Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2023

Juuso Lepistö

.NET Core -sovelluksen modernisointi Azure -ympäristöön

Opinnäytetyö (AMK) | Tiivistelmä

Turun ammattikorkeakoulu

Tieto- ja viestintätekniikka

2023 | 82 sivua

Juuso Lepistö

.NET Core -sovelluksen modernisointi Azure -ympäristöön

Opinnäytetyö käsitteli .NET Core -sovelluksen siirtämistä omien konesalien virtuaaliympäristöstä moderniin Azure -pilviympäristöön. Työn tavoitteena oli päivittää sovellus käyttämään Microsoftin Azure Platform as a Service (PaaS) komponentteja ja samalla päivittää sovellus uudempaan .NET Core -versioon. Työssä tarkasteltiin sovellusta ja sovelluksen nykyversion teknistä ratkaisua ja sen ongelmakohtia, sekä tutkittiin, miten moderni pilviympäristö kykenee tarjoamaan palveluita korvaamaan ja korjaamaan näitä ongelmia, ja miten sovelluksen ylläpito voitiin tehdä helpommaksi. Työssä käytiin läpi, miten sovelluksen modernisointiprosessi oli suoritettu ja minkä vuoksi. Työn lopuksi arvioitiin modernisoinnin vaikutuksia käytettävyyteen ja suorituskykyyn. Työn tuloksena saavutettiin päivitetty ja nykyaikainen sovellus, joka toimii modernissa pilviympäristössä ja mahdollistaa tehokkaan ylläpidon.

Asiasanat:

ohjelmistokehitys, tuotekehitys, .NET, Azure

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Information and Communications Technology

2023 | 82 pages

Juuso Lepistö

Modernizing a .NET Core application to Azure environment.

The thesis focused on migrating a .NET Core application from a virtual server to a modern Azure cloud environment. The aim of the work was to transfer the application to use the flexible and scalable resources that Azure has to offer, while updating the application to a newer .NET Core version. The application, it’s technical content and problem areas of the current version were examined, and it was investigated how a modern cloud environment could provide services to replace and fix these issues and how maintenance of the application could be made easier. The thesis described how the modernization process of the application was carried out and why. Finally, the effects of the modernization on usability and performance were evaluated. As a result, an updated and modernized version of the application was achieved, which operates flexibly in the cloud environment and enables efficient maintenance.

Keywords:

software development, product development, .NET, Azure

Sisältö

Kuvat

Käytetyt lyhenteet ja sanasto

AD	Active Directory (AD) on Microsoftin oma hakemistopalvelu, joka pyörii Windows Serverillä. AD antaa järjestelmävalvojalle oikeuden hallita eri lupia ja oikeuksia, joilla käyttäjät voivat päästä käsiksi verkon resursseihin. (TechTarget, 2021a)
API	Application Programming Interface (API) on valmiiksi kehitettyä koodia, jonka avulla kaksi osapuolta ohjelmistossa voivat keskustella toistensa kanssa. (TechTarget, 2022a)
ARM	ARM-mallit (engl. Azure Resource Manager templates) ovat Bicep- tai JSON-tiedostoja, joilla määritellään tarvittavat resurssit ratkaisun käyttöönottoa varten. (Microsoft, 2023o)
ASP	Action Server Page (ASP) ohjelmistokehys on palvelinpuolen komentosarjamoottori, joka tuottaa interaktiivisia verkkosivuja. (TechTarget, 2019a)
backend	Backend, eli palvelinpuoli tarkoittaa taas sitä koodia, joka ajetaan palvelimella, eikä selaimella. (GeeksforGeeks, 2023b.)
CLI	Command Line Interface (CLI) on komentorivityökalu, jota käytetään ohjelmien suorittamiseen, tietokonetiedostojen hallintaan ja vuorovaikutukseen tietokoneen kanssa. (TechTarget, 2021b)
CSS	Cascading Style Sheets (CSS) on tyylitaulukkokieli, jota käytetään kuvaamaan kirjoitetun HTML-koodin esitystapaa. (MDN Web Docs, 2023a)

EF Core	Enity Framework Core on avoimen lähdekoodin, monialustainen versio Entity Framework -datan käsittelyteknologiasta. (Learn Entity Framework Core, 2023)
EOL	End of Life (EOL) on tila, jonka komponentti saa sen saavuttua elinkaarensa päätökseen, eli laakaa saamasta tukea ja päivityksiä. (Microsoft, 2023w)
Framework	Ohjelmistokehys, eli uudelleenkäytettävä kehitysalusta ohjelmistoympäristöille, joka tukee ohjelmointikielikirjastoja ja komentosarjakieliä. (InterviewBit, 2022)
frontend	Puhutaan ohjelmoinnissa selainpuolesta (frontend), eli tarkoittaa koodia, joka ajetaan selaimessa. Kaikki mitä käyttäjä näkee käyttäessään sovellusta, on selainpuolta. (GeeksforGeeks, 2023b.)
GBAC	Ryhmäpohjainen autentikaatio tai englanniksi Group Based Access Control (GBAC). (Opal, n.d.)
HTML	Hypertext Markup Language (HTML) on verkkosivun rakentamisen alkeellisin osa, jolla määritellään sivun sisällön merkitys ja rakenne. (MDN Web Docs, 2023b)
http	Hypertext Transfer Protocol (http) on protokolla, jota käytetään tiedonsiirrossa verkossa. (W3Schools, n.d.)
IaaS	Infrastruktuuri palveluna (engl. Infrastructure as a Service) on yksi pilvipalvelun malleista, joka tarpeen mukaan saatavilla oleva käyttö tietokoneiden laskentaresursseihin, tallennustilaan, verkkoyhteyksiin ja virtualisointiin. (Google Cloud n.d.a)

IDE	Integrated Development Enviroment (IDE) on ohjelmistoympäristö, jolla ohjelmoija kykenee suunnittelemaan ja toteuttamaan eri ohjelmistoja. (Amazon Web Service n.d.b)
IIS	Internet Information Services (IIS) on Microsoftin joustava ja yleiskäyttöinen verkkopalvelin. (TechTarget, 2019a)
JS	JavaScript (JS) on komentosarjakieli, joka on eniten käytössä selainsivujen komentosarjoissa, mutta sitä käytetään myös ei-selainympäristöissä, kuten palvelinpuolien rakentamisessa. (MDN Web Docs, 2023d)
JSON	JavaScript Object Notation (JSON) on standardoitu tekstipohjainen formaatti, joka perustuu JavaScript-objektien syntaksiin ja jota käytetään strukturoitujen tietojen esittämiseen. (MDN Web Docs, 2023e)
LTS	Pitkäaikainen tuki (engl. Long Term Support) on yksi .NET-versioiden julkaisutyypeistä (LTS). (Microsoft, 2023w)
MFA	Multi-factor authentication (MFA) on moniosainen tunnistautumisprosessi, jossa käyttäjältä vaaditaan muutakin kuin pelkkä salasana. Salasanan lisäksi käyttäjältä saatetaan, sähköpostin tai puhelinnumeron välityksellä, pyytää vahvistuskoodia. (Amazon Web Services n.d.c)
MVC	Model – View – Controller (malli – näkymä - ohjaimet) - suunnittelumalli (engl. MVC-model), jota käytetään ASP .NET Coressa. (Microsoft, 2022d)

ORM	Object-Relational Mapper (ORM) on tekniikka, jolla kehittäjä voi työskennellä tietokannan kanssa käyttäen olio-ohjelmointikieltä, mieluummin kuin SQL-komentoja. (Learn Entity Framework Core, 2023)
PaaS	Sovellusalusta palveluna (engl. Platform as a Service) viittaa laitteisto- tai ohjelmisto- resursseihin, joita tarvitaan pilvessä toimivien sovelluksien kehitykseen (PaaS). (Google Cloud n.d.a)
RBAC	Roolipohjainen autentikaatio tai englanniksi Role Based Access Control (RBAC) rajoittaa käyttäjän verkkotoimintaa roolin pohjalta. (Digital Guardian, 2023)
REST	Representational state transfer (REST) on ohjelmistoarkkitehtuurimalli, joka määrittelee ne ehdot, kuinka API:n tulisi toimia. (Amazon Web Services n.d.a)
Runtime	Runtime on komponentti, joka on vastuussa kehittäjän tuottaman koodin suorittamisesta ja hallinnasta. (TechTarget, 2021c)
SaaS	Ohjelmisto palveluna (engl. Software as a Service) tarkoittaa koko sovelluspinon tarjoamista pilvipalveluna, mukaan lukien alustan alapuolisen infrastruktuurin ylläpitoa ja hallintaa aina sovellusohjelmistoon asti (SaaS). (Google Cloud n.d.a)
SDK	Software Development Kit (SDK) on joukko työkaluja ja ohjelmistoja, joita ohjelmistokehittäjät voivat käyttää rakentamaan sovelluksia tietyille alustoille. (TechTarget, 2022d)

SSO	Single sign-on (SSO) on istunnon ja käyttäjä todennus palvelu, joka tarjoaa käyttäjälle oikeuden käyttää yhtä sarjaa kirjautumistietoja (kuten käyttäjätunnus ja salasana) kirjautuakseen useammalle eri sovellukselle tai järjestelmälle. (TechTarget, 2022c)
SSMS	SQL Server Management Studio (SSMS) on integroitu ympäristö SQL-infrastruktuurin hallintaan, SQL Serveristä Azure SQL -tietokantaan. SSMS antaa käyttäjälleen työkalut SQL Serverin ja tietokantojen määritykseen, monitorointiin ja hallintaan. (Microsoft, 2023g)
STS	Lyhytaikainen tuki (engl. Short Term Support) on yksi .NET -versioiden julkaisutyypeistä (STS). (Microsoft, 2023w)
SQL	Structured query language (SQL) on ohjelmointikieli, jota käytetään tietokantojen hallintaan ja niiden sisällä olevien tietojen kyselyyn. (TechTarget, 2022e)
UI	Käyttöliittymä (engl. User Interface) (UI) on käyttäjän ja tietokoneen välinen rajapinta ja vuorovaikutuspiste laitteessa. (TechTarget, 2021e)
URI	URI (Uniform Resource Identifier) on merkkijono, joka tunnistaa loogisen tai fyysisen resurssin, internetiin liittyen. URI erottaa yhden resurssin toisesta. (TechTarget, 2021d)
VM	Virtuaalikone (engl. Virtual machine) (VM) on virtuaalinen versio fyysisestä tietokoneesta. (IBM, n.d.)

1 Johdanto

Laitepassi -niminen .NET Core -sovellus on 2M-IT:n kehittämä, selaimella käytettävä www-sovellus, jota Satasairaalassa käytetään laitepassien luontiin ja niiden hallintaan. Sovelluksen päätoiminnallisuutena on henkilökohtaisten laitepassien luonti ja täyttäminen yksikkökohtaisten laitepassipohjien pohjalta.

Sovelluksen kehittäjä, 2M-IT tarjoaa tietoteknisiä palveluita ja jatkuvaa kehitystyötä hyvinvointialueiden sosiaali- ja terveydenhuollon eri haasteiden ratkaisemiseksi ja kehittämiseksi. (2M-IT n.d.) Yhtiö tarjoaa sosiaali- ja terveydenhuollon tietojärjestelmä-, ratkaisu- ja tietotekniikkapalveluita Suomessa, 15 eri sairaanhoitopiirin alueella. (Wikipedia, 2022)

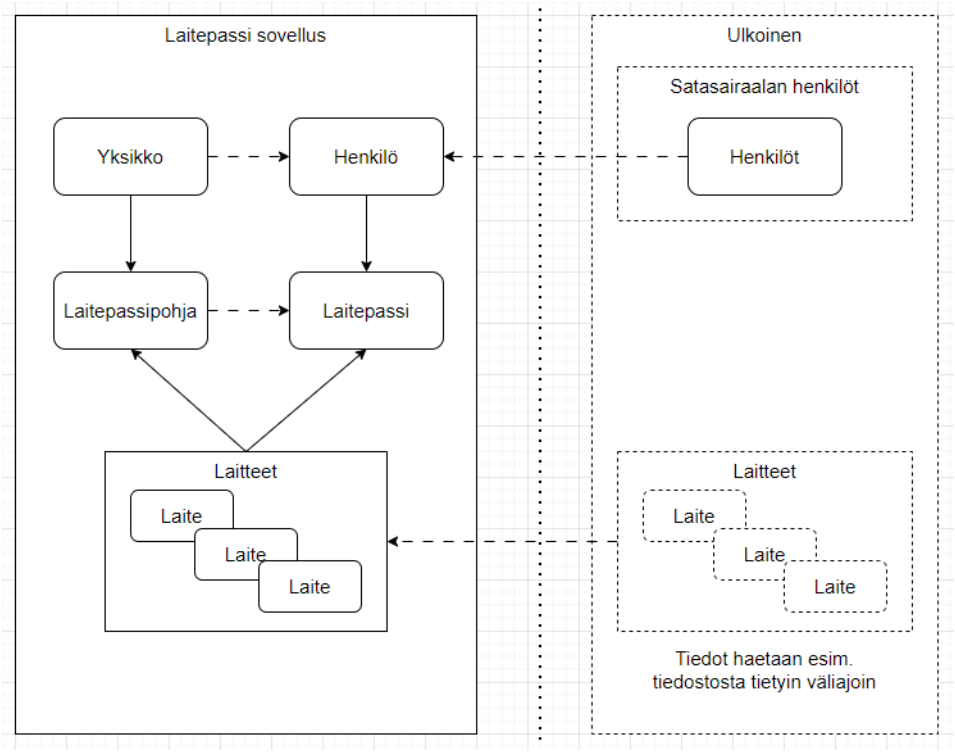
Sovelluksen on kehittänyt ja siihen sovelluspalvelua tuottaa 2M-IT. Sovellus on ollut käytössä 2M-IT- ja Satasairaala -verkoissa, millä on varmistettu myös verkkotasolla sovelluksen oikeuksien rajaus. Sovelluksen käyttämien palvelimien, ohjelmistojen ja sovelluskomponenttien vanhetessa sovelluksen päivittäminen uudelle alustalle pilveen on ajankohtaista. Tarkoituksena on myös Sote-uudistuksen myötä tehdä ratkaisu, joka skaalautuu helposti koko hyvinvointialueelle.

Tässä opinnäytetyössä käydään läpi .NET Core -sovellusta, sen teknistä sisältöä ja sovellukseen liittyviä teknisiä ongelmia. Ongelmille tehdään ratkaisuehdotuksia ja havainnollistetaan, kuinka sovelluksen parannukset toteutetaan.

2 Sovelluksen kuvaus

2M-IT on määritellyt ja suunnitellut Laitepassi-sovelluksen yhteistyössä asiakkaan (Satasairaala) kanssa. Sovelluksen on toteuttanut ja jatkokehittänyt tiimi 2M-IT:n kehittäjiä. Sovelluksen määrittely, suunnittelu ja toteutus on tehty kokonaisuudessaan ainoastaan 2M-IT:n ja asiakkaan toimesta. Sovellus on Satasairaalan käytössä ja sovelluksen tuottaa sitä kehittävä tiimi.

Laitepassi on www-sovellus, jota käytetään laitepassien luontiin ja hallintaan. Sovelluksen päätoiminnallisuutena on henkilökohtaisten laitepassien luonti ja täyttäminen yksikkökohtaisten laitepassipohjien pohjalta. Sovelluksessa on yksiköitä ja jokaisella yksiköllä on oma laitepassipohjansa. Laitepassipohjat koostuvat laitelistauksesta, jotka haetaan esimerkiksi asiakkaan toimittamasta tiedostosta. Käyttäjä kirjautuu laitepassiin, ja hän pääsee tarkastelemaan tai muokkaamaan laitepassiaan, joka on rakennettu hänen yksikkönsä laitepassipohjan perusteella. (Kuva 1)



Kuva 1 Laitepassin päätoiminnallisuudet.

Sovelluksessa on 4 eri käyttäjäryhmään: peruskäyttäjä, laitevastaava, esimies ja ylläpitäjä. Käyttäjän kirjautuessa sovellukseen häneltä haetaan sovelluksen käyttöön oleellista tietoa. Käyttäjältä haettaviin tietoihin kuuluu muun muassa käyttäjän nimi, ammattinimike, yksikkö ja käyttäjäryhmä. Riippuen siitä mihin käyttäjäryhmään käyttäjä kuuluu, asetetaan hänellä jokin seuraavista rooleista: laitepassin haltija, laitevastaava, esimies tai ylläpitäjä.

Roolien hierarkiassa laitepassin haltija sijoittuu alimmalle tasolle ja sillä on vähiten oikeuksia muihin rooleihin verrattuna. Laitevastaava ja esimies kuuluvat hierarkian keskitasoon ja ylläpitäjä kuuluu korkeimpaan tasoon. Kuvassa 2 on esimerkki, jossa tarkistetaan If-lauseella, jos käyttäjä kuuluu rooliin laitevastaava tai esimies. Mikäli jompikumpi väittämistä pitää paikkaansa, esiintyy käyttäjälle tietynlainen näkymä sovelluksen verkkosivulla.

```
@if (!(User.IsInRole(LaitepassiRoolit.LAITEVASTAAVA) || User.IsInRole(LaitepassiRoolit.ESIMIES)))
```

Kuva 2 Esimerkkikoodi käyttäjän roolin päättelystä.

Laitepassin haltijan kirjautuessa sovellukseen (Kuva 2), hänen aloitusnäkymänsä on hänen oma laitepassinsa tarkastelusivu.

Toisin kuin laitepassin haltija, muilla rooleilla toimivat käyttäjät aloittavat toisenlaisella aloitussivulla. Laitevastaava-, esimies- ja ylläpitäjäroolin saaneet käyttäjät ovat oikeutettuja sovelluksen laajempaan sisältöön. He pääsevät luomaan uusia, muokkaamaan olemassa olevia tai poistamaan sisältöä, oli kyseessä esimerkiksi henkilö, laite tai laitepasseja. Vain ylläpitäjän roolin saanut käyttäjä pystyy käsittelemään sovelluksen yksiköitä ja ammattinimikkeitä.

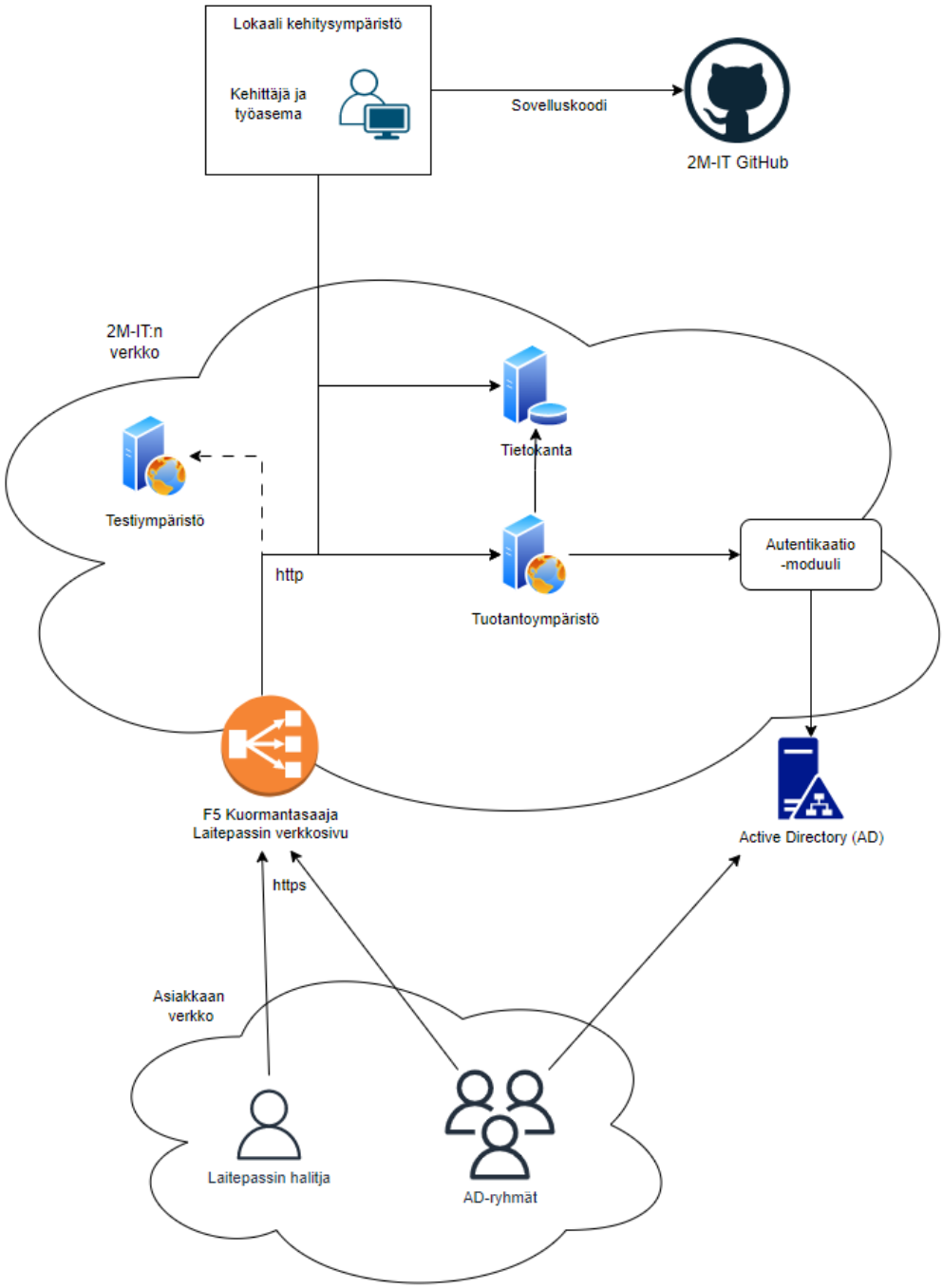
3 Nykyratkaisun teknisten komponenttien kuvaus

Tämä kappale käsittelee Laitepassi-sovelluksen nykyistä teknistä arkkitehtuuria ja komponentteja, joista tekninen ympäristö koostuu.

3.1 Tekninen ympäristö

Kuvassa 3 esitetään Laitepassi-sovelluksen teknistä arkkitehtuuria. Sovellusta kehitetään kehittäjien paikallisissa kehitysympäristöissä ja sovelluskoodia säilytetään 2M-IT:n versionhallinnassa, GitHub-verkkosivulla. Sovelluksen uusia versioita julkaistaan paikallisesta kehitysympäristöstä, 2M-IT:n hallitsemille sovelluspalvelimille, testi- ja tuotantoympäristöille. Näiden ympäristöjen kautta 2M-IT:n ja asiakkaan verkkoon oikeutetut pääsevät kirjautumaan sovellukseen.

Jos halutaan käyttää testiympäristöä, kirjaudutaan sovellukseen kehittäjien räätälöimillä testitunnuksilla. Tuotantoympäristöön kirjautumista varten käytetään Active Directoryssa olevia tunnuksia ja autentikaatio-moduulia.



Kuva 3 Arkkitehtuurikuvaus.

3.1.1 Sovelluspalvelimet

Sovelluspalvelin sijaitsee yleensä joko ulkopuolisessa datakeskuksessa tai pilviympäristössä. Palvelimen virtualisointi tarkoittaa fyysisen palvelimen muuntamista virtuaalikoneisiin. Sovelluspalvelin konfiguroidaan niin, että

useampi käyttäjä voi jakaa sen prosessointitehoa. Virtuaalipalvelimia suositaan fyysisten palvelinten ylitse, koska kustannuskulut laitteistosta ovat vähempiä ja teho- ja energiakustannukset alhaisempia. (Google Cloud n.d.c) Sovelluksen käytössä on kaksi sovelluspalvelinta: testi- ja tuotantopalvelin.

