Peter Aalto

Asiakkuudenhallintajärjestelmän integrointi ajanvarausjärjestelmään

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

4.11.2015

Metropolia

Tekijä Otsikko Sivumäärä Aika	Peter Aalto Asiakkuudenhallintajärjestelmän integrointi ajanvarausjärjestelmään 32 sivua 4.11.2015
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja	Lehtori Peter Hjort

Insinöörityön päätavoitteena oli suunnitella ja toteuttaa integraatio asiakkuudenhallintajärjestelmän ja ajanvarausjärjestelmän välille. Integraation tarkoituksena oli todentaa, onko ajanvarausjärjestelmään mahdollista integroida monen käyttäjän järjestelmää ja siirtää niiden välillä kalenteridataa. Järjestelmiä yhdistämällä ajanvarausjärjestelmään saataisiin keskitetty ohjelma, joka avustaisi käyttäjiä järjestämään tapaamisia keskenään.

Asiakkuudenhallintajärjestelmänä (CRM) käytettiin Salesforcea ja ajanvarausjärjestelmänä Misseä. Integraatio toteutettiin Salesforcessa Apex-ohjelmointikielen avulla Missen ohjelmointirajapintaa vasten.

Työn lopputuloksena ei ollut täysin toimiva integraatio Salesforcen ja Missen välille, mutta työn seurauksena saatiin arvokasta tietoa Misseen ja Salesforceen vaadituista jatkokehitystarpeista. Missen jatkokehitystarpeisiin sisältyy tuki monen käyttäjän järjestelmiin ja Salesforcen osalta pitäisi testata ja kehittää parempi käyttöliittymä.

Avainsanat	CRM, Salesforce, ajanvarausjärjestelmä, asiakkuudenhallintajärjestelmä

Author	Peter Aalto
Title	CRM system integration to appointment scheduler
Number of Pages	32 pages
Date	4 November 2015
Degree	Bachelor of Engineering
Degree Programme	Information and Communications Technology
Specialisation option	Software Engineering
Instructor	Peter Hjort, Senior Lecturer

The main goal of this final year project was the planning and implementation of an integration between a customer relationship management system and an appointment scheduler system. The point of the integration is to verify if it is possible to integrate an appointment scheduler system to a multi-user system and to move calendar data between them. Connecting systems to the appointment scheduler would result in a program which helps reserving meetings between users.

The CRM system used was Salesforce and the appointment scheduler system used was Misse. The integration in Salesforce was implemented with Apex as the programming language, using the programming interface of Misse.

The final result was not a fully functional integration between Salesforce and Misse, but the results gave valuable information for further development of Misse and Salesforce. The continued development of Misse would require building support for a multi-user system and Salesforce would require testing and development of the user interface.

Keywords	CRM, Salesforce, appointment scheduler

Sisällys

Lyhenteet

CRM	Customer relationship management. Asiakkuudenhallinta.
Apex	Force.com-alustan käyttämä javamainen ohjelmointikieli.
Visualforce	Merkkikieli, jonka avulla luodaan Apexiin linkitettyjä sivuja. Ohessa voi myös käyttää muita käyttöliittymäkehitykseen tarkoitettuja teknologioita, kuten HTML ja Javascript.
SaaS	Software as a Service. Pilvestä tarjottava ohjelma tai palvelu.
PaaS	Platform as a Service. Pilvestä tarjottava kehitysalusta.
SOAP	XML-pohjainen viestintäprotokolla.
REST	Representational State Transfer. Malli, joka parantaa ohjelmointirajapintojen toteuttamista.
SDK	Software Development Kit. Kehitystyökalu-paketti mahdollistaa sovellusten ohjelmoimisen tietylle alustalle.

1 Johdanto

Insinöörityön tarkoituksena on suunnitella ja toteuttaa integraatio asiakkuudenhallinta-järjestelmän ja ajanhallintajärjestelmän välille. Asiakkuudenhallinta- eli CRM-järjestelmänä toimii Salesforcen tarjoama Force.com, ja ajanhallintajärjestelmänä toimii prototyyppinä tehty Misse. Integraation keskeisiin aiheisiin kuuluvat suunnittelussa esiin tulevat ongelmat ja niiden ratkaisut sekä itse integraation toteutus. Suunnitteluvaiheeseen kuuluu Salesforcen ja Missen datamallien kartoitus ja niiden yhteen integrointi. Selvitän työssä myös olemassa olevia teknologioita, joita integraatiossa voi käyttää hyväksi, ja valitsen niistä parhaan perustellusti.

Misse on ajanhallintajärjestelmä, jonka tarkoituksena on keskittää tapaamisten järjestäminen riippumatta kalenterijärjestelmästä. Järjestelmä on kehitetty mobiililaitteet huomioiden, joten jo projektiin lähdettäessä tiedetään mahdolliset ongelmat integroitaessa sitä CRM-järjestelmään. Projektia aloitettaessa Missen versio on ainoastaan prototyyppi.

Force.com on Salesforcen tarjoama sovelluskehitysalusta, joka toimii pilvessä ja johon on mahdollista kehittää omia sovelluksia Salesforcen tarjoamilla työkaluilla. Niitä voidaan tarjota Salesforcen Appexchange-nimisessä sovelluskaupassa joko ilmaisena tai maksullisena. [1.]

Työn päätarkoituksena on selvittää, kuinka Missen prototyyppi joustaa integroitaessa monen käyttäjän järjestelmään ja tässä tapauksessa pilvipalveluna toimivaan alustaan. Työssä käydään ensin läpi lyhyesti CRM:n historiaa ja muita markkinoilla olevia CRM-järjestelmiä ja kuvaillaan Salesforce-järjestelmässä käytettäviä työkaluja. Loppuosa työstä on jaettu integraation ja sen ympärillä vaaditun logiikan suunnitteluun ja itse toteutuneen lopputuloksen esittelyyn.

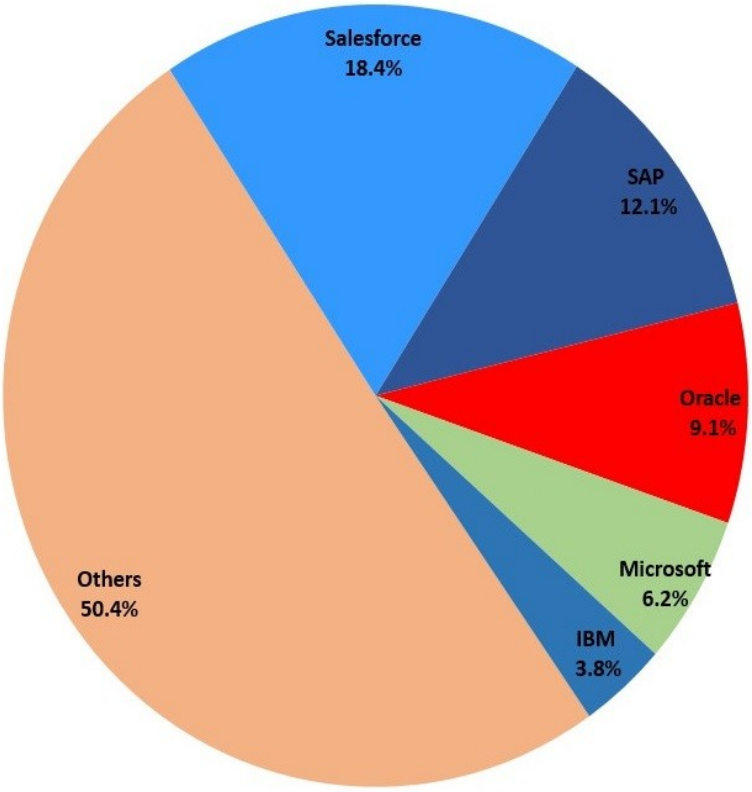
2 CRM-pilvipalvelut

CRM-järjestelmien eli asiakkuudenhallintajärjestelmien on nimensä mukaisesti tarkoitus avustaa yritystä parantamaan kontaktia uusiin asiakkaisiin sekä ylläpitämään ja parantamaan nykyisiä asiakassuhteita. Tämä mahdollistetaan järjestelmän asiakasrekisterillä ja myyntiprosesseilla sekä niistä syntyvillä tiedoilla. [2.] Yleensä CRM-järjestelmän käyttämiseen päädytäänkin sen jälkeen, kun asiakastietojen hallinta kasvaa sietämättömäksi joko Excelissä tai muussa kevyessä tiedonhallintaohjelmassa.

Ensimmäiset CRM-järjestelmät olivat asiakkaan tiloihin rakennettuja tai asennettuja ohjelmistokokonaisuuksia, joita räätälöitiin asiakkaan tarpeiden mukaisesti. Alkujaan CRM-järjestelmiä käytettiin ainoastaan asiakasrekisterinä, paikkana, johon lisätään ja muokataan asiakastietoja ilman sen suurempaa integraatiota. Projektit olivat kalliita ja toteutus kesti pitkään, mikä antoi oivan tilaisuuden uusille ratkaisuille internetin kehittyessä. [3.]

Internetin vahvistuessa 1990-luvun loppupuolella ilmaantui alalle uudenlainen CRM-tyyppi, SaaS-ratkaisut, joita tarjotaan ja käytetään pilvessä. Näistä ensimmäisten joukossa markkinoille tullut ja nykypäivän isoin tekijä on Salesforce. [4.] SaaS-ratkaisujen etuna oli nopea käyttöönotto ja pienet kustannukset sekä se, että ylläpidon vastuu on palvelun tarjoajalla. Nykyajan pilvessä toimivissa CRM-järjestelmissä on myös etuna hyvät integraatiomahdollisuudet. Tämän vuoksi on tavallista, että CRM-järjestelmää toteutettaessa se integroidaan myös esimerkiksi taloushallinto-, toiminnanohjaus- tai laskutusjärjestelmien kanssa, ja näin syntyy liiketoimintaa hyvin tukeva paketti.

Nykypäivänä markkinoilla on tarjolla lukuisia eri CRM-ratkaisuja vaihdellen asiakkaan tiloihin tehdyistä ratkaisuista pilvipalveluihin sekä erikokoisille yrityksille tarkoitettuja järjestelmiä. Suurin CRM-tarjoaja on Salesforce 18,4 prosentin liikevaihdolla (kuva 1) maailmanlaajuisesti [5].



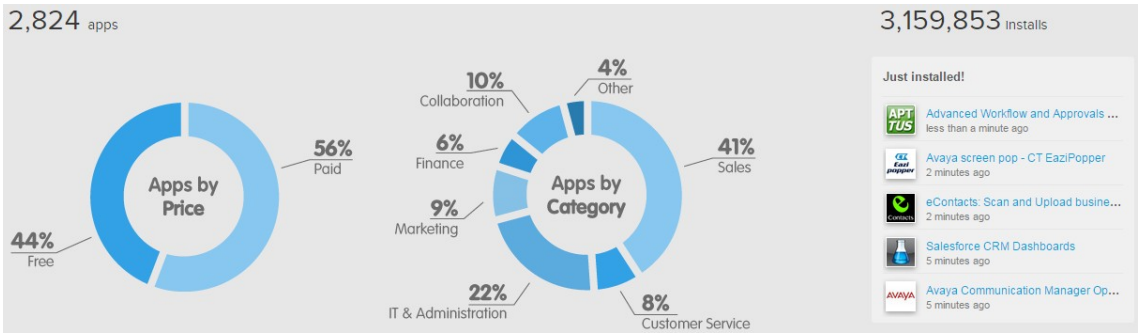
Kuva 1: CRM-toimittajien liikevaihto vuonna 2014 [5].

