OULUN AMMATTIKORKEAKOULU

Aki Perätalo & Toni Tuohimaa

Avainrekisterin tekeminen tietokantariippumattomaksi

Avainrekisterin tekeminen tietokantariippumattomaksi

Aki Perätalo & Toni Tuohimaa
Opinnäytetyö
Syksy 2015
Tietojenkäsittely
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittely, web-sovelluskehitys

Tekijät: Aki Perätalo & Toni Tuohimaa
Opinnäytetyön nimi: Avainrekisterin tekeminen tietokantariippumattomaksi
Työn ohjaaja: Teppo Räisänen
Työn valmistumislukukausi- ja vuosi: Syksy 2015 Sivumäärä: 32

Tämän opinnäytetyön aihe saatiin toimeksiantona Oamkilta ja Osekkilta, joiden ylläpitämä avainrekisteri kaipasi päivitystä. Aihe on toimeksiantajien tulevaisuutta ajatellen tärkeä, sillä sekä Oamkilla että Osekkilla on edessään vaihto Oraclen palveluista Microsoftin palveluihin. Opinnäytetyöstä saadaan tärkeää kokemusta tuota vaihtoa varten tarvittavaan käännöstyöhön PL/SQL:stä esimerkiksi PHP:lle.

Opinnäytetyön tavoitteena on kääntää vanha sovellus PL/SQL:stä PHP:lle ja saada se toimimaan PHP:lla sekä vanhassa Oraclen että uudessa Microsoftin tietokannassa. Myös käyttöliittymä joudutaan tekemään käyttämällä uutta tekniikkaa, AngularJS:ää, mutta se ei ole opinnäytetyössä pääosassa.

Vanhan sovelluksen päivittäminen suoritettiin kääntämällä PL/SQL-koodi sen toiminnallisuuden perusteella omien kykyjen mukaan PHP:lle. Vanhaa tietokantaa siistittiin ja sen jälkeen siirrettiin uuteen tietokantaan itsetehdyillä PHP-siirtoskripteillä. Käyttöliittymä luotiin AngularJS:llä vanhan sovelluksen pohjia ja Oamkin standardeja mukaillen. Opinnäytetyön raportin tietoperustana ja aineistona käytettiin pääasiassa internetistä löytyvää materiaalia.

Opinnäytetyöllä saavutettiin uudistettu avainrekisteri, joka menee vahtimestarien käyttöön. Opinnäytetyöstä saatu kokemus on arvokasta toimeksiantajille, sillä opinnäytetyön aikana ilmenneistä ongelmista, esimerkiksi tietokannan siirtoon käytettävien sovellusten ongelmat, saadusta uudesta tiedosta ja kokemuksesta on apua tulevien siirtojen aikana. Myös itsetehdyistä tietokannan siirtoskripteistä on hyötyä tuohon prosessiin, sillä vastaavaa ei ole kummassakaan organisaatiossa aiemmin tehty.

Asiasanat: Tietokanta, Oracle, AngularJS, PHP, tietokantariippumaton

ABSTRACT

Oulu University of Applied Sciences
Business Information Systems, Option of web application development

---

Authors: Aki Perätalo & Toni Tuohimaa
Title of thesis: Making key registry database-independent
Supervisor: Teppo Räisänen
Term and year when the thesis was submitted: Autumn 2015 Number of pages: 32

This thesis was produced for Oamk and Osekk, who maintain a key registry that was in need of an update. The subject of the thesis is very important to the client, because they are switching from Oracle's services to Microsoft's services in the near future. This thesis provides important experience needed in the translation process required in the switch, from PL/SQL to PHP for example.

The goal of this thesis is to translate the old application from PL/SQL to PHP and to make it work both in the old Oracle database and the new Microsoft database. The user interface will also have to be made using a new technique, and AngularJS was chosen for that, although the user interface will not be a major part of the thesis.

The old application was updated by translating the PL/SQL code to PHP by copying the functionality of the PL/SQL procedures and translating them to PHP using available knowledge. The old database was tidied up and then migrated to the new database using self-made PHP scripts. The user interface was created with AngularJS using the old application's foundations and comforming to Oamk's standards. Material found from the internet was used for the thesis' report's knowledge base.

An updated version of the key registry was achieved with this thesis. The experience gained by making the thesis is very important for the clients, for example the experience gained from encountering problems like migrating the database will be important during the switch from Oracle to Microsoft. The self-made database migration scripts will also prove useful for the process since neither of the organisations have never made anything similar.

Keywords: Database, Oracle, AngularJS, PHP, database independent

SISÄLLYS

1 JOHDANTO

Teknologian ja osaamisen kehittyessä sovellukset ja tietokannat vanhentuvat, jolloin nousee tarve päivittää nämä vanhatkin sovellukset ja tietokannat uusien tekniikoiden vaatimalle tasolle. Vanhojen sovellusten ja tietokantojen päivittämisellä uusiin tekniikoihin voidaan saavuttaa useita asioita, kuten esimerkiksi tietoturvan pitämisen ajantasalla, sekä palvelimiin kohdistuvan kuorman pitämisen kurissa uusien tekniikoiden tuomien suorituskykyparannuksien ansiosta. Myös käyttäjien kannalta uusien tekniikoiden hyödyntäminen on hyödyllistä, sillä myös heidän käyttökokemuksensa paranee paremman suorituskyvyn ja tietoturvan myötä.

Tämän opinnäytetyön aiheena on avainrekisterin tekeminen tietokantariippumattomaksi, mikä käytännössä tarkoittaa sitä, että tehdään sovellus, joka toimii riippumatta siitä, mikä tietokanta on käytössä. Tässä opinnäytetyössä kehitettävä sovellus on jo olemassa oleva Osekkin (Oulun seudun koulutuskuntayhtymä) vahtimestarien käyttämä avainrekisterisovellus, jonka avulla vahtimestarit voivat tallentaa tietokantaan avaimia ja niiden tietoja, sekä luovuttaa niitä koulun henkilökunnalle ja opiskelijoille. Sovelluksen avulla vahtimestarit voivat myös selata luovutettuja avaimia, sekä nähdä mitä avaimia ei ole vielä palautettu.

Vanha sovellus on kirjoitettu Oraclen PL/SQL-kielellä joka on yhteensopiva vain Oraclen tietokantojen kanssa. Opinnäytetyön tarkoituksena on kääntää sovellus PHP:lle sekä saada se toimimaan Oraclen sekä Microsoftin tietokannoissa yhtä aikaa. Vanhat tiedot Oraclen tietokannasta siirretään myös uuteen Microsoftin tietokantaan. Lisäksi, koska vanhan sovelluksen käyttöliittymä on tehty myös PL/SQL:llä, joudutaan uuden sovelluksen käyttöliittymä tekemään kokonaan uusiksi Googlen AngularJS:ää sekä tavallista HTML:ää hyödyntäen vanhan sovelluksen ulkoasua ja Oamkin (Oulun ammattikorkeakoulu) standardeja mukaillen.

Tässä opinnäytetyössä keskitytään PL/SQL-koodin kääntämiseen PHP:lle ja Oraclen tietokannan siirtämiseen Microsoftin tietokantaan. Käyttöliittymän toteuttamiseen käytetään vanhan sovelluksen pohjia, eikä sen kehittäminen ole pääosassa, vaikka sen toteuttamiseen tuleekin uutena sovelluksen osana AngularJS.

Tämän opinnäytetyön tekijät suorittivat työharjoittelunsa Oamkin tietohallinnossa, ja siellä tehtyjä sovelluksen pohjia ja käytettyjä tekniikoita, kuten AngularJS, hyödynnettiin myös avainrekisterin

uudelleenkehityksessä. Myös uusi hakemistorakenne saatiin lainattua harjoittelun aikana tehdystä sovelluksesta. Opinnäytetyön tekemisen aikana syntynyt valmis avainrekisteri on varmuuskopioitu Oamkin git-palvelimelle, joka toimii myös versionhallintaa varten.

Opinnäytetyön viitekehyksenä käytetään internetistä löytyvää materiaalia sekä kirjastosta lainattuja kirjoja. Tietoperustassa käsitellään tämän opinnäytetyön kannalta oleellisimmat asiat, eli tietokannat, PL/SQL, PHP, MVC-malli ja AngularJS.