Sovelluspalvelimien käyttöjärjestelmä on Windows Server 2012. Windows Serveriä hallinnoi käyttöjärjestelmäryhmä, joka on yritystason hallinnan, tietojen tallennuksen, tietojen tallennuksen, sovelluksien ja viestinnän tukemana. (Microsoft, 2022a) Laitepassin käytössä oleva Windows Server -versio on Windows Server 2012. Sovelluspalvelimet sijaitsevat 2M-IT:n omissa konesaleissa.

3.1.2 WWW-palvelin

Sovellus käyttää Internet Information Servicesiä (IIS), joka on Microsoftin joustava ja yleiskäyttöinen verkkopalvelin www-sivustojen hallinnointiin ja julkaisuun. Palvelin toimii Windows -sovelluspalvelimilla, palvellen pyydettyjä sivuja tai tiedostoja, ja käsittelee käyttäjän lähettämiä pyyntöjä ja palauttaa siihen sopivia vastauksia. Palvelin tekee yhteistyötä Microsoftin ASP (Active Server Page) .NET Core -ohjelmistokehyksen kanssa. Tämä ohjelmistokehys on palvelinpuolen komentosarjamoottori, joka tuottaa interaktiivisia verkkosivuja, mutta tätä käydään paremmin läpi luvussa 2.3.3. Ideana on, että käyttäjä lähettää verkosta IIS:lle pyynnön, joka toimittaa pyynnön eteenpäin ASP.NET Core -sovellukselle. Sovellus sitten prosessoi pyynnön ja lähettää pyynnön takaisin IIS-palvelimelle ja siitä vielä käyttäjälle, jolta pyyntö alun perin lähti liikkeelle. (TechTarget, 2019a)

IIS käyttää versiota 8.5, joka toimii Windows Server 2012 -virtuaalipalvelimen kanssa. Jotta kehitetty sovellus voi toimia, se edellyttää .NET Core Runtime 3.1 asennuksen sovelluspalvelimelle. .NET Coresta kerrotaan luvussa 2.3.2 ja Runtimesta luvussa 2.3.3.

3.1.3 SQL-palvelin ja -tietokanta

Virtuaalipalvelimelle Windows Server 2016 on asennettu relaatiotietokannan hallintajärjestelmä, joka on Microsoft SQL Server 2016. SQL Server tukee muun muassa tapahtumien käsittelyä, liiketoiminnanhallintaa ja analytiikkasovelluksia yritysten tietoteknisissä ympäristöissä. (TechTarget, 2019b) Relaatiotietokanta on tietokanta, joka on rakennettu tunnistamaan tallennettujen tietojen välisiä suhteita. SQL Server in koottu SQL-kielen päälle. SQL (Structured query language) on ohjelmointikieli, jota tietotekniikan ammattilaiset käyttävät tietokantojen hallintaan ja niiden sisällä olevien tietojen kyselyyn. (TechTarget, 2022e)

SQL Serverin keskeinen osa on SQL Server Database Engine, joka hallitsee tiedon säilytystä, sen käsittelyä ja turvallisuutta. Se sisältää relaatiomoottorin, joka käsittelee käskyjä ja kyselyitä, sekä varastomoottorin, joka hallinnoi tietokantatiedostoja, taulukoita, sivuja, indeksejä, datapuskureita ja transaktioita. Tietokannan varastoidut proseduurit, laukaisimet, näkymät ja muut objektit myös luodaan ja suoritetaan Database Enginellä. (TechTarget, 2019b)

SQL Serverin tietokanta koostuu taulukoiden kokoelmasta, joka tallentaa tietyn rakenteellisen datan. Taulukko sisältää rivien, sekä sarakkeiden kokoelman. Jokainen taulukon sarake on suunniteltu tietyn tyyppisen tiedon tallentamiseen, esimerkiksi nimiä, päivämääriä ja numeroita. (Microsoft, 2023f)

3.1.4 Satasairaala Active Directory

Active Directory on Microsoftin oma hakemistopalvelu, joka pyörii Windows Serverillä. AD antaa järjestelmävalvojalle oikeuden hallita eri lupia ja oikeuksia, joilla käyttäjät voivat päästä käsiksi verkon resursseihin. (TechTarget, 2021a)

3.1.5 F5 kuormantasaaja

Kuormantasaaja on laite, joka toimii käänteisenä välityspalvelimena ja jakaa verkon tai liikenteen useiden palvelinten kesken. Kuormantasaajaa käytetään sovelluksen kapasiteetin ja luotettavuuden lisäämiseen. Se parantaa sovelluksen yleistä suorituskykyä vähentämällä palvelimiin kohdistuvaa kuormitusta, joka liittyy sovellusten ja verkkosessioiden hallintaan ja ylläpitoon, sekä suorittamalla sovelluskohtaisia tehtäviä.

3.2 Sovelluskomponentit

Laitepassi-sovellus on rakennettu useista sovelluskomponenteista, jotka muodostavat sen toiminnallisuuden selkärangan. Tässä kappaleessa tutustutaan näihin sovelluskomponentteihin.

3.2.1 .NET Core

Laitepassi-sovellus on kehitetty .NET Corella, joka on ilmainen, monialustainen, avoimen lähdekoodin kehittäjäalusta monien erilaisten sovellusten rakentamista varten. .NET Core -projektin ovat ensisijaisesti kehittäneet Microsoftin työntekijät .NET-säätiön kautta. (Microsoft, 2023q) Projektin ylläpidosta vastaa Microsoft ja Github -yhteisö. Se on monialustainen kehys, joka toimii Windows-, Linux- ja macOS -järjestelmillä. (GeeksforGeeks, 2023a) .NET:llä voi rakentaa useita erilaisia sovelluksia, kuten verkkosovelluksia, työpöytäsovelluksia, mobiilisovelluksia, pelisovelluksia tai jopa pilvipohjaisia sovelluksia. .NET tarjoaa kehittäjälle tarvittavat työkalut näiden luomiseen. (Microsoft, 2023q)

3.2.2 ASP.NET Core

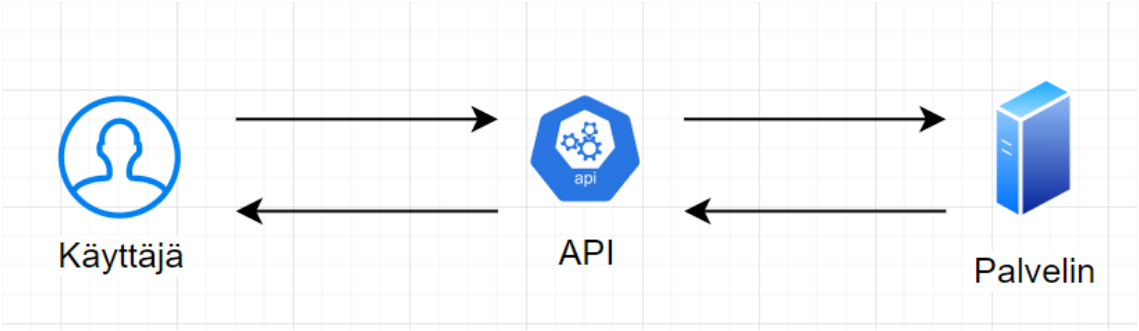
Kuten luvussa 2.1.2 jo lyhyesti käytiin asiaa lävitse, ASP .NET Core on Microsoftin kehittämä ohjelmistokehys. Tarkennettuna kyseessä on palvelinpuolen komentosarjamoottori, joka tuottaa interaktiivisia verkkosivuja.

ASP.NET Core on Windows-, macOS- ja Linux -alustoilla toimivat kehys pilvipohjaisten sovellusten, kuten tässä tapauksessa verkkosovellusten rakentamiseen. Kehys on suunniteltu niin, että se kykenee toimimaan niin pilvessä, että paikan päällä, jossa sovellus tällä hetkellä pyörii. Kehys on kohdistettu toimimaan .NET Core alustalla. ASP.NET Corella ei ole erillistä versiota, vaan se toimii samalla versiolla kuin .NET Core, jonka runtime ja SDK (Software Development Kit) sisältävät tarvittavat ASP.NET Core -kirjastot. (TutorialsTeacher n.d)

Runtime on komponentti, joka on vastuussa kehittäjän tuottaman koodin suorittamisesta ja hallinnasta. Kun ohjelmaa ajetaan, se ladataan automaattisesti runtime ympäristöön, joka sisältää tarvittavat resurssit ja palvelut ohjelman suorittamiseen. Tyypillisesti ympäristö sisältää joukon kirjastoja ja muita komponentteja, jotka tukevat ohjelman suoritusta. (TechTarget, 2021c)

Software Development Kit (SDK) on joukko työkaluja ja ohjelmistoja, joita ohjelmistokehittäjät voivat käyttää rakentamaan sovelluksia tietyille alustoille. SDK:n mukaan kuuluu dokumentaatio, API (Application programming interface eli rajapinta), esimerkkikoodia, kirjastoja ja prosesseja. Mukaan kuuluu myös oppaita, joita kehittäjät voivat käyttää ja integroida omiin sovelluksiinsa. (TechTarget, 2022d)

API on valmiiksi kehitettyä koodia, jonka avulla kaksi osapuolta ohjelmistossa voivat keskustella toistensa kanssa. (TechTarget, 2022a) Esimerkiksi jos käyttäjä haluaa hakea tietoa, hän lähettää kutsun, jonka API ottaa vastaan ja välittää palvelimelle (Kuva 4). Palvelin sitten lähettää vastauksen, jonka API ottaa vastaan ja välittää käyttäjälle.

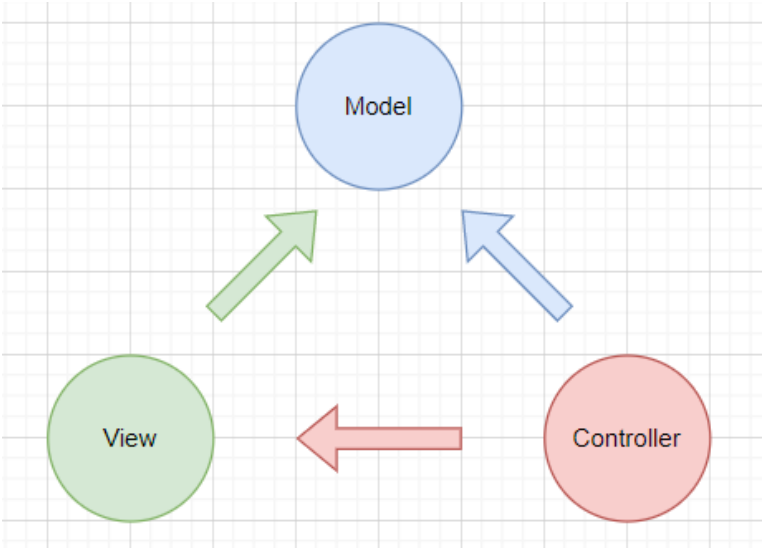


Kuva 4 API havainnollistaminen.

Kehittäjien keskuudessa suosituin rajapinta on REST API (Representational State Transfer). REST on ohjelmistoarkkitehtuuri, joka määrittelee ne ehdot, kuinka API:n tulisi toimia. Alun perin REST luotiin ohjeistukseksi, jolla hallitaan kommunikaatioita monimutkaisessa verkossa. REST-pohjaista arkkitehtuuria voidaan käyttää tukemaan tehokasta ja luotettavaa kommunikaatiota suuressa mittakaavassa. Sitä on helppo toteuttaa ja muokata, mikä tuo näkyvyyttä ja monialustaista kannettavuutta mihin tahansa API-järjestelmään. (Amazon Web Services n.d.a)

3.2.3 ASP.NET MVC

ASP.NET MVC on ohjelmistokehys, jota käytetään selainsovellusten rakentamiseen hyödyntäen MVC (model – view - controller) -suunnittelumallia. Malli toimii siten, että sovellus jaetaan kolmeen pääosaan (Kuva 5). Nämä osat ovat mallit (engl. models), näkymät (engl. views) ja ohjaimet (engl. controllers). Mallin tehtävän on edustaa sovelluksen tilaa ja logiikkaa tai toimintoja, jotka sen tulisi suorittaa. Näkymän tehtävänä on esittää käyttäjälle sisältöä käyttöliittymän kautta. Ohjaimen tehtävänä on käsitellä käyttäjän ja sovelluksen vuorovaikutusta, työskennellä mallin kanssa ja valita mikä näkymä esitetään käyttäjälle. (Microsoft, 2022d)



Kuva 5 MVC-malli.

3.2.4 Entity Framework Core – ORM (Object relation mapper)

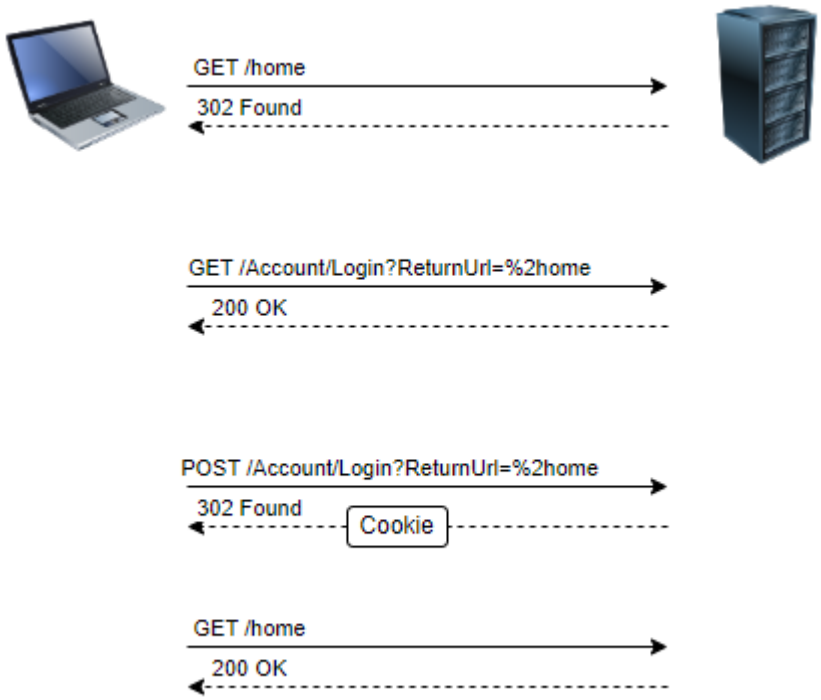
Entity Framework Core (EF Core) on avoimen lähdekoodin, monialustainen versio Entity Framework -datan käsittelyteknologiasta. EF Core on ORM (Object-Relational Mapper) työkalu, jolla kehittäjät voivat työskennellä relaatiotietokannan kanssa, hyödyntäen .NET Core Frameworkia. ORM tekniikalla kehittäjä voi työskennellä tietokannan kanssa käyttäen olio-ohjelmointikieltä, mieluummin kuin SQL-komentoja. Tämä tekee kehittäjän työstä tehokkaampaa, helpottaa koodin ylläpidettävyyttä ja sovelluksen suorituskykyä ja lisää turvallisuutta. (Learn Entity Framework Core, 2023)

3.2.5 Forms-autentikaatio

Laitepassi-sovellus käyttää Forms-autentikaatiota, tunnistamaan sovellukseen kirjautuvat käyttäjät. Formsin autentikaatio hyödyntää HTML pohjaista lomaketta lähettääkseen käyttäjän tunnistetiedot palvelimelle, taustalla hyödyntäen Active Directorya ja .NET:in Active Directory -kirjastoja. Käytännössä autentikaatio tapahtuu AD:ta vasten, räätälöidyillä koodikyselyillä. Forms-autentikointi sopii vain web-rajapinnoille, jotka kutsutaan www-

sovelluksesta, jotta käyttäjä voi olla vuorovaikutuksessa HTML-lomakkeen kanssa.

Forms-autentikaatio toimii seuraavalla tavalla (Kuva 6): Ensin käyttäjä pyytää resurssia, joka vaatii käyttäjän tunnistautumisen. Jos käyttäjä ei ole tunnistautunut, palvelin palauttaa Löydetty-vastauksen (engl. Found) ja ohjaa käyttäjän kirjautumissivulle. Tämän jälkeen käyttäjä syöttää tunnistetietonsa ja lähettää tunnistautumislomakkeen. Palvelin palauttaa toisen HTTP 302 (Löydetty)-vastauksen, joka ohjaa käyttäjän takaisin alkuperäiseen URI-osoitteeseen. Tämä vastaus sisältää tunnistautumisevästeen. Asiakas sitten pyytää resurssia uudelleen. Tämä pyyntö sisältää tunnistautumisevästeen, jolloin palvelin myöntää pyynnön käyttäjälle. (Microsoft, 2022c)



Kuva 6 Forms autentikaation havainnollistaminen.

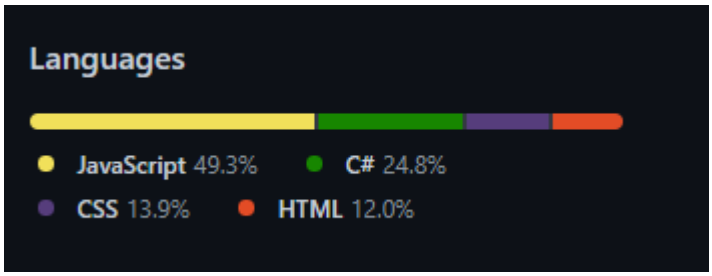
URI (Uniform Resource Identifier) on merkkijono, joka tunnistaa loogisen tai fyysisen resurssin, internetiin liittyen. URI erottaa yhden resurssin toisesta. URI:t mahdollistavat internet-protokollien välisen vuorovaikutuksen resurssien

välillä. URI:n sisältämät merkkijonot toimivat tunnisteina, kuten järjestelmänimen tai tiedostopolun. (TechTarget, 2021d)

Kuten aiemmin mainittiin, Forms-autentikaatio käyttää tunnistautumisevästeitä (authentication cookies). Evästeet ovat tietoja, joita käytetään käyttäjän ja hänen mieltymystensä tunnistamiseen. Selain palauttaa evästeen joka kerta, kun palvelimelta pyydetään sivua. Erityiset evästeet, kuten HTTP-evästeet, käytetään evästeperusteisen tunnistautumisen suorittamiseen käyttäjän istunnon ylläpitämiseksi. (Loginradius n.d)

3.3 Ohjelmointikieli

Sovellus on rakennettu neljän eri ohjelmointikielen avulla: JavaScript, C#, CSS ja HTML. Kuva 7 on otettu sovelluksen versionhallinnasta (GitHub), jossa näkyy kuinka paljon mitäkin kieltä on sovelluksessa käytetty prosentuaalisesti. JavaScript, HTML ja CSS ovat käytössä sovelluksen frontend puolella, kun taas C# käytetään sovelluksen backend puolella. Frontend, eli selainpuoli tarkoittaa koodia, joka ajetaan selaimessa. Kaikki mitä käyttäjä näkee käyttäessään sovellusta, on selainpuolta. Backend, eli palvelinpuoli tarkoittaa taas sitä koodia, joka ajetaan palvelimella, eikä selaimella.



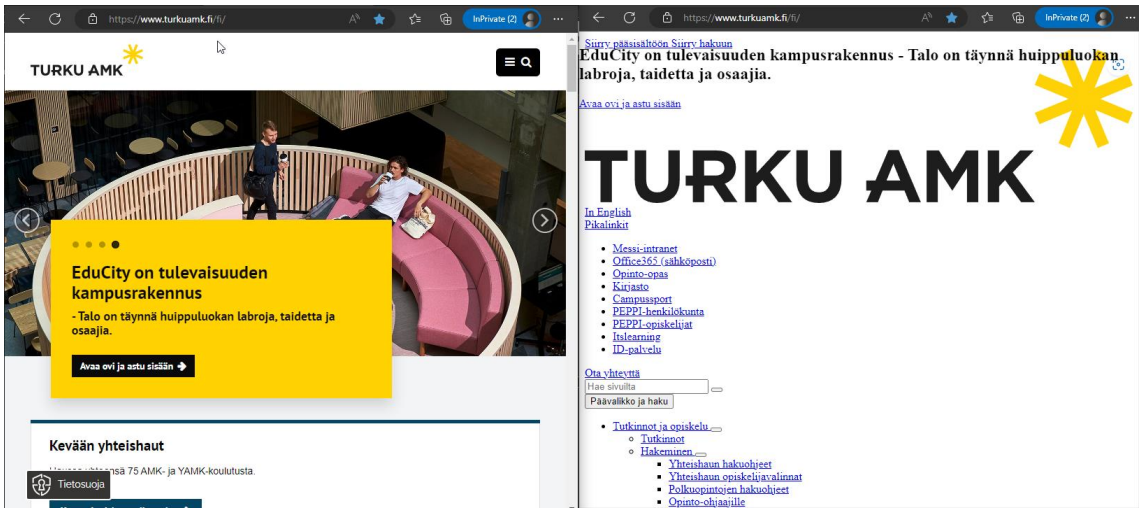
Kuva 7 Sovelluksen koodikielien käyttö.

JavaScript (JS) on kehittäjien kesken yksi tunnetuimmista komentosarjakielistä. Eniten JS on käytössä selainsivujen komentosarjoissa, mutta sitä käytetään myös ei-selainympäristöissä, kuten palvelinpuolien rakentamisessa. (MDN Web

Docs, 2023d) Esimerkkitapaus JavaScriptin käytöstä verkkosivujen parissa, on sivulla olevien painikkeiden funktio. Mitä kukin painike tekee, kun niitä painaa?

C# on moderni, oliosuuntautunut ja tyyppiturvallinen ohjelmointikieli. C# on luonnollinen kieli ohjelmistokomponenttien luomiseen ja käyttämiseen. Kielen avulla ohjelmistokehittäjät voivat rakentaa monenlaisia sovelluksia, jotka toimivat .NET:ssä, johon C# pääosin käytetäänkin. C# -kieli pohjautuu C perheen kielistä ja on tuttu vähintään C, C++, Java ja Javascript -kieliä käyttäville ohjelmoijille. (Microsoft, 2023a)

CSS (Cascading Style Sheets) on tyylitaulukkokieli, jota käytetään kuvaamaan kirjoitetun HTML-koodin esitystapaa. CSS ei ole missään nimessä pakollinen, mutta sen lisääminen helpottaa HTML-koodin luettavuutta. (MDN Web Docs, 2023a) Kuvassa 8 on kaksi esimerkkiä Turun ammattikorkeakoulun kotisivuista, vasemmalla CSS on käytössä ja oikealla CSS ei olisi käytössä.



Kuva 8 Vasemmalla CSS käytössä, oikealla CSS ei käytössä.

HTML (Hypertext Markup Language) on verkkosivun rakentamisen alkeellisin osa. Sillä määritellään sivun sisällön merkitys ja rakenne. Tähän mukaan yhdistetään sitten CSS, jota käytetään kuvaamaan ulkoasua, ja JS, jota käytetään kuvaamaan eri elementtien toimintoja. ”HyperText”, tai ”Hyperteksti” viittaa linkkeihin, jotka yhdistävät verkkosivut toisiinsa. ”Markup”, tai ”merkinnät” ovat elementtejä koodissa, joilla voidaan esittää tekstiä, kuvia tai muuta sisältöä

selaimessa. (MDN Web Docs, 2023b) Kuvassa 9 on pieni esimerkki HTML-koodista ja eri merkinnöistä.

Esimerkki

```
<!DOCTYPE html>
<html>
<head>
<title>Sivun Pääotsikko</title>
</head>
<body>

<h1>Tämä on otsikko</h1>
<p>Tämä on kappale.</p>

</body>
</html>
```

Kuva 9 HTML esimerkki.

3.4 Käyttöliittymäkirjastot

JavaScript on yleisesti käytetty käyttöliittymien (UI, eli user interface) kehityksessä. Javascriptin avulla, kehittäjä voi luoda dynaamisia, responsiivisia (mukautuvia) ja interaktiivisia (käyttäjä on vuorovaikutuksessa sisällön kanssa) sivuja, joilla pystytään parantamaan käyttökokemusta. Javascriptille on kehitetty monelaisia kirjastoja. Yksi näistä on jQuery. jQuery on kirjasto, joka tekee muun muassa HTML-dokumenttien läpikäynnistä ja käsittelystä, tapahtumien käsittelystä, animaatioista, paljon yksinkertaisempaa helppokäyttöisellä. jQuery hyödyntää sen API:ia, joka kykenee toimimaan useilla eri selaimilla. (jQuery n.d.)

