Ohjelmistokehittäjän päiväkirja

Yasin Rahimu

Haaga-Helia ammattikorkeakoulu
Tradenomi, tietojenkäsittely
Amk-opinnäytetyö
2025

Tiivistelmä

Tekijä(t)
Yasin Rahimu

Tutkinto
Tradenomi

Raportin/Opinnäytetyön nimi
Ohjelmistokehittäjän päiväkirja

Sivu- ja liitesivumäärä
45 + 0

Opinnäytetyössä tarkasteltiin ohjelmistokehittäjän ammatillista kehittymistä kahdeksan viikon ajan päiväkirjamuotoisesti. Työ toteutettiin vakituisessa ohjelmistokehittäjän roolissa, jonka alussa osallistuttiin IT-alan yrityksen tarjoamaan koulutus- ja perehdytysohjelmaan, ja jonka aikana työskentely tapahtui pilvipohjaisessa integraatiokehitystiimissä. Opinnäytetyön tavoitteena oli seurata ja analysoida, miten tekijän tekninen ja ammatillinen osaaminen kehittyi uran alkuvaiheessa päivittäisen oppimisen ja ohjatun perehdytyksen kautta. Tarkastelun pääteemoina olivat integraatiokehitykseen perehtyminen, C#-ohjelmointikielen hallinnan vahvistaminen sekä ammatillinen sopeutuminen ohjelmistokehittäjän uran alkuvaiheessa.

Työn keskeinen tietoperusta koostui kirjallisuudesta ja digitaalisista lähteistä, jotka käsittelivät pilvipohjaista integraatiokehitystä, C#-ohjelmointikieltä ja ammatillista kehittymistä uran alkuvaiheessa. Teoriaosuudessa esiteltiin integraatioratkaisujen periaatteita ja teknologioita, kuten pilvipalveluiden integraatiotyökaluja. Lisäksi tarkasteltiin C#-kielen keskeisiä piirteitä sekä työelämään sopeutumista edistäviä tekijöitä. Tietoperusta tarjosi viitekehyksen, jonka pohjalta tekijä peilasi omia kokemuksiaan ja oppimistaan viikoittaisten merkintöjen analyysissä.

Opinnäytetyö toteutettiin päiväkirjamuotoisena seuranta- ja analyysiprosessina kahdeksan viikon seurantajaksolla keväällä 2025. Tekijä kirjasi viikoittain ylös oppimansa asiat, suorittamansa työtehtävät sekä kohtaamansa haasteet ja peilasi niitä opinnäytetyön pääteemoihin. Seurantajakso sisälsi sekä yrityksen järjestämiä perehdytyskoulutuksia (pilvipohjaisista integraatiotyökaluista) sekä käytännön ohjelmistokehitystehtäviä integraatiotiimissä. Aineisto koostui viikoittaisista päiväkirjamerkinnöistä ja niiden pohjalta tehdystä analyysistä. Menetelmänä toimi itsereflektio eli oman oppimisprosessin systemaattinen tarkastelu työympäristössä.

Kahdeksan viikon seurannan tuloksena tekijän osaaminen integraatiokehittämisessä syventyi huomattavasti. Tekijä oppi hyödyntämään pilvipohjaisia integraatiotyökaluja ja ymmärtämään ohjelmistointegraatioiden suunnittelua sekä toteutusta entistä paremmin. C#-ohjelmointikielen hallinta vahvistui käytännön harjoittelun myötä. Aiempi kokemus muista ohjelmointikielistä edesauttoi uuden kielen omaksumista. Työyhteisöön sopeutuminen sujui hyvin. Tekijä omaksui tiimityöskentely- ja viestintätapoja, kartutti itsevarmuutta asiantuntijaroolissa ja hahmotti paremmin konsultointityön kokonaisuutta. Päiväkirjaseuranta osoitti, että ohjattu käytännön työskentely yhdistettynä itsereflektioon tuki tehokkaasti ammatillista kehittymistä. Opinnäytetyön tavoitteet saavutettiin ja saadut kokemukset loivat tekijälle vahvan pohjan jatkaa uraansa ohjelmistokehittäjänä.

Asiasanat
Ohjelmistokehitys, Integraatiot, Pilvipalvelut, C#-ohjelmointikieli, Azure, Ammatillinen kehittyminen

Sisällys

1 Johdanto

Päiväkirjamuotoinen opinnäytetyöni seuraa omaa kahdeksan viikon mittaista matkaani kansainvälisen IT-konsultointiyrityksen integraatiotiimissä. Toimin virallisesti Developer-nimikkeellä, joka suomalaisessa työympäristössä tarkoittaa ohjelmistokehittäjän roolia, eli vastuuta ohjelmistojen suunnittelusta, toteutuksesta, testauksesta ja olemassa olevan koodin ylläpidosta. Integraatiotiimissä vastuuni painottuu erityisesti järjestelmien välisiin rajapintoihin ja datan siirtämiseen, eli rakennan ohjelmistoja ja komponentteja, jotka mahdollistavat eri järjestelmien yhteistoiminnan luotettavasti ja turvallisesti.

Vaikka olenkin opintojen alusta asti ollut enemmän kiinnostunut ohjelmistokehityksen backend-puolesta, olen ajautunut projekteissa käytännössä kokonaan frontend-painotteiseen kehitykseen. Backend ja integraatiot ovat jäänet selvästi vähemmälle huomiolle, siksi jo tässä opinnäytetyöjakson alussa tuntuu, että lähden liikkeelle hyvin alkeellisesta asemasta. C#-ohjelmointi, Azuren pilvipalvelut ja järjestelmien välisten yhteyksien rakentaminen on minulle täysin uusia aiheita. Tämä lähtökohta tuo mukaansa oman haasteensa, mutta otan sen mielelläni vastaan.

Asetan itselleni kolme keskeistä oppimistavoitetta, jotka suuntaavat työskentelyäni integraatiotiimissä. Ensimmäinen tavoite on syventää osaamista C#-ohjelmointikielessä. C# on pääsääntöisesti käytetty kieli yrityksen integraatioratkaisussa, joten vahvempi C#-osaaminen auttaa minua etenemään omassa roolissani. Koen myös C#-kielen olevan hyvä taito tulevaisuuden kannalta. Toinen oppimistavoitteeni on perehtyä Microsoft Azure -pilvipalvelualustaan ja sen tarjoamiin erilaisiin ratkaisuihin integraatiokehityksen kannalta. Yrityksen integraatioalusta pohjautuu pilvipalveluihin, erityisesti Azureen, joten tuntemus tästä on välttämätöntä, jotta pääsen saavuttamaan onnistumisia. Tästäkään minulla ei ole entuudestaan kokemusta, joten matka tulee olemaan kivinen. Kolmantena tavoitteena on sopeutua uuteen työympäristöön uudella alalla. Haluan oppia miten, toimia konsulttina osana asiantuntijatiimiä, omaksua uuden alan käytännöt ja selvittää, miten sopeudun yleisesti ohjelmistokehityksen maailmassa. Valitsin nämä tavoitteet sekä projektin vaatimusten, että oman ammatillisen kasvuni tarpeiden perusteella.

Teen opinnäytetyön päiväkirjamuotoisena, mikä tarkoittaa sitä, että kirjoitan jokaisen työpäivän keskeiset tehtävät, havainnot ja oppimiskokemukset ylös. Viikoittain käyn läpi viikkoanalyyseissa keskeisimmät asiat reflektoiden niitä suhteessa omiin tavoitteisiin. Koen päiväkirjamuotoisen lähestymistavan olevan paras tämänhetkiseen elämäntilanteeseeni ja se mahdollistaa jatkuvaa palautetta ja itsearviointia. Pystyn reaaliaikaisesti tarkkailla omaa kehitystäni ja mahdollisesti myöhemmin palata tähän ja verrata miten urani on lähtenyt käyntiin, sekä mitä kaikkea on tullut opittua.

Seuraavissa luvuissa käyn viikko viikolta läpi kokemuksiani ja oppejani. Lopuksi pohdin kokonaisvaltaisesti, miten kehitykseni on edennyt ja missä määrin saavutin asettamani tavoitteet tämän opinnäytetyöjakson aikana.

Seuraavassa luvussa esitellään opinnäytetyön keskeisiä asiasanoja.

1.1 Asiasanasto

Artifacts
Sovelluksen rakennuksen (build) aikana syntyviä tiedostoja, kuten asennuspaketteja tai kirjastoja, joita voidaan käyttää testauksessa tai julkaista käyttäjille.

Azure
Microsoftin pilvipalvelualusta, joka tarjoaa palveluita esimerkiksi sovellusten suorittamiseen, tallennukseen ja järjestelmien integrointiin.

Azure DevOps
Microsoftin pilvipohjainen palvelukokonaisuus ohjelmistokehityksen elinkaaren hallintaan (mm. versionhallinta, CI/CD, projektinhallinta).

Azure Functions
Azuren palvelimeton (serverless) laskentapalvelu, joka suorittaa pieniä funktioita pilvessä tapahtumien perusteella.

Azure Logic Apps
Azuren integraatiopalvelu, jolla luodaan automatisoituja työnkulkuja eri järjestelmien välillä visuaalisesti.

Azure Pipelines
Azure DevOps -palvelun osa, joka mahdollistaa CI/CD-prosessien (rakennus, testaus, julkaisu) automatisoinnin.

Azure Service Bus
Azuren viestinvälityspalvelu sovellusten väliseen viestintään viestijonojen ja aiheiden (topics) avulla.

Backend
Backend on ohjelmiston taustajärjestelmä, joka toimii palvelimella ja huolehtii esimerkiksi tietojen tallennuksesta, liiketoimintalogiikasta ja rajapinnoista muihin järjestelmiin.

Bicep
Azure-resurssien määrityskieli (Infrastructure as Code), jonka avulla pilvi-infrastruktuuri kuvataan ja otetaan käyttöön koodina.

Bitbucket
Atlassianin ylläpitämä Git-versionhallintapalvelu ja koodivarasto (repository) pilvessä.

Board
Projektinhallinnan visuaalinen työlista (esim. Azure Boards), jonka avulla tiimit seuraavat tehtäviä ja työnkulkua.

Build
Prosessi, jossa sovellus käännetään ja paketoidaan ajettavaksi ohjelmaksi tai asennuspaketiksi. Synonyyminä käytetään myös termiä sovelluksen rakentaminen.

Business Central (BC)
Microsoftin toiminnanohjausjärjestelmä (ERP) pienille ja keskisuurille yrityksille, osa Dynamics 365 -kokonaisuutta.

C#
Microsoftin .NET-alustalle kehittämä olio-ohjelmointikieli.

CI/CD (Continuous Integration / Continuous Deployment)
Kehitysmalli, jossa koodimuutokset yhdistetään ja julkaistaan automaattisesti (rakennus, testaus, käyttöönotto).

CRM
Customer Relationship Management; asiakkuudenhallintajärjestelmä yrityksen ja asiakkaiden vuorovaikutuksen hallintaan.

Continious deployment
Käytäntö, jossa koodimuutokset viedään automaattisesti tuotantoon ilman manuaalista hyväksyntää.

Dataverse
Microsoftin pilvipohjainen tietovarasto, joka mahdollistaa tietojen tallentamisen ja jakamisen esimerkiksi Power Platformissa.

Deserialisointi
Tallennetun tiedon (esim. JSON, XML) muuntaminen takaisin ohjelman sisäiseksi tietorakenteeksi.

DNS
Domain Name System, internetin nimipalvelu, joka muuntaa verkkotunnukset (domain) IP-osoitteiksi.

Dreyfusin kompetenssimalli
Malli, joka kuvaa asiantuntijuuden kehittymistä vaiheittain noviisista asiantuntijaksi.

DTO (Data Transfer Object)
Yksinkertainen tietorakenne, jolla siirretään dataa sovelluksen osien välillä ilman toimintalogiikkaa.

Entra ID
Microsoftin pilvipohjainen identiteetti- ja pääsynhallintapalvelu (aiemmin Azure AD).

Endpoint
Rajapintapiste (esim. URL-osoite), jonka kautta sovellukseen voidaan ottaa yhteyttä ja käsitellä tietoa.

Entity Framework
.NET-ympäristön ORM-kehys relaatiotietokantojen käsittelyyn olioina.

ERP
ERP (Enterprise Resource Planning) on toiminnanohjausjärjestelmä, joka yhdistää ja hallinnoi yrityksen keskeisiä liiketoimintaprosesseja, kuten taloutta, varastoa ja henkilöstöä.

Frontend
Frontend tarkoittaa ohjelmiston tai verkkosovelluksen osaa, jonka käyttäjä näkee ja jonka kanssa hän on vuorovaikutuksessa. Frontendin- teknologioita ovat esimerkiksi HTML, CSS ja JavaScript.

Git
Hajautettu versionhallintajärjestelmä, jolla kehittäjät voivat hallita, seurata ja yhdistää koodimuutoksia.

Git-haara (branch)
Gitissä rinnakkainen kehityslinja, jossa voi tehdä muutoksia ilman, että ne vaikuttavat päähaaraan.

GitFlow
Git-versionhallinnan toimintamalli, jossa kehitystyö jaetaan useisiin haaroihin (master, develop, feature, release, hotfix).

HTTP-triggeri
Mekanismi, joka käynnistää tietyn toiminnon saapuvan HTTP-pyynnön perusteella (esim. Azure Functions).

HR
HR (Human Resources, henkilöstöhallinto) tarkoittaa yrityksen toimintoa, joka vastaa henkilöstöasioista, kuten rekrytoinnista, työsuhteista, palkkauksesta, koulutuksesta ja henkilöstön hyvinvoinnista.

Idempotentti
Toiminto, joka tuottaa saman lopputuloksen riippumatta siitä, kuinka monta kertaa se suoritetaan samalla syötteellä (esim. tietyn resurssin päivitys).

Infrastructure as Code (IaC)
Menetelmä, jossa infrastruktuuri määritellään ja hallitaan koodin avulla (esim. Bicep, ARM, Terraform).

Integraatio
Integraatio tarkoittaa eri järjestelmien, sovellusten tai palveluiden yhdistämistä niin, että ne voivat vaihtaa tietoa ja toimia yhdessä.

Intranet
Intranet on organisaation sisäinen verkkopalvelu, jota käytetään tiedonjakoon, sisäiseen viestintään ja yhteistyöhön vain organisaation jäsenten kesken. Intranet ei ole avoin yleisölle.

JSON
Kevyt tietojenvaihtoformaatti, jossa data esitetään avain–arvopareina.

Kerrosarkkitehtuuri
Ohjelmiston rakenne, jossa järjestelmä jaetaan useisiin tehtäväkohtaisiin kerroksiin.

Lokitus
Toiminto, jossa järjestelmän tapahtumista ja virheistä tallennetaan tietoa (lokeja) jäljitettävyyden ja virheenkorjauksen tueksi.

MFA (Multi-Factor Authentication)
Monivaiheinen tunnistautuminen, jossa käyttäjä todistaa henkilöllisyytensä useammalla tavalla.

Mockkaus
Testausmenetelmä, jossa oikeat riippuvuudet korvataan testin ajaksi jäljitellyillä (mock) olioilla.

Moq
.NET-ympäristön kirjasto, jolla voidaan luoda testien ajaksi mock-olioita.

Oauth
Avoin valtuutusstandardi, joka mahdollistaa turvallisen tunnistautumisen ja resurssien käytön palveluiden välillä ilman salasanojen jakamista.

OData
Avoin protokolla REST-rajapintojen toteuttamiseen, mahdollistaa tiedon kyselyn ja muokkauksen HTTP:n yli.

Pull request
Ehdotus koodimuutosten yhdistämisestä toiseen haaraan, jonka toinen kehittäjä tarkistaa ja hyväksyy.

Refaktorointi
Koodin rakenteen parantaminen ilman sen toiminnallisuuden muuttamista, tavoitteena esimerkiksi luettavuuden tai ylläpidettävyyden lisääminen.

Release
Ohjelmiston julkaisu- tai jakeluprosessi, jossa valmis sovellus viedään käyttäjille.

Repository (repo, koodivarasto)
Kokoelma tiedostoja ja niiden kehityshistoriaa, jota hallitaan versionhallinnassa.

Skeema
Rakennemääritys, joka kertoo, mitä kenttiä ja arvoja tietorakenne (esim. tietokanta, XML, JSON) saa sisältää.

