Tampereen ammattikorkeakoulu

C#-asiakasohjelman luominen OpenAPI-määrityksestä sekä sen käyttö Azuren API Managementin läpi

Lasse Mattila

OPINNÄYTETYÖ
Joulukuu 2021

Tieto- ja viestintätekniikan tutkinto-ohjelma
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikan tutkinto-ohjelma
Ohjelmistotekniikka

MATTILA, LASSE:
C#-asiakasohjelman luominen OpenAPI-määrityksestä sekä sen käyttö Azuren API Managementin läpi

Opinnäytetyö 31 sivua
Joulukuu 2021

Opinnäytetyön taustalla oli TietoEVRYn asiakkaan halu tehdä konseptitodistus tässä opinnäytetyössä käytettyjä teknologioita ja niiden toimintoja hyödyntäen. Tästä syystä luottamuksellinen aineisto on poistettu tästä raportista. Opinnäytetyön tavoitteena oli luoda onnistuneesti C#-ohjelmointikieltä hyödyntävä asiakasohjelma OpenAPI-määrityksestä ja sen jälkeen tehdä kutsu Azuren API Managementin läpi asiakkaan olemassa olevalle palvelimelle. Työssä avataan käytettyjä teknologioita sekä toteutetaan tavoitteen mukainen kokonaisuus. Lopuksi pohditaan työn tuloksia sekä mahdollisia jatkokehitysideoita.

Opinnäytetyön tuloksena saatiin luotua NSwagStudiolla OpenAPI-määrityksestä toimiva asiakasohjelma ja sen avulla tehtyä kutsu Azuren API Managementin läpi asiakkaan olemassa olevalle palvelimelle. Asiakasohjelman luonti OpenAPI-määrityksestä tuotti parhaimmillaan yli 30 000 koodirivin asiakasohjelman ja osoittautui näin ollen todella tehokkaaksi tavaksi luoda asiakasohjelmia.

Työtä voisi lähteä jatkokehittämään esimerkiksi selvittämällä keinoja hyödyntää Azuren API Managementin laajoja toiminnallisuuksia. Hyvä esimerkki tästä voisi olla kutsujen oikeuksien tarkistaminen API Managementin avulla sekä toiminnallisuuden toteuttaminen asiakasohjelmassa.

Asiasanat: openapi, microsoft azure, api management, nswag, .net

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in ICT Engineering
Software engineering

MATTILA, LASSE:
Generating C# client from OpenAPI specification and using it through Azure's API Management

Bachelor's thesis 31 pages
December 2021

The basis for this bachelor's thesis was TietoEVRY's customers wish to do a proof of concept for the technologies and functionality used in this thesis. For that reason, all confidential material has been removed from this report.

The goal for this thesis was to generate a functioning client that uses C# programming language from OpenAPI specification and to use it to make a successful request through Azure API Management into the existing backend of the customer. This thesis introduces the used technologies and implements the totality according to the goal. At the end of the thesis there is section where results and possible future developments are reflected.

As a result of this thesis a working client using C# programming language was generated from OpenAPI specification using NSwagStudio and with it a successful request was made through Azure API Management to the existing backend of the customer. The client generating from OpenAPI specification produced at best over 30 000 rows of code and was shown to be extremely efficient way to produce clients.

Key words: openapi, microsoft azure, api management, nswag, .net

SISÄLLYS

LYHENTEET JA TERMIT

API	Ohjelmointirajapinta, Application programming interface
C#	Ohjelmointikieli, C Sharp
HTTP	Tiedonsiirto protokolla, Hypertext Transfer Protocol
JSON	Tiedostomuoto, JavaScript Object Notation
TAMK	Tampereen ammattikorkeakoulu
URL	Web-osoite, Uniform Resource Locator
YAML	Merkintäkieli, YAML Ain't Markup Language

1 JOHDANTO

Tämä opinnäytetyö käsittelee C#-ohjelmointikieltä hyödyntävän asiakasohjelman luontia OpenAPI-määrityksestä sekä sen hyödyntämistä Microsoft Azuren API Managementin läpi. Opinnäytetyön aihe on ajankohtainen, koska julkisten pilvipalveluiden käyttö kasvaa kovalla vauhdilla (Gartner 2021). Asiakasohjelman luominen OpenAPI-määrityksestä taas mahdollistaa muun muassa pienemmän määrän ihmisestä johtuvia virheitä sekä huomattavan tehokkuushyödyn (Escott 2020).

Työn lähtökohtana ja aiheen valintana on toiminut työnantajani TietoEVRYn asiakkaan halu tehdä konseptitodistus työssä käytettyjä toiminnallisuuksia sekä teknologioita hyödyntäen. Työssä tutkitaan, miten saada onnistunut kutsu Azuren API Managementin läpi palvelimelle OpenAPI-määrityksestä luodun asiakasohjelman avulla. Työn tavoitteena on ensin luoda onnistuneesti C#-ohjelmointikieltä hyödyntävä asiakasohjelma OpenAPI-määrityksestä ja tehdä sen jälkeen kutsu sitä hyödyntäen Azuren API Managementin läpi asiakkaan olemassa olevalle palvelimelle. Tämä työ keskittyy kyseisten tavoitteiden toteuttamiseen, muttei esimerkiksi tutki Azuren tai API Managementin laajoja ja monipuolisia ominaisuuksia tai OpenAPI-määrittelyn yksityiskohtia. Tässä raportissa ei myöskään esitetä työssä oikeasti käytettyjä tiedostoja niiden arkaluontoisuuden takia vaan työn vaiheet esitetään esimerkinomaisen toteutuksen avulla.

2 KESKEISET TEKNOLOGIAT

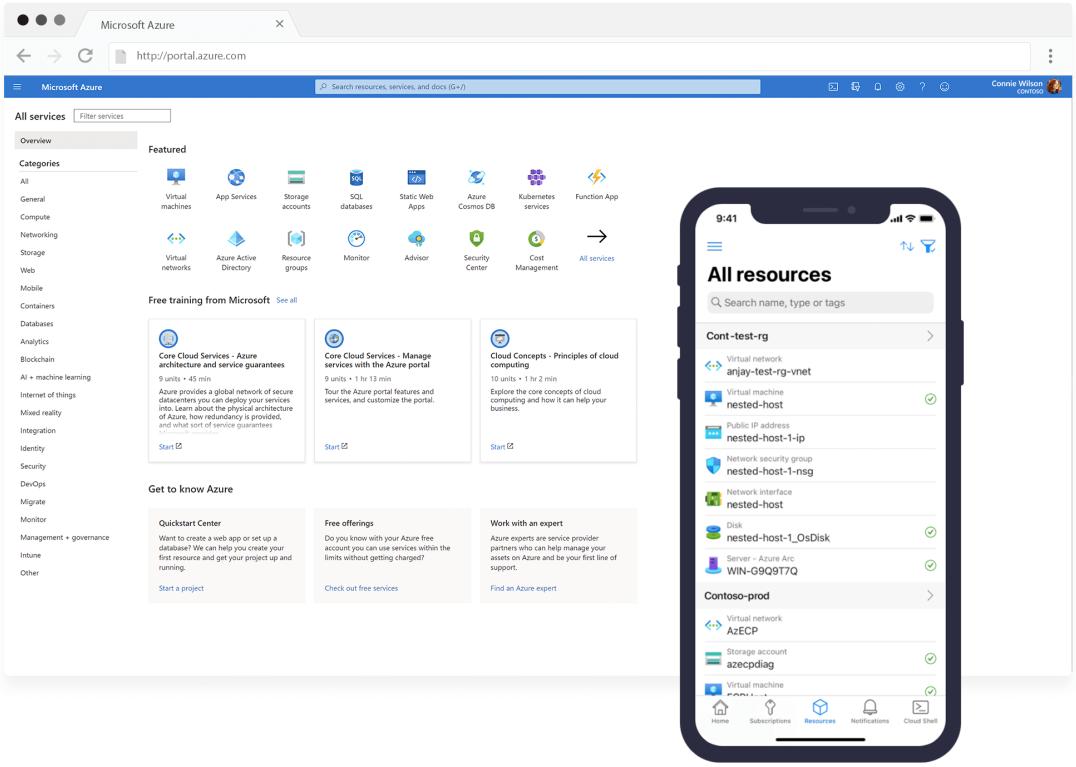
2.1 Microsoft Azure

Azure on pilvipalvelu, joka tarjoaa yli 200 tuotetta ja palvelua erilaisiin informaatiotekniikan tarpeisiin. 95 % Fortune 500 yrityksistä hyödyntää sen tuotteita ja palveluja. (What is Azure? 2021).

Pilvipalvelut ovat tietojenkäsittely palveluiden toimittamista ja tuottamista internetin välityksellä. Esimerkkejä tarjottavista palveluista ovat palvelimet, tietokannat sekä erilaiset tiedontallennustilat. (What is cloud computing? 2021)

Pilvipalvelut tarjoavat lukuisia hyötyjä verrattuna tavalliseen informaatiotekniikan resurssien hallintaan. Pilvipalveluiden keskeisimpiä hyötyjä ovat niiden matalammat kustannukset, globaali skaalautuvuus ja niiden mahdollistamat tehokkuushyödyt. (Top benefits of cloud computing 2021)

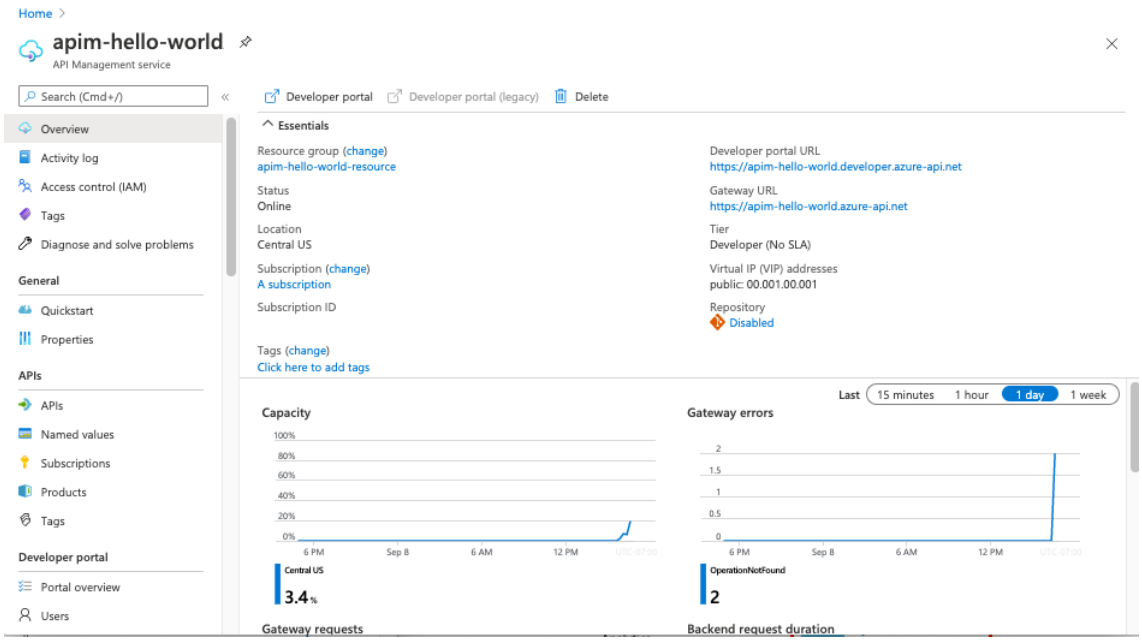
Azure portal on web-pohjainen Azuren hallintaan käytettävä graafinen käyttöliittymä. Azure portalin avulla voi hallita ja monitoroida erilaisia Azuren resursseja. Azure portalille on myös olemassa mobiilisovellus. (What is the Azure portal? 2021) Kuvassa 1 on näkymä Azure portalista ja kaikista sen tarjoamista palveluista.



KUVA 1. Näkymä Azure portalista ja kaikista sen tarjoamista palveluista (Microsoft Azure portal 2021).

2.1.1 Azure API Management

API Management on yksi Azuren tarjoamista palveluista. API Managementin tarkoituksena on tarjota ohjelmointirajapinnoille paikka, jossa niitä kaikkia voidaan hallinnoida yhdessä paikassa. API Management tarjoaa valtavasti erilaisia toiminnallisuuksia ohjelmointirajapintojen hallintaan. (API Management 2021) Kuvassa 2 on yleisnäkymä API Managementista Azure portalissa.



