Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Tuomas Mikkola

Mukautetun tuotekonfiguraattorin toteutus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

31.10.2019

Tekijä Otsikko Sivumäärä Aika	Tuomas Mikkola Mukautetun tuotekonfiguraattorin toteutus 39 sivua 31.10.2019
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	Tietotekniikan koulutusohjelma
Ammatillinen pääaine	Ohjelmistotekniikka
Ohjaajat	Lehtori Jorma Räty

Insinöörityössä tutkittiin Salesforce CPQ -myyntikonfiguraattorin toimintalogiikkaa sekä toteutettiin mukautettu ohjelmisto, joka pystyy suorittamaan tuotekonfiguraatiosääntöjä Salesforce CPQ:n tietueiden perusteella.

Työssä havaittiin, että tuotekonfiguraattoreiden toimintalogiikka pohjautuu yksinkertaisiin, teoreettisiin ajatuksiin ja matemaattisiin sääntöihin, joita noudattamalla voidaan kehittää hyvin monimutkaisia tuotekokonaisuuksia.

Julkisesti tarjolla oleva dokumentaatio oli hyvin rajoittunutta, jolloin Salesforce CPQ:n toimintalogiikan selvittäminen oli ajoittain työlästä.

Lopputuloksen perusteella todettiin, että tuotekonfiguraattorin toteuttaminen on itsessään jo hyvin haastavaa. Työssä onnistuttiin toteuttamaan ohjelma, joka pystyy suorittamaan Salesforce CPQ:n konfiguraatiosääntöjä. Lopputulos luo myös hyvät lähtökohdat mukautettujen sovellusten jatkokehittämiselle.

Olisi kuitenkin toivottavaa, että Salesforce tarjoaisi tulevaisuudessa CPQ-ohjelmointirajapinnan kautta monipuolisemman pääsyn konfiguraattorin toiminnallisuuksiin, jotta toteutuksissa voitaisiin hyödyntää jo olemassa olevaa ohjelmakoodia.

Avainsanat	CPQ, myyntikonfiguraattori, tuotekonfiguraattori

Author Title Number of Pages Date	Tuomas Mikkola Implementation of a Custom Product Configurator 39 pages 31 October 2019
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	Software Engineering
Instructors	Jorma Räty, Senior Lecturer

This thesis investigates the internal logic of Salesforce CPQ (configure, price and quote) product configuration system as well as introduces a custom software solution to execute product- and configuration rules, based on Salesforce CPQ data.

During the course of this thesis it was discovered, that product configuration is based on simple theoretical ideas and mathematical models, which enable users and developers to create rules for product combinations.

The available technical documentation for Salesforce CPQ was extremely limited from a developer perspective, which resulted in the research and development being more cumbersome than was initially anticipated.

As a conclusion it was discovered, that the development of any product configurator system requires a lot of work. Yet the development of a custom solution, one that can execute Salesforce CPQ product configuration rules, was successful and the lessons learned can be utilized in the future for possible custom solutions.

However, as this is a custom solution that more or less simulates existing functionality, it would be highly desirable should Salesforce extend the functionality and documentation of its CPQ API:s to allow a broader use of its functionalities.

Keywords	CPQ, Configure price quote, product configuration

Sisällys

Lyhenteet

API	*Application Programming Interface*. Ohjelmointirajapinta. Määritelmä, jonka avulla ohjelmat voivat kommunikoida keskenään.
CPQ	*Configure, Price and Quote.* Myyntikonfiguraattori. Järjestelmäkokonaisuus myyntityön helpottamiseksi.
CRM	*Customer Relationship Management.* Asiakkuudenhallinta. Kokonaisuus järjestelmiä ja toimintatapoja asiakassuhteiden kehittämiseksi ja ylläpitämiseksi.
DML	*Data Manipulation Language.* Kieli tietueiden tallentamiseksi Salesforceen.
ERP	*Enterprise Resource Planning.* Toiminnanohjausjärjestelmä, joka yhdistää esimerkiksi tuotannon, varastonhallinnan ja kirjanpidon toimintoja.
JSON	*Javascript Object Notation.* Tiedonvälityksessä käytetty tiedostomuoto.
PaaS	*Platform as a Service.* Kolmannen osapuolen tarjoama, ulkoistettu ohjelmistokehitysalusta.
SOQL	*Salesforce Object Query Language.* Kyselykieli tietueiden hakemiseksi Salesforcesta.

1 Johdanto

Insinöörityössä esitellään toteutusratkaisu tuotekonfiguraattorista, joka hyödyntää Salesforce CPQ -myyntikonfiguraattorin datamallia. Tavoitteena oli tutkia mukautetun tuotekonfiguraattorin vaatimuksia, toteutuksen vaativuutta sekä toteuttaa tuotekonfiguraattori, joka selviytyy yksinkertaisimmista tuotekonfiguraattorin käyttötapauksista. Tärkeimpänä tavoitteena oli toteuttaa tuotekonfiguraattorin laskennasta vastaava käyttöliittymäriippumaton ohjelma, jonka yhteyteen Salesforcen päälle voidaan tulevaisuudessa rakentaa erilaisia käyttöliittymäratkaisuja.

Salesforce CPQ tarjoaa myyntikonfiguraattorin toimintojen hyödyntämiseksi hyvin rajalliset puitteet, mutta mahdollistaa konfiguroinnin sääntöjen lukemisen ja oman logiikan kehittämisen niiden ympärille. Työssä esitellään Salesforce CPQ:n tarjoamat rajapinnat sekä datamalli, jonka perusteella Salesforce CPQ -myyntikonfiguraattori ja tuotekonfiguraattori toimivat. Datamallin kohdalla syvennytään tuotekonfiguraattorin datamallin muodostamaan konfiguroitavaan logiikkaan, jonka perusteella lasketaan tuotekokonaisuuksien sisäisiä ja tuotteiden välisiä dynaamisia ja passiivisia sääntösuhteita.

2 Myyntikonfiguraattoriohjelmistot

2.1 Myyntikonfiguraattori

Myyntikonfiguraattorilla (CPQ, Configure, Price and Quote) tarkoitetaan järjestelmää, jonka avulla yrityksen on helppo hallita monimutkaisia tuotekokonaisuuksia sekä muodostaa tarkkoja laskuja ja tarjouksia määriteltyjen sääntöjen perusteella (Anderson, 2014).

Myyntikonfiguraattorista on apua etenkin sellaisen tarjouksen muodostamisessa, jossa tarjouksen sisältämien tuotteiden lopullinen hinta riippuu useasta tuotteen ominaisuudesta ja / tai muista ennalta määritellyistä säännöistä. Tällaiset säännöt voivat olla hyvin monimutkaisia ihmisen muistettavaksi tai hahmotettavaksi, jolloin näiden käsitteleminen

myyntikonfiguraattorilla pienentää riskiä virheellisten tarjousten tai tuotteiden muodostamiseksi. (McCormick, 2016.)

Gartnerin teettämässä tutkimuksessa (Hilbert ym. 2018) esitellään seuraavat käyttötapaukset, joissa myyntikonfiguraatoria voidaan hyödyntää:

- Suoramyynnissä (Direct Sales) myyjä itse muodostaa ja hinnoittelee tarjouksen.
- Kanavoidussa myynnissä (Channel Sales) myyjä vastaanottaa valmiin tarjouksen tai tilauksen (esim. jakelijalta) ja prosessoi sen eteenpäin.
- Itsepalvelumyynnissä (Self-Service Sales) asiakas itse valitsee ja konfiguroi haluamansa tuotteet, ja tekee tilauksen tai tarjouspyynnön.
- Jatkuvien tilausten hallinnassa (Subscription Management) jatkuvia tilauksia (subscription) muokataan, uusitaan tai lakkautetaan.
- Monimutkaisten teollisuustuotteiden konfiguroinnissa (Complex Manufacturing käyttäjä konfiguroi monimutkaisia tuotepaketteja tai -kokonaisuuksia.
- Ratkaisumyynnissä (Solution Selling) yritys tarjoaa asiakkaan tarpeiden mukaan räätälöityjä tuotteita ja palveluja.

Kuten yllämainituista käyttötapauksista ilmenee, myyntikonfiguraattoria on mahdollista hyödyntää sekä yksinkertaisissa että hyvin monimutkaisissa käyttötapauksissa.

2.2 Tuotekonfiguraattori

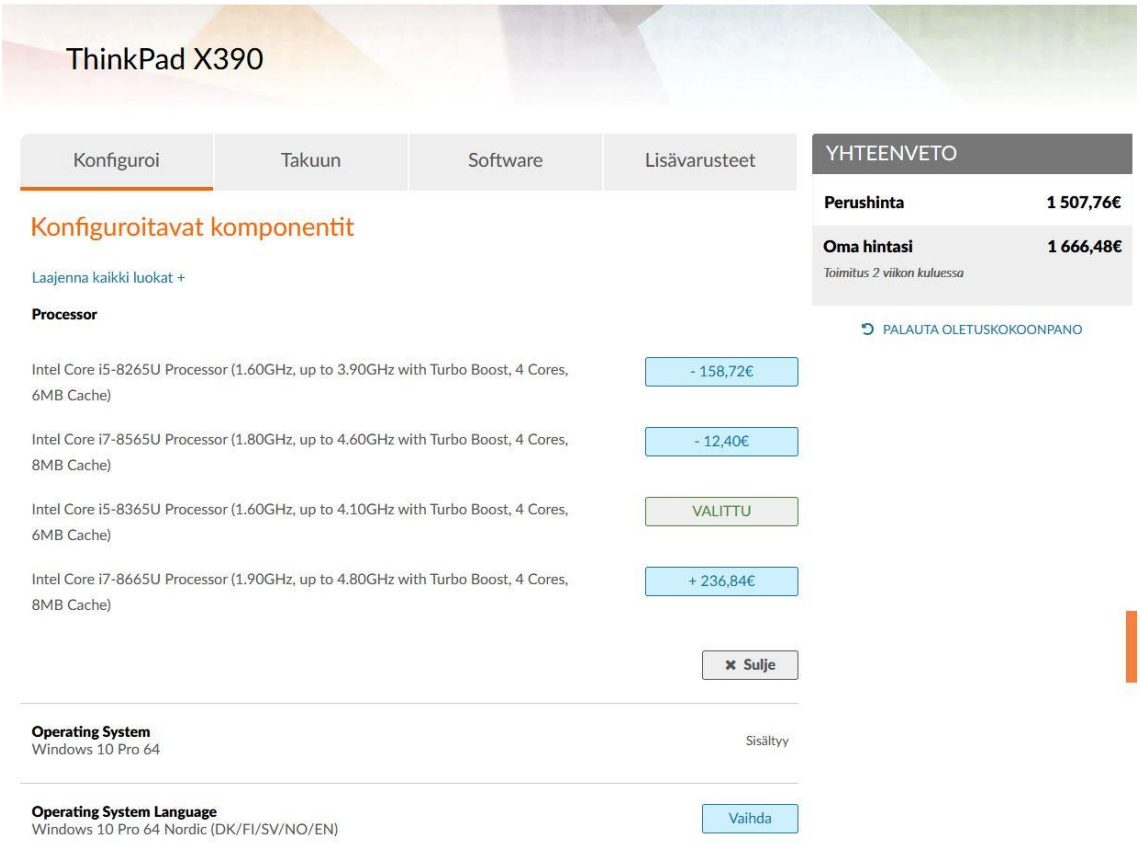
Tuotekonfiguraattorilla tarkoitetaan ohjelmaa, jolla voidaan hallitusti muokata jonkin tuotteen ominaisuuksia, joita kutsutaan myös komponenteiksi. Tällaisia komponentteja voivat olla esimerkiksi fyysisen tuotteen osat, toiset tuotteet tai palvelut, kuten takuu tai huolto. Näitä komponentteja voidaan yhdistää toisiinsa konfiguraattorin kautta, joka varmistaa valittujen komponenttien yhteensopivuuden, ennalta määriteltyjen konfiguraatiosääntöjen perusteella. (What is a Product Configurator? 2019.)

Tuotekonfiguraattori voi olla tarpeellinen jo hyvin pienellä määrällä komponentteja. Esimerkiksi tilanteessa, jossa tuote koostuu kolmesta komponentista, joista jokaisesta on

neljä eri versiota, mahdollisten tuotekokonaisuuksien määrä on 64. Jos taas tuote koostuu neljästä komponentista, joista jokaisesta on viisi eri versiota, mahdollisia kokonaisuuksia on jo 625. Lisäksi komponenttien välillä saattaa olla kokonaisuuksia rajoittavia konfiguraatiossääntöjä, jolloin tuotekokonaisuuksia hallitseminen ilman tuotekonfiguraattoria käy lähes mahdottomaksi. (What is a Product Configurator? 2019.)

