KARELIA-AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutus

Antti Liimatta

Integraatioalustojen vertailu

Opinnäytetyö
Toukokuu 2019

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Toukokuu 2019
Tietojenkäsittelyn koulutus
Alempi ammattikorkeakoulututkinto
Tikkarinne 9
80200 JOENSUU
+358 13 260 600 (vaihde)

Tekijä
Antti Liimatta

Integraatioalustojen vertailu

Toimeksiantaja:
Broman Group Oy

Tiivistelmä

Tässä opinnäytetyössä vertailtiin kolmea yhdessä Broman Group Oy:n IT-osaston kanssa valittua dataintegraatioalustaa. Testattavat alustat olivatt Anypoint Platform, Talend ja HiQ Frends. Lähteinä käytettiin Internet-artikkeleita, videoita ja muita luotettavia lähteitä.

Testaukseen käytettiin kahta testitapausta, jotka olivat samat jokaiselle dataintegraatio-alustalle. Testitapausten tavoitteena oli testata dataintegraatioalustojen yhteensopivuutta Broman Group Oy:n järjestelmien ja teknologioiden kanssa.

Opinnäytetyön lopussa luotiin testien perusteella yhteenveto ja esitettiin vertauskohtien tulokset taulukoissa luettavuuden helpottamiseksi. Lopuksi valittiin Broman Group Oy:lle sopivin integraatioalusta, HiQ Frends. Testeissä pyrittiin mahdollisimman objektiiviseen vertailuun.

Kieli

Suomi

Sivuja 40

Asiasanat

dataintegraatio, ESB, ETL

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
May 2019
Degree Programme in Business IT
Bachelor's Thesis
Tikkarinne 9
FI 80200 JOENSUU
FINLAND
Tel. +358 13 260 600 (switchboard)

Author (s)
Antti Liimatta

Title
Comparison of Data Integration Platforms

Commissioned by: Broman Group Oy

Abstract

The purpose of this thesis was to compare three data integration platforms chosen with the IT department of Broman Group Oy. The thesis describes the concept of data integration, in general, in order to make the testing and comparison processes more understandable later in the study. The platforms tested were Anypoint Platform, Talend and HiQ Frends. Articles, videos and other reliable sources from the Internet were used as sources.

Two test cases were used for testing. These were the same for each data integration platform. The aim of the test cases was to test the compatibility of data integration platforms with Broman Group Oy's systems and technologies.

At the end of the thesis, the results of the tests were summarized, and the final verdict was given on each of the tested categories. The results were presented in graphs for easier readability. Finally, the most appropriate integration platform was chosen based on the needs of Broman Group Oy. The chosen integration platform was HiQ Frends.

Language

Finnish

Pages 40

Keywords

dataintegration, ESB, ETL

Sisältö

Lyhenteet

ESB	Enterprise Service Bus. Järjestelmä- ja dataintegraatioalustojen arkkitehtuurimalli.
ETL	Extract, Transform, Load. Järjestelmä- ja dataintegraatioalustojen arkkitehtuurimalli.
REST	Representational State Transfer. Arkkitehtuurimalli ohjelmistorajapintojen toteuttamiseen.
SOAP	Simple Object Access Protocol. Verkkoyhteyksissä käytettävä tietoliikenneprotokolla.
WSDL	Web Service Description Language. Verkkopalveluiden kuvaamiseen käytettävä XML-pohjainen kieli.
HTTP	Hypertext Transfer Protocol. Internetpalvelinten tiedonsiirtoon käytettävä protokolla.
JMS	Java Message Service. Rajapinta Java-ohjelmien väliseen kommunikaatioon.
MQTT	Message Queuing Telemetry Transport. Viestintäprotokolla verkkopalveluiden väliseen kommunikaatioon.
AMQP	Advanced Message Queuing Protocol. Viestijonoprotokolla ohjelmistojen väliseen viestintään.
UDP	User Datagram Protocol. Protokolla järjestelmien väliseen tiedonsiirtoon verkon yli.
API	Application Programming Interface. Ohjelmistorajapinta, joka mahdollistaa ohjelmistojen välisen kommunikaation.
JSON	JavaScript Object Notation. Datapareihin perustuva tiedostomuoto tiedonvälitykseen.
YAML	YAML Ain’t Markup Language (rekursiivinen lyhenne). Merkintäkieli konfiguraatiotiedostojen kirjoitukseen.

XML	Extensible Markup Language. Merkintäkielistandardi. Käytetään usein tiedonvälitykseen käytettävien tiedostojen muotoiluun.
SQL	Structured Query Language.
ID	Identifier. Digitaalinen tunniste.

1 Johdanto

Tässä opinnäytetyössä vertailen kolmea dataintegraatioalustaa Broman Group Oy:lle. Tutkimuksen tarkoituksena on valita Broman Group Oy:lle heidän järjestelmiinsä mahdollisimman yhteensopiva dataintegraatioalusta nykyisten bash-skriptien korvaajaksi. Vertaan dataintegraatioalustojen projektinluontiprosessia, datankäsittelyformaattia, projektin käyttöönottoprosessia, sekä yhteensopivuutta nykyisiin järjestelmiin. Lisäksi otan kantaa kehitysympäristöjen helppokäyttöisyyteen, opittavuuteen, monipuolisuuteen ja järjestelmän hinnoitteluun.

Testattavat dataintegraatioalustat ovat amerikkalaiset Mulesoft Anypoint ja Talend ESB, sekä suomalainen Hiq Frends. Nämä järjestelmät valittiin vertailuun mainostamiensa ominaisuuksien perusteella Broman Group Oy:n IT-osaston toimesta. Toteutan jokaisella dataintegraatioalustalla kaksi testitapausta, joiden tuloksia vertailemalla valitsen mielestäni sopivimman dataintegraatioalustan Broman Group Oy:lle.

Valitsin tämän aiheen harjoittelujaksoni aikana Broman Group Oy:llä. Esimieheni otti puheeksi uuden dataintegraatioalustan hankinnan olevan lähellä ja ehdotti eri järjestelmien vertailua opinnäytetyöni aiheeksi. Mielestäni aihe oli itselleni sopiva, vaikka en ollut aikaisemmin tutustunut dataintegraatioalustoihin millään tavalla. Olin kuitenkin valmis oppimaan, sillä näiden järjestelmien käsittely tulee olemaan työni kannalta tärkeää tulevaisuudessa. Aloitin lukemalla aiheeseen liittyviä julkaisuja ja ohjetekstejä Internetissä. Katsoin myös luentoja Youtubesta, sekä dataintegraatioalustoja tarjoavien yritysten omilta verkkosivuilta. Viittaan useaan näistä teksteistä ja videoista tässä opinnäytetyössä.

Broman Group Oy:n lisäksi tästä opinnäytetyöstä on hyötyä kaikille, jotka haluavat lisätietoa markkinoiden johtavista dataintegraatioalustoista ja niiden ominaisuuksista. Yritän olla mahdollisimman objektiivinen ja puolueeton testitapausten suoritusten aikana, sekä lopullisessa vertailussa. Dataintegraatioon liittyvistä teknologioista minulla on eniten kokemusta Java-ohjelmoinnista. Testausprosessin suunnittelin yhdessä esimieheni kanssa, jotta testit vastaisivat mahdollisimman

hyvin Broman Group Oy:n tarpeita. Opinnäytetyön aikana tulen viittaamaan usein dataintegraatioalustoihin sanalla ”integraatioalusta”.

2 Mitä on dataintegraatio?

Wikipedia määrittelee dataintegraation seuraavasti: ”Dataintegraatiolla tarkoitetaan tiedon muuntamista ja kuljettamista tietojärjestelmästä toiseen.” (Wikimedia Foundation 2013). Kyseessä on siis melko laaja käsite, mikä voi tarkoittaa mitä tahansa prosessia, missä siirretään dataa ohjelmistojen välillä. Dataintegraatioratkaisut jaotellaan pääasiallisesti kolmeen integraatiomalliin: on-premises-järjestelmien välisiin integraatioihin, pilvi-integraatioihin ja hybridi-integraatioihin (Itewiki 2018). On-premises-järjestelmillä tarkoitetaan tässä yhteydessä yrityksen fyysisillä palvelimilla sijaitsevia ohjelmistoja. Pilvi-integraatio taas tarkoittaa pilvipalveluihin kytkettyjen ohjelmistojen välistä integraatiota. Hybridi-integraatiossa integraatiototeutukset voivat toimia paikallisella palvelimella, mutta tämä palvelin on yhteydessä pilvessä toimiviin komponentteihin.

Dataintegraatioita voidaan toteuttaa seuraavilla toteutustavoilla:

- Point-to-point.
- Hub and spoke.
- Enterprise service bus (ESB).

Kaksi ensimmäistä integraatiotapaa hyödyntää ETL-mallia. ETL on lyhenne englanninkielisistä sanoista extract (nouda), transform (muuta) ja load (lataa). Kaikki tätä mallia hyödyntävät integraatioratkaisut on suunniteltu noutamaan dataa erilaisista rajapinnoista, muuntamaan sitä ja lataamaan se edelleen kohderajapintaan, tiedostoon tai järjestelmään. (Zhao 2010). ESB on tästä kehittyneempi integraatioperiaate, sillä se voi sisältää pelkän siirron ja muunnon lisäksi logiikkaa, sekä pelkän datan siirron ulkopuolelle sijoittuvia operaatioita. Lisäksi ETL soveltuu paremmin suurien datamäärien siirtoon yksittäisissä kuormissa, kun taas ESB on tarkoitettu siirtämään dataa pienissä määrissä useiden sovellusten välillä jatkuvasti (Harvey 2015).

2.1 Point-to-point

Point-to-point-integraatiossa dataintegraatiojärjestelmä ottaa vastaan dataa ohjelmalta, tekee tarvittavat muutokset ja lähettää datan edelleen vastaanottavalle ohjelmalle. Suomeksi point-to-point tarkoittaakin karkeasti ”pisteestä pisteeseen”. (itewiki 2018.)
Tämän mallin heikkoutena voidaan pitää sen joustamatonta suunnittelumallia ja rajallista hyödyllisyyttä suuren mittakaavan integraatioprojekteissa. Yksittäisiin integraatiotapauksiin point-to-point on helppo ja nopeasti toteutettava integraatiomalli, mutta suuri lukumäärä yksittäisiä point-to-point integraatioita aiheuttaa nopeasti vaikeuksia ylläpidossa, varsinkin jos järjestelmä on dokumentoitu puutteellisesti. Muutos yhdessä järjestelmässä kaataa helposti koko integraatioverkoston. Itse suosittelen tämän mallisia integraatioratkaisuja yksityishenkilöiden omiin projekteihin, mutta en ammattitason järjestelmiin.

2.2 Hub and spoke

Hub and spoke-integraatiot ja point-to-point-integraatiot toimivat lähes samalla periaatteella. Hub and spoke eroaa point-to-point-integraatioista siten, että data ei automaattisesti siirry tietylle vastaanottavalle ohjelmalle, vaan viestit voidaan lähettää tietylle vastaanottajalle, datan sisällön mukaan. Viestit voidaan myös lähettää useammalle, kuin yhdelle vastaanottajalle. Tämän mallin heikkoutena voidaan pitää järjestelmän keskittynyttä arkkitehtuuria, mikä johtaa integraatiooperaatioiden epäonnistumiseen virhetilanteissa. Lisäksi tätä mallia käyttäviä järjestelmiä on vaikea skaalata tarvittaessa tukemaan suurempaa viestimäärä ilman fyysisen palvelimen päivittämistä. (Conlay 2013.)

