Riku Ketonen

VARASTOSOVELLUKSEN SUUNNITTELU MOBIILILAITTEILLE

Tietotekniikan insinöörilinja

2016

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

VARASTOSOVELLUKSEN SUUNNITTELU MOBIILILAITTEILLE

Ketonen, Riku
Satakunnan ammattikorkeakoulu
Tietotekniikan insinöörilinja, tietoverkot
Helmikuu 2016
Ohjaaja: Auramo, Yrjö
Sivumäärä: 65

Asiasanat: mobiililaite, sovellus, suunnittelu, varasto

Opinnäyteyön aiheena oli Mobiililaitteilla toimivan sovelluksen, SatVaraston, suunnittelu varaston seurantaan ja hallintaan. SatVarasto on jaoteltu eri taulukoihin, jotka ovat Asiakas, Osasto, Varasto, Hylly ja Tuote. Jokaisella taulukolla on omat parametrinsä, joista muut taulukot saavat tarkennusta, luoden kattavan varastojärjestelmän. Tilaajalla oli omat vaatimuksensa sovellukselle, jotka käydään läpi tässä työssä ongelmineen ja toteutussuunnitelmineen. Suunnittelussa pyrittiin pitämään käyttäjäriippuvuus minimissä, josta syystä eri taulukot riippuvat toisistaan.

Käyttöliittymäksi valittiin pitkän suunnittelun päätteeksi HTML5:lla toteutettu nettisivu, joka mahdollistaa käytön mobiililaitteilla, käyttää sisäänrakennettua kameraa sekä tiivistää käyttäjä-serveri välisen yhteyden muutamalle koodiriville.

Lopputuloksena on suunnitelma sovelluksesta, joka on räätälöity tarkkaan asiakkaan tarpeisiin. Tämän suunnittelun pohjalta ei sovellusta voi vielä tuotteistaa ja markkinoida, sillä työn rajoittamiseksi tiettyjä osa-alueita, kuten tietoturva, on jätetty taka-alalle.

DESIGNING A WAREHOUSE APPLICATION FOR MOBILE DEVICES

Ketonen, Riku
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Bachelor of Information Technology, Network Engineering
February 2016
Supervisor: Auramo, Yrjö
Number of pages: 65

Keywords: mobile-friendly, software, design, warehouse

The purpose of this thesis was to design mobile-friendly warehouse software to track and control a warehouse with specific needs. The software is divided in different layers, which are Customer, Unit, Warehouse, Shelf and Product. Each layer has its own parameters that are inherited on the lower layers, creating a very thorough warehouse index. The customer had its own requirements for the software, which are looked into in this thesis, with the designs and problems. The design was to keep the user interaction at minimum, which is the reason for many of the inherited parameters.

The user interface, after a long study, was chosen to be a webpage created in HTML5, which makes it possible to use the system on mobile devices, use the devices' built-in camera and the straight-forward User-Server interaction that is made possible in just a few lines of code.

The finished product was a design carefully tailored to match the customer's needs. Creating a finalized product from the design on this thesis, however, is not possible due to some limitations set by the nature of the thesis. The finalized product will have its own cyber security system in place to protect the customer.

SISÄLLYS

LYHENTEET JA TERMIT

Työtä luettaessa tulee osata muutamia alakohtaisia sanoja ja termejä, jotka on pyritty määrittelemään alla olevassa luettelossa. Vuo- ja sekvenssikaaviot kulkevat aina ylhäältä alas, vasemmalta oikealle, ellei kaaviossa päädytä tapahtumaan, jolla on kaksi mahdollista tuloa (Kyllä tai Ei). Tällöin kaaviota seurataan tapahtuman tulosta riippuvaan suuntaan, joka on aina merkitty

Sovellus
Sovellus tai sovellusohjelma on tietokoneohjelma, joka on suunniteltu ratkomaan tiettyjä ongelmia tai helpottamaan tiettyjä tehtäviä. Sovellus voi olla verkosta ladattava tai ostettava valmis tuote, tai asiakkaan tilaama ja määrittelemä tilaustuote.

Tietokanta
Yksi, tai useampia taulukoita, joita sovelluksella ylläpidetään. Taulukot sisältävät koko varaston tiedot digitaalisessa muodossa.

Prosessi
Prosessi on sarja suoritettavia toimenpiteitä, jotka tuottavat määritellyn lopputuloksen. Prosessissa tapahtumat ja suoritteet toistuvat samankaltaisina. Sisältää käyttäjän antamia arvoja, joiden perusteella prosessin lopputulos muuttuu tietyin rajaehdoin.

Ennalta Määritelty Prosessi
Prosessi, johon käyttäjällä ei ole vaikutus- tai muutosvaltaa. Arvot ovat määritelty sovellusta suunniteltaessa ja niiden muuttaminen vaatii sovelluksen avaamista koodita-solla.

Palvelin
Palvelin, arkisesti serveri tai servu, on laite joka ylläpitää sovellusta, tietokantaa sekä prosesseja. Tarkoitus ylläpitää erinäisiä palveluita mahdollisimman vikasietoisina.

IP-osoite
Numeerinen osoite, jonka perusteella laitteet ottavat yhteyden toisiinsa. Esimerkiksi IPv4 192.168.100.1 sekä IPv6 FE80::0202:B3FF:FE1E:8329.

Attribuutti

Määrite, jolla tarkennetaan jonkin yksikön tietoja tai ominaisuuksia.

Käyttöliittymä

Yhteys käyttäjän ja sovelluksen välillä. Käyttäjä näkee sovelluksesta vain käyttöliittymän, jonka kautta sovelluksen kanssa viestiminen on mahdollista.

Tasot

Varastoon liittyviä taulukoita. Esimerkiksi Osasto, Varasto, Hylly sekä Tuote.

Mobiililaite

Esimerkiksi älypuhelimet, joissa on kamera.

Optinen koodi

Viiva- tai QR-koodi, jota käyttämällä hyllyjä sekä tuotteita yksilöidään.

Olio

Perusyksikkö, joka sisältää loogisesti yhteenkuuluvaa tietoa ja toiminnallisuutta. Voivat kommunikoida keskenään lähettämällä ja vastaanottamalla viestejä.

HTTPS

Hypertext Transfer Protocol Secure (HTTPS) on protokolla, jota käytetään turvattujen yhteyksien luomiseen Internetissä. Toiminta perustuu HTTP-yhteyden salaamiseen ja lähettämiseen TLS tai SSL –protokollien yli.

String

Merkkijono (String) on järjestetty jono peräkkäisiä merkkejä. Mikä tahansa sarja merkkejä, myös lukuarvot, voidaan käsitellä merkkijonona. Tällöin lukuarvon käsittely matemaattisesti on mahdotonta ennen sen muuttamista lukuarvomuotoon.

Integer (Kokonaisluku)

Lukuarvo, jolla ei ole desimaaleja. Laitteiston arkkitehtuurista riippuen 32- tai 64-bittiset rajat lukuarvon koolla.

Float (Liukuluku, reaaliluku)
Lukuarvo desimaaleilla. Katso edellinen.

HTML5
Yleisnimitys monille nykyaikaisille web-tekniikoille. Julkaistu virallisesti 28. lokakuuta 2014. W3C:n suositus nettipalveluiden luomiselle
(W3-organisaation www-sivut 2016).

PHP
Komentosarjakieli, jossa koodi tulkitaan vasta ohjelman suoritusvaiheessa.
Käytetään eritysesti Web-palvelinympäristössä dynaamisten sivustojen luonnissa.

CSS
Cascading Style Sheets, jolla määritellään yhtenäiset säännöt sivuston ulkonäölle ja tekstin käsittelylle.

JavaScript
Komentosarjakieli, jonka tarkoitus on lisätä Web-sivuille dynaamista toiminnallisuutta. Käytetään myös pelien kehityksessä sekä useiden työpöytä- ja mobiilisovellusten luomisessa.

Wrapper
Tässä työssä käytettynä funktio, jonka tarkoituksena on kutsua toista funktiota ja kerätä yhteen data käyttäjän antamista tiedoista.

WebRTC
Web Real-Time Communication. Ohjelmointirajapinta, jonka tarkoituksena on mahdollistaa reaaliaikaiset yhteydet esimerkiksi ääni- ja videopuheluille ilman selainlaajennoksia.

Selain
Laitteen sovellus, jota käytetään Web-sivujen näyttämiseen ja käyttämiseen. Esimerkiksi Google Chrome, Internet Explorer, Microsoft Edge sekä Firefox.

Osasto

Yksi yrityksen monista mahdollisista palvelukokonaisuuksista. Esimerkkinä leikkausosasto, lääkintälaitehuolto tai teho-osasto.

Varasto

Yksittäinen varastohuone osaston alaisuudessa. Esimerkkinä hengityslaitevarasto, sähkötarvikevarasto.

Hylly

Yksittäinen taso, johon tuotteet sijoitetaan. Käyttäjän toiveiden mukaisesti kyseessä voi myös olla kokonainen varastokaappi, jos se sisältää tuotteita saman teeman mukaisesti. Esimerkkinä happikennotaso, paristokaappi. Pyritään pitämään tuotteet omilla hyllyillään, sen sijaan, että käytetään sekatavarahyllyjä, jolta löytyy kaikenlaista. Useita fyysisiä tasoja on kuitenkin mahdollista sitoa saman hyllykoodin alle, jolloin ainoa muuttunut asia on fyysisen tilan käyttö.

Tuote

Artikkeli, joita yksittäinen hyllytaso tai varastokaappi sisältää. Koska jokaista tuotetta ei välttämättä ole valmistajan toimesta viivakoodattu yksittäin, on mahdollista lisätä tuotepaketteja hyllyille. Näistä esimerkkinä 20 pariston laatikko, jonka sisältämät 20 paristoa eivät saa omia viivakoodejaan. Tällöin sovelluksen tietoihin lisätään yhden paketin viivakoodi, mutta määrään tulee 20 kappaletta. Jos useita paketteja lisätään samalle hyllylle saman viivakoodin alle, tulee kappalemäärää kasvattaa oikeassa suhteessa.

1 JOHDANTO

Työn tarkoituksena oli suunnitella sovellus, joka helpottaisi yrityksen varaston ylläpitämistä. Sovellukselle asetettiin vaatimuksiksi nopeakäyttöisyys useilla eri mobiilialustoilla.

Satakunnan sairaanhoitopiiri (SatSHP) on Satakunnan maakunnissa toimiva sairaanhoitopiiri, joka tarjoaa erikoissairaanhoitoa Satakunnassa. SatSHP:n sairaaloihin kuuluvat Satakunnan keskussairaala, Rauman aluesairaala, Harjavallan sairaala, Satalinnan sairaala sekä psykiatrian toimipisteitä useilla eri paikkakunnilla. Lisäksi kehitysvammaisten erityishuoltoon keskittyy muunmuuassa Ulvilasta löytyvä Antinkartanon kuntoutuskeskus.

Sairaalan jokaisella osastolla on oma varastonsa, jossa pidetään artikkeleita erinäisiin tarkoituksiin (huollon varaosat, leikkausosaston happilaitteet). Artikkeleita käyttävät kaikki osastoilla työskentelevät, mutta vastuu varastosta on usein yhdellä tai kahdella työntekijällä.

Osastojen varastoissa on useita eri artikkeleita, joiden määrästä ja ostopaikasta on hankala pysyä perillä. Lisäksi joillain artikkeleilla on viimeinen käyttöpäivämäärä, joka hankaloittaa tilannetta entisestään. Näiden artikkelien varastointi ja seuranta muodostavat ylimääräistä työtä jokaiselle, joka on vastuussa artikkelien tilaamisesta tai seurannasta. Esimerkiksi happikennot ovat rajallisia käyttöpäivämääriltään, eikä niiden tulisi unohtua varastojen perälle.

Sovelluksen tarkoitus on helpottaa varaston ylläpitoa, antaen reaaliaikaista tietoa varaston tilanteesta sekä mahdollistaen nopeat muutokset inventaariossa. Sovelluksen on oltava nopea sekä aina mukana, eikä se saa hidastaa päivittäistä työntekoa niin paljoa, että varaston manuaalinen ylläpito olisi kannattavampaa. Valmiilla sovelluksella varastojen nurkkiin unohtuneet happikennot tai hapertuvat kumiletkut eivät ole tulevaisuudessa ongelma, sillä varastojen tarkka seuraaminen on mahdollista.

2 KEHITYSPROSESSI

2.1 Kehitysprosessi

Kehitysprosesseja on useita erilaisia. Näistä esimerkkeinä toimivat Vesiputousmalli, Prototyyppimenetelmä sekä RUP (Rational Unified Process).

