SÄHKÖINEN MATKARAPORTTILOMAKE

TEKIJÄ: Joni Miromäki

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Joni Miromäki	
Työn nimi Sähköinen matkaraporttilomake	
Päiväys 31.5.2016	Sivumäärä/Liitteet 29
Ohjaaja(t) lehtori Keijo Kuosmanen ja ohjelmistosuunnittelija Mikko Pääkkönen	
Toimeksiantaja/Yhteistyökumppani(t) Savonia-ammattikorkeakoulu	

Tiivistelmä

Tämän opinnäytetyön aiheena oli sähköinen matkaraporttilomakesovellus, joka on integroitu CRM-järjestelmään. Työn tilaajana oli Savonia-ammattikorkeakoulu. Sovelluksen tarkoituksena on saada matkaraportit järjestelmällisempään ja yhtenäisempään muotoon aiemmin käytössä olleiden vapaamuotoisten Word-dokumenttien sijasta. Raporttilomakkeen tiedot perustuvat organisaation toiminnan ja kehittämisen arviointiin käytettävään EFQM-malliin. Osa tiedoista on tilannekohtaisia ja ne näytetään vain, jos niistä raportoidaan. Raporttien pitää olla myös tulostettavissa PDF-muodossa.

Ennen sovelluksen toteuttamista vertailtiin neljää eri toteutustapaa keskenään ja valittiin niistä sopivin sovelluksen toteuttamiseen. Vertaillut toteutustavat olivat InfoPath, Nintex Forms, ASP.NET Web Forms ja ASP.NET MVC. Lopulta toteutustavaksi valittiin ASP.NET MVC sen ominaisuuksien takia, ja koska sovelluksen pystyi yhdistämään Savonialla käytössä olevaan lomakesovellukseen.

Sovellus on toteutettu käyttämällä Microsoft Visual Studio 2015:ta. Ohjelmointikielenä käytettiin C#:ia. Käyttöliittymä on toteutettu HTML:llä, CSS:llä ja JavaScriptillä.

Työn lopputuloksena oli sovellus, joka jätettiin liitettäväksi yhteen Savonian lomakesovelluksen kanssa. Savonian lomakesovellus oli valmiiksi integroitu CRM-järjestelmään, joten sitä ei tarvinnut tehdä itse. Sovelluksella voidaan lähettää uusia raportteja, sekä katsella ja muokata lähetettyjä raportteja.

Avainsanat
ASP.NET, C#, HTML, JavaScript, MVC

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Joni Miromäki			
Title of Thesis Digital Travel Report Form			
Date	31 May 2016	Pages/Appendices	29
Supervisor(s) Mr. Keijo Kuosmanen, Lecturer and Mr. Mikko Pääkkönen, Software Developer			
Client Organisation /Partners Savonia University of Applied Sciences			

Abstract

The subject of this thesis was a digital travel report form application, which is integrated into a CRM system. The client of this project was Savonia University of Applied Sciences. The purpose of the application is to get travel reports into a more structured form instead of freeform Word documents that were used previously. The data in the forms is based on the EFQM model which is used to rate organization's activities and development. Some of the data in the form is contextual and is shown only when it is being reported about. The reports must also be available in PDF format.

Before making the application four different methods were compared to decide which one is the most suitable for the application. The methods that were compared were InfoPath, Nintex Forms, ASP.NET Web Forms and ASP.NET MVC. In the end ASP.NET MVC was chosen because of its features and because the application can be combined with the form application already used by Savonia.

The application was made in Microsoft Visual Studio 2015. The programming language used was C#. The user interface was done in HTML, CSS and JavaScript.

The result was an application that is ready to be combined with Savonia's form application. Savonia's form application was already integrated with the CRM system so there was no need to do that. The application can be used to send new reports and also browse and edit sent reports.

Keywords
ASP.NET, C#, HTML, JavaScript, MVC

SISÄLTÖ

TERMIT JA LYHENTEET

ASP.NET

Active Server Page. Microsoftin kehittämä avoimen lähdekoodin web-sovelluskehys, joka on suunniteltu dynaamisten web-sivujen, sovellusten ja palvelujen kehittämiseen. Osa Microsoftin .NET-perhettä. Tässä työssä käytetään ASP.NET:in versiota 4.6.

ASP.NET MVC

ASP.NET:iin perustuva Microsoftin kehittämä web-sovelluskehys, joka käyttää MVC (model-view-controller, malli-näkymä-ohjain) –mallia.

Bootstrap

Vapaan ja avoimen lähdekoodin HTML:ään ja CSS:ään perustuva web-sivun ulkoasuun vaikuttava kehysympäristö. Siihen sisältyy myös useita valinnaisia jQuery-liitännäisiä.

C#

Microsoftin kesäkuussa 2000 julkaistu C++:aan ja Javaan perustuva oliopohjainen ohjelmointikieli.

CSS

Cascading Style Sheets. Tyylitiedostokieli, jota käytetään rakenteisten dokumenttien, erityisesti web-sivujen ulkoasun määrittämiseen.

HTML

Hypertext Markup Language. Web-sivujen ja muiden hypertekstiä sisältävien dokumenttien sisällön ja rakenteen määrittämiseen käytettävä merkintäkieli.

JavaScript

Alunperin Netscape Communications Corporationin kehittämä, nykyään Mozilla Foundationin hallinnoima komentosarjakieli. Käytetään pääasiassa dynaamisen toiminnallisuuden lisäämiseksi web-sivuille.

jQuery

Mahdollisimman helposti ymmärrettäväksi suunniteltu avoimen lähdekoodin JavaScript-kirjasto. Siihen sisältyy esimerkiksi tapahtumia, efektejä, animaatioita ja JSON-parsinta.

Microsoft Dynamics CRM

Microsoftin kehittämä asiakkuusuhteiden hallintaan (customer relationship management) tarkoitettu sovelluspaketti.

Microsoft Visual Studio

Microsoftin kehittämä integroitu kehitysympäristö (IDE, integrated development environment). Sitä käytetään Windowsille tarkoitettujen tietokoneohjelmien, sekä web-sivujen, sovellusten ja palvelujen kehittämiseen. Visual Studiossa on sisäänrakennettuna tuki C:lle, C++:lle ja C++/CLI:lle, Visual Basic .NET:ille, C#:lle, ja F#:lle. Tässä työssä käytetty versio on Visual Studio 2015 Enterprise Edition.

XML

Extensible Markup Language. Standardisoitu merkintäkieli, jonka tavoitteena on kuvata dokumentti sekä koneelle, että ihmiselle luettavassa muodossa.

1 JOHDANTO

Tämän opinnäytetyön tavoitteena on toteuttaa sähköinen matkaraportointilomake ja sen integrointi CRM-järjestelmään. Lomakkeen käyttäjinä ovat Savonia-ammattikorkeakoulun henkilökunnan jäsenet, jotka käyvät ulkomailla esimerkiksi tutustumiskäynneillä tai opettajavaihdossa. Jokainen ulkomailla käyvä henkilö on velvollinen tekemään matkastaan raportin ennen päivärahojen ja muiden kulujen maksamista. Tähän mennessä raportit on kirjoitettu vapaamuotoisesti Word-dokumenttiin ilman minkäänlaista järjestelmällistä rakennetta.

Uusi matkaraportti on lomake, johon vaaditut tiedot kirjataan. Lomakkeen tiedot perustuvat organisaation toiminnan ja kehittämisen arviointiin käytettävään EFQM-malliin, jonka on kehittänyt EFQM-järjestö (European Foundation for Quality Management). Osa lomakkeen tiedoista on tilannekohtaisia. Esimerkiksi jos raportoidaan johtamisjärjestelmästä, avautuu aiheeseen liittyviä lisäkysymyksiä. Raporttijärjestelmään pitää pystyä kohdistamaan tehokkaasti hakuja ja tehdä koosteraportteja lomakkeen eri kohdista, esimerkiksi mitä on raportoitu muiden eurooppalaisten korkeakoulujen opetussuunnitelmista. Raporttien pitää olla myös tulostettavissa PDF-muodossa ja raportteihin olisi hyvä myös pystyä lisäämään kuvia.

CRM (customer relationship management) on asiakkaiden ja kumppaneiden tietojen hallintaan tarkoitettu järjestelmä. Lomakkeen ja Savonialla käytössä olevan Microsoft Dynamics CRM-järjestelmän välille rakennetiin integraatio. Jos matkakohde ja sen yhteyshenkilö löytyvät Savonian järjestelmästä, niiden tiedot täytetään automaattisesti lomakkeeseen. Jos kyseisiä tietoja ei löydy, ne lisätään CRM-järjestelmään. Raporteista tehdyt PDF-tiedostot viedään järjestelmään kumppaniin liittyvänä lisätietona. Lisäksi järjestelmään muodostetaan uusi entiteetti matkaraporteista, johon viedään tiedot raporttilomakkeen eri kohdista.

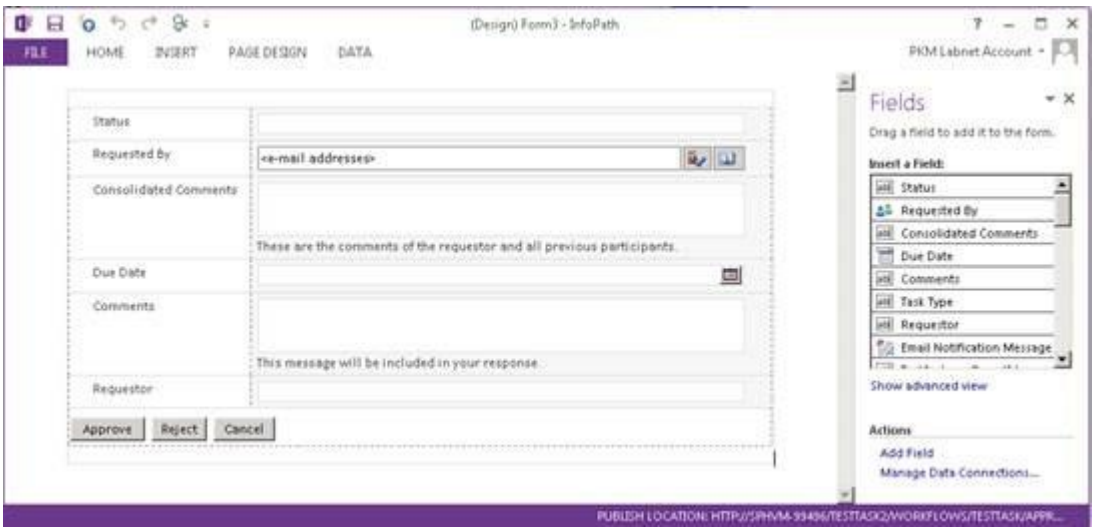
Työssä otetaan huomioon lomakkeeseen jatkossa tehtävät muutokset ja raportointimahdollisuuksien jatkokehitys, esimerkiksi ulkomaalaisten vierailijoiden Savoniasta tekemät raportit. Tämän takia täytyi selvittää hallittavuudeltaan ja ominaisuuksiltaan paras mahdollinen ratkaisu eri vaihtoehdoista.