Scrum
Kettereiden ohjelmistoprojektien hallintamenetelmä, jossa työ jaetaan lyhyisiin iteratiivisiin jaksoihin (sprintteihin) ja tiimi käy säännöllisesti läpi edistymistä sekä suunnitelmia.

Serialisointi
Tietorakenteiden muuntaminen siirrettävään tai tallennettavaan muotoon (esim. JSON, XML).

Validointi
Tietojen tarkistaminen ja varmistaminen, että ne täyttävät tietyt säännöt, muodot ja vaatimukset.

Webhook

Tapa lähettää automaattisesti tietoa toiseen järjestelmään HTTP-pyynnön avulla, kun jokin tapahtuma laukeaa.

Web API

Web-sovellusrajapinta, jonka kautta ohjelmistot voivat tarjota toimintojaan tai tietojaan muiden hyödynnettäväksi internetin yli.

XML

Merkkipohjainen kuvauskieli, jossa tieto esitetään tageilla hierarkkisesti.

XSD

XML-skeeman määrittelykieli, jolla kuvataan XML-dokumentin sallittu rakenne.

Yksikkötestaus

Testauksen muoto, jossa yksittäinen koodiyksikkö (funktio, luokka) testataan erillään muusta järjestelmästä.

YAML

YAML (YAML Ain't Markup Language) on helposti luettava tiedon kuvaus- ja konfiguraatiomuoto, jota käytetään erityisesti sovellusten asetustiedostoissa ja tiedonsiirrossa.

2 Lähtötilanteen kuvaus

2.1 Oman nykyisen työ analysointi

Aloitan työskentelyn ohjelmistokehittäjän roolissa ja työni alkuvaiheessa, noin puolentoista kuukauden ajan, tulen keskittymään lähes kokonaan yrityksen järjestämään perehdytys- ja koulutusjaksoon. Käytännössä päivät täyttyvät alkuun koulutuksista, käytännön harjoituksista sekä tiimin ja yrityksen toimintatapojen opettelusta.

Ammatillinen kehitykseni tällä hetkellä on aivan lähtötasossa. En osaa vielä kunnolla käyttää yrityksessä käytettyjä oleellisia teknologioita ja työkaluja. Nimenomaan C#-kielen, pilvipalveluiden, ohjelmistoarkkitehtuurin ja integraatioiden osaamineni on rajallista. Esimieheni ja koulutusohjelman ohjaajat ovatkin painottaneet vahvasti, että minun tulee alkuvaiheessa keskittyä erityisesti oman osaamisen kehittämiseen, eikä miettiä laajempia koulutuksen ulkopuolisia tehtäviä.

Uskon että tulen jo koulutusjaksolla saamaan hyvän perustan ohjelmistokehittäjän toimimiselle. Ja oman oppimiseni tuntien, tiedän, että ensimmäinen konkreettinen projekti tulee selkeyttämään hyvin paljon asioita minulle. Opin eniten tekemällä ja tämä auttaa minua laittamaan teoriassa opitut asiat mielessäni paikoilleen. En vältä sosiaalisia kontakteja, joten en usko, että avun pyytäminen tulee olemaan minulle ongelma. Päinvastoin uskon tulevani juttelemaan paljon kollegoiden kanssa mahdollisista ratkaisuista, joita minulla tulee olemaan käsillä. Kaiken kaikkiaan lähtökohtani on ollut se, että työni alku on nimenomaan uuden oppimista, ja osaamiseni laajenee pääasiassa koulutuksen ja ohjatun harjoittelun kautta.

2.2 Sidosryhmien esittely

Käyn tässä luvussa läpi ne sidosryhmät, jotka ovat keskeisiä työtehtävieni ja ammatillisen kehittymiseni kannalta ohjelmistokehitystyössäni integraatiotiimissä. Tavoitteena on jäsentää työympäristön kannalta olennaisimmat vuorovaikutussuhteet ja toimijat, jotka tukevat oppimistani, urakehitystäni ja päivittäistä työskentelyäni osana projektitiimiä.

Lähin sidosryhmäni on oma nelihenkinen projektitiimini, jossa toimii ohjelmistoarkkitehti (tiiminvetäjä), yksi kokeneempi kehittäjä sekä toinen uran alkuvaiheessa oleva kehittäjä lisäkseni.

Toinen merkittävä sisäinen sidosryhmä on yksikkömme esihenkilö, joka toimii koko integraatioyksikön people leadina. Hänen kanssaan pidetään säännöllisiä 1:1-keskusteluja, joissa käsitellään esimerkiksi omaa kehittymistä, työhyvinvointia ja projektien etenemistä. Koko alkuvaiheessani hän myös ohjaa minua suoraan ensimmäisiin sisäisiin projekteihin, sekä avustaa niissä, että toimii yhteyshenkilönä yksikkötasolla järjestettävissä koulutuksissa ja tapaamisissa.

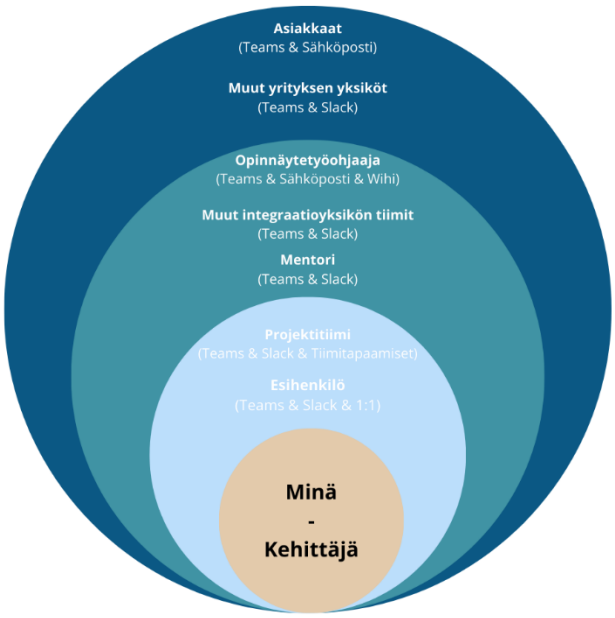
Lisäksi minulla on oma henkilökohtainen mentori, joka on matalan kynnyksen tukihenkilö ja jonka puoleen voi kääntyä arkipäiväisissä kysymyksissä. Mentorin rooli on ollut iso apu uran alkuvaiheessa, kun moni käytännön asia on vielä uutta. Hänen tukensa auttaa navigoimaan työyhteisössä ja ymmärtämään yrityksen toimintatapoja.

Laajemmin tarkasteltuna integraatioyksikön sisäiset muut tiimit ovat myös osa sidosryhmien verkostoa. Vaikka varsinaista projektipohjaista yhteistyötä muiden tiimien kehittäjien kanssa minulla ei ole vielä ollut, kysyn toisinaan apua niiltä, joilla on kokemusta samantyyppisistä projekteista. Integraatioarkkitehdit tekevät laajempaa yhteistyötä muiden yksiköiden, kuten Business Central-, Apps & Innovations- ja Data & AI -yksiköiden kanssa, erityisesti suunnittelu- ja arkkitehtuuritasolla. Tällä hetkellä nämä sidosryhmät ovat minulle etäisempiä, mutta uskon niiden merkityksen kasvavan työkokemuksen karttuessa.

Asiakasyhteistyötä ei ole tässä vaiheessa ollut, koska työni keskittyy sisäisiin integraatioihin ja koulutusprojekteihin. Jatkossa asiakasrajapinta voi kuitenkin muodostua merkittäväksi osaksi sidosryhmäkenttää.

Koulun puolelta sidosryhmään kuuluu opinnäytetyöohjaaja, jonka kanssa pidetään yhteyttä työn etenemisestä ja rakenteesta. Hänen palautteensa auttaa ohjaamaan työn fokusta ja kirjoitustapaa oikeaan suuntaan.

Päivittäinen viestintä ja yhteistyö tapahtuvat pääosin Teams-palavereiden ja Slack-viestien kautta. Slack toimii nopean viestinnän välineenä tiimin sisällä, kun taas Teamsin kautta hoidetaan etänä pidetyt palaverit, koulutukset ja 1:1-tapaamiset. Työskentely perustuu paljon oma-aloitteisuuteen, mutta apua on helppo saada, mikä luo turvallisen ja kannustavan ympäristön kehittymiselle.



Kuva 1. Sidosryhmäkaavio

Kuvaan 1 on koottu opinnäytetyöni kannalta keskeiset sidosryhmät kehämäiseen kaavioon. Kehien tarkoituksena on havainnollistaa, kuinka tiiviissä vuorovaikutuksessa eri sidosryhmien kanssa olen ollut opinnäytetyön aikana. Sisimmäisenä kehänä on oma roolini integraatiotiimin kehittäjänä. Lähimmälle kehälle sijoittuvat tiiviimmässä vuorovaikutuksessa olevat sidosryhmät, esihenkilö ja projektitiimi, joiden kanssa työskentelen päivittäin. Seuraavalle kehälle on sijoitettu harvemmin, mutta kuitenkin säännöllisesti yhteydessä olevat tahot, kuten muut integraatioyksikön tiimit, mentori sekä opinnäytetyöohjaaja. Näiden kanssa yhteistyö on satunnaisempaa, mutta ne tarjoavat tarvittaessa tukea ja ohjausta. Uloimmalle kehälle on puolestaan sijoitettu etäisimmät tai tulevaisuuden kannalta merkittävät sidosryhmät, kuten asiakkaat ja muut yrityksen yksiköt. Näiden kanssa yhteydenpito on vähäisempää, mutta niiden rooli voi korostua työuran edetessä.

2.3 Työpaikan vuorovaikutustilanteet

Työpaikalla vuorovaikutustilanteeni keskittyvät aluksi pääasiassa integraatioyksikön tiimiläisiin. Alkuvaiheessa suurin osa vuorovaikutuksesta tapahtuu koulutustilanteissa, joissa eri tiimin jäsenet pitävät meille koulutuksia eri aiheista. Näissä koulutuksissa käsitellään edellä mainittuja olennaisia osa-alueita. Integraatioyksikön sisällä työskentelee myös pienempiä, noin neljän henkilön tiimejä, jotka keskittyvät tarkemmin omiin asiakkuuksiinsa.

Varsinkin näin alkuun, vuorovaikutus koulutuksissa ja tiimien sisällä on keskeinen osa työtehtäviäni, koska se tukee oppimistani ja mahdollistaa tiedon jakamisen muiden asiantuntijoiden kanssa. Alussa saamme hyvin paljon tukea ja asioita ohjataan kädestä pitäen, mutta ajan myötä odotan, että pystyn itse myös jakamaan oppimaani ja osallistumaan aktiivisemmin keskusteluihin ja koulutuksiin. Vuorovaikutustilanteet tiimin sisällä ja muiden integraatioyksikön jäsenten kanssa ovat erittäin tärkeässä roolissa ammatillisen kehittymiseni kannalta.

3 Seurantajakson raportointi viikkoanalyyseineen

Tässä luvussa esitellään opinnäytetyön seurantajakso ja sen aikana toteutettu käytännön työskentely. Luku sisältää viikkoanalyysit, joissa kuvaan konkreettisesti oppimistani, kohtaamiani haasteita ja tekemiäni ratkaisuja viikon kuluessa. Kunkin seurantaviikon osuudessa käyn läpi keskeisimmät tehtävät ja niiden vaikutuksen omaan ammatilliseen kehittymiseen sekä pohdin teoriataustan ja käytännön työn yhdistymistä.

3.1 Seurantaviikko 1

Maanantai 3.3.

Päivä alkoi uusien työntekijöiden työvälineiden jakamisella, minkä jälkeen ohjelmassa oli yrityksen johdon puheenvuoro, jossa käytiin läpi organisaation kehityskaari sen perustamisesta tähän päivään. Samassa yhteydessä esittäytyivät myös projektinhallinnan ja liiketoiminnan tukitoiminnot sekä henkilöstöhallinto. Paikalla olivat näiden tiimien edustajat sekä uusien työntekijöiden esihenkilöt.

Näiden jälkeen lounastimme yhdessä muiden aloittavien sekä omien esimiestemme kanssa, jonka jälkeen oli vuorossa rekrytoijien perehdytys ja laitteiden asennusta. Perehdytyksessä käytiin hyvin ylätasolla yhtiön infrastruktuuria ja asennusten apuna meillä oli IT-tuen henkilöstä mukana. Asennukseen kuului työpuhelinten turvauksen asettaminen vaatimuksia vastaavaksi sekä tarvittavien ohjelmien asentaminen sekä työpuhelimeen, että työkannettavaan.

Kun asennukset alkoivat olemaan hyvällä linjalla suurimmalla osalla, oli vuorossa toimistokierros, jossa kävimme jokaisen yksikön omat työtilat sekä kaikki muut oleelliset alueet talossa. Tämän jälkeen päivän virallinen osuus oli pulkassa ja uusille aloittajille oli järjestetty työpäivän jälkeistä toimintaa yrityksen juhlatilassa, johon osallistui suurin osa uusista aloittajista, HR-tiimin jäseniä, uusien työntekijöiden mentorit sekä esimiehet. Tapahtuma oli hyvin kevyt, joissa söimme illallista yhdessä ja pelasimme ryhmäpelejä sen lisäksi, että vain yleisesti tutustuimme toisiimme. Tällöin tutustuin myös mentoriini, joka on myös valmistunut tradenomiksi Haaga-Heliasta. Hän sai minut tuntemaan hyvin vastaanotetuksi ja antoi ensivaikutelmalta luotettavan tunteen siitä, että voin olla häneen yhteydessä, kun tarvitsen apua.

Tiistai 4.3.

Tiistaina meillä oli heti aamusta HR-järjestelmän esittelyä ja ohjeistusta, jonka jälkeen käytiin läpi yrityksen sisäisen oppimisakatemian käyttöjärjestelmää sekä sisältöä. Järjestelmät olivat hyvin kattavat ja kävimme sen pääpiirteittäin läpi noin tunnissa. Jokaisen omaksi ”kotitehtäväksi” jäi tutustua mitkä ovat akatemiassa pakolliset koulutukset, ja mitä muita suositeltavia sieltä löytyy.

Lounaan jälkeen jokaisen yksikön uudet työntekijät kävivät keskustelua oman esimiehensä kanssa. Meidän osallamme käytiin esimiehen kanssa keskustelua vähän yksikkömme kokoonpanosta, mihin tiimiin meidät sijoitetaan sekä minkälainen meidän seuraavan parin kuukauden koulutusjakso yksikössämme tulee olemaan.

Päivän päätteeksi aloitimme käymään läpi perehdytystä tuntikirjauksista, niiden tärkeydestä ja toiminnasta. Yrityksellämme on kaksi tapaa kahdessa eri järjestelmässä, miten kirjata tunteja ja tässä kävimme läpi ensimmäistä niistä.

Keskiviikko 5.3.

Päivä alkoi samalla aiheella mihin eiliseen jäätiin, eli tuntikirjaukseen. Nyt kävimme toista kahdesta järjestelmästä, joka oli huomattavasti kattavampi ja johon sisältyi paljon muitakin toimintoja.

Kävimme myös läpi yrityksen kokeneiden konsulttien kanssa konsulttitaitoja. Kävimme läpi perusasioita ja ominaisuuksia, joita konsultilla pitäisi olla sekä miten asiakkaiden kanssa pitäisi käydä vuorovaikutusta. Kävimme hyvin paljon erilaisia skenaarioita läpi, hyviä sekä huonoja, ja miten niissä on toimittu sekä miten niissä voisi toimia paremmin.

Päivä päättyi 1:1-tapaamisella esimiehen kanssa, jossa kävimme läpi ensimmäisten päivien tunnelmia sekä esimiehen odotuksia minulle puolen vuoden aika välillä. Mitään tuloksellisia odotuksia esimiehellä ei ollut: Oli puhetta vain siitä, että kehitystä tapahtuu ja oman panostuksen laittamista oppimiseen. Keskustelu ei aiheuttanut yhtään painostavaa oloa ja esimieheni selvästi painotti sitä, että olen alkuun siellä oppimassa enkä tekemässä yritykselle tulosta, joka helpotti suuresti jännitystä ja ajattelua siitä, että pärjäänkö työtehtävässä.