KUVA 2. Yleisnäkymä API Managementista Azure portalissa (Create a new Azure API Management service instance by using the Azure portal 2021).

API Managementin avulla voidaan julkaista ohjelmointirajapintoja sekä sisäisille että ulkoisille ohjelmistokehittäjille. API Management koostuu kolmesta pääkomponentista, joita ovat API gateway, Azure portal ja Developer portal. (About API Management 2021)

API gateway on päätepiste, joka hyväksyy ja ohjaa ohjelmointirajapinnan kutsut palvelimelle. Sen tehtävänä on myös vahvistaa ja tarkistaa sille mahdollisesti asetetut pääsytiedot ja valvoa mahdollisten sille asetettujen kiintiöiden rajoja. (Overview 2021)

Azure portal toimii API Managementin ylläpidon käyttöliittymänä. Sen avulla voidaan tuoda ja määritellä ohjelmointirajapintoja, asettaa kiintiöitä tai muita käytäntöjä sekä analysoida ohjelmointirajapintojen käyttöä. (Overview 2021)

Developer portalin päätehtävänä on toimia porttina ja käyttöliittymänä kehittäjille. Siellä kehittäjät voivat tutustua ohjelmointirajapintojen dokumentaatioon ja saada analytiikkaa heidän omasta ohjelmointirajapintojen käytöstä. (Overview 2021)

2.2 OpenAPI

OpenAPI-määritys määrittelee standardoidun ja ohjelmointikieli agnostisen rajapinnan HTTP ohjelmointirajapinnoille. Tämä mahdollistaa sekä ihmisten että tietokoneiden ymmärtää ja hyödyntää siinä määriteltyjä palveluita ilman pääsyä lähdekoodiin tai erilliseen dokumentaatioon. OpenAPI-määritystä voidaan hyödyntää esimerkiksi rajapinnan dokumentoinnissa tai sen avulla voidaan myös luoda palvelimia ja asiakasohjelmia. OpenAPI-määritys voi olla tiedostomuodoltaan joko JSON tai YAML. (What is OpenAPI Specification? 2021) Kuvassa 3 on yksinkertainen esimerkki OpenAPI-määrityksestä YAML tiedostomuodossa.

```yaml
openapi: 3.0.0
info:
  title: Sample API
  description: Optional multiline or single-line description in [CommonMark](http://commonmark.org/help/) or HTML.
  version: 0.1.9

servers:
  - url: http://api.example.com/v1
    description: Optional server description, e.g. Main (production) server
  - url: http://staging-api.example.com
    description: Optional server description, e.g. Internal staging server for testing

paths:
  /users:
    get:
      summary: Returns a list of users.
      description: Optional extended description in CommonMark or HTML.
      responses:
        '200':    # status code
          description: A JSON array of user names
          content:
            application/json:
              schema:
                type: array
                items:
                  type: string
```

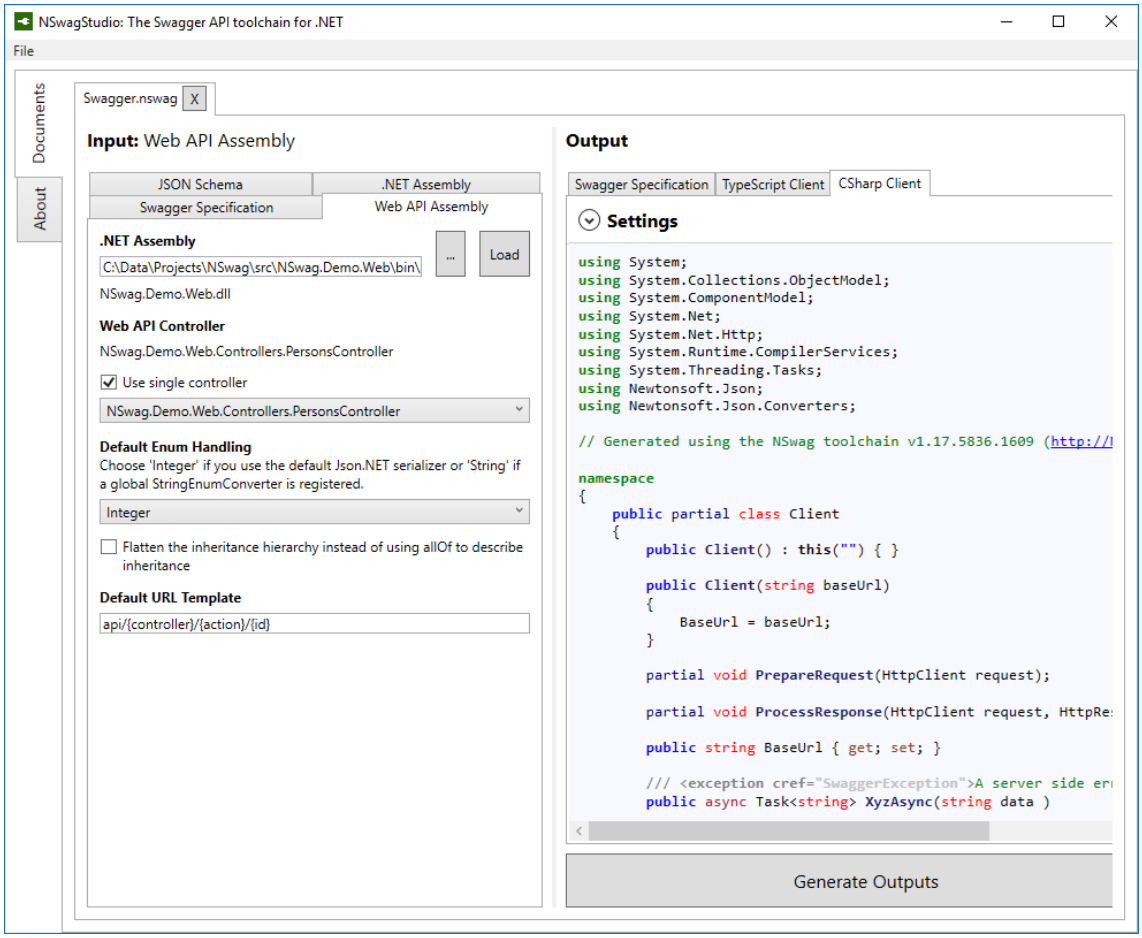
KUVA 3. Yksinkertainen esimerkki OpenAPI-määrityksestä YAML tiedostomuodossa (Basic Structure 2021).

2.3 NSwag

NSwag on joukko ohjelmointityökaluja OpenAPI-määrittelyille. NSwag tarjoaa työkaluja itse OpenAPI-määritysten luomiseen sekä asiakasohjelmien luomiseen OpenAPI-määrityksistä. (NSwag: The Swagger/OpenAPI toolchain for .NET, ASP.NET Core and TypeScript 2021)

2.3.1 NSwagStudio

NSwagStudio on yksi NSwagin tarjoamista työkaluista. NSwagStudio on verkosta ladattava Windows-työpöytäsovellus, jonka avulla voidaan luoda esimerkiksi C#-asiakasohjelmia OpenAPI-määrityksistä. (NSwagStudio 2021) Kuvassa 4 on kuvattuna NSwagStudion käyttöliittymä.



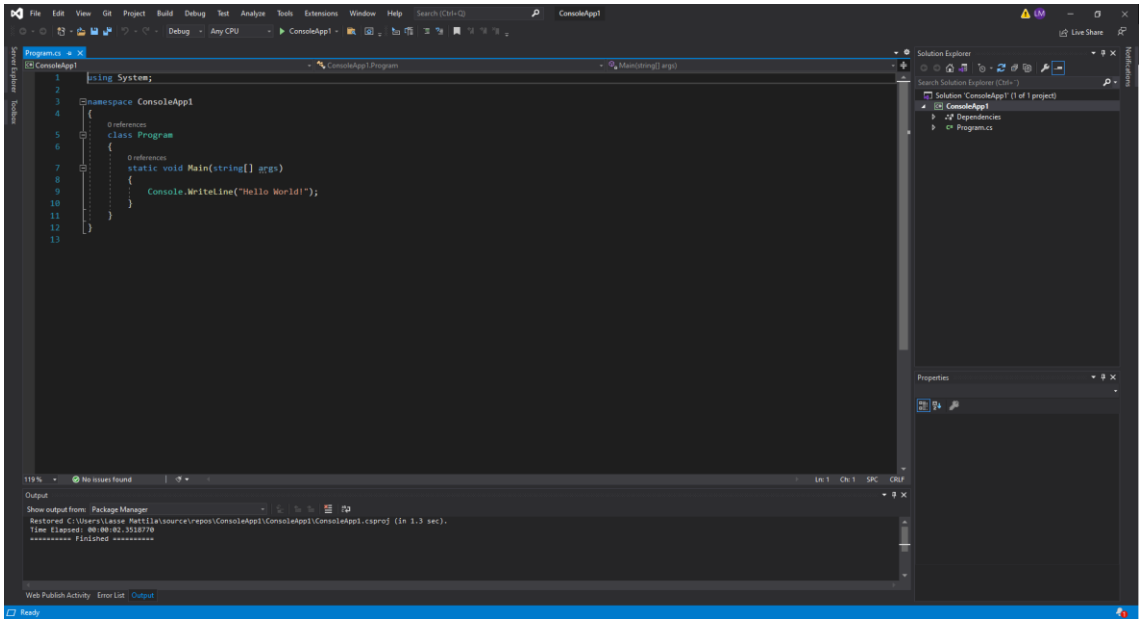
KUVA 4. NSwagStudion käyttöliittymä (NSwagStudio 2021)

2.4 .NET

.NET on käyttöjärjestelmä riippumaton ja avoimeen lähdekoodiin perustuva ilmainen ohjelmistokehitysalusta erilaisten ohjelmien kehittämiseen. .NET sovelluksia voi kehittää C#, F# tai Visual Basic ohjelmointikielillä. .NET:in paketinhallintaohjelma NuGet tarjoaa yli 90 000 pakettia toiminnallisuuden laajentamista varten ja Visual Studio kehitysympäristö tarjoaa parhaan kehitysympäristön .NET kehitykselle. (What is .NET? 2021)

2.5 Visual Studio

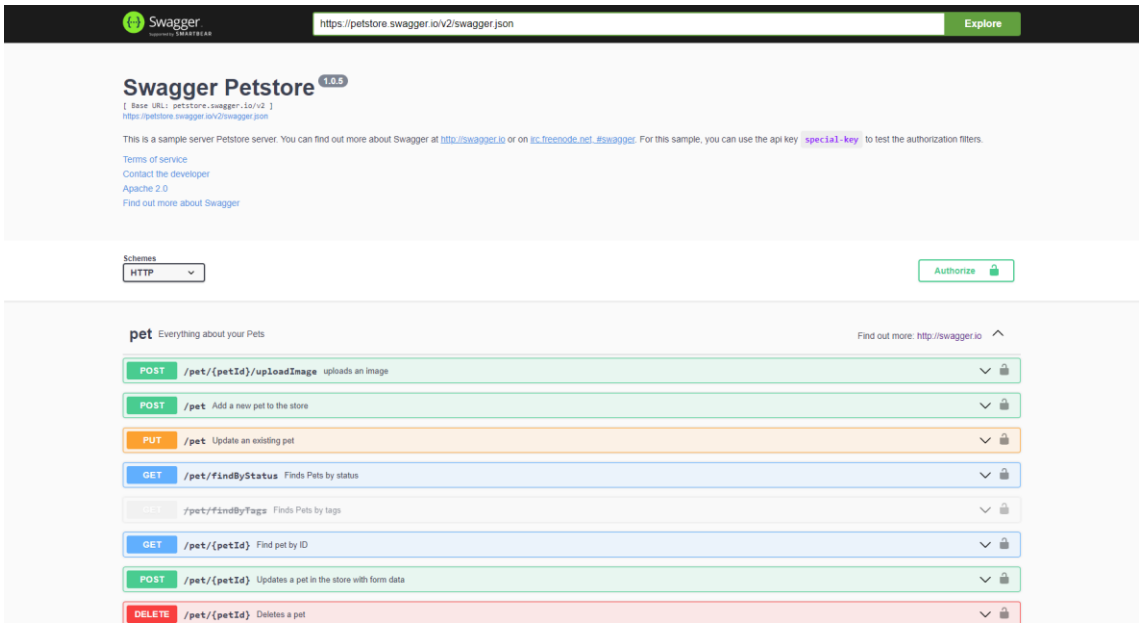
Visual Studio on kattava ohjelmankehitysmpäristö .NET ja C++ kehitykseen Windows-käyttöjärjestelmälle. Visual Studiolla voi ohjelmoida, testata ja ottaa käyttöön erilaisia ohjelmia. (Visual Studio for Windows 2021) Kuvassa 5 on yleisnäkymä Visual Studion käyttöliittymästä.



