Tampereen ammattikorkeakoulu

# Web API –rajapinnan rakentaminen Microsoft-ympäristössä

Toni Raevaara

OPINNÄYTETYÖ
Lokakuu 2020

Tieto- ja viestintätekniikan koulutus
Ohjelmistotekniikka

**TIIVISTELMÄ**

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikan koulutus
Ohjelmistotekniikka

RAEVAARA, TONI
Web API –rajapinnan rakentaminen Microsoft-ympäristössä

Opinnäytetyö 36 sivua, joista liitteitä 0 sivua
Lokakuu 2020

---

Opinnäytetyössä tutustuttiin Azure-pilvipalvelualustan tarjoamiin palveluihin sekä Web API –rajapinnan kehittämiseen, ylläpitoon sekä testaukseen Microsoftin .Net -ohjelmointiympäristössä. Käytännön tarkoituksena oli jatkokehittää CSI API – rajapinnasta käyttövalmis kokonaisuus, jonka jatkokehittäminen on mahdollisimman yksinkertaista. Opinnäytetyön toimeksiantajana oli CSI Helsinki Oy.

Kehitettävän Web API —rajapinnan tarkoituksena on toteuttaa CSI Helsinki Oy:n kehittämien CSI Mobile sekä CSI MyDesk –sovellusten tarvitsema taustajärjestelmä nykyaikaisemmalla teknologialla. Tämän lisäksi rajapinnan kautta on tarkoitus mahdollistaa kolmannen osapuolen integraatioiden nopea ja helppo toteuttaminen.

Teoria osuuteen kerättiin tietoja Microsoftin luomista dokumentaatioista sekä muista Internet lähteistä. Käytännön osuudessa esiteltiin CSI API –rajapinnan arkkitehtuuria sekä sen eri osioiden toiminnallisuutta.

---

Asiasanat: Web API –rajapinta, Azure, Microsoft, .Net

**ABSTRACT**

Tampere University of Applied Sciences
ICT Engineering
Software engineering

RAEVAARA, TONI
Developing Web API -interface in the Microsoft Ecosystem
Title of Thesis2

Bachelor's thesis 36 pages, appendices 0 pages
Month 2019

---

This thesis researched the Azure-cloud platform, the services it provides and development, maintenance and testing of Web API-interface within the Microsoft ecosystem. The practical partition of this thesis consists of developing the CSI API -interface from its current rough form to usable product, which can be easily developed further. The commissioner of this thesis is CSI Helsinki Oy.

The objective of the CSI API –interface is to implement a new backend service with modern technologies for CSI MyDesk and CSI Mobile, which are part of the ERP ecosystem developed by CSI Helsinki Oy. In addition, CSI API –interface must allow easy and fast development of third-party integrations.

Sources for the theoretic part were gathered mostly from documentation created by Microsoft and from other Internet sources. The architecture of the CSI API –interface was presented in the practical part of the thesis alongside with the functionality of the different components of the interface.

Key words: Web API –interface, Azure, Microsoft, .Net

**SISÄLLYS**

## LYHENTEET JA TERMIT

| | |
|---|---|
| .Net framework | Microsoftin kehittämä ohjelmointialusta |
| API | Ohjelmointirajapinta (Application Programming Interface), jonka kautta voidaan suorittaa toimintoja ulkopuolisesta järjestelmästä. |
| C# | Microsoftin kehittämä ohjelmointikieli .Net-alustalle |
| CSI Lawyer | CSI Helsinki Oy:n kehittämä lakitoimistojen toiminnanohjausjärjestelmä |
| CSI Mobile | CSI Helsinki Oy:n kehittämä mobiilisovellus, joka toimii CSI Lawyer -ohjelmiston rinnalla |
| CSI MyDesk | CSI Helsinki Oy:n kehittämä työpöytäsovellus, joka toimii CSI Lawyer -ohjelmiston rinnalla |
| JSON | JavaScript Object Notation, kevyt tekstipohjainen tiedonsiirtorakenne |
| Azure AD | Microsoft® Azure –alustan nimipalvelu |

# 1 JOHDANTO

CSI Helsinki Oy:llä on lakitoimistoille suunnattu CSI Lawyer –toiminnanohjausjärjestelmä sekä siihen liittyvät CSI Mobile ja CSI MyDesk –ohjelmat. CSI Mobile –sovellus on Android ja iOS –alustoille julkaistu mobiilisovellus, jonka tarkoituksena on mahdollistaa juristeille helppo toimenpiteiden kirjaus riippumatta sijainnista. CSI MyDesk –sovellus on työpöytäsovellus, jonka tarkoituksena on nopeuttaa juristin työskentelyä. Näistä syntyvä kokonaisuus mahdollistaa asiakkaiden, toimeksiantojen, toimenpiteiden ja kulujen hallinnointi sekä laskuttaminen. Tämän projektin tavoitteena on tuottaa monipuolisempi ja tehokkaampi palvelu asiakkaille sekä mahdollistaa kolmannen osapuolen integraatioiden helpompi kehittäminen CSI APIn kautta. Tämän lisäksi projektissa on tarkoituksena tutustua Microsoft Azure –alustaan ja sen tarjoamiin palveluihin projektiin soveltuvilta osilta.

## 1.1 Projektin taustat

CSI APIa lähdettiin toteuttamaan, koska CSI Mobile ja CSI MyDesk –sovellukset tarvitsevat nykyaikaisemman ja helpommin jatkokehitettävän taustajärjestelmän. Vanha järjestelmä on toimiva, mutta lisäkehitys on työlästä ja järjestelmä on toteutettu nimenomaan CSI Mobilea varten, joten se ei taivu sujuvasti kolmannen osapuolen järjestelmiä varten. Tämän takia todettiin, että on helpompi aloittaa uuden järjestelmän toteuttaminen puhtaalta pöydältä, jolloin siihen saadaan alusta asti rakennettua selkeä liitos Azure AD –nimipalveluun ja sen rakenne voidaan tehdä yleiskäyttöiseksi.

## 1.2 Lähtötilanne

CSI APIn toteuttaminen oli aloitettu jo ennen, kuin pääsin itse osallistumaan projektiin, ja alustavia versioita muutamista keskeisistä toiminnoista oli jo toteutettu. APIin oli toteutettuna tunnistautuminen Azure AD –nimipalvelun kautta sekä tie-

tokannan valinta nettisivulla rekisteröityjen kantojen joukosta. Tämän lisäksi toteutettuna oli API.V1 ja API.Implementation –rajapintojen välinen kutsujen uudelleenohjaus sekä virheiden kirjaaminen. Nämä olivat kuitenkin vielä varsin raakileita, ja niihin on tulossa muutoksia, jotta jatkokehitys olisi mahdollisimman nopeaa ja helppoa.

APIsta puuttui vielä useita toiminnallisuuksia, jotka tarvitaan APIn saattamiseksi käyttövalmiksi CSI Helsinki Oy:n omia sovelluksia varten. Tämän lisäksi tarvitaan merkittävä määrä toimintoja muita integraatioita sekä jatkokehitystä varten. Myös integraatiotestausta varten on tehty karkea runko, mutta sekin tarvitsee jalostamista, jotta lisätestien tekeminen olisi mahdollisimman nopeaa ja selkeää. Testauksesta puuttuvat vielä kokonaan varsinaiset yksikkötestit, joiden avulla voidaan varmistaa yksittäisten metodien toiminta.

Näiden lisäksi APIssa on parannettavaa reitityksen suhteen, sillä kutsujen päätepisteet eivät seuraa yleisiä REST APIen käytäntöjä. APIssa on myös muutamia muita ominaisuuksia, jotka sotivat REST APIn normaaleja käytäntöjä vastaan ja vaativat korjausta. APIin on myös suunnitteilla toiminnallisuus, jolla pyritään minimoimaan tarvittavien kutsujen määrää mahdollistamalla lisätietojen pyytäminen kutsuissa.

## 1.3 Tavoite

Opinnäytetyön tarkoituksena on toteuttaa API-rajapinta, jonka kautta on mahdollista toteuttaa CSI Mobile sekä CSI MyDesk –sovellusten nykyinen toiminnallisuus. Lisäksi APIn kautta on tarkoitus mahdollistaa integraatiot muihin ohjelmistoihin sekä CSI Helsinki Oy:n, että kolmannen osapuolen kehittäminä.

### 1.3.1 Rakenteen vaatimukset

CSI Helsinki Oy:llä on useita asiakasyrityksiä, joilla on omat CSI Lawyer –ohjelmiston tietokannat. CSI APIn pitää siis rakenteellisesti mahdollistaa useiden eri tietokantojen käsittely. Lisäksi tietokannat voivat sijaita eri palvelimilla, jotka ovat

palomuurien takana. Tästä johtuen julkisen API.V1-rajapinnan pitää olla yhdellä julkisella palvelimella ja sieltä kutsut uudelleenohjataan asiakaskohtaiseen APIin, jolla on pääsy tietokantaan.

### 1.3.2 Toiminnalliset vaatimukset

CSI APIn tarkoituksena on alustavasti korvata CSI Mobile Service –palvelu, jonka kautta on toteutettu CSI Mobile ja CSI MyDesk –sovellusten tietokantayhteys. Näitä varten tulee mahdollistaa toimenpiteiden tekeminen ja muokkaus sekä toimenpidetyyppien ja osaamisalueiden käsittely APIn kautta. Lisäksi sovellukset vaativat dataa, jonka avulla ne voivat esittää kuvaajia käyttäjän kirjauksista menneiden viikkojen ajalta. Toimenpiteiden kirjaus taas asettaa useita erilaisia vaatimuksia, kuten kirjauskausien sekä toimituskieltojen käsittelemisen.