Laitepassin ulkonäkö on rakennettu hyödyntäen Twitter Bootstrap ohjelmistokehystä. Bootstrap on ilmainen, avoimen lähdekoodin ja selainpuolen kehitystä varten oleva ohjelmistokehys. Bootstrapillä kehitetään verkkosivuja ja sovelluksia. Se on oiva ohjelmistokehys, jossa kehittäjä haluaa ensisijaisena laitteena olevan jonkinlainen mobiililaite, kun puhelin tai tabletti. Bootstrapin avulla kehittäjät voivat kehittää verkkosivuja paljon nopeammin, kun he voivat

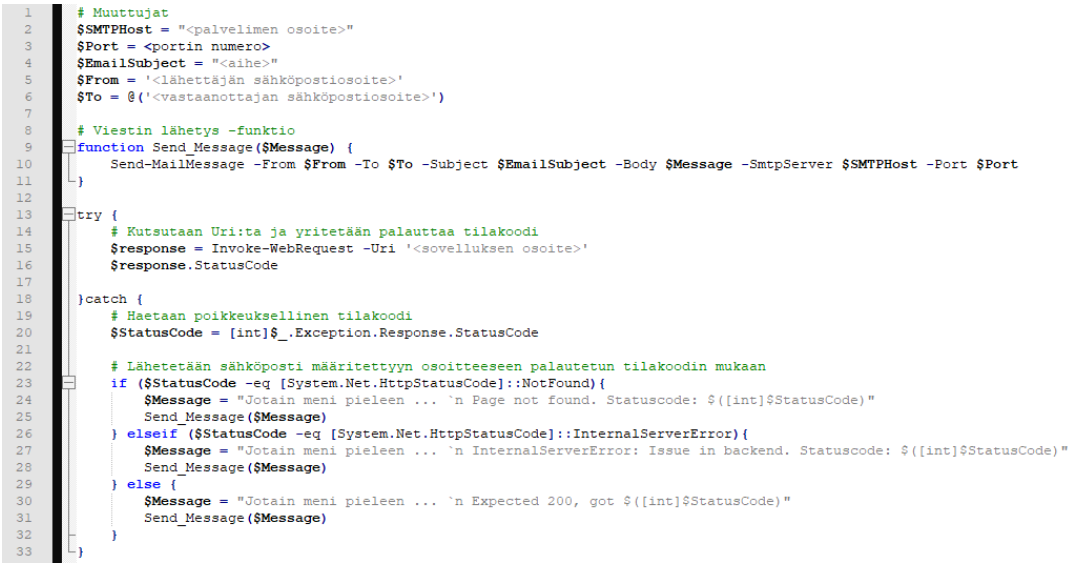
hyödyntää sen valmiiksi rakennettuja ominaisuuksia ja komponentteja, jotka pitää vain lisätä koodiin. Tällöin kehittäjän ei tarvitse murehtia yksinkertaisista komennoista tai funktioista. (TechTarget, 2022b) Bootstrapin omilta sivuilta kehittäjä voi löytää dokumentaation, sekä eri komponentteja ja funktioita, joita pystyy sitten hyödyntämään selkeän ohjeistuksen avulla.

3.5 Tietokanta

Sovellus käyttää relaatiotietokantaa datan tallentamiseen. Relaatiokanta on tiedon kokoelma, joka organisoi tiedon ennalta määritettyihin relaatioihin. Relaatiot ovat loogisia yhteyksiä eri taulujen välillä, joka muodostuu näiden taulukoiden välisen vuorovaikutuksen perusteella. Tieto tallentuu yhteen tai useampaa tauluun, joka koostuu sarakkeista ja riveistä. Tämä helpottaa näkemään ja ymmärtämään, miten eri tietorakenteet liittyvät toisiinsa. (Google Cloud n.d.b)

3.6 Tukisovellukset

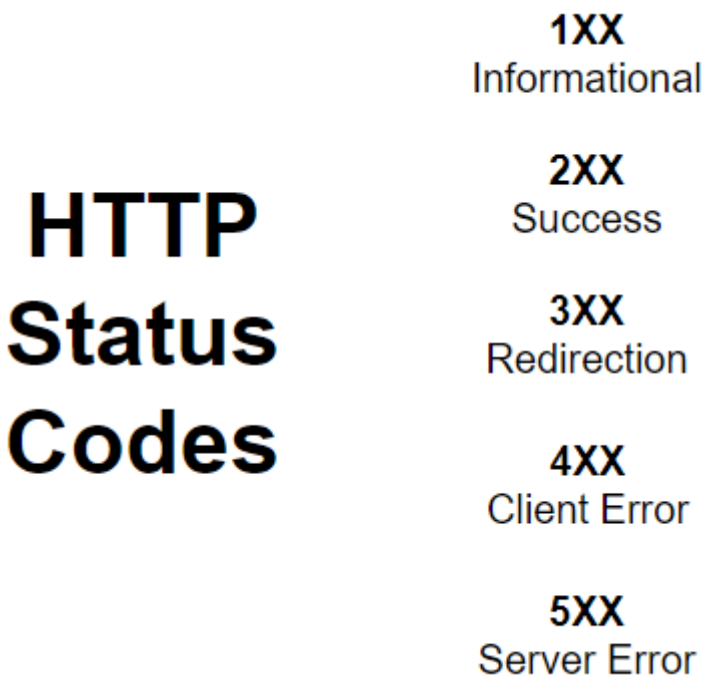
Sovelluksen ylläpidon tueksi on ohjelmoitu PowerShell-komentosarjaohjelma (Kuva 10), joka suoritetaan päivittäin 30 minuutin välein Windowsin omaa Tehtävien ajoitus -ohjelmalla. Komentosarjaohjelman kielenä PowerShell on yleisesti käytetty automatisoimaan järjestelmien hallintaa. (Microsoft, 2022i) Komentosarjaohjelman tarkoitus on saada yhteys, erikseen määritettyyn verkkosivuun. Mikäli komentosarjaohjelma ei saa tämän yhteydenoton kautta tiettyä vastausta, seuraa tietynlaisia toimenpiteitä.



```powershell
# Muuttujat
$SMTPHost = "<palvelimen osoite>"
$Port = <portin numero>
$EmailSubject = "<aihe>"
$From = '<lähettäjän sähköpostiosoite>'
$To = @('<vastaanottajan sähköpostiosoite>')

# Viestin lähetys -funktio
function Send_Message($Message) {
    Send-MailMessage -From $From -To $To -Subject $EmailSubject -Body $Message -SmtpServer $SMTPHost -Port $Port
}

try {
    # Kutsutaan Uri:ta ja yritetään palauttaa tilakoodi
    $response = Invoke-WebRequest -Uri '<sovelluksen osoite>'
    $response.StatusCode

}catch {
    # Haetaan poikkeuksellinen tilakoodi
    $StatusCode = [int]$_.Exception.Response.StatusCode

    # Lähetetään sähköposti määritettyyn osoitteeseen palautetun tilakoodin mukaan
    if ($StatusCode -eq [System.Net.HttpStatusCode]::NotFound){
        $Message = "Jotain meni pieleen ... `n Page not found. Statuscode: $([int]$StatusCode)"
        Send_Message($Message)
    } elseif ($StatusCode -eq [System.Net.HttpStatusCode]::InternalServerError){
        $Message = "Jotain meni pieleen ... `n InternalServerError: Issue in backend. Statuscode: $([int]$StatusCode)"
        Send_Message($Message)
    } else {
        $Message = "Jotain meni pieleen ... `n Expected 200, got $([int]$StatusCode)"
        Send_Message($Message)
    }
}
```

Kuva 10 Ping -komentosarjaohjelma.

Koodi voidaan jakaa kolmeen osaan: muuttujat (rivit 2–6), viestin lähetys -funktio (rivit 9-10) ja try…catch -lause (rivit 13-33). Ohjelma suorittaa ensin try-osion, yrittämällä kutsua sovelluksen kirjautumissivua ja palauttaa sen HTTP-vastauksen tilakoodin. Jos tämä ei onnistu, ohjelma antaa poikkeuksen ja suorittaa catch-osion. HTTP-vastauksen (Hypertext Transfer Protocol) tilakoodit osoittavat, jos tietty HTTP-pyyntö on pystytty suorittamaan onnistuneesti. Nämä vastaukset voidaan jakaa viiteen luokkaan, kuvan 11 tapaan. Luvulla 1 alkava koodi tarkoittaa, että pyyntö on vastaanotettu ja voidaan jatkaa. Luvulla 2 alkava koodi tarkoittaa, että pyyntö on vastaanotettu, ymmärretty ja käsitelty. Luvulla 3 alkava koodi tarkoittaa, että jatkotoimenpiteitä tarvitaan pyynnön suorittamiseksi. Luvulla 4 alkava koodi tarkoittaa, että pyyntöä ei voitu suorittaa käyttäjän päässä johtuvan virheen vuoksi. Luvulla 5 alkava koodi tarkoittaa, että pyyntöä ei voitu suorittaa palvelimen päässä johtuvan virheen vuoksi. (MDN Web Docs, 2023c)



Kuva 11 HTTP-tilakoodit.

Jos ohjelman try-osio onnistuu, sovelluksen odotetaan palauttavan 2xx (tarkemmin 200) tilakoodi. Mikäli tulee poikkeus, haetaan catch-osiossa ensin poikkeuksen tilakoodi, luodaan virheviesti ($Message -muuttuja), jossa ilmoitetaan poikkeuksen tilakoodi ja kutsutaan Send_Message-funktiota. Kyseinen funktio kutsuu Send-MailMessage-komentoa, joka käyttää koodissa määritettyjä muuttujia. Muuttujat ovat: lähettäjä, vastaanottaja, aihe, itse viesti, palvelin, joka lähettää viestin ja palvelimen portti.

3.7 Sovelluskehityksessä käytettävät työkalut

Sovelluksen kehitystiimi kehittää sovellusta paikallisessa kehitysympäristössä, omilla työasemillaan, Visual Studio kehitysympäristössä. Sovelluksen koodia varastoidaan 2M-IT omassa versionhallinnassa, GitHub-sivustolla.

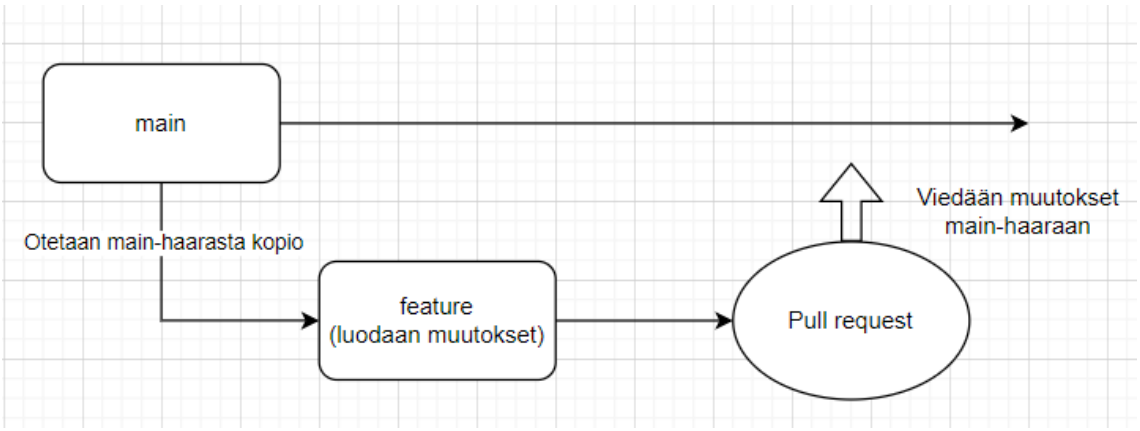
3.7.1 Visual Studio -ohjelmointiympäristö

Kaikki sovelluskehitys toteutetaan Microsoftin Visual Studio integroidussa kehitysympäristössä eli IDE:ssä (Integrated Development Enviroment). IDE on ohjelmistoympäristö, jolla ohjelmoija kykenee suunnittelemaan ja toteuttamaan eri ohjelmistoja. Se lisää tuottavuutta yhdistämällä useita ominaisuuksia, kuten ohjelmistojen muokkaus, rakentaminen, testaus ja pakkaus, kaikki yhdessä helppokäyttöisessä sovelluksessa. Ohjelmistokehittäjät käyttävät IDE:ä helpottaakseen työskentelyään, siinä missä kirjailijat saattaisivat käyttää tekstin käsittelijää tai kirjanpitäjät laskentataulukkoa. (Amazon Web Service n.d.b) Visual Studio on kattavin IDE .NET- ja C++ -kehittäjille Windowsissa. Ympäristö on täynnä erilaisia työkaluja ja ominaisuuksia, jotka nostavat ja tehostavat jokaista vaihetta ohjelmistokehityksen parissa. (Microsoft, 2023m)

3.7.2 GitHub-verkkosivusto

GitHub toimii sovelluksen versionhallintana 2M-IT:n hallinnoimana. GitHub-verkkosivusto pohjautuu Git-versionhallintajärjestelmään. Gitin idea on tarjota kehittäjille helppo ja nopea tapa hallita ohjelmistojensa lähdekoodia, erityisesti avoimen lähdekoodin projekteissa. GitHub tarjoaa ilmaisen käyttöliittymän Git-versionhallinnan hallintaan ja tallennuskapasiteetin Gitillä hallittuihin tietovarastoihin. GitHubin tarjoamat ilmaiset palvelut, kuten virheiden seuranta, kehitystoiveet ja tehtävienhallinta, ovat tehneet siitä välttämättömän työkalun ohjelmistokehittäjille. (Wikipedia, 2023)

Sovelluksen GitHub-koodivarasto koostuu useammasta haarasta, jolla voidaan eristää kehitystyö eri osa-alueisiin ilman, että se vaikuttaa sovelluksen muihin haaroihin. Kun haluttu kehitystyö on tietyssä haarassa saatu päätökseen, luodaan “pull request”, jotta voidaan yhdistää kehitetyn haaran muutokset päähaaraan. (GitHub, 2023) Alla olevassa kuvassa (Kuva 12) on esimerkki, jossa tuodaan muutos “feature” -haarasta “main”-haaraan eli päähaaraan.



Kuva 12 GitHub haarojen Pull Request -toiminto.

Päähaara on osa varastoa, josta kaikki kehityshaarat on alun perin rakennettu. Feature-haara on kopio päähaarasta, jonka päälle on tehty päivityksiä, kuten uusien ominaisuuksien, tai olemassa olevan ominaisuuden lisäämistä. On tyypillistä asettaa muunneltu koodi GitHubin sisällä toisen kehittäjän arvioitavaksi ja hyväksyttäväksi. Kun työ on arvioitu ja hyväksytty, se päivitetään varaston päähaaraan.

3.7.3 SQL Server Management Studio

SQL Server Management Studio (SSMS) on integroitu ympäristö SQL-infrastruktuurin hallintaan, SQL Serveristä Azure SQL -tietokantaan. SSMS antaa käyttäjälleen työkalut SQL Serverin ja tietokantojen määritykseen, monitorointiin ja hallintaan. SSMS:ää käytetään sovelluksien käyttämien tietotasojen komponenttien asentamiseen, seuraamiseen ja päivittämiseen, sekä kyselyjen ja skriptien luomiseen.

SSMS:llä voidaan luoda kyselyjä, suunnitella ja hallita tietokantoja ja tietovarastoja, olivat ne sitten paikallisessa tietokoneessa tai pilvessä. (Microsoft, 2023g)

4 Kehityskohteet ja ongelmat

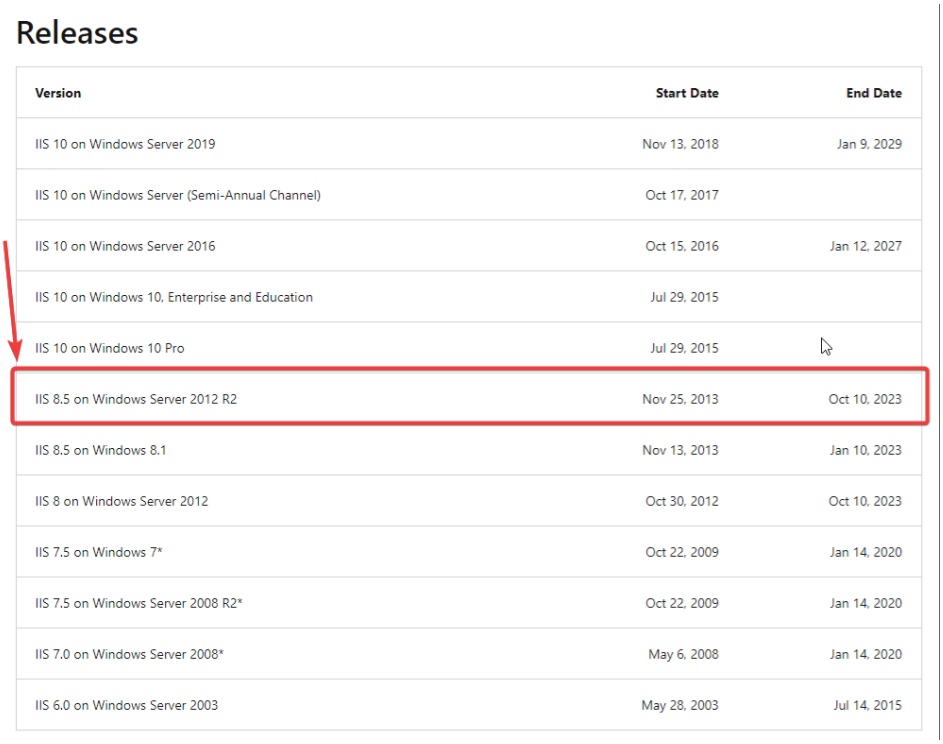
Tässä kappaleessa käsitellään vanhan sovelluksen kehityskohteita ja niiden ongelmia.

4.1 Tekniset muutokset

Sovelluksen käytössä on virtuaalipalvelimia. On kaksi sovelluspalvelinta, jotka ovat Windows Server 2012 R2 -palvelimia ja kaksi tietokantapalvelinta, jotka ovat Windows Server 2016 -palvelimia. Näillä sovelluspalvelimille on asennettu IIS 8.5, jossa WWW-sovelluksia ajetaan. Tietokantapalvelimille on asennettu SQL Server 2016, jota sovellukset käyttävät.

Sovelluksen nykyinen ympäristö 2M-IT:n omissa konesaleissa on vanhenemassa. Ratkaisuna on siirtää sovellus kokonaan uuteen ympäristöön. Laitepassi-sovelluksen tuki nyky-ympäristön palvelimille päättyy. Kehitystiimillä on kaksi vaihtoehtoa: pystyttää uusia virtuaalipalvelimia ja sitä kautta alkaa siirtämään nykysovelluksia niille. Toisen vaihtoehtona on siirtyä pilviympäristöön ja siirtyä ja ajaa kehitystä sinne. Jälkimmäinen vaihtoehto valitaan ja ratkaisuun vaikuttavia tekijöitä ovat esimerkiksi siirrytään omien konesalien virtuaalipalvelimilta PaaS -komponenttien käyttämiseen ja autentikaation modernisointi hyvinvointialueen laajuisesti, käyttämään Azure AD:ta.

Windows Server 2012 tuki tulee päättymään 10.10.2023 jälkeen (Kuva 13). Tuen päätyttyä tuote lakkaa saamasta tietoturvapäivityksiä, virheenkorjauksia, teknistä tukea tai online-teknisen sisällön päivityksiä. Microsoft kuitenkin tarjoaa siirtymisohjeita pilvipalveluille ja paikallisille ratkaisuille. (Microsoft, 2023n)



Releases

Version	Start Date	End Date
IIS 10 on Windows Server 2019	Nov 13, 2018	Jan 9, 2029
IIS 10 on Windows Server (Semi-Annual Channel)	Oct 17, 2017	
IIS 10 on Windows Server 2016	Oct 15, 2016	Jan 12, 2027
IIS 10 on Windows 10, Enterprise and Education	Jul 29, 2015	
IIS 10 on Windows 10 Pro	Jul 29, 2015	
IIS 8.5 on Windows Server 2012 R2	Nov 25, 2013	Oct 10, 2023
IIS 8.5 on Windows 8.1	Nov 13, 2013	Jan 10, 2023
IIS 8 on Windows Server 2012	Oct 30, 2012	Oct 10, 2023
IIS 7.5 on Windows 7*	Oct 22, 2009	Jan 14, 2020
IIS 7.5 on Windows Server 2008 R2*	Oct 22, 2009	Jan 14, 2020
IIS 7.0 on Windows Server 2008*	May 6, 2008	Jan 14, 2020
IIS 6.0 on Windows Server 2003	May 28, 2003	Jul 14, 2015

Kuva 13 Windows Server 2012 R2 elinkaari.

Koska ohjelmistot kehittyvät jatkuvasti, vanhentuneiden sovelluksien, palveluiden ja järjestelmien käyttö tuo mukanaan riskejä. Arcomedian artikkelissa mainitaan kolmesta riskistä tähän liittyen: Turvallisuuden vaarantuminen, luotettavuuden puute ja korkeammat kustannukset. (Acromedia, 2022)

Windows Serverin tietoturva tulee olemaan vaarantunut, eikä vain vanhentuneen palvelun alustalla, vaan myös muiden siihen yhdistävien alustojen osalta. Jatkuvien päivitysten saaminen on kriittistä. Ikääntynyt ja päivityksiä saamaton ohjelmisto ei tule olemaan yhtä luotettava ja riski vikojen esiintymiselle kasvaa. Vanhentuneet ohjelmiston käyttö vaatii myös suurempia kustannuskuluja, oli kyseessä sitten esimerkiksi menetetyn tiedon palauttaminen, päivitysten ja ylläpidon hallinnointia kolmannen osapuolen kautta. (Acromedia, 2022) Samalla tavoin kuin esimerkiksi auto, se vaatii huoltoa. Jos sen ylläpitoa laiminlyö, riski vioille kasvaa ja korjauskustannukset voivat nousta taivaisiin.

On muutamia vaihtoehtoja, joilla ratkoa tämä pulma. Ensimmäisenä ratkaisuna voidaan siirtyä Microsoftin Azure-pilviympäristöön. Toinen esimerkki, mikäli halutaan pysyä paikallisessa ympäristössä, on luoda kokonaan uusi palvelinympäristö, uuteen Windows Server 2022. Azure-pilviympäristön sovelluspalvelimet päivittämisestä vastaa Microsoft. Myös IIS:n ja SQL Serverin päivittäminen on Microsoftin vastuulla. Paikallisessa ympäristössä, uuden palvelinympäristön mukana tulee uusimmat tietoturvapäivitykset. Mikäli vanhaa Windows Serveriä käyttävät asiakkaat eivät kykene täyttämään tukipalveluiden määräaikaa ja heillä on esimerkiksi tilauslisenssi yrityksen sopimusohjelmassa, on olemassa kolmas vaihtoehto, jossa voidaan ostaa laajennettuja tietoturvapäivityksiä kolmeksi vuodeksi tälle vanhentuneelle palvelulle. (Microsoft, 2022h)

4.2 .NET -version vanheneminen

.NET -versiot jakautuvat kahteen julkaisutyyppiin: Pitkäaikaiseen tukeen, eli Long Term Support (LTS) ja normaaliaikaiseen tukeen, eli Standard Term Support (STS). LTS-julkaisuja tuetaan kolmen vuoden ajan alkuperäisen julkaisun jälkeen. STS-julkaisuja tuetaan kuuden kuukauden ajan edeltävän STS- tai LTS-julkaisun jälkeen. STS-julkaisuja tulee joka kahdentoista kuukauden välein, joten tukijakso on kahdeksantoista kuukautta. (Microsoft, 2023w)

Sovellus kehitettiin alun perin .NET Core 3.1 -version päälle, mutta kyseinen versio on tullut elinkaarensa päätökseen vuoden 2022 lopulla (Kuva 14). Vanhentuneet .NET-versiot saavat End of Life-tilan (EOL), joka tarkoittaa elinkaaren päätöstä. Kuten Windows Serverin kanssa, Microsoft ei enää tue kyseisiä EOL-tilan alla olevia versioita korjauksilla, päivityksillä, eikä tarjoa teknistä online-tukea.