3 Salesforce.com ja Force.com

3.1 Yleisesti

Salesforce.com-sovellus on rakennettu Force.com-alustalle, joka tarjoaa monia valmiita ratkaisuja, kuten tapausten-, tehtävien- ja tapahtumienhallinnan [21]. Salesforce.com on SaaS-ratkaisu, jota myydään eri variaatioina ja lisensseillä, jotka määrittävät, kuinka isoon osaan Salesforce.comin toiminnallisuuksista käyttäjä pääsee käsiksi ja kuinka korkeat käyttörajoitukset ympäristössä on. Kuten Salesforce.comin tapauksessakin, SaaS tarkoittaa ohjelmiston tarjoamista keskitetystä palvelusta, josta maksetaan lisenssimaksua [6]. Eri Salesforce.com-editioita ja lisenssejä on lukuisia, mikä tekee myös niiden sisäistämisestä ja ymmärtämisestä hankalaa.

Force.com on PaaS-ratkaisu, mikä tarkoittaa pilvipalvelualustaa, jonka päälle on mahdollista kehittää web-sovelluksia [7]. Salesforce.com on myös rakennettu Force.com-alustan päälle. Käytännössä ne näyttävät keskenään aivan samalta, ja molemmat myös mahdollistavat sisällön räätälöimisen samalla tavalla, mutta Force.com-ympäristöstä puuttuvat lähes kaikki Salesforce.comin tarjoamat ominaisuudet. Force.comilla on mahdollista tehdä uusia sovelluksia Apexin, Visualforcen ja Salesforcen tarjoamien konfiguraatiotyökalujen avulla. Force.com mahdollistaa tällöin alustallaan tehtyjen sovellusten tarjoamisen Appexchangessa, joko ilmaisena tai maksullisena. [1.] Appexchangessa on yhteensä 2 824 sovellusta sekä noin 3,1 miljoonaa sovelluksen asennuskertaa (kuva 2).



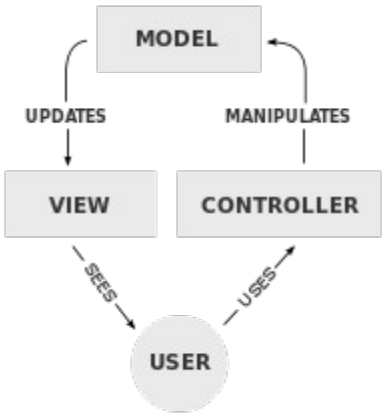
Kuva 2: Sovellusten määrä Appexchangessa [8].

3.2 Salesforce-kehittäminen

Salesforce-kehittäminen on tavalliseen verkkosovelluskehittämiseen verrattuna nopeampaa ja helpompaa, koska uusien ympäristöjen pystyttäminen on nopeaa ja kehitysympäristö saadaan nopeasti kehittäjälle käyttöön. Kehittäjän ei tarvitse huolehtia palvelimen ylläpitoon liittyvistä asioista tai tietokantojen konfiguroimisesta muutoin kuin kenttien lisäämisen ja muokkaamisen osalta. [9.] Salesforce tarjoaa valmiiksi lukuisia valmiita kenttätyyppejä, jotka täyttävät yleisimmät käyttötarpeet, kuten valintalistat, valintaruudut ja monia muita. Puuttuvia ominaisuuksia Salesforce pyrkii tarjoamaan päivityksillä käyttäjien toiveiden mukaisesti. Kaikki on käytännössä piilotettu Salesforce-ympäristön web-pohjaisen käyttöliittymän taakse, jonka kautta järjestelmän asetuksia kontrolloidaan. Tämä on myös kehittämisen kannalta huono puoli, sillä moni asia on muokkaamattomissa siihen asti, kunnes Salesforce avaa niitä kehityspäivityksillä.

Salesforce vaatii myös yksikkötestien tekoa järjestelmällisesti. Ainakin 75 %:n koodista pitää olla testiluokkien kattama, muutoin järjestelmä estää sovelluskoodin siirron tuotantoympäristöön. Tällä varmistetaan jollakin tasolla koodin ja sovelluksen toimivuus ja pakotetaan kehittäjiä kiinnittämään huomiota myös testaamiseen. Tähän Salesforce tarjoaa järjestelmän sisällä oman yksikkötestauskomponentin, jonka avulla testit suoritetaan. Tämä myös näyttää tarkalleen, mikä koodirivi on suoritettu ja mikä ei, sekä prosenttiosuuden koodin testatusta osuudesta.

Salesforce-kehittäminen toimii MVC-arkkitehtuurin mukaisesti (kuva 3). Mallissa tietokantaobjektit (Model), näkymä (View) ja käsittelijä (Controller) ovat eroteltuna omiin tehtäviinsä, jossa käsittelijä ottaa vastaan käyttäjän näkymästä viestejä, käsittelee niitä tietokannassa ja vie tämän perusteella viestin takaisin käyttäjälle. MVC:n malleina Salesforcessa toimii sObjekti, joka on kuvaus tietokannan datasta jokaiselle Salesforcessa olevalle taululle. Mukaan on luettu myös Apexin Malli-luokat. Näkymänä toimivat Visualforce-sivut ja standardit sivuasettelut, kun taas käsittelijänä toimii Apex-koodi, joka vie mallina toimivan datan näkymälle näytettäväksi.



Kuva 3: MVC-malli [10].

3.3 Toteutuksessa käytetyt Salesforce-tekniikat

Jos Salesforceen halutaan tehdä standardista poikkeavia työkaluja tai suurempia räätälöintejä, ne täytyy tehdä Apexin ja Visualforcen avulla. Apex on vahvan tyypityksen olio-ohjelmointikieli, joka on paljolti Javan kaltainen. [11.] Apexin tietokantakutsut ovat tallennettujen proseduurien kaltaisia. Visualforce on taas merkkikieli, jonka ohessa toimivat myös HTML, CSS ja Javascript. Visualforcen avulla linkitetään Visualforce-komponentit, painikkeet ja muut käyttäjän tekemät tapahtumat Apex-koodin metodeihin.

Apex

Apex mahdollistaa mukautettujen komponenttien luomisen Force.com-alustalle Apex-koodin avulla. Apex-koodia voidaan suorittaa muun muassa räätälöidyillä painikkeilla, Visualforce-sivuilla ja objektikohtaisilla triggereillä. Apexilla on myös mahdollista helposti päästä käsiksi Salesforcen taustalla toimivaan tietokantaan ja luoda käyttäjille uusia tapoja käyttää CRM:ää riippuen liiketoiminnan vaatimuksista. Apexissa tietokannan datan hakeminen onnistuu helposti SOQL-kyselyillä (kuva 4), jotka ovat samankaltaisia kuin tavallinen SQL, mutta suunniteltu Salesforcen dataa varten.

```
List<Account> accounts = [SELECT Id, Name FROM Account WHERE Name = 'Matti'];
for(Account account : accounts)
{
    account.Name = 'Teppo';
}
update accounts;
```

Kuva 4: Esimerkki SOQL:n käytöstä Apexissa.

Apexilla on käytännössä siis mahdollista tehdä räätälöityjä prosesseja Visualforce-sivujen taakse, triggereitä, jotka reagoivat tietueisiin kohdistuviin tapahtumiin, ja web service -rajapintoja ulkoisille ohjelmille [12]. Apex-sovellukset tallentuvat Salesforcen pilveen ja suoritetaan siellä, joten käyttäjäkoneille ei ole tarvetta asentaa ohjelmia. Pääosin Salesforcen toiminnallisuuksia käytetäänkin selaimen avulla.

Yksi iso puute Apexissa ovat nimiavaruudet, joita on ainoastaan standardeissa Apex-luokissa. Nimiavaruuksien avulla on ohjelmointimaailmassa tapana organisoida koodiluokat omiin toiminnallisuuksiin hierarkkisesti, mikä helpottaa koodin hallintaa. [13.] Tämän puute vaikeuttaa keski- ja suurikokoisten projektien hallintaa huomattavasti, ja siksi Apex-luokkien oikein nimeämisen painoarvo nousee tärkeäksi.

