Jani Hourunranta

SEURANTAJÄRJESTELMÄ DIABETEKSEN OMAHOITOON

Opinnäytetyö
CENTRIA AMMATTIKORKEAKOULU
Mediatekniikan koulutusohjelma
Syyskuu 2012

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Yksikkö Ylivieskan yksikkö	**Aika** Syyskuu 2012	**Tekijä** Jani Hourunranta
Koulutusohjelma Mediatekniikan koulutusohjelma		
Työn nimi SEURANTAJÄRJESTELMÄ DIABETEKSEN OMAHOITOON.		
Työn ohjaaja FM Joni Jämsä		**Sivumäärä** 50 + 3
Työelämäohjaaja FM Joni Jämsä		

Tässä opinnäytetyössä oli tavoitteena löytää uutta informaatioteknologiaa hyödyntäviä apuvälineitä länsimaissa räjähdysmäisesti leviävän diabeteksen omahoidon helpottamiseksi tilanteessa, jossa suurten ikäluokkien eläköityessä ja taloudellisten resurssien kaventuessa henkilökunnan määrää joudutaan johdonmukaisesti supistamaan. Vuosien varrella terveydenhuollon piiriin on rakennettu lukuisia heterogeenisia tietojärjestelmiä, mutta vähemmistönä tuntuvat olevan käyttäjä- ja potilaslähtöiset järjestelmät, jossa potilas itse aktiivisesti osallistuu seurantatiedon keräämiseen ja tilastoimiseen.

Ratkaisuna tutkimusongelmaan kehitettiin hajautettu järjestelmä, jossa käyttäjä syöttää mobiililaitteen avulla terveystietoja TCP/IP –yhteyden yli järjestelmän tietokantaan. Tietojen keräämisessä hyödynnetään uusimpia langattoman informaatiotekniikan välineitä kuten bluetooth-teknologiaa ja GPS-paikannusta. Järjestelmään kuuluu myös web-sovellus, jonka dedikoidut käyttäjäprofiilit tarjoavat sekä potilaalle että hoitajalle/lääkärille mahdollisuuden tarkastella hoitohistoriaa erilaisten koontien muodossa ennalta määrätyltä ajanjaksolta ja perustaa hoitopäätökset pitkän ajan seurantatietoihin. Lisäksi web-sovellus tarjoaa erilaisia palveluita kuten ruokapäiväkirja ja keskustelufoorumi nykyaikaisilla tekniikoilla toteutettuna.

Työssä käytetyistä teknologioista eräs keskeisimmistä oli android-java, jolla toteutettiin mobiililaiteohjelmointi. HTML5, Jquery ja JavaScript olivat keskeisessä roolissa web-sovelluksen rakentamisessa. Tietokanta ja palvelin toteutettiin puhtaasti Open Source –työkaluilla; Linux Ubuntu Server –käyttöjärjestelmän päälle rakennettu palvelin tarjosi MySql –tietokannan, web-palvelimen sekä postipalvelimen.

Tässä opinnäytetyössä asetetut vaatimusmäärittelyt saatiin toteutettua suurelta osin, mistä osoituksena toimiva demo-järjestelmä sekä kattava dokumentaatio opinnäytetyöraportin muodossa. Lähtökohtaisesti aikataulu oli rakennettu väljäksi ja työn kulku noudatti suunnitelmaa johdonmukaisesti.

Asiasanat
Diabetes, Android, Langattomat järjestelmät, Hajautettu järjestelmä, Mobiili, Tietokanta, Linux, MySql, HTML5, Bluetooth, Java

ABSTRACT

CENTRIA UNIVERSITY OF APPLIED SCIENCES	Date October 2012	Author Joni Jämsä
Degree programme Media Technology		
Name of thesis TRACING SYSTEM FOR DIABETES CARE		
Instructor M.Sc. Joni Jämsä		**Pages** 50 + 3
Supervisor M.Sc. Joni Jämsä		

The goal of this thesis was to find new procedures and aids for diabetes care that would make use of information technology. Diabetes is increasing rapidly while economical resources are decreasing and the so called baby boomers are retiring. Through the numerous heterogenous information systems have been built in health however, there are very few systems where the patient could participate in collecting concerning his or her own health care.

As a solution to the research problem a decentralized system was developed. The user feeds health care information to the system through a mobile device. Several modern wireless technologies such as Bluetooth-technology and GPS-positioning were utilized in this project. There is a web application included in the system as well. The web-interface offers a wide range of tools for reviewing long term statistics of health care data which helps making decisions on treatment and drug dosing. In addition, the web application offers many kinds of services such as a food diary and a discussion forum created with modern information technology.

The mobile part of Automated Health Care System was implemented with android-java which was one of the most important technologies used in this thesis project. HTML5, Jquery and JavaScript were in a central role in implementing the web-application. The database, and the server as well as the whole system was implemented using open source tools; the server built on the top of the Linux Ubuntu Server edition offered MySql database, web-server and email server.

The requirement specifications set for the thesis were mostly reached which is proven by the fully functional demo system and informational documentation. The time table was not tight so that the work could be integrated in other studies and the writer's part time job as well as possible. the work followed the timetable consistently.

Key words
Diabetes, Android, Wireless systems, Distributed systems, Mobile, Database, Linux, MySql, HTML5, Bluetooth, Java

ESIPUHE

Idea opinnäytetyöni aiheesta sai alkunsa Joni Jämsän ”Langattomat järjestelmän” –kurssin kurssityöstä. Tuolloin sain aiheeksi pohtia käytännöllisiä kohteita, jossa langattomuutta voisi hyödyntää uudella tavalla. Olen sairastanut diabetestä yli 15 vuotta ja perehtynyt siis läheltä niihin haasteisiin, joita diabeetikko kohtaa jokapäiväisessä elämässä. Olikin siis varsin luonnollinen valinta ryhtyä pohtimaan sovellusta, joka helpottaisia diabeetikon päivärutiineja.

Lisäpontta opinnäytetyöni sai Hannu Puomion ”Hajautetut järjestelmät” –kurssilta, jonka aikana pureuduttiin kovaa vauhtia yleistymässä oleviin hajautettuihin järjestelmiin, pilvipalveluihin ja niiden mahdollistamiin tekniikoihin. Mm. edellä mainituilta kursseilta ammennettuja ideoita ja taitoja pohjana käyttäen sai alkunsa opinnäytetyöni otsikolla ”Automated Health Care System”.

Haluan kiittää työni valmistumisen mahdollistaneita henkilöitäja tahoja. Opinnäytetyöni ohjaaja Joni Jämsä, Opettajani Hannu Puomio, Centrian tutkimusinsinööri Jari Kaarela, Suomen Diabetesliitto, sekä elämänkumppanini ja välittömän palautteen antajani Kati Sandroos ovat omalta osaltaan olleet ensiarvoisen tärkeässä roolissa edesauttamassa opinnäytetyöni valmistumisessa.

TIIVISTELMÄ
ABSTRACT
ESIPUHE
SISÄLLYS

1 JOHDANTO

Diabetes on lisääntynyt viime vuosikymmeninä Suomessa kuten kaikissa länsimaisissa yhteiskunnassa "kansansairauden" tunnuspiirteet täyttäviin mittasuhteisiin. Vuonna 1980 tyypin 1, eli nuoruustyypin diabetekseen sairastuneita oli sataatuhatta lasta kohden 31, luvun ollessa vuonna 2005 jo 64. Nuoruustyypin diabetekseen sairastuu joka vuosi hieman yli 600 lasta. Tyypin 2, eli vanhuustyypin diabetestä sairasti vuoden 2003 lopussa jo 190 000 suomalaista ja määrän ennustetaan nousevan 400 000:aan 30:ssä vuodessa (Ilanne-Parikka et al. 2011, 13.) Länsimaille tyypillisten elämäntapojen seurauksena tyypin 2 diabetestä tavataan yhä nuoremmilla ja sairauden kansantaloudelliset merkitykset ovat merkittävät.

Diabeteslääkkeiden Kelan erityiskorvausjärjestelmän lisäksi kustannuksia aiheuttaa terveydenhuoltojärjestelmään integroitu diabetespotilaiden kontrollijärjestelmä, joka säännöllisine lääkäri- ja hoitajatapaamisineen, laboratoriotutkimuksineen sekä mm. silmänpohjakuvauksineen muodostavat suuren kustannuserän osana terveydenhuoltomenoja. Suurten ikäluokkien eläköityessä, huoltosuhteen heikentyessä ja diabeteksen yleistyessä ollaan tilanteessa, jossa terveydenhuoltojärjestelmä on pakotettu etsimään uusia innovatiivisia ja kustannustehokkaita toimia huolehtia diabeteksen hoidosta – erityisesti omahoidon osalta.

Tässä opinnäytetyössä tutkimusongelmana on löytää teknisiä, loppukäyttäjälähtöisiä apuvälineitä diabeteksen omahoitoon osana kattavaa tilastollista seurantajärjestelmää. Terveydenhuollon piirissä on kehitetty lukuisia järjestelmiä, joissa yhteisenä nimittäjänä on varsinaisen potilaan tuottamien syötteiden jättäminen vähälle huomiolle. Diabeteksen omahoitoon liittyy lukuisia osatekijöitä, jotka järjestelmän salliessa olisi siirrettävissä sopivien automaatioratkaisujen tukemana potilaan huolehdittaviksi ja näin henkilötyövuosien säästö lääkärien ja hoitajien osalta olisi merkittävä.

Ratkaisuksi asetettuun tutkimusongelmaan suunniteltiin erityinen "Automated Health Care System" –järjestelmä, joka tarjoaa monipuolisen aikaan ja paikkaan sitomattoman viitekehyksen tallentaa ja tarkastella niin itse sairauden hoidon kannalta oleellisia kuin

välillisesti hoitotilanteeseen vaikuttavia parametreja. Terveiden elämäntapojen kuten riittävän liikunnan määrän ollessa edellä mainituista parametreista tärkeimpiä, järjestelmään otettiin mukaan syketietoa ja liikunnan määrää sekä laatua seuraava alasovellus. Järjestelmä antaa sekä potilaalle että lääkärille reaaliaikaisen kuvan potilaan hoitohistoriasta ja toimii samalla tietoperustana uusille hoitopäätöksille.

Tämä opinnäytetyö oli tutkimusprojekti, jossa sovelletaan ensisijaisesti olemassaolevia tekniikoita ja laitteita. Avoimella bluetooth –yhteydellä varustettuja sykemittareita on saatavissa markkinoilta varsin laajasti. Sen sijaan bluetooth-verensokerimittarit ovat selkeästi marginaalituote, joten bluetooth –yhteyden asentaminen tavanomaiseen verensokerimittariin oli alustavassa aihepiirirajauksessa varteenotettavin toteuttamisvaihtoehto. Sen toteuttaminen rajattiin opinnäytetyön varsinaisen aiheen ulkopuolelle ja sen toteuttaminen suunniteltiin siirrettäväksi muille toimijoille. Tämän opinnäytetyön puitteissa ei myöskään ollut tarkoituksenmukaista saada aikaan lopullista kaupallista tuotetta, vaan lähinnä demo-versio, josta myöhemmässä vaiheessa voitaisiin jatkokehittää kaupallinen formaatti. Kuitenkin kaikki järjestelmään liittyvät komponentit kuten tietoturva, järjestelmärakenne käytettävyys jne. rakennettiinn alusta alkaen täyttämään kaupallisen tuotteen edellyttämät vaatimusmäärittelyt.

Internetissä ja kirjastoissa on runsaasti saatavilla teknistä kirjallisuutta, joka tarjoaa vankan pohjan kaikkiin hajautetun järjestelmän rakentamiseen tarvittaviin osa-alueisiin. Useimpien tekniikoiden taustalla on vapaaseen lähdekoodiin perustuvat laajat kehittäjäyhteisöt, jotka jakavat tietämystään sovelluskehittäjien käyttöön täysin maksutta ja huolellisesti valmisteltujen esimerkkien kera, mikä auttaa huomattavasti näinkin laaja-alaisen projektin läpiviemistä. Lisäksi omakohtainen, vuosien kokemus diabeteksen sairastamisesta antaa monipuolisen käsityksen siitä, minkälaista järjestelmää ollaan rakentamassa, miten käyttäjät tahtovat järjestelmää käyttää ja toisaalta, minkälainen järjestelmä antaisi lääkäreille ja hoitohenkilökunnalle mahdollisimman suuren hyödyn.

2 DIABETES

Maailmassa on miljoonia diabetestä sairastavia ihmisiä ja määrä kasvaa voimakkaasti vuosi vuodelta (Aarne 2010). Diabeteksen räjähdysmäinen kasvu voidaan liittää länsimaisiin yhteiskuntiin erityisenä elintasosairautena. Vaikka varsinaista syytä ihmisen haiman insuliinintuotannon hidastumiseen tai loppumiseen johtavaan beta-solujen tuhoutumiseen ei vieläkään tiedetä, on diabeteksen hoito kehittynyt valtavasti hoidon alkuvaiheista tähän päivään. Monilla diabetes on ollut jo yli 50 vuotta ja elinikäennusteet nousevat valistuksen ja hoidon kehittymisen myötä. Aktiivinen omahoito, olemassaolevien laitteiden ja menetelmien hyödyntäminen sekä terveet elämäntavat edesauttavat diabeetikkoa elämään laadukkaan elämän sairaudesta huolimatta. Tässä luvussa on tarkoitus perehdyttää lukija siihen, mitä diabetes on sairautena, minkälainen merkitys sillä on henkilökohtaisesti potilaalle ja minkälainen merkitys yhteiskunnalle kansansairautena. Perehdymme myös siihen kuinka diabetestä hoidetaan ja mitä teknisiä apuvälineitä on käytössä.

2.1 Diabetes sairautena

Yleisesti määritellään, että diabetes on energia-aineenvaihdunnan häiriö (Ilanne-Parikka, Rönnemaa & Saha 2011, 9). Aineenvaihdunnan häiriö johtuu haiman beta-solujen osittaisesta tai kokonaisesta tuhoutumisesta, mikä lopettaa tai hidastaa elimistön oman insuliinihormoonin tuotannon. Diabetes voidaan jakaa nuoruustyypin diabetekseen sekä aikuistyypin diabetekseen vakkei sairauksilla nykyisen tutkimustiedon valossa ole juurikaan kontekstia sairastumisikään. Edellisessä lauseessa diabetekseen viitattiin monikkomuodossa perustuen tyypin 1 ja tyypin 2 diabeteksen varsin erilaiseen luonteeseen. Tyypin 1 diabeteksessä usein viruksen aiheuttaman taudin laukeamisen seurauksena haiman beta-solut tuhoutuvat täysin ja lakkaavat tuottamasta insuliinihormonia. Tyypin 2 diabetes on tyypillinen elintasosairaus ja siinä sairaus linkittyy kiinteästi ylipainoon, huonoihin elämäntapoihin sekä ns. metaboliseen oireyhtymään. Tyypin 2 diabeteksessä insuliinintuotanto ei lakkaa täysin, vaan hidastuu ja näin ollen hoitomuodotkin ovat tyypin 1 diabetestä moninaisemmat. (Suomen diabetesliitto, 1998.)

Terveellä ihmisellä verensokeripitoisuus on elimistön taholta varsin tarkoin säädeltyä. Sokeripitoisuuden noustessa aterian yhteydessä elimistön oma insuliinituotanto pitää huolen siitä, että sokeritasapaino pysyy välillä 4-9 mmol/l (Ilanne-Parikka ym. 2011, 18).

Diabetikolla näin ei tapahdu, vaan potilaan täytyy mitata verensokeritaso manuaalisesti ja mittaustulokseen sekä henkilökohtaiseen hoito-ohjelmaan nojaten päätellä elimistönsä kulloinkin tarvitseman insuliiniannoksen. Insuliiniannos annetaan joko injektiona tai tablettina. Sairauden hallittaavuuden ja hoidon kannalta varsin haasteellista on se, että verensokeritasapainoon vaikuttavat monet muutkin tekijät kuin nautittu hiilihydraattimäärä; liikunta, vireys, vuorokauden aika, stressi ja monet muut seikat – jokainen potilas on erilainen. Erityisen haastava tilanne on silloin kun nuori lapsi sairastuu diabetekseen. Lapsi ei välttämättä kykene tunnistamaan matalia verensokeritiloja (hypoglykemia) eikä ymmärrä hoidon vaatimaa tuntemusta omasta kehostaan, jolloin vanhempien rooli sekä erilaisten teknisten apuvälineiden rooli korostuu sairauden hoidossa.

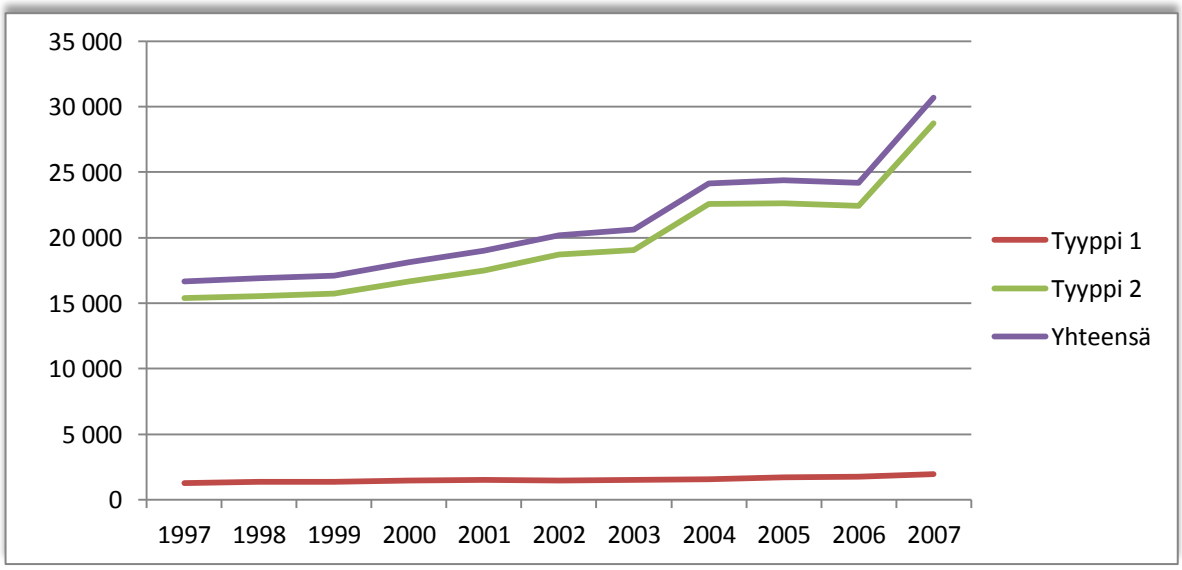
Diabetestä sairastavan ei tarvitse luopua mistään jokapäiväisestä toiminnasta. Päinvastoin, kaikille terveillekin ihmisille olisi suositeltavaa noudattaa diabetikon säännöllistä elämänrytmiä, terveellistä ruokavaliota ja kykyä kuunnellä oman elimistönsä toimintaa. Nykyiset lääkkeet, tietämys sekä tekniset innovaatiot auttavat taudin hoidossa ja luovat edellytykset täysipainoiselle elämälle.

2.2 Kansanterveydellinen ja kansantaloudellinen merkitys

Maailmassa arvioidaan olevan 300 miljoonaa diabetikkoa, joista 80-90% sairastaa tyypin 2 diabetestä (Sund & Koski 2009). Suomessa diabetes on todettu 300 000:lla ihmisellä ja arvioiden mukaan diagnosoitujen ja piilevien tapausten yhteismäärä on jopa 500 000, mikä kattaa yli yhdeksän prosenttia koko väestöstä. Itä-Suomessa diabetestä sairastaa huomattavasti useampi kuin Länsi-Suomessa. Myös sukupuolten välillä esiintyvyys on epätasapainossa: Miehistä 16% ja naisista 11% sairastaa diabetestä. Merkillepantavaa on myös diabeteksen suuri esiintyvyys lapsissa, joka on Suomessa 64/100 000 alle 15 vuotiasta lasta kohden (vuosi 2009). (Ilanne-Parikka et al. 2011, 13.) Riskitekijät ja länsimaisen elämäntyylin yhteys tyypin 2 diabeteksen puhkeamiseen tunnetaan ja jokainen voi omilla valinnoillaan vaikuttaa altistumiseensa taudille. Sen sijaan Tyypin 1 diabeteksen lisääntymiselle ei tutkimuksista huolimatta ole löydetty selitystä – ainoastaan sairauden hoitamista helpottavia ratkaisuja.