Tuen ulkopuolella olevien .NET-versioiden käyttö saattaa vaarantaa sovelluksen, sen tiedot ja tietojenkäsittely-ympäristön. (Microsoft, 2023w)

Out of support versions

The following table lists .NET Core versions no longer supported.

Version	Original release date	Latest patch version	Patch release date	End of support
.NET 5	November 10, 2020	5.0.17	May 10, 2022	May 10, 2022
.NET Core 3.1	December 3, 2019	3.1.32	December 13, 2022	December 13, 2022
.NET Core 3.0	September 23, 2019	3.0.3	February 18, 2020	March 3, 2020
.NET Core 2.2	December 4, 2018	2.2.8	November 19, 2019	December 23, 2019
.NET Core 2.1	May 30, 2018	2.1.30	August 19, 2021	August 21, 2021
.NET Core 2.0	August 14, 2017	2.0.9	July 10, 2018	October 1, 2018
.NET Core 1.1	November 16, 2016	1.1.13	May 14, 2019	June 27, 2019
.NET Core 1.0	June 27, 2016	1.0.16	May 14, 2019	June 27, 2019

Kuva 14 Vanhentuneet .NET -versiot.

Sen lisäksi, että sovellukseen asentaisi uuden .NET -version se jouduttaisiin asentamaan myös kaikilla sovelluksen käyttämille palvelimille.

4.3 Autentikaatio

Ennen sote-uudistusta, kaikilla sairaanhoitopiireillä oli omat Active Directory -ohjelmistonsa. Laitepassin nykyinen versio käyttää taustalla Satasairaalan keskussairaalan AD:ta, joka on vain Satakunnan hyvinvointialueen keskussairaalan käyttämä AD.

Nykyinen autentikaatio on sovelluksen kehitystiimin itse ohjelmoima, 2M-IT:n omien konesalien AD-palvelua vasten. Autentikaatio -ratkaisu on täysin yhden sovelluskehittäjän räätälöimä ratkaisu. Jos ratkaisuun tulisi jonkinlainen ongelma tai tietoturvapuutteita, eikä sen kehittäjä ole tavoitettavissa, sen ratkominen voi koittautua haasteelliseksi ja sen ylläpito on muutenkin työläistä. Tiimin täytyisi osata vastata autentikaatiosta ja tämä vaatii esimerkiksi henkilöstöä ja ohjelmointitaitoa.

Laitepassin käytössä on ryhmäpohjainen autentikaatio tai toisin tunnettu GBAC (Group-Based Access Control). GBAC on valtuutuksen muoto, joka perustuu

käyttäjälle määritettyihin ryhmiin. (Opal n.d.) Käyttäjän kirjautuessa Laitepassiin, hänen tietonsa haetaan AD:sta (Active Directory).

Laitepassi-sovellus on määritetty niin, että tietoturvallisuuden varmistamiseksi sovellus näkyy vain Satasairaalan tietoliikenneverkossa. Verkkorajaus on toteutettu ensisijaisesti sen vuoksi, että käytössä on räätälöity autentikaatio ja autorisointi toteutus. Sovelluksella on oma kirjautumissivunsa, johon pääsee ilman autentikaatiota (Kuva 15). Kyseiselle sivulle on mahdollista tehdä erilaisia hyökkäyksiä, jotka voi hidastaa tai pysäyttää sovelluksen toiminnan. Käytännössä räätälöityä ratkaisua ei ole tietoturvatestattu ja todettu täysin vedenpitäväksi.

Kuva 15 Laitepassin kirjautumissivu.

4.4 Julkaisuhallinta

Sovelluksen nykyinen julkaisuhallintaprosessi sisältää useita eri vaiheita, useita erityyppisiä tiedostoja ja kansioita. Tämä prosessi on tärkeä osa ohjelmistokehitystä, sillä sen avulla varmistetaan, että lopullinen sovellus on laadukas, toimiva ja turvallinen käyttäjille.

Kun kehityksen alla ollut sovellusversio on saatu valmiiksi, siitä luodaan julkaisukansio. Virtuaalipalvelimella oleva vanha sovellusversio varmuuskopioidaan, jonka jälkeen juuri luodussa julkaisukansiossa oleva uusi

versio korvataan. Kun sovellus siirretään uudenlaiseen ympäristöön, tulee tämä vanha menetelmä poistumaan ja yksinkertaistumaan.

4.5 Ylläpito

Sovelluksen käytössä olevat virtuaali- ja tietokantapalvelimet ovat 2M-IT:n konesalitiimin palvelinylläpidossa. Sovelluskehittäjät ovat vastuussa siitä, että .NET Framework on ajan tasalla ja IIS WWW-palvelin on toimintakunnossa. Osa kehittäjistä ovat esimerkiksi käyneet IIS-palvelimen ylläpitokurssin tätä varten. Muiden palvelimien, kuten SQL Server ja Windows Server, ylläpidosta ja ajantasaisuudesta vastaa muiden 2M-IT:n tiimien henkilöstö. Ongelma nykyisessä ylläpidossa on se, että useampi tiimi vastaa eri sovelluksen komponenttien ylläpidosta. Jos tulisi kiireellinen tarve korjata jonkin ongelma tai päivittää vanha komponentin versio, sovelluksen kehitystiimi ei välttämättä ole oikeutettu tähän. He joutuvat tekemään pyynnön toiselle tiimille, joka vastaa ylläpidosta, ja odottamaan, että he ratkaisevat pyynnön.

4.6 Skaalautuvuus

Mikäli sovelluksen kanssa esiintyy ongelmia, esimerkiksi sovelluksen palvelimen muistin kanssa, joutuu kehitystiimi olemaan yhteydessä toiseen osapuoleen, joka vastaa palvelimien ylläpidosta. Tämä on ongelma, koska jos sovelluksen kanssa on kiireisiä palvelinongelmia tai suoritusongelmia, joka vaatii korjausta, joutuu kehittäjä olemaan yhteydessä palvelinta hallinnoimaan osapuoleen. Palveluiden skaalautuvuudesta ja toimintakyvystä vastaa eri 2M-IT:n tiimi, joille pitää kommunikoida ongelmatapauksista erikseen ja koordinoida asioita.

4.7 Lokitus

Laitepassi-sovellus hyödyntää kehitystiimin räätälöimiä lokitus-ratkaisuja, jotka vastaavat lokien tuottamisesta sovelluksen kirjautumisesta, autentikoinnista,

tietokannan kirjoitus- ja poisto -operaatioista, sekä sovelluksen käytössä tapahtuneista hallitsemattomista poikkeuksista tai virheistä. Sovellus kirjaa lokia tiedostoon, joka sijaitsee 2M-IT:n sovelluspalvelimella. Sovelluslokia kirjoitetaan tekstitiedostoon, jolla ei ole erillistä käyttöliittymää seurata kirjattuja havaintoja- Lokia joutuu tulkitsemaan tiedostosta yhtenä massana. Koska sovellus tulee poistumaan nykyisestä suoritusympäristöstään, lokin kirjaaminen tähän sijaintiin tulee muuttumaan tarpeettomaksi ja vaatii korvaamista tai siirtämistä toiseen paikkaan.

5 Ratkaisun modernisoinnin suunnittelu

Tässä kappaleessa käsitellään Laitepassi-sovelluksen modernisoinnin suunnittelua Azure-pilviympäristöön. Kappaleessa esitellään Azure-pilvipalvelua ja sen tarjoamia palveluja, joilla ratkaistaan Laitepassi-sovellus kohtaamia ongelmia ja joita se tulee jatkossa hyödyntämään.

5.1 Pilvipalveluympäristö

Pilvipalveluilla tai pilviteknologialla viitataan tietokoneiden resursseihin verkon välityksellä, jotka vastaavat sen käyttäjän tarpeita. Tähän kuuluu palvelimet, tallennustila, tietokannat, ohjelmistot ja tietoverkot. Pilviteknologian avulla eri yritykset ja yksityishenkilöt saavat käyttöönsä nämä resurssit etänä, ilman että heidän täytyy investoida omaan infrastruktuuriinsa tai laitteistoonsa. Tämä tuo mukanaan useita etuja, kuten kustannussäästöjä, skaalautuvuutta, joustavuutta ja turvallisuutta. Lisäksi pilvipalvelut mahdollistavat helpon yhteistyön, tietojen varmuuskopioinnin ja katastrofien varalle valmistautumisen sekä kyvyn nopeasti ottaa käyttöön uusia sovelluksia ja palveluja.

Pilvipalvelut voidaan jakaa kolmeen eri malliin: IaaS (infrastructure as a service), Paas (platform as a service) ja SaaS (software as a service). IaaS, eli infrastruktuuri palveluna on tarpeen mukaan saatavilla oleva käyttö tietokoneiden laskentaresursseihin, tallennustilaan, verkkoyhteyksiin ja virtualisointiin. PaaS, eli sovellusalusta palveluna viittaa laitteisto- tai ohjelmisto-resursseihin, joita tarvitaan pilvessä toimivien sovelluksien kehitykseen. SaaS, eli ohjelmisto palveluna tarkoittaa koko sovelluspinon tarjoamista pilvipalveluna, mukaan lukien alustan alapuolisen infrastruktuurin ylläpitoa ja hallintaa aina sovellusohjelmistoon asti. (Google Cloud n.d.a)

Uusi versio Laitepassista tulee toimimaan Microsoftin Azure pilvipalveluita hyödyntäen. Azure on pilvialusta, jonka tarkoitus on yksinkertaistaa modernien sovellusten rakennusprosessia. Halusi kehittäjä sovelluksen toimia kokonaisuudessaan Azuressa, tai laajentaa paikallisia sovelluksia Azure-

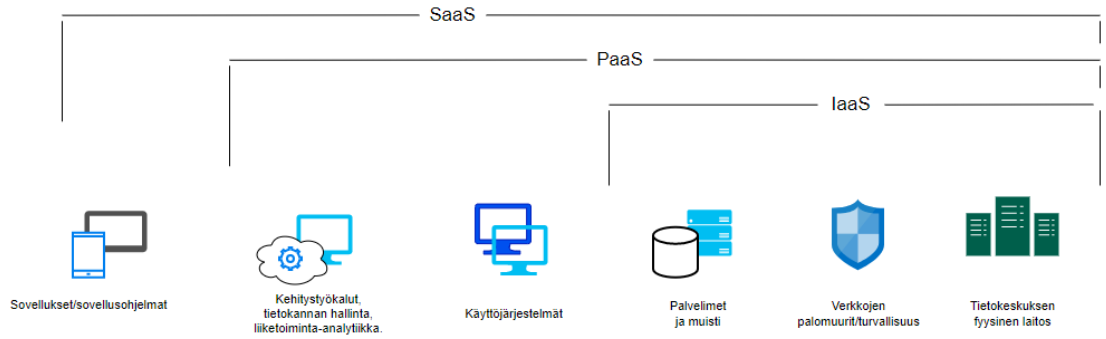
palveluissa, Azure auttaa kehittäjää luomaan sovelluksia, jotka ovat skaalautuvia, luotettavia ja ylläpidettäviä. (Microsoft, 2022b)

Syy miksi Laitepassille valittiin juuri Azure, on koska sairaanhoitopiirit käyttävät jo valmiiksi Microsoft Azurea.

5.2 Platform as a Service

Jatkossa sovellus tulee hyödyntämään ainoastaan PaaS-komponentteja. PaaS-malli tarjoaa kehitys- ja käyttöympäristöjä pilvessä, jonka resurssit mahdollistavat yksinkertaisten pilvipohjaisten sovellusten toimittamisen aina kehittämiseen asti. Tarvittavat resurssit hankitaan pilvipalveluntarjoajalta pay-as-you-go-periaatteella, eli maksetaan vain siitä, mitä käytetään, tai kuukausipainotteisella hinnoittelulla. Hankittuihin resursseihin päästään käsiksi turvallisen Internet-yhteyden kautta.

Kuten IaaS, PaaS sisältää infrastruktuurin lisäksi myös keskitason ohjelmiston, kehitystyökalut, liiketoiminta-analytiikka-palvelut, tietokannan hallintajärjestelmät ja paljon muuta (Kuva 16). PaaS on suunniteltu tukemaan koko verkkosovelluksen elinkaarta: sovelluksen rakentaminen, sen testaus, käyttöönotto, hallinta ja päivitys. (Azure n.d.)



Kuva 16 SaaS, PaaS ja IaaS.

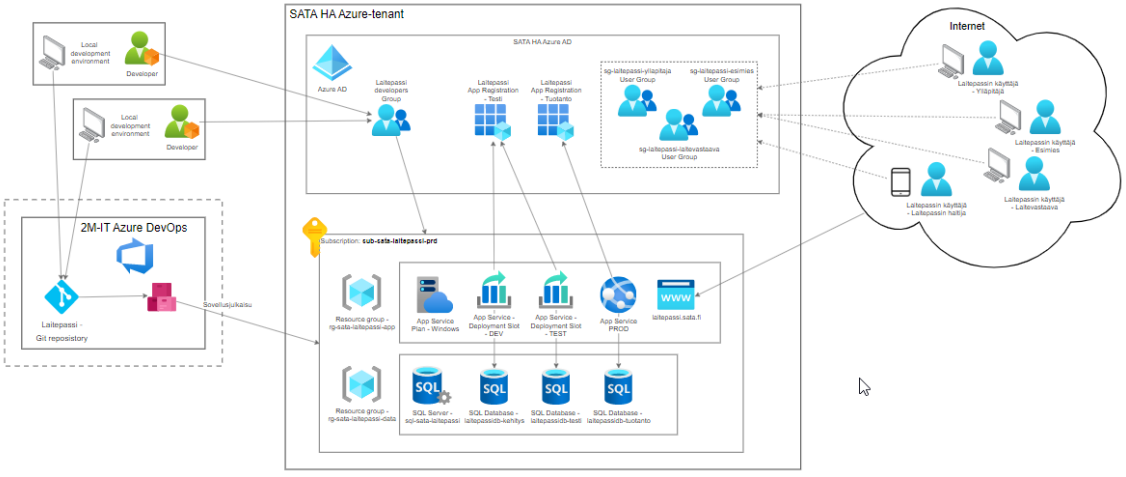
Aikaisemmin sovelluksen käytössä olleet Windows Server 2012 ja IIS WWW-palvelin, poistuvat käytöstä ja ne tullaan korvaamaan Azuren PaaS -palveluilla.

Nämä palvelut ovat Azure App Service, Azure SQL server ja Application Insights. Azuren App Service on HTTP-pohjainen palvelu, joka mahdollistaa verkkosovellusten, REST-rajapintojen toteutuksen. Kehityksessä voi käyttää haluamaansa ohjelmointikieltä. Kehitetyt sovellukset voidaan suorittaa ja skaalata vaivatta manuaalisesti ja automaattisesti, sekä Windows-, että Linux-ympäristöissä.

App Service tuo Microsoft Azuren tehon sovellukseesi, kuten tietoturvan, kuormantasauksen, automaattisen skaalauksen ja hallinnan. App Servicen avulla maksat vain niistä resursseista, joita käytät. (Microsoft, 2023b)

5.3 Uusi tekninen ympäristö

Laitepassi-sovelluksen tekninen arkkitehtuuri tulee muuttumaan (Kuva 17). Modernisoitu versio Laitepassista pystytetään Azureen, 2M-IT:n pilvipalvelu-tenantiin. Sen lisäksi, että kehittäjät tarvitsevat oikeudet toimia tämän tenantin sisällä, pitää tilata itse sovellusta varten Azure Subscription. Global adminit luovat kehittäjille tarvittavat Contributor-oikeudet toimimaan Azure-ympäristön sisällä.



Kuva 17 Laitepassin Azure arkkitehtuurikuvaus.

Tenant tarkoittaa yksittäistä omistettua ja luotettavaa Azure Active Directory (AD) -esiintymää. joka luodaan automaattisesti, kun rekisteröidytään Microsoftin

pilvipalvelutilaukseen. Tenant edustaa yhtä organisaatiota, identiteettiä tai henkilöä.

Azure AD tenant tarjoaa paikan käyttäjien, ryhmien ja niiden käyttöoikeuksien hallintaan Azure AD:ssa julkaistuille sovelluksille. Azure Active Directoryä voidaan käyttää myös käyttöoikeuksien hallintaan. Azure AD:ta voidaan käyttää myös käyttää pääsyn hallintaan moniin muihin Azure AD:lla rekisteröityihin kolmannen osapuolen sovelluksiin. Jotta sovelluksen käyttöoikeuksia voidaan hallita Azure AD:n avulla, sovellus on rekisteröitävä Azure Active Directoryyn. (Azure Training Aeries, 2022)

Azure Subscription on palvelu, joka mahdollistaa resurssien luonnin Azure-pilvipalveluiden käyttöön. Palvelu toimii yhtenäisenä laskutusyksikkönä Azure-resursseille. Azure Subscription on sidoksissa yhteen vastuuhenkilöön, jonka omistuksee Subscription luodaan ja jota käytetään myös laskutukseen. (Javatpoint n.d.)

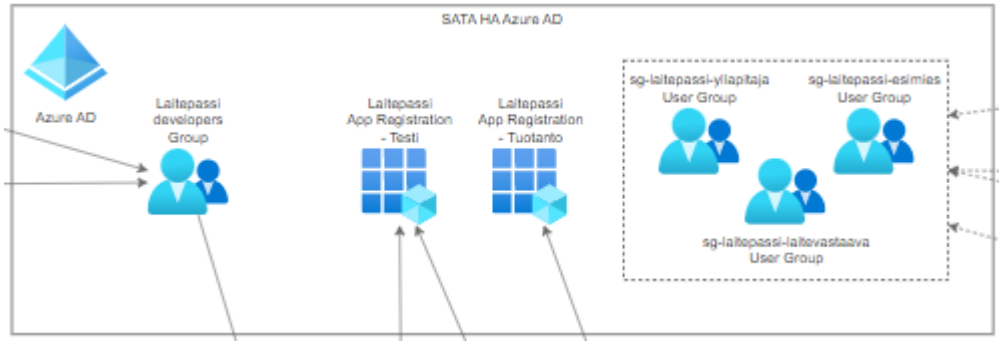
5.3.1 Azure tenant

Azure tenantin sisältö voidaan jakaa kahteen osaan: Azure AD (Kuva 18) ja Azure Subscriptioniin (Kuva 19).

Azure AD koostuu käyttäjistä, käyttäjäryhmistä ja Azure App Registrationeista eli sovellusidentiteeteistä. Azure AD käyttäjäryhmien avulla voidaan hallita käyttäjiä, jotka tarvitsevat samanlaisen pääsyn ja oikeudet resursseihin, kuten mahdollisesti rajoitettuihin sovelluksiin ja palveluihin. (Microsoft, 2023i) Laitepassi-sovelluksen tapauksessa rooleilla määritellään, mitä kirjautuva käyttäjä näkee ja pystyy tekemään. Käyttäjäryhmät voidaan luoda suoraan Azure AD:ssa tai ne voidaan synkronoida omien konesalien luotuja ryhmiä vasten ja AD:n käyttäjiä voidaan lisätä näihin ryhmiin.

Azure App Registrations, eli sovellusidentiteetit on suoritettava, jotta identiteetin ja käyttöoikeuksien hallinnointitehtävät voidaan siirtää Azure AD:lle.

Sovellusidentiteetit ovat Azure AD:n komponentteja, joiden avulla identiteetin ja käyttöoikeuksien hallinta on siirretty. (Microsoft, 2023c)

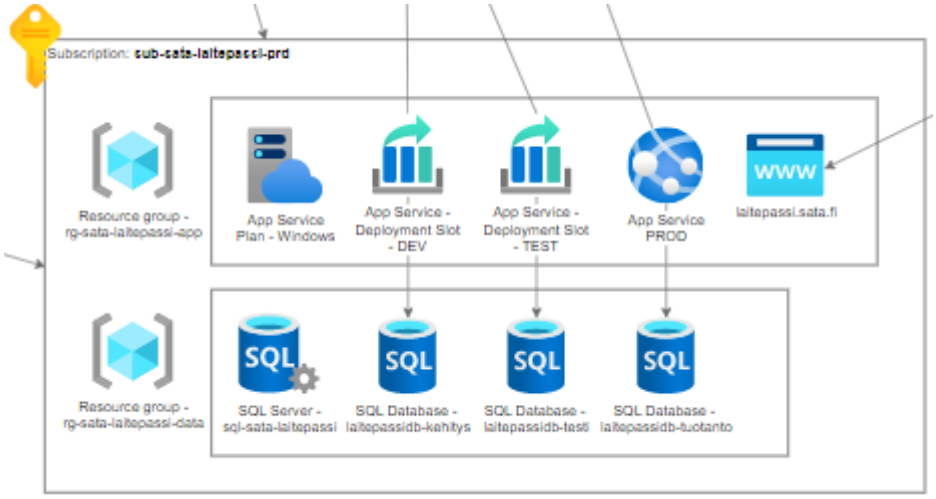


Kuva 18 Azure AD sisältö tenantissa.

App Registrationin sisällä on kaksi työhön tärkeää hallinta -ominaisuutta: Token configuration ja API permissions.

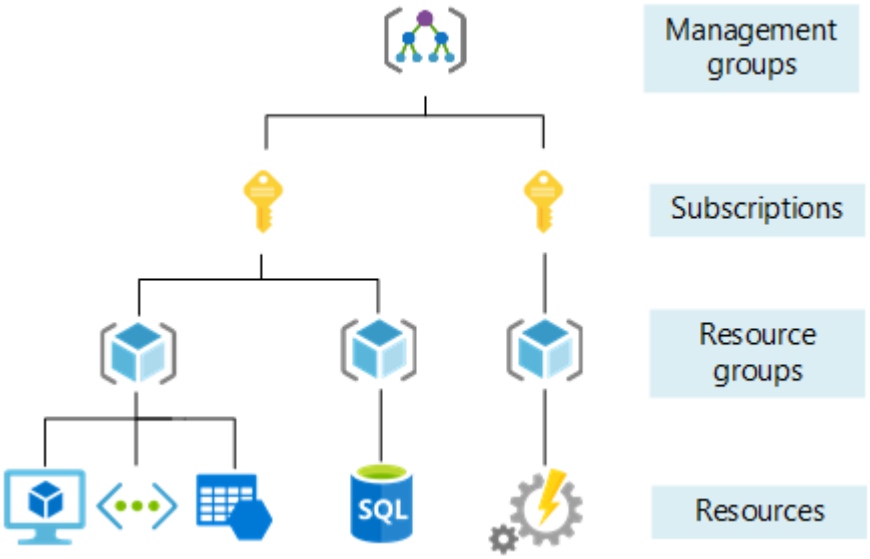
Token configurationilla voidaan valita valinnaisia claimeja, joita voidaan käyttää lisäämään lisävaatimuksia tunnuksiin, muuttamaan tiettyjen vaatimusten käyttäytymistä ja käyttämään mukautettuja hakemiston laajennusvaatimuksia. (Microsoft, 2020) Sovelluksen käyttämät valinnaiset claimit ovat groups.

API permissionsilla annetaan sovellukselle lupa päästä käsittelemään API resursseja. Määrittämällä web-API:n scopet sovelluksen rekisteröinnissä, sovellus voi hankkia Microsoftin tunnistusalustasta pääsytunnuksen, joka sisältää nämä scopet. Web-API voi sen jälkeen tarjota koodissaan lupaperusteisen pääsyn resursseihinsa, joka perustuu scopeista löytyviin access tokeneihin. (Microsoft, 2022e) Sovelluksen käyttämä scope on User.Read(), jolla luetaan vain kyseisen kirjautuvan käyttäjän tiedot.