2.1.1 Prototyyppimenetelmä

Prototyyppimenetelmässä ohjelman kehittäminen etenee spiraalimaisesti. Prototyyppimallissa ulkoasu (käyttöliittymä) rakennetaan ensin ja vasta sitten spiraalimaisesti laajenevin kehin tuotteen liiketoiminnallinen kerros (bisneslogiikka) sekä tietokantakerros. Menetelmän etuna asiakas saa nopeasti nähtäväksi tuotteen lopullisen ulkoasun. Asiakas voi jo tässä vaiheessa kertoa, jos kehittelijät ovat toteuttaneet jotain toisin kuin hän on halunnut. Mahdolliset muutokset on vielä helppo ja kustannuksiltaan edullista ja työmäärältään vähäistä tehdä tässä vaiheessa. Prototyyppimenetelmä on hyvä myös silloin kun (asiakkaan) vaatimukset muuttuvat usein. (Wikipedia-yhteisön www-sivut, 2016)

2.1.2 RUP (Rational Unified Process)

Unified Process ei ole itsenäinen prosessi vaan laajennettava kehys, joka tulee muokata vastaamaan organisaation tai projektin erityistarpeita. Rational:in yhtenäistetty prosessi eli RUP (Rational Unified Process, nimi juontuu prosessikuvauksen luoneen Rational Software Corporationin nimestä) on samalla tavoin muokattava kehys. Joskus muokatusta prosessikuvauksesta on vaikea sanoa, onko se peräisin UP:stä vai RUP:sta, joten termejä käytetäänkin usein sekaisin. (Wikipedia-yhteisön www-sivut, 2016)

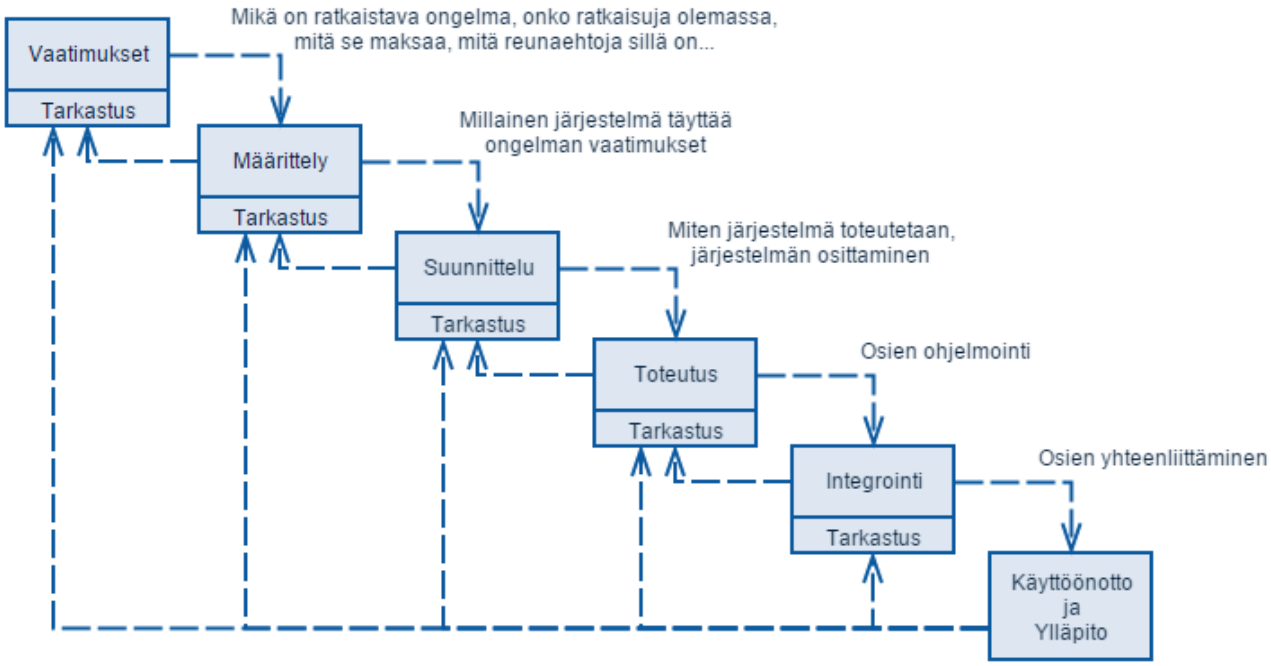
2.1.3 Vesiputousmalli

Työn luonteesta johtuen vesiputousmallin käyttö tuntui parhaalta ratkaisulta, sillä muut vaihejakomallit sisälsivät enemmän toteutusta ja prototyyppejä, kuin suunnittelua. Vesiputousmalli määritellään seuraavasti:

Vesiputousmalli on ohjelmistoprojektin vaihemalleista yksinkertaisin. Tässä vaihemallissa suunnittelu- ja toteutusprosessi etenee portaittaisesti vaiheesta toiseen aina projektin aloituksesta sen elinkaaren loppuun. Jokaisen askeleen lopussa tarkastellaan valmiutta siirtyä seuraavaan vaiheeseen. Ennen seuraavaan vaiheeseen siirtymistä tapahtuva katselmointi on tärkeää, sillä useissa projekteissa mallia sovelletaan pelkästään lineaarisena, jolloin mahdollisuutta palata edellisiin vaiheisiin ei ole. Vesiputousmallissa testaukselle varataan vain yksi vaihe. Testaus suoritetaan vasta projektin lopussa, jolloin virheiden korjauskustannukset ovat huomattavasti suuremmat kuin määrittely- tai suunnitteluvaiheessa. (Tauriainen 2005, 9-10.)

2.2 Opinnäytetyön kehitysprosessi

Ohjelmistotuotannossa kehitysprosessi kulkee suurin piirtein seuraavaa mallia noudattaen: Vaatimukset → Määrittely → Suunnittelu → Toteutus → Integrointi → Käyttöönotto ja ylläpito. Jokaisessa vaiheessa on syytä suorittaa jälkitarkastus, sillä virheet aiemmilla tasoilla vaikeuttavat toimintaa seuraavassa.



KUVA 1. Esimerkki vesiputousmallista (Haikala & Märijärvi 2006, 36)

Jokaisessa työvaiheessa luodaan useita dokumentteja, jotka helpottavat työn jatkamista. Vaatimuksiin kuuluu konkreettisten vaatimusten lisäksi vaatimusanalyysi, joka johtaa määrittelyihin. Määrittelyssä työn reunaehtoja ja toimintoja rajataan kappaleiksi, jotka suunnitellaan yksi kerrallaan omiksi valmiiksi kokonaisuuksiksi. Määrittelyssä voidaan jäädä sisäisen logiikan tasolle, tai syventyä koodiriveihin asti. Koodiriveistä siirtyminen toteutukseen on ongelmatonta, kunhan jokainen aiempi työvaihe on tarkastettu huolellisesti. Prototyypin kappaleiden integrointi, eli yhteenliittäminen, on viimeinen työvaihe ennen valmiin työn käyttöönottoa ja ylläpitoa. Tässä vaiheessa tuote on siirretty asiakkaalle ja mahdollisten muutosten ja korjausten toimittaminen ylläpidon muodossa alkaa.

2.3 Työhön käytetty kehitysprosessi ja dokumentointi

Dokumentointi on osa ohjelmistotuotantoa. Valitettavasti se hyvin usein jää tekemättä. Usein ohjelmiston suunnittelussa on kiireinen aikataulu. Dokumentointia ei pidetä niin tärkeänä, vaan aika käytetään mieluummin tuotteen koodaamiseen. Tosiasia on kuitenkin, että hyvin ja tarkasti tehty dokumentointi helpottaa tulevaisuudessa ohjelmiston päivittämistä uudempaan versioon. Toiminnallinen dokumentointi luo hyvän pohjan myös käyttöohjeiden tekemiselle. (Määttä, T & Rautio, P. 2010, 13)

Työtä tehdessäni dokumentoin ensimmäiseksi tarvittavat tiedot, kuten asiakkaan sekä tietokannan vaatimukset. Näitä suunnitellessa otin muistiin asioita, jotka tulivat eteen jokaista moduulia ja rajapintaa pohdittaessa.

Dokumentoinnin tuli kattaa tärkeimmät aiheet, jotka on kuvattu työssä. Näistä esimerkkinä ovat määritelmät, termit ja lyhenteet, sovellusalueen kuvaus, laitteisto- ja ohjelmistoympäristön kuvaus, tietokanta-arkkitehtuuri, ohjelmistoarkkitehtuuri sekä uudelleenkäytettävät komponentit. Kustakin moduulista ja prosessista esitetään yleiskuvaus, attribuutit, operaatiot sekä ohjeita toteutusta varten. Muita aihepiirejä olivat ylläpito-ohjeet, siirrettävyys, erityiset tekniset ratkaisut sekä hylättyjä ratkaisuvaihtoehtoja.

3 VAATIMUSANALYYSI

3.1 Mitä vaatimusanalyysi on?

Vaatimusanalyysin tavoitteena on selvittää ohjelmiston vaatimukset niin yksityiskohtaisesti, että niiden perusteella on mahdollista tuottaa valmis prototyyppi. Vaatimusanalyysi sisältää kakso osaa: vaatimusten kartoitus sekä analysointi. Kartoituksen lopputuloksena saatava vaatimuslista sisältää asiakkaan vaatimukset, jotka ovat oleellisia ohjelmistolle. Analyysistä on tehtävä graafinen malli korkean tason abstraktiota varten, joka muodostaa suunnittelun perustan.
(Ohjelmistotuotanto 2000, 69)

3.2 Asiakkaan vaatimukset

Lääkintälaitehuollolla oli selvät tarpeet ja vaatimukset sovellukselle:
1. Muutamaa nappia klikkaamalla nopea varaston päivitys ja tarkastus
2. Mahdollisimman vähän käyttäjäriippuvainen
3. Mobiililaitteille soveltuva
4. Automaattinen ilmoitus varaston tilanteesta eri rajaehdoin.
5. Tuotteilla tulee olla tarkat attribuutit

Kohta 4, tarkennus: Esimerkkinä tuotteen määrä alle tietyn rajan tai tuotteen päiväys vanhenemassa. Näistä tuotteista kerättäisiin lista, joka lähetetään varaston ylläpitäjälle sähköpostilla.
Kohta 5, tarkennus: Osa tuotteista saattaa sisältää käyttöpäivämääriä sekä lukumääriä per paketti.

3.3 Suunnittelun ensiaskeleet

Ajatus sovelluksen toiminnasta oli yksinkertainen. Käyttäjä avaa Sovelluksen yrityksen palvelimelta ja kirjautuu sisälle omilla tunnuksillaan. Sovellus varmistaa käyttäjän tiedot, kuten Osaston, jonka varastoja käyttäjä tulee käsittelemään. Tämän jälkeen käyttäjälle tarjotaan osaston varastohuoneista valintaikkuna, josta käyttäjä päättää oikean huoneen. Tämän jälkeen sovelluksessa avautuu sivu, jolla käyttäjä voi skannata hyllyn koodin käyttäen laitekameraa. Mahdollisuutena on myös kirjoittaa koodi käsin, mikäli optista koodia ei löydy tai se on vioittunut. Hyllyn valinnan jälkeen tarjotaan kolme painiketta, jotka ovat Lisää, Poista sekä Infot.
Infot-painikkeella käyttäjälle näytetään hyllyn tärkeimmät tiedot, kuten tuotteiden määrä sekä seuraavaksi vanhenevan tuotteen päivämäärä.
Lisää- ja Poista-painikkeen alta aukeaa uusi mahdollisuus skannata tuotteen koodi sekä antaa muita tietoja, joilla varmistetaan, että sovellus käsittelee oikeaa riviä tietokannassa. Painikkeiden toiminnot lähettävät tietokannalle pyynnöt, jotka prosessoivat toiminnon määrätyllä tavalla. Näin käyttäjän ja tietokannan välinen yhteys toimisi nopeasti ja mahdollisimman pienellä virheriskillä.

Työhön paneutuessani lähestyin varastoa alhaalta ylöspäin. Määrittelin yksittäisen tuotteen sisältämiä attribuutteja, joista yhden oli oltava hyllyn osoite. Hyllyistä korkeampi taso on huone, tai varasto, joka sisältää useita hyllyjä, joten loogisesti sen oli oltava seuraavana. Varastoista ylempänä on yrityksen osasto, jossa saattoi olla useampia varastoja, esimerkkinä Keskussairaalan leikkausosasto. Osastosta ylempänä saattoi olla vain yritys tai asiakas, jota ylemmäs ei ollut tarvetta siirtyä.

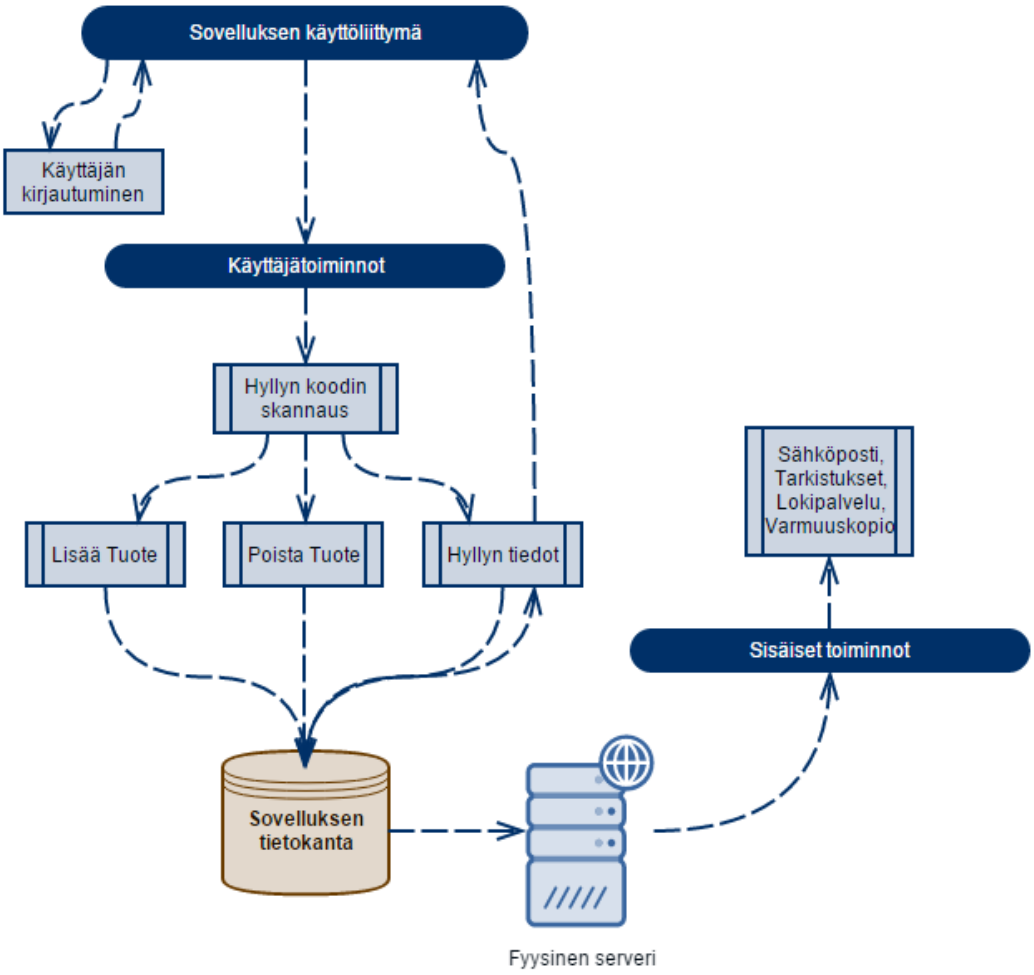
Vaatimusten perusteella sekä ylläpitoa varten piti tuotteille lisätä vielä muutama erityisattribuutti, joista kerrotaan lisää kohdassa 5.4.

