Salesforce Integraatio ulkoiseen järjestelmään

Santeri Heiskanen

Haaga-Helia ammattikorkeakoulu
Tradenomi, tietojenkäsittely
Opinnäytetyö
2022

Tiivistelmä

Tekijä(t)
Santeri Heiskanen

Tutkinto
Tradenomi

Raportin/Opinnäytetyön nimi
Salesforce Integraatio ulkoiseen järjestelmään

Sivu- ja liitesivumäärä
62

Toiminnallisen opinnäytetyön tarkoituksena oli suorittaa toimeksiantajayritykselle 10Duke Software Ltd:lle alustava analyysi ja projektisuunnitelma Salesforce integraatioprojektille, jota yritys oli aloittamassa. Tarkoituksena selvittää mitä eri integraatiomahdollisuuksia Salesforcessa oli, kuinka erosivat toisistaan ja mitä vahvuuksia ja heikkouksia niillä oli.

Analyysi, joka tehtiin, pohjautui integraatiototeutusten alustaviin käytännön toteutuksiin, josta tehtiin havaintoja. Havaintojen pohjalta opinnäytetyössä tehtiin business impact analyysi integraatiototeutusten eri riskeistä ja kustannusvaikutusarvio integraatiototeutusten vaikutuksista yrityksen eri sidosryhmiin. Näiden pohjalta toimeksiantajalle tehtiin alustava projektisuunnitelma, joka perustui valittuun integraatioteknologiaan. Opinnäytetyön käsittelemiä teknologioita olivat Salesforcen Apex-ohjelmointikieli, pub/sub API ja Amazon Web Servicen Amazon AppFlow. Toimeksiantajan tapauksessa integraatioprojekti päätettiin toteuttaa käyttämällä AppFlowia.

Analyysi teknologioiden välillä tehtiin tekemällä vertauksia ja havaintoja kunkin teknologian käytännön osuudesta, jonka pohjalta muodostettiin business impact analyysi ja kustannusarvio. Amazon AppFlowissa oli harkituista teknologioista kaikista vähiten teknisiä rajoituksia siihen nähden mitä yritys halusi saavuttaa integraatiokomponentillaan. AppFlow oli muita integraatioratkaisuja kalliimpi, mutta toimeksiantajan tahdosta projekti eteni AppFlowia käyttäen.

Työ koostui integraatiototeutusten tutkinnasta, vertailusta, alustavien demonstraatioiden, analyysin ja projektisuunnitelman tekemisestä. Tehty työ on toteutettu 1.3.2022-5.12.2022 välisenä aikana.

Asiasanat
Salesforce, integraatio, projektisuunnitelma, AWS, Apex

Sisällys

Keskeiset käsitteet:

API: Ohjelmistorajapinta, jossa kaksi tietojärjestelmää välittää tietoja toistensa välillä, ja pystyvät lähettämään pyyntöjä toistensa kesken tiedonvälitystä varten.

Cloud: Datakeskuksista muodostuva kommunikaatioverkko täynnä palvelimia, jotka on yhdistetty Internetiin. Cloud computing viittaa ohjelmistoihin ja palveluihin, jota jaetaan cloudin välityksellä. (PCMAG s.a.a.)

CRM: Customer Relationship Management, asiakkuudenhallinta, joka sisältää ne ominaisuudet, mitä organisaatio hyödyntää, kun se on vuorovaikutuksessa asiakkaan kanssa (Hargrave, 5.9.2022).

CRMS: Customer Relationship Management Software, tietojärjestelmä, joka kerää asiakkaan dataa organisaation saatavuuteen. Kerättäviä tietoja, asiakkaan yhteystiedot, ostohistoria- ja vuorovaikutukset asiakkaan kanssa (Hargrave, 5.9.2022).

PAAS: Platform-As-A-Service, alustan välitys, jossa myyjä tarjoaa alustan ohjelmistopalveluna käyttäjille. Alustan ja siihen liittyvän palvelimen ylläpidosta vastaa myyjä itse.

pub/sub: Publish-subscribe asynkroninen viestintä menetelmä, jossa kaksi ohjelmistoa kommunikoi keskenään niin, että julkaisija (tiedon alkulähde) on riippumaton tilaajasta (subscriber) (Stackpath, s.a).

REST: Representational State Transfer, hybridi ohjelmisto arkkitehtuurityyli hypermedia järjestelmille, jossa on määritellyt rajoitteet siihen, kuinka järjestelmän sisäiset elementit käyttäytyvät. Hypermedialla viitataan median muotoihin kuten Internet (Fielding, 2000, osa 5–5.1).

RSA-SHA256: Tiedonsalausmetodi, jossa SHA256 algoritmi ensiksi arpoo sekoitusarvon, sille annetusta arvosta, jonka pohjalta RSA-algoritmin avulla tieto salataan. Tarkoituksena varmistaa, että tietoa ei olla yritetty muuttaa matkan aikana (Jena, 11.11.2022).

SAAS: Software-as-a-Service, ohjelmiston välitys malli, jossa ohjelmistot ylläpitää myyjä omilla palvelimillaan- ja ne ovat asiakkaalle saatavilla tietoverkon (esim. Internet) kautta.

SSL: Secure Sockets Layer-protokolla, joka muodostaa salatun yhteyden web-palvelimen ja selaimen välillä (Kapersky s.a.).

1 Johdanto

Tietojärjestelmät ovat muuttaneet tavan, jolla yritykset järjestävät toimintojaan jokapäiväisessä elämässämme. Suurimpana vaikuttajana on ollut internet, joka globalisoi tavan, jolla yritykset ovat vuorovaikutuksessa toistensa, sekä asiakkaan kanssa 1990-luvun alusta lähtien. Tämän takia myynnin toiminnot, (yhdessä yrityksen muiden toimintojen kanssa) ovat joutuneet kehittämään toimintaansa, keräämällä tietoa vuorovaikutuksesta asiakkaan kanssa, jotta he pystyvät paremmin kohdistamaan myyntiä oikeille asiakkaille (Bose, 24.1.2020). Tämän myötä on myös syntynyt tarve, että tietojärjestelmien välillä olisi mahdollisimman saumaton integraatio, CRM-järjestelmiin.

Tämä opinnäytetyö on toiminnallinen opinnäytetyö. Sen tarkoituksena on selvittää mitä eri mahdollisuuksia Salesforce asiakkuudenhallintajärjestelmässä on integraatiolle kolmannen osapuolen ohjelmistojen kanssa. Samalla on tarkoitus selvittää, mikä olisi paras mahdollinen toteutustapa integraatioille tilanteissa, joissa organisaatioilla on eri tarpeet teknisten vaatimusten kannalta.

Opinnäytetyön toimeksiantaja yrityksenä toimii 10Duke Software Ltd., joka on kansainvälinen PK-yritys. Sen toimipisteet sijoittuvat Iso-Britanniaan ja Suomeen. Yritys erikoistuu teknologian ja palveluiden toimittamiseen, jotta ohjelmistoyritykset pystyvät toteuttamaan lisensoinnin ja asiakasidentiteettien hallinnan.

Toimeksiantajan tapauksessa, opinnäytetyö pyrkii vastaamaan minkälainen Salesforce integraatio olisi heille paras mahdollinen. Yritys on suunnitellut Salesforcen integrointia heidän toimittamiinsa ratkaisuihin. Opinnäytetyöllä pyritään tarjoamaan tietoperusta ja analyysi siitä minkälainen integraatioratkaisu olisi heille kaikista järkevin, jotta projekti itsessään pystytään aloittamaan, kuten myös alustava projektisuunnitelma integraatioprojektille valitun teknologian perusteella.

Akateemisesta näkökulmasta tutkimuskysymyksinä opinnäyte pyrkii vastaamaan siihen, mitä eri integraation toteutustapoja Salesforcella on tarjolla, miten ne eroavat toisistaan ja mitä vahvuuksia ja heikkouksia niillä on. Pyrkimyksenä on, että lukija pystyisi muodostamaan selkeän tietoperustan ja kuvan siitä, mitä asioita kehittäjien tulisi ottaa huomioon.

1.1 Rajaus

Opinnäytetyö tulee tutkimaan eri Salesforce CRM-järjestelmän integraatio mahdollisuuksien tietoperustaa. Määrittelemällä ne tekniset ratkaisut, johon kukin integraatiototeutus pohjautuu ja minkälaisiin integraatio tarpeisiin eri toteutukset soveltuvat yksittäisessä projektissa. Näihin teknisiin ratkaisuihin sisältyvät integraatioratkaisuja pohjustavat teknologiat, niiden arkkitehtuurimallit ja mistä komponenteista ne muodostuvat. Toimeksiantajan tarkoituksena on saada parempi katsaus eri

integraatio toteutusten teknisiin eroihin, ja siihen kuinka ne voivat palvella yrityksen tarpeita. Riippuen yksittäisen organisaation resursseista ja vaatimuksista integraatiolle.

Opinnäytetyö ei pyri näyttämään integraatioiden täyttä toteutusta. Vaan antamaan selkeän perustason tietoperustan siitä, kuinka eri integraatio toteutukset käytännössä voitaisiin toteuttaa. Tavoitteena on tarjota alustavat demonstraatiot siitä, miten eri integrointimahdollisuudet mahdollisesti toteutetaan ja kuinka ne eroavat toisistaan. Antaen perustelut sille miksi annetussa toimeksiannossa päädyttiin valitsemaan lopullinen integraatiototeutus.

Seuraavaksi teoriaosuudessa tulemme käsittelemään eri integraatiototeutusten teknistä puolta, ensiksi Salesforcen oman Apex-ohjelmointikielen kautta, toiseksi pub/sub API:n kautta ja viimeiseksi Amazonin AppFlowin kautta käsitteineen.

2 Teoria

Ymmärtääksemme käsitteet mitä opinnäytetyön rajauksessa tulemme käsittelemään, pitää meidän luoda asiallinen teoriapohja sille. Opinnäytetyön teoriaosuudessa tulemme ensimmäiseksi käsittelemään integraation konseptia ja mitä se tarkoittaa, jonka jälkeen tulemme siirtymään projektihallinnan konseptiin ja mitä käsitteitä siihen sisältyy. Viimeiseksi tulemme käsittelemään opinnäytetyön teknologista ja liiketoiminnallista osuutta. Ensimmäisenä aiheena on Salesforce, toisena Amazon Web Services ja sen jälkeen liiketoiminnalliset tekijät.

2.1 Integraatio

Jotta ymmärtäisimme Salesforce-integraatiota aiheena, on ymmärrettävä mitä käsite integraatio itsessään tarkoittaa. Seuraavaksi tulemme käsittelemään integraatiota käsiteenä, mitä se tarkoittaa ja määrittelemällä, kuinka aihepiirin ympärillä olevat integraation käsitteet eroavat toisistaan.

Cambridgen (s.a) mukaan sana integraatio on määritelty toiminnaksi, tai prosessiksi, jossa kaksi- tai useampi asia yhdistetään toisiinsa toimivalla tavalla. Sana järjestelmäintegraatio (System Integration) Gartnerin (s.a) mukaan taas tarkoittaa kompleksien tietojärjestelmien, kustomoidun arkkitehtuurin tai sovelluksien luontiprosessiksi, jossa integroidaan uusi tai valmiina oleva laitteisto, ohjelmisto tai tietoliikenne.

Tätä puoltaa myös Red Hat (2017), jonka mukaan IT-integraatio, tai vaihtoehtoisesti samaa tarkoittava käsite järjestelmäintegraatio tarkoittaa datan, sovellusten, API:en ja laitteiden yhdistämistä organisaation sisällä olemaan tehokkaampia ja joustavampia. Se ei kuitenkaan tarkoita samaa kuin jatkuva integraatio (continuous integration, CI), joka viittaa kehittäjien harjoittamaan tapaan, jossa toimivat kopiot koodista yhdistetään, tavoitteena automatisoida koodin rakennus ja varmistaa, että ongelmat huomataan aikaisemmin.

Grady (1994, 16) taas on määrittänyt järjestelmäintegraation taidoksi tai tieteeksi, jossa kaksi tai useampi ratkaisua yhdistetään yhdeksi ratkaisuksi. Järjestelmäintegraatio on osa järjestelmäsuunnittelun prosessia, jossa yhdistetään tuotteiden komponentteja toisiinsa.

Järjestelmäintegraatiosta eroten, itsessään ohjelmistointegraatio (software integration) on määritelty ohjelmistomoduulien, aliohjelmien ja täysien ohjelmistojen yhdistämiseksi muiden ohjelmistokomponenttien kanssa, uuden ohjelmiston kehittämiseen tai aikaisemman ohjelmiston parannukseen (PCMAG, s.a.b).

Gredin (2022) mukaan integraatio prosessia voivat helpottaa seuraavat asiat: Luomalla kokonaisvaltaisen näkymän, jossa järjestelmät kommunikoivat keskenään, helpottuu yrityksen tietojen

siirtäminen järjestelmästä toiseen. Tiedon siirtämistä asiakkaalle helpottaa, jos tieto on helposti siirrettävissä digitaalisesti käyttämällä automatisoituja työnkulkuja. Suunnittelemalla mitä tietoja yritys tarvitsee päätöksenteossaan yritys voi parantaa asiakaskokemustaan selkeämmin. Yrityksen pitäisi rakentaa sisällöntuottajilleen välineet, joiden avulla he voisivat toimittaa tietosisältöä nopeasti, luotettavalla tavalla, jotta sisälletty tieto pysyisi eheänä. Järjestelmä huolehtii aineiston automaattisesta käsittelystä ja luokittelusta. Yrityksen tulisi pitää huoli tietojen ajantasaisuudesta ja pidettävä huoli siitä, että tiedot eri järjestelmien välillä ovat synkronoituja keskenään ja ajan tasalla. Heidän tulisi myös kehittää tiedonsiirtoa kumppaneidensa kanssa. Suunnittelemalla integraatio niin, että se kerää tiedot yrityksen eri liiketoiminnoista, helpottaen liiketoiminta kumppaneiden päätöksentekoa.

2.2 Projektinhallinta

Opinnäytetyössä tullaan käsittelemään projektin alustava toimintasuunnitelma, yhdessä suoritettavan vertailun kanssa eri integraatiototeutuksista. Tämän takia, opinnäytetyössä on järkevää käsitellä projektinhallinta aiheena, määritellä mitä se tarkoittaa ja mitä ympäröiviä käsitteitä siihen sisältyy, jotta saamme paremman kontekstuaalisen viitekehyksen projektisuunnitelman tekemiseen. Ensimmäiseksi projektinhallinnan osiossa käsitellään mitä projekti ja projektinhallinta itsessään tarkoittaa käsitteenä, jonka jälkeen siirrytään käsittelemään projektin onnistumiseen vaikuttavia tekijöitä ja projektin rajoitteita. Viimeiseksi käsittelemme projektisuunnitelmaa käsitteenä ja mitä se sisältää.

Gidon ja Clementsin (2012, 3–4) mukaan projekti on pyrkimys saavuttaa jokin ennalta määritelty maalitaulu toisiinsa liitoksissa olevien tehtävien ja töiden kautta. Projektilla on oltava selkeä tavoite, missä määritellään kaikki se mitä sen tulee saavuttaa. Tavoite määritellään tämän kyseisen konkreettisen lopputuotteen avulla, siihen liittyvien aikataulun ja budjetin kanssa. Tavoitteeseen sisältyy odotus siitä, että edellä mainitut osa-alueet toteutetaan laadukkaasti ja, että asiakas jätetään tyytyväiseksi lopputuloksesta. Itsessään projekti suoritetaan erillisten liitoksissa olevien tehtävien kautta järjestyksessä ja projektissa käytetään useita erillisiä resursseja näiden saavutukseen. Projektilla on oltava myös tietty määritelty aikajana ja elinkaari missä se tullaan suorittamaan. Elinkaaressa tulee olla aloitus- ja loppupäivämäärä ja projektilla on oltava sponsori tai asiakas, jonka vastuuna on antaa sille riittävä rahoitus. Sponsori voi olla joko yrityksen sisäinen tai ulkoinen vaikuttaja. Projekteihin kuuluu aina tietty määrä epävarmuutta. Se pohjautuu ennalta asetettuihin ennakkoluuloihin siihen liittyvistä tekijöistä eli budjetista, aikataulusta ja laajuudesta.

Projektihallinta taas määritellään prosessina, jossa tieto, taidot, työkalut ja tekniikat hyödynnetään vastaamaan projektissa asetettuja vaatimuksia. Projektijohtajien pitäisi huolehtia asetettujen tavoitteiden (laajuuden, ajan, hinnan ja laadun) täyttymisestä, kuten myös siitä, että he myötävaikuttavat

koko prosessin valmistumista vastaamaan projektin tarpeita, johon sisältyy myös huolehtiminen asiakkaiden toiveista. (Schwalbe 2015, 8).

Pinton & Slevinin (1988, 68–69) mukaan projektin elinkaareen kuuluu neljä vaihetta. Ensimmäisessä aloitusvaiheessa pyritään tekemään ja määrittelemään projektin tavoite ja missio konsultoimalla asiakasta. Projektin tavoite on oltava mahdollisimman selkeä missioon nähden. Samalla keskustellaan tärkeimpien asiakkaiden kanssa heidän tarpeistaan projektin alusta lähtien. Toisessa suunnitteluvaiheessa taas jatketaan projektin mission selvittelyä ja sen hiomista samalla, kun haetaan resursseja ja valtuutuksia johtoportaasta. Tässä vaiheessa yritetään asiakas neuvottelujen kautta päästä konsultoinnista eteenpäin ja määritellään raja asiakkaan hyväksynnälle. Pyrkimyksenä vastata siihen mitä toimintoja on tehtävä, jotta tuote pystytään myymään asiakkaalle. Samalla yritetään priorisoida kiireellisyyttä asioiden hoitamisen kannalta organisaation sisällä, erityisesti työryhmän kohdalla. Kolmannessa suoritusvaiheessa, yritetään katsoa projektin missiota ja määritellä miten projektin tulee edetä, jotta pysytään tavoitteessa. Tässä vaiheessa projektijohtajan on tehokkaasti käytettävä hallinnointitaitojaan. Tehtävänä on aloittaa ongelmanratkontaprosessi määrittelemällä metodit, joilla huomata virheitä ja korjata niitä. Ongelmanratkaisun yhteydessä, tarkoituksena on asettaa ja noudattaa aikataulua ja suunnitelmia. Teknisten ratkaisujen suorittaminen aloitetaan ja näiden ratkaisujen luomisesta ja implementoinnista vastaavat niiden tekniset asiantuntijat. Suoritusvaiheessa edelleen ylläpidetään kahden suuntaista yhteydenpitoa asiakkaaseen. Neljäntenä projektin loppuvaiheena toimii projektin terminointivaihe, jossa varmistetaan, että luotu ratkaisu on hiottu ja, että se vastaa toimeksiantoa. Samalla päätetään, että onko vielä jotakin mitä pitäisi muuttaa, jotta loppuratkaisu vastaisi paremmin projektin missiossa asetettua tavoitetta. Asiakasta konsultoidaan heidän tyytyväisyydestään lopputulokseen ennen kuin projekti päätetään lopullisesti.

Gido ja Clements (2012, 6–8) tukevat myös tätä määritelmä projektin elinkaaresta mutta, asiakkaan tarpeen lisäksi lisäten, myös liiketoimintamahdollisuuden (resurssi tai tuote, jonka he voivat valmistaa, josta voisi olla heille liiketoiminnallista hyötyä) syynä projektin käynnistymiseen. Aloitusvaiheessa, tulisi olla yleiset vaatimukset ja arviot siitä minkä rahallisten ja fyysisten ehtojen tulisi täyttyä, mikäli projektia lähdettäisiin toteuttamaan. Toisessa suunnitteluvaiheessa tultaisiin käsittelemään sitä miltä onnistunut projekti tulisi näyttämään, jotta se täyttäisi siihen liittyvät budjetilliset ja ajalliset vaatimukset yhdessä sen suunnitelman kanssa, miten onnistunut projekti tultaisiin toteuttamaan. Toteutusvaiheessa, projektin tahti lisääntyy, sillä siinä vaiheessa projektissa käytettävien resurssien määrä lisääntyy. Projektissa tapahtuvia muutoksia pitäisi dokumentoida, myöntää ja kommunikoida projektin sponsoreiden ja muiden sidosryhmien välillä. Myös tarvittavat uhraukset projektin laajuuteen, budjettiin, aikatauluun ja laatuun ovat sallittuja, mikäli projektista suoriutuminen niin vaati. Projektin viimeistelyvaiheessa, tulisi suorittaa projektiin liittyvät viimeiset maksut

yhdessä projektin lopullisen arvioinnin ja projektin aikana opittujen oppien kirjaamisen kanssa. Erityisesti projektin alku- ja suunnitteluvaiheessa on tärkeää pitää mielessä ne rajoitteet mitkä tulevat rajoittamaan projektin suoriutumista.

2.2.1 Projektin onnistumistekijät & rajoitteet

Jotta projekti voisi onnistua, tulee projektia hoitaessa otettava huomioon tärkeimmät eli kriittisimmät onnistumisen tekijät (critical success factors, CSF), jotka vaikuttavat eniten projektin lopputuloksen onnistumiseen.