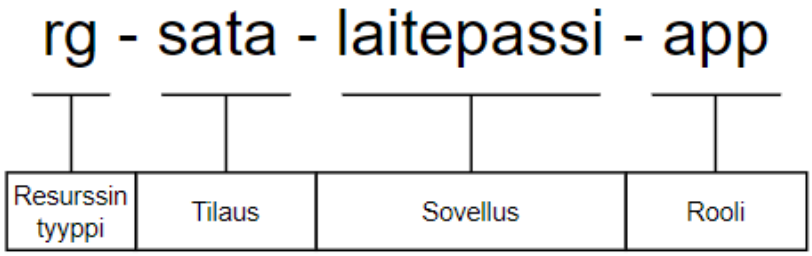
Kuva 19 Azure Subscriptionin sisältö tenantissa.

Azure Subscription alle on luotu kaksi resurssiryhmää (Kuva 19) (engl. resource group), yksi itse sovellusta varten (rg-sata-laitepassi-app) ja toinen tietokannoille (rg-sata-laitepassi-data). Resurssiryhmä on hallintaa helpottava komponentti, joka sisältää tiedon Azure-ratkaisuun liittyvistä resursseista. Tämä ryhmä voi sisältää kaikki ratkaisun resurssit yhdessä kontissa tai vain ne resurssit, joita halutaan hallita ryhmänä (Kuva 20). (Microsoft, 2023j) Järjestämisen kannalta on parasta pitää resurssit erillisissä resurssiryhmissään.



Kuva 20 Resurssien hallinta.

Resursseja nimetään käyttäen Microsoftin dokumentaation tapaista nimeämisformaattia (Kuva 21). Esimerkiksi resurssiryhmän rg-sata-laitepassi-app nimeäminen jaetaan neljään osaan. Ensimmäinen osa (rg eli resource group) vastaa resurssin tyyppiä, toinen (sata) vastaa tilausta, kolmas (laitepassi) vastaa sovellusta ja neljäs (app) vastaa resurssiryhmän sisällön merkitystä tai roolia.



Kuva 21 Esimerkki resurssin nimeämisestä.

Resursseja voidaan luoda Azuressa useilla eri menetelmillä. Näitä menetelmiä ovat esimerkiksi:

- Azure Portal
- Azure Command Line Interface (CLI)
- Azure Resource Manager (ARM) -malleilla

Azure Portalilla luodaan resursseja suoraan Azuren käyttöliittymältä.

Azure CLI:llä suoritetaan asianmukaisia Azure komentorivin komentoja luodakseen haluttuja resursseja, määrittäen vaadittuja parametreja ja asetuksia. Azure CLI on monialustainen komentorivityökalu, joka mahdollistaa yhteyden ottamisen Azureen ja hallinnollisten komentojen suorittamisen Azure-resursseissa. Se mahdollistaa komentojen suorittamisen komentorivipohjaisessa terminaalissa interaktiivisten komentorivipyyntöjen tai komentosarjaohjelman avulla. (Microsoft, 2023v)

ARM-mallit (engl. Azure Resource Manager templates) ovat Bicep- tai JSON-tiedostoja, joilla määritellään tarvittavat resurssit ratkaisun käyttöönottoa varten.

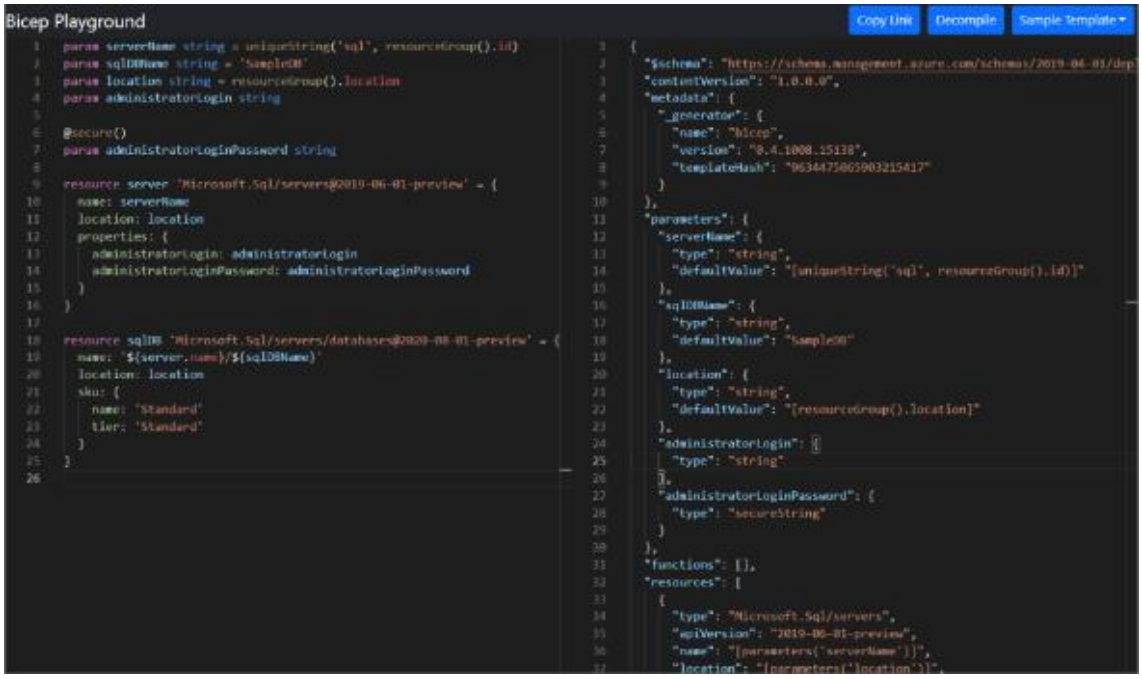
Bicep on Microsoft Azure -kohtainen kieli, joka käyttää deklaratiivista syntaksia Azure-resurssien käyttöönottoon. Bicep-tiedostossa määritellään infrastruktuuri, joka halutaan ottaa käyttöön Azureen. Kyseistä tiedostoa käytetään sitten kehityssyklin aikana infrastruktuurin toistuvaan käyttöönottoon. (Microsoft, 2023t)

JSON (JavaScript Object Notation) on standardoitu tekstipohjainen formaatti, joka perustuu JavaScript-objektien syntaksiin ja jota käytetään strukturoitujen tietojen esittämiseen. Yleisesti JSON:ia käytetään tietojen välittämiseen www-sovelluksissa, kuten datan lähettämiseen palvelimelta asiakkaalle, jotta se voidaan näyttää verkkosivulla tai päinvastoin. (MDN Web Docs, 2023e)

Resurssien hallinta järjestää riippuvaisten resurssien käyttöönottoa siten, että ne luodaan oikeassa järjestyksessä. Kun mahdollista, resurssien hallinta voi käyttöönottaa resursseja samanaikaisesti, joten käyttöönotot valmistuvat nopeammin kuin sarjalliset käyttöönotot. Käytetään mallia yhdellä komennolla sen sijaan, että käytettäisiin useita imperatiivisia komentoja. (Microsoft, 2023o)

ARM-mallit mahdollistavat koko Azure-infrastruktuurin luomisen ja käyttöönoton deklaratiivisesti. Esimerkiksi voidaan käyttöönottaa virtuaalikoneiden lisäksi verkkoinfrastruktuuri, tallennusjärjestelmät ja kaikki muut tarvittavat resurssit.

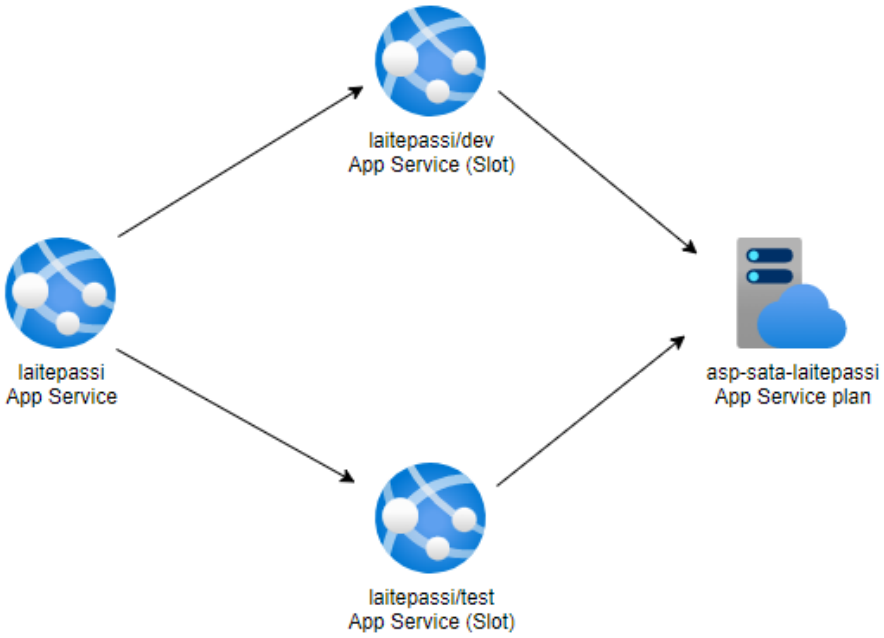
Kun vertaillaan Bicepia ja ARM JSON:ia (kuva 22), Bicep-tiedostot ovat tiiviimpiä ja helpommin luettavia. Bicep ei vaadi aikaisempaa ohjelmointikielten tuntemusta. Bicep-syntaksi on deklaratiivinen ja määrittelee, mitä resursseja ja resurssiominaisuuksia halutaan käyttöönottaa.



Kuva 22 Bicep (vas.) vs JSON (oik.).

5.3.2 Resurssiryhmien sisältö

Rg-sata-laitepassi-app -resurssiryhmä koostuu sovelluspalvelusta ja App Service planistä. (Kuva 23) Azure App Service, eli sovelluspalvelut jaetaan kolmeen deployment slotiin, pelkkä laitepassi sovelluspalvelu toimii modernin Laitepassi-sovelluksen tuontanto -ympäristöä, laitepassi/dev toimii sovelluksen uutena kehitysympäristönä ja laitepassi/test toimii sovelluksen testiympäristönä.



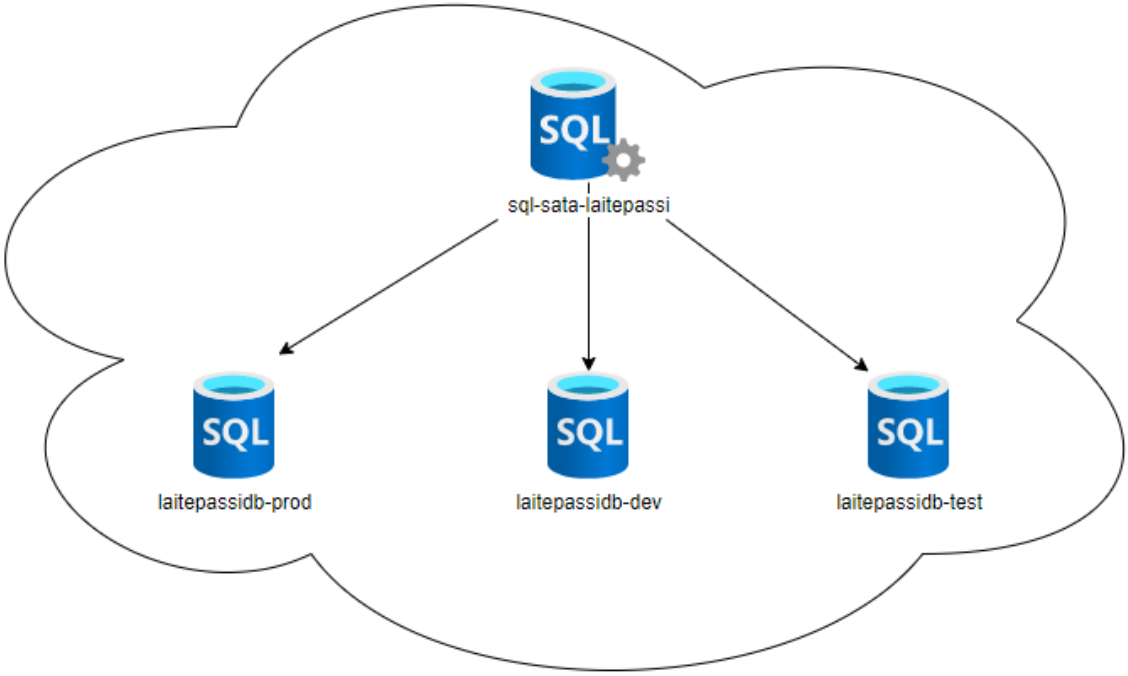
Kuva 23 rg-sata-laitepassi-app -resurssiryhmä.

App Service Plan, eli sovelluspalvelusuunnitelma on palvelu, jossa sovelluspalvelu aina toimii. Suunnitelman määrittelee joukon laskentaresursseja, joiden avulla verkkosovellus voi suorittaa toimintansa. Useampia sovelluksia voidaan määrittää toimimaan saman sovelluspalvelusuunnitelman alla.

Kun luodaan sovellus sovelluspalvelussa, se asetetaan sovelluspalvelusuunnitelmaan. Kun sovellus suoritetaan, se toimii kaikilla suunnitelmassa määritetyillä virtuaalikoneen instansseilla. Jos suunnitelma sisältää useita sovelluksia, ne silloin jakavat samat virtuaalikoneen instanssit. Koska sovelluksen käytössä on useita deployment sloteja, kaikki slotit suorittavat myös samoilla instansseilla. Jos otetaan käyttöön diagnostisia lokitiedostoja, suoritetaan varmuuskopioita, ne myös käyttävät virtuaalikoneen prosessorin suoritusaikaa ja muistia. (Microsoft, 2023e)

Rg-sata-laitepassi-data -resurssiryhmä koostuu Azure SQL Server resurssista, joka pitää sisällään kolme erillistä SQL tietokantaa. (Kuva 24) SQL Server on nimetty sql-sata-laitepassi. Palvelin nimeämisellä sql -osio viittaa SQL serveriin, sata -osio viittaa tilaukseen ja laitepassi -osio viittaa kyseessä olevaan

sovellukseen. Tietokannat ovat laitepassidb-prod (tuotanto), laitepassidb-test (testi) ja laitepassidb-dev (kehitys). Tietokantojen nimeäminen jaetaan kolmeen osioon: laitepassi viittaa sovellukseen, db viittaa tietokantaan (engl. database) ja prod, test ja dev viittaavat ympäristöön.



Kuva 24 rg-sata-laitepassi-data -resurssiryhmä.

5.4 .NET päivitys

Keväällä 2023 .NET:llä on kaksi tuettua versiota: .NET 6 ja .NET 7. .NET 6:n julkaisutyyppi on LTS ja .NET 7: julkaisutyyppi on STS. Koska .NET 6:ssa on pidempi elinkaari, on järkevämpi päivittää sovellus vanhasta LTS-versiosta uuteen. (Kuva 25)

Supported versions

The following table tracks release and end of support dates for .NET and .NET Core versions.

Version	Original release date	Latest patch version	Patch release date	Release type	Support phase	End of support
.NET 7	November 8, 2022	7.0.4	March 14, 2023	STS	Active	May 14, 2024
.NET 6	November 8, 2021	6.0.15	March 14, 2023	LTS	Active	November 12, 2024

Kuva 25 Tällä hetkellä tuetut .NET -versiot.

.NET 6 myötä .NET Core ja .NET Framework ovat yhtä. .NET 6 yhdistää SDK:n, peruskirjastot ja runtime:n mobiili-, työpöytä ja muille uudemmille teknologioille, kuten IoT- ja pilvisovelluksia. .NET 6 tarjoaa yksinkertaistetumpaa kehitystä, parempaa suorituskykyä ja parempaa tuottavuutta Microsoftin uudella IDE-versiolla. (Softensity, 2022)

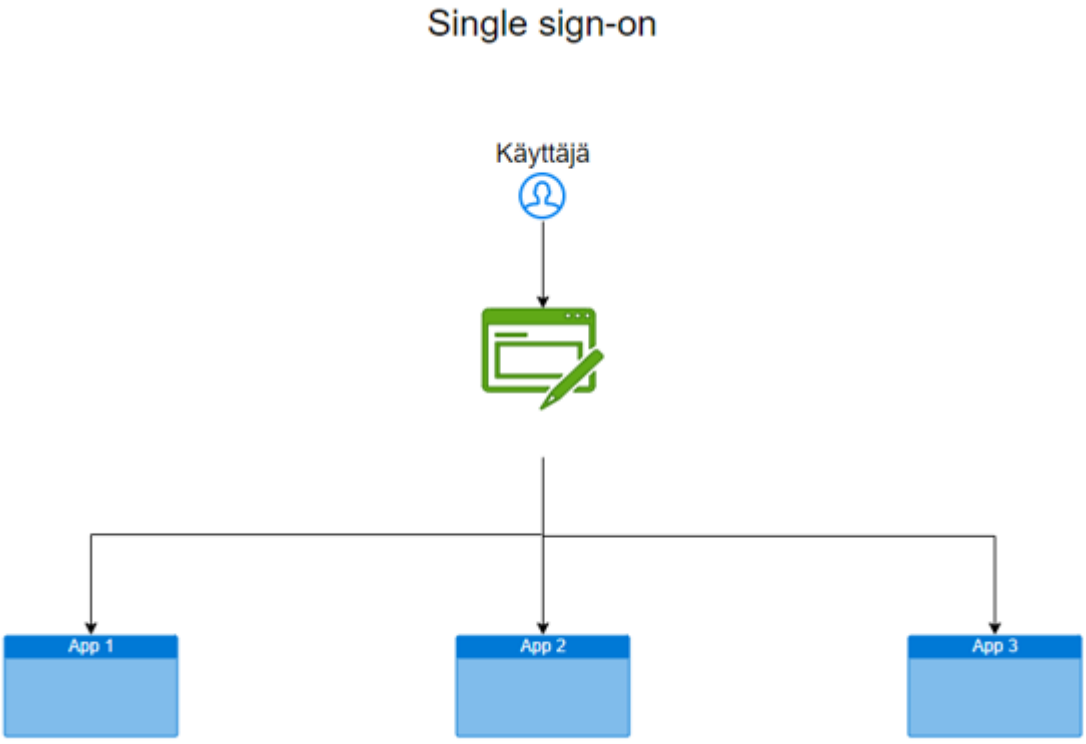
5.5 Autentikaatio

Jatkossa Laitepassi sovellus tulee käyttämään kirjautuvien käyttäjien autentikoimiseen Azure Active Directory todennusta ja Microsoft Graph APIa.

5.5.1 Azure Active Directory

Azure Active Directory (Azure AD) on yritystason identiteettipalvelu, joka tarjoaa yhden kirjautumisen (engl. single sign-on), monitekijäisen todennuksen (engl. multi-factor authentication) ja ehdollisen pääsyn (engl. conditional access) suojautuakseen 99,9 prosenttisesti tietoturvahyökkäyksiltä. (Microsoft, n.d.)

Single sign-on (SSO) on istunnon ja käyttäjä todennus palvelu, joka tarjoaa käyttäjälle oikeuden käyttää yhtä sarjaa kirjautumistietoja (kuten käyttäjätunnus ja salasana) kirjautuakseen useammalle eri sovellukselle tai järjestelmälle. (Kuva 26) Tällöin käyttäjä ei joutuisi syöttämään erillisiä kirjautumistietoja jokaiselle sovellukselle tai järjestelmällä. (TechTarget, 2022c)



Kuva 26 Esimerkki single sign-on.

Multi-factor authentication (MFA) on moniosainen tunnistautumisprosessi, jossa käyttäjältä vaaditaan muutakin kuin pelkkä salasana. Salasanan lisäksi käyttäjältä saatetaan, sähköpostin tai puhelinnumeron välityksellä, pyytää vahvistuskoodia. Tällä toiselle tunnistautumis- muodolla vältetään asiattomien pääsyä, vaikka salasana olisi vaarantunut. (Amazon Web Services n.d.c)

Conditional access, eli ehdollinen pääsy, rakentuu käytännöistä. Nämä käytännöt ovat yksinkertaisimmillaan niin sanottuja if-else-lausuntoja. Jos käyttäjä haluaa pääsyn johonkin resurssiin, niin hänen täytyy suorittaa jokin toiminta. (Microsoft, 2023u)

Azure AD:ssa todennukseen sisältyy muuta kuin pelkkä käyttäjän tunnus ja salasana. Turvallisuuden parantamiseksi ja asiakastukipalvelun tarpeen vähentämiseksi Azure AD:n tunnistus käyttää muun muassa seuraavanlaisia komponentteja: itsepalvelusalasanan nollaus, Azure AD MFA, salasanaton tunnistautuminen. Itsepalvelusalasanan nollaus tarkoittaa, että käyttäjällä on

kyky muuttaa tai nollata oma salasanansa ilman järjestelmänvalvojan tai asiakastukipalvelun puuttumista asiaan. Salasanaton tunnistautuminen tarkoittaa, että käyttäjän tunnistustiedot tarjotaan esimerkiksi biometrisillä tiedoilla, kuten kasvojen- tai sormenjälki- tunnistautumisella. (Microsoft, 2023r)

Laitepassin osalta suunnitelmana on käyttää MFA:ta. Käyttäjä tulee jatkossa ensin kirjautumaan organisaationsa sähköpostilla sovellukseen, jonka jälkeen käyttäjältä pyydetään vielä vahvistuskoodia puhelimen välityksellä. Tämän prosessin jälkeen pystyy käyttäjä toimimaan normaalisti sovelluksen kanssa.

5.5.2 Microsoft Graph

Microsoft Graph API tarjoaa yhtenäisen ohjelmointimallin, jonka avulla voidaan käyttää valtavaa määrää tietoa, muuan muassa Microsoft 365:ssä. (Microsoft, 2023k)

Graph API tarjoaa yhden rajapinnan, https://graph.microsoft.com, jonka avulla voi käyttää tietoja ja näkymiä Microsoftin pilvessä. Microsoft Graph sisältää joukon palveluita, jotka hallitsevat käyttäjän ja laitteen tunnistusta, käyttöoikeuksia, noudattamista ja turvallisuutta, sekä auttavat suojaamaan organisaatioita erilaisilta tietovuodoilta tai -hävikiltä. (Microsoft, 2023k)

Sovelluksen käytössä Graph API:lla saadaan tarkemmat lisätiedot kirjautuvasta käyttäjästä.

5.6 Julkaisuhallinta

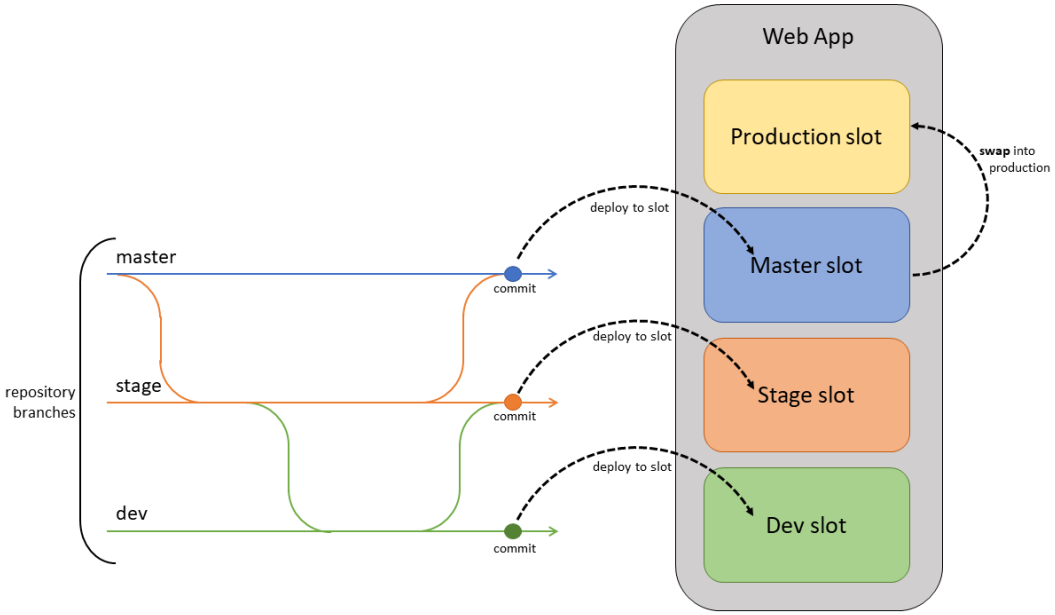
Azure App Servicen käyttöönotossa on mahdollista hyödyntää Deployment slotteja, jotka tarjoavat käyttöönotolle omat virtuaaliset instanssit (toisin sanoen VM) ja host-nimet (eli verkkosivu). Deployment slotit tarjoavat mahdollisuuden testata sovelluksen uusia ominaisuuksia ja päivityksiä, ennen siirtymistä tuotantoympäristöön.

