KARELIA-AMMATTIKORKEAKOULU
Tieto- ja viestintätekniikan koulutusohjelma

Roope Turunen
Santeri Virnes

OHOS JA MICROSOFT DYNAMICS CRM -JÄRJESTELMIEN VÄLINEN RAJAPINTASOVELLUS

Opinnäytetyö
Helmikuu 2019

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Helmikuu 2019
Tieto- ja viestintätekniikan koulutusohjelma

Tikkarinne 9
80200 JOENSUU
+358 13 260 600

Tekijät
Roope Turunen, Santeri Virnes

Nimeke
OHOS ja Microsoft Dynamics CRM -järjestelmien välinen rajapintasovellus

Toimeksiantaja
Karelia-ammattikorkeakoulu

Tiivistelmä

Tässä opinnäytetyössä tutkittiin ohjelmointirajapintoja, sekä niihin liittyviä tekniikoita ja käsitteitä. Näiden tietojen pohjalta toteutettiin kaksisuuntainen rajapinta Karelia-ammattikorkeakoulun käyttämän Microsoft Dynamics CRM 2015 -järjestelmän sekä opinnäytetyön ja harjoittelun ohjaussovelluksen (OHOS) välille siten, että rajapinnan käyttäjä voi määritellä järjestelmien välillä siirrettävät tietotyypit sekä tiedonsiirron ajastuksen. Rajapinta toteutettiin asiakkaalta saadun vaatimusmäärittelyn pohjalta.

Opinnäytetyön tutkimusvaiheessa perehdyttiin rajapinnan toteutuksen kannalta tärkeisiin tekniikoihin ja käsitteisiin, kuten sovelluskehityspaketteihin sekä erilaisiin rajapintaratkaisuihin. Lisäksi teoriaosuudessa käytiin läpi asiakkaalta saatu vaatimusmäärittely. Teorian lisäksi tutkimusvaihe sisältää molempien rajapintasovellukseen liitettävien järjestelmien yksityiskohtaisen läpikäynnin.

Toteutusvaiheessa perehdyttiin rajapintasovelluksen toteutukseen käymällä läpi itse rajapintasovelluksen käyttöliittymä ja toiminnallisuus. Toteutusvaiheen loppupuolella käytiin läpi Karelia-ammattikorkeakoulun käyttämän Microsoft Dynamics CRM 2015 -järjestelmän konfigurointi ja rajapintasovelluksen kannalta tärkeä uusien entiteettien luominen kyseiseen järjestelmään.

Kieli
suomi

Sivuja 53
Liitteet 1
Liitesivumäärä 5

Asiasanat
ohjelmointirajapinta, MVC-arkkitehtuuri, asiakkuudenhallinta, Microsoft Dynamics CRM

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
February 2018
Degree Programme in Information and Communications Technology

Tikkarinne 9
80200 JOENSUU
FINLAND
+ 358 13 260 600

Authors
Roope Turunen, Santeri Virnes

Title
Application programming interface between OHOS and Microsoft Dynamics CRM applications

Commissioned by
Karelia UAS

Abstract
This thesis explores application programming interface (API) and its related technologies and concepts. Based on studies mentioned above a two-way interface was implemented between Microsoft Dynamics CRM 2015 and OHOS (application for handling of thesis and practical training), both used by Karelia UAS, in a way that allows the user to configure the transferred entities and timing of said transfer inside the application. This interface was implemented based on the requirement specifications given by the client.

In the research stage of this thesis, technologies and concepts that are important to the implementation were familiarized with, such as software development kits and different kinds of interface solutions. In addition, the theory part deals with the client's requirement specification. Besides theory, the research stage includes detailed specification of both Microsoft Dynamics CRM and OHOS applications.

The implementation stage went through the functionality and user interface of the API. The last part of the implementation stage addressed the functionality and configuration of Microsoft Dynamics CRM 2015 software, used by Karelia UAS, as well as configuring new entities, which are vital to the application programming interface.

Language	Pages 58
Finnish	Appendices 1
	Pages of Appendices 5

Keywords
application programming interface, MVC architecture, customer relationship management, Microsoft Dynamics CRM

Sisältö

Termit ja lyhenteet

API	Application Programming Interface, ohjelmointirajapinta, jonka avulla kahden sovelluksen tai palvelun välille saadaan luotua yhteys tiedonsiirtoa varten.
ASP.NET	Microsoftin kehittämä ohjelmistokehys, jonka avulla voidaan luoda web-sivuja, -ohjelmia ja -palveluita.
BDD	Behavior Driven Development, käyttäytymisvetoinen ohjelmistokehitys, vastaa suurimmaksi osaksi TDD:n periaatteita, mutta toiminnallisuuden sijaan keskitytään ensi kädessä ohjelmiston käyttäjän oletettuun käyttäytymiseen.
C#	.NET –kirjastoa käyttävä ohjelmointikieli.
CSS	Cascading Style Sheets, HTML-ohjelmointiin liittyvät tyyliohjeet, joilla web-sivuille luodaan yksilölliset tyylit
CRM	Customer Relationship Management, käsitteellä tarkoitetaan yleensä sovelluksia, joiden avulla yritykset saavat hallittua mm. asiakassuhteitaan sekä myyntejä ja muita tuotannollisia operaatioita.
Debuggeri	Työkalu, jota käytetään ohjelmoinnissa virheiden etsintään ja korjaamiseen.
Editori	Tekstinmuokkausohjelma, jolla voidaan muokata esim. HTML-koodia. Esimerkiksi Notepad++ on tällainen ohjelma.
HTML	Hypertext Markup Language, ohjelmointikieli, jota käytetään verkkosivujen ulkoasun luomiseen.
ID	Identifier, ohjelmoinnissa usein käytettävä tunniste, jolla esimerkiksi käyttäjät saadaan yksilöityä ilman, että syntyy vaaraa duplikaateista järjestelmässä.
IDE	Integrated Development Environment, ohjelmointiympäristö, jonka avulla sovelluskehittäjä voi suunnitella ja toteuttaa ohjelmistoja tietylle alustalle.
IoT	Internet of Things, asioiden internet, tarkoittaa internet-teknologian viimeisintä kehitystä, jossa useimmat kodinkoneet ja mobiililaitteet halutaan liittää osaksi internetiä, jotta niitä voidaan ohjata ja mittailla verkon yli.
JavaScript	Pääasiassa verkkosivujen kehityksessä käytettävä ohjelmointikieli, jonka avulla sivustoille saadaan lisättyä dynaamisia toiminnallisuuksia.

MVC	Ohjelmistoarkkitehtuuri, joka muodostuu kolmesta osasta: mallista, joka hoitaa tiedon tallentamisen ja käsittelyn, näkymästä, jolla kuvataan järjestelmän käyttöliittymä, sekä kontrollerista, joka ohjaa järjestelmän toimintaa käyttäjän käskyjen perusteella.
OHOS	Opinnäytetyön ja harjoittelun ohjaussovellus, Karelia-ammattikorkeakoulun aloittama ohjelmistoprojekti, jonka tavoitteena oli sähköistää opinnäytetyön ja harjoittelun prosessit yhden järjestelmän alle.
Pop-up-ikkuna	Ponnahdusikkuna, tarkoittaa yleensä verkkosivuilla käytettävää tekniikkaa, joka avaa käyttäjän eteen uuden, pienemmän ikkunan, joka sisältää esimerkiksi mainoksen tai tarkempaa tietoa klikatusta linkistä.
PowerShell	Windows-käyttöjärjestelmälle luotu komentotulkki, jonka avulla voidaan ajaa script-komentosarjoja.
Pseudokoodi	Tapa kuvata ohjelmistojen algoritmeja ilman eri ohjelmointikielille spesifejä syntakseja, siten että algoritmi on ymmärrettävissä riippumatta siitä, onko lukija perehtynyt tiettyyn kieleen vai ei.
REST	Representational State Transfer, ohjelmointirajapintojen arkkitehtuuri, joka käyttää perustanaan HTTP-protokollaa.
SaaS	Software as a Service, palvelumallinen ohjelmisto, jossa asiakas hankkii sovelluksen palveluna siten, että sitä käytetään verkon yli, palveluntarjoaja hoitaa ylläpidon ja päivitykset.
SDK	Software Development Kit, sovelluskehittäjäpaketti, joka sisältää tarpeelliset työkalut ja dokumentaation tietylle alustalle tai tiettyä teknologiaa käyttävän sovelluksen kehitykseen.
SQL	Structured Query Language, kyselykieli, jonka avulla relaatiomallisen tietokannan tietoihin päästään käsiksi.
TDD	Test Driven Development, testivetoinen ohjelmistokehitys, joka lähtökohtana on kirjoittaa ensin testiohjelma, jonka jälkeen kirjoitetaan tuotannollinen koodi, jolla tämä testi voidaan läpäistä puhtain paperein.
Visual Basic	Microsoftin kehittämä ohjelmointikieli, perustuu jo 1980-luvulla käytössä olleeseen QuickBasic-kieleen.
Web service	Verkkopalvelu, joka voidaan toteuttaa esimerkiksi REST-menetelmällä, tarjoaa tietyn palvelun muilla tietokoneilla toimiville järjestelmille verkon kautta.

1 Johdanto

Tämän opinnäytetyön aiheena on kehittää toimiva rajapinta OHOS-verkkosovelluksen ja Microsoftin CRM-ohjelman välille. Rajapinnan tehtävänä on ylläpitää näiden kahden sovelluksen sisältämien tietojen ajankohtaisuutta päivittämällä tietoja järjestelmien välillä ajastetusti. Tämän lisäksi rajapinnan on tarkoitus myös tuoda lisäarvoa organisaation CRM-sovellukselle antamalla mahdollisuuden Karelia-ammattikorkeakoulun opinnäytetöiden seurantaan: minne päin Suomea opinnäytetöitä tehdään ja kuinka laajoja harjoitteluja opiskelijat suorittavat missäkin kunnassa.

Opinnäytetyön toimeksianto liittyy tiiviisti organisaation digitalisaatiotyöryhmän projektiin, jonka tehtävänä on kehittää verkkosovellus opinnäytetöiden ja harjoittelun ohjausta varten. Kyseisen verkkosovelluksen alkuperäisen toiminnan lisäksi sen nähtiin olevan hyvä työkalu Karelia-ammattikorkeakoulun kanssa yhteistyössä toimivien yritysten ajankohtaiseen listaamiseen. Näin OHOS-palvelusta pystyttiin saamaan kaikki mahdollinen hyöty irti, niin opiskelijoille ja opettajille kuin koulun muullekin henkilökunnalle.

Opinnäytetyön toiminnallisen osan tavoitteena on perehtyä erilaisiin rajapintaratkaisuihin ja ryhtyä niiden pohjalta toteuttamaan mahdollisimman hyvin toimivaa rajapintaa sovellusten välille. Toiminnallisen osuuden tarkoitus on täyttää kaikki asiakkaan vaatimat asiat, jotka löytyvät vaatimusmäärittelystä (liite 1).

Opinnäytetyön toinen osa on tämä raportti, joka käsittelee kaikki työhön kuuluvat osat. Raportti alkaa johdannolla, jonka jälkeen käydään läpi opinnäytetyön käsitteet ja tekniikat. Käsitteitä koskevassa luvussa käydään läpi termejä, jotka liittyvät opinnäytetyöhön. Luvun tarkoitus on antaa lukijalle tarkempi käsitys termeistä ja tekniikoista, joista raportin aikana kerrotaan.

Käsitteiden ja tekniikoiden selvittyä dokumentti käsittelee työhön kuuluvat osat. Luku kertoo lukijalle tarkemmin, mitä ja millaisia sovelluksia toiminnalliseen osuuteen kuuluu. Tarkoituksena on antaa kuvaus osista, ennen kuin opinnäytetyö alkaa käsittelemään yksityiskohtaisesti rajapinnan toimintoja.

Dokumentin neljäs luku käsittelee rajapinnan toteutuksen. Siinä käydään läpi, kuinka rajapinta on tehty, minkälaisia asioita siitä löytyy ja kuinka ne toimivat. Luku on kattava paketti, jossa käydään läpi rajapinnan käyttöliittymän toteutus, rajapintaan kuuluvat algoritmit sekä CRM-järjestelmään ja SDK-pakettiin kuuluvat muokkauksen. Sen on tarkoitus antaa lukijalle käsitys, kuinka rajapinnan jokainen yksityiskohta on toteutettu, jotta tarpeen mukaan siitä voi lainata osia tai toteuttaa uudelleen toisen järjestelmän välille.

Toteutuksen jälkeen opinnäytetyö käy läpi projektin aikana saadut saavutukset. Tulokset-luvussa käydään läpi, missä opinnäytetyön aikana on onnistuttu. Siinä arvioidaan, kuinka hyvin opinnäytetyö vastaa vaatimusmäärittelyä. Lukija saa selville kappaleesta, mitä vaatimusmäärittelyn asioita työssä on toteutettu eri tavalla sekä perustellaan ratkaisut.

Dokumentin kuudes luku pohtii opinnäytetyön aikana saatuja saavutuksia. Siinä kerrotaan lukijalle opinnäytetyön tekijöiden saamia ajatuksia jatkokehityksestä. Luvussa kerrotaan, kuinka tekijät ovat kasvaneet ammatillisesti projektin aikana sekä otetaan kantaa lähdekritiikkiin.

2 Opinnäytetyön käsitteet ja tekniikat

Lähtökohtana oli toteuttaa kehitteillä olevan OHOS:n ja Karelia-ammattikorkeakoulun Microsoft Dynamics CRM -järjestelmän välille REST-pohjaista web service -rajapintaa. Rajapinnan tuli sisältää useita tiedonsiirtotyyppejä: asiakkaat, yhteyshenkilöt, opinnäytetyöt ja harjoittelut. Siihen haluttiin myös lisätä ominaisuudet eri tiedonsiirtotyyppien käyttöön ottamiselle sekä tiedonsiirtojen tiheydelle. (Liite 1.)

Tässä luvussa käydään läpi käsitteet, joita dokumentin aikana mainitaan. Luvussa avataan käsitteet, jotta ne ovat tutut dokumentin edetessä pitemmälle. Tässä käydään myös läpi tekniikoita, joita dokumentin aikana käsitellään.

2.1 Vaatimusmäärittely

Opinnäytetyön vaatimuksia täytyi käydä tarkkaan läpi, ettei projekti laajene liian isoksi kokonaisuudeksi. Tämän työn ensimmäisessä tapaamisessa selvitettiin vähimmäisvaatimukset, mitä rajapinnan tulee sisältää. Tapaamisessa käytiin myös läpi, mitä toimintoja rajapinnalta voisi mahdollisesti vaatia vähimmäisvaatimusten lisäksi. Nämä toiminnot jäivät kuitenkin toiveiksi. Palaverissa käytiin myös läpi, kumpi järjestelmistä dominoi tiedonsiirrossa. Dominoivalla järjestelmällä tarkoitetaan järjestelmää, jonka dataa pidetään tärkeämpänä. Dominoivaksi järjestelmäksi sovittiin OHOS, koska tähän tulee aina ajankohtaisempaa dataa verrattuna CRM- järjestelmään.

Varsinainen vaatimusmäärittely saatiin palaverin jälkeen sähköpostissa Word-tiedostona. Dokumentissa käydään tarkkaan läpi, mitä tietoa siirretään. Siirtologiikkaa tarkennetaan myös siten, että tyhjällä arvolla ei missään tapauksessa korvata olemassa olevaa arvoa. Dokumentti tarjoaa myös teknisiä ratkaisuja sovelluksen toteuttamiseen. Vaatimusmäärittelyssä mainitaan, että rajapinta tulisi toteuttaa käyttämällä REST-pohjaista verkkopalvelua. Vaatimusmäärittelyssä haluttiin myös nostaa esille ominaisuus, jolla tiedonsiirron tiheyttä pystyisi säätämään.

Opinnäytetyö toteutettiin seuraamalla tarkkaan vaatimusmäärittelyn antamia linjauksia. Vaatimusmäärittely toimikin osana työsuunnitelmaa ohjaamalla mitä rajapintaan tulee tehdä seuraavaksi. Sitä seurattiin tarkasti lähes jokaisessa työnvaiheessa opinnäytetyön edetessä. REST-pohjaista verkkopalvelua rajapinnasta ei kuitenkaan tehty, sillä palvelun ajastus olisi tuottanut ongelmia. Rajapinta toteutettiin lopulta Windows Form -pohjaisena sovelluksena. Tämä ratkaisu mahdollisti toimintojen ajastamisen, sekä sitä pidettiin asiakkaan toiveiden mukaisena.