Työssä en ottanut mallia muista vastaavista sovelluksista, sillä halusin keskittyä asiakasvaatimuksiin ja suunnitteluun. Yleispäteviä vaihtoehtoja löytyy internetistä sekä Applen App Storesta sekä Googlen Play-kaupasta. Play-kaupasta esimerkiksi HUBL Inventory vaikuttaa käytettävältä sovellukselta, mutta se ei täytä asiakasvaatimuksia, joita tälle työlle on asetettu.

4 SUUNNITTELU

4.1 Toimintakaavio

Sovelluksen pohjaksi tuli suunnitella toimintakaavio ensin tavallisen käyttäjän näkökulmasta. Varaston ylläpito useiden käyttäjien toimesta vaatii vähintään käyttöliittymän, josta käyttäjä ohjataan antamaan pyydetyt tiedot. Toiminnan nopeuttamiseksi sovellus sisältää useita ennalta määrättyjä prosesseja, jolloin mahdollisuudet käyttäjävirheisiin vähentyvät. Ennalta määrätyt prosessit ovat normaalille käyttäjälle Lisää Tuote, Poista Tuote sekä Hyllyn Tiedot.



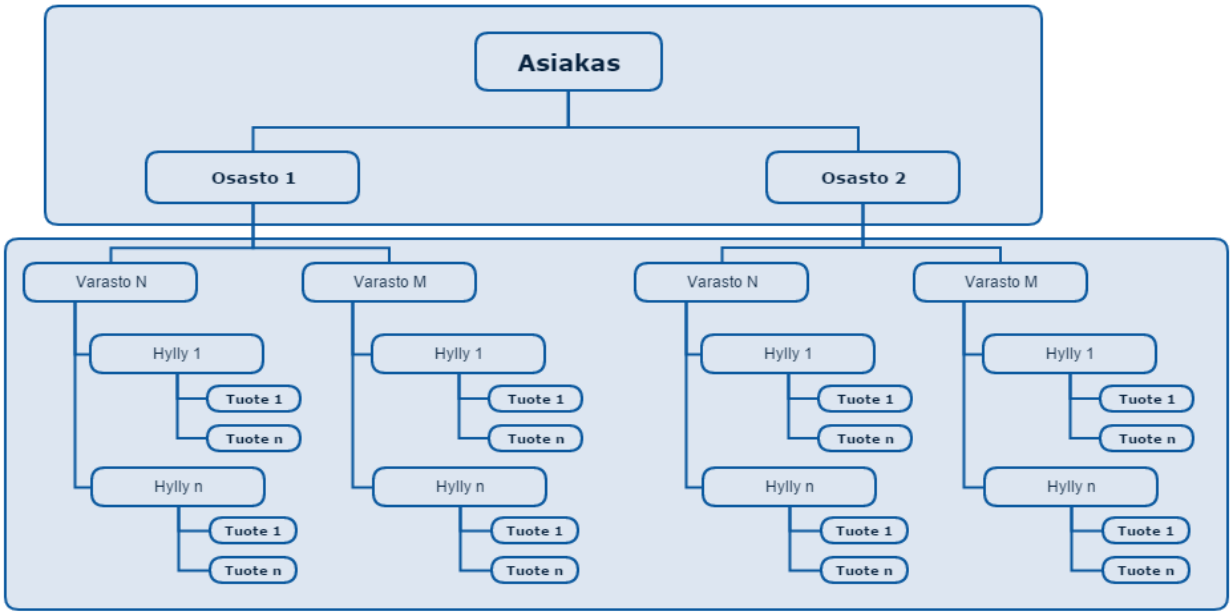
KUVA 2. Sovelluksen toimintakuvaus tavallisen käyttäjän kohdalla

Määrätyt prosessit tekevät muutoksia sovelluksen tietokantaan ja tuovat käyttäjälle takaisin tiedot hyllystä sekä tehdyistä muutoksista. Käyttäjälle näkymättömät prosessit,

kuten Sähköposti ja Tarkistukset, ovat serverillä pyöriviä tapahtumia, jotka tarkastavat tietokannan tilanteen (hyllyssä olevien tuotteiden määrä, vanhenevat tuotteet) sekä lähettävät määrätyille sähköpostiosoitteille yleiskuvan varaston tilanteesta.

4.2 Varastohierarkia

Sovellusta suunnitellessa oli parasta purkaa ongelma pienempiin osiin. Varaston käsittely ja tietojen lähetys helpottuivat merkittävästi, kun niitä käsitteli eri objekteina. Varastohierarkian hahmottamisen pohjalta oli helppo suunnitella tietokannan tarpeita ja tietuetaulujen määriä.



KUVA 3. Yrityksen varastohierarkia

Kuvasta 3 voi nähdä varastohierarkian eri objektit ja niiden väliset relaatiot.
Ylimpänä on asiakas tai yritys, joka on helpointa eritellä verkko-osoitteilla. Tämä myös poistaa tarpeen kokonaiselta tietuetaululta, jos vaikka asiakkaita olisi useampi. Toisaalta tämä asettaa vaatimuksen, jossa yhtä asiakasta tai yritystä kohden olisi yksi palvelin.
Yrityksellä voi olla useita osastoja, joten seuraavaksi määritellään Osasto, jonka alaisuudessa hyllyjen ja tuotteiden tulisi olla. Osastot on nimettävä uniikeilla nimillään, jotta tiedot pysyvät oikeassa osoitteessa.

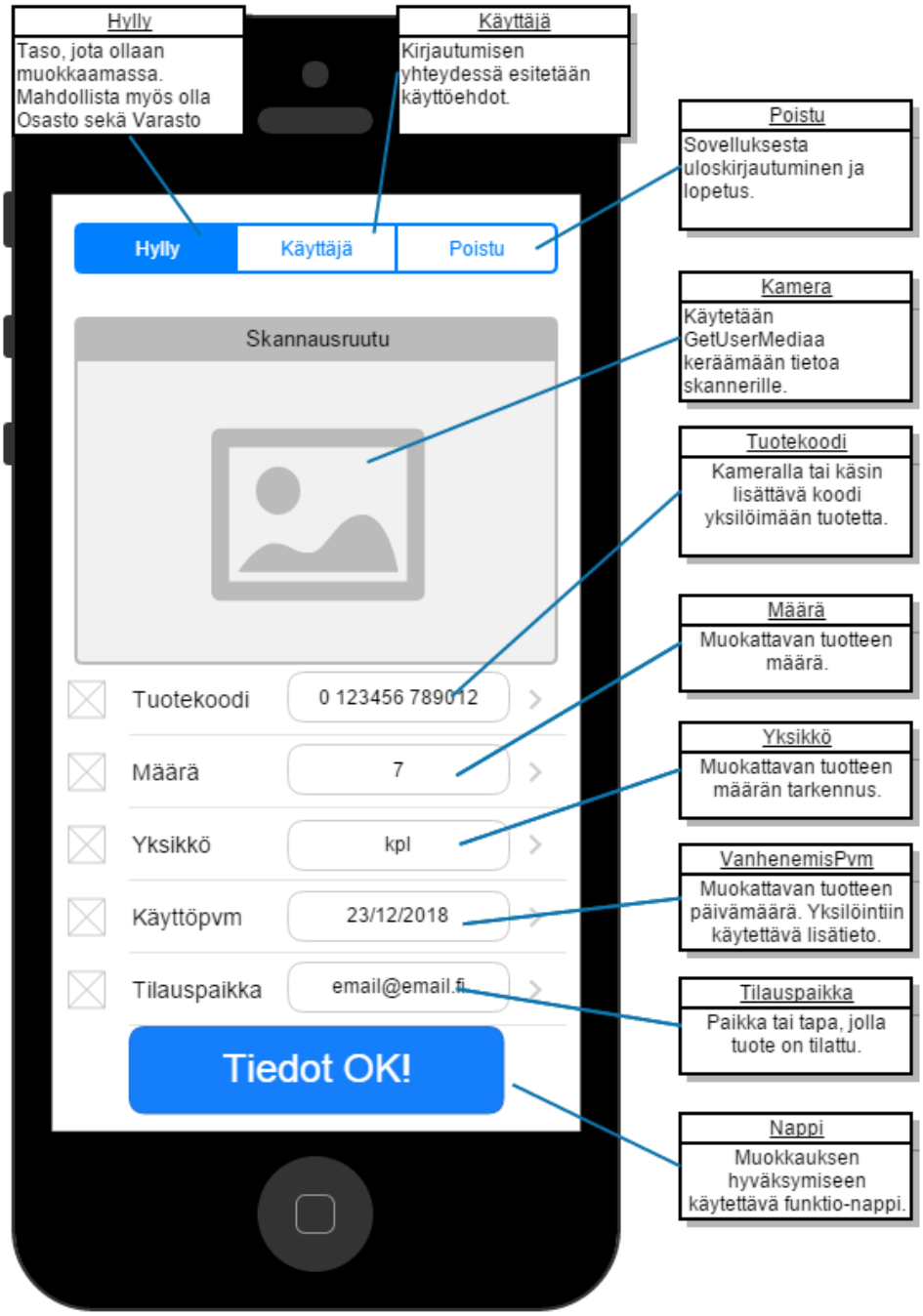
Osastoilla on mahdollista olla eri varastoja, joten nämäkin tuli ottaa huomioon. Satakunnan Keskussairaalassa leikkausosastolla oli useampia varastoja eri tuotekeskittymille, mikä aiheutti tämän tason lisäämisen sovellukseen.

Varaston alaisuudessa on useita Hyllyjä, jotka sisältävät lopulliset tuotteet, joita käyttäjät käsittelevät. Käyttäjästä riippuen näille voi myös asettaa omia tuoteteemoja, jotka antavat suuntaa, millaisia tuotteita hyllyillä tulisi olla. Hyllyillä tulee myös olla uniikit nimet ja viivakoodit, sillä tällä tasolla käytetään hyväksi laitekameraa, joka nopeuttaa sovelluksen toimintaa.

Alimmalta tasolta löytyy vihdoin yksittäiset tuotteet, jotka sidotaan oman tuotekoodinsa perusteella uniikeiksi riveiksi tietokannalle. Tuotteisiin kuuluu vielä useita attribuutteja, joista lisää myöhemmin kohdassa 5.4 sekä 5.5.

4.3 Sovelluksen käyttöliittymä

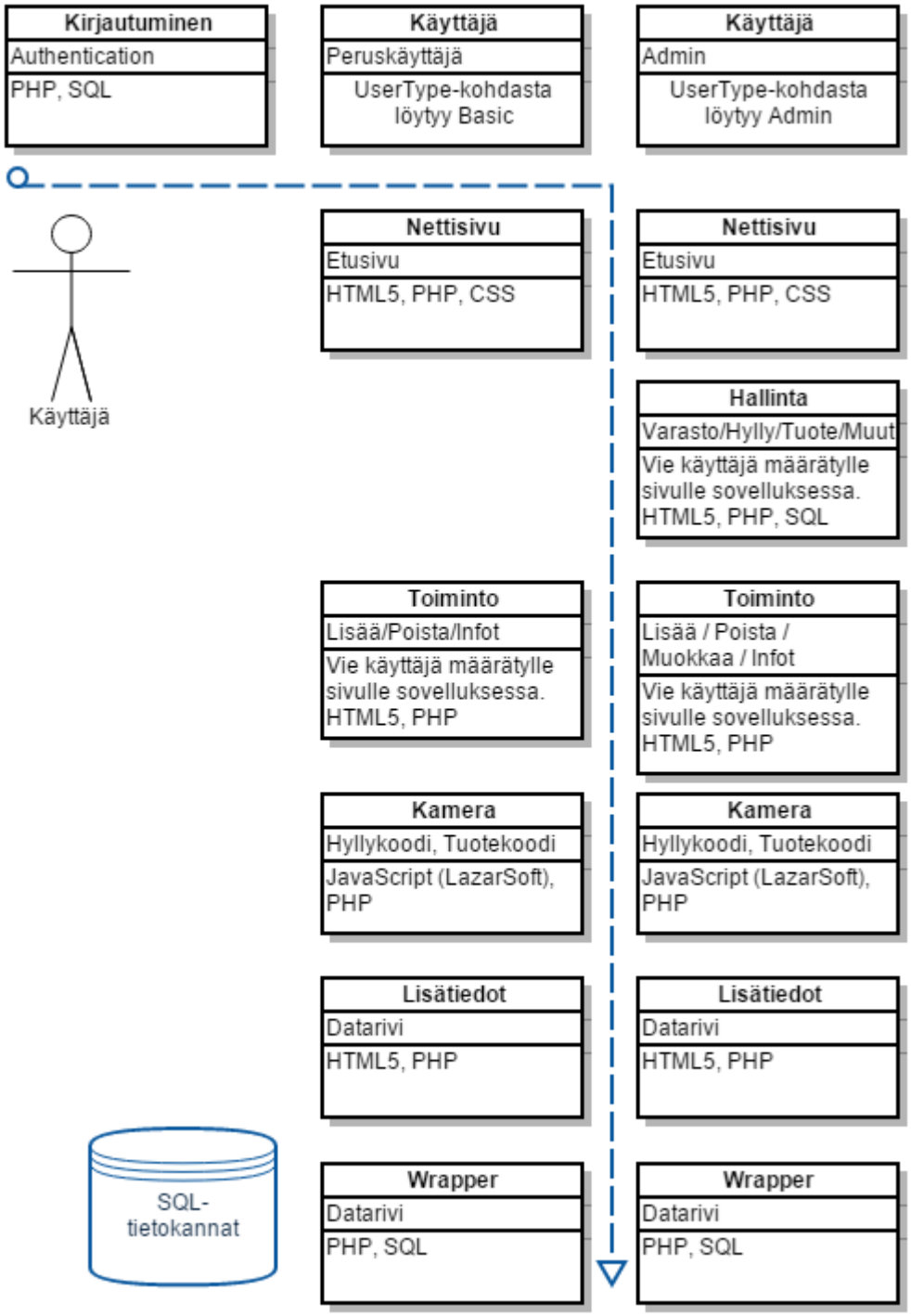
Työssä päädyttiin käyttämään HTML5, PHP sekä JavaScript -pohjaista nettisivua, joka on mahdollista sijoittaa samalle palvelimelle kuin tietokanta. Käyttöliittymä on kolmitasoinen. Päävalikko, josta löytyvät käyttäjätiedot, uloskirjautuminen sekä toiminnot. Toimintojen alta siirrytään skannausruutuun (alla), johon kerätään tapahtuman tiedot. Kolmannella tasolla on erinäisten tietojen tarkastelu. Esimerkiksi sovellukseen lisättyjen käyttäjien tai tuotteiden tietojen tarkastelu.