Diabeteksen hoidon kustannukset yhteiskunnalle on laskettu ja tilastoitu. Kustannuksia kertyy mm. lääkäri- ja laboratoriokäynneistä, hoitotarvikejakelusta, KELA:n

lääkekorvauksista sekä lukuisista diabeteksen aiheuttamista liitännäissairauksista, kuten hiusverisuonivauriot (silmänpohjat, hermot ja munuaiset). Diabeteslääkkeet, hoitotarvikkeet sekä lääkärikäyntien kustannukset ovat erittäin kalliita. Vuonna 2007 diabetes maksoi yhteiskunnalle 1350 miljoonaa euroa (Manneri 2009). Kustannukset ovat siis valtavat ja kaikki tekniset innovaatiot, jotka vähentävät esimerkiksi hoitohenkilökunnan työtä ja sitä kautta kustannuksia nousevat tulevaisuudessa yhä merkittävämpään rooliin. Kuviossa 1. on havainnollistettu uusien todettujen diabetestapausten määrää vuosien 1997-2007 välisenä aikana.



KUVIO 1. Uudet todetut diabetestapaukset vuosittain. (mukaillen Sund & Koski 2009, 14)

2.3 Diabeteksen omahoito

Diabeteksen omahoidolla on ratkaiseva rooli potilaan terveydentilan kehittymisessä sekä lisä- että liitännäissairauksien ehkäisemisessä. On arvioitu, että tyypin 1 diabeetikoilla lisäsairaudet kuusinkertaistavat ja tyypin 2 diabeetikoilla nelinkertaistavat sairauden vuosittaiset yhteiskunnalle aiheutuvat kustannukset. (Manneri 2009).

Diabeteksen omahoito perustuu täsmälliseen ruokavalioon, liikuntaan sekä näihin sovellettuun lääkeannosteluun. Olennainen osa omahoitoa liittyy siten kirjanpitoon, pitkän ajan trendien suhteuttamiseen oman kehon kuuntelemiseen. Käytettävissä olevat työkalut tehokkaan omahoidon ylläpitoon ovat kuitenkin olleet teknisessä mielessä varsin kehittymättömät. Omahoito on perustunut henkilökohtaisiin verensokerimittareihin sekä yksinkertaisiin muistiinpanovälineisiin. Joitain USB:n kautta tapahtuvaan mittarin muistin

purkuun perustuvia tietokonesovelluksia on markkinoilla, mutta tekniikan suomia mahdollisuuksia ei missään muodossa ole hyödynnetty vielä täyspainoisesti. Vaikka koko omahoito perustuu tiukkaan itsekuriin ja omaehtoiseen haluun pitää huolta terveydestään, me jokainen tarvitsemme pitkällä aikavälillä virikkeitä ja uusia näkökulmia päivittäisiin askareisiimme. Siihen suhteutettuna pienetkin ärsykkeet kuten uudet tekniset innovaatiot diabeetikon omahoidossa ovat enemmän kuin tervetulleita.

3 AUTOMATED HEALTH CARE SYSTEM (AHCS)

Tässä opinnäytetyössä tuotetulle järjestelmälle annettiin nimi ”Automated Health Care System” (AHCS). AHCS pyrkii omalta osaltaan vastaamaan diabeteksen omahoitoon liittyviin haasteisiin. Myös diabeteksen omahoito on astumassa uudelle vuosituhannelle ja se tarkoittaa mm. mobiililaitteiden hyödynnettävyyttä osana hoitoa. AHCS:n päätavoite on tallentaa terveydentilaan liittyviä parametreja kuten mitattu verensokeriarvo, syketieto sekä tiedot kyseisiin parametreihin vaikuttaneista olosuhteista älypuhelimeen, jonka jälkeen puhelin välittää tiedon edelleen TCP/IP –yhteyttä käyttäen php-skriptin kautta tietokantaan. Tämän jälkeen niin käyttäjä kuin hoitajakin voivat tarkastella potilaan profiilitietoja erilaisten graafien muodossa web-sovellusta apuna käyttäen. Web-sovellukseen on lisätty myös erilaisia käyttäjän omahoitoa helpottavia ominaisuuksia kuten ruokapäiväkirja ja keskustelufoorumi, jossa käyttäjä voi hakea vertaistukea erilaisissa diabetekseen liittyvissä arkipäivän kysymyksissä.

3.1 Järjestelmä

Hajautettujen järjestelmien ja erilaisten pilvipalveluiden merkitys kasvaa jatkuvasti mobiiliteknologian kehittyessä. Hajautettu järjestelmä koostuu tietoverkkoon liitetyistä toisistaan riippumattomista tietokoneista, jotka näyttäytyvät käyttäjälle yhtenä yhtenäisenä järjestelmänä (Systä 2012). AHCS on tyyppiesimerkki asiakas-palvelin (client-server) –ohjelmistosta, jossa asiakkaan puhelinlaitteessa tai web-selaimella sijaitseva ohjelma kommunikoi palvelimen kanssa. Siinä ei siis ole kyse varsinaisesta hajautetusta ohjelmistoarkkitehtuurista, jossa yksittäinen ohjelma olisi esimerkiksi kapasiteettioptimointisyistä hajautettu, vaan järjestelmän eri toiminnallisuuksien hajauttamisesta itsenäisiksi moduuleiksi. Palvelimelle on määritelty PHP-skriptin muodossa rajapinnat niin tiedon syöttämistä kuin lähettämistäkin varten ja niitä käyttäen se palvelee sekä web-sovellusta että puhelinaplikaatiota.

Järjestelmän tavoite on siis helpottaa ja motivoida käyttäjää uusilla keinoilla huolehtimaan paremmin omahoidosta ja näin vähentää terveydenhuoltojärjestelmän vastuuta ja kustannuksia hoitoketjussa. Järjestelmän ansiosta tilastointi ja hoitohistoria muodostuu aukottomammaksi, poikkeavuuksien ja hoidon ongelmakohtien havaitseminen helpottuu sekä erilaisille analyyseille ja hoitopäätöksille löytyy vankempi ja nopeampi perusta. Tavoitteena on myöskin kasvatus ja valistus. Erityisen ongelmallinen diabetes on

lapsiperheille, joissa pieni lapsi sairastuu diabetekseen. Lapsi ei välttämättä ymmärrä sairauden luonnetta, lääkitystä tai liikunnan ja ravinnon merkitystä. Web-sovelluksen ravintopäiväkirja opastaa seikkaperäisesti hiilihydraattien laskennassa ja puhelinsovellus lähettää automaattisesti mittaustiedon vaikkapa vanhepien puhelimeen. Näin etäällä oleva vanhempi saa realiaikaisen tiedon lapsensa verensokeritasosta. Puhelin on väline, joka on nykyään lähes poikkeuksetta mukana myös perheen nuorimmilla, joten se on luonteva ja nykyaikainen ratkaisu myös kirjanpito- ja seurantatehtäviin. Helppous ja vaivattomuus ovat peruslähtökohta toimivalle järjestelmälle.

Järjestelmän hajauttaminen TCP/IP –yhteyden ylitse tuo aina mukanaan tietoturvariskejä, jotka täytyy ottaa monin tavoin huomioon tietoturvallisessa ohjelmoinnissa. Tietoturvallinen ohjelmointi osana mobiiliohjelmointia ja hajautettuja järjestelmiä käsitellään tarkemmin luvussa 7 ”Tietoturva”.

3.2 Järjestelmän komponentit

AHCS on hajautettu itsenäisiin komponentteihin, jotka yhdessä muodostavat toimivan ja tietoturvallisen järjestelmän. Järjestelmän ydin on MySQL –tietokanta, joka toimii autentikointitietokantana sekä datan tallennustietokantana. Kuten itse tietokanta, myös varsinainen tietokantayhteys on hoidettu vapaan lähdekoodin työkaluilla. PHP-skripti avaa yhteyden tietokantaan sekä huolehtii tietokantakyselyistä niin tallennuksen kuin avaamisenkin näkökulmista. Puhelinsovellus on yhteydessä web-rajapintaan kuljettamalla avain-arvo (name-value) –parin GET-pyynnössä. Web-pyynnön tyypistä riippuen rajapinta palauttaa tietokantakyselyn perusteella määrätyn arvon, jonka perusteella puhelinsovellus päättää jatkotoimista.

Puhelinsovelluksen tehtävä on luonteeltaan välikerros –tyyppinen. Se autentikoi käyttäjän ja välittää verensokerimittaustiedon verensokerimittarilta web-rajapintaan. Optiona puhelinsovelluksen kehittämisen näkökulmasta on tuoda trenditieto graafisesti myös puhelinohjelmasta käsin käytettäväksi, mutta se on rajattu tämän opinnäytetyön ulkopuolelle. Eräs puhelinsovelluksen tärkeä rooli on myös realiaikaisen datan lähettäminen toiseen matkapuhelimeen tai sähköpostiin. Tämän opinnäytetyön rajauksessa mukana on mittaustuloksen välittäminen toiseen matkapuhelimeen tekstiviestin välityksellä.

Viimeinen järjestelmän tärkeimmistä komponenteista on web-sovellus. Siinä käyttäjä voi luoda oman profiilin, tarkastella sisäänkirjauduttuaan omia historiatietoja, tallentaa merkintöjä ruokapäiväkirjaan, keskustella keskustelufoorumilla muiden diabeetikkojen sekä hoitohenkilökunnan kanssa, nähdä viimeisimmän uutiset diabetekseen liittyen sekä olla sähköpostitse yhteydessä järjestelmän pääkäyttäjään tai vastausvuorossa olevaan hoitajaan. Kaikki nämä palvelut vähentävät varsinaiseen akuuttihoitoon kohdistuvaa kuormaa ja vähentävän osaltaan terveydenhuollon kustannuksia.

3.3 Yleiskatsaus käytettyihin tekniikoihin

Järjestelmän rakentamisessa on käytetty varsin laajaa valikoimaa avoimeen lähdekoodin perustuvia tekniikoita. Valittavana olisi ollut myös Microsoftin tarjoama .NET –ympäristö, joka olisi tarjonnut suhteellisen valmiin arkkitehtuurin niin tietokannan, web-sovelluksen sekä web-rajapinnan luontiin sekä WP7 (Windows Phone 7) sovelluksen luontiin. Microsoftin peruslähtökohta on kuitenkin bisnesvetoinen, joten AHCS:n rakentaminen puhtaasti avoimen lähdekoodin pohjalle takaa sille helpomman tien tulevaisuudessa myös kaupalliseksi tuotteeksi.

Web-sovellus rakennettiin käyttäen HTML5 –tekniikkaa. Edeltäjiinsä verrattuna siinä on monia uusia ominaisuuksia kuten video-, canvas- ja audioelementit. Eräs suuri uudistus koskee myös sivun rakennetta määritteleviä elementtejä. AHCS:n sivustorakenne nojaa vahvasti box-modeling –periaatteeseen, jossa uudet section- ja article –rakenne-elementit näyttelevät pääroolia. Web-sovellukseen kuten itseasiassa HTML5:kin vahvassa symbioosissa on avoimen lähdekoodin jQuery JavaScript –kirjasto. Se on monipuolinen ja nopea tapa luoda animointia sekä joustavaa toiminnallisuutta. Web-tekniikoista jQuery ja HTML5 ovat kovaa vauhtia syrjäyttämässä Adobe Flashin. Perinteisempää tekniikka web-sovelluksessa edustaa PHP-pohjainen keskustelufoorumi, jossa HTML5:n canvas –elementin sisään on rakennettu toiminnallisuudeltaan itsenäinen kokonaisuus. Maininnan ansaitsee myös AJAX, joka toimii linkkinä JavaScriptin sekä web-rajapinnan (PHP-skripti) välillä. Molemmilla laidoilla tieto muokataan Json-array –muotoon, ja välitetään GET-pyyntönä rajapinnan ja kutsuvan ohjelman välillä.

Avoimen lähdekoodin lisäksi myös bluetooth-tuki oli yksi ensisijaisista prioriteeteista tehtäessä valintaa android- ja windows phone 7 –käyttöjärjestelmien välillä. Windows phone 7 sen enempää kuin windows phone 7.5 ei sisällä bluetooth-tukea, joten jo siitäkin

syystä android-käyttöjärjestelmä oli luonnollinen valinta AHCS-puhelinsovelluksen alustaksi. Tässä projektissa ohjelmointikielenä päädyttiin käyttämään android-kehittäjäyhteisössä laajalti käytössä olevaa java- kieltä, vaikka saatavilla olisi ollut myös C++ ja C# -kielisten kirjastojen kääntämisen mahdollistamia lisäosia. Tästä hyvänä esimerkkinä ”mono-android” –projekti (monodevelop.com 2013).

4 MOBIILISOVELLUS

4.1 Mobiilisovellus osana järjestelmää

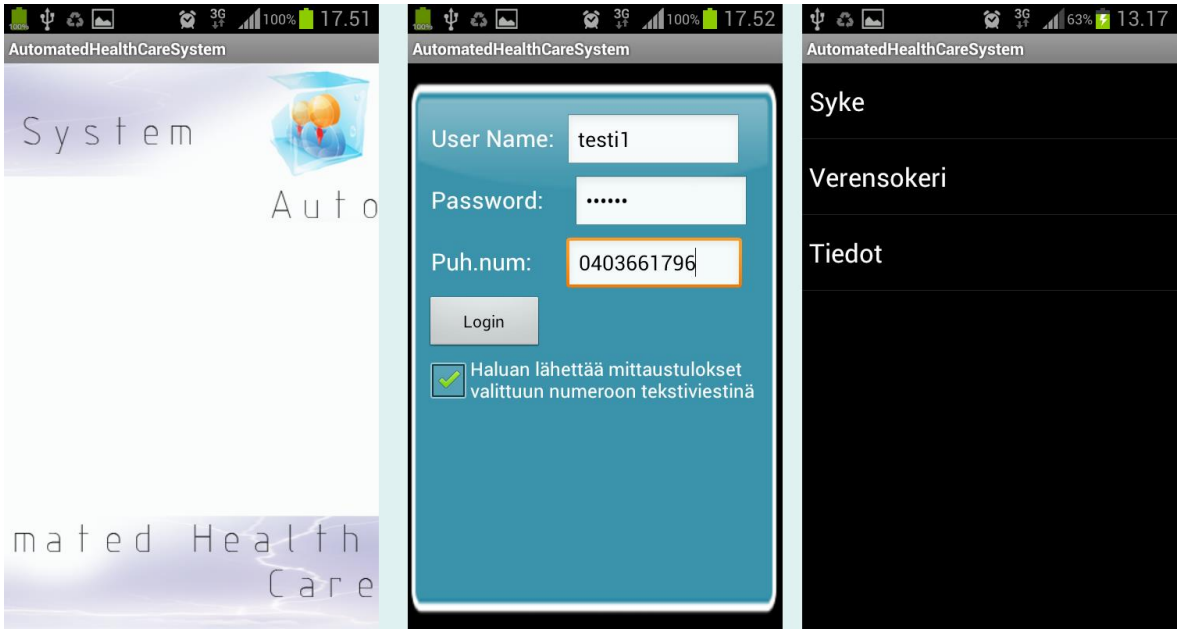
Nykyaikaiset älypuhelimet ovat toiminnallisuuksiltaan ja suorituskyvyltään kuin mitkä tahansa kotitietokoneet. Mielenkiitoisen lisän tuovat kuitenkin mobiilius eli vapaa liikuteltavuus sekä esimerkiksi integroidut GPS- ja bluetooth –ominaisuudet. Verensokerimittauksessa verensokeritason lisäksi on aina huomioitava minkälaisissa olosuhteissa kyseinen mittaus on tehty. Liikunnan määrä ja laatu, ruokailun määrä ja laatu sekä muut ympäristömuuttujat ovat merkittävässä roolissa arvioitaessa yksittäistä mittausta osana pitkän ajan seurantatietoa. Niinpä jokaiseen mittaustietoon liitetään automaattisesti paikkatieto – mitattiinko arvo urheilun yhteydessä urheiluhallilla, lenkkipolulla. Oltiinko kenties juhlimassa, ulkona syömässä vai ehkäpä pitkän työpäivän uuvuttamana ja huonosti syöneenä työpaikalla. Lisäksi käyttäjä voi syöttää lisätietokenttään lyhyen selvityksen mittaustietoon vaikuttaneista muista tekijöistä kuten pistetyn insuliinin määristä jne.

Mobiilisovellus sisältää myös liikuntapäiväkirjan mahdollistavan osion, jossa käyttäjä voi seurata reaaliaikaisesti kartalta liikuntasuoritustaan, mitata keskisykkeen, huippusykkeen, kuljetun matkan ja nopeuden ja tallentaa harjoitusdatan tietokantaan myöhempää tarkastelua varten.

4.2 Mobiilisovelluksen toiminnot

Sovelluksen käynnistyessä käyttäjälle avautuu aloitusruutu, joka noudattelee graafiselta ilmeeltään järjestelmän yleistä teemaa: vaaleat sävyt sekä tuttu liikemerkki matkapuhelimen resoluutioon sovitettuna. Aloitusruutu ja sen taustalla soiva musiikki on toteutettu erillisellä säikeellä, mikä mahdollistaa luokan tuhoamisen ja estää mahdollisuuden palata ruutuun puhelimen palautuspainikkeella. Toinen ruutu on sisäänkirjautuminen, ja jatkaminen edellyttää web-sovelluksessa luotua käyttäjätiliä. Kirjautuessaan käyttäjä voi valita antaako hän luvan käyttää puhelimen tekstiviestitoimintoa mittaustulosten lähettämiseen toiseen puhelimeen. Ominaisuus on suunniteltu palvelemaan erityisesti lapsiperheitä, joissa vanhempien on tärkeä saada reaaliaikaista tietoa diabetestä sairastavan lapsensa verensokeritasosta. Samalla käyttäjätunnus, salasana ja puhelinnumero tallennetaan erillisen getter-setter –luokan avulla muistiin, jotta tietoja ei tarvitsisi jokaisen käyttökerran yhteydessä syöttää uudelleen.

AHCS:ssä menuvalikko on toteutettu arrayAdapter –tyyppisellä ratkaisulla, jossa sovelluksen osat voidaan käynnistää pyyhkäisemällä listasta haluttua alasovelluksen nimeä. Uuden aktiviteetin (Activity) aloittaminen määritellään android-manifest –tiedostossa. Tiedosto sisältää jokaisen aktiviteetin - voidaan pelkistetysti kuvata ruuduksi tai luokaksi - tunnisteen, jonka perusteella haluttua alasovellusta voidaan kutsua. Kuviossa 2. on esitelty mobiilisovelluksen kolme ensimmäistä aktiviteettia vasemmalta oikealle: aloitusruutu, sisäänkirjautuminen ja menuvalikko.

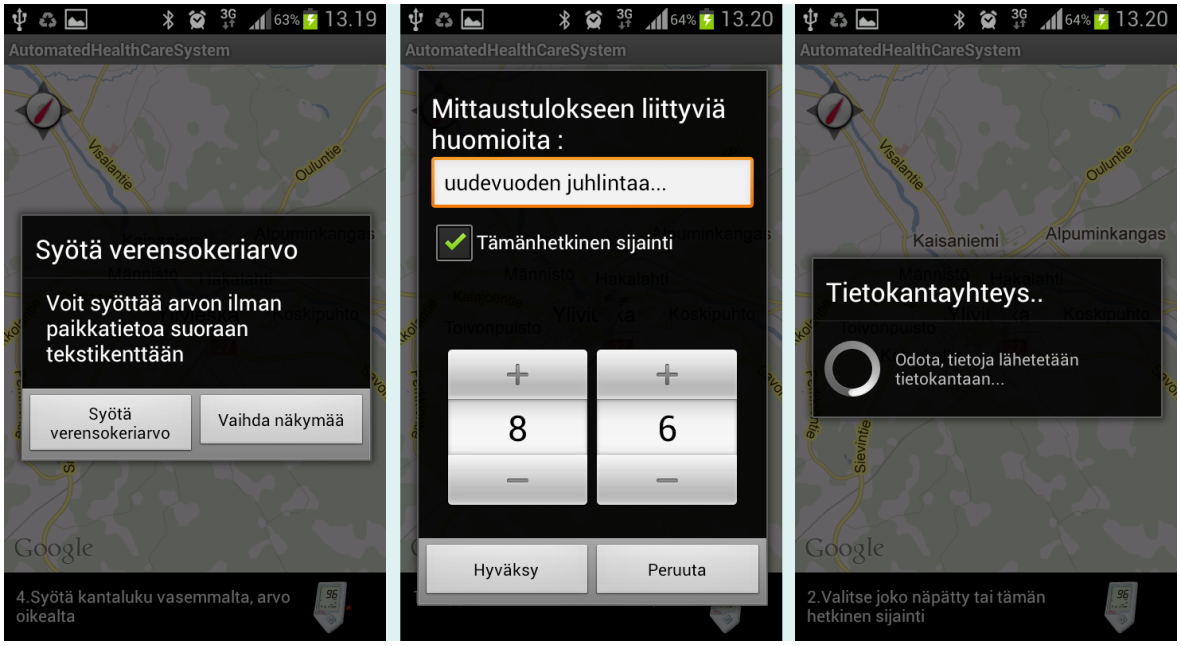


KUVIO 2. Automated Health Care System –mobiilisovelluksen aloitusnäkymä, sisäänkirjautuminen sekä menu-valikko

Avatessaan ensimmäistä kertaa ohjelman käyttäjän on luonnollista vierailla käyttäjätiedot –sivulla, jossa hän voi täydentää omaa profiiliaan yksityiskohtaisilla tiedoilla kuten pituus, paino ikä jne. Ne ovat tärkeitä tietoja laskettaessa mm. painoindeksejä, arvioitaessa riskirymään kuulumista ja laadittaessa vaikkapa henkilökohtaista kunto-ohjelmaa tai ruokavaliota. Tiedot tallentuvat tietokantaan rajapinnan kautta, jossa ”ALTER TABLE” –komennolla täydennetään ensimmäisessä kirjautumisessa avomeksi (ALLOW NULL) jääneitä kohtia.