KUVA 5. Yleisnäkymä Visual Studion käyttöliittymästä

3 TOTEUTUS

Tässä kappaleessa käytetyt kuvat ja niissä olevat tiedot eivät ole oikeassa työssä käytettyjä vaan esimerkinomaisia kuvia oikeaa työtä vastaavasta esimerkkitoteutuksesta. Tässä kappaleessa kuvatussa toteutuksessa käytettiin esimerkkinä Swagger Petstore palvelinta ja siitä luotua OpenAPI-määritystä, koska ne olivat julkisesti saatavilla ja käytettävissä (Swagger Petstore 2021). Kuvassa 6 on esimerkkitoteutuksessa käytetyn Swagger Petstore ohjelmointirajapinnan kuvaus Swagger UI:ta käyttäen.

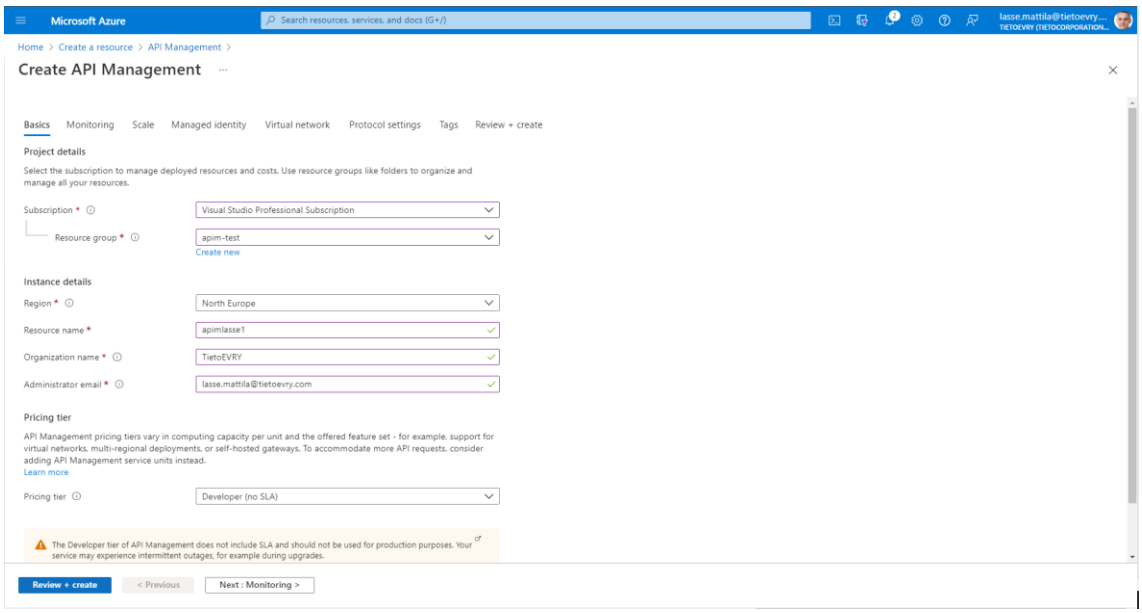


KUVA 6. Esimerkkitoteutuksessa käytetyn Swagger Petstore ohjelmointirajapinnan kuvaus Swagger UIta käyttäen (Swagger Petstore 2021)

Oikeassa työssä käytetty OpenAPI-määritys saatiin asiakkaalta. OpenAPI-määrityksessä oli määriteltynä asiakkaan olemassa olevan palvelimen toiminta.

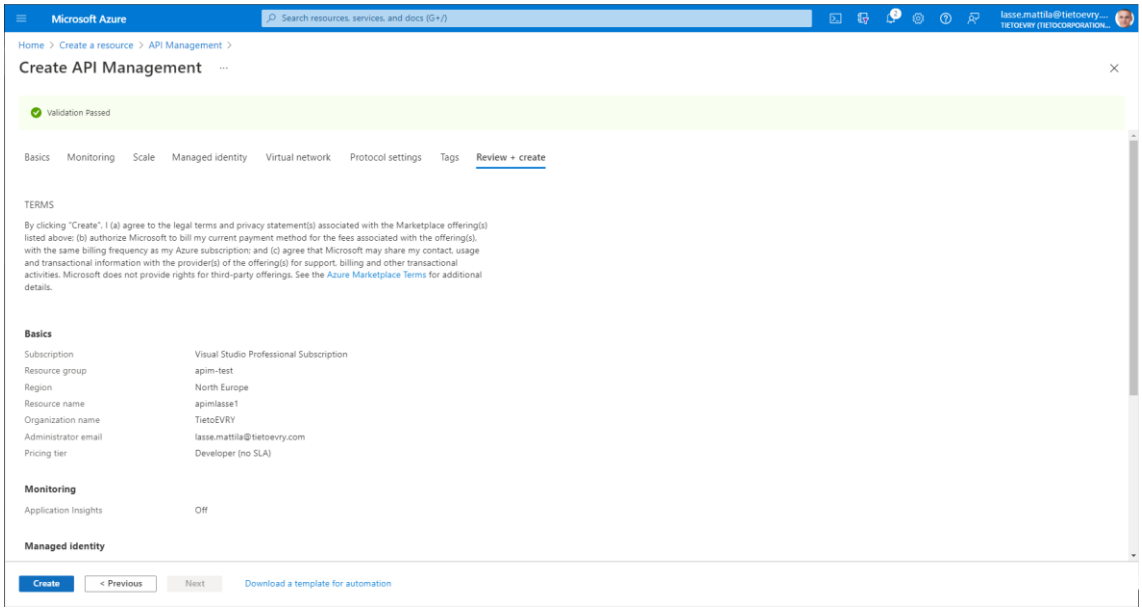
3.1 API Managementin luominen Azure Portalissa

Työ alkoi luomalla API Management service Azure portalissa. Luomisen yhteydessä API Managementille määriteltiin sille vaaditut tiedot kuten sen nimi, sijainti ja resurssiryhmä. Kuvassa 7 on esitettynä esimerkkitoteutuksessa käytetyn API Management servicen käyttämät tiedot. Tietojen syöttämisen jälkeen painettiin Review + Create-nappia.



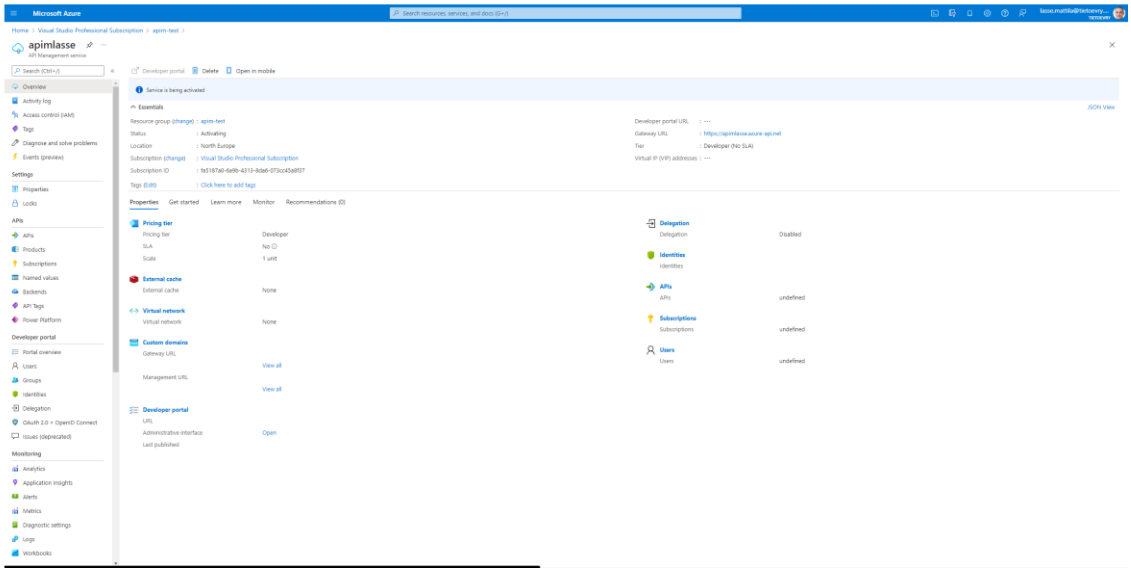
KUVA 7. API Management servicessä käytetyt tiedot esimerkkitoteutuksessa

Review + Create-napin painamisen jälkeen API Management vielä vahvisti kaikki sille annetut tiedot. Kuvassa 8 on tilanne Review + Create-napin painamisen jälkeen. Vahvistuksen jälkeen kuvassa 8 olevaa Create-nappia painamalla Azure alkoi luomaan varsinaista API Management serviceä.



KUVA 8. API Management service sen luonti vaiheessa sille annettujen tietojen vahvistamisen jälkeen

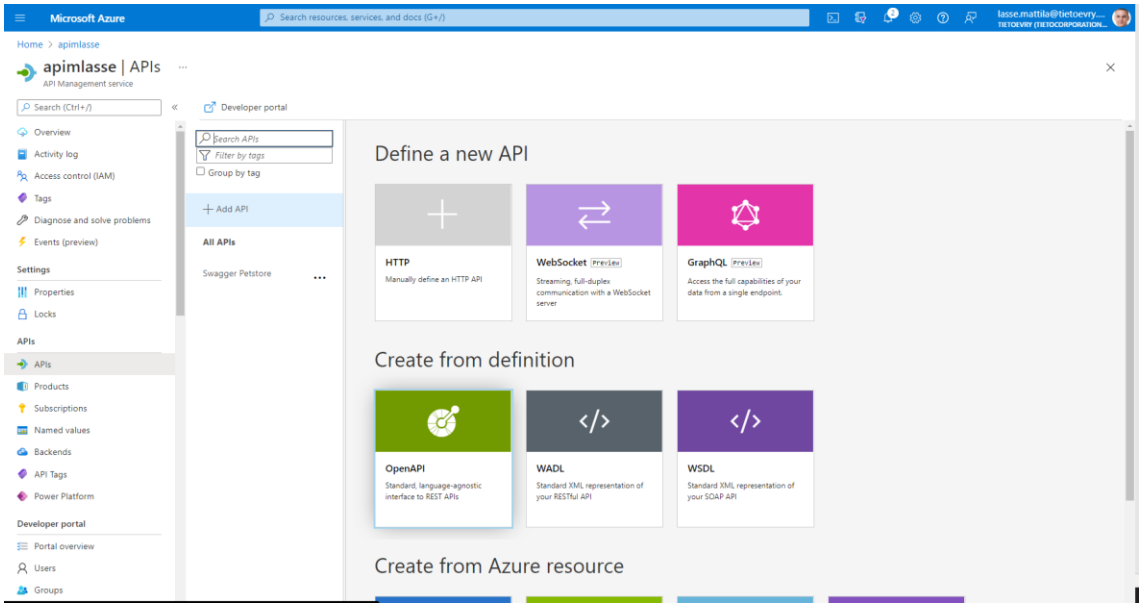
API Management servicen luomiseen ja aktivointiin voi kulua jopa 40 minuuttia. (Create a new Azure API Management service instance by using the Azure portal 2021) Kuvassa 9 on esimerkkitoteutuksessa luodun API Management servicen yleisnäkymä Azure portalissa sen luonnin jälkeen.