2.3 Enterprise Service Bus (ESB)

ESB on kehittyneempi versio hub and spoke-mallista. Pelkän yksittäisen välittäjän sijaan ESB-järjestelmissä käytetään reititinjärjestelmää, johon kaikki ulkopuoliset yhteydet liitetään. Tämä reititin (bus) on vastuussa datan hausta, muunnoksista ja reitittämisestä. Reititinjärjestelmä toimii hajautetusti ja sisältää toisiinsa liitettyjä ohjelmistomoduuleja, joiden avulla reititin voi toimia joustavasti. Reititin voi esimerkiksi sisältää erilaisia datan muuntamiseen liittyviä moduuleja, joita käytetään aina tarpeen mukaan, riippuen käsiteltävän datan tyypistä, sekä dataa vastaanottavasta rajapinnasta. Nämä moduulit toimivat järjestelmässä omina prosesseinaan, joten yhden prosessin kaatuminen ei aiheuta koko järjestelmän toiminnan estymistä. ESB-malli mahdollistaakin joustavamman virheenhallinnan, kuin aikaisemmin mainitut integraatiomallit. (Ciurana 2007.)

3 Dataintegraatioalustat

Tässä osiossa kerron myöhemmin vertailtavista dataintegraatioalustoista, sekä muutamasta muusta, yleisesti käytössä olevasta integraatioalustasta. Tarkoitukseni on antaa mahdollisimman tarkka kuva siitä, mitä integraatiomallia nämä integraatioalustat noudattavat, mitä teknologiaa ne sisältävät, millainen yritys niitä markkinoi, sekä millaiselle asiakkaalle kyseisiä integraatioalustoja markkinoidaan. Käyn myös läpi projektinluonnin prosessin jokaisen alustan kehitysympäristöllä.

3.1 Talend

Talend on saman nimisen kalifornialaisyrityksen markkinoima ESB-dataintegraatioalusta. Talend julkaistiin vuonna 2006 nimellä Talend Open Studio. Talendista on myöhemmin julkaistu kaupallinen versio, joka sisältää operaatioidentarkkailujärjestelmän pilvessä, sekä muita suurille yrityksille hyödyllisiä ominaisuuksia. Tässä opinnäytetyössä keskityn kuitenkin vain vapaasti käytettävissä olevaan,

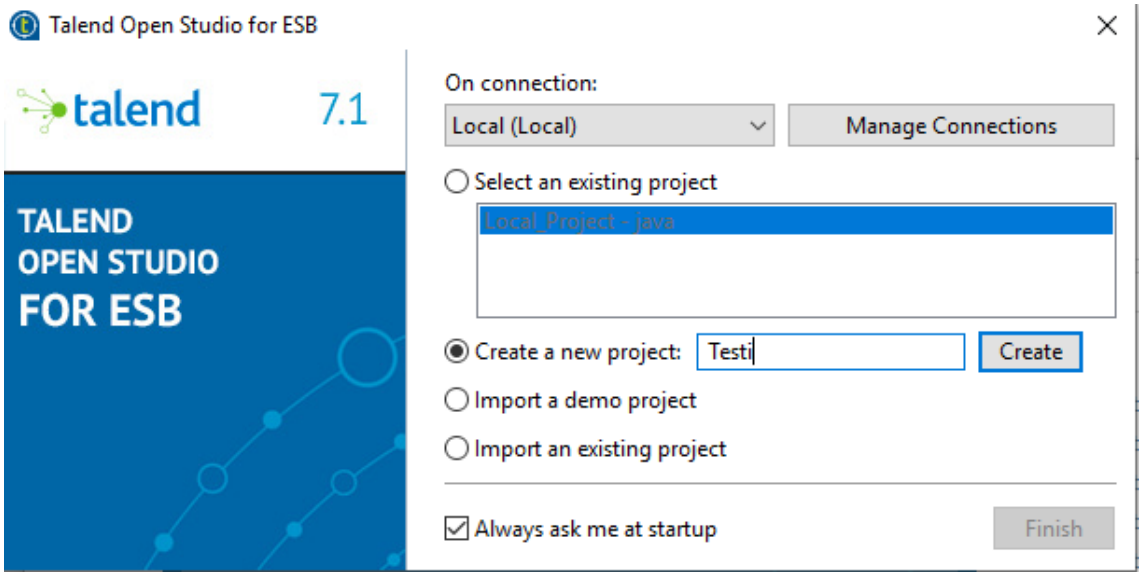
vapaan lähdekoodin versioon. Talendin vapaan lähdekoodin alustaa käyttää tällä hetkellä pohjoismaissa yli 150 yritystä, kun tilausmallilla markkinoitavaa suljetun lähdekoodin versiota käyttää 12 yritystä (Fredriksson 2018).

Talend tukee suurinta osaa suosituista rajapintateknologioista, joista Broman Group Oy:n kannalta tärkeimmät ovat REST, SOAP, OAuth ja WSDL. Lisäksi Talend tukee suurta määrää siirtoprotokollia, kuten HTTP, JMS, MQTT, AMQP ja UDP. (Talend 2018.)

Talend Open Studio For ESB on saatavilla Talendin kotisivuilta Windows- ja OSX-käyttöjärjestelmille vapaalla Apache-lisenssillä. Kyseessä on suositun Eclipse-kehitysalustan pohjalta rakennettu kehitysympäristö, jolla käyttäjä pystyy luomaan ja testaamaan Talend-projekteja. Kehitysympäristö pohjautuu Java-ohjelmointikieleen. Ladattavaan pakettiin kuuluu myös runtime-ohjelma, johon studiolla rakennetut integraatiototeutukset asennetaan ajettavaksi.

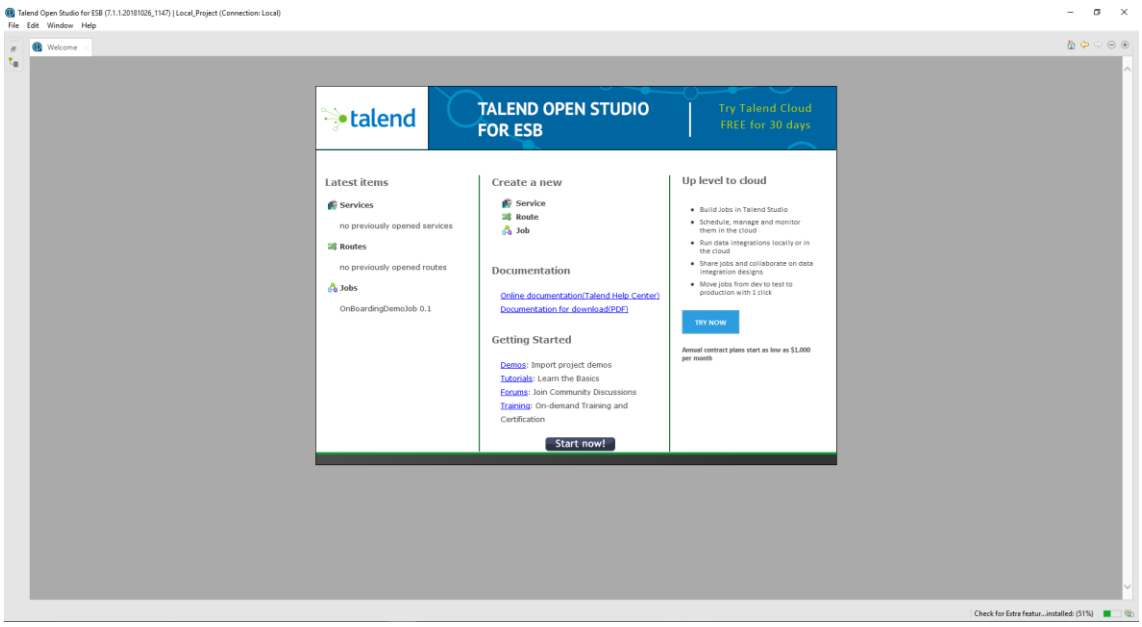
3.2 Talend-projektin luonti

Talend-projektin luonti aloitetaan käynnistämällä Talend Studio for ESB. Käynnistyessään Studio pyytää käyttäjää luomaan uuden projektin, tuomaan projektin ulkoisesta tiedostosta, tai valitsemaan aikaisemmin luodun projektin listasta (Kuvio 1). Uutta projektia luodessa projektille annetaan nimi, joka lisätään olemassa olevien projektien listaan. Tämän jälkeen projekti voidaan avata valitsemalla se listasta ja painamalla oikean alanurkan painiketta.

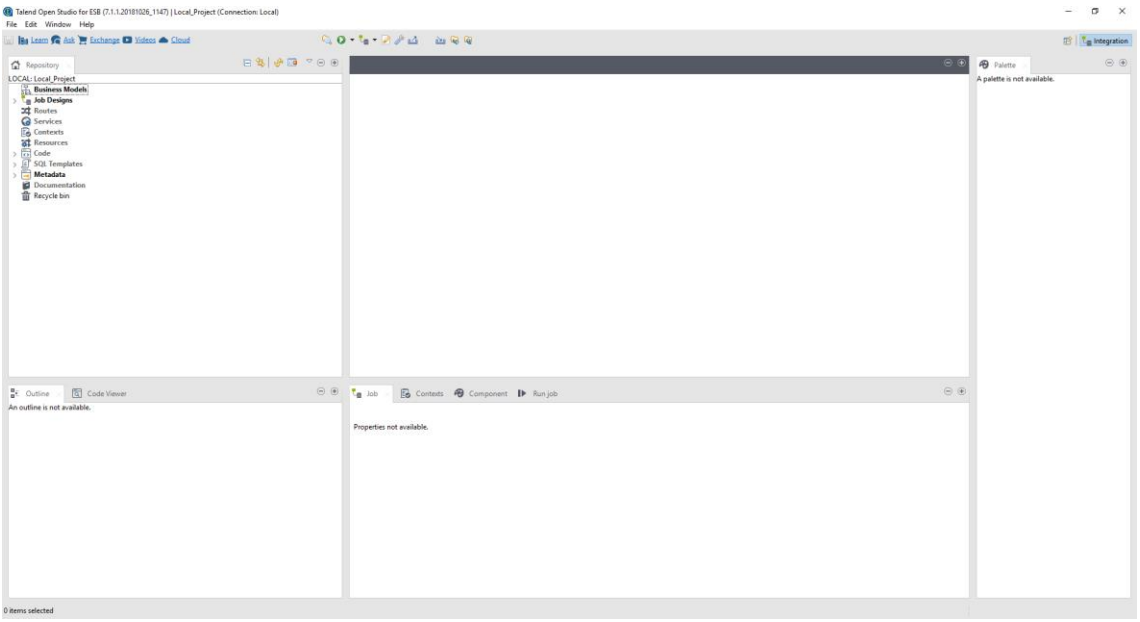


Kuvio 1. Projektin luonti.

Projektin avaamisen jälkeen Studio avaa tervetuloa-ikkunan (Kuvio 2), josta käyttäjä voi palata aikaisemmin muokkaamiinsa kohteisiin, luoda uusia komponentteja, lukea dokumentaatiota ja päivittää lisenssinsä. Tervetuloa-ikkunan voi sulkea vasemmasta ylänurkasta, välilehden nimen vierestä. Ohjelma siirtyy tällöin työtilaan (Kuvio 3). Työtila koostuu useista ikkunoista, jotka ovat melko samanlaisia Eclipse-kehitysympäristön kanssa. Vasemmalta löytyvät paneelit kertovat projektin sisällön, keskellä olevat kangas, sekä hienosäätöikkunat, sekä oikealla paletti, joka sisältää kankaalle lisättävät komponentit.