Konfiguraatiosäännöt voidaan jakaa ominaisuuksiin perustuviin sääntöihin (engl. characteristic-based) ja tietoon perustuviin sääntöihin (engl. knowledge-based). Ominaisuuksiin perustuvat säännöt, joita kutsutaan myös rajoituksiksi (engl. constraint), ovat passiivisia ja toistuvia riippumatta ulkoisista tekijöistä. Ominaisuuksiin perustuva sääntö voi olla esimerkiksi, "jos komponentti A on valittu, niin komponenttia B ei voi valita". Tietoon perustuvat säännöt muuttavat dynaamisesti saatavilla olevia konfiguraatioita riippuen konfiguraation ulkopuolisesta datasta. Esimerkiksi saatavilla olevia komponentteja voidaan rajoittaa niin, että eri maihin on saatavilla eri komponentteja. (Felgerning ym. 2014.)

Erilaisia tuotekonfiguraattoreita on olemassa niin kuluttajien kuin yritysten käyttöön. Esimerkiksi Lenovon verkkokaupassa asiakas voi räätälöidä itselleen sopivan kannettavan tietokoneen. Verkkokaupassa oleva tuotekonfiguraattori (kuva 1) esittää käyttäjälle tuotteeseen (tietokone) saatavilla olevat komponentit (prosessori, muistin määrä, ...), huolehtii komponenttien yhteensopivuudesta ja ilmoittaa tuotteen lopullisen hinnan.



Kuva 1. Tuotekonfiguraattori Lenovon verkkokaupassa.

3 Salesforce-alusta

Asiakkuudenhallinnalla (CRM, customer relationship management) tarkoitetaan niitä yritysten toimintatapoja ja järjestelmiä, joilla yritykset voivat kehittää ja ylläpitää asiakassuhteitaan. Asiakkuudenhallintarjärjestelmiä (CRM-systems) käytetään asiakkuudenhallinnan prosessien tukena, esimerkiksi myynnin, markkinoinnin ja asiakaspalvelun yhteydessä. (Sahlsten, 2012.)

Salesforce on amerikkalaisen Salesforce.com-yrityksen kehittämä pilvipohjainen asiakkuudenhallintajärjestelmä, joka tarjoaa toiminnallisuuskokonaisuuksia erilaisiin asiakkuudenhallinnan tarpeisiin (What is Salesforce?, 2019). Yksi tällainen kokonaisuus on esimerkiksi Sales Cloud, joka sisältää perustoiminnallisuudet ja datamallin myynti- ja markkinointityön tueksi (Sales Cloud, 2019).

Lisäksi Salesforce tarjoaa palvelualustaa (PaaS, Platform as a Service), joka mahdollistaa omien sovellusten ja toiminnallisuuksien kehittämisen Salesforcen päälle, jotka integroituvat suoraan Salesforceen ja pystyvät käsittelemään sen sisältämää dataa (What is PaaS? - Platform as a Service Explained. 2019).

3.1 Datamalli

Salesforcessa data esitetään tietueena (record), joka on tallennettu objektille (object), joka koostuu kentistä (field). Tietue vastaa hyvin pitkälti taulukkorakenteen riviä, objektitaulua ja kenttäsolua.

Objektit jaetaan vakio-objekteihin (Standard object) ja mukautettuihin objekteihin (Custom object). Vakio-objektit ovat Salesforcen mukana tulevia objekteja, jotka mallintavat esimerkiksi liidiä (Lead), yhteystietoa (Contact) tai asiakasta (Account, suomennettu myös ’tili’). Mukautetut objektit ovat käyttäjien tai ylläpitäjien luomia objekteja, jotka on tarkoitettu mallintamaan yrityksen tai sovelluksen tarvitsemaa dataa, jota vakio-objektit eivät mallinna jo ennestään. Mukautetut objektit erottuvat vakio-objekteista niiden API-nimen lopun ”__c”-päätteestä. Esimerkiksi ”Account” API -nimi viittaa vakio-objektiin ja ”SalesObject__c” viittaa mukautettuun objektiin. Molempia objekteja kutsutaan Salesforce-objekteisi, eli sobjekteiksi. (Understanding Custom & Standard Objects, 2019.) Jatkossa objektilla ja Salesforce-objektilla tarkoitetaan oletuksena näitä objekteja.

Kuten aikaisemmin mainittiin, kaikki objektit koostuvat kentistä. Kentät jaetaan neljään päätyyppiin:

- Tunnistekenttä (Identity field), joka sisältää uniikin, tietuekohtaisen tunnisteen (ID).
- Järjestelmäkenttä (System field), joka sisältää luettavaa (read-only) dataa, kuten tietueen luontihetken aikaleima.
- Nimikenttä (Name field), joka sisältää tietueen selkokielisen tunnisteen.
- Mukautettu kenttä (Custom field), joka on käyttäjän objektille luoma kenttä.

Tunniste-, järjestelmä- ja nimikentät ovat vakiokenttiä, jotka löytyvät kaikilta niin vakio kuin mukautetuilta objekteilta. Mukautetut kentät ovat käyttäjien objektille luomia kenttiä.

Kuten mukautetut objektit myös mukautetut kentät voivat erottaa vakiokentistä API-nimen lopussa olevasta ”__c”-päätteestä. (Understanding Custom & Standard Objects, 2019.)

Kaikilla kentillä on nimi, API-nimi ja tietotyyppi. Nimi on käyttöliittymässä näytettävä kentän selkokielinen nimi. API-nimi on kentän uniikki, objektikohtainen tunniste, jota käytetään kenttäviittauksissa esimerkiksi ohjelmakoodissa. Tietotyyppi kertoo kentän sisältämän datan tyypin, kuten päivämäärän (Date), tekstin (Text) tai valuutan (Currency). Kentän on mahdollista viitata myös toisen tietueen tunnistekenttään (id), jolloin puhutaan Lookup-kentästä. Lookup-kenttää luodessa kentälle on määriteltävä kohdeobjekti, jonka tietueisiin kenttä viittaa. (Understanding Custom & Standard Objects, 2019.)

Taulukossa 1 on esimerkki kolmesta tiliobjektin (Account) tietueesta.

Taulukko 1. Esimerkki Salesforce-objektiin tallennetusta datasta.

Id	**Name**	**CreatedDate**	**Description__c**
001D000000IRt53	Yritys Oy	2015-01-21T03:00:000Z	Pieni osakeyhtiö
001D000000IRt54	Firma Oyj	2015-01-21T03:00:000Z	Suuri osakeyhtiö
001D000000IRt55	Korporaatio ky	2015-01-21T03:30:000Z	Keskikokoinen kommandiittiyhtiö

Ensimmäisellä rivillä on objektin kenttien API-nimiä. Ensimmäisessä sarakkeessa on ID-tyyppinen tietuetunnistemerkkijono, joka on uniikki jokaisella tallennetulla tietueella.

3.2 Apex-ohjelmointikieli

Apex on vahvasti tyypitetty, oliopohjainen ohjelmointikieli, jonka avulla kehittäjät voivat kirjoittaa ohjelmalogiikkaa ajettavaksi Salesforcen päällä. Apex on syntaksiltaan hyvin lähellä Java-ohjelmointikieltä. (Introducing Apex, 2019.) Suorituksen alkaessa Apex-luokka käännetään Java-tavukoodiksi, jonka Apex-tulkki ajaa Java-virtuaalikoneen päällä (Fee & Gallagher. 2010).

Apex hyödyntää Javan tapaan primitiivejä, kuten Integeriä ja Stringiä sekä myös abstrakteja tietotyyppejä, kuten hakurakennetta (Map) ja joukkoa (Set). Lisäksi Apex tarjoaa pääsyn Salesforcen objektien ja näiden tietueiden käsittelyyn. Apex käsittelee kaikkia Salesforce-objekteja luokkina ja tietueita luokkien ilmentyminä. (Understanding Apex Core Concepts. 2019.)

Esimerkkikoodissa 1 on Apex-esimerkkiohjelma, joka tulostaa Salesforcen debug-konsoliin luvun 10.

```
// Primitiivimuuttujien määrittely
String mjono = 'Merkkijono';
Integer kluku = 10;
Boolean onTosi = true;

// Hakurakenteen käyttäminen
Map<String, Integer> keyToValue = new Map<String, Integer>();
keyToValue.put(mjono, kluku);

if (onTosi) {
     for (String key : keyToValue.keySet()) {
          System.debug(keyToValue.get(key));
     }
}
```

Esimerkkikoodi 1. Apex-syntaksia ja esimerkkiohjelma.

Kuten aikaisemmin mainittiin, Apex tarjoaa toimintoja Salesforce-objektien käsittelyyn. Nämä ovat Data Manipulation Language (DML, jota ei tule sekoittaa samannimiseen käsitteeseen) ja Salesforce Object Query Language (SOQL). Yksinkertaistettuna SOQL:a käytetään tietueiden kutsumiseen, Apexia tietueiden käsittelyyn ja DML:a tietueiden muokkaamiseen.

Apex sisältää neljä DML-operaatiota:

- Insertiä käytetään uuden tietueen luontiin.
- Updatea käytetään olemassa olevan tietueen päivittämiseen.
- Deleteä käytetään olemassa olevan tietueen poistamiseen.
- Upsertia käytetään uusien tietueiden luontiin ja olemassa olevien päivittämiseen samassa tapahtumassa (transaction).

DML-operaatioista kaikki paitsi insert vaativat toimiakseen tietuetunnisteen. Mikäli esimerkiksi update-operaatiota yrittää suorittaa tietueelle, jolla ei ole tietuetunnistetta, tietokantaoperaatio epäonnistuu. (sObject Types. 2019.)

Esimerkkikoodissa 2 esitellään tietueen hakeminen SOQL:a, muokkaaminen Apexilla ja tallentaminen DML:a.

```
// Luodaan uusi Account-sObject ja tallennetaan se muuttujaan
Account acc = new Account(Name = 'Yritys Oy');

// Tallennetaan sObjekti
insert acc;

// Haetaan SOQL:a Account-objektilta yksi tietue muuttujaan
acc = [SELECT Id, Name, Description__c FROM Account WHERE Name = 'Yritys Oy'
LIMIT 1];

// Muutetaan tietueen kentän arvoa
acc.Description__c = 'Muuttunut kentän kuvaus';

// Tallennetaan muutos
update acc;
```

Esimerkkikoodi 2. Tietokantaoperaatioita ja sObjektin käsittelyä.

4 Salesforce CPQ:n myyntikonfiguraattori

Salesforce CPQ on Salesforcen Sales Cloudin päällä toimiva myyntikonfiguraattori, joka integroituu suoraan Salesforcen muihin toiminnallisuuksiin. Salesforce CPQ tarjoaa toiminnallisuuksia tarjousten luontiin ja hallintaan, tuotteiden ja tarjousten automaattiseen hinnoitteluun ja tuotekonfiguraattorin. (What is Salesforce CPQ? 2019.)

Tässä luvussa esitellään työn kannalta tärkeät Salesforce CPQ:n datamallin objektit sekä tutustutaan Salesforce CPQ:n sisältämään tuotekonfiguraattoriin.

4.1 CPQ-datamalli

Salesforce CPQ tuo mukanaan lukuisia mukautettuja objekteja, jotka muodostavat myyntikonfiguraattorin datamallin. Osa uusista objekteista mallintaa kokonaan uusia

ominaisuuksia, kuten tuoterajoituksia, kun taas osa korvaa tai laajentaa jo olemassa olevia toiminnallisuuksia, kuten tarjous tai tarjousrivi.

Seuraavaksi esitellään tämän työn kannalta oleelliset mukautetut objektit.

Tarjous (SBQQ__Quote__c)

Tarjous kuvastaa käsittelyssä olevaa tarjousta, johon konfiguroidut tuotteet lisätään tarjousriveinä. Tarjoukseen liittyy aina asiakastili (Account) sekä hintakirja (Pricebook2).

Tarjousrivi (SBQQ__QuoteLine__c)

Tarjousrivi kuvastaa tarjoukselle liitettyä tuotetta. Jokaisesta tarjoukselle lisätystä tuotteesta on olemassa yksi tarjousrivi, joka sisältää tiedon mm. tuotteen lopullisesta hinnasta, rivikohtaisesta alennuksesta sekä tuotteen lukumäärästä. Tuotepakettien kohdalla jokainen paketin tuote muodostaa oman tarjousrivinsä, jotka liittyvät lookup-kenttäviittauksella tuotepaketin ylimmän tason tarjousriviin.