2 TOTEUTUKSESSA KÄYTETYT TEKNIIKAT

Tässä työssä oli ensimmäiseksi tarkoitus selvittää sopivin tapa toteuttaa matkaraporttilomakesovellus vertailemalla vähintään kahta eri toteutustapaa. Vaatimuksina totutustavalle olivat mahdollisuus integroida sovellus CRM-järjestelmään, lomakkeen kehittämismahdollisuus ja mobiiliystävällisyys. Vertaillut toteutustavat olivat InfoPath, Nintex Forms, ASP.NET Web Forms ja ASP.NET MVC.

2.1 InfoPath

InfoPath on Microsoftin sähköisten lomakkeiden suunnitteluun, täyttöön ja lähettämiseen tarkoitettu sovellus. Se julkaistiin alunperin osana Microsoft Office 2003 –ohjelmistopakettia. InfoPath-lomakkeet tallennetaan XML-formaatissa. Lomakkeiden suunnittelu InfoPathilla tapahtuu helposti WYSIWYG-näkymässä (What You See Is What You Get). (Kuva 1.) Mahdollisuudet vaikuttaa lomakkeen ulkoasuun ovat kuitenkin melko rajalliset. Lomakkeet eivät myöskään toimi välttämättä kunnolla mobiililaitteilla eikä InfoPath tue mukautuvaa HTML-muotoilua. (George 2014-04-27.)



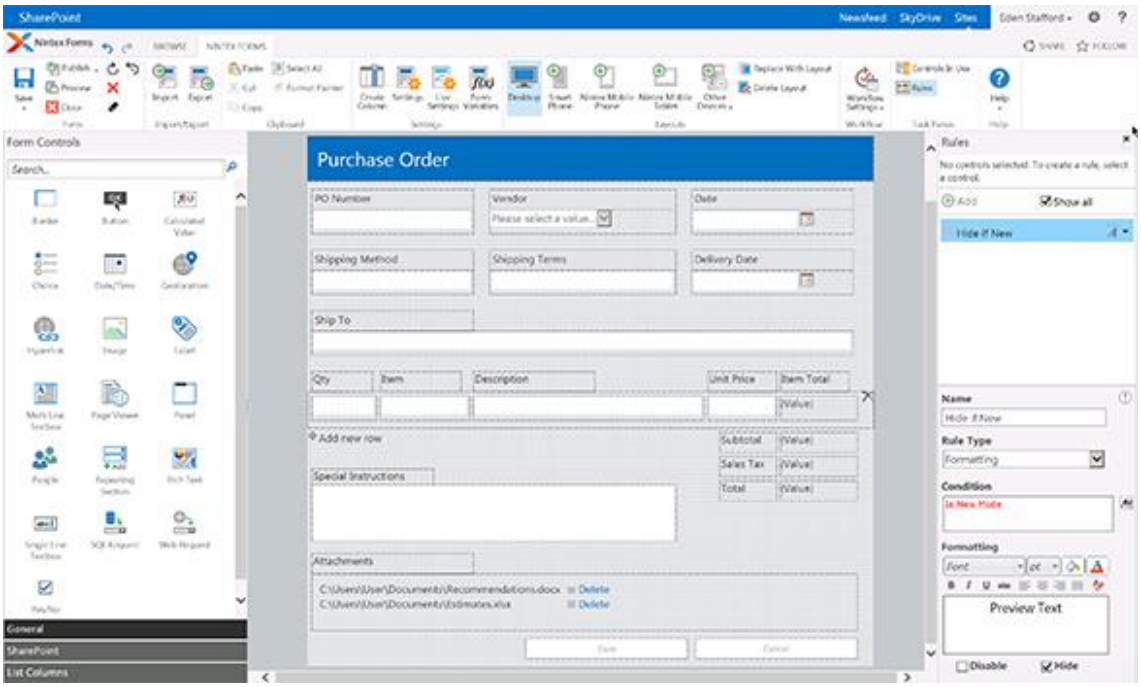
KUVA 1. InfoPath 2013 suunnittelunäkymä

InfoPath-lomakkeideet integroidaan CRM-järjestelmän kanssa Microsoft SharePoint-palvelinalustan kautta. Lomake hostataan SharePointissa ja se näytetään iframen kautta. CRM:stä otetaan ja sinne laitetaan dataa web-palvelujen ja .NET-koodin avulla. (Miley 2011-7-7.) Toinen tapa integroida InfoPath CRM-järjestelmän kanssa on lähettää lomakkeet CRM:ään sähköpostin avulla (Colter 2011-11-21).

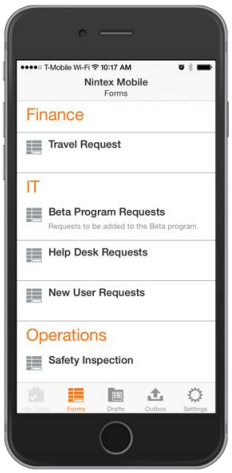
Tärkeimpänä syynä sille, miksi tässä työssä ei käytetty InfoPathia on se, että sen kehitys on lopetettu (Microsoft 2014-1-31). InfoPath 2013 jää sovelluksen viimeiseksi versioksi. Microsoft ei myöskään ole vielä julkistanut sille mitään korviketta. Vaikka InfoPathin tuki jatkuu huhtikuuhun 2023 asti, ei sovelluksen tekemisessä hyllytetyllä tekniikalla ole järkevää.

2.2 Nintex Forms

Nintex Forms on Nintex Global Ltd:n lomakesovellus. Se on yritysmaailmassa suosittu InfoPathin korvike. Lomakkeiden suunnittelu Nintexillä tapahtuu InfoPathin tavoin WYSIWYG-näkymässä (kuva 2). Toisin kuin InfoPathin lomakkeet, Nintexillä suunnitellut lomakkeet toimivat suoraan mobiililaitteilla (kuva 3). Nintex Formsilla on suora integraatio Microsoft SharePointin kanssa. Microsoft Dynamics CRM-integraatiota varten tarvitaan erillinen Nintex Connectors -lisäosa. Esteenä Nintexin käyttämiselle tässä työssä oli sen maksullisuus. Myös CRM-integraatioon tarvittavasta lisäosasta pitää maksaa erikseen. (Nintex Global Ltd 2016.)



KUVA 2. Nintexin suunnittelunäkymä

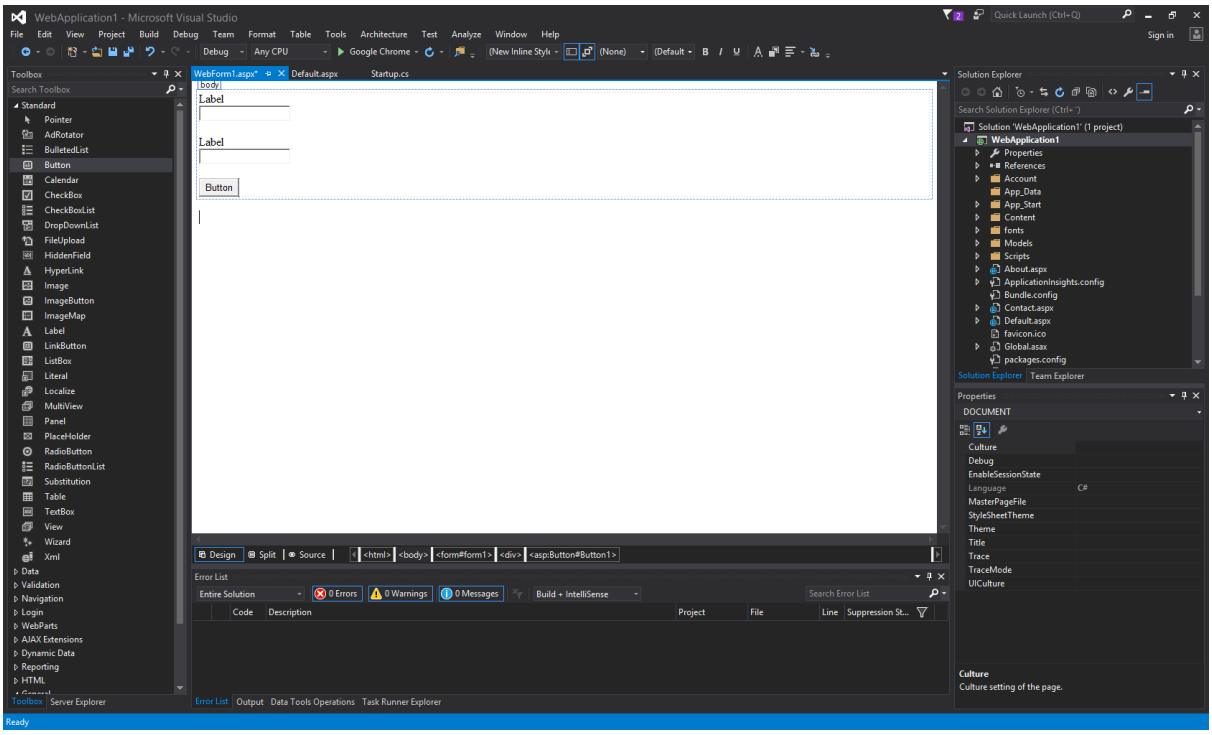


KUVA 3. Nintex-lomake mobiililaitteen näytöllä

2.3 ASP.NET Web Forms

ASP.NET Web Forms on osa Microsoftin kehittämää ASP.NET-sovelluskehystä. Web Forms -sovelluksen käyttöliittymä voidaan rakentaa aiempien vaihtoehtojen tapaan WYSIWYG-näkymässä