Gido ja Clements (2012, 11) ovat määritelleet projektin kriittisiksi onnistumisen tekijöiksi suunnittelun ja kommunikaation yhdessä ajanottamisen toimivan suunnitelman kehittämiseen. Projektilla tulisi myös olla selvä tavoite siitä mitä projektilla halutaan suorittaa, lopputuotteen, aikataulun tai budjetin heijastamana. Projektilla tulisi myös olla sponsori tai asiakas, joka ottaa osaa projektiin aktiivisen yhteistyön kautta. Onnistunut projekti vaatii jatkuvaa yhteydenpitoa asiakkaan kanssa, pitääkseen heidät tasalla projektin etenemisestä, jotta tiedetään mikäli heidän odotuksensa projektilta ovat muuttuneet. Projektin hallinnassa tehokas keino sen hallitsemiseen on mitata projektin etenemistä ja vertailtava sitä suunniteltuun edistykseen säännöllisin aikavälein. Projektin lopussa, projektin suoriutumista tulisi myös arvioida ja punnittava mitä siitä voitaisiin oppia, mikäli yritys työskentelisi samankaltaisen projektin kanssa tulevaisuudessa.

Alias, Zawawi, Yusof ja Aris (2014, 64) ovat vaihtoehtoisesti määritelleet projektin onnistumisen tekijöiksi ne luonteenpiirteet, olosuhteet ja muuttujat, joilla on merkittävä vaikutus projektin onnistumiseen, kun ne ovat tarpeeksi hallittuja. Heidän mukaansa, vaikka tarkoista ja erillistä onnistumisen tekijöistä ja vaikutuksista on historiallisesti oltu eri mieltä eri kriittiset onnistumisen tekijät ja niiden määrittelyprosessit ovat käsitelleet melkein aina projektin kolmiorajoitteita (triple constraints), jotka tullaan esittelemään seuraavaksi. He myös toteavat, että syynä siihen miksi onnistumisen tekijöistä ollaan eri mieltä luultavasti, aiheutuu siitä, että eri projekteilla on erilaiset tarpeet ja olosuhteet ja täten eri projekteilla tulee olemaan myös eri kriittiset onnistumistekijät.

Wyngaardin, Pretoriuksen ja Pretoriuksen (2012, 1991–1994) mukaan projektissa on kolme rajoitetta (triple constraints) aika, hinta ja laajuus. Ne vaikuttavat toisiinsa muodostaen kolmiorajoite pyramidin, jossa nämä kolme rajoitetta ovat vuorovaikutuksessa toisiinsa ja riippuen toistensa onnistumisesta ne pystyvät vaikuttamaan toisiinsa niin positiivisella kuin negatiivisella tavalla. Aikarajoite sisältää itseensä kaikki ne asiat, jotka liittyvät projektin ajallisiin tekijöihin, sen aikatauluun ja kestoon. Kun taas hintarajoite asettaa projektille sen budjetin- ja aikarajoitteet. Laajuus määrittelee projektin vaatimuksia, maalitauluja ja niistä muodostuvaa työtä.

Kuva 1.Triangle constraints kuvaa kuinka projektirajoitteet vaikuttavat kokonaislaatuun (mukaillen Gido & Clements 2012, 11)

Gido ja Clements (2012, 5) ovat taas esittäneet seitsemän jakoisen mallin projektirajoitteille, jossa rajoitteiksi on määritelty: laajuus (scope), laatuodotukset (quality expectations), aikataulu (schedule), budjetti (budget), resurssit (resources), riskit (risks) ja asiakastyytyväisyys (customer satisfaction). Laajuus kuvaa valmistettavia tuotteita, joista projekti vastaa yhdessä asiakkaan asettamien vaatimusten ja kriteereiden kanssa. Budjetti vastaa projektissa käytettävää budjettia, jonka asiakas tai sponsori on myöntänyt ja se pohjautuu projektin aikana tehtyyn tutkimukseen. Aikataulu vastaa asetettuja aikoja, mihin mennessä projektin laajuudessa asetetut tehtävät on suoritettava. Resurssit taas vastaavat projektissa käytettäviä voimavaroja mitä yrityksellä on käytettävissä tehtävien suorittamiseen, kuten ihmistyövoima, materiaalit ja työtilat. Riskit vastaavat niitä vastavoimia mitkä negatiivisesti vaikuttavat projektin suoriutumiseen. Laatuodotukset asetetaan projektin alussa ja projektille asetettu laajuus on suoritettava niin, että se vastaa odotuksia. Asiakastyytyväisyyden taas on ylitettävä mitä projektissa asetettujen laajuudesta, aikataulusta ja budjetista muodostuneet oletukset määrittäisivät.

2.2.2 Projektisuunnitelma

Mäntynevan (2016, 49–50) mukaan projektisuunnitelman tarkoituksena on vastata miksi ja miten projekti suoritetaan. Hyvä projektisuunnitelma toimii tukevana dokumenttina projektin käynnistämiselle ja siinä käydään läpi asiat, jotka voivat vakuuttaa projektin tilaajaa ja ohjausryhmää siitä, kuinka projektin tavoitteet vastaavat heidän tarpeitansa ja kuinka projektin budjetti ja aikataulu hoidettaisiin. Vaikkakin todellisuudessa usean projektin kohdalla onkin päätös projektin toteutuksesta jo tehty. Projektisuunnitelma käsittelee kuinka projektin tavoitteet vastaavat yrityksen tarpeita ja miten projektin aikataulu ja budjetti pysyvät hallinnassa. Projektisuunnitelmassa kuvataan myös projektista muodostuvat tuotokset. Se on paras pitää mahdollisimman ytimekkäänä ja helposti

ymmärrettävänä. Projektisuunnitelmassa ei oteta kantaa yksityiskohtaisiin teknisiin tehtäviin vaan siihen määritellään suhteellisen pieni määrä tehtäviä, joita päivitetään ja suunnitellaan kuukausi- tai vuosineljännestasolla. Tarkoituksena on, että projektisuunnitelmassa olevia tehtäviä päivitetään koko projektin elinkaaren ajan.

2.3 Salesforce

Salesforce on keskeisessä asemassa opinnäytetyön aihealueen ja toimeksiannon puolesta, joten selvittäminen mikä Salesforce on ja mitä käsitteitä sen aihealueen ympärille kuuluu, on yksi tärkeimmistä tutkimuskysymyksistä. Aluksi Salesforcesta käsittelemme alustan pohjatietoja ja arkkitehtuuria, jonka jälkeen käymme läpi alustassa toimivia integraatioratkaisuja ja niiden teknisiä yksityiskohtia. Integraatiototeutuksien teknologiota mitä työssä käydään läpi ovat Apex ja pub/sub API. Amazon Web Services (AWS) ja siihen liittyvää AWS AppFlowta tulemme käsittelemään myöhemmässä kappaleessa. Kolmen edellä mainitun teknologian lisäksi Salesforcelle on tarjolla kolmannen osapuolen tarjoamia ratkaisuja integraatiolle, kuten Mulesoft (Mulesoft s.a.), mutta niitä ei tulla käsittelemään tässä opinnäytetyössä.

Salesforce on Salesforce nimisen yrityksen kehittämä asiakkuudenhallinta ohjelmisto, joka on julkaistu vuonna 1999 Salesforce Inc:in toimesta ex-Oracle työntekijöiden Marc Benioffin, Frank Dominiquezen ja Parker Harriksen aikaansaamana (O'Connell 27.2.2020).

Salesforcen alusta on CRM (Customer Relationship Management) -ratkaisu ja järjestelmä, jonka avulla yritykset pystyvät parantamaan asiakasvuorovaikutuksia käyttämällä Salesforcen tarjoamia palveluita ja Cloud-teknologiaa. Salesforcen alustan avulla yritykset voivat monitoroida asiakkaidensa toimintaa, markkinoida palvelujaan heille Salesforcen CRM avulla, parantaen yrityksen asiakassuhteiden hallintaa. Salesforce Cloud-alustan avulla yritykset voivat hallita oman yrityksensä myyntiä, markkinointia ja asiakaspalvelun osa-alueita toiminnassaan (Sunkari, 2022).

Salesforcen alusta toimii Software-As-A-Service (SAAS)-periaatteella, eli palvelu tarjotaan ohjelmistona, johon asiakas voi ostaa käyttöoikeuden. Ohjelmistojen ylläpidosta vastaa itse palvelua tarjoava yritys (Yazar, s.a.).

Arkkitehtuurilta Salesforce perustuu multitenant cloudiin eli useamman haltijan pilviarkkitehtuuriin, jossa useampi käyttäjä käyttää Salesforcen jakamia ydinpalveluita pilven kautta. Kaikilla käyttäjillä on käytössään samat resurssit tietojenkäsittelylle ja palvelut datan tallentamiselle, jossa Salesforce pitää huolen siitä, että vaikka käyttäjät jakavat samoja resursseja keskenään, yksittäisen käyttäjän data pysyy salassa muilta käyttäjiltä (Trailhead, s.a.a).

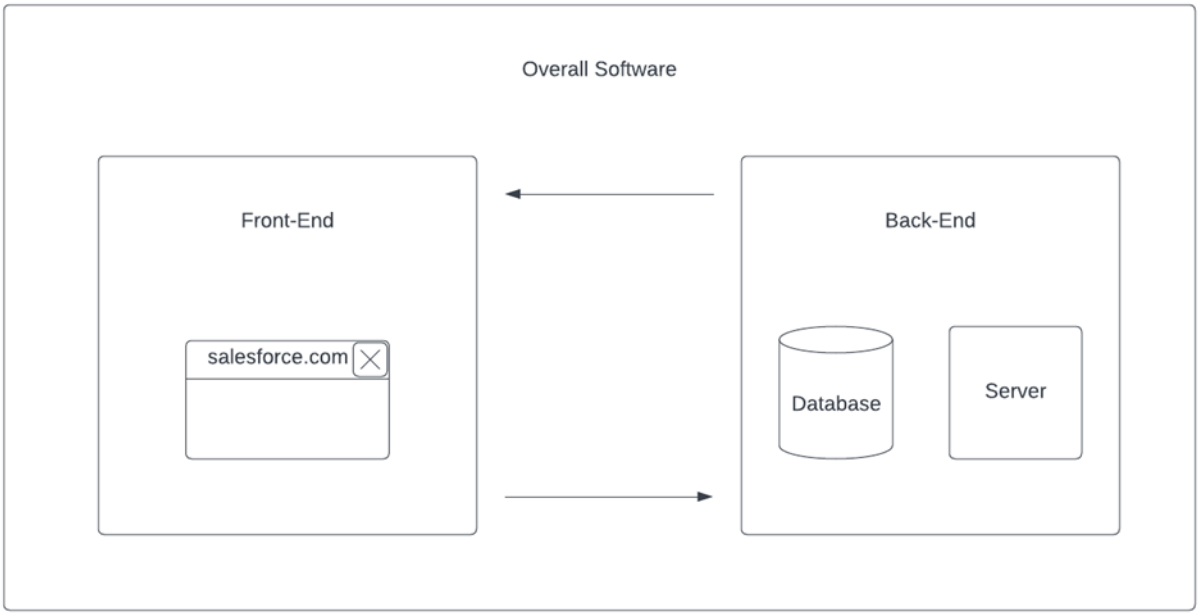
Tätä puoltaa myös Sunkari (2022), jonka mukaan Salesforcen palvelu pohjautuu yhtä tietokanta rakennetta käyttävään arkkitehtuuriin, jossa ylläpidetään dataa usealta käyttäjältä, jossa palvelimella voi olla useampi käyttäjä yhden instanssin sisällä. Tällaisessa arkkitehtuurissa Salesforce toimii sekä SAAS-palvelun tarjoajana, että PAAS (Platform as a Service) -tarjoajana. Välittämällä asiakkaalle arkkitehtuurin ja alustan, jolla luoda sovelluksia ja ominaisuuksia Salesforcen alustaan.

Metadata on Trailheadin (s.a.a.) mukaan on dataa toisesta datasta, eli metadata kuvastaa jonkin muun olevan datan kenttiä, ominaisuuksia ja konfiguraatioita. Metadata helpottaa datan talletta-mista ja käyttämistä antamalla datalle tallennuspaikan. Tätä tukee myös Red Hat (s.a.), jonka mukaan metadata on dataa datasta ja se sisältää tietoa jostain tietystä tietorakenteesta riippumatta tämän yksilöllisen rakenteen sisällöstä.

Salesforce tarjoaa käyttäjilleen laajat mahdollisuudet muokata alustaa yrityksen tarpeiden mukaan, ja jatkokehittää alustaa paremmin vastaamaan yrityksen tarpeita alustan tarjoamien API:en ja muiden teknologioiden kautta. Eri Salesforce integraatioratkaisuista tulemme käsittelemään ratkaisun käyttäen Apex-ohjelmointikieltä, sekä pub/sub API-pohjaisen ratkaisun. Myöhemmässä kappaleessa tulemme käymään myös läpi Amazon AppFlow pohjaisen ratkaisun, joka perustuu Amazonin kehittämään Amazon Web Services-palveluun.

2.3.1 Apex

Ohjelmistokehityksessä, erityisesti web-ohjelmoinnissa, backend on palvelin puolella sijaitseva osa ohjelmistoa. Se yleensä vastaa tiedon talletuksista, tiedon organisoimisesta ja varmistaa, että sovellus toimii asiakasohjelman puolella. Asiakas ei ole yhteydessä backend puolen kanssa suoraan, vaan asiakasohjelma ottaa yhteyden backendiin epäsuorasti käyttäen, asiakasohjelman puolella toimivaa frontend ohjelmistoa yhteydenpitoon. Frontend taas on se puoli ohjelmistosta, jonka kanssa käyttäjä pystyy olemaan vuorovaikutuksessa, joka sijaitsee asiakasohjelman puolella sovelluksessa. Siihen sisältyy ohjelmiston ulkoasu, kuvat, teksti eli asiat minkä kanssa asiakas on vuorovaikutuksessa (Geeksforgeeks, 2022).



(Kuva 2. frontendin ja backendin kahtiajako. Backend sisältää tietokannan ja palvelimen, sillä välin kun asiakasohjelman käsittelee frontend.)

Apex on Salesforcen kehittämä API ja objektiorientoitunut backend ohjelmointikieli, jonka avulla pystytään luomaan kustomoitua koodia eritoten tietovirtojen ja transaktioiden suorittamiseen Salesforcen alustan sisällä. Apexin avulla Salesforcessa olevaa dataa pystytään muokkaamaan ja kehittämään eteenpäin. Se muistuttaa syntaksiltaan Javaa ja sen kehitykselle löytyy oma konsoli Salesforcen sisältä. (Salesforce s.a.a).

Stroustrup (1995, 2) mukaan, ohjelmointikielen objektiorientoituneisuus tarkoittaa, että se rakentuu objektityyppisten rakenteiden varaan ja, että ohjelmointikieli tukee kolmea sääntöä.

- Abstraktio eli ohjelmointikieli antaa mahdollisuuden tai tuen luokkien ja objektien käyttöön niin, että dataa pystytään perimään toisesta.
- Periytyvyys, joka tarkoittaa, että ohjelmointikielissä käytettävät objektit, metodit ja luokat pystyvät perimään ominaisuuksia ja uusia abstraktion tasoja toisistaan.
- Suoritusaika polymorphismi, eli ohjelmointikieli tukee jotain ajettavaa ajoaikaa (runtime) kytköksenään.

Itsessään termi polymorphismi tarkoittaa, että ohjelmoinnissa käytössä oleville erityyppisille entiteeteille pystytään tarjoamaan yksi rajapinta (Stroustrup s.a.). Craig (2007, 3) mukaan kuitenkin aiemmin listatun kolmen säännön lisäksi sisältyy myös neljäs sääntö, ohjelmointikielen objektiorientoitumiselle, joka on metodien dynaaminen sitominen toisiinsa.

Java on vuonna 1991, Sun Microsystemsin kehittämä objektiorientoitunut ohjelmointikieli, joka käyttää abstraktion, kapseloinnin ja periytyvyyden käsitteitä toiminnassaan. Se rakentuu pitkälti luokkamuotoisten rakenteiden varaan, jotka toimivat objektien pohjapiirrustuksina. (TechMetrix Research 2011, 1–4). Javan käsitteen määrittelemisen tarkoituksena on selventää, kuinka Apex toimii logiikaltaan.

Stroustrupin (s.a) mukaan kapselointi tarkoittaa abstraktion pakottamista mekanismeilla, jotka välttävät pääsyä objektin tai objektijoukon yksityiskohtiin, suunnittelussa rajapinnassa. Käytännön tasolla kapselointi tarkoittaa sisäisten yksityiskohtien piilottamista.

Apex-ohjelmoinnin perustana toimii se, että koodi sijaitsee ja suoritetaan Salesforcen Force.com alustassa. Se eroaa tyypillisesti muista ohjelmointikielistä siten, että sen koodi suoritetaan ja kasataan pilvessä ja sen avulla pystytään suorittamaan SOQL-pyyntöjä (Poniszewska-Maranda, Matusiak & Kryvinska, 2017). SOQL eli Salesforce Object Query Language on Salesforcen sisällä oleva tietokanta pyyntöihin kehitetty ohjelmointikieli, jonka avulla käyttäjä pystyy suorittamaan SELECT-lausekkeita Salesforcessa olevan datan käyttöön, joka muistuttaa rakenteeltaan SQL-tietokanta kieltä. Se on kehitetty erityisesti käsittelemään Salesforcen dataa (Salesforce, s.a.b.).

Salesforcen sisällä objektit ovat tietokanta relaatioita, joihin pystytään tallentamaan yrityksen tai organisaation dataa. Salesforce objektit jakautuvat kahteen eri osioon, standardi objekteihin (standard objects) ja kustomoidut objektit (custom objects). Standardi objektit ovat objekteja, jotka Salesforce itsessään tarjoaa asiakkailleen. Kustomoidut objektit taas ovat käyttäjän itsensä luomia. Niillä on itse määritellyt kustomoidut kentät ja suhteet alustan muihin objekteihin. (Thakkar & Rajaan 2022, 15–16). Salesforcen kustomoidut kentät (custom fields), ovat kenttiä, jolle käyttäjä itse pystyy määrittelemään nimet, ominaisuudet ja datatyypin. Käyttäjä pystyy itse määrittelemään kustomoidut kentät, sekä kustomoiduille, että standardi objekteille (Salesforce s.a.c.).

Apexin avulla kehittäjät pystyvät suorittamaan tietovirtoja ja transaktioita Salesforcen palvelimilla kuten myös lisäämään logiikkaa Salesforcen järjestelmille liittyen tietojenkäsittelyyn. Suorittamalla INSERT-, UPDATE- ja DELETE-operaatiota ja lisäämällä komponentteja järjestelmän sisäistä koodia pystytään alustamaan ja aktivoimaan objekteihin liittyvien tapahtumien ja muiden web-palvelupyyntöjen kautta. (Salesforce s.a.a.)

Apex koodi aktivoidaan Apex-triggereiden (Apex-triggers) kautta. Apex-triggerit ovat Apex-luokkia, jotka käynnistyvät siinä vaiheessa, kun niihin kirjatussa tapahtumassa tapahtuu muutos tallenteeseen Salesforcen tietokannassa ja sen avulla pystytään tekemään kustomoituja komentoja Salesforcen sisällä (Trailhead, s.a.b.).

Salesforcen sisällä oleva avainten hallinta on järjestetty niin, että Salesforcessa olevia sertifikaatteja (certificates) ja avainpareja käytetään varmistamaan, että Salesforcen suuntaan tuleva pyyntö on asiakas organisaatiolta itseltään. Niin kuin myös SSL-pohjaiseen kommunikointiin ulkoisten nettisivujen kanssa (Salesforce, s.a.d.).

Api Client sertifikaattia käytetään Salesforcesta ulosmeneviä viestejä varten, delegoitujen HTTPS-pohjaisten autentikointikutsujen kanssa. Näiden sertifikaattien pitäisi olla ainoastaan käyttäjän oman organisaation tiedossa ja niiden avulla pystytään tekemään:

- Itse luotuja sertifikaatteja (self-signed certificate), jonka avulla pystytään varmistamaan, että tieto tulee yksinomaan asiakasorganisaatiosta.
- Generoimaan sertifikaatteja, jotka sertifikaatti auktoriteetti on allekirjoittanut ja täten pätevä. Sertifikaatin käyttäjä voi itse ladata Salesforceen.
- Asettaa yhteisautentikointi sertifikaatin, jonka avulla pystytään varmistamaan, ettei kukaan teeskentele olevansa allekirjoittava pääty.
- Hallitsemaan master encryption-avaimia, jonka avulla pystytään salaamaan Salesforcessa olevien yksityisten custom-kenttien sisältöä. Kyseistä avainta hallitsee Salesforcen omaava asiakasorganisaatio ja se kenellä on pääsy siihen, riippuu organisaation omista turvallisuus ja sääntö konfiguraatioista (Salesforce, s.a.d.).

Salesforcen alustan sisällä applikaatioiden ja komponenttien välitys hoidetaan Salesforcen omien packageiden ja managed eli hallituiden packageiden kautta. ISV (independent software vendor) ja Salesforcen kumppanit pystyvät paketoimaan ja välittämään sovelluksia ja komponenttejaan toisille käyttäjille ja organisaatioille näiden kautta. (Salesforce s.a.e.)

Packaget ovat Salesforcessa kontteja, jotka sisältävät yksittäisiä komponentteja tai yhtenäisiä sovelluksia. Packageja voidaan välittää Salesforcessa asiakkaille ja kolmansille osapuolille yrityksen ulkopuolella (Salesforce s.a.f.). Packageja täytetään metadatalla ja package sisältää tähän metadataan liitetyt ominaisuudet, muokkaukset ja tietokantarakenteen. Packageilla on oma eritelty elämänkaari, johon sisältyy metadatan lisäys ja uuden package-version luonti. Packaget ovat omia muuttumattomia artefakteja (eli paketoitua, valmista koodia), eli joka kerta kun sen sisältämää metadataa muutetaan joko lisäämällä, poistamalla tai muokkaamalla, luodaan myös uusi package-versio (Salesforce s.a.g.). Packageiden metadata pystyy sisältämään erilaisia paketoituja ratkaisuja kuten Apex-koodia ja Salesforce lightning-komponentteja.