APIn pitää mahdollistaa tunnistautuminen ja käyttäjienhallinta Azure AD –tunnuksilla ilman erillistä käyttäjän rekisteröitymistä CSI Lawyer –sovelluksessa. Tämän lisäksi CSI APIn julkisen rajapinnan tulee toimia Azure-pilvipalvelussa, jolla varmistetaan hyvät yhteydet APIin sekä tarvittaessa palvelimen kapasiteetin kasvattaminen. Asiakkaiden toimialasta johtuen käsitellään järjestelmässä usein GDPR:n alaisia tietoja, joten tietoturvan tulee olla erinomainen.

# 2 Microsoft Azure –alusta

Suurimpana syynä Microsoft Azure –alustan valintaan olivat käytössämme jo olevat useat Microsoftin Azure –alustaan liittyvät palvelut, minkä lisäksi Azure AD tunnistautuminen on toteutettu jo muihin ohjelmiimme. Pystymme tämän johdosta hyödyntämään osaamistamme Microsoft Azure –ympäristöstä, jonka lisäksi tuotteidemme käyttökokemus pysyy yhdenmukaisena.

## 2.1 Alustan tarjoamat palvelut

Azure tarjoaa nykyisin yli 600 erilaista palvelua, sisältäen virtuaalipalvelimia, käyttäjien hallintaa ja tunnistautumista, tiedonhallintaa, tietokantoja ja lukuisia muita palveluita. Alusta mahdollistaa useiden eri ohjelmointikielten sekä Microsoftin ja kolmannen osapuolen kehittämien ohjelmistokehysten käyttämisen.

### 2.1.1 Azure AD –nimipalvelu

Azure AD (Azure Active Directory), on Microsoftin kehittämä pilvipohjainen nimipalvelu, joka kehitettiin alkujaan Office 365 –kirjautumista varten. Ilman Azure AD:ta olisi jokaiseen Office 365 –pakettiin kuuluvaan ohjelmaan pitänyt kirjautua erikseen omilla käyttäjätunnuksilla. Azure AD mahdollistaa kuitenkin yhden käyttäjätunnuksen käyttämisen kaikissa Microsoftin tuotteissa sekä tuhansissa kolmannen osapuolen palveluissa. Tämän lisäksi palvelu kerää tietoa käyttäjien kirjautumisista, muun muassa laitteen, sijainnin sekä sen, mihin aikaan kirjautuminen on tapahtunut. Tällä mahdollistetaan tietohallinnolle tietoturvallisuuden parantaminen sekä mahdollisten tietomurtojen havaitseminen. Esimerkiksi jos Helsingissä työskentelevän henkilön tunnuksilla kirjaudutaan ulkomailta, voidaan helposti päätellä, että tunnukset ovat vuotaneet. (What is Azure Active Directory? 2020.)

Azure AD –palvelu mahdollistaa myös kolmannen osapuolen ohjelmistoihin kirjautumisen. Tätä varten löytyy valmiita ohjelmointikirjastoja useilla eri ohjelmointikielillä ja useisiin eri ohjelmistokehyksiin. Tässä erittäin suurena etuna on mahdollisuus käyttää suuren yrityksen kehittämää ja hallinnoimaa tietoturvallista tunnistautumispalvelua, jolloin sitä ei tarvitse rakentaa itse. Lisäksi Azure AD mahdollistaa monivaiheisen tunnistautumisen, joka varmistaa, ettei pelkkä käyttäjätunnusten vuotaminen päästä ulkopuolisia kirjautumaan palveluun. (What is Azure Active Directory? 2020.)

### 2.1.2 Azure Sovellukset

Azure mahdollistaa myös kolmannen osapuolien kehittämien sovellusten rekisteröinnin Azure AD –nimipalveluun, jonka avulla käyttöoikeuksien hallinta voidaan keskittää yhteen paikkaan. Tämän avulla mahdollistetaan myös sovellusten pääsy tietoturvallisesti Azure Key Vault –palveluun tallennettuihin tietoihin. (How and why applications are added to Azure AD? 2019.)

### 2.1.3 Azure Key Vault -avainsäilö

Azure Key Vault on palvelu, jonne voidaan tallentaa salausavaimia, kirjautumistunnuksia ja muita sovellusten tarvitsemia sertifikaatteja. Avaimiin pääsevät käsiksi vain sellaiset sovellukset, jotka ovat tunnistautuneet Azureen ja joille on annettu käyttöoikeus. Azure Key Vault mahdollistaa myös avainten käytön seurannan, jonka avulla voidaan havaita väärinkäytökset ja poistaa väärinkäytetyt avaimet nopeasti. (Azure Key Vault basic concepts, 2019.)

### 2.1.4 Azure SQL –pilvipalvelu

Azure SQL pilvipalvelu tarjoaa SQL Server –tietokantoja erinäköisissä ympäristöissä. Palvelusta voidaan ottaa käyttöön pelkkä tietokanta Azure SQL Database –palvelun kautta, mikäli ympäristöä ei ole tarvetta hallinnoida. Tässä palvelussa Microsoft hallinnoi tietokantaa ja sen päivityksiä, mutta palvelu ei tarjoa aivan

kaikkia laajemmissa palveluissa tarjolla olevia ominaisuuksia. Azure SQL Managed Instance –palvelu on hyvin samanlainen, mutta mahdollistaa laajemman valikoiman ominaisuuksia. Esimerkkinä eroavista ominaisuuksista on se, että Azure SQL Managed Instance –palvelu mahdollistaa tietokannan manuaalisen varmuuskopion luonnin. Automaattinen varmuuskopioiden luonti löytyy kuitenkin kummastakin. (Features comparison: Azure SQL…, 2020)

Viimeisenä vaihtoehtona on SQL Server on Azure VMs –palvelu, täysi virtuaalikone, jossa SQL tietokanta sijaitsee. Tämä mahdollistaa käyttäjälle täyden hallinnan sekä tietokantaan että sen toimintaympäristöön. (What is Azure SQL?, 2020.)

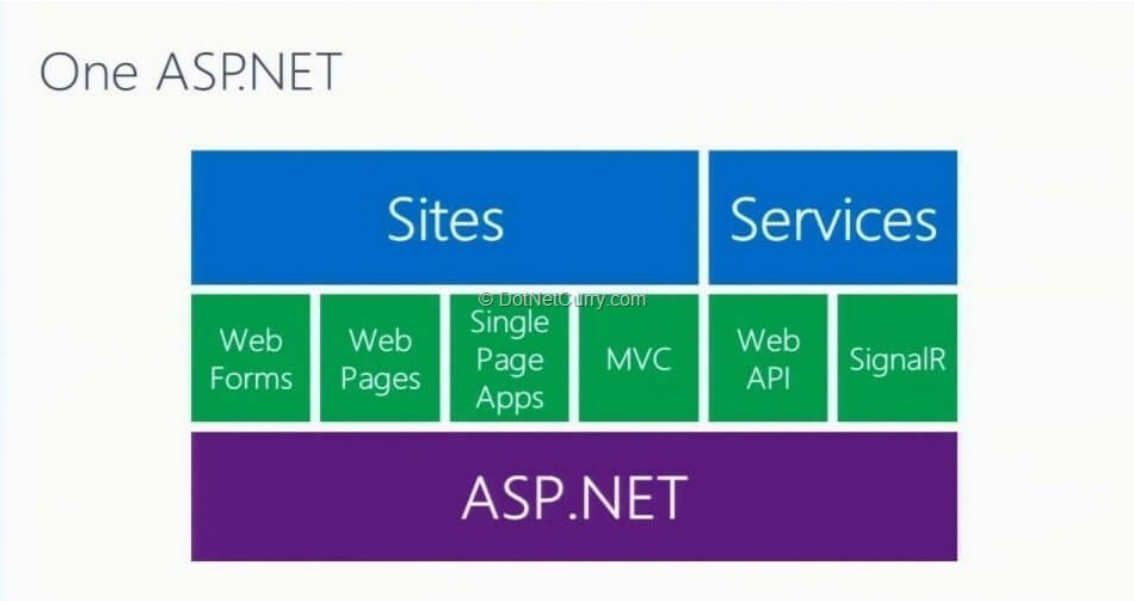
## 2.2 Kilpailijat

Microsoft Azuren suurimmat kilpailijat ovat Amazon Web Services (AWS) sekä Google Cloud Platform. Käytännössä hyvin pitkälti samat asiat on mahdollista toteuttaa palveluntarjoajasta riippumatta. AWS tarjoaa hieman muita laajemman valikoiman erilaisia palveluita ja Google Cloud parhaan koneoppimis-alustan sekä eniten avoimen lähdekoodin ratkaisuja. (AWS vs Azure vs Google Cloud… 2020)

# 3 ASP.NET –Ympäristö

## 3.1 Yleiskatsaus

ASP.Net –ympäristö on Microsoftin kehittämä web-sivujen sekä palveluiden kehittämiseen tarkoitettu ohjelmointiympäristö. Ympäristön ensimmäinen versio on julkaistu vuonna 2002, joten se on nykyisin hyvin testattu ja tarjoaa käytännössä kaikki mahdollisuudet, mitä web-ympäristössä voi tarvita.




Kuva 1. ASP.NET –kokonaisuus (The History of ASP.NET, 2019)

ASP.NET –ympäristö jakautuu karkeasti kahteen osaan Kuva 1. ASP.NET –kokonaisuus (The History of ASP.NET, 2019) mukaisesti. Ensimmäinen osa on web-sivustot, joiden toteutukseen on neljä erilaista tapaa. Tämän lisäksi ASP.NET –ympäristö mahdollistaa web-palveluiden kehittämisen. ASP.NET –alustan lisäksi .NET-alustalla on mahdollista kehittää eri teknologioilla sekä työpöytä- että mobiilisovelluksia. (The History of ASP.NET, 2019)

## 3.2 Ohjelmointikielet .Net-ympäristössä

.Net ympäristössä voidaan käyttää useita eri ohjelmointikieliä, joista käytetyin on C# (C sharp). Tämän lisäksi käytettävissä ovat F# sekä Visual Basic –ohjelmointikielet. (.Net Programming Languages, n.d.)