2.2 Ohjelmointirajapinnat

Rajapintoja on ollut olemassa ensimmäisten karkeiden tietokoneohjelmien syntymästä saakka ja nykypäivänä ne ovat erittäin iso osa modernia ohjelmointia, sillä ne sallivat tiedonjaon erillisten sovellusten välillä sekä useampien erillisten ominaisuuksien helpon liittämisen osaksi uutta sovellusta. Ohjelmointirajapinnalla tarkoitetaan siis eräänlaista siltaa, jonka avulla kaksi sovellusta tai palvelua saadaan keskustelemaan toistensa kanssa siten, että tiedonjako näiden välillä on mahdollista. Käytännössä ohjelmointirajapinta sisältää määritykset siitä, mitä tietoa ja missä muodossa sovellusten välillä voidaan jakaa. Tämän lisäksi rajapinta sisältää listan funktioita, joita kehittäjät voivat käyttää oman sovelluksensa sisällä, sekä kuvaukset kyseisistä funktioista. Rajapintojen tärkein ominaisuus onkin se, että ne tarjoavat kehittäjille erilaisia valmiita ratkaisuja ohjelmistojen integroimiseen, eikä näitä ratkaisuja tarvitse ohjelmoida erikseen itse sovellukseen. (Hoffman 2018.)

Näitä ratkaisuja on saatavilla sekä yksittäisten kehittäjien luomina, että isompien ohjelmistoalan yritysten tarjoamana. Yksi esimerkki suuren yrityksen tarjoamasta API:sta on Google Maps API, jonka avulla kehittäjät saavat sovelluksensa käyttöön Googlen karttaominaisuuden ilman tarvetta ohjelmoida sovellukseen omaa karttaominaisuutta alusta lähtien. (Hoffman 2018). Tämän lisäksi esimerkiksi sosiaalisen median jättiläiset Twitter ja Facebook tarjoavat kehittäjille rajapintaa omiin sovelluksiinsa. Näiden alustojen rajapinnat tarjoavat siis käyttäjälle mahdollisuuden saada yhteys palveluiden dataan ohjelmoinnin kautta ja vaikkapa lisätä sovellukseensa mahdollisuus päivittää käyttäjän Twitter- tai Facebook-sivua oman sovelluksensa kautta. (Freeman 2018). Nykypäivänä lähes kaikki suuret verkkopalvelut tarjoavat kehittäjille rajapinnan omaan sovellukseensa, jotta ne saavat laajennettua oman sovelluksensa käyttäjäkuntaa.

Esimerkkejä erilaisista rajapinnoista ovat muun muassa paikalliset rajapinnat, web-rajapinnat sekä ohjelmistomalliset rajapinnat (Rouse, Nolle & Li 2017). Rajapintoja on satoja erilaisia ja tässä osiossa mainitut ovat vain muutama esimerkki niistä. Paikallisella rajapinnalla tarkoitetaan ohjelmointirajapintaa, joka tarjoaa käyttöjärjestelmän tai muun paikallisen sovelluksen palveluita toisen paikallisen

sovelluksen käyttöön. Esimerkkejä tällaisesta rajapinnasta ovat muun muassa Microsoftin .NET API:t, joilla saadaan integroitua sovelluksiin erilaisia .NET –ominaisuuksia.

Web-rajapinnat ovat rajapintoja, joiden avulla verkossa olevaan palveluun saadaan yhteys HTTP-protokollaa käyttäen. Käytännössä tämä tapahtuu niin, että esimerkiksi mobiilisovellukselta lähetetään tiedonhakupyyntö rajapinnalle, joka sitten noutaa kyseisen tiedon tietyssä muodossa toisesta verkkosovelluksesta ja palauttaa sen samassa muodossa mobiilisovellukselle. Kenties tunnetuin esimerkki tällaisesta rajapinnasta on REST-rajapinta, jossa haettu tieto palautetaan yleensä JSON tai XML –muodossa. (Rouse, Nolle & Li 2017.)

Näiden lisäksi on olemassa ohjelmistomallisia rajapintoja, jotka perustuvat RPC-protokollaan (Remote Procedure Call), jolla tarkoitetaan palveluiden etäkäyttöä. Tällaista rajapintaa käytettäessä sovellus siis käyttää muualla verkossa sijaitsevaa palvelua tai palvelun osaa hyväkseen siten, että tämä ilmenee sovellukselle itselleen paikallisena palveluna ja on siten normaalisti käytettävissä ilman ylimääräistä ohjelmointia. (Rouse, Nolle & Li 2017.)

Rajapintojen merkitys on kasvamassa, sillä niillä on suuri rooli asioiden internetin kehityksessä (Schöne 2017). Tämä johtuu suurimmaksi osaksi siitä, että tänä päivänä kaikki kodinkoneet jääkaapeista televisioihin ja pesukoneisiin halutaan liittää osaksi internetiä, sekä saada keskustelemaan toistensa ja pilvipalveluiden kanssa. Jotta edellä mainitut asiat saadaan toteutettua sekä järkevästi että tietoturvallisesti, vaaditaan toteuttamiseen erilaisia rajapintoja, ja erityisesti tässä vaiheessa korostuu API management eli rajapintojen hallinta. Rajapintojen hallinnalla tarkoitetaan kyseiseen tarkoitukseen kehitettyjä sovelluksia ja työkaluja, joiden avulla rajapintoihin saadaan ajettua päivityksiä ja niiden tietoturvaa saadaan hallittua (Thielens 2013). Tällaisia rajapintojen hallintasovelluksia on tarjolla sekä avoimen lähdekoodin että suurempien palveluntarjoajien kuten Microsoftin ja Amazonin kautta.

Perinteisten rajapintojen hallintasovelluksien lisäksi tarjolla on myös rajapintojen hallinta-alustoja, jotka sisältävät kokoelman API:en luomiseen ja hallintaan tarkoitettuja työkaluja. Tällaisten alustojen avulla voidaan myös hallita rajapintojen saapuvaa ja lähtevää liikennettä, minkä kautta organisaatiot voivat varmistaa, että rajapinnat eivät kaadu tai ruuhkaudu liiallisen tietoliikenteen seurauksena (Rouse & Churchville 2016). Hallinta-alusta siis myös lisää osaltaan tietoturvaa. Useat hallinta-alustat sisältävät edellä mainittujen ominaisuuksien lisäksi niin sanotun kehittäjäportaalin, jonka avulla rajapinnat saadaan kätevästi levitykseen ja joka tarjoaa kehittäjille nopean keinon etsiä tarpeeseensa sopivat rajapinnat.

Hallintasovelluksien ja –alustojen lisäksi eräs ratkaisu rajapintojen hallintaan on API-portti, joka toimii eräänlaisena reitittäjänä sovelluksien ja mikropalvelujen välillä. Mikäli sovellus käyttää esimerkiksi mikropalveluarkkitehtuuria ja kehittäjä haluaa jonkin tietyn mikropalvelun sovelluksensa käyttöön, hän voi ottaa sovelluksen kautta yhteyden API-porttiin ja lähettää pyynnön käyttää tiettyä mikropalvelua portin toisella puolella. Portti toimii siis yhteyspisteenä sovelluksien ja lukuisien mikropalvelujen välillä. Tällaisen portin hyötyjä on varsinkin se, että sen avulla saadaan vähennettyä edestakaisen liikenteen ja pyyntöjen määrää, koska portin kautta saadaan haettua tietoa useammista mikropalveluista yhdellä pyynnöllä. Yhtenä esimerkkinä tällaisesta API-portista on Netflixin API-portti, joka toimii siten, että kullakin päätelaitteella toimiva sovellus ottaa käytön aikana yhteyden tälle päätelaitteelle suunniteltuun API-porttiin, joka sitten reitittää kyseisen sovelluksen pyynnöt sitä vastaaviin mikropalveluihin. (Richardson 2018.)

2.3 SDK (Software Development Kit)

SDK eli Software Development Kit tarkoittaa tietyn alustan sovellusten ohjelmointia varten luotua ns. kehittäjäpakettia, joka useimmiten sisältää työkaluja, esimerkkikoodia, kirjastoja ja dokumentaatiota kyseisen alustan sovelluskehitystä varten. Usein nämä kehittäjäpaketit sisältävät myös useita rajapintoja, jotka voivat olla hyödyksi sovelluksien kehityksessä.

Ohjelmistokehityspakkaus sisältää tyypillisesti integroidun kehitysympäristön, joka palvelee ohjelmoinnin keskusliittymänä. Tämä kehitysympäristö voi sisältää ohjelmointi-ikkunan lähdekoodin kirjoittamiselle, debugger-työkalun virheiden ratkomiseen sekä visuaalisen editorin, jolla kehittäjät voivat rakentaa ja muokata sovelluksen graafista käyttöliittymää. Ohjelmistokehityspakkaukset sisältävät usein myös kääntäjän, jonka avulla lähdekoodista voidaan rakentaa lopullinen sovellus.

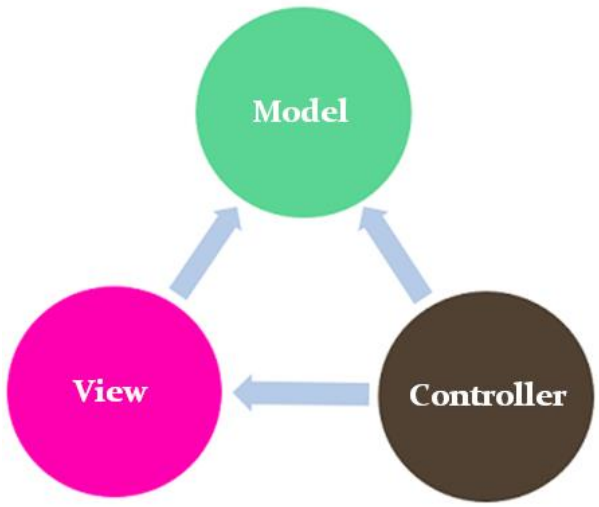
Monet SDK:t sisältävät esimerkkikoodin, joka antaa kehittäjälle esimerkkisovellukset sekä -kirjastot. Nämä esimerkit auttavat kehittäjää oppimaan, kuinka kehittää yksinkertaisia sovelluksia kehittäjäpaketin avulla, mikä mahdollistaa jatkossa yhä monimutkaisempien sovellusten kehityksen. SDK:t tarjoavat myös teknisiä dokumentteja, jotka voivat sisältää ohjeita sekä vastauksia usein kysyttyihin kysymyksiin. Tämän lisäksi kehittäjäpaketit sisältävät esimerkkigrafiikkaa, kuten näppäimiä tai ikoneja, joita sovelluksissa voi käyttää. (Christensson 2010.)

Koska monet yritykset haluavat ohjelmoijien kehittävän sovelluksia omille alustoilleen, SDK:t ovat usein tarjolla ilmaiseksi. Eräs esimerkki tällaisista yrityksen tarjoamista kehityspaketeista on tässä työssä käytetty Microsoft CRM SDK. Kehittäjät voivat ladata kehittäjäpaketin yrityksen kotisivuilta ja aloittaa ohjelmoinnin välittömästi. Kuitenkin, koska jokainen ohjelmistokehityspakkaus on erilainen, kehittäjillä saattaa kulua aikaa opetellessa uuden kehittäjäpaketin ominaisuuksia. Sen takia useimmat modernit SDK:t sisältävät yksityiskohtaiset dokumentaatiot ja intuitiiviset ohjelmointialustat, jotka auttavat sovelluskehityksessä.

Kehittäjäpaketit ovat myös rajapintojen kannalta tärkeitä, sillä niiden avulla rajapintojen kehittäjät saavat levitettyä tuotteensa laajempaan käyttöön, sekä saavat ohjeistettua kehittäjiä niiden käyttämisessä. SDK:n julkaisu voi aiheuttaa myös ongelmia, sillä vaikka se tarjoaa tietyn lähtökohdan sovelluskehitykseen kyseiselle alustalle, samalla se saattaa myös rajoittaa kehittäjien ajattelutapaa, kun sovelluksia lähdetään kehittämään. Kehittäjien on helppo tyytyä SDK:n tarjoamiin raameihin ja hylätä laatikon ulkopuolinen ajattelu. Juuri tästä syystä kunkin kehittäjäpaketin julkaisijoiden on syytä panostaa paketin kehitykseen ennen sen julkaisua. (Sandoval 2016.)

2.4 MVC-arkkitehtuuri

MVC on arkkitehtuurimalli, joka alun perin kehitettiin työpöytäsovelluksia varten vuonna 1970. Nykyisin sitä kuitenkin käytetään laajasti myös verkkosovelluksissa. Tämän seurauksena monet rakenteet on kehitetty seuraamaan tätä kaavaa. MVC-arkkitehtuuri on ollut pitkään suosittu ohjelmistokehittäjien keskuudessa. Monet ohjelmointikielet omaksuvat tätä kuviota monilla eri variaatioilla. (Shahid 2018). Arkkitehtuuri koostuu nimensä mukaan kolmesta erillisestä komponentista: malli (model), näkymä (view) ja käsittelijä (controller). Arkkitehtuurin kolmen komponentin yhteistyö esiintyy kuvassa 1.



Kuva 1. MVC-arkkitehtuuri.

Mallin tarkoitus on kuvata sovellukseen liittyvää tietoa. Se voi olla yksittäinen kappale tai rakentua useammasta kappaleesta. Yksittäinen malli voi myös koostua useasta mallista. OHOS-sovellukseen on rakennettu malleja käyttäen jokaista vaihtoehtoa. (Atwood 2008.)

Näkymä on visuaalinen esitys sen mallista. Yleensä se korostaa joitain osia mallista ja peittää loput. Näin se toimii esittävänä suodattimena. Näkymä on liitetty sen malliin ja se saa tietoa tarvittaessa esittämällä kysymyksiä mallille. Se voi

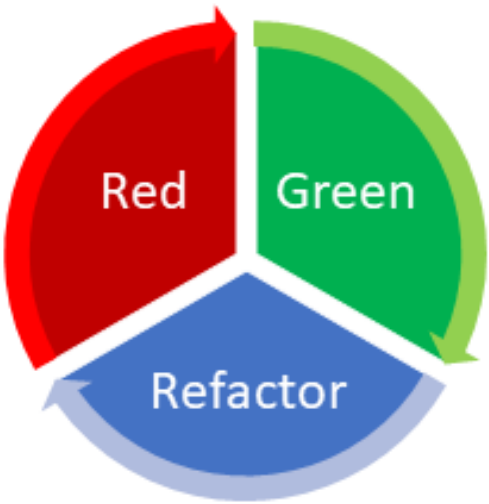
myös päivittää mallia lähettämällä asianmukaisia viestejä. Kaikkien näiden kysymysten ja viestien täytyy olla mallin terminologiassa, ja näkymän täytyy silloin olla tietoinen mallin ominaisuuksien semantiikasta, jota se esittää. (Atwood 2008.)

Käsittelijä on yhteys järjestelmän ja käyttäjän välillä. Se tarjoaa käyttäjälle syötettä järjestelemällä merkitykselliset näkymät esittämään itsensä asianmukaisille paikoille näytöllä. Se antaa käyttäjälle työkalut suorittaa komentoja tai vaihtaa näkymiä. Käsittelijä siis vastaanottaa käyttäjän syötteet, kääntää ne asianmukaisiksi viesteiksi ja toimittaa nämä viestit yhteen tai useampaan näkymään. (Atwood 2008)

2.5 Testivetoinen kehitys

Testivetoisella ohjelmistokehityksellä tarkoitetaan ohjelmointimallia, jossa ohjelmaa kirjoitetaan testaus edellä, eli ennen varsinaisen tuotannollisen koodin kirjoittamista täytyy ensin kirjoittaa testi, joka halutaan läpäistä. Tällaisen kehityksen lähtökohtana toimii yksikkötestaus, jonka tarkoituksena on testata kaikki sovelluksen yksittäiset komponentit, kuten funktiot ja luokat, läpi yksitellen niin kauan, että jokainen osa saadaan läpäisemään sitä varten luotu testi. Tätä testaamista suoritetaan läpi sovelluksen kehitysprosessin, ja se tapahtuu ohjelmiston kehittäjän toimesta. Useimmiten yksikkötestaus suoritetaan automatisoidusti käyttämällä tähän tarkoitukseen suunniteltuja työkaluja, joiden avulla saadaan luotua testiluokkia. Testaus voidaan suorittaa myös manuaalisesti, mutta tämä ei ole yhtä kustannustehokasta kuin automaattinen testaus, jota useimmiten suositellaan käytettäväksi. (Guru99 2014.)

Käytännössä testivetoinen kehitys koostuu eräänlaisesta kehästä (kuva 2), joka voidaan jakaa aluksi kahteen osioon, punaiseen vaiheeseen eli testin kirjoittamiseen ja vihreään vaiheeseen eli tämän testin läpäisevän tuotannollisen koodin kirjoittamiseen. (Maxwell 2018.)



Kuva 2. Testivetoisen kehityksen prosessi (Maxwell 2018.)