Packaget jaotellaan kahteen tyyppiin hallitsemattomiin (unmanaged) ja hallittuihin (managed) packageihin. Unmanaged packaget on yleensä tarkoitettu open-source eli avoimen lähdekoodin sovelluksiin ja komponentteihin, joita muut kehittäjät voivat käyttää rakenteina omissa sovelluksissaan. Kehittäjät pystyvät muokkaamaan niitä organisaation sisällä, ja alkuperäisellä kehittäjällä ei

ole hallintaa ladatuista komponenteista, eikä hän pysty muuttamaan tai päivittämään niitä. Managed packaget taas ovat packageja joiden tarkoituksena on välittää ja myydä sovelluksia ja komponentteja asiakkaille käyttämällä Salesforcen AppExchangea ja License Management Applicationia (LMA). Näiden packageiden myynti ja välitys tapahtuu käyttämällä käyttäjäpohjaisia lisenssejä applikaatioille ja ne ovat täysin päivitettävissä. Managed packageissa käyttäjä ei itse pysty tekemään muutoksia koodiin tai poistamaan objekteja tai kenttiä. (Salesforce s.a.f.). Managed packageiden sisällä protected custom settingit ovat kustomoituja Salesforce asetuksia, johon vain saman packagen sisällä oleva Apex koodi pystyy yltämään (Salesforce s.a.h).

AppExchange on Salesforcessa oleva myyntialusta, jossa myydään ja välitetään sovelluksia, komponentteja ja konsultointipalveluja. Myyjinä toimivat Salesforcen itse hyväksymät kumppanit eli Salesforce partnerit (Salesforce s.a.i). Salesforce License Management App on taas Salesforcen tarjoama sovellus, jonka avulla käyttäjä pystyy hallinnoimaan omia myynnin johtolankoja ja lisenssejä AppExchangen rinnalla (Salesforce s.a.j.).

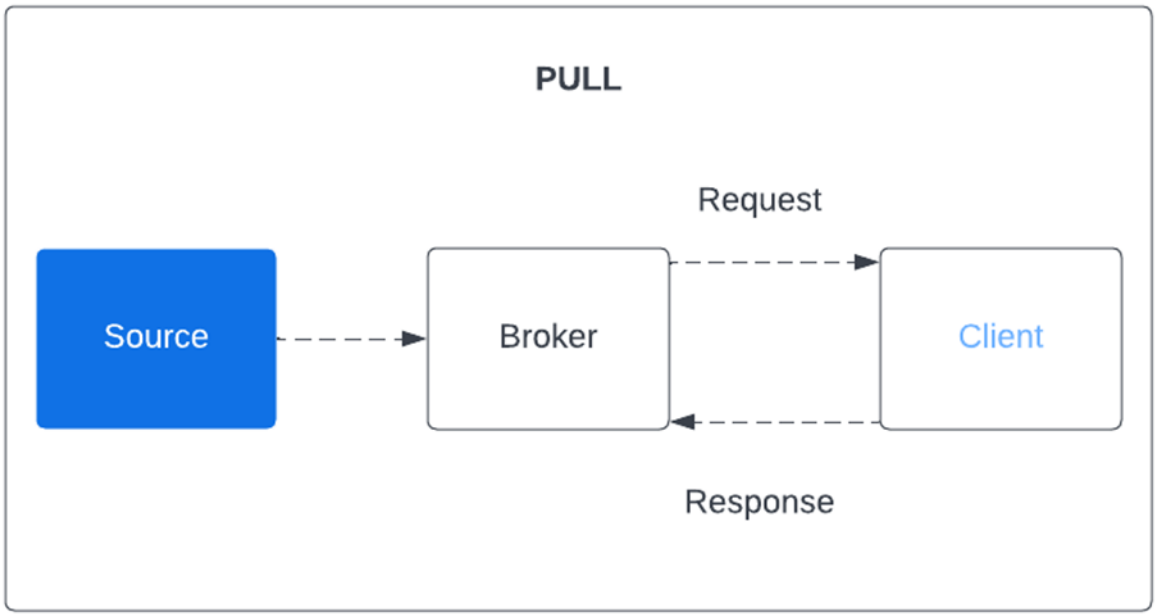
Platform eventit ovat osa Salesforcen yritysviestintäalustaa. Alustassa on tapahtumapohjainen viestintä arkkitehtuuri, jonka kautta ohjelmistot kommunikoivat keskenään Salesforcen kanssa (Salesforce s.a.k.). Platform eventit ovat turvattuja, skaalautuvia viestejä, jotka sisältävät dataa. Platform eventejä käyttämällä kehittäjät pystyvät tilaamaan dataa platform eventeistä suoraan Salesforcen tietovirtoihin, Apex-triggereihin ja CometD-asiakasohjelmaan (Salesforce s.a.l.). Salesforcen Developer Console on osa Salesforcen kehitysympäristöä, jossa kehittäjä pystyy ajamaan SOQL-lauseita, luomaan Apex-koodia ja testaamaan omia sovelluksia (Salesforce s.a.m.).

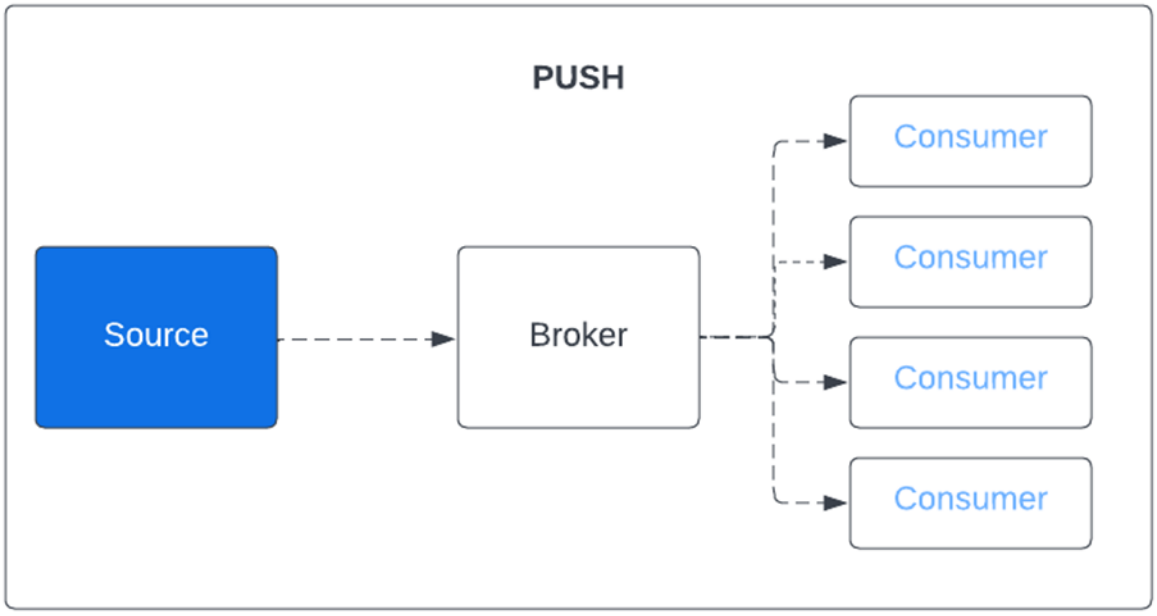
2.3.2 pub/sub API

Tapahtumapohjainen viestintä arkkitehtuuri on arkkitehtuuri malli, mikä koostuu tapahtumien tuottajista (event producers), tapahtumien käyttäjistä (event consumers) ja kanavista (channels). Arkkitehtuuri jaottelee tapahtumien tuottajat keskenään, helpottaen kommunikointia järjestelmien välillä. Platform eventeja julkaistaan tässä arkkitehtuurissa event bus nimisen sanoman välittäjän kautta, mihin ne väliaikaisesti talletetaan. Platform eventejä pystytään vastaanottamaan Salesforcesta käyttäen Streaming API:eja kuten CometD:tä tai Salesforcen pub/sub API:a. Tapahtumapohjaisessa viestintä arkkitehtuurissa: Tapahtuma (event) vastaa muuttuvaa tilaa keskellä liiketoimintaprosessia ja tapahtumaviesti (event message) vastaa tapahtumassa olevaa dataa. Tapahtuman tuottaja (event producer) taas on tapahtumaviestin julkaisija ja kuluttaja (event consumer) on sen kanavan tilaaja, joka vastaanottaa viestin. Tapahtumakanava (event channel) on se viestintävirta missä julkaisija lähettää tapahtumaviestit liikkeelle ja tapahtuman kuluttaja lukee ne (Salesforce s.a.n.).

Kuva 3. Kuvastaa tapahtumapohjaisen pub/sub mallin rakennetta (mukaillen Salesforce, s.a.i.)

Jansenin ja Saladaksen (11.4.2021) mukaan tapahtumapohjainen arkkitehtuuri määritellään järjestelmäksi, jossa yhdistetyt mikropalvelut siirtävät dataa toistensa välillä tapahtumien luonnin ja käytön kautta. Järjestelmä mahdollistaa tapahtumien vastaanottamisen tapahtumapohjaisen ekosysteemin kautta, josta ne lähetetään palveluille, jotka ovat aiheista kiinnostuneita. Tapahtumat (events) on määritelty tallenteiksi jostain ennalta tapahtuneesta. Niiden tilaa ei voida muuttaa ja ne on järjestelty sekvenssissä luontijärjestyksen mukaan. Näistä kiinnostuneet osapuolet voivat vastaanottaa tietoa näistä tilan muutoksista tilaamalla niitä pub/sub-järjestelmän kautta. Tapahtumapohjainen järjestelmä eroaa pull-pohjaisista järjestelmistä, jossa datan välitys toimii request/response mekanismilla, jossa vastaanottaakseen dataa joudutaan ensiksi lähettämään pyyntö välittävälle osapuolelle. Tapahtumapohjaisissa järjestelmissä tiedon välitys perustuu push-pohjaiseen datan kuljetukseen, jossa tiedon välitys tapahtuu ilman erillisiä pyyntöjä. Viestin vastaanottajat saavat tapahtumien päivitykset suoraan lähteestä välittäjän kautta ilman kyselyä.





(Kuvat 4 & 5. Demonstroi eroa pull- ja push- tiedonvälitys tapojen välillä (mukaillen Jansen & Saladas, 11.4.2021)

Pub/sub eli publish subscribe-malli on ohjelmointiparadigma, missä informaation julkaisija eli publisher välittävät dataa injektoimalla sitä järjestelmään dokumentteina. Tämän jälkeen tiedon kuluttajat (eli tilaajat), käyttämällä erilaisia tilauksia, tilaavat dataa julkaisijalta ja näiden muodostama

pub/sub-järjestelmä on sen jälkeen vastuussa julkaistujen dokumenttien välittämisestä suoraan tilaajalle. Tilaajalla on oltava näitä dokumentteja vastaava tilaus. Järjestelmä mahdollistaa skaalattavan ratkaisun, jonka avulla tietoa voidaan välittää useampien julkaisijoiden, ja tilaajien kesken niin, että tilaajat yksinkertaisesti vastaanottavat heitä kiinnostavan tiedon ilman, että heidän tarvitsee tietää julkaisijasta erillisiä tietoja (Majumder, Shrivastava, Rastogi & Srinivasan 2009, 567).

Salesforcella on käytössään oma pub/sub-API, jonka avulla käyttäjä pystyy yhden rajapinnan kautta julkaisemaan ja tilaamaan platform eventejä, datamuutos tapahtumia, tapahtuman monitorointitapahtumia ja Salesforcen yleisiä standardi tapahtumia. Sallien skaalattavan ja turvallisen tavan tapahtumien julkaisemiselle. Tapahtumat julkaistaan event bus tapahtuma varastossa (jossa useampi asiakas voi käyttää varastoa samanaikaisesti yhdessä) ja tapahtumat varastoidaan sinne väliaikaisesti. Salesforcessa event busin toiminta perustuu aikajärjestykselliseen tapahtumalokiin, jossa tapahtumat varastoidaan siinä järjestyksessä missä ne on vastaanotettu Salesforcen sisällä. Tilaaja voi vastaanottaa tapahtumansa event busista pub/sub API:n kautta tekemällä tilauksen siihen. Jokaisella event busiin tallennetulla tapahtumalla on oma replay-id eli identifioiva tunnus, jonka avulla tunnistetaan tapahtumia tietovirrassa toisistaan. (Salesforce s.a.o.). Tapahtumat säilyvät Salesforcen event busin sisällä 24-tuntia, jos kyseessä on tavallinen Salesforce tapahtuma, mutta korkeamman volyymin tapahtumat kuten Platform eventit ja Change Data Capture Eventit säilyvät Event Busissa 72-tuntia (Salesforce, s.a.p.).

2.4 Amazon Web Services

Amazon Web Services, eli AWS on vuonna 2002 Amazon.com Inc:in kehittämä pilvialusta, joka tarjoaa IT-infrastruktuuria pilvipalvelun välitykselle yhdessä muiden Cloud Computing palveluiden kanssa (AWS, s.a.a.). Vaikka itsessään AWS:n alusta tarjoaa useita erilaisia sisäisiä palveluja, löytyy sieltä myös Salesforcelle tukea kehittämänsä Amazon AppFlow nimisen web-palvelun kautta. Seuraavaksi aiheena käsitellään AWS:n tarjoamaa Amazon AppFlow-palvelua ja kuinka integraatio Salesforceen pystyttäisiin toteuttamaan sitä käyttäen teknisellä tasolla.

2.4.1 Amazon AppFlow

Amazon AppFlow on AWS:n tarjoama integraatiopalvelu, jonka avulla voidaan turvallisesti siirtää dataa erilaisten SAAS-palveluiden (kuten Salesforce) välillä. Se käyttää itsestään skaalautuvia tietovirtoja ilman, että kehittäjän itse tarvitsee ohjelmoida koodia datan siirtoon. (AWS s.a.b.). AppFlow on kaksisuuntainen tiedonsiirtopalvelu, jolla on oma lähde(source-destination), joka vastaa AppFlowin tukemia SAAS-palveluita ja määränpää (end-destination), joka vastaa sitä palvelua johon data lopullisesti siirretään. Lähde ja määränpää ovat molemmat asiakkaan itse

päätettävissä. AppFlow kutsuu lähdettä ja siirtää datan määränpäähän (AWS s.a.c.), jossa sen tietolähteenä yleensä toimii kyseessä oleva SAAS-palvelu.

AppFlowin avulla käyttäjä pystyy luomaan tietovirtauksen lähteen ja määränpään välille, suorittamaan niitä tarpeen mukaan ja täten pitämään dataa ajan tasalla eri sovellusten välillä. AppFlow palvelun kautta pystytään keräämään dataa useammasta lähteestä, monitoroimaan dataa, pitämään siellä oleva tieto turvassa (sillä AppFlow salaa sen sisällä olevan, sekä suoritettavan datan) käyttäen AWS:n omaa PrivateLink palvelua, kuljettaen datan yksityisesti AWS:n sisäisessä verkossa julkisen verkon sijaan. (AWS, s.a.d.). AppFlowin pystyy yhdistämään useaan erilliseen AWS-palveluun, joista yksi on EventBridge.

2.4.2 EventBridge

EventBridge on AWS:ssä toimiva serverless-pohjainen event bus sanomanvälitys palvelu, jonka tarkoituksena on yhdistää sovellukset toisiinsa välittämällä tietoa niiden välillä (AWS, s.a.e.). Sana serverless tarkoittaa pilvipohjaista kehitysmallia, missä kehittäjän ei tarvitse itse järjestää palvelimia tai siihen liittyvää palvelun skaalausta. Palvelinten hallinta on abstraktoitu pilvipalvelun tarjoajalle, joka hoitaa palvelimien järjestämisestä, ylläpidosta ja skaalautumisesta aiheutuvat kustannukset, jättäen sovelluslogiikan rakentamisen ja deploymentin kehittäjän hoidettavaksi (Red Hat, 2022).

Adzicin ja Chatleyn (2017) mukaan serverless vastaa uudenlaista sovelluslogiikkaa ja PaaS (Platform as a Service) tarjontaa, missä infrastruktuurin tarjoaja ottaa vastuun asiakkaan lähettämistä pyynnöistä, niihin vastaamisesta, kapasiteetin suunnittelusta ja näiden aikatauluttamisesta. Kehittäjän vastuulla on huolehtia ainoastaan prosessointilogiikasta, eroten huomattavasti logiikasta, jossa PaaS-tarjoaja huolehtii ainoastaan alustan tarjoamisesta asiakkaalle. Serverlessissä ideana on, että palvelun tarjoaja toimittaa funktioita, jotka toimivat tapahtuman käsittelijöinä. Asiakas maksaa ainoastaan CPU-kustannuksista, kun funktiot ovat käytössä eikä silloin, kun ne ovat olemassa passiivisena.

EventBridge toimii niin, että se välittää reaaliajassa jonon dataa eri SAAS-sovelluksista AWS:n omiin kohteisiin tai vastaavanlainen toiseen SAAS-sovellukseen. EventBridge vastaanottaa tapahtuman, jonka jälkeen se käyttää siihen yksittäisessä EventBridgessä asetettuja sääntöjä päättämään mihin data välitetään. Nämä säännöt (event rules), voivat pohjautua joko tapahtuman rakenteeseen (event pattern) tai säännössä määriteltyyn aikatauluun. Jokainen EventBridgeen tuleva tapahtuma sidotaan johonkin event busiin, johon säännöt on asetettu, jonka avulla event bus arvioi datalle lopullisen päämäärän. EventBridgessä olevat määränpäät ovat HTTP-päätepisteitä, joiden

avulla EventBridgestä pystytään tekemään kutsuja reitittämään tapahtumia AWS-palveluihin ja integroituihin SAAS-palveluihin. (AWS, s.a.e).

2.4.3 S3 – Simple Storage Service

Amazon Simple Storage Service (S3) on AWS:n tarjoama pilvitallennuspalvelu, jonka tarkoituksena on tarjota skaalautuva ja matalan maksukynnyksen palvelu datan tallennukselle AWS-palvelualustan sisällä. Data S3:n sisällä on jaettu kahteen tasoon, bucket- ja objektitasoon. S3-bucketit luonteeltaan muistuttavat kansioita ja kontteja, eli niillä on globaalisti uniikki nimi, jonka avulla käyttäjä pystyy organisoimaan dataa, tunnistamaan käyttäjä, jota laskutetaan ja helpottamaan omaa datan siirtoa (Palankar, Iamnitchi, Ripeanu & Garfinkel 2008, 65). Yksittäisellä AWS-tilillä voi ennalta olla maksimissaan 100 erillistä bucketia, jota voi kuitenkin lisätä lähettämällä pyynnön siihen käyttämällä AWS:n Service Quotas-konsolia (AWS, s.a.f.).

Objektit S3:ssa taas vastaavat yleisiä objekteja, jotka tallennetaan S3:een ja ne muodostuvat käyttäjän määrittelemistä nimiarvopareista, kuten myös omasta blob-datasta ja metadatasta, joka muodostuu pienistä ennalta määritellyistä arvoista. S3:n avulla käyttäjät pystyvät suorittamaan perinteisiä CRUD (create, read, update, delete) -operaatioita, ja näitä operaatioita rajoittaa niille määritellyt pääsynhallinta määritykset S3:ssa (Palankar ym. 2008, 65).

2.5 Liiketoiminnalliset tekijät

Muodostaaksemme projektisuunnitelman on projektinhallinnan ja teknologioiden lisäksi hyvä käsitellä liiketoiminnallisia tekijöitä ja työkaluja, jotka liittyvät aihealueeseen. Seuraavaksi käsittelemme business impact analyysin käsitteen, jonka jälkeen käsittelemme Euroopan Unionin yleistä tietosuoja asetusta. Näiden jälkeen siirrymme opinnäytetyön empiiriseen osaan.

Business impact analyyysi (BIA) on prosessi, jossa määritellään liiketoiminta aktiviteettien ja siihen liittyvien resurssi vaatimusten kriittisyys. Näin pystytään ennalta varmistamaan operaatioiden kesto liiketoiminnan häiriöiden aikana ja kyseisten häiriöiden jälkeen. BIA:ssa arvioidaan häiriöiden vaikutus palveluun, sen toimittamiseen ja palautumisajan tavoitteet ja siihen liittyvät palautumisajan asteet. BIA:ssa analysoidaan kriittisiä resursseja eli tässä tapauksessa avainsidosryhmiä, tuotteita ja palveluja yhdessä näiden tärkeyden kanssa, kun niitä mitataan ja verrataan tärkeimpiin suoritettaviin aktiviteetteihin arvoketjussa (Gartner s.a.).

Tjoan, Jakoubin & Quirchmaurin (2008, 3) mukaan tärkeimmät asteet BIA:n suorittamisessa ovat:

- Liiketoiminta aktiviteettien tunnistaminen yhdessä niistä vastuussa olevien johdon omistajien (toimitusjohtaja, tuoteomistaja ym.) kanssa.
- Riittävän informaation keruuseen tarvittavan henkilökunnan tunnistaminen.