Mobiilisovelluksen ehkä keskeisin alaohjelma on verensokeritietojen tallentamiseen suunnattu sovellus. Siinä käyttäjälle avautuu kotinäkymä, joka kohdistuu sen hetkiseen

sijaintiin mikäli GPS-signaali on saatavilla. Muutoin näkymä on kotikohteessa. Google tarjoaa karttapalvelunsa myötä andorid-kirjaston, jonka ominaisuuksiin sekä verensokerisovellus, että myöhemmin esiteltävä sykkeen mittaamiseen tarkoitettu sovellus perustuu suurilta osin. Verensokerisovelluksessa käyttäjä näppää kartalta halutun sijainnin ja esiinnousevasta dialogista valitsee haluaako hän tallennettavan sijaintitiedoiksi tämänhetkinen sijainnin (oletus) vai kartalta näpätyn kohteen sijainnin. Tämä mahdollistaa siis myös jälkikäteen tehtävien päiväkirjamerkintöjen tekemisen. Sijainti tallennetaan desimaalimuotoisina itä-länsi ja pohjois-eteläsuuntaisina koordinaatiston pisteinä. Lisäksi tietokantaan tallennetaan myös katuosoite, joka saadaan google-maps api-kirjaston getFromLocation –metodista. Käyttäjä saa välittömän palautteen tietokantaan tallentamisen onnistumisesta. Tietokantaan tallentaminen sekä automaattisen tekstiviestin lähettäminen suoritetaan hyvän ohjelmointitavan mukaisesti rinnakkaisessa säikeessä, joten http-pyynnön ollessa suoritusvaiheessa käyttäjä voi huoletta käyttää alasovelluksen muita ominaisuuksia. Kuviossa 3. on esitelty verensokerimittausten tallentamiseen suunnatun aliohjelman toiminnallisuuksia.



KUVIO 3. Automated Health Care System –mobiilisovelluksen verensokerimittausten tallentamiseen suunnattu aliohjelma

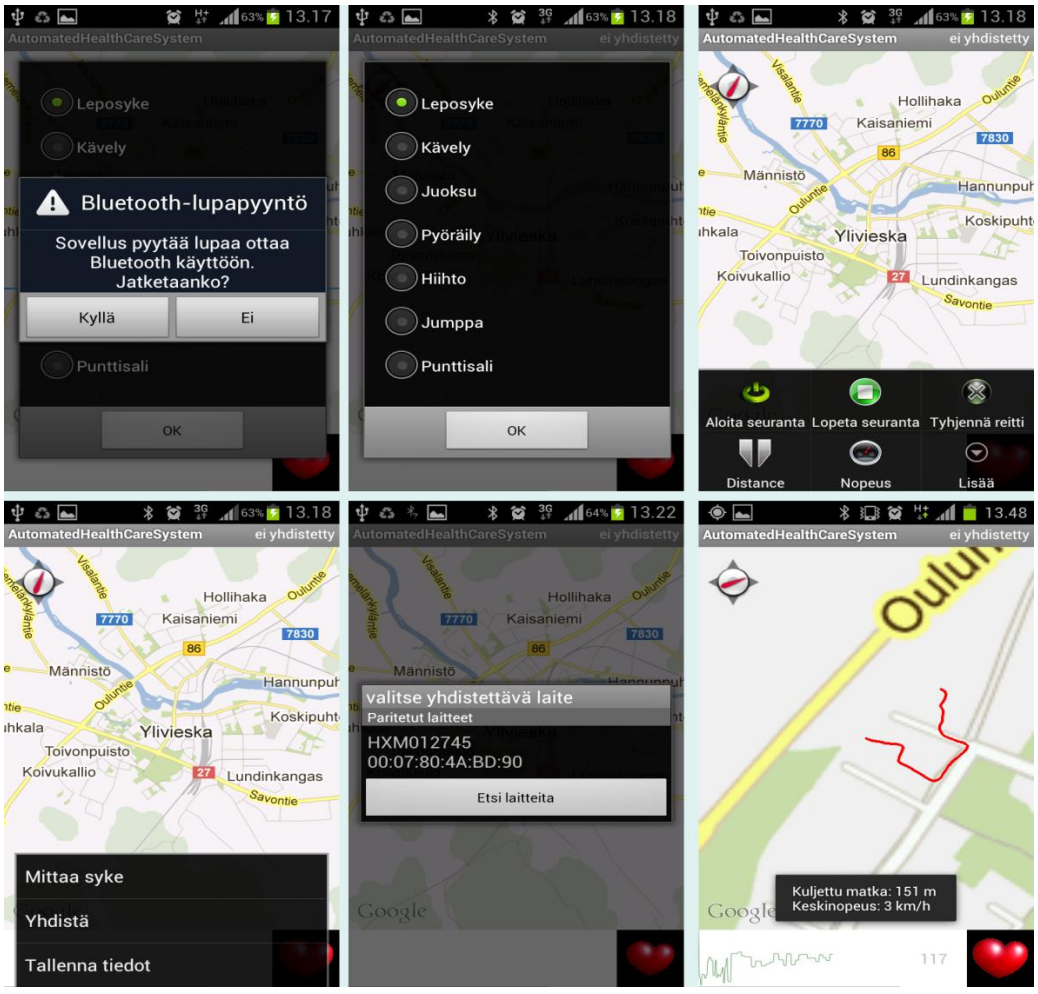
Sekä verensokerisovellukseen että sykkeen mittaamiseen suunniteltuun sovellukseen on tuotu myös graafisesti havainnollistavia ja silmää miellyttäviä elementtejä kuten aiheeseen liittyviä animaatioita ja ohjekenttien animointia. Vaikka android ei suoraan tuekaan gif-animointia, voidaan kyseinen puute kiertää hyödyntämällä AnimationDrawable –luokkaa.

Sykkeen mittaamiseen suunnattu alasovellus käynnistyy monivalintaruutuna, jossa käyttäjä valitsee sopivan harjoitusmuodon. Liikuntapäiväkirjaa täytettäessä on oleellista tietää, minkälaisen harjoitteen kautta mitattu syke on saavutettu. Tämän jälkeen itse sovellus avautuu verensokerisovelluksen tavoin karttanäkymänä, jonka alaosa jakautuu sykekäyrän piirtämiseen tarkoitettuun, sykkeen numereeniseen näyttöön tarkoitettuun ruutuun sekä äärimmäisenä alhaalla oikealla olevaan havainnollistavaan sykeanimaatioon. Käyttäjä aloittaa sykkeen mittaamisen käynnistämällä menuvalikosta löytyvällä ”yhdistä” –painikkeella joko parittamattomien tai paritettujen laitteiden haun tai valitsemalla oma laite suoraan listasta. Yhdistämisprosessin aikana käyttäjä saa palautteen yhdistämisen onnistumisesta jonka jälkeen mittaus alkaa välittömästi, sykekäyrä piirtyy erillisellä graafin piirtämiseen tarkoitetulla luokalla graafialueelle ja keskisykkeen laskemiseen tarvittavat syketiedot alkavat tallentua tiedoille varattuun taulukkoon. Sykenäkymän alaosassa verensokerisovelluksen kanssa yhteneväisesti sijoitettu animaatio havainnollistaa käyttäjälle osaltaan sen, minkälaisesta sovelluksesta on kyse.

Tyypillisen android-sovelluksen tapaan alasovelluksen eri toiminnot on sijoitettu alavalikkoon (menubar), jotta varsinaiselle päänäytölle jäisi riittävästi tilaa. Sykkeen ja huippusykkeen mittaamisen lisäksi alasovelluksen toimintoihin kuuluu matkan ja nopeuden mittaaminen. Kuljetun reitin varrelta tallennetaan tasaisin välimatkoin reittipisteitä, joiden välille androidin Paint –luokkaa apuna käyttäen piirretystä viivasta muodostuu yhtenäinen reittiesitys. Piirtäminen on toteutettu liittämällä kartan päälle eri tasoja (OverLay), tallentamalla tasot listaan (OverLaylist) ja piirtämällä lopuksi listan eri tasot kartalle. Reittitason lisäksi käytössä on karttamerkkien (Pinpoint) piirtämiseen tarkoitettu taso. Jos GPS-signaalia ei ole saatavilla tai käyttäjä haluaa tallentaa ainoastaan esimerkiksi leposykkeen, syketietojen tallentamiseen tarkoitettu tietokantataulu sallii nolla-arvot nopeuden, matkan, ajan ja koordinaattitietojen sarakkeisiin. Tällöin näiltä osin tulevassa tiedonlouhinnassa otetaan huomioon ainoastaan syke ja huippusyke.

Ohjelma tarkistaa valitun toiminnon suhteen, onko riittävästi dataa tallentunut suoritteen läpiviemiseksi. Mikäli kyseessä on toiminto, joka ei salli nolla-arvoja tietokannassa, tietokantaan tallennusta ei sallita. Edellä kuvatussa tilanteessa suoritus jatkuu normaalina kunnes käyttäjä yrittää tallennusta uudelleen. Käyttäjä saa viipymättä palautteen, jossa kerrotaan tulokset mitatuista parametreista sekä tietokantaan tallentamisen onnistuminen.

Android-ohjelmoinnissa on tärkeää huomioida, että voimakkaasti järjestelmää kuormittavat toiminnot on säikeistettävä huolellisesti. Esimerkiksi tietokantaan tallentamiseen käytettävät web-palvelupyynnöt ovat riippuvaisia verkon kuormitustilasta ja voivat aika-ajoin vaatia pitkänkin ajan onnistuakseen. Jos androidin pääsäie keskeytyy kuuden sekunnin ajaksi, järjestelmä pysäyttää sovelluksen suorittamisen, joten erilaiset kuormittavat toimenpiteet kuten web-pyynnöt ja ajastusta vaativat toiminnat on suoritettava omissa säikeissään. Syketietojen sekä muun kuntoiluun liittyvän harjoitusdatan hallinnointiin suunnatun aliohjelman toimintoja on kuvatta kuviossa 4.



KUVIO 4. Harjoitusdatan hallintaan suunnattu aliohjelma

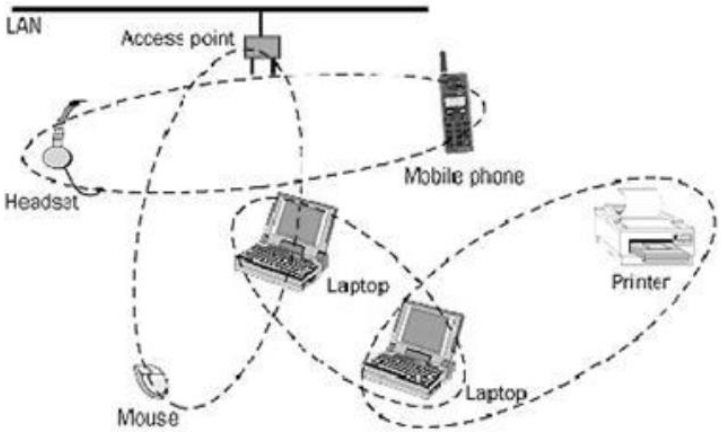
4.3 Android, java ja Eclipse –kehitysympäristö

Yksinkertaistettuna Android on Googlen mobiilikäyttöön muokkaama Linux-käyttöjärjestelmä. Vaikka Android perustettiin vasta vuonna 2003 (Darell 2011), on sen ekosysteemin kehittyminen vertaansa vailla oleva menestystarina. Osasyy menestykseen on varmasti se, että Google osti sen pian perustamisen jälkeen ja kehitti sitä voimakkaasti kunnes ensimmäin android-käyttöjärjestelmä julkaistiin vuonna 2007. On arvioitu, että vuonna 2013 maailmassa on miljardi aktivoitua android-laitetta (Dediu 2012). Android-koodia kirjoitetaan java-kielellä johon Google on kehittänyt ja julkaissut lukuisia android-kirjastoja. Lisäksi käyttöjärjestelmän ympärille on kehittynyt vakuuttava kehittäjäyhteisö, joka kehittää ns. kolmannen osapuolen kirjastoja sovelluskehittäjien käyttöön.

Nykyisin androidin kehittämisestä vastaa Open Handset Aliance (Linux-wiki 2012), mutta google tarjoaa sovelluskehittäjille ilmaisen kehitysympäristön (IDE) ja ilmaisen android–lisäosan(android-sdk) java-kehittäjien suosimaan Eclipse–kehitysympäristöön. Android–ohjelmien kehitys ja julkaiseminen on siis täysin ilmaista ja siksi myös AHCS kehitettiin Eclipse IDE-kehitysympäristössä, joka on valitun linjan mukaisesti avoimeen lähdekoodiin perustuva sovelluskehitystyökalu.

4.4 BlueTooth Androidissa

Bluetooth on IIIE 802.11 –stantardiin pohjautuva radiotie, jossa mitkä tahansa kaksi tai useampi samalla bluetooth-profiililla varustettua laitetta kytkeytyy yhteiseen langattomaan ad-hoc –verkkoon. Verkko on kaksipisteyhteyteen perustuva pakettikytkentäinen verkko, jossa taajuushyppely (frequency hopping) –tekniikkaan pohjautuen käyttäjä-data pilkotaan kehyksiin ja lähetetään 1600 kertaa sekunnissa vaihtuvalla bluetooth-kantoaallolla. (Brax 2011, 5-8 a.) Bluetooth-laite voi olla esimerkiksi matkapuhelin, tietokoneen näppäimistö, handsfree-laite, gps-paikannin, tulostin, verensokerimittari, verenpainemittari, sykemittari jne. Bluetoothin kantomatka on lähetystehosta riippuen noin 0,1-10 metriä ja yhteys ei vaadi toimiakseen esteetöntä näköyhteyttä. Datansiirtonopeudessa on huomioitavaa, että bluetooth-siirto on asyncronista ja jakaa sekä siirtotien nopeuden että suunnan. (Brax 2011, 5-8 a.) Bluetooth 2.0 siirtonopeus on 3Mbit/s. Bluetooth voi toimia myös pääsyverkkona internetiin. Tällöin puhutaan ”IP-over-Bluetooth” –tekniikasta. Kuviossa 2. on esitetty periaatekaavio bluetoothin toiminnasta.



KUVIO 5. Periaatekaavio bluetoothin toiminnasta (Brax 2011, 83.)

Android-alusta tarjoaa bluetooth-rajapinnan (API, Application Programming Interface) kautta laajan tuen bluetooth-toiminnallisuuden toteuttamiseksi. ”BluetoothAdapter” luo pohjan koko kommunikaatiolle. Sen pohjalta voidaan etsiä näkyväksi asetettuja bluetooth-laitteita. BluetoothDevice edustaa kuuluvuusalueella olevia laitteita (paritettuja tai parittamattomia), joilla jokaisella on oma uniikki MAC-osoite. Bluetooth toimii kuten TCP/IP –protokolla: toisen laitteista on oltava palvelimen roolissa (server socket) eli kuunneltava sisääntulevia pyyntöjä toisen ollessa asiakkaan roolissa (client). (Android developers 2012a.)

Ennen kuin bluetooth-laitteiden etsintään voidaan ryhtyä, on bluetooth otettava käyttöön android-manifest –tiedostossa. Lisäksi hyvän ohjelmointitavan mukaista on testata, tukeeko laite bluetoothia ja onko bluetooth sallittu. Varsinainen bluetooth-laitteiden etsintä voidaan suorittaa joko parittamattomien laitteiden etsintään tarkoitetulla StartDiscovery() –funktiolla tai paritetuille laitteille suunnatulla getBondedDevices() –funktiolla. Parittamattomat ja paritetut laitteet voidaan AHCS:n tapaan listata listanäkymään, josta käyttäjä voi pyyhkäisemällä joko parittaa parittamattomaan laitteeseen tai yhdistää valmiiksi paritettuun laitteeseen. Paritettujen ja yhdistettyjen laitteiden ero on siinä, että yhdistettyjen laitteiden välillä avautuu RFCOMM –kanava, jota pitkin suojattu tiedonsiirto on mahdollista. (Android developers 2012b.) On myös jälleen huomioitava, että laitteiden etsintä, parittaminen, yhteyden avaaminen sekä yhteyden ylläpitäminen ovat runsaasti resursseja vieviä toimenpiteitä, joten kaikki edellä mainitut toimenpiteet on toteutettu AHCS –puhelinsovelluksessa omissa säikeissään. Lisäksi bluetoothin suhteen

erityishuomiota vaatii runsaasti laitteen resursseja kuormittava laitteiden etsintä, joka on hyvä lopettaa heti kun yhteys on luotu.

Yhteyden ollessa auki, yhteyden molemmilla osapuolilla on avoin yhteyskehys (bluetoothSocket) samassa RFCOMM –kanavassa. Tällöin tiedonsiirto noudattaa yleisiä tiedonsiirron lainalaisuuksia: tiedon kirjoitus ja lukeminen tapahtuu sisään tulevan ja ulos menevän vuon (imput-/outputStream) kautta bitti kerrallaan.

AHCS:n puhelinsovelluksen bluetoothia hyödyntävissä alasovelluksissa tarkistetaan käynnistysvaiheessa, onko bluetooth sallittu (getDefaultAdapter() palauttaa nollan, jos bluetooth ei ole sallittu) ja käytössä (jos ei käytössä, voidaan käynnistää sisäänrakennettu bluetooth -pyyntö). Tässä vaiheessa riippumatta alasovelluksesta, käyttäjän on valittava menuvalikosta ”etsi laitteet” –toiminto, joka käynnistää ”DeviceListActivity” –luokan. DeviceListActivity alustaa ja esittää listanäkymät (ArrayAdapter - ListView) ja käynnistää tarvittaessa aikaisemmin käsitellyt funktiot parittamattomien ja paritettujen laitteiden etsimiseen. Laitteet esitetään omissa listanäkymissään ja käyttäjän pyyhkäistyä haluttua laitetta, välitetään laitteen MAC –osoite takaisin pääluokkaan, missä tunnettu osoite parametrina kutsutaan ”ConnectionService” –luokan connect() -funktiota. Kutsutun luokan tehtävänä on hoitaa laitteiden yhdistämiseen sekä olemassa olevan yhteyden ylläpitämiseen vaadittavat toimenpiteet. Uusien laitteiden etsintä lopetetaan ja ellei yhteyden uudelleenrakentamiseen johtavaa poikkeusta tapahdu, kutsutaan tässä vaiheessa yhteyden ylläpitämiseen tarvittavien funktioiden toiminnallisuudesta vastaavaa säiettä. Parametrina ”connected” –säikeeseen viedään yhteyden luonnissa saatu kehys (socket) ja mm. laitteen MAC-osoitteen sisältävä laite-olio.