### 3.2.1 C# verrattuna C++ -kieleen

Nämä kielet on alkujaan kehitetty C-ohjelmointikielen pohjalta ja tästä johtuen niiden syntaksi on hyvin vastaava. C on hyvin matalan tason ohjelmointikieli ja C++ on keskitason ohjelmointikieli. C# taas vastaavasti on korkean tason kieli, joka tekee automaattisesti osan asioista, joita alemman tason kielissä pitää tehdä manuaalisesti. Yksi esimerkki tällaisesta on muistin käsittely. C++ -kielessä muistinkäsittely tulee hoitaa manuaalisesti, mutta C#-kieli hallinnoi muistin automaattisesti. Automaattisuus helpottaa huomattavasti ohjelmointia, mutta vähentää jonkin verran suorituskykyä. Usein suorituskyvyn menetys on merkityksetön, mutta esimerkiksi suuren mittakaavan laskennassa tällä on merkitystä, joten tällaisissa sovelluksissa suositaankin matalan tason ohjelmointikieliä. (Understanding the Differences… 2018)

Suurin ohjelmointiin vaikuttava ero vaihdettaessa C++:sta C#:iin on se, että C# mahdollistaa ohjelmakokonaisuuden rakentamisen useista komponenteista, jotka eivät tiedä toisistaan muuta kuin julkisen rajapinnan. Toisinpäin vaihdettaessa haasteita on huomattavasti enemmän, johtuen erityisesti muistin hallinnoinnista sekä osoittimista. (Understanding the Differences… 2018)

## 3.3 .Net-ohjelmistokehykset

.Net on Microsoftin kehittämä ohjelmistokehys, joka perustuu avoimeen lähdekoodiin ja on käytettävissä ilmaiseksi. .NET kokonaisuutena kattaa useita eri ohjelmointikehyksiä, merkittävimpänä .NET Framework ja .NET Core, jotka ovat lähitulevaisuudessa yhdistymässä .NET 5.0:ksi. Eri kehysten suurimmat erot ovat

niiden yhteensopivuudessa eri alustoiden kanssa. Erittäin suuri etu .NET ympäristössä on NuGet -paketinhallintajärjestelmä, jossa on yli 90000 erilaista .Net ympäristöön kehitettyä ohjelmointikirjastoa. (What is .NET? n.d.)

### 3.3.1 .Net Framework

.Net Framework mahdollistaa nettisivujen, palveluiden ja sovellusten toteuttamisen Windows-alustalle. .Net Framework koostuu kahdesta osasta, ensimmäinen on Common Language Runtime (CLR), joka hoitaa sovellusten suorittamisen ja toinen .NET Framework Class Library, joka tarjoaa laajan valikoiman ohjelmistokirjastoja. .NET Framework on yksi vaihtoehdoista, joka toteuttaa .Net Standard –rajapinnan. .NET Standard on kaikkien .NET-alustaan kuuluvien ohjelmistokehysten välinen rajapinta. .NET Standard määrittelee, mitkä kaikki mahdollisuudet ohjelmistokehyksistä pitää löytyä. Tämän avulla on helppo käyttää yhdessä ohjelmistokokonaisuudessa useampia .NET-ympäristöön kuuluvia kehyksiä. (What is .NET Framework? n.d.)

### 3.3.2 Entity Framework

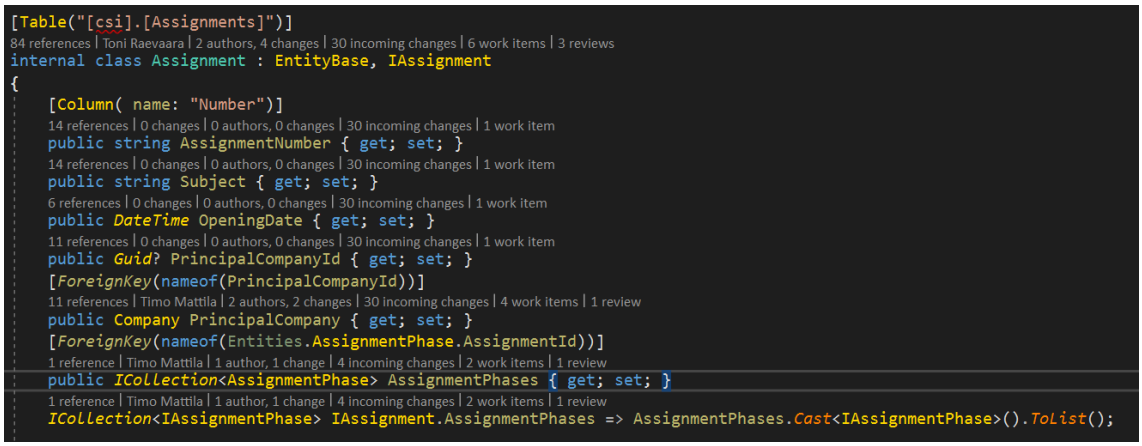
Entity Framework on Microsoftin tukema avoimen lähdekoodin ORM (Object-Relational Mapping) ohjelmistokehys .NET-sovelluksiin. Kehys mahdollistaa tietokannan käsittelyn olioiden kautta ilman, että kehittäjän tarvitsee keskittyä tietokannan varsinaiseen rakenteeseen. Entity Framework käsittelee sisäisesti tietokantakutsujen rakentamisen sekä välimuistin käytön. Kehyksen toiminta perustuu tietokantakontekstiin, POCO-luokkiin (Plain old CLR object) ja LINQ-to-Entities hakuihin. (What is Entity Framework? n.d.)

```
11 references | Toni Raevaara | 3 authors, 11 changes | 30 incoming changes | 17 work items | 14 reviews
internal class CsiDataContext : CodeFirstDbContext
{
    constructors
    1 reference | 0 changes | 0 authors, 0 changes | 30 incoming changes | 1 work item
    protected override void OnModelCreating(DbModelBuilder modelBuilder)...
    12 references | 0 changes | 0 authors, 0 changes | 30 incoming changes | 1 work item
    public virtual DbSet<Entities.Assignment> Assignments { get; set; }
    0 references | Toni Raevaara | 1 author, 1 change | 23 incoming changes | 3 work items | 1 review
    public virtual DbSet<Entities.AssignmentPhase> AssignmentPhases { get; set; }
    1 reference | 0 changes | 0 authors, 0 changes | 30 incoming changes | 1 work item
    public virtual DbSet<Entities.AssignmentTransactionTemplate> AssignmentTransactionTemplates { get; set; }
    0 references | 0 changes | 0 authors, 0 changes | 30 incoming changes | 1 work item
    public virtual DbSet<Entities.AssignmentType> AssignmentTypes { get; set; }
```

Kuva 2. Tietokanta kontekstin rakenne

Tietokantakonteksi rakentuu DbSet-kokoelmista, jotka vastaavat karkeasti tietokannassa olevia tauluja tai näkymiä. Jokaiselle DbSet-kokoelmalle kerrotaan, mitä entiteettejä kokoelma sisältää, esimerkki Kuva 2.

Luokat voivat sisältää sekä yksinkertaisia arvoja, kuten uniikkeja tunnisteita (Guid) ja tekstiä (string). Ne löytyvät entiteettiä vastaavasta taulusta, joka kerrotaan Table-attribuutilla. Mikäli entiteetin kenttien nimet vastaavat tietokannassa olevia sarakkeita, Entity Framework osaa automaattisesti yhdistää ne, muuten tulee sarakkeen nimi ilmoittaa Column-attribuutilla. POCO–entiteetit (Plain Old CLR Object) voivat sisältää lisäksi relaatioita muihin tauluihin.




Kuva 3. POCO luokka

Kuva 3 on esimerkki POCO-luokasta, joka sisältää yksinkertaisia ja komplekseja entiteettejä.

```
DbQuery<Assignment> dbQuery = CsiContext.Assignments;
dbQuery = dbQuery.Include("PrincipalCompany");
var assignments =  dbQuery.Where(assignment =>
    assignment.PrincipalCompany.CountryId == COUNTRY_ID_FINLAND).ToList();
```

Kuva 4. Tietokantahaku

Tietokantahaut Entity Frameworkin kautta tehdään LINQ-to-entities kirjaston avulla. Kirjasto rakentaa entiteetin perusteella tehdystä hausta tietokantaa vastaavan haun. Kuva 4 on esimerkkihaku tietokannasta, jossa ladataan Assignment-olioita. Samalla ladataan niihin liittyvät PrincipalCompany-oliot käyttämällä include-toiminnallisuutta. Hakua suodatetaan PrincipalCompanyn maatunnisteen mukaan.

## 3.4 ASP.NET WEB API

API eli Application Programming Interface, on rajapinta, jonka kautta voidaan kommunikoida eri ohjelmien, käyttöjärjestelmien tai palveluiden välillä. Web API on rajapinta, jonka kanssa voidaan kommunikoida HTTP protokollan mukaisesti.

ASP.NET Web API on yksi osa Microsoftin kehittämää avoimen lähdekoodin ASP.Net –ohjelmistokehystä. Kehys mahdollistaa sekä selainsivujen toteuttamisen että web-palveluiden kehittämisen. (What is Web API? n.d.)