Tuotevaihtoehdot (SBQQ__ProductOption__c)

Tuotevaihtoehdot yhdistävät tuotepakettien ylimmän tason pakettituotteen siihen saatavilla oleviin tuotteisiin. Tuotevaihtoehto tietueessa voidaan myös määrittää oletusarvoja esimerkiksi alennusprosentille ja vähimmäistuotemäärälle, jotka Salesforce CPQ huomioi, kun linkitetystä tuotteesta luodaan tarjousriviä.

Vaihtoehtorajoitukset (SBQQ__OptionConstraint__c)

Vaihtoehtorajoitukset mallintavat tuotevaihtoehtojen rajoitussääntöjä silloin, kun tuotepakettia konfiguroidaan. Jokaiselle rajoitukselle on määriteltävä

- rajoittava tuotevaihtoehto
- rajoitettu tuotevaihtoehto
- tuotepaketti, jonka tuotevaihtoehtoja rajoitus koskee

- rajoitusoperaatio
- valinnainen rajoitusryhmittely.

Rajoittava tuotevaihtoehto ja rajoitusoperaatio muodostavat yhdessä rajoituslausekkeen, jonka perusteella estetään tai sallitaan rajoitettavan tuotevaihtoehdon lisääminen tuotepakettiin.

Rajoitusoperaatio määrää sen, onko kyseessä eksklusiivinen vai inklusiivinen vaihtoehtorajoitus. Inklusiivisessa vaihtoehtorajoituksessa rajoitettu tuotevaihtoehto voidaan valita tuotepakettiin vain, jos rajoittava tuotevaihtoehto on valittu. Päinvastoin eksklusiivisessa vaihtoehtorajoituksessa rajoitettu tuotevaihtoehto voidaan valita vain, jos rajoittava tuotevaihtoehto ei ole valittu.

Vaihtoehtorajoituksia voidaan myös ryhmittää. Tällöin jokaisen samaan ryhmään kuuluvan vaihtoehtorajoituksen rajoitusoperaation ja rajoitetun tuotevaihtoehdon on oltava sama. Esimerkiksi tuotevaihtoehto A on valittavissa vain, jos kaikki vaihtoehdot B, C ja D eivät ole valittuna.

Tuoteominaisuudet (SBQQ__ProductFeature__c)

Tuoteominaisuudet kuvaavat ryhmiä, joiden alle tuotevaihtoehtoja voidaan lisätä esitettäväksi konfiguraattorissa. Yksittäinen tuoteominaisuus voi vaihtoehtorajoitusten lisäksi rajoittaa (tai pakottaa) käyttäjän valitsemaan vähimmäis- / enimmäismäärän tuotevaihtoehtoja. Esimerkiksi jos tuoteominaisuudella ”Feature A” on määriteltynä minimimäärä yksi ja enimmäismäärä kolme, siihen liittyvistä tuotevaihtoehdoista tulee konfiguraatiossa valita vähintään yksi, mutta enintään kolme vaihtoehtoa.

Tuotesäännöt (SBQQ__ProductRule__c)

Tuotesäännöt ovat konfiguroitavia sääntöjä, jotka koostuvat yhdestä tai useammasta ehtolausekkeesta tai hakukyselystä sekä mahdollisista tuotetoiminnosta. Tuotesäännöt ajetaan valinnan mukaan yhdessä seuraavista neljästä tapauksesta:

- Käyttäjä avaa konfiguraattorin tai tarjouseditorin (quote line editor).

- Käyttäjä muokkaa tarjousrivitietueen kenttää.
- Käyttäjä tallentaa tarjouksen tarjouseditorissa.
- Käyttäjä avaa, muokkaa tai tallentaa tarjouksen tarjouseditorissa.

Kun tuotesääntö ajetaan, Salesforce CPQ ratkaisee kaikki sääntöön liittyvät ehtolausekkeet ja ratkaisee niiden perusteella tuotesäännön oman ehtolausekkeen. Tämä ehtolauseke määrää, millä ehtolausekkeiden arvoilla tuotesääntöön liittyvät tuotetoiminnot ajetaan. Jokaiselle tuotesäännölle on määritettävä yksi kolmesta ehtolausekkeesta:

- Kaikki tuotetoiminnon ehtolausekkeet ovat totta (All).
- Yksikin tuotetoiminnon ehtolauseke on totta (Any).
- Käyttäjän määrittelemä ehtolauseke on totta (Custom).

Mikäli tuotesäännössä käytetään mukautettua ehtolauseketta, on erillisessä kentässä määriteltävä ehtolauseke, jossa viitataan numeroin tuotesäännön ehtolausekkeisiin.

Esimerkiksi lausekkeessa (1 AND 2) OR (3 AND 4) tuotesäännön ehtolausekkeiden 1 ja 2 tai 3 ja 4 on oltava totta, jotta tuotesääntöön liittyvät tuotetoiminnot ajetaan.

Ehtolausekkeet (SBQQ__ErrorCondition__c)

Ehtolausekkeet ovat tietueita, joista jokainen sisältää sellaiset tiedot, kuten kenttäviittauksen, vertailuoperaation ja vertailuarvon, joiden avulla voidaan muodostaa lauseke, jonka arvo on tosi tai epätosi (true / false). Ehtolauseke kohdistuu aina tarjoukseen, tarjousriviin tai konfiguroitavan tuotteen tuotevaihtoehtoihin.

Tuotetoiminnot (SBQQ__ProductAction__c)

Tuotetoiminnot ovat tietueita, joiden perusteella Salesforce CPQ tekee tuotevaihtoehtoihin (Product option) kohdistuvia toiminnallisuuksia, kuten vaihtoehdon piilottamisen tai automaattisen valinnan. Tuotetoiminnot ovat vaihtoehtorajoitusten kanssa hyvin samanlaisia, mutta mahdollistavat myös esimerkiksi tuotevaihtoehtojen automaattisen valitsemisen konfiguraatiosta.

Tuotevaihtoehtoihin voidaan kohdistaa seuraavat toimenpiteet:

- valitse
- poista valinta
- aktivoi valinta (enable)
- inaktivoi valinta (disable)
- aktivoi ja valitse
- inaktivoi ja poista -valinta
- näytä valinta
- piilota valinta
- näytä ja valitse
- piilota ja poista -valinta

Tuotetoiminto sisältää yhden edellä mainituista toimenpiteistä sekä viittauksen tuotteeseen, jota vastaavalle tuotevaihtoehdolle kyseinen toimenpide tehdään.

Hakukyselyt (SBQQ__LookupQuery__c)

Hakukyselyt ovat ehtolausekkeita, joiden avulla voidaan kohdistaa vertailuja sellaisten objektien tietueisiin, jotka eivät liity konfiguroitavaan tarjoustietueeseen. Tuotesääntötietue sisältää kentän (SBQQ__LookupObject__c), jonka sisältämää objektia vasten sääntöön liittyviä hakukyselyjä verrataan. Yksi hakukyselytietue sisältää tiedot sellaisesta tuotesäännöllä määritellyn objektin kentästä, jota halutaan verrata johonkin tilauksen, tilausrivin tai tuotevaihtoehdon kenttään.

Yhteenvetomuuttujat (SBQQ__SummaryVariable__c)

Yhteenvetomuuttujat ovat tietueita, joiden avulla usean tietueen sisältämään dataan voidaan kohdistaa matemaattinen operaatio. Yhteenvetomuuttujan kenttien arvojen perus-

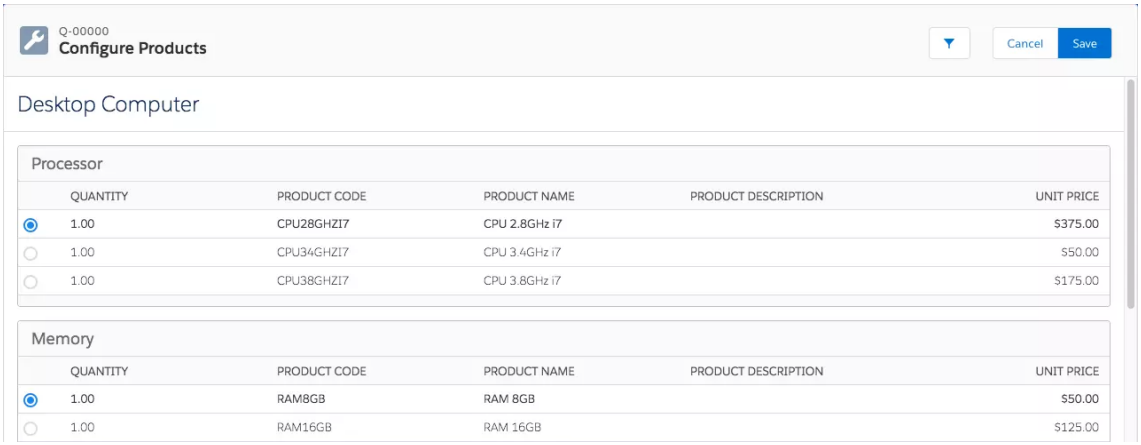
teella Salesforce luo taustalla SOQL-kyselyn, joka palauttaa lukuarvon. Jokainen yhteenvetomuuttuja kohdistuu aina yhden objektin yhteen numeerista dataa sisältävään kenttään.

Yhteenvetomuuttujilla voidaan määrittää neljä matemaattista operaatiota: summa, keskiarvo, pienin ja suurin. Operaatio kohdistuu sellaisiin tarjoukseen liittyviin tietueisiin, jotka vastaavat yhteenvetomuuttujalla määriteltyä objektia ja suodatinta.

Esimerkiksi tarjousriveihin voidaan kohdistaa yhteenvetomuuttuja, joka laskee yhteen rivien hintakentät, huomioiden vain sellaiset rivit, joiden sisältämän tuotteen tyyppi on ”komponentti”.

4.2 Salesforce CPQ:n tuotekonfiguraattori

Salesforce CPQ sisältää myös oman tuotekonfiguraattorin. Konfiguraattori (kuva 2) käynnistyy, kun Salesforcen sisäinen käyttäjä lisää tarjoukselle tuotteen, joka on määritelty konfiguroitavaksi.



Kuva 2. Salesforce CPQ:n tuotekonfiguraattorin käyttöliittymä.

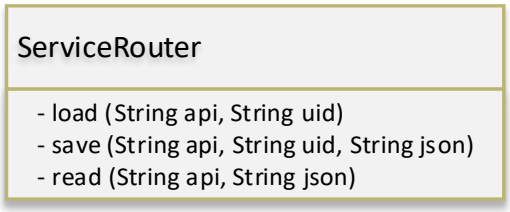
Koska Salesforce CPQ on niin kutsuttu hallittu paketti (engl. managed package), kaikki siihen liittyvät Apex-luokat on määritelty niin, että niiden lähdekoodia ei voi tarkastella

(Components Available in Managed Packages. 2019). Täten näiden Apex-luokkien hyödyntäminen mukautetun tuotekonfiguraattorin toteutuksessa ei ole mahdollista.

4.3 QuoteAPI-ohjelmointirajapinta

Salesforce CPQ mahdollistaa tarjousten käsittelyn ohjelmointirajapinnan kautta. Rajapintaa pystyy käyttämään joko autentikoiduilla REST-kutsuilla tai Salesforcen sisältä Apexilla hyödyntämällä Salesforce CPQ:n mukana tulevaa ServiceRouter-luokkaa. ServiceRouter-luokan käyttämiseksi vaaditaan myös Apex-luokat, jotka mallintavat QuoteAPI:n datamallia. (Service Router, 2019.)

ServiceRouter-luokka tarjoaa käyttöön kolme metodia, joista jokainen vastaanottaa parametrina käytetyn rajapinnan nimen, Salesforce ID:n ja / tai JSON-muotoisen merkkijonon, joka sisältää käytetyn toiminnon mukaan tarvittavaa dataa (Service Router, 2019). Kuvassa 3 on esitelty ServiceRouter-luokan metodit.



Kuva 3. ServiceRouter-luokan UML.

Taulukossa 2 on esitelty ServiceRouterin metodien hyväksymien rajapintojen nimet sekä käyttötarkoitukset.

Taulukko 2. QuoteAPI-rajapintojen nimet ServiceRouterissa.

Nimi	**Service-Router metodit**	**Kuvaus**
SBQQ.ProductAPI.ProductLoader	load	Hakee tuotten (Product2) Salesforcesta.
SBQQ.QuoteAPI.QuoteReader	read	Hakee tarjoustietueen Salesforcesta.

SBQQ.QuoteAPI.QuoteProductAdder	load	Lisää ProductLoaderilla haetun tuotteen QuoteReaderilla haetulle tarjoukselle.
SBQQ.QuoteAPI.QuoteSaver	save	Tallentaa QuoteReadeilla haetun tarjoustietueen Salesforceen.