AHCS –järjestelmässä sykemittarin bluetooth –moduuli käyttää sarjaliikenteen mahdollistavaa Serial Port bluetooth –profiilia. Yhteyden muodostamiseen kuuluvan proseduurin jälkeen kommunikaatio on pelkkää sarjamuotoista tiedonsiirtoa, joka perustuu bitti kerrallaan tapahtuvaan tiedon lähettämiseen ja vastaanottamiseen. Tiedon lähettäminen ja vastaanottaminen tapahtuu input- ja outputStream:n avulla. AHCS-mobiilisovellus ei kirjoita bluetoothin välityksellä mitään ulospäin, joten tästä näkökulmasta ainoa huolehdittava asia on sisään tuleva datavuo. Se luetaan bitti kerrallaan datataulukkoon (bytearray) ja muutetaan numeerisesta arvostaan tulostettavaan tekstimuotoon (String). Sykevyön valmistajan toimittama datalehti kertoi selkeästi

vastaanotettavien tavujen merkityksen ja järjestyksen. Tarvittavat tiedot kuten varsinainen syketieto oli annettujen tietojen perusteella helposti parsittavissa

Android:in sisäinen luokka ”Handler” huolehtii säikeiden ja pääluokan välisestä tiedonsiirrosta. AHCS –mobiilisovelluksessa luokkaa on hyödynnetty esimerkiksi lähettämällä säikeitten välillä ”lippuja”, jotka ilmoittavat pääluokalle säikeen tilan. Tällä tavoin pääluokka voi informoida käyttäjää siitä missä tilassa kunkin tehtävän suorittaminen milloinkin on. Myös varsinainen tiedonsiirto kuten mittaustulosten lähettäminen yhteyttä ylläpitävältä säikeeltä pääluokkaan ja toisaalta edelleen saman informaation lähettäminen pääluokasta web-pyynnöstä huolehtivalle säikeelle tapahtuu juuri Handler-luokan instanssin avulla.

4.5 Puhelinsovellus teknisestä näkökulmasta

AHCS -mobiilisovelluksessa on käytetty muutamia tekniikoita, jotka jo läpikäytyjen aiheiden lisäksi on syytä nostaa lähempään tarkasteluun. Android tarjoaa kaksi erilaista tapaa tehdä web-kyselyjä: HttpURLConnection sekä Apache:n tarjoama HTTP-Client. Tässä opinnäytetyössä käytettiin Apachen tarjoamaa kirjastoa lähinnä kehittäjäyhteisöltä saadun kirjaston vakautta ja monipuolisuutta koskevan positiivisen palautteen perusteella. Http –yhteyttä varten AHCS:ssä on luotu oma luokka: ”CustomHttpClient.java”. Sen metodit ”getHttpClient()” sekä ”executeHttpPost/Get()” alustavat httpclient –instanssin sekä huolehtivat varsinaisesta yhteydenotosta ja datasiirrosta. Luokan metodeja kutsutaan esimerkiksi sisäänkirjautumista hoitavasta login-luokasta sekä lähetä-säikeestä, jolloin parametrina funktioon viedään web-pyynnön kohteena oleva osoite sekä avain-arvo –pareja sisältävä taulukko-objekti.

Varsinainen järjestelmän tuotteistus ei kuulu tämän opinnäytetyön tutkimusongelman piiriin, mutta tuotteistus on huomioitu muunmuassa rakentamalla mobiilisovellus joustavaksi kääntää nopeasti eri kielille. Androidissa erilaiset tekstikentät voidaan joko ohjelmoida (hard code) suoraan layout.xml –tiedostoon, jossa määritellään jokaisen luokan ulkoasu (layout) joko graafisen käyttöliittymän kautta tai suoraan xml-komennoin. Toinen – ja samalla joustavampi – tapa on määritellä jokaiselle erilliselle tekstille erillinen tekstimuuttuja, johon ainoastaan viitataan layout.xml –tiedostossa. Jokaiselle käytettävälle kielelle voidaan näin muodostaa oma tekstiviitteitä sisältävä xml-tiedosto ja haluttaessa

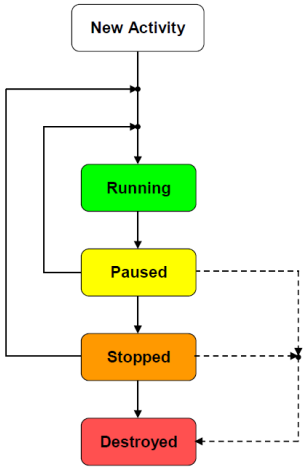
kääntää sovellus eri kielelle, tarvitsee ainoastaan kyseinen tiedosto vaihtaa sen sijaan että jokainen yksittäinen teksti vaihdettaisiin erikseen.

Eräs alkuperäisen vaatimusmäärittelyn hyödyllisimmiksi määreiksi nousi järjestelmän kyky lähettää saadut mittaustulokset tekstiviestinä toiseen matkapuhelimeen. Sen lisäksi, että sovelluskehittäjä voi kutsua itse kehittämiään sovelluksen osia (Activity/Class), Android-käyttöjärjestelmä tarjoaa monipuolisen joukon sisäänrakennettuja aktiviteetteja. Nämä aktiviteetit voivat suoraan hyödyntää puhelimen omia resursseja, joista muutamana esimerkkinä tekstiviestin lähettäminen, puhelu sekä web-selaimen ja kameran käyttö. Uuden aktiviteetin käynnistäminen tapahtuu ”Intent” –olion avulla, johon on tallennettu kaikki oleellinen tieto kommunikoinnin sujumiseksi halutulla tavalla. (Juen 2012, 103-106.) Mobiilisovelluksessa päädyttiin kuitenkin soveltamaan edellä opittua tekniikkaa ja hyödynnettiin ”SmsManager” –luokkaa, jota käytettäessä järjestelmä huolehtii tekstiviestin lähettämisestä automaattisesti eikä joka kerta vaadi erikseen käyttäjän suostumusta viestin lähettämiseen.

Tekstiviestin lähettäminen siinä missä mikä tahansa muukin sovellukseen, käyttöjärjestelmään tai käyttäjään vaikuttava tekijä on osa Androidin turvallisuusarkkitehtuuria ja siten huomioitava erikseen android-manifest.xml –tiedostossa. Vaikka eri sovellusten tulee toimiakseen jakaa tiettyjä resursseja, turvallisuussyistä lähtökohtaisesti nämä on erotettava toisistaan sekä käyttöjärjestelmästä (adnroid developers 2012a). Kysymys on muunmuassa käyttäjän tietosuojasta sekä siitä millä tavalla eri osapuolilla on lähtökohtainen oikeus käyttää käyttäjätietoja ja millä tasolla käyttäjällä on oltava mahdollisuus hallita sovelluksen oikeutta käyttää puhelimen resurseja. Käyttöoikeudet annetaan tapauskohtaisesti android-manifest.xml -tiedostossa muodossa ”<uses-permission>”.

Android-aktiviteetin elinkaarimalli (lifecycle) antaa sovellukselle paljon joustavuutta, mutta samalla tuo mukanaan monia seikkoja, jotka sovelluskehittäjän on osattava ottaa huomioon. Käynnistettäessä ohjelmaa, se voi olla joko täysin kuolleessa tilassa tai se voi olla elossa taustalla, jolloin sen suorittaminen jatkuu siitä aktiviteetista johon suorittaminen edellisellä kerralla keskeytyi. Kuviosta 6. voimme nähdä android-aktiviteetin elinkaaren eri vaiheet. Running-tilassa aktiviteetti on suorituslistan päälimmäisenä - aktiivinen ja näkyvä käyttäjälle. Paused –tilassa aktiviteetti on näkyvillä, mutta jostain syystä

menettänyt asemansa suhteessa käyttöjärjestelmän ykköprioriteettina. Tämä on tilanne esimerkiksi mobiilisovelluksessamme ruutuorientaation muuttuessa. Tällöin ilman lisätoimenpiteitä muunmuassa sykkeen mittaamiseen keskittyvä säie keskeytyy. Paused- ja stopped –tilassa aktiviteetti ei menetä käyttäjädataa eikä tilatietoa, mutta on siinä mielessä vaarallinen, että järjestelmä voi muistin loppuessa tappaa automaattisesti sovelluksen. (Juen 2012, 106-122.) Onkin hyvän ohjelmointitavan mukaista varmistaa käyttäjädatan säilyminen esimerkiksi varastoimalla sovelluksen tila ja tiedot esimerkiksi käyttöjärjestelmän sisäiseen SqlLite –tietokantaan. Lopullisesti aktiviteetin elinkaaren lopettaa onDestroy –tila, jossa finish() –funktion kutsuminen tuhoaa kaiken käyttäjädatan sekä tilainformaation ja vapauttaa lopullisesti käytetyn muistin muuhun käyttöön.



KUVIO 6. Android-aplikaation elinkaarimalli

5 WEB-SOVELLUS

5.1 Web-sovellus osana järjestelmää

Diabeteksen omaseuranta on sen tärkeästä roolista huolimatta tähän saakka hoidettu potilaan käsin pidettävän kirjanpidon avulla. Hoitaja on saanut käytettäväkseen muutamien kuukausien välein laboratoriossa tehtävien Hba1c (suuntaa-antava pitkän ajan keskiverensokeri) mittausten lisäksi ainoastaan paperivihkoseen kirjatut mittaustulokset. Tulosten analysointi on kiireellisistä käynneistä johtuen hyvin pintapuolista ja merkittävätkin hoitopäätökset joudutaan tekemään varsin kevein perustein.

AHCS –web-sovellus tarjoaa hienon työkalun tallentaa ja esittää kotona tapahtuvan omahoidon tuloksia. Lisäksi se tarjoaa tietoa tavallisimmistä diabetestä käsittelevistä aiheista ja tehokkaan työkalun tilastoida ruokailutottumuksia sekä opetella eri ruokalajeille ominaisia hiilihydraattimääriä. Web-sovelluksen kautta käyttäjä voi turvallisesti hallinoida profiiliaan ja tietojaan sekä mahdollisesti jatkokehityksen myötä tulevaisuudessa asettaa erilaisia hälytysarvoja trendipoikkeavuuksien varalle.

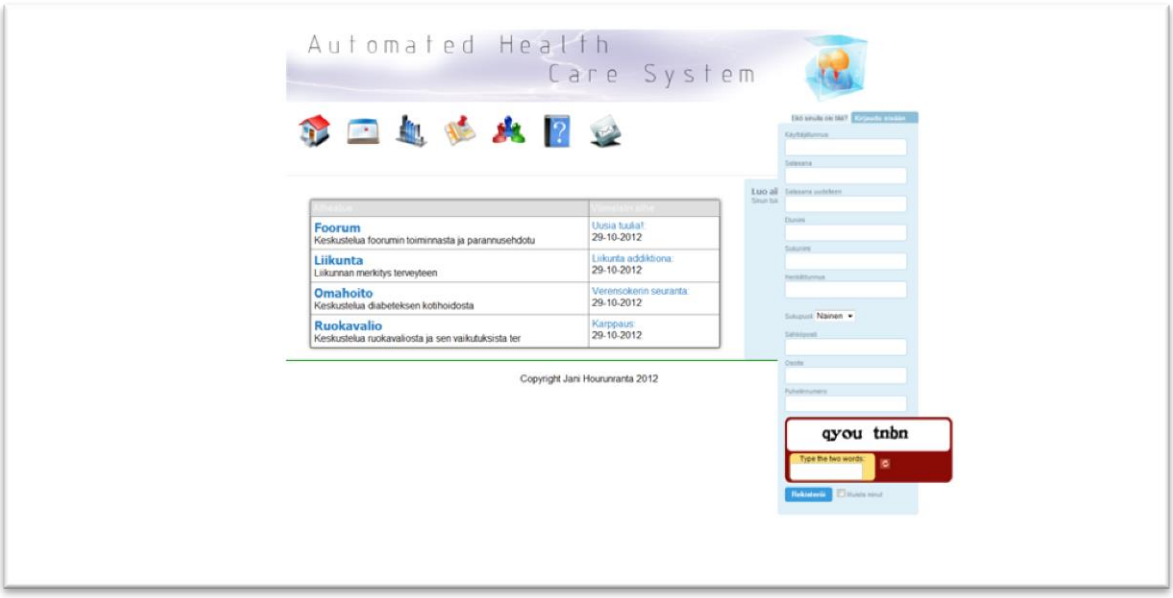
Web-sovelluksen rooli osana järjestelmää on tärkeä myös siinä mielessä, että se antaa kasvot ja graafisen ilmeen koko järjestelmälle. Raikas, vaalea värimaailma on pyritty luomaan kuvaamaan järjestelmän sijoittumista terveydenhuoltosektorille. Myös puhelinsovellus noudattaa – käytettävyydestä tinkimättä – valittua layoutia.

5.2 Web-sovelluksen toiminnot

Web-sovelluksen suunnittelussa toteutusideoita oli varsin runsaasti, mutta opinnäytetyön rajauksen puitteissa mukaan otettiin tässä vaiheessa ainoastaan järjestelmän toiminnan ja idean kannalta elintärkeät toiminnallisuudet. Tärkein toiminnallisuus koko järjestelmän kannalta on profiilin luonti ja sisäänkirjautuminen. Kuten kuviosta 7. on havaittavissa, toiminnot sijoittuvat käytettävyystutkimustenkin mukaan luontevalle paikalle sivun oikeaan yläreunaan liikemerkin alle. Toiminnallisuus muuttuu dynaamisesti sen mukaan, onko ”$_SESSION” –muuttujassa tieto voimassaolevasta kirjautumisesta joko sisään –tai uloskirjautumiseksi. Sisäänkirjautumisen sekä profiilinluonnin animointiin jQuery tarjoaa valmiin hide/unhide –funktion, joka voidaan määritellä toimivaksi esimerkiksi hiiren painalluksella tai kohdistimen siirrolla. Kirjautumisen sekä profiilinluonnin taustalla AJAX-kysely hoitaa datan kuljettamisen JavaScriptiltä PHP-skriptille, jossa suoritetaan

datan validointi ennen varsinaisia tietokantakyselyjä. Datan validoinnista sekä tietoturvallisesta ohjelmoinnista tarkemmin kappaleessa ”7. Tietoturva”.

Web-sovelluksen etusivulta löytyy yleistä tietoa ja viimeisimpiä uutisia diabeteksestä sekä järjestelmästä ja ylläpidosta itsestään. Sivuston header-osio säilyy sivusta riippumatta samana. Se on liitetty PHP-include –funktiolla yhtenä sivuna sivuston rakenteeseen, minkä ansiosta tehtävät muutokset aiheuttaa mahdollisimman vähän lisätyötä sivuston ylläpitäjälle. Header-osiossa sijaitseva navigointi on toteutettu perinteisellä JavaSkriptilla. Yksinkertaisuudessaan sen toiminta perustuu suuriin ja pieniin kuvakkeisiin, jotka vaihtuvat hiiren osoittimen osoittaessa kuvaketta. ”Easing” –funktio pehmittää liikettä ja antaa animoidun vaikutuksen Kuvakkeet on määrätty kellumaan kohti vasenta laitaa, joten yhden kuvakkeen ollessa kerrallaan suuremmassa tilassa, siirtyy muut kuvakkeet tarvittavan määrän oikealle.



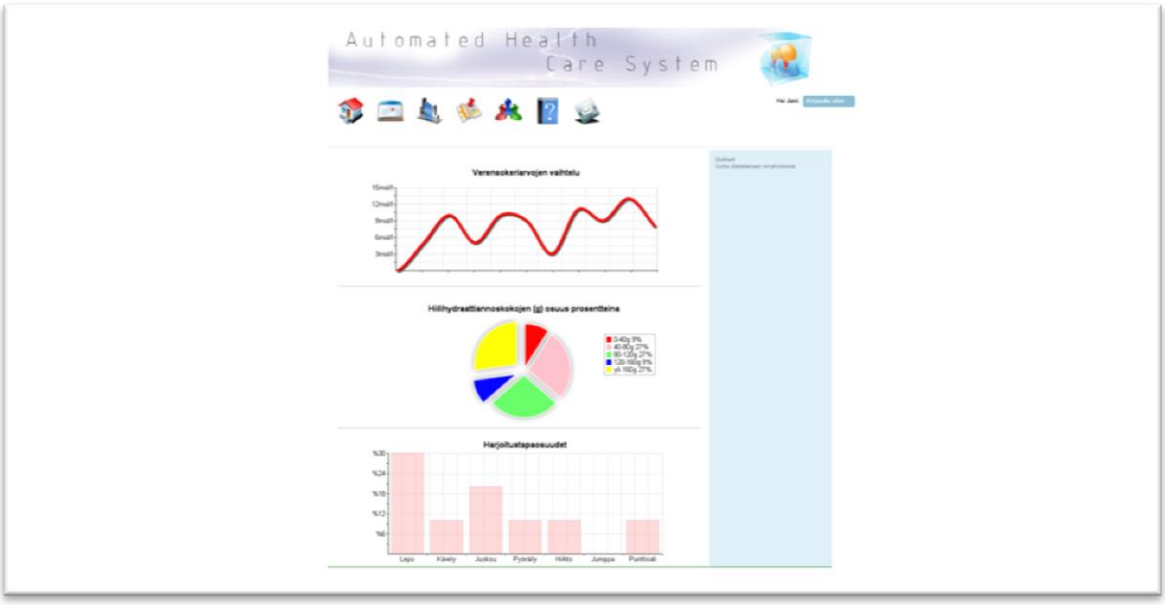
KUVIO 7. AHCS-järjestelmän kirjautumis- ja profiilinluontitoiminnallisuudet

Päiväkirjaosion kantava idea on ”vedä ja pudota” (drag and drop) –toiminnallisuudessa. Käyttäjä voi vetää hiirelle oikealta sarakkeesta haluamansa ruoka-aineen tarjottimelle ja järjestelmä ilmoittaa ruoka-aineen hiilihydraattimäärän sekä lisää sen automaattisesti laskuriin. Päiväkirjamerkinnän oletuspäivämäärä on kyseinen ajankohta, mutta käyttäjä voi halutessaan muuttaa sitä jQueryllä toteutetulla kalenteritoiminnolla. Käyttäjä voi halutessaan myös liittää päiväkirjamerkintään lisäinformaatiota koskien esimerkiksi

aterialla pistettyä insuliiniannosta, päivän liikuntamäärää ja –tyyppiä tai vaikkapa muistiinpanoja vireystilasta tai muista mahdolliseen verensokeritasoon vaikuttavista tekijöistä. Päiväkirja on myös oiva apuväline tuoreille diabeetikoille tai vaikkapa perheen pienimmille opetella eri ruoka-aineiden hiilihydraattipitoisuuksia. Kuviossa 8. on havainnollistettu päiväkirja-osion graafista ilmettä.

KUVIO 8. AHCS-järjestelmän ruokapäiväkirjaosio

Tilasto-osiossa on hyödynnetty ns. kolmannen osapulen kirjastoja ”RGraph” historiatietoja kuvaavien graafien piirtämiseksi. Tilastotietojen näyttämisen edellytyksenä on onnistunut sisäänkirjautuminen, jotta järjestelmä osaa suorittaa tietokantakyselyn oikean käyttäjätunnuksen perusteella. Jälleen AJAX –teknologiaa hyväksikäyttäen tietokantayhteyttä ylläpitävälle PHP-skriptille lähetetään kysely, johon on liitetty oikea käyttäjätunnus. Vastauksena saadaan tällä string –tyyppiseen merkkijonoon tallennetut tiedot, jotka voidaan taulukoksi muutettuna tulostaa graafin muotoon. Erilaisia tietokantakyselyvariaatioita ja koonteja voidaan suorittaa rajattomasti – tämän opinnäytetyön puitteissa tyydytään ainoastaan esittelemään muutaman esimerkkigraafin puitteissa teknologian mahdollisuuksia (Kuvio 9).



KUVIO 9. AHCS-järjestelmän tilasto-osio

Google maps API tarjoaa tehokkaan työkalun esittää karttoja ja sijaintitietoja niin mobiililaitteissa kuin web-sivuillakin. JavaScript –pohjainen ohjelmointirapinta tarjoaa kehittäjälle maksuttoman ja ainakin tätä kirjoitettaessa mainosvapaan karttapalvelun, jonka ominaisuuksia siis hyödynnettiin myös AHCS -järjestelmän rakentamisessa laajasti (Google Developers 2012). Web-sovelluksessa perinteisen linja-diagrammin lisäksi verensokeriarvot esitetään karttanäkymässä käyttäjäprofiilin perusteella valikoiduilla merkeillä (pinpoint). Jokainen verensokerimerkintä näkyy kartalla omana ikoninaan ja ikonia klikkaamalla käyttäjälle avautuu lisätietoikkuna, josta on nähtävissä verensokeriarvon lisäksi katuosoite, aika sekä mittaukseen liittyvät lisätiedot. Toiminnon taustalla on siis tietokanta, johon merkinnät paikkatietoineen on syötetty puhelinsovelluksessa aijemmin kuvatulla tavalla. Php -skripti lataa verensokeri- ja liitännäistiedot tietokannasta ja muodostaa niistä xml-tietueen tulostaen ne ”echo” –komennolla eteenpäin. Itse karttapohja ladataan Googlen karttapalvelusta ja keskitetään annettujen koordinaattien perusteella haluttuun lähtöarvoon. JavaScript, josta edellä kuvattua php-tiedostoa kutsutaan, käy läpi jokaisen verensokerimittaustietueen liittäen niihin toimintatapahtuman (Click Event). Mittaustietuetta edustavat ikonit piirretään kartalle ja käyttäjän valitessa erään ikonin JavaScriptin ”getElementById” –funktio tunnistaa id-numeron perusteella tietueen ja avaa tähän yhdistetyn infoikkunan. Kuviossa 10. on havainnollistettu verensokerimittaustietueiden esitystä karttapohjalla.