Visualforce

Visualforce (kuva 5) on Salesforcen HTML:n oheen kehittämä merkkikieli, joka mahdollistaa näkymien teon käyttäjille helposti ja samalla integroimalla sen taustalla suoritettavaan Apex-koodiin. Nämä tagit käyttävät Salesforcen vakiosivujen tyyliä, mikä myös vähentää kehittäjän työtä huomattavasti CSS-tyylien kehittämisessä. Visualforce-sivut mahdollistavat myös HTML:n, CSS:n ja Javascriptin käytön.

```
<apex:page standardController="Account">
    <apex:form>
        <apex:pageBlock title="Edit Account for {!$User.FirstName}">
            <apex:pageMessages/>
            <apex:pageBlockButtons>
                <apex:commandButton value="Save" action="{!save}"/>
            </apex:pageBlockButtons>
            <apex:pageBlockSection>
                <apex:inputField value="{!account.name}"/>
            </apex:pageBlockSection>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

Kuva 5: Visualforce-esimerkki [14].

Kiteytettynä Visualforcea voidaan käyttää upottamaan Salesforcen vakiosivuille pieniä Visualforce/Apex-toiminnallisuuksia tai tehdä täysin räätälöityjä työkaluja, kuten esimerkiksi kokonaisia myyntiprosesseja tai muita kevyempiä prosesseja auttamaan käyttäjää. [15.]

3.4 Toteutuksessa käytetyt työkalut

Konfiguraatiot ja kevyemmät Salesforcen räätälöinnit tehdään selainkäyttöliittymän kautta Salesforcen ympäristöön kirjautuneena. Niihin liittyy objektien, kenttien, validointien ynnä muiden standardien ominaisuuksien hallinta.

Työkaluina Apexin ja Visualforcen kehittämisessä voi käyttää joko Salesforceen tehtyä selainpohjaista editoria tai Salesforcen rajapintoja käyttäviä, tietokoneille asennettavia ohjelmia. Salesforcen selaineditori on kätevä ainoastaan silloin, jos tavoitteena on tehdä nopeita ja pieniä muokkauksia Apexiin tai Visualforceen. Se tukee kevyttä sisällön avustajaa, joka täydentää sillä hetkellä kirjoitettua koodia. Siitä puuttuvat muut tärkeät sovelluskehitystyökalujen ominaisuudet.

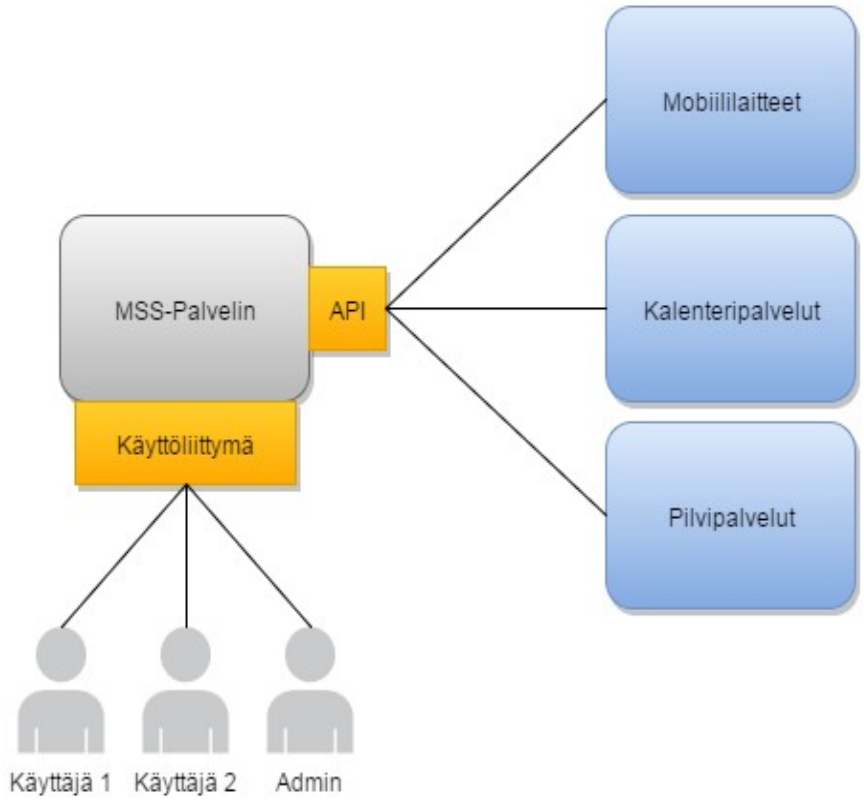
Ulkoiset ohjelmat toimivat laajemmassa sovelluskehityksessä paremmin. Näistä yksi vaihtoehto on Eclipse, joka on avoimen lähdekoodin ohjelmointiympäristö, johon on tehty Force.com-laajennus. Tämä laajennus käyttää hyväkseen Salesforcen SOAP-rajapintaa ja täten mahdollistaa koodin tallentamisen suoraan haluttuun Salesforce-instanssiin.

Käytin itse IDE:nä pääosin Eclipseä Force.com-laajennuksen kanssa. Testauksessa ja ohjelmalogiikan seuraukseen käytin Salesforcen omaa Developer consolea, joka mahdollistaa lokien seuraamisen ja virheiden korjaamisen. Sen avulla myös suoritetaan yksikkötestit ja nähdään niiden testikattavuus.

4 Nykyinen ajanvarausjärjestelmä

4.1 Yleisesti

Misse-ajanvarausjärjestelmän tarkoitus on avustaa tapaamisten organisoinnissa ja ajoittamisessa (kuva 6). Tapaamista järjestettäessä järjestelmä tutkii, onko kutsutuilla tilaa valituilla aikajaksoilla, ja lähettää järjestäjälle lopulta mahdolliset aikavalinnat, jotka kävisivät kaikille osallistujille.



Kuva 6: Misse-ajanvarausjärjestelmä.

Misse toimii palvelinkoneella, jonka tarkoitus on olla keskitetty järjestelmä, johon integroidaan erinäiset kalenterijärjestelmät. Tällöin Misse saisi kalenterijärjestelmästä riippumatta haettua kalenteripäiväysdatan, jonka perusteella yhteiset vapaat ajat saataisiin

laskettua. Laskettu data lähetetään ja näytetään tapaamista järjestävälle henkilölle, joka voi tarjotuista vaihtoehdoista valita parhaan päivämäärän.

Misse-järjestelmään integroitujen laitteiden tai järjestelmien ei tarvitse jakaa koko kalenteridataa, vaan ainoastaan tiettyjen päivämäärävälien ajalta vapaat ajat. Nämä ajat jaetaan riippuen siitä, mitkä tapahtuman järjestäjä on asettanut alku- ja loppupäivämääräksi. Tämä mahdollistaa käyttäjien yksityisyyden ylläpitämisen ja tarkkojen tapaamistietojen salassa pitämisen.

4.2 Prototyyppi

Nykyinen Misse-ajanvarausjärjestelmä on vain prototyyppi, joten kaikkia siihen suunniteltuja ominaisuuksia ei ole vielä toteutettu. Prototyyppi on koodattu Pythonilla ja rajapintoja testattu nopeasti koodatulla ohjelmalla. Ohjelmaa testattiin Nokian N900-mallin matkapuhelimella. Yksi Missen puutteista oli käyttäjän järjestelmiin tai pilvipalveluihin tehty integraatiomahdollisuus, mikä jo alkuvaiheessa tiedostettiin. Prototyyppi tuki siis parhaiten vain mobiilikäyttäjien integroimista järjestelmään, joten CRM-järjestelmän integroimisen oletettiin olevan haasteellista. Prototyypin valmiit REST-rajapinnat kuitenkin mahdollistivat jonkintasoisen integraation, ja siksi toteutusta jatkettiin suunnitteluvaiheesta eteenpäin.

4.3 Tekninen kuvaus

Misse tarjoaa mobiilikäyttöön pääosin tarkoitettua long polling -rajapintaa, jota kutsumalla laitteet pitävät jatkuvaa HTTP-yhteyttä Misseen. Kun Misseen järjestetään uusi tapaaminen, palautetaan jokaiselle rajapintaa yhteyttä pitävälle laitteelle viesti kalenteritietojen tarpeesta. Tämän jälkeen laitteet lähettävät Misseen halutulta aikaväliltä vapaat kalenteriajat, joiden avulla Misse päättelee kaikille avoimet ajat. Tämä kokonaisuudessaan tarkoittaa sitä, että jokaisen päätelaitteen pitää käyttää sille toteutettua ohjelmaa, joka pystyy tämän prosessin toteuttamaan. Tällaisia ohjelmia tarjottaisiin päätelaitteiden sovelluskaupoissa käyttäjille. Kalenteripalveluille, jotka toimivat palvelimella tai työpöytäkoneilla, Misse tarjoaa REST-rajapinnan long polling -rajapinnan sijaan.

4.4 REST-rajapinta

REST on ohjelmointirajapintojen kehittämisen malli, jossa käytetään lähes ainoastaan HTTP-protokollaa. REST:n avulla voi kehittää yksinkertaisia rajapintoja, joita vasten integroiminen on helppoa [16]. REST:ssä käytetään hyväksi HTTP-kutsujen jokaista metodia, jotta tietueiden luku, luonti, muokkaus ja poisto onnistuu. Missen REST-rajapinnan vuoksi spesifikaatioiden lukeminen ja ymmärtäminen oli huomattavasti helpompaa, ja se vauhdittikin projektin alkua.

Missen tarjoamat rajapinnat mahdollistavat käyttäjä-, ryhmä- ja tapaamistietojen hallinnan. Rajapinta hyväksyy JSON-viestiin pakatun HTTP-yhteydellä lähetetyn paketin ennalta määriteltyihin osoitteisiin (taulukko 1).

Taulukko 1: Missen REST-rajapinnan metodit.

REST Resurssi	URL	Metodit
Meeting List	/rest/v1/meeting/	GET, POST
Meeting	/rest/v1/meeting/<meeting_id>/	GET,PUT, PATCH
Time period list	/rest/v1/meeting/<meeting_id>/timeperiod/	GET, POST
Time period	/rest/v1/meeting/<meeting_id>/timeperiod/<time_period_id>/	GET
Request response list	/rest/v1/meeting/<meeting_id>/response/	GET, POST
Request response	/ rest/v1/meeting/<meeting_id>/response/ <response_id>/	GET, PIT
User list	/rest/v1/user/	GET, POST
User	/rest/v1/user/<user_id>/	GET,PUT,PATCH, DELETE
Group list	/rest/v1/group/	GET, POST
Group	/rest/v1/group/<group_id>/	GET,PUT,PATCH, DELETE

Nykyisellään Missen palvelimeen saa yhteyden ainoastaan salaamattomalla http-yhteydellä ja tunnistautuminen tapahtuu Basic access -autentikaatiota käyttäen. Tämä on käyttäjätunnuksien liikuttamiseen HTTP-yhteyden välityksellä tietoturvan kannalta riittämätöntä. Basicin avulla tunnukset siirretään Base64-koodattuna HTTP:n ylätunnisteessa, joka ei anna suojaa hyökkäyksiä vastaan ilman HTTPS:ää. [17.] Tämä jouduttaisiin muuttamaan, ennen kuin Misseä voitaisiin valjastaa tuotantokäyttöön.

5 Integraation suunnittelu