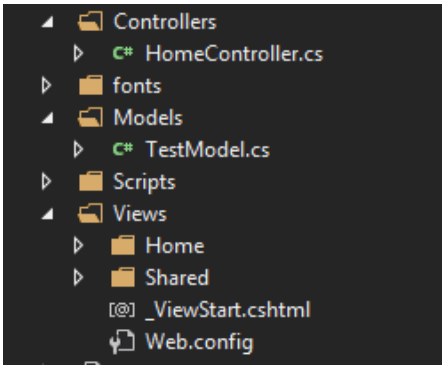
(kuva 4) tai HTML:ää muokkaamalla. Lomakkeen ulkoasua muokataan CSS-tyylien avulla ja asiakaspuolen toiminnallisuutta saadaan lisää JavaScriptillä. Palvelinpuolen kielenä voidaan käyttää C#:ia tai Visual Basic .NET:ä. (Microsoft 2016a.)



KUVA 4. ASP.NET Web Formsin suunnittelunäkymä Visual Studio 2015:ssa

2.4 ASP.NET MVC

ASP.NET MVC on Web Formsin tavoin osa Microsoftin ASP.NET-sovelluskehystä. MVC-sovelluksissa on kolme kerrosta: malli (model), näkymä (view) ja ohjain (controller) (kuva 5). Malliin haetaan ja siellä säilytetään käsiteltävää dataa. Näkymässä näytetään sovelluksen käyttöliittymä. Ohjaimen tehtävänä on keskustella näkymän ja mallin välillä. MVC-sovelluksessa käytetään samoja kieliä kuin Web Formsissakin. HTML:ää, CSS:ää ja JavaScriptiä asiakaspuolella ja palvelinpuolella C#:ia tai Visual Basic .NET:iä. (Microsoft 2016b.)



KUVA 5. Ohjaimet, mallit ja näkymät ASP.NET MVC-projektissa

2.5 Yhteenveto

Tutkituista vaihtoehdoista InfoPath ja Nintex Forms olivat poissuljettuja, joten jäljelle jäävät ASP.NET Web Forms ja ASP.NET MVC. Web Formsin etuna MVC:hen verrattuna on sen helppous. Käyttöliittymäelementit voidaan lisätä näkymään vetämällä ja pudottamalla, tapahtumavetoinen rakenne on helpommin ymmärrettävissä ja kenttien validoinnit yms. saadaan säädettyä niiden ominaisuuksista. Helppouden vastapainona on rajoittuneisuus. HTML:ään pystyy vaikuttamaan vähemmän kuin MVC:ssä, mikä tekee jQueryn kaltaisten JavaScript-kirjastojen käyttämisestä sovelluksessa vaikeampaa. Web Formsissa ei myöskään ole ennalta määrättyä projektiarkkitehtuuria, mikä saattaa aiheuttaa sen, että laajemmista sovelluksista tulee epäjärjestelmällisiä.

ASP.NET MVC:ssä käyttöliittymä tehdään kokonaan HTML:ää muokkaamalla, mikä tarkoittaa Web Formsiin verrattuna enemmän työtä, mutta tarjoaa kuitenkin enemmän vapauksia. Tämä on huomattava etu, jos ja kun sovelluksessa käytetään paljon JavaScriptiä. MVC:ssä on myös mahdollista käyttää samaa syöttölogiikkaa useammassa käyttöliittymässä, toisin kuin Web Formsissa. Yhtenä huomionarvoisena etuna MVC-sovelluksessa oli lisäksi se, että se voidaan liittää jo Savonialla käytössä olevaan lomakesovellukseen, joka on yhteydessä CRM-järjestelmään. (Hambrick 2013-12-5, Koirala, 2014-9-27, Sukesh 2014-9-26.)

Lopulta sopivammaksi tavaksi toteuttaa työ valittiin ASP.NET MVC. Syinä tälle olivat suurempi vapaus vaikuttaa HTML:ään, parempi soveltuvuus JavaScriptin kanssa ja mahdollisuus liittää tehty sovellus osaksi isompaa kokonaisuutta, missä CRM-integraatio on jo valmiina.

3 MATKARAPORTIN MÄÄRITTELY

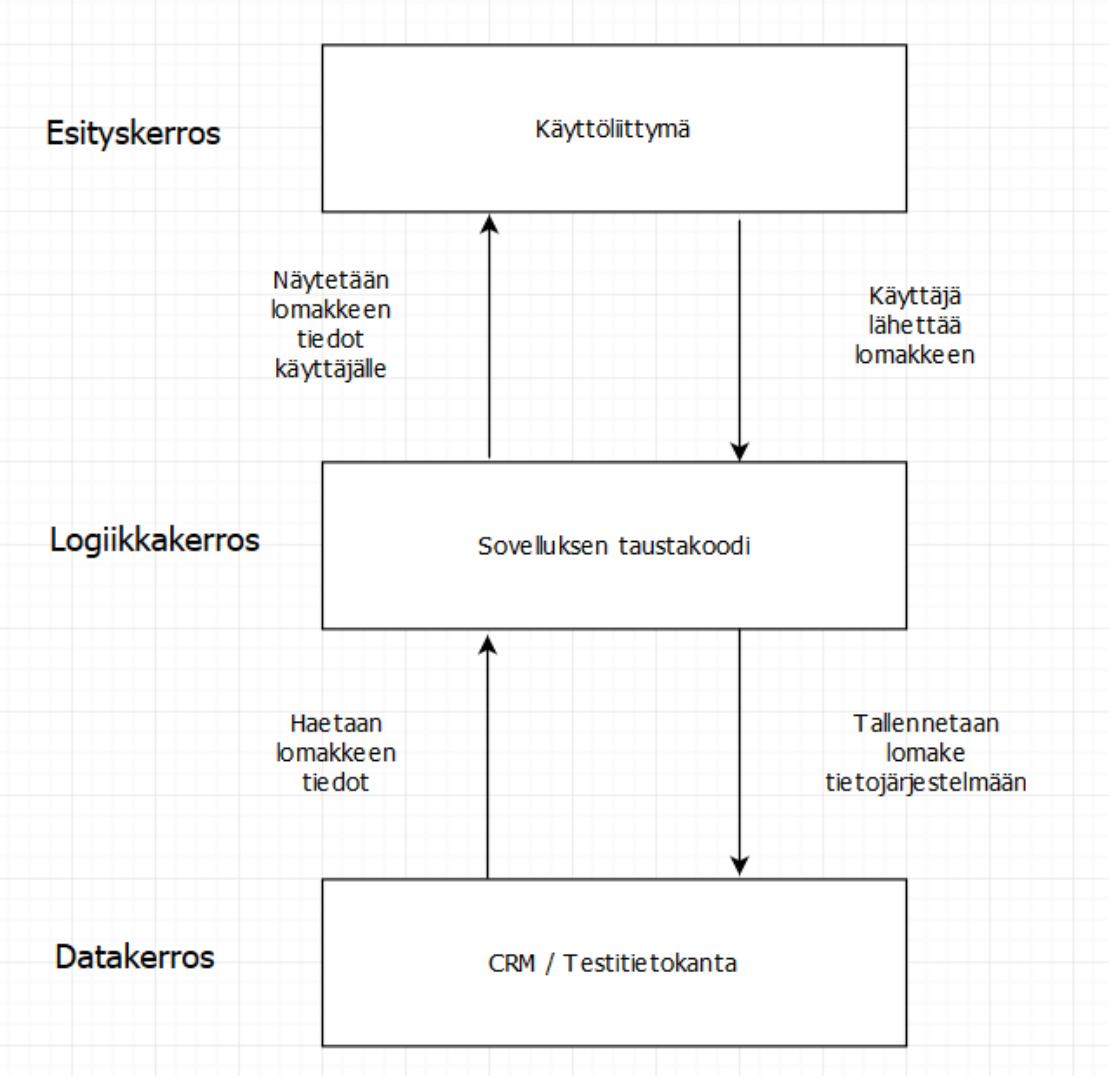
Lomakkeen ensimmäisessä osiossa kirjataan tiedot matkan tekijästä ja matkakohteesta. Matkan tekijän tarvittavat tiedot ovat etu- ja sukunimi, jotka viedään myös CRM-järjestelmään. Kohdeorganisaation tiedoista kysytään esimerkiksi vierailukohteena toimivan organisaation virallinen nimi. Kohteen nimi haetaan CRM:stä, jos sitä ei ole, niin kirjataan loput vierailukohteen tiedot. Jos tiedot ovat CRM:ssä, niin niiden kentät täytetään automaattisesti. Matkakohteen yhteyshenkilön tiedot haetaan CRM:stä erikseen sähköpostiosoitteen avulla. Jos tietoja ei löydy, kirjataan loput tiedot ylös, muuten ne täytetään automaattisesti. Muut tarvittavat yhteyshenkilön tiedot ovat etunimi, sukunimi, puhelinnumero ja titteli. Lisäksi listataan tarvittaessa myös muut tavatut henkilöt ja lisätään mahdolliset liitetiedostot, jos niitä on. Kysytään myös lupa tietojen käyttöön Savonian kehitystyössä.

