TIETOKANTARATKAISUT MOBIILI- JA VERKKOSIVUYMPÄRISTÖISSÄ

Case oldtimerTimer

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulun opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Visamäki, syksy 2014

Janne Remes

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

Visamäki
Tietojenkäsittelyn koulutusohjelma
Systeemityö

Tekijä Janne Remes **Vuosi** 2014

Työn nimi Tietokantaratkaisut mobiili- ja verkkosivuympäristöissä
Case oldtimerTimer

TIIVISTELMÄ

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa tietokantaratkaisut oldtimerTimer-mobiilisovellukseen ja -internetkäyttöliittymiin. OldtimerTimer-sovelluksesta toteutettiin kesän 2014 aikana prototyyppiversio, jota on tarkoitus kehittää eteenpäin kesän aikana tehtyjen havaintojen pohjalta.

OldtimerTimer-mobiilisovelluksella vanhus voi selailla tulevia tapahtumia ja lisätä niitä mobiilikalenteriinsa. Tapahtuman jälkeen mobiilisovellus kysyy vanhukselta, osallistuiko tämä tapahtumaan. Vanhuksen lähiomainen pystyy internetkäyttöliittymän kautta lisäämään vanhuksen seurannan alle. Lähiomaisen nähtäville tulee myös, mihin tapahtumiin vanhus on osallistunut ja mihin ei.

Opinnäytetyön toimeksiantajatahoina toimivat hämeenlinnalainen Linnan Ateria Oy, jonka on tarkoitus ottaa oldtimerTimer-sovellus tulevaisuudessa käyttöön, sekä Hämeen ammattikorkeakoulu, jonka opiskelijat vastaavat järjestelmän teknisestä toteutuksesta.

Opinnäytetyössä hyödynnettiin kehittämisprojektin tietoperustaa, koska valmiin tuotoksen aikaansaamiseen hyödynnetään projektimaista tuotekehitystä. Tietoperustassa käsitellään käytettäviä järjestelmiä tietokantaratkaisujen kannalta, tietokannan peruskäsitteitä, tietokannan normaalimuotoisuutta sekä tietokannan haku- ja luontikomentoja MySQL-kyselykielellä. Myös scrum-projektinhallintamenetelmän vaikutusta tietokantaratkaisuihin käsitellään. Aineistona käytettiin tietokantasuunnittelua käsittelevää kirjallisuutta sekä internetsivuja käytetyistä järjestelmistä, scrum-projektinhallintamenetelmästä ja MySQL-kyselykielestä.

Opinnäytetyön tuloksena saatiin aikaan tietokantaratkaisuja hyödyntävä mobiilisovellus- ja internetkäyttöliittymäratkaisu. Prototyyppiversiossa esille nousseista haasteista sekä parannusehdotuksista laadittiin myöhempiä versioita varten erillinen raportti, jonka pohjalta myöhemmät työryhmät voivat kehittää tietokantaratkaisuja toimivammiksi.

Avainsanat tietokannat, MySQL, projektinhallinta

Sivut 25 s. + liitteet 12 s.

HAMK
UNIVERSITY OF APPLIED SCIENCES

Visamäki
Degree Programme in Business Information Technology
System Engineering

Author Janne Remes **Year** 2014

Subject of Bachelor's thesis Database solutions in mobile and web page environments
Case oldtimerTimer

ABSTRACT

The goal for the thesis was to design and create database solutions for oldtimerTimer mobile application and internet interfaces. During summer 2014 a prototype of the oldtimerTimer application was implemented. It is intended to be developed further based on findings made on summer.

OldtimerTimer mobile application is intented for senior citizens to browse events and add them to mobile calendar. After the event senior citizen is asked if he took part to the event. Senior citizen's next of kin is able to add senior citizen under his following through internet interface. Next of kin is also able to see in who events senior citizen has taken and who not.

The clients for the thesis were Linnan Ateria Ltd. with the intention of using oldtimerTimer in the future and HAMK University of Applied Sciences which students are responsible for system's technical execution.

Thesis' knowledge-base was a development project because project-like research and development is used to get the prototype. The theory part contains database solutions for the used systems, the basic concepts of database, normal forms of database and database's search and create commands by MySQL. Also scrum software development framework's affects to database solutions is handled. By material literature of creating databases and internet pages of used systems, scrum software development framework and MySQL were used.

The achievement of the thesis was a database solutions that mobile application and internet interface use. The challenges and the suggestions for improvement that was faced in prototype version were mentioned in separate report that will give later working groups possibility to create much more functional database solutions.

Keywords databases, MySQL, project management

Pages 25 p. + appendices 12 p.

API (Application programming interface)
Ohjelmointirajapinta, jonka avulla yksittäiset ohjelmat pystyvät vaihtamaan tietoja keskenään.

MySQL
Avoimen lähdekoodin tietokantaohjelmisto.

PHP (PHP: Hypertext Preprocessor)
Internetympäristön ohjelmointikieli.

Relaatiotietokanta
Relaatiotietokannassa on tauluja, joiden välillä on yhteyksiä. Taulut yhdistyvät toisiinsa yksilöllisillä avainarvoilla.

SQL (Structured Query Language)
Relaatiotietokantojen kyselykieli, jolla voidaan suorittaa tietokantaan hakuja, muutoksia tai lisäyksiä.

XML (eXtensible Markup Language)
Tiedon rakennetta kuvaava tekstimuotoinen merkintäkieli.

SISÄLLYS

1 JOHDANTO

Opinnäytetyössä laadittiin tietokantaratkaisut oldtimerTimer-mobiilisovelluksen ja -internetkäyttöliittymien prototyyppiversioon. Tietokantaratkaisujen työstäminen aloitettiin keväällä 2014 Hämeen ammattikorkeakoulun Digital Service Development -opintojaksolla. Opintojaksolla kartoitettiin kaikkia oldtimerTimer-sovelluksen toteutuksessa huomioitavia seikkoja.

Digital Service Development -opintojakson pohjalta suunnitelman keskiössä oli vanhuksille suunnattu mobiilipalvelu, jonka kautta vanhus voi selailla eri oppilaitoksissa olevia päivittäisiä ruokalistoja ja lisätä sopiva ajankohta mobiililaitteen kalenteriin. Vanhuksen lähiomaisella ja hoitohenkilöstöllä tuli olla mahdollisuus lisätä ruokailutapahtumia internetportaalin kautta vanhuksen mobiililaitteeseen. Ruokalistan ylläpitäjän tuli pystyä tallentamaan ruokalista internetportaalin kautta luettavaan muotoon. Ruoan loppumisesta yksittäisellä koululla tiettynä päivänä tuli tehdä erillinen merkintä, ja tiedon ruoan loppumisesta tuli siirtyä vanhuksen mobiililaitteeseen, mikäli hänellä oli tehty varaus ruokailla koululla kyseisenä päivänä.

Mobiilisovellus- ja internetkäyttöliittymistä vastaavat työryhmät ja Hämeen ammattikorkeakoulun projektista vastaava opettaja kävivät kesällä 2014 luotavasta oldtimerTimer-prototyyppiversiosta myös erillisiä keskusteluja. Näiden keskustelujen pohjalta oldtimerTimer-palveluun haluttiin mahdollisuus lisätä myös muita säännöllisesti toistuvia tai satunnaisempia tapahtumia kalenteriin. Projektin edetessä muiden tapahtumien rooli nousi keskeisempään asemaan, joten esimerkiksi ruokalistan ylläpitäjärooli jäi varsinaisesta tietokantatoteutuksesta pois. Ruokailumahdollisuuksien selailu limitettiin muiden tapahtumien kanssa samanlaiseen tietorakenteeseen, ruokalistaominaisuuksia ei erikseen korostettu tehdyissä ratkaisuissa.

Opinnäytetyön toimeksiantajatahoina toimivat Hämeen ammattikorkeakoulu ja Linnan Ateria Oy. Hämeen ammattikorkeakoulun opiskelijat toteuttavat järjestelmän, joka Linnan Ateria Oy:n on tarkoitus ottaa tulevaisuudessa käyttöön.

Tietokantaratkaisut toteutettiin mobiilisovelluksesta ja internetkäyttöliittymistä vastaavien työryhmien tietokannoille esittämien vaatimusten pohjalta. Järjestelmästä toteutettiin kesän 2014 aikana asiakkaalle myöhemmin syyslukukaudella esiteltävä prototyyppiversio. Prototyyppiversion pohjalta järjestelmää on tarkoitus kehittää eteenpäin Hämeen ammattikorkeakoulun ICT-projekteissa, jotka alkavat syyslukukaudella 2014. Projektinseurannassa hyödynnettiin prototyyppiversiota toteutettaessa scrum-projektinhallintamenetelmää.

Aiheen valintaan vaikuttivat opinnäytetyön tekijän kiinnostus tietokantoihin liittyviin aihealueisiin sekä halu toteuttaa tietokantaratkaisuja käytännössä, myös tulevaisuuden työelämää silmällä pitäen.

Opinnäytetyön tutkimuskysymyksiä ovat:

- Miten tietokanta suunnitellaan ja toteutetaan projektin aikana?
- Miten pitkälle mobiilisovelluksessa ja internetkäyttöliittymissä voidaan hyödyntää samoja tietokantaratkaisuja?
- Miten valittu projektinhallintajärjestelmä (scrum) tukee seurantansa osalta tietokantaratkaisujen syntyä?

2 KÄYTETTÄVÄT JÄRJESTELMÄT

Kesän 2014 aikana toteutetun oldtimerTimer-prototyyppiversion esiselvitysvaihe käynnistyi keväällä 2014 Digital Service Development -opintojaksolla. Opintojaksolla kartoitettiin projektin aikana mahdollisesti käyttöön otettavia järjestelmiä. Opintojakson jälkeen esille nousseita vaihtoehtoja pohdittiin tarkemmin prototyyppiversiota työstävän työryhmän kesken. Lopulliset valinnat käyttöönotettavista järjestelmistä tehtiin alkukesästä, jolloin tietokantaratkaisujen kartoitus oli jo aloitettu. Luvussa järjestelmät käsitellään sillä tasolla, miten ne kytkeytyvät tietokantaratkaisuihin.

2.1 MySQL

MySQL on alunperin suomalaissyntyinen avoimen lähdekoodin tietokantajärjestelmä, jonka kehityksestä vastaa Oracle Corporation (Hovi 2004, 4; MySQL n.d-b). MySQL pohjautuu SQL-standardiin, jonka kautta tietokantaan voi tehdä muun muassa tietokannan rakenteen muutoksia sekä tietokannan sisällön päivittämistä (MySQL-opetusmateriaali n.d.; Hovi 2004, 10).

MySQL valittiin oldtimerTimer-palvelun tietokantajärjestelmäksi sen aiemman tuttuuden ja helpoksi todetun käytettävyyden vuoksi. Myös projektin aikana mahdollisesti ilmeneviä tietokantoihin liittyviä haasteita pyrittiin minimoimaan sen osalta, ettei uuden ja vieraan tietokantajärjestelmän opetteluun kulutettaisi aikaa ja pitkitettäisi täten tietokantaratkaisujen syntyä.

2.2 WordPress

WordPress on internetsivujen sisällönhallintajärjestelmä, jolla on mahdollista tehdä muun muassa sivupohjia (Leiniö 2012). Ennen kuin WordPress asennetaan palvelimelle, tulee palvelimelle hallintapaneelin kautta asentaa tietokanta, johon tallentuvat WordPressin tekstitiedot, linkitykset ja toiminnot. Hallintapaneelin kautta siirrytään tietokannan luontiin, jossa luodaan tietokannan nimi, tietokannan käyttäjätunnus sekä tämän salasana. (vaisalamika 2011b.)

Kun WordPressin asennuspaketti puretaan, päästään käsiksi wp-config-sample-tiedostoon (vaisalamika 2011a; vaisalamika 2011c). Tiedostoon muokataan käytössä oleviksi WordPressin käyttämän tietokannan nimi se-

kä MySQL-tietokannan käyttäjätunnus ja salasana. Muutoksen jälkeen tiedosto uudelleennimetään wp-config-nimiseksi. Uudelleennimeämisen jälkeen asennuspaketti siirretään palvelimelle. (vaisalamika 2011c.)

2.3 phpMyAdmin

phpMyAdmin on php-ohjelmointikielellä toteutettu, erikseen ladattavissa oleva vapaan ohjelmiston tietokannanhallintatyökalu MySQL-tietokannoille. phpMyAdminin hallintapaneelin kautta pystyy hallinnoimaan tietokantoja sekä niiden taulusisältöjä ja taulujen välisiä suhteita. (phpMyAdmin n.d.)

phpMyAdmin-hallintapaneeliin pääsee syöttämällä internetselaimen osoiteriville palvelinosoitteen jatkoksi /phpmyadmin. Tämän jälkeen syötetään tietokannan käyttäjätunnus sekä salasana.

2.4 Android

Android on älypuhelimien ja mobiililaitteiden avoimen lähdekoodin käyttöjärjestelmä. Android-sovellusten toteutus tehdään Java-ohjelmointikielellä. (Android-Suomi 2012.)

Javassa, kuten muissakin ohjelmointikielissä, voidaan kieleen upottaa SQL-käskyjä. Java-kielessä käytetään kutsuttaessa aktivoituvaa JDBC (*Java Database Connectivity*)-SQL API -rajapintaa. (Hovi 2004, 16 & 234.)