Ohjelmointi aloitetaan kirjoittamalla yksikkötesti, jonka tiedetään epäonnistuvan. On erittäin tärkeää, että juuri kirjoitettu testi epäonnistuu, koska sitä kautta saadaan varmistettua koodin pysyminen yksinkertaisena ja vain tarpeellisen laajana tämän testin läpäisemiseksi. Kun on varmistuttu siitä, että kirjoitettu testi epäonnistuu, lähdetään kirjoittamaan tuotannollista koodia, jolla tämä testi saadaan läpäistyä. Tässä vaiheessa on tärkeää pitää mielessä, että koodia kirjoitetaan vain sen verran, että aluksi kirjoitettu testi saadaan läpäistyä. Koodin voi pitää myös hyvinkin yksinkertaisena, sillä sen ainoa tarkoitus on läpäistä yksinkertainen testi. Kun testi on saatu läpäistyä, jatketaan kirjoittamalla seuraava testi, jonka lähtökohta on jälleen epäonnistua, ja siitä jatketaan kirjoittamalla tuotannollista koodia, jolla tämä testi läpäistään. (Bender & McWherter 2011, 6-9.)

Testin läpäisyn jälkeen kehään voidaan liittää myös kolmas vaihe, joka kulkee nimellä refaktorointi (Maxwell 2018). Tässä vaiheessa kehittäjä voi muokata testauksen läpäissyttä koodia vastaamaan tuotantovaatimuksia, pitäen sen samalla ehjänä niin, että se läpäisee aiemmin kirjoitetun testitapauksen. Refaktorointi pitää sisällään mahdollisten toistuvien koodinpätkien poistamisen tai uudelleen muokkaamisen, ns. kovakoodattujen arvojen tai funktioiden poistamisen sekä muita koodin tyylittelyyn liittyviä toimenpiteitä. Tässä vaiheessa testin läpäisseen koodin rakennetta siis muutetaan vastaamaan sekä organisaation asettamia, että kehittäjän omia laatuvaatimuksia, pitäen kuitenkin samalla koodin toiminnallisuuden muuttumattomana (Koutifaris 2018).

Testivetoisen kehityksen kehää seuraamalla ohjelmoija voi varmistua siitä, että hänen kirjoittamansa sovellus vastaa kaikkia esitettyjä vaatimuksia, sillä hänellä on valmiiksi kirjoitetut testit varmistamassa, että sovelluksen jokainen osa toimii oikein. Tämän lisäksi testivetoinen kehitys paljastaa mahdolliset virheet koodissa jo aikaisessa vaiheessa, sillä testejä ajetaan jatkuvasti ja kun jokainen koodirivi tulee testattua useampaan kertaan, eivät kriittiset virheet jää piiloon. Testivetoinen kehitys on myös erinomainen metodi tiimityöskentelyn kannalta, sillä poissaolojen sattuessa kehitystiimin jäsen voi helposti jatkaa toisen jäsenen työtä, koska kehittäjillä on olemassa olevat testit, joilla koodin toimivuus voidaan varmistaa myös kehittäjän vaihtuessa. (Kulkarni 2019.)

Testivetoisen kehityksen lisäksi on olemassa muutamia muitakin kehityssuuntauksia, jossa sovelluksen kehityksessä mennään testaaminen edellä. Eräs esimerkki näistä on käyttäytymisvetoinen kehitys, jota voi käyttää erikseen tai yhdessä testivetoisen kehityksen kanssa. Tässä kehitysmallissa on otettu huomioon testivetoisen kehityksen parhaat puolet, mutta kehitys tapahtuu toteutukseen keskittymisen sijaan keskittymällä sovelluksen käyttäjien oletettuun käyttäytymiseen. Tämän lisäksi suurin ero testi- ja käyttäytymisvetoisen kehityksen välillä on testien kuvaamisessa. Käyttäytymisvetoisessa kehityksessä testit pyritään kuvaamaan enemmän puhekielen kaltaisella kielellä, jotta ne voitaisiin paremmin selittää sekä kuvailla asiakkaalle. Tässä kehitysmallissa asiakkaan kanssa kommunikointi onkin suuremmassa roolissa kuin testivetoisessa kehityksessä, koska testit pyritään tekemään käyttäjien mahdollisen käyttäytymisen pohjalta. (Hit Subscribe 2018.) Testivetoisessa kehityksessä testit pyritään taas tekemään toiminnallisuus edellä, ja testattava ominaisuus sekä testien tarkoitus jäävät usein epäselväksi muille paitsi kehittäjätiimille itselleen.

3 Rajapinnan osat

Opinnäytetyön tavoitteena oli toteuttaa rajapintasovellus kahden järjestelmän välille. Rajapinnan toteutukseen paneudutaan tarkemmin luvussa 4. Ennen toteutukseen tutustumista on ensin selvitettävä mistä osista rajapinta koostuu. Tässä luvussa käydään läpi kaikki osa-alueet, joista koko opinnäytetyön toiminnallinen osuus rakentuu, eli OHOS- järjestelmä, Microsoft Dynamics CRM sekä CRM-järjestelmää varten julkaistu kehityspakkaus

3.1 OHOS

OHOS eli opinnäytetyön ja harjoittelun ohjaussovellus on Karelia-ammattikorkeakoulun aloittama projekti, jonka tarkoituksena oli sähköistää opinnäytetyön ja harjoittelun prosessit sekä niiden ohjaus yhden järjestelmän alle. Tällä haluttiin parantaa opiskelijan ja ohjaajan välistä kommunikointia, sekä päästä eroon vanhanaikaisesta paperisodasta opinnäytetyön ja harjoittelun yhteydessä. Sovelluksen avulla haluttiin myös helpottaa opinnäytetöiden ja harjoittelujen etenemistä tilanteessa, jossa näiden ohjaaja vaihtuu. Sovelluksesta oli määrä löytyä muun muassa molempiin osa-alueisiin kuuluvat lomakkeet sekä ominaisuus tiedostonpalautukselle sekä chat-ominaisuus, jolla ohjaaja pystyi pitämään yhteyttä opiskelijaan järjestelmän sisällä sekä seuraamaan tämän opinnäytetyön tai harjoittelun etenemistä. Tämän lisäksi sovelluksen tuli olla muokattavissa siten, että koulutusalojen käytännölliset erot voitiin ottaa huomioon. Tällä tarkoitetaan esimerkiksi sitä, että eri koulutusaloilla voi olla käytössä eri määrä palautettavia harjoitustehtäviä opinnäytetyöhön liittyen, minkä lisäksi harjoittelu voi sisältää erilaisia seurantatehtäviä ja päiväkirjoja ynnä muuta koulutusalasta riippuen.

Sovelluksen kehitys aloitettiin tammikuussa 2017 ja aluksi sen parissa työskenteli kaksi erillistä työryhmää; toisen tehtävänä oli suunnitella ja toteuttaa harjoittelusovellus, kun taas toinen ryhmä vastasi osaltaan opinnäytetyösovelluksesta. Projektin lopulla nämä sovellukset oli tarkoitus yhdistää saman katon alle, mistä syystä molemmissa sovelluksissa oli määrä noudattaa samoja ohjelmointikäytäntöjä ja visuaalista linjaa. Projektia lähdettiin toteuttamaan käyttäen ASP.NET-

ohjelmointikehyksen tarjoamaa MVC-arkkitehtuuria ja sen tietokannaksi valittiin Microsoftin SQL-palvelin. Sovelluksen oli määrä sisältää käyttäjäroolit opiskelijan ja ohjaajan lisäksi myös koulutusalojen päälliköille sekä opinnäytetöiden tarkastajille.

Projektin aikana molempien työryhmien kokoonpano vaihteli jatkuvasti, ja deadlinen lähestyessä kävi selväksi, että sovellus ei valmistu ajallaan. Lopulta sovelluksen harjoitteluosa todettiin toimimattomaksi ja sitä lähdettiin kehittämään uudestaan opinnäytetyöosiosta vastanneen työryhmän toimesta. Tästä huolimatta sovellusta ei koskaan saatu julkaisukuntoon, mikä taas johti osaltaan siihen, että tätä opinnäytetyötä tehdessä OHOS ei sisältänyt juurikaan toiminnallisuutta harjoittelun ohjausta varten, mikä vaikutti myös opinnäytetyön sisältöön.

3.2 Asiakkuudenhallinta

CRM eli asiakassuhteiden hallintasovellus on liiketalouden työkalu, joka auttaa yrityksiä hallitsemaan asiakassuhteita sekä parantaa yrityksen ja asiakkaiden välistä keskustelua. CRM-sovelluksen avulla yritys voi mm. hallita myyntejään entistä paremmin, pitää kirjaa nykyisistä asiakkaista, hallita varastotoimintojaan sekä analysoida kaikkea myyntiin liittyvää dataa. (Rouse, Ehrens & Kiwak 2013)

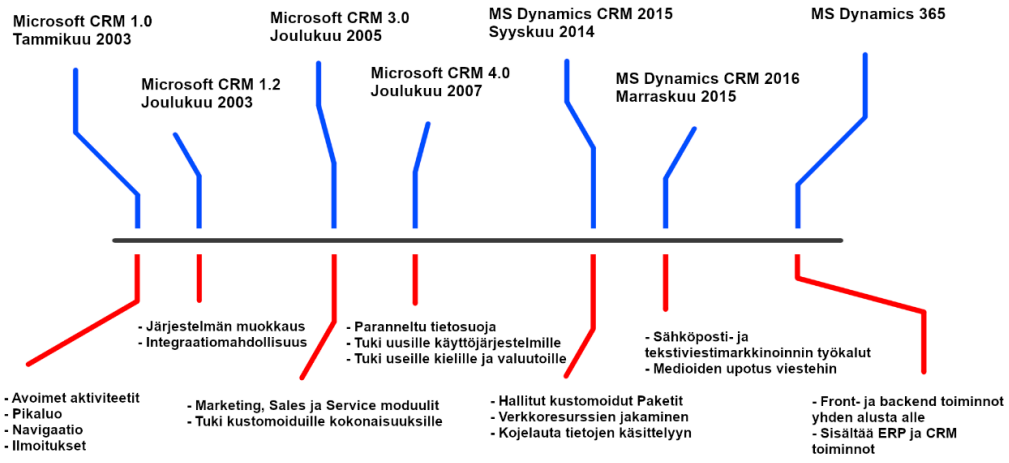
Karelia-ammattikorkeakoulu on käyttänyt asiakkuudenhallintajärjestelmää tallentamalla koulun yhteistyöhenkilöt opinnäytetöiden ja harjoittelujen osalta. Järjestelmään on syötetty niiden yritysten nimet, joihin opiskelijat ovat tehneet tai tekevät opinnäytetöitä ja harjoitteluja. Tämä toimenpide on suoritettu manuaalisesti työntekijöiden toimesta. Tämän opinnäytetyön tarkoitus on automatisoida toimenpide niin, että jatkossa opiskelijoiden verkkoon täyttämistä sopimuksista tieto kulkee suoraan CRM:n järjestelmään.

Karelia-ammattikorkeakoulun asiakkuudenhallintajärjestelmä on vielä keskeneräinen sen datan suhteen. Koska dataa on syötetty tähän mennessä käsin, se on hyvin vajavaista. Esimerkiksi yritysten osoitteita ei ole syötetty järjestelmään läheskään jokaisen yrityksen kohdalta. Tämän opinnäytetyön aikana toteutettu

rajapinta automatisoi sen syöttämisen, sekä pitää sen päivitettynä. Tiedot olisivat ajankohtaisia, koska opiskelijoiden palauttamista lomakkeista löytyisi aina ajankohtaiset yrityksen tiedot osoitteineen ja puhelinnumeroineen.

3.3 Microsoft Dynamics CRM

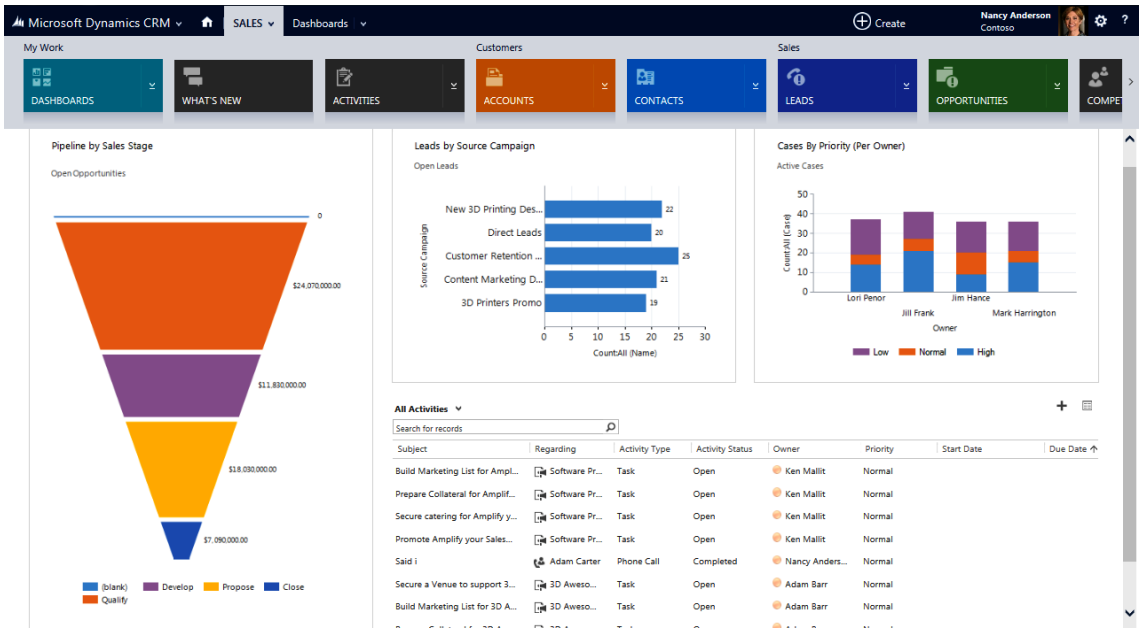
Dynamics CRM on Microsoftin luoma sovellusratkaisu asiakkuudenhallintaan. Pitkän elinkaarensa ansiosta se on vallannut paikan yhtenä suosituimmista asiakkuudenhallintajärjestelmistä. Lisäksi Microsoft on jo vuosia ollut suosittu sovelluskehittäjä yritysten keskuudessa, mikä on tehnyt myös Dynamics CRM:stä luotettavan vaihtoehdon asiakkuudenhallintaan. Yritykset, jotka suosivat Microsoftin tuotteita, saavat Dynamics CRM:stä helposti opittavan ja integroitavan osan yrityksensä sovelluskokonaisuutta (Chorus 2015). Microsoft on julkaissut sovelluksestaan vuosien varrella useita versioita, joista tässä osiossa perehdytään versioon Microsoft Dynamics CRM 2015. Kuvassa 3 on avattu Dynamics CRM:n versiohistoriaa sekä kunkin version tuomia uusia ominaisuuksia.



Kuva 3. Microsoft Dynamics CRM:n historia.

Microsoft tarjoaa Dynamics CRM-sovellusta sekä paikallisesti että SaaS –mallisena palveluna. Paikallisesti ostettuna Dynamics CRM vaatii Microsoftin Windows –käyttöjärjestelmän sekä Microsoftin palvelinteknologian toimiakseen, kun taas palveluna tilattuna Dynamics CRM –järjestelmän verkkoversioon, nimeltään

Dynamics CRM Online, voi muodostaa yhteyden useimmilta verkkoselaimilta. (Houpes 2015.)



Kuva 4. Microsoft Dynamics CRM 2015 käyttöliittymä (Chorus 2015.)

Microsoft Dynamics CRM 2015 julkaistiin vuoden 2014 viimeisellä neljänneksellä. Dynamics CRM 2015 ei enää tukenut kaikkia vanhempia Microsoft-perheen tuotteita, esimerkiksi Internet Explorer -tuki puuttui vanhemmista versioista, mikä tarkoitti sitä, että yritykset joutuivat päivittämään selaimet 10-versiota vanhemmasta uusimpaan. Dynamics CRM 2015 version käyttöliittymä on esitetty kuvassa 4. Tämän lisäksi 2015 versio toi mukanaan muun muassa mahdollisuuden tietojen hakuun kaikista järjestelmään kirjatuista entiteeteistä sekä parannuksia järjestelmän mobiiliin toimintaan; järjestelmä tukee viimeisimpiä mobiilikäyttöjärjestelmiä sekä sisältää mahdollisuuden lisätä myyntitapahtumia offline-tilassa, jonka jälkeen tiedot synkronoidaan järjestelmään, kun käyttäjä palaa online-tilaan (D365 2014). Tässä versiossa sovelluksen mobiiliversio oli rakennettu siten, että voitiin varmistaa myös mobiilikäyttäjien saavan käyttöönsä samat monipuoliset ominaisuudet kuin muutkin käyttäjät. Näitä ominaisuuksia olivat esimerkiksi täysi taulukkotyyppien tuki ja ulkoasun muokkaus. Mobiiliversio sisälsi myös mahdollisuuden piilottaa tietyt entiteettityypit tai ominaisuudet suorituskyvyn parantamiseksi. (Microsoft 2015). 2015-versiossa sähköpostimarkkinointia oli myös paranneltu. Sähköposteja varten mainostajat pystyivät löytämään valmiita kehikkoja, joita käyttää, tai tekemään niitä itse tyhjästä käyttämällä interaktiivista ”raahaa ja tiputa” -

prosessia. Kokeneemmille käyttäjille löytyi myös CSS ja HTML –muokkausmahdollisuus web-resurssien muodossa. (Microsoft 2015).