Esimerkkikoodissa 3 esitellään QuoteAPI-rajapinnan käyttöä ServiceRouter-luokan kautta. Esimerkissä rajapinnan kautta haetaan ja tallennetaan Salesforcessa oleva tarjoustietue. Ennen tallentamista ServiceRouterin palauttama JSON-muotoinen vastaus muunnetaan QuoteAPI:n datamallin mukaiseksi QuoteModel-luokan ilmentymäksi.

```
// Tarjouksen tietue ID
String quoteId = 'a0Wf100000J1vk1';

// Tuotteen tietue ID
String productId = '01tj0000003P1SN';

// Tarjouksen hintakirjan tietue ID
String pricebookId = '01sj0000003THhKAAW';

// Käytetyn QuoteAPI:n nimi tallennettuna muuttujaan
String READ_QUOTE = 'SBQQ.QuoteAPI.QuoteReader';

// Haetaan tietue rajapinnasta
String json = SBQQ.ServiceRouter.read(READ_QUOTE, quoteId);

// Muunnetaan JSON QuoteModel-luokan ilmentymäksi
QuoteModel model = (QuoteModel)JSON.deserialize(json, QuoteModel.class);
```

Esimerkkikoodi 3. QuoteAPI-rajapinnan kutsuminen tarjouksen hakemiseksi.

5 Mukautetun tuotekonfiguraattorin toteutus

Työssä toteutettu tuotekonfiguraattori pystyy rakentamaan Salesforce CPQ:ssa määriteltyjen sääntöjen ja rajoitusten mukaisen tuotepaketin. Työssä toteutettu ohjelma keskittyy tuotekonfiguraattorin backend-toiminnallisuuksiin, kuten tuotesääntöjen laskentaan, eikä ota kantaa mahdollisiin käyttöliittymätoteutuksiin. Työssä kehitetyt Apex-luokat ja datamalli rakennettiin niin, että tuotekonfiguraattorin toiminnallisuuksia on mahdollista hyödyntää helposti käyttöliittymätoteutuksissa.

5.1 Toteutuksen rajaus

Ennen työn toteuttamista tunnistettiin tarpeelliset Salesforce CPQ:n toiminnallisuudet:

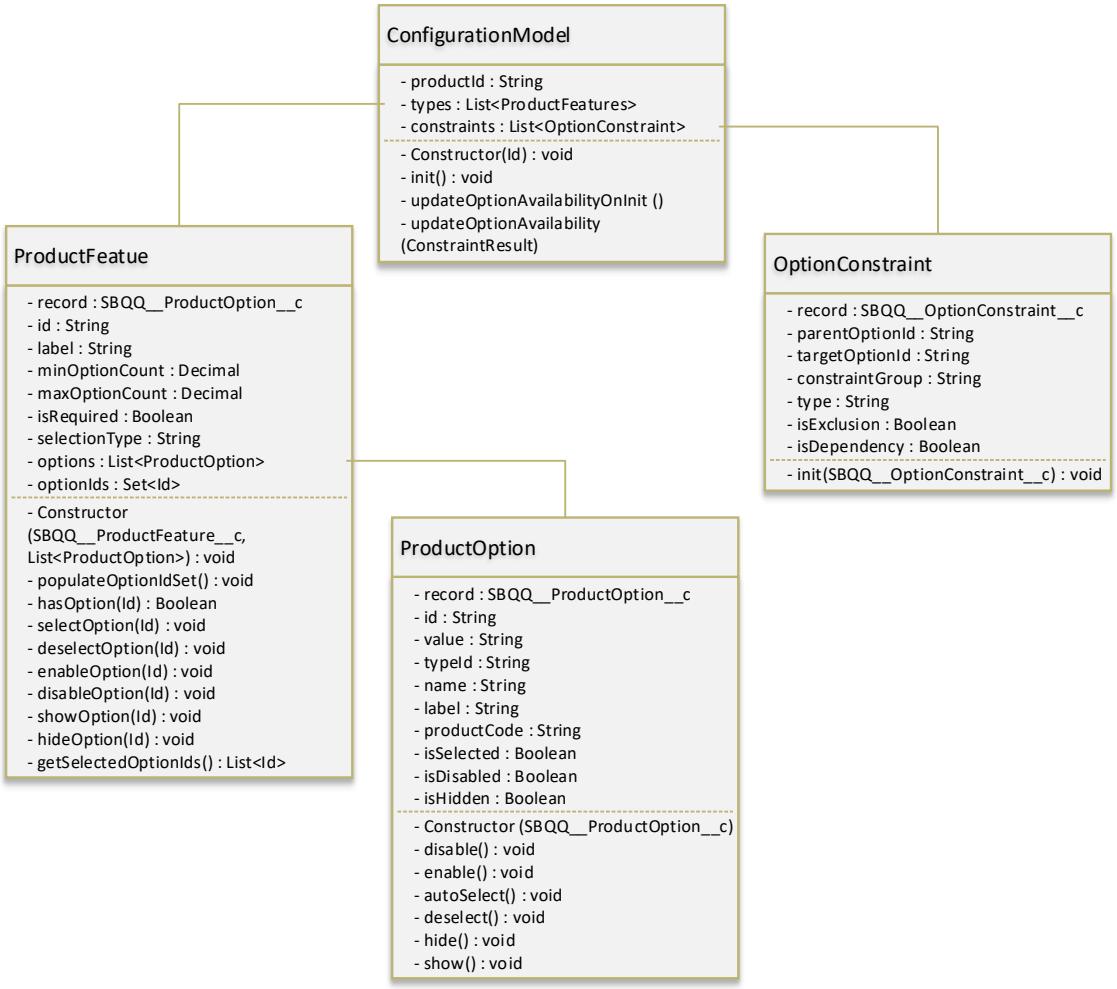
- konfiguroitavan tuotteen valitseminen
- tuotevaihtoehtojen valitseminen
- konfiguroidun tuotteen tallentaminen / lisääminen tarjoukselle.

Näiden toimintojen pohjalta määriteltiin konfiguroinnin kannalta tärkeimmät Salesforce CPQ:n datamallin mukautetut objektit ovat tuotevaihtoehdot, vaihtoehtorajoitukset sekä tuotesäännöt. Tuotesääntöihin liittyvät lisäksi ehtolausekkeet, yhteenvetomuuttujat, tuotetoiminnot, hakukyselyt ja konfigurointimäärittelyt. Konfiguroidun tuotteen lisäämiseen tarjoukselle hyödynnettiin QuoteAPI-ohjelmointirajapintaa, joka huolehti tietueiden tallentamisesta ja muokkaamisesta.

Seuraavissa luvuissa tarkastellaan yksi kerrallaan tuotevaihtoehtojen, vaihtoehtorajoitusten ja tuotesääntöjen toteutuksia.

5.2 Konfiguroitavan tuotteen datamalli

Tuotteen konfigurointia varten luotiin konfiguraation sen hetkistä tilaa kuvaava datamalli, joka koostui useasta Apex-luokasta. Luokat mallinsivat läheisesti Salesforce CPQ:n mukautettujen objektien datamallia, joihin lisättiin konfigurointia helpottavia toiminnallisuuksia, kuten metodikutsu tuotevaihtoehdon valitsemiseksi.



Kuva 4. Konfiguroidun tuotteen tilaa kuvaava datamalli.

Kuvan 4 datamallissa ConfigurationModel kuvastaa konfiguroitavaa tuotetta. Tämän alle datamallissa rakennettiin tuoteominaisuudet (kuvassa ProductFeature), jotka sisältävät listan kyseiseen ominaisuuteen kuuluvista tuotevaihtoehdoista.

Esmierkkikoodissa 4 esitellään, kuinka ConfigurationModel-luokasta luodaan olio tuote-tietueen tietuetunnisteen perusteella. Ilmentymisen yhteydessä ConfigurationModel-olio hakee Salesforcesta tietueeseen liittyvät tuotevaihtoehdot, vaihtoehtorajoitukset sekä tuoteominaisuudet.

```
// Haetaan Salesforcen Tuote-objektilta yksi tietue
Product2 product = [SELECT Id FROM Product2 LIMIT 1];
```

```
// Luodaan ConfigurationModel-olio
ConfigurationModel productModel = new ConfigurationModel(product.Id);
```

Esimerkkikoodi 4. ConfigurationModel-olion luominen tuotetietueen tunnisteen perusteella.

Toisin kuin Salesforce CPQ:n datamallissa, luodussa datamallissa vaihtoehtorajoitukset (kuvassa OptionConstraint) eivät liity suoralla viittauksella tuotevaihtoehtoihin. Sen sijaan ConfigurationModel sisältää listan vaihtoehtorajoituksia mallintavista OptionConstraint-olioista. OptionConstraint-oliot sisältävät tietuetunnisteen rajoittavasta- ja rajoitetusta tuotevaihtoehdosta sekä rajoitustyypin, jotka haetaan olion luonnin yhteydessä SBQQ__OptionConstraint__c-tietueelta.

Esimerkkikoodissa 5 on JSON-muodossa kuvattu yksinkertaisen ConfigurationModel-olion sisältämä data.

```
{
      "productId": "product_record_id",
      "types": [
      {
            "selectionType": "checkbox",
            "Name": "Vaihtoehtoryhmä A",
            "minOptionCount": 0,
            "maxOptionCount": 2,
            "label": "Tuoteominaisuus 1",
            "isRequired": false,
            "id": "feature_group_1",
            "options": [
            {
                  "typeId": "feature_group_1",
                  "productCode": "P-1234",
                  "name": "Tuotevaihtoehto A",
                  "label": "Tuotevaihtoehto A",
                  "isSelected": false,
                  "isDisabled": false,
                  "id": "product_option_a"
            },
            {
                  "typeId": "feature_group_1",
                  "productCode": "P-1200",
                  "name": "Tuotevaihtoehto B",
                  "label": "Tuotevaihtoehto B",
                  "isSelected": false,
                  "isDisabled": false,
                  "id": "product_option_b"
            },
      ],
      "optionIds": [
            "product_option_a",
            "product_option_b"
      ]
      }
      ],
```

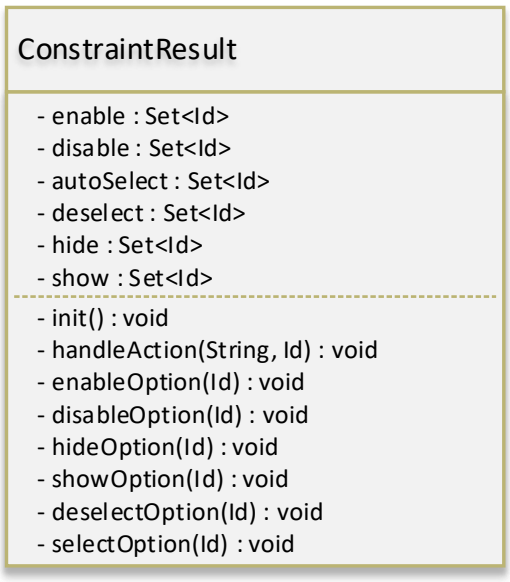
```
      "constraints": {
            "product_option_a": [
            {
                  "type": "Exclusion",
                  "targetOptionId": "product_option_b",
                  "parentOptionId": "product_option_a",
                  "isExclusion": true,
                  "isDependency": false,
                  "constraintGroup": null
            }
            ]
      },
      "messages" : []
}
```

Esimerkkikoodi 5. JSON-muotoinen ConfigurationModel-olio ilmentymisen jälkeen. Tietuetunnisteet muutettu.

Esimerkkikoodissa konfiguroitavalla tuotteella on kaksi tuotevaihtoehtoa ja yksi vaihtoehtorajoitus. Esimerkkikoodissa tietueiden tunnisteet on lukemisen helpottamiseksi korvattu selventävillä merkkijonoilla, kuten ”product_option_a”.

5.3 Vaihtoehtorajoitusten laskennan toteutus

Kuten luvussa 4.1 mainittiin, konfiguroitavalle tuotteelle on mahdollista määritellä tuotevaihtoehtoja, joiden valitsemista pystytään rajoittamaan vaihtoehtorajoituksilla. Rajoitusten käsittelemiseksi luotiin ConfigurationCommander-niminen Apex-luokka, joka laskee tuotevaihtoehdoille tarvittavat valintamuutokset. Laskenta tehdään perustuen tuoterajoituksiin, aikaisemmin valittuihin tuotevaihtoehtoihin ja nyt valittuun tuotevaihtoehtoon. Laskennan avuksi luotiin ConstraintResult-niminen luokka, jonka ilmentymään vaihtoehtorajoitusten aiheuttamat tuotevaihtoehtojen valintamuutokset tallentuvat ajon ajaksi.