3 TIETOKANNAN PERUSKÄSITTEET

Tietokannaksi käsitetään yleensä tietojoukko, jonka sisällä olevat tiedot kuuluvat loogisesti yhteen (Hovi, Huotari & Lahdenmäki 2005, 4). Nykyään käytettävät tietokannat ovat pääosin relaatiotietokantoja, jotka perustuvat muun muassa matematiikkaa ja joukko-oppia hyödyntävään relaatiomalliin (Hovi ym. 2005, 5 & 7). OldtimerTimer-järjestelmän tietokantaratkaisut ovat relaatiomallin mukaisia.

3.1 Taulu, sarake ja rivi

Yksittäinen tietokanta koostuu yhdestä tai useammasta taulusta. Taulut muodostuvat sarakkeista, joiden tietotyypit määräävät, missä muodossa sarakkeen tietosisältö on. Rivillä tarkoitetaan yhtä sarakkeiden tietosisällöistä koostuvaa tietokokonaisuutta. (Hovi 2004, 5–6.)

Taulukon 1 mukaisessa ratkaisussa Opiskelija-taulu muodostuu Opiskelijatunnus, Etunimi- ja Sukunimi-sarakkeista. Taulusta on haettavissa yksittäinen rivi, jonka Opiskelijatunnus-sarakkeen arvo on 1, Etunimi-sarakkeen arvo on Ossi ja Sukunimi-sarakkeen arvo on Opiskelija.

Opiskelija		
Opiskelijatunnus	Etunimi	Sukunimi
1	Ossi	Opiskelija
2	Karoliina	Koululainen

Taulukko 1. Opiskelija-taulu. Opiskelijatunnus-, Etunimi- ja Sukunimi-sarakkeet, sekä sarakkeiden tietotyypin mukaisen rivin arvot.

Tietokannan ja sarakkeen nimissä tulee välttää skandinaavisia ym. erikoiskirjaimia, välilyöntejä sekä tiettyjä erikoismerkkejä (välilyönnin tilalla voi käyttää alaviivaa). Myös tietokannoille ja SQL:lle varattuja sanoja, esimerkiksi USER, DATE, ROWID sekä ID, ei saa käyttää taulujen niminä. (Polvinen 1999, 5.)

3.2 Perus- ja viiteavaimet

Jokaisessa taulussa käytetään rivin yksilöivänä tunnisteena perusavainta (primary key). Rivin perusavaimena voi olla joko yksi taulun sarake, tai perusavain voi koostua useamman sarakkeen yhdistelmästä. Koska perusavainarvon tulee olla jokaisella rivillä yksilöllinen, sama perusavainyhdistelmä saa esiintyä samassa taulussa vain yhden kerran. (Hovi 2004, 6.)

Kun tietokannan eri taulut halutaan yhdistää toisiinsa, käytetään viiteavaimia (foreign key). Viiteavainarvot viittaavat toisten taulujen perusavaimiin. (Hovi 2004, 6.) Viiteavainarvo voi toimia samanaikaisesti myös taulun perusavaimena tai sen osana.

3.3 Tietotyypit

Tietokannan luomisen yhteydessä määritetään yleensä, mitä tietotyyppiä kukin sarake sisältää. Tämän opinnäytetyön kannalta keskeiset tietotyypit ovat char, varchar, int, date ja datetime.

Char on kiinteämittainen, varchar taas vaihtelevamittainen merkkijono (Hovi 2004, 105). Esimerkiksi tiettyyn mittaan määritelty käyttäjätunnus kannattaa olla char-tietotyypissä, kun taas esimerkiksi henkilön etu- tai sukunimeä ei pystytä määrittämään etukäteen tietyn pituiseksi, joten varchar-tietotyyppi tulee tällöin kyseeseen. Merkkijonon pituus määritetään taulun luontivaiheissa tietotyypin perään sulkeissa, esimerkiksi char(5) ja varchar(20) (Hovi 2004, 103).

Int on kokonaislukumuodossa oleva tietotyyppi. Kokonaisluvuille on viisi erisuuruista lukujonoa mahdollistavaa tietotyyppiä: tinyint (luvut -128 – 127), smallint (luvut -32768 – 32767), mediumint (luvut -2^{23} – 2^{23}-1), int (luvut -2^{31} – 2^{31}-1) sekä bigint (luvut -2^{63} – 2^{63}-1). (MySQL n.d-a; Mic-

rosoft Developer Network n.d.) Kokonaislukuja on tarkoituksenmukaista käyttää esimerkiksi silloin, kun niitä hyödynnetään laskutoimituksissa. Täten esimerkiksi postinumeroille int ei ole ensisijainen tietotyyppi.

Kokonaislukuja voidaan käyttää myös, jos taulun sarakkeelle halutaan määrittää juokseva, automaattisesti kasvava sarjanumero. Tällöin luontikomentoon lisätään int-tietotyypin yhteyteen auto_increment-määre. (w3schools.com n.d.)

Päivämäärän kuvaamiseen käytetään tietokannoissa joko date- tai datetime-tietotyyppiä. Date-tietotyyppi käsittää päivämäärän, datetime-tietotyyppi kuvaa päivämäärän lisäksi myös kellonajan. (Hovi 2004, 105.)

4 TIETOKANNAN NORMAALIMUOTOISUUS

Tietokannan normalisoinnilla pyritään muun muassa siihen, että tietoa ei tarvitse turhaan toistaa, ja että päivitettäessä tieto yhteen paikkaan sitä ei tarvitse päivittää enää toisaalle. Tietokannan tauluille on määritelty eri normaalimuotoja, joista kolme ensimmäistä ovat yleisimmin käytettyjä, ja käytännössä ne riittävät. (Hovi ym. 2005, 86.)

4.1 Tietokannan ensimmäinen normaalimuoto

Tietokannan ensimmäisen normaalimuodon tavoitteena on toistuvien ryhmien ja moniarvoisten sarakkeiden karsiminen (Hovi ym. 2005, 87–89). Kuvan 1 mukaisessa tauluratkaisussa työntekijän nimi on moniarvoinen sarake, jos ajatellaan etu- ja sukunimeä erillisinä objekteina. Mikäli halutaan etsiä työntekijöitä tietyn etu- tai sukunimen mukaan, haku moniarvoisesta kentästä vaikeutuu (myös esimerkiksi tietoa syötettäessä ei voida aina olla varmoja käytänteestä, kirjataanko etunimi ennen sukunimeä, vai toisinpäin). Myös nykyinen kuukausipalkka ja aiempi palkka historiatietona ovat tietyissä tauluratkaisuissa käytännöllisiä, mutta eteen voi tulla tilanne, jossa halutaan tarkastella historiatietoja pidemmältä ajalta. Myös päivitystilanteessa tehtävä työ moninkertaistuu, kun uuden kuukausipalkan lisäämisen yhteydessä entinen palkka joudutaan siirtämään Aiempi_palkka-sarakkeeseen.

Tyontekija			
Tyontekijatunnus	Nimi	Kuukausipalkka	Aiempi_palkka
1	Seppo Suutari	2500	2300

Kuva 1. Ensimmäistä normaalimuotoa rikkova tauluratkaisu.

Kuvan 2 mukainen tietokantaratkaisu on käytännöllisempi: Tyontekija-taulussa henkilön etu- ja sukunimi on eriytetty erillisiin sarakkeisiin, sekä Tyontekijan_palkka-taulussa päivämäärän mukaan pystytään selaamaan palkkahistoriaa työntekijöittäin taaksepäin.

Tyontekija		
Tyontekijatunnus	Etunimi	Sukunimi
1	Seppo	Suutari

Tyontekijan_palkka		
Tyontekijatunnus	Palkka	Paivays
1	2500	2014-01-01
1	2300	2012-01-01

Kuva 2. Ensimmäisen normaalimuodon mukainen tauluratkaisu.

4.2 Tietokannan toinen normaalimuoto

Tietokannan toinen normaalimuoto liittyy useampisarakkeiseen perusavaimeen: mikään taulun ”tavallisista” sarakkeista ei saa olla riippuvainen yksittäisestä perusavainsarakkeesta, vaan sarakkeiden tulee olla riippuvaisia koko perusavaimesta (Hovi ym. 2005, 90–92). Kuva 3 havainnollistaa tuotetilausta käsittelevää taulua, jossa yksilöllinen arvo saadaan tuote- ja valmistajatunnuksen sekä tilausajan yhdistelmänä (perusavain-sarakkeet alleviivattuna). Kun tarkastellaan perusavaimettomia kenttiä, huomataan että tuotteen nimi on riippuvainen tuotetunnuksesta, ja toisaalta valmistajan nimi on riippuvainen valmistajatunnuksesta. Nämä sarakearvot eivät ole siis riippuvaisia koko perusavainyhdistelmästä, vaan sen tietystä osasta. Tilattu kappalemäärä sen sijaan riippuu aina tietyn ajankohdan mukaisesta tilausajasta yhdistettynä tiettyyn tuotetunnukseen ja tiettyyn valmistajatunnukseen, joten Kappalemaara-sarake on toisen normaalimuodon mukainen.

Tuotetilaus					
Tuotetunnus	Valmistajatunnus	Tilausaika	Tuotteen_nimi	Valmistajan_nimi	Kappalemaara
100	15	2014-08-01 16:00	Kuparijohto, 50 mm	Suomen Kuparitalo	10

Kuva 3. Toista normaalimuotoa rikkova tauluratkaisu.

Jotta taulu saataisiin toiseen normaalimuotoon, lisätään jälleen uusia tauluja (kuva 4). Tuotetilaus-taulun Tuotetunnus-sarake viittaa Tuote-tauluun, ja vastaavasti Valmistajatunnus-sarake viittaa Valmistaja-tauluun. Esimerkin ratkaisu on perusavaimineen suuntaa antava: monijäsenisessä organisaatiossa voi olla tapauksia, joissa kahdessa eri yksikössä tehdään täsmälleen samaan aikaan samojen tuote- ja valmistajatunnusten mukainen tilaus.

Tuote	
Tuotetunnus	Tuotteen_nimi
100	Kuparijohto, 50 mm

Valmistaja	
Valmistajatunnus	Valmistajan_nimi
15	Suomen Kuparitalo

Tuotetilaus			
Tuotetunnus	Valmistajatunnus	Tilausaika	Kappalemaara
100	15	2014-08-01 16:00	10

Kuva 4. Toisen normaalimuodon mukainen tauluratkaisu.

4.3 Tietokannan kolmas normaalimuoto

Tietokannan kolmannen normaalimuodon mukaan jokaisen sarakkeen tulee olla riippuvainen pelkästään perusavaimesta, ei muista sarakkeista (Hovi ym. 2005, 93–94). Kuvan 5 Tyontekija-taulussa on nähtävillä suomalaisissa lomakejärjestelmissä varsin tyypillinen ratkaisu: postinumero ja -toimipaikka esitetään yleensä samassa yhteydessä. Kuitenkin tietokannan normaalimuotoisuutta ajatellen käytäntö ontuu: postitoimipaikka on riippuvainen siitä, mikä sen postinumero on.

Tyontekija					
Tyontekijatunnus	Etunimi	Sukunimi	Katuosoite	Postinumero	Postitoimipaikka
1	Seppo	Suutari	Lehtikuja 2	13100	Hämeenlinna

Kuva 5. Kolmatta normaalimuotoa rikkova tauluratkaisu.

Kuvan 6 mukaisessa tauluratkaisussa postinumero haetaan erillisestä Posti-taulusta. Tällöin Tyontekija-taulussa ei tarvitse tehdä postinumeron yhteyteen kuuluvia muutoksia jokaiselle saman postinumeron sisältäville taulun riville.

Tyontekija				
Tyontekijatunnus	Etunimi	Sukunimi	Katuosoite	Postinumero
1	Seppo	Suutari	Lehtikuja 2	13100

Posti	
Postinumero	Postitoimipaikka
13100	Hämeenlinna

Kuva 6. Kolmannen normaalimuodon mukainen tauluratkaisu.

5 TIETOKANNAN LUONTIKOMENNOT

Palvelimelle voidaan syöttää suoria kirjoitettuja komentoja, jotka luovat halutut tietokannat ja taulut niihin. Komentojen avulla pystytään myös syöttämään tietoa luotuihin tauluihin. Esimerkeissä ja varsinaisessa toteutuksessa tietokannat luotiin palvelimelle phpMyAdmin-hallintapaneelin kautta. Erilliset sisäiset tapahtumat erotetaan toisistaan puolipisteellä.

5.1 Tietokannan ja taulun luontikomennot

Siirtymällä phpMyAdminin etusivulle voidaan valita SQL-editori, jonka kautta pystytään palvelintasolle syöttämään tietokannan luontikomento (Hovi 2004, 142).

```
CREATE DATABASE opinnaytetyo;
```

Samaiseen Create database -komentoon voidaan liittää myös Create table -komento, joka luo halutun taulun kyseiseen tietokantaan (Hovi 2004, 102). Koska järjestelmä ei palvelintasolla tietokannan luomisen jälkeen kuitenkaan tunnista, mihin tietokantaan taulu halutaan lisätä, tulee tietokannan nimi lisätä halutun taulun eteen pisteellä erotettuna.

```
CREATE DATABASE opinnaytetyo;
CREATE TABLE opinnaytetyo.opiskelija;
```

Hallintapaneelin kautta voidaan myös siirtyä suoraan luotuun tietokantaan, jossa niin ikään SQL-editori avaamalla pystytään lisäämään taulu automaattisesti aktiivisena olevaan tietokantaan, ilman tarvetta nimetä tietokanta erikseen.

```
CREATE TABLE opiskelija;
```

Create table -komento ei kuitenkaan yksinään riitä taulun lisäämiseen tietokantaan, vaan komennon yhteydessä tulee lisätä vähintään yksi sarake tauluun.