Seuraavassa osiossa kirjataan ylös tiedot matkasta. Kysyttyjä tietoja ovat esimerkiksi matkasuunnitelmassa käytetty perustelu matkalle ja matkan ajankohta. Loput matkaraportin kysymyksistä perustuvat EFQM-järjestön (European Foundation for Quality Management) kehittämään Euroopan laatupalkintoon, eli EFQM-malliin. Nämä kysymykset ovat oletuksena piilotettuna ja näytetään kun käyttäjä laittaa rastin ruutuun. Raportin tarkempi määrittely löytyy liitteestä (liite 1).

4 SOVELLUKSEN ARKKITEHTUURI

Sovelluksen arkkitehtuurimallina on n-kerrosarkkitehtuuri (n-tier architecture) (kuva 6). N-kerrosarkkitehtuurin erona samankaltaiseen monitasoarkkitehtuuriin (multilayer architecture) on se, että kerrosarkkitehtuurin eri kerrokset ovat fyysisiä, kun taas tasoarkkitehtuurin tasot ovat loogisia. (Meier 2008-9-5.) Esityskerros renderöidään käyttäjälle verkkoselaimessa ja logiikka- ja datakerrokset ovat omilla erillisillä palvelimillaan.

Sovellus koostuu kolmesta kerroksesta: esityskerros, logiikkakerros (myös nimellä bisneskerros) ja datakerros. Esityskerros on mallin päällimmäinen kerros, mihin sisältyy sovelluksen käyttöliittymä. Tässä työssä sovelluksen viewit, eli näkymät kuuluvat sovelluskerrokseen. Keskimmäinen kerros on logiikkakerros. Se sisältää sovelluksen bisneslogiikan ja muun toiminnallisuuden. Sen tehtävänä on siirtää dataa pohjimmaisesta kerroksesta, eli datakerroksesta esityskerrokselle, ja toisinpäin. Logiikkakerrokseen kuuluu tässä työssä sovelluksen controllerit, eli ohjaimet. Datakerrokseen kuuluu CRM-järjestelmä tai kehitystilanteessa käytössä ollut testitietokanta.



KUVA 6. Sovelluksen arkkitehtuurikuva

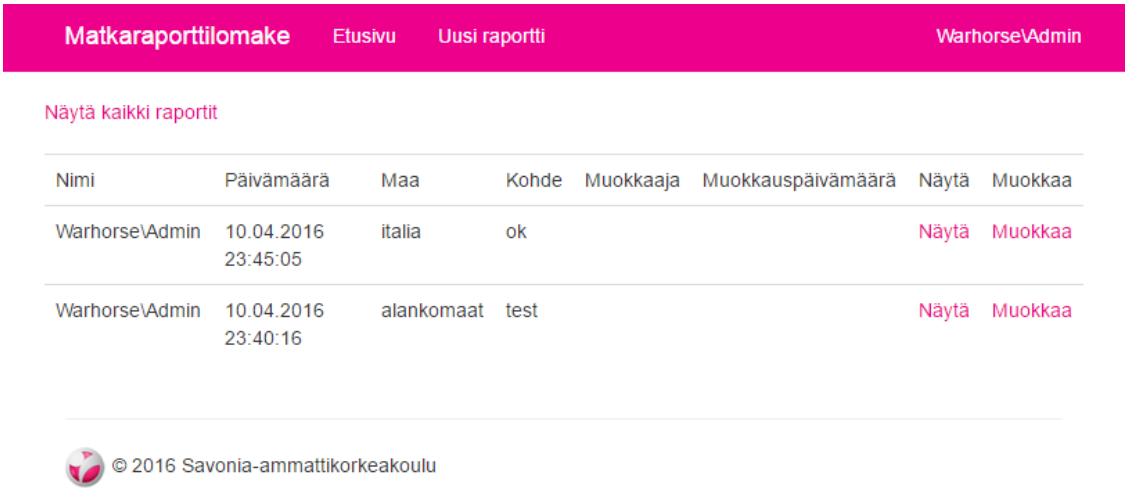
5 TYÖN TOTEUTUS

5.1 Ulkoasu

Sovelluksen ulkoasuna on muokattu Bootstrapin oletusulkoasu. Vakiotyylitiedoston ylikirjoittavassa custom.css-tyylitiedostossa ylälaidassa olevan valikkopalkin väri on muutettu mustasta ja harmaasta Savonia-ammattikorkeakoulun visuaaliseen ilmeeseen sopivampiin magentaan ja valkoiseen. Myös linkkien väri on muutettu sinisestä magentaan. Ulkoasun alatunnisteeseen on lisätty Savonia-ammattikorkeakoulun pallologo.

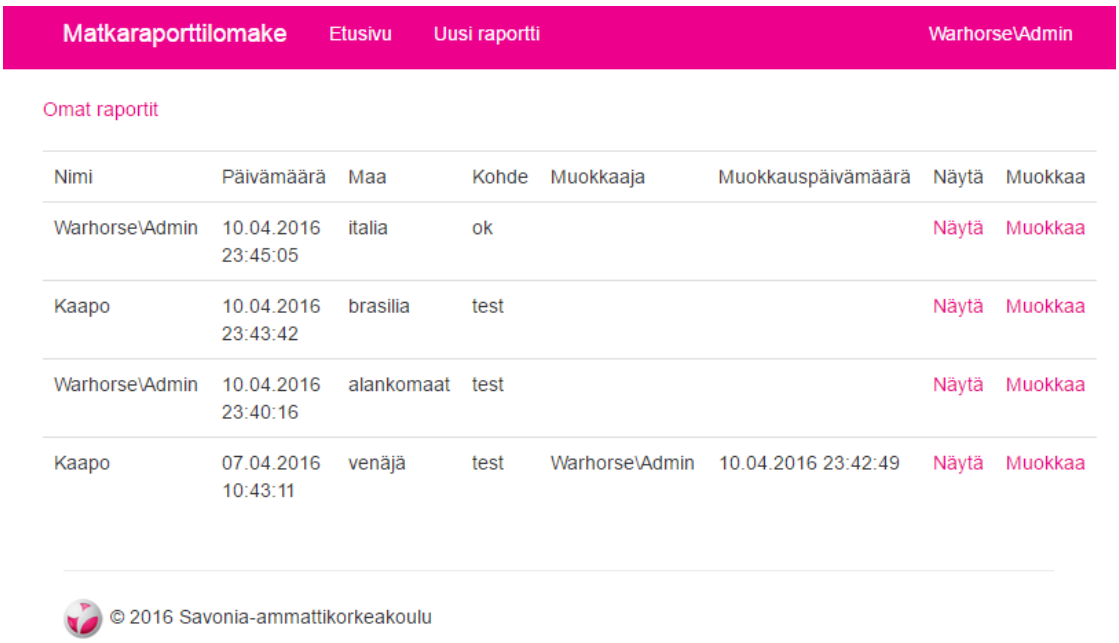
5.2 Etusivu

Etusivulla näytetään oletuksena kaikki sisäänkirjautuneen käyttäjän lähettämät raportit (kuva 7). Raportin tiedoista näytetään lähettäjän nimi, lähetyspäivämäärä, kohdemaa, kohde, muokkaajan nimi ja muokkauspäivämäärä. Jokaisen raportin kohdalla on myös linkit, joista pääsee katsomaan raportin tiedot tai muokkaamaan niitä.



KUVA 7. Etusivun oletusnäkymä

Valitsemalla ”näytä kaikki raportit” saadaan näkyviin kaikki lähetetyt raportit (kuva 8).



KUVA 8. Etusivu kaikki raportit näytettynä

Etusivun ActionResultissa (kuva 9) raporttien suodatus tapahtuu all-muuttujan avulla. Raportit haetaan CRM:stä (testitilanteessa tietokannasta) GetForms-funktiolla IndexViewModel-malliin, joka lähetetään Index-näkymään. IndexViewModelin sisältönä on lista FormDatabaseModel-olioista, joissa on etusivulla näytettävät tiedot raportista, raportin tunniste, raportin rakenteen tunniste ja raportin sisältö XML-muodossa.

```
0 references
public ActionResult Index(string all)
{
    bool getAll = false;
    if (all == "true")
    {
        getAll = true;
    }
    List<FormDatabaseModel> forms = new List<FormDatabaseModel>();
    forms = GetForms(forms, getAll);
    IndexViewModel vm = new IndexViewModel();
    vm.Forms = forms;
    return View(vm);
}
```

KUVA 9. Index ActionResult

Jos GetForms-funktioon lähetetty bool-muuttuja getAll on epätosi, raportit haetaan sisäänkirjautuneen käyttäjän käyttäjätunnuksen perusteella. Jos muuttuja on tosi, haetaan kaikki raportit. (Kuva 10.)

```
public List<FormDatabaseModel> GetForms(List<FormDatabaseModel> forms, bool getAll)
{
    string query = "";

    if (getAll == false)
    {
        query = "SELECT * FROM dbo.Test WHERE NAME = @Name ORDER BY Date DESC;";

    }
    else
    {
        query = "SELECT * FROM dbo.Test ORDER BY Date DESC;";
    }

    using (SqlConnection conn = new SqlConnection(ConfigurationManager.AppSettings["connStr"]))
    {
        using (SqlCommand command = new SqlCommand(query, conn))
        {
            conn.Open();
            if (getAll == false)
            {
                command.Parameters.AddWithValue("@Name", User.Identity.Name);
            }
            using (SqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    FormDatabaseModel f = new FormDatabaseModel();
                    f.id = reader[0].ToString();
                    f.name = reader[1].ToString();
                    f.date = DateTime.Parse(reader[2].ToString());

                    f.location = reader[3].ToString();
                    f.target = reader[4].ToString();
                    f.templateid = int.Parse(reader[5].ToString());
                    f.data = reader[6].ToString();
                    f.mname = reader[7].ToString();
                    if (!String.IsNullOrEmpty(reader[8].ToString()))
                    {
                        f.mdate = DateTime.Parse(reader[8].ToString());
                    }
                    forms.Add(f);
                }
            }
        }
    }

    return forms;
}
```

KUVA 10. GetForms-funktio

5.3 Uusi raportti

Uuden raporttilomakkeen lähetyksen näkymään (kuva 11) mennään HomeControllerissa olevan Form-actionin kautta. Action tarkastaa, onko parametrinä olevaa uniikkitunnistetta olemassa. Jos ei, niin action palauttaa Form-näkymän, jossa uuden raportin lähetys tapahtuu. Näkymässä lomake näytetään käyttämällä ASP.NET MVC:n HTML helperiä.