KUVA 9. Azure portalissa luodun API Management servicen yleisnäkymä sen luonnin jälkeen esimerkkitotetuksessa

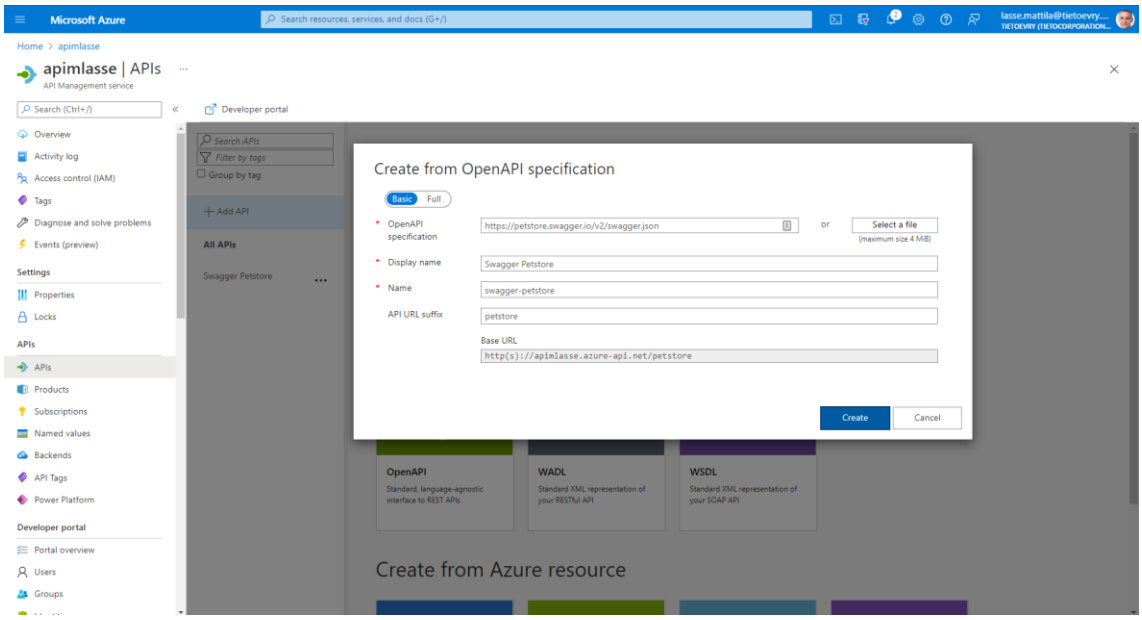
3.2 OpenAPI-määrityksen tuominen API Managementtiin

Seuraavaksi asiakkaalta saatu OpenAPI-määritys tuotiin API Managementtiin (Import an OpenAPI specification 2021). Kuvassa 10 on API Managementin valikosta avattuna kohta APIs ja painettu Add API-nappia. Työssä haluttiin lisätä ohjelmointirajapinta OpenAPI-määrityksestä, joten kohdasta Create from definiton valittiin OpenAPI.



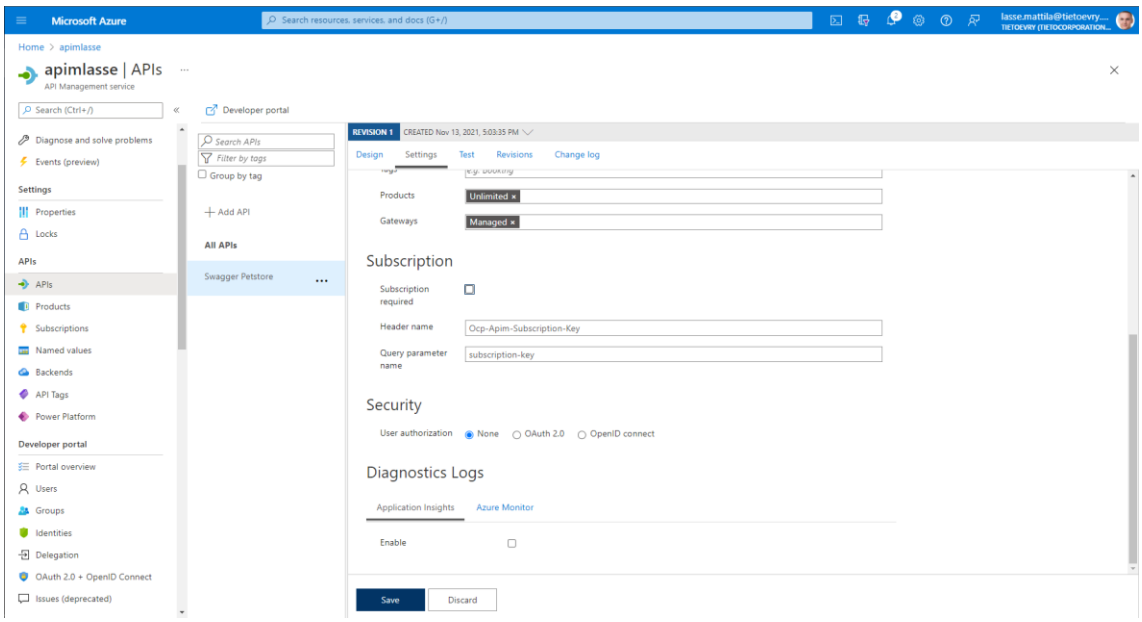
KUVA 10. API Managementin valikosta avattuna kohta APIs ja siitä painettu Add API-nappia

Kuvassa 11 on OpenAPI-painikkeen painamisen jälkeinen tilanne, johon syötettiin esimerkkitoteutuksessa käytetyn OpenAPI-määrityksen tiedot. Create from OpenAPI specification-ponnahdusikkunassa kohdassa OpenAPI specification annettiin joko OpenAPI-määritelty tiedosto tai sen URL. API Management osasi OpenAPI-määrityksen perusteella asettaa muut pakolliset tiedot, mutta niitä pystyi myös halutessaan muokkaamaan. Tietojen syöttämisen jälkeen painettiin Create-nappia, joka lisäsi ohjelmointirajapinnan API Managementtiin.



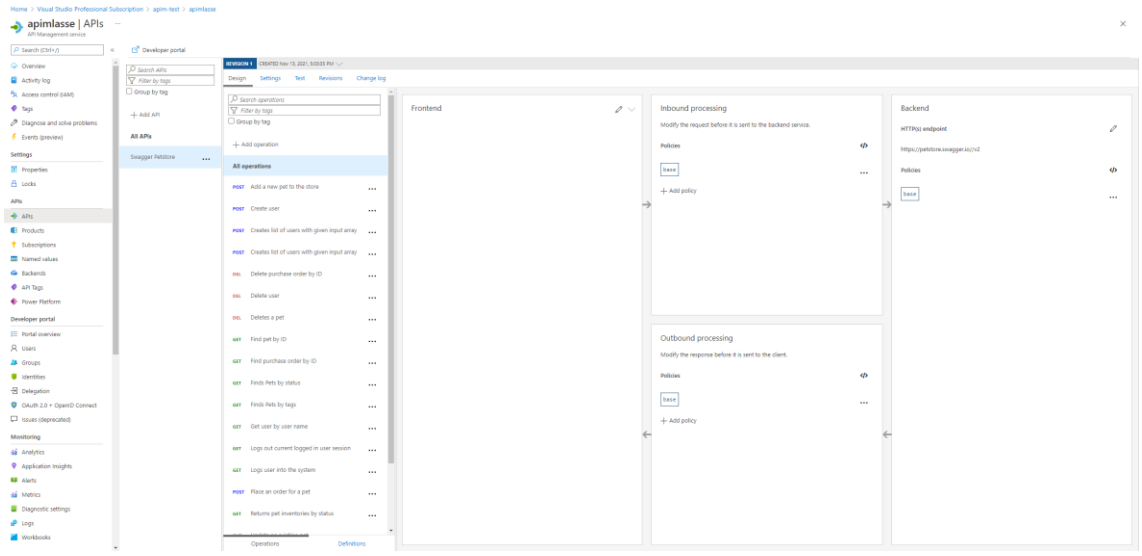
KUVA 11. Esimerkkitoteutuksessa käytetyn OpenAPI-määrityksen tietojen syöttäminen Create from OpenAPI specification ponnahdusikkunassa

Työssä ei haluttu tehdä minkäänlaista ohjelmointirajapinnan käyttöön liittyvää oikeuksien tarkastamista, joten ohjelmointirajapinnan lisäämisen jälkeen valittiin lisätty ohjelmointirajapinta APIs-sivulta painamalla Swagger Petstore-nappia. Tämän jälkeen siirryttiin Settings-välilehdelle ja poistettiin käytöstä Subscription required-ominaisuus. Tämän jälkeen asetukset tallennettiin painamalla Save-nappia. Kuvassa 12 on navigoitu Settings-välilehdelle ja poistettu käytöstä Subscription required-ominaisuus.



KUVA 12. Subscription required-ominaisuuden poistaminen käytöstä lisätyn ohjelmointirajapinnan Settings-välilehdeltä

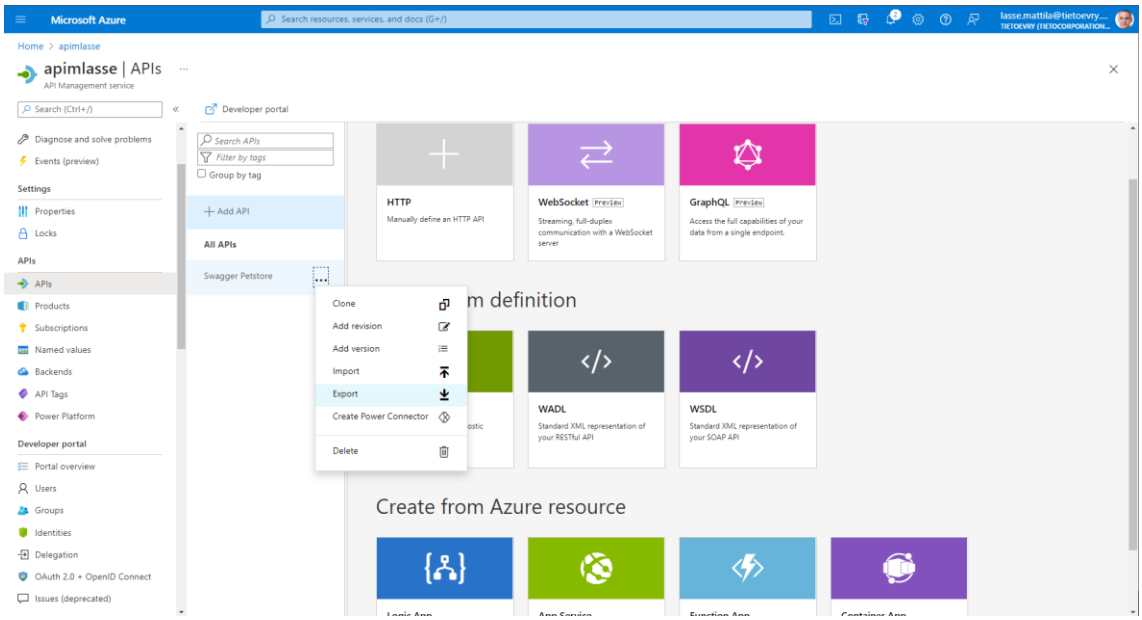
Tämän jälkeen siirryttiin Design-välilehdelle, jossa varmistettiin, että palvelun backendin päätepiste oli määritelty oikein ohjelmointirajapinnan Backend-kohdassa. API Management osasi asettaa palvelimen päätepisteen oikein suoraan OpenAPI-määrityksestä. Kuvassa 13 näkyy esimerkkitoteutuksessa käytetyn ohjelmointirajapinnan Design-välilehti.