KUVA 4. Sovelluksen käyttöliittymä

4.4 Sovelluksen kulku

Perus- sekä pääkäyttäjällä on hieman erilainen käyttöympäristö. Merkittävin ero löytyy pääkäyttäjän oikeuksista muokata eri tasojen arvoja ja tietoja. Normaalilla käyttäjällä mahdollisuudet ovat rajatummat. Alla olevasta kuvasta käy ilmi eri käyttäjätasojen väliset eroavaisuudet.



KUVA 6.Sovelluksen kulku normaalin ja ylläpitäjän kohdalla

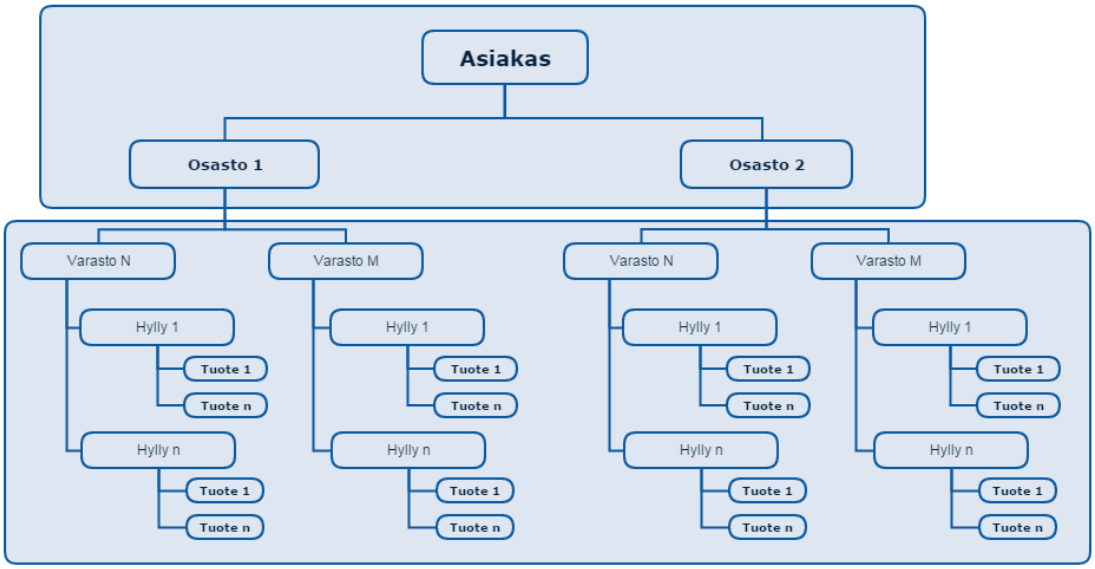
4.5 Yleinen käyttötapaus

Kirjautumisen jälkeen käyttäjän oikeudet tarkistetaan, ja näkymä siirretään käyttöliittymään. Peruskäyttäjälle esitetään Lisää-, Poista- sekä Infot-toiminnot, jotka koskevat vain Hylly- ja Tuoteobjekteja. Pääkäyttäjälle esitetään varastonhallintaa varten valikko objekteja varten. Objektin ja/tai toiminnon valitsemisen jälkeen Sovellus siirtyy kameran käyttöön, jolla käyttäjä voi ohittaa erinäisten koodien käsin kirjoittamisen. Koodien lukemisen ja lisätietojen keräämisen jälkeen kerätään saadut tiedot yhteen ja lähetetään ne palvelimen tietokannalle käsiteltäväksi.

5 TAULUKUVAUKSET

5.1 Objektien määrittely

Jokainen sovelluksen objekti tulee määritellä järkevää käyttöä varten.



KUVA 7. Sovelluksen objektit ja niiden relaatiot kuvana.

Asiakas-objekti määritellään yksinkertaisesti IP-osoittella, jolla otetaan yhteys palvelimeen, jossa sovellus ja tietokanta sijaitsevat.

Kaikki objektit Asiakkaan alla järjestetään erillisillä tietuetauluilla, jolloin tietokannan koko pienenee ja ylläpito helpottuu. Yksittäisen osastonimen muutos ei vaadi usean solun arvon muutosta, vaan yhden solun muutos päivittyy muihin tietoihin.

Tuote-objekti sisältää sovelluksen perustan. Kaikki normaalin käyttäjän toiminnot sovelluksen sisällä muuttavat Tuote-objektin arvoja taulukolla, joka sisältää Tuotteet. Tuote-objekteilta luetaan arvot sisäisille tapahtumille, kuten määritellyille tarkastuksille, sähköposti-ilmoituksille sekä Infot-toiminnolle.

5.2 Taulukuvaukset

Taulukuvaus on tiivis merkintätapa kuvata taulua ja sen sisältämiä sarakkeita. Yleisesti käytetty syntaksi on Taulunimi (sarake-1, sarake-2, ..., sarake-n). Jotta eri tyyppiset sarakkeet erotetaan toisistaan, perusavainsarakkeet alleviivataan ja viiteavainsarakkeet kursivoidaan alla olevan mallin mukaisesti:

Taulunimi (perusavainsarake, sarake, *viiteavainsarake*)
Perusavain on sarake, jonka perusteella rivit tunnistetaan ja erotetaan toisistaan. Perusavaimeen liittyy muutama sääntö:

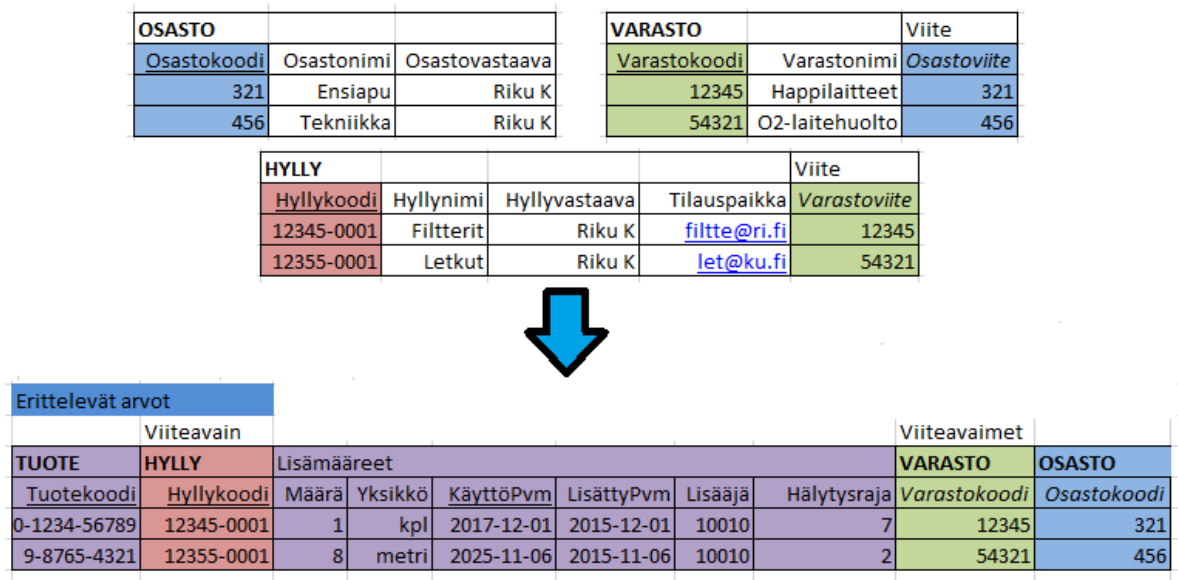
- perusavain on yksikäsitteinen muodostuen yhdestä tai useammasta sarakkeesta
- jokaisella taulun rivillä perusavaimella on oltava eri arvo
- perusavaimen arvo ei saa puuttua yhdeltäkään riviltä
- perusavaimen eheys (entity integrity)

Viiteavain on saraketyyppi, jota käytetään luomaan relaatioita eli suhteita taulujen välille. Viiteavainsarake luodaan tauluun, jossa viitataan (tyypillisesti) toisessa taulussa olevaan riviin tämän toisen taulun jonkin rivin perusavainta vastaavalla arvolla. Näin saadaan linkitettyä eri tauluissa olevat tiedot yhteen. Viiteavaimeen liittyviä sääntöjä ovat:

- viiteavaimen arvoalue on sama kuin viitattavan taulun perusavaimen arvoalue
- sarakkeet ovat vastaavat kuin viitattavassa taulussa
- viite-eheys (referential integrity)

(Kilpeläinen, A. www-sivut 2016)

Taulukoita muodostuu seuraavasti:



OSASTO		
Osastokoodi	Osastonimi	Osastovastaava
321	Ensiapu	Riku K
456	Tekniikka	Riku K

VARASTO		Viite
Varastokoodi	Varastonimi	*Osastoviite*
12345	Happilaitteet	321
54321	O2-laitehuolto	456

HYLLY				Viite
Hyllykoodi	Hyllynimi	Hyllyvastaava	Tilauspaikka	*Varastoviite*
12345-0001	Filtterit	Riku K	filtte@ri.fi	12345
12355-0001	Letkut	Riku K	let@ku.fi	54321

Erittelevät arvot

	Viiteavain							Viiteavaimet	
TUOTE	HYLLY	Lisämääreet						VARASTO	OSASTO
Tuotekoodi	Hyllykoodi	Määrä	Yksikkö	KäyttöPvm	LisättyPvm	Lisääjä	Hälytysraja	*Varastokoodi*	*Osastokoodi*
0-1234-56789	12345-0001	1	kpl	2017-12-01	2015-12-01	10010	7	12345	321
9-8765-4321	12355-0001	8	metri	2025-11-06	2015-11-06	10010	2	54321	456

KUVA 8. Tietuetaulujen kuvaus

Osasto-taulukon perusavaimena toimii Osastokoodi. Taulukko ei kaipaa viitearvoja, sillä ainoa objekti ylempänä hierarkiassa olisi Asiakas/Yritys, joka tässä työssä määriteltiin verkko-osoitteen perusteella.

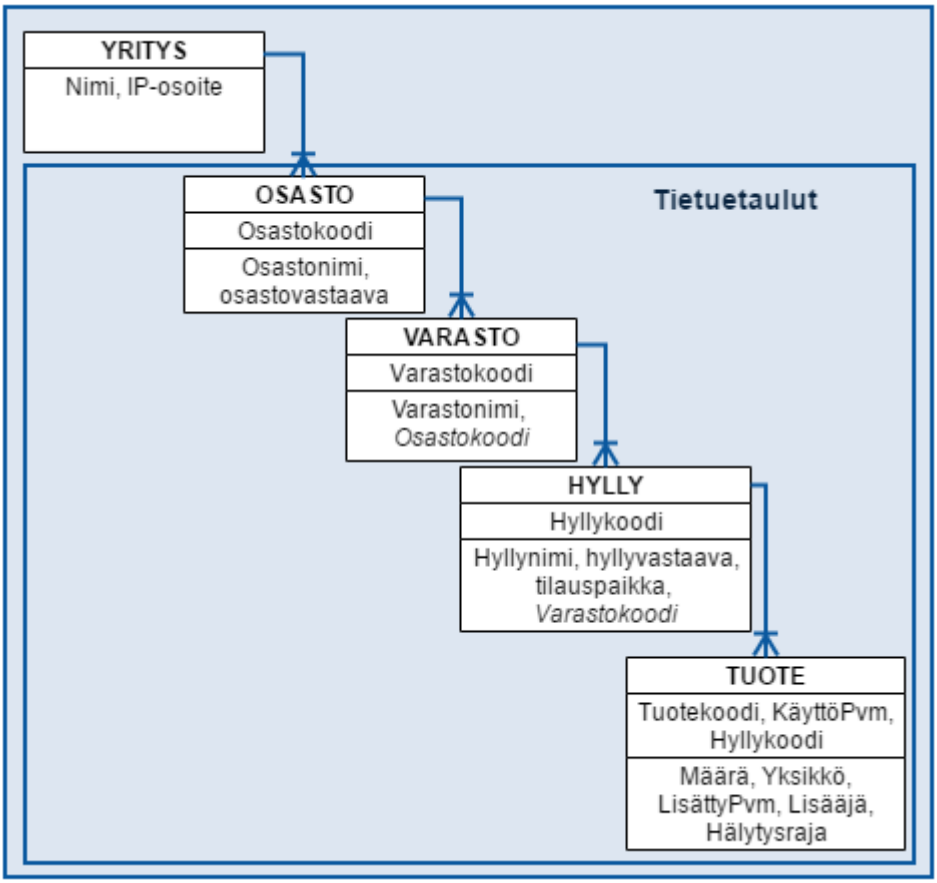
Varasto-taulukkojen perusavaimina toimivat Varastokoodi, sekä viitearvona käytetään Osastokoodia.

Hylly-taulukkojen perusavaimena käytetään kaksiosaista koodia, joka muodostetaan Varastokoodin sekä hyllyn järjestysnumeroa käyttäen. Täten koodiksi muodostuu esimerkiksi kuvassa näkyvä 12345-0001, jossa 12345 on Varastokoodi ja 0001 on Hyllyn järjestysnumero.

Lopulta Tuote-taulukko käyttää perusavaimenaan Tuotekoodia sekä KäyttöPvm:ää. Syy kahteen arvoon on oletus siitä, että tuotekoodi saattaa olla kahden eri artikkelin välillä samanlainen, mutta käyttöpäivämäärä on todennäköisesti eri. Lisäksi Tuote-objektin sijainti tulee määritellä käyttäen Osasto-, Varasto-, sekä Hyllykoodeja.

5.3 Tietuetaulujen väliset relaatiot

Tietokannan ja tilan säästämisen vuoksi Tuotteet perivät attribuuttejaan ylemmiltä tasoilta. Näin vältytään käyttäjän ajan tuhlaamiselta sekä datan toistamiselta. Jokaiselle tasolle määriteltiin yksi pääavain, sekä useampia tukea antavia attribuutteja.