Suunnitteluvaiheessa piti ottaa huomioon monia asioita ennen työn tekoa. Suunnittelutyötä tehtiin paljon myös projektityön edetessä, mutta suurimmat päätökset tehtiin ennen työn aloittamista.

Ensimmäisessä vaiheessa tutustuin Missen nykyiseen toiminnallisuuteen sen tarjoaman selainkäyttöliittymän avulla. Tästä pidimme samassa yhteydessä työpajan, jossa Missen prototyypin luoja esitteli järjestelmän ominaisuudet ja keskustelimme mahdolli-

sista jatkokehitysmahdollisuuksista ja siitä, mitä asioita pitäisi ottaa huomioon integraatiota tehdessä.

Tämän jälkeen kartoitin Missen tarjoamia REST-rajapinnan metodeita ja niiden arvokenttiä. Niiden perusteella pystyin päättelemään, mitä tietoja rajapinnan kautta voi luoda ja muokata, ja pystyin myös arvioimaan, mitä Salesforcen tarjoamia standardiobjekteja tarvitaan ja mitkä täytyy tehdä täysin räätälöitynä. Tein myös valmiiksi Apexiin REST:ä vastaavat resurssiluokat, jotka sisälsivät valmiin rakenteen ottamaan rajapinnasta tullutta dataa vastaan.

5.1 Tietomalli

Tietomallia suunnitellessa piti ottaa huomioon Salesforcen olemassa olevat yleisesti käytetyt tietomallit. Tämä sen vuoksi, että integraatiopaketin saisi käyttöön helposti myös olemassa oleviin Salesforce-ympäristöihin käyttäen olemassa olevaa kantadataa.

Nämä standarditaulut ovat Salesforcessa asiakkaat (Account), yhteyshenkilöt (Contact), tapahtumat (Event) ja käyttäjätaulu. Näitä standardiobjekteja käytetään lähes poikkeuksetta jokaisessa Salesforce-ympäristössä, joten integraation rakentaminen niiden ympärille on tärkeää. Tämä mahdollistaisi helpon käyttöönoton jo käytössä olevaan Salesforce-ympäristöön.

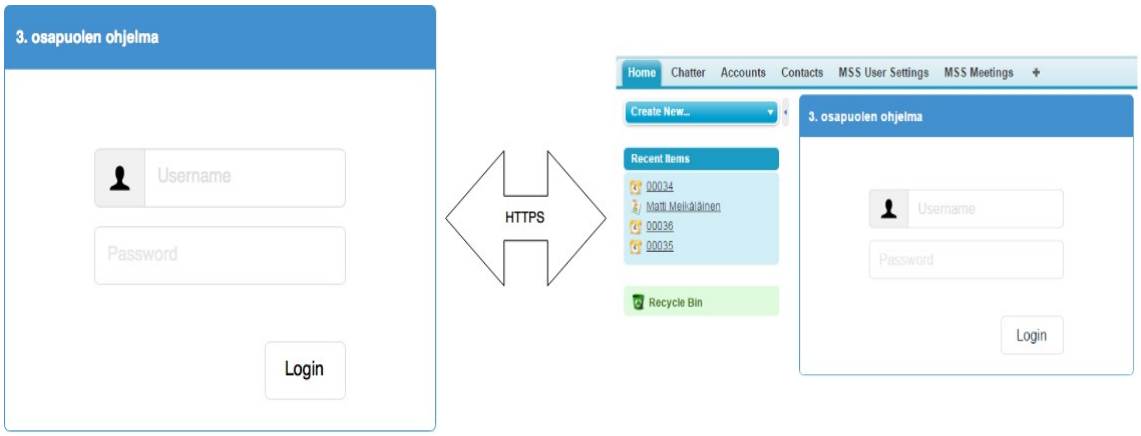
Asiakastauluissa tärkeäksi muodostuvat yhteyshenkilöt, sillä heidän on tarkoitus heijastaa Missen vastaavia käyttäjiä, jotka eivät kuitenkaan ole Salesforce-käyttäjiä. Tällöin täytyy tehdä linkitys yhteyshenkilöiden ja Misse-käyttäjien välillä. Yhteyshenkilöllä voi olla Salesforcessa linkitettynä yksi yritys. Missessä ei vielä tallenneta yritystietoja, mutta jos järjestelmää aiotaan käyttää yritysmaailmassa, tulee tuon tiedon lisäys oleelliseksi.

Käyttäjätaulun tiedot ovat samassa käytössä kuin yhteyshenkilötkin, eli ne linkitetään suoraan Missen käyttäjiin. Jokaista käyttäjää kohden, joka on linkitetty Misseen, on tallennettu omat tunnistautumistiedot erilliseen piilotauluun.

Tapahtumataulu tulee oleelliseksi siinä vaiheessa, kun tapaaminen on luotu Salesforcen kautta ja aikataulut hyväksytty Missessä. Tässä vaiheessa Salesforce loisi uuden tapahtuman Salesforcen tapahtumatauluun, johon on linkitetty tapaamiseen kutsutut Salesforcen käyttäjät ja ulkopuoliset käyttäjät. Tällöin Salesforcen kalenteriin tulevat myös näkyviin nämä tapahtumat.

5.2 Käyttöliittymä

Käyttöliittymää suunnitellessa vastaan tuli kaksi mahdollista toteutusvaihtoehtoa. Ensimmäinen vaihtoehto oli käyttää Salesforcen Canvas-tekniikkaa, jonka tarkoituksena on auttaa integroimaan kolmannen osapuolen sovelluksia Salesforceen. Canvas tarjoaa Force.com Canvas SDK:n, joka on Javascriptillä tehty kirjasto. Tämä kirjasto avustaa ulkopuolisen sovelluksen upottamisen Salesforceen. Salesforce ja ulkoinen sovellus keskustelisivat tällöin automaattisesti HTTPS:n välityksellä välittäen autentikaatiotiedot keskenään ja varmistaen yhteyden luotettavuuden (kuva 7).



Kuva 7: Käyttöliittymän upottaminen Canvas SDK:lla.

Tämä mahdollistaisi Missen käyttöliittymän upottamisen Salesforceen, mikä poistaisi tarpeen kehittää käyttöliittymäalusta mutta vaatisi myös Misse-järjestelmään muutoksia.

Toisena vaihtoehtona oli tehdä käyttöliittymä Visualforcen ja Apexin avulla, jäljitellen Missen käyttöliittymää. Tällöin tiedonvälitys tehtäisiin Missen REST-rajapintoja käyttäen. Tämä vaatisi enemmän tekemistä verrattuna Force.com Canvasiin, sillä käyttöliittymä ja logiikka pitäisi tehdä uudestaan Salesforceen. Tässä on kuitenkin etuna käyttöliittymän muokattavuus ja käytettävyys, sillä Missen käyttöliittymää ei ole suunniteltu siinä mielessä, että sitä voisi upottaa ulkoiseen sovellukseen kätevästi.

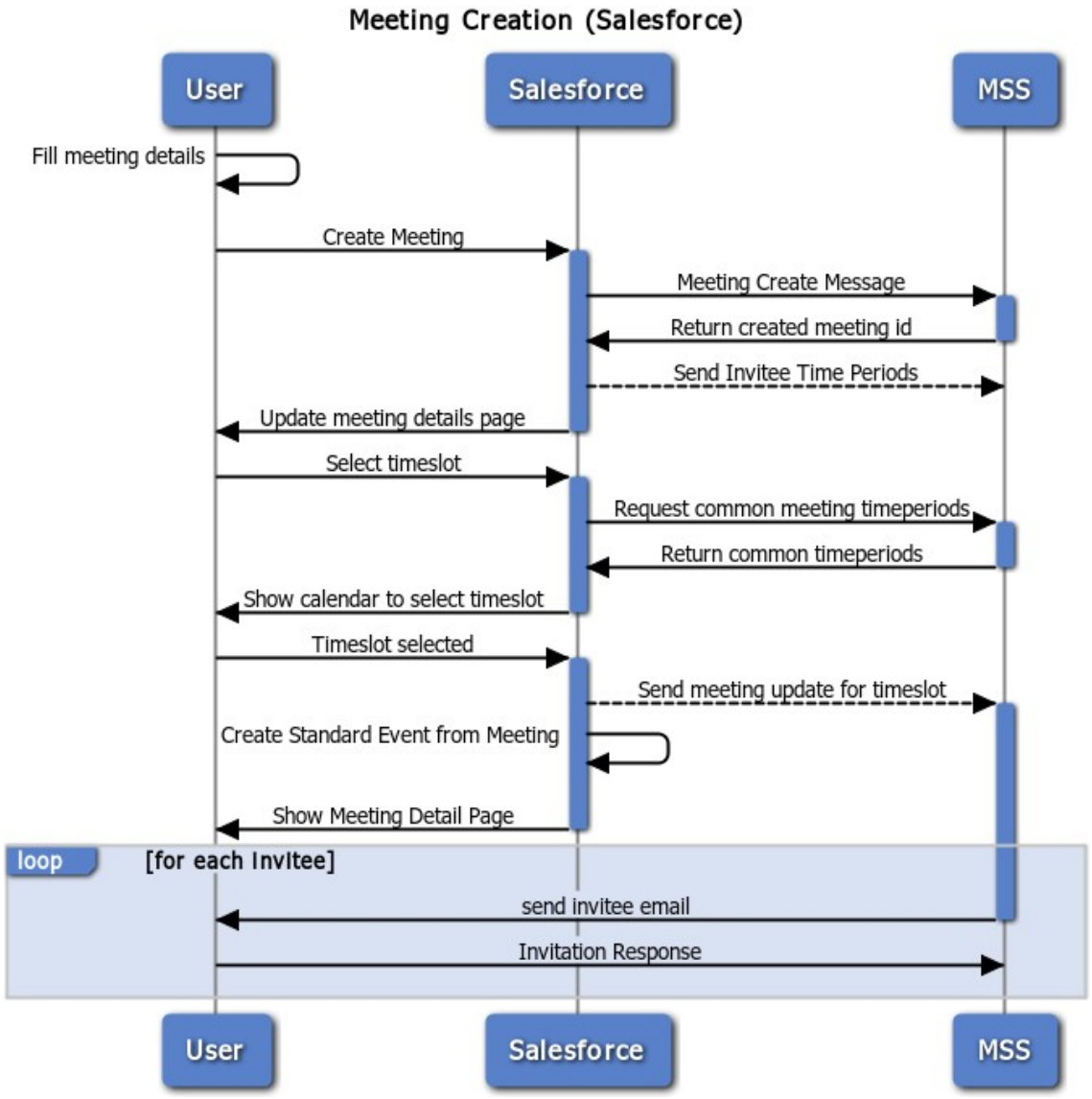
Kahdesta vaihtoehdosta päädyttiin jälkimmäiseen eli oman käyttöliittymän tekemiseen Apexilla ja Visualforcella. Tämä pääosin sen vuoksi, että Canvasilla kehitystä olisi vaadittu myös Missen osalta, mutta sitä ei valitettavasti tämän projektin puitteissa voitu tehdä.

Missen kehittäjä keskittyi jo melko paljon siihen, että käyttöliittymä on hyvin yksinkertainen ja helppo käyttää. Tavoitteena oli, että käyttöliittymä Salesforcessa tulisi olemaan hyvinkin samankaltainen, ja tämä myös varmistaisi sen, että ristiriitaisuuksia toiminnallisuuksien suhteen ei tulisi. Tämän vuoksi Salesforceen tehtävät toiminnallisuudet myötäilisivät mahdollisimman paljon Missen tarjoamaa REST-rajapintaa.

5.3 Tapaamisten luominen

Tapaamisten luomiselle täytyy tehdä uusi käyttöliittymä, joka tukee Missen tarjoamia ominaisuuksia. Salesforcen oma tapaamisten luontisivu ei taivu Missen vaatimuksiin, eikä sitä voi myöskään muokata tarpeeksi, jotta siitä saisi vaaditun kaltaisen.

Tapaamisten luomisprosessi on monivaiheinen, mutta käyttöliittymän kannalta kaksivaiheinen. Ensimmäisessä vaiheessa tapaamisen järjestäjä luo tapaamisen, valitsee sopivan alku- ja loppupäivämäärän ja ajan, tapaamisen pituuden ja kutsutut henkilöt ja lähettää kutsun valituille. Toisessa vaiheessa, kun Misse on selvittänyt jokaisen kutsutun kalenterista sopivat tapaamisajat, voi tapaamisen järjestäjä avata toisen sivun, jossa hän voi valita Missen tarjoamista aikavaihtoehdoista haluamansa. Valitun ajan jälkeen kutsutuille lähtee Missestä sähköposti, josta kutsuttu voi valita tapaamisen hyväksymisen, hylkäämisen tai pyytää uutta tapahtuma-aikaa. Kun tapahtuma on hyväksytty, generoituu Salesforceen uusi Event-tietue, jonka avulla tapaaminen saadaan näkyviin käyttäjien kalenteriin (kuva 8).



Kuva 8: Tapaamisen luontia kuvaava sekvenssikaavio.

5.4 Käyttäjien hallinta

Käyttäjätietoja hallitaan sekä Missessä että Salesforcessa. Missen kirjautumistietoja tarvitaan käyttäjän tunnistautumiseen, ja tämä onnistuu joko verkkoselaimen avulla tai rajapintojen kautta. Koska rajapinnat ottavat kirjautumistiedot vastaan selkokielisenä, täytyy myös Salesforcen puolella pitää käyttäjätunnuksia tallennettuna. Käyttäjätunnukset luotaisiin Misseen admin-käyttäjällä.

Tähän suunniteltiin ratkaisuna oma asetuskäyttöliittymä Salesforceen, jossa käyttäjä voi asettaa Misse-käyttäjätunnukset tai vaihtaa sen toiseen. Itse asetuskäyttöliittymä pitäisi sisällään aluksi vain syöttökentän käyttäjänimelle ja kaksi syöttökenttää, joihin asettaa salasana, ja jatkossa tänne samaan asetusikkunaan voisi laittaa muita käyttäjäkohtaisia Misseen liittyviä asetuksia.

Asetussivu tallentaisi käyttäjätunnukset Salesforcen kantaan salatun avaimen avulla. Salattu avain olisi myös tallennettuna kantaan, johon vain Salesforcen ohjelmakoodi pääsee käsiksi. Itse kirjautumisvaiheessa kryptattu salasana avattaisiin salatun avaimen avulla ja lähetettäisiin HTTPS-yhteydellä basic access -autentikaatiota käyttäen.

5.5 Integraatio Misseen

Käyttäjät

Misse-ajanvarausjärjestelmä tarjoaa käyttäjienhallinnan CRUD-mallin mukaisesti rajapinnan kautta, mikä mahdollistaa käyttäjähallinnan tekemisen ulkoiseen järjestelmään. Tässä tapauksessa tämä kannattaa tehdä myös Salesforceen, jolloin jokainen käyttäjä voi luoda ja muokata omaa Misse-käyttäjätunnustaan Salesforcen käyttöliittymästä. Salesforce admin-käyttäjilläkään ei ole näihin kirjautumistietoihin pääsyä.