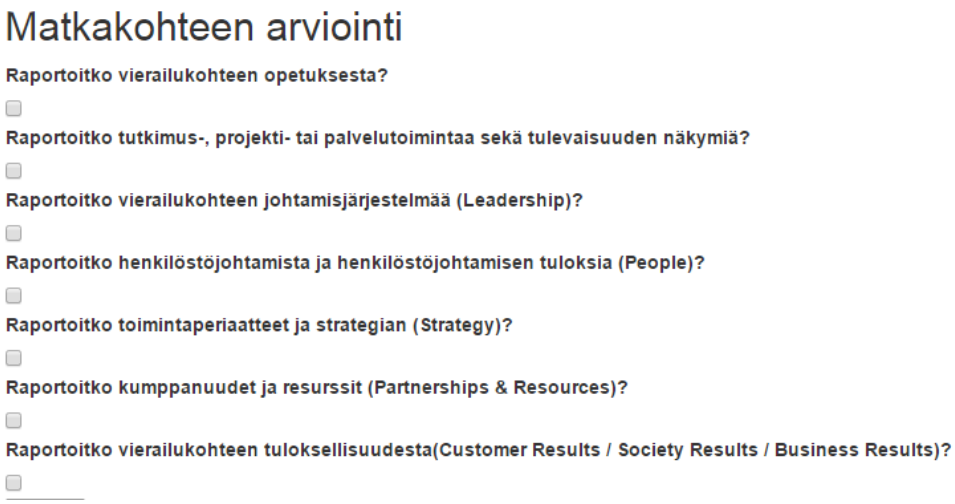
KUVA 11. Raporttilomake

Lomakkeen tietojen esitäyttöön käytetään Typeahead-Javascriptlaajennusta. Syöttämällä vähintään kolme kirjainta kenttiin, joihin Typeahead on liitetty, haetaan CRM:stä samankaltaisia tuloksia. Hakutulokset näytetään tekstikentän alapuolella olevassa listassa. Kun listasta valitaan jotain, täytetään loput siihen liittyvät kentät automaattisesti. Esimerkiksi vierailukohteen nimen perusteella haetaan kohteen tyyppi ja osoitetiedot. Vierailukohteen yhteyshenkilön tiedot haetaan sähköpostiosoitteen perusteella.

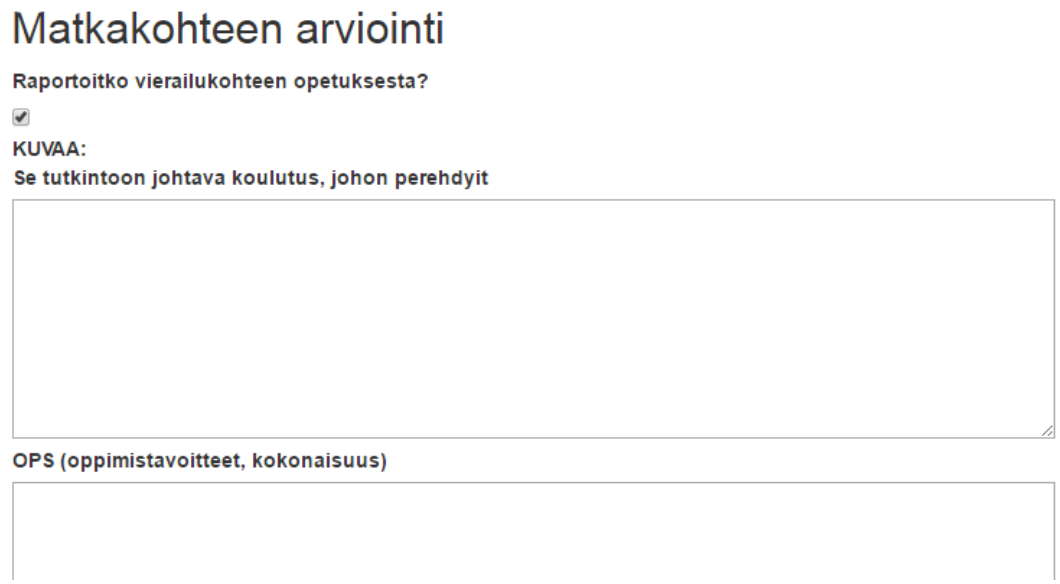
Päivämäärien syöttöön käytetään Bootstrap-skriptikirjaston Datepicker-laajennusta. Päivämääräkenttää klikkaamalla ilmestyy kalenteri, josta päivämäärä valitaan (kuva 12). Datepickerin kalenteri on saatu suomenkieliseksi erillisellä lokalisaatiotiedostolla.

KUVA 12. Päivämäärän valinta

EFQM-arviointiin liittyvät kentät ovat oletuksena piilotettu (kuva 13). Kentät näytetään, kun rastitetaan checkbox siitä kohdasta, josta halutaan raportoida (kuva 14). Checkboxeihin on liitetty jQueryn toggle-funktio, jolla halutut kentät saadaan näytettyä ja piilotettua.



KUVA 13. EFQM-arvioinnin kentät piilotettuna



KUVA 14. EFQM-arvioinnin kentät näytettynä

Liitetiedostojen lisäyksessä käytetään HTML:n file-tyyppistä input-kenttää. Multiple-attribuuttia käyttämällä saadaan lisättyä useampia liitetiedostoja. Liitteet lähetetään palvelimelle raportin uniikkitunnisteen mukaan nimettyyn hakemistoon.

Jos pakollisia kenttiä ei ole täytetty tai lomakkeen täytössä on muita virheitä (esim. päivämäärä ei ole validi) lomaketta ei lähetetä eteenpäin vaan pysytään lomakesivulla ja käyttäjälle näytetään lista siitä mitä meni väärin. Lista virheistä saadaan näkyviin Html.ValidationSummary()-funktion avulla. Pakollisten kenttien kohdalla on mallin puolella käytetty Required-attribuuttia. Päivämäärät validoituvat Typeof(Date.Time):llä. "Lupa kysytty tietojen käyttöön Savonian kehittämistyössä"-

checkboxin validoinnissa käytetään bool-tyyppistä Range-attribuuttia, jonka ainoa hyväksytty arvo on tosi.

Kun ”Lähetä”-painiketta klikataan, lomakkeeseen täytetyt tiedot lähetetään HTTP-protokollan POST-metodilla FormControllerissa olevaan PostForm-actioniin (kuva 15). Action tarkastaa ensin, onko lähetetty lomake validi. Jos lomake on validi, lomakkeen tiedot serialisoidaan XML-muotoon. Lisäksi raportille generoidaan oma uniikkitunniste. Lomakkeen tiedot, uniikkitunniste, lähettäjä, matkakohde, kohdemaa ja lomakkeen rakenteen tunniste lisätään CRM:ään tai testikantaan omaan tauluunsa. Matkakohteen tiedot, kohteen yhteyshenkilö ja lähettäjän tiedot lisätään myös omiin tauluihinsa, jos niitä ei siellä ole. Jos tietojen lähetys onnistui, käyttäjä ohjataan sivulle, jolla onnistumisesta ilmoitetaan käyttäjälle.

```
[HttpPost]
0 references
public ActionResult PostForm(FormViewModel model, HttpPostedFileBase file)
{
    if (ModelState.IsValid)
    {
        XmlSerializer xs = new XmlSerializer(typeof(FormModel));
        StringWriter sw = new CustomStringWriter();
        string name = HttpContext.User.Identity.Name;
        string location = model.Form.Organization.Address.Country;
        string target = model.Form.Organization.Name;
        int tempid = 1;
        xs.Serialize(sw, model.Form);

        SqlXml xml = new SqlXml(new XmlTextReader(new StringReader(sw.ToString())));

        Guid guid = Guid.NewGuid();

        //Insert to database
        using (SqlConnection conn = new SqlConnection(ConfigurationManager.AppSettings["connStr"]))
        {
            using (SqlCommand command = new SqlCommand())
            {
                conn.Open();
                command.Connection = conn;
                command.CommandText = "INSERT INTO dbo.Test (Id, Name, Date, Location, Target, TemplateId, Data) VALUES
                command.Parameters.Add("@Id", SqlDbType.UniqueIdentifier).Value = guid;
                command.Parameters.AddWithValue("@Name", name);
                command.Parameters.AddWithValue("@Location", location);
                command.Parameters.AddWithValue("@Target", target);
                command.Parameters.AddWithValue("@TemplateId", tempid);
                command.Parameters.AddWithValue("@Data", xml);
                command.ExecuteNonQuery();
```

KUVA 15. PostForm-action

5.4 Raportin muokkaus

Raportin muokkaukseen pääsee etusivun raporttilistassa olevasta linkistä. Muokkausnäkymään mennään uuden raportin tavoin HomeControllerin Form-actionin kautta. Tässä tapauksessa kuitenkin lähetetään actioniin parametrinä muokattavan lomakkeen uniikkitunniste. Tunnisteen perusteella haetaan muokattava raportti tietojärjestelmästä GetFormModel-funktion (kuva 16) avulla, missä se de-serialisoidaan XML-muodosta modeliin. Model lähetetään eteenpäin näkymään, jossa siihen sidotut kentät ovat automaattisesti täytettynä. Muokkausnäkymä on ulkoisesti samanlainen kuin uuden raportin täyttönäkymä. Taustalla ero on uuden raportin lähetykseen verrattuna lomakkeen näyttävään HTML helperiin liitetyssä actionissa, johon lomake lähetetään. Toinen ero on EFQM-arvioinnit sisältävissä diveissä, joissa on if-lauseet. Lauseissa asetetaan divien näkyvyys sen perusteella, onko checkboxeihin liitettyjen bool-muuttujien arvona tosi vai epätosi.

```
public FormModel GetFormModel(FormModel form, string guid)
{
    string query = "SELECT Data FROM dbo.Test WHERE Id = @Id;";
    string xmlStr = "";

    if (!String.IsNullOrEmpty(guid))
    {
        using (SqlConnection conn = new SqlConnection(ConfigurationManager.AppSettings["connStr"]))
        {
            using (SqlCommand command = new SqlCommand(query, conn))
            {
                conn.Open();
                command.Parameters.Add("@Id", SqlDbType.UniqueIdentifier).Value = new Guid(guid);
                using (SqlDataReader reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        xmlStr = reader[0].ToString();
                    }
                }
            }
        }

        XmlSerializer xs = new XmlSerializer(typeof(FormModel));
        object result;

        using (TextReader reader = new StringReader(xmlStr))
        {
            result = xs.Deserialize(reader);
            form = (FormModel)result;
        }
    }

    return form;
}
```

KUVA 16. GetFormModel-funktio

Uusia liitetiedostoja saa listättyä raportin lähetyksen tavoin file-tyyppistä input-kenttää käyttämällä. Vanhat liitteet saa poistettua yksitellen klikkamalla liitteen vieressä olevaa roskakorikuvaketta. Tiedoston poistoon käytettävää DeleteFile-funktiota kutsutaan käyttämällä Ajax Callia. Ajax Callin käyttäminen välttää sen, että sivu päivittyy tai käyttäjä uudelleenohjautuu toiselle sivulle ja menettää tallentamattomat tiedot.