Deployment slotien käyttö tarjoaa useita etuja, kuten mahdollisuuden validoida sovelluksen muutokset staging-ympäristössä ennen julkaisua tuotantoon. Deployment slotit mahdollistavat myös sovelluksen käynnistämisen deployment slotissa ja sen jälkeen vaihtamisen tuotantoon, jolloin kaikki slotin instanssit ovat valmiina tuotantokäyttöön. Tämä vähentää käyttökatkosten riskiä julkaisuprosessin aikana.

Deployment slotien käyttö on suositeltavaa, koska se mahdollistaa nopean ja helpon palautuksen edellisempään versioon, jos julkaisuprosessi ei onnistu tai muutokset eivät toimi odotetulla tavalla tuotannossa. (Microsoft, 2023l)

Kuvassa 27 nähdään esimerkki, jossa Web App kuvastaa Azure App Serviceä, jolla on useita Deployment sloteja. Versionhallinnasta (kuvan vasemmasta osasta) tuodaan muutoksia dev, stage ja master sloteihin (deploy to slot). Master slotista päivitetään uusin versio aina production slotiin (swap into production).

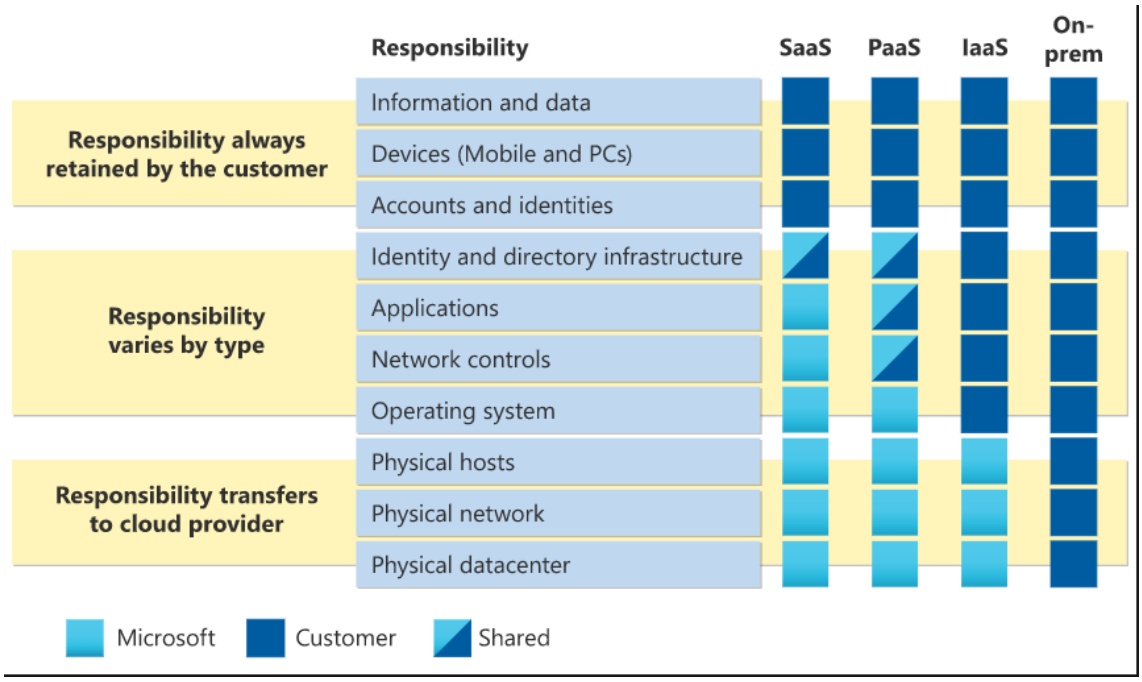
Slotin vaihdossa (engl. swap), yksi slotti toimii lähteenä ja toinen kohteena. Lähde slotissa on ilmentymä sovelluksesta, jota käytetään kohde slotissa.



Kuva 27 Deployment slots.

5.7 Ylläpito

Azureen siirtymisen myötä, organisaation taakka ylläpidon vastuullisuudesta kevenee. Kuten kuvasta 28 näkee, on-prem (eli omien palvelimien) sovelluksilla organisaation asiantuntijat ovat vastuussa kaikesta. PaaS-komponentteja hyödyntämällä tätä vastuuta voidaan siirtää Microsoftille. Kaikissa pilvipalvelun käyttöönotoissa, Microsoft ei vastaa datasta ja käyttäjätiedoista. Käyttäjät omistavat omat tietonsa ja tunnistetietonsa. He ovat vastuussa tietojensa ja tunnistetietojensa turvallisuuden suojaamisesta, sekä omassa fyysisessä tilassaan, että pilvipalvelun komponenteissa, joita he hallinnoivat. (Microsoft, 2022g)



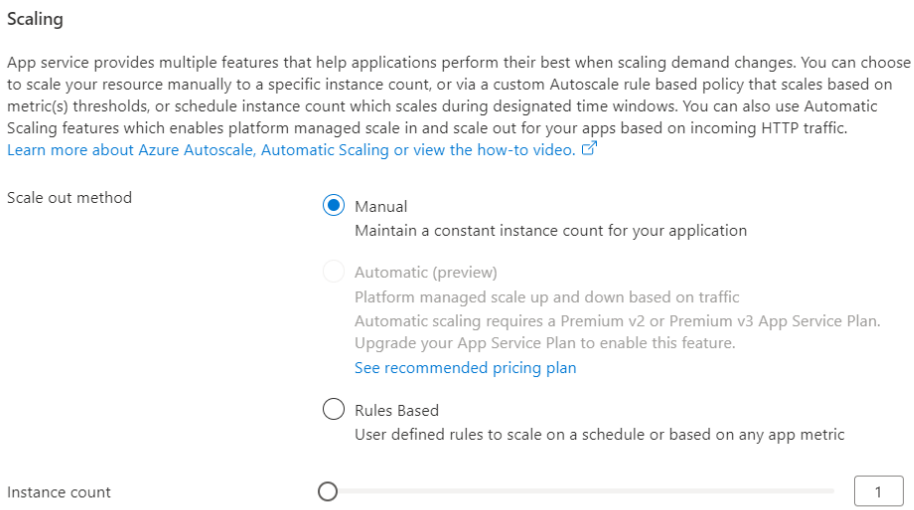
Kuva 28 Vastuun jakaminen.

5.8 Skaalautuvuus

Pilvipalvelut tarjoavat enemmän joustavuutta. Resursseja ja tallennustilaa voidaan nopeasti skaalata vastaamaan liiketoiminnan vaatimuksia, ilman investointia fyysiseen infrastruktuuriin. Yritysten ei tarvitse maksaa tai rakentaa

infraa, joka tukisi korkeimpia kuormitustasoja. Samoin voidaan nopeasti skaalata alas, jos resursseja ei käytetä. (Google Cloud n.d.)

Azure sovelluspalvelulla on olemassa kaksi työkulkua skaalaamiseen (Kuva 29), ylöspäin (engl. scale up) ja ulospäin (engl. scale out). Ylöspäin skaalaamalla saat lisää prosessoritehoa, muistia, levytilaa ja muita ominaisuuksia, kuten omistetut virtuaalikoneet (VM), mukautetut verkkotunnukset ja sertifikaatit, tuotantokäyttöönottopaikat, automaattinen skaalaus ja muut. Sovelluksia skaalataan ylöspäin muuttamalla sovelluksen sovelluspalvelusuunnitelman hinnoittelutasoa. Ulospäin skaalaaminen tarkoittaa, että kasvatat sovelluksesi käyttämiä virtuaalikoneiden määrää. Riippuen hinnoittelutasosta, jopa 30 virtuaalikoneeseen voidaan skaalata. (Microsoft, 2022f)



Kuva 29 App Servicen skaalaaminen Azuressa.

5.9 Lokitus

Nykyistä lokituksen ratkaisua voidaan jatkossa käyttää, viemällä se Azureen. Suunnitelman pohjalta on valittu Azuren valmis ratkaisu. Tämä on Azuren valmis palvelu Application Insights. Azuren Application Insights on laajennus Azure Monitorista, tarjoaa sovelluksen suorituskyvyn seurantatoimintoja (APM,

eli Application Performance Monitoring). Application Insightsin avulla voidaan kerätä ja tallentaa sovelluksen jäljityslokidataa yhdessä metriikkojen ja sovelluksen telemetriatiedon kanssa. Nämä kuvaavat sovelluksen toimintaa ja tilaa. (Microsoft, 2023d)

5.10 Komentosarjaohjelma

Aikaisemmin käytössä sovelluksen ollutta PowerShell komentosarjaohjelmaa voidaan edelleen hyödyntää Azuressa, mutta on olemassa yksinkertaisempiakin ratkaisuja. Ohjelma voidaan korvata Azure Logic Appsillä, tai Azure Functionsilla tai jopa Azure Application Insightsilla.

Azure Logic Apps on pilvipohjainen alusta, jolla voi luoda ja ajaa automatisoituja työnkulkuja, joko vähäisellä koodilla tai ilman koodia. Logic Appsillä voi käyttää visuaalista suunnittelijaa ja valita valmiiksi luotuja toimintoja. Näillä voi nopeasti rakentaa työnkulun, joka integroi ja hallitsee sovelluksiasi, tietojasi, palveluitasi ja järjestelmiäsi. (Microsoft, 2023s)

Azure Functions on palvelinratkaisu, jonka avulla mahdollistetaan vähäinen koodin kirjoittaminen, infrastruktuurin vähäisempi ylläpito ja kustannussäästöt. Sen sijaan, että huolehdittaisiin palvelinten käyttöönotosta ja ylläpidosta, pilven infrastruktuuri tarjoaa kaikki ajan tasalla olevat resurssit, joita tarvitaan sovelluksen käyttämiseen. (Microsoft, 2023h)

Paras valinta tähän tilanteeseen on Logic Apps, koska siinä on valmiita komponentteja, jotka sopivat nykyiseen ratkaisuun.

6 Ratkaisun modernisoinnin toteutus

Tässä kappaleessa käsitellään itse modernisoinnin toteutusta. Kappaleessa käydään läpi, kuinka luodaan Azuren tekninen ympäristö, mitä ohjelmoitavia muutoksia itse sovellus vaati ja kuinka lopuksi sovellus julkaistaan rakennettuun pilviympäristöön.

Ratkaisun pohjalle on luotu 2M-IT:n asiakkaan Azure tenatiin uusi Azure tilaus (engl. Azure Subscription), johon kehittäjät ovat saanet roolipohjaiset Avustaja -oikeudet (engl. Contributor) käyttöoikeudet (RBAC, engl. Role Based Access Control).

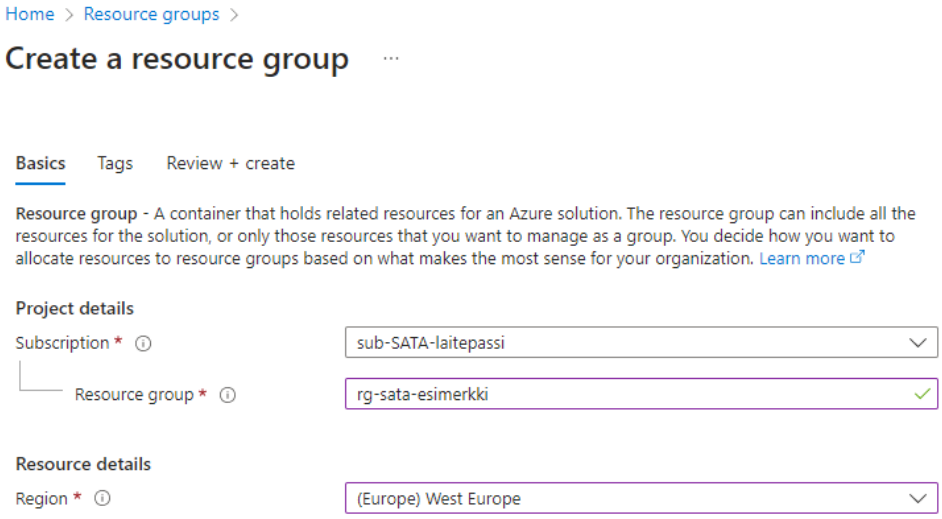
6.1 Azuren resurssin luonti

Suunnitelmana oli luoda resurssit Bicep-tiedostoja hyödyntämällä. Resurssit kuitenkin luodaan dev (kehitys), test (testi) ja prod (tuotanto) nimisiin ympäristöihin Azuren käyttöliittymän, eli Azure Portalin kautta. Kaikki Azure-ympäristön resurssit vaativat resurssiryhmän (Kuva 20), jotta resursseja voidaan hallinnoida. Resurssiryhmät sijoitetaan Azure tilaukseen. Seuraavaksi käsitellään, miten luodaan resurssi Azure Portalin ja ARM-mallin kautta.

Tässä tapauksessa luodaan resurssit käsin, Azuren käyttöliittymän kautta. Jatkossa on kuitenkin kannattavampaa luoda resursseja Bicepilla, esimerkiksi uusia asiakkuuksia varten. Tällöin voidaan luoda samantyyppisiä ympäristöjä luotettavasti käyttäen samoja bicep-malleja, eikä luoda resursseja aina alusta asti, Azuren käyttöliittymän kautta käsin.

6.1.1 Esimerkkiresurssin luonti Azuren käyttöliittymältä

Kuva 30 näyttää resurssiryhmän luonnin käyttöliittymältä, eli Azure Portalin kautta. Ryhmän luontia varten vaaditaan subscription (tilaus), resource group (resurssiryhmä) ja region (alue). Tags -sivua ei tarvinnut tässä tapauksessa huomioida, sillä ne luodaan automaattisesti.



Kuva 30 Esimerkki resurssin luontilomakkeen täytöstä.

Azure-alueet (engl. Azure region) on suunniteltu saavuttamaan luotettavuutta työkuormille, jotka ovat liiketoimintakriittisiä. Azuressa on useita maantieteellisiä alueita. Alueiden erilliset rajat määrittävät katastrofien varautumisen ja tietojen sijaintipaikan rajat yhden tai useamman Azure-alueen yli. Usean alueen ylläpidolla varmistetaan, että asiakkaita voidaan tukea ympäri maailmaa.

Jokaisessa alueessa on tietokeskuksia, jotka on sijoitettu viiveeseen perustuvan kehän sisälle. Nämä tietokeskukset on yhdistetty omistettuun alueelliseen vähäviiveiseen verkkoon. Tämä suunnittelu varmistaa, että Azuren palvelut tarjoavat parhaan mahdollisen suorituskyvyn ja turvallisuuden missä tahansa alueessa. (Microsoft, 2023p)

6.1.2 Esimerkkiresurssin luonti Bicep teknologialla

Kuva 31 näyttää samankaltaisen resurssiryhmän luonnin Bicepilla, hyödyntäen Bicep-kieltä. Luodaan Bicep-tiedosto haluamaamme sijaintiin ja annetaan tiedostolle nimi. Tässä tapauksessa tiedoston nimi on bicep-rg-example.

```
targetScope = 'subscription'
param location string = 'westeurope'

resource bicepResourceGroup 'Microsoft.Resources/resourceGroups@2022-09-01'= {
    name: 'rg-sata-bicepEsimerkki'
    location: location
}
```

Kuva 31 Bicep-tiedoston sisältö resurssiryhmän luontia varten.

Ensin määritetään targetScope, jonka arvo on tässä tapauksessa ’subscription’ eli Azure tilaus. Jokaisella Bicep-tiedostolla on targetScope määritys, jota käytetään resurssien validointiin ja tarkistuksiin mallipohjissa. Kun halutaan ottaa resurssiryhmä käyttöön, targetScope tulee todennäköisesti olla tilaus (’subscription’), koska targetScope ei voi olla oletusarvoinen resourceGroup, koska itse resurssiryhmää luodaan. (ochzhen.com, Create Resource Group With Azure Bicep and Deploy Resources In It, n.d.) Seuraavana määritellään resurssiryhmän Azure-alue (engl. location) omaan parametriinsa, jonka arvo on ’westeurope’. Tämä alue valitaan, koska kyseinen alue on lähimpänä Suomea.

Kun luodaan itse resurssia, sille täytyy antaa symboolinen nimi. Nimi on tässä esimerkkitapauksessa bicepResourceGroup. Nimen perään määritellään, minkälaista resurssia ollaan luomassa, eli 'Microsoft.Resources/resourceGroups@2022-09-01'. Tämä merkkijono rakentuu seuraavasti: Microsoftin resurssi, joka on resurssiryhmä (’Microsoft. Resources/resourceGroups’) ja sen luomiseen käytettävä uusin ehdotettu API-versio (’@2022-09-01’). Resurssin sisälle määritellään vielä nimi (engl. name) ja sijainti (engl. location).

6.2 .NET version päivittäminen

Microsoft hoitaa jatkossa Azure App Service ja Azure SQL Server versioiden ylläpidon, joten kehitystiimin täytyy vain päivittää sovellustasolla .NET -versiota.

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
-    <TargetFramework>netcoreapp3.1</TargetFramework>
+    <TargetFramework>net6.0</TargetFramework>
  </PropertyGroup>