Kuva 5. ConstraintResult-luokka.

ConstraintResult-luokka sisältää joukkomuuttujan (engl. Set variable) jokaiselle käsittelytapahtumalle, jonka tuotevaihtoehtoihin voi kohdistaa. Mahdolliset käsittelytapahtumat on listattu luvussa 4.1 tuotesääntöjen yhteydessä. ConstraintResult-oliota kutsutaan aina esimerkkikoodin 6 mukaisella metodilla, jonka perusteella olio päättelee ne joukkomuuttujat, joihin sille välitetty tietuetunniste tulee lisätä.

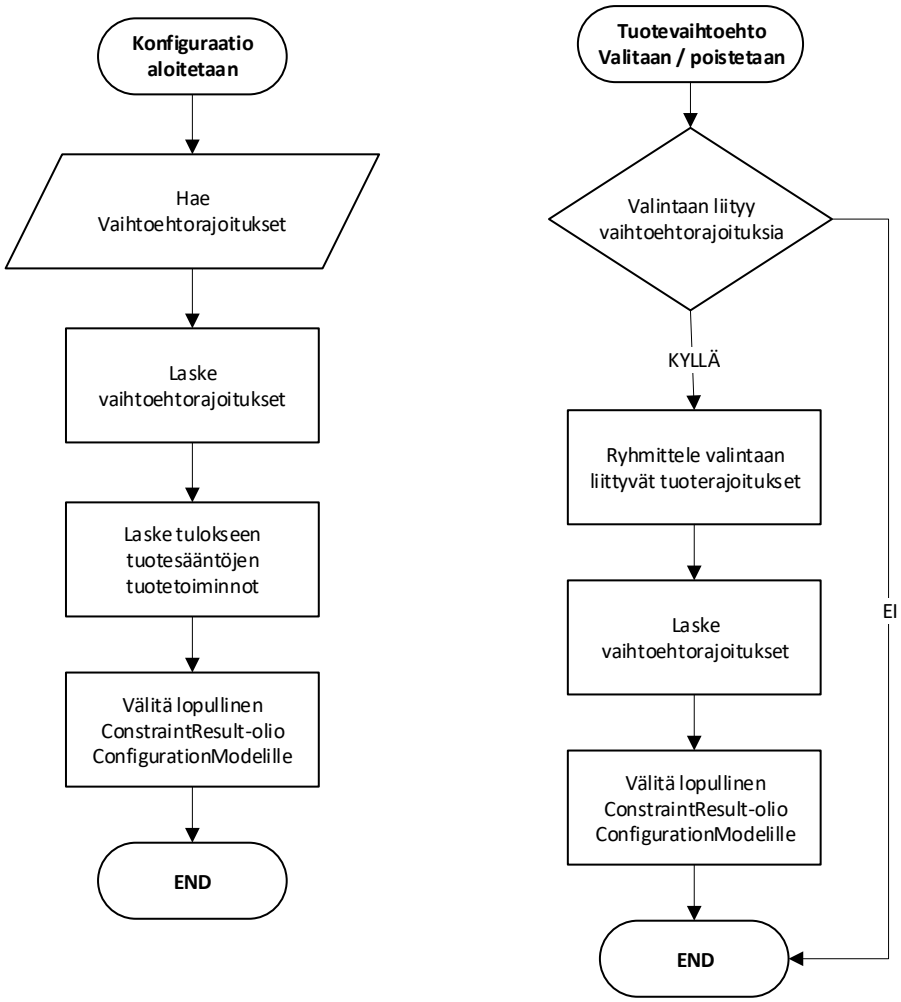
```
public void handleAction(String actionDescription, Id optionId) {
      switch on actionDescription {
            when 'Add' {
                  selectOption(optionId);
            }
            when 'Remove' {
                  deselectOption(optionId);
            }
            when 'Enable' {
                  enableOption(optionId);
            }
            when 'Disable' {
                  disableOption(optionId);
            }
            when 'Enable & Add' {
                  enableOption(optionId);
                  selectOption(optionId);
            }
            when 'Disable & Remove' {
                  disableOption(optionId);
                  deselectOption(optionId);
            }
            when 'Show' {
                  showOption(optionId);
```

```
            }
            when 'Hide' {
                  hideOption(optionId);
            }
            when 'Show & Add' {
                  showOption(optionId);
                  selectOption(optionId);
            }
            when 'Hide & Remove' {
                  hideOption(optionId);
                  deselectOption(optionId);
            }
      }
}
```

Esimerkkikoodi 6. ConstraintResult-luokan metodikutsu tietuetunnisteen lisäämiseksi oikeisiin joukkomuuttujiin.

Ohjelmoinnin helpottamiseksi metodin switch-when-rakenteessa päädyttiin käyttämään samoja arvoja, kuin Salesforce CPQ:n tuotetoiminnoissa, jotka esiteltiin luvussa 4.1.

Vaihtoehtorajoitusten laskemishetkiä on kaksi: kun konfigurointi aloitetaan, ja kun jokin tuotevaihtoehto valitaan tai poistetaan. Kun laskenta aloitetaan, ConfigurationCommander hakee Salesforcesta kaikki konfiguroitavaan tuotteeseen liittyvät vaihtoehtorajoitustietueet. Jokaisen tietueen rajoitetun tuotevaihtoehdon tietuetunniste lisätään rajoitustoiminnon mukaan ConstraintResult-olion aktivointi tai inaktivointi joukkomuuttujaan. Tällöin luodaan konfiguraatiolle alkutilanne, jossa osa tuotevaihtoehdoista on valittavissa ja osa ei. Aloituslaskennassa huomioitiin myös tuotesääntöjen yhteydessä mahdollisesti jo valitut tuotevaihtoehdot, sekä näiden vaikutus tuoterajoituksen, tai rajoitusryhmän, lopputulokseen. Tuotesääntöjen vaikutusta laskennassa tarkastellaan tarkemmin luvussa 5.4.4. Kuvan 6 kaavioissa on esiteltynä molempien laskutapahtumien kulku.



Kuva 6. Tuoterajoitusten laskutapahtumat.

Tuotevaihtoehdon valinnan ja poiston yhteydessä tapahtuneessa laskennassa ConfigurationCommanderille välitetään käsitellyn tuotevaihtoehdon tietuetunniste, sekä lista kaikista jo valituista tuotevaihtoehdoista. Tämän jälkeen ConfigurationCommander selvittää, liittyykö nyt valittu tuotevaihtoehto johonkin vaihtoehtorajoitukseen, jonka jälkeen aikaisempien valintojen perusteella päätellään, ollaanko tuotevaihtoehtoa valitsemassa vai poistamassa. Sen jälkeen ConfigurationCommander ryhmittelee tuotevaihtoehtoon liittyvät vaihtoehtorajoitukset mahdollisten rajoitusryhmien mukaan. Sitten jokaisen rajoitusryhmän kohdalla tarkastetaan, ovatko kaikki ryhmään kuuluvien vaihtoehtorajoitusten tuotevaihtoehdot valittuina, jolloin ryhmän vaihtoehtorajoitukset otetaan laskettaviksi.

Lopuksi vaihtoehtorajoitukset lasketaan ja tallennetaan ConstraintResult-olioon. Olio välitetään takaisin ConfigurationModelille, joka välittää tuotevaihtoehtojen tietuetunnisteet sisältämilleen ProductFeature-olioille, jotka toteuttavat tarvittavan valintamuutoksen.

```
public void selectOption(Id optionId) {
     if (!this.hasOption(optionId))
          return;

     for (ProductOption option : this.options) {
          if (this.selectionType == 'radio') {
               if (option.id == optionId) {
                    if (!option.isSelected)
                         option.autoSelect();
               } else {
                    option.deselect();
               }
          } else {
               if (option.id == optionId) {
                    option.isSelected = !option.isSelected;
                    return;
               }
          }
     }
}
```

Esimerkkikoodi 7. Esimerkkimetodi ProductFeature-luokasta tuotevaihtoehdon valitsemiseksi.

Esimerkkikoodissa 7 enableOption-metodille välitetään tuotevaihtoehdon tietuetunniste. Metodi varmistaa ensin hasOption-joukosta, että kyseinen tunniste liittyy olion mallintamaan tuoteominaisuuteen. Tämän jälkeen metodi käy olion tuotevaihtoehdot läpi ja valitsee tunnistetta vastaavan vaihtoehdon.

Mikäli laskennan aikana jokin jo valittu tuotevaihtoehto, johon vaikuttaa muita vaihtoehtorajoituksia, inaktivoidaan, suoritetaan laskenta rekursiivisesti uudestaan kyseisen tuotevaihtoehdon rajoituksille.

5.4 Tuotesääntöjen laskennan toteutus

Tuotesääntöjen laskemiseksi luotiin ProductRuleCalculator-niminen Apex-luokka. Koska osa tuotesäännöistä saattaa kohdistua konfiguroitavan tuotteen tai tarjouksen ominaisuuksiin, ilmentymisen yhteydessä luokka vaatii, että sille välitetään konfiguroitavan tuot-

teen tietuetunniste, tarjouksen tietuetunniste ja lista mahdollisista jo valittujen tuotevaihtoehtojen tietuetunnisteista. Laskentaa varten luokka hakee ilmentyessään seuraavat tietueet:

- konfiguroitavaan tuotteeseen liittyvät tuotesäännöt
- tuotesääntöjen ehtolausekkeet
- tuotesääntöjen hakukyselylausekkeet
- ehtolausekkeiden yhteenvetomuuttujat
- tuotesääntöjen tuotetoiminnot.

Luokka muodostaa, esimerkkikoodissa 8 kuvatulla tavalla, tuotetoiminnoille sekä hakukysely- ja ehtolausekkeille hakurakenteet, jotka palauttavat tuotesäännön tietuetunnisteen perusteella listan sääntöön liittyvistä tietueista.

```
public Map<Id, List<SBQQ__ProductAction__c>> mapRulesToAc-
tions(List<SBQQ__ProductRule__c> rules) {
          Map<Id, List<SBQQ__ProductAction__c>> result = new Map<Id,
List<SBQQ__ProductAction__c>>();

     for (SBQQ__ProductAction__c condition : getRuleProductActions(rules) {
          Id reference = condition.SBQQ__Rule__c;
          if (!result.containsKey(reference))
               result.put(reference, new List<SBQQ__ProductAction__c>());

          List<SBQQ__ProductAction__c> tmp = result.get(reference);
          tmp.add(condition);
          result.put(reference, tmp);
     }

     return result;
}
```

Esimerkkikoodi 8. Hakurakenteen muodostaminen tuotetoiminnoille.

Haettujen tuotesääntöjen laskemiseksi kutsutaan esimerkkikoodissa 9 esiteltyä metodia, jolle välitetään tieto laskentatapahtumasta (engl. evaluation event). Tämän perusteella ProductRuleCalculator käy läpi kaikki ne ladatut tuotesäännöt, joiden tapahtuma vastaa välitettyä tapahtumaa.

```
public void processProductRules(String evaluationEvent) {
     for (SBQQ__ProductRule__c rule : this.rules) {
          if (rule.SBQQ__EvaluationEvent__c == 'Always' || rule.SBQQ__Evalua-
tionEvent__c == evaluationEvent) {
               if (evaluate(rule)) {
                    processSuccessfulRule(rule);
```

```
                    }
                }
            }
        }
```

Esimerkkikoodi 9. Metodi laskentatapahtumaa vastaavien tuotesääntöjen laskemiseksi.