Torstai 6.3.

Päivä oli hyvin kevyt, sillä kävimme aamulla noin tunnin läpi yrityksen sisäistä harrastetoimintaa, jonka jälkeen kävin itse työpaikan kuntosalilla. Myöhemmin päivällä oli yksikkömme bi-weekly training-tilaisuudessa, jossa yksi tiimi esittää valitsemaansa ratkaisua, jonka he ovat tässä lähiaikoina tehneet. Tässä he samalla esittivät, mitä ongelmia on tullut eteen ja miten he lähtivät sitä selvittämään.

Perjantai 7.3.

Tämä oli vielä lyhyempi päivä, sillä kävimme aamulla läpi, kuinka hakea kulukorvauksia yrityksen sisäisestä järjestelmästä ja myöhemmin päivällä oli vain noin puolen tunnin keskustelu ensimmäisen viikon kokemuksista

1. Viikon analyysi: Tutustuminen työyhteisöön

Tällä viikolla ei vielä päästy suoraan syventämään teknistä osaamistani Azure-ympäristössä, sillä työtehtävät keskittyivät enemmän perehdytykseen ja organisaatioon liittyviin asioihin. Koulutuksissa, joissa kävimme läpi yrityksen infrastruktuuria ja järjestelmiä, oli kuitenkin tärkeää ymmärtää, miten nämä järjestelmät tukevat integraatioiden kehittämistä, erityisesti työpaikan sisäisessä oppimisakatemiassa.

Erityisesti tulevaisuudessa odotan, että voin syventyä tarkemmin Azure-pohjaisiin integraatioratkaisuihin, koska työtehtäväni keskittyvät tähän. Olen myös havainnut, että Azure-ympäristön ymmärtäminen on oleellinen osa työtehtäviäni, sillä se vaikuttaa merkittävästi työskentelytapojemme sujuvuuteen. Koulutustilanteet ja työympäristön esittely antavat hyvän perustan, mutta varsinaiset käytännön kokemukset ja oppiminen tulevat vasta myöhemmin

C#-ohjelmointikielen oppimista ei vielä tällä viikolla ole ollut. Perehdytyksen aikana käytiin kylläkin läpi yrityksen käytössä olevia ohjelmointikieliä ja työkaluja, ja latasin näitä jo valmiiksi tietokoneelleni ja katselin hieman erilaisia toimintoja. Koska minulla on rajallinen kokemus C#-kielestä, tämä alkuvaihe on erityisen tärkeä, ja pyrin tukeutumaan työpaikan tarjoamiin resursseihin ja koulutuksiin.

Mentorini kanssa keskustelu on myös ollut hyödyllistä, sillä hän on auttanut minua hahmottamaan, kuinka pääsen alkuun C#-kielen käytössä ja minkälaiset ovat työni alkuaskeleet.

Työpaikan kulttuuri on ollut tosi avoin ja tukevainen. Vaikka olen ollut paastolla, jonka vuoksi en ole osallistunut yhteisiin ruokailuhetkiin, olen silti ollut tiimiläisten kanssa tekemisissä ja saanut paljon tukea. Varsinkin mentorini kanssa käydyt keskustelut ovat olleet hyödyllisiä, koska hän on antanut minulle luottamuksen tunteen ja varmuuden siitä, että voin aina kysyä apua ja odottaa kehittymistä työssäni.

On myös ollut tosi mukavaa osallistua tiimin sisäiseen pelituokioon, joka pidetään usein lounaan jälkeen. Tämä on ollut rento ja hauska tapa tutustua tiimiin ja kokea itseni tervetulleeksi sekä osaksi yhteisöä. Nämä yhteiset aktiviteetit ovat tärkeitä yhteishengen vahvistamisessa ja tiimin yhteistyön kehittämisessä. Yhteisten kokemusten jakaminen lisää luottamusta ja avoimuutta tiimin

jäsenten välillä, mikä tukee sopeutumista ja vahvistaa ryhmädynamiikkaa (Ideagroup.fi 2023). Tämä on ollut selkeästi nähtävissä omassa sopeutumisprosessissani.

Vaikka alku on ollut totta kai jännittävä, on tärkeää muistuttaa itseäni, että olen tässä alussa oppimassa. Esimieheni on painottanut, että ensimmäiset kuukaudet keskittyvät ennen kaikkea oppimiseen, ei suorituksiin. Tämä ajattelutapa on auttanut vähentämään painetta ja antanut minulle tilaa kasvaa roolissanni.

3.2 Seurantaviikko 2

Maanantai 10.3.

Viikko alkoi ohjelmistokehitys-koulutuksella aamulla. jossa käytiin tarkemmin läpi, mitä ohjelmointikieliä ja työkaluja meillä on yleisesti käytössä. Näitä oli mm. C#, SQL ja joissakin yksiköissä myös JavaScript sekä muita työkaluja kuten Bruno ja Visual Studio. Koulutuksessa ei käsitelty vain integraatioyksikön käytössä olevia välineitä, vaan myös muiden yksiköiden työkaluja, kuten O365-työkaluja ja niiden soveltamista. Erityisesti puhuttiin myös eri järjestelmistä kuten Dataverse ja OData, jotka ovat keskeisiä työkaluja yrityksen projektien hallinnassa. Käytiin läpi myös ketterän kehityksen malleja ja prosessimalleja, kuten Scrum, jotka tukevat projektinhallintaa ja kehitystyötä. Turvallisuus oli myös keskeinen osa tätä koulutusta, jossa käsiteltiin sitä, miten yritys varmistaa tietoturvan käytännöissä. Lisäksi kävimme läpi meidän käyttämää GitFlow-mallia, joka ohjaa versionhallintaa ja koodin kehitystä tiimien sisällä.

Iltapäivällä käytiin Azure Basics -koulutusta, jossa kävimme läpi Azure-portaalia ja katsottiin, miltä projektien sisältö näyttää Azure-ympäristössä. Aloitimme koulutuksen käymällä Azure-ympäristön teoriaa ja sen peruskäsitteitä, kuten resurssien hallintaa ja projektien rakennetta. Sen jälkeen pääsimme katsomaan käytännössä, miltä projektit näyttävät Azure-portaalissa ja miten ne on organisoitu ja hallittavissa. Tämä oli tosi hyödyllistä, koska se antoi enemmän käsitystä siitä, miten projekteja käsitellään ja milläkin työkaluilla.

Tiistai 11.3.

Tiistaina oli Azure Integration Components -koulutusta, joka oli käytännössä samanlainen teemoilta kuin maanantainen ohjelmistokehitys-koulutus, mutta tällä kertaa keskityimme vain integraatiotiimin käytäntöihin, työkaluihin ja prosesseihin. Tässä käytiin läpi, miten Azure-ympäristössä toimitaan integraatioidemme kehittämisessä ja mitkä työkalut ovat keskeisiä tiimimme työssä. Esimerkiksi integraatiotiimi hyödyntää erityisesti Azure-infrastruktuuria ja -työkaluja, kuten Azure Logic Appsia ja Azure Functionsia (Azure-funktioita), joiden avulla luodaan ja hallitaan integraatioratkaisuja eri järjestelmien välillä.

Iltapäivällä oli Git- ja Bitbucket-koulutus, jossa taas kävimme läpi tarkemmin GitFlow-mallia, joka on olennainen osa tiimin versionhallintaprosessia. Kävimme läpi GitFlow-mallia, joka jakautuu Master-, Hotfix-, Release-, Develop- ja Feature-kehityshaaroihin (branches), ja sen eri vaiheita, kuten näiden branchien luomista, pull requestien tekemistä ja koodin yhdistämistä Master-haaraan. Käytiin läpi myös kuinka tärkeä GitFlow on ja mitä hyötyjä siitä on tiimin työskentelyn ja versionhallinnan kannalta, erityisesti isoimmissa projekteissa. Demosimme käytännössä, miten malli toimii vaiheittain ja miten GitFlow auttaa ylläpitämään järjestystä tiimissä, jossa on useita eri työstäjiä, jotta projekti pysyy hallittuna. Demoaminen oli todella kivaa ja hyödyllistä ja koin sen auttavan hahmottamaan teoriaosuutta todella paljon, sillä se oli alkuun vähän sekavaa.

Keskiviikko 12.3.

Kävimme koko päivän Azure Logic Apps -koulutusta. Koulutuksen aluksi käytiin läpi yleistä agendaa, jossa käytiin perusteellisesti läpi Logic Appsin ymmärtämistä ja sen käyttömahdollisuuksia Azure-ympäristöissä. Koulutuksessa perehdyimme siihen, miten Logic Apps voi yhdistää eri järjestelmiä ja automatisoida prosesseja, kuten tietojen siirtoa ja käsittelyä eri palveluiden välillä.

Teimme itse pari simppeliä Logic Apps -sovellusta, joissa haimme tietoja tietokannasta ja käsittelimme niitä. Kävimme läpi Logic Appsin peruskomponentteja, kuten actions (toiminnot) ja triggers (laukaisijat), joiden avulla määritetään, mitä tapahtuu, kun tietyt olosuhteet täyttyvät. Koulutuksessa opimme, kuinka määritämme erilaisia toimintoja, kuten tietojen hakemista, lisäämistä tai päivittämistä, ja kuinka voimme liittää ne toisiinsa luodaksemme monivaiheisia prosesseja.

Lopussa teimme itsenäisesti testejä, joissa haimme aktiivisia käyttäjiä, tarkastelimme käyttäjien tietoja, lisäsimme uusia käyttäjiä ja jopa importoimme käyttäjät Flow-toiminnon kautta. Tämä käytännön harjoittelu auttoi minua ymmärtämään Logic Appsin toiminnan konkreettisesti ja sain hyvän alkukäsityksen siitä, miten näitä voisi käyttää tulevissa projekteissa.

Torstai 13.3.

Tänään kävimme Dataverse integraatiot - ja CRM integraatiot -koulutusta, jossa käsiteltiin useita keskeisiä teemoja, jotka liittyivät pääsääntöisesti tietojen hallintaan ja integrointiin CRM-järjestelmien kanssa. Aloitettiin käymällä Dataverse-järjestelmän perusperiaatteita, kuten sen kerrokset: tietoturva, logiikka, data ja tallennus sekä integraatio. Kävimme myös läpi Web API- ja CRUD-operaatioita, joissa käsiteltiin, kuinka Web API:t mahdollistavat tietojen käsittelyn Dataversessa. Esimerkiksi Web API:n avulla voidaan tehdä CRUD-operaatioita. Ja tämä tukee JSON-muotoa, jota yleisesti käytetään tietojen siirtämiseen ja vastaanottamiseen. Esimerkkinä käytiin läpi kontaktien hakeminen Web API:n kautta, mikä antoi käytännön esimerkin siitä, miten Dataverse ja Web API voivat toimia yhdessä.

Myös autentikointi oli keskeinen osa koulutusta. Kävimme läpi OAuth-protokollan ja sen roolin pääsyoikeuksien välittämisessä palvelusta toiseen. Lisäksi käsiteltiin, miten JWT-tokenia käytetään tunnistautumiseen ja valtuutukseen, mikä on keskeinen osa nykyaikaisia API-pohjaisia järjestelmiä.

Koulutuksessa käytiin myös läpi event- ja service bus -elementtejä, jotka mahdollistavat viestien ja tapahtumien käsittelyn ja välittämisen järjestelmässä. Nämä ovat tärkeitä elementtejä erityisesti suurissa ja monimutkaisissa järjestelmissä, joissa tarvitaan tehokasta ja luotettavaa viestinvälitystä.

Lopetimme päivän tekemällä harjoituksia, joissa käytimme Logic Appseja integraatioiden luomiseen ja sen jälkeen testasimme Brunolla integraatiotoimintoja.

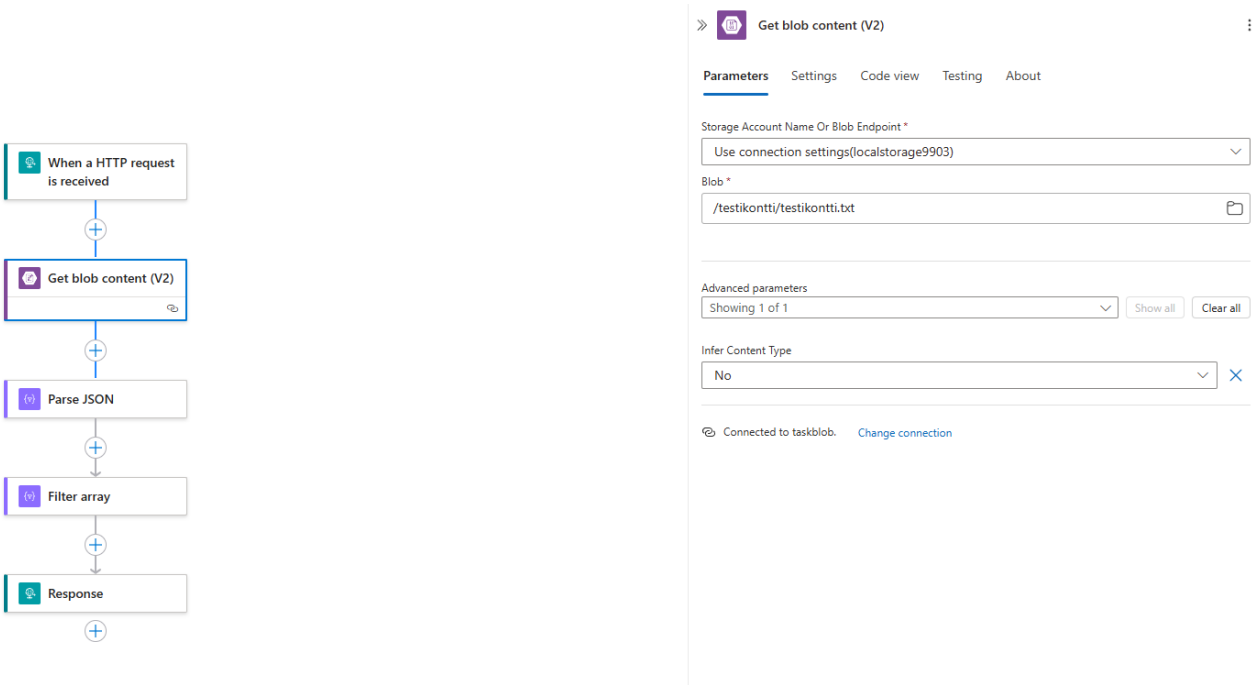
Perjantai 14.3.

Meillä piti olla perjantaina samaa koulutusta kuin torstaina, mutta etenimme niin nopeasti, että saimme kaikki tarvittavat käytyä läpi, joten perjantai oli vapaa.

2. Viikon analyysi: Integraatioharjoituksia Azure-ympäristössä

Tällä viikolla syvensimme osaamista Azure-ympäristön käytössä erityisesti integraatiokehityksessä. Azure Logic Apps ja Azure-funktiot olivat koulutusten keskiössä, ja ne ovat keskeisiä työkaluja prosessien automatisoinnissa ja järjestelmien välisessä tiedonkäsittelyssä (Microsoft, 2025b; 2025a). Logic Apps mahdollistaa no-code/low-code-ratkaisujen rakentamisen visuaalisesti, kun taas funktiot tarjoaa ohjelmallisen hallinnan ja laajennettavuuden tarkemmin säädeltäviin toimenpiteisiin.

Erityisesti keskiviikkona tekemämme käytännön harjoitukset syvensivät ymmärrystäni Logic Appsin toiminnasta. Rakensimme työnkulun, jossa HTTP-pyynnön vastaanoton jälkeen luettiin tiedosto Blob Storage -kontista, jäsennettiin sen sisältö JSON-muotoon, suodatettiin tietoa Filter array -toiminnolla ja palautettiin suodatettu tulos HTTP-vastauksena (kuva 2). Tämä harjoitus konkretisoi, miten Logic Appsilla voidaan automatisoida tiedonkäsittelyä eri palveluiden välillä ja yhdistää useita vaiheita visuaalisessa työnkulussa.