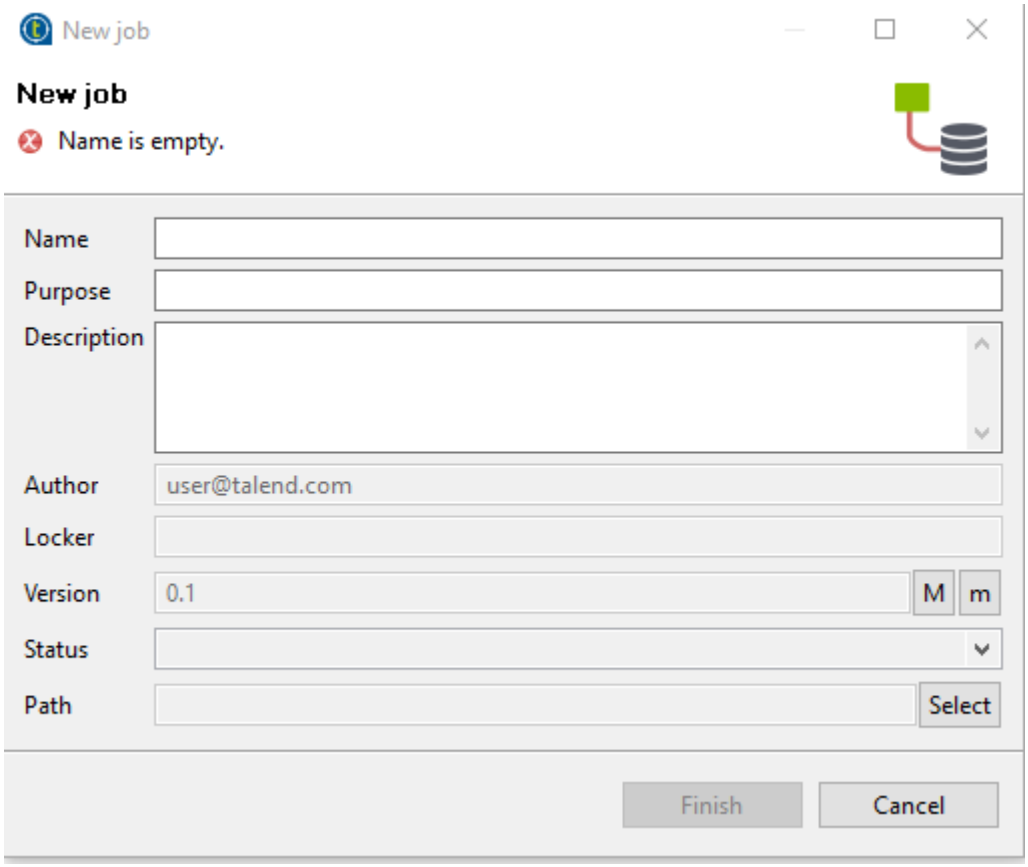


Kuvio 2. Aloitusruutu.

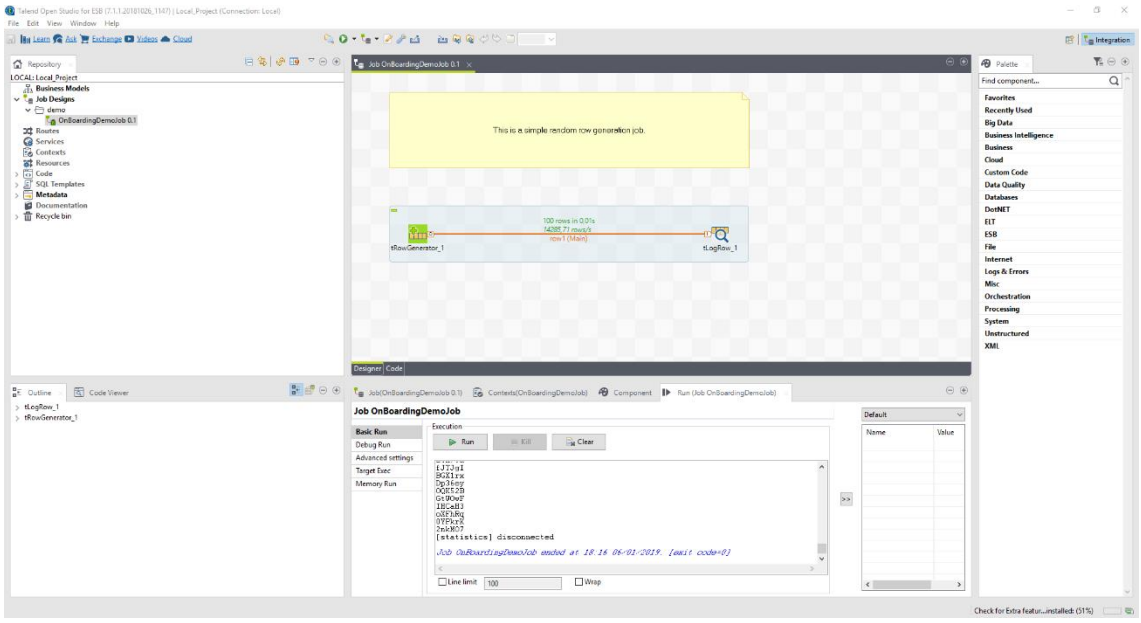


Kuvio 3. Käyttöliittymä.

Uuden Talend-työn luonti aloitetaan klikkaamalla Repository-paneelin Job Designs-valikkoa oikealla hiiren näppäimellä. Avautuvassa ikkunassa uudelle työlle annetaan nimi, sekä valinnaisesti muita kuvaavia ominaisuuksia (Kuvio 4). Tämän jälkeen käyttäjä voi rakentaa haluamansa työn käyttäen paletissa saatavilla olevia moduuleja yhdistämällä. (Kuvio 5).



Kuvio 4. Projektin luontiasetukset.



Kuvio 5. Projektikaavio

3.3 MuleSoft Anypoint

Anypoint Platform on kalifornialaisen MuleSoft-yrityksen markkinoima ESB-integraatioalusta. Mulesta on saatavilla vain maksullinen versio, johon on saatavilla 30 päivän ilmainen kokeilujakso. Kehitys tapahtuu Eclipse-kehitysympäristöön pohjautuvalla Anypoint Studio-ohjelmalla, joka on saatavilla Windows- ja OSX- käyttöjärjestelmille. Anypoint Studiolla luodut integraatiototeutukset voidaan siirtää testaamisen jälkeen Mulen pilvipalveluun (Kuvio 6), jossa käyttäjä voi myös suunnitella API-rajapintoja. Tässä opinnäytetyössä en ota kantaa Anypointin pilvipalveluihin, sillä ne eivät olleet tekemieni testien kannalta oleellisia. Anypoint Platformista on saatavilla paikalliselle palvelimelle asennettava versio, joka toimii Dockerin avulla (MuleSoft 2018.)

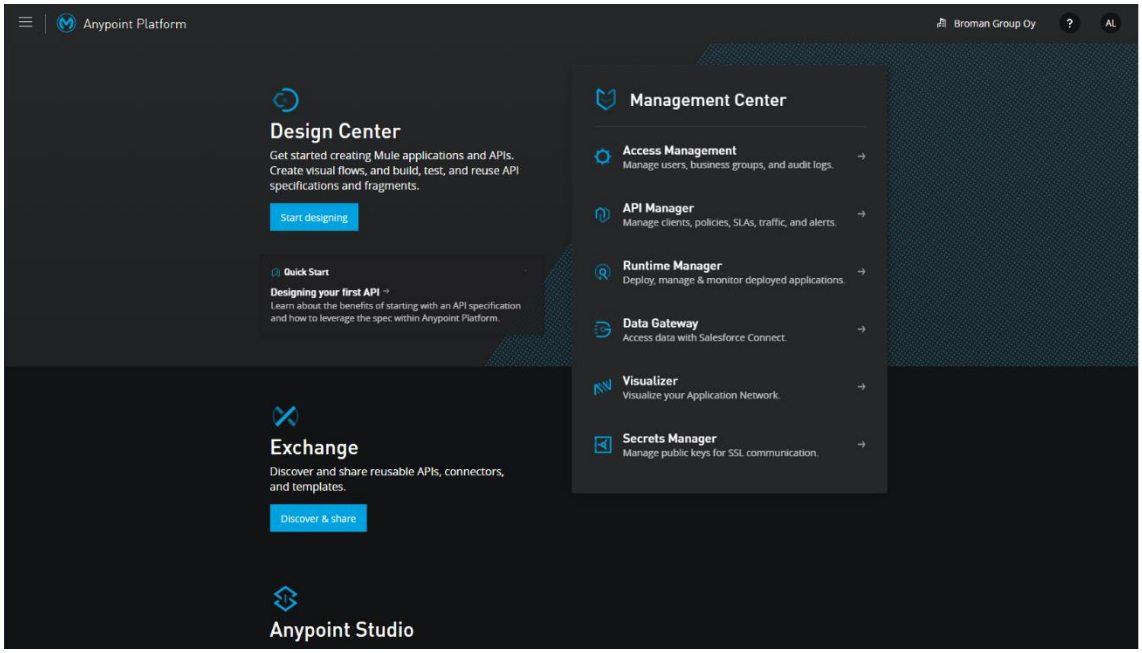
Anypoint Platform tukee valtavaa määrää erilaisia yhteys- ja rajapintateknologioita, myös samoja kuin Talend. Näitä rajapintoja lisätään myös jatkuvasti lisää. (MuleSoft 2018.)

Anypoint Studio muistuttaa ulkoisesti puhdasta Eclipse-kehitysympäristöä todella paljon (Kuvio 7), mutta on toiminnallisuudeltaan melko erilainen. Anypoint Studio ei mahdollista projektin Java-lähdekoodin tarkastelua, vaan tämä on korvattu XML-standardin mukaisella muotoiluskriptillä.

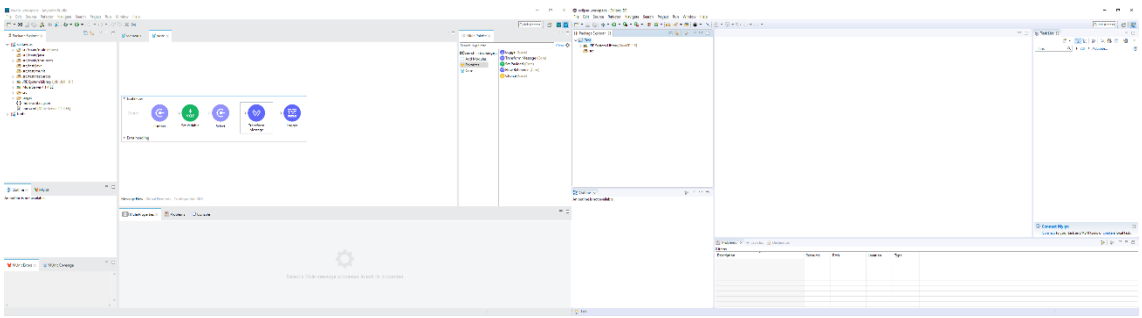
3.4 Projektin luonti Anypointissa

Projektin luonti aloitetaan Anypointissa valitsemalla uusi Mule-projekti (Kuvio 8). Samassa listassa on myös vaihtoehtoja luoda projekti valmiin pohjan päälle. Listasta on myös mahdollista valita Eclipse-kehitysympäristön Java-versiolle tyypillisiä projektityyppejä. Mule-projektin valinnan jälkeen projektille annetaan nimi ja se voidaan liitää joko paikalliseen Mule serveriin, tai Anypoint platformin pilvipalveluihin (Kuvio 9). Tämän jälkeen käyttäjä voi rakentaa canvas-paneeliin haluamansa toteutuksen käyttämällä paletissa saatavilla olevia moduuleja (Kuvio 10). Toteutusta voidaan testata paikallisesti kääntämällä ja ajamalla projekti. Projekti suoritetaan paikallisessa versiossa Mule-runtimesta. Kun projekti on valmis, se