</Project>
```

Kuva 32 .NET version päivitys projektitiedostosta.

Sovelluksen .NET -version päivittämiseksi avataan sovelluksen .csproj projektitiedosto ja korvataan ”TargetFramework”-kentän sisällä oleva ”netcoreapp3.1”-arvo ”net6.0”-arvolla. (Kuva 32) Tämän päivityksen myötä saadaan sovellukselle uuden Microsoftin tuen piirissä .NET Core ja ASP .NET Coren päivitykset. Lisäksi saadaan muut sovelluskomponentit, kuten EF Core, päivitettyä uusiin versioihin.

6.3 Autentikaation päivitys

Sovelluksen aiemmin käytössä olleesta, kehitystiimin räätälöimästä, autentikaatio-moduulista ja kirjautumissivusta luovutaan kokonaan. Modernisoitu versio käyttää Azure AD:n App Registrationsien tarjoamia ominaisuuksia, jolla saadaan uudistettu autentikaatio ja kirjautumissivu toteutettua.

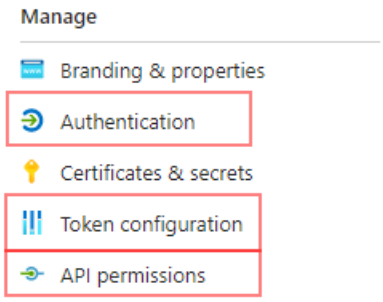
6.3.1 App Registration

Microsoftin ratkaisu sovellusidentiteetin luontiin Azure AD:ssa, jolla voidaan toteuttaa sovellukseen kirjautumistoiminto. App Registrationissa konfiguroidaan kolme sivua, joiden kautta kirjautuvan käyttäjän autentikaatio tulee tapahtumaan (Kuva 33).

App Registrationin sivut ovat:

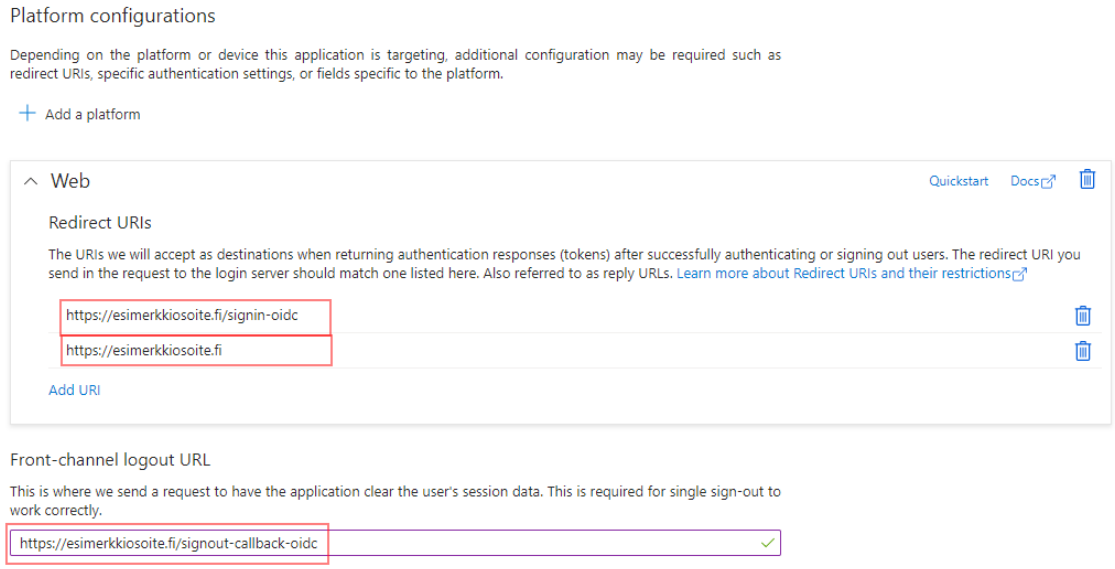
- Authentication
- Token configuration

- API permissions



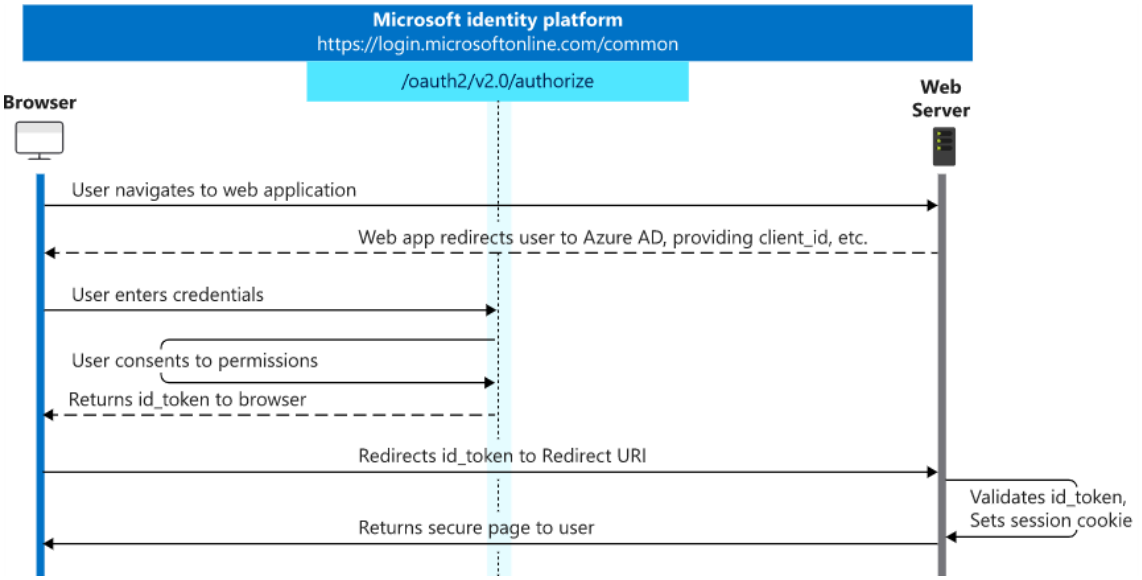
Kuva 33 Azure App Registration konfiguroitavat sivut autentikaatiota varten.

Authentication -sivulla määritetään uudelleenohjaus URI:t (engl. Redirect URIs) ja uloskirjautumisosoite (Kuva 34). Tässä tapauksessa lisätään kaksi URI:a uudelleenohjaukseen, ’https://esimerkkiosoite.fi’ ja ’https://esimerkkiosoite.fi/signin-oidc’. Uloskirjautumista varten ’https://esimerkkiosoite.fi/signout-callback-oidc’**.** URI:n loppuosa perustuu ASP .NET Coren autentikaation oletustoteutukseen.



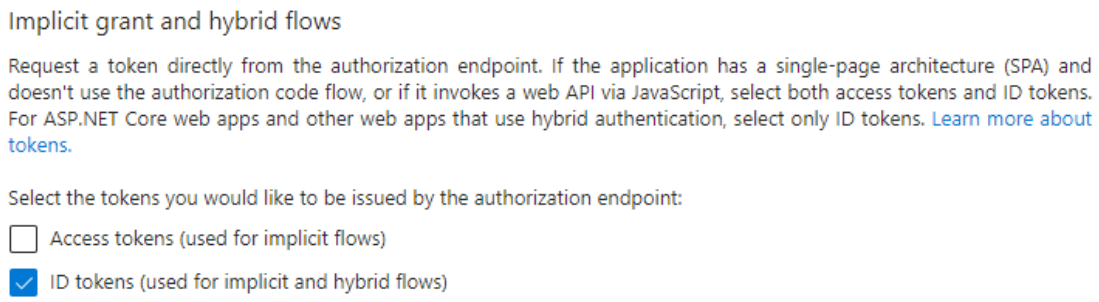
Kuva 34 Authentication -sivulla lisätään uudelleenohjaus URI:t.

Käyttäjän onnistuneesti tunnistauduttuaan, ’signin-oidc’ -päätteinen osoite käsittelee tunnistautumistietoja, kerää tarvittavat ID tokenit (Kuva 36), jonka jälkeen käyttäjä ohjataan itse sovelluksen sivulle (Kuva 35).



Kuva 35 Kirjautumisprosessi.

Kuvassa 35 havainnollistetaan uutta autentikaatiota. Uudessa sovellusversiossa voidaan ulkoistaa autentikaatio Azure AD:lle.

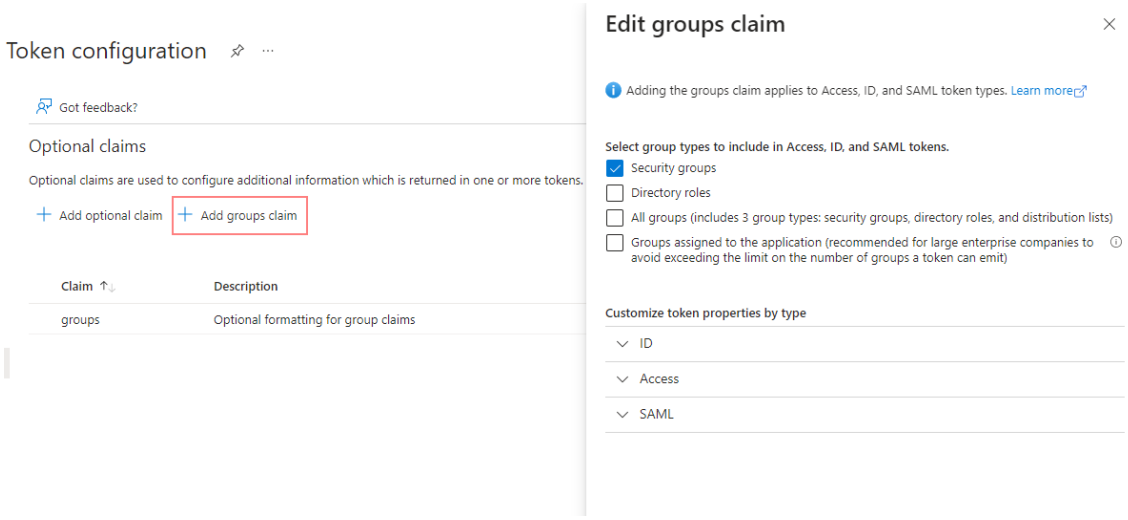


Kuva 36 Tokeneita, joita '/signin-oidc' kerää.

’/signout-call-oidc’ -osoitetta kutsutaan, kun käyttäjä kirjautuu ulos ja hänen istuntotietonsa halutaan tyhjentää.

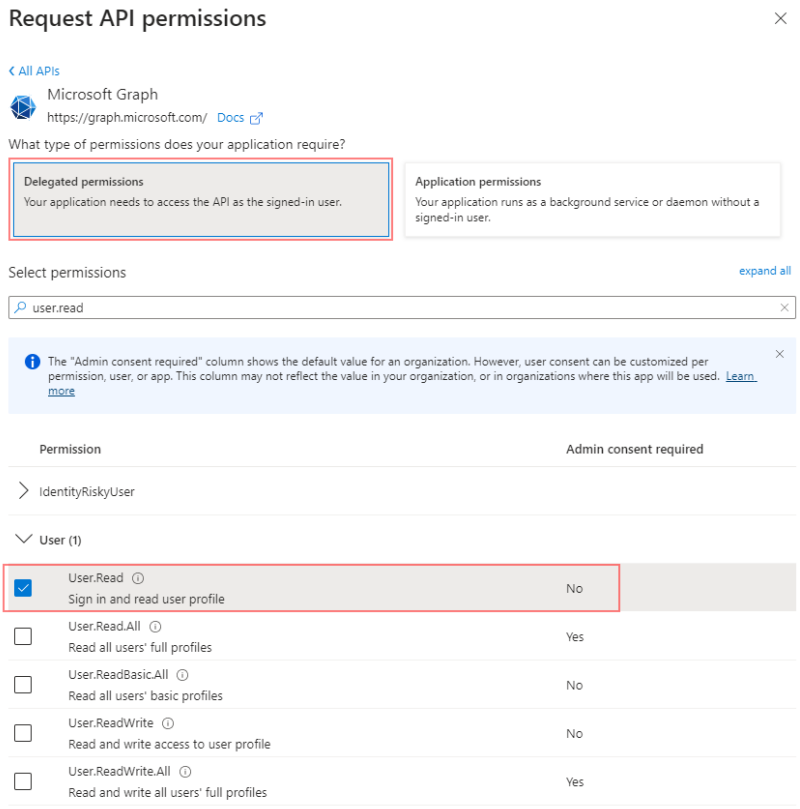
Oletusarvoisesti kun uusi App Registration on luotu, ryhmät eivät tule käyttäjätietojen mukaan. Tämän takia Token configuration sivua muokataan.

Token configuration -sivulla lisätään uusi group claim ja valitaan ryhmän tyypiksi Security group -vaihtoehto (Kuva 37). Group claims tarjoaa tietoa ryhmästä tai ryhmistä, joiden jäsen sovellukseen kirjautuva käyttäjä kuuluu. Tästä on hyötyä, kun on tehtävä valtuutuspäätöksiä ryhmän jäsenyyden perusteella. Lisättiin ainoastaan Security group, koska ryhmiä käytetään käyttäjän autentikaatiossa ja autorisoinnissa, eikä esimerkiksi jakelulistoja tarvita sovelluksessa.



Kuva 37 Token congifuration -sivu.

API permissions -sivulla lisätään uusi API oikeus (engl. permission), johon valitaan Microsoft Graph. Valitaan delegoidut oikeudet (engl. Delegated permissions), joka tarkoittaa, että sovellus tarvitsee oikeuden API:in ainoastaan kirjautuneesta käyttäjästä. Sitten valitaan oikeuksien listasta User.Read -oikeus, eli kirjautuessaan sovellukseen, käyttäjän profiili luetaan (Kuva 38). Kirjautuessaan sovellukseen, luetaan vain kirjautuneen käyttäjän profiili, eikä muita oikeuksia tarvita.



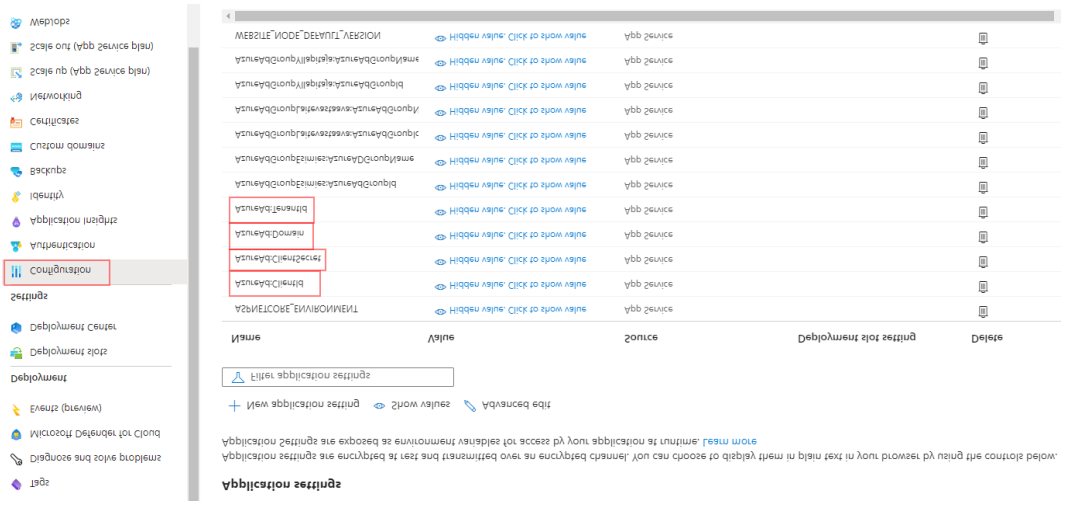
Kuva 38 API permissions -sivun oikeuksien lisäys Graph API:lle.

6.3.2 Ohjelmointimuutoksia

Kirjautumisen toteuttamiseksi Azure AD:lla, .NET Core -sovelluksessa, sovelluksen appsettings -tiedoston arvoja lisätään kuvan 39 mukaisesti. JSON-tiedoston kautta määritellään Azure AD:n tarvitsemat arvot kehitysaikana. Tuotannossa ja asiakkaan testiympäristössä arvot määritetään Azure App Servicen ympäristön muuttujien kautta.

Kuva 39 appsettings.ympäristö.json -tiedoston sisältöä.

Instance -kenttään kirjataan Azure AD autentikaation päätepiste (kirjautumissivu), joka on oletusarvona ”https://login.microsoftonline.com/”. Domain -kenttään lisätään verkkotunnus, joka on liitettynä Azure AD:hen. Esimerkkinä tunnuksena on ”esimerkki.onmicrosoft.com”. TenantId on yksilöllinen tunniste Azure AD tenantille. TenantIdn löytää Azure portalin Azure Active Directory -sivulta. ClientId on itse sovelluksen tunniste, joka saadaan sovelluksen rekisteröinnin yhteydessä Azure portalissa. ClientSecret on suojattu merkkijono, jota käytetään todentamaan sovellus Azure AD:ssa Graph API:a käytettäessä. Tämän saa myös määritettyä sovelluksen rekisteröinnin yhteydessä.Tuotannossa ja asiakkaan testiympäristössä nämä arvot määritetään suoraan Azuressa, sovelluksen App Servicen Configuration -sivulla (Kuva 40).



Kuva 40 Asiakkaan tuotanto- ja testiympäristön Azure -tietojen täyttö App Servicessä.

Sovelluksen Startup.cs, eli käynnistyksen ja kokoonpanon määrittelemässä tiedostossa, tehdään muutamia muutoksia, jotta saadaan yhteys Azureen toimimaan (Kuva 41).

Ensimmäisessä kuvan korostuksessa on koodirivi ”Configuration.Bind(”AzureAd”, options)”. Tällä sidotaan Azure AD:n tunnistautumiseen liittyvät konfiguraatioarvot, Azure AD:n vaihtoehtoja edustavaan objektiin. Seuraava korostus kuvassa, jonka sisältö on ”options.Events.OnTokenValidated”, viittaa tapahtumankäsittelijään, jota kutustaan, kun token validoidaan onnistuneesti autentikaatioprosessin aikana. Kolmannessa korostuksessa, kutsutaan Microsoft Graph API:a ja haetaan käyttäjän token, jotta voidaan kutsua Graph API:a User.Read -oikeuksilla ja hakea käyttäjän tiedot. Viimeisessä korostuksessa määritetään kirjautunut käyttäjä user -muuttujaan, johon liitetään User.Readista haettuja tietoja omiin arvoihinsa. Nämä arvot ovat:

- DisplayName, eli käyttäjän näyttönimi, joka on vain käyttäjän koko nimi.
- GivenName, eli käyttäjän etunimi.
- Surname, eli käyttäjän sukunimi.
- JobTitle, eli käyttäjän ammattinimike.
- Mail, eli käyttäjän sähköpostiosoite.

- Department. eli käyttäjän yksikkö.

```
services
// Use OpenId authentication
.AddAuthentication(OpenIdConnectDefaults.AuthenticationScheme)
// Specify this is a web app and needs auth code flow
.AddMicrosoftIdentityWebApp(options =>
{
    Configuration.Bind("AzureAd", options);

    // This causes the signin to prompt the user for which
    // account to use - useful when there are multiple accounts signed
    // into the browser
    options.Prompt = "select_account";

    options.Events.OnTokenValidated = async context =>
    {
        var tokenAcquisition = context.HttpContext.RequestServices
            .GetRequiredService<ITokenAcquisition>();

        var graphClient = new GraphServiceClient(
            new DelegateAuthenticationProvider(async (request) =>
            {
                var token = await tokenAcquisition
                    .GetAccessTokenForUserAsync(new string[] { "User.Read" }, user: context.Principal);
                request.Headers.Authorization =
                    new AuthenticationHeaderValue("Bearer", token);
            })
        );