Kuva 2. Logic Apps rakenne

Tällä viikolla olin etänä, sillä olin flunssassa koko viikon. Tämän vuoksi en ollut mukana tiimin sisäisessä toiminnassa mukana, mutta oli kivaa, että jokaisessa koulutuksessa minulta kysyttiin kuinka voin ja toivotettiin pikaisia paranemisia. Erityisesti maanantain ja tiistain koulutukset, kuten Git-Flown-käyttö ja versionhallinta, auttoivat ymmärtämään tiimityöskentelyn merkityksen ja sen, miten tiimissä toimitaan tehokkaasti yhdessä. GitFlow on tärkeä työkalu, joka varmistaa, että tiimi voi hallita versionhallinnan ja jakaa työt sujuvasti eri työvaiheisiin (GitHub Docs, 2024).

Etänä työskentely oli hieman haasteellinen, sillä kotona on huomattavasti vaikeampaa keskittyä kuin paikan päällä töissä, mutta selviydyin viikosta hyvin.

Viikon aikana saadut käytännön kokemukset tukivat Microsoftin suosittamaa pilviarkkitehtuuria, jossa Logic Apps toimii integraatioiden ytimenä ja Azure-funktio tukee tarvittaessa kooditasolla (Microsoft, 2025a).

3.3 Seurantaviikko 3

Maanantai 17.3.

Viikko alkoi Azure-Funktiot-koulutuksella, jossa kävimme ensiksi läpi koulutussuunnitelmaa tämän aiheen suhteen. Sen jälkeen lähdimme alustamaan projektia, jossa olisi tarkoitus lähteä testaamaan erilaisia toimintoja. Loimme alkuun tietokantaan yhteyden ja malleja (models), jonka jälkeen

teimme erilaisia toimintoja, joilla haettiin ja luotiin tietokantaan tietoja. Tässä oli uusia asioita, mutta myös paljon tuttua, sillä opinnoissa on tullut käytettyä Javaa alkuvuosina jonkun verran ja vaikka en siitä ihan hirveästi muistanut, niin sitä alkoi tehtäessä muistamaan aika paljon sillä C# muistutti tältä osalta paljon Javaa.

Tiistai 18.3.

Koulutukset jatkuvat ja tänään oli vuorossa asiaa integraatioyksikön toimintatavoista. Käytiin aika perustavasti läpi yrityksen toimintamalleja myyntivaiheesta tukivaiheeseen ja missä vaiheessa projektia mikäkin yksikkö tekee ja mitäkin hommia. Käytiin siis läpi eri prosesseja, mitä proseduureja noudatetaan kussakin prosessissa, mitä työkaluja ja mitä järjestelmiä näissä kussakin käytetään. Tässä tuli aika paljon asiaa muistettavaksi, mutta onneksi esitetyt kaaviot ovat yrityksen intranetissä saatavilla. Koitin keskittyä muistamaan erityisesti omaan yksikköni koskevat asiat.

Keskiviikko 19.3.

Keskiviikkona oli tarkoitus käydä Azure API Management -koulutusta, mutta koulutuksen vetäjä oli sairaana, joten se peruuntui. Esimieheni antoi minulle ja toiselle juuri aloittaneelle kollegalleni koulutuksen ulkopuolisen projektin ja työstin sitä. Tarkoitus oli luoda ohjelma, joka käsittelee sille syötettyä JSON-tiedostoa, muuntaa sen XML:ksi ja validoi sen. Tutkin lähtökohtaisesti tähän alkuun netistä materiaalia tähän.

Torstai 20.3.

Tänään piti olla integraatiotiimin yksikön sisäinen koulutus, mutta se peruuntui ja siirtyi seuraavalle viikolle, joten jatkoin eilistä projektia ja sain sen toimimaan, vaikka en ehtinyt vielä näyttää sitä kenellekään osaavammalle kollegalle, jotta saisin varmistuksen siitä, että se toimii kuten pitäisi. Lähdin suorittamaan ratkaisua niin, että ensin luin JSON:n tiedoston objektiin, jonka jälkeen tein XML:lle oman objektin. Sitten yhdistin JSON:n XML-objektiin, jonka jälkeen ohjelma luo XML-tiedoston ja validoi sen XSD-tiedostolla.

Perjantai 21.3.

Viikko päättyi koulutukseen, jossa perehdyttiin Entra ID:n (entinen Azure AD) toimintoihin ja tietoturvaominaisuuksiin. Kävimme läpi DNS-nimipalvelujärjestelmän merkityksen ja sen roolia brändätyissä verkkotunnuksissa. Lisäksi tutustuimme Entra ID:n keskeisiin komponentteihin, kuten identiteetteihin, ryhmiin ja rooleihin, sekä tietoturvaominaisuuksiin, joista monivaiheinen tunnistautuminen (MFA) jäi erityisesti mieleeni.

MFA perustuu siihen, että käyttäjä vahvistaa henkilöllisyytensä yhdistämällä vähintään kaksi erillistä vahvistusmenetelmää. Nämä voivat olla jotakin, mitä käyttäjä tietää, kuten salasana; jotakin, mitä käyttäjällä on, kuten luotettu laite (esimerkiksi puhelin), jota on vaikea kopioida; tai jotakin, mitä käyttäjä on, kuten biometrinen tunniste – esimerkiksi sormenjälki tai kasvojen tunnistus.

Lisäksi käsiteltiin Entra ID Connectin toimintaa hybridiympäristössä sekä Privileged Identity Management (PIM) -ratkaisua roolien hallintaan. Lopuksi tarkasteltiin käyttäjien ja sovellusten hallintaa sekä Zero Trust -periaatteen soveltamista. Koulutuksen aikana kävi myös ilmi, että minulta puuttui LastPass-tunnusten aktivointi, ja sain tehtäväksi hoitaa asian ottamalla yhteyttä Service Deskiin.

3. Viikon analyysi: Itsenäistä ongelmaratkaisua

Vaikka osa koulutuksen aiheista jatkoi samoja teemoja kuin aiemmilla viikoilla, kuten Azure-funktiot ja Logic Apps, pääsin nyt syventämään ymmärrystäni omaehtoisten tehtävien kautta. Viikko oli rakennettu sekä ohjatusta koulutuksesta että itsenäisestä työskentelystä, mikä auttoi siirtämään teoriaa konkreettisiksi taidoiksi. Tällä viikolla tuntuu, että otin ensimmäiset kunnon askeleeni ammatillisessa kehityksessä, erityisesti itsenäisen ongelmanratkaisun ja epävarmuuden kohtaamisen osa-alueilla.

Viikko alkoi maanantaina Azure-funktio-projektin alustuksella. Rakensimme C#-ohjelmointikielellä ohjelman, joka muodostaa yhteyden tietokantaan ja käsittelee tietoja erilaisten funktioiden avulla. Vaikka tekniikat, kuten HttpTrigger, DTO:t ja tietokantakyselyt, olivat jo teoriasta tuttuja, huomasin oppivani uutta siitä, miten järjestelmän osat pelaavat yhteen. Samalla aloin hahmottaa omia vahvuuksiani ja heikkouksiani esimerkiksi virheenkäsittelyn toteutuksessa.

Keskiviikon koulutuksen peruunnuttua sain vastuulleni ulkopuolisen projektin, jossa tuli toteuttaa tiedostopohjainen integraatioratkaisu: JSON → XML sekä validointi XSD:llä. Tämä oli ensimmäinen isompi itsenäinen tehtäväni, ja vaikka alkuun epäröin, pääsin nopeasti vauhtiin, kun rakensin ohjelman vaihe vaiheelta pieniksi tehtäviksi. Käytin lähteinäni mm. W3C:n dokumentaatiota (W3C, 2024) ja aiempia muistiinpanojani C#-kurssilta. Huomasin tässä, kuinka oppiminen oli vähän erilaista kuin luentopohjainen taikka ohjeistettu oppiminen. Tämä oli enemmän ongelmalähtöistä oppimista (problem based learning) ja siinä ongelman ratkominen johtaa syvempään ymmärrykseen kuin pelkkä malliratkaisun seuraaminen.

Fowler (2002, luku 15) kuvaa DTO-kaavassaan, että tiedon esitysmuodot kuten XML ja JSON ovat keskeisiä järjestelmien välisessä tiedonsiirrossa, niiden avulla data voidaan välittää selkeästi ja yhteisesti ymmärretyssä muodossa eri järjestelmien välillä. Projektissani konkretisoitui Fowlerin esiin nostama ajatus siitä, että serialisointi ja validointi ovat oleellinen osa integraatioratkaisujen luotettavuutta.

Perjantain koulutus peruuntui, mutta sen sijaan sain mahdollisuuden syventyä itsenäisesti AZ-204-materiaaliin. Vaikka koulutus kiinnosti, niin koin tämän myös positiivisena asiana, sillä IT-alalla kehittyminen on myös isolta osalta itsenäistä oppimista, joten mitä aikaisemmin siihen totun niin sitä parempaa oppimispohjaa se lue minulle tulevaisuuden kannalta. Lemmetty ja Collin (2020, 48) korostavatkin, että nykyisessä ICT-työssä vastuu oppimisesta siirtyy yhä enemmän työntekijöille itselleen, ja suurin osa oppimisesta tapahtuu epämuodollisesti ja itseohjautuvasti työtehtävien yhteydessä.

Lisäksi tiistain koulutuksessa käytiin perusteellisesti läpi integraatioyksikön prosesseja ja vastuujakoa. Opin, missä vaiheessa integraatiotiimi tulee mukaan projektin elinkaareen, miten tukivaihe ja ylläpito järjestetään, ja millaisia työkaluja ja järjestelmiä missäkin vaiheessa käytetään. Tämä auttoi minua hahmottamaan oman paikkani laajemmassa kokonaisuudessa, mikä on tärkeä osa sopeutumistani uutena työntekijänä uuteen yritykseen ja sen toimintamalleihin.

Kolmas viikko erottui aiemmista siinä, että sain ensimmäistä kertaa itse rakentaa kokonaisen ratkaisun ilman ohjausta. Tämä toi vähän lisävarmuutta tekniseen tekemiseen ja loi perustan ymmärrykselle siitä, mitä ohjelmointi ammattimaisessa ympäristössä oikeasti on.

Sopeutuminen uuteen rooliin on selvästi jatkuva prosessi. Se vaatii paitsi teknisiä taitoja, myös taitoa pyytää apua, ymmärtää projektien sykliä ja uskaltaa kokeilla, vaikka ei ole täysin varma. Tämä ammatillinen kasvu, vaikkakin totta kai vielä pienin askelin eteneekin, on juuri sitä, mitä Dreyfusin kompetenssimalli (Dreyfus & Dreyfus 1980, 5) kuvaa matkalla noviisista kohti asiantuntijaa. Mallin mukaan kehittyminen tapahtuu vaiheittain. Aluksi työssä nojaudutaan sääntöihin ja ohjeisiin, mutta kokemuksen karttuessa opitaan tunnistamaan tilanteita, tekemään itsenäisiä päätöksiä ja toimimaan yhä sujuvammin sekä intuitiivisemmin. Jokainen askel, myös epävarmuuden ja kokeilun kautta, vie lähemmäs asiantuntijuutta. Tämä viikon ensimmäinen itsenäinen tehtävä korostaa omaa henkilökohtaista kompetenssimallin mukaista ensimmäistä askelta noviisista kohti asiantuntijaa, sillä tässä on alkukantaista siirtymistä ohjatusta koulutuksesta, itsenäisesti ratkaisujen tuottamiseen ja uuden opettelemiseen.

3.4 Seurantaviikko 4

Maanantai 24.3.

Jatkoimme edellisen viikon maanantain Funktiot-aiheella ja tänään vuorossa oli ODataan perehtyminen. Ennen lounastaukoa alustettiin projektia OData-toimintoja varten. Tässä oli kyllä paljon uutta, sillä OData on itselleni uusi aihealue. Tuntui, että nyt projektin eri osat alkoivat liittyä toisiinsa

yhä enemmän, esimerkiksi tietokanta, DTO-oliot, funktiot ja sovelluksen reititys (endpoints), mikä loi kokonaisuuteen moniulotteisuutta. Toisaalta tämä teki asioista myös vähän sekavia, koska välillä oli vaikea hahmottaa, missä käytetään mitäkin objekteja ja mistä tiedostoista ne tulevat.

Iltapäivällä rakensimme kolme Azure-funktio-pohjaista rajapintaa: GetPlaylists, GetTracks ja CreatePlaylist. Harjoituksessa haettiin tietoa Chinook-tietokannasta käyttäen Entity Frameworkiä ja muotoiltiin tulokset DTO-oliorakenteisiin. GetPlaylists haki soittolistoja ja niihin liittyviä kappaleita, kun taas GetTracks haki kappaleita ja sisälsi tiedot albumista, artistista ja genrestä. CreatePlaylist tarkisti, löytyykö saman nimistä soittolistaa, ja jos ei, se loi uuden.

Tämä auttoi hyvin hahmottamaan serialisointia, virheenkäsittelyä ja sitä, miten integraatioratkaisu rakentuu vaihe vaiheelta.

Tiistai 25.3.

Tänään oli itsenäistä oppimista ja kertasin itsekseni edellisenä päivänä tehtyjä ratkaisuja ja opiskelin hieman AZ-205 sertifikaattia varten.

Keskiviikko 26.3.

Aamupäivällä keskityttiin Finance & Operations (F&O) -integraatioihin. Kävimme läpi, miten F&O-järjestelmä integroidaan muihin ratkaisuihin, erityisesti Dataversen ja muiden Microsoftin pilvipalvelujen kanssa. Esillä oli esimerkiksi se, miten tietoa voidaan hakea OData-rajapinnan kautta ja miten F&O:n tietorakenne ohjaa sitä, miten integraatiot suunnitellaan. Tähän tarvitaan monesti useiden taulujen yhdistämistä, ja niiden suhteet vaikuttavat siihen, miten tietoa käsitellään.

Iltapäivällä siirryttiin Azure Service Bus -aiheeseen. Rakensimme tapahtumapohjaisen Azure-funktion, joka kuuntelee viestejä ja lähettää ne eteenpäin Service Busiin. Käytännön esimerkkinä toteutettiin OrderPlaylist-funktio, jossa asiakkaan valitsema soittolista "tilataan" ja sen sisältö paketoidaan viestiksi. Viesti lähetettiin eteenpäin Service Busin kautta myöhempää käsittelyä varten. Funktio käytti HTTP-triggeriä ja luki pyynnön sisällön OrderPlaylistDto-objektiin, haki soittolistan ja asiakkaan tietokannasta, rakensi niistä OrderTracksMessage-viestin ja palautti sen OrderTracksServiceBusOutput-rakenteessa.

Torstai 27.3.

Oli taas itsenäistä oppimista ja opiskelin AZ-205 sertifikaattia varten.

Perjantai 28.3.

Päivän aiheena oli Infrastructure as Code (IaC), ja keskityimme erityisesti Bicep-kieleen, jonka avulla voidaan määritellä ja ottaa käyttöön Azure-resursseja suoraan koodista. Koulutuksessa käytiin läpi, miten esimerkiksi Logic Apps -työnkulku voidaan luoda ja julkaista Bicepin avulla hallitusti ja toistettavasti.

Harjoituksessa laadimme main.bicep-nimisen mallin, jossa määriteltiin muun muassa työnkulku, resurssiryhmä, yhteydet ja käyttöoikeudet. Lisäksi käytettiin JSON-muotoista parametritiedostoa, jossa määriteltiin ympäristökohtainen arvo, esimerkiksi:

```
"parameters": {
  "environmentCode": {
    "value": "test"
  }
}
```

Käyttöönotto tehtiin komentoriviltä seuraavasti:

```
az deployment group create --resource-group example-rg --template-file "main.bicep" --parameters "parameters.test.json" --mode Incremental
```

Lisäksi käsiteltiin autentikointia ja identiteettien määrittelyä. Käytimme esimerkkinä mallia, jossa määritellään sekä järjestelmän hallitsema että käyttäjän määrittämä identiteetti:

```
identity: {

  type: 'SystemAssigned, UserAssigned'

  userAssignedIdentities: {

    '${ExampleManagedIdentity.id}': {}

  }

}
```

Tämä koulutus osoitti, miten Logic Apps -pohjaiset integraatiot voidaan rakentaa koodina ja julkaista osana automatisoitua käyttöönottoa. Oli hyödyllistä nähdä, miten tämä liittyy kokonaisarkkitehtuuriin, jossa Logic Appsiin yhdistyy erilaisia resursseja, kuten viestinvälityspalveluja ja identiteetinhallintaa.