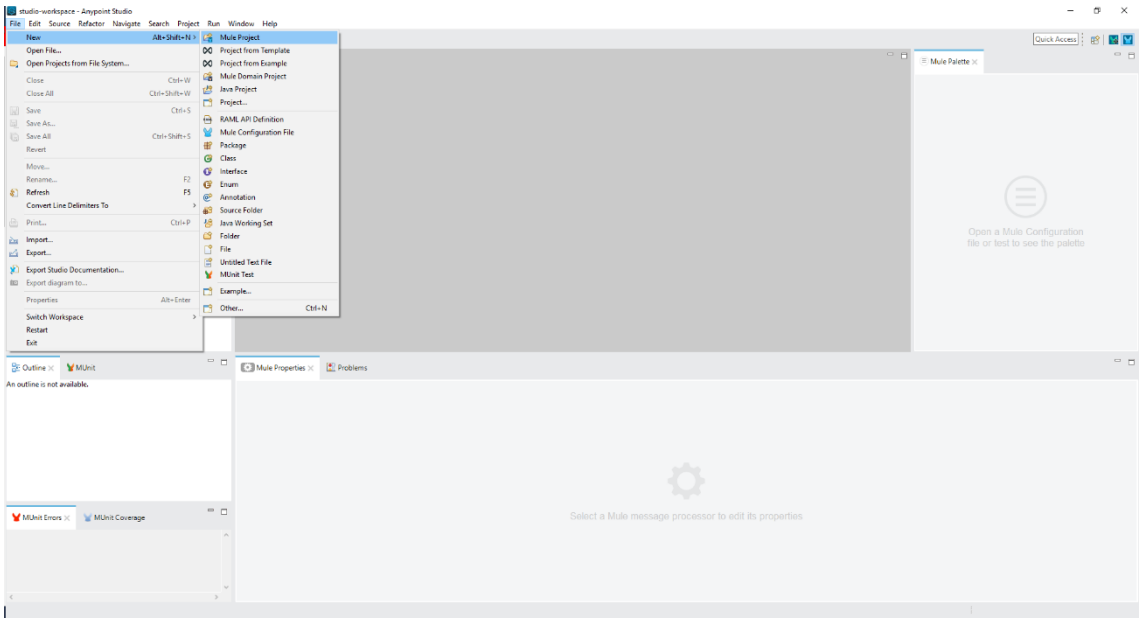
voidaan joko lähettää suoraan Anypoint Platformiin, joka voi olla joko paikallisella palvelimella tai pilvessä. Paikalliselle Mule Runtime palvelimelle asennettava paketti voidaan luoda suorittamalla komento mvn clean package käyttöjärjestelmän komentorivillä.



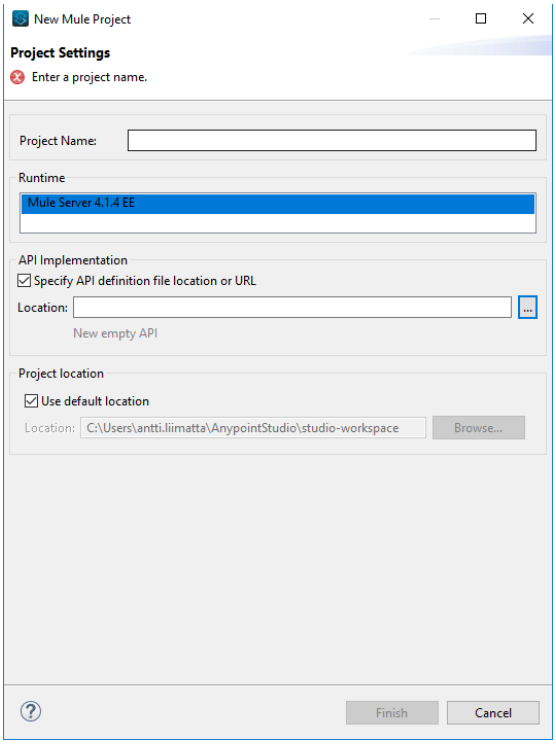
Kuvio 6. Anypoint Platform Management Center.



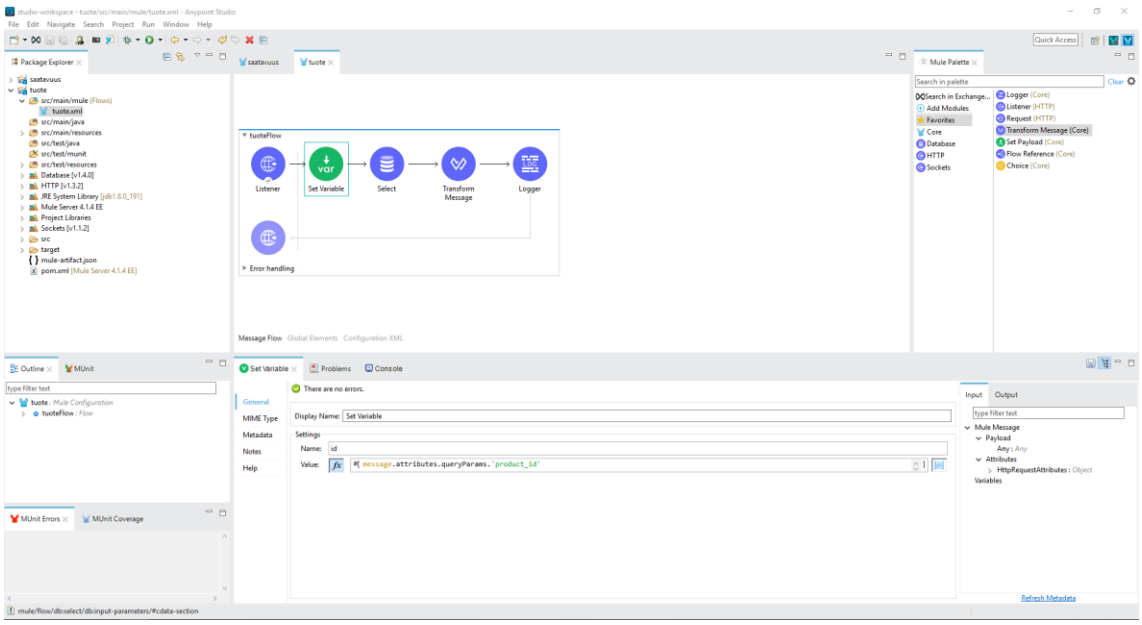
Kuvio 7. Anypoint Studio (vasen) ja Eclipse (oikea).



Kuvio 8. Projektin luonti.



Kuvio 9. Projektin luontiasetukset.



Kuvio 10. Projektikaavio.

3.5 Frends

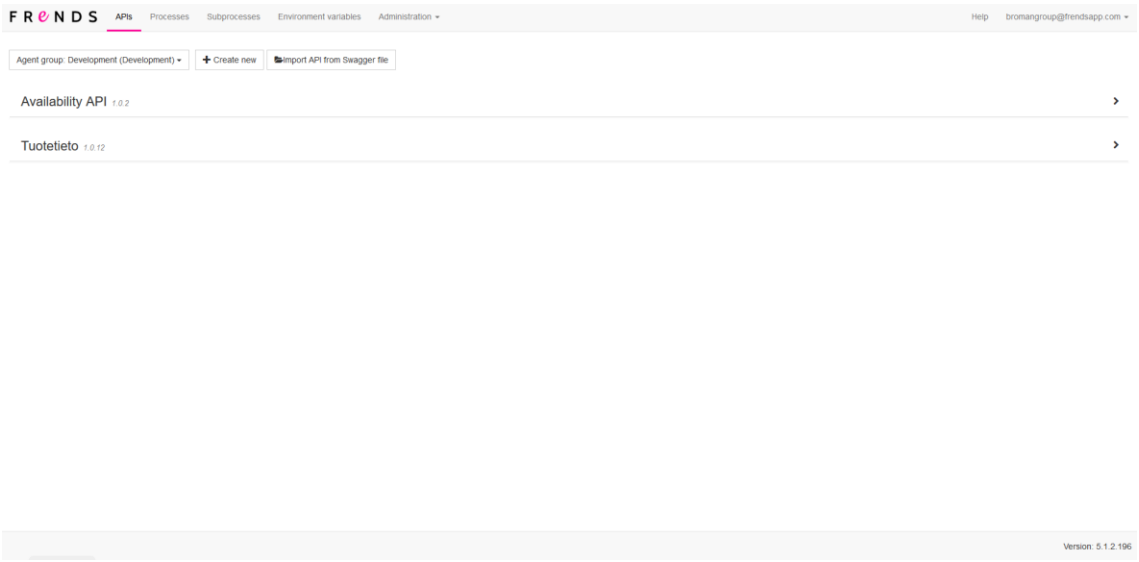
Frends on suomalaisen HIQ Finland Oy:n kehittämä järjestelmäintegraatioalusta. Ohjelma noudattaa ESB-mallia ja pohjautuu C#-ohjelmointikieleen. Frendsiä käyttää pohjoismaissa 250 yritystä, mukaan lukien Aalto-yliopisto (HiQ Finland 2018; Moisio 2018).

Frendsillä integraatiototeutusten kehitys tapahtuu verkkokäyttöliittymän avulla. Ympäristö käyttää tunnistautumiseen Office365-tunnusta. Kyseessä on siis ainoa vertailtavista integraatioalustoista, jonka kehitysympäristö ei pohjaudu Eclipseen tai Java-ohjelmointikieleen. Näin voidaan kiertää kehityksen aloittamista vaivaavat ongelmat, kuten Java-kehityspaketin asentaminen. Kehitys ei vaadi tehokasta tietokonetta, vaan kehittäjä voikin työskennellä Frendsin parissa missä tahansa ja millä tietokoneella hyvänsä. Tämä on huomattava etu kilpailijoihin verrattuna, jotka vaativat vähintään keskitehokkaalla prosessorilla ja tarpeeksi suurella keskusmuistilla varustetun tietokoneen varsinkin Java-virtuaalikoneen vuoksi. Lisäksi käyttäjän ei tarvitse huolehtia palomuurista tai muista verkkoasetuksista.

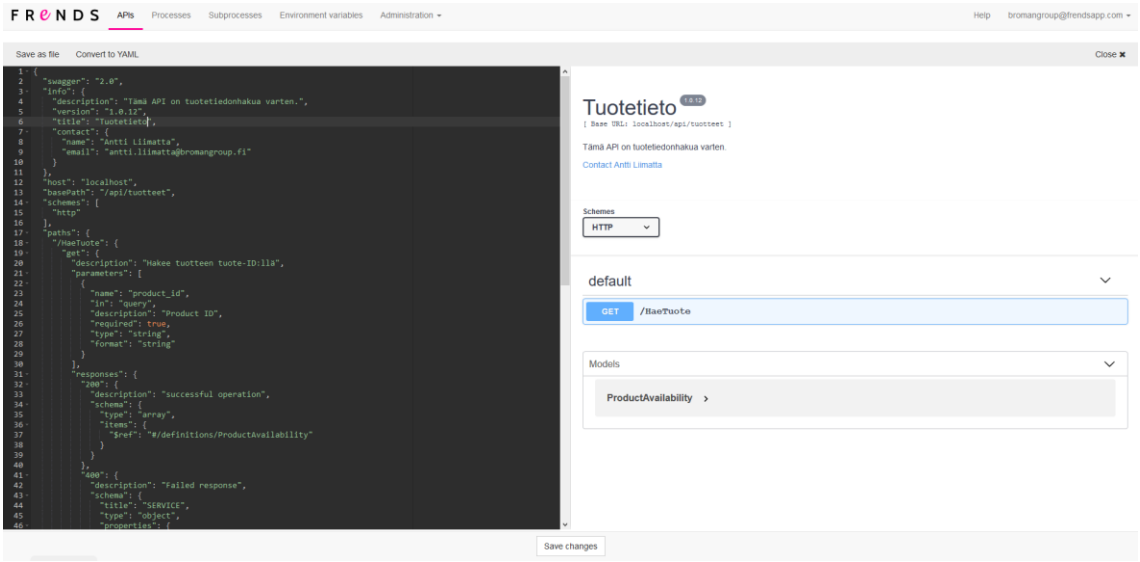
Frendsiä käyttäessä käyttäjän tietokoneen käyttöjärjestelmällä ei ole merkitystä, kunhan se sisältää modernin verkkoselaimen (Galkin 2019).