KUVA 9. Attribuuttien relaatiot tasolta seuraavalle

Jokaista ylempää tasoa kohden saattaa olla yksi tai useampi alataso. Kuitenkaan jokaista alatasoa kohden ei voi olla useampaa ylätasoa. Kuvaa voisi ajatella eräänlaisena puuna. Yritys on yksittäinen runko, josta eriytyy useampia osastoja, eli oksia. Oksat jakautuvat edelleen varastoihin, hyllyihin ja lopulta tuotteisiin, eli lehtiin. Yksittäistä lehteä kohden ei voi olla useampia oksia tai puita.

Yksittäiselle tuotteelle keräytyy viitteitä ylemmiltä tasoilta mm. hyllyn, varaston sekä osaston koodi.

5.4 Attribuuttien määrittely

Mahdollisten virheiden välttämiseksi jokainen attribuutti tulee rajata tietyin ehdoin. String-arvo on puhdas merkkijono, Integer puhdas lukuarvo ja Float on desimaaleja sisältävä lukuarvo.

Tuotekoodi (String, 24 merkkiä)
Tuotteen paketissa tai näkyvällä paikalla sijaitseva optinen koodi, joka on ISO/IEC 15416 (linear) määrittelyjen mukainen. Tämä tarkoittaa sitä, että käytössä voi olla vain yksiuloitteisia viivakoodeja. Otetaan vastaan pelkkiä numeroja, kirjaimia sekä viivoja sisältäviä String-arvoja.

Määrä (Float, 5 merkkiä)
Desimaaliluku, jolla määritellään hyllyssä olevan tuotteen määrä. Esimerkiksi yksi (1.0) kappale happikennoja tai kolme ja puoli (3.5) metriä putkea. Float-arvo.

Yksikkö (String, 7 merkkiä)
Tuotteen määrän yksikkö. Vaihtoehtoina Kappale, Metri, Litra. Annetaan käyttäjälle pudotusvalikon muodossa String-arvona.

VanhenemisPvm (Date, ei merkkipituutta)
Attribuutti, joka sisältää takarajan tuotteen käyttöpäivämäärälle. Muodossa YYYY-MM-DD, eli Vuosi-Kuukausi-Päivä. Päivä ja kuukausi muodostuvat kahdesta numerosta, kun taas vuosi muodostuu neljästä. Jos käyttäjä ei määrittele päivämäärää itse, sovellus määrittelee arvon LisättyPvm-attribuutin perusteella kaavalla LisättyPvm + 10 vuotta. Käsitellään Integer-arvona, johon on mahdollista lisätä vuosia yksinkertaisella Plus-laskulla.

LisättyPvm (Date, ei merkkipituutta)
Sisältää päivämäärän, jolloin tuote lisättiin hyllylle. Kuten VanhenemisPvm, on tämäkin arvo muodossa YYYY-MM-DD.

Käyttäjäkoodi (String, 5 merkkiä)
Kuusinumeroinen juokseva luku, johon sidotaan käyttäjänimi sekä käyttäjätyyppi. Käyttäjä-taulukon pääarvo 8-merkkisenä lukuarvona. Käsitellään kuitenkin String-arvona tietoturvan vuoksi. Integer-arvojen käsittely vuotavan koodin tapauksessa olisi turhan helppoa, joten on turvallisempaa käyttää String-arvoja, joita ei tulla käsittelemään laskutoimituksissa.

Käyttäjätyyppi (String, 5 merkkiä)
Arvo, jolla määritellään käyttäjän oikeudet. Arvoina käytetään ”Basic” tai ”Admin”. Admin-tyypillä on mahdollista tehdä muutoksia taulukoihin ja tasoihin muunmuuassa lisäämällä tai poistamalla arvoja.

Käyttäjänimi (String, 6 merkkiä)
Käyttäjän nimi, joka muodostuu etu- ja sukunimestä käyttämällä kolmea ensimmäistä kirjainta kummastakin (Riku Ketonen => RikKet). Sidotaan yhteen muiden käyttäjäarvojen kanssa käyttäen Käyttäjäkoodia.

Hyllykoodi (String, 10 merkkiä)
Optisella koodilla kirjoitettu 10-merkkinen numerosarja. Esimerkkinä 12345-0001. Koodi muodostuu varaston koodista, sekä erikseen määritellystä hyllyn numerosta. Katso Tuotekoodi.

Hyllyvastaava (String, 5 merkkiä)
Henkilö, joka on vastuussa hyllyn täyttämisestä, tai tietää mistä vastaavia tuotteita tilataan. Arvo haetaan Henkilökoodin perusteella.

Tilauspaikka (String, 40 merkkiä)
Sähköpostiosoite tai yritys, josta asiakas yleensä tilaa tuotteensa.

Varastokoodi (String, 5 merkkiä)
QR-koodilla kirjoitettu 5-merkkinen numerosarja. Esimerkkinä 12345. Asiakas määrittelee itse merkkitilan käytöstä. Katso Tuotekoodi.

Varastonimi (String, 30 merkkiä)
Yleinen nimi varastolle. Käytetään vain datan kuvailuun. Merkkitilan käyttö on asiakkaan päätettävissä.

Osastokoodi (String, 3 merkkiä)
Yrityksen osastolle liitetty koodi, jolla sidotaan muita arvoja aputaulukoihin. 3-merkkinen numerosarja.

Osastonimi (String, 20 merkkiä)
Osastokoodiin sidottu nimi, jolla kuvataan osaston tarkoitus.

Osastovastaava (String, 5 merkkiä)
Henkilö, joka vastaa osaston varastoista. Käytetään Käyttäjäkoodia tilan säästämiseksi.

5.5 Tietuetaulujen normalisointi

Normalisointi on tietokannan tietojen järjestämisen prosessi. Siihen sisältyy taulukoiden luomista ja suhteiden järjestämistä taulukoiden välille noudattamalla sääntöjä, jotka on suunniteltu sekä suojaamaan tietoja että tekemään tietokannasta entistä joustavamman poistamalla redundanssin ja epäyhtenäiset riippuvuussuhteet.

Redundantit tiedot kuluttavat turhaan levytilaa ja aiheuttavat ylläpito-ongelmia. Jos useammassa kuin yhdessä sijainnissa olevia tietoja on muutettava, tiedot on muutettava täsmälleen samalla tavalla kaikissa sijainneissa. Asiakkaan osoitteen muutos on paljon helpompi toteuttaa, jos tiedot on tallennettu vain Asiakkaat-taulukkoon, eivätkä mihinkään muuhun sijaintiin tietokannassa. (Microsoftin Tuki www-sivut, 2016)

Tietuetauluja määritellessäni lähdin yhden Tuote-rivin tiedoista, joista keräsin mahdollisesti toistuvat Attribuutit toisiin taulukoihin.

Tuotekoodi	Määrä	Yksikkö	VanhenemisPvm	LisättyPvm	Käyttäjäkoodi	Käyttäjätyyppi	Käyttäjä
123-123123	2	kpl	1.1.2025	1.1.2015	10010	Admin	Riku K
123-123124	3	kpl	1.1.2025	1.1.2015	10010	Admin	Riku K
123-123125	1	kpl	1.1.2025	1.1.2015	10010	Admin	Riku K

Hyllykoodi	Hyllynimi	Hyllyvastaava	Tilauspaikka	Varastokoodi	Varastonimi	Osastokoodi	Osastonimi	Osastovastaava
12-345	Happikenno	Riku K	happi@kenno.fi	1-234	Happi Jne	01-244	Lääk.Huolto	Riku K
12-345	Happikenno	Riku K	happi@kenno.fi	1-234	Happi Jne	01-244	Lääk.Huolto	Riku K
12-345	Happikenno	Riku K	happi@kenno.fi	1-234	Happi Jne	01-244	Lääk.Huolto	Riku K

KUVA 10. Tuote-rivin attribuutit kahdessa osassa

Toistuvia Attribuutteja ovat Käyttäjäkoodi, Käyttäjä, Hyllykoodi, Hyllynimi, Hyllyvastaava, Varastokoodi, Varastonimi, Osastokoodi, Osastonimi sekä Osastovastaava. Näistä ylimääräisiä ovat Käyttäjä, Hyllynimi, Hyllyvastaava, Varastonimi, Osastonimi sekä Osastovastaava. Nämä Attribuutit voidaan sitoa vastaaviin koodi-attribuutteihin alla olevien taulukoiden mukaan.

	Käyttäjät	
Käyttäjäkoodi	Käyttäjätyyppi	Käyttäjänimi
10010	Admin	Riku K
10011	Basic	Saku K
10012	Basic	Jukka K

Hyllyt		
Hyllykoodi	Hyllynimi	Tilauspaikka
12-345	Happikenno	happi@kenno.fi
12-456	Tiivisteet	tiivi@steet.fi
12-567	Letkut	let@kut.fi

Varastot		
Varastokoodi	Varastonimi	Var.Vastaava
1-234	Happi Jne	10010
1-345	Yleiset	10010
1-456	Röntgen	10010

Osastot		
Osastokoodi	Osastonimi	Os.Vastaava
01-244	Lääk. Huolto	10010
01-255	Röntgen	10010
01-266	Ensiapu	10010

KUVA 11. Osa attribuuteista sidottu koodeihin tilansäästöä varten

Useampia tietuetaulukoita käyttäen Kuvan 11. suuri määrä toistuvia arvoja saadaan vähennettyä alla olevan kuvan näköiseksi. Puuttuvat arvot kerätään useista tietuetauluista.

Tuotekoodi	Määrä	Yksikkö	VanhenemisPvm	LisättyPvm	Lisääjäkoodi	Hyllykoodi	Varastokoodi	Osastokoodi
123-123123	2	kpl	1.1.2025	1.1.2015	10010	12-345	1-234	01-244
123-123124	3	kpl	1.1.2025	1.1.2015	10010	12-345	1-234	01-244
10-125-54	1	kpl	5.7.2024	5.7.2014	10012	12-567	1-456	01-255
10-125-55	1	kpl	1.1.2024	1.1.2014	10011	12-567	1-456	01-255

KUVA 12. Tuotekoodin perusteella määriteltävät Tuote-rivit

Yhden Tuoterivin 17 attribuuttia saatiin tiivistettyä 9:ään, jolloin tietokannan koko pienenee noin 40%:lla. Kaava tähän arvioon on $\frac{(Arvio\ lopullisista\ soluista * 100)}{Arvio\ alkuperäisistä\ soluista}$.

100 000 tuoterivin taulukko sisältäisi 1,7 miljoonaa solua ilman aputaulukkoja, kun taas aputaulukoiden kanssa soluja olisi 900 000, plus aputaulukkojen koko.

Suurelle yritykselle aputaulukot voisivat olla seuraavan arvion mukaisia:
Käyttäjät-taulukossa on kolme solua per rivi, jolloin noin 1000 käyttäjän taulukon koko olisi 3000 solua. Hyllyt-taulukon sisältö on 3 solua per rivi, joten 1000 hyllyn yrityksessä taulukon koko on 3000 solua. Varastot-taulukossa soluja on 3 per rivi, arviolta 100 varastoa per yritys, taulukon koko 300 solua. Osastot-taulukossa 3 solua per rivi, 20 osastoa per yritys, taulukon koko 60 solua. Tuote-taulukossa 100 000 tuotetta,

joilla 9 solua. Aputaulukkojen koko yhteensä 7360 solua. Tuotetaulukon koko 900 000 solua. Yhteensä 907 360 solua. Alkuperäisten solujen määrä on 1 700 000. Aiempaa kaavaa käyttäen saadaan uuden tietuetaulun suhteellinen koko verrattuna alkuperäiseen arvioon.

$$\frac{(907\,360 * 100)}{1\,700\,000} = 53.37\%$$

Aputaulukkoja käyttäen tietuetaulun solujen määrä putoaa noin 46,63%:lla. Säästö levytilan kohdalla ei ole suoraan verrannollinen aiempiin laskutoimituksiin, sillä soluissa olevan datan koko vaihtelee solusta toiseen. Säästö on kuitenkin tarpeeksi merkittävä, jotta usean taulukon järjestelmä on kannattava.

5.6 Käyttäjät

Jokaiselta sovellukseen lisätyltä käyttäjältä kerätään perustiedot. Nämä ovat koko nimi, puhelinnumero sekä sähköpostiosoite. Lisäksi jokaiselle käyttäjälle määräytyy sovelluksen kautta oma käyttäjänumero. Näistä muodostetaan oma tietuetaulu, jota käytetään aputaulukkona sovelluksen muiden toimintojen suorittamiseen.

5.6.1 Peruskäyttäjä

Peruskäyttäjän oikeuksiin kuuluu henkilörekisterin tarkastelu yhteistyön edistämiseksi, sekä alla kuvatut toiminnot liittyen varaston muokkaamiseen. Vastuisiin kuuluu sovelluksen vastuullinen käyttö sekä kaiken sovelluksen sisältämän datan pitäminen asiaankuuluvien henkilöiden piirissä.

5.6.2 Ylläpitäjä

Normaaliin käyttäjään verrattuna ylläpitäjä-tyyppisellä käyttäjällä on käytössään isompi lista toimintoja. Osaston, Varaston sekä Hyllyn lisääminen tarpeen tullen kuuluu ylläpitäjän oikeuksiin ja vastuisiin, sekä niiden muokkaaminen ja poistaminen.

6 TOIMINNOT JA PROSESSIT

Sovelluksen sisältämiä toimintoja, joiden vaikutuksen käyttäjä voi nähdä normaalissa käyttötilanteessa. Nämä liittyvät varaston ylläpitoon, muiden käyttäjien tietojen tarkasteluun sekä tiedon välittymiseen pääkäyttäjille.