4. Viikon analyysi: Integraatioratkaisun kokonaisarkkitehtuurin hahmottaminen

Tällä viikolla ammatillinen kehittymiseni jatkui Azure-pohjaisessa integraatiokehityksessä. Harjoittelimme OData-rajapintoja, Azure-funktio-pohjaista tietokantakäsittelyä ja lopuksi tutustuimme Bicep-kieleen, joka mahdollistaa infrastruktuurin hallinnan koodina. Yksi tärkeimmistä oivalluksista tällä viikolla oli se, kuinka eri teknologiat, kuten DTO-rakenteet, Entity Framework, Azure-funktiot ja HTTP-endpointit, alkavat muodostaa selkeämpää kokonaiskuvaa integraatioratkaisun arkkitehtuurista. Tämä syvensi ymmärrystäni siitä, mitä “integraatio” todella tarkoittaa arjessa, nimittäin kyse ei ole vain tiedon siirrosta, miten minä sen isoksi osaksi kuvittelin päässäni, vaan kokonaisvaltaisesta suunnittelusta, hallinnasta ja ylläpidosta.

Kehityin teknisesti etenkin serialisointien ja rajapintojen rakentamisessa. Esimerkiksi maanantain CreatePlaylist-funktio käytti HttpTrigger-pohjaista rajapintaa vastaanottaakseen JSON-objektin, josta muodostettiin soittolista tietokantaan. Tämä erottelun tärkeys tuli viikon aikana esiin erityisesti silloin, kun hahmotin eri komponenttien (esim. DTO:t vs. liiketoimintalogiikka) roolit konkreettisissa funktioissa ja rajapinnoissa. Huomasin, että selkeä raja niiden välillä teki ratkaisuista sekä selkeämpiä että helpommin ylläpidettäviä, juuri kuten Fowler (2002, luku 15) mallissaan painottaa.

Viikko päättyi Bicep-koulutukseen, jossa infrastruktuuri määriteltiin koodina. Tässä päästiin lähemmäs DevOps-aihetta, josta minulla ei aiemmin ollut kokemusta. Vaikka en ollut aiemmin käyttänyt Bicepiä, huomasin nopeasti, miten se liittää yhteen oppimaani Logic Appsien ja funktioiden konfiguroinnin kanssa. Microsoftin dokumentaation mukaan Infrastructure as Code (IaC) tukee automaatiota ja ylläpidettävyyttä pilviympäristössä, mikä on olennainen osa modernia DevOpsia (Microsoft, 2024).

Reflektoidessani viikkoa huomasin, vaikka olenkin enemmän pihalla kuin perillä asioista, että itseluottamukseni kasvoi hieman. Monimutkaisempien kokonaisuuksien ymmärtäminen, esimerkiksi miten webhook vastaanottaa datan BC:stä ja funktio vie sen eteenpäin, alkoi pikkuhiljaa hahmottumaan päässä. Knuth (1997, Esipuhe) kuvaa tätä prosessia ohjelmoinnin "taiteena". Syvempi ymmärrys ei synny hetkessä, vaan rakentuu vähitellen harjoittelun, kokemuksen ja kerrosten kautta. Kuten Knuth esipuheessa toteaa, ohjelmoinnin omaksuminen vaatii aikaa ja käytännön kokeiluja, eikä kaikkea voi sisäistää yhdellä kertaa.

Lopuksi on tärkeää mainita myös sopeutuminen uuteen rooliin. Vaikka osa koulutuksista oli toistoa aiemmista viikoista, huomasin käyttäväni niitä tukemaan aiempien oppien yhdistämistä. Tämä on hyvin tyypillistä yleisesti urien alkuvaiheessa: asiat alkavat loksahdella paikalleen vähitellen, kuten (Dreyfus & Dreyfus 1980, 10, 13–14) kompetenssimallissa kuvataan, matka noviisista asiantuntijaksi kulkee jatkuvan reflektion ja käytännön kautta.

3.5 Seurantaviikko 5

Maanantai 31.3.

Käytettiin koko päivä perehtymään enemmän Logic Appseihin, ja syvennyttiin tänään erityisesti virheenkäsittelyyn, muuttujien hallintaan ja ehtolauseisiin Logic Apps -työnkulussa. Rakensimme työnkulun, joka vastaanotti HTTP-pyynnön, käsitteli JSON-taulukon For each -silmukassa ja laski summan. Epäonnistuneet tapaukset kerättiin Errors-muuttujaan, ja työnkulku palautti joko onnistumis- tai virheilmoituksen tilanteen mukaan.

Lisäksi käytettiin Scope-lohkoja ryhmittelemään toimintoja ja hallitsemaan virheitä keskitetysti. Jos jokin vaihe epäonnistui, virheviesti muotoiltiin ja palautettiin Compose- ja Terminate-toiminnoilla. Tutustuimme myös Kusto Query Languageen (KQL), jota käytetään erityisesti Azure Monitorin ja Microsoft Fabricin yhteydessä lokien ja analytiikan kyselyyn. Harjoittelimme yksinkertaisia kyselyitä, kuten taulukon suodatuksia ja ryhmittelyjä, ja keskustelimme siitä, miten KQL:ää voidaan hyödyntää valvonnassa ja diagnostiikassa osana integraatioratkaisuja.

Tiistai 1.4.

Päivän aiheena oli Business Central (BC) -integraatiot. Koulutuksessa käytiin läpi erilaisia integraatiotapoja, kuten valmiit Microsoftin palveluihin perustuvat ratkaisut, OData-rajapinnat ja tiedostopohjaiset integraatiot. Erityisesti käsiteltiin BC:n sisäänrakennettuja dataversiointegraatioita ja valmiita laajennuksia, joiden avulla integraatiot voidaan toteuttaa nopeasti ilman merkittävää räätälöintiä.

Aamupäivän aikana tutustuimme BC:n käyttöliittymään ja siihen, miten eri ympäristöissä ja yrityksissä navigointi tapahtuu. Kävimme myös läpi hakutoimintojen ja näkymien merkitystä tehokkaassa käytössä.

Iltapäivällä toteutimme käytännön harjoituksen, jossa integroitiin Business Central Azureen webhookien avulla. Rakensimme funktion, joka vastaanottaa BC:stä tulevia ilmoituksia ja käsittelee ne eteenpäin.

Keskiviikko 2.4.

Tänään keskityttiin lisään integraatioiden monitorointiin ja loggaukseen Azure-ympäristössä. Koulutuksessa käytiin läpi, miten Azure Log Analytics -palvelua käytetään tehokkaasti seuraamaan ja analysoimaan integraatiossa tapahtuvia prosesseja. Opettelimme Kusto Query Languagea (KQL),

joka mahdollistaa joustavan ja tehokkaan kyselyiden kirjoittamisen Azure Monitorin keräämistä lokeista. KQL:n avulla haimme esimerkiksi epäonnistuneita suorituksia, mittasimme vasteaikoja ja tarkastelimme tiettyjen vaiheiden suoritushistoriaa.

Käytiin myös läpi käytännön esimerkkejä siitä, miten Logic Apps -työnkulkujen logit voidaan siirtää Log Analyticsiin ja miten näitä lokeja voi hyödyntää vikatilanteiden jäljittämisessä. Rakensimme myös virheenkäsittelylogiikkaa Logic Apps -ratkaisuihin siten, että virheet kerätään taulukkomuuttujaan ja palautetaan lopuksi KQL-analyysia varten. Näimme käytännössä, miten virheilmoitukset voidaan kerätä ja käsitellä rakenteellisesti niin, että niistä muodostuu selkeä kokonaiskuva raportoinnin ja seurannan tarpeisiin.

Torstai 3.4.

Aamupäivällä jatkoimme edellisen päivän integraatioiden monitorointi- ja lokitusaiheiden parissa. Kävimme vielä läpi käytännön esimerkkejä KQL-kyselyistä ja pohdimme, miten lokitietoja voidaan hyödyntää tuotantoympäristön valvonnassa.

Iltapäivällä osallistuin yksikkömme kuukausittaiseen tapaamiseen, jossa käsiteltiin ajankohtaisia asioita ja katsottiin hieman tulevaan. En kuitenkaan kirjaa yksityiskohtia tähän. Tämän jälkeen meillä oli yksikköpäivä, jonka ohjelmaan kuului ohjattu fatbike-retki Kallahdessa koko tiimin kanssa. Retken jälkeen menimme vielä yhdessä illalliselle. Oli todella mukavaa päästä viettämään vapaa-aikaa työkavereiden kanssa ja tutustumaan heihin paremmin työn ulkopuolella. Tämä auttoi huomattavasti minua tuntemaan oloni osaksi tiimiä sekä olemaan rennommin työkavereitten kanssa.

Perjantai 5.4.

Koulutus oli perjantailta peruttu, joten opiskelin taas itsekseni AZ-204-sertifikaattia varten.

5. Viikon analyysi: Integraation virhekäsittelyyn perehtyminen ja ajatuksia tiimiytymisen tärkeydestä

Viides viikko toi mukanaan syvennettyä teknistä osaamist a tuttujen työkalujen, kuten Logic Appsien, Azure-funktioiden ja Business Centralin, parissa. Vaikka monet asiat olivat jatkoa aiemmista koulutuksista, sain nyt uusia mahdollisuuksia soveltaa aiempaa tietoa laajemmassa kokonaisuudessa ja hahmottaa, miten yksittäiset palaset kytkeytyvät isompaan arkkitehtuuriin.

Maanantain ja keskiviikon koulutuksissa keskityttiin virheenkäsittelyyn, monitorointiin ja lokien analysointiin Azure-ympäristössä. Rakensimme Logic Apps -pohjaisen työnkulun, jossa otettiin käyttöön array-muuttujia virheiden keräämiseen ja Scope-lohkoja virheiden hallintaan, tämä malli auttaa rakentamaan entistä vakaampia ja ylläpidettäviä integraatioratkaisuja. Samalla pääsin käyttämään Kusto Query Languagea (KQL), joka toi uuden tason työnkulkujen läpinäkyvyyteen. Se auttoi

hahmottamaan, mitä Azure Monitorin ja Log Analyticsin keräämällä datalla voidaan oikeasti tehdä, ei vain valvoa, vaan myös ennakoida ongelmia (Microsoft, 2025c).

Tämä liittyy myös laajempaan teemaan havainnoitavuuden periaatteista, joita käsitellään Microsoftin arkkitehtuurimalleissa: havainnoitavuus tarkoittaa kykyä seurata ja ymmärtää järjestelmän tilaa telemetrian, lokien ja metrikoiden avulla. Näiden tietojen avulla voidaan paitsi valvoa ratkaisua, myös ennakoida ongelmia ja optimoida toimintaa automaattisten palautesilmukoiden avulla (Microsoft, 2024).

Maanantain funktio-rajapintojen (GetPlaylists, GetTracks, CreatePlaylist) rakentaminen auttoi vahvistamaan C#-osaamista ja kieli alkaa pikkuhiljaa olemaan tutumpi. Projektissa harjoiteltiin DTO-rakenteiden sarjallistamista, tietojen hakemista Entity Frameworkillä sekä virheiden hallintaa. Saman päälle rakentaminen on selvästi tuonut ohjelmointikielen käyttämiseen rutiinia ja itseluottamusta.

Samalla hahmotin paremmin, miten kerrosarkkitehtuuri toimii käytännössä: datakerros, logiikkakerros ja rajapinta limittyvät. Tällaiset harjoitukset yhdistivät aiempaa teoriaa, kuten Martin (2008, luku 6) korostaa, on tärkeää erottaa DTO-rakenteet ja liiketoimintalogiikka omiin kerroksiinsa, jotta ratkaisut pysyvät ylläpidettävinä ja ohjelmiston arkkitehtuuri säilyy selkeänä.

Yksinkertaistettu kerrosarkkitehtuuri (mukaillen Martin, 2008, luku 6 ja 11):

Rajapinta	(Rajapintakerros / API-kerros)
Sovelluslogiikka	(Sovelluslogiikkakerros)
Datakerros	(Datakerros, DTO/repository)

Tynjälä, Välimaa ja Sarja (2003, 153–154) korostavat, että asiantuntijuuden kehittyminen tapahtuu vuorovaikutuksessa yhteisön kanssa, eli osallistumalla aitoihin käytäntöihin ja sosiaaliseen vuorovaikutukseen. Juuri tällaiset yhteiset tekemiset ja epämuodolliset kohtaamiset mahdollistavat sen, että uusilla työntekijöillä on tilaisuus oppia kokeneemmilta kollegoilta, rakentaa keskinäistä luottamusta ja muodostaa suhteita, jotka tukevat osaamisen kehittymistä.

Kokeneemmilta oppiminen ja tiimityö nähdään olennaisina osina sekä asiantuntijuuden että työyhteisöön sopeutumisen kannalta (Tynjälä ym., 2003, s. 154–155). Artikkelissa todetaan, että asiantuntijuuden kehittyminen ei ole pelkkää tiedon omaksumista, vaan siihen liittyy yhteisössä tapahtuvaa dialogia, tiedon jakamista ja yhdessä oppimista. Tällainen yhteisöllinen ilmapiiri rohkaisee

myös epäonnistumisista puhumiseen ja niiden käsittelyyn, mikä edistää avointa ja oppimista tukevaa kulttuuria.

Torstain ohjattu fatbike-retki Kallahdessa ja yhteinen illallinen kollegoiden kanssa loivat varmuutta ja vahvistivat tunnetta siitä, että kuulun joukkoon. Tämänkaltaiset tapahtumat ja yhteiset hetket kollegoiden kanssa madaltavat kynnystä lähestyä muita, kysyä neuvoa ja oppia kokeneemmilta. Koen, että juuri tällainen yhteisöllisyys on edellytys sille, että pääsen osalliseksi edellä kuvattua aitoa vuorovaikutusta ja oppimista työyhteisössä, sillä ilman yhteyttä kollegoihin olisi paljon vaikeampaa hakeutua näihin oppimistilanteisiin ja rakentaa luottamuksellisia suhteita, jotka tukevat omaa kehittymistäni.

Tällä viikolla minulle alkoi hahmottua selkeämmin, että osaaminen ei kehity vain teknisiä asioita opiskelemalla, vaan myös toimimalla tiimissä, ymmärtämällä järjestelmien kokonaisuuksia ja reflektoimalla omaa oppimistaan. Esimerkiksi Logic Appsin virheenkäsittely on tekninen asia, mutta sen merkitys avautuu vasta, kun ymmärtää, miten vikatilanteet vaikuttavat loppukäyttäjään ja millainen vastuu kehittäjällä on järjestelmän toimivuudesta.

3.6 Seurantaviikko 6

Maanantai 7.4.

Viikko alkoi uudella koulutusteemalla, jonka aiheena oli Azure DevOps ja Azure Pipelines. Alkuun perehdytyksessä käsiteltiin yleistä teoriaa sekä Azure DevOpsin rakenteellista kokonaisuutta. Kävi selväksi, että Azure DevOps -alustaa käytetään usein synonyyminä Azure Pipelinesille (julkaisuputket), vaikka todellisuudessa kyseessä on laajempi kokonaisuus, johon kuuluu esimerkiksi työjonot (Boards), versionhallinta (Repos), pakettivarastot (Artifacts), testaussuunnitelmat (Test Plans) sekä julkaisujen hallinta (Release Management).

Kävimme läpi, miten julkaisuputki (build and release pipeline) toimii ja miten sen voi määritellä joko YAML-tiedostojen tai perinteisen käyttöliittymän (classic interface) avulla. Esimerkkinä tarkastelimme, miten rakennusagentit (build agents) toimivat, kuinka julkaistavia artefakteja (artifacts) siirretään eri vaiheiden välillä, sekä miten automaattisia laukaisimia (triggers), kuten jatkuva julkaisu (continuous deployment trigger), voidaan hyödyntää julkaisuprosessin aloittamiseksi ilman manuaalisia toimenpiteitä.

Koulutuksessa painotettiin myös CI/CD-prosessin merkityksestä nykyajan ohjelmistokehityksessä. Virheitä ja manuaalista työtä voidaan vähentää automatisoinnilla huomattavasti, sekä varmistaa että kaikki asennukset ja julkaisut tapahtuvat toistettavasti ja jäljitettävästi. Esimerkkinä käytiin

idempotenttien Bicep-tiedostojen käyttöä infrastruktuurin hallinnassa, eli miten samoja määrittelyjä voidaan ajaa useamman kerran ilman, että lopputulos muuttuu, mikä lisää luotettavuutta.