2 CASE OSEKK / AVAINREKISTERI

Osekk eli Oulun seudun koulutuskuntayhtymä on organisaatio, joka tuottaa erilaisia palveluita Hailuodon, Iin, Kempeleen, Limingan, Lumijoen, Muhoksen, Oulun ja Tyrnävän alueen kouluille. Tämän opinnäytetyön kannalta oleellisin tieto on se, että heidän IT-osastonsa on vastuussa vahtimestarien käyttämän avainrekisterisovelluksen kehityksestä ja ylläpidosta.

Toisena osapuolena opinnäytetyössä on Oamkin tietohallinto, joka hoitaa erilaisia Oamkin opiskelijoiden opiskeluihin liittyviä asioita ja ylläpitää sovelluksia. Heidän tehtäviänsä ovat esimerkiksi opiskelijoiden käyttämien sovellusten Asion, Oivan ja Moodlen ylläpito, sekä avustaminen opiskelijoiden ja opettajien ongelmatilanteissa. Tietohallinto on osana opinnäytetyötä, sillä he ovat tiiviissä yhteistyössä Osekkin kanssa.

Opinnäytetyön aihetta ehdotti Oamkin tietohallinnon tietohallintopäällikkö Samuli Malinen, joka oli yhdessä Ville Valtosen ja Osekkin IT-henkilöiden Jarno Temosen ja Sampo Juntusen kanssa keskustelleet avainrekisterin kehitystarpeista. Opinnäytetyöstä on myös hyötyä jatkon kannalta, sillä sekä Oamk että Osekk ovat tulevaisuudessa irtautumassa Oraclen palveluista, jolloin Oraclen tietokannoissa olevat ja PL/SQL-kielellä toteutettujen muiden sovellusten siirtäminen esimerkiksi Microsoftin tietokantoihin ja PHP-kielelle tulee olemaan heille myös edessä tulevaisuudessa, joten tämä opinnäytetyö antaa pohjaa myös tuota urakkaa varten. Siirron on tarkoitus tapahtua alustavien keskusteluiden perusteella vuonna 2017.

2.1 Tilanteen kuvaus

Avainrekisteri luotiin 2000-luvun alussa nykyiseen muotoonsa, ja sitä on sittemmin muokattu ja paikkailtu aina tarpeen mukaan, toisinaan juurikaan ajattelematta suuremmin toiminnallisuutta tai tietokantarakenteen selvyyttä. Tietokannassa tai sen käyttöön luodussa sivustossa ei kummassakaan ole minkäänlaisia tietojen tarkistuksia, ja se on johtanut erittäin vapaamuotoiseen tietojen täyttämiseen ja täyttämättä jättämiseen, joka taas vaikeuttaa tietojen hakemista ja tekee lisättyjen tietojen lajittelun erittäin vaikeaksi.

Avainrekisteriä käyttävät lähinnä talonmiehet hallitessaan kulkijoiden avaimia, joten vanhassa kannassa on paljon sellaista tietoa johon heidän ei tarvitsisi päästä käsiksi ollenkaan, tai ainakaan voida muokata. Uudelle kannalle siirryttäessä voidaan helposti rajoittaa saatavilla olevia tietoja ja selkeyttää lisättäviä tietoja rajoittamalla ne asioihin joita oikeasti tarvitaan ja tarkastamalla ne.

Microsoftin palveluihin siirron johdosta kaikki vanhat järjestelmät tulee kääntää Microsoftin alustalle. Avainrekisterin siirrolla Microsoftin alustalle saadaan hankittua tietoa mahdollisista ongelmakohdista siirrossa sekä saadaan valmista koodia avuksi muita siirtoja varten.

2.2 Kuvaus avainrekisteristä

Osekkin avainrekisteri sisältää noin 6000 avaimen tiedot, joita on mahdollista luovuttaa yli 50000 henkilölle, jotka ovat joko Oamkin henkilökuntaa tai opiskelijoita. Avainrekisterin tiedot ovat Oraclen tietokannassa ja opinnäytetyöllä järjestelmästä halutaan tehdä tietokantariippumaton, eli sellainen, että järjestelmä toimisi riippumatta siitä, mitä tietokantaa käytetään. Käytännössä tämä toimii niin, että tietokantalausekkeet olisivat tietokannasta riippumatta samat, mutta tietokannan muuttuessa hakulausekkeita käsittelevät PHP-luokat vaihtuvat käytettävän tietokannan mukaisiksi.

Avainrekisterin käyttöliittymä ja tietokantahaut on vanhassa sovelluksessa toteutettu PL/SQL-kielellä, ja opinnäytetyössä erotetaan käyttöliittymä ja tietokannan käsittely, jolloin käyttöliittymä toteutetaan AngularJS:llä ja tietokannan käsittely kokonaisuudessaan PHP:lla.

3 TIETOPERUSTA

Tässä luvussa käsitellään opinnäytetyölle oleelliset tekniikat, eli tietokannat, PL/SQL, PHP, MVC-malli sekä AngularJS. Lisäksi mainitaan sovelluksen kehitykselle oleelliset työkalut.

3.1 Tietokannat

Tietokanta on kokoelma tietoja, joka on järjestelty niin, että se on helposti saatavilla, käsiteltävissä ja päivitettävissä. Yleisimmät tietokantamallit ovat hierarkkinen-, relaatio- ja olio-opohjaiset mallit. (TechTarget 2006, viitattu 26.10.2015.)

Tässä opinnäytetyössä käsiteltävät Oraclen ja Microsoftin tietokannat ovat molemmat malliltaan relaatiotietokantoja. Muita relaatiomallisia tietokantajärjestelmiä ovat esimerkiksi MySQL, PostgreSQL sekä SQLite. Taulukkolaskentaohjelma Excel puolestaan on esimerkki oliotietokantajärjestelmästä. IBM Information Management System puolestaan on nykyisin laajimmin käytetty kaupallinen hierarkkinen tietokanta. (Ibm, viitattu 23.11.2015.)

Relaatiotietokantamallissa tieto esitetään taulukoissa, joissa on rivejä ja sarakkeita. Taulujen välille luodaan yhteyksiä luomalla riveistä ja sarakkeista avaimia, joihin toisissa tauluissa viitataan. (Oracle b, viitattu 26.10.2015.)

Tietokantojen siirtämiseen käytetään yleensä juuri sitä tehtävää varten kehitettyjä sovelluksia, kuten esimerkiksi Oraclen SQL Developer tai Microsoftin SQL Server Migration Assistant (SSMA). Sovelluksia käyttämällä tiedon siirto automatisoidaan, jolloin vähennetään tarvetta ihmisen väliintulolle sekä minimoidaan järjestelmien alhaallaoloajat (Janalta Interactive Inc., viitattu 03.11.2015.).

3.1.1 Oracle

Oraclen ensimmäinen versio julkaistiin 1978, ja se oli aikanaan ensimmäinen kaupallinen relaatiotietokanta, se nousi nopeasti suosituksi yritysmaailmassa, ja on yhä nykyään suurin tietokantapalvelujen tarjoaja yritysmaailmassa. (Oracle 2007, viitattu 17.09.2015.)

Vuonna 1980 Oracle teki historiaa julkaisemalla relaatiotietokantojen hallintajärjestelmänsä kaupalliseen käyttöön. Se oli suunniteltu tietojen muokkaukseen kyselyitä ja liitoksia käyttäen. Ensimmäisen version julkistamisesta asti Oracle on toiminut suunnannäyttäjänä, ja Gartnerin mukaan Oraclella olikin vuonna 2011 jopa 50% osuus relaatiotietokantojen hallintajärjestelmien markkinoista. (Lee J. 2013, viitattu 3.11.2015.)

Oracle 2 sisälsi julkaisussa mukanaan vain perus SQL-toiminnot, mutta seuraavien 10 vuoden aikana Oracle kehitti ohjelmaa jatkuvasti alati muuttuvien tarpeiden mukaan. Yksi syy Oraclen voittokulkuun yhä nykyäänkin on sen tapa pysyä kehityksen aallonharjalla sekä reagoida markkinoiden muutoksiin nopeasti. (Lee J. 2013, viitattu 3.11.2015.)