3.6 Projektin luonti Frendsillä

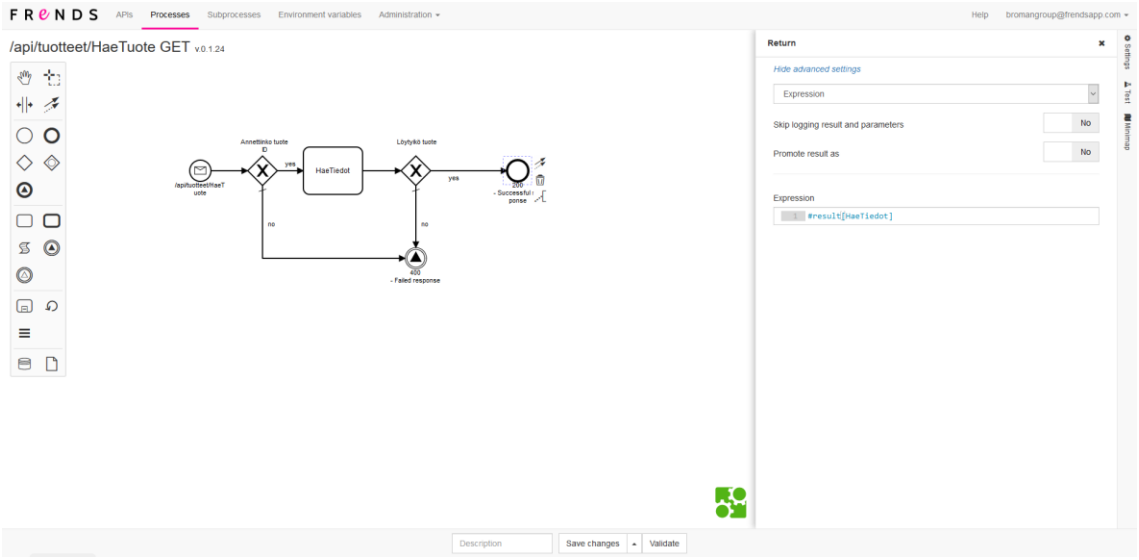
Projektin luonti aloitetaan kirjautumalla sisään Frendsin kehitysympäristöön. Kehitysympäristössä voidaan valita yläpalkista joko API:t, prosessit, aliprosessit, ympäristömuuttujat ja asetukset (Kuvio 11). Tässä esimerkissä käyn läpi API-prosessin luonnin. Aluksi on määriteltävä API, jota prosessi tulee käyttämään. Valitsemme siis kohdan API kehitysympäristön yläpalkista. Tässä kehitysympäristössä voidaan määritellä REST-rajapinta käyttäen Swagger-notaatiota. Kirjoittamiseen voidaan käyttää JSON- tai YAML-tekstiformaattia (Kuvio 12). Kun API on valmis, se tallennetaan ja voidaan siirtyä suunnittelemaan sitä käyttävä prosessi valitsemalla prosessit välilehdessä ”Create new”. Avautuvalla sivulla voidaan rakentaa integraatiototeutus vetämällä sivupalkista moduuleja työskentelyalueelle (Kuvio 13). Toteutus voidaan lopuksi tallentaa joko keskeneräisenä, tai valmiina toteutuksena. Valmiit toteutukset otetaan välittömästi käyttöön. Aliprosessien luonti toimii lähes samalla tavalla, mutta niitä ei oteta tallentamisen jälkeen käyttöön, vaan ne on ajettava muiden prosessien sisältä (Galkin 2019).



Kuvio 11. Käyttöliittymä.



Kuvio 12. API-kuvaus.



Kuvio 13. Projektikaavio.

4 Testitapauksista yleisesti

Testausta suunnitellessani otin huomioon seuraavat teknologiat, joita integraatioalustojen on tuettava:

- REST (rajapintojen kutsuminen ja luominen).
- JSON.
- XML.
- PostgreSQL.
- OAuth.

Arvioin myös kehitysympäristöjen helppokäyttöisyyttä ja käyttökokemusta omasta näkökulmastani, sekä toteutuksien yksinkertaisuutta. Arviointini on käyttökokemuksen ja helppokäyttöisyyden kannalta subjektiivinen, mutta toteutuksien yksinkertaisuutta voin testata numeerisesti, laskien toteutuksiin käytettyjen moduulien määrän. Tämä on mahdollista, sillä jokainen testattava integraatioalusta käyttää integraatiototeutusten rakentamiseen hyvin samantyyppistä käyttöliittymää. Pyrin myös vertaamaan alustojen hinnoittelua.

Muodostan tuloksista taulukon, jossa annan kullekin integraatioalustalle arvosanan kussakin kategoriassa, poislukien numeerisesti mitattavat seikat, kuten hinnoittelu ja moduulien määrä. Hinnoittelua vertaillessani mainitsen myös hintaan

kuuluvan asiakastuen laajuuden. Yhteensopivuus-kategoriassa käytän arvoa ”hyvä”, mikäli testitapauksien aikana ei kohdattu yhteensopivuusongelmia. ”Välttävä”-arvo annetaan, mikäli ilmeni yhteensopivuusongelmia, jotka on mahdollista kiertää. ”Epäsopiva”-arvo merkitsee hyväksymättömiä yhteensopivuusongelmia. Monipuolisuuteen otan kantaa tarkastelemalla integraatioalustojen tukemien teknologioiden kokonaismäärää ja monimuotoisuutta.

5 Ensimmäinen testitapaus

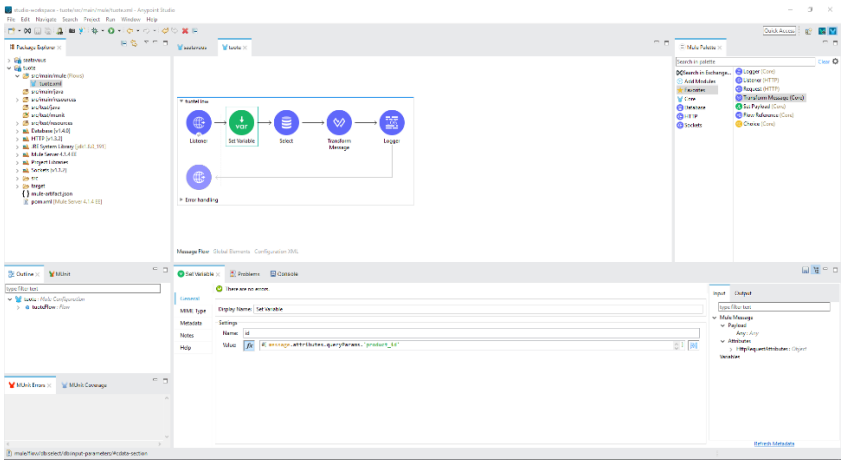
Ensimmäinen testitapaus tähtäsi testaamaan REST-rajapinnan luontia ja tiedon hakua PostgreSQL-tietokannasta. Läpäistäkseen tämän testin, integraatioalustan on kyettävä luomaan REST-rajapinta, josta on mahdollista pyytää tietokannassa olevan myymälätuotteen tiedot syöttämällä GET-kutsun parametrina tuotteen ID. Tiedot on palautettava JSON- tai XML-formaatissa.

5.1 Tulokset Anypoint Studiolla

Anypoint Studiolla toteutuksen luonti oli melko vaivatonta. Määritin Listener-moduulilla REST-rajapinnan, joka ottaa vastaan tuote-ID:n. Tämän jälkeen ID välitetään Select-moduulille, joka hakee tuotteen tiedot PostgreSQL-tietokannasta tämän ID:n avulla. Palautettu data muunnetaan JSON-muotoon ja palautetaan kysyjälle. Suurimmaksi ongelmaksi toteutuksen luonnissa muodostui Anypoint Studiosta puuttuva PostgreSQL-tuki. Tietokantayhteys oli määritettävä geneerisenä SQL-yhteytenä ja JDBC-ajuri oli ladattava erikseen PostgreSQL:n verkkosivuilta. Yhteys toimii tämän jälkeen, mutta natiivi tuki PostgreSQL-yhteyksille olisi ollut toivottua näin menestyksekkäältä integraatioalustalta.

Toteutus vaati yhteensä 5 moduulia, joista aikaa vievin oli Select-moduuli. Testaus oli myös helppoa, sillä Anypoint Studio päivittää taustalla ajettavaa Mule Runtime-palvelinta ilman että projektia tarvitsee kääntää erikseen ennen jokaista

testikertaa. Harmia tosin aiheutti piilotettu Java-koodi. Pääsin ainoastaan tarkastelemaan kaavion taustalla olevaa XML-skriptiä, joka ei ollut kovinkaan paljoa avuksi ongelmanratkaisussa. Kohtasin myös ongelmia SQL-kyselyn kirjoittamisessa, sillä Anypoint Studion sijoittaessa kyselyn taustalla olevaan XML-skriptiin, jotkin kyselyssä olleet merkit rikkoivat XML-syntaksin. Ongelma vaikutti olevan regexp-funktiossa, jota käytin kyselyssä merkkijonojen vertailuun. Tähän ongelmaan en löytänyt ratkaisua, vaan jouduin poistamaan merkkijonojen vertailun kokonaan kyselystä. Tämä ei ole vakava puute tämän toteutuksen sisällä, sillä kyseinen vertailu vain tarkisti haettavan tuotteen olevan myymälän, eikä varaston puolella. Muissa toteutuksissa regexp-funtiot tulevat kuitenkin olemaan tarpeellisia, joten tämä voidaan laskea kriittiseksi puutteeksi.Toteutuksen kaavio on nähtävissä kuviossa 14 ja tulos kuviossa 15.



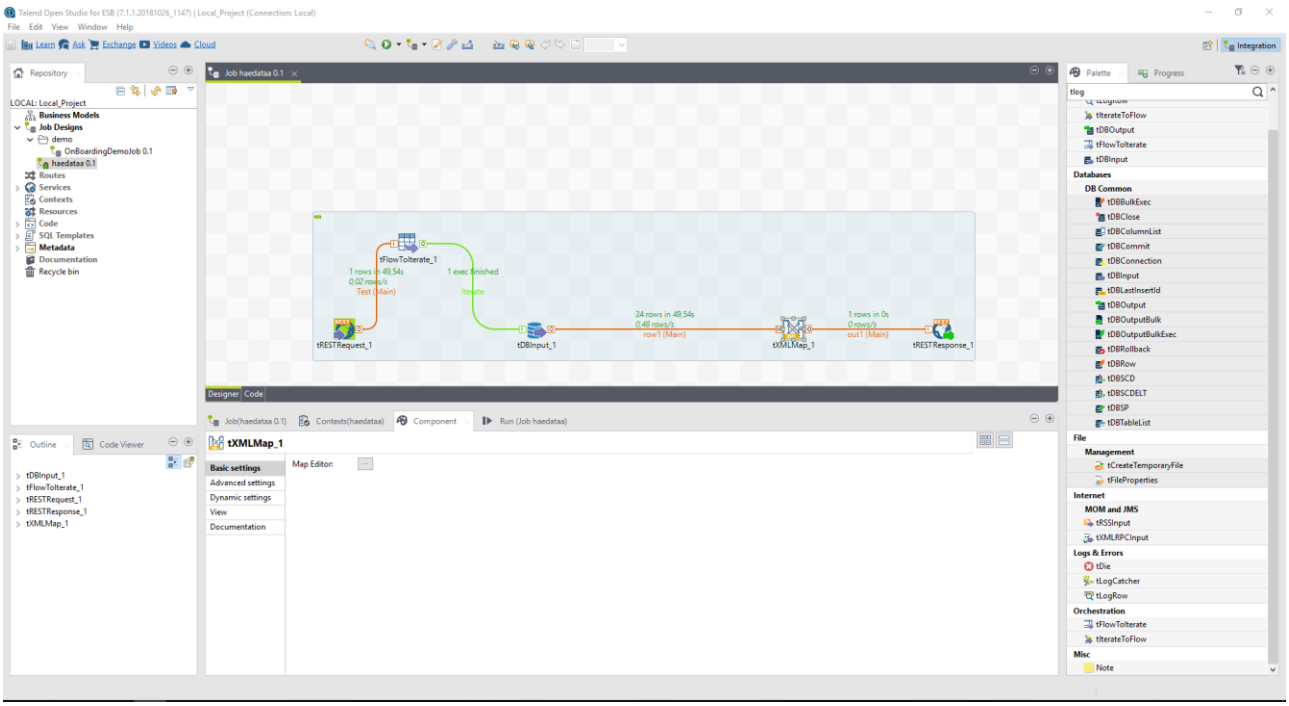
Kuvio 14. Projektikaavio.