        // Get user information from Graph
        User user = await graphClient.Me.Request()
            .Select(x => new
            {
                x.DisplayName,
                x.GivenName,
                x.Surname,
                x.JobTitle,
                x.Mail,
                x.Department
            })
            .GetAsync();
```

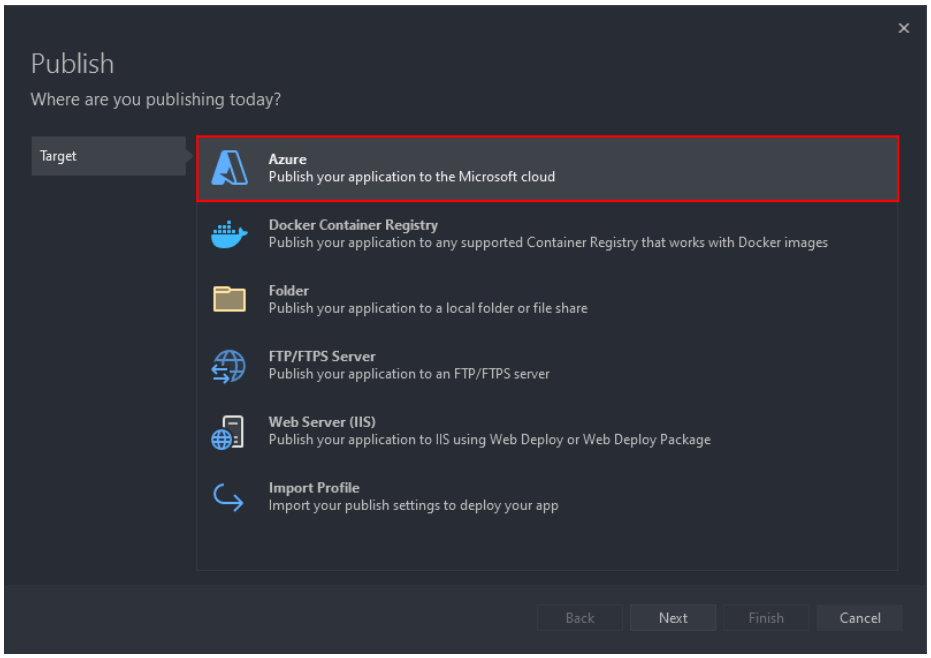
Kuva 41 Aloitustiedoston konfigurointi.

6.4 Sovelluksen julkaisu pilveen

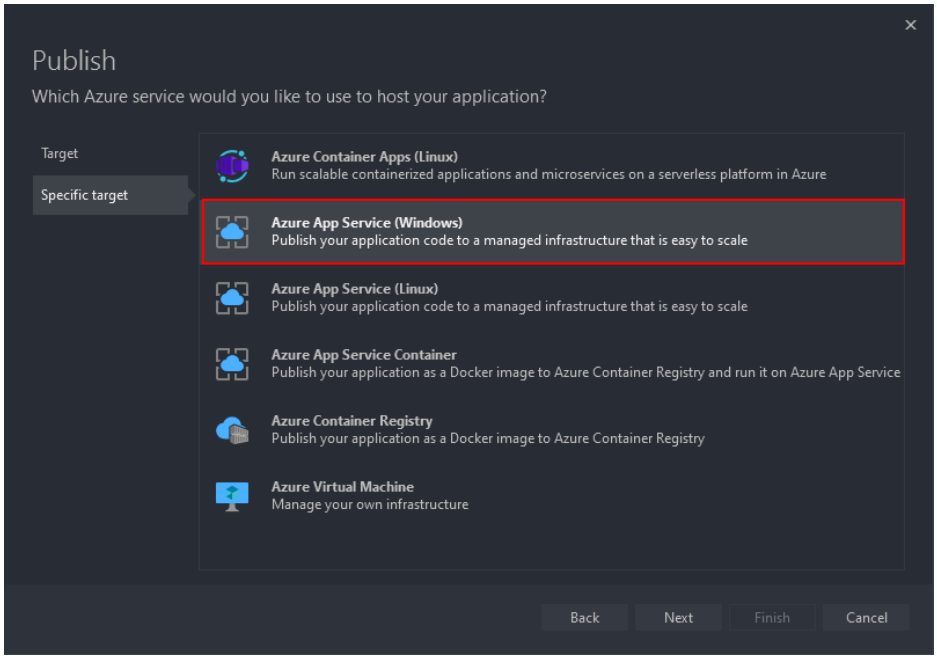
Sovellus julkaistaan kehitys- ja testivaiheessa, sekä tuotantokäytön alkuvaiheessa Azure -pilviympäristöön Visual Studio kautta. Sovellukselle kuitenkin tehtiin myös suunnitelma, miten se voidaan julkaista myös Azure DevOpsin ja GitHubin kautta, muun muassa käyttäen Bicep-malleja.

Sovelluksen julkaisussa Azureen, käytetään Visual Studion Azure Publish-metodia. (Kuva 42) Aikaisemmin käytettiin Folder Publish-metodia. Azure Publish-metodilla saadaan julkaistua sovellus suoraan Azuren App Serviceen, haluttuun deployment slotiin. Aiemmin Folder Publish-metodia käyttäessä sovellus julkaistiin tiedostona paikalliseen ympäristöön, jonka jälkeen sovellus siirrettiin useamman vaiheen kautta haluttuun virtuaaliympäristöön.

Sovellukselle voi luoda niin monta julkaisutiedostoa kuin haluaa. Laitepassi-sovellukselle luodaan kolme tiedosto, yksi per ympäristö (tuotanto, testi ja kehitys). Seuraavaksi käydään läpi, miten tuotantoympäristöä varten luotaisiin julkaisutiedosto. Sovellus halutaan julkaista Azure App Serviceen (Kuva 43).

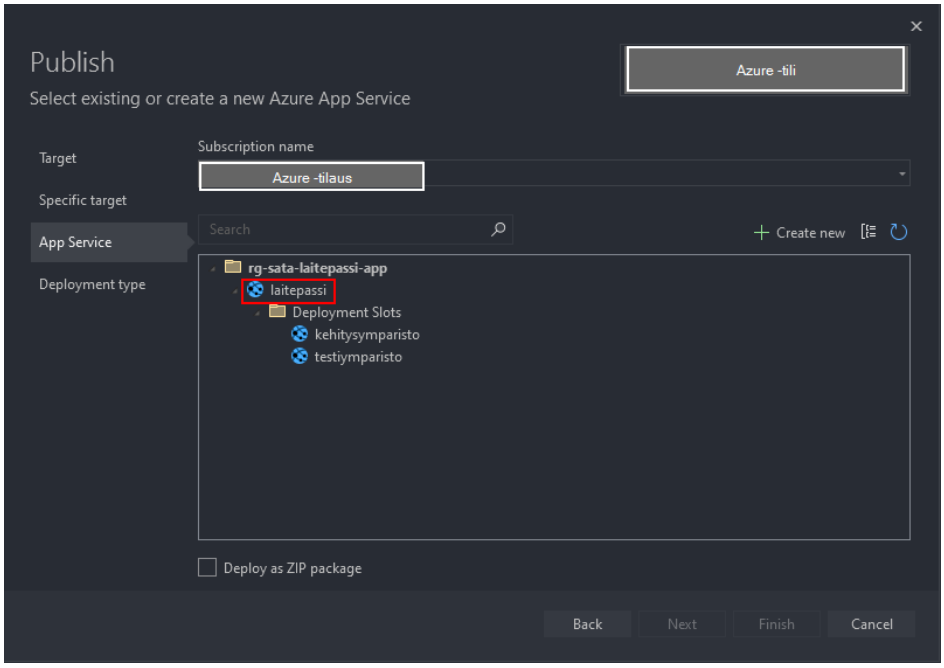


Kuva 42 Visual Studio Publish -sivu.



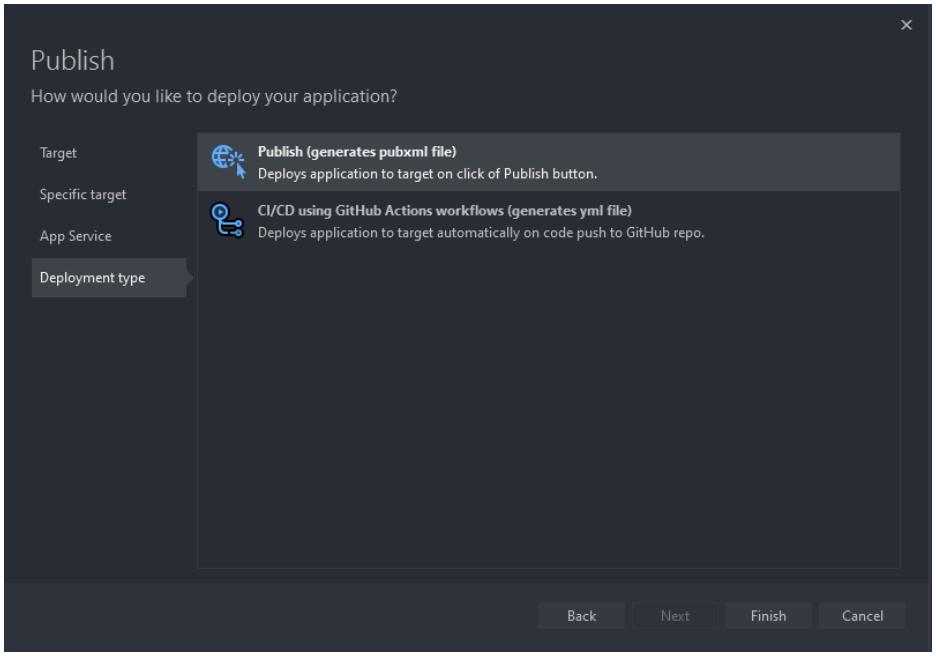
Kuva 43 Sovellusta ylläpitävän palvelun valinta.

Seuraavaksi valitaan rg-sata-laitepassi-app -resurssiryhmän sisälle valmiiksi luotu Azure App Service, jonka sisällä Laitepassi-sovellusta suoritetaan (Kuva 44). Tätä varten täytyy kehittäjän olla kirjautunut Azure -tilillensä ja valita sovellusta varten tehty Azure-tilaus. Seuraavaksi valitaan tapa, miten sovellus julkaistaan Azure -pilviympäristöön (Kuva 45).



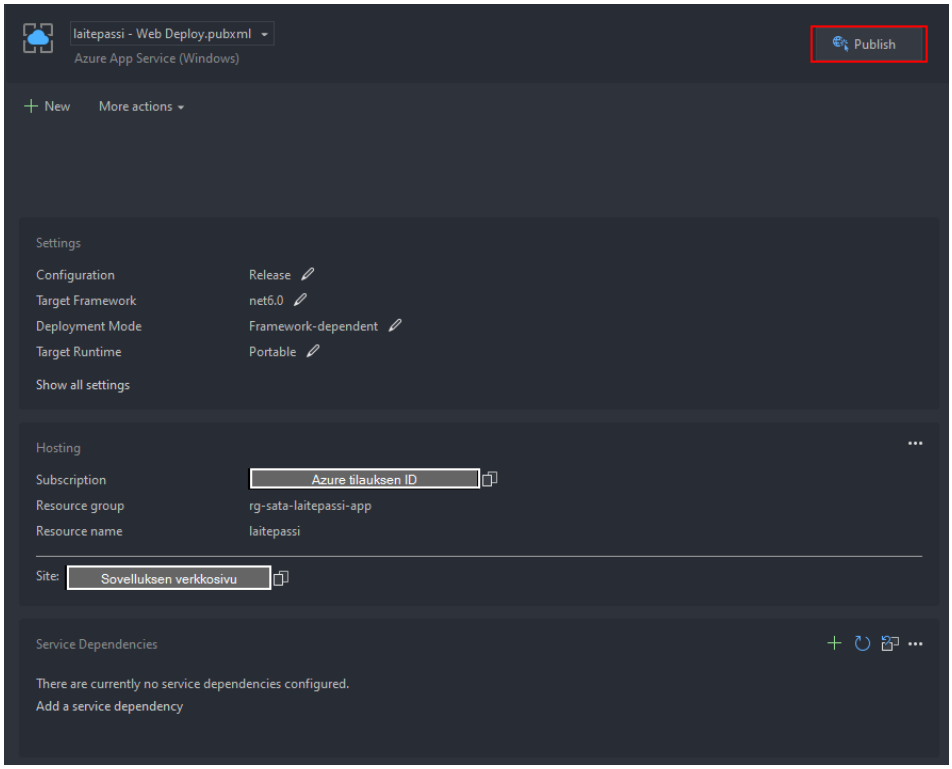
Kuva 44 Azure App Servicen valinta.

Julkaisutiedoston luonti testi- ja kehitysympäristöille on samanlainen kuin tuotantoympäristölle, erona on vain kuvassa 44 valittava Azure App Service.



Kuva 45 Julkaisutavan valinta.

Tämän jälkeen sovellus on julkaisua varten valmis (Kuva 46). Sovelluksen uusi .pubxml -julkaisutiedosto sisältää tarkistusta varten Azure -tilauksen ID:n, resurssiryhmän ja resurssin nimen. Lisäksi voidaan nähdä sivuston, jossa sovellus tulee toimimaan. Painettuaan Publish -painiketta, sovellus julkaistaan pilviympäristöön ja on käyttövalmis.

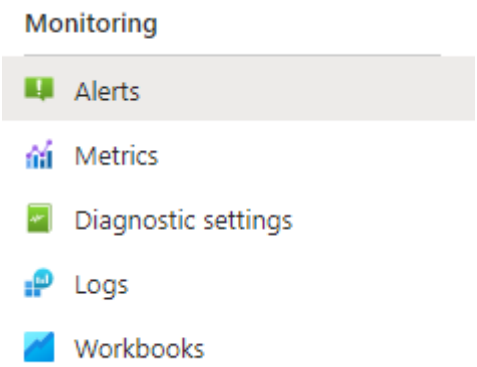


Kuva 46 Sovelluksen julkaisutiedosto.

Jatkossa jos kehittäjä tekee muutoksia sovelluksen koodiin, täytyy hänen vain hakea juuri luotu julkaisutiedosto ja painaa uudelleen Publish -painiketta. Täysin uutta julkaisutiedostoa ei tarvitse luoda. Azure Publishia käytetään jatkossa testi- ja kehitysympäristöihin julkaisussa. Tuotantoon siirrossa käytetään deployment slotien siirtoa (ks. luku 5.6).

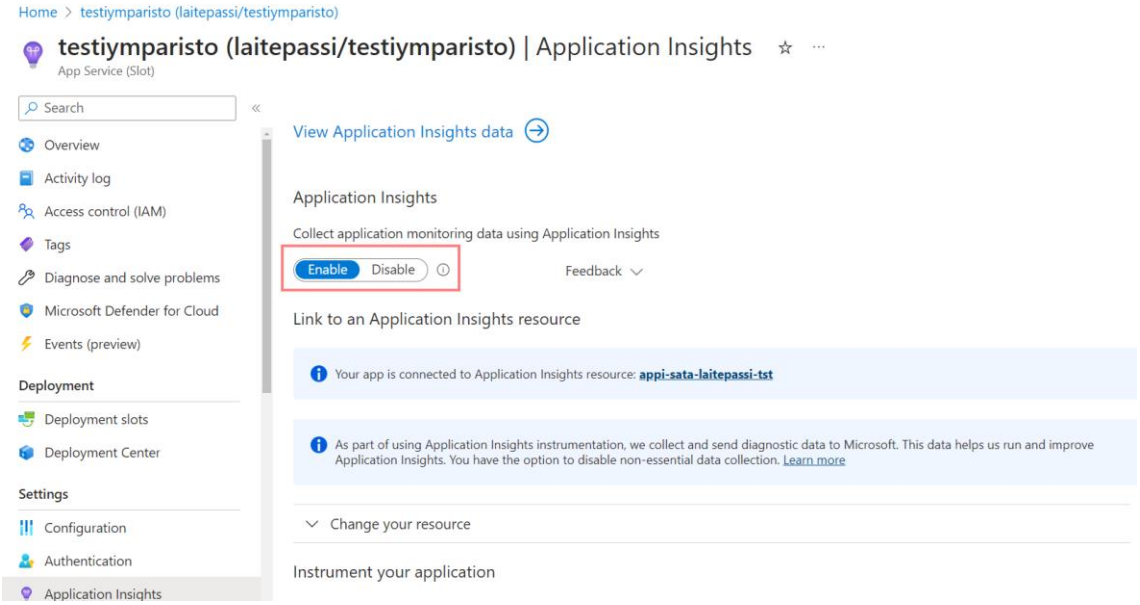
6.5 Uusi Lokitus

Sovelluksen uusi lokitus toteutetaan Azuren Application Insightsilla. Azure Application Insights otettiin käyttöön, koska se on Mcirosoftin valmis tuote www-sovelluksien monitorointiin ja hallintaan. Application Insightsin Monitorointi -osa (engl. Monitoring) sisältää tarvittavia työkaluja monitoroida haluttuja tapahtumia sovelluksessa (Kuva 47).



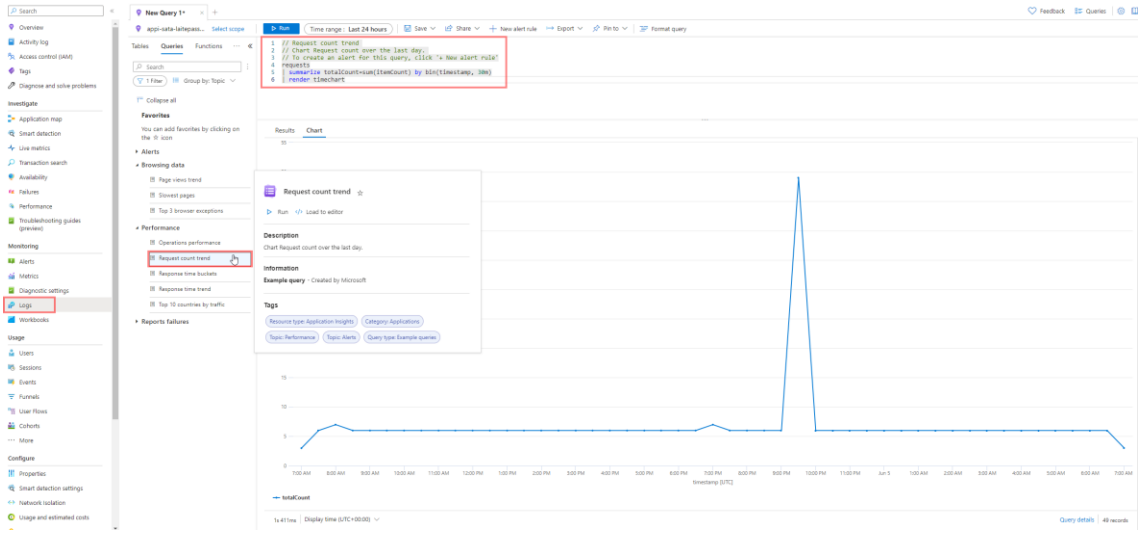
Kuva 47 Application Insights – Monitoring.

Application Insights voidaan ottaa käyttöön ilman riviäkään koodia. App Servicen kautta pitää vain kytkeä palvelu päälle. (Kuva 48)



Kuva 48 Application Insightin kytkeminen päälle.

Esimerkiksi Logs -sivulla voidaan luoda uusia kyselyitä tai käyttää olemassa olevia kyselyitä, jotka tuottavat sovelluksesta lokia. Kuvassa 49 käytetään valmista kyselyä, jossa näytetään kaavion muodossa käyttäjien lähettämien pyyntöjen määrän palvelimelle, viimeisen päivän aikana.



Kuva 49 Logs-kyselyn luonti.

Kyselyitä voidaan luoda vastaamaan vanhan sovellusversion käytössä olleita kyselyitä. Voidaan luoda Logs-sivulla uusi kysely, joka tuottaa lokia esimerkiksi kirjautuvien käyttäjien sähköposteista ja sovellukseen kirjautumisajankohdista.

7 Päätelmät

Uudistuneen Laitepassi-sovelluksen siirto Microsoft Azure -pilviympäristöön onnistui, mutta modernisointiprosessi ei ollut yhtä suoraviivainen kuin alun perin oletettiin. Vaikka Azure-ympäristön pystyttäminen sujui helposti Microsoftin tarjoaman dokumentaation avulla, itse sovelluksen modernisointi osoittautui haastavammaksi, erityisesti ohjelmointipuolen päivityksessä. Koin käyttäjän tunnistautumisen päivittämisen erityisen haastavaksi.

Laitepassi-sovelluksen julkaiseminen Azure-pilviympäristöön ei tuottanut ongelmia, mutta ohjelmointipuolella modernisointiprosessi vaati työtä ja aikaa. Tämä johtuu pääosin siitä, että kaikki oli kehitystiimille melko uutta. Kuitenkin onnistuneen siirron myötä sovellus hyötyy nyt Azure-ympäristön tarjoamista eduista, kuten paremmasta skaalautuvuudesta, ylläpidosta ja valmiista palveluista.

Vaikka modernisointiprosessi toi mukanaan omia haasteita, sovelluksen kehitystiimi sai tärkeää oppia ja kokemusta sovelluksen modernisoinnista Azureen. Aikaisemmin olen työskennellyt Azuren kanssa opiskelun merkeissä ja opiskelu oli melko teoreettista. Tämä projekti tarjosi paljon enemmän oppimiskokemuksia, niin teoreettisesti kuin käytännöllisesti, kun työskentely oli monipuolisempaa ja tunsin enemmän painetta projektin onnistumisesta.

Nyt, kun projektin tavoitteet on saavutettu ja sovellus on valmis, tunnen, että opinnäytetyö voi toimia hyödyllisenä oppimisvälineenä vastaavanlaisten projektien läpiviennissä, olivat ne sitten 2M-IT:n sisäisiä tai ulkoisia. On tärkeää käsitellä opinnäytetyössä myös kehitettyjä työkaluja ja menetelmiä, joita voitaisiin käyttää vastaavissa projekteissa tulevaisuudessa. Näin 2M-IT:n kehitystiimi voi jakaa oppimaansa ja parantaa organisaation kykyä modernisoida sovelluksiaan tehokkaasti Azure-ympäristöön.

Lähteet

Acromedia, 2022. The risks of running end-of-life software. Viitattu 20.3.2023. https://www.acromedia.com/article/the-risks-with-running-end-of-life-software

Amazon Web Services, n.d.a What Is A RESTful API? Viitattu 13.4.2023. https://aws.amazon.com/what-is/restful-api/

Amazon Web Services, n.d.b What is an IDE (Integrated Development Environment). Viitattu 24.2.2023 https://aws.amazon.com/what-is/ide/

Amazon Web Services, n.d.c What Is Multi-Factor Authentication (MFA)? Viitattu 12.4.2023. https://aws.amazon.com/what-is/mfa/

Azure, n.d. What is PaaS? Viitattu 10.4.2023. https://azure.microsoft.com/en-gb/resources/cloud-computing-dictionary/what-is-paas/

Digital Guardian, 2023. What is Role-Based Access Control (RBAC)? Viitattu 15.4.2023. https://www.digitalguardian.com/blog/what-role-based-access-control-rbac-examples-benefits-and-more

GeeksforGeeks, 2023a. Differences Between .NET Core and .NET Framework. Viitattu 5.3.2023. https://www.geeksforgeeks.org/differences-between-net-core-and-net-framework/

GeeksforGeeks, 2023b. Frontend vs backend. Viitattu 6.3.2023. https://www.geeksforgeeks.org/frontend-vs-backend/

Github Docs, 2023. GitHub. Viitattu 6.3.2023. https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-branches

Google Cloud, n.d.a Advantages and Disadvantages of Cloud Computing. Viitattu 10.4.2023. https://cloud.google.com/learn/advantages-of-cloud-computing#section-2

Google Cloud, n.d.b What is a relational database? Viitattu 26.3.2023. https://cloud.google.com/learn/what-is-a-relational-database

Google Cloud. n.d.c What is a Virtual Server? Viitattu 14.3.2023. https://cloud.google.com/learn/what-is-a-virtual-server

IBM, n.d. What are virtual machines (VMs)? Viitattu 15.4.2023. https://www.ibm.com/topics/virtual-machines

InterviewBit, 2022. .NET Core vs .NET Framework. Viitattu 14.3.2023. https://www.interviewbit.com/blog/net-core-vs-net-framework/

Javatpoint, n.d. What is Microsoft Azure Subscription? Viitattu 13.5.2023. https://www.javatpoint.com/what-is-microsoft-azure-subscription

jQuery, n.d. jQuery API. Viitattu 24.3.2023. https://api.jquery.com/

Learn Entity Framework Core, 2023. Welcome to Learn Entity Framework Core. Viitattu 29.2.2023. https://www.learnentityframeworkcore.com/

Loginradius, n.d. Cookie-based vs. Cookieless Authentication: What's the Future? Viitattu 12.5.2023. https://www.loginradius.com/blog/engineering/cookie-based-vs-cookieless-authentication/

MDN Web Docs, 2023a. CSS: Cascading Style Sheets. Viitattu 6.3.2023. https://developer.mozilla.org/en-US/docs/Web/CSS

MDN Web Docs, 2023b. HTML: HyperText Markup Language. Viitattu 6.3.2023. https://developer.mozilla.org/en-US/docs/Web/HTML

MDN Web Docs, 2023c. HTTP response status codes. Viitattu 6.3.2023. https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

MDN Web Docs, 2023d. JavaScript. Viitattu 6.3.2023. https://developer.mozilla.org/en-US/docs/Web/JavaScript

MDN Web Docs, 2023e. Working with JSON. Viitattu 12.5.2023. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON

Microsoft, n.d. Azure Active Directory (Azure AD). Viitattu 12.4.2023. https://azure.microsoft.com/en-us/products/active-directory#overview

Microsoft, 2020. Token configuration in Azure AD app registrations now generally available. Viitattu 14.5.2023.

https://devblogs.microsoft.com/microsoft365dev/token-configuration-in-azure-ad-app-registrations-now-generally-available/

Microsoft, 2022a. Windows Server. Viitattu 14.3.2023. https://learn.microsoft.com/en-us/windows/win32/srvnodes/windows-server

Microsoft, 2022b. Azure for developers overview. Viitattu 22.4.2023. https://learn.microsoft.com/en-us/azure/developer/intro/azure-developer-overview

Microsoft, 2022c. Forms Authentication in ASP.NET Web API. Viitattu 14.3.2023. https://learn.microsoft.com/en-us/aspnet/web-api/overview/security/forms-authentication

Microsoft, 2022d. Overview of ASP.NET Core MVC. Viitattu 14.3.2022. https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?WT.mc_id=dotnet-35129-website&view=aspnetcore-7.0

Microsoft, 2022e. Quickstart: Configure a client application to access a web API. Viitattu 14.3.2022. https://learn.microsoft.com/en-us/azure/active-directory/develop/quickstart-configure-app-access-web-apis

Microsoft, 2022f. Scale up an app in Azure App Service. Viitattu 22.4.2023. https://learn.microsoft.com/en-us/azure/app-service/manage-scale-up

Microsoft, 2022g. Shared responsibility in the cloud. Viitattu 22.4.2023. https://learn.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility

Microsoft, 2022h. Three options to prepare for Windows Server 2012/R2 end of support. Viitattu 14.4.2023. https://techcommunity.microsoft.com/t5/windows-server-news-and-best/three-options-to-prepare-for-windows-server-2012-r2-end-of/ba-p/3645211

Microsoft, 2022i. What is PowerShell? Viitattu 18.4.2023. https://learn.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.3

Microsoft, 2023a. A tour of the C# language. Viitattu 6.3.2023. https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

Microsoft, 2023b. App Service overview. Viitattu 15.4.2023. https://learn.microsoft.com/en-us/azure/app-service/overview

Microsoft, 2023c. Application and service principal objects in Azure Active Directory. Viitattu 13.4.2023. https://learn.microsoft.com/en-us/azure/active-directory/develop/app-objects-and-service-principals#application-registration

Microsoft, 2023d. Application Insights overview. Viitattu 20.4.2023. https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview?tabs=net

Microsoft, 2023e. Azure App Service plan overview. Viitattu 15.4.2023. https://learn.microsoft.com/en-us/azure/app-service/overview-hosting-plans

Microsoft, 2023f. Databases. Viitattu 26.3.2023. https://learn.microsoft.com/en-us/sql/relational-databases/databases/databases?view=sql-server-ver16

Microsoft, 2023g. Download SQL Server Management Studio (SSMS). Viitattu 28.4.2023. https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16

Microsoft, 2023h. Introduction to Azure Functions. Viitattu 17.4.2023. https://learn.microsoft.com/en-us/azure/azure-functions/functions-overview

Microsoft, 2023i. Manage Azure Active Directory groups and group membership. Viitattu 15.4.2023. https://learn.microsoft.com/en-us/azure/active-directory/fundamentals/how-to-manage-groups

Microsoft, 2023j. Manage Azure resource groups by using the Azure portal. Viitattu 15.4.2023. https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/manage-resource-groups-portal

Microsoft, 2023k. Overview of Microsoft Graph. Viitattu 15.4.2023. https://learn.microsoft.com/en-us/graph/overview

Microsoft, 2023l. Set up staging environments in Azure App Service. Viitattu 15.4.2023. https://learn.microsoft.com/en-us/azure/app-service/deploy-staging-slots

Microsoft, 2023m. Visual Studio, It’s how you make software. Viitattu 6.3.2023. https://visualstudio.microsoft.com/#vs-section

Microsoft, 2023n. Windows Server 2012 and 2012 R2 reaching end of support. Viitattu 14.4.2023. https://learn.microsoft.com/en-us/lifecycle/announcements/windows-server-2012-r2-end-of-support

Microsoft, 2023o. What are ARM templates? Viitattu 12.5.2023. https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/overview

Microsoft, 2023p. What are Azure regions and availability zones? Viitattu 23.5.2023. https://learn.microsoft.com/en-us/azure/reliability/availability-zones-overview

Microsoft, 2023q. What is .NET? Introduction and overview. Viitattu 22.3.2023. https://learn.microsoft.com/en-us/dotnet/core/introduction

Microsoft, 2023r. What is Azure Active Directory authentication? Viitattu 22.3.2023. https://learn.microsoft.com/en-us/azure/active-directory/authentication/overview-authentication

Microsoft, 2023s. What is Azure Logic Apps? Viitattu 17.4.2023. https://learn.microsoft.com/en-us/azure/logic-apps/logic-apps-overview

Microsoft, 2023t. What is Bicep? Viitattu 12.5.2023. https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/overview?tabs=bicep

Microsoft, 2023u. What is Conditional Access? Viitattu 12.4.2023. https://learn.microsoft.com/en-us/azure/active-directory/conditional-access/overview

Microsoft, 2023v. What is the Azure CLI? Viitattu 12.5.2023. https://learn.microsoft.com/en-us/cli/azure/what-is-azure-cli

Microsoft, 2023w. .NET and .NET Core support Policy. Viitattu 22.3.2023. https://dotnet.microsoft.com/en-us/platform/support/policy/dotnet-core

Softensity, 2022. Differences Between .NET Core 3.1 and .NET Core 6. Viitattu 22.3.2023. https://www.softensity.com/blog/differences-between-net-core-3-1-and-net-core-6/

ochzhen.com, n.d. Create Resource Group With Azure Bicep and Deploy Resources In It. Viitattu 18.5.2023. https://ochzhen.com/blog/create-resource-group-azure-bicep

Opal, n.d. Group-based access control (GBAC). Viitattu 19.5.2023. https://opal.dev/glossary/group-based-access-control-gbac

TechTarget, 2019a. Internet Information Services (IIS). Viitattu 22.3.2023. https://www.techtarget.com/searchwindowsserver/definition/IIS

TechTarget, 2019b. Microsoft SQL Server. Viitattu 22.3.2023. https://www.techtarget.com/searchdatamanagement/definition/SQL-Server

TechTarget, 2021a. Active directory. Viitattu 22.3.2023. https://www.techtarget.com/searchwindowsserver/definition/Active-Directory

TechTarget, 2021b. command-line interface (CLI). Viiattu 25.5.2023. https://www.techtarget.com/searchwindowsserver/definition/command-line-interface-CLI

TechTarget, 2021c. runtime. Viitattu 12.4.2023. https://www.techtarget.com/searchsoftwarequality/definition/runtime

TechTarget, 2021d. Uniform Resource Identifier (URI). Viitattu 14.3.2023. https://www.techtarget.com/whatis/definition/URI-Uniform-Resource-Identifier

TechTarget, 2021e. user interface (UI). Viitattu 6.3.2023. https://www.techtarget.com/searchapparchitecture/definition/user-interface-UI

TechTarget, 2022a. Application Programming Interface. Viitattu 14.3.2023. https://www.techtarget.com/searchapparchitecture/definition/application-program-interface-API

TechTarget, 2022b. Bootstrap. Viitattu 12.4.2023. https://www.techtarget.com/whatis/definition/bootstrap

TechTarget, 2022c. Single sign-on. Viitattu 12.4.2023. https://www.techtarget.com/searchsecurity/definition/single-sign-on

TechTarget, 2022d. Software Development Kit (SDK). Viitattu 12.4.2023. https://www.techtarget.com/whatis/definition/software-developers-kit-SDK

TechTarget, 2022e. Structured Query Language (SQL). Viitattu 12.4.2023. https://www.techtarget.com/searchdatamanagement/definition/SQL

TutorialsTeacher, n.d. ASP.NET Core Overview. Viitattu 22.3.2023 https://www.tutorialsteacher.com/core/aspnet-core-introduction

W3Schools, n.d. What is HTTP? Viitattu 6.3.2023. https://www.w3schools.com/whatis/whatis_http.asp

Wikipedia, 2022. 2M-IT. Viitattu 12.3.2023 https://fi.wikipedia.org/wiki/2M-IT

Wikipedia, 2023. GitHub. Viitattu 12.3.2023 https://fi.wikipedia.org/wiki/GitHub

2M-IT. n.d. Viitattu 12.3.2023 https://2m-it.fi/