- Vakavia yrityksen imagoon, varoihin tai taloudelliseen asemaan vaikuttavien tilanteiden tunnistaminen.
- Aikarajan tunnistaminen missä liiketoiminnan aktiviteettien keskeyttämistä ei voi hyväksyä ja riippuen tilanteesta voidaan tässä kohdin soveltaa erilaisia tiedonhakumenetelmiä.

Suorittaessaan toimintaansa, yritysten on myös huomioitava asiakkaiden tietoihin liittyvät lait. Yrityksillä ja rekisterien ylläpitäjillä on velvollisuus poistaa rekisteröityjen henkilöiden tiedot pois rekisteristä, mikäli tietoja ei enää tarvita (Euroopan parlamentin ja neuvoston asetus luonnollisten henkilöiden suojelusta henkilötietojen käsittelyssä sekä näiden tietojen vapaasta liikkuvuudesta ja direktiivin 95/46/EY kumoamisesta 2016/679/EU). Tätä tietoa tullaan käyttämään myöhemmin opinnäytetyön empiirisessä osiossa, kun käsittelemme projektisuunnitelmaa.

3 Opinnäytetyön suunnitelma

Toimeksiannon mukaan, tarkoituksena opinnäytetyössä on suorittaa tutkimus ja muodostaa 10Dukelle analyysi toimeksiantajan toivomista integraatiototeutuksista, ja siitä kuinka ne eroavat toisistaan. Analyysin pohjalta muodostetaan integraatiovalinta 10Dukelle sen teknologian pohjalta mikä yritykselle parhaiten soveltuu heidän tarpeensa integraatiolle huomioon ottaen. Yrityksellä on tarkoitus aloittaa integraatioprojekti, jonka pohjalta toimeksiantaja tekisi omaan käyttöönsä Salesforce integraatiokomponentin tuotteena, jonka he voisivat myydä asiakkailleen. Analyysin ja projektisuunnitelman tarkoituksena on vastata mitä teknologiaa hyödyntämällä heidän kannattaa lähteä suorittamaan projektia, ja mikä olisi integraatioprojektin alustava suunnitelma, jotta he tietävät kuinka edetä prosessissa.

Analyysissä tarkoituksena on selvittää, kuinka eri integraatio toteutukset, eroavat toisistaan eroavat ja mitkä ovat niiden yksittäiset tekniset vahvuudet ja heikkoudet. Analyysi tullaan suorittamaan eri integraatiototeutusten käytännöllisten toteutusten pohjalta, jotka ovat alustavia demonstraatioita siitä, kuinka integraatio voisi tapahtua kyseisiä teknologioita käyttäen. Tarkoituksena demonstraatioissa ei ole luoda täydellistä tai kokonaista konkreettista versiota siitä kuinka eri integraatiototeutukset toimisivat, vaan pikemminkin demonstraatioina siitä kuinka teknologiat toimivat pohjimmiltaan ja mitä teknisiä yksityiskohtia kannattaisi ottaa huomioon, kun yritys on tekemässä omaa tuotettaan. Analyysi tulee pohjautumaan käytännön toteutuksissa tehtyihin havaintoihin ja analyysissä tullaan vertaamaan käytännön toteutuksessa läpikäytyjä teknisiä yksityiskohtia toisiinsa. Näiden pohjalta analyysi osiossa, tullaan tekemään business impact analyysi eri integraatiovaihtoehdoista ja arvioidaan riskejä mitä yksittäisessä teknologiassa on. Näiden pohjalta tehdään myös kustannusvaikutus arvio siitä kuinka yksittäinen integraatiototeutus tulisi vaikuttamaan kuhunkin yrityksen sisäiseen sidosryhmään. Hyödyllisin integraatiototeutus on se, millä on vähiten teknisiä rajoituksia, joka olisi estämässä yritystä tekemästä Salesforce integraatiokomponenttia tuotteena ja se millä on vähiten vaikutusta yrityksen sisäisiin sidosryhmiin.

Opinnäytetyöstä muodostuva analyysi ja projektisuunnitelma on tarkoitettu 10Duken projektijohdon käytettäväksi, jotta he pystyvät konkretisoimaan miksi integraatioprojektissa käytetään tiettyä teknologiaa ja selkeyttämään kuinka projektin tulisi edetä ja minkälaisia haasteita integraatioprojektissa voisi tulla vastaan. Tarkoituksena on myös esittää ehdotuksia näiden estämiselle.

Rajoittavat tekijät opinnäytetyön empiirisessä osiossa ovat, että kaikkia Salesforce integraatiototeutuksia ei ehditä opinnäytetyön ja yrityksen projektin aikarajoitteissa tutkimaan ja sen takia opinnäytetyössä keskitytään pääosin kolmeen seuraavaan teknologiaan, jotka ovat Salesforcen Apex ohjelmointikieli, pub/sub API ja Amazon AppFlow. Mulesoftia tai muita kolmannen osapuolen integraatioratkaisuja ei pystytä opinnäytetyön rajoissa käymään läpi.

3.1 Tuotosten tuottaminen

Tuotosten tuottaminen tapahtuu tekemällä alustavat demonstraatiot kustakin kolmesta integraatioteknologiasta eli Apexista, pub/sub API:sta ja Amazon AppFlowista. Tuotokset luodaan esittelemällä niissä käytettävät teknologiat ja työkalut, jonka perusteella tehdään alustava demonstraatio kustakin teknologiasta, niin että niissä päästään vaiheeseen missä ne pitäisi enää vain yhdistää ulkoiseen järjestelmään. Tarkoituksena, ei ole, että tuotokset olisivat valmiita yhdistämiseen toimeksiantajan omaan järjestelmään. Kunkin konkreettisen demonstraation aikana tehdään havaintoja niiden vahvuuksista ja heikkouksista, jonka perusteella muodostetaan vertailu ja analyysi niiden toimivuudesta. Tämän pohjalta luodaan eri integraatiototeutuksista analyysi, joka vertaa integraatiovalintoja toisiinsa ja tekee business impact analyysin riskeistä, joita kussakin integraatiototeutuksessa esiintyy. Business impact analyysin myötä tehdään myös kustannusvaikutus analyysi niiden vaikutuksesta yrityksen sisäisiin sidosryhmiin. Analyysin pohjalta, luodaan alustava päätös ja projektisuunnitelma minkä integraatioteknologian kanssa yritys etenee projektissaan ja kuinka projekti suoritettaisiin, joka tulee analyysin kanssa toimimaan opinnäytetyön tuotoksena.

4 Käytännön toteutus

Käytännön toteutuksissa luodaan alustavat demonstraatiot opinnäytetyössä käsitellyistä teknologioista. Tarkoituksena ei ole luoda täyttä, tuotantovalmista prototyyppiä tai ratkaisua vaan pikemminkin rakentaa perusrakenne kyseisestä integraatiototeutuksesta. Ensimmäisenä demonstraationa tulee toimimaan Apex, jonka jälkeen käsittelemme pub/sub API- ja AWS AppFlow-pohjaiset ratkaisut.

4.1 Apex toteutus

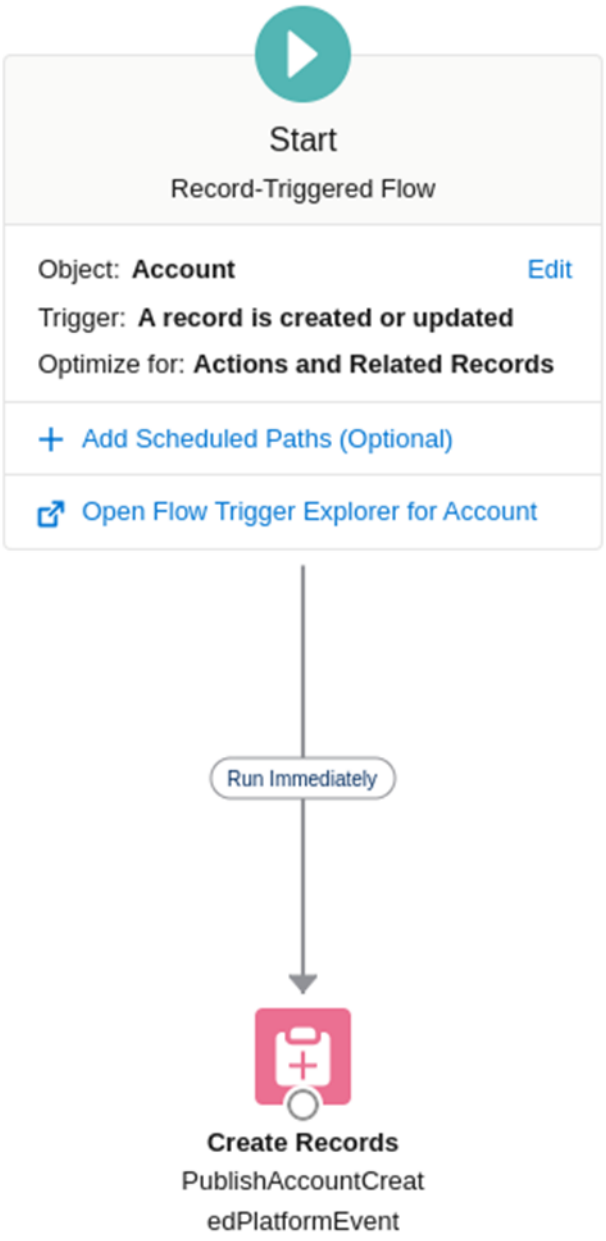
Ensimmäisenä käymme läpi käytännön toteutuksen käyttämällä Salesforcen Apex ohjelmointikieltä. Asiat mitä toteutuksessa selvitetään ovat, kuinka data saadaan Salesforcen Change Data Capture Eventistä Apex-luokkaan saakka, yhdessä alustavan luuranko rakenteen Apex-luokasta, joka hypoteettisesti tekisi kutsun ulkoisen järjestelmän HTTP-osoitteeseen.

4.1.1 Apex työkalut

Apexissa suurimpana käytettynä työkaluna tulee toimimaan edellä mainittu Salesforce Developer konsoli. Siellä käyttäjä pystyy luomaan erillisiä Apex-pohjaisia luokkia, joita käyttäjä pystyy hyödyntämään Salesforce alustassaan. Kaikki Apex-koodi luodaan näiden luokkien varaan ja niissä oleva koodi ajetaan käyttöön käyttämällä Apex-trigger luokkia.

4.1.2 Apex demonstraatio

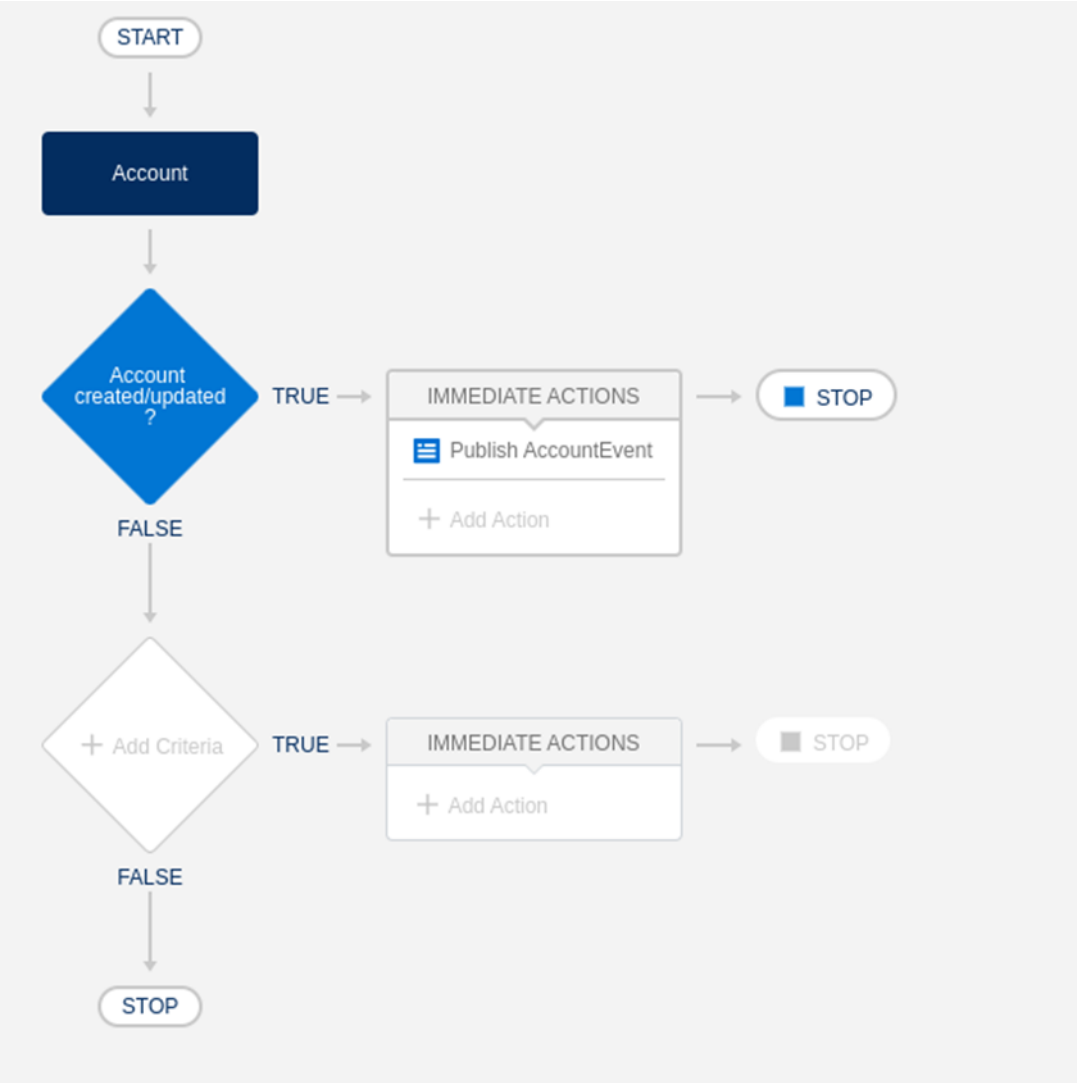
Jotta Salesforcen sisäinen data saataisiin kulkemaan Apex-triggereihin Salesforcen alustan läpi, tarvitsemme sisäisen tavan datan kuljetukseen alustan sisällä. Tähän toimii edellä mainitut Salesforcen tietovirtaukset (flow), jotka kuuntelevat Salesforcen sisäisiä tapahtumia. Flowien avulla Salesforcen sisäisten tapahtumien tiedot välitetään määrättyihin päätepisteisiin, jotka pystytään määrittämään Salesforcen Flow Builder-työkalussa. Toimeksiannon tapauksessa nämä välitetään Salesforcessa sijaitseviin, edellä mainittuihin Platform eventeihin. Tiedon kulkeminen flowin kautta on kuvattu kuvassa 6.



(Kuva 6. Kuvankaappaus Salesforcen Flow näkymästä. Flow eli tietovirta aktivoidaan tilan muutos tapahtumassa, Salesforcen Account-objektissa, joka laukaisee platform eventin.)

Vaihtoehtoisesti kehittäjä pystyisi luomaan tiedon platform eventeihin käyttämällä Salesforcessa olevaa Process Builder työkalua (kuva 7). Process Builder on kuitenkin vanhempi työkalu kuin

Flow Builder ja Salesforce on aikeissa poistaa sen. Tiedon kulku käyttäen Process Builderia tulisi näyttämään seuraavalta.



(Kuva 7. Kuvankaappaus Salesforcen Process Builderista. Vaihtoehtoinen näkymä kuvaan 6, jossa tietovirta luodaan Salesforcen vanhaa Process Builderia käyttäen, Flow-näkymän sijaan.).

Platform eventit kuljettavat tiedon Apex-triggereihin, jotka kuuntelevat tapahtumien aktivoitumista ja vastaanottavat tapahtuman tiedot sisälleen. Apex-triggerit vastaanottavat tiedon, jonka jälkeen ne siirtävät tiedot itse Apex-luokan koodiin parametreina.

Apex-luokissa pystymme luomaan yhteyden kolmanteen osapuoleen tekemällä HTTP-pyynnön loppukohteen URL-osoitteeseen. Toimeksiannon mukaisesti kuitenkin suurin huomio, mikä tässä vaiheessa esiintyi, oli tietoturvan kannalta luotettava ja riittävä toteutus. Apexin avulla tietoturva pystytään järjestämään käyttämällä algoritmipohjaista tiedon salausta. Tiedon kulun salaamiseen toimeksiannon mukaisesti päätimme, että parhaimpana tapana tiedon salaukseen tulisi toimimaan

RSA-SHA256-pohjainen algoritmi salaus, jossa tiedon lähettäjä tunnistaa ainoastaan julkisen avaimen tai sertifikaatin. Tiedon vastaanottaja tulisi tietämään yksityisavaimen tai oman sertifikaatin. Tiedon julkaisija allekirjoittaa digitaalisen allekirjoituksen salattuun tietoon käyttäen hänen julkista avaintaan, jonka tiedon vastaanottaja varmentaisi yksityisavaimensa tai sertifikaatin kautta. Tämän pohjalta tiedon vastaanottaja pystyisi varmistamaan, että tuleva tieto on alkuperältään lähtöisin oikeasta lähteestä. Tämän jälkeen tiedon vastaanottaja pystyisi määrittämään mitä HTTP-yhteydellä vastaanotetulla tiedolla tehtäisiin yrityksen sisäisessä logiikassa. Sertifikaatti, jolla viesti salataan, pystytään Salesforcen alustassa tallentamaan Key Management-palveluun ja sitä voidaan kutsua Apexin koodista seuraavasti (kuva 8).

```
Blob data = Blob.valueOf(eventDataFromSalesforce);
Blob signature = Crypto.signWithCertificate('RSA-SHA256', data, 'signingCert');
```

(Kuva 8. Kuvakaappaus Apex-koodista. Demonstroi kuinka SF-dataa kryptataan käyttäen Apex:issa toimivaa Crypto-luokkaa.)

Vaihtoehtoisesti avain voidaan myös tallettaa Salesforcen custom objektin salattuun kenttään tai custom asetuksiin. Ongelmana kuitenkin sen salaus sillä tämä vaatii, että asiakasorganisaation Salesforcen sisällä on oltava riittävät suojaukset ja pääsyoikeudet. Niiden on oltava määritelty kullekin käyttäjälle, jotta avain ei ole tiedossa jokaiselle käyttäjälle organisaation sisällä ja vain pakollisilla käyttäjillä on siihen pääsy. Toisena ongelmana on, että custom objektien salatuissa kentissä on maksimi merkkipituus (175 merkkiä pitkä) ja tämän takia mm. käyttäen RSA-SHA256-salausmenetelmää ongelmaksi nousee, että avaimen merkit jouduttaisiin pilkkomaan eri osiin ja täten tallettaa useampaan kenttään. Tämä sama ongelma toistuisi protected custom settingienkin kanssa. Nämä osat jouduttaisiin kokoamaan yhteen, joka on huonoa koodi käytäntöä ja täten sitä pitäisi välttää. Konseptuaalisesti, koodin ulkoasu olisi tullut näyttämään rakenteellisesti seuraavalta (kuva 9).

```
public class DemoRequest {

        @future(callout = true)
        public static void sendData(String accountId){

            String username = 'myname';
            String password = 'mypwd';

            //RETRIEVING DATA FROM ACCOUNT OBJECT WITH MATCHING ID
            Account acc  = [SELECT FIELDS(STANDARD) FROM Account WHERE id=:accountId LIMIT 200];

            //SIGNING A DIGITAL CERTIFICATE ON APEX TO VERIFY THE INTEGRITY OF THE EVENT SOURCE
            Blob data = Blob.valueOf(username + ':' + password);
            Blob signature = Crypto.signWithCertificate('RSA-SHA512', data, 'MyTestKey');
            String authorizationHeader = EncodingUtil.convertToHex(signature);

            //GENERATING JSON BODY FOR THE REQUEST
            JSONGenerator generator = JSON.createGenerator(true);
            generator.writeStartObject();
            generator.writeStringField('salesforceId', AccountId);
            generator.writeEndObject();
            String jsonBody = generator.getAsString();

            //BUILDING THE HTTP REQUEST
            Http http = new Http();
            HttpRequest request = new HttpRequest();
            request.setEndpoint('THE HTTP ENDPOINT FOR WHERE THE DATA IS SENT');
            request.setHeader('Authorization', authorizationHeader);
            request.setMethod('POST');
            request.setbody(jsonBody);
            HttpResponse response = new HttpResponse();

            try {
            response = http.send(request);
            } catch(System.CalloutException e) {
            System.debug('Error e happened: '+ e);
            System.debug(request.toString());
            }
    }

}
```

(Kuva 9. Kuvankaappaus Apex-luokan luurankorakenteesta, jossa demonstroidaan HTTP-pyynnön tekemistä ulkoiseen lähteeseen.)

4.1.3 Apex vahvuudet & heikkoudet

Edellä esitetyn demonstraation perusteella, pystymme todentamaan, että Apex on toimiva tapa integraatiolle Salesforcen ja kolmannen osapuolen välillä. Suurimpina vahvuuksina Apexilla on sen räätälöitävyys ja siinä on hyvin vähän rajoituksia. Apexissa suurin osa vastuusta siirretään kehittäjälle, joka määrittää kirjoitettavan koodin kautta tiedonkulun Salesforcen sisällä ja sen, miten yhteys ulkoiseen järjestelmään järjestetään. Se on myös kustannustehokas yritykselle, sillä ainoa kustannus siinä on Salesforcen käyttö itsessään ja kehitysryhmän kulut.