6.1 Näkyvät toiminnot – Toimintakaaviot ja virheenkäsittely

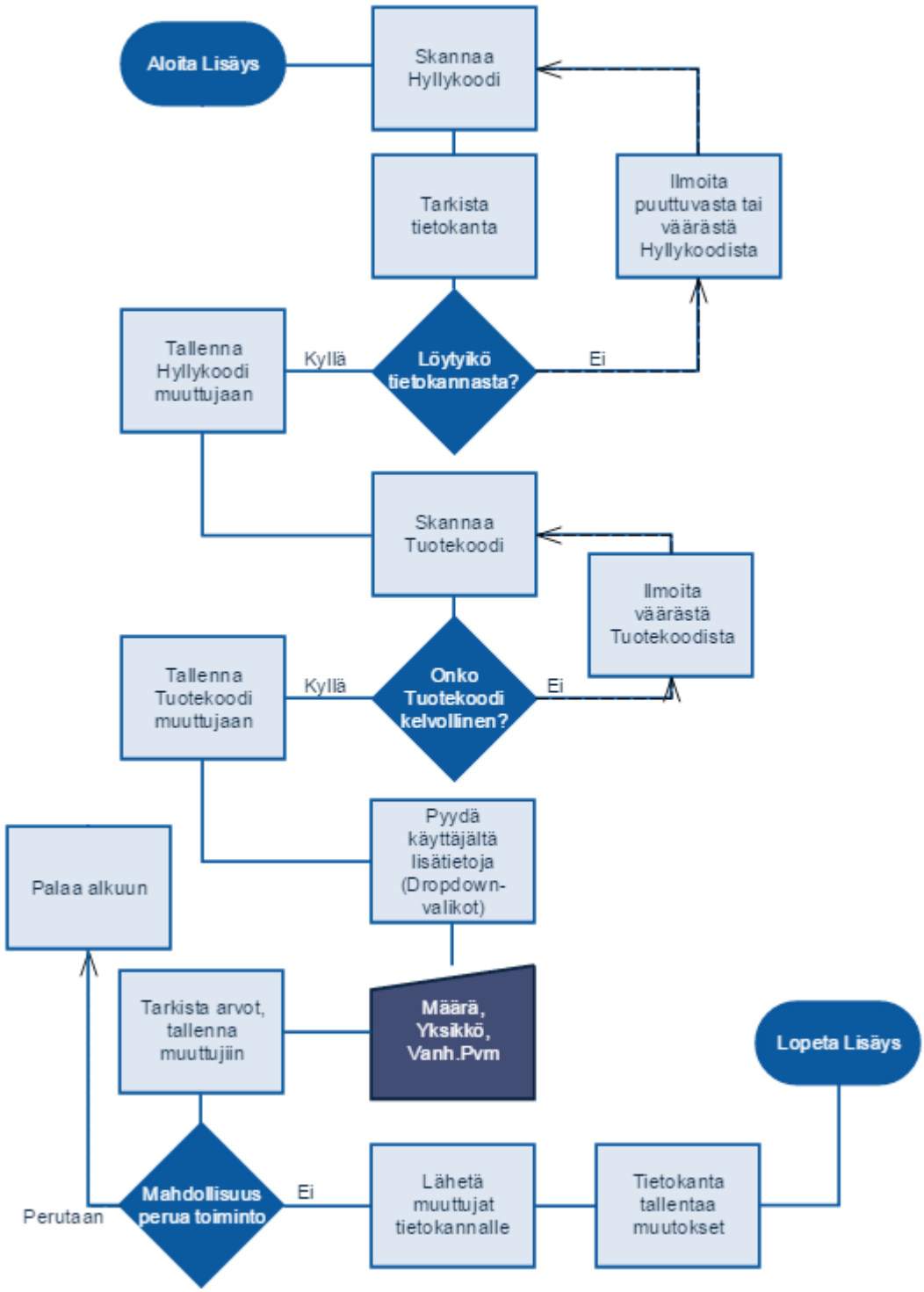
Kaikki toiminnot on kuvattu käyttäen vuokaavioita, sillä ne kuvaavat toimintaa ymmärrettävämmin kuin sanallisesti esitetyt tapahtumat.

Vuokaavioissa pitäydytään yleisen toiminnan tasolla, ilman moduulien sisäisiä toimintoja tai kutsuja, sillä ne ovat mahdollista toteuttaa monella eri tapaa. Toteutustapa riippuu Sovelluksen ohjelmoijasta, eikä työssä ole tarkoitus rajata toteutustapoja yhdenlaiseen ratkaisuun.

6.1.1 Toiminnot – Lisää

Normaalin käyttäjän suorittama Tuotteen lisäys Hyllyyn toteutuu alla olevan vuokaavion mukaisesti.

Toiminto aloitetaan valitsemalla Lisää-toiminto käyttöliittymästä. Tämän jälkeen sovellus avaa sivun, jolla skannataan laitekameralla Hyllykoodi. Skannaamisen jälkeen sovellus tarkastaa, onko koodia tietokannassa. Puuttuvasta tai väärästä koodista annetaan käyttäjälle ilmoitus, oikealla koodilla sovellus tallentaa Hyllykoodin muuttujaan myöhempää käyttöä varten. Sovellus siirtyy Tuotekoodin skannaamiseen ja skannaamisen jälkeen tarkistetaan, onko Tuotekoodi aiemmin tehtyjen määriteltyjen mukainen. Vääränlainen koodi antaa käyttäjälle ilmoituksen ja palaa Tuotekoodin skannaukseen. Oikean koodin löytyessä se tallennetaan kuten Hyllykoodi, myöhempää käyttöä varten. Käyttäjä antaa Tuotteen lisätiedot sovellukselle (Määrä, Yksikkö, Vanhenemispäivämäärä), jotka tallennetaan taas muuttujiin. Sovellus näyttää käyttäjälle tapahtuman kaikki tiedot (Varaston, Hyllyn sekä Tuotteen tiedot) ja pyytää vahvistusta näille. Vahvistuksen jälkeen sovellus kerää tiedot yhteen Wrapper-funktiolla ja lähettää muutoksen palvelimelle, joka suorittaa muutoksen tietokantaan.

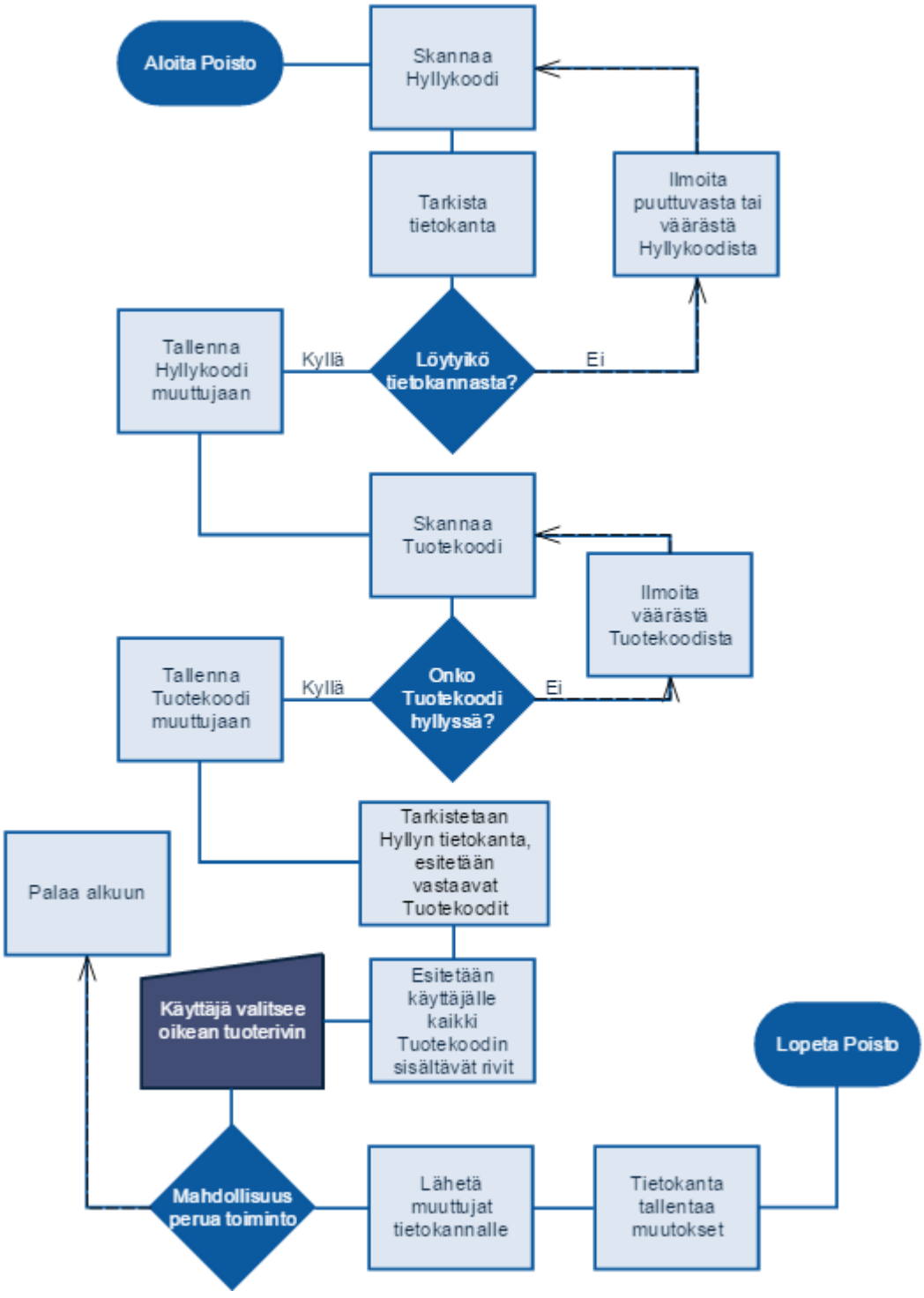


KUVA 13. Vuokaavio Lisää-toiminnosta

6.1.2 Toiminnot – Poista

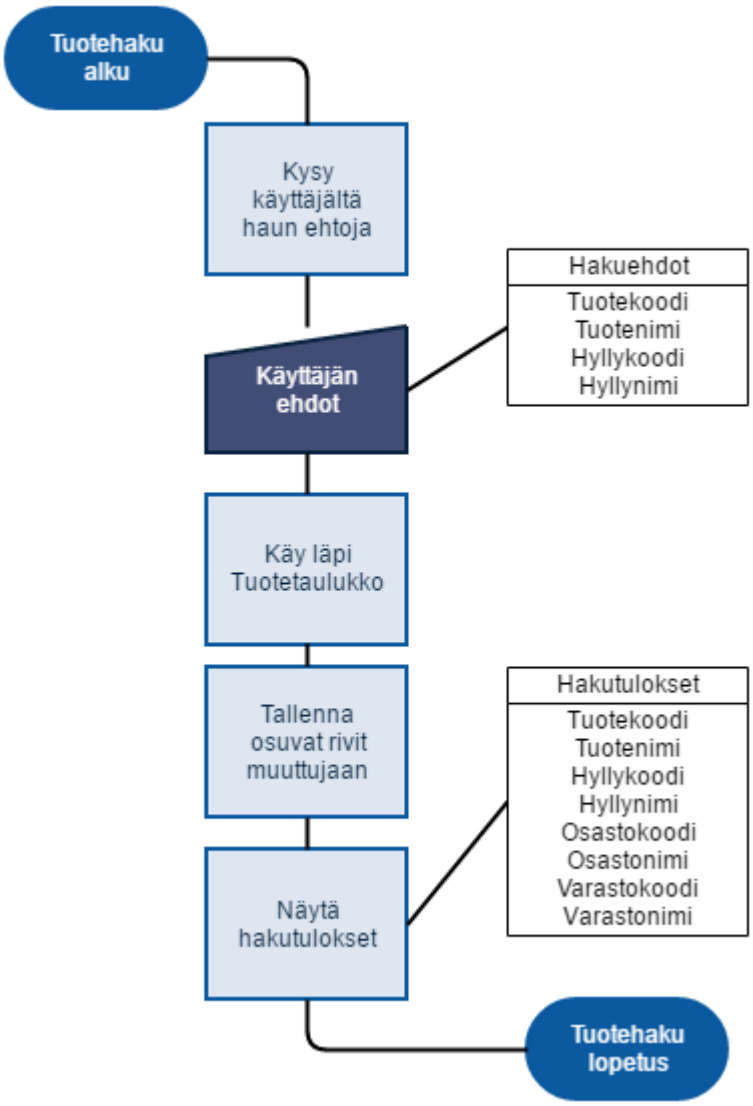
Poisto-toiminto ei eroa suuremmin Lisää-toiminnosta. Koska tuotekoodeja saattaa olla samanlaisia, tulee käyttäjälle esittää mahdolliset toistuneet rivit. Ideaalitilanteessa toistuvia tuotekoodeja ei ole samalla hyllyllä, mutta väärän rivin muokkaaminen mahdollistaa vääristyneen varastotilanteen.

Toiminto aloitetaan valitsemalla Poista-toiminto käyttöliittymästä. Tämän jälkeen sovellus avaa sivun, jolla skannataan laitekameralla Hyllykoodi. Skannaamisen jälkeen sovellus tarkastaa, onko koodia tietokannassa. Puuttuvasta tai väärästä koodista annetaan käyttäjälle ilmoitus, oikealla koodilla sovellus tallentaa Hyllykoodin muuttujaan myöhempää käyttöä varten. Sovellus siirtyy Tuotekoodin skannaamiseen ja skannaamisen jälkeen tarkistetaan, onko Tuotekoodi aiemmin tehtyjen määriteltyjen mukainen. Vääränlainen koodi antaa käyttäjälle ilmoituksen ja palaa Tuotekoodin skannaukseen. Oikean koodin löytyessä se tallennetaan kuten Hyllykoodi, myöhempää käyttöä varten. Käyttäjä antaa Tuotteen lisätiedot sovellukselle (Määrä, Yksikkö, Vanhenemispäivämäärä), jotka tallennetaan taas muuttujiin. Näillä tiedoilla varmistetaan oikean rivin käsittely tietokannassa. Sovellus näyttää käyttäjälle tapahtuman kaikki tiedot (Varaston, Hyllyn sekä Tuotteen tiedot) ja pyytää vahvistusta näille. Vahvistuksen jälkeen sovellus kerää tiedot yhteen Wrapper-funktiolla ja lähettää muutoksen palvelimelle, joka suorittaa muutoksen tietokantaan.