Kuvio 15. Tulos.

5.2 Tulokset Talendilla

Talendilla toteutus oli melko samanlainen Anypoint Studion kanssa. Loin REST-rajapinnan tRESTRequest-moduulilla, joka otti parametrina tuotteen ID:n. Tämä ID välitettiin tDBInput-moduulille, joka otti yhteyden PostgreSQL-tietokantaan ja haki tuotteen tiedot ID:n perustella. Tämä data sijoitettiin XML-dokumenttiin XML-Map-moduulissa ja palautettiin kysyjälle RESTResponse-moduulissa. REST-rajapinnan luonti tuntui mielestäni intuitiivisemmalta, kuin Anypoint Studiossa. Moduuli vaikutti monipuolisemmalta ja vaikutti tukevan autentikaatioprotokollia, joita en nähnyt Anypoint-studiossa. Ainoa ongelmaksi laskettava kummallisuus oli Talendin vaikeus käsitellä JSON-formaattista tekstiä. Tämä saattoi tosin johtua omasta osaamattomuudestani. XML-formaattisen datan käsittely tuntui mielestäni Talendissa todella yksinkertaiselle, joten toteutin palautuksen sillä.

Ohjelma tuntui helppokäyttöiseltä ja monipuoliselta, vaikka moduulit ovatkin nimetty melko vaikeaselkoisesti. Onneksi internetissä on saatavilla laaja dokumentaatio ja käyttäjäfoorumit. Myös mahdollisuus lukea taustalla ajettavaa Java-koodia, auttoi ongelmanratkonnassa huomattavasti. Toteutusta testattaessa se oli kuitenkin käännettävä ja Talendin palvelin käynnistettävä. Tämä teki testaamisesta hieman rasittavaa. Kaavio on nähtävissä kuviossa 16 ja tulokset kuviossa 17.



Kuvio 16. Projektikaavio

```
−<Tuote product_id="538595dd-d7b6-473f-b1bf-ef595fb93d70">
    <taxprice>349.0000</taxprice>
  −<name>
      GAMO Replay 10 lippaallinen ilmakivääri 4,5 mm 4X32WR kiikaritähtäimellä
    </name>
    <tuotekoodi>725501449541</tuotekoodi>
    <yhinta>349.0000000000000000</yhinta>
    <unitabbreviation>pcs</unitabbreviation>
    <size>1.000000</size>
  </Tuote>
```

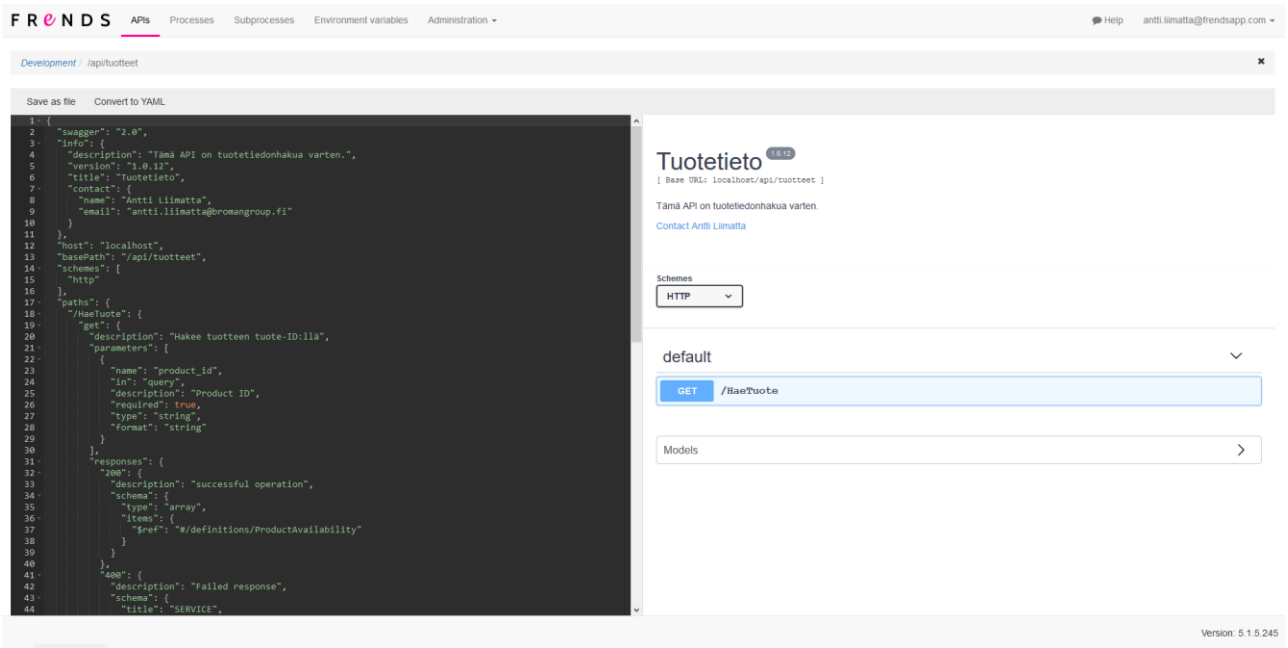
Kuvio 17. Tulos.

5.3 Tulokset Frendsillä

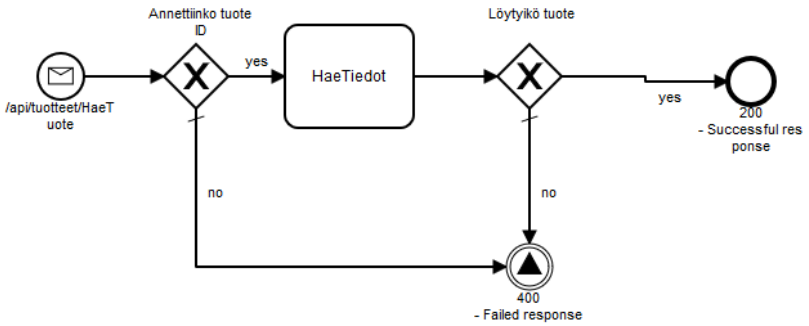
Frends vaati huomattavasti pidemmän opettelujakson verrattuna Anypoint Studioon tai Talendiin. Aluksi toteutukselle oli luotava kutsuttava API, joka onnistui käyttäen Frendsin Swagger editoria (kuvio 18). Pelkästään tämä vaihe vei huomattavan määrän aikaa, mutta myöhemmin tajusin, että saman toiminnallisuuden olisi voinut toteuttaa Frendsin HTTP-moduulilla. Swagger API:t ovat käytännöllisempiä suuremmille integraatiototeutuksille.

Käytin toteutukseen yhteensä 6 moduulia, mutta olisin voinut karsia tämän vain kolmeen moduuliin, jos en olisi lisännyt virheenhallintaa. Virheenhallinnan lisäsin kokeena, sillä se vaikutti melko helppokäyttöiseltä. Swaggerilla luotu REST-rajapinta ottaa vastaan tuotteen ID:n, joka välitetään PostgreSQL-moduulille, joka hakee tuotteen tiedot tietokannasta ja palauttaa ne JSON-muotoisena merkkijonona. Tämä merkkijono palautetaan kysyjälle. Virheenhallinta pitää huolen siitä, että ID on syötetty ja että tuote löytyy tietokannasta.

API-kuvaus löytyy kuviosta 18, projektikaavio kuviosta 19 ja tulos kuviosta 20.



Kuvio 18. API-kuvaus.



Kuvio 19. Projektikaavio.

Kuvio 20. Tulos.

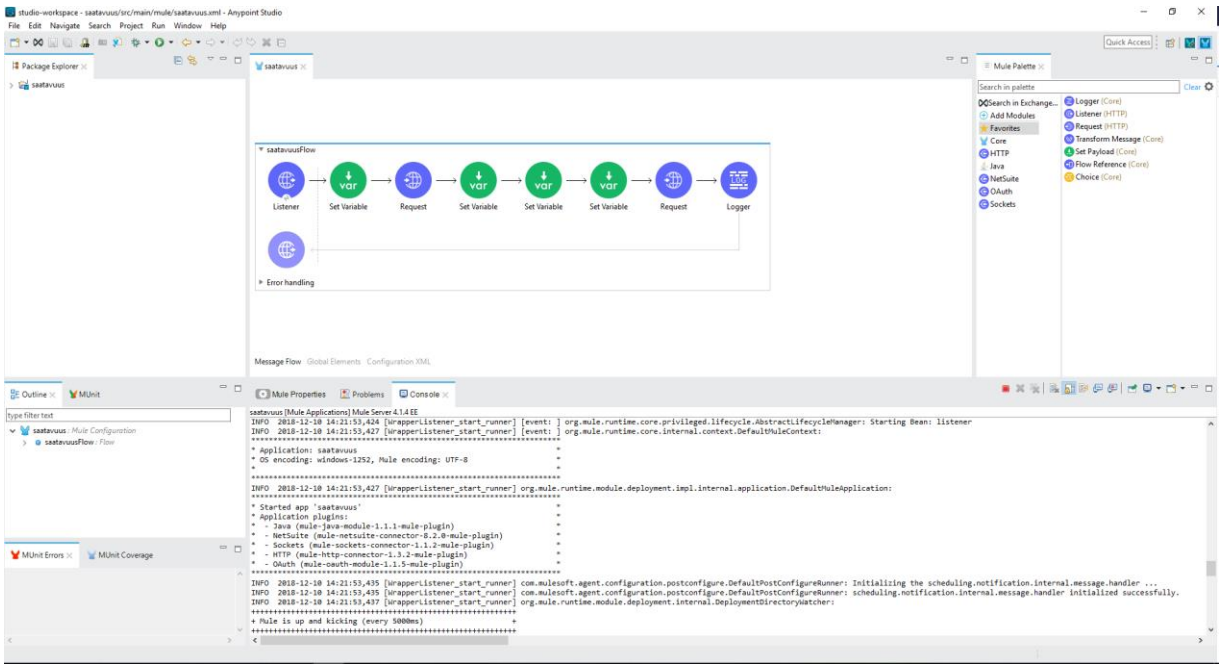
6 Toinen testitapaus

Toisen testitapauksen tavoitteena on testata REST-kutsun lähettämistä ulkopuoliseen REST-rajapintaan. Toteutuksen on välitettävä toteutuksen omaan REST-rajapintaan parametrina annettu tuotekoodi ulkopuoliseen rajapintaan ja palautettava vastauksena saatava JSON kysyjälle. Testattavana rajapintana toimi erään Broman Group Oy:n toimittajan saatavuus-rajapinta, joka vaatii tunnistautumisen OAuth-tokenilla. Rajapinta palauttaa tiedot JSON-muotoisena merkkijonona.