Räätälöitävän koodin ongelmana on se, että sen toimittaminen asiakkaalle on monimutkaista, sillä se vaatii yhteyshenkilön, joka osaa implementoida sen Salesforceen. Tämä ei ole ongelma

integraatioprojekteissa yksittäiseen Salesforce-alustaan, sillä niissä usein itse kehittäjä tulee olemaan se, joka suorittaa koodin implementoinnin alustaan ja tekee tarvittavan kehityksen, sekä asennuksen varmistaakseen sen toimivuuden. Tilanteissa, jossa tarkoituksena on toteuttaa integraatio useamman asiakkaan Salesforce-alustoihin voi tämä kuitenkin olla ongelma, koska useilla yrityksillä ongelmana voi olla, että heiltä puuttuu riittävä tekninen osaaminen koodin implementointiin heidän omaan alustaansa. Tämä voi olla työläs prosessi, sillä se voi johtaa ulkoisen konsultaation palkkaamiseen. Tämä vaikuttaa lopputuotteen skaalautuvuuteen tilanteessa, jossa tuotteen pitäisi olla tarjolla useammalle asiakkaalle ja voi mahdollisesti nostaa käytettyä budjettia teknisen tuen kautta ja täten vaikuttaa projektin edellä mainittuun kustannusrajoitteeseen. Syy, miksi tämä on ongelma, että vaikka itsessään koodi olisi jaettavissa useammalle asiakkaalle AppExchangen kautta, salaisuuksien ja avaimien tallettaminen asiakkaan vastuulle.

Toimeksiantajan tapauksessa integraation ideana on, että lopputuote olisi mahdollisimman helposti paketoitava ja toimitettava useammalle asiakkaalle. Salesforcen AppExchangessa tämän ongelman voi ratkoa, tekemällä olemassa olevasta koodista managed packagen, joka pystytään myymään asiakkaalle ja asiakas voi ladata olemassa olevat ohjelmistokomponentit yhdessä muiden asiakkaiden kanssa. Asiakkaalla ei itsellä tulisi olemaan pääsyä kirjoitettuun koodiin managed packagessa. Ongelmana tässä ratkaisussa tietoturvan kannalta muodostuu se, että sillä Apex-koodissa oleva salaisuus (oli se sitten avain tai sertifikaatti) joutuisi olemaan jokaisella asiakkaalla sama, koska muuten asiakas joutuisi itse konfiguroimaan oman salaisuutensa. Toimeksiantajan mukaan tämä on huonoa koodikäytäntöä ja täten sitä pitäisi välttää. Se nostaisi projektin laajuutta eli skaalaa huomattavasti, sillä 10Duke joutuisi rakentamaan jonkin ratkaisun, jonka avulla jakaa näitä salaisuuksia niin, että asiakkaan olisi mahdollisimman helppo konfiguroida nämä itselleen.

Täten multitenant ratkaisu voisi osoittautua haasteelliseksi. Apex on kuitenkin hyödyllinen erityisesti tilanteissa, joissa tarkoitus on toteuttaa integraatio yksittäisille asiakkaille, sen halvan kustannuksen takia ja kuinka tiivis resursseiltaan se on toteutuksessa. Erityisesti integraatioissa, joissa kustomoitavuus on etusijalla, on Apex erittäin tehokas, sillä käyttäen Salesforce komponentteja, kenttiä ja custom-objekteja pystytään asiakkaalle luomaan varsin yksilöitävä kokemus.

4.2 pub/sub toteutus

Toisin kuin muissa integraatiototeutuksissa pub/subin tapauksessa itse integraation käytännön toteutus tulee pohjautumaan mahdollisen käytännön toteutuksen tiedonkulun kuvaukseen. Syynä tähän, että kyse on kustomoitavissa olevasta koodista, joten alustavan käytännön toteutuksen tekeminen ei palvelisi opinnäytetyön tarkoituksia, sillä todellinen toteutus riippuisi pelkästään kehittäjän omasta tavasta tehdä toteutus.

4.2.1 pub/sub API työkalut

Toisin kuin muissa integraatiototeutuksissa pub/subin tapauksessa itse integraation käytännön toteutus tulee pohjautumaan mahdollisen käytännön toteutuksen tiedonkulun kuvaukseen. Syynä tähän, että kyse on kustomoitavissa olevasta koodista, joten alustavan käytännön toteutuksen tekeminen ei palvelisi opinnäytetyön tarkoituksia, sillä todellinen toteutus riippuisi pelkästään kehittäjän omasta tavasta tehdä toteutus.

4.2.2 pub/sub demonstraatio

Pub/sub API toteutus on pääosittain koodia, joten sen käytännön toteutus tulee perustumaan pitkälti siitä minkälaista teknologiaa ja ohjelmointikieltä kehittäjä päättää käyttää sen toteutukseen. Koodin sisältöön tultaisiin tekemään verkkoprotokollapyyntö tilaamaan tapahtumia Salesforcen pub/sub API:sta, jonka jälkeen itse koodi kuuntelisi näitä tapahtumia rajapinnan kautta ja vastaanottaisi niitä. Tämän jälkeen prosessointilogiikassa tieto käsiteltäisiin jälki jalostusta varten. Järjestelmä voi tehdä HTTPS- tai TCP-protokolla pohjaisen pyynnön Salesforcen osoitteeseen, käyttäen käyttäjän omia Salesforce tunnistautumistietoja. Tämän jälkeen event bus käsittelee pyynnön ja alkaa lähettämään tapahtumia käsittelijälle, johon kehittäjä pystyy lisäämään logiikkaa tapahtumien käsittelylle oman koodinsa kautta.

Opinnäytetyössä ei käytännön demonstraatiota pub/sub API:sta tulla näyttämään sillä ratkaisu on kokonaan koodipohjainen ja riippuen yrityksestä voi näyttää hyvinkin erilaiselta riippuen yrityksen tarpeista. Koodipohjainen ratkaisu myös riippuu siitä millä ohjelmointikielillä se toteutetaan.

4.2.3 pub/sub API:n vahvuudet ja heikkoudet

Pub/sub tarjoaa erittäin joustavan ja skaalautuvan ratkaisun, joka antaa ohjelmoijalle mahdollisimman suuren vapauden toteuttaa ratkaisu heidän suosimalla tavalla, erityisesti jos sitä verrataan Apexiin. Samalla, se kuitenkin lisää vastuuta tuotteen laadukkaasta toteuttamisesta, sekä lisää kehittäjän ohjelmointityön määrää. Sisäisen prosessointilogiikan lisäksi, kehittäjä joutuisi pitämään huolen yhteyden muodostamisesta pub/sub API:n ja sen suojauksesta. Erityisesti tilanteessa, jossa toteutettavan ratkaisun pitäisi olla skaalattavissa, kehittäjän pitäisi pystyä rakentamaan toiminnallisuus tiedon saamiselle usean asiakkaan Salesforcen sisältä, joka vaatisi useamman eri yhteyden eri Salesforceen. Myös tapahtumien vastaanotto ja huolehtiminen, että ne tulisivat aikajärjestyksessä voi olla haasteellista.

Tämä lisäisi projektin kompleksisuutta, sen nostetun skaalan kautta. Tämän takia myös projektin aikarajoite tulisi nousemaan, kuten myös budjettirajoite, sillä projektista aiheutuvat ohjelmointikulut

tulisivat nousemaan. Projektin aikataulu voisi täten venyä ja sen skaala nousisi huomattavasti ja tämä voisi aiheuttaa ongelmia projektin suorittamiselle kokonaisuudessaan.

4.3 Amazon AppFlow toteutus

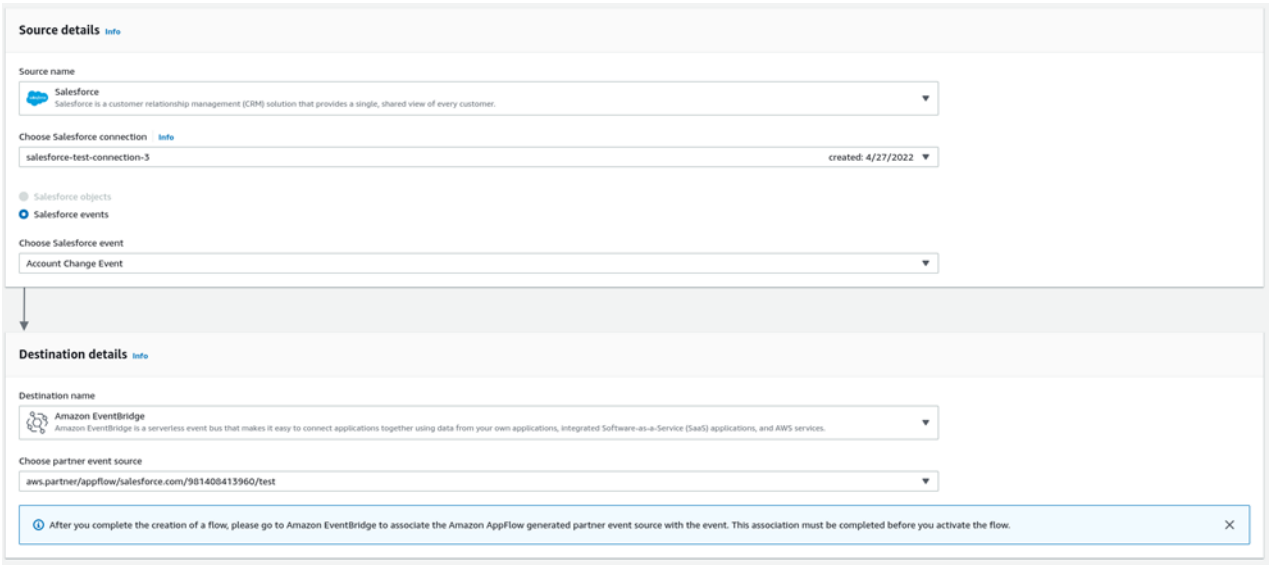
Viimeisenä käymme läpi Amazon AppFlowia käyttävän toteutuksen. Amazon AppFlowin toteutuksessa tulemme käsittelemään kuinka saadaan pub/sub yhteys muodostettua käyttämällä AppFlowia Salesforcen kanssa ja kuinka tieto reititetään EventBridgeä käyttämällä Amazonin S3-tallennustilaan.

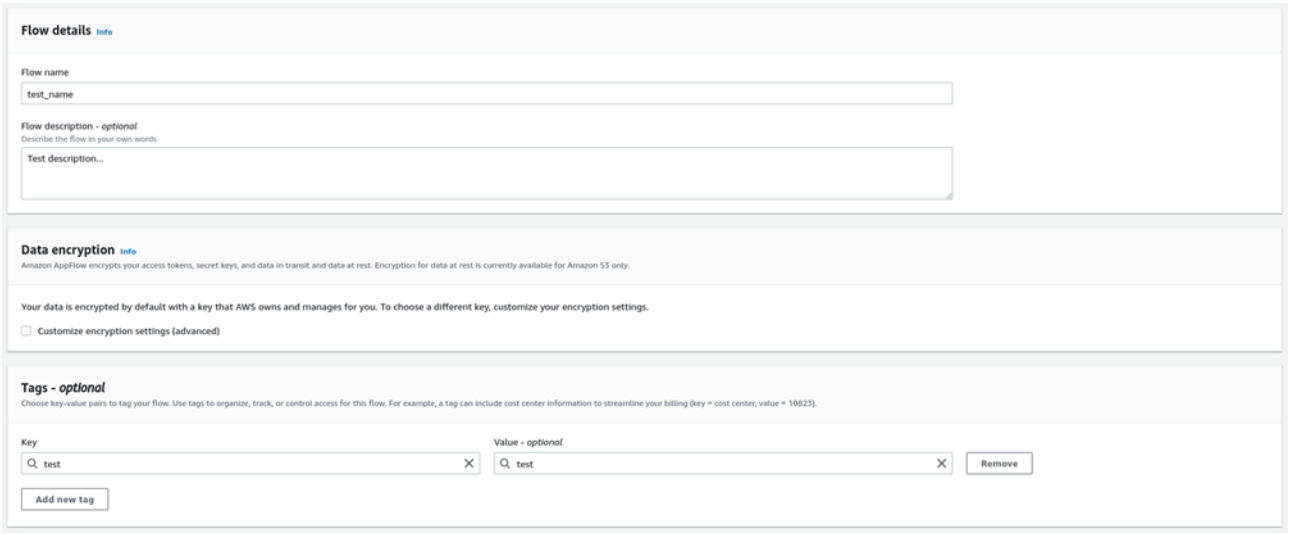
4.3.1 Amazon AppFlow työkalut

Amazon AppFlow integraatioissa, suurimpina työkaluina toimii Amazon Web Services eli AWS:n pilvialustan palvelut AppFlow, EventBridge, developer konsoli ja S3-tallennuspalvelu. AppFlow, tiedon vastaanottamista varten, EventBridge sen reitittämistä varten ja S3 datan tallettamista varten. Developer konsolia käytetään tiedon monitorointiin. S3:sta organisaatio pystyy vastaanottamaan dataa sisäisen prosessointilogiikkansa kautta. Näitä palveluja käyttääkseen on kehittäjällä oltava kuitenkin AWS-tili. Itsessään toteutus käyttäen Amazon AppFlowta tulisi näyttämään seuraavalta.

4.3.2 Amazon AppFlow demonstraatio

Yhteys Amazon AppFlowiin voidaan luoda käyttäen AWS:n sisäistä konsolia tai koodillisesti Amazon AppFlow API:n kautta. Tässä tutkielmassa kuitenkin tulemme demonstroimaan AppFlowin luontia AWS:n konsolia käyttäen. Muodostaakseen yhteyden graafista käyttöliittymää käyttäen joudumme luomaan sen menemällä AWS-konsolin AppFlow osioon, josta voimme edetä tietovirran luontiin. Tietoja mitä se tarvitsee, on Salesforcesta source- ja destination name nimiset osoitteet (kuva 10), yhdessä yhteyden nimen kanssa (kuva 11). Sourcella tarkoitetaan Salesforcen tarkkaa osoitetta ja destinationilla paikkaa mihin data sijoitetaan AWS:n sisällä (toimeksiannon tapauksessa tapahtumanvälitys). Tämän lisäksi käyttäjä voi vaihtoehtoisesti lisätä leimoja (tags) osana yhteyttä, jotta ne ovat helpompia erottaa toisista floweista (kuva 11). Käyttäjä pystyy myös luomaan itse hallinnoimansa avaimen, AWS:n hallinnoiman vakioavaimen sijaan.





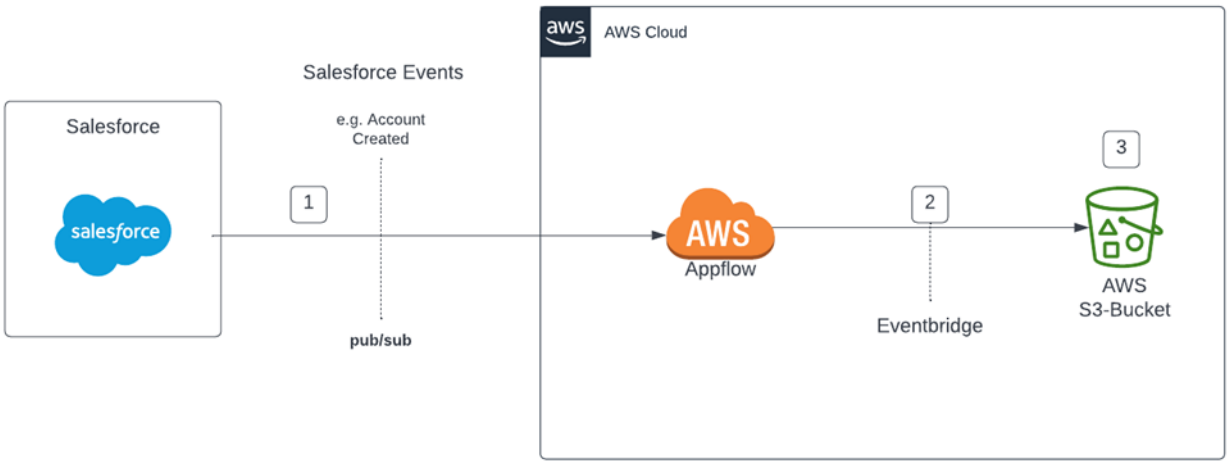
(Kuva 10 & 11. Kuvankaappaus AWS-AppFlown konsolista, jossa täytetään AppFlowlle tarpeelliset tiedot yhteyden luomiseen Salesforcen kanssa.)

Tämän jälkeen asiakas voi itse valita haluaako hän käyttää yhteyden luomiseen normaalia yhteyttä, julkisen internetin kautta vai Amazonin sisäisen verkon kautta kulkevaa PrivateLink palvelua. Nämä asetukset päätettyään, ja navigoiden konsolin kautta loppuun yhteys on luotu. Käyttäen EventBridge palvelua, integraatio AppFlowin avulla jatkuisi seuraavalla tavalla.

EventBridgeä käytetään EventBridgen konsolin kautta. Asetuksia mitä siihen kuuluu asettaa, on tapahtuman lähde (event source), joka tässä tapauksessa vastaa Amazon AppFlowia. Tapahtuman lähde liitetään yhteen event busiin, jonka jälkeen näiden välinen tietovirta voidaan aktivoida välittämään tietoa.

Jotta saisimme tiedon kulkemaan sen lopulliseen kohteeseen (toimeksiannon tapauksessa S3:seen), pitää se reitittää erillisen palvelun kautta sinne. Tähän AWS:ssä ratkaisuna toimii event rules eli tapahtuma säännöt, jotka käyvät läpi tapahtumista tulevan tiedon ja sen tiedon perusteella päättävät mihin tieto ohjataan. Tapahtuma säännöille annetaan oma identifioiva nimi ja sille voidaan määritellä kustomoidut tapahtuma mallit (event pattern), jonka avulla määritellään sääntöjen mukaiset rakenteet, joita EventBridgeen saapuvien tapahtumien on noudatettava.

Toimeksiannon tapauksessa tarkoituksena on vastaanottaa kaikkia tapahtumatyyppejä, sillä tapahtumatyypit on erikseen jo muutenkin jaoteltu. Mallinnuksessa käytetään “pre-defined pattern by services”-vaihtoehtoa ja service providerina toimii “All Events”. Tämän jälkeen säännölle valitaan siihen liittyvä event bus, jonka jälkeen säännölle valitaan sen lopullinen kohde. Toimeksiannon tapauksessa S3:seen tulevan tapahtuma datan kulkua demonstroidaan kuvassa 12 ja S3:seen muodostunutta payloadia demonstroidaan kuvassa 13).



(Kuva 12: Tiedon kulku Salesforcesta S3-buckettiin.)

(Kuva 13. Kuvankaappaus S3:seen saapuneesta JSON-objektista.)

4.3.3 Appflow vahvuudet & heikkoudet

AppFlow integraatiossa on omat selkeät vahvuudet ja heikkoutensa. Vahvuuksina siihen kuuluu, että se vaatii erittäin vähän koodausta. AppFlowin avulla kehittäjät pystyvät vähentämään ohjelmoinnista aiheutuvaa aikaa, vähentäen projektin suoritusaikaa ja täten aikarajoitteita, kuten myös ohjelmoinnista aiheutuvia kuluja. AppFlowin avulla kehittäjät pystyvät kehittämään skaalautuvia integraatioita Salesforcen kanssa, käyttäen AWS:n omaa API:a AppFlow-yhteyksien luomiseen, joten se sopii tilanteisiin, joissa integraation on toimittava usean asiakkaan kanssa.

AppFlowin heikkous kuitenkin on, että se ei ole ratkaisuna joustava. Eli mikäli se ei ratkaise tarkalleen tuotteen tarvetta on sitä vaikea ottaa käyttöön. Erityisesti tilanteissa, joissa sinne tuleva data on muuttuvaa eli sillä ei ole vakio kenttiä. Tämä tekee reitittämisen vaikeaksi käyttäen event ruleseja. On myös otettava huomioon, että Amazon AppFlow on maksullinen kolmannen osapuolen palvelu ja täten siitä aiheutuvat kustannukset on otettava huomioon integraatiossa. Erityisesti tilanteessa, joissa yrityksellä on valtava määrä tapahtumia useammalta asiakkaalta tulossa AppFlowin kautta. Jokainen ajettu AppFlow kierros maksaa 0.01 USA:n dollaria ja EventBridgen käyttö kustantaa 1.00 USA:n dollaria per miljoona tapahtumaa. Datan lisääminen S3:seen ja sen tallettaminen myös nostaa projektin kustannuksia. Viimeinen ongelma AppFlowin ratkaisussa on myös, että Salesforcen tapauksessa yksi AppFlow pystyy tukemaan vain yhtä Salesforcen tapahtuma tyyppiä. Mikäli halutaan tehdä skaalautuva ratkaisu, jossa pystytään käsittelemään useita tapahtuma tyyppejä, joudutaan yhdelle asiakkaalle luomaan monta erillistä AppFlowia huolehtimaan eri

tapahtumien tyypeistä. Onneksi kuitenkin itsessään AppFlowin luominen on ilmaista, joten siitä ei aiheudu lisäkustannuksia.

5 Analyysi

Toimeksiannossa osana tutkimusta kuului se, että yritys pystyisi tunnistamaan tavan integraatiolle, jossa olisi paras tuki useamman asiakkaan kytkemiseen osaksi integraatioratkaisua. Jotta tämä kuitenkin pystyttäisiin toteuttamaan, piti eri integraatio metodien vahvuuksia ja heikkouksia verrata keskenään. Tarkoituksena on tehdä johtopäätöksiä siitä, mikä olisi projektihallinnallisesti järkevin tapa toteuttaa integraatio, kun ajatellaan integraatiota projektin aika-, budjetti- ja laajuus rajoitteiden kautta.