3.1.2 Microsoft SQL Server

Microsoft SQL Server on relaatiotietokantojen hallintaohjelma, joka kehitettiin kilpailemaan Oraclen ja IBMn vastaavien palvelujen kanssa, ja jonka ensimmäinen Windows-versio Microsoft SQL Serveristä julkaistiin 1992. (Preston 2007, 562)

Ensimmäiset SQL Server -versiot pohjautuivat tiukasti Sybaselta hankittuun koodiin, mutta myöhemmin Microsoft vähitellen korvasi Sybasen koodin omallaan. Vuonna 2000 julkaistu SQL Server 2000 oli ensimmäinen versio, jossa ei ollut ollenkaan Sybasen koodia. (Lee J. 2013, viitattu 3.11.2015.)

3.2 PL/SQL

PL/SQL on Oraclen vuonna 1992 julkaisema proseduraalinen laajennus SQL:lle ja Oraclen relaatiotietokannoille. Se mahdollistaa SQL:n sisällyttämisen proseduraaliseen koodiin. PL/SQL:n funktiot, proseduurit, paketit, tyypit ynnä muut tallennetaan suoraan tietokantaan. Tämä mahdollistaa sen, että tietokantaa ei tarvitse kutsua useita kertoja yhtä tapahtumaa suorittaessa, mikä vähentää verkon kuormaa. (Hall T, viitattu 4.11.2015.)

PL/SQL koostuu kahdesta toisistaan eroavasta kielestä. Proseduraalinen koodi käsitellään PL/SQL-moottorin avulla, kun taas SQL lauseet lähetetään lausekkeen suorittajaan. Proseduraalisen koodin ja SQL lauseiden tiukasti yhteensidottu toiminta saa koodin näyttämään yhdeltä ohjelmointikieleltä kahden sijaan. (Hall T, viitattu 4.11.2015.)

3.3 PHP

PHP eli PHP: Hypertext Preprocessor on Rasmus Lerdorfin kehittämä, vuonna 1995 julkaistu avoimen lähdekoodin ohjelmointikieli. Nykyinen PHP 5 julkaistiin vuonna 2004 ja se toi mukanaa uuden version Zed-moottorista jonka ensimmäistä versiota käytettiin jo aiemmassa PHP 4 -versiossa. (The PHP Group a, viitattu 14.09.2015.)

Mikä erottaa PHP:n muista ohjelmointikielistä kuten Javascriptistä on se, että PHP koodin toteuttaa palvelin eikä sivun käyttäjän tietokone. Sivun käyttäjälle lähetetään siis vain koodin suorituksen tulos, eikä suoritettavaa koodia kuten JavaScriptiä käytettäessä.
(The PHP Group b, viitattu 14.09.2015.).

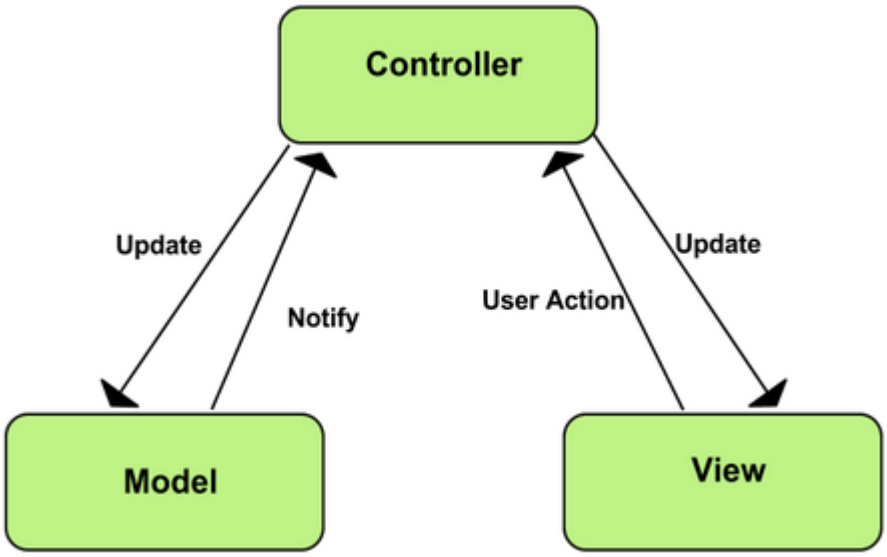
3.4 MVC, Model-View-Controller

MVC eli Model-View-Controller on sovellusarkkitehtuurin malli, joka määrittelee sen, miten käyttäjältä saatua ja käyttäjälle lähetettyä dataa käsitellään. (Google a, viitattu 21.9.2015.)

Modeliin eli Malliin talletetaan sovelluksen data-oliot, eikä se ole tietoinen käsittelijöiden tai näkymien olemassaolosta. Mallin muuton yhteydessä se lähettää tiedon muutosta sitä tarkkaileville tahoille. View eli näkymä sisältää sen osan sovelluksesta, jonka käyttäjä näkee, ja jonka tietoja käyttäjä voi muokata. Näkymä koostuu yleensä HTML-, CSS-, ja JavaScript-koodista. Controllerin eli käsittelijän tehtävä on toimia mallin ja näkymän välikappaleena. Se muuttaa näkymää mallin muutosten perusteella, sekä päivittää mallia käyttäjän näkymään tekemien muutosten perusteella lisäämiensä tapahtumankuuntelijoiden toimesta. (Google a, viitattu 21.9.2015.)

Käytännössä MVC-malli ja siitä periytyneet mallit, kuten MVVM (Model-View-View-Model) tai MVP (Model-View-Presenter), mahdollistavat sen, että koodi on paremmin uudelleenkäytettävissä malliin, käsittelijään ja näkymään jaon ansiosta. Kyseinen jaottelu mahdol-

listaa myös sen, että sovelluksen teko voidaan jakaa useammalle henkilölle, sekä sen, että sovelluksen eri osia voidaan kehittää yhtä aikaa. (Kanjilal, J., viitattu 22.9.2015.)



KUVIO 1. Model-View-Controller. (Malli-Näkymä-Käsittelijä) (Google a, viitattu 21.9.2015)

3.5 AngularJS ja hakemistorakenteet

Googlen ylläpitämä AngularJS on alunperin vuonna 2009 julkaistu avoimen lähdekoodin JavaScript-ohjelmistokehys. AngularJS kehitettiin helpottamaan dynaamisten verkkosivujen luomista vähentämällä tarvittavan koodin määrää ja tehostamalla yhteistoimintaa HTML-koodin kanssa muihin JavaScript-pohjaisiin kehyksiin nähden. (Google b, viitattu 16.09.2015.)

AngularJS:ää pidetään yleisesti MVVM-mallisena (Model-View-View-Model) arkkitehtuurina. MVVM perustuu MVC (Model-View-Controller) ja MVP (Model-View-Presenter) -malleihin. MVVM pyrkii erottelemaan käyttöliittymälogiikan muusta logiikasta mahdollistaen esimerkiksi käyttöliittymän ja muun logiikan yhtäaikaisen kehittämisen. (Osmani, A., viitattu 17.09.2015.)

AngularJS sopii ominaisuuksiensa ansiosta varsinkin yhden sivun sovelluksia varten. Yhden sivun sovelluksessa sivulataukset on pyritty vähentämään minimiin. Niissä sisältö ladataan aina tarvittaessa dynaamisesti, eikä koko sivua ladata koskaan uudestaan. Verrattuna esimerkiksi toiseen MVC-mallia käyttävään CodeIgniter-sovelluskehykseen AngularJS:llä tehdyt sovellukset voivat olla nopeampia ja kuormittavat vähemmän palvelinta dynaamisen sisällönlataamisen ansi-

osta. CodeIgniter on PHP-pohjainen sovelluskehys, joten dynaaminen lataaminen ei ole siinä mahdollista perusominaisuuksia käyttäen, vaan kyseistä toimintoa varten tarvitaan erillisiä JavaScript- ja jQuery-lisäosia. (Google c, viitattu 23.9.2015.)