Edellä mainittujen ominaisuuksien lisäksi eräs Microsoft Dynamics CRM 2015 version täysin uusista ominaisuuksista olivat siihen lisätyt Microsoft Social Engagement –työkalut, joiden avulla järjestelmän käyttäjä voi pitää asiakkaaseen suoraa yhteyttä yrityksensä sosiaalisen median sivujen kautta. Tämä antaa yritykselle myös mahdollisuuden seurata, mitä heistä puhutaan sosiaalisen median eri kanavilla ja kerätä nämä tiedot taulukoihin, joiden avulla voidaan muun muassa seurata viimeisimpien trendien kehitystä, sekä kohdentaa mainontaa erilaisten tehostettujen sosiaalisen median kampanjoiden avulla. Lisäksi Dynamics CRM 2015 sisälsi mahdollisuuden Help-osion muokkaukseen, jolla voitiin huomattavasti parantaa käyttäjäkokemusta muokkaamalla sovelluksen jokaiselle osalle oma Help-osio, joka vastaa mahdollisimman kattavasti käyttäjien mahdollisiin ongelmiin. Muokkaus tarjosi kehittäjille myös mahdollisuuden asettaa Help-osioon parametreja, joiden perusteella käyttäjä ohjattiin ongelman sattuessa käyttäjän kieltä vastaavalle sivulle, joka on tärkeä ongelman ratkaisun kannalta (Microsoft 2015).

Microsoft on ollut suosittu sovelluskehittäjä myös Karelia-ammattikorkeakoulussa, mistä syystä organisaatio on päättynyt Dynamics CRM asiakkuudenhallintaan. Karelia-ammattikorkeakoululla on paikallisesti toimiva CRM-sovellus, joka sijaitsee organisaation omalla paikallisella palvelimella.

3.4 Microsoft Dynamics CRM SDK

Microsoft Dynamics CRM 2015 SDK on kehittäjäpaketti, joka on tyypilliseen tapaan ladattavissa yrityksen kotisivuilta ilmaiseksi. Se sisältää kaiken tarvittavan järjestelmän muokkaamista varten. Paketin mukana tulee paljon erilaisia työkaluja järjestelmän testaamista varten. Se sisältää myös paljon kuvia, joita kehittäjä voi käyttää sovelluksensa ulkoasun parantamiseksi. Tärkeimpänä kaikista paketin mukana tulleista välineistä on kuitenkin esimerkkikoodi, jonka avulla kehittäjät pääsevät suoraan työn pariin. Esimerkilliseen tapaan Microsoft Dynamics CRM 2015 SDK tarjoaa kehittäjälle aloituskoodin neljällä eri ohjelmointikielellä. Paketista löytyy esimerkkikoodit C#-kielelle, JavaScriptille, Visual Basic-kielelle sekä Windows PowerShellille.

Tässä opinnäytetyössä keskityttiin C#-kielelle tehtyyn pakettiin, koska paketti oli selvästi muille ohjelmointikielille tehtyjä paketteja laajempi. Kyseistä ohjelmointikieltä oli käytetty myös OHOS-järjestelmän kehityksessä, joten oli luontevampaa käyttää yhtenäistä kieltä. Paketin esimerkkikoodi oli selkeästi rakennettu ja siitä löytyi helposti kaikki tarvittavat komennot työn aloittamiseen. Tämän lisäksi ohjelmistokehityspakkauksen esimerkkikoodille tarkoitettu Readme-tiedosto sisälsi ohjeet pelkästään C#-paketille, joten siihen löytyi neuvoja suoraan paketin sisältä.

Microsoft Dynamics CRM SDK on kaiken kaikkiaan laaja paketti, josta löytyy lähes kaikki, mitä kehittäjä voi tarvita sovelluslaajennusten toteuttamiseksi. Monipuolinen valikoima ohjelmointikielien esimerkkikoodia antaa kehittäjälle vapaat kädet laajentaa järjestelmäänsä ilman rajoituksia. Hienoja graafisia elementtejä sekä laajoja esimerkkikoodeja lukuun ottamatta paketti jää kuitenkin vajaaksi, sillä paketista ei löydy integroitua kehitysalustaa. Paketin mukana tulleissa tiedostoissa kuitenkin neuvotaan käyttämään kehitysalustana Windowsin omaa Visual Studiota, joka on ladattavissa ilmaiseksi Microsoftin kotisivuilta.

4 Rajapinnan toteutus

Tämän opinnäytetyön toteutusvaihe on ollut vaiheikas. Toteutuksen aikana sovellusta on muutettu suunnitelmista poiketen moneen suuntaan. Koska kommunikointi asiakkaan suuntaan oli huonoa, työssä jouduttiin ottamaan vapauksia rajapinnan toteutuksen suhteen, mikä aiheutti paljon työtä, josta olisi selvinnyt helpomminkin.

Toteutus koostui kolmesta kokonaisuudesta: suunnittelusta, toimintojen toteutuksesta ja käyttöliittymän hahmottelusta. Suunnitteluvaihe oli käytännössä esimerkkikoodiin tutustumista, jonka pohjalta tehtiin ratkaisut toteutuksen tekniikoiden suhteen. Toteutus tapahtui testivetoista kehitystä mukaillen, koska vaatimusmäärittelyssä esitetyt toiminnot oli mahdollista jakaa yksittäisiksi testeiksi. Testivetoisen kehityksen valitseminen mahdollisti myös työskentelyn aloittamisen välittömästi ilman yksityiskohtaisen suunnitelman luomista.

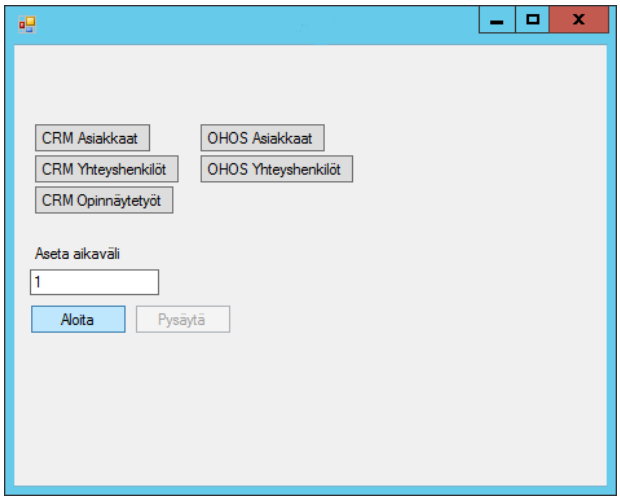
Sovelluksen käyttöliittymän toteutus tapahtui hahmottelemalla, minkälaisia kenttiä käyttöliittymään tulee. Käyttöliittymä toteutettiin lopulta minimivaatimukset täyttäen. Käyttöliittymästä ei mainittu mitään vaatimusmäärittelyssä, joten sen yksityiskohtiin ei juuri panostettu. Lopulta käyttöliittymästä löytyi pelkästään toiminnot, joilla käyttäjä pystyi luomaan yhteyden CRM- palvelimeen, valitsemaan tiedonsiirtotyypit, asettamaan ajanjakson ja käynnistämään sovelluksen.

4.1 Käyttöliittymä

Käyttöliittymän toteutukseen oli useita mahdollisia vaihtoehtoja. Rajapinnan olisi mahdollisesti voinut toteuttaa esimerkiksi verkkosovelluksena, jolloin käyttöliittymänä olisi toiminut verkkosivu. Verkkosovelluksen toteuttamisessa olisi ollut etuna esimerkiksi käyttöliittymän toimivuus useammalla eri päätelaitteella. Sovelluksen toimivuus ajastettuna olisi kuitenkin tuottanut suuria vaikeuksia, jos verkkosovellus olisi valittu toteutettavaksi tekniikaksi. Toisaalta käyttöliittymänä olisi voinut toimia myös yksinkertainen konsoli, johon olisi syötetty tietoa. Konsolisovellus olisi ollut käytettävyydeltään haastavin.

Rajapinta päädyttiin toteuttamaan Windows Forms -käyttöliittymänä, sillä sen avulla sovelluksen ajastus oli yksinkertaisin toteuttaa. Se oli myös hieman helppokäyttöisempi kuin konsolisovellukset. Sovelluksen käyttöliittymään kuului yksi Windows Forms -ikkuna sekä yksi konsoli-ikkuna, jolla sovelluksessa valitaan CRM-palvelin, johon se halutaan yhdistettävän. Konsoli-ikkuna oli osa Microsoft Dynamics CRM SDK:n esimerkkikoodia, eikä sitä sen alkuperäisen toimivuuden kannalta lähdetty toteuttamaan uudelleen.

Kokonaisuudessaan käyttöliittymä koostui kolmesta erillisestä ikkunasta. Ensimmäinen ikkuna, joka käyttäjälle avautuu, on pääkäyttöliittymä. Pääkäyttöliittymä on toteutettu Windows Forms -ikkunana. Ikkunasta löytyy valinnat tiedonsiirron tyypeille, aikaväli tiedonsiirroille sekä näppäimet sovelluksen käynnistämiselle ja pysäyttämiselle.



Kuva 5. Rajapintasovelluksen käyttöliittymä.

Käyttöliittymästä valittavilla tiedonsiirtotyypeillä (Katso kuva 5) tarkoitetaan sitä, minkälaisia kokonaisuuksia siirretään järjestelmästä toiseen. Kokonaisuuden edessä oleva järjestelmän nimi viittaa siihen, mihin järjestelmään kyseistä tietoa ollaan viemässä. CRM-järjestelmään meneviä tietotyyppejä on enemmän kuin OHOS-järjestelmään meneviä, koska CRM-järjestelmästä ei löydy opinnäytetöitä, joita OHOS-järjestelmään olisi mahdollista viedä.

Tietotyypeistä seuraavana lomakkeelta löytyy kenttä aikavälin asettamiselle. Aikavälillä tarkoitetaan jaksoa, millä välillä rajapinta suorittaa halutut tietotyyppien siirrot. Aika syötetään kenttään kokonaislukuna, joka vastaa minuutteja. Vaatimusmäärittelyssä ehdotettiin ajan olevan joko tunteja tai päiviä, mutta aikavälin toimivuuden testaamista varten oli yksinkertaisempaa pysytellä minuuteissa. Aikavälin muuttaminen tunneiksi on kuitenkin helppo muokkaus ohjelman lähdekoodiin kuvan 6. mukaisesti.

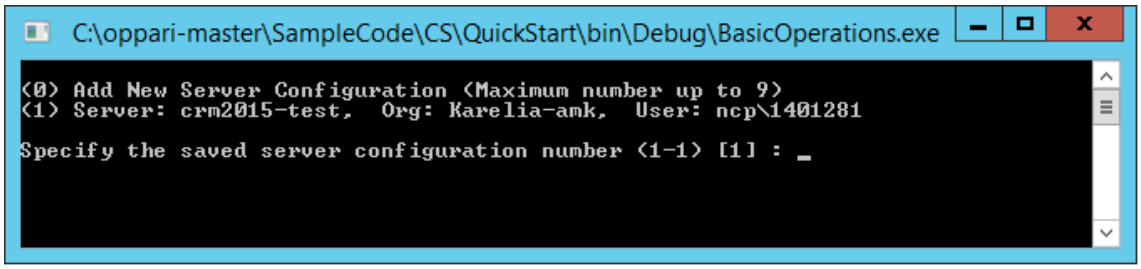
```
115             if (start)
116             {
117             MyTimer.Interval = (Convert.ToInt32(var) * 60 * 1000); // 1 mins
118             MyTimer.Tick += (sender2, e2) => MyTimer_Tick(sender2, e2, serviceProxy, records);
119             MyTimer.Start();
120             } else
121             {
122                 MyTimer.Stop();
123             }
124         }
125
```

```
115             if (start)
116             {
117             MyTimer.Interval = (Convert.ToInt32(var) * 60 * 1000 * 60); // 60 mins
118             MyTimer.Tick += (sender2, e2) => MyTimer_Tick(sender2, e2, serviceProxy, records);
119             MyTimer.Start();
120             } else
121             {
122                 MyTimer.Stop();
123             }
124         }
125
```

Kuva 6. Rajapintasovelluksen aikamääreen muuttaminen minuuteista tunneiksi Form.cs -tiedostossa.

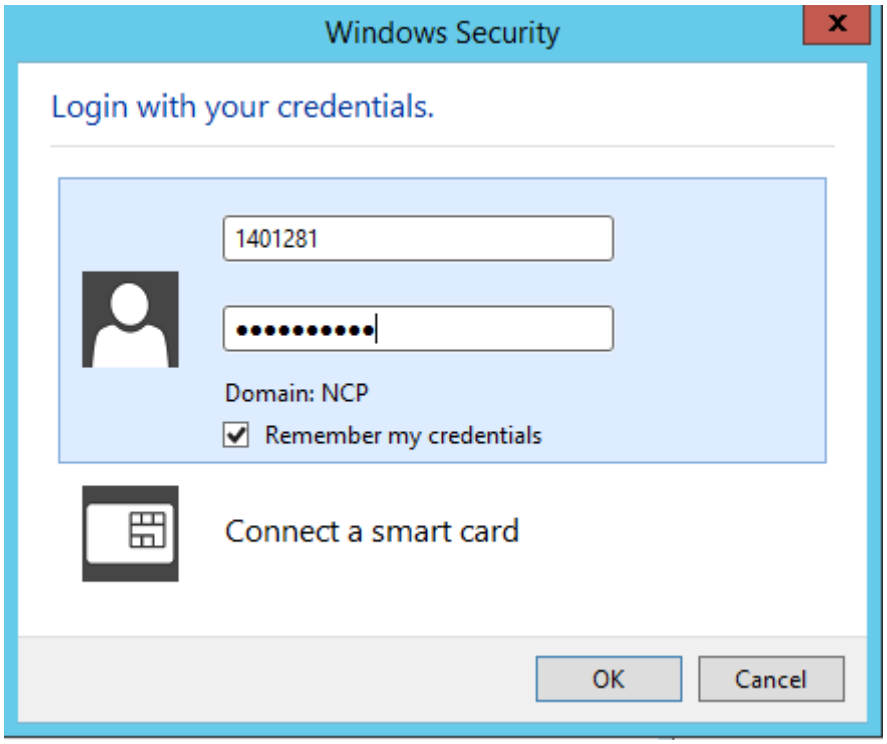
Käyttäjän täytettyä ensimmäisen lomakkeen hänen tulee painaa ”Käynnistä” -näppäintä. Tätä näppäintä painaessa käyttäjälle avautuu toinen sovelluksen ikkunoista, joka on toteutettu konsoli-ikkunana (kuva 7). Konsoli-ikkunasta käyttäjän tulee valita yhdistettävä CRM:n palvelin tai lisätä uusi yhdistettävä palvelin

seuraamalla konsoliin piirtyviä ohjeita. Ikkuna päätettiin toteuttaa käyttäen tätä tekniikkaa, koska CRM SDK:n mukana tulevassa esimerkkikoodissa palvelimen valinta oli toteutettu samalla tavalla, eikä tämän muuttamiseen nähty mitään syytä.



Kuva 7. Rajapintasovelluksen palvelimen valinta.

Kuvan mukaisesti konsoliin syötetään numero listassa olevista CRM palvelimista, johon rajapinta halutaan yhdistää. Jos haluttua palvelinta ei löydy, konsoliin voi syöttää numeron 0 ja lisätä uuden palvelimen tiedot. Palvelimia voi lisätä yhteensä 9. Syötettyään palvelimen numeron ja painamalla enter-näppäintä käyttäjälle avautuu rajapinnan viimeinen ikkuna.



Kuva 8. Rajapintasovelluksen sisäänkirjautuminen.

Lopuksi käyttäjälle avautuu tavanomainen Windows Security -ikkuna, jolle annetaan CRM-palvelimen käyttäjätiedot (kuva 8). Käyttäjän tulee omistaa muokkausoikeudet valitulle CRM-palvelimelle. Sovellus käynnistyy ilman muokkausoikeuksia, mutta halutut tietotyypit eivät siirry niiden puuttuessa.