Kun muokkaukset tallennetaan, lomakkeen tiedot lähetetään FormControllerissa olevaan EditForm-actioniin. (Kuva 17.) Erona uuden lomakkeen lähetyksessä käytettävään PostForm-actioniin on lähinnä se, että se ottaa parametrina vastaan muokatun lomakkeen uniikkitunnisteen. Tunnistetta käytetään ehtona muokattujen tietojen päivityksessä.

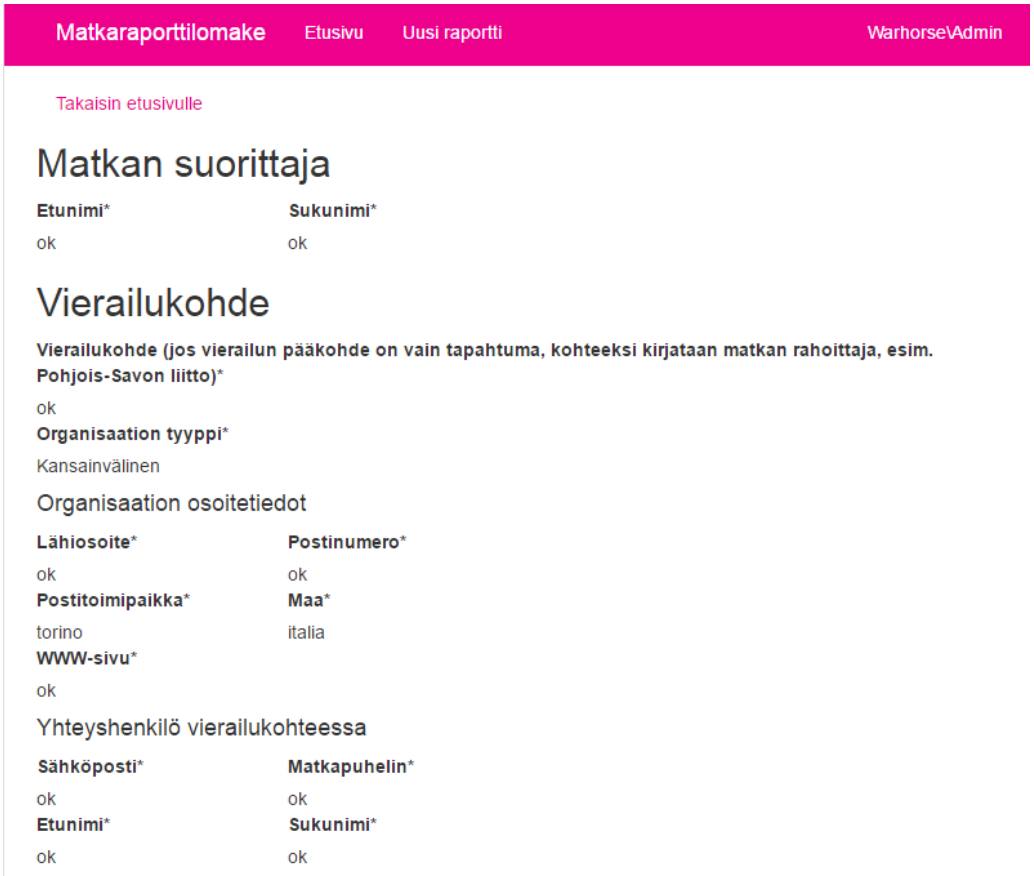
```
[HttpPost]
0 references
public ActionResult EditForm(string guid, FormViewModel model, IEnumerable<HttpPostedFileBase> files)
```

```
conn.Open();
command.Connection = conn;
command.CommandText = "UPDATE dbo.Test SET Location=@Location,Target=@Target,Data=@Data,MName=@MName,MDate=GETDATE() WHERE Id = @Id;";
command.Parameters.AddWithValue("@Location", location);
command.Parameters.AddWithValue("@Target", target);
command.Parameters.AddWithValue("@Data", xml);
command.Parameters.AddWithValue("@MName", name);
command.Parameters.Add("@Id", SqlDbType.UniqueIdentifier).Value = new Guid(guid);
command.ExecuteNonQuery();
```

KUVA 17. EditFormin erot PostFormiin verrattuna

5.5 Raportin näyttö

Raportin näyttö (kuva 18) on samankaltainen raportin muokkauksen kanssa. Tekstikenttien sijasta raportin tiedot näytetään normaalina tekstinä.



KUVA 18. Raportin näyttö

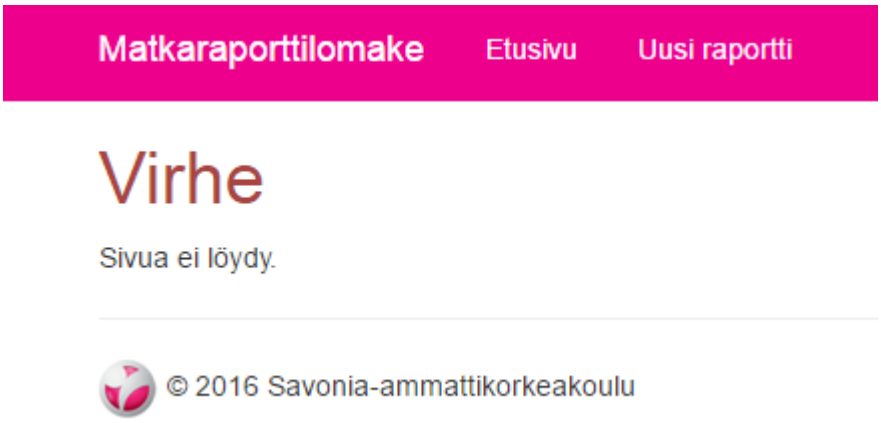
Raportin näytön action (Kuva 19) on hyvin yksinkertainen. Näytettävä raportti haetaan raportin muokkauksen tavoin GetFormModel-funktiolla FormModel-malliin raportin uniikkitunnisteen (GUID) perusteella. Funktiossa raportti de-serialisoidaan XML-muodosta malliin.

```
0 references
public ActionResult ViewForm(string guid)
{
    FormModel form = new FormModel();
    form = GetFormModel(form, guid);
    return View(form);
}
```

KUVA 19. ViewForm-action

5.6 Virhetilanteiden näyttö

Luvussa 6.2 mainittujen validaattoreiden ja ValidationSummaryn lisäksi työssä käytetään muitakin tapoja tarkistaa ja näyttää virhetilanteet. Virhetilanteessa, missä sivua ei löydy, käyttäjä ohjataan NotFound-näkymään. (Kuva 20.) Tämä saadaan toimimaan laittamalla CustomErrors päälle sovelluksen Web.configissa ja asettamalla 404-virhekoodille uudelleenohjaus. (Kuva 21).

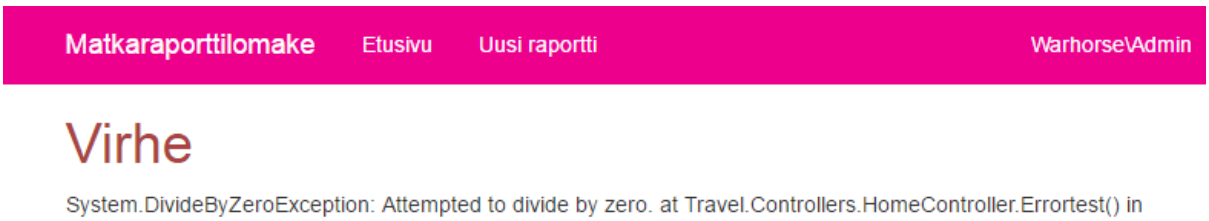


KUVA 20. ”Sivua ei löydy.”-virhesivu

```
<customErrors mode="On">
  <error statusCode="404" redirect="~/Home/NotFound"/>
</customErrors>
```

KUVA 21. customErrors ja 404-virheen asetukset Web.configissa

Controllerissa tapahtuvat virheet saadaan havaittua ja näytettyä (kuva 22) käyttämällä globaalia virhesivua ja molemmissa sovelluksen controllerissa olevaa OnException-metodia (kuva 23) käyttämällä. Virheen tarkat tiedot näytetään virhenäkymässä olevassa Model.Exception-muuttujassa. OnExceptionin käyttö ei vaadi customErrorsin päälle laittamista.



KUVA 22. Virhesivu

```
1 reference
protected override void OnException(ExceptionContext filterContext)
{
    if (filterContext.Exception is InvalidOperationException)
    {
        var controllerName = (string)filterContext.RouteData.Values["controller"];
        var actionName = (string)filterContext.RouteData.Values["action"];
        var model = new HandleErrorInfo(filterContext.Exception, controllerName, actionName);

        RedirectToAction("~/Views/Shared/Error.cshtml", model);
    }
}
```

KUVA 23. OnException-metodi

Virheiden tarkastukseen actionkohtaisella tasolla voidaan käyttää myös actioniin lisättävää HandleError-attribuuttia. Tätä ei kuitenkaan käytetä tässä työssä.