Julkaisuputkien monivaiheisuus ja se millaisia haasteita voi tulla vastaan esimerkiksi salaisuuksien hallinnassa tai testauksen automatisoinnissa, tuli esiin, kun kävimme läpi todellisia esimerkkejä julkaisuputkista eri ympäristöissä, kuten F&O, BC yms. Huomasin, että jopa yksinkertaisissa projekteissa julkaisuputkien rakenteet noudattivat samoja periaatteita.

Tuntui että sain hyvän yleisen kokonaiskuvan Azure DevOps -ympäristöstä ja miten automatisoitu julkaisuputki toimii, mutta käytännön toimivuus ehkä vielä vähän kysymysmerkki, mutta se varmasti selkeentyy, kun pääsen itse toimimaan näiden parissa.

Tiistai 8.4.

Päivän koulutuksessa palattiin OData-integraatioihin, jossa jatkettiin työskentelyä Chinook-projektissa, jota työstimme aikaisemmin. Saimme koulutuksessa tehtävänannon, jossa teimme projektiin uudet pyydetyt toiminnot. Päivä oli jaettu kahteen vaiheeseen. Ensimmäisessä vaiheessa saimme tehtävän, johon saimme aikaa toteuttaa se itsenäisesti, ja tämän jälkeen siirryimme seuraavaan vaiheeseen, joka sisälsi toisen vähän laajemman tehtävän.

Ensimmäinen tehtävänanto oli toteuttaa uusi Azure-funktio, joka hakee genre-tiedon CRM:stä ID-arvon perusteella. Sain tämän tehtävän tehtyä annetussa ajassa. Funktio piti nimetä GetGenreName-nimellä ja se hyödynsi ChinookODataClient-luokkaa ja GetGenreByIdAsync-metodia ja palauttaa vastauksena uuden GenreDto-olion, jossa on genren nimi.

Toisessa tehtävässä tarkoituksena oli toteuttaa samaan CRM:ään soittolistan päivittävä toiminto. Funktion oli tarkoitus vastaanottaa JSON-muotoisen UpsertPlaylistDto-olion, joka sisälsi soittolistan ja kappaleiden nimet. Logiikan tuli siis käsittää soittolistan luonti, olemassa olevan päivittäminen sekä kappaleiden synkronoinnin. Alkuun minulla meni aika pitkä aika miettiessä, miten lähden tätä toteuttamaan ja myöhemmin törmäsin ongelmaan, jossa päivitys loi uusia soittolistoja vanhojen päivittämisen sijaan. En saanut asiaa ratkaistua annetussa ajassa ja tämä tehtävä jäikin meille kotitehtäväksi, jota oli perjantaihin asti aikaa tehdä.

Keskiviikko 9.4.

Päivä meni pääosin siihen, että työskentelin tiistain UpsertPlaylist-toiminnon parantamiseen ja viimeisteliin. Suurin haaste oli edelleen se, että päivitys ei kohdistunut aiemmin luotuun soittolistaan, vaan järjestelmä loi joka suorituksella uuden soittolistan uudella ID:llä. Ongelma liittyi siihen, miten

hain entiteettiä ja miten liitin sen kontekstiin ennen kuin käytin SaveChanges-kutsua. Sain kuitenkin lopulta ongelman ratkaisua esimieheni avulla, kun tarkistimme, missä päin koodissa entiteetin tunnistus oikein tapahtuu.
Sain myös esihenkilöltäni uuden tehtävän, jossa minun pitää toteuttaa toiminto, joka muuntaa JSON-muotoisen aineiston XML:ksi ja validoi sen XSD-skeemaa käyttäen. En aloittanut vielä varsinaista koodausta, koska alkuun tutkin annetun skeeman rakennetta ja hahmottelin, miten lähestyisin tehtävää. Tutkin myös netistä, miten XML-validaatio toimii tarkemmin C#-koodissa ja miten JSON -> XML -muunnos kannattaa toteuttaa. Minulla oli kyllä ideaa siitä, miten sen toteuttaisi, esim. JavaScriptillä, mutta koska C# on vielä vähän uudempi kieli, niin lähdin liikkeelle perusteiden tutkimisesta ennen kuin lähin toteuttamaan toimintoa.

Torstai 10.4.

Tänään jatkoi siitä mihin eilen jäin JSON -> XML-muunnos- ja validointitehtävän parissa. Aloitin toteuttamaan tehtävää ja sain luotua toimivan version metodista, joka vastaanottaa JSON-aineiston, muuntaa sen oikean rakenteen mukaiseen XML-muotoon ja validoi tulosta XSD-skeemaa vasten.

Ratkaisulogiikkani toimi niin, että alkuun JSON-tiedot deserialisoidaan ItemJson-olioksi, jonka jälkeen muunnetaan XML-rakenteeseen käyttäen sitä varten tehtyä luokkaa. Lopuksi käytin XmlSerializer-luokkaa muodostamaan tiedoista XML-tiedoston ja XmlReader-pohjaista toteutusta skeemavalidaatioon.

Päivään sisältyi myös tutkiskelua siitä, mitä työkaluja käyttää XML-tiedostojen luontiin ja skeemavalidaatioon.

Perjantai 11.4.

Kävimme aamupäivällä läpi keskiviikolta jäänyttä UpsertPlaylist-toimintoa. Esitin, miten lähestyin ratkaisua ja miten toteutin sen. Kerroin kanssa ongelmista, johon törmäsin ja miten sain ne ratkaistua. Myös kollegani, joka käy samaa koulutusta läpi, esitteli oman ratkaisunsa ja oli opettavaa nähdä, miten hän on tehnyt sen vähän erilaisella logiikalla.

Iltapäivällä tutustuimme yksikkötestaukseen ja Visual Studion tarjoamiin testaustyökaluihin, jotka kuitenkin olivat aika mitäänsanomattomia eikä niin käytännöllisiä. Loimme testiprojektia, jossa hyödynnettiin Moq-kirjastoa riippuvuuksien eri olioiden mockkaamiseen. Testasimme käsillä olevan ChinookFunction-luokan OrderPlaylist- ja GetPlaylist- metodeja sekä onnistuneilla että virheellisillä syötteillä. Esimerkiksi yksi testi varmisti, että OrderPlaylist palauttaa oikean asiakkaan nimen, kun syöte on oikea, ja toinen tarkisti, että GetPlaylist-metodi palauttaa BadRequest-vastauksen, jos nimi-parametri puuttuu.

6. Viikon analyysi: Integraation luotettavuus ja viestien uudelleenlähetyksen hallinta

Tämä kuudes viikko toi yhä selvemmin esille sen, kuinka vahvasti moderni ohjelmistokehitys perustuu automatisoituun julkaisuun, testattavuuteen ja rakenteelliseen yhteensopivuuteen järjestelmien välillä. Azure DevOpsin opiskelu ja käsittely antoi parempaa käsitystä siitä miten eri vaiheiden kuten buildin, testauksen ja tuotantoon viennin määrittely YAML-muodossa helpottaa ja mahdollistaa kontrolloitua ja jäljitettävää käyttöönottoa. Tämä selvästi liittyy Infrastruktuuri koodina (Infrastructure as Code) -ajatteluun, jossa koko infrastruktuuri rakennetaan, testataan ja versioidaan aivan kuten sovelluskoodikin. Kim (2021, luku 9) tukee tätä sillä hänen mukaansa infrastruktuuri koodina mahdollistaa ympäristöjen versioinnin ja testauksen samalla tavoin kuin sovelluskoodin, mikä tukee jatkuvan julkaisemin laatua ja toistettavuutta.

Integraatiokehityksen kannalta viikon pääaiheena oli saada UpsertPlaylist-funktio kehittäminen niin, että se olisi idempotentti. Integraation on oltava luotettava tilanteessa, jossa sama pyyntö voi saapua useammankin kerran erilaisten syiden takia, esimerkiksi verkko viiveen taikka toistuvan päivitystoiminnon vuoksi. Tässä huomaa, että vaikka koodin palauttaa onnistuneen vastauksen on tärkeää tarkistaa vielä hieman pintaa syvemmältä, että kai tieto on oikeasti tallentunut haluttuun paikkaan, ja onko se tallentunut oikein. Wolff (2016, luku 8.4) korostaa tätä niin paljon, että sanoo viestien uudelleenlähetyksen olevan lähtökohta, eikä poikkeus, ja että järjestelmän täytyy olla suunniteltu tämän mukaisesti. Viestien uudelleenyritysten hallinta kuuluu siis järjestelmän luotettavuuden ytimeen, sillä onnistuneen käsittelyn tunnistaminen, tilan tarkistaminen ennen muutoksen tekemistä ja tallennuksen idempotentti toteutus ovat kaikki keskeisiä keinoja, joiden avulla järjestelmä pysyy vakaana myös poikkeustilanteissa. Tämä korostui konkreettisesti oman toteutukseni aikana, kun ensin luulin tallennuksen menneen juuri kuten pitikin, mutta kun hieman enemmän tarkisteli niin huomasi toistuvien viestien käsittelyn olevan vääränlainen.

Viikon lopussa pääsin rakentamaan myös yksikkötestejä C#-koodilla, joissa testattiin loogisia reittejä ja virhetilanteita mockatuilla riippuvuuksilla. Käytimme xUnitia ja Moq:ia, mikä konkretisoi testattavan koodin eriyttämisen merkityksen. Freeman ja Pryce (2009, luku 5) kuvaavat testivetoista kehitystä prosessina, jossa ensin luodaan epäonnistuva testi, sitten tehdään juuri sen verran koodia, jotta testi menisi läpi, ja lopuksi refaktoroidaan rakenne. Tämä kuvaus on tosi osuva ja se auttoi selvästi jäsentelemään prosessin päässäni niin, että osaisin sanoittaa sen hyvin myös muille.

Oli opettavaa, kun kollegani kanssa ratkoimme samaa tehtävää eri tavoin. Käytiin yhdessä läpi eriäviä lähestymistapojamme, joka antoi uudenlaista ymmärrystä sekä itse ongelmasta että toistemme tekniikasta lähestyä ratkaisuja. Tämä on yksi tiimiperustaisen oppimisen hyödyistä, nimittäin se parantaa tiedon omaksumista, kehittää ongelmanratkaisutaitoja ja tukee pehmeiden taitojen, kuten empatian ja viestinnän, kehittymistä (Mullen 29.11.2024).

3.7 Seurantaviikko 7

Maanantai 14.4.

Kävimme läpi aamupäivästä läpi viimeisen koulutuksen, joka liittyy juuri omaan rooliini integraatiokehittäjänä. Aiheena oli rajapintojen hallinta ja miten se toteutuu meidän yrityksemme omissa ympäristöissä.

Jatkoin lounaan jälkeen XML-muuntajan kanssa ja aloitin toteuttamaan siitä Azure-funktiota. Lähdin siirtämään aiemmin tehdyn muuntaja-toiminnon logiikkaa Azure-funktion sisään niin, että se vastaanottaa JSON-pyynnön ja palauttaa muunnetun XML-tiedoston. Polut XSD-validointia varten onnistuivat hyvin ja sain vaivattomasti lisättyä toiminnon, joka tallentaa palautetun tiedoston lokaalisesti, jotta uudet XML-tiedostot jäävät talteen. Rakensin tämän ratkaisun HttpTrigger-pohjalle ja siinä toimii POST- ja GET-pyynnöt.

Tiistai 15.4.

Lähdin viimeistelemään XML-muuntaja Azure-funktiota ja sain sen ongelmitta julkaistua Azureen jonka jälkeen testasin molempia päätoiminnallisuuksia Brunolla, eli JSON:sta muunnoksen XML:ään ja viimeisimmän XML-tiedoston haun GET-pyynnöllä. Sain loppuun vielä esimieheltäni ohjeen, että funktion pitäisi palauttaa XML-sisältö suoraan selaimeen eikä pelkkänä datavastauksena. Sain nopeasti ja vaivattomasti tämän hoidettua, jonka jälkeen parin testausyrityksen jälkeen paketoin tehtävän kasaan.

Keskiviikko 16.4.

Tänään oli aika vähäsanainen päivä. Kävin aamupäivällä läpi Azure-funktiotani ja katselin, olisiko jotain tapoja millä saisin parannettua sitä, esimerkiksi kirjoittamalla hieman tehokkaampaa koodia, mutta en keksinyt mitään parannettavaa. Odotin iltapäivään, että esimiehelläni on vapaata ja esittelin nopeasti tämän pienen sivutehtävän ja hän sanoi, että hänellä on minulle yrityksen sisäiseen integraatioon liittyvä tehtävä. Varasimme seuraavalle aamulle tapaamisen, jossa käymme sitä tarkemmin läpi.

Torstai 17.4.

Aamun palaverissa saamani tehtävänanto liittyi uuteen integraatiotehtävään, jossa tarkoituksena olisi toteuttaa toiminto yrityksemme HR-järjestelmän ja projektinhallinnan työkalun välille siten, että kun työntekijöiden poissaolot tai lomat syötetään HR-järjestelmään, ne siirtyvät myös projektinhallinnassa käytettyyn järjestelmään ja tallennetaan ne suunnitelmiksi kyseisille päiville. Toiminnon on

tarkoitus myös poistaa mahdollisesti jo valmiiksi olevat suunnitelmamerkinnät tälle samalle ajanjaksolle, jotta sinne ei jää minkäänlaisia päällekkäisyyksiä.

Loppupäivä meni aika lailla siinä, että tutustuin projektinhallinnan API-dokumentaatioon ja hahmottelin, mitkä päätoiminnot ja tietorakenteet ovat oleellisimpia juuri tämän toteutuksen kannalta. Kirjasin ylös alustavasti, mitä ohjelmointirajapinnan endpointteja tulen todennäköisimmin kutsumaan ja minkälaiset datamallit niihin liittyvät. Lähdin luomaan omaa versionhallinnan kehityshaaraa integraatiota varten ja alustin sen sisällön niin, että sain ympäristön valmiiksi oman toteutukseni kehittelyyn. Päivän lopuksi tein alustavaa suunnitelmaa siitä, miten lomatietojen tuonti ja olemassa olevien merkintöjen poisto tulisi jäsennellä, mutta se jäi vielä ylätasolle, sillä huomasin, että tällä osaamistasolla oli hieman vaikeaa käsittää ratkaisun koko arkkitehtuuria.

Perjantai 18.4.

Oli pääsiäinen, mutta päätin kuitenkin tänään alkaa kirjoittamaan varsinaista koodia tähän annettuun projektiin. Aloitin toteuttamaan metodia, jonka tarkoituksena oli lähettää poissaolotiedot rajapintaan ja luoda näiden perusteella uudet suunnitelmamerkinnät valituille päiville. Metodin logiikka toimi niin, että se muodostaa HTTP POST -pyynnön ja lähettää sen eteenpäin JSON-muodossa. Metodi lopussa käsittelee vastauksen ja palauttaa sekä onnistumisen tilan, että palvelimen vastauksen.

Illalla palasin takaisin integraatiotehtävän ääreen ja viimeistelin kesken jääneen metodin virheenkäsittelyn ja lokituksen. Nyt metodi palauttaa asialliset virhekoodit selkeällä palautteella ja lokimerkinnöillä.

7. Viikon analyysi: Tiedon esitysmuodon hallitut muunnokset ja XSD-validointi

Tällä viikolla keskeiseksi tavoitteeksi asetin hoitaa loppuun harjoitustehtävän, jonka tarkoitus oli luoda Azure-funktioon pohjautuva muunnnospalvelu, joka vastaanottaa JSON-muotoista tietoa ja palauttaa sen XML-muotoon. Tehtävä vei syvemmälle siihen, miten tiedon esitysmuotoja voidaan muuntaa hallitusti, ei vain syntaksin vaan myös rakenteellisten sääntöjen mukaisesti. Tässä tärkeäksi nousi XSD-skeemaan perustuva validointi ja siihen liittyvä virheenkäsittely.