### 3.4.1 Tiedonsiirto formaatit

Olennainen osa WEB API –palvelua on tiedonsiirto APIn kutsujalta APIin sekä toisinpäin. Tätä varten on olemassa useita erilaisia tietorakenteita, joista jokaisella on omat etunsa sekä haittapuolensa. Nykyisin yleisimmät käytössä olevat rakenteet ovat JSON (JavaScript Object Notation) sekä XML (EXtensible Markup Language).

```
{
    "subject": "Ohjelmoinnin Alkeet",
    "courseCode": "kooodaus-101",
    "id": "77ac6d84-4d07-4e2e-a2dc-00476630723f"
},
<Class>
    <Id>77ac6d84-4d07-4e2e-a2dc-00476630723f</Id>
    <Subject>Ohjelmoinnin Alkeet</Subject>
    <CourseCode>kooodaus-101</CourseCode>
</Class>
```

Kuva 5: JSON ja XML –vastaukset

Kuva 5: JSON ja XML –vastau on yksinkertainen esimerkki JSON (ylempi) sekä XML (alempi) –rakenteista. Siihen on kuvattuna Class-tietoluokka, joka sisältää aiheen, kurssikoodin sekä objektin tunnisteen. JSON rakenne koostuu avain/arvo

–pareista ja XML vastaavasti useista elementeistä. Molemmat rakenteet ovat varsin helposti luettavia ja niiden rakenne on varsin kevyt, mutta molemmissa JSON on hieman parempi vaihtoehto. ASP.NET WEB API –ohjelmistokehys mahdollistaa molempien vaihtoehtojen käyttämisen automaattisesti.

## 3.5 API tyylivaihtoehdot

Web APIn toteuttamiseen on olemassa useita erilaisia tyylejä. Nykyisin yleisimmät tyylit ovat REST ja merkittävästi viime aikoina kasvanut GraphQL. Molemmilla voidaan mahdollistaa CRUD-toiminnot, mutta niiden toiminnallisuus eroaa kutsujen sekä vastausten rakenteessa.

### 3.5.1 GraphQL

Pääsääntöisesti yhteyksien tiedonsiirtonopeus on niin suuri, että API kutsuissa suurin osa ajasta johtuu verkkoviiveestä. Tietomäärä ei siis merkittävästi vaikuta suoritusaikaan, mutta kutsujen suuri määrä hidastaa toimintaa merkittävästi. Tämän takia GraphQL:n suurin etu RESTiin nähden on se, että GraphQL mahdollistaa kyselyiden rakentamisen kutsukohtaisesti. APIn tarjoaja määrittelee mitä kaikkia tietoja on mahdollista hakea ja kuinka tiedot liittyvät toisiinsa. Kutsuja määrittää mitä tietoja vastaukseen halutaan. Lopputuloksena tietokannasta haetaan vain ne tiedot, jotka kutsuja oikeasti tarvitsee. Näin saadaan minimoitua sekä siirrettävän datan ja tarvittavien kutsujen määrä kutsujan ja APIn sekä APIn ja tietokannan välillä. (GraphQL vs. REST. 2017.)

Haittapuolena GraphQL:ssä on teknologian tuoreus. Ensimmäinen vakaa versio julkaistiin 2018 kesällä, joten kyseisestä teknologiasta ei CSI Helsinki Oy:ssä vielä ole kokemusta, mikä saattaa hidastaa tuotekehitystä merkittävästi.

### 3.5.2 REST

REST tyylinen API on kutsujan kannalta kankeampi, koska kutsuja ei yleensä voi määrittää, mitä tietoja hän tarvitsee. APIn tarjoaja on määritellyt tarjolla olevat kutsut ja niiden palauttamat tiedot. Tämän seurauksena kutsuja ei välttämättä saa yhdellä kutsulla kaikkia tarvittavia tietoja, joten kutsujen määrä moninkertaistuu tai vastauksessa on ylimääräistä dataa. Ongelmaa voidaan minimoida tekemällä uusia kutsuja, jotka mahdollistavat tarvittavien tietojen haun yhdellä kutsulla. Selkeästi määritettyjen kutsujen etuna on se, että tietokantahakuja voidaan nopeuttaa välimuistilla, koska haut ovat aina samanlaisia.

REST APIin voidaan toteuttaa lisätietojen hakumahdollisuus, jonka avulla voidaan mahdollistaa GraphQL-tyylinen kutsu, ja kutsuja voi määritellä kutsuun haluamiaan lisätietoja. Tämä ei mahdollista yhtä kattavaa toiminnallisuutta kuin GraphQL:n kutsut, mutta sallii kuitenkin pienimuotoisen kutsujen kustomoinnin. (GraphQL vs. REST. 2017.)

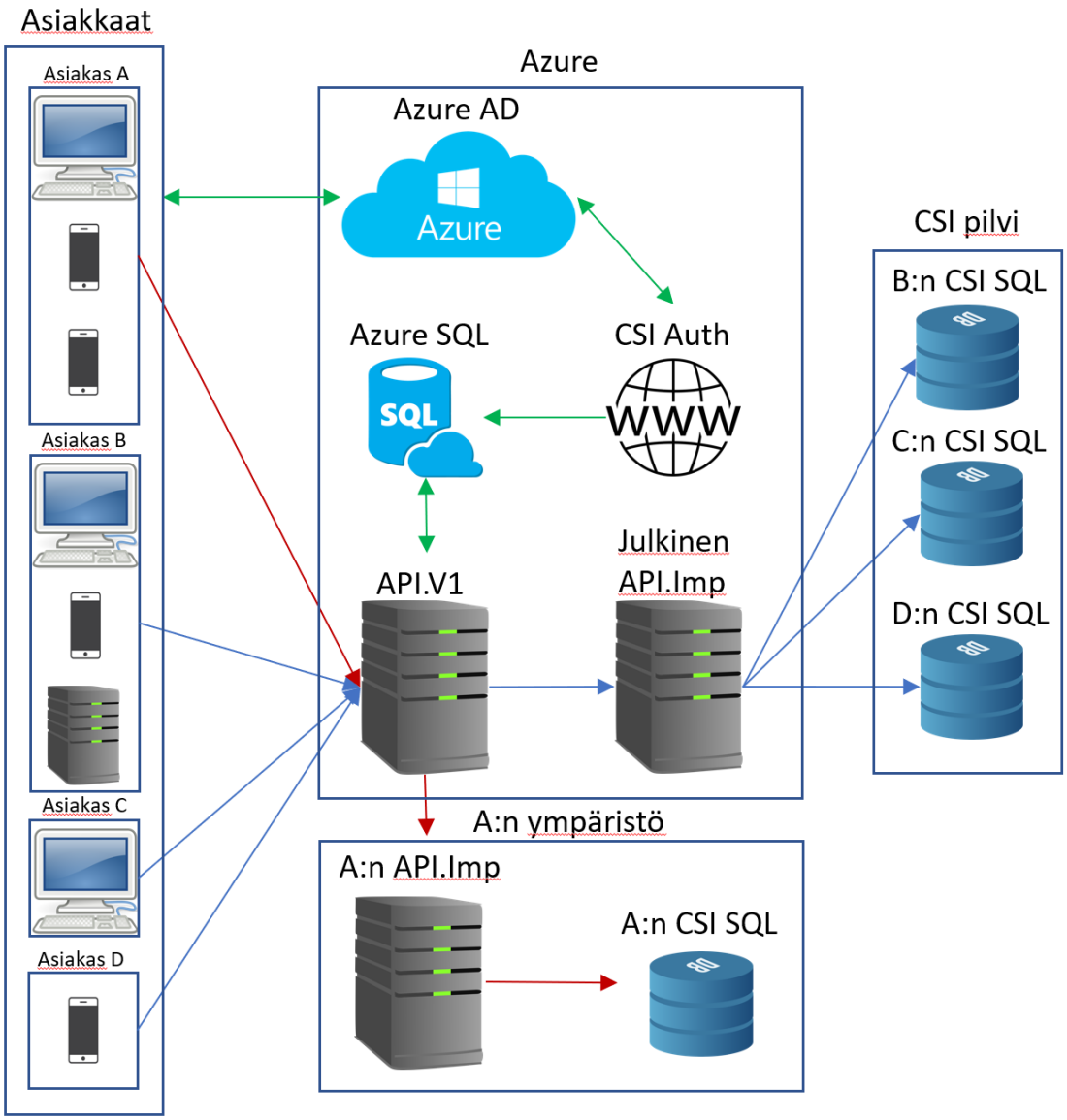
# 4 Palvelinpään kehitys

## 4.1 CSI API kokonaisuutena

CSI API koostuu kahdesta lähes identtisestä rajapinnasta, jotka ovat API.V1 ja API.Implementation. API.V1 –rajapinta tunnistaa käyttäjän, valitsee tietokannan sekä ohjaa kutsun eteenpäin joko julkiseen tai asiakkaan ympäristössä olevaan API.Implementation-rajapintaan. API.Implementation sisältää varsinaisen toimintalogiikan sekä asiakkaiden tietokantojen käsittelyn.

### 4.1.1 Toiminnallinen kokonaisuus

CSI APIn toiminnallinen kokonaisuus koostuu kahdesta REST-rajapinnasta sekä auth.csihelsinki.fi sivustosta. Näiden lisäksi käytössä on Azure AD –nimipalvelu, johon käyttäjät kirjautuvat ja johon sovellukset rekisteröidään sekä Azure SQL –tietokanta, jota käytetään asiakkaiden tietokantayhteyksien ja mahdollisten omien API.Implementation päätepisteiden tallentamiseen. Järjestelmä tarvitsee myös asiakkaiden CSI Lawyer –sovelluksen tietokannan sekä jonkin APIa kutsuvan sovelluksen.