6.1 Tulokset Anypoint Studiolla

Toteutus oli odotettua yksinkertaisempi Anypoint Studiolla. Loin REST-rajapinnan listener-moduulilla, joka otti vastaan haettavan tuotteen tuotenumeron parametrina. Tämä tuotenumero asetettiin muuttujaan Set Variable-moduulilla. Tämän jälkeen haettiin toimittajan autentikointirajapinnasta tokeni ja tokenityyppi, jotka asetettiin muuttujiin ja yhdistettiin yhdeksi merkkijonoksi. Lopuksi lähetettiin GET-kutsu toimittajan saatavuus-rajapintaan ja palautettiin saatu JSON-muotoinen merkkijono kysyjälle.

Toteutus vaati yhteensä 8 moduulia, vaikkakin uskoisin olevan mahdollista karsia ainakin yksi pois. En kohdannut kehitysprosessin aikana mitään suuria vaikeuksia, vaan kaikki sujui hyvin johdonmukaisesti, eikä ohjelmistossa ilmennyt puutteita, jotka olisivat haitanneet kehitysprosessia. Projektikaavio on nähtävillä kuviossa 21 ja tulos kuviossa 22.



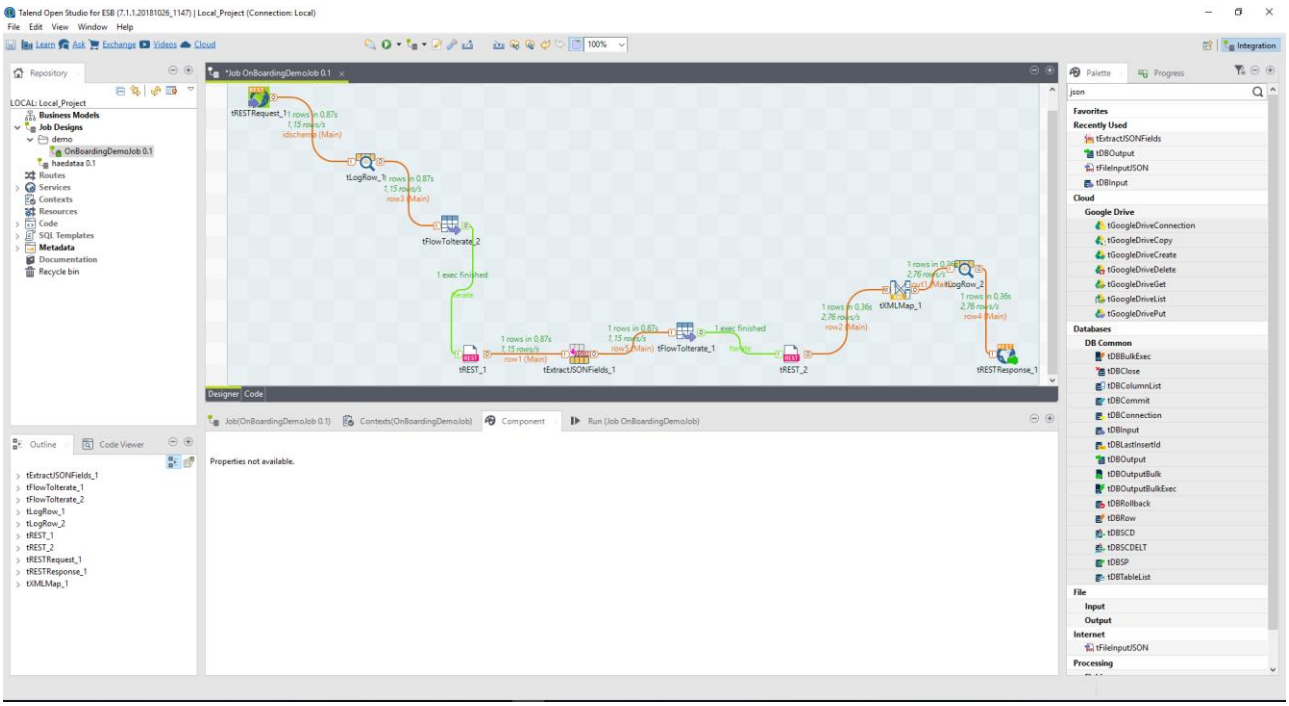
Kuvio 21. Projektikaavio.

Kuvio 22. Tulos.

6.2 Tulokset Talendilla

Talendilla toteutus oli yllättävän haasteellinen. Loin kutsuttavan REST-rajapinnan tRESTRequest-moduulilla, joka otti parametrina vastaan tuotenumeron. Tämän jälkeen siirryttiin noutamaan tokeni autentikointi-rajapinnasta tREST-moduulilla. Itse tokeni poimittiin vastauksesta tExtractJSONFields-moduulilla. Lopuksi suoritettiin saatavuuskysely kirjautumalla tokenin avulla toimittajan rajapintaan ja lisäämällä vastaus XML-dokumenttiin. Tämä XML-dokumentti palautettiin kysyjälle tRESTResponse-moduulilla. Tämä XML-dokumentti tosin sisälsi vain yhden kentän, joka oli toimittajan rajapinnan palauttama JSON-merkkijono. En saanut palautettua Talendista pelkkää JSON-merkkijonoa.

REST-kutsujen lähettäminen Talendilla oli mielestäni turhan monimutkaista. Talend on melko vaikeaselkoinen sen suhteen, mitkä moduulit voivat asettua peräkkäin samassa sekvenssissä, sekä millaista dataa ne voivat ottaa sisään. Talendin dokumentaatio ei myöskään selittänyt kovin hyvin, mitä moduulit, kuten tFlowIterate tekevät. Ne vain olivat ohjeistuksen mukaan tarpeellisia tämän kaltaisten toteutusten luomisessa. Lopulta toteutukseni toimii karkeasti, enkä ole täysin perillä sen kaikkien osien tarkoituksesta. Toteutus vaati yhteensä 10 moduulia. Kaavio on nähtävissä kuviossa 23 ja tulos kuviossa 24.



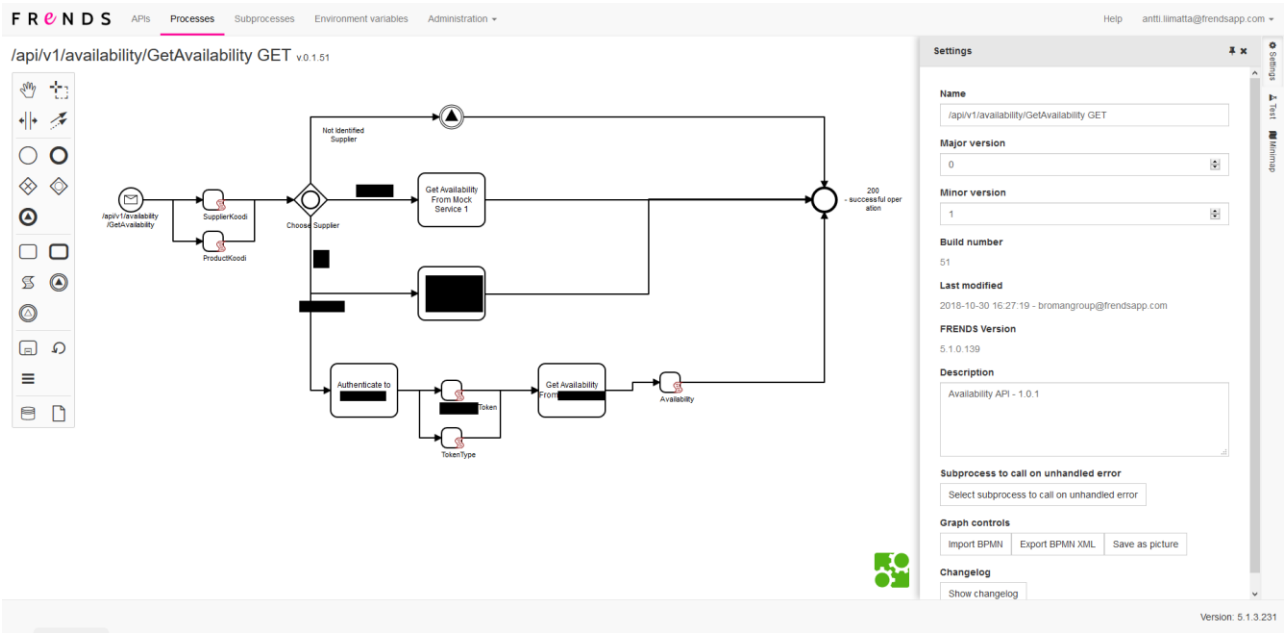
Kuvio 23. Projektikaavio.


```
−<root>
  −<Body>
    {"ArticleId":18370,"EAN":null,"ArticleText":"175/65R14 82T Antares Toimitetaan Maxtrek Nasta","QuantityAvailable":10,"QuantityExternal":0,"ExternalDeliveryTime":null,"NetPrice":40.33,"DotYear":2016}
  </Body>
</root>
```


Kuvio 24. Tulos.

6.3 Tulokset Frendsillä

Frendsillä en luonut koko toteutusta täysin alusta loppuun, sillä tehdessäni testitapauksia muilla alustoilla, Frendsin kehittäjät olivat luoneet pohjan saatavuuskyselyn API-rajapinnalle ja esimieheni oli rakentanut itse toteutuksen. Ohjelma otti vastaan tuotekoodin REST-rajapinnan parametrina, josta tämä tallennettiin muuttujaan. Lisäksi rajapinta otti vastaan toimittajakoodin, jolla määriteltiin, minkä yrityksen rajapintaan haluttiin olla yhteydessä. Tämä lisättiin tulevaisuutta varten, jos halutaan lisätä testitoimittajan lisäksi muiden yritysten rajapintoja samaan toteutukseen. Tämäkin koodi tallennettiin muuttujaan, jonka avulla valittiin oikea polku inclusive choise-moduulissa. Siirryttyään oikaan haaraan, ohjelma haki autentikointi-rajapinnasta tokenin ja tokenityypin, jotka tallennettiin muuttujiin. Näiden muuttujien avulla haetaan haluttu data välittämällä tuotekoodi toimittajan saatavuus-rajapintaan. Tästä tuloksesta poimittiin saatavuutta merkitsevä luku ja palautettiin se kysyjälle. Itse toteutus vaati 10 moduulia, vaikka kaaviossa onkin niitä enemmän. Nämä ylimääräiset moduulit ovat jatkokehitystä varten. Kaavio on nähtävissä kuviossa 25.



Kuvio 25. Projektikaavio (liikesalaisuudet sensuroitu).

7 Yhteenveto

Kaikki integraatioalustat ovat hyviä omilla tavoillaan. Anypoint Platform on lähes minkä tahansa ohjelmiston kanssa, mutta tämä yhteensopivuus ei välttämättä ole tuettu kovin hyvin, kuten PostgreSQL:n kanssa kävi ilmi. Itse kehitysympäristön asentaminen oli myös melko tuskainen prosessi, sillä jouduin poistamaan Java Runtime Environmentin kokonaan työkoneeltani, jotta Anypoint Studio havaitsi Java Development Kitin. Tämä ei ole ollut aikaisemmin ongelma käyttämieni Eclipse-pohjaisten kehitysympäristöjen kanssa, eikä se myöskään ollut ongelma Talendin asennuksessa. Anypoint Studio oli myös melko raskas ajaa, mikä johti kehityksen hitauteen. Lisäksi taustalla ajettavan koodin piilotus tuntui mielestäni todella kummalliselta. Hyvänä puolena tosin mainittakoon toteutusten kompaktius.