Itse muunnoksen tein C#-kielellä XDocument-luokan avulla, joka tarjosi riittävän joustavan tavan rakentaa XML-rakenteita ohjelmallisesti. Kuten on tullut kuitenkin ilmi, niin pelkkä muunnos ei riitä, vaan rakenteen pitää myös vastata tarkasti odotettua muotoa. Tähän käytin XmlReaderSettings-luokkaa, jonka avulla sainkin määriteltyä skeemaan perustuvat tarkistukset ja liittämään mukaan virheenkäsittelijän. Huomasin käytännössä, kuinka XmlReaderSettings-luokan tarjoamat ominai-

suudet, kuten skeeman määrittely ja virheenkäsittelijän liittäminen, toi projektin kehityksen etenemisen kannalta varmuutta, sillä virheet saatiin havaittua jo ennen kuin data edes lähti jatkokäsittelyyn. Tämä tulee myös esille Microsoftin dokumentaatiosta (2024a), jonka mukaan nämä asetukset mahdollistavat sen, että XML-datan rakennetta voidaan tarkastella ohjelmallisesti määriteltyjä sääntöjä vasten, ja poikkeamista ilmoitetaan välittömästi. Tämän auttaa siinä, ettei ongelmat jää piiloon tai siirry mahdollisiin myöhempiin järjestelmiin asti, vaan ne voidaan käsitellä jo alkumetreillä.

Binstock, Peterson ja Smith (2002, luvut 1 ja 2.3) kuvaavat XML-skeemaa nimenomaan sopimuksena, joka määrittelee, mitä tietoja järjestelmä voi hyväksyä ja mitä ei. Tämä konkretisoitui omassa työssäni, kun huomasin, ettei toiselle järjestelmälle riittänyt pelkkä teknisesti oikea XML, vaan sen rakenteen piti täsmätä juuri skeeman määrityksiä. Näin skeemasta tulee osa toiminnallisuutta, ei vain dokumentti tai ohje.

Viikon toisella puoliskolla aloitin uuden integraation, jossa HR-järjestelmään syötetyt lomatiedot siirretään projektihallinnan järjestelmään ja mahdolliset vanhat merkinnät poistetaan päällekkäisyyksien estämiseksi. Lähdin etenemään projektin kanssa tutustumalla kohdejärjestelmän API-dokumentaatioon ja hahmottamalla, mitä tietoa tarvitaan ja miten se lähetetään. Itse koodaamisen aloitin toteuttamalla metodin, joka lähettää poissaolotiedot rajapintaan HTTP POST -pyynnöllä. Rakensin mukaan myös virheenkäsittelyn ja lokituksen, ongelmien havaittavuuden ja jäljitettävyyden helpottamiseksi.

Tällä viikolla olennaiseksi nousi miten virheenkäsittely ja validointi ovat erittäin kriittisiä vaiheita, ei siksi, että ne olisivat vaikeita toteuttaa, vaan koska ne vaikuttavat suoraan siihen, toimiiko koko integraatio alun perinkään luotettavasti.

Tämä tehtävä loi lisäarvostusta suunnittelulle ja testaamisen roolille. Jo pienillä ratkaisuilla, kuten XML:n oikean muotoilun palauttamisella selaimeen, voi olla iso vaikutus käytettävyyteen ja yhteistyöhön muiden järjestelmien kanssa. Samalla ymmärrykseni vahvistui siitä, että kehittäjän työssä ei riitä, että jokin “toimii”, sen on myös toimittava oikein, selkeästi ja ennakoitavasti. Tämä ajattelutapa on alkanut siirtyä vahvemmaksi osaksi ajattelutapaani, sillä vaikka näin totta kai olen aina ajatellutkin, mutta kouluprojekteissa harvemmin ei tapahdu mitään niin kriittistä, jos toiminnossa tulee myöhemmin virheitä, verrattuna siihen mitkä ovat seuraukset saman asian suhteen työympäristössä.

3.8 Seurantaviikko 8

Maanantai 21.4.

Tänään oli pääsiäispäivä, mutta aloitin työstämään seuraavaa metodia, jonka tarkoituksena oli hakea kaikki olemassa olevat suunnitelmamerkinnät annetusta päivämäärästä eteenpäin ja poistaa ne. Tarkoituksena oli, ettei samalle ajanjaksolle jää ristiriitaisia tai päällekkäisiä merkintöjä, kun HR-järjestelmään päivitetyt tiedot tulevat käyttöön.

Metodin logiikan toteutin ensin nykyisten merkintöjen haun API-kutsulla, minkä jälkeen käytin DELETE-kutsua poistamaan nämä merkinnät. Ajatuksena oli, että uudet merkinnät tallennettaisiin aina uudelle pohjalle, vaikka osa näistä merkinnöistä olisikin muuttumattomia. Aikavälien suodatuksessa painin hetken, mutta sain loppujen lopuksi metodin toimimaan.

Tiistai 22.4.

Tänään aloitin käymällä läpi tehtyjä metodeja ja myös vähän tarkemmin koko ratkaisua. Huomasin, että olin tehnyt useita uusia malleja ja määrittelyjä metodeja varten, vaikka projektissa olikin jo valmiiksi olemassa olevia datamalleja, joita olisin voinut hyödyntää suoraan metodeissa. Palasin takasin metodien ääreen ja korvasin ylimääräisiä rakenteita niillä, joita löytyi jo valmiiksi muista projekteista.

Oli hieman turhauttavaa huomata, että olin edennyt liian nopeasti ja alkanut toteuttaa asioita tutkimatta kunnolla, mitä kaikkea ratkaisusta jo valmiiksi löytyy. Päivä meni aika lailla siihen, että kävin koodia uudelleen läpi, poistin turhat osat ja yhtenäistin toteutusta olemassa olevan rakenteen mukaiseksi. Tässä tuli toisaalta parempaa yleiskuvaa ratkaisun sisällöstä, joka varmasti auttaa jatkossa.

Keskiviikko 23.4.

Päivä alkoi 1:1-keskustelulla esimieheni kanssa, jossa kävimme läpi lähtökohtaisesti tähänastista etenemistä ja yleistä fiilistä työn suhteen. Kävimme läpi asioita, mitkä on minusta tuntunut sujuneen hyvin, missä on ollut haasteita ja mitä kehityskohteita haluaisin jatkossa painottaa. Oli myös puhetta kesäloman ajankohdasta sekä syksyn mahdollisista tulevista projekteista, jotka todennäköisesti osuvat minulle. Keskustelu oli kannustavaa ja oli kiva saada vähän kuvaa siitä miltä tulevaisuus pidemmällä aikavälillä näyttää.

Tapaamisen lopussa sanoin, että olin saanut integraatiometodit alkuvaiheeseen nähden riittävään kuntoon ja sovimme parin tunnin päähän toisen tapaamisen. Tässä toisessa tapaamisessa esimie-

heni antoi minulle ohjeet Azure-funktion toteuttamiseen ja minkälaiset toiminnat sillä pitäisi olla. Tavoitteena oli, että funktio hakee sähköpostiosoitteen perusteella API:n kautta käyttäjän ja sitten tekee suunnitelmamerkintöjen poiston ja luonnin.

Päivän lopussa aloitin tämän funktion alustamisen ja hahmottelin sen rakennetta sekä riippuvuuksia. Koodausta en pahemmin enää tehnyt.

Torstai 24.4.

Tänään sain jatkettua projektin toteutusta ja sain pääpirteittäin yhteysvalmiin Azure-funktion, joka oli siinä kunnossa, että se muodosti oikeat pyynnöt rajapintoihin.

Ensimmäinen testiajo ei kuitenkaan onnistunut, sillä sain virheilmoituksen, joka viittasi siihen, että minulla ei ollut tarvittavia oikeuksia käyttää testirajapintaa. Kysyin asiasta esihenkilöltäni, ja sain ohjeet olla yhteydessä helpdeskiin. Ensimmäisen ja toisen saamani autentikaatiotokenin kanssa oli ongelmia, mutta hetken säätämisen jälkeen saatiin ne toimimaan toivotusti. Sen jälkeen aloin testaamaan funktion toimintaa, mutta törmäsin uuteen ongelmaan, joka liittyi uuden suunnitelman talletukseen. Azure-funktio onnistui lähettämään pyynnön rajapintaan, mutta vastaanotettu virheilmoitus oli erittäin ympäripyöreä eikä suoraan kertonut, mikä oli pielessä. Taistelin tämän ongelman kanssa aika pitkään, kunnes päätin jättää ratkomisen seuraavalle päivälle.

Perjantai 25.4.

Aamulla meillä oli yrityksen yhteinen kuukausittainen aamupala ja siihen perään kuukausittainen tapaaminen, jossa kävimme yrityksen nykyistä tilaa, miltä tulevaisuus näyttää sekä minkälaisia sisäisiä tapahtumia on meneillään. Tällä hetkellä on päällä äänestys siitä, mitä uutta tekoälyprojektia lähdetään yrityksen sisäisesti kehittämään seuraavassa yrityksen hackathonissa. Voittajien idealle ja toteutukselle oli luvattu palkkio, joka teki tästä hauskan kisan.

Kuukausittaisen tapaamisen jälkeen oli lounastauko ja sen jälkeen minulla oli tapaaminen esimieheni kanssa, jossa pyysin häneltä apua eilen kohtaamassani ongelmassa. Virhe liittyi rajapinnan datamallin määrittelyyn, sillä käyttämäni malli ei ollut täysin odotetun mukainen, vaikka tein sen rajapintadokumentaation mukaan oikein. Lähdimme esimieheni kanssa selvittämään tilannetta ja se ratkesi sillä, että yksinkertaisesti vähän testailtiin yksi muuttuja kerrallaan, että mikä aiheuttaa virheen. Esimieheni sanoi, että tällaiset kummalliset virhekoodi ovat välillä vähän sellaisia ”tietäjät tietää”-tilanteita, joita alkaa vaan oppia kokemuksen kautta.

Kun virhe saatiin korjattua, toimi tallennus toistaiseksi odotetusti ja funktio loi uudet suunnitelmat rajapinnan kautta onnistuneesti.

8. Viikon analyysi: Laadukkaan integraation toteutettaminen

Tällä viikolla tekninen kehitys painottui suunnitelmamerkintöjen käsittelyyn, jossa rajapinnan DELETE-pyyntöjä hyödynnettiin olemassa olevien merkintöjen poistamiseen ennen uusien lisäämistä. Huomio kiinnittyi erityisesti rajapinnan-arkkitehtuurin perusperiaatteisiin, kuten idempotenssiin, joka oli ollut jo aikaisemmilla viikoilla esillä. RESTful Web APIs -teoksen (Richardson, Amundsen & Ruby 2013, luku 11) mukaan DELETE-pyynnön idempotenttius takaa sen, että pyynnön voi suorittaa useamman kerran ilman pelkoa siitä, että järjestelmän tila muuttuisi odottamattomasti. Tämä oli oleellista juuri tässä tehtävässä, koska oli tärkeää pystyä poistamaan kaikki vanhat merkinnät luotettavasti, vaikka poistometodia kutsuttaisiin useamman kerran. Ilman idempotenttiutta olisi riskinä, että useampi poisto aiheuttaisi virhetilanteita tai poistaisi liikaa tietoa. Nyt pystyin rakentamaan logiikan, jossa DELETE-pyyntöä voi käyttää turvallisesti aina ennen uusien merkintöjen lisäämistä, mikä selkeytti toteutusta ja vähensi virheiden mahdollisuutta.

Koodin refaktorointi ja datamallien yhtenäistäminen nousivat esiin, kun aiemmin olin luonut ylimääräisiä rakenteita sellaisiin tilanteisiin, joihin olisi löytynyt valmiit, uudelleenkäytettävät mallit projektista. Davis (2019, luku 5) painottaa pilvinatiivien sovellusten kehittämisessä toiston välttämistä ja yhteisten rakenteiden käyttöä, sillä tämä parantaa järjestelmän ylläpidettävyyttä ja mahdollistaa joustavan kehityksen. Refaktoroinnin aikana yhtenäistin projektin mallit ja poistin ylimääräisen koodin, mikä teki kokonaisuudesta selkeämmän ja laadukkaamman.

Azure-funktion rakentaminen toi esille palvelimettoman-arkkitehtuurin edut, kuten resurssien automaattisen skaalaamisen ja käyttöönoton helppouden. Davis (2019, luku 10) kuvaa, kuinka palvelimettomat-mallit nopeuttavat kehitystyötä ja mahdollistavat toimintojen yksittäisen hallinnan, mutta edellyttävät huolellista riippuvuuksien ja autentikoinnin suunnittelua. Tämä konkretisoitui projektissa, kun alku oli vähän mutkikasta puuttuvien oikeuksien vuoksi. Toisaalta sen jälkeen, kun oikeudet oli hoidettu kuntoon, oli eteneminen tältä kannalta sulavaa omassa kehitysympäristössä.

Virheenkäsittelyn näkökulmasta viikon merkittävin havainto oli se, että rajapintojen virheilmoitukset eivät aina olleet riittävän informatiivisia. Boten (2022, luku 3) korostaa järjestelmän havainnoitavuuden, eli tilan seurannan ja mittaamisen, roolia modernissa pilvikehityksessä. Hänen mukaansa järjestelmän sisäisten tapahtumien mittaaminen ja seuranta on olennainen osa virheiden diagnosointia ja jatkuvaa laadun parantamista. Käytännössä havahduin siihen, että pelkkä virheilmoituksen lukeminen ei aina riitä, vaan tarvitaan laajempi näkyvyys järjestelmän tilaan ja tapahtumiin, esimerkiksi lokien ja jäljitettävyyden avulla.

Viikon kokonaisuus osoitti, miten kirjallisuudessa korostetut rajapinta-periaatteet, koodin selkeys ja järjestelmän havainnoitavuus eivät ole pelkkiä teoreettisia suosituksia, vaan näkyvät myös konkreettisesti arkipäiväisessä kehitystyössä. Reflektoimalla näitä oppeja oman projektin ratkaisuihin, uskon vahvistavani pohjaani, jolla rakentaa luotettavampia ja helpommin ylläpidettäviä integraatioita.

4 Pohdinta

Tässä luvussa tulen käsittelemään ammatillista kehittymistäni viimeisen kahdeksan viikon mittaisen opinnäytetyöprosessin aikana vertaamalla lähtötilanteen osaamistani päiväkirjamerkintöjen analyysiin. Pohdinnassa käsittelen sekä teknistä että ammatillista kehittymistäni, löytämiäni ratkaisuja sekä uusia menetelmiä, oppimiani asioita ja sitä miten olen hyödyntänyt työn analysointia oppimisessani. Lopuksi tulen pohtimaan myös tulevia kehittämiskohteitani ja sitä, miten voin jatkaa osaamiseni kehittämistä urallani.

4.1 Oma kehittyminen

Opinnäytetyön alussa oma lähtötasoni C#-ohjelmoinnin ja Azure-ympäristön suhteen oli melko matala. Ensimmäiset viikot menivät pitkälti perusasioihin tutustuessa, sillä jouduin omaksumaan uudet ohjelmistoprojektin rakenteet, integraatiotiimin käytännöt ja kokonaan uuden työkalupakin. Usein huomasin, että joudun palaamaan aivan perusteisiin, kysymään neuvoa kokeneemmilta kollegoilta vaikkapa C#-koodin syntaksin yksityiskohdista tai Azuren palveluiden käytöstä. Alussa päästyäni projekteihin mukaan työskentely tuntui epävarmalta, virheilmoitukset tulivat nopeasti vastaan ja moni asia tuntui hankalalta.

Pikkuhiljaa toistuvat tehtävät ja arjen rutiinit alkoivat kuitenkin vahvistamaan itseluottamustani. Viikkojen myötä opin kirjoittamaan C#-koodilla pieniä ohjelmamoduuleja ja skriptejä, hallitsemaan .NET-ympäristön perusasioita sekä hyödyntämään Azure-palveluita projektin tarpeiden mukaan. Yksi konkreettinen esimerkki oli Azure-funktion toteuttaminen, jossa rakensin toiminnon, joka haki käyttäjän tietoja ulkoisesta API-rajapinnasta ja päivitti tietoja eteenpäin integraation kautta. Tämän tehtävän myötä jouduin perehtymään C#-koodin HTTP-pyyntöihin ja Azuren autentikointiin käytännön tasolla. Alun kompastelut, virheilmoitukset ja puutteelliset käyttöoikeudet, opettivat minua hyödyntämään vahvemmin lokitietoja ja hakemaan ratkaisuja määrätietoisesti, ja koen, että tämä antoi minulle hyvää kokemusta, miten selvittää ja välttää vastaavia ongelmia tulevaisuudessa.