KUVIO 10. AHCS-järjestelmän verensokerimittaustietueiden esitys

Kuviossa 11 esitelty keskustelufoorumi sekä erillinen palautelomake hyödyntävät sekä perinteisempää PHP -tekniikkaa että modernimpaa HTML5 –tekniikkaa. Keskustelufoorumissa ainoastaan ylläpitäjällä on oikeus lisätä aihealueita, mutta jokaisella kirjautuneella käyttäjällä on mahdollisuus avata aihealueen alle uusia keskustelusäikeitä. Rakenne on toteutettu perinteisellä ”table” –rakenteella, jolloin esimerkiksi sivun tulostusehdot voidaan määritellä muuttumaan sen mukaan onko käyttäjä kirjautunut sisään vai ei. Kuka tahansa voi lukea foorumin viestejä, mutta säikeiden lisääminen ja viesteihin vastaaminen onnistuu ainoastaan sisäänkirjautumisen jälkeen. Palauteosion sähköpostin lähetys oli suurin yksittäinen syy järjestelmään liitetyn oman postipalvelimen perustamiselle. PHP:n mail –funktio on lähes mahdotonta saada toimimaan ellei järjestelmässä ole omaa SMTP (Simple Mail Transfer Protocol) –palvelinta. Mail –funktio lähettää käyttäjän hyväksyttyä viestin suoraan ohjelmakoodissa määriteltyyn sähköpostiosoitteeseen, tässä tapauksessa osoitteeseen webmaster@kalliopaja.zapto.org.

KUVIO 11. AHCS-järjestelmän keskustelualue

5.3 HTML5

HTML5 on yksinkertaisuudessaan uusi standardi HTML –tekniikalle (w3schools.com 2012 b). Uusi standardi on huomattava laajennus ja se mahdollistaa monien nykyaikaisten sisältöjen esittämisen entistä joustavammin web-sivulla. Videon ja audion tuominen suoraan lähteestä ilman lisäosia mahdollistavien elementtien lisäksi mukana on esimerkiksi karttaelementti, joka on varsin käytännöllinen ominaisuus tämän päivän paikkatieto-orientoituneessa sovelluskentässä. Mukana on myös erityinen email –elementti, joka sisältää automaattisen validoinnin syötetyn sähköpostiosoitteen oikean muodon osalta. Lukuisista uusista elementeistä AHCS:ssä on hyödynnetty erityisesti sivuston rakenteen määrittäviä elementtejä kuten ”section”, ”fieldset”, ”article” jne. Puhutaan erityisesti käsitteestä ”flexible-box-modeling”, joka on lunastamassa paikkansa johtavana sivunasettelutekniikkana (Roberts 2012). Kaikki selaimet ei sitä vielä tue, joten tämä CSS:ään (Cascade Style Sheet) ja HTML5:n perustuva tekniikka tarvitsee toimiakseen –moz-box ja –webkit-box –lisäosat. Valitettavasti tätä kirjoittaessa flexible-box-modelling toimii ainoastaan Firefox ja Safari –selaimissa. Yleisellä tasolla web-sovelluskehittäjän näkökulmasta voidaan todeta, että HTML5 kokonaisuudessaan on ennen kaikkea uskomattoman taipuisa ja monipuolinen yhdistelmä jQueryä, CSS:ää sekä html/xhtml:ää, mikä nostaa dynaamisen web-ohjelmoinnin täysin uudelle tasolle. Nopeiden laajakaistayhteyksien yleistyessä kokonaisia järjestelmiä tulee siirtymään kokonaan selainpohjaisiksi ratkaisuiksi.

5.4 AJAX, JQuery, JavaScript ja PHP

AJAX:n määritelmä on mukautunut ajan kuluessa käsittämään kohtalaisen laajasti proseduurin, joka välittää pieniä palasia dataa käyttäjän huomaamatta web-sovelluksen ja palvelimen välillä. Vaikka alkuperäisen tiukan määritelmän mukaan AJAX tarkoitti asynkronista xml-pohjaista tiedonsiirtoa XMLHTTP-pyynnön avulla, käsitteen perustavoite on säilynyt samana: lisätä web-sovelluksien vuorovaikutteisuutta ja suorittaa enemmän tehtäviä suoraan selaimessa tarvitsematta joka kerta ladata sivua uudelleen palvelimelta (Aleson & Schutta 2007, 15).

AJAX:ssa ei sinänsä ole mitään uutta, vaan se on pikemminkin kokoelma olemassa olevia tekniikoita. AJAX kokonaisuutena käsittää siis käyttäjädatan vastaanottamisen esimerkiksi web-lomakkeelta, XMLHTTPRequest –objektin luomisen sekä käyttäjädatan tallentamisen siihen, pyynnön lähettämisen palvelinskriptille (PHP/ASP) ja myös palvelimelta tulevan palautteen vastaanottamisen ja prosessoinnin. JavaScriptin esittelyyn ei tässä kohtaa muutoin kiinnitetä huomiota kuin toteamalla, että sen päätehtävä on mahdollistaa HTML –elementtien dynaaminen käsittely. AJAX:n näkökulmasta se tarkoittaa sitä, että käyttäjän web-lomakkeelle syöttämä data voidaan identifioida ja käsitellä selaimessa ja käsitellystä datasta voidaan luoda XMLHTTPRequest –objekti. Jquery:n rooli AJAX:ssa on lähinnä JavaScript:ä laajentava – sen avulla käyttäjä voi pyytää mm. tekstitiedoston, html-, xml- tai JSON –dataa ja esittää sen suoraan html –elementissä.

PHP (Hypertext Preprocessor) on säilyttänyt vapaan lähdekoodin ratkaisuna vahvan asemansa esimerkiksi suhteessa kaupalliseen ASP (Active Server Pages) -skriptikieleen. PHP on C-, Java- ja Perl –pohjainen komentosarjakieli, jossa ohjelmakoodi tulkitaan vasta suoritusvaiheessa (Php-manual 2012). PHP on myös alusta- ja käyttöjärjestelmäriippumaton kieli, mikä takaa useassa vertailussa sen aseman johtavana dynaamisten web-palveluiden tuottamiseen tarkoitettuna kielenä (Php-wikibooks 2012). AHCS:n osalta on todettu, että PHP -skripti toimii linkkinä varsinaisen sovelluksen ja tietokannan välillä. AJAX ei käsitteenä muuta kuvaa millään tavalla; AHCS:n kommunikoidessa PHP:n kautta tietokantaan, on web-sovelluksen näkökulmasta rajapintana Jqueryn ohjaama AJAX –proseduuri.

5.5 CSS

CSS on www-dokumentin tyylejä määrittävänä tyylimääritysten kokoelmana kiinteä osa nykyistä sivunkuvauskielien muodostamaa kokonaisuutta yhdessä HTML5:n, Jqueryn ja JavaScriptin kanssa. Alunperin CSS syntyi ratkaisemaan osin sekasortoisen ongelman, jossa HTML elementtien perhe laajeni käsittämään sisältöä merkkaavien elementtejen lisäksi myös tyylejä kuvaavia elementtejä (w3scoohls.com 2012 a). Dynaamisessa web-sovelluksessa JavaScriptin/JQueryn, HTML:n sekä CSS:n roolit ovat selvät; HTML –elementissä voidaan määrittää elementtiä määrittävä identifikaatiomääre esimerkiksi elementtikohtaisesti, elementtiluokkakohtaisesti tai antaa elementille ainutkertainen, uniikki tunnus. CSS –tiedostossa määritellään näitä identifikaatiotunnuksia vastaavat tyylimääreet. Sivun dynaamisuutta voidaan lisätä muuttamalla JavaScriptin tai Jqueryn avulla reaaliaikaisesti mm. CSS-tyylimääreitä. Elementtien kokoa, väriä, varjoa, muotoa, sijaintia, läpinäkyvyyttä ja monia muita ominaisuuksia voidaan säätää CSS-määrittelyjen avulla – tarvitaan ainoastaan tapahtuma (Event), jolla uusi määritys otetaan käyttöön.

Tämän päivän olioperustaisessa ohjelmoinnissa ns. kolmannen osapuolen kirjastot ovat varsin laajassa käytössä. Niiden tehtävänä on tuoda lisäominaisuuksia ohjelmointikielen rajoitettuun perusasuun. Vaikka CSS ei varsinaisesti ohjelmointikieli olekaan, voidaan senkin kautta tuoda laajennuksia kuten aikaisemmin esitellyt –webkit ja –moxkit –selainmoottorit (Rendering engine). Huolimatta siitä, että mm. Google ja Nokia ovat olleet vahvasti mukana kehittämässä –webkit:ä, se on avoimen lähdekoodin projekti, joka on mullistanut mm. mobiili-web –kehityksen. Selainmoottoreiden kehitys on kuitenkin monin osin vielä kesken ja mitään universaalia kaikkien selaimien tukemaa yhteistä moottoria ei ole. Tilanne on kehittäjän näkökulmasta osin sekava ja työläs sillä jokainen selain täytyy näiltä osin huomioida erikseen.

6 TIETOKANTA

Hajautetussa järjestelmässä tietokanta on koko järjestelmän ydin jonne kaikki kerättävä data käyttäjätietoineen ja relaatioineen tallennetaan. On siis selvää, että luvussa 7. ”Tietoturva” käsiteltävät aiheet ovat ensiarvoisen tärkeässä roolissa puhuttaessa terveydenhuoltoon suunnatusta järjestelmästä. Henkilö- ja potilastietotyyppisen datan säilyttäminen vaatii aina huolellisesti suunniteltua arkkitehtuuria sekä suojautumista asiattomia hyökkäyksiä vastaan. Tietosuoja sisältää lain määrittelemät vaatimukset siitä kuinka tietojärjestelmän tulee suojata yksityisiä tietoja oikeudettomalta käytöltä (Brax 2012, 8-10). Myös tiedon normalisointi on syytä otta huomioon suunniteltaessa runsaasti tietueita sisältävää tietokantaa . AHCS kerää runsaasti käyttäjän lähettämiä mittaustuloksia – jatkokehitysvaiheessa myös verenpainemittaustietoa - ja jotta tietokanta ei paisuisi hallitsemattomasti, tarpeettoman tiedon tallentaminen kahteen kertaan tulee minimoida. Kolmas kriittisen tärkeä näkökohta – joka sekin liittyy tietoturvaan – fyysinen tietoturvallisuus. Kukaan ei kykene ennustamaan koneiden ja laitteiden toiminta-aikaa, joten tietokannan varmuuskopiointi ja kaksintaminen ovat ehdottomia edellytyksiä toimintavarmalle järjestelmälle. Valitettavasti resurssisyistä tämän opinnäytetyön puitteissa edellä kuvattuja varotoimenpiteitä ei AHCS:n kohdalla ole tehty. Aloittelevan järjestelmäkehittäjän on toki hyvä tallentaa tietokannan rakentamisessa käytetyt SQL-lauseet, jolloin kuva järjestelmästä säilyy muutoksia tehtäessä. Toinen hyvä vaihtoehto on käyttää ”mysqldump” –komentoa, joka tekee tiedostoon täydellisen tietokantakuvan muodostimineen sekä sisältöineen, jolloin koko tietokanta on siirrettävissä tai palautettavissa uuteen kohteeseen.

6.1 Tietokanta osana hajautettua AHCS-järjestelmää

AHCS –järjestelmän päätehtävä on huolehtia käyttäjädatasta, johon sekä autentikointi että historiatietojen esitys perustuu. Tietokannassa on omat taulut niin käyttäjätiedoille, syketiedoille kuin verensokeri- ja verenpainetiedoille. Taulu tietokannan yksikkönä on yksinkertaisuudessaan kaksiulotteinen taulukko, joka sisältää tietueita (rivejä ja sarakkeita). Tietojen tallentaminen omiin tauluihinsa tarkoittaa karkeasti yleistäen tietokannan normalisointia. Esimerkiksi verensokerin mittaustietoja sisältävä taulu ei sisällä käyttäjätietoja, ainoastaan viittauksen käyttäjään käyttäjätaulun kanssa yhteisen käyttäjätunnisteen avulla.

Ennen sisäänkirjautumista käyttäjällä on käytössään varsin rajoitetut toiminnot. Esimerkiksi puhelinsovellus on rakennettu niin, että introruudun jälkeen käyttäjälle avautuu heti kirjautumisruutu – ennen kirjautumista muu toiminnallisuus ei ole käytettävissä. Web-sovelluksessa ennen kirjautumista toiminnallisuutta on hieman enemmän käytössä: ohje-osio on luonnollisesti käytössä, keskustelupalstalta voi lukea viestejä jne. Kirjautumisprotokolla on kuitenkin välttämätön käydä läpi edes perustason suoritteiden onnistumiseksi – autentikointitiedoista huolehtivan tietokannan on siis toimittava luotettavasti.

Kirjauduttuaan käyttäjällä on oikeus suorittaa kaikki järjestelmän ydintoiminnot, joista verensokeritason mittaaminen ja tietokantaan lähettäminen ovat oleellisimmat. Tietokantayhteydestä huolehtiva PHP-skripti ottaa vastaan puhelinsovellukselta http-post – pyynnön yhteydessä parametrina tulevan postparameter-objektin, joka sisältää avain-arvo – pariin sidottuna käyttäjätunnuksen, kryptatun salasanan sekä tietokantaan tallennettavan arvon. Validoituaan vastaanottamansa datan, skripti suorittaa tietokantakyselyn, jossa se tallettaa tietokantaan mittaustuloksen sekä käyttäjätunnusta ja kryptattua salasanaa vastaavan käyttäjätunnisteen. Huomioitavaa on, että salasanat tallennetaan tietokantaan aina kryptattuna, jolloin salasanat ei ole selkokielisenä edes järjestelmän pääkäyttäjän nähtävillä. Lisäksi tietokantaan on ohjelmoitu automaattinen ajankohdan poimiva toiminnallisuus (timestamp), jonka avulla jokaiseen tietueeseen saadaan liitettyä myös aikaleima. Edellä kuvatun kaltainen proseduuri on varsin yleinen ja myös tietoturvallinen tapa hoitaa mobiileja tietokantayhteyksiä – tietoturvallisessa ohjelmoinnissa mobiililaitteen ei koskaan tulisi ottaa suoraa yhteyttä tietokantaan.

Proseduuri tietokantakyselyjen näkökulmasta on hieman erilainen puhelinsovelluksessa suhteessa web-sovellukseen jo siinäkin mielessä, että web-sovellus sijaitsee fyysisesti samalla palvelimella kuin itse tietokantakin. Näin ollen http-kyselyä ei tarvita. PHP-kieliset tietokantakyselyt voidaan suorittaa suoraanweb-sivulta edellyttäen, että sivu ei sisällä tietokantayhteyden avaavaa skriptin osaa. Tietoturvallisen ohjelmoinnin periaatteiden mukaisesti tietokantayhteyden ylläpitävä skripti on aina säilytettävä erillään varsinaisesta sivunkuvauksesta.

Tietokanta on tärkeässä roolissa myös käyttäjäpalautetta keräävän yhteyslomakkeen ja erityisesti keskustelufoorumin sekä ruokapäiväkirjan taustalla pyörivänä tallennusmediana.

SQL tarjoaa perusteelliset työkalut erilaisiin tietokantakyselyihin, joiden avulla esitettävä tieto voidaan koostaa eri tauluista monipuolisilla haku- ja järjestelyehdoilla. Tällä tavalla lääkärille on esitettävissä mitä perusteellisimmin variaatioin potilaan hoitohistoria suhteessa eri muuttujiin. Keskustelufoorumin osalta tietokantaan tallennetut viestit voidaan esittää mm. ajan, käyttäjän, aiheen tai aihealueen mukaan järjestettynä.

6.2 Tietokannan rakentaminen

Tietokannan rakentaminen aloitetaan jo palvelintietokoneen käyttöjärjestelmän asentamisen yhteydessä. LAMP (Linux-Apache-MySQL-PHP) –palvelinohjelmistoon kuuluva MySQL on alun perin suomalaisen Michael Wideniuksen kehittämä tietokantaohjelmisto, jota luonnehditaan maailman suosituimmaksi vapaan lähdekoodin tietokannaksi (MySQL.com 2012). Varsinaisen asennuksen yhteydessä määritellään tietokannan juurikäyttäjän käyttäjänimi sekä salasana. Tietoturvan kannalta tärkeää on antaa myös tietokantakohtaiset käyttäjätunnukset ja salasanat, varsinkin jos palvelimen etäkäyttö asetetaan sallituksi. Etäkäyttö vaatii ssh-yhteyden, mikä tavallisesti asennetaan jo käyttöjärjestelmän asennuksen yhteydessä. On huomioitava myös, että MySQL –konfigurointitiedostossa käyttöoikeudet rajataan oletuksena ainoastaan isäntäkoneeseen (localhost), joten tämä komentorivi on etäyhteyden mahdollistamiseksi ”kommentoitava ulos”.

Varsinainen relaatiotietokannan rakentaminen on suunniteltava huolellisesti ennen varsinaiseen ohjelmointiin ryhtymistä. Taulut, relaatiot, datatyypit sekä normalisointi on suunniteltava, jotta tietokannasta muodostuisi tehokkaasti toimiva kokonaisuus. Tietokannan on katettava kaikki järjestelmän tietokantakyselyiden edellyttämät tiedot. Hyvin suunnitellun tietokannan tunnusmerkkejä ovat myös selkeät tietorakenteet, joustavuus muutoksille sekä suorituskyky (Luimula 2010). AHCS-tietokannan suunnittelu aloitettiin laatimalla tietokannasta ER-kaavio (Entity-Relationship model) (LIITE 1). ER-kaavio on graafinen esitys tietokannan käsitteellisestä tietomallista (Ahola 2011). ER-kaavion kuvaamat käsitteet (Entity) ovat asioita, olioita tai tapahtumia joita tietokantaan halutaan tallentaa mukaan lukien käsitteiden väliset sidokset. Liitteessä 1. esitelty ER-kaavio noudattelee yleistä ER-kaavion rakennetta, joka erittelee käsitteet, relaatiot sekä käsitteiden ominaisuudet. ER-kaaviossa käistteet on kuvattu standardin mukaan neliöllä, käsitteiden ominaisuudet ellipsillä ja käsitteiden väliset suhteet ”salmiakkikuviolla”.

Kardinaalisuhteet on piirretty kaavioon käyttämällä merkkejä ”1” ja ”M” kuvaamaan käsitteiden suhdetta joko yksi-yhteen tai yksi moneen.

AHCS-tietokannassa esimerkkinä käsitteestä voidaan nostaa esiin vaikkapa käyttäjä, verensokerin mittaustulos tai sykkeen mittaustulos, joista käyttäjä on ns. vahva käsite, jonka olemassaolo ei riipu muiden käsitteiden olemassaolosta. Käsitteen tietosisältö koostuu alatason tieto-osioista (Attribute), jotka määrittävät ja identifioivat käsitteen. Käsitteessä ”käyttäjä” atribuutteja ovat esimerkiksi nimi sekä sosiaaliturvatunnus ja mittaustuloksissa puolestaan id, aikaleima sekä relaation määrittävä käyttäjä-id. Relaatiot eri käsitteiden välillä määritellään viiteavaimen (Foreign Key) avulla. Käsitteiden väliset relaatiot ovat olennaisessa roolissa relaatiotietokannan toiminnassa, sillä juuri ne auttavat vähentämään turhaa kahteen kertaan tallennettua dataa. Relaatiotyyppejä eli kardinaalisuhteita on olemassa kolmea tyyppiä: Yksi-yhteen, yksi-moneen sekä monta-moneen. Käyttäjä-käsitteen näkökulmasta relaatiot ovat yksi-moneen – yhdellä käyttäjällä voi olla monta mittaustulosta, mutta yhdellä mittaustuloksella voi olla relaatio ainoastaan yhteen käyttäjään.

Varsinaiseen käytännön rakennustyöhön on saatavilla runsaasti oppimateriaalia niin sähköisenä kuin painetussakin kirjallisuudessakin. Huolellisesti valmistellun ER-kaavion perusteella tuo työ oli lähinnä teknistä ohjelmointia. Tietokannan rakentamiseen käytetyt SQL-lauseet ovat nähtävissä liitteessä 2.