KUVA 13. Esimerkkitoteutuksessa käytetyn ohjelmointirajapinnan Design-välilehti

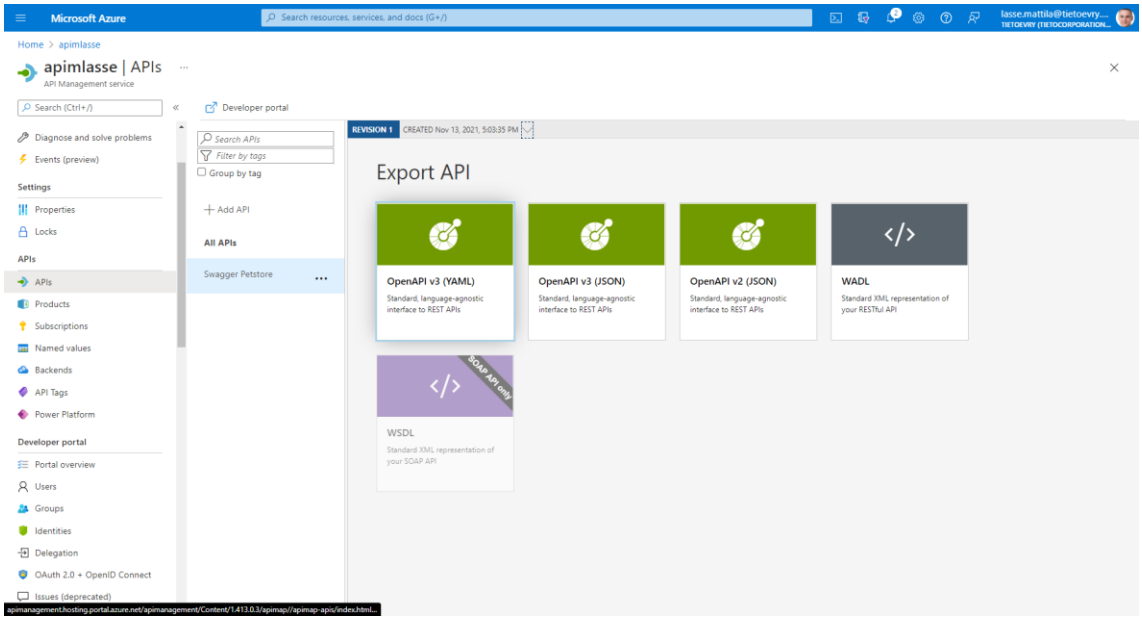
Onnistuneen ohjelmistorajapinnan lisäämisen jälkeen ohjelmointirajapinta täytyi viedä pois API Managementista OpenAPI-määritettynä, jotta siitä voitaisiin luoda asiakasohjelma. Tämä tehtiin painamalla kolmea pistettä ohjelmointirajapinnan

nimen vieressä ja painamalla Export-nappia. Kuvassa 14 on painettu kolme pistettä esimerkki toteutuksessa käytetyn ohjelmointirajapinnan vieressä.



KUVA 14. Esimerkkitoteutuksessa käytetyn ohjelmointirajapinnan vieressä olevien kolmen pisteen painamisen jälkeinen tilanne

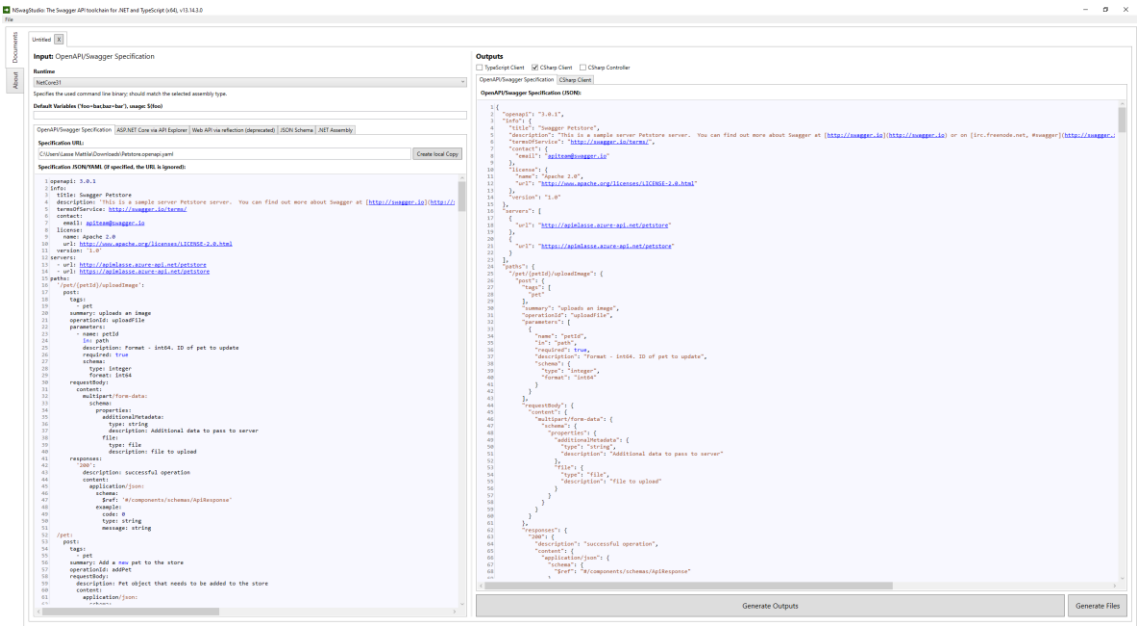
Export-napin painaminen avasi Export API-välilehden, josta valittiin haluttu OpenAPI-versio sekä tiedostomuoto. Halutun OpenAPI-määrityksen painamisen jälkeen selain latasi OpenAPI-määrityksen sisältävän tiedoston. Esimerkkitoteutuksessa OpenAPI-määritys vietiin API Managementista YAML-tiedostomuodossa ja OpenAPI-versiona kolme. Kuvassa 15 on Export-napin painamisen jälkeinen Export API-välilehti.



KUVA 15. Export API-välilehti Export-napin painamisen jälkeen

3.3 Asiakasohjelman luominen NSwagStudiolla

API Managementista viety OpenAPI-määrittely tuotiin NSwagStudioon. NSwagStudiossa on runsaasti erilaisia asetuksia liittyen esimerkiksi asiakasohjelman koodikielen valintaan sekä sen erilaisiin ominaisuuksiin. Tässä työssä haluttiin generoida C#-ohjelmointikieltä ja .Net Core 3.1-ohjelmointialustaa hyödyntävä asiakasohjelma asiakkaan toiveesta, joten valittiin asetuksista ne ja muuten käytettiin NSwagStudion asettamia oletusasetuksia. Specification URL-kenttään syötettiin API Managementista viedyn OpenAPI-määrityksen tiedostosijainti ja painettiin Create local Copy-nappia, jonka jälkeen määrittely tuli näkyviin Specification JSON/YAML-kenttään. Tämän jälkeen painamalla ensin Generate Outputs-nappia NSwagStudio loi C#-asiakasohjelman ja tämän jälkeen painamalla Generate Files-nappia NSwagStudio loi asiakasohjelman sisältävän C#-tiedoston NSwagStudion asetuksissa määriteltyyn tiedostosijaintiin. Kuvassa 16 on edellä mainittujen toimenpiteiden tuottama tilanne esimerkkitoteutuksessa.

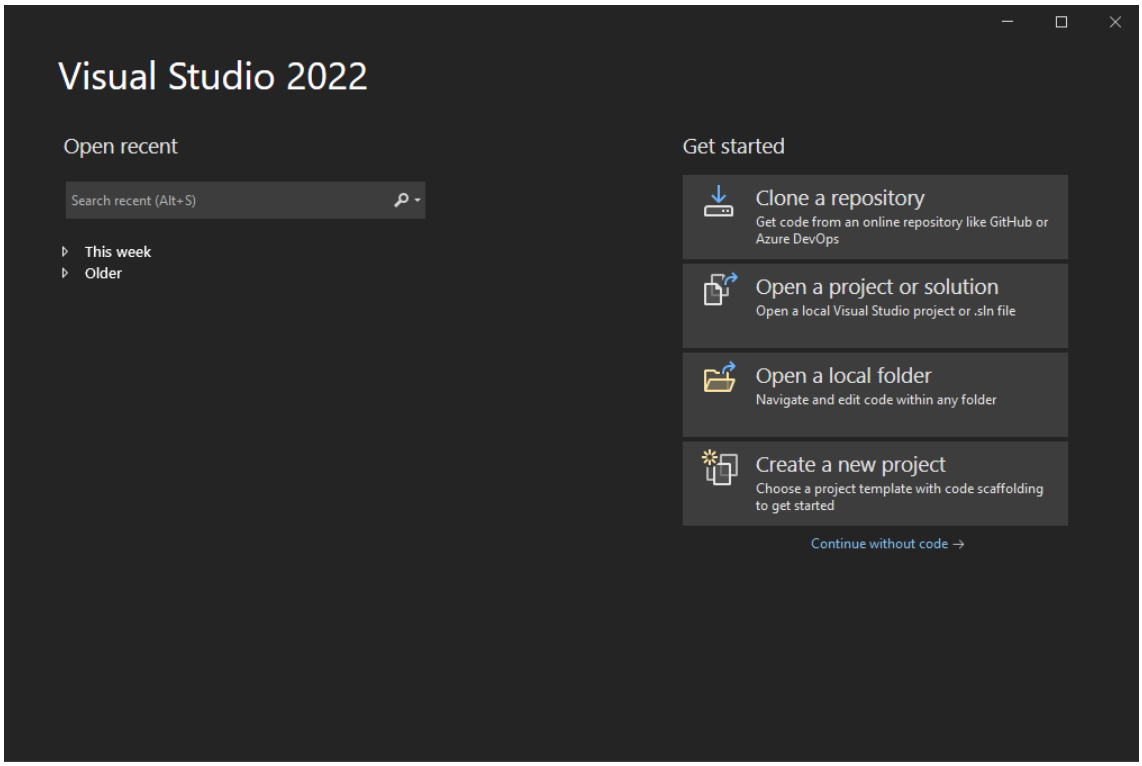


KUVA 16. API Managementista viedystä OpenAPI-määrityksestä NSwagStudiolla luotu C#-asiakasohjelma esimerkkitoteutuksessa

Varsinaisessa työssä käytetyt ja asiakkaalta saadut OpenAPI-määritykset loivat parhaimmillaan yli 30 000 koodirivin asiakasohjelmia. Esimerkkitoteutuksessa käytetystä OpenAPI-määrityksestä syntyi 1285 riviä koodia.

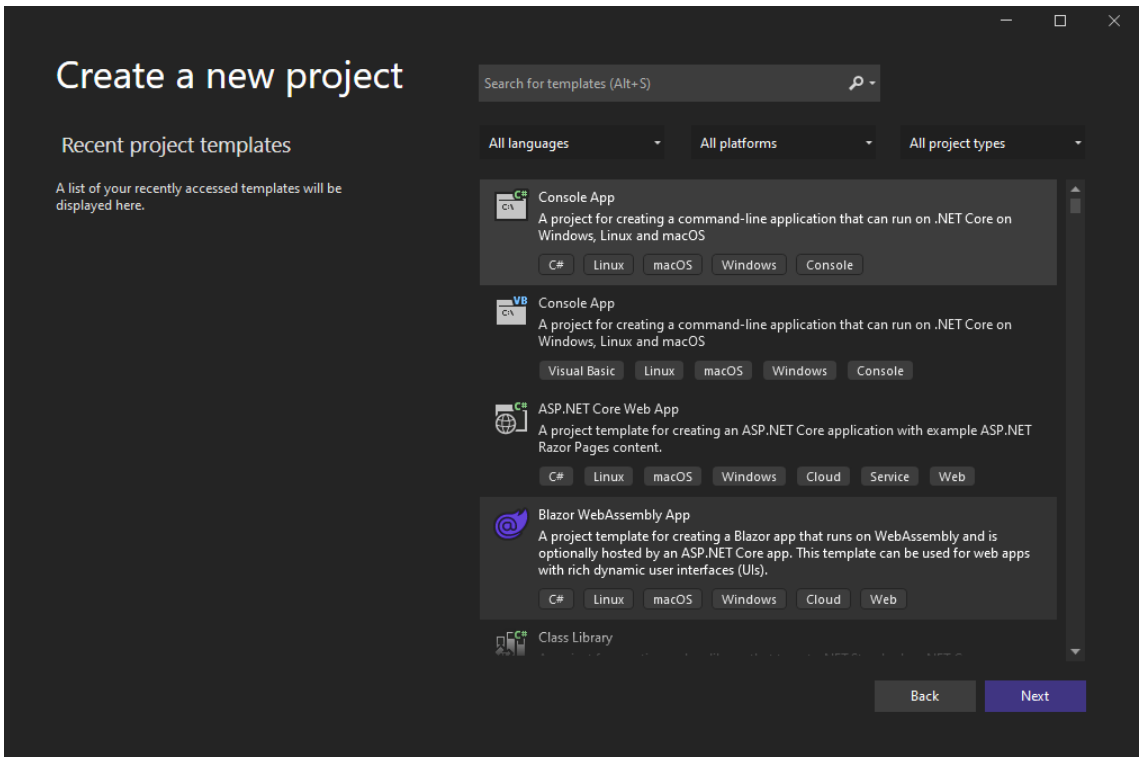
3.4 Konsoliohjelman luonti Visual Studiossa

API Managementin sekä luodun asiakasohjelman toiminnan testaamista varten täytyi luoda konsoliohjelma, johon luotu asiakasohjelma voitaisiin lisätä. Konsoliohjelma luotiin Visual Studio ohjelmointiympäristössä ja kehitysalustana käytettiin asiakkaan käytössä olevaa kehitysalustaa .Net Core 3.1. (Create a .NET console application using Visual Studio 2021) Konsoliohjelman luonti alkoi avaamalla Visual Studio. Sen jälkeen painettiin Create a new project-nappia, koska haluttiin luoda uusi projekti. Kuvassa 17 on Visual Studion aloitusnäkymä ohjelman avaamisen jälkeen.