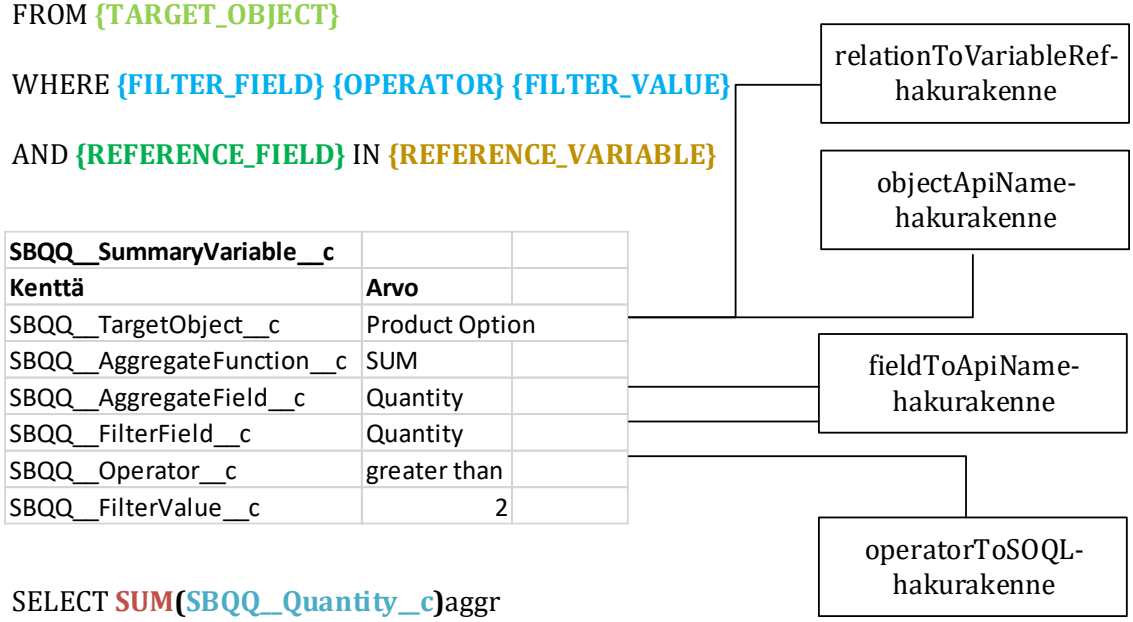
Moni tuotesääntöihin liittyvien tietueiden kentistä, kuten yhteenvetomuuttujien ”SBQQ__TargetObject__c”, sisältää tekstimuotoisen sobjekti- tai kenttänimen, joka ei välttämättä kuitenkaan ole kentän tai sobjektin rajapintanimi. Tällaisia tilanteita varten ProductRuleCalculator luo hakurakenteet ”fieldToApiName” sekä ”objectApiName” niiden rajapintanimien selvittämiseksi.

Tuotesäännöt voidaan jakaa ehto- ja hakukyselylausekkeita sisältäviin tuotesääntöihin. Ehtolausekkeita sisältävät tuotesäännöt, eli ehtolausekesäännöt, voidaan jakaa vielä yhteenvetomuuttujia sisältäviin ja sisältämättömiin sääntöihin. Seuraavissa luvuissa syvennytään tarkemmin erilaisten tuotesääntöjen ratkaisemiseen.

5.4.1 Yhteenvetomuuttujien muodostaminen

ProductRuleCalculator-luokan ilmentyessä, yhteenvetomuuttujista muodostetaan valmiit SOQL-lausekkeet, joita käytetään ehtolausekkeiden ratkaisemiseen. Jokainen yhteenvetomuuttuja-tietue sisältää tarvittavat tiedot yhden sellaisen SOQL-lausekkeen muodostamiseen, jonka lopputuloksena on jokin lukuarvo.

SOQL-lausekkeet muodostetaan välittämällä tietue constructSummaryVariableQuery-metodille. Metodi korvaa valmiin merkkijonon merkittyihin kohtiin tarvittavat tiedot tietueen kentistä. Kuvassa 7 on kuvattuna, miten yhteenvetomuuttujan tiedot täydentyvät valmiiseen merkkijonoon ja muodostavat lopullisen SOQL-lausekkeen.



Kuva 7. SOQL-lausekkeen muodostaminen yhteenvetomuuttujan perusteella.

Koska yhteenvetomuuttujille on valittavissa rajallinen määrä kohdeobjekteja, kyselylausekkeen ehto-osan vertailumuuttuja voidaan määritellä ennakkoon ja tallentaa hakurakenteeseen. Esimerkkikoodissa 10 on esitelty hakurakenne, joka palauttaa vertailumuuttujan, haettavan sobjektin nimen perusteella.

```
Map<String, String> relationToVariableRef = new Map<String, String> {
      'Quote' => ':quoteId',
      'Quote Line' => ':quoteId',
      'Quote Line Group' => ':quoteId',
      'Product Option' => ':selectedProductOptionIds',
      'Configuration Attributes' => ':productId'
};
```

Esimerkkikoodi 10. Hakurakenne sobjektien vertailumuuttujille.

Lisäksi SBQQ__TargetValue__c-kentän tekstimuotoinen vertailuarvo muutetaan SOQL-lausekkeen ymmärtämään muotoon, riippuen SBQQ__Operation__c-kentän sisältämästä operaatiosta sekä siitä, onko vertailuarvo numeerinen.

Kun jokaisesta yhteenvetomuuttujasta on muodostettu SOQL-lauseke, ProductRuleCalculator pystyy ratkaisemaan niitä hyödyntävät ehtolausekkeet.

5.4.2 Ehtolausekesääntöjen ratkaiseminen

Ehtolausekkeita sisältävää sääntöä ratkaistessa ProductRuleCalculator iteroi läpi jokaisen sääntöön liittyvän ehtolausekkeen. Sellaisten ehtolausekkeiden tiedoista, jotka eivät kohdistu yhteenvetomuuttujiin, muodostetaan fieldsPerObject-niminen hakurakenne, joka sisältää sobjektikohtaisesti tiedot kentistä, joihin ehtolausekkeissa viitataan. Samalla kaikista ehtolausekkeista muodostetaan lopullista laskentaa mallintava, indexEvaluationResult-niminen hakurakenne, jonka avaimena on ehtolausekketietueen SBQQ__Index__c-kentän arvo ja arvona false.

Tämän jälkeen jokaisesta fieldsPerObject-hakurakenteen mallintamasta objektista luodaan SOQL-lauseke, jonka jälkeen ajetaan kaikki aikaisemmin rakennetut, sääntöön liittyvien yhteenvetomuuttujien SOQL-kyselyt. Jokaisen yhteenvetomuuttujan laskennan jälkeen, tuloksena saatuun numeroarvoon kohdistetaan vielä kyseisen yhteenvetomuuttujan tietueessa määritelty mahdollinen laskutoimitus, jonka jälkeen tulos tallennetaan summaryVariableEvaluatedValue-nimiseen hakurakenteeseen.

Seuraavaksi ProductRuleCalculator ajaa jokaisen fieldsPerObject-hakurakenteen perusteella luodun SOQL-lausekkeen. Jokaisen lausekkeen kohdalla käydään läpi ne ehtolausekkeet, joiden SBQQ__TestedObject__c-kenttä sisältää arvonaan lausekkeen objektin nimen. Tämän iteroinnin aikana ratkaistaan ehtolausekkeen sisältämä vertailu. Vertailussa ehtolausekkeen SBQQ__TestedField__c-kentässä määritellystä iteroidun tietueen kentän arvoa verrataan joko ehtolausekkeen SBQQ__FilterValue__c-kentän arvoon tai ehtolausekkeen viittaamaan yhteenvetolausekkeen arvoon. Vertailu suoritetaan evaluateOperatorCondition-metodilla, jolle välitetään kohdearvo, vertailuarvo ja ehtolau-

sekkeen SBQQ__Operator__c-kentän sisältämä operattori. Jokaisen vertailun lopputulos, tosi tai epätosi, tallennetaan indexEvaluationResult-hakurakenteeseen, ehtolausekkeen indeksiarvoa vastaavan avaimen arvoksi.

Lopuksi käsitellään sellaiset ehtolausekkeet, jotka eivät viittaa objekteihin, vaan vertaavat yhteenvetomuuttujien arvoja keskenään tai kentässä määriteltyyn arvoon. Näiden ehtolausekkeiden vertailu suoritetaan muuten samalla tavalla, paitsi SOQL-lausekkeiden ajamisen sijaan kohdearvoksi haetaan summaryVariableEvaluatedValue-hakurakenteesta ehtolausekkeessa viitatun yhteenvetomuuttujan arvo.

Kun kaikki tuotesäännön ehtolausekkeet on ratkaistu ja tallennettu hakurakenteeseen, ProductRuleCalculator lukee tuotesääntötietueen SBQQ__ConditionsMet__c-kentän ehdon. Mikäli ehtona on ”All” tai ”Any”, hakurakenne välitetään metodille, joka käy rakenteen läpi ja palauttaa tosi / epätosi riippuen siitä, vastaavatko hakurakenteen arvot määriteltyä ehtoa. Mikäli taas ehtona on ”Custom”, siirrytään tarkastelemaan tuotesäännölle määriteltyä mukautettua ehtolauseketta (SBQQ__AdvancedCondition__c).

Kuten luvussa 4.1 mainittiin, mukautettu ehtolauseke on tekstimuotoinen lauseke, esimerkiksi 1 AND (2 OR 3), jossa numerot viittaavat ehtolausekkeiden SBQQ__Index__c-kentän arvoihin. Tällaisen lausekkeen ratkaisemiseksi kehitettiin Edsger Dijkstran shunting-yard -algoritmia (Shunting Yard Algorithm. 2019.) hyödyntävä samanniminen Apex-luokka.

ShuntingYard-luokka toteuttaa infix-notaation muuttamisen postfix-notaatioksi, hyödyntäen algoritmia sillä erolla, että kaikki operaattorit (AND ja OR) ovat saman arvoisia. Luokka sisältää kaksi listamuuttujaa, joita käytetään jonorakenteina sekä toteutuksen pinorakenteesta, jota käytetään algoritmin operaattoripinona.

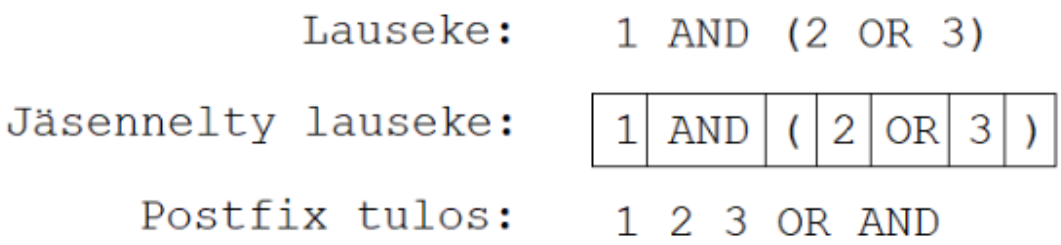
Ennen lausekkeen läpikäyntiä, luokka jäsentää lausekkeen yksittäisiksi operandeiksi (luvut), operaattoreiksi (AND ja OR) ja sulkumerkeiksi.

Jäsennetyn lausekkeen muunnoslogiikka on seuraava:

- Jos alkio on luku, vie luku tulosjonoon.

- Jos alkio on operaattori, vie alkio pinoon.
- jos alkio on vasen sulkumerkki, vie alkio pinoon.
- Jos alkio on oikea sulkumerkki, vie pinon alkiot tulosjonoon, vasempaan sulkumerkkiin asti.
- Lopuksi vie pinoon jääneet alkiot tulosjonoon.

Tämän jälkeen tulosjonossa on postfix-muotoinen lauseke. Kuvassa 8 on esimerkki luokan saamasta infix-muotoisesta lausekkeesta, jäsennellystä lausekkeesta sekä lopullisesta postfix-muotoisesta lausekkeesta.



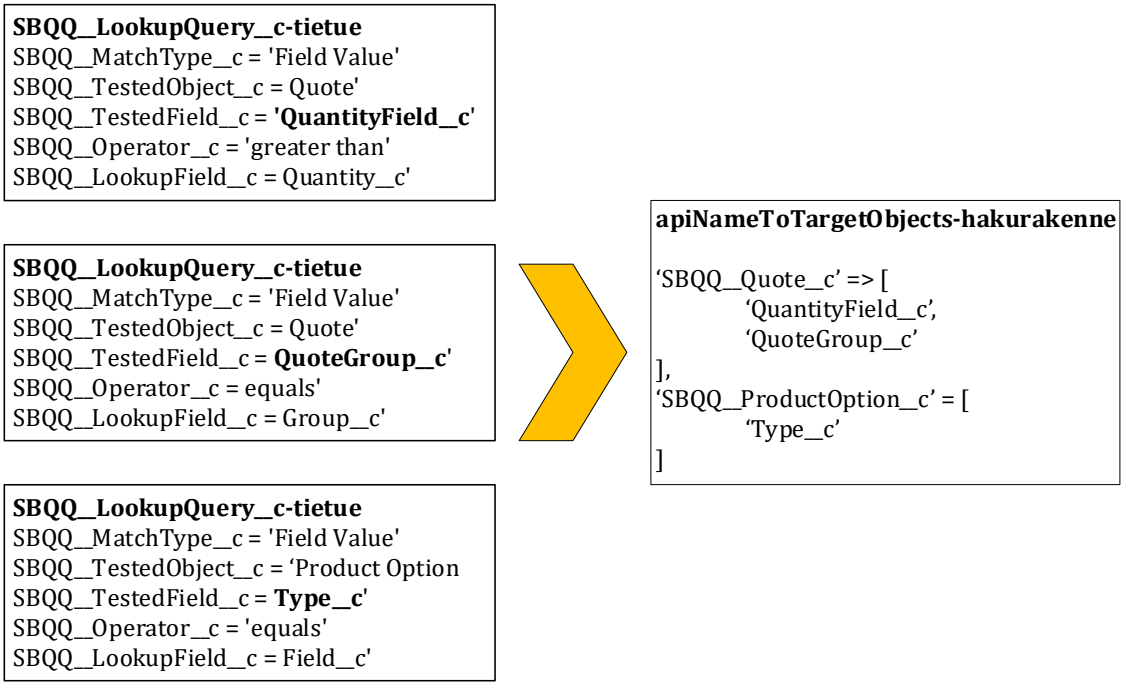
Kuva 8. Postfix-lausekkeen muuttuminen infix-muotoon.