Teknisten taitojen lisäksi opin paljon ammatillisessa mielessä. Vaikka olen jo entuudestaan luonteeltani avoin kommunikoija, oli alussa kuitenkin ehkä hieman aristusta, että keneltä kysyy apua ja missä vaiheessa, mutta aika nopeasti tunsin oloni mukavaksi nojautua kollegoiden tukeen. Esihenkilön ja kollegoiden palaute on ollut tärkeää, sillä sen avulla tunnistin omia kehityskohteitani, kuten koodin selkeydessä ja dokumentoinnissa.

Ajanhallintani parani projektin aikana, kun totuin suunnittelemaan työviikot päiväkirjaan asetettujen tavoitteiden mukaisesti. Loppua kohden huomasin, että itseluottamukseni kasvoi, etenin projek-

teissa paljon varmemmin kuin alkuun. Alussa kysyin paljon varmistelua välivaiheissa, että meneekö oikein ja miten lähden toteuttamaan ratkaisua seuraavaksi, vaikka minulla olikin jo mielessä, miten se pitäisi tehdä, mutta epävarmuuteni vuoksi halusin kysyä tästä vielä varmistusta. Koen että tämä kehitys on ollut urani alkuvaiheessa erityisen tärkeää ja antaa hyvän pohjan toimia jatkossa entistä varmemmin konsulttina.

4.2 Uudet ratkaisumallit ja menetelmät

Työskentelyn aikana olen löytänyt useita uusia ratkaisumalleja ja menetelmiä, jotka ovat olleet merkittävässä roolissa omassa kehittymisessäni. Yksi keskeisimmistä oivalluksista liittyi pilvi-infrastruktuurin hallintaan koodin avulla. Opinnäytetyöjakson aikana osallistuin Azure Bicep -koulutukseen, jonka kautta avautui täysin uusi näkökulma infrastruktuurin hallintaan. Bicep-mallien avulla pystyin kuvaamaan esimerkiksi Azure-funktio-palveluita ja Logic App -työnkulkuja deklaratiivisesti koodina, mikä oli minulle aiemmin vieras ajattelutapa. Vielä koulutuksen jälkeen tämä oli käytännössä minulle hämärää, mutta juuri tämän kahdeksan viikon ajan jakson loputtua pääsin suorittamaan julkaisua tuotantoon ensimmäisen kerran ja silloin tämä konkretisoitui minulle lopullisesti.

Toinen tärkeä uusi toimintamalli oli systemaattisempi lähestyminen ohjelmointitehtäviin. Opinnäytetyön aikana otin versionhallinnan (Git) päivittäiseen käyttöön aiempaa syvällisemmin. Opin hyödyntämään haaroja uuden ominaisuuden kehittämisessä ja tekemään pull request -koodikatselmointeja yhdessä tiimin kanssa. Tähän mennessä kouluprojekteissa eri haarojen välillä työskentely ei ole ollut niin olennainen osa kuin nyt, joten tämä oli siltä kannalta uutta menetelmää, ja koin sen hyvin tehokkaaksi tällaisessa monitoimisessa ympäristössä.

Aloin myös soveltaa testauslähtöistä kehitystä, vaikkei se aluksi ollut vahvin osa-alueeni. Kirjoitin yksinkertaisia yksikkötestejä C#-koodilla ja testasin Azure-funktioiden toiminnallisuuksia paikallisessa kehitysympäristössä. Lisäksi tutustuin Bruno-työkaluun API-kutsujen testaamisessa, mikä tehosti integraatioratkaisujen kehittämistä ja antoi parempaa varmuutta siitä, että toteutetut toiminnot toimivat odotetusti. Brunon käyttöä minun on kyllä vielä tehostettava, sillä huomasin, että monissa tilanteissa olisin päässyt vähemmällä muokkailulla, jos olisin jo aikaisemmin testannut kunnolla.

Olen kokenut nämä työkalut todella hyödyllisiksi ja koen, että ne tukevat vahvasti omaa työntekoani integraatiokehittäjänä. Olen tyytyväinen siihen, että työkalupakkini on laajentunut, ja sitä kautta olen saanut uutta ymmärrystä sekä osaamista alaan liittyen.

4.3 Oppiminen päiväkirjamuotoisen opinnäytetyön kirjoittamisesta

Päiväkirjamuotoisen opinnäytetyön tekeminen tarjosi minulle uudenlaisen oppimiskokemuksen. Viikoittaiset päiväkirjamerkinnät pakottivat minut reflektoimaan säännöllisesti omaa tekemistäni ja edistymistäni. Huomasin, että kirjoittaessa jokaisen viikon päätteeksi ylös, mitä olen tehnyt, mitä ongelmia kohtasin ja miten ne ratkaisin, opin ymmärtämään omaa työskentelyprosessiani paremmin. Erittäin motivoivaa on varsinkin se, että pystyn seuraamaan kuinka nopea kehitys oikeasti on näin alkuvaiheessa. Nyt kun katson taaksepäin alkuvaiheitani, tuntuu ihan kummalliselta, että miten niin yksinkertaiset asiat eivät hoituneet välittömästi vaan vaativat keskittymistä.

Päiväkirjan pitäminen auttoi myös pitämään yllä kirjoitustaitoani ja itseilmaisua ammatillisessa kontekstissa. Alussa viikkoraporttien tekeminen tuntui haastavalta, koska en ollut tottunut dokumentoimaan työtäni näin tarkasti. Siksi alkuun moniin osioihin kirjoitinkin vain alkeellisesti asioita, sillä ajatuksella, että korjaan ne myöhemmin sellaisiksi, miksikä ne oikeasti haluan. Huomasin isoimman ongelman olevan siinä, kuinka nopeasti saan tuotettua tekstiä. Ajan myötä kirjoittaminen alkoi kuitenkin tuntua luontevammalta ja huomasin kirjoittamisen nopeutuvan ja olevan vähemmän epämiellyttävää. Kokonaisuutena päiväkirjamuotoisen opinnäytetyön kirjoittaminen opetti minulle systemaattista itsearviointia ja jatkuvan oppimisen asennetta, josta toivon olevan hyötyä minulle myös jatkossa.

4.4 Kiinnostavat havainnot ja niiden hyödyt tulevaisuudessa

Yksi keskeisimmistä havainnoista tämän kahdeksan viikon aikana liittyi siihen, että integraatiokehittäjän ajasta menee yllättävän paljon aikaa olemassa olevan järjestelemän ymmärtämiseen, varsinkin ympäristöön uutena tulleena kehittäjänä. Aikaa meni myös paljon virheiden jäljittämiseen ja järjestelmien välisen yhteensopivuuden varmistamiseen eikä vaan pelkästään uuden koodin kirjoittamiseen. Koen tämän olevan arvokas oppi itselleni kehittäjänä, sillä se auttaa minua arvioimaan jatkossa paremmin sitä, kuinka paljon aikaa joudun varaamaan projektin tekemiseen. Tässä kahdeksan viikon aikana oli useampi kohta, jossa luulin olevani hetkessä valmis, mutta vastaan kuitenkin tuli aikaansa vieviä hidasteita. Tajusin myös, kuinka tärkeää on suunnitella ratkaisuja jo etukäteen ja todella käyttää resursseja myös testaukseen, jotta näitä hidasteitä tulisi mahdollisimman vähän.

Toinen havainto liittyi oman yritykseni tapaan kannustaa jatkuvaan oppimiseen ja innovointiin. Pääsin seuraamaan sivusta, miten yrityksessä järjestettiin sisäinen ideakilpailu ja siihen liittyvä hackathon-tapahtuma. Huomasin, että tällainen kokeilukulttuuri ja uusien tekniikoiden tutkimiseen kannustaminen lisäsi hyvää ilmapiiriä sekä työmotiivia työympäristössäni. Koin tätä jopa henkilökohtaisella tasolla, kun esimieheni, koulutusohjaajat ja muut kollegat kannustivat innoissaan, mutta kuitenkin puskematta liikaa päälle, oppimaan lisää ja oppimaan rauhassa.

4.5 Työn analysoinnin hyödyntäminen

Työn jatkuva analysointi osoittautui yllättävän hyödylliseksi ja tuotteliaaksi kehittymiseni kannalta. Opinnäytetyöni päiväkirjamerkintöihin sisältyi viikoittaisia analyysiosuuksia, joissa arvioin kuluneen viikon onnistumisia ja haasteita suhteessa asetettuihin tavoitteisiin. Näiden viikkoanalyysien avulla pystyin tunnistamaan toistuvia ongelmakohtia työskentelyssäni. Esimerkiksi huomasin alkupuolella projektia toistuvasti aliarvioineeni tehtäviin kuluvan ajan, mikä johti siihen, että usein tuntui siltä, että on kauhea kiire, vaikka mitään aikarajaa ei välttämättä ollut edes annettu. Analyysin kautta tunnistin tämän ajanhallintaongelman ja koen, että jo sen tunnistaminen on saanut minut aikatauluttamaan paremmin sekä antamaan tarkempia aika-arvioita.

Analysointi selvästi auttoi havaitsemaan omia vahvuuksiani ja heikkouksiani. Kun kirjasin ylös, tekemistäni ja pohdin samalla omia hyviä suoriutumisia (esimerkiksi uuden teknologian nopeassa oppimisessa käytännön esimerkeillä) ja missä tarvitsin enemmän tukea (kuten kokonaisuuksien hahmottamisessa), sain selkeämmän kuvan kehittymisprofiilistani. Näiden havaintojen perusteella minulle on nyt selvää mitä osa-alueita minun tulee eniten parantaa, jotta saan vahvistettua heikkouksiani.

Myös koko päiväkirjajakson kertaaminen oli arvokasta. Käymällä läpi kaikkien viikkojen merkinnät ja havainnot, huomasin miten paljon olin oppinut lyhyessä ajassa. Lukemalla ja muistelemalla päiviä, pystyn hahmottamaan, mitä tein väärin ja mitä tein hyvin.

4.6 Tulevaisuuden kehittämiskohteet

Vaikka olenkin kehittynyt paljon kahdeksan viikon aikana, näen selvästi myös useita osa-alueita, joilla haluan kehittää osaamistani edelleen tulevaisuudessa. Teknisen osaamisen kannalta minulla on vieläkin erittäin paljon opittavaa ja aionkin syventää tietojani erityisesti pilvipalveluista ja integraatioarkkitehtuurista. Olen huomannut kertaamalla päiväkirjamerkintöjäni ja omaa työtäni, että isoimmat haasteeni minulla on ollut hahmoittaan kokonaisuuksia, ja kunkin toiminnon roolia osana tätä kokonaisuutta. Azure-ympäristöstä olen oppinut vasta pintaraapaisun. Haluan perehtyä syvemmin Azure-palveluiden, kuten Azure Service Bus -viestijonojen ja Azure Logic Apps -työnkulkujen tehokkaaseen käyttöön. Lisäksi suunnittelen tutustuvani tarkemmin DevOps-käytäntöihin ja työkaluihin. Tämäkin on vähän tämän kahdeksan viikon ulkopuolista, mutta heti seurantajakson perään pääsin luomaan ja käsittelemään Azuren julkaisuputkia ja ne ovat olleet todella kiinnostavia. Mahdollinen Azure DevOps -sertifiointi tai muu pilviteknologioiden sertifikaatti olisi yksi konkreettinen tavoite, joka tällä hetkellä kiinnostaa sekä kannustaa opiskelemaan lisää.

Ohjelmointitaitojeni kehittämiseksi aion laajentaa osaamistani myös C#-kielessä. Erityisen olennaisena pidän sitä, että ymmärrän paremmin, miten sovellusarkkitehtuuri rakentuu pilvipohjaisissa integraatioympäristöissä. Haluan perehtyä esimerkiksi siihen, miten erilaiset komponentit (kuten Azure-funktiot, Logic Apps, Service Bus) toimivat yhteen kokonaisuutena yhdessä, miten oikeanmuotoinen tiedonkulku varmistetaan järjestelmien välillä sekä millaiset arkkitehtuurimallit ovat yleisesti tehokkaimpia integraatioratkaisuissa. Lisäksi yksi tärkeä aihe, jossa minun pitää kehittyä, tehokkaampien ja kattavimpien testien luonti.

Työympäristöön liittyen haluan kehittää erityisesti projektinhallintataitojani. Aion jatkossakin pyytää palautetta työstäni ja osallistua aktiivisesti tiimin yhteistyöhön, koska jo tähän mennessä koen oppineeni kokeneemmilta kollegoilta tosi paljon. Aion pitää keskeisenä myös ylläpitää omaa oppimisen kulttuuria arjessa seuraamalla alan kehitystä ja kokeilemalla uusia teknologioita pienissä kokeiluprojekteissa vapaa-ajalla. IT-ala on jatkuvasti kehittyvä ala, ja nykyinen taloudellinen tilanne sekä tekoälyn nopea kehitys luovat hieman epävarmoja aikoja, jolloin ei ole varaa vain passivoitua pysyttelemään pinnalla, vaan on pakko kehittää itseään jatkuvasti, jotta pystyy kilpailemaan töistä. Tämä ei luo minulle niinkään stressiä vaan antaa hyvää haastetta ja painetta kehittyä paremmaksi.

Lähteet

Boten, A. 2022. Cloud Native Observability with OpenTelemetry. Packt Publishing. E-kirja. Luettu 17.5.2025.

Davis, C. 2019. Cloud Native Patterns. Manning Publications. E-kirja. Luettu 18.5.2025.

Dreyfus, S. E. & Dreyfus, H. L. 1980. A Five-Stage Model of the Mental Activities Involved in Directed Skill Acquisition. University of California, Berkeley, Operations Research Center. Luettu 25.3.2025.

Fowler, M. 2002. Patterns of Enterprise Application Architecture. Addison-Wesley. E-kirja. Luettu 25.3.2025.

GitHub Docs. 2024. GitHub Flow. Luettavissa: https://docs.github.com/en/get-started/quickstart/github-flow. Luettu 20.5.2025.

Ideagroup.fi. 2023. Tiimipäivä – unelmatiimi ei rakennu itsestään. Luettavissa: https://ideagroup.fi/tiimipaiva-unelmatiimi-ei-rakennu-itsestaan/. Luettu 8.3.2025.

Knuth, D. E. 2011. The Art of Computer Programming, Volume 1: Fundamental Algorithms. Addison-Wesley. E-kirja. Luettu 24.3.2025.

Martin, R. C. 2008. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall. E-kirja. Luettu 31.3.2025.

Microsoft. 2024a. Bicep Documentation – Infrastructure as Code. Luettavissa: https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/. Luettu 30.3.2025.

Microsoft. 2024b. Infrastructure as Code (IaC). What is Infrastructure as Code (IaC)? Luettavissa: https://learn.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code. Luettu 20.5.2025.

Microsoft. 2025a. Azure Logic Apps documentation. Luettavissa: https://learn.microsoft.com/en-us/azure/logic-apps/. Luettu 24.3.2025.

Microsoft. 2025b. Integration Services on Azure. Luettavissa: https://learn.microsoft.com/en-us/azure/architecture/example-scenario/integration/enterprise-integration-modern. Luettu 24.3.2025.

Microsoft. 2025c. Monitoring and diagnostics guidance. Luettavissa: https://learn.microsoft.com/en-us/azure/architecture/best-practices/monitoring. Luettu 24.3.2025.

Mullen, A. 29.11.2024. 4 ways to use team-based learning in the workplace. Thrive Learning. Blogi. Luettavissa: https://www.thrivelearning.com/blog-news/4-ways-to-use-team-based-learning-in-the-workplace. Luettu 13.5.2025.

Richardson, L. & Amundsen, M. 2013. RESTful Web APIs. O'Reilly Media. E-kirja. Luettu 27.4.2025.

Tynjälä, P., Välimaa, J. & Sarja, A. 2003. Pedagogical perspectives on the relation between higher education and working life. Kluwer Academic Publishers. E-kirja. Luettu 7.4.2025.

W3C. 2024. XML Schema Validation. Luettavissa: https://www.w3.org/XML/Schema. Luettu 24.3.2025.