Ryhmät

Missessä ryhmät koostuvat käyttäjistä, ja yksi käyttäjä voi kuulua useampaan ryhmään. Missen käyttöliittymässä voi pyytää samassa ryhmissä olevia samaan tapaamisiin, mutta muuten järjestelmä ei estä ryhmän ulkopuolisten ihmisten kutsumista.

Salesforceen ei integraatiossa tehdä ryhmäkohtaista kutsumista, vaan kaikki Salesforce-instanssiin linkitetyt Misse-käyttäjät voidaan kutsua tapaamisiin ryhmätiedoista välittämättä. Tämän pohjalle olisi kuitenkin mahdollista kehittää ryhmäkohtainen kutsuminen ja näkyvyyden rajoittaminen.

Tapaamiset

Tapaamiset ovat järjestelmän keskeisin osa. Tapaamiset jakautuvat kahteen osaan: suunniteltavissa olevaan tapaamiseen ja itse tarkkaan tapahtuma-aikaan sijoittuvaan tapaamiseen.

6 Salesforce-integraatio

6.1 Tietomalli

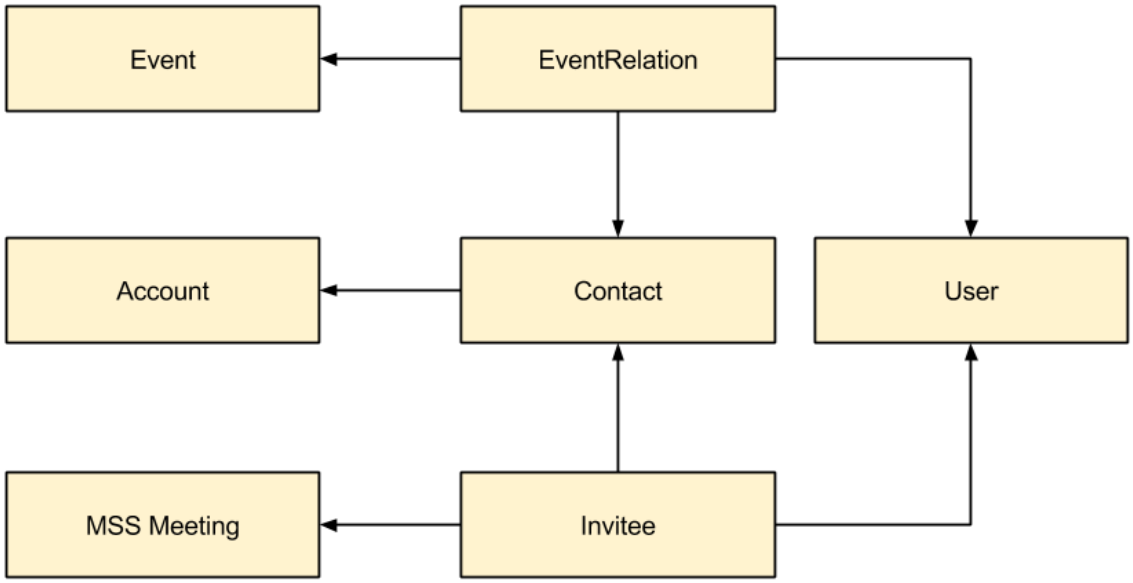
Salesforceen toteutettiin tietomalli seuraavilla standardiobjekteilla: Account, Contact, User, Event ja EventRelation. Mukautettuina objekteina toimivat Mss Meeting sekä Invitee.

Contactit ovat Salesforcen ulkopuolisia yhteystietoja, ja ne voivat olla linkitettyjä myös Missen käyttäjiin. Contacteilla voi olla yksi yritystieto, johon se kuuluu. Yritykset eivät ole Misseen linkitettyjä tietoja, mutta se on tarpeen tullen mahdollista.

User-tietueet ovat Salesforcen käyttäjiä, jotka voivat olla linkitettyjä Missen käyttäjään. Tämä tapahtuu sen jälkeen, kun käyttäjä on luonut Misse-tunnuksensa Salesforcessa. User-objektin taakse on tallennettu Missessä vastaavan user-tietueen tunnus, jota käytetään integraation apuna linkittämään tietue Missen vastaavaan tietueeseen.

Tapaamistiedot on tallennettu standardiobjekteihin Event ja EventRelation ja mukautettuihin objekteihin MSS Meeting sekä Invitee. MSS Meeting sisältää MSS:ssä suunnitteilla olevan tapaamisen tiedot, kuten tapaamisen nimen, halutun pituuden ja sijainnin.

Invitee-objekti on välitaulu, joka linkittää tapahtuman kutsuttuihin henkilöihin ja Salesforce-käyttäjiin. Nämä relaatiotietueet luodaan siinä yhteydessä, kun tapaaminen luodaan, ja sitä luodessa valitaan kutsutut henkilöt. Invitee-tietue sisältää kenttäarvoina muun muassa linkin Missessä vastaavaan kutsutietueeseen ja kutsutun henkilön vastauksen kutsuun. Kun tapaamisen ajankohta on sovittu ja käyttäjät ovat hyväksyneet sen, generoidaan Event-tietue. Siihen tallentuvat muun muassa tapahtuman tarkka ajankohta, sijainti ja muut tiedot. EventRelation on vastaavanlainen kuin Invitee-objektikin, eli se linkittää tapahtuman siihen osallistuneisiin henkilöihin (kuva 9).

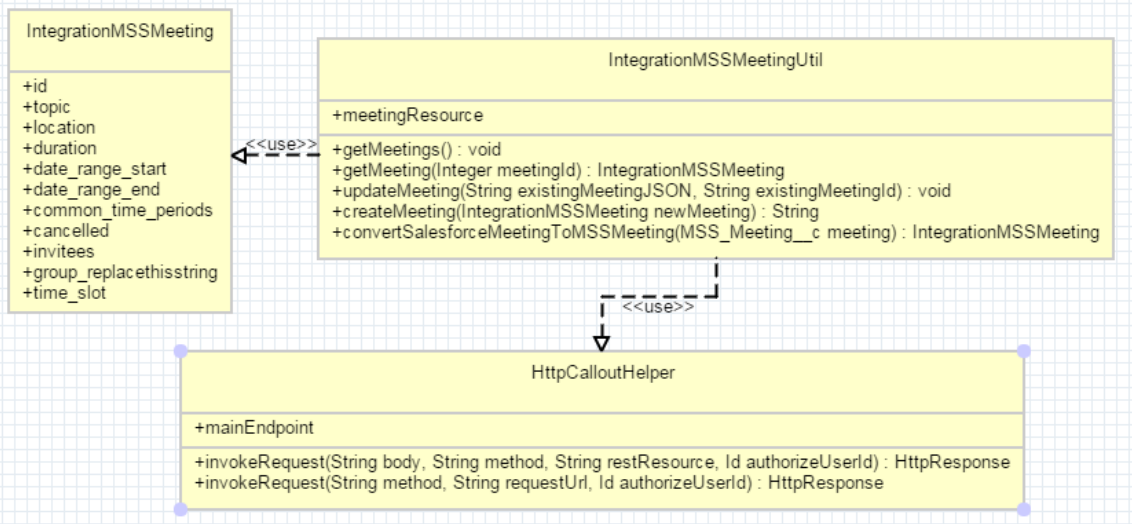


Kuva 9: Salesforcen objektien tietomalli.

6.2 Apex-integraatioluokkien rakenne

Integraatiotyötä aloittaessa työstin ensimmäisenä dataa ”kuljettavat” Apex-luokat kuntoon. Nämä luokat sisältävät ainoastaan kenttiä ja niiden datatyypitykset, jotka loin Missen rajapintakuvausten mukaisesti. Kun dataa vihdoin viedään tai haetaan Missen rajapinnasta, luodaan tarvittavat instanssit näistä integraatioluokista, minkä jälkeen ne serialisoidaan JSON-muotoon, jota Missen rajapinta suostuu ottamaan vastaan.

Jokaiselle resurssille on tehty oma Utility-luokka, jonka kautta Missen rajapintametodien kutsu onnistuu. Nämä luokat mahdollistavat esimerkiksi tapaamisten hakemisen, luonnin tai päivittämisen Missestä sekä avustavat Salesforce tietokantaobjektien muuttamisen integraatiossa käytettäviin instansseihin. Niitä on täten helppo käyttää, ilman että olisi tarvetta jatkossa pureutua syvemmälle rajapinnan toiminnallisuuksiin (kuva 10).



Kuva 10: Kaavio tapaamisten integraatiota avustavasta luokasta.

6.3 Käyttäjätunnusten hallinta

Koska Missen rajapintaa kutsuessa käytetään käyttäjäkohtaisia tunnuksia, täytyy näiden tunnusten olla tallennettuna ja hallittavissa Salesforcessa. Tällöin on hyvin tärkeää, että tiedot ovat salattuna Salesforcen tietokannassa, ilman että niihin pääsee kukaan käsiksi. Salesforce tarjoaa tähän kolmea eri vaihtoehtoa: mukautetut asetukset, Apex-salausluokat ja -metodit sekä salatut tietokentät. [18.]

Mukautetut asetukset

Mukautetut asetukset (Custom settings) ovat käytännössä mukautettuja tietokantatauluja, joiden tyypiksi voidaan laittaa joko lista tai hierarkia sekä tieto, ovatko listat julkisia vai suojattuja. Hierarkkiset asetukset ovat muun muassa Salesforce-käyttäjiin tai -profiileihin linkitettyjä tietueita, kun taas listat ovat linkittämättömiä tietueita. Näihin asetuksiin voidaan lisätä samalla tavalla eri kenttätyyppejä kuin Salesforcen mukautettuihin objekteihinkin. Julkisten asetusten avulla voidaan tarjota esimerkiksi staattista dataa käytettäväksi mukautetuille Visualforce-sivuille, käyttäjästä tai profiilista riippuen tai riippumatta.

Suojattujen mukautettujen asetuksien käyttäminen hallittujen asennuspakettien kanssa mahdollistaa käyttäjätunnusten tai muun yksityisen tiedon salauksen [18]. Kun asennuspaketti on asennettu Salesforce-instanssiin, voi suojattuihin asetuksiin päästä käsiksi ainoastaan suoritetun Apex-koodin avulla, joka kuuluu vastaavaan asennuspakettiin.

Tunnusten salaus

Apexissa käytettävän Crypto-luokan metodit avustavat tiedon salauksessa ja autentikoinnissa ulkoisiin järjestelmiin. Metodien avulla voidaan muun muassa luoda tiivisteitä,

salata tietoa tai purkaa salauksia. Salausmetodien käyttö vaatii myös salausavaimen tallentamisen Salesforcen suojattuun asetukseen. [18.]

Salausmetodit yhdessä suojattujen asetuksien ja hallittujen pakettien kanssa mahdollistaisivat turvallisen kokonaisuuden tallentaa ja käyttää Misse-käyttäjätunnuksia. Yksi parhaista ja käytetyimmistä salaustavoista olisi tiivisteen luonti salasanasta siihen tarkoitetulla algoritmilla. Tässä tapauksessa kuitenkin Missen rajapinta ottaa vastaan tunnukset lähes selkokielellä, minkä vuoksi salauksen täytyy olla kaksisuuntainen.

Salatut tietokentät

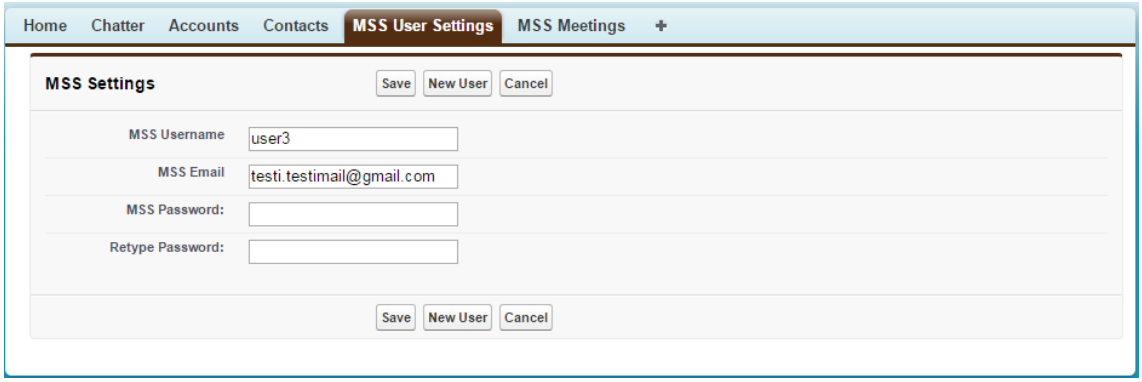
Salatut tietokentät ovat melko uusi ominaisuus Salesforcessa. Niiden avulla on mahdollista luoda objekteihin kenttiä, joihin syötetyt arvot ovat salattuja 128-bittisillä avaimilla AES-algoritmilla. [18.] Kenttään syötettyihin arvoihin pääsee käsiksi tietyllä Salesforcen admin-käyttäjäoikeudella, minkä vuoksi tämä ei olisi tietoturvan kannalta aukoton ja turvallisin tapa tallentaa käyttäjätunnuksia. Nämä kentät soveltuvatkin parhaiten arkaluontoisemman tiedon piilottamiseen, kuten henkilötunnusten tai luottokorttitietojen.

6.4 Käyttöliittymä

Integraatiota varten tehdyt käyttöliittymät tehtiin omina sivukomponentteinaan Visualforcella, käyttäen apuna HTML:ää, CSS:ää ja Javascriptia JQueryn kanssa. Joitakin Salesforcen standardi luomis- ja muokkaamissivuja on korvattu näillä, kuten tapaamisten luontisivu.

6.4.1 Käyttäjähallintasivu

Käyttäjähallintasivulla on mahdollista luoda uusi tunnus Misseen, minkä jälkeen käyttäjä voidaan kutsua Misse-tapahtumiin. Käyttäjäsivulla on tämän jälkeen mahdollista muuttaa ainoastaan käyttäjän nimitiedot tai sähköpostiosoite. Salasanaa ei ole mahdollista vaihtaa, sillä Missen rajapinta ei tätä mahdollista. Hallintasivulta on myös mahdollista luoda aikaisemman tunnuksen tilalle uudet tunnukset, mikä korvaa vanhan tunnuksen Salesforcessa (kuva 11).