KUVA 14. Tuotteen poisto tietokannasta

6.1.3 Toiminnot – Tuotehaku



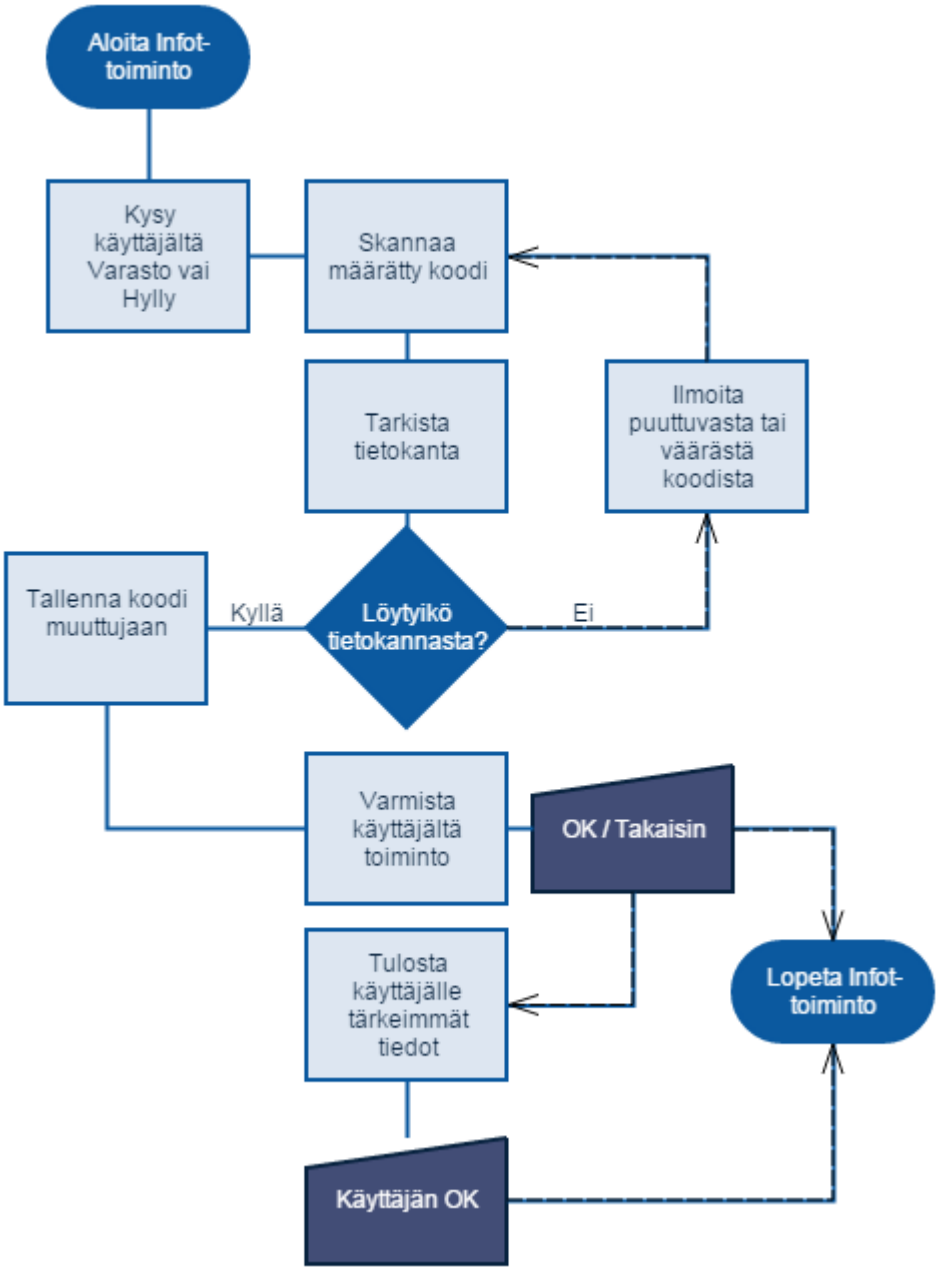
KUVA 15. Tuotehaku-ominaisuus

Tuotteiden haku sovelluksesta on mahdollista käymällä Tuotetaulukko läpi käyttäjän antamilla ehdoilla. SQL-komentojen avulla on mahdollista käydä koko taulukko läpi tarkoin ehdoin tehokkaasti.

Tuotehaku-ominaisuus kysyy käyttäjältä hakuehdot, joiden perusteella tehdään kysely palvelimelle. Palvelin käy läpi Tuotetaulukon ja tallentaa osuneet rivit, jotka lopulta esitetään käyttäjälle. Tuloksista käy ilmi tarkat tiedot jokaisesta rivistä, muunmuuassa fyysinen sijainti osaston, varaston ja hyllyn perusteella.

6.1.4 Toiminnot – Infot (Hyllyltä/Varastosta)

Käyttäjällä on mahdollisuus tarkastella kokonaisen Varaston tai Hyllyn tietoja. Muunmuassa Varastossa olevien Hyllyjen määrä, osastovastaava sekä varastonimi tulee käydä ilmi Varastoa tarkasteltaessa. Hyllyn kohdalla taas tärkeintä on tietää artikkelien määrä, seuraava päivämäärä, jolloin Tuote vanhenee sekä hyllystä vastaavan henkilön nimi.

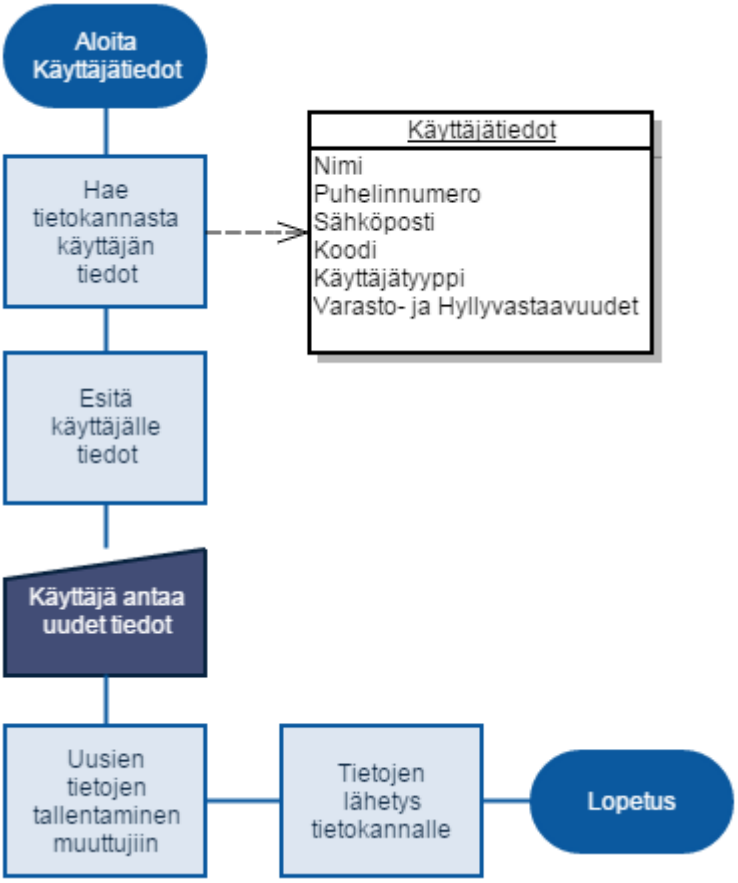


KUVA 16. Varaston tai Hyllyn tietojen tarkistus

Toiminto aloitetaan Infot-toimintoa painamalla. Sovellus pyytää käyttäjältä Tason määritelmää. Käyttäjän on mahdollista skannata hyllyn tai varaston koodi, tai valita se käsin kaikkien vastaavien joukosta. Tietokannasta tarkistetaan kyseisen tason tiedot ja tallennetaan Taso sekä koodi muuttujiin. Käyttäjältä varmistetaan toiminnon suorittaminen, jonka jälkeen tietokannasta poimitut tiedot esitetään käyttäjälle. Tiedot sisältävät kaikki muuttujat sekä alempien tasojen lukumäärän.
Käyttäjän painettua OK sovellus lopettaa toiminnon ja siirtyy päävalikkoon.

6.1.5 Toiminnot – Omien tietojen muokkaus

Erityistä mainittavaa tässä toiminnossa ei ole. Sovellus antaa tietokannalle käskyn esittää sisäänkirjautuneen käyttäjän tiedot, jotka tallennetaan ja esitetään käyttäjälle. Tämän jälkeen käyttäjän on mahdollista muokata tietojaan ja tallentaa ne takaisin tietokantaan.

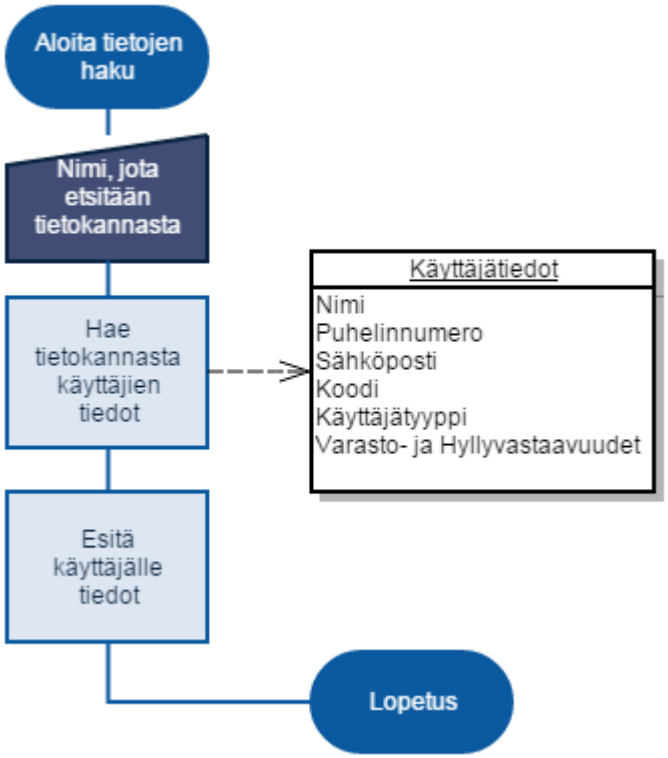


KUVA 17. Käyttäjätietojen muokkaaminen

Toiminto alkaa keräämällä käyttäjän tiedot tietokannasta ja esittämällä ne käyttäjälle. Käyttäjälle annetaan mahdollisuus muuttaa tietojaan tietyin rajaehdoin. Tietojen antamisen jälkeen sovellus kerää tiedot Wrapper-funktion avulla ja lähettää ne tietokannalle käsiteltäväksi.

6.1.6 Toiminnot – Muiden tarkastelu

Käyttäjän on mahdollista hakea toisien käyttäjien tietoja antamalla etu- tai sukunimi, jota käyttämällä haetaan käyttäjätietokannasta vastaava rivi. Koko tietokantaa ei esitetä peruskäyttäjälle väärinkäytösten estämiseksi, mutta pääkäyttäjälle kyseinen toiminto on kätevä työkalu.

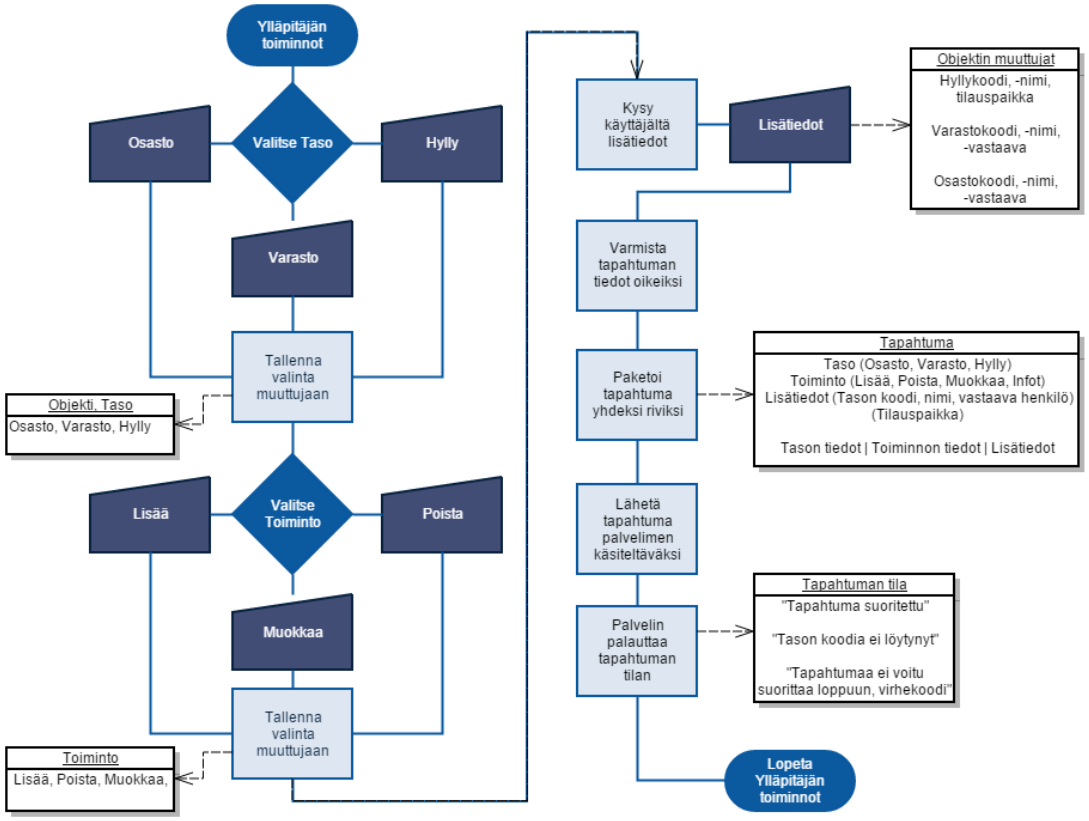


KUVA 18. Muiden tietojen tarkastelu

Toiminto kysyy käyttäjältä etsittävää etu- tai sukunimeä, jonka perusteella esitetään rivit, joilta etsittävät termit löytyivät. Useampien osumien tapauksessa käyttäjälle esitetään linkit henkilöihin, jotka sopivat hakutermeihin. Linkkien takaa löytyy haetun käyttäjän tiedot, kuten nimi, puhelinnumero sekä sähköposti.

6.1.7 Toiminnot – Tasojen käsittely

Tasojen käsittely on ylläpitäjän tai pääkäyttäjän vastuulla. Toiminnolla on mahdollista muokata eri attribuutteja, jotka on sidottu erinäisiin tasoihin. Esimerkiksi Osasto-tasolta on mahdollista vaihtaa Osastovastaavan nimi tai koko Osaston nimi.