6.3 Tietokantakuvaus

AHCS-tietokannan tärkein käsite on käyttäjätiedot. Käyttäjätiedot –taulu sisältää automaattisesti generoituvan (auto increment) uniikin käyttäjä-id:n, henkilötunnuksen, etunimen, sukunimen, sukupuolen, käyttäjänimien, kryptatun salasanan, sähköpostiosoitteen, osoitteen sekä puhelinnumeron. Hyvin normalisoidussa tietokannassa käyttäjätiedot –käsitteestä voitaisiin vetää yksi-yhteen –tyyppinen relaatio vielä erilliseen potilastiedot –käsitteeseen, joka sisältäisi yksityiskohtaiset potilastiedot. Näin pitkälle viety ja monipuolinen tietokantarakenne kuitenkin rajattiin tämän opinnäytetyön ulkopuolelle.

Järjestelmätietokantaan on varattu omat taulut sekä verensokerin-, verenpaineen- että sykkeen mittaustietojen tallentamiseen. Taulut ovat rakenteeltaan identtiset: Merkinnän identifikaationumero, aikaleima, itse arvo sekä käyttäjä-id, jolla arvo sidotaan oikeaan

käyttäjään. Vastaavaa yksinkertaista taulurakennetta noudattaa myös ruokapäiväkirjan taustalla pyörivä taulu. Merkinnän identifikaatiotiedon sekä käyttäjärelaation lisäksi tauluun tallennetaan hiilihydraattimäärä, kalorimäärä, aikaleima sekä päiväkirjamerkinnän yhteydessä tallennettu käyttäjäviesti. Kuviossa 12. on kuvattu AHCS-tietokantarakenne jokaisen taulun osalta poikkeuksena ”arvot ” –taulu, joka on yhdistetty malli verensokeri-, syke- ja verenpainetaulun rakenteesta.

kayttaja

kayttaja_id	henkilotunnus	etunimi	sukunimi	sex	username	password	email	address	phone

syke

syke_id	timestamp	syke	huippu_syke	matka	nopeus	kayttaja_id	type

paivakirja

paivakirja_id	hiilihydraatti	kalori	tiedot	kayttaja_id	aika

verensokeriarvot

verensokeriarvo_id	timestamp	arvo	lat	lon	address	tiedot	kayttaja_id	type

categories

cat_id	cat_name	cat_description

topics

topic_id	topic_subject	topic_date	topic_cat	topic_by

posts

post_id	post_by	post_topic	post_content	post_date

KUVIO 12. Tietokantamalli

7 TIETOTURVA

Kuten aiemmin todettiin, järjestelmän hajauttaminen tuo mukanaan monia etuja. Kuitenkin järjestelmän eri osien kommunikoidessa TCP/IP-verkon ylitse, hajauttaminen altistaa järjestelmän monille ulkoisille uhille. Siksi tietoturvan käsittely kiinteänä osana hajautettuja järjestelmiä on näiltä osin välttämätöntä. Yleisesti tietoturvaan voidaan katsoa kuuluvaksi mm. järjestelmän turvaaminen oikeudettomalta käytöltä, järjestelmässä käsiteltävän tiedon luotettavuuden säilyttäminen sekä pääsykontrollin avulla ylläpidettävä tietosisällön suojaaminen (Lehtonen 2005, 2-4). Edellä mainitut tietoturvan periaatteet eivät ole yksiselitteisiä, sillä aihe on laaja ja sitä on lähestytty usean eri organisaation näkökulmasta. Tietoturvan voidaan ajatella jakaantuvan edelleen luottamuksellisuuteen, eheyteen, saatavuuteen, todentamiseen, pääsynvalvontaan sekä kiistämättömyyteen (Brax 2011a). Lisäksi tietoturvaratkaisuissa merkittävässä roolissa on tietojärjestelmän fyysiset turvaratkaisut ja ennen kaikkea henkilöstön riittävä koulutus, huolellisuus ja luotettavuus kiinteänä osana tietoturvallista järjestelmää. Tämän opinnäytetyön yhteydessä hajautettujen järjestelmien tietoturvan käsittely rajataan kattamaan sovellusohjelmoinnissa ja lähiverkon rakentamisessa käytettävissä olevia keinoja.

7.1 Palomuuri

Palomuurin rooli tämän päivän lähiverkossa on jo niin vakiintunut käsite, ettei sitä ole syytä lähteä tässä yhteydessä käsittelemään perusteellisesti. Todettakoon kuitenkin, että palomuurin tehtävä on valvoa lähiverkon ja internetin välistä liikennettä ja estää asiaton pääsy verkosta toiseen (FiCom 2012). Palomuuri suodattaa IP-paketteja ip-osoitteen, porttinumeron, käytettävän protokollan tai vaikkapa kellonajan perusteella. Tosin raskasta suoritustehoa vaativat ip-suodatus ja erilaiset pääsylistat (acces control list) ovat käytössä lähinnä raskaammissa laitteistopohjaisissa palomuureissa. Kotikoneiden ja pienten lähiverkkojen suojaus perustuu pitkälle kevyempien ohjelmallisten palomuurien käyttämiseen. Viimekädessä käyttäjä on itse vastuussa palomuurinsa asianmukaisesta konfiguroinnista niin, että palomuuri tarjoaa riittävän suojan web-hyökkäyksiä vastaan, mutta ei kuitenkaan häiritse järjestelmän toimivuutta. Lisäksi käyttäjällä on aina suuri vastuu tietoturvallisesta käyttäytymisestä lähiverkossaan. Jos käyttäjä huolimattomuuttaan tulee paljastaneeksi salasanansa tai käyttää heikkoja salasanoja, ei mikään palomuuri kykene tarjoamaan riittävää tietoturvaa. (Brax 2011b, 22-29.)

AHCS –järjestelmän simulointiympäristössä palomuuri sijoittuu TP-LINK TL-MR3220 –reitittimeen, joka toimii reunareitittimenä lähiverkon ja internetin välillä. Oletuksena kaikki portit ovat suljettuna – pop3(143), http(80), DNS(53) ja SMTP(25) ohjataan palvelinkoneeseen. Lisäksi internet-palveluja tarjoava yritys (ISP, Internet Service Provider) – simulointiympäristön tapauksessa TeleFinland – suodattaa tahollaan liikennettä omista perustelluista lähtökohdista. Internet palveluntarjoajan roolista sekä portinohjauksesta enemmän luvussa ”Testiympäristö”.

7.2 Salasanat

Älykkäistä tietoturvaratkaisuista huolimatta yksi suurimmista tietoturvaongelmista on käyttäjän huolimattomuus tai suoranainen laiskuus salasanojen suhteen. Ns. vahvojen salasanojen huolellinen käyttö ja riittävän tiheä vaihtosykli on koko järjestelmän tarkoituksenmukaisuuden kulmakivi, sillä parhainkaan järjestelmä ei pysty ehkäisemään vääriin käsiin joutuneen salasanan väärinkäyttöä. Bodgan Calin vuonna 2009 tekemän tutkimuksen mukaan vertailussa olleen 10000:n Hotmail-salasanan joukossa salasana ”123456” esiintyi 64 kertaa (Walzer 2012, 13). Kokonaan toinen luku on lukuisat salasanojen urkintaan ja varastamiseen erikoistuneet käytännöt, joita vastaan suojautumiseen mailman tietoyhteisö on varautunut mm. teknisten innovaatioiden, koulutuksen ja valistuksen keinoin.

AHCS vaatii käyttäjän kirjautuessa tältä vahvan salasanan, jossa esiintyy vähintään yksi iso kirjain, yksi pieni kirjain ja yksi numero. Muutoin salasanaa ei hyväksytä. Järjestelmässä edellä kuvattu toiminta on toteutettu PHP:n standardikirjastoon kuuluvalla ”preg_match()” –functiolla, joka ottaa parametrina rajatun ASCII –merkistöalueen sekä syötetyn salasanan, jonka sisältämiä merkkejä verrataan edellä määrätyihin spekseihin. Menetelmää kuvataan nimellä ”Regular Expressions”.

7.3 XSS-hyökkäys

XSS eli Cross-Site Scripting on hyökkäystapa, jossa hyökkääjä pyrkii syöttämään suojaamattomalle www-sivulle omaa koodia esimerkiksi palauteenantolomakkeen kautta. Koodi voi olla esimerkiksi JavaScriptiä, mutta helpoimmillaan HTML –merkkauskieltä, jossa esimerkiksi keskustelupalstalle syötetään hyökkääjän oma scripti. Tällä keinoin voidaan alkuperäiselle sivustolle integroida uutta sinne kuulumatonta sisältöä mm. salasanan urkintatarkoituksessa. AHCS:ssä XSS –hyökkäykset on pyritty estämään

niinikään PHP:n standardikirjastoon kuuluvalla strip_tags() –funktiolla, joka suodattaa kaikki HTML-tagit käsiteltäessä käyttäjän sisään syöttämää informaatiota. Toisaalta myös htmlspecialchars() on käyttökelpoinen funktio, jonka avulla HTML-merkit voidaan muuttaa entiteeteiksi (Walzer 2012, 41).

7.4 SQL-injektio

SQL-injectiota käytetään SSX-hyökkäyksen tavoin syöttämällä vahingollista informaatiota web-lomakkeen kautta kohteena olevaan järjestelmään. SQL-injektiossa erona on se, että varsinaisena kohteena on web-sivuston taustalla pyörivä SQL-tietokanta. Tietokannan toimintaa pyritään häiritsemään, tietoja poistamaan ja anastamaan – jopa uuden root-tyyppisen käyttäjän lisääminen on mahdollista huonosti suojatuissa järjestelmissä. SQL-injektiossa hyökkääjä syöttää web-lomakkeen kautta sql-kyselyjä pyrkien selvittämään tietokannan rakenteen, taulujen sekä tietueiden nimet jne. (Walzer 2012, 18).
AHCS:n suunnittelussa ja toteutuksessa on pyritty noudattamaan tietoturvallisen www-ohjelmoinnin periaatteita. SQL-injektion torjumiselle hyvän perustan luo tietoturvallinen hakemistorakenne. Järjestelmässä julkiseen kaikkien käytettävissä olevaan kansioon on sijoitettu ainoastaan ne tiedostot, joiden yleinen saatavuus on välttämätöntä järjestelmän toimivuuden kannalta. Esimerkiksi tietokantayhteyden avaava ja ylläpitävä skripti ja informaation syöttämisestä tietokantaan sekä tietokantakyselyt suorittavat skriptit on sijoitettava erilleen muista tiedostoista ja suojattava ulkopuolisilta hyökkäyksiltä. Julkisissa tiedostoissa löytyy ainoastaan viittaukset PHP-include –funktiolla edellä mainittuihin tiedostoihin.

Ohjelmointitasolla on tärkeää, että web-lomakkeelta vastaanotettava informaatio valitoidaan ennen tietokantaan syöttämistä. AHCS –toteutuksessa varmistetaan aina, että esimerkiksi numeerinen arvo on numeerinen tietotyyppi ja että syötteiden pituus on oikea. Lisäksi eräs tehokas SQL-injektioita ehkäisevä keino on käyttää mysql_real_escape_string –funktiota. Se lisää ”/” –merkit erikoismerkkien eteen. Menetelmää kutsutaan ”koodinvaihdoksi” ja sen tehtävänä on estää SQL-kyselyjen manipulointi.

Yleisellä tasolla voidaan todeta, että huolellisuus on korkein prioriteetti suojautumisessa niin SQL-injektiota kuin muitakin verkkohyökkäyksiä vastaan. AHCS:n ohjelmoinnissa on noudatettu seuraavaa turvallisen www-ohjelmoinnin proseduuria: tärkeän tiedon salaaminen, tietokantaan liittyminen minimaalisin käyttöoikeuksin, datan asianmukainen

validointi, parametrisoidut tietokantakyselyt sekä virhe- ja logiviestien rakentaminen niin, ettei niissä paljasteta tietokannan tai järjestelmän rakennetta (Mackay 2005).

7.5 Tiivistefunktiot

Tiivistefunktiot kuuluvat myös olennaisena osana tietoturvalliseen www-ohjelmointiin. AHCS:ssä PHP:n standardikirjastoon kuuluvaa md5() funktiota käytetään tallennettaessa salasanoja tietokantaan. Salasanojen tallentaminen sellaisenaan tietokantaan on selkeä turvallisuusriski ja niiden sijaan tietokantaan kuuluu tallentaa salasanojen tiivisteet. Käyttäjäsyöte syötetään parametrina md5() funktiolle ja funktio laskee syötteestä 32-merkkisen vakiopituisen merkkijonon, joka sisältää merkkejä alueelta 0-9 ja a-f. Vaikka käytetyimmistä tiivistefunktioista mm. juuri md5 ja sha-1 ovat kohtuullisen pienellä vaivalla murrettavissa, hankaloittaa niiden käyttö tietomurtoja monilta osin.

Md5 -tiivisteestä ei pitäisi kyetä päättelemään mitään alkuperiäisestä syötteestä eikä funktion tulisi tuottaa samaa tiivistettä useammalla syötetekstillä. Md5:n keksi aikoinaan Yhdysvaltalainen kryptografi Ronald Rivest, joka tunnetaan myös laajasta joukosta muita tiivistefunktioita kuten RC2, RC4 sekä RC5. (Teeriaho 2006, 1-7.)

7.6 Recaptcha

Web-hyökkäykset ovat usein automaattisesti generoituja tunkeutumisia, jotka suuntautuvat suureen joukkoon homogeenisesti toimivia laitteita. Haittaohjelmat etsivät suuresta massasta ennalta määrätyn proseduurin mukaisesti järjestelmän heikkouksia. Recaptcha on ohjelma, joka pystyy päättelemään onko käyttäjä ihminen vai tietokone ja pystyy siten ehkäisemään automaattisesti generoituja hyökkäyksiä. Recaptchan idea on varsin yksinkertainen: Ohjelma generoi sanoja, merkkijonoja tai taltioitua audiota. Käyttäjän on ainoastaan palautettava vastaanottamansa informaatio ja näin tämä on tunnistettu inhimilliseksi toimijaksi. Recaptchan käyttö on levinnyt varsin laajalle – vuonna 2012 se oli mukana käyttäjäautentikoinnissa 200 miljoonaa kertaa päivässä. (Google 2012.)

Recaptcha on maksuton sovellus ja monet web-alustat tarjoavat recaptcha-plug-ineja sovelluskehittäjien käyttöön. Toiminnallisuus on pienellä vaivalla rakennettavissa myös itse juuri oman sovelluksen yhteyteen sopivaksi. Näin toimittiin myös AHCS:n yhteydessä. Rekisteröitymiskaavakkeeseen liitettiin recaptcha-div, jonka kuvaparametrin taustalla php-scripti generoi satunnaislukugeneraattorilla kaksi samanpituista merkkijonoa ja tallensi

tämän ”SESSION” –muuttujaan. Käyttäjän jäljentäessä captchan tekstikenttään ja painaessaan lähetä-painiketta, javascript –skripti välittää captchan muun lomakkeelta saatavan informaation ohessa rekisteröimisestä huolehtivalle php-skriptille. Validoinnin yhteydessä verrataan captchaa ”SESSION” –muuttujan sisältöön ja näiden täsmätessä lomakkeen sisältö voidaan turvallisesti tallentaa tietokantaan.

7.7 Mobiililaitteiden tietoturva

Tähän mennessä tässä opinnäytetyössä päähuomio on keskittynyt pääasiassa web-sovelluksen ja tietokantayhteyden tietoturvaongelmiin sekä niiden ehkäisemiseen. Kuitenkin tätä opinnäytetyötä kirjoitettaessa hajautettujen järjestelmien, sosiaalisen median ja erilaisten pilvipalveluiden yleistyessä mobiililaitteiden tietoturvakysymykset ovat yhä merkittävämpi osa ajankohtaista tietoturvakeskustelua. Tietoturvan näkökulmasta tietokoneella ja älypuhelimella ei ole eroa kun kasvaneiden älypuhelinmarkkinoiden myötä laitteiden houkuttelevuus kasvaa hakkereiden kohteena. Hyvä esimerkki älypuhelimen haavoittuvuudesta on bluetooth. Bluetooth tarjoaa lupa- ja tunnistamismenettelyn, jonka ollessa käytössä luvaton pääsy laitteeseen on epätodennäköistä (Franklin & Layton 2011). Mutta huolimattomasti asetetut näkyvyysasetukset saattavat altistaa laitteen tarkoituksettomille parituspyynnöille. Tähän saakka matkapuhelinvirusten laajemman leviämisen on estänyt se, että käyttäjän on itse täytynyt hyväksyä viruksen asentuminen puhelinlaitteeseen. Uusi lukunsa on ”bluejacking” ja ”bluebugging” –tyyppiset hyökkäykset, jossa käyttäjän laite saadaan lähettämään käyntikortteja bluetoothin välityksellä lähiympäristöön ja pahaa-aavistamaton vastaanottaja hyväksyessään lähetyksen tulee asentaneeksi tiedon osoitekirjaansa ja sitä kautta aukaisseen portin jopa luvattomalle etäkäytölle. (Franklin & Layton 2011.) AHCS olettaa käyttäjän matkapuhelimessa bluetoothin olevan oletuksena poiskytkettynä. Sovelluksen käynnistyessä se tiedustelee bluetooth-yhteensopivuutta ja tämän jälkeen käyttäjän lupaa käynnistää bluetooth. Bluetooth –yhteydestä ja sen rakentamisesta android-sovellukseen enemmän luvussa ”Mobiilisovellus”

Matkapuhelinten virustorjuntaohjelmistot ovat tänä päivänä vielä kehitysaskeleen jäljessä verrattuna tietokonemaailmaan. Ohjelmistoja on toki tarjolla, mutta mikään riippumaton taho ei takaa niiden toimivuutta. Google Play ei ota mitään vastuuta android-kehittäjien yleiseen jakoon asettamista sovelluksista ja näin android-käyttäjäyhteisö on ainut consortio arvioimaan sovellusten – virustorjuntasovellukset mukaan lukien – luotettavuutta.

Windows Phone 7:n (WP7) kohdalla käyttöjärjestelmän valmistajalla on huomattavasti tiukempi ote sovelluspolitiikassa. Windows Phone Marketplace käy läpi kaikki sovellukset ennen yleiseen jakoon päästämistä.

8 TESTIYMPÄRISTÖ

Toimivan järjestelmän rakentamiseen liittyy aina testaus autenttisessa ympäristössä. AHCS rakennettiin todelliseen lähiverkkoon, todelliseen domainiin ja käyttäen todellisia järjestelmäkomponentteja. Ohjelmointityön alussa projektille laadittiin testiohjelma, jossa suunniteltiin järjestelmän testaaminen. Järjestelmän jokainen yksittäinen komponentti testattiin yksikkötestin periaatteiden mukaisesti. Siinä testaus kohdistuu olio –ja metoditasolle ja jokainen yksikkö testattiin osana kokonaisuutta, mikä helpottaa ylläpitovaiheessa tehtävien muutosten testausta (Tietojenkäsittelytieteen laitos 2011).
Yksikkötestauksessa noudatettiin ns. ”Lasikattoperiaatetta”, joka pohjaa riittävään ja kattavaan määrään erilaisia testitapauksia. Lasikattoperiaatteessa kriittinen kattavuus saavutetaan kun testissä jokainen lause tulee suoritetuksi vähintään kerran, jokainen ehto saa vähintään kerran kunkin yksittäisen tulosmahdollisuutensa, jokainen silmukka suoritetaanvähintään kerran, jokainen polku tulee suoritetuksi vähintään kerran ja jokainen muuttuja ja parametri saa arvokseen jokaisen mahdollisen arvon. (Darcey 2011.) Android –sovelluksen rakentaminen ja testaus on mahdollista suorittaa kokonaan Eclipse –kehitysympäristössä, johon on asennettu android-sdk. Junit on yleisin java-kielen yksikkötestauskehys ja se on integroitu myös Eclipse –kehitysympäristöön.

Integrointitestissä testataan yksitellen järjestelmän komponenttien integraatio toisiinsa nähden. AHCS:n tapauksessa voitiin testata erikseen puhelinohjelmiston ja sykemittarin yhteistoiminta, puhelimen ja web-rajapinnan toiminta, web-rajapinnan ja SQL-tietokannan toiminta sekä web-sovelluksen ja tietokantarajapinnan toiminta.

Järjestelmätestaus suoritettiin siis tätä opinnäytetyötä varten rakennetussa autenttisessa testiympäristössä. Testihenkilöiksi valittiin 5 henkilöä, joissa oli mukana sekä diabetesta sairastavia että täysin terveitä yksilöitä. Heidän tehtävänään oli kiinnittää huomiota järjestelmän toiminnallisuuteen, käyttöliittymään, opasteisiin, käyttäjädokumentaatioon, virhetilanteisiin jne.

