käyttöoikeuksien jaon kanssa. Käyttäjien oikeudet tulisi kuitenkin varmistaa ylläpidon puolesta ennen kuin he voivat käyttää mitään käyttäjähallintaan rekisteröityä sovellusta.

11. YHTEENVETO

Työn tavoitteena oli luoda roolipohjainen käyttäjähallinta käyttäen Azure AD B2C:tä. Samaan käyttäjähakemistoon oli tavoitteena saada useita eri käyttäjäryhmiä, joilla on erilaisia oikeuksia erinäisiin sovelluksiin. Käyttäjähallinnan täytyi kyetä palvelemaan useampaa asiakasympäristöä. Azure AD B2C ei ennen tämän työn muokkauksia täyttänyt yhtäkään näistä vaatimuksista. Työ saatiin toimimaan halutulla tavalla vaikka vielä ei täyttä integraatiota Solitan Agile Data Engineen tehty.

Työssä ei voitu käyttää kovin paljoa valmista tieteellistä materiaalia sillä vastaavia muokkauksia Azure AD B2C käyttäjähallinnasta ei löytynyt. Toteutus tehtiin hyödyntämällä Microsoftin tekemiä dokumentaatioita ja käymällä läpi monia esimerkkejä, joissa vastaavia asioita oli tehty. Työn eteneminen suurimmaksi osaksi tapahtui yrityksen ja erehdyksen kautta.

Käyttäjähallinta saatiin palvelemaan useampaa asiakasympäristöä rekisteröimällä asiakasympäristöille omat sovellukset. Rekisteröityjä sovelluksia käytetään rajapintoina asiakasympäristöjen tunnistautumiselle. Tällöin kuitenkin asiakasympäristöt jakoivat vain käyttäjähakemiston ilman erillistä käyttäjien pääsyn rajausta. Käyttäjien pääsyn rajaus toteutettiin käyttäen käyttöoikeusryhmiä, joihin käyttäjät voitiin lisätä. Nämä käyttöoikeusryhmät lisättiin rekisteröityjen sovellusten jäseniksi, jolloin käyttäjät voitiin käyttöoikeusryhmien avulla määritellä sovellusten käyttäjiksi. Näiden tarkastaminen toteutettiin osana ulkoista logiikkaa, joka toteutettiin Azure Funktion avulla. Roolit määriteltiin käyttäjähallinnan puolella ja vastuu niiden tarkastamisesta jätettiin asiakasympäristöiden sovelluksille. Käyttäjähallinnassa roolit nimettiin systemaattisella tavalla, jonka avulla sovellukset voivat tulkita käyttäjien roolit niiden nimien perusteella. Käyttäjähallinnan toiminnallisista vaatimuksista saatiin suurin osa täytettyä.

Käyttäjähallintaan toteutettiin monivaiheinen tunnistautuminen käyttäjän kirjautumiseen ja salasanan palauttamiseen. Kun käyttäjä on rekisteröinyt monivaiheisen tunnistautumisen, ei kukaan muu voi kirjautua tai muuttaa käyttäjän salasanaa ilman hänen puhelintaan. Käyttäjän rekisteröiminen toteutetaan ylläpitäjien toimesta ohjelmallisesti, jolloin käyttäjien salasanat generoidaan satunnaisesti. Salasanoja ei tallenneta erillisiin tietokantoihin tai lähetetä käyttäjille sähköpostilla. Käyttäjät saavat käyttäjätunnustensa salasanat ainoastaan suorittamalla salasanan palautusprosessin vahvistamalla heidän sähköpostinsa ja asettamalla monivaiheisen tunnistautumisen.

Käyttäjähallinnan hallinnoiminen toteutettiin hyödyntämällä Microsoft Graph Api ohjelmistorajapintaa. Sen avulla sai tehtyä kaikki operaatiot käyttäjien, käyttöoikeusryhmien ja sovellusten rekisteröimiseen liittyen. Työssä kuitenkin jäi selvittämättä, kuinka käyttäjähallinnan infrastruktuuri saadaan ohjelmallisesti pystytettyä. Työn toteutuksessa oletetaan käyttäjähallinnan olevan toiminnassa jo valmiiksi. Käyttäjähallinnan ohjelmallista hallinnointia joudutaan vielä tutkimaan osana jatkokehitystä.

Loppujen lopuksi työn tulos koettiin onnistuneeksi. Työssä tehtyä käyttäjähallinnan kehitystä tullaan jatkamaan ja integroimaan osaksi Solitan Agile Data Engine -tuotetta. Luotu käyttäjähallinta vaikuttaa hyvin lupaavalta ja sen odotetaan vähentävän huomattavasti käsityön määrää uusien asennusten luomisen yhteydessä ja käyttäjähallinnan ylläpitämisessä.

12. LÄHTEET

[1] K. Chard and I. Foster, "Serverless science for simple, scalable, and shareable scholarship," in - *2019 15th International Conference on eScience (eScience),* 2019, .

[2] R. Boyd, *Getting Started with OAuth 2.0.* (1st ed.) 2012.

[3] Juan R. Rodriguez, et al., "Masterworkshop IBM WebSphere DataPower SOA Appliances: Part II: Authentication and Authorization," 2008. Available: https://masterworkshop.skillport.com/skillportfe/assetSummaryPage.action?assetid=RW$28239:_ss_book:31375#summary/BOOKS/RW$28239:_ss_book:31375.