Sitten luokka suorittaa pinon avulla postfix-muotoisen lausekkeen arvon laskennan. Laskennan aikana luokka hakee indexEvaluationResult-hakurakenteesta lausekkeessa esiintyviä numeroita vastaavat boolean arvot, jotka puolestaan vastaavat aikaisemmin ratkaistujen ehtolausekkeiden tuloksia. Täten esimerkiksi kuvan 8 mukainen postfix-lauseke ”1 2 3 OR AND” ratkaistaan muodossa ”true true false OR AND”. Viimeinen pinoon jäävä boolean arvo on tuotesäännön ehtolausekkeen loppuarvo.

5.4.3 Hakukyselylausekesääntöjen ratkaiseminen

Yhteenvetomuuttujien tapaan hakukyselylauseketietueet sisältävät tietoa SOQL-lausekkeiden muodostamiseksi. Toisin kuin ehtolausekesäännöissä, hakukyselylausekesäännöissä ehtolauseke muodostetaan hakukyselylauseketietueista ja sääntö kohdistuu tuotesäännön SBQQ__LookupObject__c-kentässä määriteltyyn objektiin. Aluksi sääntöön

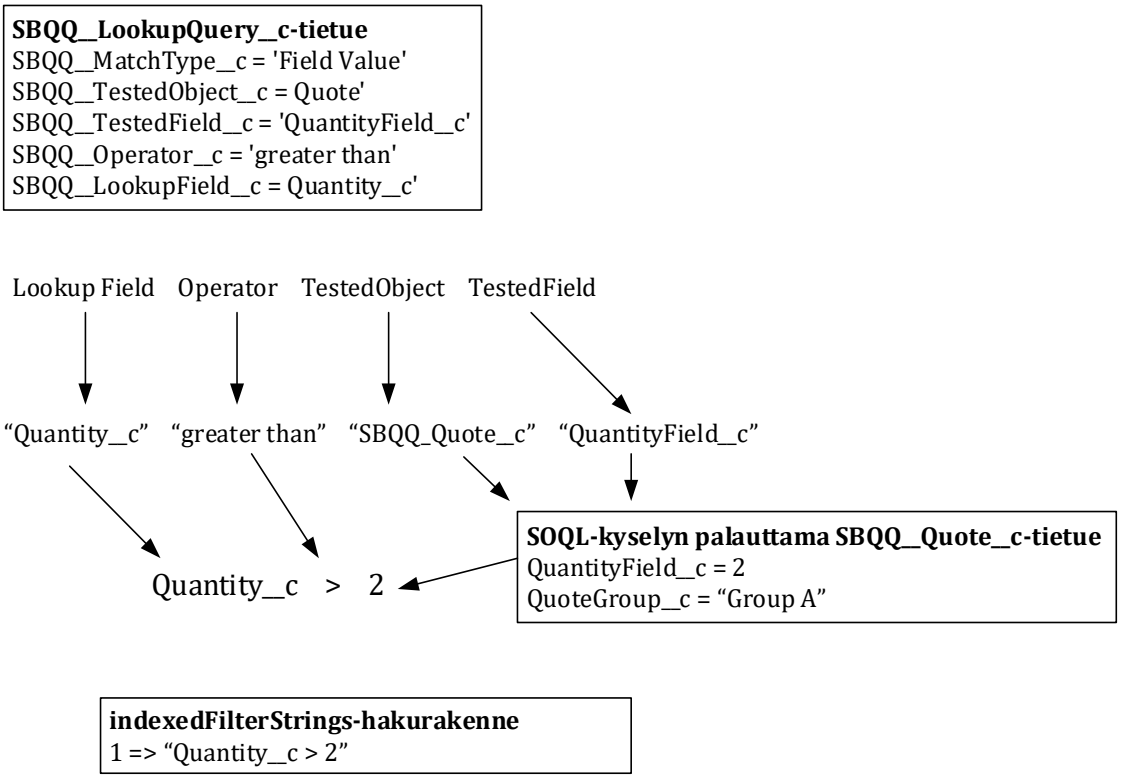
liittyvät hakukyselylausekkeet iteroidaan, mikä muodostaa lookupTestedObjectFields-niminen hakurakenteen, joka sisältää hakukyselytietueiden SBQQ__TestedObject__c-kentän sisältämien objektien, SBQQ__TestedField__c-kentässä mainitut kentät. Kuvassa 9 on havainnollistettu hakurakenteen muodostuminen tietueiden datan pohjalta.



Kuva 9. Hakurakenteen muodostuminen hakukyselylausekkeeseen liittyvän datan pohjalta.

Tämän jälkeen jokaisesta hakurakenteen objektista muodostetaan SOQL-lauseke. Lausekkeet ajetaan ja Salesforcen palauttamat tietueet viedään apiNameToTargetObjects-nimiseen hakurakenteeseen. Palautuneet tietueet sisältävät hakukyselylausekkeiden vertailudataa, jonka perusteella muodostetaan SBQQ__LookupObject__c-kentässä määriteltyyn objektiin kohdistuvan SOQL-lausekkeen where-lause. Where-lausetta muo-

dostaessa hakukyselylauseketietueet iteroidaan uudelleen. Jokaisesta hakukyselylausekkeesta muodostetaan yksi where-lauseen lauseenjäsen, jotka viedään indexedFilterStrings-nimiseen hakurakenteeseen, jonka avaimena on hakukyselylauseketietueen järjestystuku (Name-kenttä) ja arvona muodostettu lauseenjäsen. Kuvassa 10 havainnollistetaan lauseenjäsenen muodostamista hakukyselytietueen pohjalta.



Kuva 10. Lauseenjäsenen muodostuminen hakukyselutietueen datan perusteella.

Kun kaikista hakukyselylauseketietueista on muodostettu lauseenjäsenet, ProductRuleCalculator-luokka rakentaa SOQL-lausekkeen where-lauseen. Where-lause rakennetaan tuotesääntötietueen SBQQ__ConditionsMet__c-kentän ehdon perusteella. ”Any”-ehdossa kaikkien lauseenjäsenten väliin lisätään ”OR” ja ”All”-ehdossa ”AND”. ”Custom”-ehdossa where-lause korvataan SBQQ__AdvancedCondition__c-kentän mukaisella ehtolauseella, jonka jälkeen ehtolauseen numeroarvot korvataan indexedFilterStrings-hakurakenteen osoittamilla lauseenjäsenillä. Kuvassa 11 havainnollistetaan where-lauseen muodostuminen eri ehtojen perusteella.

```
indexedFilterStrings
1 => "Quantity__c > 2"
2 => "Group__c = 'Group A' "
3 => "Field__c = 'Value' "
```

```
"Any"
Quantity__c > 2 OR Group__c = 'Group A' OR Field__c = 'Value'
```

```
"All"
Quantity__c > 2 AND Group__c = 'Group A' AND Field__c = 'Value'
```

```
"Custom" - 1 AND (2 OR 3)
Quantity > 2 AND (Group__c = 'Group_A' OR Field__c = 'Value')
```

Kuva 11. Where-lauseen muodostuminen hakurakenteen datan pohjalta, erilaisilla muodostamisehdoilla.

Lopuksi ProductRuleCalculator luo lopullisen SOQL-lausekkeen ja asettaa rakentamansa where-lauseen sen where-lauseeksi. Tämän SOQL-lausekkeen where-lause toimii tuotesäännön ehtolausekkeena. Mikäli Salesforce palauttaa lausekkeen mukaisia tietueita, ovat tuotesäännön ehdot toteutuneet. Jokainen Salesforcen palauttama tietue sisältää tiedot yhden tuotetoiminnon toteuttamiseksi. Näiden tietueiden kentissä, joiden nimet määritellään tuotesääntötietueen kentissä SBQQ__LookupProductField__c ja SBQQ__LookupTypeField__c, on tallennettuna lookup-viittaus tuotteeseen (Product2) ja tuotetoiminnon tyyppi (esimerkiksi ”Add”). Jokaisesta tietueesta luodaan väliaikainen tuotetoimintotietue, jotka lisätään ProductRuleCalculatorin ruleActions-hakurakenteseen odottamaan käsittelyä.

5.4.4 Toteutuneen tuotesäännön prosessointi

Kun tuotesääntö on ratkaisu onnistuneesti, ProductRuleCalcualtor käsittelee siihen liittyvät tuotetoiminnot. Koska tuotetoimintotietueet viittaavat tuotetietueisiin, eivätkä tuote-

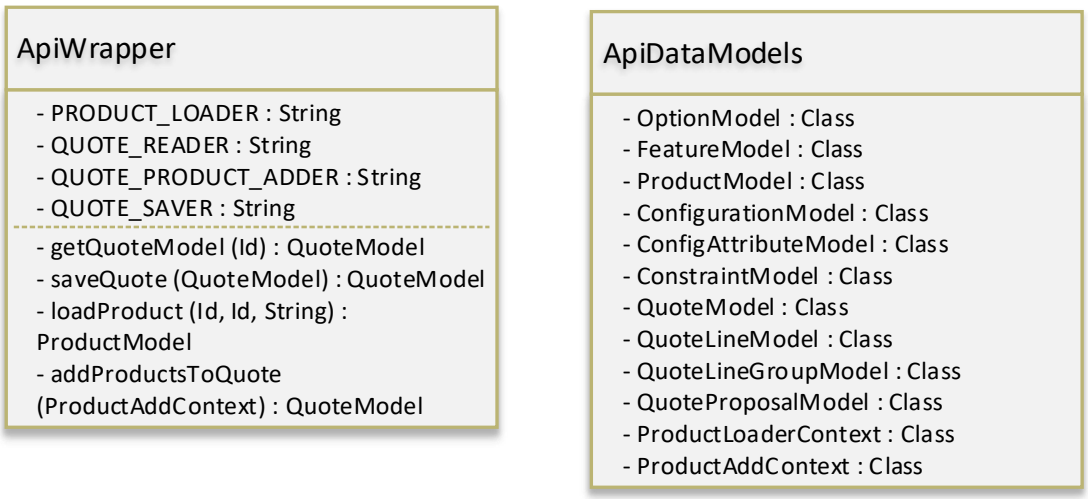
vaihtoehtotietueisiin, ennen käsittelyä ProductRuleCalculator rakentaa productIdToOptionId-nimisen hakurakenteen, jossa avaimena on tuotetietuetunniste ja arvona tuotetta vastaava, konfiguroitavan tuotteen tuotevaihtoehtotietueen tunniste. Tämän jälkeen tuotetoimintotietueet iteroidaan ja tallennetaan ProductRuleCalculatorin sisältämään ConstraintResult-tyypin result-nimiseen muuttujaan, ja käyttää esimerkkikoodissa 6 esiteltyä handleAction-metodia.

Mikäli tuotesääntöön ei liity tuotetoimintoja, vaan kyseessä on validointi- tai huomautussääntö, tuotesääntötietueen SBQQ__ErrorMessage__c-kentän arvo tallennetaan sääntötyyppiä vastaavaan validation- tai alertMessage-muuttujaan.

Tämän jälkeen muuttujat voidaan viedä ConfiguraitonModel-olion mallintamaan konfiguraatioon, jolloin result-muuttuja käsitellään luvussa 5.3 mainitun ConstraintResult käsittelyn mukaisesti ja validation- ja alertMessaget tallennetaan ConfiguraitonModelin ”messages”-listaan.

5.5 QuoteAPI-ohjelmointirajapinnan hyödyntäminen

Konfiguroidun tuotteen tallentamiseksi tarjoukselle hyödynnettiin luvussa 4.1 esiteltyä QuoteAPI-ohjelmointirajapintaa. Rajapintakutsuja varten luotiin ApiWrapper-niminen Apex-luokka, jolla rajapintakutsut kapseloidaan helpommin käytettäviksi metodikutsuiksi. Kuvassa 12 on esiteltynä ApiWrapper-luokan julkiset metodikutsut, joita kutsutaan QuoteUtils-nimisestä Apex-luokasta, joka huolehtii tuotekonfiguraation tallentamisesta tarjoukselle.