5.1 Aika

Ajallisesti yksi suurimmista kysymyksistä integraatiotapaa pohtiessa on, kuinka paljon organisaatiolla on varaa käyttää aikaa kehittämiseen yhdessä organisaation olevien työvoima resurssien kanssa. Kuinka paljon heillä on työvoimaa kehitysvaiheesta suoriutumiseen ja kuinka paljon osaamista projektissa työskentelevillä työntekijöillä on. Tämän vuoksi erityisesti pienemmän skaalan ratkaisuissa, nopeuttaakseen kehitystyötä, on parempi toimeksiantajan mukaan, jos organisaatio pyrkii välttämään räätälöidyn koodin tekemistä mahdollisimman paljon. Tämän vuoksi toimeksiantajan tapauksessa, jossa Salesforce integraatioratkaisun pitäisi toimia useammalle asiakkaalle, mitä todennäköisemmin tulisi pub/sub API:n käyttäminen olemaan liian pitkäkestoinen. Kehitysryhmä joutuisi kirjoittamaan valtavan määrän koodia saadakseen toimivan ja turvallisen ratkaisun luotua. Tätä pystyy tietysti helpottamaan suurempi kehitysryhmä, lisäten organisaation inhimillisiä resursseja saada integraatioratkaisu valmiiksi ajoissa. Toimeksiantajan tapauksessa kuitenkin projektissa kehitysryhmä on pieni, ja täten pub/sub API:in kehitys tulisi ajallisesti olemaan liian suuri, sillä integraation joutuisi kehittämään räätälöitynä koodina.

Pub/sub API:sta eroten Apex ja AppFlow ovat ajallisesti tehokkaampia, sillä molemmista ratkaisuista löytyy valmiita ratkaisuja tiedonvälitykseen Salesforcen läpi. Apexin tapauksessa tietoa pystytään kuljettamaan suoraan Apex-koodiin, käyttämällä Salesforcen sisäistä Process Builderia tai floweja kuuntelemaan tapahtumia, joka aktivoi platform eventin ja Apex-triggerin, josta tieto pystytään välittämään koodin kautta vastaanottajalle. Se ei vaadi suurta määrää koodia ja koodi olisi helposti uudelleen tuotettavissa useammalle asiakkaalle samannimiseen Apex-luokkaan, joten oikean ratkaisun löydyttyä olisi se ajallisesti tehokas ratkaisu.

Sama asia AppFlowin kanssa, joka ajallisesti on vielä nopeampi, sillä se ei tarvitse lainkaan koodia organisaation oman järjestelmän ulkopuolelle, vaan se voidaan konfiguroida täysin yrityksen järjestelmän sisällä. Erityisesti yksittäisessä tilanteessa, jossa organisaation ei tarvitse muodostaa yhteyttä useamman asiakkaan Salesforceen se on erittäin suoraviivainen. Koska yhteyttä AppFlowiin

ei tarvitse koodata ollenkaan, voidaan se suorittaa AWS:n konsolin kautta, käyttäen API-yhteyksiä. Ainoana vaatimuksena on, että kehittäjällä on AWS-tunnukset.

5.2 Budjetti

Budjetilla on suuri osuus integraatiomallin valitsemisessa, sillä usein organisaation toimintaa hallitsee budjettirajoitteet, joten on erittäin tärkeää ottaa ne huomioon, kun päättää mihin suuntaan projektia viedään. Se on myös tärkeä asia sen takia, koska sillä on suora yhteys siihen, onko projektin suorittaminen ylipäätään kannattavaa, ja voiko yritys rahallisesti hyötyä siitä. Kaikissa kolmessa läpikäydyssä integraatiotavassa on omat budjetilliset riskinsä siitä, mikä aiheuttaa kuluja ja mikä ei.

Apexin ja pub/sub:in asemassa suurimmat kysymykset johtuvat kehittäjien työstä aiheutuvista kustannuksista ja kuluista. Siihen vaikuttavat muut asiat, kuten esimerkiksi kehitysryhmä koko yhdessä projektin skaalan ja aikataulun kanssa. Sillä, mitä laajempi projekti sitä enemmän aikaa siinä tulee kestämään, ja tämän kautta kustannukset kehittäjien palkoista tulevat nousemaan.

AppFlow erityisesti eroaa kahdesta edellä mainitussa tässä suhteessa, sillä erityisesti integraatio projekteissa, missä tarkoituksena on vain integroida yhden asiakkaan Salesforce yrityksen sisäiseen koneistoon, toimii AppFlow niin, että se säästää aikaa ja täten rahaa. Sillä kehittäjien ei tarvitse kehittää ratkaisua, vaan he voivat käyttää AWS:n AppFlow-konsolia yhteyden luomiseen. Tilanteessa taas missä ratkaisun on oltava skaalattava useammalle asiakkaalle, voivat kehittäjät myös luoda yhteyden käyttämällä AWS:n API-yhteyksiä.

AppFlowin tapauksessa suurin osa kuluista syntyy AWS:n käytöstä aiheutuvista kuluista eli AppFlowin instanssien ajosta. Näiden lisäksi AWS veloittaa sinne tallennetun datan määrästä (toimeksiannon tapauksessa S3-käytöstä) ja event busin käytöstä aiheutuvista kustannuksista, jotka kaikki skaalautuvat niiden kokonaismäärän mukaisesti ja voivat täten vaihdella suuresti. Tähän vaikuttaa asiakkaan Salesforcen käytön määrä, ja se kuinka monta asiakasta organisaatiolla on käyttämässä integraatioratkaisua. Organisaation tulisi täten laskea tarkasti, kuinka monta asiakasta heillä tulisi olemaan, ja kuinka paljon he palvelusta laskuttavat projektin aikana, että integraation toteutus pystyisi kattamaan itsensä.

5.3 Laajuus

Apexin käytön kautta projektin laajuus tulisi laajenemaan liikaa pienen kehitysryhmän tuettavaksi. Täten projektin laajuus ei vastaisi yrityksen käytössä olevia resursseja. Jotta integraatio tulisi olemaan järkevä, ratkaisun kehityksen lisäksi joutuisi kehitysryhmä pitämään huolen sen paketoimisesta asiakkaalle.

Apexissa paketointi asiakkaalle voidaan hoitaa käyttämällä Salesforcen omaa Salesforce package-paketointi tapaa, käyttäen managed packagea, joka myöhemmin pystytään välittämään asiakkaille, käyttäjille ja organisaatioille, josta package pystytään välittämään Salesforcen App Exchangen kautta käyttöön. Koodi ei itsessään tulisi näkymään asiakkaalle, joten tämän kautta projektin laajuus tulisi pysymään hallittavissa.

Toimeksiannon tapauksessa kuitenkin ongelmaksi osoittautui tietoturva, sillä mitään erillistä salausmetodia Apexissa ei esitetty, jonka avulla salaisuuksia erityisesti RSA-256 kaltaisia avainpareja pystyttäisiin salaamaan ilman, että niitä laitettaisiin joko koodin sisälle tai custom fieldin tai settingin sisälle. Toimeksiantajan mukaan erityisesti avaimen tallentaminen itse koodin sisälle olisi erittäin huono ratkaisu, kuten myös sen tallettaminen custom fieldin tai protected custom settingin sisälle. Syy minkä takia avaimen tai muun salaisuuden tallettaminen itse koodiin on huono asia, on koska se ei ole ylläpidettävä kokonaisuus ja jos ulkopuolinen pääsee siihen käsiksi, jouduttaisiin vähintäänkin kyseinen avain muuttamaan, jokaiselle asiakkaalle tietoturva syiden vuoksi. Sama pätee myös muihin vaihtoehtoihin. Jokaisella asiakkaalla olisi sama avain ja täten mikäli se paljastuisi, integraatiopalvelu ei olisi toimimaan yhdelläkään asiakkaalla. Täten itse avaimien ja muiden salaisuuksien tallettaminen tulisi vaatimaan paljon koodausta tai muuta konfigurointia, joista kehittäjien täytyisi huolehtia, nostaen projektin laajuutta. Key managementiin pystytään tallentamaan ainoastaan sertifikaatteja.

Toiseksi toimeksiantajan mukaan integraatioratkaisun toimittaminen asiakkaalle tulisi olla mahdollisimman suoraviivainen toteutukseltaan ja tämän takia Apexin toimittaminen AppExchangen kautta osoittautui ongelmaksi, sillä integroidakseen palvelun asiakas joutuisi itse konfiguroimaan avaimen tai salaisuuden omaan Salesforce-ympäristöönsä, lisäten projektin laajuutta.

Pub/sub API puolestaan on konfiguroitavaa koodia, jossa valta on pääosin kehittäjän omalla vastuulla. Täten projektin laajuus tulee riippumaan vahvasti siitä, mitä vaatimuksia ja ominaisuuksia rakennettavalla sovelluksella tulee olemaan. Toimeksiannon kaltaisessa tapauksessa, yksi ongelma erityisesti mikä aiheutuisi pub/sub API:n käytöstä on se, että koska kyse on koodista, pitäisi palvelun kestävyydestä, ja luotettavuudesta pitää huolta. Se erityisesti nostaisi projektin laajuutta, jotta se pystyisi kestämään mahdolliset palvelukatkot, mikäli esimerkiksi palvelin kaatuisi organisaatiossa ja täten Salesforcen sisäinen tapahtumadata katoaisi. Erityisesti sillä tapahtumat säilyvät Salesforcen event busin sisällä vain 24 tuntia vakiotapahtumien kohdalla ja 72 tuntia Change Data Capture-tapahtumien kohdalla, joten järjestelmän pitäisi pystyä toipumaan siinä ajassa.

AppFlowin tapauksessa itse projektin skaalaan vaikuttaa eniten siihen kuuluvien AppFlowien luonti. Vaatimukset siitä mitä eri tapahtumia käyttäjä haluaa vastaanottaa vaikuttavat siihen, kuinka monta erilaista AppFlowia tarvitaan ratkaisuun. AppFlow voidaan soveltaa vain yhteen

Salesforcessa olevaan tapahtumatyyppiin kerrallaan, joka voi nostaa laajuutta riippuen millaisella ratkaisulla he päättävät AppFlowien luonnin toteuttaa. Yksittäisen flowin tapauksessa helpoin ratkaisu todennäköisesti olisi manuaalisesti luoda flow AWS:n konsolin kautta, kun taas tapauksessa missä tapahtumatyyppejä on useita, voi projektin skaala laajentua. Mikäli ratkaisussaan käyttää EventBridgeä on siinä käytettävät event rulesit huomioitava myös määrittelemään minkälaiset tapahtumat se hyväksyy jatkojalostukseen. Toimeksiannon tapauksessa, EventBridgen palvelua käytetään tapahtumien lähettämiseen Amazon S3-talletustilaan.

Yksi suuri vahvuus AppFlowin käytön kannalta edelliseen kahteen integraatiototeutukseen verrattuna on se, että se on valmiina oleva ratkaisu, jota kehittäjän ei itse tarvitse koodata. Se alentaa projektin laajuutta sen käytännön toteutuksen kannalta huomattavasti. Laajuus tulee myös vähentymään tietoturvallisista syistä, sillä koska AppFlow on AWS:n tarjoama palvelu, itse tiedonvälitys kanavan turvallisuudesta tulee vastaamaan AWS, jossa heillä on käytössään aikaisemmin mainittu PrivateLink-palvelu. Tämä on huomattava etu kahteen edelliseen integraatiototeutukseen verrattuna, sillä näissä toteutuksissa suurin osa tietoturvan hoitamisen vastuusta sisältyy itse kehittäjille ja sen toteutus voi laajentaa projektin skaalaa huomattavasti. Myös ratkaisun toimintavarmuus on huomattavasti parempi, sillä AWS tulee vastaamaan mahdollisista palvelu ja järjestelmä katkoista ja täten riski mahdollisesta datan häviämisestä vähentyy, sillä organisaatiolla tulisi olemaan kolmas osapuoli, jonka vastuulla olisi pitää huoli tapahtumien luotettavasta tallettamisesta.

5.4 Toimeksiannon tilanne

Nämä asiat läpikäytyämme voimme ryhtyä käsittelemään toimeksiantoa ja sitä mihin integraatiototeutukseen päädyimme yrityksen tulevan projektin kannalta. Erityisesti suurimpina tekijöinä tulevat toimimaan projektiin liittyvät rajoitukset ja vaatimukset liittyen siihen, mitä tulevan integraatiototeutuksen pitää pystyä tekemään. Toimeksiannon vaatimuksina oli, että asiakkaiden Salesforcien sisäistä dataa pystyttäisiin siirtämään Salesforcesta toimeksiantajan sisäiseen järjestelmään multitenant pohjaisessa ratkaisussa, turvallisesti ja, että tapahtumista tuleva data pysyisi eheänä, kuten myös ajallisesti järjestyksessä toisiinsa nähden. Ratkaisun olisi myös hyvä olla helposti konfiguroitava asiakkaalle niin, että asiakkaan ei itse tarvitsisi konfiguroida ratkaisua itselleen ja, että integraatioratkaisun implementointi heidän järjestelmäänsä olisi mahdollisimman yksinkertaista. Näiden vaatimusten pohjalta, kunkin edellä esitetyn integraatiototeutuksen pitäisi pystyä vastaamaan näihin vaatimuksiin ja täyttämään nämä osana ratkaisuaan.

Apexin tapauksessa, ratkaisu on toteutettava luomalla kustomoitua koodia ja komponentteja itse Salesforcen alustaan, jotka jaetaan asiakkaalle AppExchangen kautta, tekemällä package luodusta koodista ja konfiguraatiosta. AppExchangen ratkaisun ideana on, että kehittäjä pystyy jakamaan sovelluksiaan suoraan asiakkaille, joten Apexissa luotu koodi on helposti monistettavissa ja

jaettavissa asiakkaille. Apex-koodin päätepiste tulisi olemaan jokaisessa yksittäisessä ratkaisussa sama. Jopa tilanteissa, joissa päätepiste olisi oltava yksilöitävä, pystytään tieto lähettämään parametrina pyynnön yhteydessä päätepisteeseen. Tietoturvaa Apexissa ei kuitenkaan pystytä täysin ratkaisemaan pelkän koodin avulla vaan asiakkaan olisi määriteltävä salaisuudet tai avaimet itse. Toimeksiantajan mukaan, vaikka avain olisi talletettu managed packageen ja täten siihen ei pääse muu kuin sitä käyttävä Apex-koodi, se olisi riskialtista, että asiakkaat jakaisivat palvelussa saman avaimen tietoturvan vuoksi.

Pub/sub API:n tapauksessa, ratkaisu on samalla tavalla ratkaistavissa, sillä kyseessä on kustomoitua koodia, voidaan itse yrityksen sisäisessä logiikassa pitää huoli pub/sub-yhteyden muodostamisesta Salesforcen kanssa. Vaikka kyseessä olisikin multitenant pohjaisen ratkaisun luonnista, voidaan useampi yhteys luoda kirjoittamalla logiikka, jossa asiakkaan kirjaamilla Salesforce tiedoilla, voi toimeksiantaja muodostaa yhteyden pub/sub API:iin. Toimeksiantajan oman sisäisen järjestelmän rakenne olisi suunniteltava niin, että sen olisi mahdollista käsitellä useampaa pub/sub API-yhteyttä. Ongelma kuitenkin pub/sub API:n tapauksessa on toimeksiantajan mukaan, se että koska se on räätälöitävää koodia, jouduttaisiin koodia kirjoittamaan kohtuuttoman paljon pienellä kehitysryhmällä, jossa on vain kaksi kehittäjää.

AppFlowin tapauksessa, palvelu olisi toimeksiantajan tapauksessa helppo tehdä palvelu tietoturvalliseksi. Toimeksiantajalle on ollut AWS yrityksen käytössä jo aikaisemmin ja täten teknologiallisesti se olisi helposti implementoida Salesforceen. Asiakkaalla olisi turvallista käyttää PrivateLink-palvelua laittamalla yhteydet kulkemaan AWS:n yksityisen verkon kautta ja palvelun toimintavarmuus parantuisi. Vastuu tultaisiin siirtämään kolmannelle osapuolelle ja täten vaikka itsessään toimeksiantajan omassa palvelussa olisi käyttökatko, tapahtumien tieto ei katoaisi. Toimeksiantajan tapauksessa, ongelma AppFlowin kanssa on kuitenkin problematiikka, että mikäli palvelun halutaan olevan skaalattavissa useammalle asiakkaalle, jouduttaisiin luomaan useampi flow per asiakas vastaamaan eri tapahtumatyyppejä, koska AppFlowissa yksi tapahtumatyyppi Salesforcessa vastaa yhtä flowia. Toimeksiantajan olisi myös laskettava floweista aiheutuvat kustannukset tarkasti.

5.5 Business impact analyysi & kustannusvaikutukset

Jotta saisimme paremman kuvan eri integraatiototeutuksista muodostuvista riskeistä ja kuinka ne voivat haitata projektin onnistumista, kannattaa meidän tehdä business impact analyysi aiheelle. Business impact analyysin avulla pystymme tekemään laskuja mahdollisista riskeistä ja arvioimaan kuinka mahdollinen integraatioprojekti vaikeutuisi siitä aiheutuvien keskeytysten takia.

Alustavana Business impact analyysinä tulemme arvioimaan mahdollisia riskejä mitä integraatio projektissa voisi tapahtua ja heijastamme niitä kuhunkin läpikäytyyn integraatio toteutukseen ja kuinka ne voisivat heikentää lopputuotteen mahdollista tulosta, selittäen ne syyt mistä tämä voisi johtua.

Analyysissä tulemme käyttämään mallia missä riskien arvioiminen tapahtuu tarkastelemalla seuraavia kategorioita (aika, laajuus, budjetti, toimintavarmuus, tietoturva ja helppokäyttöisyys), ja niistä mahdollisesti koituvia ongelmia toimimaan perusteluina kullekin edellä esitetylle integraatiototeutukselle. Arvioiden arvosanat kuvaavat riskiä mitä yksittäinen integraatiototeutus tulee mahdollisesti aiheuttamaan kussakin kategoriassa ja niiden arviointi kulkee asteikolla 1–3, jossa 1. vastaa pientä riskiä, 2. keskisuurta riskiä ja 3. suurta riskiä.

Kyseinen business impact analyysi ei tule arvioimaan häiriöstä aiheutuvaa toipumisaikaa, tai taloudellista vahinkoa, sillä kyseessä on projektia alustava business impact analyysi ja täten niitä ei vielä tässä vaiheessa projektia pystytä arvioimaan. Tarkoituksena on antaa arvio kunkin integraatiototeutuksen riskeistä ja kuinka todennäköisiä ne ovat:

Taulukko 1. Business impact analyysi. Kuvaa integraatiototeutuksista aiheutuvien riskien mahdollisuutta.

Kriteeri/teknologia	**Apex**	**pub/sub API**	**AppFlow**
Aika. Projektin aikamääreet voivat venyä ja mahdollisesti myöhästyä.	2. Keskisuuri. Apex on kustomoitavaa koodia, joten aika sen luomiseen tulee kestämään enemmän kuin valmiina olevassa ratkaisussa.	3. Suuri. Pub/sub API:n ratkaisu perustuu täysin kustomoidun koodin varaan integraatiojärjestelmän päädyssä, joten riippuen tarpeista se vaatii enemmän koodausta.	1. Pieni. AppFlow on valmiina oleva ratkaisu, joka voidaan rakentaa ilman koodia, säästäen projektin aikamääreistä paljon.
Laajuus. Projektin laajuus ja tehtävät tekniset ratkaisut voivat paisua yli projektin oman kokonaisuuden ulkopuolelle.	2. Keskisuuri. Apexissa on kustomoitua koodia, joka vaatii konfigurointia, testausta ja implementointia, mutta	3. Suuri. Sillä kyseessä on täysin kustomoitava ratkaisu, joka rakentuu koodin varaan, tulee ratkaisun laajuus olemaan	1. Pieni. AppFlow on valmiina oleva ratkaisu, jota pystytään hyödyntämään säästäen kehitystiimiltä turhaa koodausta ja

	yhteydenluonti Salesforcen ja ulkoisen järjestelmän välillä on helppo toteuttaa.	suurempi kuin muissa. Erityisesti jos otetaan huomioon payloadien ja jonojen kronologinen käsittely.	siinä on määritellyt rajoitteet tiedonsiirtoa kohden, joiden varassa integraatio on suunniteltava
Budjetti. Projektin sisäisen lopputuotteen budjettirajoitteet tulisivat ylittymään.	1.Pieni. Suurimmassa osassa tapauksissa Apexin hinnoittelu pohjautuu asiakkaan käytössä olevaan Salesforce-pakettiin, joka määrittää, kuinka paljon heille integraatio maksaa.	1.Pieni. Suurimmassa osassa tapauksista Apexin hinnoittelu pohjautuu asiakkaan käytössä olevaan Salesforce-pakettiin, joka määrittää, kuinka paljon heille integraatio maksaa.	3. Suuri. Toimeksiantaja joutuu maksamaan Amazonille AppFlowin käytöstä jokaista ajettua instanssia kohden.
Toimintavarmuus. Yrityksen sisäinen järjestelmä kaatuu ja siihen liittyvä data katoaa.	2. Keskisuuri. Apexissa oleva koodi sijaitsee Salesforcen eli kolmannen osapuolen palvelimilla eli vaikka käytössä tulisi katkos, pysyisi tapahtumien data Salesforcessa ainakin 24-tunnin ajan.	3. Suuri. Koska kyseessä on kustomoitua koodia, olisi ratkaisu suunniteltava niin, että se minimoi datan katoamisen mahdollisen palvelin katkoksen aikana.	1.Pieni. AWS:n ollessa kolmas osapuoli, vaikka toimeksiantajan palvelimet menisivät nurin, tieto voidaan tallentaa AWS:n sisälle esimerkiksi S3:seen tai muuhun ratkaisuun.
Tietoturva. Integraatiototeutuksessa tieto ei pysy turvallisesti piilotettuna ja yksityisenä.	3. Keskisuuri. Mikäli asiakas ei konfiguroi omaa salaisuuttaan tai avaintaan, ratkaisussa jouduttaisiin käyttämään samaa salaisuutta	2. Keskisuuri. Tietoturvan suunnitteleminen on pääosin ohjelmoijien omissa käsissä, mutta sen implementointi voi olla vaikeaa.	1. Pieni. Tiedon turvallinen kuljettaminen pystytään järjestämään Salesforcesta itselle käyttäen PrivateLinkiä ja yksityisavain pystytään konfiguroimaan

	jokaiselle asiakkaalle.		ulkoisessa järjestelmässä.
Helppokäyttöisyys. Vaikuttaako projektin vaikeudet sen myymisen asiakkaalle?	3. Suuri. Asiakas joutuisi konfiguroimaan itse salaisuudet ja avaimen ratkaisussa.	1. Matala. Asiakkaan ei tarvitsisi tehdä mitään vaan yhteyden voisi luoda API-kutsuja käyttäen.	1. Matala. Asiakkaan ei tarvitsisi tehdä mitään vaan yhteyden voisi luoda API-kutsuja käyttäen.