Kokonaisuudessaan käyttöliittymän toteutuksessa pyrittiin visuaalisesti yksinkertaiseen lopputulokseen. Toimeksiantajalta saatujen tietojen mukaan rajapintaa ei tule käyttämään usea eri henkilö, joten käyttöliittymän ulkonäköön ei kulutettu ylimääräisiä resursseja. Toteutuksen yhteydessä ajatuksena oli koko ajan, että käyttö opetetaan korkeintaan kahdelle eri henkilölle. Nämä kaksi henkilöä olisivat tietotekniset taidot omaavia, joten heille tällaisen käyttöliittymän pystyisi ohjeistamaan nopeasti.

4.2 Toiminnallisuus

Rajapinnan toiminnallinen osuus toteutettiin omana kokonaisuutena irrallaan käyttöliittymästä. Toiminnot liitettiin käyttöliittymään myöhemmin molempien ollessa valmiita. Toiminnoilla tarkoitetaan käyttöliittymästä valittavia tiedonsiirtotyyppien siirtoa. Ohjelmiston koodissa toiminnot koostuivat viidestä erillisestä algoritmista. Tämän luvun tarkoitus on syventyä tarkemmin siihen, mistä vaiheista algoritmit koostuvat ja kuinka ne toimivat.

Opinnäytetyön toiminnallisen osuuden alussa oli suunnitelmana toteuttaa yksi algoritmi, joka suorittaisi kaiken tiedonsiirron valittujen tiedonsiirtotyyppien perusteella. Tiedonsiirtotyyppien kohdat muistuttavat niin paljon toisiaan, että ilman testausta niiden olisi voinut olettaa toimivan yhdellä algoritmilla. Ennen tarkempaa tutustumista tiedonsiirron suunnallakaan ei oletettu olevan vaikutusta toimivuuteen. Lopulta huomattiin, että algoritmeja on myös helpompi muokata ja testata, jos ne on rakennettu yksilöllisesti. Sovellukseen tehdyt mallit eivät myöskään olleet yhteensopivat molempiin suuntiin menevissä siirroissa. Algoritmit päätettiin lopulta toteuttaa omina toimintoina juuri näistä syistä. Niiden todettiin myös olevan helpompi yhdistää käyttöliittymään erillisinä ratkaisuina.

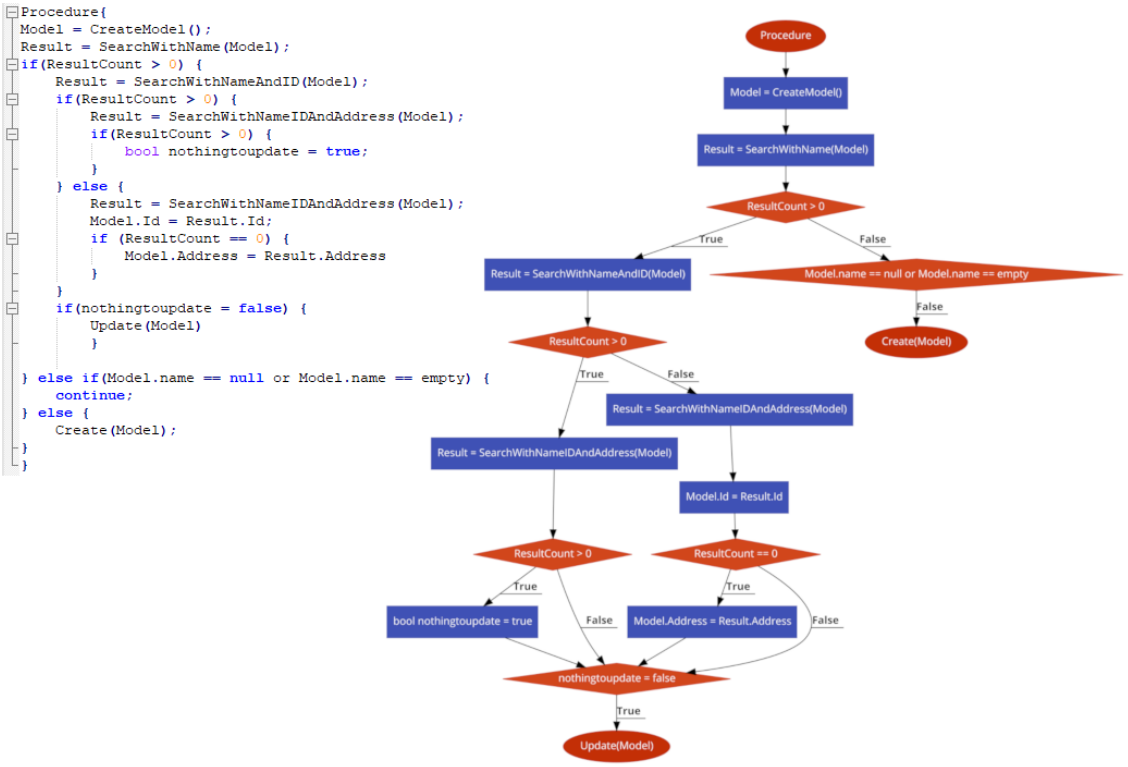
Toteutus aloitettiin kokeilemalla CRM SDK:sta löytyvien siirtofunktioiden toimivuutta. Kehityspaketin alkuperäinen toiminto loi CRM-järjestelmään uuden asiakkaan sen käynnistyessä ja poisti tämän asiakkaan sammuessaan. Toimintoa päätettiin laajentaa testivetoisella kehityksellä, koska alkuperäinen toiminto läpäisi jo ensimmäisen testitapauksen. Ensimmäisessä testissä haluttiin luoda käyttäjä CRM-järjestelmään. Funktiota laajennettiin luomaan kyseinen käyttäjä OHOS-järjestelmän tietokannasta löytyvillä tiedoilla. Laajennuksen onnistuessa huomattiin toteutuksen olevan helppoa tekemällä yksi kohta kerrallaan.

Toiminnot koostuivat useista eri kohdista. Toimintojen kohdat muistuttavat hyvin paljon toisiaan tiedonsiirtotyypistä riippumatta. Vaiheiden määrä kuitenkin vaihtelee tiedonsiirtotyypistä ja siirron suunnasta johtuen. Jokaisesta algoritmista on luotu vuokaavio vaiheiden selventämiseksi. Vuokaavioista ei kuitenkaan käy ilmi, että jokainen algoritmi luo aluksi listan tutkittavasta datasta, jonka jälkeen kyseistä dataa lähdetään selaamaan läpi vuokaavion mukaan. Vuokaaviot on luotu algoritmien pseudokoodista käyttämällä automaattista generaattoria.

4.2.1 Asiakastietojen siirto CRM-järjestelmään

Asiakkaiden siirtoa varten OHOS-järjestelmästä CRM-järjestelmään algoritmi vaati 16 eri vaihetta, jotka ilmenevät proseduurin vuokaaviosta (katso kuva 9). Jokainen vaihe on tärkeä tiedon yksilöimistä ja oikeaa paikkaa varten. Kappaleessa käydään tarkkaan läpi, mitä vaiheet tekevät ja paneudutaan algoritmin yksityiskohtiin.

CRM-järjestelmään asiakkaita siirtävä algoritmi hakee ensiksi OHOS-järjestelmän tietokannasta kaikki sieltä löytyvät asiakkaat. Tämän jälkeen se alkaa tarkastella listaa kohta kerrallaan vuokaavion mukaan.



Kuva 9. OHOS asiakastietojen siirron pseudokoodi ja kaavio.

Pseudokoodissa on käytetty hakuun funktioita, jotka on nimetty alkamaan englannin kielen sanasta search. Hakufunktioiden toiminnasta riittää, että tiedetään niille annettavan listasta löytyvä malli, johon on tallennettu kyseisen asiakkaan kaikki järjestelmästä löytyvät tiedot. Hakufunktio palauttaa tämän jälkeen listan malleja haettavasta järjestelmästä. Algoritmeista tällaisia hakufunktioita löytyy useita ja ne on nimetty niin, että niissä ilmenee mitä mallin tietoja käytetään hakua varten. Esimerkiksi kuvan mukaisessa algoritmissa ensimmäisessä haussa haetaan mallista löytyvän nimen perusteella. Funktiot ovat rakennettu haettavan järjestelmän mukaisesti. CRM-järjestelmästä hakiessa on käytetty CRM SDK:n mukana tulleita hakufunktioita.

Kokonaisuudessaan algoritmin tarkoitus on selvittää, löytyykö haettavasta järjestelmästä kyseistä asiakasta. Se suorittaa hakuja järjestelmään asiakkaan tietojen perusteella. Haut on järjestelty niin, että aluksi haetaan pelkästään nimen perusteella, jonka jälkeen haun ehtoja kasvatetaan. Nimen perusteella hakemisessa on tärkeä huomata, että haettavan mallin nimen ollessa tyhjä siirrytään suoraan seuraavaan listan malliin. Tämä on tärkeää siksi, koska CRM-järjestelmästä löy-

tyy useita asiakkaita, joille ei ole asetettu yrityksen nimeä, mutta esimerkiksi puhelinnumero löytyy. Tällaisten tietojen yksilöiminen on haastavaa. Pelkän puhelinnumeron perusteella on mahdotonta tietää, onko se jo olemassa olevan yrityksen numero vai pelkkä numero, joka on haluttu laittaa järjestelmään talteen.

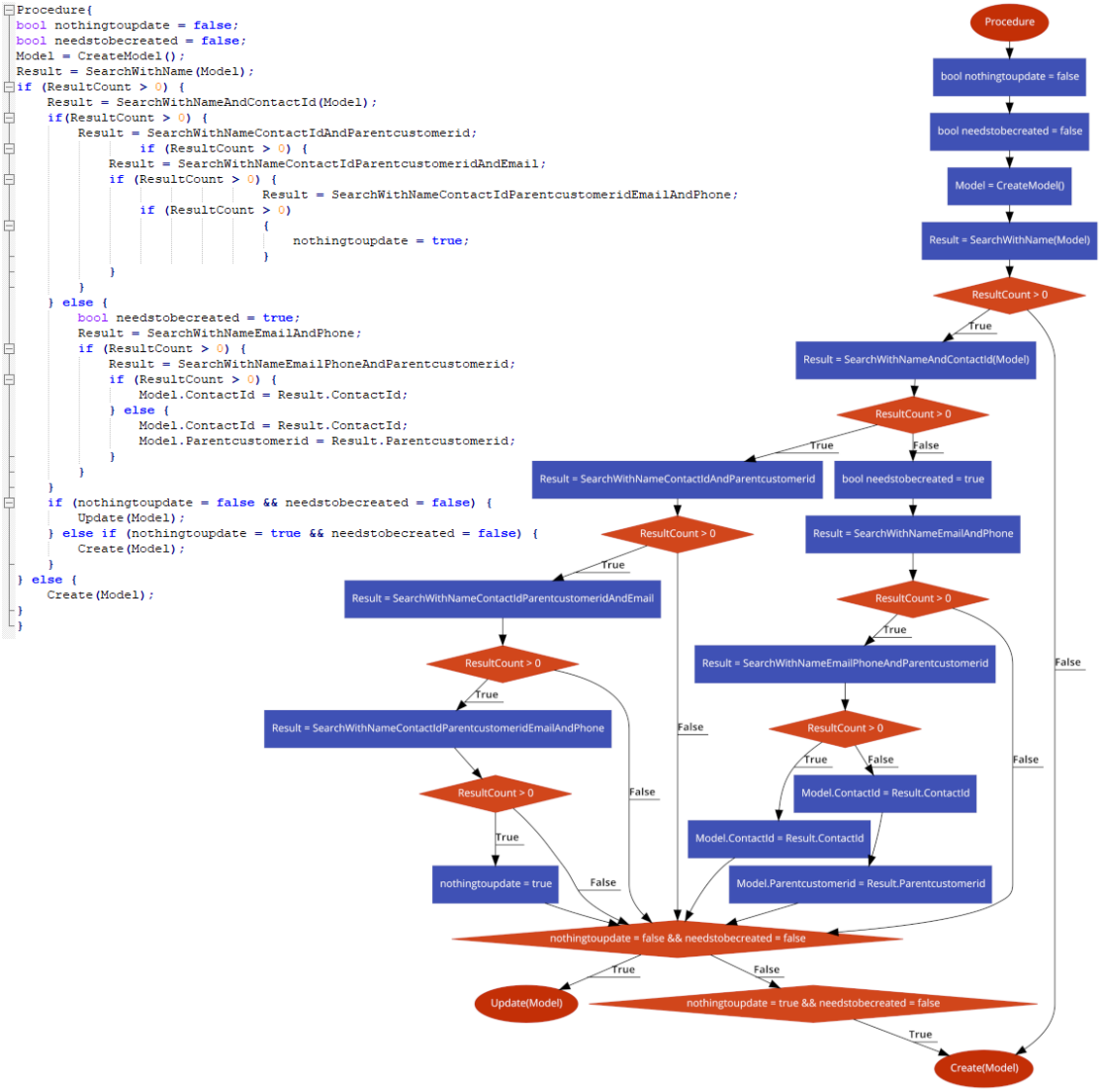
Toinen yksityiskohta algoritmissa on nimen ja ID-numeron perusteella hakiessa. Järjestelmiä käytettäessä voi olla mahdollista, että sama yritys lisätään molempiin järjestelmiin rajapinnan ollessa lepotilassa. Tällaisessa tilanteessa molemmista järjestelmistä löytyy yritys muuten samoin tiedoin paitsi ID-numeron kohdalta, koska molemmat järjestelmät luovat ID-numeron automaattisesti uuden tiedon tullessa järjestelmään. Nämä tilanteet algoritmi ratkaisee niin, että OHOS-järjestelmän ID korvataan CRM:stä löytyvällä ID:llä, jotta jatkossa tiedot ovat synkronoitu.

CRM-järjestelmän asiakastietojen siirtoa varten oleva algoritmi oli ensimmäinen, joka opinnäytetyöhön saatiin valmiiksi. Algoritmin rakennetta käytettiin myöhemmin OHOS-järjestelmän asiakastietoja varten, koska sen tiedettiin koostuvan hyvin paljolti samoista tiedoista. Algoritmi rakentui lopulta testivetoista kehitystä soveltaen. Vaiheet rakennettiin yksi kerrallaan pyrkimällä yksilöimään haettavia tietoja entistä tarkemmin. Kehityksen kulkiessa poikkeuksien, kuten tyhjien nimien ja ID-numeroiden eroavuuksien, suhteen onnistuttiin luontevasti löytämään ratkaisut. Testivetoisen kehityksen ansiosta poikkeukset löydettiin tehokkaasti ja niihin pystyttiin puuttumaan välittömästi, eikä algoritmia tarvinnut muokata myöhemmin.

4.2.2 Yhteyshenkilötietojen siirto CRM-järjestelmään

Yhteyshenkilöiden siirtoa varten oleva algoritmi vaati eniten vaiheita kaikista algoritmeista. Yhteyshenkilöitä yksilöidään CRM-järjestelmässä henkilöiden oman ID-numeron perusteella ja heidän yrityksensä ID-numeron perusteella. Lisäksi yhteyshenkilöiden suhteen pelkkä nimi ei riitä yksilöimään heitä, sillä saman nimisiä ihmisiä voi olla useita työskentelemässä eri yrityksissä, toisin kuin yrityksiä, joita ei ole samalla nimellä yhtä useampaa.

CRM-järjestelmästä yhteyshenkilöitä siirtävä algoritmista on luotu vuokaavio pseudokoodin perusteella (kuva 10).



Kuva 10. CRM yhteyshenkilötietojen siirron pseudokoodi ja vuokaavio.

Algoritmissa on käytetty samaa periaatetta funktioiden nimeämisessä, kuin kohdassa 4.2.1. Ne toimivat myös samalla tavalla, eli funktiolle annetaan malli, joka sisältää kaikki yhteyshenkilön tiedot. CRM-järjestelmästä hakiessa dataa, yhteyshenkilön malli sisältää myös mallin yrityksen tiedoista. Yrityksen tiedoista ei kuitenkaan tarkastella muita kuin yrityksen ID-numeroa.

Asiakastietoja varten olevan algoritmin tavoin myös yhteyshenkilöiden algoritmi pyrkii selvittämään, löytyykö tarkasteltavaa dataa siirrettävästä järjestelmästä.