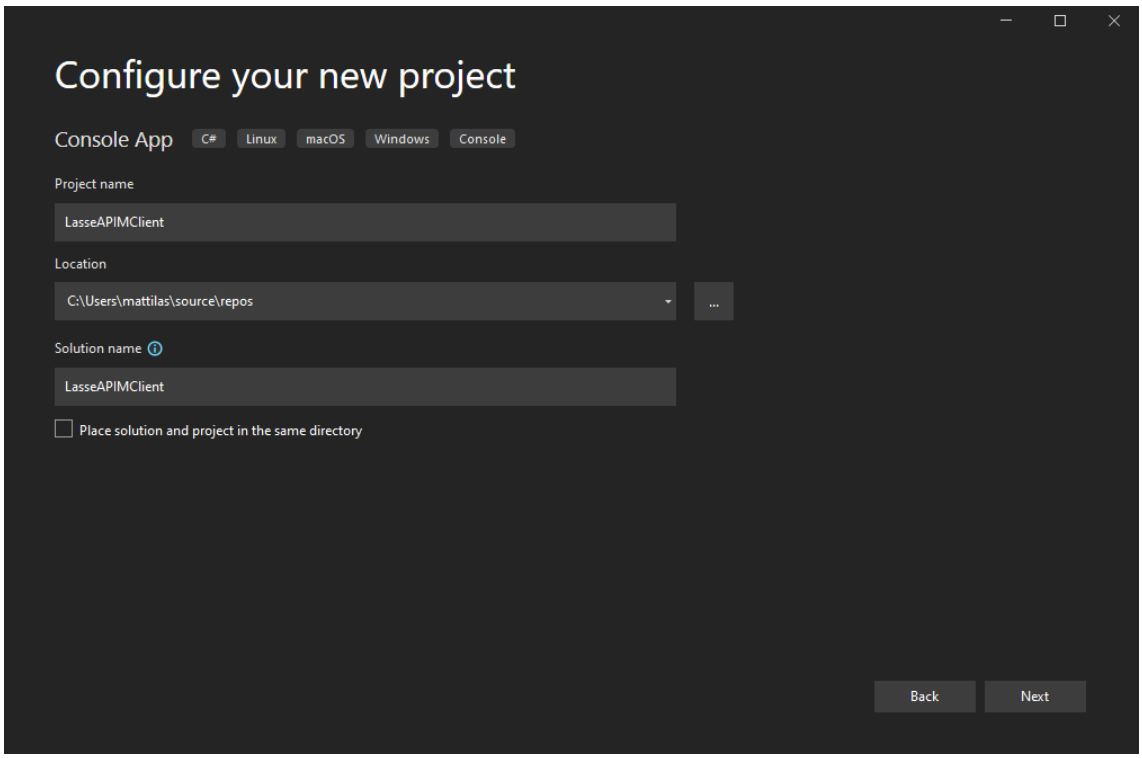
KUVA 17. Visual Studio aloitusnäkymä ohjelman avaamisen jälkeen

Create a new project-napin painamisen jälkeen valittiin haluttu projektipohja. Tässä työssä käytettiin C#-ohjelmointikieltä hyödyntävää Console App-projektipohjaa, joten valittiin se ja painettiin Next-nappia. Kuvassa 18 on Create a new project-napin painamisen jälkeinen näkymä.



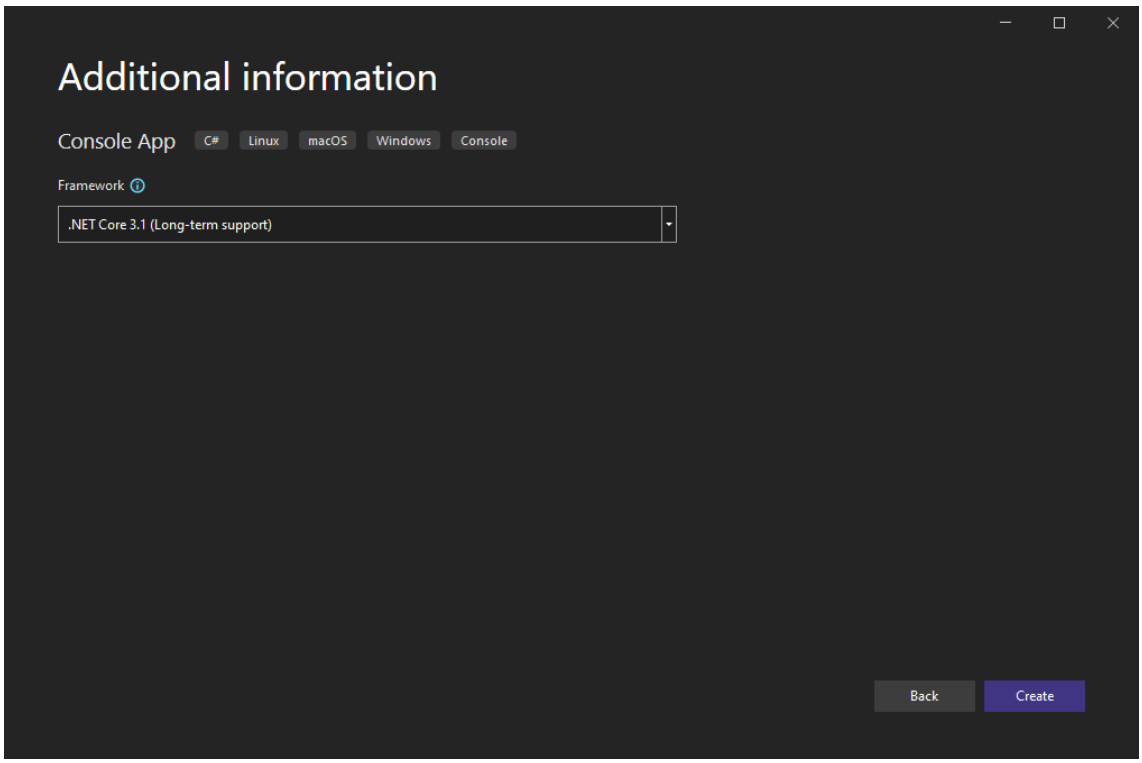
KUVA 18. Create a new project-napin painamisen jälkeinen näkymä

Projektipohjan valinnan jälkeen määriteltiin Configure your new project-näkymässä projektin nimi, tiedostosijainti sekä ratkaisun nimi. Tietojen määrittelyn jälkeen painettiin Next-nappia. Kuvassa 19 on esimerkkitoteutuksessa määritellyt tiedot Configure your new project-näkymässä.



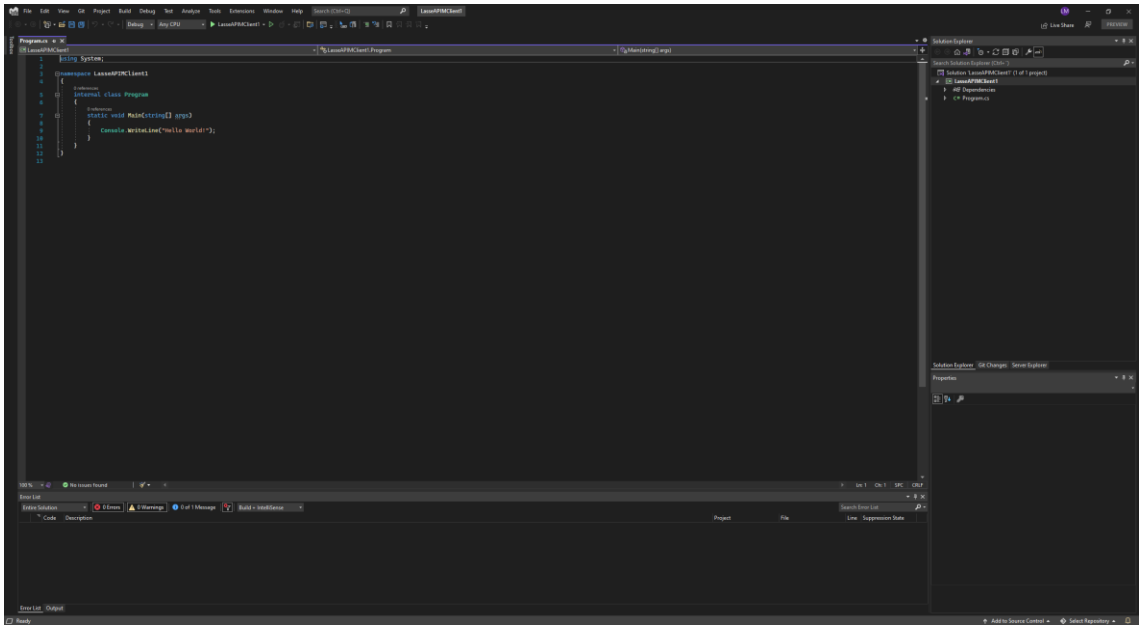
KUVA 19. Esimerkkitoteutuksessa määritellyt tiedot Configure your new project-näkymässä

Projektin tietojen määrittelyn jälkeen valittiin vielä projektin käyttämä ohjelmistokehys Additional information-näkymässä. Asiakkaan toiveesta ohjelmistokehykseksi valittiin .Net Core 3.1 ja sen jälkeen painettiin Create-nappia. Kuvassa 20 on Additional information-näkymä ja siinä valittuna .Net Core 3.1-ohjelmistokehys.



KUVA 20. Additional information-näkymä ja siinä valittuna .Net Core 3.1-ohjelmistokehys

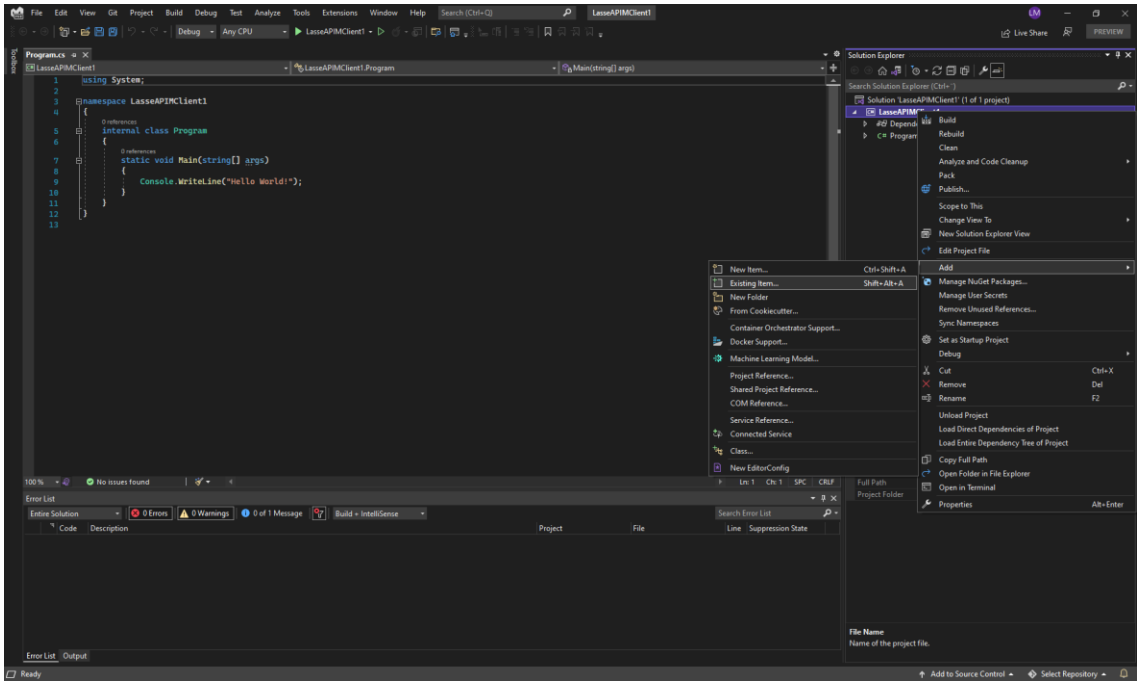
Create-napin painamisen jälkeen Visual Studio loi määritellyn konsoliohjelman. Kuvassa 21 on esimerkkitoteutuksessa luotu konsoliohjelma Visual Studiossa.