5.2 Perusavaimen luonti

Taulun ja sarakkeiden lisäämisen yhteydessä pystytään määrittämään myös taulun perusavain (primary key). Koska rivillä tulee vähintään perusavainarvon olla pakollinen, taulun luontivaiheessa perusavaimen tyhjä arvo estetään Not null -määreellä. (Hovi ym. 2005, 107.) Myös muille kuin perusavainkentille voidaan käyttötapaus huomioiden laittaa tyhjien arvojen esto taulun luontivaiheessa.

```
CREATE TABLE opiskelija
(opiskelijatunnus INT AUTO_INCREMENT NOT NULL,
etunimi VARCHAR(30) NOT NULL,
sukunimi VARCHAR(30) NOT NULL,
PRIMARY KEY(opiskelijatunnus));
```

```
CREATE TABLE ohjaaja
(ohjaajatunnus INT AUTO_INCREMENT NOT NULL,
etunimi VARCHAR(30) NOT NULL,
sukunimi VARCHAR(30) NOT NULL,
PRIMARY KEY(ohjaajatunnus));
```

5.3 Viiteavaimen luonti

Viiteavaimet (foreign key) voidaan määrittää tauluun Create table-komennon yhteydessä, mikäli taulu sisältää viiteavainsarakkeen. Luontikomennossa luotavan taulun viiteavainsarake nimetään Foreign key -määritteen perään sulkeissa, minkä perässä References-määritteessä nimetään taulu, jonne viitataan, ja tämän taulun perään sulkeissa kyseisen taulun viitattava sarake.

```
CREATE TABLE opiskelijan_ohjaajat
(opiskelijatunnus INT,
ohjaajatunnus INT,
PRIMARY KEY(opiskelijatunnus, ohjaajatunnus),
FOREIGN KEY(opiskelijatunnus)REFERENCES
opiskelija(opiskelijatunnus),
FOREIGN KEY(ohjaajatunnus) REFERENCES
ohjaaja(ohjaajatunnus));
```

Viiteavainluonnin yhteydessä voidaan määrittää myös se, miten toimitaan tapauksissa, joissa viitattavaan tauluun tehdään perusavainarvon päivittämistä (lähtökohtaisesti taulu on hyvä suunnitella siten, että perusavain säilyy muuttumattomana) tai perusavainarvon sisältävä rivi poistetaan kokonaan taulusta. Määritys tehdään On update- ja On delete -määreillä perusavainmäärityksen perään. (Stack Overflow 2011a).

MySQL mahdollistaa kolme eri tapaa sille, miten perusavaimen muokkaus tai poisto vaikuttaa tauluihin, jossa perusavainarvo toimii viiteavaimena:

- Restrict estää perusavaimen poiston tai muokkaamisen, mikäli sillä on viittauksia muihin tauluihin (tämä on automaattinen oletusarvo MySQL:n viiteavaimissa).
- Cascade vyöryttää uuden perusavainarvon toisiin tauluihin, tai poiston yhteydessä perusavainarvon sisältävät rivit poistuvat kokonaan muista tauluista.
- Set Null asettaa viiteavain arvoksi tyhjän null-arvon, mikäli perusavainarvo poistuu viitattavasta taulusta. Tällöin tulee huomioida, että viiteavaimelle ei luontikomennon yhteydessä määritetä tyhjän arvon estävää Not null -määrettä.

(Stack Overflow. 2011b.)

```
CREATE TABLE opponointitilaisuus
(aika DATETIME NOT NULL,
paikka VARCHAR(20) NOT NULL,
vastuuopettaja INT,
PRIMARY KEY(aika, paikka),
FOREIGN KEY(vastuuopettaja)
REFERENCES ohjaaja(ohjaajatunnus)
ON UPDATE CASCADE ON DELETE SET NULL);
```

5.4 Tiedon syöttäminen tauluun

Kun taulu on luotu tietokantaan, Insert into -komennolla pystytään syöttämään tauluun haluttuja tietoja. Insert into -rivillä määritetään, mihin tietokannan tauluun tietoa ollaan syöttämässä, Values-rivillä määritetään syötettävät arvot. Insert into -riville suositellaan aina lisättävän taulun perään sulkuihin sarakkeet, joihin tietoa ollaan syöttämässä, vaikka esimerkiksi MySQL:llä on mahdollista olla nimeämättä sarakkeita, jos kaikkiin taulun sarakkeisiin ollaan syöttämässä arvot. (Hovi 2004, 122–123.)

```
INSERT INTO opiskelija(etunimi, sukunimi)
VALUES('Ossi', 'Opiskelija');
INSERT INTO opiskelija(etunimi, sukunimi)
VALUES('Karoliina', 'Koululainen');
INSERT INTO ohjaaja(etunimi, sukunimi)
VALUES('Anni', 'Assistentti');
INSERT INTO ohjaaja(etunimi, sukunimi)
VALUES('Ossi', 'Opponentti');
```

Tiedon syöttäminen tauluun voidaan toteuttaa Values-rivillä myös hakemalla syötettävät arvot toisesta taulusta Select-kyselykäskyllä (Stack Overflow 2013).

```
INSERT INTO opiskelijan_ohjaajat(opiskelijatunnus,
ohjaajatunnus)
VALUES((SELECT opiskelijatunnus FROM opiskelija
WHERE etunimi='Ossi' AND sukunimi='Opiskelija'),
(SELECT ohjaajatunnus FROM ohjaaja WHERE etunimi='Anni'
AND sukunimi='Assistentti'));
```

6 TIEDON HAKEMINEN TIETOKANNASTA

Olemassa olevasta tietokannan tauluista pystytään hakemaan tietoa Select-kyselykäskyn avulla. Kyselykäskyssä määritetään, mitkä sarakkeet halutaan näkyviin (Select-osa, kaikki taulun saraketiedot saa näkyviin Select * -merkinnällä) ja mistä taulusta tiedot haetaan (From-osa). Kyselykäskyyn voidaan myös määrittää, jos halutaan hakea tietyn sarakearvon mukainen rivi (Where-osa). (Hovi 2004, 27.)

```
SELECT *
FROM opinnaytetyo.opiskelija
WHERE etunimi='Ossi' AND sukunimi='Opiskelija';
```

Kun haetaan tietoa useampaa taulua käyttäen, taulut liitetään toisiinsa määriteltyjen perus- ja viiteavaimien avulla. Liitos tapahtuu Join-käskyllä, On-määreellä määritetään käytettävien perus- ja viiteavainten nimet taulukohtaisesti. (Hovi 2004, 81.)

```
SELECT opiskelija.etunimi, opiskelija.sukunimi
FROM opinnaytetyo.opiskelija
JOIN opinnaytetyo.opiskelijan_ohjaajat ON
opiskelija.opiskelijatunnus =
opiskelijan_ohjaajat.opiskelijatunnus
JOIN opinnaytetyo.ohjaaja ON
opiskelijan_ohjaajat.ohjaajatunnus = ohjaaja.ohjaajatunnus
```

```
WHERE ohjaaja.etunimi = 'Anni' AND
ohjaaja.sukunimi = 'Assistentti';
```

Mikäli eri taulujen avaimilla on sama sarakenimi, MySQL:llä liitos on mahdollista tehdä On-määreen sijaan myös Using-määreellä (Using-määreen perään sulkuihin sarakkeen nimi), mutta luotettavampi ja alustariippumattomasti varmemmin toimiva tapa on On-määreen käyttäminen.

```
SELECT opiskelija.etunimi, opiskelija.sukunimi
FROM opinnaytetyo.opiskelija JOIN
opinnaytetyo.opiskelijan_ohjaajat USING(opiskelijatunnus)
JOIN opinnaytetyo.ohjaaja USING(ohjaajatunnus)
WHERE ohjaaja.etunimi = 'Anni' AND
ohjaaja.sukunimi = 'Assistentti';
```

7 OLDTIMERTIMER-TIETOKANTARATKAISUT

OldtimerTimer-prototyyppiversion tietokantaratkaisuja hahmoteltiin jo Digital Service Development -opintojaksolla. Alkukesästä 2014 mobiilisovellus- ja internetkäyttöliittymätyöryhmien työsuunnitelma kesän osalta tarkentui, ja myös heidän tarpeet tietokantojen suhteen tulivat tällöin selkeämmiksi.

7.1 Tietokantojen määrittely

OldtimerTimer-prototyyppivaiheen tietokantaratkaisujen toteutus käynnistyi kartoittamalla internetkäyttöliittymä- ja mobiilisovellustyöryhmien tarpeita tietokantojen osalta. Kartoitus tehtiin työryhmien kanssa keskustelemalla ja heidän toiveitaan ylöskirjaamalla. Kirjatuista muistiinpanoista pyrittiin muodostamaan kokonaiskuva molempien työryhmien yhteisistä ja erillisistä tarpeista tietokantojen tietosisällön suhteen.

7.1.1 Mobiilisovelluksen tarve tietokannoille

Vanhuksen tekemä toiminta tapahtuu prototyyppiversiossa yksinomaan mobiililaitteella. Vanhuksen rekisteröityessä oldtimerTimer-sovelluksen käyttäjäksi käyttäjätiedot tallennetaan sekä mobiililaitteen omaan tietokantaan että palvelimen käyttäjätietokantaan. Mikäli palvelimen käyttäjätietokannasta löytyy sama käyttäjätunnus, kuin mitä vanhus on syöttämässä rekisteröitymisen yhteydessä, mobiililaitteeseen tulee kehotus vaihtaa käyttäjätunnus toiseksi.

Vanhuksen mobiililaitteen kalenteriin lisättävät tapahtumat ovat kahden tyyppisiä: säännöllisesti toistuvia rutiineja, tai satunnaisesti tai yksittäistapahtumina tapahtuvia eventtejä. Tapahtuman tiedot luetaan joko palvelimella sijaitsevista xml-tiedostoista tai tietokantatauluista.

Kun tapahtuman suorittamisesta on kulunut tietty aika, vanhuksen mobiililaitteeseen tulee kysely siitä, suorittiko vanhus kyseisen tapahtuman. Myönteisen vastauksen jälkeen käyttäjää pyydetään antamaan palaute tapahtumasta, kielteisen vastauksen jälkeen pyydetään kertomaan syy tapah-

tuman toteutumattomuudelle. Käyttäjälle tulee antaa mahdollisuus myös halutessaan olla vastaamatta jatkokysymyksiin. Vastaukset tallentuvat palvelimen seurantatietokantaan, josta ne ovat myös vanhuksen lähiomaisen nähtävillä.

Mobiilisovelluksen painikkeet ohjelmoidaan siten, että niiden painamisista siirtyy tieto palvelimen analyysitietokantaan. Painikkeiden painamiskerroista voidaan päätellä esimerkiksi, jos sovelluksen käyttö on hidastunut tai sovellus on ollut pitkään käyttämättömänä.

7.1.2 Internetkäyttöliittymän tarve tietokannoille

Internetportaalin kautta ruokalistan ylläpitäjä pystyy syöttämään lomaketäytön kautta ruokalistan, minkä jälkeen ohjelmakoodi muuttaa syötetyn tekstin xml-tiedostoksi. Tyylittelyn kautta xml-tiedostona oleva ruokalista on luettavassa muodossa internetsivuilla ja mobiilisovelluksessa.

Vanhuksen lähiomaisen rekisteröityminen palvelun käyttäjäksi tapahtuu internetportaalin kautta. Rekisteröitymisen yhteydessä lähiomaisen tiedot tallentuvat käyttäjätietokantaan. Samoin kuin vanhuksen rekisteröityessä järjestelmän käyttäjäksi, lähiomaiselle tulee olemassa olevasta käyttäjätunnuksesta ilmoitus ja kehotus vaihtaa käyttäjätunnus toiseksi.

Kirjautuneena käyttäjänä lähiomainen pystyy lisäämään vanhuksen omalle listalleen seurattavaksi. Vanhuksen lisäys tapahtuu lähiomaisen syöttäessä vanhuksen käyttäjätunnuksen ja salasanan (näiden tulee olla siis lähiomaisen tiedossa).

Lähiomaisella on mahdollisuus selata tallennettuja ruokalistoja sekä niiden aikoja ja paikkoja, ja sopivan yhdistelmän löydyttyä lähiomainen pystyy tiedonsiirron myötä lisäämään tapahtuman muistutuksen kera vanhuksen mobiililaitteeseen.

7.2 Tietokantojen suunnittelu

Kun tietokantojen määrittely saatiin pääpiirteissään tehtyä, siirryttiin tietokannan suunnitteluvaiheeseen. Suunnitteluvaiheessa tietokantakaaviot toteutettiin paperiversioina, joita kehitettiin eteenpäin työryhmien kanssa keskustellen.

7.2.1 Käyttäjätietokanta

Rekisteröitymisen yhteydessä palvelimella sijaitsevaan käyttäjätietokantaan tallentuvat vanhuksen ja lähiomaisen tiedot omiin tauluihinsa. Vanhuksen ja lähiomaisen käyttäjätunnus-sarakkeita perus- ja viiteavaimenaan käyttävästä vanhuksen_lahiomainen-taulusta on haettavissa yksittäisen vanhuksen lähiomaiset, kuin myös lähiomaisen itselleen merkitsemät vanhukset.