Talend tuntui käyttöliittymänsä ja nimeämiskäytäntöjensä puolesta vanhanaikaiselta ja kankealta. Toteutuksen rakentaminen tuntui kuitenkin Anypoint Studioon verrattuna monipuolisemmalta, vaikkakin vaikeammin ymmärrettävältä. Tämä tosin ei olisi ongelma alun opettelujakson jälkeen. Talend vaikuttaisi tukevan suurempia, yksittäisiä toteutuksia, kun taas Anypoint Studio vaikuttaisi olevan suunnattu pienemmille ratkaisuille, joita voidaan yhdistellä toisiinsa. En ehtinyt testaamaan tätä toiminnallisuutta, mutta uskoisin saman olevan mahdollista myös Talendissa. Mainittakoon, että suurten toteutusten kääntäminen vie molemmilla alustoilla vaihtelevan määrän aikaa, riippuen käyttäjän tietokoneen tehokkuudesta.

Frends tuntui joukon moderneimmalta integraatioalustalta. Frendsin kehitysympäristö on käytettävissä täysin verkkoympäristössä, poistaen vaatimuksen tehokkaalle tietokoneelle. Lisäksi työympäristö on hyvin selkeä ja helppokäyttöinen. Saatavilla olevien moduulien määrä on kilpailijoitaan suppeampi, mutta uusia moduuleja julkaistaan jatkuvasti lisää HiQ:n ja vapaaehtoisten kehittäjien toimesta. Moduuleja on myös mahdollista luoda itse C#-ohjelmointikielellä. Frendsin API-

kehitys on kilpailijoihinsa verrattuna modernimpaa ja rajapintojen käyttöönotto on vaivatonta. Tosin Anypoint Platformilla on tarjolla verkkoympäristössä saatavilla oleva erillinen API-kehitysympäristö, mutta koska keskityin integraatioalustojen kehitysympäristöihin, jättäen pilvipalvelut vertailun ulkopuolelle, en tätä API-kehitysympäristöä tarkastellut. Frendsillä kaikki kehitystyökalut ovat saatavilla samassa ympäristössä. Frends on myös joukon ainoa suomalainen tuote, mikä tekee kehittäjien kanssa kommunikoimisen huomattavasti helpommaksi verrattuna kilpailijoihin.

Kaikki alustat läpäisivät testitapaukset ilman suuria vaikeuksia. Teknologinen yhteensopivuus oli hyväksyttävää kaikissa tapauksissa ja kohtaamani ongelmat eivät olleet kriittisiä ja johtuivat mahdollisesti omasta kokemattomuudestani. Keräämieni havaintojen perusteella suosittelen kuitenkin Broman Group Oy:lle HiQ Frendsiä sen modernin kehitysympäristön ja kehittyvän koodipohjan perusteella. Vaikka Frends onkin kilpailijoitaan suppeampi toiminnallisuudeltaan, tämän ei pitäisi olla ongelma tulevaisuudessa. Suurin osa puuttuvista ominaisuuksista liittyy kaupallisiin teknologioihin, kuten Azureen ja Salesforceen, joita Broman Group Oy ei käytä. Lisäksi virheenhallinta oli kilpailijoihin verrattuna helppo toteuttaa, mikä helpottaa kehitystyötä.

7.1 Vertailutaulukot

Vaadittujen moduulien määrä	Testitapaus 1	Testitapaus 2	Yhteensä
Talend	5	10	15
Anypoint Platform	5	8	13
Frends	3	10	13

Listasin tähän taulukkoon testitapauksiin vaadittujen moduulien määrän. Talend vaati eniten moduuleja, kun taas Anypoint ja Frends olivat tasoissa. Työskentely Talendin kanssa oli tämän takia hitaampaa, kuin muilla integraatioalustoilla. Tämä vertailu ei ota kantaa moduulien monimutkaisuuteen, mutta monelle ohjelmistokehittäjälle graafisen käyttöliittymän käyttö ohjelmoinnissa hidastaa työskentelyä, joten useampien moduulien sijoittelu kasvattaa työhön käytettävää aikaa ja opettelua.

Pisteytys	Talend	Anypoint Platform	Frends
Helppokäyttöisyys	2	3	5
Opittavuus	2	3	4
Tuetut teknologiat	4	5	3
Hinnoittelu	0€/kk, Ei tukea	?€/kk, Täysi tuki	3000€/kk, Täysi tuki
Yhteensopivuus	Välttävä	Hyvä	Erinomainen

Pisteytysmetodini on tässä seuraava:

1. Huono.
2. Välttävä.
3. Neutraali.
4. Hyvä.
5. Erinomainen.

Helppokäyttöisyyden puolesta Frends oli ehdoton voittaja. Käyttöliittymä oli erittäin luonnollinen käyttää ja moduulit oli nimetty järkevästi. Anypoint Platform oli myös nimennyt moduulinsa hyvin, mutta Eclipse-pohjainen kehitysympäristö voi olla varsinkin ensikertalaiselle vaikea ymmärtää. Talend kärsi myös Eclipse-pohjaisen kehitysympäristön monimutkaisuudesta, sekä sekavasta moduulien nimeämiskäytännöstä. Suurin ongelma Talendin ja Anypoint Platformin kanssa oli kuitenkin asennusprosessin hankaluus, josta Frends ei kärsi, sillä se toimii täysin Internet-ympäristössä.

Helpoiten opittava ympäristö oli ehdottomasti Frends, sillä se ei kärsinyt Eclipse-pohjaisen kehitysympäristön kömpelöstä rakenteesta tai hämmentävistä nimeämiskäytänteistä.

Anypoint Platform tuki ehdottomasti suurinta määrä teknologioita. Talend tuki myös melko laajasti eri teknologioita, mutta joissakin tapauksissa tuen laatu oli kyseenalainen. Esimerkiksi REST-moduuleja oli useita ja niiden toiminnoilla oli hämmentäviä päällekkäisyyksiä.

Hinnoittelun puolesta Talend oli ehdottomasti paras vaihtoehto, sillä sen vapaan lähdekoodin versio on vapaasti käytettävissä. Tälle versiolle ei kuitenkaan ollut saatavilla tukea Talendin puolelta ongelmatilanteissa. Anypoint Platform ja Frends tarjosivat kaupallisen lisenssin mukana täyden tuen, mutta Anypoint Platformin hinnoittelu olisi tullut neuvotella markkinoijan kanssa. Frendsin alimman tason lisenssi maksoi 3000€/kk. Tämä sisältää myös täyden tuen ongelmatilanteissa.

Otettuani kaikki nämä seikat huomioon, totesin Frendsin olevan yhteensopivin integraatioalusta Broman Group Oy:n olemassa olevaan IT-infrastruktuuriin.

Kaikki integraatioalustat olisivat tukemiensa teknologioiden puolesta yhteensopivia, mutta varsinkin helppokäyttöisyys ja opittavuus nostavat työskentelyn mielekkyyttä ja työn laatua pitkällä aikavälillä. Yrityksen on myös luonnollisesti järkevää valita vaihtoehto, jolla on selkein ja järkevin hinnoittelu.

8 Pohdinta

Valitsin mielestäni oleellisen aiheen nykyisen työtilanteeni huomioon ottaen. Aihe oli myös mielenkiintoinen, sillä en ollut aikaisemmin perehtynyt data- tai järjestelmäintegraatioihin. En ollut myöskään saanut koulutusta aiheeseen liittyen, mikä oli mielestäni kummallista, sillä integraatiototeutukset ovat suuri osa yrityksissä tapahtuvasta ohjelmistokehityksestä, ellei tätä ulkoisteta. Testitapausten suunnittelu tapahtui yhdessä esimieheni kanssa ja mielestäni ne sopivat opinnäytetyön tavoitteisiin hyvin. Jos aikaa olisi ollut enemmän käytettävissä, olisin lisännyt vielä yhden testitapauksen tiedostojen käsittelylle, mutta työmäärä olisi silloin noussut kohtuuttomaksi ottaen huomioon, että minun täytyi myös keskittyä muihin työtehtäviin tämän opinnäytetyön toteutuksen aikana.

Jos aloittaisin opinnäytetyön alusta, olisin enemmän yhteydessä integraatioalustojen kehittäjiin ja niitä käyttäviin yrityksiin. Yhteydenpito tämän opinnäytetyön aikana oli melko rajallista, mikä oli yksi syy opetteluvaiheen hitauteen. Halusin perehtyä asioihin omilla ehdoillani ilman kädestä pitoa, mikä oli jälkikäteen ajateltuna virhe. Olen kuitenkin tyytyväinen lopulliseen laatuun ja nautin opinnäytetyön tekemisestä odotettua enemmän. Pelkäsin aiheen olevan puuduttavan tylsä, mutta opittuani dataintegraatioiden oleellisuuden yritysympäristöissä, mielenkiintoni kasvoi huomattavasti. Data- ja järjestelmäintegraatiota pitäisi mielestäni opettaa varsinkin ammattikorkeakouluissa aiheen yrityskeskeisyyden vuoksi.

Tulen mahdollisesti itsekin työskentelemään integraatiototeutusten parissa tulevaisuudessa ja opintojen aikana saama kokemus olisi varmasti hyödyksi. Toivon tämän opinnäytetyön olevan hyödyksi kaikille dataintegraatiosta kiinnostuneille ja

aloitteleville integraatiokehittäjille. En väitä tämän opinnäytetyön olevan paras tiedon lähde aiheeseen liittyen, sillä se on tehty yhden yrityksen tarpeisiin, mutta yritin esitellä dataintegraation mahdollisimman selkeästi aiheesta tietämättömällekin lukijalle.

Lähteet

1. Wikimedia Foundation. Dataintegraatio. Päivitetty 05.05.2013. [Viitattu 18.04.2019.] Saatavissa: https://fi.wikipedia.org/wiki/Dataintegraatio.
2. Flashnode Oy. Integraatiot. [Viitattu 18.04.2019.] Saatavissa: https://www.itewiki.fi/opas/integraatiot/
3. Zhao, Shirley. What is ETL? (Extract, Transform, Load). Päivitetty: 20.10.2017. [Viitattu 18.04.2019.] Saatavissa: https://www.edq.com/blog/what-is-etl-extract-transform-load/
4. Harvey, Robert. Difference between ESB and ETL. Päivitetty 13.11.2015. [Viitattu 18.04.2019.] Saatavissa: https://softwareengineering.stackexchange.com/a/302470
5. Conlay, John III. Basic Concepts: Hub/Spoke vs ESB. 27.09.2013. Saatavissa: http://soalink.blogspot.com/2009/08/difference-between-esb-and-hub-and.html
6. Talend. Talend ESB Container Administration Guide. Päivitetty 2019. [Viitattu 18.04.2019.] Saatavissa: https://help.talend.com/reader/yovCMqvJzyaSSSI-drlB4FQ/wC1xyZN9kSCRCfruACWGCw
7. Ciurana, Eugene. Mule: A Case Study. 01.01.2007. Saatavissa: https://www.theserverside.com/news/1365047/Mule-A-Case-Study
8. Fredriksson, Peter. Talend [yksityinen sähköpostiviesti]. Vastaanottaja: Antti Liimatta. Lähetetty: 5.11.2018.
9. Mulesoft. Anypoint Exhange. Päivitetty 2019. [Viitattu 18.04.2019.] Saatavissa: https://www.mulesoft.com/exchange/?type=connector
10. HiQ Finland. Aalto University Chooses HiQ as Integration Partner. 30.10.2018. Saatavissa: https://www.youtube.com/watch?v=yuSla8Zoi8U
11. Moisio, Juha. Käyttäjämäärät pohjoismaissa. [yksityinen sähköpostiviesti]. Vastaanottaja: Antti Liimatta. Lähetetty: 21.12.2018.
12. Galkin, Ossi. Quick start for those who don't read documentations. Päivitetty 2019. [Viitattu 18.04.2019.] Saatavissa: https://docs.frends.com/getting-started/quick-start/quick-start-for-those-who-dont-read-documentations
13. Galkin, Ossi. How to use FRENDS UI. Päivitetty 2019. [Viitattu 18.04.2019.] Saatavissa: https://docs.frends.com/getting-started/introduction/how-to-use-frends-ui