KUVA 21. Esimerkkitoteutuksessa luotu konsoliohjelma Visual Studiossa

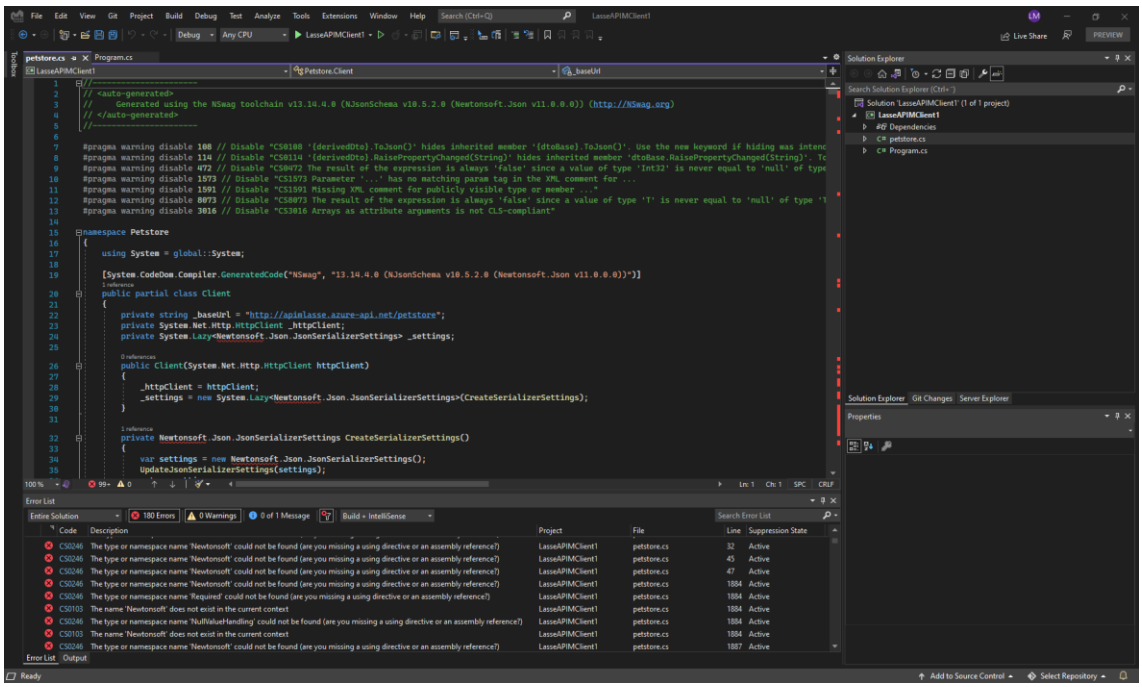
3.5 Asiakasohjelman luominen ja käyttäminen Visual Studiossa

Seuraavaksi Visual Studiossa luotuun konsoliprojektiin lisättiin NSwagStudiolla luotu C#-tiedosto. (Add files to a solution 2021) Kuvassa 22 projektiin lisättiin NSwagStudiolla luotu C#-tiedosto sen määrittämästä tiedostosijainnista.



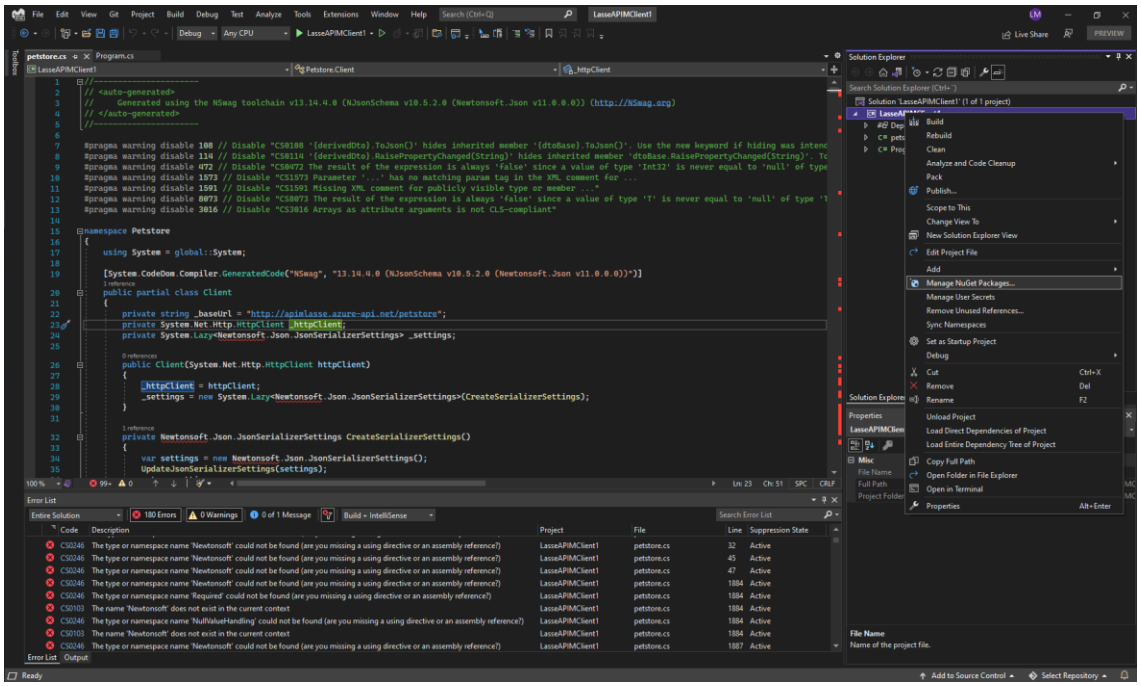
KUVA 22. NSwagStudiolla luodun tiedoston lisääminen konsoliohjelmaan

Tiedoston lisäämisen jälkeen NSwagStudiolla luotu C#-tiedosto aiheutti paljon virheitä, koska tiedosto hyödyntää Newtonsoft.Json NuGet-pakettia, jota ei ollut asennettu projektiin. Kuvassa 23 on tilanne NSwagStudiolla luodun C#-tiedoston lisäämisen jälkeen.



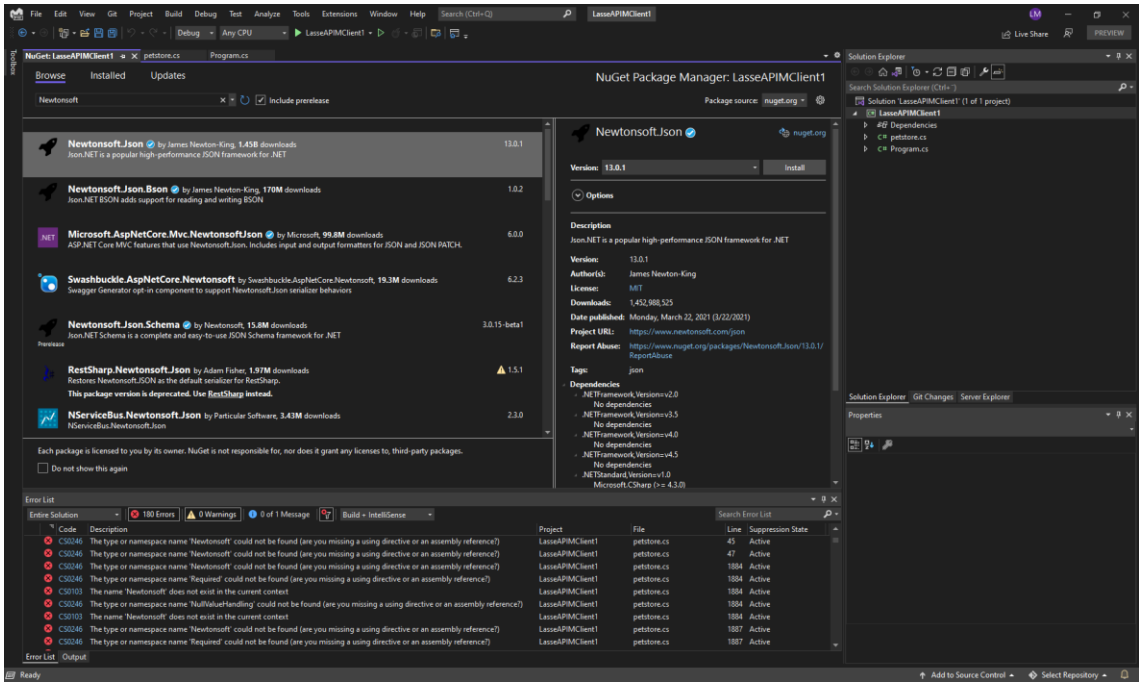
KUVA 23. Tilanne NSwagStudiolla luodun C#-tiedoston lisäämisen jälkeen

Puuttuva Newtonsoft.Json NuGet-paketti täytyi asentaa Visual Studio ohjelmointiympäristön avulla projektiin, jotta generoitua asiakasohjelmaa voitiin käyttää. (Install and use a package in Visual Studio 2021) Kuvassa 24 valitaan kohta Manage NuGet Packages, jotta päästään asentamaan tarvittava Newtonsoft.Json NuGet-paketti.



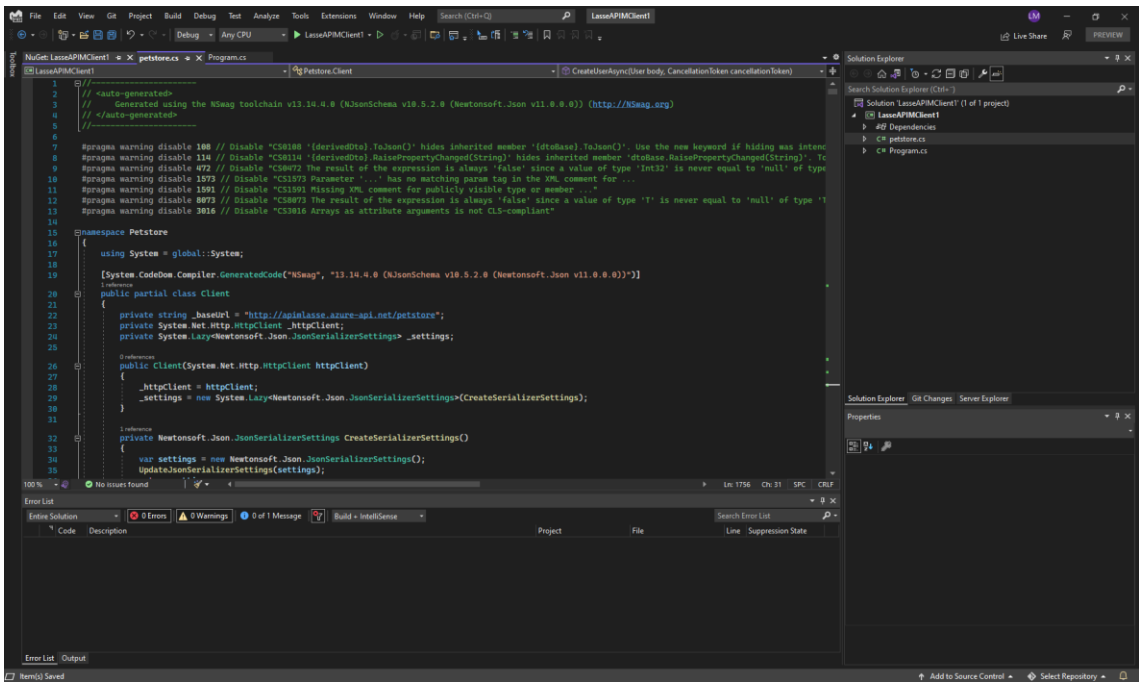
KUVA 24. Tilanne ennen Manage NuGet Packages-napin painamista

Manage NuGet Packages-napin painamisen jälkeen Browse-välilehden tekstikenttään kirjoitettiin NuGet-paketin nimi Newtonsoft ja valittiin kyseinen paketti. Tämän jälkeen painettiin Install-nappia. Kuvassa 25 on valittuna haluttu NuGet-paketti Newtonsoft.Json ennen Install-napin painamista.