Kuva 6: Toiminnallinen kokonaisuus

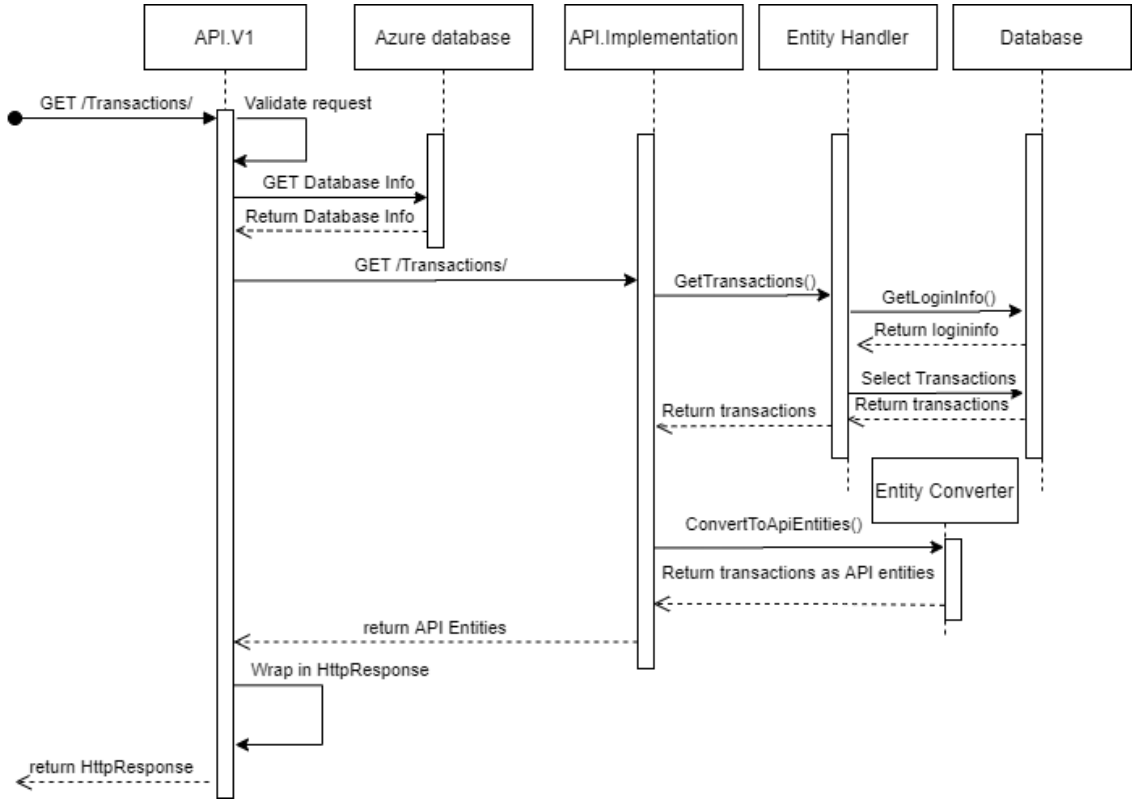
Kuva 6: Toiminnallinen kokonaisuus on esitettynä CSI API, siihen liittyvät muut palvelut sekä asiakkaiden kutsujen kulku. Vasemmassa reunassa on CSI Helsinki Oy:n asiakkaiden käyttäjiä, ylhäällä keskellä Azure-pilvipalvelussa olevat toiminnallisuudet ja oikealla reunassa CSI Helsinki Oy:n pilvipalvelussa olevat asiakkaiden CSI Lawyer –sovellusten tietokannat. Keskellä alhaalla on asiakkaan A omassa ympäristössä oleva yksityinen API.Implementation –rajapinta ja heidän CSI Lawyer –sovelluksensa tietokanta.

Vihreillä nuolilla on kuvattu kaikkien asiakkaiden käyttämät yhteydet eri palveluiden välillä. Siniset nuolet kuvaavat yleistä API.Implementation versiota käyttävien asiakkaiden yhteydet ja punaiset nuolet asiakkaan A käyttäjien yhteydet, jotka

kulkevat heidän omassa ympäristössään olevan API.Implementation version kautta.

### 4.1.2 Kutsun kulku CSI APIssa

CSI API koostuu kahdesta erillisestä API-rajapinnasta, joten kokonaisuuden toiminta poikkeaa hieman yleisestä APIen toiminnasta. Yleensä, kun kutsu vastaanotetaan, voidaan suoraan rajapintaluokassa käsitellä tietokantaa ja palauttaa halutut tiedot tai tehdä tietokantaan muutoksia.



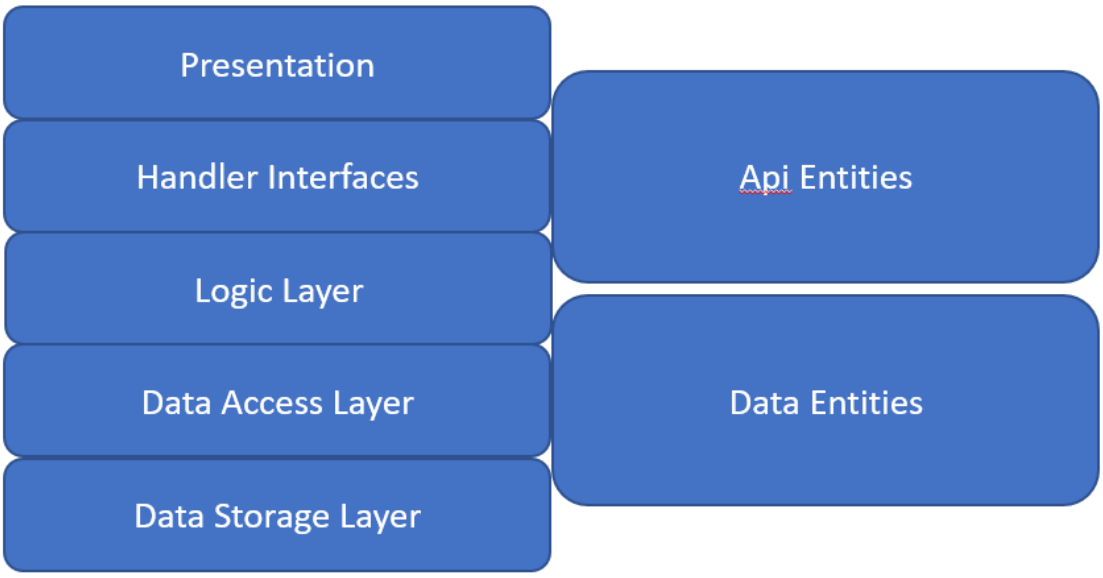

Kuva 7: Kutsun kulku CSI APIssa

Kuva 7: Kutsun kulku CSI APIssa on nähtävillä kutsujen kulku CSI APIn läpi, kun vastaanotetaan GET Transactions –kutsu, jolla haetaan tietokannassa olevat toimenpiteet. Ensin API.V1 tarkastaa kutsusta tunnisteen sekä kutsun tiedot, jonka jälkeen API.V1 hakee Azuren SQL -tietokannasta tietokannan sekä mahdollisesti API.Implementation –rajapinnan tiedot. Tämän jälkeen kutsu uudelleenohjataan API.Implementation –rajapintaan, jonka kautta käsitellään tietokantaa. Api.Imple-

mentationin sisällä on useita Entity Handler –luokkia, jotka vastaavat logiikkatason toiminnasta ja joiden kautta päästään tietokantaan käsiksi. Luokka kirjautuu tietokantaan ja hakee sen jälkeen pyydetyt tiedot ja muokkaa ne vastaamaan julkista luokkaa. Tämän jälkeen kutsu palautetaan API.V1:lle ja sen kautta alkuperäiselle kutsujalle.

### 4.1.3 APIen arkkitehtuuri

CSI APIn molemmat rajapinnat rakentuvat .Net WEB API 2 –ohjelmistokehyksen päälle ja niiden rakenne seuraa yleistä APIn eri tasojen jakoa, joka on Kuva 8. Jokainen erillinen taso on pyritty toteuttamaan siten, että taso tietää vain yhdestä alemmasta tasosta sekä näiden välillä jaetuista entiteettityypeistä. Tällä erottelulla pyritään mahdollistamaan se, että tulevaisuudessa voidaan tarvittaessa toteuttaa useampia toisistaan poikkeavia logiikkatasoja. Esimerkiksi eri CSI Lawyer -ohjelmaversioiden mukaan voitaisiin rakentaa erilliset logiikkatasot, mutta yhteinen rajapinta mahdollistaa kuitenkin saman Presentation-tason käyttämisen.




Kuva 8: APIen rakenteelliset tasot

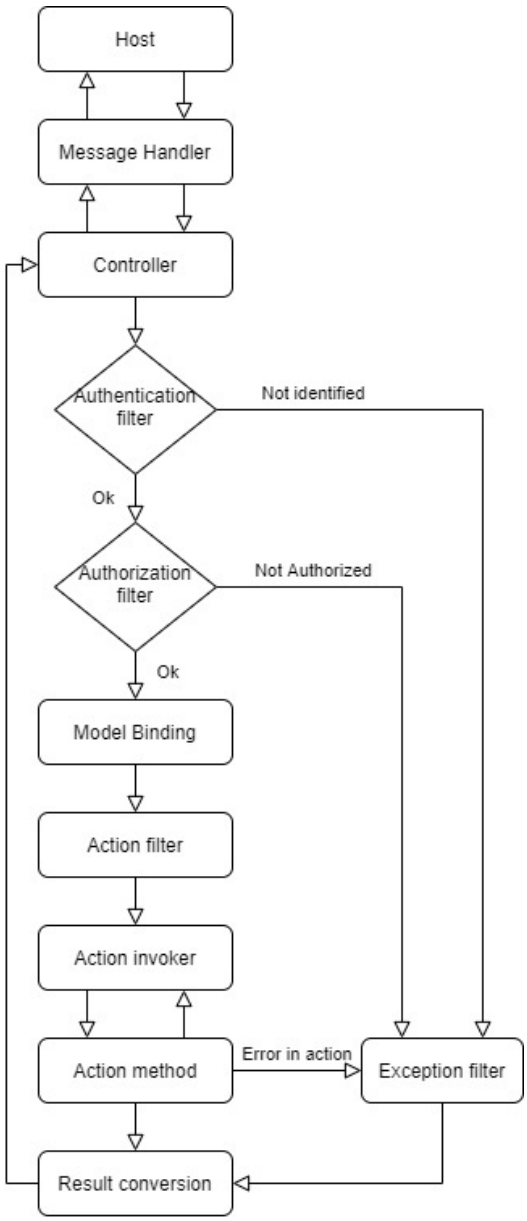
## 4.2 Presentation-taso ja API.Entities luokat

Presentation-taso sisältää varsinaisen julkisen rajapinnan. Siellä sijaitsevat Controller-luokat, joiden perusteella API-kutsujen päätepisteet määräytyvät. Presentation-taso on lisäksi tietoinen Api Entities –projektista, joka sisältää kaikki rajapintojen ulospäin näkyvät julkiset entiteettiluokat sekä Handler Interfaces -projektista, jonka kautta se pääsee käsittelemään logiikkatasoa.