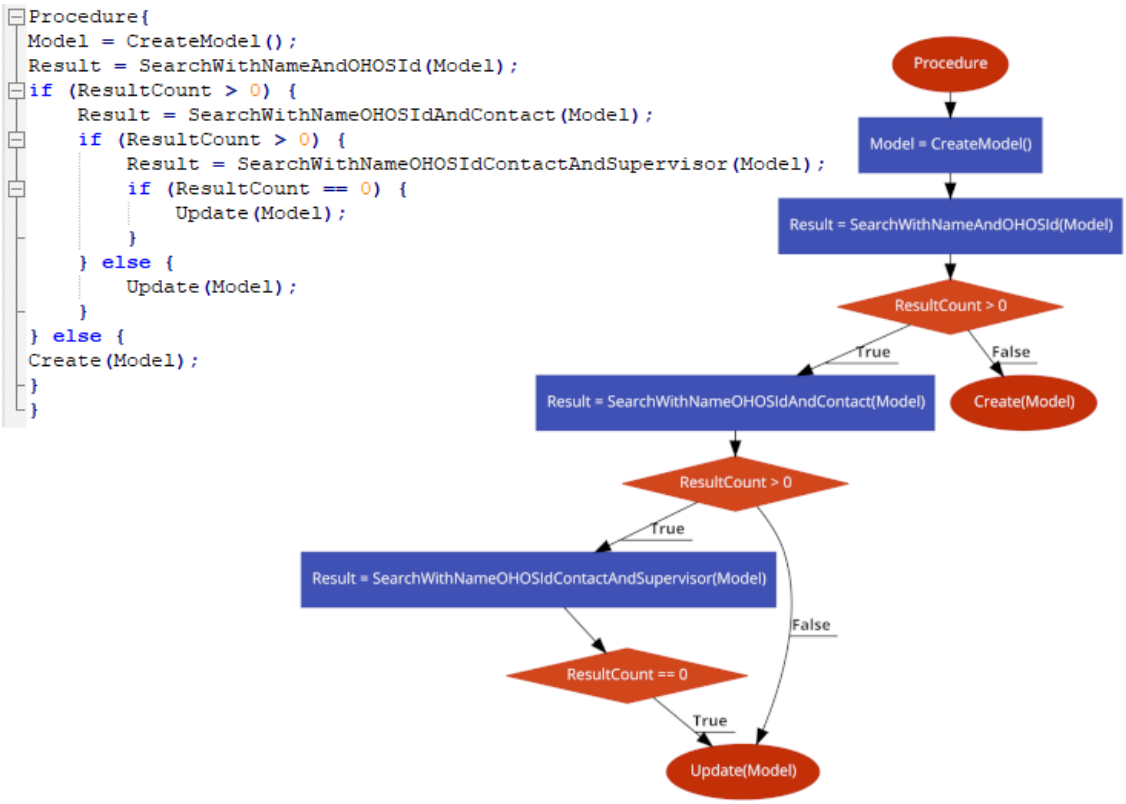
Ensimmäisessä haussa haetaan etu- ja sukunimen perusteella henkilöä. Henkilön puuttuessa se voidaan suoraan luoda. Oikea prosessi alkaa tilanteessa, jossa henkilö löytyy järjestelmästä. Tässä tilanteessa algoritmi pyrkii selvittämään, onko kyseessä kokonaan uusi henkilö, jolla vain sattuu olemaan sama etu- ja sukunimi, kuin jollain toisella järjestelmässä jo olevalla henkilöllä. Tilannetta selvitetään tarkastelemalla henkilön ID-numeroa. Henkilökohtaisen ID:n täsmätessä selvitetään myös muut henkilön tiedot: yrityksen ID-numero, sähköpostiosoite ja puhelinnumero. Jos jokin tiedoista poikkeaa, se päivitetään järjestelmään. Henkilökohtaisen ID-numeron poiketessa tarkastetaan ensin pelkästään nimen, puhelinnumeron ja sähköpostin perusteella, löytyykö henkilöä toisella ID-numerolla järjestelmästä. Henkilön löytyessä selvitetään vielä, täsmääkö hänen yrityksensä ID-numero. Jos kaikki muut tiedot, kuin henkilökohtainen ID täsmäävät, todetaan henkilön olevan sama ja synkronoidaan hänen ID-numero järjestelmien kesken. Yrityksen ID-numeron poiketessa todetaan henkilön yrityksen vaihtuneen ja vaihdetaan hänelle uuden yrityksen tiedot, sekä synkronoidaan henkilökohtainen ID.

Kokonaisuudessaan algoritmin toteutus mukaili hyvin paljon asiakkaita varten tehtyä algoritmia. Toteutus alkoi kopioimalla asiakkaiden algoritmin alku ja rakentui loppuun testivetoista kehitystä mukaillen. Algoritmia käytettiin myöhemmin mallina OHOS-järjestelmään siirtävän algoritmin kehityksessä, koska sen tiedettiin tottelevan samoja sääntöjä kuin tämä algoritmi. Algoritmista löytyy toistaiseksi ylimääräisiä vaiheita. Esimerkiksi viimeiset yrityksen ID-numeron täsmäykset eivät ole algoritmin toiminnan kannalta tärkeät, sillä OHOS-järjestelmä ei koskaan kehittynyt niin pitkälle, että järjestelmä tallentaisi yhteyshenkilöitä tietokantaansa.

4.2.3 Opinnäytetöiden siirto CRM-järjestelmään

Opinnäytetöiden siirtoa varten oleva algoritmi oli tämän rajapintasovelluksen tärkein osa. Toimeksianto koko opinnäytetyötä varten tuli juuri näiden tietojen siirtämistä varten. Asiakas halusi saada OHOS-järjestelmään tulevat opinnäytetyöt talteen CRM-järjestelmään, jotta niistä voitaisiin tehdä yhteenvetoja, minne opinnäytetöitä tehdään ja kuinka paljon. Asiakkaita ja yhteyshenkilöitä varten rakennettuihin algoritmeihin verrattuna opinnäytetöiden algoritmi oli yksinkertaisin ja

vaati huomattavasti vähemmän vaiheita. Vaiheiden lukumäärä käy hyvin ilmi vuokaaviosta ja pseudokoodista (kuva 11).



Kuva 11. Opinnäytetyötietojen siirron pseudokoodi ja vuokaavio.

OHOS-järjestelmää tehdessä ajatuksena oli, että opinnäytetöitä ei olisi samalla otsikolla yhtä useampaa. Tilanne ei kuitenkaan välttämättä ole aina niin yksinkertainen. Tästä syystä opinnäytetöille luodaan OHOS-järjestelmässä jokaista työtä varten työkohtainen ID-numero. Algoritmin kannalta pelkän ID-numeron vertaaminen pitäisi riittää opinnäytetöiden yksilöimiseen, mutta sitä tehtäessä ei voitu olla täysin varmoja, lisääkö koulun henkilökunta CRM-järjestelmään opinnäytetöitä. Tällaisessa tilanteessa CRM-järjestelmä loisi opinnäytetyölle oman ID-numeron ja opinnäytetyöllä voisi mahdollisesti olla sama nimi, kuin jollain OHOS-järjestelmän opinnäytetyöllä. Tästä syystä ensimmäisessä tarkastelussa verrataan sekä nimeä että ID-numeroa. Mikäli nämä tiedot eivät täsmää, algoritmi huomaa sen ja luo CRM-järjestelmään uuden opinnäytetyön.

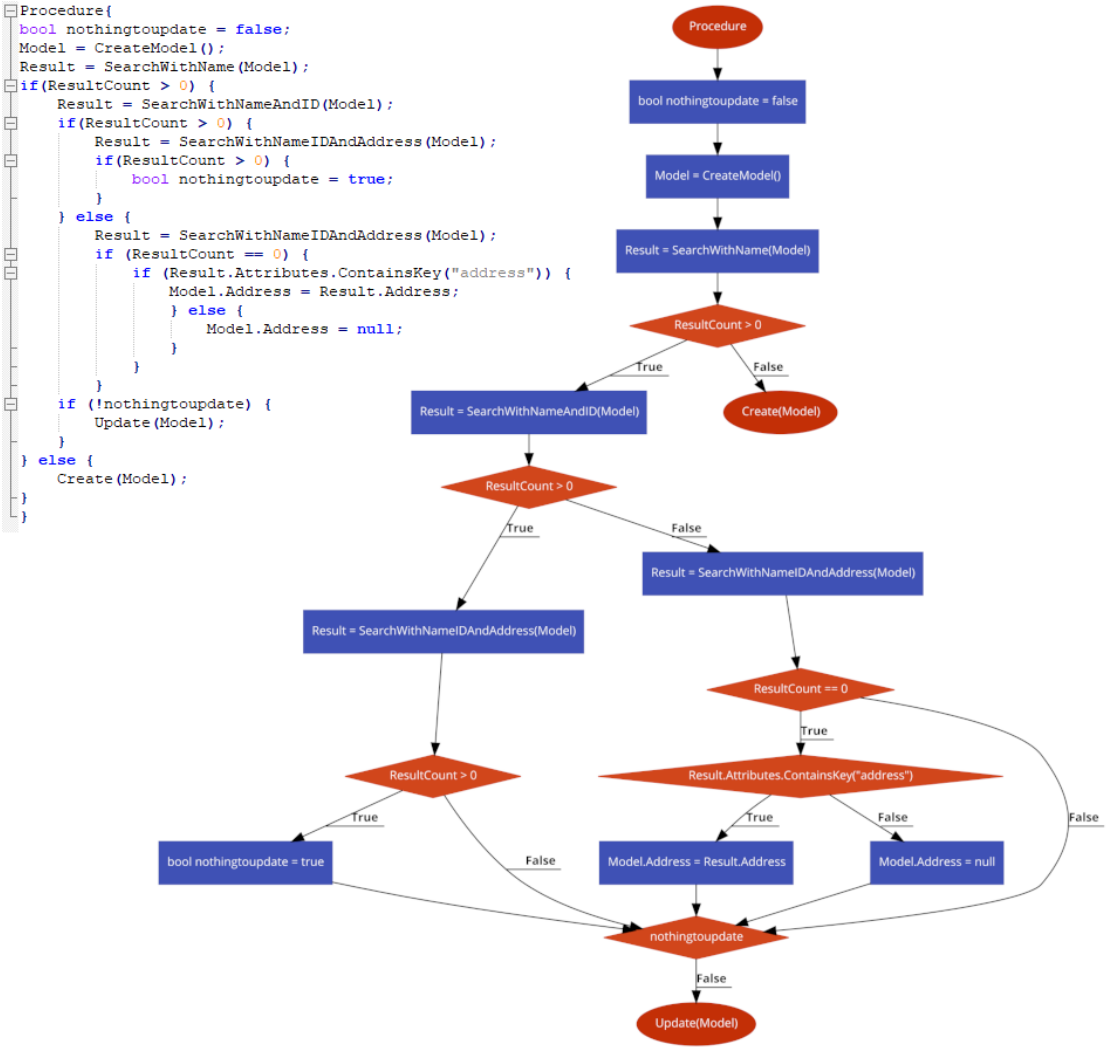
Seuraavat toiminnot vertaavat opinnäytetyön yhteyshenkilöä ja ohjaajaa. Ne ovat ainoita tietoja, joiden oletetaan vaihtuvan opinnäytetyön aikana. Siksi algoritmi

tarkastaa ensiksi, onko opinnäytetyön yhteyshenkilö vaihtunut. Jos se on vaihtunut opinnäytetyön aikana, algoritmi päivittää uudet tiedot CRM-järjestelmään. Yhteyshenkilön täsmätessä tarkastellaan vielä molempia yhteyshenkilöä sekä ohjaajaa. Näiden tietojen poiketessa päivitetään uusi ohjaaja kohdejärjestelmään ja tiedot pysyvät synkronoituna.

Opinnäytetöiden algoritmi oli nopein ja yksinkertaisin toteuttaa. Niiden tietojen pysyessä samana algoritmi vaati selvästi vähemmän vaiheita kuin muut algoritmit. Algoritmia ei kahden edellisen tapaan tarvinnut käyttää OHOS-järjestelmän suuntaan tulevaa algoritmia varten mallina, sillä opinnäytetöitä ei siirretä missään tapauksessa CRM-järjestelmästä OHOS-järjestelmään. Tiedon siirron yksisuuntaisuus auttoi algoritmin toteutuksessa, sillä siihen ei tarvinnut kiinnittää huomiota.

4.2.4 Asiakastietojen siirto OHOS-järjestelmään

Opinnäytetyön alussa ajatuksena oli, että CRM-järjestelmään tallennettuja tietoja voisi yhtä hyvin käyttää OHOS-järjestelmässä; tietoja voisi käyttää esimerkiksi asiakasyritysten tietojen automaattiseen täyttämiseen opinnäytetyössä vaadittavalle aihe- ja ohjauslomakkeelle. Vaikka OHOS-järjestelmän toiminnan kannalta tiedot eivät ole mitenkään oleellisia, olisi tarpeetonta olla käyttämättä hyödyksi tallennettuja tietoja myös OHOS-järjestelmässä. Tätä varten rajapinta tarvitsi algoritmin tietojen siirtämiseen myös toiseen suuntaan. Algoritmin toteutus olikin suoraviivaista, sillä sen tiedettiin noudattavan samoja sääntöjä kuin toiseen suuntaan kulkevan algoritmin. Tästä syystä valmiina olevaa algoritmia voitiin käyttää mallina vaiheita hieman muuntelemalla. Algoritmiin tehdyt vaiheet tulevat hyvin selville vuokaaviosta ja pseudokoodista (kuva 12).



Kuva 12. CRM asiakastietojen siirron pseudokoodi ja vuokaavio.

Vertaamalla algoritmin vuokaaviota ja pseudokoodia sen toiseen suuntaan (Katso kuva 10) toimivaan algoritmiin on helppo huomata, että valtaosa vaiheista ovat täysin samoja. Poikkeuksia algoritmien väliltä löytyy vain kahdesta kohtaa. Ensimmäisenä on ID-numerojen vaihdokset, mitkä löytyvät CRM-järjestelmään menevien tietojen algoritmista. OHOS-järjestelmään mennessä vaihdosta ei tarvitse tehdä, koska tiedon tullessa sinne CRM-järjestelmän ID-numero saa korvata OHOS-järjestelmän numeron. Tiedon korvautuessa tiedot pysyvät jatkossa samoina ja algoritmien toiminta on sujuvampaa.

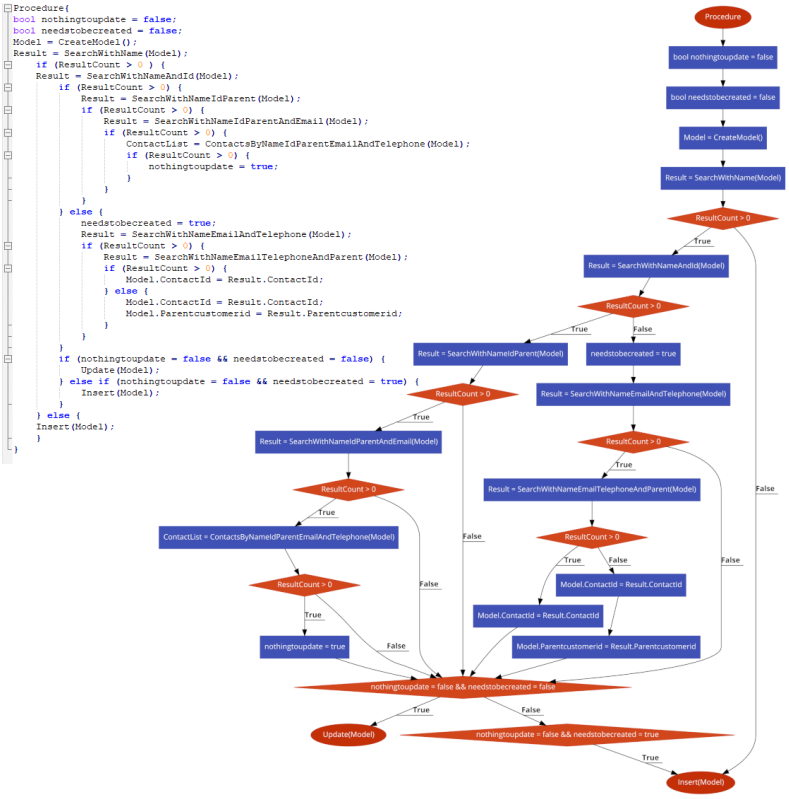
Toinen poikkeus algoritmien välillä on yrityksen osoitteen muutos. CRM-järjestelmässä voi olla yrityksiä, joiden tietoihin ei ole merkattu heidän postiosoitettaan, joten algoritmi vastaanottaa tämän datan tyhjänä kenttänä. OHOS-

järjestelmään siirrettäessä dataa tyhjä osoite on muutettava null-arvoksi, koska järjestelmän tietokanta ei hyväksy tyhjiä arvoja.