Kuva 11: Käyttäjätietojen luonti ja muokkaus.

Kun käyttäjätunnukset luodaan, lähtee automaattinen luontiviesti Missen rajapintaan. Jos käyttäjän luonti onnistui, tulee pyyntöviestiin vastauksena Misseen luodun käyttäjän tunnus, joka laitetaan Salesforcessa käyttäjän kenttään User ID. Päivitettäessä olemassa olevaa käyttäjää käytetään tunnusta yhdistämään kyseinen käyttäjä Misseen.

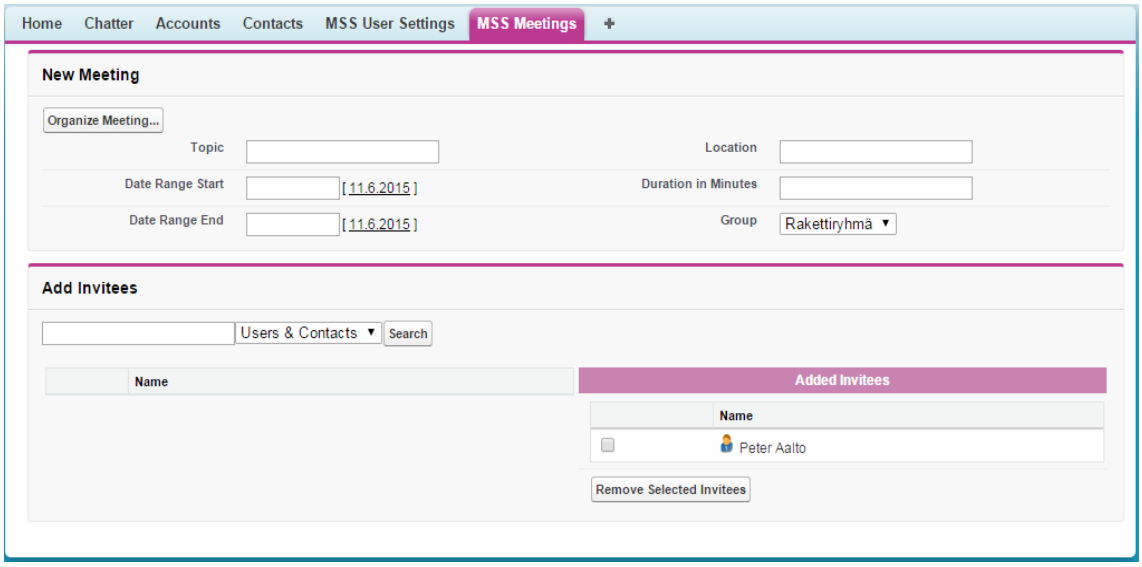
6.4.2 Tapaamisen luontisivu

Tapaamisen luontisivu on räätälöitynä tehty Visualforce-sivu, johon Salesforcella ei ole tarjota mitään standarditoiminnallisuutta, joka tukisi vaadittuja käyttötapauksia.

Tapaamisen luontisivu on kaksiosainen (kuva 12). Ylemmässä osassa täydennetään yleiset tapaamiskohtaiset tiedot, kuten nimi, tapaamisen pituus ja alku- ja loppupäivämäärä. Tapaamisen päivämäärätiedot määrittävät, miltä väliltä osallistujilta kerätään

kalentereista vapaat ajat ja lähetetään Misseen. Tapaamisen kesto taas määrittää, kuinka isoja tapaamisaukkoja käyttäjien kalentereista haetaan.

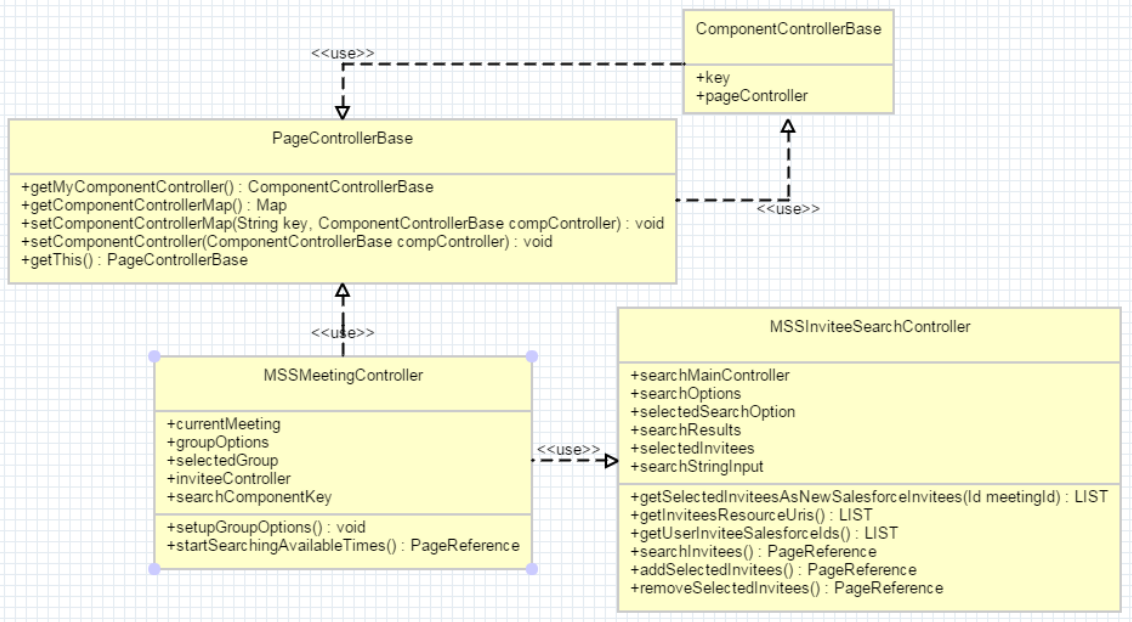
Alemmassa osiossa lisätään tapahtumaan kutsuttavat henkilöt hakemalla heitä nimen perusteella. Tätä hakua voi kohdentaa joko hakemalla vain ulkoisia Misse-yhteyshenkilöitä tai sisäisiä Salesforce-käyttäjiä. Haun tuloksista on tämän jälkeen mahdollista valita henkilöitä rastittamalla ja painikkeesta lisäämällä, jolloin ne listautuvat hakutulosten oikealle puolelle. Salesforcen käyttäjiä ja yhteyshenkilöitä ei voi tulla hakuvastauksina, jos niitä ei ole integroitu Misseen.



Kuva 12: Tapaamisen luontisivu.

Kun käyttäjä vihdoin luo tapaamisen, kokoaa Salesforce tapaamisen aikatietojen perusteella kutsuttujen kalentereista vapaat ajat. Tietojen keräyksessä kootaan kutsuttujen vapaat ajat, joihin tapaamisen kesto mahtuisi. Tämän jälkeen itse tapaamistietue lähetetään Misseen annettujen tietojen kanssa. Jos lähetys onnistuu, tulee vastauksena Misseen luodun tapaamisen tunnus, joka tallennetaan Salesforceen. Tapaamisen luonnin jälkeen lähetetään jokaisen kutsutun käyttäjän vapaat aikajaksot erillisenä http-kutsuna Misseen. Niiden avulla Misse kykenee laskemaan jokaisen käyttäjän vapaiden aikatietojen avulla kaikille yhteiset vapaat ajat. Kutsutut käyttäjät tallennetaan myös Salesforceen erillisinä tietueina, joihin myös tallennetaan lähetettyjen aikatietojen Misse-tunnukset.

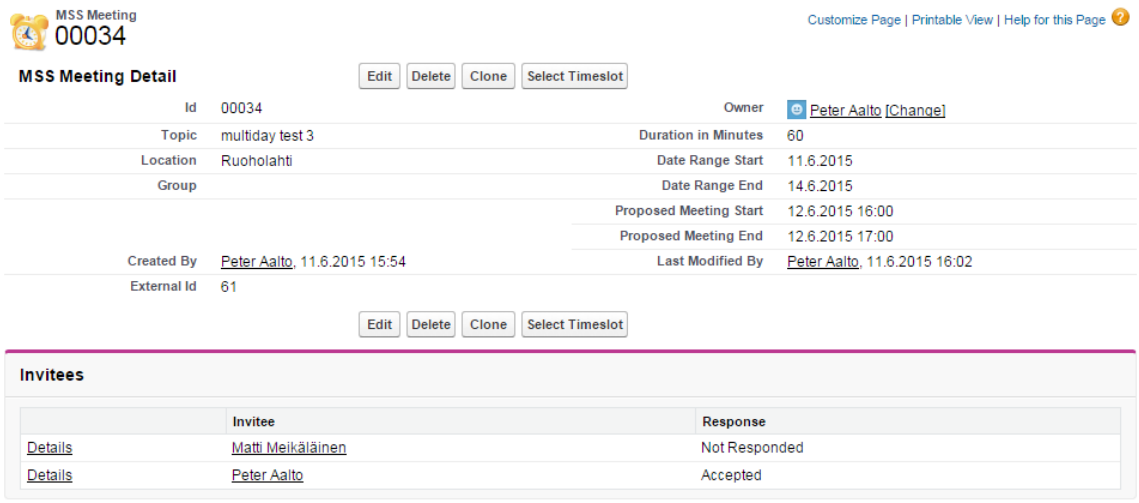
Tapaamisen luontinäkymä ja taustalla toimiva integraatiologiikka vaati muihin Visualforce-näkymiin verrattuna eniten työtä. Luontinäkymän yläosa on toiminnallisuudeltaan hyvin yksinkertainen ja sisältää vain kenttäviittaukset tapaamisobjektiin. Alanäkymä on taas luotu Visualforce-komponenttina, mikä mahdollistaa hakunäkymän uudelleenkäytön tarvittaessa toisilla Visualforce-sivuilla. Tämä tapa erotella Visualforce-komponentteja mahdollistaa myös taustalla olevan Apex-koodin loogisen erottelun muusta toiminnallisuudesta (kuva 13). Tässä käytetään hyväksi myös suunnittelumallia, joka mahdollistaa yhteyden luonnin pääsivun kontrollerin ja komponentin kontrollerin välille. [19.] Tapaamisten luonnissa voi tämän avulla saada komponentista pääkontrollerille tarvittaessa tapaamiseen haetut ja kutsutut käyttäjät.



Kuva 13: Tapaamiskontrollerin suhde hakukomponenttiin.

6.4.3 Tapaamisten yksityiskohdat

Kun tapaaminen on luotu, avautuvat käyttäjälle tapaamisen tietosivut (kuva 14), joista näkee aikaisemmin syötetyt tiedot sekä muut toiminnot mahdollistavat painikkeet. Koko sivu on Visualforce-sivu, joka koostuu tapaamisen perustiedoista ja tapaamiseen kutsutuista henkilöistä. Joka kerta, kun käyttäjä avaa tapaamisen tiedot, tehdään Misseen rajapintakutsu, joka hakee ajantasaisen tiedon tapaamiseen kutsutuista henkilöistä. Tällöin käyttäjä näkee heti, kuinka moni kutsutuista on hyväksynyt tai hylännyt tapaamiskutsun.



Kuva 14: Tapaamistietosivu.

Kun kutsuttujen tiedot on haettu tapaamisen luonnin jälkeen, on järjestäjän mahdollista mennä valitsemaan itse tapaamisen aika järjestelmän ehdottamista aikavaihtoehdoista. Tähän on tehty räätälöity aikavalinta-sivu (kuva 15), josta voi valita Missen laskemia aikoja, jolloin kaikilla osanottajilla pitäisi olla vapaata aikaa kalentereissaan. Ajat ovat eritelty päivämäärän mukaan, jokaisen päivän vapaat aikalohkot allekkain. Kun päivä on valittu, on sille mahdollista asettaa tarkempi aika. Nyt tapaamisajan valinta mahdollistaa minkä pituisen aikajakson tahansa alkumäärityksistä huolimatta, mutta tähän voi mahdollisesti kehittää validoinnin, jotta muun pituiset aikajaksot eivät olisi mahdollisia.

Kuva 15: Tapaamisajan valinta.

Tapaamisajan tietojen hakemisessa tuli vastaan yksi suurimmista ongelmista ja esteistä projektin etenemisen kannalta. Kuten aikaisemmin todettiin, nykyinen Missen prototyyppi ei tue muuta kuin mobiilikäyttöä mainiosti. Tämän vuoksi tapa lähettää Salesforcesta käyttäjien vapaita aikatietoja ei laukaise Missessä yhteisten kalenteriaikojen laskentaa. Tämä toimii ainoastaan, jos Missen selainkäyttöliittymästä käy manuaalisesti painamassa vapaiden aikatietojen ehdotusta, minkä jälkeen Misse suostuu laskemaan vapaan aikatiedon. Vasta tämän jälkeen saa rajapinnan kautta kaikkien käyttäjien yhteiset vapaat ajat. Ainoa mahdollisuus tämän ongelman korjaamiseen olisi Missen logiikan muuttaminen, mitä ei kuitenkaan voitu tehdä tämän projektin puitteissa.

Kun tapaamisen järjestäjä on valinnut tapaamiselle ajan, lähtee tästä tallentaessa tieto Misseen. Misse lähettää tämän jälkeen jokaiselle kutsutulle henkilölle sähköpostiviestin, josta jokainen voi valita tapaamisen hyväksymisen, hylkäämisen tai uudelleen ajoittamisen sähköpostissa tulleiden URL:ien avulla. URL:ssä on id-viittaukset tapahtuman kutsuun sekä parametreina tapaamiseen valittu vastaus. Tämän jälkeen tapaamisen järjestäjä voi seurata tapaamisen vastauksia joko Salesforcesta tapaamistietosivulta tai Missen selainkäyttöliittymästä ja päättää jatkotoimista.

6.5 Yksikkötestit

Jotta integraatiopakettia voisi valjastaa tuotantokäyttöön, pitäisi testien kattaa ainakin 75 % käytettävästä koodista. Salesforce ei voi kuitenkaan vaikuttaa testiluokkien laatuun, eli tämä on täysin kehittäjän vastuulla. Yksikkötestien tarkoitus ei ole vikojen etsintä tai selvitys, vaan tapa pitää koko sovellus määrittelyiden mukaisena ja estää mahdollisesti uusien vikojen syntyminen [20]. Kun sovellusta kehitetään ja testit hajoavat, ne kertovat suoraan kehittäjälle, mikä osa sovelluskoodista ei toimi, jolloin ongelman korjaus on nopeaa.

Testaaminen oli projektin aikana pääosin manuaalista testausta, ja koska projekti muutenkin viivästyi, en käyttänyt paljoa aikaa yksikkötestien tekemiseen. Integraation testaamiseen tein kuitenkin yksikkötestipohjan, jonka avulla saataisiin rakennettua testit

myös muille rajapintakutsumetodeille. Web service -rajapintoja testaavat testit eivät käytännössä ota yhteyttä Missen rajapintaan, vaan niille tehdään mock-luokka, joka imitoi rajapintaa (kuva 16). Yksikkötestissä määritetään kyseinen mock-luokka esittämään rajapintaa, joka palauttaa HTTP-kutsuihin vastauksen. Kun yksikkötestissä käytetään rajapintaa kutsuvia metodeja, saadaan näin aina testivastauksia testejä varten. Tässä tapauksessa palautettavien JSON-viestien pitää olla vastaavia, kuin mitä Missen oikea rajapinta palauttaisi.