Hoitaja- ja hoitolaitos-taulujen perusavaimet hakemalla saadaan muodostettua hoitolaitoksen_hoitajat-taulu. Vanhuksen_hoitajat-tauluun saadaan vanhuksen ja hoitajan käyttäjätunnukset hakemalla koottua tiedot yksittäisen vanhuksen hoitajista. Vanhus_hoitolaitosasukas-tauluun pystytään hakemaan perusavaimet vanhus- ja hoitolaitos-tauluista, mikäli vanhus asuu hoitolaitoksessa. Myös vanhuksen itsenäinen kotiasuminen huomioidaan vanhus_kotiasukas-taulussa, jossa vanhuksen käyttäjätunnuksen lisäksi sijaitsevat asunnon osoitetiedot.

Koulun työntekijät ovat henkilöitä, jotka tekevät ilmoituksen siitä, mikäli ruoka loppuu tiettynä ajankohtana tietyllä koululla. Koulu- ja tyontekija-taulujen perusavaimet hakemalla saadaan yksittäisen koulun kaikki työntekijät haettua koulun_tyontekijat-taulusta. Tietty hoitolaitos voi suosia tiettyjä kouluja ruokailupaikkoinaan, joten hoitolaitos_suositut_koulut-tauluun on haettavissa perusavaimet koulu- ja hoitolaitos-tauluista. Myös yksittäinen vanhus voi suosia tiettyjä kouluja esimerkiksi välimatkojen vuoksi, joten vanhus_milla_koululla-tauluun haetaan vanhus- ja koulu-taulujen perusavaimet.

Posti-taulussa perusavaimena toimii postinumero-sarake, paikkakunta-sarake sisältää tiedon postitoimipaikasta. Postinumero-sarake toimii viiteavaimena lahiomainen-, vanhus_kotiasukas-, hoitolaitos-, hoitaja-, koulu- ja tyontekija-tauluissa postinumeroa kysyttäessä.

7.2.2 Ruokatietokanta

Ruokatietokannassa on ruokalajit- ja erityisruokavalio-taulut. Ruokalajit-taulu sisältää peruskuvauksen ruokalajista, erityisruokavalio-taulu sisältää ruokiin liittyviä erityisruokavaliopiirteitä. Ruokalajin_erityisruokavaliot-tauluun saadaan haettua kunkin ruokalajin yksittäiset erityisruokavaliot.

Päivämääräkohtaisesti koulun ruoat kootaan koulun_ruoat_pvm-tauluun, jossa perusavaimina toimivat ruokalajin ja koulun tunnusten lisäksi päivämäärä. Mikäli tiettynä päivänä ruokalaji loppuu tietyltä koululta, koulun työntekijän laittaman ilmoituksen pohjalta ruoka_loppui_koululla-tauluun tallentuvat koulu- ja ruokalajitunnusten lisäksi aika datetime-tietotyypissä. Ruoka_loppui_koululla-taulusta tulee siirtyä hälytys vanhuksen mobiililaitteeseen, mikäli vanhuksella on kyseiseen kouluun tehty samaisena päivänä aikavaraus tietyn ruokalajin osalta.

7.2.3 Seurantatietokanta

Vanhuksen mobiililaitteeseen lisätyt tapahtumatiedot ovat luettavissa seurantatietokannassa sijaitsevista tauluista. Palvelimen tietokannassa sijaitsevat tapahtumatyyppi- sekä tapahtumapaikka-taulut. Tapahtumatyyppi-taulusta on haettavissa tapahtuman kategoriat, tapahtumapaikka-taulusta fyysiset tapahtuman toteutuspaikat osoitetietoineen. Yksittäinen tapahtuma koostetaan tapahtuma-taulussa hakemalla tapahtumalle tyyppi- ja paikka-tunnukset, joiden lisäksi taulussa on myös tapahtuman aloitus- ja lopetus-ajat. Yksittäisen vanhuksen tapahtumat ovat haettavissa vanhuk-

sen_tapahtumat-taulussa, jossa perusavaimena käytetään vanhuksen käyttäjätunnusta sekä tapahtuma-taulun tapahtumatunnusta. Vanhuksen_tapahtumat-taulussa on myös luontipvm-sarake, josta on haettavissa aika, jolloin tapahtuma on lisätty vanhuksen kalenteriin.

Kun tapahtuman suorittamisesta on kulunut tietty aika, vanhuksen mobiililaitteeseen tulee kysely, suorittiko vanhus kyseisen tapahtuman. Myönteisen vastauksen jälkeen käyttäjää pyydetään antamaan palaute tapahtumasta, kielteisen vastauksen jälkeen pyydetään kertomaan syy tapahtuman toteutumattomuudelle. Käyttäjälle tulee antaa mahdollisuus myös halutessaan olla vastaamatta jatkokysymyksiin. Palautteiden vastausvaihtoehdot on etukäteen ohjelmoitu mobiilisovellukseen, ja ne löytyvät myös palvelimen tietokannasta. Vastaukset tallentuvat palvelimen seurantatietokantaan, josta ne ovat myös vanhuksen lähiomaisen nähtävillä.

7.3 Tietokantojen toteutus

Kun tietokantasuunnitelmista oli muodostunut riittävä kokonaiskuva, tietokannat luotiin palvelimelle. Luontikomennot laadittiin etukäteen Notepad-tekstieditorilla, ja ne toteutettiin palvelimelle phpMyAdminin kautta.

Suunnitelmasta poiketen ruokatietokanta jätettiin kokonaan toteuttamatta. Myöskään määrittelyvaiheessa esille noussutta mobiilisovelluksen painikkeiden ohjelmointia analyysitietokannan tiedonkeruuta varten ei toteutettu lainkaan.

7.3.1 Käyttäjätietokanta

Käyttäjätietokannan prototyyppiversiossa päätettiin keskittyä kahteen käyttäjäryhmään, vanhukseen ja tämän lähiomaiseen. Luontikomennossa (liite 1) luotiin vanhus_lahiomainen_hoitaja-tietokanta, jonne siirryttiin luomisen jälkeen hallintapaneelin kautta. Tietokantaan luotiin vanhus- ja lahiomainen-taulut sekä vanhuksen ja lähiomaisen toisiinsa yhdistävä vanhuksen_lahiomainen-taulu.

Yllämainittujen taulujen lisäksi tietokantaan luotiin myös muita tauluja, joille ei kuitenkaan muodostunut prototyyppivaiheen osalta käyttötarvetta: hoitolaitos- ja hoitaja-taulut, hoitolaitoksessa työskentelevät hoitajat näyttävä hoitolaitoksen_hoitajat-taulu sekä vanhuksen hoitajat ilmoittava vanhuksen_hoitajat-taulu. Myös vanhuksen asumispaikkaan kiinnitettiin huomiota joko hoitolaitosasumisen tai itsenäisen kotiasumisen muodossa vanhus_hoitolaitosasukas- ja vanhus_kotiasukas-tauluissa.

Liitteen 1 mukaisesta luontikomennosta huomataan, että tietokantaratkaisu rikkoo tietokannan kolmatta normaalimuotoa: henkilöiden ja paikkojen yhteystiedoissa perusavaimeton sarake postinumero vaikuttaa postitoimipaikkaan. Tietokantaratkaisuja mobiilisovelluksessa ja internetkäyttöliittymissä testatessa ei kuitenkaan nähty järkeväksi luoda erillistä posti-taulua, koska tauluun olisi joko tarvinnut luoda valmis rajapintaratkaisu postinumeroille, tai tiedonhaku olisi muutoin pitänyt tehdä posti-taulun

tarkastuksen tai tiedonsyötön kautta. Prototyyppiversion tietokantaratkaisussa päädyttiin siis nykyiseen ratkaisuun yhteystietojen osalta, mutta erillisestä posti-taulusta tehtiin maininta tulevia versioita ajatellen.

Sen jälkeen, kun vanhuksen lähiomaiselle oli luotu oma taulu, WordPressin omat tietokantaratkaisujen vuoksi alkuperäistä suunnitelmaa jouduttiin muuttamaan: kun lähiomainen kirjautuu sivustolle, tiedot täytyi tallentaa WordPressin omaan tietokantaan. Tähän ei lähdetty etsimään kiertotietä, koska se olisi voinut rikkoa järjestelmää. Siksi alkuperäistä tietokantasuunnitelmaa muutettiin siten, että vanhuksen_lahiomainen- ja lahiomainen-taulu poistettiin tietokannasta, ja luotiin uusi vanhuksen_lahiomainen-taulu hakien lähiomaisen käyttäjätunnus WordPressin omasta tietokantataulusta (liite 2).

7.3.2 Seurantatietokanta

Seuranta-tietokantaan luotiin (liite 3) tapahtumatyyppi-, tapahtumapaikka- ja tapahtuma-taulut. Aluksi näiden taulujen perusavaimina oli erikseen määriteltävä 5 merkin mittainen merkkijono, mutta mobiilisovellustyöryhmän kanssa käydyn keskustelun myötä päädyttiin luomaan taulujen perusavain automaattisesti kasvavalla sarjanumerolla. Vanhuksen_tapahtumat-taulun perus- ja viiteavaimina toimivat tapahtuma-taulun tapahtuma_id-sarake sekä vanhus_lahiomainen_hoitaja-tietokannassa sijaitsevan vanhus-taulun kayttajatunnus_v-sarake.

Tapahtuma-taulussa huomattiin taulun luonnin jälkikäteen muutamia aukkoja toimintalogiikassa, joten taulumäärityksiä muutettiin jälkikäteen. Liitteessä 4 oleva komento tekee tapahtuma-tauluun ennen syöttämistä aikatarkastus-nimisen herättimen, joka tarkastaa, ettei tapahtuma-tauluun syötettävä aloitusaika ole myöhäisempi kuin lopetusaika. Liitteen 5 mukainen komento huolehtii, että taulussa ei voi useampaa saman tapahtumatyypin tapahtumaa samassa tapahtumapaikassa samanaikaisesti järjestettynä. Mobiilisovellustyöryhmän toiveesta tapahtuma-tauluun lisättiin jälkikäteen vielä sisalto-sarake, johon voi määrittää tapahtuman tarkempaa sisältöä, paikkatunnus-sarakkeen perään (liite 6).

Osallistuiko_tapahtumaan_vaihtoehdot-, syyt_tekemattomyyteen- sekä palautekuvaus-taulut sisältävät etukäteen määritellyt vastausvaihtoehdot, joita vanhukselta kysytään mobiilisovelluksessa sen jälkeen, kun tapahtumasta on kulunut tietty aika. Taulujen luontikomentojen jälkeen niihin syötettiin tietokantojen suunnitteluvaiheessa sopivaksi todetut vastausvaihtoehdot.

Osallistuiko_tapahtumaan-tauluun tallentuu kyllä- tai ei-vastaus siten, kuin vanhus vastaa kysymykseen mobiilisovelluksessa. Mikäli käyttäjä osallistui tapahtumaan, osallistui_palaute-tauluun tallentuu käyttäjän vastauksen mukainen palautetieto kyseisestä tapahtumasta. Mikäli käyttäjä ei osallistunut tapahtumaan, ei_osallistunut_syy-tauluun tallentuu käyttäjän vastausvaihtoehdon mukainen tieto.

Liitteen 3 komennosta huomataan, että osallistuiko_tapahtumaan-, ei_osallistunut_syy- sekä osallistui_palaute-tauluihin vanhuskäyttäjän sekä tapahtuman kohdalta viittaus tehtiin käyttäjätietokannan vanhus-tauluun sekä seurantatietokannan tapahtuma-tauluun. Kuitenkin tietokannan loogisen toimivuuden vuoksi vanhuksen ei tule olla mahdollista antaa palautetta tapahtumasta, johon tämä ei ole osallistunut, joten viittausta yritettiin muuttaa jälkikäteen oikeaoppiseksi suorittamalla näiden taulujen vanhuskäyttäjä- ja tapahtumaviittaus tehtäväksi vanhuksen_tapahtumat-taulusta (liite 7). Viiteavaimien muutos aiheutti kuitenkin myöhemmin ongelmia: vanhuksen_tapahtumat-taulusta ei pystynyt poistamaan yksittäistä riviä, mikäli rivillä olevalla vanhuksella tai tapahtumalla oli erillisiä viittauksia alitauluissa. Tämä siis siitäkin huolimatta, että poistettavan rivin mukaista perusavainparia alitauluista ei löytynyt. Viittausmäärityksen muutos ja siitä aiheutunut ongelma tapahtui projektin loppuvaiheessa, eikä siihen ennätetty löytää ratkaisua.

7.4 Tietokantojen hyödyntäminen käytännössä

Vanhuksen rekisteröityminen oldtimerTimer-sovelluksen käyttäjäksi onnistuu mobiililaitteella, ja tiedonsiirron myötä vanhuskäyttäjän tiedot tallentuvat käyttäjätietokannan vanhus-tauluun. Mobiilisovellukseen ei kuitenkaan ehditty suorittaa käyttäjätietokannasta tapahtuvaa reaaliaikaista käyttäjätunnustarkastusta, joka ilmoittaisi vanhukselle jo olemassa olevasta käyttäjätunnuksesta rekisteröitymisen yhteydessä..

Lähiomaisen rekisteröityminen onnistuu internetportaalin kautta, ja lähiomaisen tiedot tallentuvat WordPressin omaan tietokantatauluun. Näiden kahden taulun perusavainsarakkeista pystyy muodostamaan yksilöllisen sarakeyhdistelmän vanhuksen_lahiomainen-tauluun, jonne tiedot tallentuvat lähiomaisen lisätessä vanhuskäyttäjän internetportaalin kautta seurattavakseen.