Algoritmi toimii täysin samalla logiikalla kuin kohdassa 4.2.1 läpikäyty algoritmi. Kahden poikkeuksen lisäksi suurin ero on hakufunktioissa. Koska algoritmi siirtää dataa OHOS-järjestelmän tietokantaan, on Hae, Lisää ja Päivitä -funktiot tehty käyttämällä SQL-kyselyitä, mistä syystä algoritmi toimii ajallisesti hieman hitaammin kuin CRM-järjestelmään siirrettäessä. SQL-kyselyiden takia jokainen funktio joutuu avaamaan yhteyden tietokantaan ja sammuttamaan sen kyselyn jälkeen. Tämä aiheuttaa hieman ylimääräistä aikaa algoritmin suoritukseen, mutta ajallisesti rajapinnan ei vaadittu olevan tehokkain mahdollinen.

4.2.5 Yhteyshenkilötietojen siirto OHOS-järjestelmään

Yhteyshenkilöitä varten oleva algoritmi oli viimeinen, joka tehtiin opinnäytetyön aikana. Algoritmi on kopio CRM-järjestelmään menevästä algoritmista (katso kuva 11). Algoritmien välillä ei ole yhtään eroavaa vaihetta. Sen vaiheet selviävät tutkimalla pseudokoodia ja vuokaaviota (kuva 13).



Kuva 13. CRM yhteyshenkilötietojen siirron pseudokoodi ja vuokaavio.

Algoritmi käy läpi samat vaiheet kuin CRM-järjestelmän versio. Samalla tavoin kuin kohdan 4.2.4 algoritmi, tässä algoritmissa hakufunktioissa on jouduttu käyttämään SQL-kyselyjä. Tässä algoritmissa ei ole kuitenkaan tarvetta muuntaa tyhjiä kohtia null-arvoiksi, koska toisin kuin yritysten osalta, CRM-järjestelmään syötetyillä yhteyshenkilöillä ei ole tyhjiä kohtia.

4.3 CRM-järjestelmän ja SDK-paketin konfigurointi

CRM-järjestelmä oli opinnäytetyön alussa muokattu pelkästään Karelia-ammattikorkeakoulun henkilökunnan tarpeiden perusteella. Tämä opinnäytetyö kuitenkin vaati muokkauksia järjestelmän asetuksiin, jotta siitä saatiin vaatimusmäärittelyn vaatimusten mukainen. CRM-järjestelmästä puuttui kokonaan kokonaisuus opinnäytetöiden tietojen tallentamista varten ja se täytyi rakentaa erikseen järjestelmän asetuksien kautta. Tähän tarvittiin järjestelmän pääkäyttäjän käyttöoikeudet, koska järjestelmä ei salli asetusten muokkaamista opiskelijakäyttäjien toimesta.

SDK-paketti tarjosi ohjelmoijalle kaiken tarvittavan työn aloittamiseen, mutta paketin joidenkin oletusluokkien muokkaaminen oli tarpeellista. Oleellisena muokkauksena oli uuden luokan tekeminen SDK-paketin HelperCode -nimisen kansion sisältävään MyOrganizationCrmSdkTypes.cs -tiedostoon. Tiedosto sisälsi kaikki tarvittavat luokat CRM-järjestelmän oletuskokonaisuuksia varten. SDK-paketti vaati kuitenkin luokan kirjoittamisen kyseiseen tiedostoon uuden kokonaisuuden tullessa järjestelmään, jotta sitä pystyi käsittelemään samalla tavalla kuin oletuskokonaisuuksia.

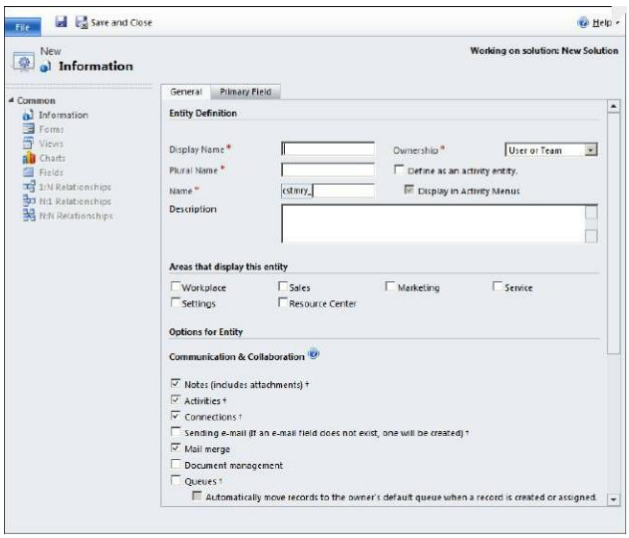
4.3.1 Uusi entiteetti

Microsoft Dynamics CRM -järjestelmässä entiteetti tarkoittaa kohdetta, jota organisaatio tarvitsee hallitakseen markkinointiosuutta. Esimerkiksi, jos organisaatio tarjoaa vakuutuksia, sen tarvitsee hallita asiakkaita, välittäjiä, menettelytapoja

sekä väittämiä (nämä ovat vain esimerkkejä entiteeteistä vakuutusliiketaloudessa), osana sen markkinointiprosessia. (Benson 2012, 99)

Microsoft Dynamics CRM sovelluksissa tulee mukana tarvittavat työkalut, jotka hoitavat valmiiksi monimutkaisten entiteettien luomiseen ja hallinnointiin. Luodessa uuden entiteetin CRM-alusta luo valmiiksi tietokantataulut datan tallentamiseen, suodatetun näkymän tietokannan tiedoista, lomakkeet tietojen hakuun ja syöttöön, sekä turvaroolit käyttäjille, jotka voivat hallita tietoja. (Benson 2012, 100)

Uusien entiteettien luominen ei vaadi käyttäjältä ohjelmointitaitoa, minkä vuoksi niitä voivat luoda kaikki käyttäjät, joilla on riittävät oikeudet. Entiteettien luominen on suoraviivaista ja se tapahtuu kokonaan CRM- järjestelmän käyttöliittymän kautta (Katso kuva 14). Entiteettejä käyttäjä pääsee luomaan valitsemalla navigointipaneelista ”Asetukset”. Tämän jälkeen Muokkaus -ryhmästä täytyy valita ”Muokkaa”, jonka jälkeen on valittava ”Muokkaa järjestelmää”. Vasemmalle puolelle ikkunaa aukeaa uusi navigointi-ikkuna, josta voi valita entiteetit. Tämän jälkeen valitaan kohta ”Uusi” joka avaa ”Uusi entiteetti” ikkunan.



Kuva 14. Uuden entiteetin luominen (Benson 2012, 101.)

Ikkunasta löytyy laaja valikoima erilaisia toimintoja entiteetin luomiseen. On tärkeää tietää mitä toimintoja tarvitaan, jotta entiteetti vastaa haluttuja vaatimuksia. Tämän opinnäytetyön toteutuksessa tarvitsee keskittyä pelkästään entiteetin nimeämiseen. Entiteetille voi antaa nimen kirjoittamalla sen kenttään ”Display

Name”. Lomake täyttää automaattisesti kentän ”Name” edellisen kentän perusteella. Tässä opinnäytetyössä ”Display Name” kenttään annettiin nimi ”Opinnäytetyö”, jolloin ”Name” kenttään tuli nimeksi ”new_opinnytety”. Sovellus ei anna kirjoittaa ”Name” kenttään ääkkösiä, koska tämä nimi tulee suoraan ohjelman tietokantaan ja tietokannassa ei voi käyttää ääkkösiä nimeämiseen.

Seuraavaksi kokonaisuudelle on asetettava kentät, joihin voi syöttää tietoa. Kenttiä pääsee lisäämään valitsemalla ikkunan vasemmalta löytyvästä navigaatiosta painamalla kohtaa ”Fields”. Microsoft Dynamics CRM-järjestelmässä jokaiselle kentälle voi asettaa erilaisia datatyyppejä, jotka määrittävät millaista tietoa kenttiin syötetään. Datatyyppejä ei voi vaihtaa myöhemmin, joten on tärkeää asettaa ne oikein ensimmäisellä yrittämällä. Esimerkkejä, mitä datatyyppejä Microsoft Dynamics CRM järjestelmästä löytyy ja mitkä niiden vastaavat SQL Server datatyypit ovat, löytyy taulukosta 1.

Taulukko 1. Microsoft Dynamics CRM datatyypit (Benson 2012, 111-112.)

CRM datatype	**SQL Server datatype**
Single line of text	Nvarchar
Option set	Int
Two options	Bit
Whole number	Int
Floating point number	Float
Decimal number	Decimal
Currency	Money
Multiple lines of text	nvarchar(max)
Date and time	Datetime
Lookup	Uniqueidentifier

Tätä opinnäytetyötä varten tietotyypeistä käytettiin pelkästään yksittäisiä tekstirivejä eli ”nvarchar”, ja ID-numeroita eli ”uniqueidentifier”. Valtaosa kentistä oli tyyppiä yksittäinen tekstirivi. Näitä kenttiä olivat muun muassa ohosid, ohjaaja, opiskelijaid, koulutusohjelma, toimeksiantaja sekä yhteyshenkilö. ID-numero kenttiä oli vain kaksi ja ne olivat nimeltään accountID ja opinnäytetyöID. Nämä

kaksi vaativat ID-numeroinnin, jotta CRM-järjestelmä pystyy yhdistämään opinnäytetyöt oikeille asiakkaille (accountID). OpinnäytetyöID taas antoi CRM-järjestelmälle mahdollisuuden yksilöidä opinnäytetyöt toisistaan.

Nämä vaiheet suoritettua CRM-järjestelmän kustomointiin rajapinta voi toimia oikein. Vaiheet ovat elintärkeitä opinnäytetöiden algoritmin toiminnan osalta. Muut algoritmit toimivat kyllä ilman näiden vaiheiden suorittamista. Jotta rajapinta täyttäisi kaikki vaatimusmäärittelyn antamat määritteet, tämä vaihe on kuitenkin tehtävä. CRM-järjestelmän helppokäyttöisen käyttöliittymän ansiosta vaihe ei vaadi tekijältä lainkaan ohjelmoinnin osaamista ja sen voi tehdä käytännössä kuka vain.

4.3.2 SDK-paketin konfigurointi

Microsoftin SDK-paketti on alun perin tehty toimimaan pelkästään CRM-järjestelmän kokeiluversion kanssa. Paketti ei ymmärrä, jos CRM-järjestelmää muokataan itsenäisesti. Tästä syystä myös pakettia pitää muokata, jos itse järjestelmään tekee isompia muutoksia.

Tässä opinnäytetyössä CRM-järjestelmään luotiin uusi entiteetti, joka sisälsi kokonaan omat datatyyppinsä. Tämä aiheutti tarpeen muokata SDK-pakettia. Pakettiin on Microsoftin puolesta ohjelmoitu valmiiksi luokat CRM- järjestelmän testiversion kokonaisuuksia varten. Tämän opinnäytetyön kannalta helpotti se, että Karelia-ammattikorkeakoulu käyttää näitä entiteettejä asiakkaiden ja yhteyshenkilöiden tallentamiseen, vaikka kokonaisuuksien nimet eivät vastaa tietoa, jota ne sisältävät. Nämä nimet ovat Account ja Contact. Tämä helpotti opinnäytetyön toteutusta, koska näille entiteeteille ei tarvinnut tehdä uusia luokkia pakettiin.

Uusi opinnäytetyö-kokonaisuus, jota käsiteltiin luvussa 4.3.1, vaatii kuitenkin luokan luomisen SDK-paketin MyOrganizationCrmSdkTypes -nimiseen CS-tiedostoon. Uuden luokan tekeminen vaatii hieman ohjelmointitaitoa, mutta sen tekeminen onnistuu kuitenkin toisen luokan kopioimisella ja nimien muuttamisella. Tässä opinnäytetyössä otettiin kopio account-luokasta ja vaihdettiin sen nimeksi new_opinnytety. Tämän jälkeen luokkaan täytyi tehdä get ja set -metodit kohdassa 4.3.1 luoduille datatyypeille. Metodeja tehdessä on tärkeää käyttää oi-

keaa datatyyppiä. Metodien luomisessa voi käyttää hyödyksi kopioidusta luokasta löytyviä metodeja. Näin metodille tarvitsee vaihtaa vain oikea nimi ja datatyyppi.

Kokonaisuudessaan uuden luokan rakentaminen on suoraviivaista. On kuitenkin tärkeää, että tekijällä on hieman ohjelmointikokemusta. Rakentaminen onnistuu ilman C# -kielen osaamista, jolla tiedosto on ohjelmoitu, mutta kielen osaaminen helpottaa nimeämistä ja oikean datatyypin valitsemista. Tehtävän voi suorittaa millä tahansa tekstieditorilla, mutta on suotavaa käyttää jotain juuri C# -ohjelmointiin soveltuvaa editoria.

5 Tulokset

Projektin vaatimusmäärittelyssä rajapinta haluttiin toteutettavan REST-arkkitehtuuria käyttäen, mutta tämä lähtökohta siirrettiin syrjään jo aikaisessa vaiheessa, kun huomattiin ajastuksen toteuttamisen haastavuus kyseistä tekniikkaa käyttäen. Rajapinta päädyttiin toteuttamaan Point-to-Point –protokollaa käyttämällä, koska sen avulla ajastus saatiin toteutettua kätevästi ja tiedonsiirto pystyttiin edelleen rakentamaan vaatimusmäärittelyn mukaiseksi. Lopputuloksena syntyi erillinen rajapintasovellus, joka on käynnissä palvelimella ympäri vuorokauden.

Vaatimusmäärittelyn tärkein vaatimus oli eri entiteettityyppien siirtomahdollisuus siten, että entiteettien siirto voidaan ottaa käyttöön tai poistaa käytöstä yksittäin. Nämä entiteetit olivat: asiakkaat, yhteyshenkilöt, opinnäytetyöt ja harjoittelut. Yhtä vaille kaikki edellä mainituista entiteeteistä saatiin ohjelmoitua rajapintasovellukseen, ja niiden siirto toimi kuten vaatimusmäärittely edellytti. Ainoastaan harjoittelutietojen siirto sovellusten välillä tuotti ongelmia, sillä OHOS-sovellukseen ei tätä opinnäytetyötä tehdessä oltu rakennettu ominaisuuksia harjoittelutietoja varten. Tästä syystä rajapintasovellukseen ei lisätty lainkaan mahdollisuutta siirtää tietoja opiskelijoiden harjoitteluista sovellusten välillä. Tämän lisäksi sovelluksessa ei ole mahdollisuutta siirtää opinnäytetöitä CRM-

järjestelmästä OHOS-järjestelmään, sillä CRM:stä ei löydy opinnäytetöitä, joita voitaisiin synkronoida toiseen järjestelmään.

Yksittäiset entiteetit oli vaatimusmäärittelyssä määritelty siten, että ensin määriteltiin mikä entiteetti on kyseessä ja mihin suuntaan sitä halutaan rajapinnan avulla siirtää. Tämän jälkeen käytiin läpi mahdolliset entiteettien yksilölliset piirteet ja kuvattiin tiedot, joita entiteettiin haluttiin sisällyttää. Esimerkiksi Asiakas-entiteetti haluttiin siirtää OHOS-sovelluksen ja CRM-järjestelmän välillä molempiin suuntiin, minkä lisäksi sen yksilöivänä attribuuttina käytettiin accountid-tietoa. Edellä mainitun tiedon lisäksi asiakkaasta haluttiin saada kirjattua tiedot yrityksen nimestä, laskutus- ja käyntiosoitteista, yritystunnuksesta, mahdollisesta www-sivustosta sekä yrityksen sähköpostiosoitteesta. Jokainen entiteetti ja niiden siirtologiikka saatiin rakennettua täysin vaatimusmäärittelyn mukaisiksi. Tarkemmat tiedot kunkin entiteetin eri attribuuteista ja yksilöivistä tekijöistä on esitetty liitteessä 1.

Rajapintasovelluksen käyttöliittymä haluttiin pitää mahdollisimman yksinkertaisena, ja se sisältääkin vain entiteettien valinnan sekä ajastuksen, joka määritetään minuutteina toimivuuden takaamiseksi. Ajastus toteutettiin siten, että jatkokehittäjillä on mahdollisuus muuttaa käytettävä aikamääre tunneiksi, mikäli se nähdään paremmaksi käytännöksi. Tämän lisäksi käyttöliittymästä löytyy näppäimet eri tietotyyppien valinnoille siten, että valittavana on mitkä tietotyypit siirretään CRM-järjestelmästä OHOS-sovellukseen sekä toiseen suuntaan.

6 Opinnäytetyön pohdinta

OHOS-järjestelmän jäädessä keskeneräiseksi opinnäytetyössä jouduttiin tekemään ratkaisuja, joita ei olisi tarvinnut ihanteellisissa olosuhteissa tehdä. Opinnäytetyön aihe oli todella haastava, sillä sen toimeksiannon yhteydessä oletettiin, että keskeneräinen järjestelmä tulee valmistumaan samaan aikaan kuin tämä opinnäytetyö. Näistä haasteista huolimatta rajapintasovellus pystyttiin toteuttamaan vaatimusmäärittelyn mukaisesti. Prosessin aikana valmistui täysin toimiva rajapinta, joka sisältää kaikki vaatimusmäärittelyssä mainitut ominaisuudet.