### 4.2.1 Controller-luokat

Molemmissa API-rajapinnoissa on tällä hetkellä noin kaksikymmentä erilaista Controller-luokkaa, jotka vastaavat APIssa olevista kutsujen päätepisteistä. Jotta kutsujen uudelleenohjaus toimisi, on molemmissa rajapinnoissa oltava vastaavat luokat. Tätä varten jokainen Controller-luokka toteuttaa sitä vastaavan IController-luokan, jonka avulla luokkien funktiot saadaan pakotettua vastaamaan toisiaan.

APIn virheiden käsittelyä varten kaikkiin Controller-luokkiin on liitettyä virhesuodatin. Virhesuodattimen tarkoituksena on vastaanottaa kaikki virheet, jotka tapahtuvat API-kutsuja käsiteltäessä, ja muotoilla niistä virheviesti, jonka avulla APIn kutsujalle voidaan antaa selkeä käsitys siitä, mikä kyseisessä kutsussa on pielessä.




Kuva 9: Kutsun käsittely Controller-luokassa

Kuva 9: Kutsun käsittely Controller-luokassa on nähtävillä ohjelman kulku, kun APIin tulee jokin kutsu ulkopuolelta. .Net WEB API –kehyksen MessageHandler-luokka etsii reititysmäärityksistä kutsun osoitetta vastaavan Controller-luokan ja sen tunnistautumissuodattimen. Mikäli suodatin päästää kutsun läpi, haetaan seuraavaksi käyttöoikeudet tarkastava suodatin ja kutsutaan sitä. Tämän jälkeen kutsun osoitteen sekä parametrien perusteella etsitään oikea funktio. Funktio voi olla sidottu omaan suodattimeen, jossa voidaan esimerkiksi tarkastaa, että kutsussa olevat parametrit täyttävät vaatimukset. Mikäli tästäkin suodattimesta päästään läpi, voidaan suorittaa funktio. Kutsun suorittamisen jälkeen paluuarvo muotoillaan http-vastaukseksi ja palautetaan kutsujalle.

### 4.2.2 Reititys

Jotta APIn kutsut saadaan ohjattua oikeaan funktioon, .NET WEB API –kehykselle täytyy kertoa, kuinka kutsun osoite kohdistuu Controller-luokkiin sekä niiden funktioihin. Yleisin tapa hoitaa reititys on muodostaa kutsun osoite Controller-luokkien nimien mukaan niin, että Controller osa jätetään pois ja yleisimmät toiminnot, jotka ovat Taulukko 1. http-verbien toiminnallisuus. (Using HTTP Methods… n.d.) , ohjataan http-verbien mukaan. Erikoisempien ja vähemmän käytettyjen kutsujen osoitteessa on lisäksi toiminnolle annettu nimi.

Taulukko 1. http-verbien toiminnallisuus. (Using HTTP Methods… n.d.)

| **http verbi** | **CRUD** | **Selite** |
|---|---|---|
| POST | Create | Luodaan uusia entiteettejä |
| GET | Read | Haetaan olemassa olevia entiteettejä |
| PUT | Update/Replace | Korvataan olemassa oleva entiteetti tai luodaan uusi, mikäli ei ole olemassa |
| PATCH | Update/Modify | Muokataan olemassa olevaa entiteettiä |
| DELETE | Delete | Poistetaan olemassa oleva entiteetti |

### 4.2.3 APIn paluuarvot

Kutsujen paluuarvot seuraavat yleisiä http-paluukoodeja, joista yleisimmät ovat kuvattuna Taulukko 2. http paluuarvot. . Kun http-paluukoodeja käytetään yleisen käytännön mukaisesti, on rajapinnan käyttäjän helppo jo pelkän paluukoodin perusteella päätellä, mikä on ollut kutsun lopputulos.

Taulukko 2. http paluuarvot. (HTTP Status Codes n.d.)

| **Koodi** | **Syy** | **Käyttö** | **Selite** |
|---|---|---|---|
| 200 | Ok | Get | Toiminto suoritettu onnistuneesti |
| 201 | Created | Post / Put | Entiteetti luotu onnistuneesti |
| 204 | No Content | Put / Patch | Entiteetti päivitetty onnistuneesti |
| 400 | Bad Request | Kaikki | Kutsussa on jotain vikaa, joka kutsujan on korjattava itse |
| 401 | Unauthorized | Kaikki | Kutsusta puuttuu AAD tunniste |
| 403 | Forbidden | Kaikki | AAD tunniste löytyy, mutta nykyisellä käyttäjällä ei ole oikeuksia toimintoon |
| 404 | Not Found | Get / Put / Patch / Delete | Kutsun endpointia ei ole olemassa tai annetulla tunnisteella ei ole olemassa entiteettiä. |
| 405 | Method Not Allowed | Kaikki | Annettu http verbi ei vastaa löydettyä toimintoa |
| 500 | Internal server error | Kaikki | Palvelimen päässä tapahtunut jokin virhe, jota ei osattu käsitellä. |

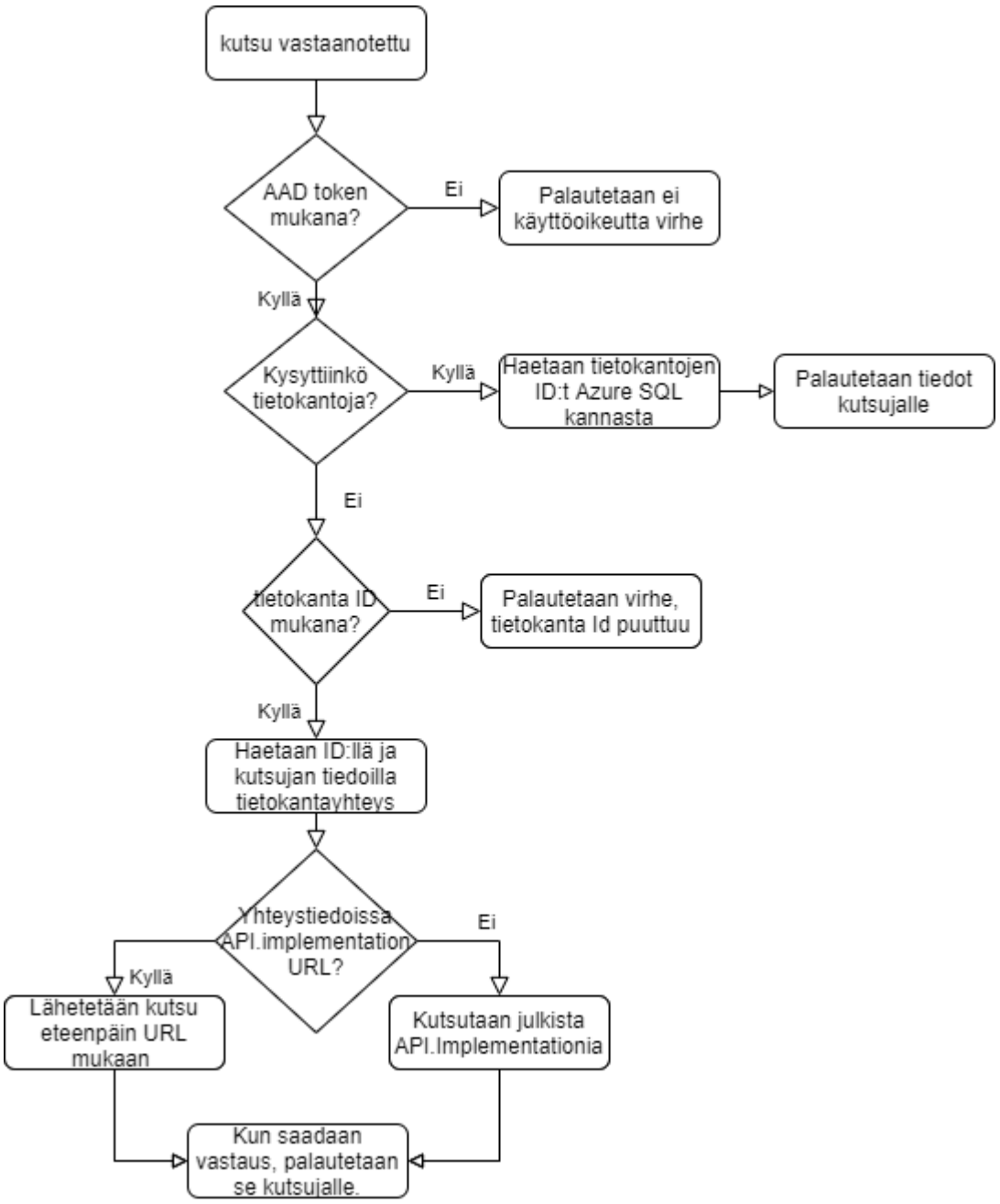
2xx-koodit viittaavat toiminnon onnistumiseen, 4xx-koodit virheelliseen kutsuun ja 5xx-koodit palvelinpään virheeseen. Näiden lisäksi on 1xx-koodeja, jotka ovat informatiivisia sekä 3xx-koodeja, jotka viittaavat uudelleenohjaukseen.