Seurantatietokannassa oleviin tapahtumatyyppi-, tapahtumapaikka- ja tapahtuma-tauluihin syötettiin kesän 2014 aikana tietoa phpMyAdmin-hallintapaneelin kautta, koska tapahtumien ylläpitäjälle ei määritelty erillistä roolia portaalin kautta. Mobiilisovelluksen kautta yksittäinen vanhus pystyy lisäämään tietyn tapahtuman itselleen, mikä tallentui seurantatietokannan vanhuksen_tapahtumat-tauluun. Myös tapahtuman jälkeen mobiililaitteeseen tulleen tiedustelun vastauksen pohjalta tallentuu tieto osallistuiko_tapahtumaan-tauluun. Samalla periaatteella on toteutettavissa myös tiedon lisäys käyttäjän antaman vastauksen perusteella ei_osallistunut_syy- sekä osallistui_palaute-tauluihin, mutta näiden taulujen osalta tiedonsiirtoa mobiilista palvelintietokantaan ei ehditty toteuttaa.

Vanhuksen lähiomainen pystyy internetportaalissa tarkastelemaan tapahtumakuvauksen mukaan tapahtumia, joita vanhus on suorittanut. Liitteessä 8 Select-kyselykäsky on lisätty php-koodin yhteyteen siten, että tiedot näkyvät internetportaalissa taulukkomuodossa. Select-kyselykäsky hakee tapahtumakuvauksen, tapahtumapaikan, tapahtuman aloitusajan sekä vanhuksen osallistumisen tai osallistumattomuuden mukaiset tiedot. Where-määreellä haetaan niitä vanhuksia, joiden lähiomainen rekisteröitynyt

käyttäjä on, minkä lisäksi kullakin tapahtumakuvauksen mukaisella alasivulla näytetään vain kyseisen tapahtumakuvauksen mukaiset tapahtumat. Order by -määre näyttää taulukkotiedot ensisijaisesti tapahtuman aloitusajan, toissijaisesti tapahtuman tyyppitunnuksen mukaisessa järjestyksessä.

7.5 Tietokannan kehittämiskohteita

OldtimerTimer-prototyyppiversion tultua valmiiksi ja projektin päättyessä pohdittiin tehtyjä tietokantaratkaisuja ja mietittiin, mitä parannettavaa tietokantaratkaisuissa oli myöhempiä versioita ajatellen. Todetut haasteet ja kehittämiskohdat kirjattiin ylös, ja ne ovat tulevien työryhmien luettavissa.

7.5.1 Käyttäjätietokanta

Vanhus_lahiomainen_hoitaja-tietokannassa sijaitseva lahiomainen-taulu jouduttiin korvaamaan WordPressin omalla tietokantataululla järjestelmän sisäisten tietokanta-asetusten vuoksi. Täten suunnitelman mukaiset tiedot vanhuksen lähiomaisesta (lähiomaisen etu- ja sukunimi sekä osoitetiedot) jäivät tietokantaratkaisussa tallentumatta. Mikäli vanhuksen lähiomaisesta halutaan kerätä edellämainittuja tietoja, järjestelmän valmiisiin tietokantaratkaisuihin tulee etsiä vaihtoehtoinen toteutustapa.

Erillistä posti-taulua, jossa perusavaimena toimivan postinumeron lisäksi olisi postitoimipaikka-sarake, ei toteutettu. Prototyyppiversiossa katsottiin parhaaksi yksinkertaisempi ratkaisu, jossa esimerkiksi vanhuksen rekisteröityessä mobiililaitteella järjestelmään ei tarvitse tehdä välissä palvelintietokannan posti-taulusta postinumerotarkastusta. Tietokannan loogisen toimivuuden kannalta myöhemmissä versioissa erillinen posti-taulu on kuitenkin syytä toteuttaa.

Seurantatietokannan luontikomennosta (liite 3) huomataan, että vanhuksen käyttäjätunnusta viiteavaiminaan käyttävien taulujen viiteavainmäärityksissä ei otettu huomioon tilannetta, jossa vanhuksen käyttäjätunnusta päivitetään, tai käyttäjätunnuksen sisältävä rivi poistuu kokonaan. Kyseiseen seikkaan kannattaa kiinnittää huomiota jo tietokantojen suunnitteluvaiheessa, jolloin voidaan keskustella, miten toimitaan perusavainta päivitettäessä tai poistettaessa, jos perusavaimella on viittauksia muihin tauluihin.

Vanhus-taulussa oleva salasana on prototyyppiversion jäljiltä selkokielisessä muodossa. Tulevissa ratkaisuissa tietokantoihin tallennettavat salasanat on syytä salata selvittämättömään muotoon salausalgoritmin avulla. Tietokannan suunnitteluvaiheessa salatun tiedon merkkijono tulee määrittää riittävän pitkäksi, jotta merkkimäärä riittää salatun tiedon tallentamiseen, eikä kirjautumisvaiheessa synny ongelmia.

7.5.2 Seurantatietokanta

Tapahtumat, tapahtumatyypit sekä tapahtumapaikat sijoitettiin prototyyppiversiossa samaan tietokantaan kuin tapahtumien seurantatiedot. Kuitenkin, jos tavoitteena on pitää yksittäisen tietokannan sisältö mahdollisimman loogisena, tapahtumat olisi syytä eriyttää omaan tietokantaansa.

Prototyyppiversiossa onnistuttiin siirtämään mobiililaitteelta tehtyä toimintaa palvelimen tietokantaan, mutta palvelimelta ei ehditty toteuttaa siirtoa vanhuksen mobiililaitteeseen. Täten esimerkiksi vanhuksen lähiomaisen mahdollisuus lisätä sopiva tapahtuma vanhuksen mobiililaitteen kalenteriin muistutuksen kera jäi prototyyppiversiossa toteutumatta.

Prototyyppiversiossa ei tehty lopullista valintaa siitä, tulisiko tapahtumatietojen luvun tapahtua xml-tiedostojen vai tietokannan taulusisällön kautta. Testausvaiheessa mobiilisovellustyöryhmä käytti xml-tiedostoja, joiden pohjalta mobiilisovelluksessa saatiin aikaan kalenteriin lisättäviä tapahtumia. Mikäli päädytään tietokantatauluista tapahtuvaan tapahtumatietojen lukuun, tiedonsiirtoon palvelimelta mobiililaitteeseen tulee kiinnittää huomiota.

Säännöllisesti toistuvat rutiinit sekä satunnaisempina yksittäistapahtumina tapahtuvat eventit eivät soveltuneet samanlaiseen tapahtuma-taulun rakenteeseen. Koska rutiini voi toistua esimerkiksi päivittäin usean vuoden ajan, tietokantaan kertyy liikaa toisteisuutta samankaltaisen tiedon osalta. Täten tulevissa tietokantaratkaisuissa tapahtumatiedon ja -seurannan osalta tulisi miettiä mahdollisimman vähän toisteisuutta sisältäviä ratkaisuja. Myös useamman henkilön yhteen kokoava tapahtuma käsittää kaikille yhteisen aloitus- ja lopetusajan, kun taas tietyissä tapauksissa käyttäjän tulee olla mahdollista määrittää itse haluamansa osallistumisaika. Tällä hetkellä kaikki aikatiedot ovat ennalta määrättyjä.

Liitteen 7 mukainen jälkikäteen tehty viiteavaimien muokkaus esti rivien poiston vanhuksen_tapahtumat-taulusta, mikäli vanhuksella tai tapahtumalla oli yksittäisiä viittauksia osallistuiko_tapahtumaan-taulussa, huolimatta siitä, ettei poistettavan rivin mukaista perusavainsarakeparia viitattavasta taulusta löytynyt. Oikeaoppiseen tauluviittaukseen on hyvä kiinnittää huomiota siis jo taulun luontivaiheessa.

8 TIETOKANTARATKAISUT OSANA PROJEKTINHALLINTAA

OldtimerTimer-prototyyppiversiota tehtiin usean henkilön ja työryhmän voimin. Täten myös tietokantaratkaisujen synnyn osalta eri tahojen organisoitu toiminta oli tärkeää. Ilman työryhmien keskinäistä kommunikointia tietokantaratkaisujen osalta ei olisi tiedetty, mitkä prototyyppiversion tavoitteet ovat ja missä vaiheessa niiden saavuttamisessa ollaan. OldtimerTimer-prototyyppiversion projektinhallintamenetelmäksi valikoitu scrum Digital Service Development -opintojaksolla tehdyn menetelmäkartoituksen myötä.

8.1 Scrum-projektinhallintamenetelmä

Scrum on tuotekehityksessä käytetty, ongelmien ratkaisuun pyrkivä viitekehys. Scrum koostuu tekijöidensä pohjalta erilaisista rooleista sekä muista tapahtumista ja tuotoksista. (Schwaber & Sutherland 2013, 3.)

Yksittäinen scrumin tuoteversioita kehittävä scrumtiimi muodostuu kolmesta eri tahosta: kehitystiimistä, tuoteomistajasta sekä scrummasterista. Kehitystiimi on noin 3–9 henkilön kokoinen, ja sen tehtävänä on itseohjautuen saada aikaan varsinainen tuoteversio. Tuoteomistaja, joka on aina yksittäinen henkilö, vastaa tuotekehityksen hallinnasta, johon kuuluu muun muassa tiedon antaminen siitä, mitä scrumtiimin tulee tehdä. Scrummasterin tehtävänä on muun muassa valvoa projektimaisen työskentelyn etenemistä ja mahdollistaa kehitystiimin esteetön työskentely valmentaen kehitystiimiä samalla itseohjautuvuuteen. (Schwaber & Sutherland 2013, 4–6.)

Scrumin päätoteutus on sprintti, jonka aikana tavoitteena on tuottaa mahdollisimman valmis tuoteversio. Sprintin maksimipituus on yksi kuukausi. Mahdollinen uusi sprintti alkaa heti edellisen perään. (Schwaber & Sutherland 2013, 8.)

Sprintti suunnittellaan scrummasterin johdolla koko scrumtiimin kesken. Suunnitelmassa päätetään, mitä sprintin aikana valmiiksi saatava tuoteversio sisältää, ja miten työskentely tapahtuu sprintin aikana. Scrummasterin johdolla joka päivä pidettävässä päiväpalaverissa kukin kehitystiimin jäsen käy läpi, mitä on tehnyt edellisenä päivänä, olemassa olevan päivän suunnitelman sekä mahdollisia havaittuja esteitä sprintin tavoitteiden saavuttamisessa. Maksimissaan 15 minuuttia kestävän päiväpalaverin jälkeen kehitystiimin jäsenillä on mahdollisuus keskenään käydä läpi akuutteja tilanteita sprintin osalta. (Schwaber & Sutherland 2013, 8–10.)

Sprintin loppuun tehtävässä sprintin katselmoinnissa scrumtiimi ja mahdolliset sidosryhmät käyvät läpi, mitä sprintin aikana tehtiin nostaen samalla esille mahdolliset ongelmat. Sprintin katselmoinnissa tuoteomistajan johdolla todetaan, miten valmis työ on, ja koko ryhmän kesken pohditaan mahdollisia parannuskohtia seuraavien mahdollisten sprinttien suunnittelupalavereita ajatellen. Ennen mahdollista seuraavan sprintin suunnittelupalaveria pidetään sprintin retrospektiivi. Sprintin retrospektiivin aikana scrumtiimi pohtii scrummasterin johdolla, miten työskentely sprintissä sujui eri tekijöiden osalta, mikä työskentelyssä sujui hyvin ja missä on parannettavaa, sekä suunnitellaan kuinka scrumtiimin työskentelytapoja voidaan parantaa. (Schwaber & Sutherland 2013, 10–11.)

8.2 Projektinhallinnan merkitys oldtimerTimer-tietokantaratkaisujen synnyssä

Kesällä 2014 toteutetun scrumin aikana työskennellyt scrumtiimi koostui tuoteomistajaa lukuun ottamatta opiskelijoista, joilla ei ollut aiempaa ammattimaista kokemusta scrumtyöskentelystä. Scrumin aikana pidettiin yhteensä kolme sprinttiä, aluksi kaksi kahden viikon ja lopuksi yksi neljän viikon mittainen. Kaksi ensimmäistä sprinttiä koostuivat tietokantaratkai-

sujen osalta kartoituksesta internetkäyttöliittymä- ja mobiilisovellustyöryhmien kanssa keskustellen ja mahdollisia tietokantamalleja laatien. Viimeisen sprintin aikana tietokantaratkaisut toteutettiin käytännössä, ja tietokantoihin syötettiin ja niistä haettiin tietoa työryhmien tarpeiden mukaan.

Kahden ensimmäisen sprintin aikana käydyt keskustelut kehitystiimien kanssa edesauttoivat viimeisen sprintin aikana tehtyä tietokantojen käytännön toteutusta. Scrumia kuitenkin leimasi tiiviin aikataulun läsnäolo, jolloin ratkaisuja työstettiin osittain eteenpäin jo kartoitusvaiheen aikana: esimerkiksi palvelinratkaisua jouduttiin odottamaan suhteellisen pitkään, jolloin internetkäyttöliittymätyöryhmä työsti tietokantaratkaisuja työryhmän jäsenen omalle palvelimelle ennen, kuin tietokantasuunnitelmia oli saatu valmiiksi. Muutoinkin työnteko ja ratkaisujen testaus oli aloitettu jo ennen scrumin käynnistystä, jolloin tietokantaratkaisujen projektimainen eteneminen kartoituksesta suunnitelmaan ja siitä käytännön toteutukseen ei täysin toteutunut. Mikäli scrum olisi aloitettu aiemmin ennen varsinaisten ratkaisujen toteutusta, tietokantaratkaisut olisi mahdollisesti saatu suunniteltua selkeiksi etukäteen, eikä kartoitusvaiheen aikana olisi tarvinnut tehdä päällekkäistä työtä eri työryhmien itsenäisten tietokantaratkaisujen osalta.