8.1 WAMP

Monet avoimen lähdekoodin konsortiot mahdollistavat sovelluskehittäjille toimivat työkalut sovellus -ja järjestelmäkehitykseen. WAMP eli Windows-Apache-MySQL-PHP sisältää yhdessä ratkaisussa kaikki edeltävissä kappaleissa esitetyt järjestelmäkehityksen järeät työkalut täysin ilmaisena kokonaisuutena. Kokonaisuudesta on tarjolla myös XAMPP (Cros Platform) ja LAMP (Linux) –versiot. WAMP:n ansiosta kokonainen hajautettu järjestelmä web-palveluineen on mahdollista perustaa periaatteessa yhdelle kotitietokoneelle. On kuitenkin monia syitä (mm. tietoturvanäkökohdat) miksi WAMP ei sovellu pysyväksi ratkaisuksi. WAMP kehitettiin ennen kaikkea simulointi -ja testausympäristöksi palvelinsovellusten kehittämiseen. Lisäksi ei ole suositeltavaa omaksua toimintamallia, jossa kehityskokeilut tehdään suoraan web-palvelimelle ftp:n välityksellä. WAMP oli AHCS:n kehityksessä vertaansa vailla oleva työkalu: tiedostojen siirrettävyys, tietokannan rakentaminen sekä koodin testaus onnistui joustavasti selkeän ja loogisen järjestelmän ansiosta.

8.2 Avoin Lähdekoodi (Open Source)

Monopolia muistuttavat aikamme suuret tietotekniikkayhtiöt ovat olleet suurelta osin määrittämässä tietotekniikkakomponenttien ja ohjelmistojen hintakehitystä. Suuret yhtiöt ovat ostaneet pieniä kilpailijoita markkinoilta ja joko liittäneet näiden innovaatiot osaksi omia tuotteitaan tai toisaalta jättäneet huomattaviakin keksintöjä hyödyntämättä. Ohjelmistokehittäjällä on usein käytettävissä kaupallisista tuotteista suppeita kokeiluversioita, mutta lähes poikkeuksetta ne eivät kata läheskään kaikkia kehityksen kannalta välttämättömiä työkaluja. Avoimen lähdekoodin (Open Source) yhteenliittymien tarkoituksena on vapautua kaupallisuudesta, tuottaa ohjelmistoja ja kehitystyökaluja lähdekoodeineen vapaaseen levitykseen. Virallisen määritelmän mukaan avoin lähde kattaa keinot kehittää ja jakaa tietokoneohjelmistoja. Käyttäjä voi vapaasti käyttää, kopioida, jakaa sekä jatkokehittää yksi tunnetuimmista vapaan lähdekoodin projekteista on Linux –käyttöjärjestelmä. AHCS perustuu täysin avoimen lähdekoodin sovellusten ja kehitystyökalujen hyödyntämiseen. Linux Ubuntu-Server, Apache –web-palvelin ja MySQL –tietokanta ovat kaikkien vapaassa käytössä olevia avoimen lähdekoodin ohjelmistoja.

8.3 Testiverkko

AHCS:n rakentaminen toteutettiin pääasiassa siis avoimen lähdekoodin komponentteja hyödyntäen. Varsinaiset ohjelmistojen kehitysympäristöt (IDE) olivat Eclipse (android-puhelinsovellus) ja NotePad++ (web-sovellus). Kaupallisista tuotteista mukana olivat TP-LINK TL-MR3220 –reitittimen mukana tullut configurointi/palomuurisovellus sekä kehitystietokoneessa alustana toiminut Windows7 –käyttöjärjestelmä. Testiympäristön fyysiset laitteet olivat KotiPC windows –käyttöjärjestelmällä, edellä mainittu reititin, Palvelinkone Linux Ubuntu-Server käyttöjärjestelmällä sekä android 4.0 –käyttöjärjestelmällä varustettu Samsung Galaxy S2 –kosketusnäyttöpuhelin. Laitteet olivat kytketty kategorian 6 kierretyllä parikaapelilla. Järjestelmäkaavio on nähtävissä liitteessä 3.

8.3.1 Linux Ubuntu-Server

Unix-pohjaiset käyttöjärjestelmät levisivät 80-luvun taitteessa pääasiassa yliopistojen serveri- työasematietokoneisiin. Avoimen lähdekoodin filosofia oli alusta lähtien kantava voima ja unix sai nopeasti laajan huomion it-kehittäjien keskuudessa. Suomalainen Linux Torvalds täydensi amerikkalaisen FSF:n (Free Software Foundation) yksittäisistä kirjastoista kokoaman vapaan käyttöjärjestelmän kehittämällään käyttöjärjestelmän ytimellä, Linux-kernelillä. (Jones 2007.)

Linux jakaantuu käyttötarkoituksen ja kehittäjäyhteisön perusteella useisiin eri jakeluihin ja on saavuttanut laajan suosion luotettavuutta, notkeutta ja keveyttä vaativissa tehtävissä. Linux:n eri jakeluita käytetään niin maailman tehokkaimmissa supertietokoneissa, pienimmissä sulautetuissa järjestelmissä kuin esimerkiksi suuressa osassa web-palvelimista. Todellista Linux-palvelimien osuutta kokonaismäärästä on tosin vaikea arvioida, sillä Linux-järjestelmän maksuttomasta luonteesta johtuen niistä ei pidetä kattavaa rekisteriä.

Linux-serverin asennus asennusmediasta tyhjälle kovalevylle noudattaa periaatteessa yksinkertaista proseduuria. Asennuksen merkittävin vaihe on LAMP (Linux-Apache-MySQL-PHP) valitseminen. Muutoin asennus noudattaa tyypillistä Linux-käyttöjärjestelmän asennusta: Varsinaisen asennuksen jälkeen käyttöjärjestelmä on syytä päivittää sudo apt-get update/upgrade –komennoilla. Versiosta 9.04 lähtien Linuxissa on ollut tarjolla varsin toimiva graafinen käyttöliittymä.Vaikka monet web-palvelua

ylläpitävät toimenpiteet tapahtuvatkin komentokehotepohjaisesti, antaa graafisen käyttöliittymän asentaminen työskentelyyn tiettyä konkretiaa. Jotta web-palvelu olisi löydettävissä omasta lähiverkosta ja ennen kaikkea lähiverkon ulkopuolelta, on sille annettava staattinen ip-osoite. Ip-osoitteistuksesta enemmän luvussa ”Reunareitittimeltä palveluntarjoajalle”.

LAMP-palvelimen jokaisella komponentilla on oma tärkeä roolinsa: MySQL huolehtii relaatiotietokantajärjestelmästä, Apache web-palvelusta ja PHP (Hyppertext Preprocessor) skriptikielestä. Mitään sen suurempaa konfiguraatiota järjestelmä ei edellä mainittujen komponenttien suhteen tarvitse. Asennusvaiheessa käyttäjältä kysytään MySQL-juurikäyttäjän salasana sekä järjestelmän pääkäyttäjän käyttäjätunnus sekä salasana. Web-sivusto on tallennettu kansioon \var\www ja ip-osoitteeseen ja domain-nimeen osoitettu http/GET –request ohjautuu suoraan siellä sijaitsevaan index-tiedostoon. MySQL-tietokannan rakentaminen, rakenne ja toiminta kuvataan tarkemmin luvussa ”6 TIETOKANTA”.

AHCS –järjestelmässä palvelimelle asennettiin myös postipalvelintoiminnallisuus. Saatavilla on monia selainpohjaisia ja maksuttomia sähköpostipalveluja, mutta omaan domain-nimeen yhdistetty sähköpostiosoitteistus on tärkeä osa toimivan ja huolitellun järjestelmän julkisivua. Postipalvelimen asennusproseduuri Linuxissa on jakelukohtainen. Asentajan on huomioitava, että mm. konfigurointitiedostot asentuvat eri kansioihin käyttöjärjestelmän versiosta riippuen. Postipalvelimen asentamiseen löytyy paljon oppimateriaalia internetistä, joskin käyttäjän on osattava soveltaa lukemaansa tietoa juuri oman jakelun kanssa yhteensopivaksi.

AHCS –projektissa postipalvelimen asentaminen aloitettiin Postfix – SMTP palvelinohjelmalla. Sähköpostijärjestelmässä SMPT:n (Mail Transfer Agent) tehtävänä on välittää lähtevä posti päätelaitteelta postipalvelimelle ja postipalvelimelta edelleen eteenpäin omaan lähiverkkoon kuulumattomat viestit. SMTP –palvelin tekee suuria päätöksiä koskien sähköpostien reitittämistä sekä viestin välitysehtoja. (Syotec 2007.) SMPT:n hyvin tunnettu porttinumero on 25 ja tämä täytyy huomioida palomuurin asetuksissa sekä reunareitittimen porttiohjauksessa. Postfix –konfigurointitiedoston (/etc/postfix/main.cf) tärkeimmät tehtävät on liittää SMPT –palvelin haluttuun

nimiavaruuteen ja ohjata postit kayttäjien kotihakemistoihin, jotka sijaitsevat hakemistossa /etc/Maildir/new.

Postipalvelin tarvitsee toimiakseen myös toiminnon, joka noutaa saapuneet sähköpostit palvelimelta käyttäjän sähköpostin asiakasohjelmaan. Järjestelmässämme päädyttiin käyttämään hyväksi ja turvalliseksi osoittautunutta dovecot-ohjelmaa. Dovecotin konfigurointitiedostossa (/etc/dovecot/dovecot.conf) voidaan määrittää lukuisia turvallisuusasetuksia, valita käytettävä protokolla sekä valita esimerkiksi autentikointimekanismi. AHCS:ssä päädyttiin käyttämään imap-protokollaa. Imap-protokolla eroaa pop3 –protokollasta siinä mielessä, että käytettäessä imap:a postit ainoastaan kopioidaan palvelimelta asiakasohjelmaan ja siten säilyvät luettavissa useilta eri koneilta ja asiakasohjelmilta (client). Käytettäessä pop3 –protokollaa, asiakasohjelma vetää postit pysyvästi paikalliselle koneelle ja ne ovat käytettävissä ainoastaan kyseisessä kohteessa. (Syotec 2007.)

Vaikka DNS (Domain Name System) –nimipalvelinjärjestelmä on merkittävä osa sähköpostijärjestelmää, tyydytään tässä kohtaa toteamaan yleisellä tasolla sen tehtävän olevan domain-nimen muuntaminen IP-osoitteeksi. IP-osoitteet ovat hankalia numerosarjoja ihmisen muistettavaksi, joten internetin osoitteet muutetaan nimipalvelun avulla helpommin muistettavaan muotoon kuten mail@example.com. Julkisessa verkossa SMTP-palvelimet keskustelevat keskenään ja DNS-palvelin ratkaisee domain-nimen perusteella mihin sähköpostipalvelimeen mikäkin sähköposti kuuluu lähettää. AHCS Ubuntu-palvelimelle asennettiin laajasti käytetty bind9 –nimipalvelinohjelmisto (eHowstuff 2012). DNS- ja postipalvelimen ylläpitäjän velvollisuuksiin kuuluu varmistaa asianmukainen roskapostinsuodatus ja virustarkistus sekä toisaalta myös se ettei oma palvelin toimi roskapostin lähettäjänä. Markkinoilla on tähän tarkoitukseen runsaasti täysin ilmaisia palveluja, mutta tämä ulottuvuus on rajattu tämän opinnäytetyön ulkopuolelle.

8.3.2 Reunareitittimeltä palveluntarjoajalle

Järjestelmän kehittäjän näkökulmasta palvelun näkyminen omasta lähiverkosta ulospäin vaatii käytettävissä olevat työkalut huomioon ottaen muutamia erityistoimenpiteitä. Tavallisessa kiinteässä ADSL-laajakaistaliittymässä operaattori sallii ns. ulkoa päin tulevat yhteydenotot asiakkaidensa koneisiin. AHCS-testiympäristössä pääsyverkko perustui kuitenkin TeleFinland Oy:n tarjoamaan 3G –langattomaan laajakaistaliityntään, jossa

oletuksena palveluntarjoaja esti ulkoa päin tulevat palvelupyynnöt sulkemalla kaikki portit. Tähän lienee syynä osaksi turvallisuus ja osaksi myös operaattoreiden halu hallita verkkonsa rajallisen kapasiteetin käyttöä. Lukuisten lähestymisyritysten jälkeen operaattori suostui tarjoamaan palvelua, joka mahdollisti tarvittavien porttien (80 http, 143 imap, 25 SMPT, 53 DNS) avaamisen liikenteelle.

Jotta web-palvelu olisi näkyvillä internetistä käsin, sillä tulee olla staattinen IP-osoite. Pienten järjestelmien kohdalla se ei kuitenkaan ole käytännössä mahdollista, sillä tietokoneiden määrä suhteessa Ipv4 –osoiteavaruuden rajallisuuteen on kasvanut räjähdysmäisesti eikä omaa IP-osoitetta riitä kaikille tietokoneille. Internet-operaattori (ISP, Internet Service Operator) jakaa asiakkailleen väliaikaisia IP-osoitteita DHCP:n (Dynamic Host Conficuration Protocol) kautta, joten vaatimus staattisesta web-palvelun osoitteesta ei täyty automaattisesti. Markkinoilla on kuitenkin tähän tarkoitukseen niin kutsuttuja dynaamisia DNS-palveluita, jotka yhdistävät domain-nimen asiakkaan internet-operaattorilta saamaan väliaikaiseen IP-osoitteeseen. AHCS käytää NO-IP.com:n tarjoamaa palvelua, jossa palvelinkoneelle asennettu client-ohjelma päivittää määrätyin väliajoin internet-operaattorin tarjoaman väliaikaisen IP-osoitteen palveluntarjoajan tietokantaan. Dynaamisen DNS-palvelun tarjoaja yhdistää kaikki AHCS:n domain-nimeen (http://kalliopaja.zapto.org) tulevat palvelupyynnöt asikkaan sillä hetkellä käytössä olevaan IP-osoitteeseen ja siten yhteys web-palvelimeen tai sähköpostipalvelimeen voidaan muodostaa. Samaa domain-nimeä hyödyntää myös AHCS:n mobiilisovellus, joka lähettää http-pyynnöt samalla web-palvelimella sijaitsevaan rajapintaan.

Internet-operaattorin tarjoama väliaikainen IP-osoite on reunareitittimeen kytketyn 3G-modeemin osoite. Reunareititin välittää lähiverkon liikenteen internetiin yhden käytössä olevan julkisen osoitteen kautta mikä on mahdollista Network Address Translation (NAT) –osoitemuunnostekniikkan avulla. TP-LINK TL-MR3220-reunareitittimessä on ohjelmallinen palomuuri, joka sulkee oletuksena kaikki lähiverkkoon kohdistuvat pyynnöt ja on ohitettavissa ainoastaan tietyistä ennalta määritellyistä porteista. Reunareitittimeen voidaan ohjelmoida portinohjaus (port forwarding), joka välittää tiettyihin palveluihin kohdistuvat pyynnöt oikeaan lähiverkon koneeseen ja ennen kaikkea oikeaan palveluun. AHCS:ssä kaikki palvelut kuuntelevat niiden yleisesti tunnettuja oletusportteja, joten palvelupyyntöön ei tarvitse erikseen liittää palvelun porttinumeroa.

9 JOHTOPÄÄTÖKSET

9.1 Työn haasteet ja ongelmapaikat

Työn sujumista ja etenemistä voidaan pääosin pitää varsin sujuvana, mutta kuten aina laajan järjestelmän kehitystyössä eteen tulee myös erilaisia haasteita, joiden ratkaiseminen vaatii suuriakin ponnisteluja. AHCS –järjestelmän kohdalla suurin haaste oli ehdottomasti sen kompleksisuus: oli suuri taidollisen kasvun paikka kehittää itseään kaikissa niissä tekniikoissa, joiden hallitsemista laaja-alaisen järjestelmän rakentaminen vaati. Internetin valtakaudella tarjolla on kuitenkin lukuisia yhteisöjä ja tahoja, jotka tarjoavat täysin korvauksetta ohjeita, neuvoja sekä sovellettavissa olevia esimerkkejä, mikä tekee oppimisesta nopeaa ja vaivatonta.

Yksittäisistä haasteista nostaisin esiin Linux –ympäristössä toimivan postipalvelimen asennuksen ja konfiguroinnin. Linuxista tekee haasteellisen toimintaympäristön sen eri jakeluiden heterogenisyys. Windows –maailmasta tuleva saattaa kokea hämmentävänä eri versioiden välillä muuttuneet hakemistorakenteet ja jopa toimintalogiikat. Oman haasteen postipalvelimen asennukseen toi myös järjestelmän domain-nimen yhdistäminen (binding) vaihtuvaan IP-osoitteeseen.

Projektinhallinnollisesta näkökulmasta otettiin suuri riski koko opinnäytetyön idean perustuessa vahvasti verensokerimittarin bluetooth –yhteyden varaan. Vapailta markkinoilta laitteita oli saatavissa varsin niukasti. Entra Health Systems sekä Polyamp Glucose Device olivat ainoat varteenotettavat tekijät ja opinnäytetyöprojektin rajallisten taloudellisten resurssien myötä päädyttiin lopulta käyttämään Polyamp Polytell –bluetoothlähetintä. Se on ns. kolmannen osapuolen rakentama lähetin Suomessa laajalti käytössä oleviin Bayer Contour –mittareihin ja toimiessaan erittäin hyvä ratkaisu Suomen markkinoita ajatellen. Laitteen rakentajan tarjoama android-kirjasto ei kuitenkaan toiminut ja pitkäjänteisestä kommunikaatiosta ja vuorovaikutuksesta kehittäjäyritykseen huolimatta tämän projektin yhteydessä tuota laitetta ei saatu toimimaan. Bluetooth –toiminnallisuuden rajoittuessa ainoastaan sykkeenmittauksessa käytettyyn yhteyteen, opinnäytetyön strategiaa muutettiin painottumaan hieman enemmän geolokaation suuntaan. Sekään ei kuitenkaan ole huono suunta, sillä paikkatietopalvelut ovat yhä voimakkaammin tulleet osaksi tämän päivän mobiilipalveluita. Mikään ei kuitenkaan anna ymmärtää, etteikö tuota

verensokerimittaukseen suunnattua bluetooth-laitetta tultaisi jatkossa kehittämään, joten aihe tullee olemaan esillä jatkossakin ja tehtävälistalla suurella prioriteetilla.

Oman haasteensa loi myös järjestelmän testaamiseen käytetyn lähiverkon internet yhteys, joka toteutettiin 3G-modeemilla. Sisään päin tuleva yhteysnopeus Soneran verkossa oli varsin kiitettävää tasoa, mutta ulos päin menevä – mikä serveriä ylläpidettäessä on ensiarvoisen tärkeä – yhteys oli tarkoitukseen aivan liian hidas. Lisäksi vaati runsaasti selvitystyötä saada palveluntarjoajan oletuksena estämät portit avattua, mikä ei virallisesti pitäisi lähinnä turvallisuussyistä olla lainkaan mahdollista Soneran 3G-verkossa.

9.2 Onnistumiset ja kehitysnäkökohdat

Työhön tartuttaessa tiesin kirjoittamisen olevan eräs vahvuuksistani eikä ajatusten ja tapahtumien pukeminen kirjalliseen muotoon ole koskaan tuottanut minulle hankaluuksia. Tosin tiukkaan, tieteelliseen ja napakkaan asiatekstiin ei kuulu liika rönsyileminen, joten vaati itsekuria pysyä määrätietoisesti asiassa ja pyrkiä kertomaan vain ja ainoastaan oleellinen. Suuri riskinotto verensokerimittarin saatavuuden osalta olisi voinut päättyä myös epäonnisemmin, joten uudelleen projektiin ryhdyttäessä pyrkisin takuulla varmistamaan huolellisemmin jokaisen kriittisen osatekijän riskit ja todennäköisyydet konkretisoitua vahingoiksi. Ja vaikka tämän opinnäytetyön myötä olen saavuttanut uskomattoman määrän tietämystä, ammattitaitoa ja kokemusta juuri oman alan taidoissa, koen projektin olleen laajuudessaan hieman tarpeettoman korkealla ammattikorkeakoulutason opinnäytetyöksi.