### 4.3 Logiikkataso

Logiikkataso sisältää kaiken taustatoiminnallisuuden. Tasoon kuuluvat esimerkiksi kaikki parametrien käsittelyt, erilaiset laskennalliset toiminnallisuudet, kuten toimenpiteiden hintojen laskenta sekä entiteettien muokkaus julkisista luokista tietokantaluokkiin ja toisinpäin.

### 4.3.1 API.V1-toiminta

API.V1 on julkinen kaikkien asiakkaiden kesken jaettu rajapinta, jonka tarkoituksena on mahdollistaa useiden API.Implementation rajapintojen toiminta eri ympäristöissä. Tälle ominaisuudelle on tarvetta, koska useilla suurilla asiakkailla on tietokanta heidän omassa ympäristössään, jonne eie ole pääsyä julkisesta API.Implementation rajapinnasta. Tämä mahdollistaa myös sen, että eri sovelluksiin ja kolmannen osapuolen integraatioihin ei tarvitse tehdä käsittelyä eri asiakasympäristöille. API.V1 rajapinnan toiminta seuraa alla olevaa Kaavio 1: API.V1 toiminnan vuokaavio.




Kaavio 1: API.V1 toiminnan vuokaavio

### 4.3.2 API.Implementation toiminta

API.Implementation rajapinta voidaan julkaista kahdella eri tavalla. Ensimmäinen tapa on julkaista rajapinta julkisena, jolloin se palvelee CSIn pilvipalveluasiakkaita. Toinen tapa on julkaista rajapinta asiakkaan omaan ympäristöön, jolloin sinne on pääsy vain kyseisen asiakkaan käyttäjillä. API.Implementation –rajapinta vastaa CSI API –rajapinnan varsinaisesta toiminnallisuudesta. Jokaista kut-

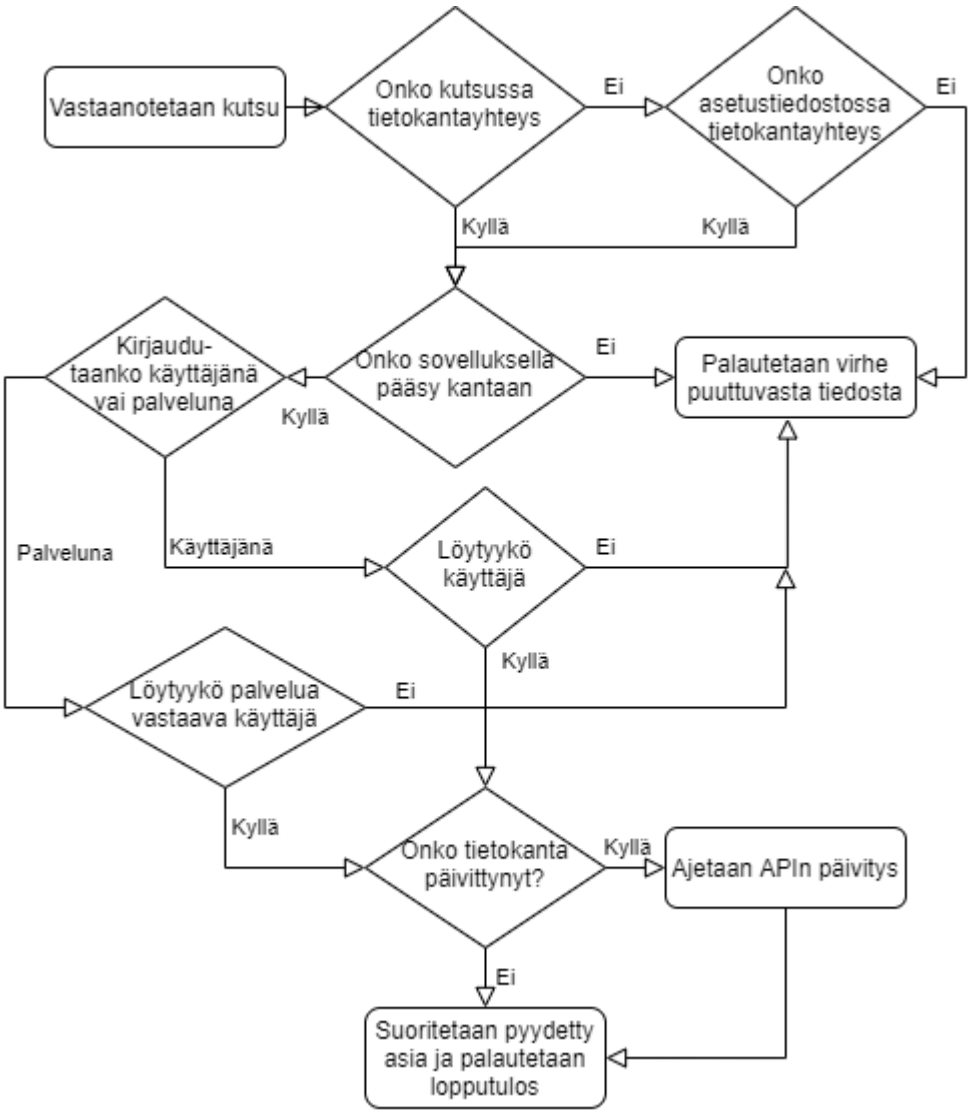
sua varten alustetaan kutsuun liittyvän logiikan sisältävä Handler-luokan instanssi sekä yhteys data access –tasoon, jonka kautta se voi käsitellä tietokantaa.

## 4.4 Data Access ja Data Storage –tasot

Alimmaisena arkkitehtuurissa on Data Access –taso, joka käsittelee tietokantaa sekä siellä olevia entiteettejä ja Data Storage -taso, joka sisältää sekä tietokannan, että sen näkymien ja taulujen luonnin ja muokkaamisen.

### 4.4.1 Käyttöoikeudet

CSI.API mahdollistaa kaksi erilaista tapaa kirjautua tietokantaan. Ensimmäinen vaihtoehto on, että CSI.API kirjautuu aina omana palvelukäyttäjänään ja tekee kirjaukset muiden käyttäjien puolesta. Tämä on tarkoitettu kolmannen osapuolen integraatioita varten, joissa palveluun tunnistautuminen suoritetaan jollain muulla tavalla kuin Azure AD –nimipalvelulla. Toinen vaihtoehto on, että jokainen käyttäjä kirjautuu omilla Azure AD tunnuksillaan. Tämä on tarkoitettu ensisijaisesti CSI Mobile sekä CSI MyDesk –sovellusten käyttöön, joihin käyttäjät kirjautuvat Azure AD –tunnuksilla.




Kaavio 2: API.Implementation tietokantaan tunnistautumisen vuokaavio

Kaavio 2: API.Implementation tietokantaan tunnistautumisen vuokaavio on nähtävillä tietokantaan tunnistautumisen vaiheet. Ensimmäisenä vaaditaan tietokannan yhteystieto joko kutsusta tai paikallisesta asetustiedostosta. Tämän jälkeen varmistetaan tietokannasta, että CSI API –rajapintaa kutsuva sovellus on rekisteröity tietokantaan ja että tietokannasta löytyy CSI Lawyer –sovelluksessa oleva käyttäjä, jolla on asianmukainen Azure AD –tunnus asetettuna. Käyttöoikeuksien varmistamisen jälkeen tarkastetaan, tarvitsevatko rajapinnan käyttämät tietokantanäkymät päivittämistä. Mikäli tarvetta on, suoritetaan uusien tietokantanäkymien muodostaminen automaattisesti.

### 4.4.2 Tietokannan käsittely

CSI Lawyer -ohjelmisto on tällä hetkellä noin kymmenen vuotta vanha, ja iän myötä sen tietokanta on todella monimutkainen. Tämän takia APIa ei haluta päästää käsiksi suoraan tietokannan tauluihin, vaan käsittely tapahtuu tietokantanäkymien kautta. Näin voidaan helposti luoda näkymät siten, että API-rajapinnalla on aina käytössään yhdenmukaiset näkymät riippumatta siitä, minkä CSI Lawyer –ohjelmistoversion tietokanta on kyseessä.

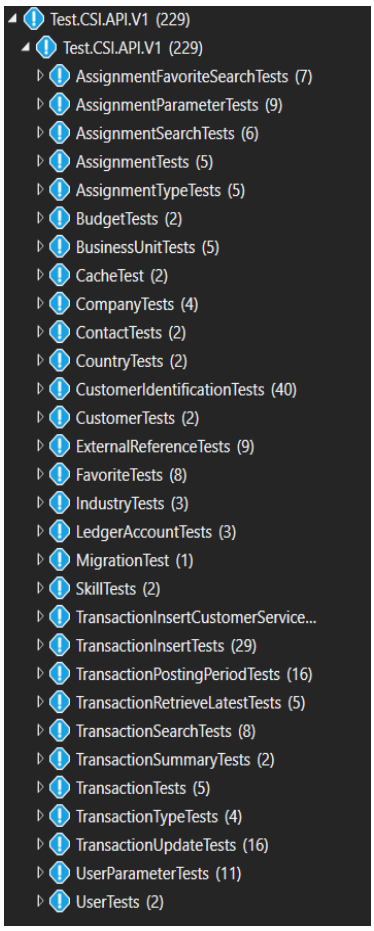
```
string tableName = "Transaction";
List<Helpers.Column> columns = new List<Helpers.Column>
{
    new Helpers.Column(tableName, "assignment_guid", nameof(Transaction.AssignmentId)),
    #region
    new Helpers.Column(tableName, "exchange_rate", nameof(Transaction.ExchangeRate))
};
if (currentDbVersion == MajorVersion.V8_1)
    columns.Add(new Helpers.Column("null", nameof(Transaction.SinglePriceLocked)));
else
    columns.Add(new Helpers.Column(tableName, "single_price_locked", nameof(Transaction.SinglePriceLocked)));
return Helpers.ViewHelper.GetCommandsForEntityView("Transactions", tableName, columns);
```

Kuva 10: Versiokohtaisten näkymien muodostaminen

Kuva 10: Versiokohtaisten näkymien muodostaminen on esimerkki siitä, kuinka versiokohtaisia näkymiä muodostetaan. Koska CSI Lawyer -ohjelmiston versiossa 8.1 ei vielä ollut mahdollista lukita yksikköhintaa, kyseistä saraketta ei ole tietokannoissa, joten sen sijaan näkymässä palautetaan sarakkeen sijasta aina null-arvo.