AngularJS:n ja tämän opinnäytetyön tapauksessa sivulatauksia ja verkon kuormaa on vähennetty käyttämällä HTML-pohjia eli templateja. Jokainen sovelluksen sivu lataa samalla headerin ja footerin pohjat, ja sivun oma sisältö ladataan headerin sisältämään div-elementtiin. Headerissä on määritelty käytettävän AngularJS-sovelluksen nimi esimerkiksi “ng-app=’test’”, jolloin sivusto saa käyttöönsä kaikki tuon nimisen AngularJS-sovelluksen sisältämät controllerit, direktiivit ynnä muut, kunhan headerissä on myös muistettu ottaa käyttöön AngularJS-sovelluksen käyttämät JavaScript-tiedostot. (Google d, viitattu 23.9.2015.)

AngularJS:n routing-ominaisuuden ansiosta voidaan jokaiselle controllerille määritellä latauksen yhteydessä oma HTML-pohjansa. Jokainen erillinen sivu sisältää siis omat HTML-koodinsa, joka aina sivun osoitetta kutsuttaessa ladataan headerin div-elementtiin dynaamisesti. HTML-pohjien ollessa kytkettynä routing-ominaisuuden ansiosta controllereihin, voidaan HTML-koodin sekaan laittaa AngularJS:n direktiivejä, joilla sidotaan esimerkiksi käyttäjän tekemät muutokset muuttujiin jotka ovat controllerien käytettävissä. Controllerit reagoivat käyttäjän tekemisten perusteella, esimerkiksi lähettävät käyttäjän antamia tietoja rajapinnoille, jotka taas välittävät tiedot eteenpäin tietokantoja käsitteleville luokille, tai muuttavat näkymää reaaliajassa dynaamisesti. (Google e, viitattu 23.9.2015.)

4 TOTEUTUKSEN KUVAUS

Tässä kappaleessa käsitellään opinnäytetyöllä uudelleenkehitettävän avainrekisterisovelluksen kääntäminen PL/SQL-kieleltä PHP:lle, sovellukseen uutena osana tulevan AngularJS:n hyödyntäminen käyttöliittymän tekemiseen ja käyttäjän tekemien muutoksien käsittelijänä, tietokannan siirto Oraclesta Microsoftin SQL:ään, sekä käytetyt työkalut.

Opinnäytetyön ensimmäinen vaihe oli kuvausdokumentin kirjoittaminen toimeksiantajalle. Sen tarkoituksena oli kertoa mitä tehdään ja miten tehdään. Kuvausdokumenttiin suunniteltiin myös uusi tietokantarakenne vanhaa tietokantarakennetta mukaillen, esiteltiin sovelluksessa käytettävä kansiorakenne, sekä suunniteltiin alustavat ratkaisut sovelluksen tietokannan vaihtoon sekä tietojen siirtoon Oraclen tietokannasta Microsoftin tietokantaan.

4.1 PHP toteutuksessa

Sovelluksen kehitys aloitettiin PL/SQL-koodin kääntämisurakalla. Vanha PL/SQL:llä koodattu sovellus haluttiin ensin saada toimimaan PHP:lle käännettynä Oraclen tietokannassa, joten sovelluksen kehittäminen päätettiin aloittaa käännöstehtävällä.

Opinnäytetyössä tapahtuva käännöstyö toteutettiin siten, että toimeksiantajan toimittamilla kannettavilla tietokoneilla yhdistettiin Toad-sovellusta käyttäen Oamkin Oracle-tietokantaan josta päästiin tutkimaan avainrekisteriin kuuluvia PL/SQL-paketteja. Pakettien sisältämät SQL-lausekkeet olivat pääosin suoraan hyödynnettävissiä PHP-koodiin, muilta osin PHP-koodi kirjoitettiin omaa osaamista soveltaen, kunhan toiminnot olivat täysin samat kuin PL/SQL-koodilla.

Suuri osa Oamkin web-sovelluksista on toteutettu PL/SQL:llä, joten Oraclesta irtaantumisesta aiheutuva käännöstyö tulee olemaan laaja. Tästä opinnäytetyöstä saadusta kokemuksesta tulee olemaan apua käännöstyössä.

PL/SQL:llä koodatusta vanhasta sovelluksesta saatiin hyödynnettyä SQL-hakulausekkeet suurimmaksi osaksi kokonaisuudessaan, joskus vaadittiin pieniä muutoksia. SQL-hakulausekkeiden lisäksi PL/SQL-koodista ei ollut käännöstehtävään paljoa apua, vaan se

jouduttiin kääntämään omien kykyjen mukaan. Toiminnallisuukset piti siis kääntää PHP:lle niin, että PL/SQL-sovelluksen toiminnallisuuksia testattiin ja PHP-koodi tehtiin tämän toiminnallisuuden perusteella.

```
public function henkilo_avaimet($paketti) {

        $sql = "select a.a, a.b as luovutuspvm, a.c, a.d, b.a, b.b, c.a
            from taulu1 a, taulu1 b, taulu1 c
            where a.a = :asiakas_id and a.b = b.a
            and b.b = c.a order by a.a";
        $this->db->doParse($sql);

        $res = $this->db->doBindSql([
            ':asiakas_id' => $paketti['asiakas_id']
        ]);

        if (!$res) {
            $this->error = $this->db->getError();
            return false;
        }

        $avaimet = array();

        foreach($this->db->getRows() as $row) {
            array_push($avaimet, $row);
        }

        $avaimet['count']=count($avaimet);

        return $avaimet;
    }
```

KUVIO 2. PHP-funktio. Muuttujien nimiä muutettu toimeksiantajan pyynnöstä

Kuviossa 2 nähtävää PHP-funktiota vastaava PL/SQL-proseduuri on noin 200 riviä pitkä, sillä se sisältää myös käyttöliittymäkoodin, joten kuvaa siitä ei ole kannattavaa laittaa.

Sovelluksen käyttämät rajapinnat koodattiin myös PHP:lla. Rajapinnoissa AngularJS:n lähettämän JSON-dataa (JavaScript Object Notation) käytetään tietokantaluokkien funktioiden kutsuissa parametreinä, ja funktion palauttama data palautetaan takaisin AngularJS:n käsiteltäväksi, oli se sitten epäonnistunut tai onnistunut funktiokutsu.

```php
<?php
include_once "../../resources/config.php";

$res = $tietokantaolio->funktio($_POST);

if (!$res) {
    $response['errors']['message'] = $tietokantaolio->getError();
}
else {
    $response['success']= 'true';
    $response['message']='Onnistumisiesti';
    $response['palautuspaketti']=$res;
}

echo json_encode($response);

?>
```

KUVIO 3. Esimerkki rajapinnasta. Muuttujien nimiä muutettu.

Sovelluksen asetustiedostossa (config.php) määritellään muun muassa sovelluksen käyttämä tietokanta. Tietokannan vaihto Oraclesta Microsoftiin ja toisinpäin onnistuu vaihtamalla haluttu tietokanta kuviossa 4 nähtäville riveille. Asetustiedosto sisällytetään jokaiselle sitä tarvitsevalle tiedostolle, kuten rajapinnat, käyttämällä PHP:n include-toimintoa. Näin muut tiedostot osaavat käyttää oikeaa tietokantaa.

Kuviossa 4 nähtävät -load-tiedostot sisältävät nimeänsä vastaavan tietokannan yhdistämiseen vaadittavat funktiot. Näin ollen esimerkiksi rajapinnoissa saadaan yhdistettyä oikeaan tietokantaan aina, kunhan config.php on otettu tiedostossa käyttöön ja sieltä löytyy oikean tietokannan nimi.

```php
// tietokannan includaaminen
include_once "microsoftload.php";
//include_once "oracleload.php";
$config['kanta']='microsoft';
```

KUVIO 4. Tietokannan valinta config.php -tiedostossa.