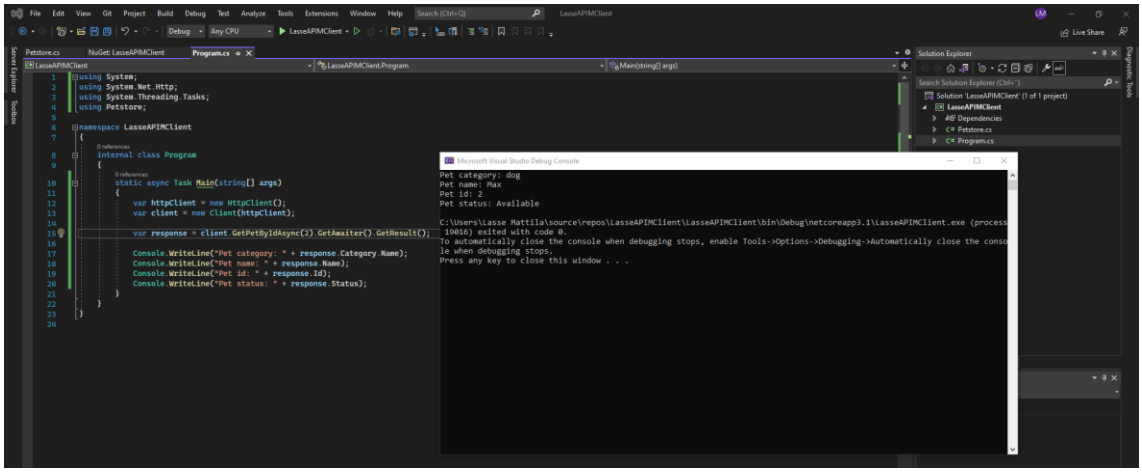
KUVA 25. Newtonsoft.Json NuGet-paketti valittuna ennen Install-napin painamista

Newtonsoft.Json-paketin asentaminen poisti kaikki virheet NSwagStudiolla luodusta C#-tiedostosta. Kuvassa 26 on näkymä NSwagStudiolla luodusta C#-tiedostosta ja sen virheettömyydestä Newtonsoft.Json-paketin asentamisen jälkeen.



KUVA 26. Näkymä NSwagStudiolla luodusta C#-tiedostosta ja sen virheettömyydestä Newtonsoft.Json-paketin asentamisen jälkeen Visual Studiossa

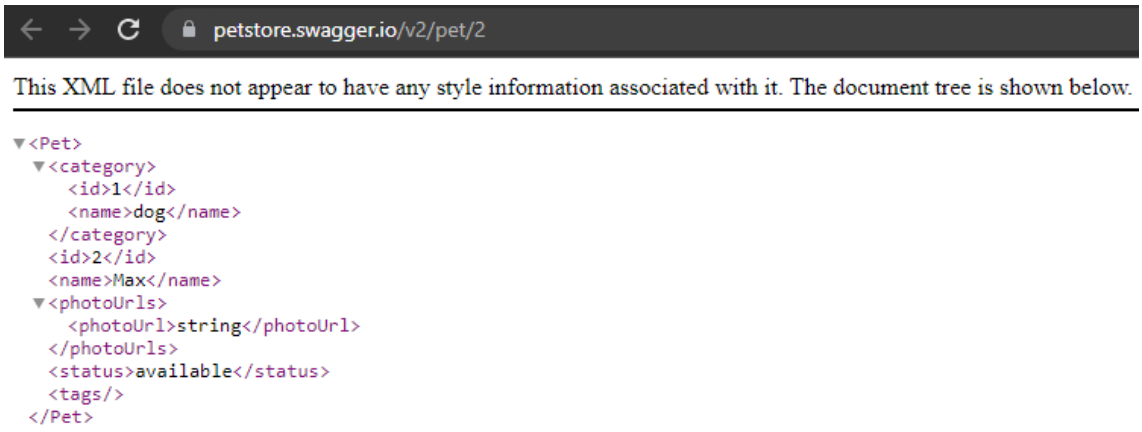
Tämän jälkeen konsoliohjelman Program.cs tiedostoon lisättiin viittaus NSwagStudiolla luodusta C#-tiedostosta ja luotiin haluttu kutsu hyödyntäen luotua asiakasohjelmaa. Kutsun toimivuuden testaamista sekä todentamista varten kutsun tulos kirjattiin konsoliin. Esimerkkitoteutuksessa Program.cs tiedostoon lisättiin viittaus NSwagStudiolla luodusta C#-tiedostosta Petstrore.cs ja käytettiin siinä määriteltyä GetPetByIdAsync-kutsua. (Suter 2017) Kuvassa 27 on esitettynä esimerkkitoteutuksessa tehdyt yllä olevat toimenpiteet sekä niiden tulokset konsoliohjelman käynnistämisen jälkeen.



KUVA 27. Visual Studiossa luotu konsoliohjelma ja sillä tehdyn kutsun tulos esimerkkitoteutuksessa

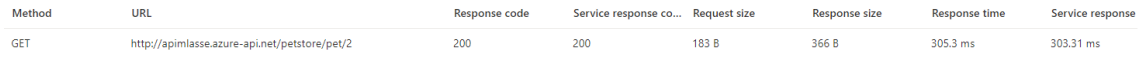
3.6 Tuloksen todentaminen ja varmentaminen

Konsoliohjelman konsoliin kirjaaman tuloksen oikeellisuuden varmentamiseksi varmistettiin, että suoraan selaimessa tehty kutsu tuottaa saman tuloksen. Selaimessa tehty kutsu tuotti saman tuloksen kuin konsoliohjelma, joten sen tuottama tulos oli oikea. Kuvassa 28 on esitettynä selaimessa tehdyn kutsun tulos.



KUVA 28. Suoraan selaimessa tehdyn esimerkkitoteutuksessa käytetyn kutsun tulos

Kutsun API Managementin käytön varmistamiseksi tarkastettiin Azure portalista API Management servicen Analytics-kohtaa, joka kerää tietoa API Management-tiin lisättyjen ohjelmointirajapintojen käytöstä. Analytiikassa näkyi konsoliohjelman tekemä kutsu ja sen tietoja. API Management servicen Analytics-kohdan tuloksen perusteella voidaan todeta, että kutsu meni myös onnistuneesti API Managementin läpi. Kuvassa 29 on esimerkkitoteutuksen API Management servicen Analytics-kohdassa tehty havainto onnistuneesta kutsusta ja sen tuloksista.



KUVA 29. Onnistuneen kutsun todentaminen Azure portalissa API Mananagement servicen Analytics-kohdassa

4 JOHTOPÄÄTÖKSET JA POHDINTA

Tässä työssä tutkittiin, miten OpenAPI-määrityksestä luodaan C#-asiakasohjelma sekä miten sillä luotu kutsu saadaan menemään API Managementin kautta asiakkaan olemassa olevalle palvelimelle. Tavoite saavutettiin tuomalla ensin OpenAPI-määrittely Azureen ja sen jälkeen viemällä se sieltä NSwagStudioon, jossa siitä luotiin C#-asiakasohjelma. Tämän jälkeen luotiin Visual Studiolla C#-konsoliohjelma, johon OpenAPI-määrityksestä luotu C#-asiakasohjelma lisättiin. Lopuksi konsoliohjelman avulla tehtiin kutsu API Managementin läpi palvelimelle ja varmennettiin sillä saadun tuloksen oikeellisuus. Toinen tärkeä työssä huomattu tulos oli asiakasohjelman luomisen tuoma tehokkuushyöty. Tässä työssä käytetyistä OpenAPI-määrityksistä asiakasohjelman luonti tuotti parhaimmillaan yli 30 000 koodiriviä. Vastaavan koodimäärän kirjoittamiseen käsin menisi huomattavasti enemmän aikaa kuin OpenAPI-määrityksestä luontia hyödyntämällä. Työtä ja sen tuloksia voidaan pitää erittäin luotettavina, koska työssä hyödynnettiin useita erilaisia OpenAPI-määrityksiä ja niiden versioita.

Tätä työtä voisi lähteä jatkokehittämään esimerkiksi selvittämällä, miten kutsujen oikeuksien tarkastamiseen voisi parhaiten hyödyntää API Managementin ominaisuuksia sekä sitä, miten se toteutettaisiin asiakasohjelman puolella.

LÄHTEET

About API Management. Microsoft. Luettu 28.11.2021.
https://docs.microsoft.com/en-us/azure/api-management/api-management-key-concepts

Add files to a solution. Microsoft. Luettu 20.11.2021.
https://docs.microsoft.com/en-us/visualstudio/ide/creating-solutions-and-projects?view=vs-2022#add-files-to-a-solution

API Management. Microsoft. Luettu 6.11.2021.
https://azure.microsoft.com/en-us/services/api-management/#overview

Basic Structure. Swagger. Viitattu 13.11.2021.
https://swagger.io/docs/specification/basic-structure/

Create a .NET console application using Visual Studio. Microsoft. Luettu 13.11.2021.
https://docs.microsoft.com/en-us/dotnet/core/tutorials/with-visual-studio?pivots=dotnet-5-0

Create a new Azure API Management service instance by using the Azure portal. Microsoft. Luettu 8.11.2021.
https://docs.microsoft.com/en-us/azure/api-management/get-started-create-service-instance

Escott, E. 2020. The pro's of code generation. Luettu 20.11.2021.
https://codebots.com/app-development/what-is-a-code-generator

Gartner Forecasts Worldwide Public Cloud End-User Spending to Grow 23% in 2021. Gartner. Luettu 20.11.2021.
https://www.gartner.com/en/newsroom/press-releases/2021-04-21-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-grow-23-percent-in-2021

Import an OpenAPI specification. Microsoft. Luettu 6.11.2021.
https://docs.microsoft.com/en-us/azure/api-management/import-api-from-oas

Install and use a package in Visual Studio. Microsoft. Luettu 13.11.2021.
https://docs.microsoft.com/en-us/nuget/quickstart/install-and-use-a-package-in-visual-studio

Microsoft Azure portal. Microsoft. Viitattu 24.11.2021.
https://azure.microsoft.com/en-us/features/azure-portal/

NSwag: The Swagger/OpenAPI toolchain for .NET, ASP.NET Core and TypeScript. NSwag Project. Luettu 6.11.2021.
https://github.com/RicoSuter/NSwag/blob/master/README.md

NSwagStudio. NSwag Project. Luettu 6.11.2021.
https://github.com/RicoSuter/NSwag/wiki/NSwagStudio

Overview. Microsoft. Luettu 28.11.2021.
https://docs.microsoft.com/en-us/azure/api-management/api-management-key-concepts#overview

Top benefits of cloud computing. Microsoft. Luettu 20.11.2021.
https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/#benefits

Suter, R. 2017. NSwag v11 Tutorial: How to integrate NSwag into your ASP.NET Core Web API project. Youtube-video. Julkaistu 10.2.2017. Viitattu 20.11.2021.
https://www.youtube.com/watch?v=lF9ZZ8p2Ciw

Swagger Petstore. Swagger. Luettu 13.11.2021.
https://petstore.swagger.io/

Visual Studio for Windows. Microsoft. Luettu 13.11.2021.
https://visualstudio.microsoft.com/

What is .NET? Microsoft. Luettu 13.11.2021.
https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet

What is Azure? Microsoft. Luettu 6.11.2021.
https://azure.microsoft.com/en-us/overview/what-is-azure/

What is cloud computing? Microsoft. Luettu 20.11.2021.
https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/

What is the Azure portal? Microsoft. Luettu 24.11.2021.
https://docs.microsoft.com/en-us/azure/azure-portal/azure-portal-overview#what-is-the-azure-portal

What is the OpenAPI Specification? OpenAPI Initiative. Luettu 6.11.2021.
https://spec.openapis.org/oas/latest.html