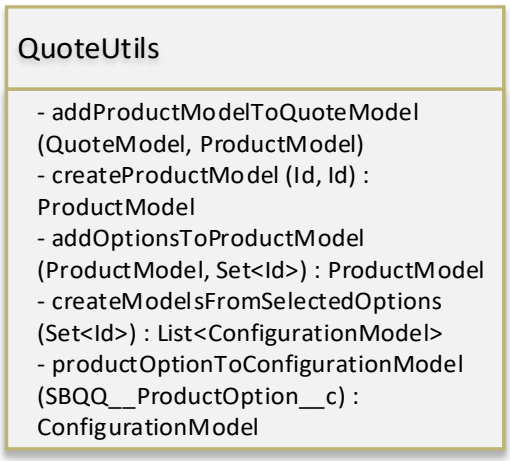


Kuva 12. ApiWrapper ja ApiDataModels luokkien uml UML.

QuoteAPI-ohjelmointirajapinnan vaatima datamalli (CPQ API Models, 2019) rakennettiin ApiDataModels-nimiseen wrapper-luokkaan. QuoteAPI sisältää työssä kehitetyn Apex-luokan kanssa samannimisen luokan, ConfigurationModel. Tässä luvussa ConfigurationModelilla tarkoitetaan oletuksena tätä luokkaa.

5.5.1 Konfiguriodun tuotteen rakentaminen

Kuten luvussa 5.5 mainittiin, tuotekonfiguraatioiden tallentamisesta huolehtii QuoteUtils-niminen Apex-luokka. Kuvassa 13 on esiteltynä QuoteUtils-luokan metodit.



Kuva 13. QuoteUtils-luokan UML.

Jotta konfiguroidun tuotteen tallentaminen QuoteAPI:n kautta on mahdollista, tulee konfiguraatio muuttaa ohjelmointirajapinnan datamallin muotoon. Aluksi muodostetaan ProductModel-luokan olio. Olio muodostetaan JSON-muotoisesta merkkijonosta, joka saadaan QuoteAPI:n ServiceRouter-luokalta, esimerkkikoodin 11 mukaisella loadProduct-metodilla. Luokalle välitetään konfiguroitavan tuotteen tietuetunniste sekä JSON-objektina tilauksella käytössä olevan hintakirjan tietuetunniste ja valuuttakoodi.

```
public static final String PRODUCT_LOADER = 'SBQQ.ProductAPI.ProductLoader';

private static String load(String name, String recordId, Object payload) {
    return SBQQ.ServiceRouter.load(name, recordId, JSON.serialize(payload));
}
public static ApiDataModels.ProductModel loadProduct(Id productId, Id price-
bookId, String currencyCode) {
    ApiDataModels.ProductLoaderContext productLoaderPayload = new ApiDataMod-
els.ProductLoaderContext(pricebookId, currencyCode);
    String jsonResultProduct = load(PRODUCT_LOADER, (String) productId,
productLoaderPayload);
    ApiDataModels.ProductModel productModel = (ApiDataModels.ProductModel)
JSON.deserialize(jsonResultProduct, ApiDataModels.ProductModel.class);
    return productModel;
}
```

Esimerkkikoodi 11. Tuotteen lataaminen ProductModel-luokan olioksi, QuoteAPI:n kautta

Datamallin dokumentaatiossa (CPQ API Models, 2019) ei kerrota selkeästi, miten konfiguroitavaa tuotetta mallintavalle ProductModel-luokan oliolle lisätään tuotevaihtoehtoja. Tämä selvitettiin työn aikana lisäämällä QuoteAPI:n kautta erilaisia kombinaatioita tarjoukselle, kunnes haluttu rakenne tallentui onnistuneesti. ProductModel-luokka mallintaa omaa konfiguraatiotaan yhdellä ConfigurationModel-luokalla. Kyseisen luokan configured-nimisen muuttujan arvoksi asetetaan true. Konfiguroidun tuotteen valitut tuotevaihtoehdot lisätään, kukin omana ConfigurationModel-oliona, luokan sisältämään optionConfigurations-listaan, joka on dokumentaatiossa virheellisesti ilmoitettu yksittäisenä muuttujana. Jokainen olio rakennetaan esimerkkikoodin 12 mukaan SBQQ__ProductOption__c-tietueen perusteella.

```
public static ApiDataModels.ConfigurationModel optionToConfigurationModel(
SBQQ__ProductOption__c option) {

    // Asetetaan tietueen valituksi, ennen lisäämistä oliolle
    option.SBQQ__Selected__c = true;
```

```
     ApiDataModels.ConfigurationModel optionConfigModel = new ApiDataMod-
els.ConfigurationModel();
     optionConfigModel.configuredProductId = option.SBQQ__ConfiguredSKU__c;
     optionConfigModel.optionId = option.Id;
     optionConfigModel.optionData = option;

     return optionConfigModel;
}
```

Esimerkkikoodi 12. ApiDataModels.ConfigurationModel-olion muodostaminen tuotevaihtoehdon perusteella.

Kun kaikki valitut tuotevaihtoehdot on lisätty optionConfigurations-listaan, ProductModel lisätään tarjoukselle.

5.5.2 Konfiguroidun tuotteen lisääminen tarjoukselle

Tuotteen lisäämiseksi tarjoukselle myös tarjoustietueesta muodostetaan datamallin mukainen QuoteModel-olio. ServiceRouterin load-metodille välitetään tarjouksen tietuetunniste sekä QuoteReader-rajapinnan nimi, jolloin luokka palauttaa QuoteModelin JSON-muotoisena merkkijonona. Tästä QuoteModelista, sekä aikaisemmin muodostetusta ProductModel-oliosta, luodaan ProductAddContext-olio. Tämän jälkeen ServiceRouterin load-metodille välitetään tuotteen lisäämiseen käytettävän QuoteProductAdder-rajapinnan nimi sekä JSON-muotoinen ProductAddContext-olio. Metodi palauttaa JSON-muotoisen QuoteModel-olion, joka sisältää ProductModelin perusteella muodostetut QuoteLineModelit.

Lopuksi tarjous tallennetaan välittämällä ServiceRouterin save-metodille QuoteSaver-rajapinnan nimi sekä JSON-muotoinen QuoteModel. Tallentamisen yhteydessä, Salesforce luo taustalla tietueet uusille tarjousriveille, luo näiden väliset yhteydet lookup-kentillä ja täydentää tarvittavaa tietoa mm. tarjoukselta tai muilta tarjoukseen liittyviltä tietueilta.

6 Yhteenveto

Insinöörityön tarkoituksena oli tutkia Salesforce CPQ:n tuotekonfiguraattorin datamallia ja toimintalogiikkaa sekä kehittää tarvittavat ohjelmakomponentit sen hyödyntämiseen. Työssä saatiin aikaiseksi ohjelma, joka onnistuneesti replikoi tuotekonfiguraattorin toiminnallisuuksia ja logiikkaa yksinkertaisissa käyttötapauksissa, samalla hyödyntäen sen datamallia. Työn edetessä opin myös Salesforce CPQ:n ominaisuuksista, hyödyntämismahdollisuuksista ja rajoituksista. Lisäksi sain pintakosketuksen matemaattiseen teoriaan ja matemaattisiin malleihin, joihin tuotekonfiguraatio yleisesti ottaen perustuu.

Työn toteutuksen aikana suurimmat ongelmat esiintyivät Salesforce CPQ:n tuotekonfiguraattorin logiikan selvittämisen aikana, sillä sen dokumentaatio oli käyttöohjeita lukuun ottamatta lähes olematonta. Dokumentaation puutetta kompensoitiin mm. luomalla testitapauksia, joiden aikana ajettiin haluttu CPQ-toiminnallisuus ja tarkkailtiin datan muutoksia ennen ja jälkeen.

On kuitenkin korostettava, että Salesforce CPQ sisältää lukuisia toiminnallisuuksia, joita tässä työssä ei käsitelty; myös tuotekonfiguraattorin osalta. Työ luo kuitenkin hyvän pohjan jatkaa puuttuvien toiminnallisuuksien kehittämistä tulevaisuuden tarpeiden mukaan. Salesforce CPQ on kuitenkin massiivinen kokonaisuus, jonka kehittämiseen on mennyt varmasti tuhansia työtunteja, ja joka kehittyy jatkuvasti. Olisikin toivottavaa, että Salesforce lisäisi tarjoamaansa rajapintaan toimintoja, joiden avulla Salesforce CPQ:n omaa logiikka voitaisiin hyödyntää monipuolisemmin.

Lähteet

Anderson, Kristina. 2014. What is CPQ? Let Us Explain. Verkkoaineisto. <https://www.salesforce.com/blog/2014/07/what-is-cpq--let-us-explain-.html>. 15.7.2014. Luettu 1.6.2019.

Components Available in Managed Packages. 2019. Verkkoaineisto. Salesforce.com Ltd. <https://developer.salesforce.com/docs/atlas.en-us.packaging-Guide.meta/packagingGuide/packaging_packageable_components.htm>. Luettu 15.10.2019.

CPQ API Models. 2019. Verkkoaineisto. Salesforce.com, Inc. <https://developer.salesforce.com/docs/atlas.en-us.cpq_dev_api.meta/cpq_dev_api/cpq_api_models.htm>. Luettu 6.10.2019.

Fee, Gregory & Gallagher, William. 2011. Methods and systems for utilizing bytecode in an on-demand service environment including providing multi-tenant runtime environments and systems. Verkkoaineisto. <https://patents.google.com/patent/US20110264861>. 21.4.2010. Luettu 30.5.2019.

Felgerning, Alexander; Hotz, Lothar; Bagley, Claire & Tiihonen, Juha. 2014. Knowledge-Based Configuration: From Research to Business Cases. E-kirja. Morgan Kaufmann.

Hilbert, Melissa; Klock, Christina & Lewis, Mark. 2018. Critical Capabilities for Configure, Price and Quote Application. Verkkoaineisto. <https://www.gartner.com/en/documents/3892769>. Luettu 20.5.2019.

McCormick, Moira. 2016. How can CPQ solutions boost your bottom line? Verkkoaineisto. <https://blog.blackcurve.com/why-use-configure-price-quote-cpq-software>. 29.6.2016. Luettu 1.6.2019.

Sahlsten, Pekka. 2012. Asiakkuudenhallinta eli CRM - mistä oikein on kysymys? Verkkoaineisto. <https://www.myynti20.fi/asiakkuudenhallinta-crm-mista-on-kysymys> 3.9.2012. Luettu 29.5.2019.

Sales Cloud. 2019. Verkkoaineisto. Salesforce.com, Inc. <https://www.salesforce.com/products/sales-cloud/overview/>. Luettu 29.5.2019.

Salesforce CPQ. 2019. Verkkoaineisto. Salesforce.com, Inc. <https://resources.docs.salesforce.com/216/latest/en-us/sfdc/pdf/final_cpq_map.pdf>. Luettu 6.10.2019.

Service Router. 2019. Verkkoaineisto. Salesforce.com, Inc. <https://developer.salesforce.com/docs/atlas.en-us.cpq_dev_api.meta/cpq_dev_api/cpq_api_service_router.htm>. Luettu 3.10.2019.

Shunting Yard Algorithm. 2019. Verkkoaineisto. Brilliant.com. <https://brilliant.org/wiki/shunting-yard-algorithm/>. Luettu 3.10.2019.

sObject Types. 2019. Verkkoaineisto. Salesforce.com, Inc. <https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/langCon_apex_SObject_types.htm>. Luettu 30.5.2019.

Understanding Custom & Standard Objects. 2019. Verkkoaineisto. Salesforce.com, Inc. <https://trailhead.salesforce.com/en/content/learn/modules/data_modelling/objects_intro>. Luettu 11.6.2019.

Understanding Apex Core Concepts. 2019. Verkkoaineisto. Salesforce.com, Inc. <https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_qs_core_concepts.htm>. Luettu 30.5.2019.

What is a Product Configurator? 2019. Verkkoaineisto. Sepia GmbH & Co. KG. <https://www.sepia.de/produktkonfigurator-definition.html?&L=1> Luettu 7.10.2019.

What is Apex? 2019. Verkkoaineisto. Salesforce.com, Inc. <https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_intro_what_is_apex.htm> Luettu 30.5.2019.

What is PaaS? - Platform as a Service Explained. 2019. Verkkoaineisto. Salesforce.com, Inc. <https://www.salesforce.com/au/learning-centre/tech/paas >. Luettu 29.5.2019.

What is Salesforce? 2019. Verkkoaineisto. Salesforce.com, Inc. <https://www.salesforce.com/eu/products/what-is-salesforce/>. Luettu 29.5.2019.