Päiväpalaverit sprinttien aikana edesauttoivat osaltaan tietokantaratkaisujen syntyä, kun kehitystiimin muille jäsenille kerrottiin edellispäivänä tehdyistä saavutuksista, tulevan päivän suunnitelmista ja mahdollisista ongelmista. Työn ulkopuolinen tarkastelu saattoi osaltaan muodostua haasteelliseksi siksi, että päiväpalaverin vetäjänä toimiva scrummaster oli itse osa varsinaista tuoteversiota tekevää kehitystiimiä. Tällöin myös tietokantaratkaisuihin oli hankalaa saada ulkopuolista näkökulmaa, sillä kaikki scrumtiimin jäsenet tuoteomistajaa lukuun ottamatta olivat myös sisällä varsinaisen käytännön työn teossa.

9 TULOSTEN JA PROSESSIN ANALYSOINTI

OldtimerTimer-sovelluksen prototyyppiversion tietokantaratkaisujen laatiminen aloitettiin keskustelemalla internetkäyttöliittymä- ja mobiilisovellustyöryhmien kanssa heidän tarpeistaan tietokantojen ja niiden tietosisällön suhteen. Keskustelujen aikana esille nousseet asiat kirjattiin ylös, minkä pohjalta pystyttiin muodostamaan alustavat tietokantasuunnitelmat. Tietokantasuunnitelmista keskusteltiin uudelleen työryhmien kanssa, ja mahdolliset muutostarpeet kirjattiin ylös ja ne toteutettiin uusiin suunnitelmiin. Kun suunnitelmat oli saatu tarpeeksi pitkälle ja tietokantojen käytännön toteutus oli mahdollista, tietokannat toteutettiin MySQL-komentoina palvelimelle. Toteutusvaiheeseen mennessä osa kartoitus- ja suunnitelmavaiheiden tietokantaratkaisuista jätettiin pois, koska niitä ei katsottu aikataulullisesti olevan mahdollista toteuttaa.

Tiiviissä aikataulussa tehdyn suunnitteluvaiheen myötä käytännön toteutuksessa huomattiin muutamia logiikka-aukkoja, jotka korjattiin jälkikäteen taulujen muokkauskomennoilla. Suurin osa tehdyistä muokkauksista onnistui, yksi muokkaus aiheutti ongelmia tietojen poiston yhteydessä.

Tietokannoissa havaituista ongelmista ja muista mahdollisista kehittämiskohteista laadittiin erillinen raportti oldtimerTimer-sovelluksen myöhempiä versioita varten.

Tietokantaratkaisut saatiin osittain laadittua siten, että ne ovat sekä mobiilisovelluksen että internetkäyttöliittymän hyödynnettävissä. Mobiilisovellus ei kyennyt hyödyntämään palvelimella sijaitsevia tietokantoja, koska tiedonsiirtoa palvelimelta mobiililaitteeseen ei ennätetty prototyyppiversiossa toteuttaa, mobiililaitteelta saatiin kuitenkin siirrettyä tietoa palvelimella sijaitsevaan tietokantaan.

Vanhuskäyttäjän mobiililaitteella suorittamat rekisteröintitiedot tallentuvat palvelimen käyttäjätietokannan vanhus-tauluun. Vanhuksen lähiomainen rekisteröityy järjestelmään internetportaalin kautta, ja lähiomaisen tiedot tallentuvat WordPressin omista tietokanta-asetuksista johtuen WordPressin omaan käyttäjätietokantaan. Vanhuksen lähiomainen pystyy internetportaalin kautta lisäämään tietyn vanhuksen seurattavakseen, kun hän syöttää oikein vanhus-taulussa olevan käyttäjätunnus- ja salasanayhdistelmän. Vanhuksen ja lähiomaisen perusavaimet tallentuvat perus- ja viiteavainpariksi käyttäjätietokannan vanhuksen_lahiomainen-tauluun.

Vanhuksen lisätessä yksittäisen tapahtuman mobiilikalenteriinsa, tapahtuman suorituksen jälkeen tulee tiedustelu, suorittiko vanhus kyseisen tapahtuman vai ei. Vastaus tallentuu palvelimen seurantatietokannassa sijaitsevaan osallistuiko_tapahtumaan-tauluun. Internetportaalissa vanhuksen lähiomaisen nähtävillä on tapahtumatyypeittäin ne tapahtumat, joita vanhuksen kalenterissa on ollut merkittynä. Tapahtumatyypin mukaista painiketta painamalla lähiomainen pääsee sivulle, joka näyttää lähiomaiselle taulukkomuodossa tietokannasta tiedon hakien tapahtumatyypin, tapahtumapaikan, tapahtuman aloitusajan sekä sen, osallistuiko vanhus tapahtumaan vai ei.

Tietokantaratkaisut toteutettiin scrum-projektinhallintamenetelmää hyödyntäen. Scrumin ytimen muodostivat 2–4 viikon mittaiset sprintit. Sprinttien aikana tietokantaratkaisujen etenemistä raportoitiin päivittäin tapahtuvissa päiväpalavereissa, joissa kerrottiin mitä edellisenä työpäivänä oli saatu aikaan, mitä tulevan työpäivän aikana on suunnitelma tehdä sekä mainittiin mahdollisista havaituista ongelmista.

Työryhmän tiivis keskinäinen kommunikointi scrumin aikana edesauttoi tietokantaratkaisujen syntyä. Osittain scrumia leimasi kuitenkin suuri päällekkäisen työn määrä: ennen scrumin sekä tietokantaratkaisujen kartoituksen ja suunnittelun aloittamista mobiilisovellus- ja internetkäyttöliittymätyöryhmät olivat tehneet testiversioina omia tietokantaratkaisuja. Nämä ratkaisut osaltaan leimasivat scrumin aikana tehtyjä tietokantaratkaisuja, ja tietyissä kohdin tietokantaratkaisuista ei saatu muodostettua lopullista kantaa. Esimerkiksi mobiilisovelluksessa hyödynnettiin laitteen sisäisesti tapahtumatietojen luvun osalta xml-tiedostoja, kun taas palvelimella tapahtuman tiedot olivat haettavissa tietokantatauluista.

Scrumia leimasi muutoinkin kauttaaltaan nopea tahti etenemisen suhteen, minkä johdosta tietokantaratkaisuista ei kartoitus- ja suunnitteluvaiheessa saatu työryhmien kesken muodostettua täysin selvää ja yhtenäistä kuvaa, kunnes tietokantoja oli jo ryhdyttävä toteuttamaan. Pidempi scrum ja sinä aikana kunnolla tehdyt tietokantasuunnitelmat työryhmien kesken olisivat mahdollisesti edesauttaneet selkeämpien tietokantaratkaisujen syntymistä.

10 OPINNÄYTETYÖPROSESSI PROJEKTIN AIKANA

Opinnäytetyön aihe tuli esille kevättalvella 2014, kun mahdollisia opinnäytetyöaiheita kesää varten esiteltiin Hämeen ammattikorkeakoululla. Mobiililaitteita ja internetkäyttöliittymiä varten kehitettävät tietokantaratkaisut tarjosivat aiheena monipuolisen alustan tietokantojen suunnitteluun ja toteutukseen käytännössä. Jo aiemmassa vaiheessa suunnitelmana oli tehdä opinnäytetyö tietokantoihin liittyvästä aihealueesta.

OldtimerTimer-prototyyppiversion varsinaiset ratkaisut tulivat mietittäväksi keväällä Digital Service Development -opintojaksolla. Samanaikaisesti pidetyn Tutkiva ja kehittävä osaaja -opintojakson tavoitteena oli opinnäytetyöprosessin käynnistys. Tutkiva ja kehittävä osaaja -opintojaksolla tehtävänpalautuksina olleet aiheenvalintalomake ja opinnäytetyösuunnitelma laadittiin siten, että ne voitaisiin esittää idea- ja suunnitelmaseminaareissa.

Alkukesästä aloitetun scrumin ohessa aloitettiin myös opinnäytetyöprosessi ideaseminaarilla. Ideaseminaariin mennessä oli selvää, että kesän aikana mobiilisovellukseen ja internetkäyttöliittymiin tehtävät tietokantaratkaisut ovat prototyyppiversioita, joista tehtyjen havaintojen pohjalta myöhemmät työryhmät jatkavat järjestelmän työstämistä sitä edelleen kehittäen. Ideaseminaari sisälsi lopullisessa opinnäytetyöprosessissa toteutuneen työkulun: tietokantaratkaisut toteutetaan työryhmien kanssa käytyjen keskustelujen pohjalta. Se, mitkä työryhmien tarkemmat toiveet tietokantojen sisällön suhteen olivat, eivät olleet ideaseminaarin mennessä vielä täysin selviä. Ideaseminaarissa mukana olleet tutkimuskysymykset koskien tiedonsiirtoa palvelintietokannan ja mobiililaitteen välillä sekä kannanottoa siihen, mitä tietoa säilytetään tietokantatauluissa ja mitä xml-tiedostoissa, jätettiin lopullisesta versiosta pois. Niiden tilalle tuli kysymys siitä, miten tietokannan suunnittelu- ja toteutusprosessi tapahtuu projektin aikana.

Tietokantaratkaisujen kartoitusvaiheen aikaan pidetyssä suunnitelmaseminaarissa oli tiedossa, että järjestelmään halutaan vanhuksen mahdollisuus selailla mobiililaitteella tapahtumia ja lisätä ne omaan kalenteriinsa. Myös vanhuksen lähiomaisen mahdollisuutta lisätä internetportaalin kautta tapahtuma vanhuksen mobiililaitteen kalenteriin toivottiin. Suunnitelmassa korostettiin ennen muuta ruokatapahtumien lisäämistä kalenteriin, mikä jäi kuitenkin lopullisesta toteutuksesta pois.

Opinnäytetyön varsinainen kirjoitus alkoi suunnitelmaseminaarin jälkeen. Samanaikaisesti käynnistyi scrumin tavoitteiden saavuttamisen kannalta keskeisin, viimeinen yhden kuukauden mittainen sprintti. Sprintin aikana tietokantaratkaisujen suunnittelu ja toteutus hoidettiin käytännössä, minkä

yhteyteen opinnäytetyön kirjoittaminen tuotti tiettyjä haasteita: sprintin aikana tehty työ oli käytännön toteutusta, toisaalta opinnäytetyön teoriaosa oli vielä kesken. Tehdyistä käytännön ratkaisuista otettiin ylös sekä suunnitelmat että tietokantojen ja taulujen luonti- ja muutoskomennot, joista molempia hyödynnettiin myöhemmin opinnäytetyön käytännön osassa. Opinnäytetyön etenemisen kannalta ratkaisevia olivat pidemmät tauot käytännön työn teosta: mobiilisovellus- ja internetkäyttöliittymätyöryhmien viikon mittainen loma viikkoa ennen sprintin loppua antoi aikaa laatia opinnäytetyön teoriaosaa, ja se saatiinkin viikon aikana etenemään runsaasti. Samoin scrumin päätyttyä opinnäytetyön kirjoittamiselle jäi aikaa ennen sen suunniteltua valmistumisajankohtaa.

Opinnäytetyön kirjoittamiselle oli scrum-suunnitelmassa varattu yksi päivä viikossa, jolloin oli tarkoitus keskittyä pelkästään siihen. Työtä organisoitaessa kyseisinäkin ajankohtina tehtiin kuitenkin oman valinnan seurauksena usein tietokantaratkaisujen laadintaa. Jälkikäteen ajatellen opinnäytetyön kirjoitusprosessi olisi loppua kohden ollut keveämpi, mikäli scrumin aikana olisi yhtenä päivänä viikossa keskitytty opinnäytetyön kirjoittamiseen järjestelmällisellä otteella.

11 YHTEENVETO

Opinnäytetyön tutkimuskysymykset muuttuivat osittain prosessin edetessä. Alun perin tutkimuskysymysten joukossa oli kysymys mobiililaitteen ja palvelintietokannan välisestä tiedonsiirrosta. Kysyttiin myös, mitä tietoa on tarkoitus säilyttää xml-tiedostoissa ja mitä tietokantatauluissa. Scrumin aikana huomattiin kuitenkin, että mobiilisovellustyöryhmän toimenkuvaan kuuluu luontevammin huolehtia tiedonsiirron järjestämisestä mobiililaitteesta palvelimen tietokantaan. Samoin scrumin aikana ja sen lopputuloksessa jäi tekemättä lopullinen päätös siitä, säilytetäänkö esimerkiksi tapahtumatietoja mobiililaitteella luettavissa olevasta xml-tiedostosta, vai hakeeko mobiililaite tapahtumatiedot palvelimen tietokannan tauluista. Opinnäytetyössä päätettiin lopulta keskittyä teknistä puolta enemmän yleiseen tietokantasuunnittelun prosessikuvaukseen sekä SQL-kielen läpikäyntiin tietokantoja luotaessa ja niistä tietoa haettaessa.