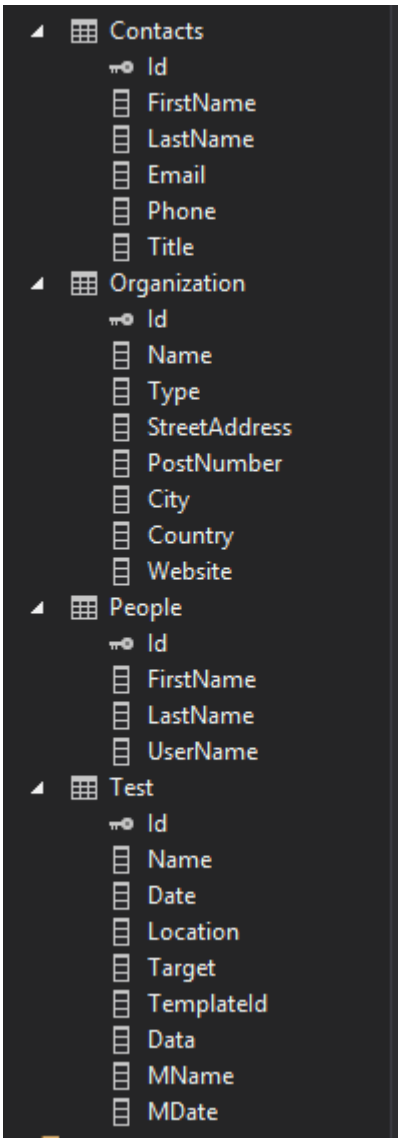
5.7 CRM-integraatio

Matkaraporttilomakesovelluksen tarkoitus on toimia osana tämän opinnäytetyön toisen ohjaajan Mikko Pääkkösen kehittämää lomakesovellusta. Sovellus on jaettu eri alueisiin (area), joista yksi on matkaraportit. Integraatio Savonian lomakesovelluksen ja CRM-järjestelmän välillä tapahtuu Entity Frameworkin avulla. Entity Framework on ORM (object-relational mapping) -ohjelmistokehys, joka mahdollistaa tiedonsiirron sovelluksen ja suhteellisen tietojärjestelmän välillä ilman, että ohjelmoijan tarvitsee kirjoittaa useita rivejä koodia. (Microsoft 2015b.) Lähetettyjen raporttien tulostus PDF-muodossa tapahtuu CRM:n kautta.

6 TESTAUS

Sovellusta tehtäessä CRM-järjestelmän korvikkeena käytettiin testitietokantaa. Testikantana toimi Microsoft SQL Server -tietokantatiedosto. Testitietokannassa on neljä taulua (kuva 24):

- Test, joka sisältää lähetetyt raportit.
- Organization, joka sisältää matkakohteet ja niiden tiedot.
- Contacts, joka sisältää matkakohteiden yhteyshenkilöt ja heidän yhteystietonsa.
- People, joka sisältää raporttien lähettäjien nimet ja käyttäjätunnukset



KUVA 24. Testikannan taulut.

Raporttien lähetys testitietokantaan XML-muodossa ei sujunut virheittä. Ääkkösiä sisältävät raportit eivät tallentuneet tietokantaan. Tämä korjautui käyttämällä tavallisen StringWriterin sijasta omaa CustomStringWriter-luokkaa (kuva 25), jolla XML-muodossa olevat raportit saadaan tallennettua ISO-8859-1-enkoodauksella oletuksena käytettävän UTF-8-enkoodauksen sijasta. Raportteja lähetettäessä CRM-järjestelmään enkoodausta ei tarvitse vaihtaa.

```
2 references
public class CustomStringWriter : StringWriter
{
    0 references
    public override Encoding Encoding => Encoding.GetEncoding("ISO-8859-1");
}
```

KUVA 25. CustomStringWriter-luokka

7 JATKOKEHITYS

Ominaisuus, joka työstä jäi pois, oli mahdollisuus käyttää sovellusta offline-tilassa. Yksi mahdollinen vaihtoehto sovelluksen offline-käytölle on se, että käyttäjä täyttää valmiin Word- tai Excel-lomakepohjan ja lähettää sen sovellukseen. Kun internetyhteys on käytettävissä, dokumentti lähetetään eteenpäin ja serialisoidaan XML-muotoon. Tämän menetelmän huono puoli on sen alttius käyttäjän virheille. Käyttäjä saattaa rikkoa lomakkeen rakenteen, jolloin XML-serialisointi ei onnistu.

Parempi ratkaisu olisi käyttää HTML5 Application Cachea ja Web Storagea. Application Cachen käytössä on muitakin etuja kuin offlinekäytön mahdollisuus. Cachetettujen resurssien lataus on nopeampaa ja palvelinta rasitetaan vähemmän. Application Cache toimii kolmeen osaan jaettua Manifest-tiedostoa käyttämällä. Ensimmäisessä osassa, Cache Manifestissä määritellään mitä Application Cachessa halutaan säilyttää. Toisessa osassa Networkissa taas määritellään mitä ei haluta säilyttää. Network on käytettävissä pelkästään verkkoyhteyden kanssa. Kolmas osa on Fallback, jossa määritetään mitä näytetään jos internetyhteys ei ole käytössä.

Web Storage säilyttää dataa selaimessa evästeen tavoin, mutta sen kapasiteetti on huomattavasti suurempi ja se on myös evästeitä turvallisempi. Evästeeseen mahtuu 4 kilobittiä dataa, kun taas Web Storageen mahtuu selaimen mukaan 5-25 megabittiä. Web Storageen kuuluu kaksi erilaista ratkaisua, Local Storage ja Session Storage. Tässä työssä sopivampi on Local Storage, joka ei vanhene. Session Storage on ikkunakohtainen, ja vanhenee kun ikkuna suljetaan. Application Cachea ja Web Storagea tukevat selaimet ovat Internet Explorer, Mozilla Firefox ja siihen perustuvat selaimet, Google Chrome, Safari, Opera ja Edge.

8 YHTEENVETO

Opinnäytetyön tilaajana oli Savonia-ammattikorkeakoulu. Työn tavoitteena oli vertailla eri tapoja toteuttaa sähköinen matkaraporttilomakesovellus ja toteuttaa se valitulla tavalla. Toteutuksessa täytyi ottaa huomioon jatkokehitysmahdollisuudet, integraatio CRM-järjestelmään ja mobiiliystävällisyys. Vertaillut toteutustavat olivat InfoPath, Nintex Forms, ASP.NET Web Forms ja ASP.NET MVC. Toteutustavaksi valittiin ASP.NET MVC sen ominaisuuksien takia, ja koska MVC-sovelluksen pystyi yhdistämään Savonialla käytössä olevaan lomakesovellukseen.

Haasteena työssä oli aikataulussa pysyminen. Liian usein juutuin johonkin ongelmaan ja yritin itsepäisesti ratkoa sitä sen sijaan, että olisin siirtynyt seuraavaan asiaan ja yrittänyt myöhemmin uudelleen. Myös useampaan asiaan keskittyminen samanaikaisesti tuotti vaikeuksia, mikä vaikutti myös aikataulussa pysymiseen. Lisäksi raporttilomakkeeseen tulevan kysymyspatterin saaminen viivästyi ja jouduin käyttämään sitä ennen itse tekemää testilomaketta. Työn loppupäässä aikataulusta lipsuminen lähti niin pahasti käsistä, ettei mitään aikataulua enää noudatettu.

Opettavaisin asia tässä työssä oli eri JavaScript-kirjastojen ja –liitännäisten hyödyntäminen. Aikaisemmin olin käyttänyt niitä melko vähän. Opin myös kantapään kautta, ettei kannata jäädä miettimään yhtä ongelmaa liian pitkäksi aikaa.

Lopputuloksena saatiin aikaiseksi sovellus, jolla on mahdollisuus lisätä uusia raportteja liitetiedostoineen, sekä muokata ja katsella lähetettyjä raportteja. Alunperin suunnitelluista ominaisuuksista jäi puuttumaan mahdollisuus käyttää sovellusta offlinetilassa, joten se jäi jatkokehitykseen. Myös raporttien tulostus PDF-muotoon jäi pois. Sovelluksen Visual Studio –projekti jätettiin yhdistettäväksi Savonian lomakesovelluksen kanssa.