Olen äärimmäisen onnellinen ja ylpeäkin itsestäni kirjoittaessani tässä vaiheessa jo opinnäytetyöni yhdeksättä lukua. Monista pieniä hankaluuksia ja viivästyksiä aiheuttaneista haasteista on selvitty periksiantamattomalla asenteella. Tuloksena on toimiva järjestelmä, joka kelpaa referenssiksi työnhakuun ja jopa pohjaksi kaupalliselle tuotteelle. Ajanhallinan suhteen projekti ei olisi voinut onnistua paremmin: Työ valmistui suurimmilta osin opintojen kolmannen ja neljännen vuoden välisenä kesänä vaikka tuona aikana olin kokopäiväisessä työsuhteessa. On tuottanut suurta henkistä tyydytystä opinnäytetyön aiheen osuessa täysin oman alan haasteisiin ja olen samalla kyennyt kehittämään taitojani tähtäimessä työllistyminen juuri oman alan tehtäviin.

9.3 Tavoitteet ja niiden toteutuminen

Diabetes sairautena on niin monisyinen kokonaisuus, ettei edes lääketieteen parissa työskentelevä ei-diabetikko kykene aina ymmärtämään mitä diabeksen sairastaminen kokonaisuudessa merkitsee. Vuosien kokemus diabeteksen sairastamisesta antoikin vakaan tietopohjan tartuttaessa insinöörin näkökulmasta haasteelliseen aihealueeseen. Tavoitteena oli luoda uudenlainen - ehkä jopa uusi sukupolvi kohdeyleisönä painottuen – järjestelmä, joka auttaisi ja motivoisi diabetikkoja terästämään omahoitoa, sillä tiukkenevat resurssit siirtävät yhä enemmän diabeteksen perushoidon vastuuta terveydenhuoltojärjestelmältä yksittäisille potilaille. Hoitohistorian kerääminen laajoina otoksina ja sitä kautta hoitopäätösten yleisvaikutusten tilastoiminen suuressa mittakaavassa tuskin onnistuisi nykyisillä hoitomenetelmillä, joten luodun kaltaisella järjestelmällä suurella todennäköisyydellä pystyttäisiin saamaan uutta tilastoitua ja yksilöityä tietoa sairaudesta. Lisäksi järjestelmään saatiin toteutettua lähes kaikki suunnitellut ominaisuudet, jotka auttavat moderneilla tavoilla kaiken ikäisten mahdollisuuksia saada oppia sairaudesta sekä sen hoitamisesta. Tämän päivän ihmiset ovat tottuneita tietokoneiden ja älypuhelimien käyttäjiä. Myös vanhempi sukupolvi on löytänyt nuorempien suosiollisella avustuksella informaatiotekniikan ja ovat keskimäärin innokkaita kokeilemaan ja myös ottamaan käyttöön uusia mahdollisuuksia. Tästä näkökulmasta kehittämäni järjestelmä syntyi suureen tilaukseen ja vastaa mielestäni varsin mallikaasti asetettuun tutkimusongelmaan.

9.4 Tulevaisuudennäkymät

Informaatioteknologian mahdollisuudet terveydenhuollossa tuntuvat tällä hetkellä mittaamattoman lupaavilta. Väestö ikääntyy, mutta samalla keskimääräinen elinikä nousee. Uudenlaisia päätelaitteita ja tekniikoita tulee markkinoille vuosittain. Kun terveydenhuollon resursseja pienennetään, korostuu potilaan oma aktiivisuus myös käytännän hoitotietojen ylläpitämisessä. Potilailla ei tähän saakka ole juuri ollut pääsyä terveydenhuollon järjestelmiin, mutta AHCS:n myötä terveydenhuollon virallinen järjestelmä tarjoaisi rajapinnan, jolla järjestelmä kohtaisi paremmin yksittäiset käyttäjät ja näin priorisoisi huomattavasti terveydenhuoltoon kohdistuvaa kuormaa.

Tätä opinnäytetyötä varten AHCS –järjestelmästä luotiin niinsanotusti runkoversio, jossa järjestelmälle rakennettiin toimivat ja tietoturvalliset raamit. Ajateltaessa tuotteen kaupallistamista, tehtävälistalla on lukuisia uusia ominaisuuksia, joita olisi hyvä saada mukaan; Monipuolisempia historiatietoja, perusteellinen ohje-osio videotallenteineen ja

ehkäpä järjestelmän sisäinen wiki, josta käyttäjä voi hakea tietoa indeksoidusta hakemistosta. Tulevaisuudessa järjestelmään voitaisiin liittää videopuheluominaisuus, jossa käyttäjä voi kommunikoida videon välityksellä lääkärinsä kanssa. Myös monet jo tässä vaiheessa järjestelmään mukaan saadut ominaisuudet kuten ruokapäiväkirja ja graafit ovat varsin pelkistettyjä versioita mahdollisesta lopullisesta tuotteesta. Opinnäytetyön rajauksen mukaan tyydyttiin ainoastaan osoittamaan erilaisten tekniikoiden suuri potentiaali tämän kaltaisten järjestelmien rakentamisessa.

9.5 Loppusanat

Matka oli pitkä ja osin kivinen. Periksiantamaton luonteenlaatu, oma kiinnostus aiheeseen sekä halu luoda näyttävä referenssi tulevaisuuteen antoivat voimia viedä tämä projekti loppuun saakka. Tätä kirjoitettaessa minulle on syntyt kattava kuva hajautettujen järjestelmien yksittäisten osien toiminnasta osana kokonaisuutta. Se on hyvä raamittava indeksilista ryhdyttäessä hiomaan henkilökohtaisia taitoja spesifeillä ohjelmoinnin osa-alueilla.

LÄHTEET

Aarne, M. 2010. Diabeteksen kustannukset Suomessa 1998-2007. Www-dokumentti. Saatavissa: http://www.diabetes.fi/files/1264/Kustannusraportti.pdf. Luettu 30.2.2013.

Ahola, H. 2011. Relaatiotietokannan suunnittelu ja toteutus. Opinnäytetyö. Keski-Pohjanmaan ammattikorkeakoulu. Mediatekniikan koulutusohjelma.

Android. 2012. Linux-wiki. Www-dokumentti. Saatavissa: http://linux.fi/wiki/Android. Luettu 18.7.2012.

Android developers. 2012a Bluetooth. Www-dokumentti. Saatavissa: http://developer.android.com/guide/topics/connectivity/bluetooth.html. Luettu 18.7.2012.

Android developers. 2012b Permissions. Www-dokumentti. Saatavissa: http://developer.android.com/guide/topics/security/permissions.html. Luettu 18.7.2012.

Asleson, R. & Nathaniel, T. 2007. Rakenna dynaamisia verkkosivuja. Kääntänyt Mikko Kamppila. Helsinki: Readme.fi.

Brax, V. 2011a. Tietokoneverkot-tuntimoniste. Pdf-dokumentti. Ylivieska: Keski-Pohjanmaan ammattikorkeakoulu.

Brax, V. 2011b. Tietoturva-tuntimoniste. Pdf-dokumentti. Ylivieska: Keski-Pohjanmaan ammattikorkeakoulu.

Coss ry. 2012. Avoin lähdekoodi. Www-dokumentti. Saatavilla: http://coss.fi/avoimuus/avoin-lahdekoodi/. Luettu 18.7.2012.

Darell, R. 2011. The Complete Android History Timeline. Www-dokumentti. Saatavissa: http://www.bitrebels.com/technology/the-complete-android-history-timeline-infographic/. Luettu 18.7.2012

Dediu, H. 2012. When will Android reach one billion users. Www-dokumentti. Saatavissa: http://www.asymco.com/2012/02/29/when-will-android-reach-one-billion-users/ Luettu 18.7.2012.

Developers.google.com. 2012. Google Maps Api. Www-dokumentti. Saatavissa: https://developers.google.com/maps/?hl=fi . Luettu 17.11.2012

Darcey, L. 2011. Android SDK: Unit Testing with the Junit Testing Framework. Www-dokumentti. Saatavissa: http://mobile.tutsplus.com/tutorials/android/android-sdk-junit-testing/. Luettu 18.7.2012.

eHowstuff. 2012. How to Install and Configure Bind9 DNS on Ubuntu 11.10. Www-dokumentti. Saatavilla: http://www.ehowstuff.com/how-to-install-and-configure-bind9-dns-on-ubuntu-11-10/. Luettu 18.7.2012.

Franklin, C. & Layton, J. 2011. How Bluetooth Works. Bluetooth Security. Www-dokumentti. Saatavissa: http://electronics.howstuffworks.com/bluetooth4.htm. Luettu 18.7.2012.

FiCom. 2012. Php-programming. FiCom Internet Palomuuri. Saatavissa: http://www.ficom.fi/tietoa/tietoa_4_1.html?Id=1057649869.html. Luettu 18.7.2012.

Google 2012. Recaptcha. What is reCaptcha. Www-dokumentti. Saatavissa: http://www.google.com/recaptcha/learnmore. Luettu 18.7.2012.

Ilanne-Parikka, P., Rönnemaa, T. & Saha, M. 2011. Diabetes. 7. Helsinki: Duodecim.

Jones, M. 2007. Anatomy of the Linux kernel. Www-dokumentti. Saatavilla: http://www.ibm.com/developerworks/linux/library/l-linux-kernel/. Luettu 18.7.2012.

Juen, G. 2012. Android Workshop The Basics. Pdf-dokumentti. Cambus Bocholt. Mestfäliche Hochschule.

Lehtonen, J. 2005. Hajautetun tietojärjestelmän tietoturva. Pro gradu –tutkielma. Turun yliopisto. Informaatioteknologian laitos. Tietojenkäsittelytiede.

Luimula, M. 2010. Tietokannan suunnittelu. Pdf-dokumentti. Ylivieska: Keski-Pohjanmaan ammattikorkeakoulu.

Mackay, C. 2005. SQL Injection Attacks and Some Tips on How to Prevent Them. Www-dokumentti. Saatavissa: http://www.codeproject.com/Articles/9378/SQL-Injection-Attacks-and-Some-Tips-on-How-to-Prev. Luettu 18.7.2012.

Manneri, T. 2009. Diabetikot voivat entistä paremmin. Www-dokumentti. Saatavissa: http://www.diabetes.fi/diabetesliitto/lehdet/diabetes-lehden_juttuarkisto/yleista_diabeteksesta/diabeetikot_voivat_entista_paremmin.html. Luettu 17.7.2012

Monodevelop.com. 2013. Project home page. Www-dokumentti. Saatavissa: http://monodevelop.com/Download/Mono_For_Android. Luettu 13.3.2013.

MySWL.com. 2012. Commercial home page. Www-dokumentti. Saatavissa: http://www.mysql.com/. Luettu 18.7.2012.

Php Manual. 2012. General Information. Www-dokumentti. Saatavissa: http://fi2.php.net/manual/en/faq.general.php. Luettu 18.7.2012.

Php-wikibooks. 2012. Php-programming. Www-dokumentti. Saatavissa: http://en.wikibooks.org/wiki/PHP_Programming. Luettu 18.7.2012.

Roberts, B. 2012. Flexible Box Model. Www-dokumentti. Saatavissa: http://thenewboston.org/watch.php?cat=43&number=14. Luettu 18.7.2012.

Sund, R. & Koski, S. 2009. Diabeteksen ja sen lisäsairauksien esiintyvyyden ja ilmaantuvuuden rekisteriperusteinen mittaaminen, Suomen Diabetesliitto, tekninen

raportti. Www-dokumentti. Saatavissa: http://www.diabetes.fi/files/274/FinDM_II._Diabeteksen_ja_sen_lisasairauksien_esiintyvyyden_ja_ilmaantuvuuden_rekisteriperusteinen_mittaaminen_Tekninen_raportti_pdf_361_kt.pdf. Luettu 17.7.2012.

Suomen diabetesliitto. 1998. Nuorten diabetesleiri, luentomuistiinpanot. 1.7.1998

Syotec. 2007. Linux Server Kommunikaatio. Www-dokumentti. Saatavilla: http://wiki.syotec.fi/index.php?title=JA210_-_Linux_Server_-_Kommunikaatio#contentSub. Luettu 18.7.2012.

Systä, K. 2012. Hajautettujen järjestelmien perusteet. Tampereen teknillinen yliopisto. Www-dokumentti. Saatavissa: http://www.cs.tut.fi/~hajap/luennot/. Luettu 18.7.2012

Teeriaho, J. 2006. Salausmenetelmät. Www-dokumentti. Saatavissa: http://ta.ramk.fi/~jouko.teeriaho/krypto2006/salausmenetelmat2_7tiivisteetMACitallekirjoitus.pdf. Luettu 18.7.2012.

Tietojenkäsittelytieteen laitos. 2011. Yksikkötestaus. Www-dokumentti. Saatavissa: http://www.cs.helsinki.fi/u/avihavai/edutainment/2011/ohma/7-yksikkotestaamisesta.pdf. Luettu 18.7.2012.

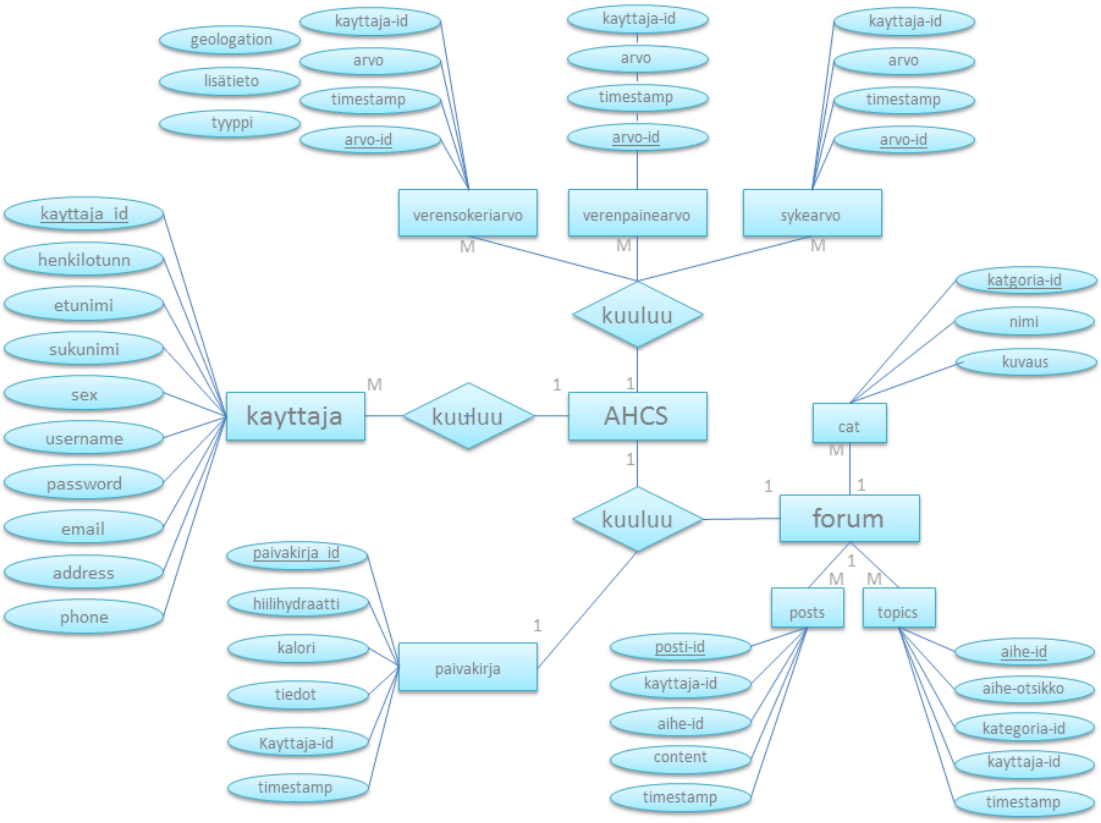
Walzer, J. 2012. Tietoturvallinen www-ohjelmointi. Opinnäytetyö. Tampereen ammattikorkeakoulu. Tietojenkäsittelyn koulutusohjelma.

w3schools.com. 2012 a. CSS Introduction. Www-dokumentti. Saatavissa: http://www.w3schools.com/css/css_intro.asp. Luettu 18.7.2012.

w3schools.com. 2012 b. HTML5 Introduction. Www-dokumentti. Saatavissa: http://www.w3schools.com/html5/html5_intro.asp. Luettu 18.7.2012.

LIITTEET

LIITE 1. TIETOKANNAN ER-KAAVIO



LIITE 2. TIETOKANNAN SQL-LAUSEET

```
CREATE TABLE kayttaja(
kayttaja_id             MEDIUMINT NOT NULL AUTO_INCREMENT,
henkilotunnus           VARCHAR(32) NOT NULL UNIQUE,
etunimi                 VARCHAR(32) NOT NULL,
sukunimi                VARCHAR(32) NOT NULL,
sex                     TINYINT(1)  NOT NULL,
username                VARCHAR(64) NOT NULL UNIQUE,
password                VARCHAR(64) NOT NULL UNIQUE,
email                   VARCHAR(64),
address                 VARCHAR(64) NOT NULL,
phone                   VARCHAR(64) NOT NULL,
PRIMARY KEY             (kayttaja_id)
)
CHARACTER SET utf8 COLLATE utf8_general_ci;
```

```
CREATE TABLE syke(
syke_id                 MEDIUMINT NOT NULL AUTO_INCREMENT,
timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
syke                    INTEGER UNSIGNED NOT NULL,
huippusyke  INTEGER UNSIGNED NOT NULL,
matka                   INTEGER UNSIGNED,
nopeus                  DECIMAL (10,1) UNSIGNED,
kayttaja_id  MEDIUMINT NOT NULL,
type                    VARCHAR(30) NOT NULL DEFAULT 'lepo',

CONSTRAINT              arvot_cons
                        PRIMARY KEY (syke_id),
CONSTRAINT              kayttaja_cons
                        FOREIGN KEY (kayttaja_id)
                        REFERENCES kayttaja (kayttaja_id)
);
```

```
CREATE TABLE Verensokeriarvot(
Verensokeriarvot_id     MEDIUMINT NOT NULL AUTO_INCREMENT,
timestamp               TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP,
arvo                            DECIMAL (10,2) UNSIGNED NOT NULL,
kayttaja_id             MEDIUMINT NOT NULL,
lat                             FLOAT(10, 6) DEFAULT '64.071067',
lon                             FLOAT(10, 6) DEFAULT '24.533049',
address                         VARCHAR(80),
tiedot                          LONGTEXT,
type                            VARCHAR(30) NOT NULL DEFAULT
'verensokeri',

CONSTRAINT                      arvot_cons
                                PRIMARY KEY (Verensokeriarvot_id),
CONSTRAINT                      kayttaja_cons
                                FOREIGN KEY (kayttaja_id)
                                REFERENCES kayttaja (kayttaja_id)
);

CREATE TABLE paivakirja(
paivakirja_id           MEDIUMINT NOT NULL AUTO_INCREMENT,
hiilihydraatti          DECIMAL (10,2) UNSIGNED,
kalori                  DECIMAL (10,2) UNSIGNED,
tiedot                  LONGTEXT,
kayttaja_id             MEDIUMINT NOT NULL,
aika                    DATETIME,

PRIMARY KEY             (paivakirja_id),

CONSTRAINT              paivakirja_cons
                        FOREIGN KEY (kayttaja_id)
                        REFERENCES kayttaja (kayttaja_id)
)
CHARACTER SET utf8 COLLATE utf8_general_ci;


CREATE TABLE categories(
cat_id                  MEDIUMINT NOT NULL AUTO_INCREMENT,
cat_name                VARCHAR(32) NOT NULL,
cat_description         VARCHAR(50) NOT NULL,
//kayttaja_id           MEDIUMINT NOT NULL,

CONSTRAINT              cat_cons
                        PRIMARY KEY (cat_id),
);
```

```
CREATE TABLE topics(
topic_id                TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
topic_subject           VARCHAR(50) NOT NULL,
topic_date              TIMESTAMP NOT NULL,
topic_cat               VARCHAR(32) NOT NULL,
topic_by                MEDIUMINT NOT NULL,
PRIMARY KEY(topic_id),

CONSTRAINT              kayttaja_cons
                        FOREIGN KEY (topic_by)
                        REFERENCES kayttaja (kayttaja_id)
);


CREATE TABLE posts(
post_id                 INT UNSIGNED NOT NULL AUTO_INCREMENT,
post_by                 MEDIUMINT NOT NULL,
post_topic              TINYINT UNSIGNED NOT NULL,
post_content            LONGTEXT NOT NULL,
post_date               TIMESTAMP NOT NULL,

PRIMARY KEY(post_id),
CONSTRAINT              post_cons
                        FOREIGN KEY (post_by)
                        REFERENCES kayttaja (kayttaja_id)

CONSTRAINT              topic_cons
                        FOREIGN KEY (post_topic)
                        REFERENCES topics (topic_id)
);
```

LIITE 3. JÄRJESTELMÄKAAVIO