Kuten pystymme havainnoimaan, AppFlow on ainoa kolmesta ratkaisusta, jossa ainoastaan budjettiin liittyvässä kategoriassa riski on arvioitu suureksi. Apexin tapauksessa sen suurimmat vahvuudet löytyivät sen budjetista ja pub/sub API:ssa sen budjetista ja helppokäyttöisyydessä asiakkaalle.

Seuraavaksi laskemme kustannusvaikutteen vaikutusta, kun suhteutetaan niitä yrityksen sisäisten sidosryhmien kokonaiseen työmäärään, ja kuinka edellä mainitut integraatiotavat voisivat vaikeuttaa sidosryhmien toimintaa, lisäämällä heidän tuntipanostansa. Kyseinen laskelma tulee olemaan hypoteettinen arvio, joka tulee pohjautumaan projektiarvioon, jossa kokonaisten työtuntien määrä tulee olemaan 2000 tuntia. Toimeksiantajan mukaan, heidän yrityksessään sisäisten sidosryhmien työpanos yksittäisessä projektissa on:

- Kehitysryhmä: 70 %
- Projektinjohto (Sisäinen asiakas, tuoteomistaja): 20 %
- Myynti ja tuki: 10 %

Tämän pohjalta pystymme laskemaan, että keskimääräisessä projektissa, jossa arvio osuu kohdalleen, tulisi kokonaistyötuntien määrän per sidosryhmä laskettua seuraavasti: projektin kokonaistyötuntimäärä * sidosryhmän prosenttiosuus * riskikerroin, joka pohjautuu integraatiotavan vaikeuteen. Kertoimina tulemme käyttämään 25 % riskiprosenttia, joko alentamaan projektin vaikeustasoa, tai lisäämään sitä, integraatiotavan mukaan. Riskikertoimet tulisivat olemaan seuraavat:

- + 25 % Vaikea integraatiototeutus sidosryhmälle.
- 0 % Keskivaikea integraatiototeutus sidosryhmälle.
- -25 % - Helppo integraatiototeutus sidosryhmälle.

Seuraavalla sivulla on esitelty alustava kustannusvaikutusten lasku perustuen edellä mainittuihin riskeihin, kussakin integraatiototeutuksessa:

Taulukko 2. Kustannusvaikutusarvio integraatiovalinnan vaikutuksista yrityksen sisäisille sidosryhmille.

	Kehitysryhmä	Projektinjohto	Myynti/tuki
Apex	1400 h. Kerroin: 0 %. Apexin avulla koodauksen tarvetta ei eliminoida täysin integraatiototeutuksessa.	400 h. Kerroin: 0 %. Apexin avulla sisäisen asiakkaan ja projektin johdon on helppo määritellä tarpeet ja tiedot mitä he haluavat integraatioratkaisusta.	250 h. Kerroin: 25 %. Myynnissä, tuotteen myynti asiakkaalle olisi vaikeampaa, koska asiakas joutuisi olemaan valmis konfiguroimaan omat salaisuudet.
pub/sub API	1750 h. Kerroin: 25 %. Koska kyseessä on kustomoitava ratkaisu, joutuu kehitystiimi koodaamaan enemmän ja olemaan suuremmassa vastuussa järjestelmänsä toimintavarmuudesta.	500 h. Kerroin: 25 %. Sisäisten asiakkaiden ja johdon mietittävä tarkemmin mitä tietoa he haluavat pub/sub API:n lähettävän, jotta liika kuormalta vältytään.	200 h. Kerroin 0 %. Teknologian implementoinnilla ei ole vaikutusta myynnin toimintoihin.
AppFlow	1050 h. Kerroin: -25 %. AppFlow vähentää kehitysaikaa, koska se on jo valmiina löytyvä ratkaisu.	500 h. Kerroin: 25 %. Sisäinen asiakas ja projektijohto joutuu laskemaan AppFlowista aiheutuvia kuluja ja miettimään kuinka tehdä ratkaisusta kannattava.	200 h. Kerroin 0 %. Teknologian implementoinnilla ei ole vaikutusta myynnin toimintoihin.

Kuten pystymme tuloksista näkemään, AppFlowin tapauksessa kustannusvaikutus on kaikista pienin. Syynä, koska se vähentää kaikista eniten työtunteja pois kehitysryhmän työarviosta, joka toimii suurimpana sidosryhmänä projektin sisällä työpanokseltaan. AppFlowin tapauksessa myös ainoastaan yhdellä sidosryhmällä riski ylimääräisestä työmäärästä lisääntyisi. Toimeksiannon

tapauksessa Apexin ja pub/sub API:n käytöstä ei olisi yhdellekään yrityksen sisäiselle sidosryhmälle erityisen suurta hyötyä AppFlowin käyttöön verrattuna.

6 10Duken projektisuunnitelma

Analyysin ja vertailun pohjalta, toimeksiannon tilanteessa päätimme, että etenisimme Amazon AppFlowin kanssa, integraatioprojektin toteutukseen. Syinä, sen tuoma toimintavarmuus, sillä vaikka yrityksen sisäisessä palvelussa olisi katko, tapahtumien data tulisi kuitenkin säilymään AWS:n S3-tallennustilassa. Se myös säästäisi kehitysryhmän aikaa integraatiototeutuksen kehittämisessä, sillä yrityksen ei alustavasti tarvitsisi demoympäristössään erillisesti koodata päätyä mistä data siirretään yrityksen sisäiseen järjestelmään, vaan he voivat käyttää AppFlowin konsolia flowin muodostamiseen ja siirtää data S3:seen käyttämällä EventBridgeä. Tietoturva pysyisi myös hallittavissa AppFlowin PrivateLink-palvelun kautta. Seuraavaksi käsittelemme alustavan projektisuunnitelman, jonka perusteella teemme opinnäytetyön pohdinnan osuuden.

6.1 Alustus

10Duke Software Ltd. haluaa tehdä Salesforce integraatiokomponentin, yrityksen sisäiseen lisensointi- ja identiteetinhallintajärjestelmään asiakkaiden käyttöön. 10Duken asiakkaista monella on Salesforce käytössään. Ideana, että asiakkaat pystyisivät suorittamaan lisensoinnin- ja identiteetinhallintansa Salesforcen kautta.

6.1.1 Hyödyt

Usealla 10Duken asiakkaista on Salesforce käytössään, ja he toteuttavat pääosan liiketoiminnastaan Salesforcen kautta, täten heillä olisi toive, että he voisivat toteuttaa identiteetinhallintansa Salesforcen kautta. Projektin toteuttamisesta tuleva hyöty ja lisäarvo pohjautuisi asiakassuhteiden parantamiseen tarjoamalla 10Duken palveluja heidän haluamalla tavalla ja uusien asiakkaiden saamiseen. Kaikilla lisensoinnin- ja identiteetinhallinta ohjelmistoja tarjoavilla yrityksillä ei ole Salesforcea käytössä ja täten 10Duken palvelut voisivat erottua kilpailusta.

6.1.2 Tavoitteet

Projektilla tavoitellaan multitenancy eli useammalle asiakkaalle pohjautuvaa integraatiokomponenttia, käyttäen Amazon AWS:n AppFlow, EventBridge ja S3-teknologiaa. Ideana, että AppFlowta käyttäen Salesforcessa tapahtuvat tapahtumat kulkevat tietovirrassa Amazon EventBridgen reitittämänä S3-tallennustilaan. Tapahtumatiedot vastaanotettaisiin 10Duken sisäiseen järjestelmään käyttäen API-kutsua S3:seen, josta tiedot vastaanotetaan säännöllisin väliajoin. Käyttäjien pitäisi pystyä luomaan tilejä 10Duken järjestelmään Salesforcen kautta, kuten myös muokkaamaan asiakastietoja ja poistamaan niitä. Salesforce käyttäjien pitäisi myös pystyä lisäämään, muokkaamaan ja poistamaan lisenssejä, kuten myös toteuttaa asiakkaidensa identiteettien hallinta, kuten roolitus ja luvat.

6.1.3 Tuotevaatimukset

Projektin toivottu tuote, jonka se elämänkaarensa aikana tuottaa on Salesforce integraatiokomponentti osaksi 10Duken lisensoinnin- ja identiteetinhallintajärjestelmää. Vaatimuksina luoda multitenant pohjainen integraatiokomponentti, joka perustuu Amazonin AppFlow, EventBridge ja S3-teknologioihin. Muina tuotoksina projektista syntyy integraatiokomponentin lisäksi sen pohjalta luodut markkinointimateriaali ja tekninen dokumentaatio. Vaatimukset itse päätuotteelle on listattu tämän tekstin alapuolelle taulukkoon 3.

Taulukko 3. Projektin minimivaatimukset, yhdessä siihen liittyvän prosessin kanssa ja mihin sidosryhmään se eniten vaikuttaa.

Prosessi	Minimivaatimus	Sidosryhmä
Asiakkaan on pystyttävä tilaamaan integraatiokomponenttipalveluna ja yhdistettävä Salesforceen.	Asiakas kirjautuu 10Duken graafiseen käyttöliittymään ja tekee kirjautumisen Salesforce tunnuksillaan.	Asiakas
Asiakkaan on pystyttävä lisäämään, poistamaan, jakamaan lisenssejä omille asiakkailleen.	CRUD eli CREATE-, READ-, UPDATE- ja DELETE-operaatiot toimivat Salesforcen kautta.	Asiakas
Asiakkaan on pystyttävä jakamaan eri rooleja asiakkailleen lisensseistä.	Roolitus tapahtuu Salesforcea käyttämällä, ilman että asiakkaan tarvitsee konfiguroida mitään.	Asiakas
Asiakkaan on pystyttävä luomaan 10Duke tilejä omille asiakkailleen Salesforcen kautta.	Asiakkaalla on pystyttävä luomaan uusia asiakastilejä Salesforcen kautta, käyttäen Salesforcen standardi objekteja.	Asiakas
10Dukella on oltava admin näkymä integraatiokomponentille, jotta he pystyvät valvomaan palvelun suorituskykyä.	10Duken työntekijä pystyy kirjautumaan sisään yrityksen admin palveluun ja valvomaan palvelun suorituskykyä käyttämällä admin-tunnuksia.	10Duke

Tietoturva järjestelmässä on hoidettava käyttäen salattuja yhteyksiä ja sertifikaatteja.	RSA-SHA256-pohjainen algoritmi salaus ja digitaalisten kirjoitusten tekeminen.	Kaikki
Laskutus, 10Duken pitää pystyä hoitamaan laskujen lähetys asiakkaille.	Järjestelmän pitää pystyä lähettämään lasku 10Duken asiakkaalle säännöllisin väliajoin.	10Duke
Raportointi. 10Duken pitää pystyä saamaan raportteja järjestelmän suoriutumisesta ja sen toiminnasta.	Integraatiokomponentti pystyy muodostamaan raportteja suoriutumisestaan ja lähettää viestejä yrityksen muihin järjestelmiin.	10Duke
Yksityisyys. Järjestelmän on poistettava asiakkaiden tietoja säännöllisin väliajoin noudattamalla EU:n tietosuoja-asetusta.	Järjestelmä pystyy poistamaan asiakkaiden tietoja silloin, kun se ei niitä enää tarvitse Euroopan parlamentin ja neuvoston asetus luonnollisten henkilöiden suojelusta henkilötietojen käsittelyssä sekä näiden tietojen vapaasta liikkuvuudesta ja direktiivin 95/46/EY kumoamisesta 2016/679/EU mukaisesti.	Kaikki

6.1.4 Organisoituminen

Projektissa olevat resurssit tulevat pääosin pohjautumaan tehtävänjakoon yrityksen eri sisäisten sidosryhmien kanssa ja projektissa käytettäviin teknologioihin. Projektin työryhmä projektin kehitysvaiheessa tulee muodostumaan kehitysryhmästä, johon kuuluu kaksi kehittäjää. Kehitysryhmän tekemisen valvonnasta vastaa teknologiajohtaja ja projektin etenemisen valvonnasta ja aikataulusta vastaa projektijohtaja, joka toimii tuoteomistajana. Kehitysryhmän sisällä tulee toimimaan yksi pääkehittäjä ja yksi vanhempi kehittäjä, joka toimii konsultoivana osapuolena pääkehittäjälle ja neuvoo tätä. Kun tuote on valmis, sen myynnistä ja toimituksesta asiakkaalle tulee vastaamaan yrityksen sisäiset myynnin ja tuen toiminnot. Työalueittain työnjako näyttää seuraavalta:

(Taulukko 4. Tehtävien jako yrityksen eri sidosryhmien välillä)

Työalue	**Sidosryhmä**
Ohjelmisto- ja tuotekehitys	Kehitysryhmä, teknologiajohtaja
Projektin hallinta ja valvonta	Projektijohtaja, teknologiajohtaja
Tuotteen myynti ja toimitus asiakkaalle	Myynti- ja tukitoiminnot
Markkinointimateriaalin tuotto	Myynti- ja tukitoiminnot
Teknisen dokumentaation tuotto	Kehitysryhmä

Teknologisesti projektin resursseina tullaan käyttämään yrityksen sisäisiä ohjelmointiresursseja ja teknologiaa kuten ohjelmointikieliä (Java tms.), yrityksen sisäisiä kirjastoja ja valmiita komponentteja.

6.1.5 Riskit

Projektin tulevat riskit tulevat pohjautumaan opinnäytetyössä tehtyihin arvioihin. Tämä riskiarviointi tulee sisältämään aikaisemmin tehdyssä riskianalyysissa todetut riskit yhdessä muiden projektin hallintaan kohdistuvien riskien kanssa, niiden selvityssuunnitelmineen. Riskit mitä integraatioprojektissa voi tulla vastaan ovat:

Taulukko 5: Kuvaa projektin mahdollisia riskejä ja ratkaisuja.

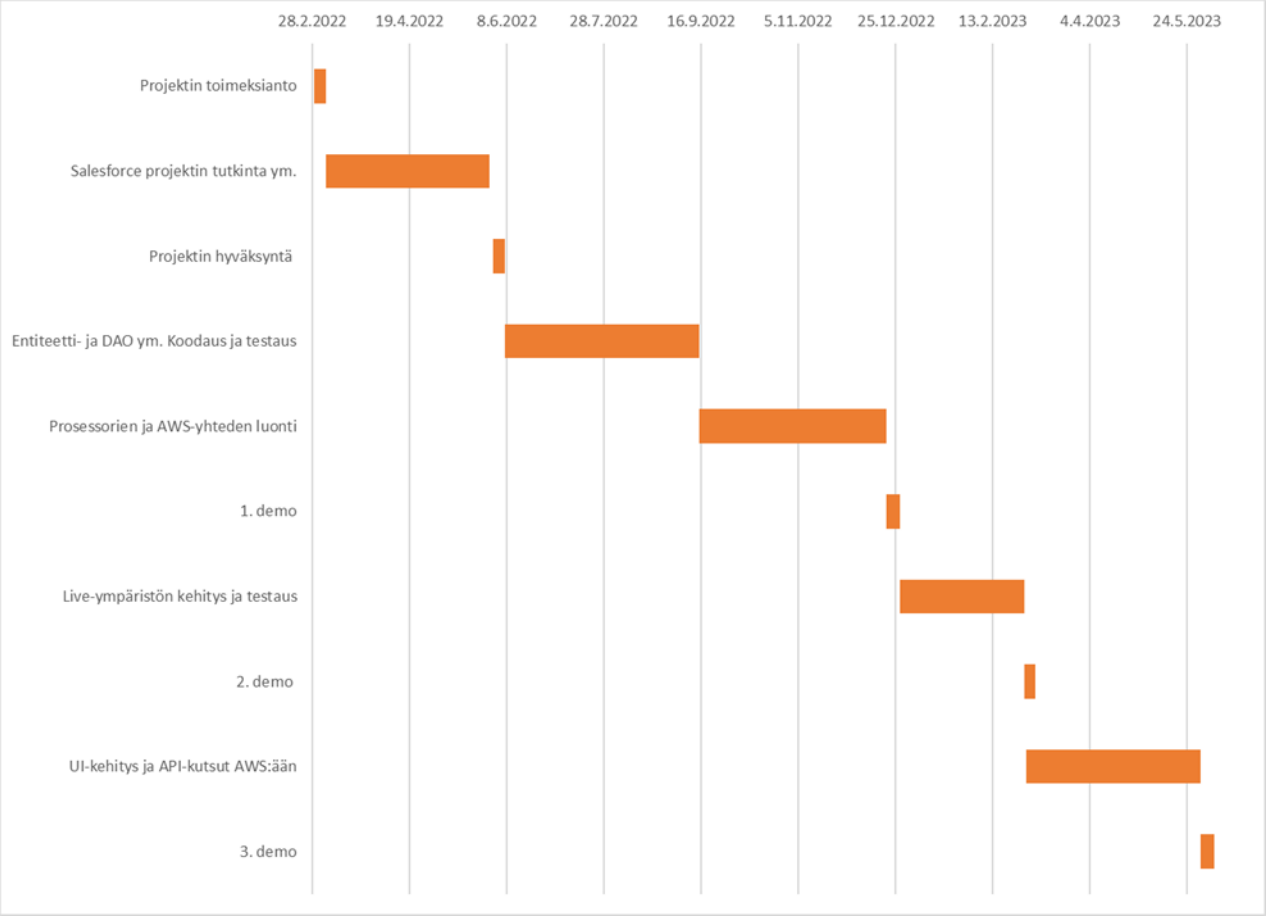
Riski	**Ratkaisu**
AppFlowin käyttö osoittautuu liian kalliiksi per asiakas.	Projektinjohto suunnittelee kustannusmallin projektille niin, että se on tuottelias.
Kehitysryhmässä koetaan yhteyden muodostaminen vaikeaksi useampaan Salesforceen käyttäessään AWS:n API:a	Vanhempi kehittäjä ja teknologiajohtaja keskittyvät enemmän problematiikan ratkomiseen.
Kehitysryhmä kokee kehityksen haastavaksi ja täten projekti viivästyy.	Projektinjohto asettaa kehitykselle tarpeeksi aikaa demojen välille, että kehitysryhmällä on enemmän aikaa ongelmanratkaisuun.
Projektin laajuus kasvaa liikaa ja tuotteeseen laitetaan liikaa ominaisuuksia	Projektinjohto ja kehitysryhmä karsivat olennaiset ominaisuudet epäolennaisista, heijastamalla niitä projektisuunnitelmaan.
S3:seen tulevat objektit ovat liian suuria payloadiltaan.	Kehitysvaiheessa, ennen demoja varmistetaan, että Salesforcesta tulevat payloadit ja niiden määrä ei ylitä AWS:n prosessoinnin rajoitteita.
Projektissa käytettävät teknologiat eivät ole yhteensopivia.	Luodaan alustavat demonstraatiot teknologioiden yhteensopivuudesta.
Projektille ei löydy tarpeeksi kattavaa loppu asiakaskuntaa.	Projektinjohto tekee kyselyn yrityksen asiakkailta heidän halukkuudestaan saada integraatiokomponentti.
Projektissa muodostuva koodi ei toimi oletetusti.	Kehitysryhmä tekee demoja esittelyä varten ja tekee tarvittavat testit ja metodit koodiin.

6.1.6 Aikataulut

Projektin aikataulu pohjautuu siihen perustuvien demojen esityksiin, jotka järjestetään säännöllisin väliajoin projektin aikana. Demojen ajat järjestetään alustavasti seuraavasti.

- 1.demo: 20.12.2022
- 2.demo: 1.3.2023
- 3.demo: 1.6.2023

Alempana esitetyn aikataulun tarkoituksena on esittää projektin ensimmäisen vuoden aikataulua 1.3.2022-1.6.2023 välisenä aikana.



(Kuva 14. Gantt-taulukko projektin etenemisestä.)

Projektin aikataulu ei sisällä projektin terminointivaihetta tai asiakkaan hyväksyntää, sillä projektin kehittymistä tullaan arvioimaan projektin elinkaaren läpi ja aiemmin esitetyn aikataulun tarkoitus on olla vain alustava versio aikataulusta.