Microsoftin tietokantaluokka toteutettiin PHP:lla siten, että Oraclea varten kirjoitettu tietokantaluokka kopioitiin, ja sen SQL-hakulausekkeet käännettiin uutta Microsoftin tietokantaa vastaavaksi ja Oracle-tietokannan käsittelyyn käytetty PDO vaihdettiin Osekkin käyttämään

MSSQL:n omaan käsittelytyyliin. Jos sovellus halutaan saada toimimaan myös esimerkiksi MySQL-tietokannassa, se onnistuu kopioimalla vastaavaan tapaan sille oma tietokantaluokkansa, johon muutetaan SQL-lausekkeet ja haluttu käsittelytyyli.

Kuvioissa 5 ja 6 kutsuttavat funktiot, kuten doBindSQL ja doParseParam kuuluvat omiin luokkiinsa. On siis olemassa kaksi luokkaa; toisella ajetaan Microsoftin SQL-lausekkeet ja sidotaan haluttu data lausekkeisiin, ja toinen jossa tehdään sama Oraclen SQL-lausekkeille ja halutulle datalle. Näissä luokissa on myös funktiot merkistön vaihtamiselle, jotta käsiteltävästä datasta saadaan tietokannan käyttämää utf-8 -muotoista merkistöä.

.

```
public function hae_yksikko_idlla($paketti){
        $yksikko= array();
        $sql = "select a from taulu where id=:id";
            $this->db->doParse($sql);

            $res = $this->db->doBindSql([
                ':id' => $paketti['yksikko'],
            ]);

            if (!$res) {
                $this->error = $this->db->getError();
                return false;
            }

            foreach($this->db->getRows() as $row) {
                array_push($yksikko, $row);
            }
            return $yksikko;
    }
```

KUVIO 5. PHP-funktio PDO:lla. Muuttujien nimiä muutettu.

```
public function hae_yksikko_idlla($paketti){
        $yksikko= array();
        $sql = "select a as a_nimi from taulu where id=(?) and rivi2=(?)";
        $param= array (
            $paketti['yksikko'],'0'
        );

        $res=$this->db->doParseParam($sql, $param);

        if (!$res) {
            $this->error = $this->db->getError();
            return false;
        }

        foreach($this->db->getRowsAssoc() as $row) {
            array_push($yksikko, $row);
        }
        return $yksikko;
    }
```

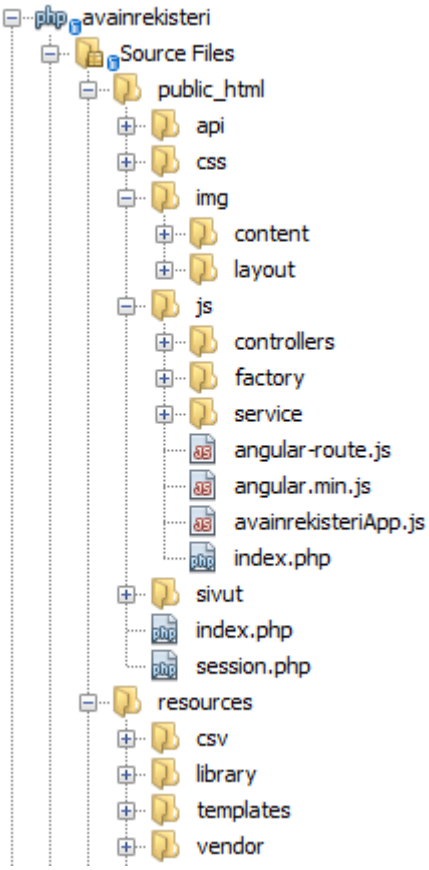
KUVIO 6. PHP-funktio MSSQL:n binditoiminnoilla. Muuttujien nimiä muutettu.

4.2 AngularJS

AngularJS on sovelluksessa kokonaan uusi osa. Koska vanha sovellus oli kokonaisuudessaan tehty PL/SQL:llä, oli sen käyttöliittymä tehty myös PL/SQL:llä. Uuteen sovellukseen käyttöliittymä haluttiin erottaa kokonaan muusta logiikasta, joten käyttöliittymän tekoon valittiin AngularJS.

AngularJS oli tämän opinnäytetyön tekijöille tuttu entuudestaan harjoittelun ajalta, eikä sitä ole kuullun perusteella käsitelty paljoa opinnäytetöissä, joten AngularJS on monelta kantilta olennainen osa tätä opinnäytetyötä.

Koska AngularJS:ää käytettiin myös opinnäytetyön tekijöiden työharjoittelun aikana, saatiin sinä aikana tehdyistä sovelluksista hakemistopohjan valmiina. Kuten kuviosta 7 näkee, kaikki tiedostot on järjestelty loogisesti omiin hakemistoihinsa ja alihakemistoihinsa. Tämä järjestely auttaa varsinkin silloin, kun sovelluksen sisältämien tiedostojen määrä kasvaa suureksi.



KUVIO 7. Uuden sovelluksen kansiorakenne.

Tämän opinnäytetyön tekijöiden työharjoittelun aikana tekemästä AngularJS-pohjaisesta ASAPv2-sovelluksesta saatiin myös paljon muutakin pohjaa, jota pystyttiin käyttämään hyväksi tätä uutta sovellusta kehitettäessä. Mallit AngularJS:n käsittelijöitä ja sovelluksia varten olivat suoraan hyödynnettävissä uudessa sovelluksessa.

Käyttöliittymätiedostojen sisältämät HTML-koodit saatiin suoraan kopioitua vanhan sovelluksen lähdekoodista selaimen avulla, joten käyttöliittymää ei jouduttu aloittamaan ihan tyhjästä.

AngularJS-sovelluksen kehitys aloitetaan yleensä määrittelemällä Angular-app omaan tiedostoonsa. Sovelluksen header-tiedostossa määritellään käytettävä Angular-app. Header-tiedosto sisältää myös div-elementin, jonka sisälle aina syötetään dynaamisesti sivu, johon selaimella on menty.

Controllerit, eli käsittelijät yhdistetään oikeisiin HTML-koodia sisältäviin pohjiin käyttämällä AngularJS:n routing-ominaisuutta, kuten kuviosta 8 nähdään. Käytännössä tämä toimii siis niin, että selaimen osoite osoittaa käsittelijään, ja kun tämä käsittelijä ladataan, ladataan samalla

siihen liitetty HTML-pohja ja tämä syötetään header-tiedoston sisältämään div-elementtiin. Tärkeää on myös, että käsittelijöitä luodessa käsittelijä on lisätty oikeaan Angular-appiin. Käsittelijöiden JavaScript-tiedostot täytyy myös ottaa käyttöön kuten tavallisessa JavaScriptiä käyttävässä sovelluksessa.

```
var avainrekisteriApp = angular.module('avainrekisteriApp', ['ngRoute', 'services.breadcrumbs']);

    avainrekisteriApp.config(function($routeProvider) {
            $routeProvider

                .when('/', {
                        templateUrl : 'sivut/sivu1.php',
                        controller  : 'sivu1Controller'
                })
                .when('/sivu2', {
                        templateUrl : 'sivut/sivu2.php',
                        controller  : 'sivu2Controller'
                })
```

KUVIO 8. Angular-app ja routing-toiminto.

Käsittelijöissä käsitellään näkymien data, eli käyttäjän näkemien sivujen tiedot käyttäjän tekemien muutosten ja toimintojen perusteella. Käsittelijästä tehdään tarvittaessa kutsuja rajapintoihin, joiden kautta tehdään hakuja tietokantaan (KUVIO 9). Rajapintakutsut tehdään jQueryn http-toimintoja hyödyntäen. Rajapintojen saama ja palauttama tieto on JSON-muodossa.

```
angularApp.controller('avainrekisteri1Controller', function ($scope, httpFactory, $location, loginFactory) {

    $scope.muuttuja1 = loginFactory.funktioKutsu();

    $scope.taulukko = {yksi : "", kaksi : "", kolme: "henkilokunta"};

    $scope.muuttuja2 = "1";

    $scope.funktio = function () {

        if ($scope.taulukko.yksi.length < 1) {
            $scope.statusfail = "Riittämättömät hakuehdot";
        }
        else {
            // rajapintakutsu
            httpFactory.doHttpPost('api/rajapintatiedosto.php', $scope.taulukko, function(data) {
                $location.path("/sivu2");
            });
        }
    };

});
```

KUVIO 9. Esimerkki controllerista eli käsittelijästä. Muuttujia muutettu.