```
@isTest
private class IntegrationMSSGroupUtilTest
{
    static testMethod void testGettingAllGroups_ShouldReturnListOfGroups()
    {
        Test.setMock(HttpCalloutMock.class, new IntegrationMssHttpMock());
        Test.startTest();

        List<IntegrationMSSGroup> mssGroups = IntegrationMSSGroupUtil.getGroups();

        Test.stopTest();

        system.assert(mssGroups.size() > 0);
    }
}

global class IntegrationMSSHttpMock implements HttpCalloutMock
{
    global HTTPResponse respond(HttpRequest request)
    {
        String requestURL = request.getEndpoint();
        HttpResponse response;
        if(requestURL.contains('group/'))
        {
            response = getGroupResponse(request);
        }

        return response;
    }

    private HttpResponse getGroupResponse(HttpRequest request)
    {
        HttpResponse groupResponse = new HttpResponse();
        if(!String.isEmpty(request.getMethod()) && request.getMethod().equals('GET'))
        {
            groupResponse.setBody(groupListResponseBody);
            groupResponse.setStatusCode(200);
        }

        return groupResponse;
    }

    private static final String groupListResponseBody = '{"meta": {"limit": 20, "next"
}
```

Kuva 16: Ryhmäintegraation testiluokka ja mock-luokka.

Kun testiluokkien ajoja suoritetaan, Salesforce ilmoittaa, kuinka monta tietyn luokan testimetodeista menee läpi ja kuinka moni epäonnistuu. Epäonnistuessa pääsee tarkemmin suorituksen tietoihin käsiksi, mikä kertoo suoraan, mikä Assert-kutsuista ei mennyt lävitse ja onko koodissa tapahtunut poikkeuksia. Suoritusten onnistuessa voi Salesforcen developer consolesta tarkistaa jokaisen Apex-luokan kattaman testiosuuden. Avaamalla luokan tiedot pääsee myös rivikohtaisesti näkemään, mitkä koodirivit testit ovat kattaneet (kuva 17).