## 4.5 Järjestelmän testaus

Nykyaikaiseen ohjelmistokehitykseen kuuluvat oleellisena osana erilaiset automaattiset sekä manuaaliset testitapaukset. Näiden tarkoituksena on varmistaa, että uusia toiminnallisuuksia kehitettäessä vanhat toiminnallisuudet eivät muutu tai mene rikki ja uudet ominaisuudet toimivat suunnitelman mukaan. Tämä on ollut tavoitteena myös CSI.APIn tapauksessa. Tällä hetkellä testitapauksia on hieman yli 200 ja ne on jaettu muutamaan kymmeneen eri luokkaan, jotka ovat Kuva 11.




Kuva 11: Testiluokat

Kaikki CSI API –rajapinnan testit ovat tällä hetkellä niin sanottuja integraatiotestejä eli ne testaavat kokonaisuuden toimivuutta, eivät yksittäisiä funktioita.

Jotta testit voidaan suorittaa useita kertoja yhdenmukaisessa ympäristössä, testien suorittamista varten on rakennettava joka kerralla tietokanta. Tätä varten on luotu järjestelmä, joka osaa luoda tietokannat testejä varten. Koska järjestelmän tulee olla yhteensopiva CSI Lawyer –sovelluksen eri versioiden kanssa, luodaan tällä hetkellä jokaista testiajoa varten neljä tietokantaa, jotka päivitetään eri versioita vastaaviksi. Jatkossa julkaistaessa CSI Lawyer –sovelluksesta uusia versioita, testeihin tulee lisätä yksi tietokanta testejä varten.

Jokainen yksittäinen testitapaus koostuu neljästä osiosta. Ensimmäisenä tietokantaan alustetaan tiedot, jotka tarvitaan testitapausta varten. Usein tämä sisältää ohjelmistoparametrien asettamista, asiakkaiden, käyttäjien ja toimeksiantojen lisäämistä sekä välimuistin tyhjentämistä rajapinnasta. Seuraavassa vaiheessa suoritetaan varsinainen toiminnallisuus ja otetaan vastaus talteen. Kolmannessa

vaiheessa tarkastetaan, että saatu vastaus vastaa odotettua. Vastauksesta tarkastetaan http-tilakoodi, mahdolliset virheviestit sekä varsinainen palautunut data. Mikäli testissä on syötetty tietoa tietokantaan, tarkastetaan tässä vaiheessa myös tietokantaan luodun entiteetin tiedot. Viimeisessä vaiheessa tietokanta palautetaan alkuperäiseen tilaan, jotta seuraava testitapaus pääsee aloittamaan samasta alkutilanteesta.

### 4.6 APIn dokumentointi käyttäjille

Yksi erittäin tärkeä osio julkisesta rajapinnasta on sen dokumentointi käyttäjiä varten. Jos rajapinnan dokumentaatio on virheellinen tai puutteellinen, sen käyttäminen on ulkopuolisille käytännössä mahdotonta. Dokumentointia varten API.V1 –rajapinnassa käytetään Microsoftin kehittämää ASP.NET Web API Help Page -kirjastoa, jonka avulla saadaan varsin pienellä työllä luotua automaattisesti päivittyvä ohjesivusto.

Kirjaston toiminta perustuu useisiin eri attribuutteihin, joiden avulla kerrotaan esimerkiksi eri kutsujen paluuarvojen tyyppi, kutsuissa olevat parametrit, sekä niiden pakollisuus. Tämän lisäksi kirjasto vie sivustolle automaattisesti summary-kommentit, joissa voidaan antaa tarkempi kuvaus kutsun toiminnasta ja sen parametreistä. Näiden lisäksi on mahdollista määrittää omia attribuutteja, joiden avulla voidaan luoda erikoistuneempia ohjeistuksia.

# 5 LOPPUTILANNE

CSI API –rajapinta saatiin opinnäytetyön aikana vietyä hyvin lähelle käyttövalmiutta, mutta siinä on vielä joitain lisäkehitystä vaativia asioita. Suurin osa havaituista tarpeista on saatu täytettyä, mutta CSI Mobile ja CSI MyDesk –sovellusten kehityksen yhteydessä havaitaan edelleen puuttuvia ominaisuuksia, jotka ovat joko jääneet huomioon tai ovat vielä työlistalla.

Opinnäytetyön jälkeen CSI API –rajapinnan kehittäminen keskittyy ensisijaisesti rajapinnan viimeistelyyn, jotta CSI Mobile ja CSI MyDesk –sovellukset saadaan käyttämään uutta rajapintaa. Tämän jälkeen jatkokehityksessä keskitytään tarpeisiin, joita havaitaan kolmansien osapuolten integraatioiden yhteydessä.

# 6 POHDINTA

Opinnäytetyön tavoitteena oli tutustua Azure-alustaan sekä jatkokehittää CSI API –rajapintaa, jonka avulla mahdollistetaan CSI Mobile ja CSI MyDesk –sovellusten toiminnallisuudet. Rajapinnan kehittäminen on edistynyt hyvin, mutta se vaatii viel hieman viimeistelyä ennen kuin se saadaan varsinaiseen käyttöön. Aikataulusyistä opinnäytetyö päättyi ennen rajapinnan valmistumista, mutta rajapinnan kehittäminen jatkuu. Yksi tärkeimmistä osioista eli CSI API –rajapinnan tietoturvallisuus on vielä tarkastamatta.

Azure-alusta on hyvin monipuolinen palvelukokonaisuus. Palveluiden valtava määrä tuottaa ongelmia etenkin alussa, koska juuri oikean palvelun valitseminen useiden vastaavien joukosta voi olla haastavaa. Palveluun tutustuessa alusta kuitenkin vähitellen avautuu, jolloin omaan käyttöön soveltuvat ominaisuudet alkavat löytyä kohtalaisen helposti.

Suurimmat ongelmat kehitystyössä muodostuivat integraatiotestien toteuttamisesta. Testirakenteen toteutuksen jälkeen varsinainen testien tekeminen oli varsin yksinkertaista. Testien epäonnistuessa tietokantaan jäi kuitenkin usein asetuksia, jotka kaatoivat muitakin testejä. Tämä vaikeutti huomattavasti virheellisen testin löytämistä, koska ensimmäisen testin epäonnistuttua suuri osa muistakin testeistä epäonnistui, ja vian varsinaisen lähteen selvittäminen oli aikaa vievää.

**LÄHTEET**

Microsoft. 6.5.2020, What is Azure Active Directory? Luettu 15.8.2020. https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-whatis

Microsoft. 26.11.2019. How and why applications are added to Azure AD? Luettu 16.8.2020. https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-how-applications-are-added

Microsoft. 18.1.2019. Azure Key Vault basic concepts. Luettu 16.8.2020. https://docs.microsoft.com/en-us/azure/key-vault/general/basic-concepts

Microsoft. 27.7.2020. What is Azure SQL?. Luettu 21.8.2020. https://docs.microsoft.com/en-us/azure/azure-sql/azure-sql-iaas-vs-paas-what-is-overview

Microsoft. 22.7.2020. Features comparison: Azure SQL Database and Azure SQL Managed Instance. Luettu 30.9.2020. https://docs.microsoft.com/en-us/azure/azure-sql/database/features-comparison

Scott Carey. 23.1.2020. AWS vs Azure vs Google Cloud: What's the best cloud platform for enterprise? Luettu 16.7.2020. https://www.computerworld.com/article/3429365/aws-vs-azure-vs-google-whats-the-best-cloud-platform-for-enterprise.html

Daniel Jimenez Garcia. 27.4.2019. The History of ASP.NET Luettu 1.8.2020. https://www.dotnetcurry.com/aspnet/1493/aspnet-history-part-2-mvc

Microsoft. n.d. .Net Programming Languages. Luettu 4.9.2020 https://dotnet.microsoft.com/languages

Janice Friedman. 17.5.2018. Understanding the Differences Between C#, C++, and C. Luettu 14.9. https://csharp-station.com/understanding-the-differences-between-c-c-and-c/

Microsoft. n.d. What is .NET? Luettu 17.9.2020. https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet

Microsoft. n.d. What is .NET Framework? Luettu 22.7.2020. https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet-framework

entityframeworktutorial.net. n.d. What is Entity Framework? Luettu 23.7.2020. https://www.entityframeworktutorial.net/what-is-entityframework.aspx

tutorialsteacher.com. n.d. What is Entity Framework? Luettu 23.7.2020. https://www.tutorialsteacher.com/webapi/what-is-web-api

Sashko Stubailo. 27.6.2017. GraphQL vs. REST. Luettu 24.7.2020. https://www.apollographql.com/blog/graphql-vs-rest-5d425123e34b

RestApiTutorial.com. n.d. Using HTTP Methods for RESTful Services. Luettu 26.7.2020. https://www.restapitutorial.com/lessons/httpmethods.html

restfulapi.net. n.d. HTTP Status Codes. Luettu 30.7.2020. https://restfulapi.net/http-status-codes/