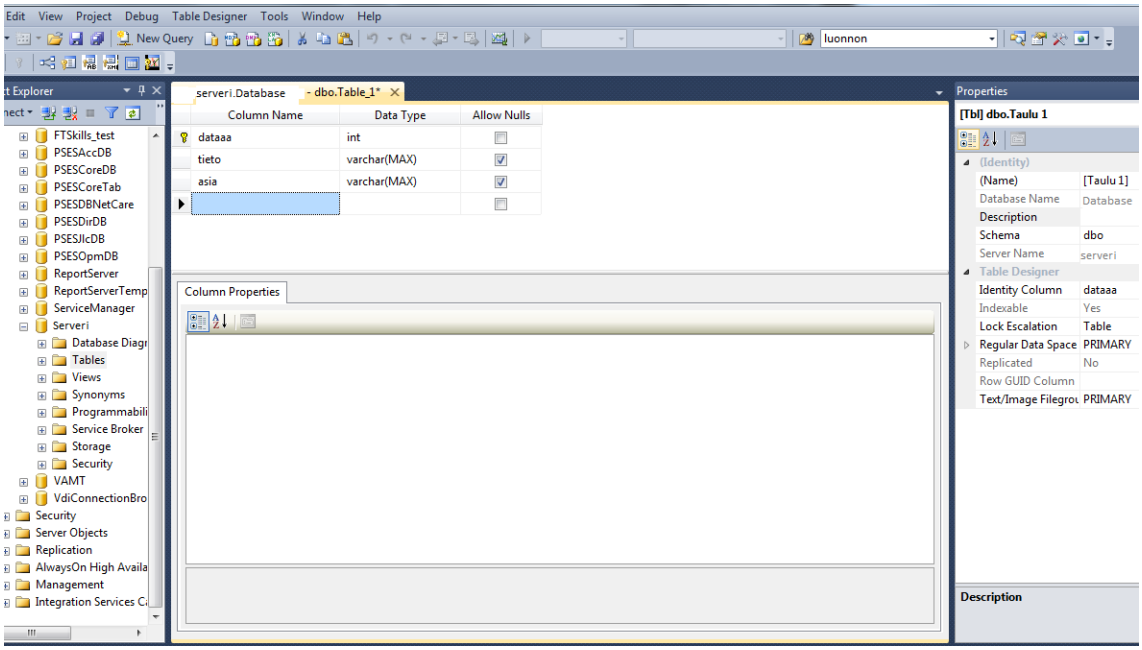
4.3 Tietokannan siirto

Vanha avainrekisterin tietokanta on alunperin tehty 2000-luvun alussa, ja sitä on sittemmin viimeisen lähes 15 vuoden aikana muokattu vastaamaan muuttuneita vaatimuksia lisäämällä tarvittaessa uusia kenttiä, jonka takia tietokanta on rakenteeltaan levinnyt ja muuttunut hankalasti luettavaksi.

Opinnäytetyössä tehtävänä oli käydä vanha avainrekisterin tietokanta läpi ja tehdä sen pohjalta uusi tietokantapohja Microsoftin SQL:llä. Ennen kuin uutta tietokantaa voitiin kuitenkaan alkaa rakentaa, täytyi selvittää monimutkaiseksi kasvaneen vanhan tietokannan sisältö sekä erotella turhat tietokentät tärkeistä.

Uuden tietokannan luonnissa päätavoitteeksi otettiin selkeys ja yksinkertaisuus. Vanhassa tietokannassa vaivanneet kymmenet eri puhelinnumerot ja muut vastaavat erikoisuudet, sekä muuten täysin tarpeettomat tiedot, poistettiin kerralla, ja tietokannan käyttämien taulujen määrää vähennettiin kuudesta taulusta viiteen tauluun yhdistämällä henkilöiden tiedot yhteen tauluun riippumatta siitä mistä heidän tietonsa on kantaan haettu.

Uusi tietokanta luotiin Microsoft SQL Server Management Studio -ohjelmalla, jonka avulla tietokantoja voidaan luoda sekä muokata tarpeen mukaan. Kuviosta 10 on nähtävissä tietokantataulun hallintasivu, jossa käyttäjä voi muokata ja luoda uusia tietokantatauluja tarpeen sekä omien hallintaoikeuksiensa mukaan. Visuaalisen design-näkymän lisäksi kantoja voi muokata myös skriptejä käyttäen, jolloin esimerkiksi taulun poisto ja uudelleen luonti käy nopeasti tallennetun skriptin avulla.



KUVIO 10. Microsoft SQL Server Management Studio

Tietokannan siirtoon oli alunperin tarkoitus käyttää Microsoftin omaa SQL Server Migration Assistant -ohjelmaa joka on tarkoitettu juuri tietokantojen siirtoon Oraclen kannasta Microsoftin kantaan. Siirtoa ohjelmalla yritettiin useita kertoja, mutta ohjelman huomattiin olevan tarkoitettu vain kannan kopioimiseen uudelle alustalle, kun taas tarkoituksena oli siirtää vain vanhan kannan tiedot uuteen tietokantapohjaan.

SQL Server Migration Assistant -ohjelman oltua käyttöön kelpaamaton, täytyi jokaiselle siirrettävälle taululle kehittää oma PHP-skriptinsä, jolla taulun tiedot siirrettiin CSV-tiedostoon (Comma-separated Values). Koska vastaavaa ei ollut aiemmin tehty, jouduttiin etsimään ja selvittämään parhaan keinon toteuttaa siirtokoodi itse kokeilemalla useita erilaisia funktiopohjia ja mallikoodeja ennen oikean löytämistä. Koodin yksinkertaisuuden kannalta hyvä asia on se, että PHP 5 sisältää tehtävää huomattavasti helpottavan fputcsv()-funktion jonka ansiosta koko siirtokoodi on pituudeltaan vain muutamia rivejä, kuten kuviosta 11 on nähtävissä.

```php
$sql = "select lause jossa haetaan exportattavan taulun tiedot";
$db->doSql($sql);

$tiedot = $db->getRowsAssoc();

$output = fopen("C:/csvfilu.csv",'w') or die("Can't open file");
fprintf($output, chr(0xEF).chr(0xBB).chr(0xBF));

foreach($tiedot as $per_rivi) {
    fputcsv($output, $per_rivi,",");
}
```

KUVIO 11. PHP-skripti CSV-tiedostoon tallennus, sisältää pseudokoodia ja muuttujien nimet muutettu.

Joidenkin taulujen tallennuksessa CSV-muotoon esiintyi ongelmia koska vanhassa kannassa ei ollut minkäänlaista tietojen tarkistusta. Tarkistusten puute johti CSV:n luontiskriptin ajoittaiseen kaatumiseen virheellisten merkkien takia. Ongelmaa ratkottiin sekä etsimällä sopivaa välimerkkiä jota ei esiintynyt missään itse kannassa, että käymällä muokattava taulu PHP-skriptissä ensin läpi ja muokaten virheellinen tieto oikeaan muotoon ennen CSV-muunnosta. Lisäksi kantaan siirrettäessa, varsinkin kahta eri taulua yhdistettäessä yhdeksi uudeksi tauluksi, tarvittiin skriptejä jotka yhtenäistivät taulujen tiedot jottei uuteen tauluun tulisi puutoksia riippumatta lähtötaulujen tiedoista.

Taulujen foreign key-avainten luontia varten vaadittiin PHP-skripteissä myös omaa lisäystään, jolloin jo uuteen tietokantaan tallennetun taulun tiedot haettiin ja verrattiin niitä tallennettavaan toiseen tauluun jotta voitiin määrittää oikeat rivit vastaamaan toisiaan myös uudessa kannassa.

```php
$filename='C:/csvtiedosto.csv';
$file=array_map('str_getcsv',file($filename),array_fill(0,count(file($filename)),","));
for ($a=0;$i<count($file);$i++){

    //muokataan $file-data microsoft tietokannan ymmärtämään muotoon
    $file[$a]=mb_convert_encoding ($file[$a][0], 'WINDOWS-1252', 'UTF-8');

    $sql="insert lause jossa tallennetaan csv-tiedostosta haettu data uuteen tauluun";
    $param=array
        ($file[$a][0],$file[$a][1]);
    $db->doParseParam($sql, $param);
};
```

KUVIO 12. PHP-skripti CSV-tiedon luku ja tallennus uuteen tietokantaan. Sisältää pseudokoodia ja muuttujien nimet muutettu.