6.1 Jatkokehitys

Rajapinnassa tehtyjä ratkaisuja voitaisiin parantaa monellakin tapaa. Ensimmäisenä parannuskohteena on rajapinnan näkyvin osuus, eli käyttöliittymä. Kuten luvussa 4.1 mainittiin, käyttöliittymässä ei panostettu visuaalisuuteen. Siitä olisi kuitenkin mahdollista tehdä näyttävämpi pienelläkin hiomisella. Esimerkiksi valittavat painikkeet käyttöliittymässä voisivat olla samankokoisia. Toinen hieman isompi muutos olisi konsolisovelluksen liittäminen ensimmäiseen Windows Form -ikkunaan. Nämä muutokset olisivat kuitenkin vain visuaalisia eivätkä vaikuttaisi millään tavalla rajapinnan toiminnallisuuteen.

Myöskään algoritmien toiminta ei ole ihanteellinen aika- ja tilavaatimuksien suhteen. Algoritmien toiminnassa voisi käyttää hyödykseen CRM-järjestelmässä käytettävää ModifiedOn – tietokenttää. Tietokentän avulla pystyisi tarkistamaan, mitä kokonaisuuksia on muokattu viimeisen ajanjakson aikana. Kentän hyödyntäminen vaatisi kuitenkin vastaavan kentän löytymisen myös OHOS- järjestelmästä. Kenttä ehdittiin luoda järjestelmän tietokantaan, mutta järjestelmä itse ei täytä kenttää muokkausten tapahtuessa. Tästä syystä kenttää ei pystytty hyödyntämään tässä rajapintasovelluksen versiossa.

Muutosten aiheuttama työ ei kuitenkaan aiheuta tarpeeksi hyötyä rajapinnan toiminnan kannalta, joten ne jätettiin jatkokehitysideoiksi. Käyttöliittymän suhteen

muutokset on helppo toteuttaa, mutta ne eivät juurikaan paranna toimivuutta. Visuaaliset parannukset kyllä parantaisivat käyttökokemusta ja helpottaisivat käyttöä, mutta toimivuuden kannalta niillä ei ole merkitystä. Algoritmien parannukset vaatisivat useita päiviä OHOS-järjestelmän uudelleen ohjelmointia. Parannus säästäisi palvelimen resursseja huomattavasti, mutta opinnäytetyön tarkoituksena ei ollut luoda resursseja säästävää rajapintaa.

6.2 Lähdekritiikki

Opinnäytetyön alkuvaiheissa tutustuttiin tarkkaan erilaisiin lähdevaihtoehtoihin. Lähteinä haluttiin käyttää mahdollisimman paljon kirjalähteitä. Kirjoja haettiin lähinnä verkosta löytyvistä kirjatietokannoista. Kirjalähteiden suurimpana ongelmana koettiin niiden ikä, sillä tietotekniikka on kehittynyt 2000-luvulla niin nopeasti, että suuri osa kirjalähteiden sisältämästä tiedosta oli ehtinyt vanhentua jo vuosia sitten.

Vanhentuneen kirjallisuuden puitteissa opinnäytetyössä jouduttiin turvautumaan paljolti verkkolähteiden tarjoamiin tietoihin. Haettavien tietojen ollessa hyvin kaupallisiin asioihin liittyviä, lähteiksi löytyi paljon markkinoivia verkkosivuja ja blogeja. Joukkoon kuitenkin mahtui paljon informatiivisia verkkosivuja, joten tietoa on tarkasteltu monesta eri lähtökohdasta.

Vanhasta kirjallisuudesta huolimatta opinnäytetyössä on käytetty lähteinä sellaisia asioita, jotka eivät ole muuttunut vuosien varrella. Teknisiin ratkaisuihin kirjoista ei ollut apua, sillä jokainen verkkokirja itse CRM-järjestelmästä keskittyi sen vuoden 2011 tai viimeisimpään Dynamics 365 versioon. Verkkosivut taas olivat ajan tasalla olevia lähteitä, joten niitä käytettiin opinnäytetyön pääasiallisena lähteenä.

6.3 Ammatillinen kasvu

Opinnäytetyön alkaessa rajapinnat olivat melko uusi asia tekijöille. Täysin tuntematon termi se ei ollut. Niiden toiminnasta oli hyvin lähtötason tietoa ja niitä oli käytetty aikaisemmin erilaisissa projekteissa. Toimiva rajapinnan toteutus oli kuitenkin täysin uusi käsite. Tästä syystä opinnäytetyö oli haastava, mutta lopulta antoisa prosessi.

Suurimpia virheitä prosessin aikana oli rajapintasovelluksen suunnittelu, sillä siihen ei käytetty läheskään tarpeeksi aikaa prosessin alkupuolella. Suunnittelu kuitenkin korvattiin mukailemalla testivetoista kehitystä, joka on ollut tuttu työskentelytapa. Kehitystapa on todettu toimivaksi lähestymistavaksi projekteihin, joiden alkuvaiheissa ei oikeastaan ole ideaa mistä lähteä liikkeelle. Tällaisissa tilanteissa on vaikea suunnitella koko sovelluksen toimintaa ennen tekemistä. Tekemisen voi kuitenkin aloittaa valitsemalla jonkin toiminnon, jota lähteä toteuttamaan ja edetä näin vaihe vaiheelta eteenpäin.

Menetelmän korvatessa suunnittelun toteutus onnistui mutkitta. Menetelmän ansiosta ongelmat oli helppo ratkaista heti niiden ilmetessä. Opetuksena kuitenkin oli, että tekemisen vaiheet olisi ollut hyvä ottaa selkeämmin paperille ylös, jolloin itse dokumentin kirjoittaminen olisi ollut yhtä nopeaa.

Lähteet

Atwood, J. 2008. Understanding Model-View-Controller. Coding Horror – Programming and human factors. 5.5.2008. https://blog.codinghorror.com/understanding-model-view-controller/ 10.1.2019.

Bender, J. & McWherter, J. 2011. Professional Test Driven Development with C#: Developing Real World Applications with TDD. New Jersey: John Wiley & Sons, Incorporated 27.1.2019.

Benson, J. 2012. Microsoft Dynamics CRM 2011 Customization & Configuration (MB2-866) Certification Guide. Birmingham: Packt Publishing Ltd. 30.1.2019.

Chorus. 2015. Microsoft Dynamics CRM: A History. https://www.chorus.co/resources/news/microsoft-dynamics-crm-a-history 8.10.2018.

Christensson, P. 2010. SDK (Software Development Kit) Definition. TechTerms https://techterms.com/definition/sdk 22.1.2019.

Christensson, P. 2018. PPP Definition. TechTerms https://techterms.com/definition/ppp 20.1.2019.

Freeman, J. 2018. What is an API? Application programming interfaces explained. InfoWorld https://www.infoworld.com/article/3269878/apis/what-is-an-api-application-programming-interfaces-explained.html 14.1.2019.

Guru99. 2014. Unit Testing Tutorial: What is, Types, Tools, EXAMPLE. Guru99 https://www.guru99.com/unit-testing-guide.html 25.1.2019.

Hit Subscribe. 2018. TDD vs. BDD: What Are They and How Do They Differ? Remco Software Ltd. https://blog.ncrunch.net/post/tdd-vs-bdd-what-how-differ.aspx 27.1.2019.

Hoffman, C. 2018. What Is an API? How-to-Geek https://www.howtogeek.com/343877/what-is-an-api/ 14.1.2019.

Houpes T.J. 2015 What to expect from Microsoft Dynamics CRM 2015. TechTarget https://searchcrm.techtarget.com/feature/What-to-expect-from-Microsoft-Dynamics-CRM-2015 7.2.2019.

Kulkarni, K. 2019. What is Test Driven Development (TDD)? Tutorial with Example. Guru99 https://www.guru99.com/test-driven-development.html 26.1.2019.

Koutifaris, A. 2018. Test Driven Development: what it is and what it is not. freeCodeCamp. https://medium.freecodecamp.org/test-driven-development-what-it-is-and-what-it-is-not-41fa6bca02a2 26.1.2019.

Maxwell, I.A. 2018. Is Test Driven Development Right for You? Blog – Our thoughts on technology and design. 24.8.2018. https://blog.scottlogic.com/2018/08/24/is-test-driven-development-right-for-you.html 25.1.2019.

Microsoft Corporation. 2015. Administering CRM 2015 for online and on-premises

https://www.microsoft.com/en-gb/download/details.aspx?id=45022&fbclid=IwAR0EJ8qE9BxYPm5d12mdvnr5bNfyFvPtP8quQ2ui3Zw4ana98dSXCyWH7mg 7.2.2019.

Richardson, C. 2018. Pattern: API Gateway / Backend for Front-End. Microservices.io https://microservices.io/patterns/apigateway.html 17.1.2019.

Rouse, M., Ehrens, T. & Kiwak, K. 2013 CRM (customer relationship management). TechTarget. https://searchcrm.techtarget.com/definition/CRM 27.10.2018.

Rouse, M. & Churchville, F. 2016. API Management. TechTarget. https://searchmicroservices.techtarget.com/definition/API-management 16.1.2019.

Rouse, M., Nolle, T. & Li, T. 2017. application program interface (API). TechTarget. https://searchmicroservices.techtarget.com/definition/application-program-interface-API 15.1.2019.

Sandoval, K. 2016. What is the Difference Between an API and an SDK? Nordic APIS https://nordicapis.com/what-is-the-difference-between-an-api-and-an-sdk/ 22.1.2019.

Schöne, P. 2017. APIs in the world of IoT. APIfriends. https://apifriends.com/api-management/iot-api/ 18.1.2019

Shahid, U. 2018. Learn About MVC Architecture. C# Corner. https://www.c-sharpcorner.com/article/learn-about-mvc-architecture/ 18.9.2018.

Thielens, J. 2013. Without API management, the Internet of Things is just a big thing. Wired. 23.7.2013 https://www.wired.com/insights/2013/07/without-api-management-the-internet-of-things-is-just-a-big-thing/ 18.1.2019.

KARELIAN CRM-RAJAPINTA

OHOS – RAJAPINTAKUVAUS

1. Yleistä

Opinnäytetyö- ja harjoittelusovelluksen (OHOS) ja Microsoft Dynamics CRM järjestelmän välisen rajapinnan toteutus on kuvattu tässä dokumentissa. Rajapinta toteutetaan käyttämällä REST-pohjaista web service -rajapintaa. OHOS-CRM –rajapinta toteutetaan yhtenä kokonaisuutena, joka sisältää useita tiedonsiirtoja/entiteettityyppejä: Asiakkaat, yhteyshenkilöt, opinnäytetyöt ja harjoittelut. Eri tiedonsiirtotyypit voidaan konfiguroida käyttöön tai pois käytöstä yksittäin. Tiedonsiirrot suoritetaan ajastettuna. Tiedonsiirtojen tiheyttä / ajastusta voidaan säätää tiedonsiirtosovelluksen asetuksista (esim. kerran tunnissa tai kerran vuorokaudessa). Tiedonsiirtorajapinta on kahdensuuntainen.

Tämä dokumentti sisältää Dynamics CRM 2015 on-premises – OHOS välisessä rajapintatoteutuksessa siirrettävät tietoentiteetit sekä näiden yhteydet.

2. Tietojen siirrot

Tietojen siirto CRM:stä OHOSeen:

- Organisaatio / yhteisötiedot
  Yksilöinti ulkoisen järjestelmän tunnisteella
- Yhteisön henkilöt (linkitys)
  Yksilöinti ulkoisen järjestelmän tunnisteella (henkilönumero)
- Opinnäytetyöt
- Harjoittelut

Tietojen siirto OHOSta CRM:ään:

- Organisaatio / yhteisötiedot
  Yksilöinti ulkoisen järjestelmän tunnisteella. Siirto kaksisuuntainen.
- Yhteisön henkilöt (linkitys)
  Yksilöinti ulkoisen järjestelmän tunnisteella (henkilönumero). Siirto kaksisuuntainen.
- Opinnäytetyöt
- Harjoittelut

Siirtologiikassa tyhjällä arvolla ei korvata missään tapauksessa kohdejärjestelmässä mahdollisesti olevaa arvoa.

3. Tietotyyppien siirtomäärittelyt

3.1. Asiakas (account)

Yhteisö/organisaatio-tiedot siirretään CRM:stä OHOS:en ja OHOS:sta CRM:ään. Yksilöivänä tekijänä käytetään CRM:n accountid-attribuuttia, joka tallennetaan OHOS:n yhteisötiedon lisätiedot-kenttään.

Tietojen siirto on luonteeltaan lisäävä ja päivittävä. Tyhjää tietoa ei päivitetä mahdollisen OHOS:n tallennetun tiedon päälle.

OHOS yhteisön tietoissa on valinta ”Tietojen automaattinen päivittäminen estetty”, jolla voidaan yhteisökohtaisesti estää tietojen päivittäminen rajapinnan kautta. Sama asetus estää ko. yhteisön tietojen siirron OHOS:sta CRM:ään.

Rajapinnan käyttöönoton yhteydessä OHOS:ssa jo oleville yhteisöille päivitetään CRM-ID-tiedot käyttämällä apuna y-tunnus ja nimi-tietoja.

Kenttä OHOS	Tyyppi	Kenttä CRM	CRM attribuutin nimi	Tyyppi	Näyttönimi	Lisätieto / valinnat
Nimi	Teksti	Yrityksen nimi	name	Yksi tekstirivi	Asiakkaan nimi	
Postiosoite	Teksti	Laskutus-osoite	address2_line1, address2_line2, address2_line3,	Teksti		
Käyntiosoite	Teksti	Käyntiosoite	address2_line1, address2_line2, address2_line3,	Teksti		
Y-tunnus	Teksti	Y-tunnus	new_ytunnus	Yksi tekstirivi	Y-tunnus	
URL	Teksti	WWW-sivusto	websiteurl	Yksi tekstirivi	WWW-sivusto	
E-mail	Teksti	Sähköposti	emailaddress1	Yksi tekstirivi	Sähköposti	
Ulkoisen järjestelmän tunniste	Teksti	accountid	accountid	Ensisijainen avain	Asiakas	

3.2. Yhteyshenkilö (contact)

Henkilöiden tiedot siirretään rajapinnan kautta CRM:stä OHOS:en ja OHOS:sta CRM:ään. Henkilöt lisätään OHOS:ssa henkilötiedoiksi ja henkilöt linkitetään yhteisöihin. Henkilöt yksilöidään CRM:n contactid-attribuutilla, joka tallennetaan OHOS:sen henkilön CRM-contactid –kenttään.

Tietojen siirto on luonteeltaan lisäävä ja päivittävä. Tyhjää tietoa ei päivitetä mahdollisen OHOS:en tallennetun tiedon päälle.

Henkilöiden siirrossa huomioidaan, onko henkilön tietojen päivittäminen estetty OHOS:ssa. Mikäli henkilön automaattinen tiedonsiirto on estetty OHOS:ssa, ei henkilön tietoja siirretä myöskään OHOS:sta CRM:ään.

Kenttä OHOS	Tyyppi	Kenttä CRM	CRM attri-buutin nimi	Tyyppi	Näyttönimi	Valinnat
Sukunimi	Teksti	Sukunimi	lastname	Yksi teksti-rivi	Sukunimi	
Etunimi	Teksti	Etunimi	firstname	Yksi teksti-rivi	Etunimi	
Puhelin	Teksti	Puhelin	telephone1	Yksi teksti-rivi	Työpuhelin	
E-mail	Teksti	Sähköposti	emailaddress1	Yksi teksti-rivi	Sähköposti	
Tehtävänimike	Teksti	Tehtä-vänimike	jobtitle	Yksi teksti-rivi	Tehtä-vänimike	
Yhteisö	Moniva-linta	Yrityksen nimi	parentcustome-rid	Asiakas	Yrityksen nimi	
Ulkoisen järjestelmän tunniste	Teksti					

3.3. Opinnäytetyö

Kenttä OHOS	Tyyppi	Kenttä CRM	CRM attribuutin nimi	Tyyppi	Valinnat
Aihe	Teksti	Aihe	topic	Yksi tekstirivi	
Työntantajan nimi	Teksti	Työnantaja	client	Yksi tekstirivi	
OpiskelijaID	Teksti	OpiskelijaID	studentID	Yksi tekstirivi	
Koulutusohjelma	Teksti	Koulutusoh-jelma	degreeprogramme	Yksi tekstirivi	
Ohjaaja	Teksti	Ohjaaja	supervisor	Yksi tekstirivi	
Hanke	Boolean	Hanke	project	Yksi tekstirivi	

3.4. Harjoittelu

- Tätä osiota ei toteuteta OHOS-järjestelmän puutteellisuuden vuoksi