Kysymykseen tietokannan suunnittelusta ja toteutuksesta projektin aikana pystyttiin vastaamaan tutkimalla käytännön osan toteutusjärjestystä. Tietokannan kartoitusvaiheessa käydään tietokantojen tarpeista keskustelua niiden henkilöiden kanssa, joiden tekniset toteutukset tarvitsevat niitä. Kartoitusvaiheen pohjalta voidaan luoda tietokantasuunnitelmia, joiden toiminnallisuutta voidaan läpikäydä ilman, että teknistä puolta tarvitsee vielä huomioida. Suunnitelmien pohjalta voidaan luoda valmiit tekniset tietokantaratkaisut, joiden yhteydessä tiedonhaku tietokannoista tulee myös pohdittavaksi.

OldtimerTimer-prototyyppiversion tietokantaratkaisuissa havaituista haasteista voidaan myös päätellä, että suunnittelu- ja toteutusvaiheen yhteyteen on syytä lisätä vielä mahdollisuuksien mukaan tietokantojen tekninen testaus ennen varsinaista käyttöönottoa. Tällöin suunnitteluvaiheessa havait-

tuja loogisia virheitä voidaan korjata ja uusien suunnitelmien pohjalta luoda uudet mahdolliset testiversiot tai suora käyttöönotto.

OldtimerTimer-tietokantaratkaisuja tarkastelemalla vastaaminen siihen, miten pitkälle mobiilisovellus ja internetkäyttöliittymät voivat hyödyntää samoja tietokantaratkaisuja, onnistui riittävän hyvin. Mobiililaitteelta onnistuttiin siirtämään tietoa palvelimen tietokantaan, josta mobiililaitteen lähettämät tiedot ovat internetportaalilla luettavissa. Toisaalta mobiilisovellus ei onnistunut hyödyntämään sellaisia palvelimella sijaitsevia tietokantoja, jotka ovat internetkäyttöliittymän kautta muokattavissa, koska tiedonsiirtoa palvelimelta mobiililaitteeseen ei onnistuttu toteuttamaan. Ei siis saatu näkökulmaa siihen, kuinka hyödynnettävissä mobiililaitteella olisivat internetkäyttöliittymällä muokattavat tietokannat.

Vastattaessa kysymykseen, miten scrum-projektinhallintamenetelmä tukee seurantansa osalta tietokantaratkaisujen syntyä, vastauksen sävy on aina vastaajasta riippuva: toiselle henkilölle tietynlainen projektityöskentelytapa sopii paremmin kuin toinen, ja toisen henkilön ollessa kyseessä tilanne voi olla päinvastainen. Opinnäytetyön tekijän näkökulmasta tarkasteltuna scrum palveli tietokantaratkaisujen syntyä hyvin, sillä scrumin kautta kaikki työryhmät olivat tietoisia toistensa tilanteesta. Kuitenkin scrumin aikana haasteena oli tiivis aikataulu, jonka seurauksena tietokantaratkaisut eivät syntyneet luontevalla tahdilla kunnollisen kartoituksen ja suunnittelun kautta viimeisteltyyn toteutukseen.

Opinnäytetyön käytettiin kehittämisprojektin tutkimusmenetelmää, koska opinnäytetyön keskeisenä aikaansaannoksena oli projektimaisen työskentelyn kautta syntyvä tuote. Opinnäytetyötä tehtäessä ei kuitenkaan ennätetty perehtyä tarkemmin kehittämisprojektin tapaan lähestyä projektia ja sen vaiheita.

Opinnäytetyön teoriaosa painottui kuvaamaan tietokannan peruskäsitteitä ja tietokantojen käsittelyä komentotasolla. Alun perin suunnitelmana olleet tietokantoihin liittyvät teknisemmät käsitteet niitä käsittelevine lukuineen jäivät projektin edetessä ja jäsenten keskinäisiä töitä organisoitaessa pois. Jälkikäteen ajateltuna teoriaosassa olisi voitu mennä syvemmälle tietokantojen kartoitus- ja suunnitteluvaiheen tekemiseen erilaisine dokumentteineen ja lähestymistapoineen, jotta käytännön osuudessa kartoitus- ja suunnitteluvaihe olisi voitu toteuttaa teoriatiedon pohjalta tehokkaammin.

Opinnäytetyön käytännön osuus painottui teoriaosan pohjalta ennen muuta tietokantoihin liittyviin komentoihin. Käytännön osuudessa kuvataan kronologisesti ne työvaiheet, joita tietokannan työstämisprosesseissa on. Kun tietokantojen työstäminen oli aktiivisimmillaan, resurssit keskittyivät työntekoon opinnäytetyön kirjoittamisen sijaan. Täten käytännön osuus laadittiin kokonaisuudessaan sen jälkeen, kun tietokantojen työstäminen oli jo ohi. Työstämisen aikana oli kuitenkin otettu talteen projektin tietokantoihin liittyviä komentoja, joten niitä tutkimalla päästiin käsiksi tehtyihin työvaiheisiin. Niin ikään kartoitus- ja suunnitteluvaiheen suunnitelmadokumentit ja piirretyt tietokantakaaviot auttoivat käytännön osaa kirjoitettaessa.

Opinnäytetyön tekijän tavoitteena oli saada opinnäytetyön kautta käytännön kokemusta tietokantaratkaisujen suunnittelusta ja toteutuksesta. Tavoite saavutettiin, ja lopputuloksena on kokonaisuutena suurelta osin toivotunlaisesti toimiva järjestelmä. Tietokantaratkaisujen työstämisvaiheessa havaitut ongelmat opettivat tekijälleen, mitä tulevissa tietokantaratkaisuissa kannattaa suunnittelu- ja toteutusvaiheen aikana huomioida.

Kesällä 2014 aikaansaatu oldtimerTimer-sovellus on tietokantaratkaisuineen riittävän monipuolinen, jotta niiden ideaa pystyy esittämään muille tahoille. Tietokantaratkaisujen pohjalta tehty raportointi scrumin aikana havaituista haasteista ja parannusehdotuksista edesauttaa tietokantaratkaisujen kehitystä oldtimerTimer-sovelluksen myöhemmissä versioissa.

LÄHTEET

Android-Suomi. 2012. Mikä on Android? Viitattu 20.8.2014. http://blog.androidsuomi.fi/mika-on-android/

Hovi, A. 2004. SQL-opas. Jyväskylä: Docendo Finland Oy.

Hovi, A., Huotari, J., Lahdenmäki, T. 2005. Tietokantojen suunnittelu & indeksointi. Jyväskylä: Docendo Finland Oy.

Leiniö, T. 2012. WordPress - Muutakin kuin blogi. Viitattu 9.8.2014. http://www.sofokus.com/blogi/wordpress-muutakin-kuin-blogi/

Microsoft Developer Network. n.d. Int, bigint, smallint, and tinyint (Transact-SQL). Viitattu 7.8.2014. http://msdn.microsoft.com/en-us/library/ms187745.aspx

MySQL. n.d-a. Integer Types (Exact Value). Viitattu 7.8.2014. http://dev.mysql.com/doc/refman/5.1/en/integer-types.html

MySQL. n.d-b. What is MySQL? Viitattu 14.8.2014. http://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

MySQL-opetusmateriaali. n.d. MySQL. Viitattu 19.8.2014. http://www.ratol.fi/opensource/mysql/

phpMyAdmin. n.d. About phpMyAdmin. Viitattu 10.8.2014. http://www.phpmyadmin.net/home_page/

Polvinen, T. 1999. Tietokannat käytännön työssä. Jyväskylä: Teknolit Oy.

Schwaber, K., Sutherland, J. 2013. The Scrum Guide. Viitattu 21.8.2014. http://scrumwell.files.wordpress.com/2014/03/scrum-guide-2013-fi-v1-1.pdf

Stack Overflow. 2011a. Exact Meaning of MySQL's Foreign Key 'on delete restrict' Clause. Viitattu 10.8.2014. http://stackoverflow.com/questions/8072876/exact-meaning-of-mysqls-foreign-key-on-delete-restrict-clause

Stack Overflow. 2011b. Setting up table relations what do “Cascade”, “Set Null” and “Restrict” do? Viitattu 10.8.2014. http://stackoverflow.com/questions/5383612/setting-up-table-relations-what-do-cascade-set-null-and-restrict-do/5383648#5383648

Stack Overflow. 2013. MySQL INSERT INTO … VALUES and SELECT. Viitattu 7.8.2014. http://stackoverflow.com/questions/15523597/mysql-insert-into-values-and-select/15523692#15523692

vaisalamika. 2011a. Wordpress-julkaisujärjestelmän lataus verkosta. Viitattu 9.8.2014.
https://www.youtube.com/watch?v=dB7Vae0bejc

vaisalamika. 2011b. Tietokannan luominen palvelimelle. Viitattu 9.8.2014.
https://www.youtube.com/watch?v=_3UVcXD02-U

vaisalamika. 2011c. wp-config tiedoston rakentaminen. Viitattu 9.8.2014.
https://www.youtube.com/watch?v=BqiJqMi3EIM

w3schools.com. n.d. SQL AUTO INCREMENT Field. Viitattu 7.8.2014.
http://www.w3schools.com/sql/sql_autoincrement.asp

Liite 1

KÄYTTÄJÄTIETOKANNAN LUONTIKOMENTO

(Komento ajettu läpi osissa, myös muissa liitteissä väliviivat ovat tästä erottimina.)

```
CREATE DATABASE vanhus_lahiomainen_hoitaja;
- - - - - - - - - -
CREATE TABLE vanhus
(kayttajatunnus_v VARCHAR(20) NOT NULL,
salasana VARCHAR(40) NOT NULL,
etunimi VARCHAR(30) NOT NULL,
sukunimi VARCHAR(30) NOT NULL,
PRIMARY KEY(kayttajatunnus_v));

CREATE TABLE lahiomainen
(kayttajatunnus_lo VARCHAR(20) NOT NULL,
salasana VARCHAR(40) NOT NULL,
etunimi VARCHAR(30) NOT NULL,
sukunimi VARCHAR(30) NOT NULL,
katuosoite VARCHAR(40) NOT NULL,
postinro CHAR(5) NOT NULL,
postitoimipaikka VARCHAR(40),
PRIMARY KEY(kayttajatunnus_lo));

CREATE TABLE vanhuksen_lahiomainen
(kayttajatunnus_v VARCHAR(20) NOT NULL,
kayttajatunnus_lo VARCHAR(20) NOT NULL,
PRIMARY KEY(kayttajatunnus_v, kayttajatunnus_lo),
FOREIGN KEY(kayttajatunnus_v) REFERENCES
vanhus(kayttajatunnus_v),
FOREIGN KEY(kayttajatunnus_lo)
REFERENCES lahiomainen(kayttajatunnus_lo));

CREATE TABLE hoitolaitos
(hoitolaitos_id CHAR(5) NOT NULL,
hoitolaitoksen_nimi VARCHAR(50),
katuosoite VARCHAR(40) NOT NULL,
postinro CHAR(5) NOT NULL,
postitoimipaikka VARCHAR(40),
PRIMARY KEY(hoitolaitos_id));

CREATE TABLE vanhus_kotiasukas
(kayttajatunnus_v VARCHAR(20) NOT NULL,
katuosoite VARCHAR(40) NOT NULL,
postinro CHAR(5) NOT NULL,
postitoimipaikka VARCHAR(40),
PRIMARY KEY(kayttajatunnus_v),
FOREIGN KEY(kayttajatunnus_v)
REFERENCES vanhus(kayttajatunnus_v));

CREATE TABLE vanhus_hoitolaitosasukas
(kayttajatunnus_v VARCHAR(20) NOT NULL,
hoitolaitos_id CHAR(5) NOT NULL,
PRIMARY KEY(kayttajatunnus_v, hoitolaitos_id),
FOREIGN KEY(kayttajatunnus_v) REFERENCES
vanhus(kayttajatunnus_v),
FOREIGN KEY(hoitolaitos_id)
REFERENCES hoitolaitos(hoitolaitos_id));
```

```
CREATE TABLE hoitaja
(kayttajatunnus_h VARCHAR(20) NOT NULL,
salasana VARCHAR(40) NOT NULL,
etunimi VARCHAR(30) NOT NULL,
sukunimi VARCHAR(30) NOT NULL,
katuosoite VARCHAR(40) NOT NULL,
postinro CHAR(5) NOT NULL,
postitoimipaikka VARCHAR(40),
PRIMARY KEY(kayttajatunnus_h));

CREATE TABLE hoitolaitoksen_hoitajat
(hoitolaitos_id CHAR(5) NOT NULL,
kayttajatunnus_h VARCHAR(20) NOT NULL,
PRIMARY KEY(hoitolaitos_id, kayttajatunnus_h),
FOREIGN KEY(hoitolaitos_id)
REFERENCES hoitolaitos(hoitolaitos_id),
FOREIGN KEY(kayttajatunnus_h)
REFERENCES hoitaja(kayttajatunnus_h));

CREATE TABLE vanhuksen_hoitajat
(kayttajatunnus_v VARCHAR(20) NOT NULL,
kayttajatunnus_h VARCHAR(20) NOT NULL,
PRIMARY KEY(kayttajatunnus_v, kayttajatunnus_h),
FOREIGN KEY(kayttajatunnus_v)
REFERENCES vanhus(kayttajatunnus_v),
FOREIGN KEY(kayttajatunnus_h)
REFERENCES hoitaja(kayttajatunnus_h));
```