CSV-tiedoston lukemiseen ja uuteen kannan tiedon siirtämiseen luotiin myös jokaiselle taululle oma PHP-skriptinsä kuten kuviosta 12 on nähtävissä. Str_getcsv-funktion avulla aiemmin luodusta CSV-tiedostosta luotiin taulukko joka sitten tallennettiin uuteen tietokantaan insert-lauseella.

Siirtoa tehdessä paljon ongelmia aiheutti se, ettei Microsoftin tietokanta ymmärrä UTF-8 merkistöä, joten ennen kantaan tallennusta tiedot täytyi kääntää WINDOWS-1252 -merkistöön. Merkistön kääntäminen olisi voitu välttää mikäli nvarchar kenttiin tallentaminen olisi onnistunut, jolloin tietokanta olisi itse hoitanut kääntämisen myös UTF-8 muodosta. Käytettyjen funktioiden rajoituksista johtuen insert-lauseisiin ei voinut lisätä nvarcharin vaatimaa N-merkkiä talletettavien tietojen eteen, vaan jouduttiin tyytymään pitkän yrittämisenkin jälkeen muuttamaan tietokannan nvarchar kentät varchareiksi ja käyttämään merkistön kääntöä PHP-skriptissä.

4.4 Työkalut

Opinnäytetyössä tapahtuvaan sovelluksen kehitykseen käytettiin NetBeans -nimistä sovellusta. NetBeans on Oraclen kehittämä ja ylläpitämä Java-. HTML-, ja PHP-sovellusten kehitykseen tarkoitettu sovellus. Tässä opinnäytetyössä NetBeansiä käytettiin sovelluksen uusimiseen kokonaisuudessan. Sillä siis toteutettiin sovelluksen PHP-luokat, -apit, sekä HTML-sivut ja AngularJS:n JavaScript.

Toad on Dellin luoma Oraclen tietokantasovellusten ja tietokantojen kehittämiseen ja ylläpitoon tarkoitettu sovellus. Tässä opinnäytetyössä Toadia käytettiin vanhan sovelluksen proseduurien ja tietokantojen selaamiseen.

Windowsin komentoriviä käytettiin tietokantojen siirtoa varten tehtyjen PHP-skriptien ajamiseen. PHP-skriptien ajaminen komentorivillä on mahdollista jos XAMPP on asennettuna ja se on käynnissä.

XAMPP on Apachen kehittämä ja ylläpitämä sovellus, joka mahdollistaa mm. käyttäjän oman koneen toimimisen HTTP-palvelimena.
XAMPPia käytettiin uuden sovelluksen testaamiseen Oracle-tietokannan kanssa. Microsoftin tietokanta on Oamkin palvelimella, joten XAMPPia ei sen testaamiseen tarvittu.

Opinnäytetyön aikana syntyneet koodit ladattiin Oamkin git-palvelimelle, joka on versionhallintaan ja varmuuskopiointiin soveltuva järjestelmä. Koodin siirtämiseen git-palvelimelle käytettiin SourceTree -nimistä avoimen lähdekoodin ohjelmaa. SourceTree on Atlassianin kehittämä ja ylläpitämä ohjelma, joka toimii käyttöliittymänä Git-varastolle. Uusi versio ohjelmasta syntyy aina, kun SourceTree:llä lataa koodit gitiin, esimerkiksi kun päivän työt on saatu valmiiksi tai virheitä on korjattu, ja versiot voidaan nimetä haluamalla tavalla ja niille voidaan kirjoittaa haluttu viesti, kuten esimerkiksi "Korjattu koodista bugeja". Jos myöhemmissä versioissa ilmenee niin pahoja virheitä, ettei jatkaminen onnistu, gitin ansiosta on mahdollista palata takaisin versioon, jossa kyseistä virhettä ei enää ilmene.

Internetistä löytyy myös git-projekteja sisältävä gitHub-palvelu, johon sen käyttäjät voivat ladata omia projektejaan. Julkisiksi asetetut projektit ovat muiden käyttäjien nähtävissä ja ladattavissa. Käyttäjät voivat kommentoida projektien eri versioita ja jopa koodin yksittäisiä rivejä. Näin ollen gitHub voi olla todella hyödyllinen järjestelmä sellaisen sovelluksen kehitykseen, johon halutaan ottaa yhteisön mielipiteet ja apu mukaan.

4.5 Ongelmat ja niiden ratkaisut

Kuten aiemmin onkin jo mainittu, suurimmat ongelmat tulivat eteen tietokannan siirrossa. Vanha ja uusi tietokanta eivät olleet identtisiä, joten siirtoon valittu sovellus ei suostunut ollenkaan yhteistyöhön. Koska kyseessä oli vanha sovellus, ei tätä koettu ongelmaksi, sillä siirto voitaisiin suorittaa myös skripteillä, ja samalla saataisiin valittua vanhasta tietokannasta vain halutut tiedot. Jos taas tehdään esimerkiksi kokonaan uusi sovellus aivan alusta, voidaan käytettävät tietokannat suunnitella identtisiksi, jotta tietojen siirto tietokannasta toiseen onnistuisi vaivattomammin siirtoon tarkoitettujen työkalujen avulla.

Kun tietokannan tietojen siirrossa päädyttiin skriptien käyttöön, tuli eteen jälleen uusia ongelmia. Vanhan tietokannan tiedot siirrettiin skriptien avulla CSV-tiedostoihin, jokaista tietokannan taulua kohden oma tiedostonsa. Tietokannan ollessa vanha ja siivoton, CSV-tiedostot menivät skriptejä ajettaessa usein rikki, eli tiedot katkesivat kesken kaiken. Tiedoston rikkova data oli kohtuullisen helppo paikantaa, ja sen saamiseksi mukaan tiedostoon yritettiin siitä poistaa mahdollisia ongelmia aiheuttavia merkkejä, tuloksetta. Jotta CSV-tiedostoista saatiin ehjiä, päädyttiin siihen,

että tiedoston rikkovia datoja on turha kuljettaa mukana uuteen kantaan, joten ne jätettiin pois lisäämällä kutakin rikkovaa dataa kohden oma koodinpätkänsä siirtoskriptiin. Näin ollen uudesta tietokannasta saadaan siistimpi, ja käyttöliittymään laitettujen tietojen tarkistuksien ansiosta tietokannan väärin täyttäminen ei ole enää mahdollista.

Muilta osin sovelluksen kehitys onnistui ilman suurempia ongelmia. AngularJS, PL/SQL, ja PHP olivat tekijöille jo ennestään sen verran tutut, ettei mitään etenemistä lamauttavia ongelmia tullut vastaan. Jos ongelmia tuli, niihin saatiin nopeasti ja vaivattomasti apu internetistä, sillä se on pullollaan ihmisiä, joilla on ollut samankaltaisia ongelmia ja jotka ovat saaneet niille jo vastauksen.

5 YHTEENVETO JA POHDINTA

Idea opinnäytetyön tekemiselle Oamkin tietohallinnolle lähti liikkeelle jo harjoittelumme aikana, jolloin tietohallintopäällikkö Samuli Malinen kertoi meille mahdollisuudesta tehdä heille opinnäytetyö. Opinnäytetyön aihe tarkentui vasta harjoittelun jälkeen, ja sen aiheeksi saatiin tulevaisuuden tarpeen valossa avainrekisterin käännösprosessi. Aiheeseen ei ollut vaikea tarttua, sillä tiesimme, että opinnäytetyöstä olisi heille oikeasti hyötyä tulevaisuudessa, kun Oamk siirtyy Microsoftin palveluihin Oraclen palveluista.