6.1.7 Budjetti

Budjetinlaskenta tulee perustumaan pitkälti toimeksiantajan teknisten asiantuntijoiden arvioon, ja alustaviin laskelmiin projektin sisäisten sidosryhmien työpanoksesta. Työpanosten laskelmassa käytetään opinnäytetyössä toimeksiantajan antamia prosenttiarvioita kunkin sidosryhmän osuudesta kokonaistyötunteihin.

Projektissa yksi kehittäjä tulee toimimaan projektin pääkehittäjänä, jonka viikoittainen työtuntimäärä tulee olemaan arvioltaan 37.5 tuntia. Projektin toinen kehittäjä, jonka tarkoituksena on toimia konsultoivana osapuolena pääkehittäjälle arvioitu, tuntimäärä tulee olemaan noin 25 prosenttia pääkehittäjän tunneista, eli pyöristettynä 9.5 tuntia. Kehitysryhmän kokonaistuntimäärät viikolta tulevat olemaan siis noin 47 tuntia. Tämän perusteella pystymme arvioimaan muiden sidosryhmien työmäärän, jos kehittäjien aika projektista on noin 70 prosenttia, eli:

Projektin kokonaistuntimäärä:

x * 0,7 = 47 || :0,7

x = 67.142857

Eli tämän perusteella pystymme arvioimaan, että projektin kokonaistyötunnit viikossa ovat noin 67 tuntia, josta 10 prosenttia kuuluu myynnin- ja tuen funktioille ja 20 prosenttia projektinjohdolle, eli pyöristettynä sidosryhmien tunnit ovat.

- Projektinjohto: 13,5 h.
- Myynti- ja tuki: 7,5 h.
- Kehitysryhmä: 47 h.

Tämän perusteella pystymme laskemaan työvoimasta koituvan budjettiarvion. Yhdessä vuodessa on pyöristettynä noin 52 viikkoa, joten tämän ja sidosryhmien arvioidulla viikoittaisella työmäärällä pystymme laskemaan sidosryhmien koko vuoden työtunnit eli viikoittaiset työtunnit kertaa viisikymmentäkaksi.

- Projektinjohto: 702 h.
- Myynti- ja tuki: 390 h.
- Kehitysryhmä: 2444 h.

Kunkin sidosryhmän vuotuisten työtuntien määrillä pystymme laskemaan hinta-arvion sidosryhmien tehdyn työn arvosta.

Tunnillisen keskipalkan laskemiseen käytämme opinnäytetyössä esitettyjä keskiarvo palkkoja työnimikkeen perusteella, jossa tulemme jakamaan kuukausittaisen palkan keskimääräisellä työmäärällä kuukaudessa, jonka jälkeen jaamme arvon 7,5 tunnilla, joka vastaa työtä, jonka yksittäinen työntekijä tekisi päivässä eli:

- Keskimääräinen kuukausipalkka / 22 / 7,5 = tuntipalkka

Tämän avulla saamme arvion kunkin sidosryhmän yksittäisen työtunnin arvosta. Alempana esitetty taulukko kuvastaa kutakin sidosryhmää ja kuinka paljon yksi työtunti on arvoltaan.

Taulukko 6. Sidosryhmien keskiarvoiset kuukausi- ja tuntipalkat.

Sidosryhmä	**Käytetty keskiarvo**	**Tuntipalkka**
Projektijohto	5500 € / Projektijohtaja	33,30 €
Kehitysryhmä	3275 € / Ohjelmistokehittäjä	19,85 €
Myynti- ja tuki	3400 € / Myyntiedustaja	20,60 €

Näiden arvojen perusteella pystymme laskemaan työvoimasta aiheutuvan budjetin määrän laskemalla aikaisemmin esitettyjen työtuntiarvion perusteella palkoista aiheutuvat kulut kertomalla tuntipalkka kunkin sidosryhmän vuotuisella työtuntiarviolla. Sidosryhmistä aiheutuvat vuosikulut ovat listattuna alempana taulukossa.

Taulukko 7. Sidosryhmistä aiheutuvat kulut

Sidosryhmä	**Vuosikulut**
Projektijohto	23 727,60 €
Myynti ja tuki	8034 €
Kehitysryhmä	48 513,40 €

Näiden avulla pystymme laskemaan, että projektin vuotuinen minimibudjetin pitäisi olla yhteenlaskettuna 80 275 euroa. Budjetissa ei olla laskettu mukaan projektin ja tuotteenkehityksestä aiheutuvia palvelin tai resurssikuluja, koska toimeksiantajan mukaan ne ovat erittäin minimaalisia, eivätkä aiheuta yritykselle erityisiä muutoksia budjettiin heidän muiden projektien ohella. Myöskään kehittäjien työkalut eivät kuuluneet budjettiin, koska ne olivat valmiiksi kustannettuja ennen projektia.

6.1.8 Projektin raportointi & viestintä

Projektissa oleva raportointi tapahtuu yrityksen sisäisten järjestelmien kautta, jotka heillä ovat käytössä. Projektin aikana kukin yrityksen sisäinen sidosryhmä huolehtii heidän vastuualueellensa kuuluvien asioiden dokumentaation kirjoittamisesta ja hoitamisesta. Kehittäjät huolehtivat lopputuotteen kautta muodostuvan teknologian dokumentaatiosta (API-dokumentaatio, arkkitehtuurimallit ym.), sillä välin kun projektin johto huolehtii projektin sisäisten resurssien dokumentaatiosta. Myynti ja tuki ovat vastuussa tuote esitteistä ja markkinointimateriaalista.

Projektin sisäinen raportointi tapahtuu yrityksen käytössä olevien kanavien kautta. Käytössä olevia kanavia ovat:

- Google Chat: Yrityksen sisäiseen kommunikointiin ja viestittelyyn.
- Harvest: Tuntikirjanpito-ohjelma, työtuntien ja tehtävien raportointiin.
- Trello: Ohjelmisto tehtävien jakamiselle, jotta työtehtäviä pystytään jakamaan työntekijöiden kesken.

Projektissa on käytössään viikoittainen tapaaminen, jossa kehitysryhmä jäsenet raportoivat johdolle edistyksestään ja käydään läpi seuraavan viikon aikana tapahtuvat asiat ja suunnitelmat. johon sisältyy tavoitteet projektin edistymisestä.

7 Pohdinta

Opinnäytetyön projektin tarkoituksena oli tehdä 10Dukelle alustava analyysi Salesforcen eri integraatiototeutuksista, yhdessä alustavan projektisuunnitelman kanssa, joka pohjautuu analyysin pohjalta tehtyyn integraatiovalintaan. Sekä analyysin tuotto, että projektisuunnitelman tekeminen onnistui ja yritys pystyy jatkamaan integraatioprojektia projektisuunnitelman pohjalta, tietäen analyysin pohjalta, mitä teknisiä yksityiskohtia pitää mielessä tehdessään integraatiota AppFlowia käyttäen.

Analyysin kautta päästiin arvioimaan eri integraatiototeutusten business impactin vaikutusta ja eri integraatiototeutusten vaikutusta yrityksen sisäisiin sidosryhmiin. Yritys pääsi myös näkemään alustavat demonstraatiot ja havainnot heidän harkitsemistaan vaihtoehdoista integraatiolle, vaikka niiden täyttä käytännön toteutusta ei tullutkaan luotua. Projektisuunnitelman pohjalta yritys on myös tietoisempi projektiin liittyvistä tehtävistä, aikatauluista, budjetista ja organisaatiollisesta resurssien jakamisesta ja projektisuunnitelmaa pystytään päivittämään projektin elinkaaren aikana. Kehittäjät pystyvät hyödyntämään analyysista ja vertailusta syntyneitä havaintoja integraatiovalinnan tekemiseen teknologioiden välillä.

Yleisesti ottaen opinnäytetyö on onnistunut, sillä se vastasi johdannossa esitettyihin tutkimuskysymyksiin siitä mitä eri integraatio vaihtoehtoja Salesforcessa on, miten ne eroavat toisistaan ja mitä vahvuuksia ja heikkouksia niillä oli. Vaikkakin, opinnäytetyön skaalassa kaikkia integraatiototeutuksia ei päästy tutkimaan. Opinnäytetyöstä tuli selväksi, että integraatiovalintaa tehdessä, kannattaa erityisesti huomioida käytössä olevien teknologioiden rajoitukset ja tehdä valinnat niiden pohjalta, pikemminkin kuin sen pohjalta mikä alustavasti vaikuttaa kustannustehokkaimmalta. Jatkokehityksen kannalta kannattaisi selvittää AppFlowin suoriutumiskykyä mitattavien payloadien alla ja integraatiototeutusten välillä voisi tehdä testauksia nopeuden kannalta.

Lähteet

Adzic G. & Chatley R. 2017. Serverless Computing: Economic and Architectural Impact. In Proceedings of 2017 11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering. Paderborn. s.884–889.

Alias Z., Zawawi E.M.A, Yusof K. & Aris N.M. 2014. Determining Critical Success Factors of Project Management Practice: A conceptual framework. Procedia - Social and Behavioral Sciences. 153. s.61–69.

AWS, s.a.a. What is AWS? Luettavissa: https://aws.amazon.com/what-is-aws/. Luettu: 11.11.2022.

AWS. s.a.b. Amazon AppFlow. Luettavissa: https://aws.amazon.com/AppFlow/. Luettu: 11.11.2022.

AWS. s.a.c. Overview of Amazon AppFlow source and destination connectors. Luettavissa: https://docs.aws.amazon.com/AppFlow/latest/userguide/requirements.html. Luettu: 11.11.2022.

AWS. s.a.d. Amazon AppFlow FAQ. Luettavissa: https://aws.amazon.com/AppFlow/faqs/. Luettu: 18.11.2022.

AWS. s.a.e. What is Amazon Eventbridge? Luettavissa: https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-what-is.html. Luettu: 18.11.2022.

AWS. s.a.f. Buckets Overview. Luettavissa: https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingBucket.html. Luettu: 18.11.2022.

Bose, C. 24.1.2020. Brief History of Digital Marketing (1990-2020). Webcentralin blogi. Luettavissa: https://www.webcentral.au/blog/brief-history-of-digital-marketing-1990-2020/. Luettu: 2.12.2022.

Cambridge. s.a. Meaning of integration in English. Luettavissa: https://dictionary.cambridge.org/dictionary/english/integration. Luettu: 5.12.2022.

Craig, ID. 2007. Object-Oriented Programming Languages:Interpretation. Springer. Lontoo.

Euroopan parlamentin ja neuvoston asetus luonnollisten henkilöiden suojelusta henkilötietojen käsittelyssä sekä näiden tietojen vapaasta liikkuvuudesta ja direktiivin 95/46/EY kumoamisesta (yleinen tietosuoja-asetus) (EU) 2016/679. Annettu 27.4.2016.

Fieding, R. 2000. Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine. Tohtorin väitöskirja. Luettavissa: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. Luettu: 18.11.2022.

Gartner s.a. Business Impact Analysis (BIA). Luettavissa: https://www.gartner.com/en/information-technology/glossary/bia-business-impact-analysis. Luettu: 21.11.2022.

Geeksforgeeks 2022. Frontend vs Backend. Luettavissa: https://www.geeksforgeeks.org/frontend-vs-backend/. Luettu: 9.11.2022.

Grady, J. 1994. System Integration. CRC Press. Florida.

Gredi 2022. Integrointi lisää tiedon arvoa. Luettavissa: https://www.gredi.fi/tietojarjestelmien-integraatio-lisaa-tiedon-arvoa/#:~:text=Integrointi%20lis%C3%A4%C3%A4%20tiedon%20arvoa. Luettu: 9.11.2022.

Gido, J. & Clements, P. 2012. Instructor´s Manual. Successful Project Management. 5. painos. Cengage Learning. Massachusetts.

Hargrave, M. 5.9.2022. What Is CRM? Customer Relationship Management Defined. Luettavissa: https://www.investopedia.com/terms/c/customer_relation_management.asp. Luettu 9.11.2022.

Jansen, G. & Saladas, J. 11.4.2021. IBM. Advantages of the event-driven architecture pattern. Luettavissa: https://developer.ibm.com/articles/advantages-of-an-event-driven-architecture/. Luettu: 9.11.2022.

Jena KB. 11.11.2022. A Definitive Guide to Learn The SHA-256 (Secure Hash Algorithms). Luettavissa: https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm. Luettu: 5.12.2022.

Kapersky s.a. What is an SSL certificate – Definition and Explanation. Luettavissa: https://www.kaspersky.com/resource-center/definitions/what-is-a-ssl-certificate. Luettu: 5.12.2022.

Majumder, J, Shrivastava N., Rastogi R., Srinivasan.A. Scalable Content-Based Routing in pub/sub Systems. Institute of Eletrical and Electronics Engineers (IEEE) Annual Joint Conference, INFOCOM 2009. s. 567-575.

Mulesoft s.a. https://www.mulesoft.com/integration-solutions/saas/salesforce. Luettu: 5.12.2022.

Murch, R. Project Management: Best Practices for IT Professionals. 2000. 1.painos. Prentice Hall PTR. New Jersey. E-Kirja. Luettu: 5.12.2022.

Mäntyneva, M. Hallittu Projekti, Jäntevästä suunnittelusta menestykselliseen toteutukseen. 2016. Helsingin seudun kauppakamari. Helsinki.

O'Connel, B. 27.2.2020. History of Salesforce: Timeline and Facts. The Street. Luettavissa: https://www.thestreet.com/technology/history-of-salesforce. Luettu: 4.12.2022.

Oikotie s.a.a. Ohjelmistokehittäjä, palkka. Luettavissa: https://tyopaikat.oikotie.fi/palkkavertailu/ohjelmistokehitt%C3%A4j%C3%A4. Luettu: 4.12.2022.

Oikotie s.a.b. Projektijohtaja, palkka. Luettavissa: https://tyopaikat.oikotie.fi/palkkavertailu/Projektijohtaja. Luettu: 4.12.2022.

Oikotie s.a.c. Myyntiedustaja, palkka. Luettavissa: https://tyopaikat.oikotie.fi/palkkavertailu/Myyntiedustaja. Luettu: 4.12.2022.

Palankar M, Iamnitchi I, Ripeanu M & Garfinkel S. 2008. Amazon S3 for Science Grids: A Viable Solution? 08: Proceedings of the 2008 international workshop on Data-aware distributed computing. s.55-64.

PCMAG, s.a.a. Cloud. Luettavissa: https://www.pcmag.com/encyclopedia/term/cloud. Luettu: 9.11.2022.

PCMAG, s.a.b. Software Integration. Luettavissa: https://www.pcmag.com/encyclopedia/term/software-integration. Luettu: 9.11.2022.

Pinto, J.K & Slevin, D.P. 1988. Critical success factors across the project life cycle: definitions and measurement techniques. Project Management Journal, 19, 3, s.67–75.

Poniszewska-Maranda A., Matusiak R. & Kryvisnka N. 2017. Use of Salesforce Platform for Building Real-time Service Systems in Cloud. IEEE 14th International Conference on Services Computing, Honolulu. s.491–494.

Red Hat 2017. What is integration? Luettavissa: https://www.redhat.com/en/topics/integration/what-is-integration. Luettu: 9.11.2022.

Red Hat 2022. What is serverless? Luettavissa: https://www.redhat.com/en/topics/cloud-native-apps/what-is-serverless. Luettu: 18.11.2022.

Red Hat s.a. What is metadata? Luettavissa: https://access.redhat.com/documentation/it-it/red_hat_jboss_data_virtualization/6.4/html/user_guide_volume_1_teiid_designer/what_is_metadata. Luettu: 23.11.2022.

Schwalbe, K. 2015. An introduction to Project Management. 5.painos. Schwalbe Publishing. Minneapolis.

Salesforce s.a.a What is Apex? Luettavissa: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_intro_what_is_apex.htm. Luettu: 11.11.2022.

Salesforce s.a.b. Introduction to SOQL and SOSL. Luettavissa: https://developer.salesforce.com/docs/atlas.en-us.soql_sosl.meta/soql_sosl/sforce_api_calls_soql_sosl_intro.htm. Luettu: 11.11.2022.

Salesforce s.a.c Custom Fields. Luettavissa: https://developer.salesforce.com/docs/atlas.en-us.234.0.object_reference.meta/object_reference/custom_fields.htm. Luettu: 11.11.2022.

Salesforce s.a.d Certificates and Keys. Luettavissa: https://help.salesforce.com/s/articleView?id=sf.security_keys_about.htm&type=5. Luettu: 11.11.2022.

Salesforce s.a.e Distributing Applications and Components. Luettavissa: https://developer.salesforce.com/docs/atlas.en-us.198.0.lightning.meta/lightning/apps_packaging.htm. Luettu: 17.11.2022.

Salesforce s.a.f. Overview of Packages. Luettavissa: https://developer.salesforce.com/docs/atlas.en-us.234.0.packagingGuide.meta/packagingGuide/packaging_about_packages.htm. Luettu: 17.11.2022.

Salesforce s.a.g. What is a Package? Luettavissa: https://developer.salesforce.com/docs/atlas.en-us.234.0.sfdx_dev.meta/sfdx_dev/sfdx_dev_unlocked_pkg_whats_a_package.htm. Luettu: 17.11.2022.

Salesforce s.a.h. Protection and Privacy Options for Custom Settings. Luettavissa: https://help.salesforce.com/s/articleView?id=sf.cs_schema_settings.htm&type=5. Luettu: 5.12.2022.

Salesforce s.a.i. What is AppExchange? Luettavissa: https://developer.salesforce.com/docs/atlas.en-us.appExchangeInstallGuide.meta/appExchangeInstallGuide/appexchange_install_whatis.htm. Luettu: 17.11.2022.

Salesforce s.a.j. License Management App. Luettavissa: https://developer.salesforce.com/docs/atlas.en-us.204.0.packagingGuide.meta/packagingGuide/lma_intro.htm. Luettu: 17.11.2022.

Salesforce s.a.k. Delivering Custom Notifications with Platform Events. Luettavissa: https://developer.salesforce.com/docs/atlas.en-us.platform_events.meta/platform_events/platform_events_overview.htm. Luettu: 11.11.2022.

Salesforce s.a.l. Platform Events Developer Guide. Luettavissa: https://developer.salesforce.com/docs/atlas.en-us.platform_events.meta/platform_events/platform_events_intro.htm. Luettu: 11.11.2022.

Salesforce s.a.m. Developer Console. Luettavissa: https://help.salesforce.com/s/articleView?id=sf.code_dev_console.htm. Luettu: 11.11.2022.

Salesforce s.a.n. Event-Driven Software Architecture. Luettavissa: https://developer.salesforce.com/docs/atlas.en-us.platform_events.meta/platform_events/platform_events_intro_architecture.htm. Luettu: 11.11.2022.

Salesforce s.a.o. Get started with pub/sub API. Luettavissa: https://developer.salesforce.com/docs/platform/pub-sub-api/guide/intro.html. Luettu: 11.11.2022

Salesforce s.a.p. Message Durability. Luettavissa: https://developer.salesforce.com/docs/atlas.en-us.api_streaming.meta/api_streaming/using_streaming_api_durability.htm. Luettu: 20.11.2022.

Stackpath s.a. What is pub/sub Messaging? Luettavissa: https://www.stackpath.com/edge-academy/what-is-pub-sub-messaging/. Luettu: 9.11.2022.

Stroustrup B. s.a. Bjarne Stroustrup's C++ Glossary. Luettavissa: https://www.stroustrup.com/glossary.html. Luettu: 9.11.2022.

Stroustrup B. 1995. Why C++ is not just an Object-Oriented Programming Language. AT&T Bell Laboratories. New Jersey. Luettavissa: https://www.stroustrup.com/oopsla.pdf. Luettu: 9.11.2022.

Sunkari, S. 2022. A Brief Review On CRM, Salesforce, and Reasons Stating Salesforce as One of the Top CRM´s. Dextera Digital Private Limited. Hyderabad.Luettavissa: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4158451. Luettu: 5.12.2022.

TechMetrixResearch. 1999. Java Application Servers Report. Luettavissa: https://web.archive.org/web/20101229090912/http://www.fscript.org/prof/javapassport.pdf. Luettu: 5.12.2022.

Thakkar M. & Rajaan R. 2020. Salesforce CRM: A new way of managing Customer Relationship in a cloud environment. International Journal of Eletrical, Electronics and Computers. 5, 3, s.14-17.

Tjoa S., Jakoubi S. & Quirchmaur G. s.a. Enhancing Business Impact Analysis and Risk Assessment applying a Risk-Aware Business Process Modeling and Simulation Methodology. 2008 International Conference on Availability, Reliability and Security (ARES), Barcelona.

Trailhead s.a.a. Understanding Understand the Salesforce Architecture. Luettavissa: https://trailhead.salesforce.com/content/learn/modules/starting_force_com/starting_understanding_arch. Luettu: 22.11.2022.

Trailhead s.a.b. Get Started with Apex Triggers. Luettavissa: https://trailhead.salesforce.com/content/learn/modules/apex_triggers/apex_triggers_intro. Luettu: 11.11.2022.

Wyngaard C.J, Pretorius J.H.C, Pretorius L. 2012. Theory of Triple Constraint - a Conceptual Review. 2012 IEEE International Conference on Industrial Engineering and Engineering Management. s.1991-1997.

Yazar K. s.a. Salesforce. TechTarget. Luettavissa: https://www.techtarget.com/searchcustomerexperience/definition/Salesforcecom. Luettu: 21.11.2022.