[4] l. o. J. 11. (-06-11T05:14:53+00:00). *Authentication vs. Authorization Defined: What's the Difference? [Infographic]*. Available: https://securityboulevard.com/2020/06/authentication-vs-authorization-defined-whats-the-difference-infographic/.

[5] S. Michael and Z. J. Anna, "An identity provider as a service platform for the eduGAIN research and education community," in - *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM),* 2019, .

[6] "What is ADFS? | Active Directory Federation Service (ADFS)," Available: https://docs.miniorange.com/articles/what-is-adfs.

[7] *SaaS: Single Tenant vs Multi-Tenant - What's the Difference?*. Available: https://digitalguardian.com/blog/saas-single-tenant-vs-multi-tenant-whats-difference.

[8] S. Ahmed and Q. Mahmood, "An authentication based scheme for applications using JSON web token," in *2019 22nd International Multitopic Conference (INMIC)*Anonymous 2019, pp. 1-6.

[9] *Overview of tokens - Azure Active Directory B2C*. Available: https://docs.microsoft.com/en-us/azure/active-directory-b2c/tokens-overview.

[10] *OAuth 2.0*. Available: https://docs.oracle.com/cd/E82085_01/160023/JOS%20Implementation%20Guide/Output/oauth.htm.

[11] B. Leiba, "OAuth Web Authorization Protocol," *Mic,* vol. 16, *(1),* pp. 74-77, 2012. . DOI: 10.1109/MIC.2012.11.

[12] I. P. Pratama, L. Linawati and N. P. Sastra, "Token-based Single Sign-on with JWT as Information System Dashboard for Government," *Telkomnika,* vol. 16, *(4),* pp. 1745-1751, 2018. . DOI: 10.12928/telkomnika.v16i4.8388.

[13] A. Bertram, "Understanding Azure App Registrations | Petri," 2020. Available: https://petri.com/understanding-azure-app-registrations.

[14] D. Mosley. (-06-01T17:00:57+00:00). *Understanding Security Principals - Active Directory Implementation Windows Server 2003*. Available: https://www.serverbrain.org/active-directory-implementation-2003/understanding-security-principals.html.

[15] Alexander S. Gillis. (). *What is Active Directory (AD)?*. Available: https://searchwindowsserver.techtarget.com/definition/Active-Directory.

[16] O. Media. (). *Microsoft Azure Infrastructure Services for Architects*. Available: https://learning.oreilly.com/library/view/microsoft-azure-infrastructure/9781119596578/.

[17] *Azure AD, B2B, B2C Puzzled Out - What Makes The Difference?*. Available: https://www.predicagroup.com/blog/azure-ad-b2b-b2c-puzzled-out/.

[18] *Azure Active Directory B2C custom policy overview*. Available: https://docs.microsoft.com/en-us/azure/active-directory-b2c/custom-policy-overview.

[19] *User flows and custom policies in Azure Active Directory B2C - Azure AD B2C*. Available: https://docs.microsoft.com/en-us/azure/active-directory-b2c/user-flow-overview.

[20] "Using Groups in Azure AD B2C – Found it interesting," Available: https://mrochon.azurewebsites.net/2019/05/06/using-groups-in-azure-ad-b2c/.

[21] O. Media. (). *1. Introduction to Azure Functions*. Available: https://learning.oreilly.com/library/view/beginning-azure-functions/9781484244449/html/473508_1_En_1_Chapter.xhtml.

[22] Davide Taibi, Nabil El Ioini, Claus Pahl and Jan Raphael Schmid Niederkofler, "Patterns for Serverless Functions (Function-as-a-Service): A Multivocal Literature Review," 2020. Available: http://urn.fi/URN:NBN:fi:tuni-202008286730.

[23] *Triggers and bindings in Azure Functions*. Available: https://docs.microsoft.com/en-us/azure/azure-functions/functions-triggers-bindings.

[24] *What is Azure Event Grid? - Azure Event Grid*. Available: https://docs.microsoft.com/en-us/azure/event-grid/overview.

[25] P. Pialorsi, *Programming Microsoft Office 365: Covers Microsoft Graph, Office 365 Applications, SharePoint Add-Ins, Office 365 Groups, and More.* (1st ed.) 2016.

[26] *Get access without a user - Microsoft Graph*. Available: https://docs.microsoft.com/en-us/graph/auth-v2-service.

[27] *Create application - Microsoft Graph v1.0*. Available: https://docs.microsoft.com/en-us/graph/api/application-post-applications.

[28] *Create group - Microsoft Graph v1.0*. Available: https://docs.microsoft.com/en-us/graph/api/group-post-groups.

[29] *Create User - Microsoft Graph v1.0*. Available: https://docs.microsoft.com/en-us/graph/api/user-post-users.

[30] *Apps & service principals in Azure AD - Microsoft identity platform*. Available: https://docs.microsoft.com/en-us/azure/active-directory/develop/app-objects-and-service-principals.

[31] *Groups in Microsoft 365 and Azure, and Which is Right for You*. Available: https://docs.microsoft.com/en-us/microsoft-365/community/all-about-groups.

[32] *Securing Azure Functions*. Available: https://docs.microsoft.com/en-us/azure/azure-functions/security-concepts.

[33] *Azure Functions HTTP trigger*. Available: https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-http-webhook-trigger.