Aluksi opinnäytetyön aihe aiheutti pientä pakokauhua, sillä emme tienneet eri tietokantojen eroja, emmekä olleet ennen kääntäneet sovellusta kieleltä toiselle tai siirtäneet tietokantaa. Pientä toivoa antoi harjoittelun aikana saatu kokemus tekemästämme sovelluksesta. Uusi avainrekisteri oli tarkoitus tehdä samalla tyylillä ja samoilla työkaluilla kuin tuo harjoittelun aikana tekemämme sovellus. Ainoina eroina oli koodin kääntäminen kieleltä toiselle, sekä tietokannan siirto.

Opinnäytetyön kehitystehtävä, eli avainrekisterin kääntäminen PHP:lle ja tietokannan siirto Oraclesta Microsoftiin aloitettiin kesäkuun viimeisellä viikolla, ja se valmistui elokuun viimeisellä viikolla. Sovelluksen kehitykseen meni karkeasti arvioiden 200-300 tuntia per henkilö. Suurin osa ajasta meni tietokannan siirtoon, sillä sen kanssa ilmeni paljon ongelmia, eikä vastaavaa siirtoa ollut ennen tehty koko Oamkin organisaatiossa. Vanhan tietokannan tietoja joutui myös siivoamaan, jotta siitä saatiin muodostettua ehjä CSV.

Uuden sovelluksen kääntäminen ja kehitys onnistui pääosin ilman suurempia ongelmia, mutta suurimmat ongelmat tulivat vastaan tietokannan siirtämisessä. Vastaavaa siirtoa ei aiemmin ollut Oamkissa tehty, joka osaltaan auttoi motivoimaan ja viemään siirron loppuun, vaikka siirron aikana moraali murenikin useaan otteeseen. Siirtoon kului loppujen lopuksi aikaa noin kuukausi, tosin emme tehneet täysien työpäivien edestä joka päivä töitä, ja osa ajasta kului meistä riippumattomien seikkojen takia, kuten esimerkiksi puuttuvien oikeuksien ja tietokannan muiden ongelmien takia.

Opinnäytetyön raportti aloitettiin vasta käytännön osuuden toteuttamisen valmistumisen jälkeen syyskuun alussa. Näin tehtiin, koska olimme varmoja, että asiat olisivat vielä tuoreessa muistissa, sekä jo valmiina ollut alustava sisällysluettelo auttoi painamaan asiat vielä paremmin muistiin.

Opinnäytetyön raportin kirjoittaminen sujui ennalta odotettua paremmin. Aloitimme kirjoittamisen noin 2,5 kuukautta ennen suunniteltua määräaikaa, joten emme tunteneet olevamme kovassa kiireessä. Tästä johtuen sovimme, että molemmat kirjoitamme joka päivä vähintään yhden sivun. Tämä auttoi vähentämään opinnäytetyöstä aiheutuvaa stressiä, mutta silti edisti opinnäytetyötä hyvää vauhtia ilman kiirettä.

Sovitun tahdin auttamana kirjoittaminen ei aiheuttanut suuria ongelmia, vaan jokaiseen suunniteltuun otsikkoon saatiin tekstiä ja kuvia sopivasti. Ainoa huoli syntyi siitä, että koska meitä oli kaksi, jouduimme käyttämään kirjoittamiseen Google Docsia, joka myös tarkoitti sitä, ettei opinnäytetyöhön tarkoitettua pohjaa voitu vielä kirjoittamisen alkuvaiheessa käyttää, vaan valmis teksti siirretään vasta raportin valmistuttua viralliseen pohjaan. Tämä aiheutti sen, ettei lopullisen tekstin pituudesta ollut täyttä varmuutta, vaikka Docsin tiedostolle oli aseteltu virallisen pohjan marginaalit ja muut tärkeimmät asetukset.

Opinnäytetyön tuloksena oli valmis avainrekisterisovellus, joka menee vahtimestareille käyttöön. Opinnäytetyön toimeksiantaja Oamkin tietohallinto ja Osekk saavat opinnäytetyöstä arvokasta kokemusta tulevaisuudessa tapahtuvaa siirtoa Oraclen palveluista Microsoftin palveluihin varten.

Oman oppimisen kannalta opinnäytetyö oli todella antoisa. Saimme hyvän käsityksen siitä, minkälainen urakka sovelluksen kääntäminen kieleltä toiselle on, sekä opimme siirtämään tietokantoja käsintehdyillä PHP-skripteillä kun siirtoa varten tehdyillä sovelluksilla siirtäminen ei onnistunut. Myös uuden tietokannan rakenteen suunnittelimme itse hyödyntäen vanhan tietokannan pohjaa.

Jos jotain tekisimme toisin, luultavasti miettisimme uudestaan sen, mistä opinnäytetyössä lähdetään liikkeelle. Jälkeenpäin katsottuna lähtisimme luultavasti liikkeelle tietokannan siirrosta, sillä siihen meillä meni eniten aikaa. Myös raporttia olisi voinut jälkeenpäin ajateltuna kirjoittaa edes hieman yhtä aikaa kehityksen kanssa, vaikkei meille koitunutkaan suurta ongelmaa siitä, että aloitimme kirjoittamisen vasta kehitystyön valmistuttua.

Myös opinnäytetyön aikataulua olisi voinut tiivistää. Aloitimme kesäkuun lopulla ja tarkoituksena oli saada valmiiksi marraskuun puolessa välissä, joten aikaa oli reilusti, eikä joka päivä tarvinnut

tehdä kovin montaa tuntia. Toisaalta väljä aikataulu mahdollisti sen, että pystyimme pitämään välissä jopa viikonkin taukoja, eikä opinnäytetyö näin ollen päässyt aiheuttamaan liikaa stressiä.

Loppujen lopuksi selviydyimme mielestämme urakasta kiitettävästi. Kun otetaan huomioon, että opinnäytetyössä oli asioita jotka eivät olleet edes toimeksiantajalle tuttuja, niin siitä suoriutuminen on mielestämme varteenotettava teko.

LÄHTEET

Google a, MVC Architecture, viitattu 21.09.2015. https://developer.chrome.com/apps/app_frameworks.

Google b, What is Angular?, viitattu 4.11.2015. https://docs.angularjs.org/guide/introduction.

Google c, AngularJS, viitattu 23.9.2015. https://angularjs.org/.

Google d, Tutorial: 2, viitattu 23.9.2015. https://docs.angularjs.org/tutorial/step_02.

Google e, Developer Guide, viitattu 23.9.2015. https://docs.angularjs.org/guide/controller.

Hall T. Introduction to PL/SQL, viitattu 14.09.2015. https://oracle-base.com/articles/misc/introduction-to-plsql.

Ibm, Information management system, viitattu 23.11.2015. http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/ibmims/.

Janalta Interactive Inc., viitattu 03.11.2015. https://www.techopedia.com/definition/1180/data-migration.

Kanjilal, J., Implementing the MVC Design Pattern in ASP.NET, viitattu 22.9.2015. http://aspalliance.com/1562_Implementing_the_MVC_Design_Pattern_in_ASPNET.3.

Lee J 2013, Oracle vs. MySQL vs. SQL Server: A Comparison of Popular RDBMS, viitattu 3.11.2015. https://blog.udemy.com/oracle-vs-mysql-vs-sql-server/.

Oracle 2007, viitattu 17.09.2015. http://www.oracle.com/us/corporate/profit/p27anniv-timeline-151918.pdf.

Oracle b, A Relational Database Overview, viitattu 26.10.2015 https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html.

Osmani A 2012, Understanding MVVM – A Guide For JavaScript Developers, viitattu 17.09.2015. http://addyosmani.com/blog/understanding-mvvm-a-guide-for-javascript-developers/.

Preston C 2007. Backup & Recovery: Inexpensive Backup Solutions for Open Systems. O'Reilly Media, Inc.

TechTarget, What is database?, viitattu 26.10.2015. http://searchsqlserver.techtarget.com/definition/database.

The PHP Group a, viitattu 14.09.2015. http://php.net/manual/en/history.php.php.

The PHP Group b, What is PHP?, viitattu 14.09.2015. http://php.net/manual/en/intro-whatis.php.