LÄHTEET

COLTER, Carlton 2011-11-21. InfoPath to CRM: Accepting Emailed InfoPath Forms. [verkkoaineisto]. [viitattu 2016-05-25]. Saatavissa: https://mscrmblogger.com/2011/11/21/infopath-to-crm-accepting-emailed-infopath-forms/

GEORGE, Suzanne 2014-04-27. SharePoint 2013 – Is Access 2013 the New InfoPath? [verkkoaineisto]. [viitattu 2016-05-25]. Saatavissa: http://blogs.perficient.com/microsoft/2014/04/sharepoint-2013-is-access-2013-the-new-infopath/

HAMBRICK, PJ 2013-12-5. .NET Web Forms vs. MVC: Which Is Better?. [verkkoaineisto]. [viitattu 2016-1-13]. Saatavissa: http://www.seguetech.com/blog/2013/12/05/dotnet-web-forms-vs-mvc-which-better

KOIRALA, Shivprasad 2014-9-27. Webforms vs MVC and Why MVC Is Better? [verkkoaineisto]. [viitattu 2016-1-13]. Saatavissa: http://www.codeproject.com/Articles/821275/Webforms-vs-MVC-and-Why-MVC-is-better

MEIER, JD 2008-9-5. Layers and Tiers. [verkkoaineisto]. [viitattu 2016-4-11]. Saatavissa: https://blogs.msdn.microsoft.com/jmeier/2008/09/05/layers-and-tiers/

MICROSOFT 2013. Introduction to InfoPath Forms Services. [verkkoaineisto]. [viitattu 2016-1-3]. Saatavissa: https://support.office.com/en-us/article/Introduction-to-InfoPath-Forms-Services-e599b4f0-1d1d-4249-8251-5d28afb648d8?CorrelationId=a2cd296c-c09d-4592-8718-7c2fa4074aa0&ui=en-US&rs=en-US&ad=US

MICROSOFT 2013-1-10. Customize the Duet Workflow Task form in InfoPath 2013. [verkkoaineisto]. [viitattu 2016-1-3]. Saatavissa: https://blogs.msdn.microsoft.com/sharepointdev/2013/01/10/customize-the-duet-workflow-task-form-in-infopath-2013/

MICROSOFT 2014. ASP.NET MVC Overview. [verkkoaineisto]. [viitattu 2016-4-6] Saatavissa: https://msdn.microsoft.com/en-us/library/dd381412%28v=vs.108%29.aspx

MICROSOFT 2014-1-31. Update on InfoPath and SharePoint Forms. [verkkoaineisto]. [viitattu 2016-5-25]. Saatavissa: https://blogs.office.com/2014/01/31/update-on-infopath-and-sharepoint-forms/

MICROSOFT 2015a. ASP.NET web forms and data binding. [verkkoaineisto]. [viitattu 2016-1-13]. Saatavissa: https://msdn.microsoft.com/en-us/library/gg695802%28v=crm.7%29.aspx

MICROSOFT 2015b. Entity Framework. [verkkoaineisto]. [viitattu 2016-5-7]. Saatavissa: https://msdn.microsoft.com/en-us/data/ef.aspx

MICROSOFT 2016a. Learn About ASP.NET WebForms. [verkkoaineisto]. [viitattu 2016-1-13]. Saatavissa: http://www.asp.net/web-forms

MICROSOFT 2016b. Learn About ASP.NET MVC. [verkkoaineisto]. [viitattu 2016-1-13]. Saatavissa: http://www.asp.net/mvc

MILEY, Jamie 2011-7-7. HOW INTEGRATE INFOPATH IN CRM? [foorumikeskustelu]. [viitattu 2016-1-3]. Saatavissa: https://social.microsoft.com/Forums/en-US/5bfff309-7cab-4046-af32-969ff53250b8/how-integrate-infopath-in-crm-2011

NINTEX GLOBAL LTD 2016. Exploring InfoPath Alternatives. [verkkoaineisto]. [viitattu 2016-1-3]. Saatavissa: http://www.nintex.com/workflow-platform/nintex-forms/forms-for-sharepoint/exploring-infopath-alternatives

NINTEX GLOBAL LTD 2016. SharePoint Forms. [verkkoaineisto]. [viitattu 2016-1-3]. Saatavissa: http://www.nintex.com/workflow-platform/nintex-forms/forms-for-sharepoint

SUKESH, Marla 2014-9-26. WebForms vs. MVC. [verkkoaineisto]. [viitattu 2016-1-13]. Saatavissa: http://www.codeproject.com/Articles/528117/WebForms-vs-MVC

LIITE 1: RAPORTTILOMAKKEEN MÄÄRITTELY

Matkan suorittaja	Etunimi	Sukunimi (matkan tekijän tiedot tärkeää viedä myös CRM:n puolelle)			
Vierailukohde Jos kohteen tiedot on CRM:ssä, ne saadaan lomakkeelle näkyviin (kuten opinnäytetyölomakkella)					
Vierailukohde (jos vierailun pääkohde on vain tapahtuma, kohteeksi kirjataan matkan rahoittaja, esim. Pohjois-Savon liitto) *	*Organisaation virallinen nimi (ETSII CRM:stä -> jos ei ole CRM:ssä, kirjataan oheiset perustiedot)*				
Organisaation tyyppi *	*VALIKKO. Ainoat vaihtoehdot: kansainvälinen tai rahoittaja. CRM/Savonian kumppanikategoria.*				
Organisaation osoitetiedot *	*Lähiosoite* *	*Postinro* *	*Postitoimipaikka* *	*Maa* *	*www-sivu**
Yhteyshenkilö vierailukohteessa	*Sähköposti (hakee CRM:stä)* *	*Etunimi* *	*Sukunimi* *	*Matkapuhelin* *	*Titteli*
Lisätietoja vierailukohteesta					
Muut tavatut henkilöt	*Kirjataan, tekstimuodossa talteen*				
Liitteet	*Yksi tai useampi tiedosto mahdollista liittää.*				
Lupa kysytty tietojen käyttöön Savonian kehittämistyössä *	*Kyllä*	*Ei*			

Tiedot matkasta	
Matkasuunnitelmassa käytetty perustelu matkalle (esim. esitys seminaarissa x) *	*Kirjataan, tekstimuodossa Käytetään matkaraportin nimenä*
Matkan ajankohta *	*Aloitus pvm* / *Lopetus pvm*
Matkan keskeinen anti *	*Kirjataan, tekstimuodossa*
Vapaa kuvaus vierailusta	*Kirjataan, tekstimuodossa*
Mitä yhteistyömahdollisuuksia matkasta kehittyi (esim. opiskelija- tai henkilöstövaihto, opetusjärjestelyt, opiskelu)?	*Kirjataan, tekstimuodossa*
Kohdeorganisaation edustajien näkemyksiä	*Kirjataan,* tekstimuodossa. Substanssin ja osaamisen kehittyminen. Mihin suuntaa esim opetus ja tekniikka kehittyvät? Tulevat suunnitelmat, trendit (alueella, maassa, globaalisti).

Matkakohteen arviointi (EFQM, vertailuihin)	*Kyllä*	*Ei*
Raportoitko vierailukohteen opetuksesta? *	*Kirjataan kuvauksia TAI liitetiedosto.* *KUVAA:* *- Se tutkintoon johtava koulutus, johon perehdyit* *- OPS (oppimistavoitteet, kokonaisuus)* *- Opintojen rakenne (opintopisteet, aikataulutus). Ennen kaikkea hyvät käytänteet.* *- Opintojaksokuvaukset* *- Arviointi (hyvät käytänteet)* *- Toimivat pedagogiset ratkaisut (hyvät käytänteet)* *- Oppimisympäristöt käytännössä ja mitä meillä opittavaa* *- E-learning, miten käytetään ja mitä meillä opittavaa* *- Mahdollisuudet yhteistyön kehittämiseen opetuksessa (OPS, Double Degree)* *- Mahdollisuudet yhteistyön kehittämiseen opiskelija- ja henkilöstövaihdossa (opetus, opiskelu,...)* *- Tärkeimmät havainnot opetustoiminnasta* *- Tärkeimmät havainnot muusta kuin opetustoiminnasta*	
Raportoitko tutkimus-, projekti- tai palvelutoimintaa sekä tulevaisuuden näkymiä? *	*Kirjataan kuvauksia TAI liitetiedosto.* *- Millasta tutkimustoimintaa kohdeorganisaatiossa on, miten laaja sen on ja miten se on kytketty opetukseen (jos on)?* *- Millasta kehittämistoimintaa kohdeorganisaatiossa on, miten laaja sen on ja miten se on kytketty opetukseen (jos on)?* *- Millasta maksullista palvelu/liiketoimintaa kohdeorganisaatiossa on, miten laaja sen on ja miten se on kytketty opetukseen (jos on)?* *- Mahdollisuudet yhteistyön kehittämiseen opiskelija- ja henkilöstövaihdossa (tutkimus, projektit,...)* *- Mahdollisuudet tutkimusyhteistyöhön (tutkimusintressi)* *- Mahdollisuudet projekti- ja kehittämisyhteistyöhön*	
Raportoitko vierailukohteen johtamisjärjestelmää (Leadership)? *	*Kirjataan kuvauksia TAI liitetiedosto.* *KUVAA JOHTAJUUS. Kuvaa tähän yhteyteen kuinka tarkastelemasi organisaation johtajat luovat organisaatiolle mission, vision ja arvot, sekä edistävät niiden toteutumista henkilökohtaisella toiminnallaan. Kerro esimerkiksi miten johtajat osallistuvat henkilökohtaisesti organisaation johtamisjärjestelmän kehittämiseen ja toteuttamiseen, pitävät yhteyttä asiakkaisiin, yhteistyökumppaneihin ja yhteiskunnan edustajiin sekä motivoivat ja tukevat henkilöstöä.* *KUVAA ORGANISAATIO.*	
Raportoitko henkilöstöjohtamista ja henkilöstöjohtamisen tuloksia (people) *	*Kirjataan kuvauksia TAI liitetiedosto.* *KUVAA HENKILÖSTÖJOHTAMINEN.* *KUVAA HENKILÖSTÖJOHTAMISEN TULOKSET.*	
Raportoitko toimintaperiaatteet ja strategian (Strategy)? *	*Kirjataan kuvauksia TAI liitetiedosto.* *Kuinka organisaation sidosryhmien nykyisiä ja tulevia tarpeita sekä odotuksia tutkitaan, hyödynnetään strategian ja toimintaprosessien suunnittelussa ja käytännön työssä?* *KUVAA STRATEGIA.* *KUVAA YLEISET TOIMINTAPERIAATTEET.* *KUVAA KESKEISET KEHITTÄMISKOHTEET.*	
Raportoitko kumppanuudet ja resurssit (Partnerships & Resources)? *	*Kirjataan kuvauksia TAI liitetiedosto.* *KUVAA TÄRKEIMMÄT KUMPPANUUDET* *KUVAA KÄYTETTÄVISSÄ OLEVAT RESURSSIT (henkilöstö, raha yms.)*	
Raportoitko vierailukohteen tuloksellisuudesta (Customer Results / Society Results /Business Results)?	*Kirjataan kuvauksia TAI liitetiedosto.* *KUVAA:* *- Tärkeimmät asiakastulokset (esim. opiskelijapalautteet)* *- Toiminnan tärkeimmät yhteiskunnalliset tulokset* *- Toiminnan tärkeimmät suorituskykytulokset*	