Liite 2

KÄYTTÄJÄTIETOKANNAN MUUTOSKOMENTO

```
DROP TABLE vanhuksen_lahiomainen;
- - - - -
DROP TABLE lahiomainen;
- - - - -
CREATE TABLE vanhuksen_lahiomainen
(kayttajatunnus_v VARCHAR(20) NOT NULL,
ID BIGINT(20) UNSIGNED NOT NULL,
PRIMARY KEY(kayttajatunnus_v,ID),
FOREIGN KEY(kayttajatunnus_v) REFERENCES
vanhus(kayttajatunnus_v),
FOREIGN KEY(ID) REFERENCES rutiinit.wp_users(ID)
);
```

Liite 3

SEURANTATIETOKANNAN LUONTIKOMENTO

```
CREATE DATABASE seuranta;

CREATE TABLE seuranta.tapahtumatyyppi
(tapahtuma_tyyppitunnus INT AUTO_INCREMENT NOT NULL,
tapahtumakuvaus VARCHAR(50) NOT NULL,
PRIMARY KEY(tapahtuma_tyyppitunnus));

CREATE TABLE seuranta.tapahtumapaikka
(paikkatunnus INT AUTO_INCREMENT NOT NULL,
paikan_nimi VARCHAR(50) NOT NULL,
katuosoite VARCHAR(40) NOT NULL,
postinro CHAR(5) NOT NULL,
postitoimipaikka VARCHAR(40) NOT NULL,
PRIMARY KEY(paikkatunnus));

CREATE TABLE seuranta.tapahtuma
(tapahtuma_id INT AUTO_INCREMENT NOT NULL,
tapahtuma_tyyppitunnus INT NOT NULL,
paikkatunnus INT NOT NULL,
aloitusaika DATETIME,
lopetusaika DATETIME,
PRIMARY KEY(tapahtuma_id),
FOREIGN KEY(tapahtuma_tyyppitunnus) REFERENCES
tapahtumatyyppi(tapahtuma_tyyppitunnus),
FOREIGN KEY(paikkatunnus) REFERENCES
tapahtumapaikka(paikkatunnus));

CREATE TABLE seuranta.vanhuksen_tapahtumat
(kayttajatunnus_v VARCHAR(20) NOT NULL,
tapahtuma_id INT NOT NULL,
luontipvm DATETIME,
PRIMARY KEY(kayttajatunnus_v, tapahtuma_id),
FOREIGN KEY(kayttajatunnus_v) REFERENCES
vanhus_lahiomainen_hoitaja.vanhus(kayttajatunnus_v),
FOREIGN KEY(tapahtuma_id) REFERENCES
tapahtuma(tapahtuma_id));

CREATE TABLE seuranta.osallistuiko_tapahtumaan_vaihtoehdot
(vastaustunnus CHAR(2) NOT NULL,
kuvaus VARCHAR(10) NOT NULL,
PRIMARY KEY(vastaustunnus)
);

INSERT INTO seuranta.osallistuiko_tapahtumaan_vaihtoehdot
(vastaustunnus, kuvaus)
VALUES('01', 'Ei');

INSERT INTO seuranta.osallistuiko_tapahtumaan_vaihtoehdot
(vastaustunnus, kuvaus)
VALUES('02', 'Kyllä');

CREATE TABLE seuranta.osallistuiko_tapahtumaan
(kayttajatunnus_v VARCHAR(20) NOT NULL,
tapahtuma_id INT NOT NULL,
osallistuiko CHAR(2),
PRIMARY KEY(kayttajatunnus_v, tapahtuma_id),
FOREIGN KEY(kayttajatunnus_v) REFERENCES
vanhus_lahiomainen_hoitaja.vanhus(kayttajatunnus_v),
```

```
FOREIGN KEY(tapahtuma_id) REFERENCES
tapahtuma(tapahtuma_id),
FOREIGN KEY(osallistuiko) REFERENCES
osallistuiko_tapahtumaan_vaihtoehdot(vastaustunnus));

CREATE TABLE seuranta.syyt_tekemattomyyteen
(tekemattomyystunnus CHAR(2) NOT NULL,
syy VARCHAR(30) NOT NULL,
PRIMARY KEY(tekemattomyystunnus));

INSERT INTO seuranta.syyt_tekemattomyyteen
(tekemattomyystunnus, syy)
VALUES('01','En muistanut');

INSERT INTO seuranta.syyt_tekemattomyyteen
(tekemattomyystunnus, syy)
VALUES('02','En jaksanut');

INSERT INTO seuranta.syyt_tekemattomyyteen
(tekemattomyystunnus, syy)
VALUES('03','Tuli muu este');

INSERT INTO seuranta.syyt_tekemattomyyteen
(tekemattomyystunnus, syy)
VALUES('04','Ei vastausta');

INSERT INTO seuranta.syyt_tekemattomyyteen
(tekemattomyystunnus, syy)
VALUES('05','Tapahtuma peruttiin');

CREATE TABLE seuranta.ei_osallistunut_syy
(kayttajatunnus_v VARCHAR(20) NOT NULL,
tapahtuma_id INT NOT NULL,
tekemattomyystunnus CHAR(2),
vastausaika DATETIME,
PRIMARY KEY(kayttajatunnus_v, tapahtuma_id),
FOREIGN KEY(kayttajatunnus_v) REFERENCES
vanhus_lahiomainen_hoitaja.vanhus(kayttajatunnus_v),
FOREIGN KEY(tapahtuma_id) REFERENCES
tapahtuma(tapahtuma_id),
FOREIGN KEY(tekemattomyystunnus) REFERENCES
syyt_tekemattomyyteen(tekemattomyystunnus));

CREATE TABLE seuranta.palautekuvaus
(palautetunnus CHAR(2) NOT NULL,
palautekuvaus VARCHAR(40) NOT NULL,
PRIMARY KEY(palautetunnus));

INSERT INTO seuranta.palautekuvaus(palautetunnus,
palautekuvaus)
VALUES ('01','Erittäin positiivinen');

INSERT INTO seuranta.palautekuvaus(palautetunnus,
palautekuvaus)
VALUES ('02','Positiivinen');

INSERT INTO seuranta.palautekuvaus(palautetunnus,
palautekuvaus)
VALUES ('03','Neutraali');

INSERT INTO seuranta.palautekuvaus(palautetunnus,
palautekuvaus)
VALUES ('04','Negatiivinen');
```

```
INSERT INTO seuranta.palautekuvaus(palautetunnus,
palautekuvaus)
VALUES ('05','Erittäin negatiivinen');

INSERT INTO seuranta.palautekuvaus(palautetunnus,
palautekuvaus)
VALUES ('06','Ei vastausta');

CREATE TABLE seuranta.osallistui_palaute
(kayttajatunnus_v VARCHAR(20) NOT NULL,
tapahtuma_id INT NOT NULL,
palautetunnus CHAR(2),
vastausaika DATETIME,
PRIMARY KEY(kayttajatunnus_v, tapahtuma_id),
FOREIGN KEY(kayttajatunnus_v) REFERENCES
vanhus_lahiomainen_hoitaja.vanhus(kayttajatunnus_v),
FOREIGN KEY(tapahtuma_id) REFERENCES
tapahtuma(tapahtuma_id),
FOREIGN KEY(palautetunnus) REFERENCES
palautekuvaus(palautetunnus));
```

Liite 4

TAPAHTUMA-TAULUUN AIKATARKASTUKSEN LISÄYS

```
DELIMITER ;;

CREATE TRIGGER aikatarkastus
BEFORE INSERT ON tapahtuma
FOR EACH ROW IF NOT (NEW.aloitusaika < NEW.lopetusaika)
THEN CALL invalid_aika;
END IF;;

CREATE TRIGGER aikatarkastus
BEFORE UPDATE ON tapahtuma
FOR EACH ROW IF NOT (NEW.aloitusaika < NEW.lopetusaika)
THEN CALL invalid_aika;
END IF;;

DELIMITER ;
```

Syöttökomento, jolla testattiin yllä olevan tarkastuksen toimivuus:

```
INSERT INTO tapahtuma(tapahtuma_tyyppitunnus, paikkatunnus,
aloitusaika, lopetusaika)
VALUES(1, 4, '2014-08-19 11:00', '2014-08-19 10:00');
```

Liite 5

TAPAHTUMA-TAULUUN YKSILÖLLISTEN SARAKKEIDEN LISÄYS

```
ALTER TABLE tapahtuma
ADD UNIQUE INDEX tapahtuma_unique (tapahtuma_tyyppitunnus,
paikkatunnus, aloitusaika, lopetusaika);
```

Liite 6

TAPAHTUMA-TAULUUN SISALTO-SARAKKEEN LISÄYS

```
ALTER TABLE seuranta.tapahtuma
ADD COLUMN sisalto VARCHAR(100)
AFTER paikkatunnus;
```

Liite 7

SEURANTATIETOKANNAN TAULUJEN VIITEAVAINTEN MUOKKAUSKOMENTO

(Viiteavaimien tunnukset saadaan näkyville komennolla: SHOW CREATE TABLE [taulun nimi])

```
ALTER TABLE ei_osallistunut_syy
DROP FOREIGN KEY `ei_osallistunut_syy_ibfk_1`;

ALTER TABLE ei_osallistunut_syy
ADD FOREIGN KEY(kayttajatunnus_v) REFERENCES
vanhuksen_tapahtumat(kayttajatunnus_v);

ALTER TABLE ei_osallistunut_syy
DROP FOREIGN KEY `ei_osallistunut_syy_ibfk_2`;

ALTER TABLE ei_osallistunut_syy
ADD FOREIGN KEY(tapahtuma_id) REFERENCES
vanhuksen_tapahtumat(tapahtuma_id);
- - - - - -
ALTER TABLE osallistuiko_tapahtumaan
DROP FOREIGN KEY `osallistuiko_tapahtumaan_ibfk_1`;

ALTER TABLE osallistuiko_tapahtumaan
ADD FOREIGN KEY(kayttajatunnus_v) REFERENCES
vanhuksen_tapahtumat(kayttajatunnus_v);

ALTER TABLE osallistuiko_tapahtumaan
DROP FOREIGN KEY `osallistuiko_tapahtumaan_ibfk_2`;

ALTER TABLE osallistuiko_tapahtumaan
ADD FOREIGN KEY(tapahtuma_id) REFERENCES
vanhuksen_tapahtumat(tapahtuma_id);
- - - - - -
ALTER TABLE osallistui_palaute
DROP FOREIGN KEY `osallistui_palaute_ibfk_1`;

ALTER TABLE osallistui_palaute
ADD FOREIGN KEY(kayttajatunnus_v) REFERENCES vanhuk-
sen_tapahtumat(kayttajatunnus_v);

ALTER TABLE osallistui_palaute
DROP FOREIGN KEY `osallistui_palaute_ibfk_2`;

ALTER TABLE osallistui_palaute
ADD FOREIGN KEY(tapahtuma_id) REFERENCES
vanhuksen_tapahtumat(tapahtuma_id);
```

Liite 8

VANHUKSEN OSALLISTUMISTIETOJEN HAKULAUSE TIETOKANNASTA PHP-KOODIN YHTEYDESSÄ

```
[insert_php]

global $current_user;
get_currentuserinfo();

$con=mysqli_connect("palvelin.ip", "käyttäjä", "salasana");

if(mysqli_connect_errno())
{
echo "Failed to connect to MySQL: " .
mysqli_connect_error();
}

echo '<table border="1">';
echo '<tr>';
echo '<td>';

$result = mysqli_query($con, "SELECT
seuranta.tapahtumatyyppi.tapahtumakuvaus AS Tapahtuma,
seuranta.tapahtumapaikka.paikan_nimi AS Paikka,
seuranta.tapahtuma.aloitusaika AS Aloitusaika,
seuranta.osallistuiko_tapahtumaan_vaihtoehdot.kuvaus AS
Osallistuiko

FROM seuranta.tapahtuma JOIN seuranta.tapahtumatyyppi ON
seuranta.tapahtuma.tapahtuma_tyyppitunnus =
seuranta.tapahtumatyyppi.tapahtuma_tyyppitunnus
JOIN seuranta.tapahtumapaikka ON
seuranta.tapahtuma.paikkatunnus =
seuranta.tapahtumapaikka.paikkatunnus
JOIN seuranta.osallistuiko_tapahtumaan ON
seuranta.tapahtuma.tapahtuma_id =
seuranta.osallistuiko_tapahtumaan.tapahtuma_id
JOIN seuranta.osallistuiko_tapahtumaan_vaihtoehdot ON
seuranta.osallistuiko_tapahtumaan.osallistuiko =
seuranta.osallistuiko_tapahtumaan_vaihtoehdot.vastaustunnus

WHERE seuranta.osallistuiko_tapahtumaan.kayttajatunnus_v IN
(SELECT vanhus_lahiomainen_hoitaja.vanhuksen_lahiomainen.
kayttajatunnus_v
FROM vanhus_lahiomainen_hoitaja.vanhuksen_lahiomainen
WHERE vanhus_lahiomainen_hoitaja.vanhuksen_lahiomainen.ID =
'$current_user->ID')

AND seuranta.tapahtumatyyppi.tapahtumakuvaus='projekti'
ORDER BY seuranta.tapahtuma.aloitusaika,
seuranta.tapahtuma.tapahtuma_tyyppitunnus")

or die(mysqli_errno($con));

$array = array();
while($row = mysqli_fetch_assoc($result))
{
$array[] = $row;
}
```

```
print_r($array);

echo '</td>';
echo '</tr>';
echo '</table>';

mysqli_close($con);

[/insert_php]
```