```
public static void createGroup(IntegrationMSSGroup newGroup)
{
    String newGroupJson = JSON.serialize(newGroup);
    HttpResponse response = HttpCalloutHelper.invokeRequest(newGroupJson, 'POST',
}

public static List<IntegrationMSSGroup> getGroups()
{
    HttpResponse response = HttpCalloutHelper.invokeRequest(null, 'GET', groupRest
    List<IntegrationMSSGroup> groups = new List<IntegrationMSSGroup>();
    if(response.getStatusCode() == 200)
    {
        groups = JSONParserUtil.convertJsonToMSSGroupList(response.getBody());
    }
    return groups;
}
```

Kuva 17: Testien kattama koodi sinisellä, testaamaton punaisella.

Testien kattama koodi on projektissa siis lähes olematon. Jos ja kun uusi versio Missestä valmistuu ja siihen integroidaan Salesforce, pitäisi suurin osa koodista todennäköisesti kirjoittaa uusiksi. Vasta tällöin yksikkötestit kannattaisi tehdä kunnolla.

6.6 Työhön käytetty aika

Työpaikkani tarjosi tätä insinöörityötä alustavasti vuoden 2013 kesäkuun paikkeilla, jolloin projekti myös aloitettiin. Ensimmäiset kolme kuukautta meni suunnittelupalavereissa, Missen spesifikaatioiden lukemisessa ja itse projektin suunnittelussa. Tämän jälkeen aloitin työn ympäristön pystyttämisellä ja pohjatyön tekemisellä. Viimeinen muutos työhön tehtiin lähes tasan kaksi vuotta aloituksen jälkeen, jolloin todettiin, että työ

on tavoitteeseensa nähden valmis. Yhteensä kaikkiaan työtunteja työhön ja siihen liittyviin palavereihin kului minulta yhteensä noin 188 tuntia (taulukko 2). Suurin osa käyttöliittymäkehitykseen käytetystä ajasta kului tapaamisen luomisnäkymään ja tapaamisajan valintasivuun. Integraatiotyö vei taas liikaa aikaa vain tapaamisajan integrointiongelman selvittämisen vuoksi. Osa palavereihin käytetystä ajasta oli myös Missen Java-toteutuksen suunnittelua. Raportoin integraatiota tehdessä mahdollisista kehityskohteista ja ongelmista, joita korjattiin hiljalleen uuteen Misseen.

Taulukko 2: Eri työtehtäviin kulutettu aika.

Tehtävä	**Kulutettu aika tunteina**
Keskustelut & palaverit	20,72
Integraatiotyö	78,67
Käyttöliittymäkehitys	55,5
Virheiden korjaukset	21,5
Muut tehtävät	11,75
Yhteensä	188,13

Apex-luokkia oli lopullisessa työssä 37, joissa yhteensä 1 933 riviä Apex-koodia, noin 58 000 kirjaimen verran. Suurin osa Apex-koodia liittyy integraatioon ja loppuosa Visualforce-sivujen takana olevaan logiikkaan. Visualforce-sivuja oli yhteensä vain 6, mukaanlukien yksi komponentti. Testiluokkien kattava teko olisi vienyt tämän lisäksi kymmeniä tunteja.

7 Yhteenveto

Salesforce-integraatio saatiin valmiiksi määrittelyiden mukaisesti, lukuun ottamatta täysin toimivaa integraatiota. Salesforce-integraation tarkoituksena olikin selvittää, onko integraatio mahdollinen ja minkälaisia parannuksia Salesforceen saataisiin tehtyä nykyisen tapaamisten järjestämisen tilalle. Projektin keskivaiheilla huomattiin prototyypin kankeus integroitaessa sitä monen käyttäjän järjestelmiin, minkä vuoksi aloitettiin projektin sivuhaara, jossa kehitettiin uutta ja parempaa versiota Missestä Javalla. Tähän uuteen versioon tehtäisiin tämän jälkeen integraatio Salesforceen, ja sen pohjana käytettäisiin tämän projektin aikana syntynyttä integraatiototeutusta. Nykyiseen Misse-ver-

sioon erona olisi järjestelmäkohtaisen integraatiologiikan siirtyminen osittain myös Misseen. Tämä tarkoittaisi suurempaa työtä Missen osalta, mutta poistaisi integraatiopalikoiden tarpeen tehtäessä integraatioita monen käyttäjän järjestelmiin. Tällöin valittaisiin alustavasti käytetyimpiä kalenterijärjestelmiä, joihin integraatio tehtäisiin, ja tämän kautta nähtäisiin ajanhallintajärjestelmän todellinen tarve.

Jatkokehitysmahdollisuuksia Salesforcen kannalta olisi monia, ja ne vaatisivat sitä ennen myös käyttöliittymän laajaa testausta ja parantelua. Muutamissa käyttötapauksissa huomasin nopeasti ärsyttäviä ominaisuuksia, kuten nappien epäintuitiiviset sijainnit ja muita epäloogisuuksia. Kehittäessä en panostanut niin paljoa käyttöliittymäpuoleen, ja eniten aikaa menikin integraatiokoodia tehtäessä ja testattaessa. Myös testiluokat jäivät lähes kokonaan tekemättä, koska nykyinen toteutus tulisi muuttumaan sen verran jyrkästi muutettaessa integraatio uuteen Misseen.

Tietoturvan kannalta on myös paljon kehitettävää. Tämänhetkinen toteutus on ainakin Salesforcen tietoturvan kannalta ajateltu loppuun, sillä salasanat salataan ja piilotetaan mukautettujen asetusten taakse. Kuitenkin yhteydet Missen rajapintaan tapahtuvat edelleen salaamattoman HTTP-yhteyden kautta, mikä vaatii vielä kehittämistä. Tämä otetaan huomioon Missen Java-toteutuksessa.

Lähteet

1 Salesforce Appexchange. Verkkodokumentti. FinancialForce.com. <http://www.financialforce.com/archive/whitepapers-and-ebooks/salesforce-platform/salesforce-appexchange/> Luettu 8.10.2015.

2 What is CRM? 2014. Verkkodokumentti. Really Simple Systems. <http://www.reallysimplesystems.com/faq/what-is-crm/> Luettu 12.10.2015.

3 SaaS CRM and Cloud CRM Software Evolution. Verkkodokumentti. Online-crm.com. <http://www.online-crm.com/saas_crm_evolution.htm> Luettu 9.7.2015.

4 A brief History of Customer Relationship Management. 2013. Verkkodokumentti. CRM Switch. <http://www.crmswitch.com/crm-industry/crm-industry-history/> Luettu 10.7.2015.

5 Gartner CRM Market Share Update. 2015. Verkkodokumentti. Forbes. <http://www.forbes.com/sites/louiscolumbus/2015/05/22/gartner-crm-market-share-update-47-of-all-crm-systems-are-saas-based-salesforce-accelerates-lead/> Luettu 10.7.2015.

6 What Is 'SaaS' (Software as a Service)? Verkkodokumentti. About.com. <http://netforbeginners.about.com/od/s/f/what_is_SaaS_software_as_a_service.htm> Luettu 13.10.2015.

7 Understanding the Cloud Computing Stack: SaaS, PaaS, IaaS. 2013. Verkkodokumentti. Rackspace. <http://www.rackspace.com/knowledge_center/whitepaper/understanding-the-cloud-computing-stack-saas-paas-iaas> Luettu 13.10.2015.

8 Appexchange. Verkkodokumentti. Salesforce.com. <https://appexchange.Salesforce.com/> Luettu 17.7.2015.

9 Force.com drives faster development. 2009. Verkkodokumentti. Nucleus Research. <https://www.Salesforce.com/assets/pdf/analysts/Nucleus_Force.com_drives_faster_development.pdf> Luettu 31.8.2015.

10 Model-view-controller. 2015. Verkkodokumentti. Wikipedia. <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller> Luettu 31.8.2015.

11 Apex Code. 2015. Verkkodokumentti. Salesforce.com. <https://developer.Salesforce.com/page/Apex> Luettu 10.7.2015.

12 What is Apex? 2015. Verkkodokumentti. Salesforce.com. <https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_intro_what_is_apex.htm> Luettu 16.6.2015.

13 Namespace definition. 2005. Verkkodokumentti. TechTarget. <http://search-soa.techtarget.com/definition/namespace> Luettu 13.10.2015.

14 An Introduction to Visualforce. 2014. Verkkodokumentti. Salesforce.com. <https://developer.Salesforce.com/page/An_Introduction_to_Visualforce> Luettu 10.7.2015.

15 Visualforce: an overview. 2015. Verkkodokumentti. Salesforce.com. <https://developer.Salesforce.com/page/Visualforce:_An_Overview> Luettu 16.6.2015

16 Learn REST: A Tutorial. 2008. Verkkodokumentti. M. Elkstein. <http://rest.elkstein.org/> Luettu 20.9.2015

17 Basic access authentication. 2015. Verkkodokumentti. Wikipedia. <https://en.wikipedia.org/wiki/Basic_access_authentication> Luettu 10.7.2015

18 Storing Sensitive Data. 2015. Verkkodokumentti. Salesforce.com <https://developer.Salesforce.com/page/Secure_Coding_Storing_Secrets#Apex_and_Visualforce_Applications> Luettu 17.6.2015.

19 Controller Component Communication. 2010. Verkkodokumentti. Salesforce.com <https://developer.Salesforce.com/page/Controller_Component_Communication> Luettu 12.9.2015

20 The Beginner's Guide to Unit Testing: What Is Unit Testing? 2012. Verkkodokumentti. Envato. <http://code.tutsplus.com/articles/the-beginners-guide-to-unit-testing-what-is-unit-testing--wp-25728> Luettu 13.10.2015

21 Salesforce.com Unveils Force.com Cloud Computing Architecture. 2008. Verkkodokumentti. EWeek. <http://www.eweek.com/c/a/Enterprise-Applications/Salesforcecom-Unveils-Forcecom-Cloud-Computing-Architecture> Luettu 12.10.2015.