KUVA 19. Tasojen käsittely

Tasojen käsittely alkaa tason valinnalla. Sovellus esittää vaihtoehdoiksi Osaston, Varaston sekä Hyllyn. Tuotetasolla tapahtuvia muutoksia ei tarjota, sillä niitä voi suorittaa jokainen käyttäjä Lisää- ja Poista-toiminnoilla. Tason valinnan jälkeen sovellus tallentaa tapahtuman ja jatkaa Toiminnon valitsemiseen. Toiminnon ja Tason tallentamisen jälkeen sovellus pyytää käyttäjältä lisää tietoja, riippuen aiemmista valinnoista. Tason valinnasta riippuen käyttäjältä vaaditaan tason koodi, nimi sekä vastuussa olevan henkilön käyttäjänumero. Kaikki tiedot esitetään käyttäjälle tapahtuman varmistamiseksi. Hyväksymisen jälkeen koko tapahtuma paketoidaan Wrapper-funktiolla ja lähetetään palvelimelle käsiteltäväksi. Palvelin suorittaa tapahtuman, tekee muutokset tietokantaan ja palauttaa tapahtuman tilan käyttäjälle. Tapahtuman tilan palauttamisen

tarkoituksena on antaa käyttäjälle palaute tapahtumasta ja mahdollisista virhetilanteista, jotka voivat aiheutua täyttyneestä tallennustilasta tai muusta tietokantavirheestä.

6.1.8 Toiminnot – Käyttäjämuutokset

Pääkäyttäjällä on oikeus käsitellä sovellukseen oikeutettuja henkilöitä. Näiden tietojen muokkaaminen on helpointa toteuttaa käsittelemällä Käyttäjät-tietokantaa, jonka attribuutteja voi muokata, sekä kokonaisia rivejä voi lisätä sekä poistaa.

1	Käyttäjäkoodi	Käyttäjätyyppi	Käyttäjänimi
2	10010	Admin	Riku K
3	10011	Basic	Saku K
4	10012	Basic	Jukka K

KUVA 20. Käsiteltävä käyttäjärivi

Toiminto lisää pääkäyttäjälle vastuun henkilörekisteristä, joka on laissa määritelty ja rajattu.

Henkilötietolain soveltamisen kannalta keskeinen vaatimus on määritellä henkilötietojen käsittelyn tarkoitus. Henkilötietojen käsittelyn tarkoitus tulee määritellä siten, että siitä ilmenee, minkälaisten rekisterinpitäjän tehtävien hoitamiseksi henkilötietoja käsitellään.

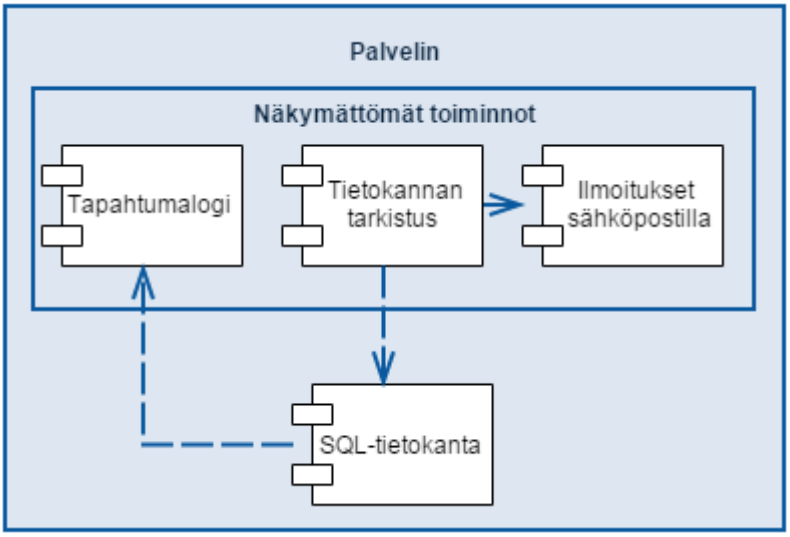
Vain tämän määrittelyn pohjalta voidaan arvioida, mitkä ovat käsittelyn tarkoituksen kannalta tarpeellisia ja virheettömiä henkilötietoja. Rekisteröidyn suostumuksellakaan ei saa kerätä tai muutoin käsitellä käsittelytarkoituksen kannalta tarpeettomia henkilötietoja.

Henkilörekisterillä tarkoitetaan henkilötietolain määritelmän mukaan käyttötarkoituksensa vuoksi yhteenkuuluvista merkinnöistä muodostuvaa henkilötietoja sisältävää tietojoukkoa, jota käsitellään osin tai kokonaan automaattisen tietojenkäsittelyn avulla taikka, joka on järjestetty kortistoksi, luetteloksi tai muulla näihin verrattavalla tavalla siten, että tiettyä henkilöä koskevat tiedot voidaan löytää helposti ja kohtuuttomitta kustannuksitta.

Lain määrittämä henkilörekisteri on siten looginen henkilörekisteri. Samaan loogiseen rekisteriin kuuluvat sekä automaattisen tietojenkäsittelyn avulla ylläpidetyt osat, että manuaalisesti pidetyt osat, jos rekisterinpitäjä käyttää tietoja saman tehtävän hoitamisessa. (Tietosuoja.fi www-sivut 2016).

6.2 Näkymättömät toiminnot

Palvelin suorittaa Sovellukseen kuuluvia prosesseja taustalla täysin läpinäkyvästi. Tietokantaan tehdyt muutokset tallennetaan tapahtumalogiin, tietokantaa tarkastellaan mahdollisten vanhenevien tuotteiden varalta ja näistä luodaan viesti, joka lähetetään erikseen määritellyille henkilöille määrätyin väliajoin. Näillä toiminnoilla pidetään huoli varaston tuhansista tuotteista, jotka mahdollisesti voivat vanhentua, tai jos tietyn tuotteen lukumäärä putoaa alle halutun rajan.



KUVA 21. Näkymättömien toimintojen suhteet tietokantaan

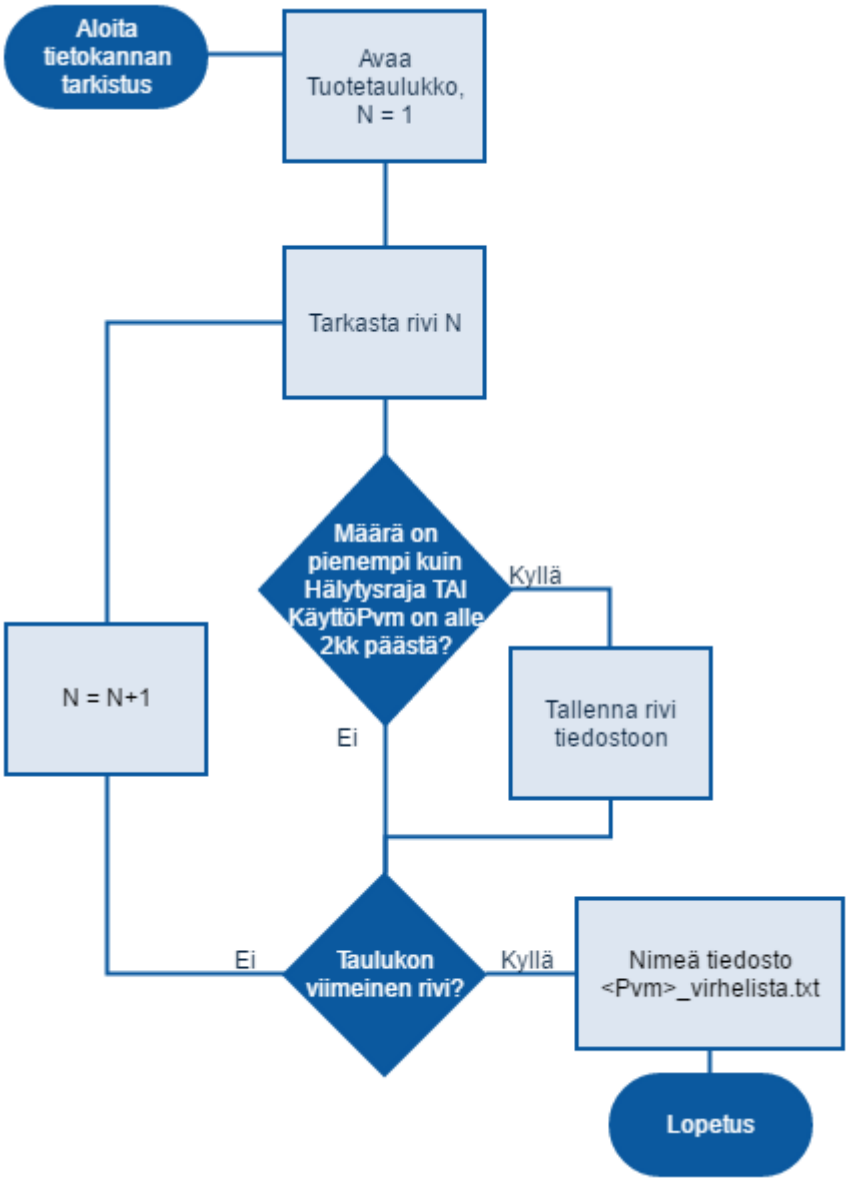
Yllä esitetyt Näkymättömät toiminnot tukeutuvat täysin SQL-tietokantaan. Tietokannan tarkistus käy läpi toivottujen rajojen perusteella jokaisen tietuetaulun ja kerää niistä tiedot pyydettyjen ehtojen mukaisesti. Esimerkkinä voisi olla ”Kaikki Tuoterivit, joissa Määrä on alle (2 Kpl), (10 metriä)”. Näin prosessi keräisi kaikki tuoterivit, joissa

tuotteen määrä on käymässä vähiin. Tämä tieto siirrettäisiin Sähköposti-toiminnolla tietokannasta löytyvien, määrättyjen henkilöiden sähköpostiin ja tuotteiden määrä ei koskaan putoaisi nollille huomaamatta.

Tietokannasta kerättäisiin muutos- ja tarkastelutapahtumat talteen erilliseen lokitiedostoon myöhempää tarkastelua varten. Toimintojen ajaminen eri väliajoin on mahdollista toteuttaa esimerkiksi ajastettuina toimintoina (Scheduled Tasks, Windows) tai lisäämällä skriptin alkuun suoritusajan (at-komento, Unix).

6.2.1 Tietokannan tarkastus

Tietokannan tarkastaminen on mahdollista toteuttaa serverillä tapahtuvin skriptein ilman käyttöliittymää.



KUVA 22. Tietokannan tarkastaminen

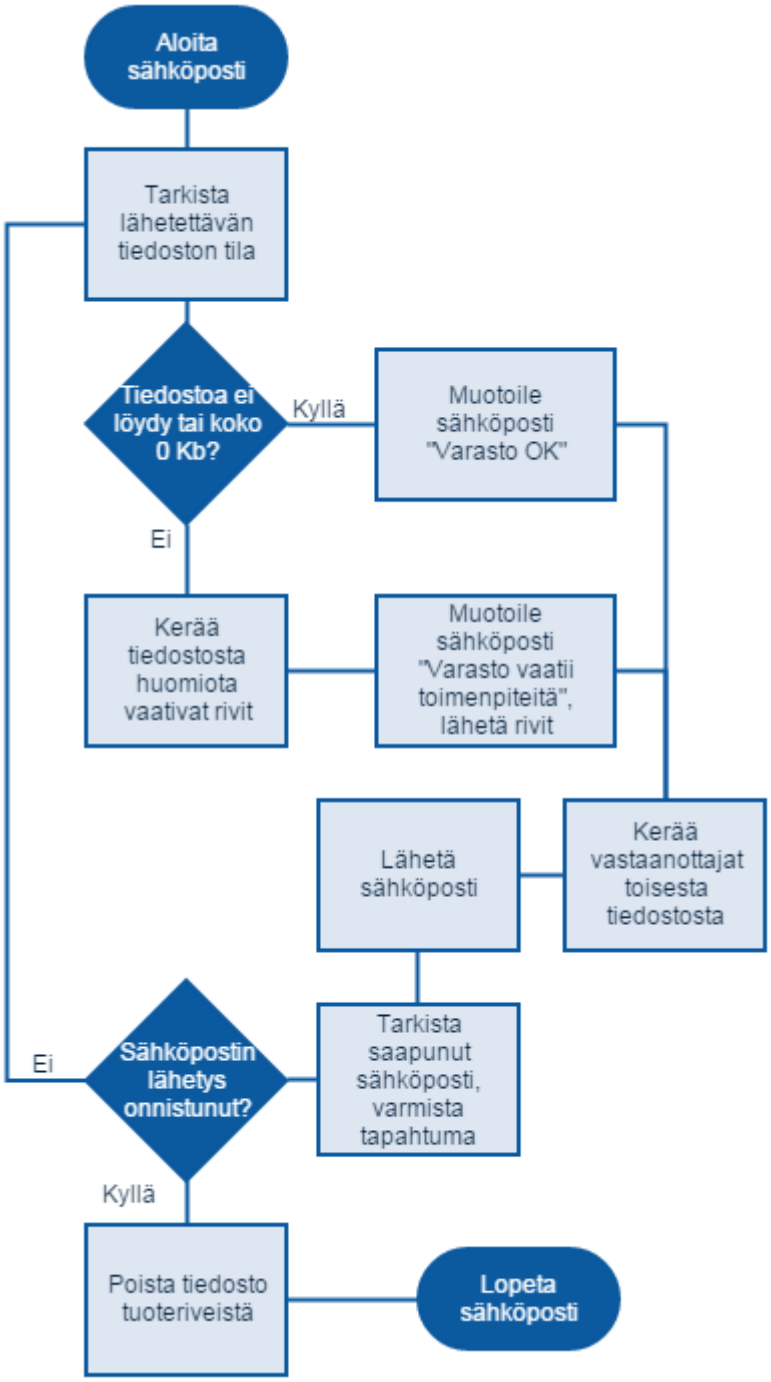
Määrätyin väliajoin palvelimella ajettava skripti suorittaa automaattisesti yllä kuvatun prosessin. Prosessi käy läpi tuotetaulukon ja kerää tuoteriveillä esiintyvät puutteet. Esimerkiksi lähestyvät vanhenemispäivämäärät tai alle määrätyn rajan olevat tuotteet kerätään talteen sähköposti-prosessia varten.

Prosessi avaa tuotetaulukon käyttöönsä ja tarkastaa rivi kerrallaan koko taulukon. Jos riviltä löytyvä Määrä-attribuutti on pienempi kuin Hälytysraja-attribuutti, tai Käyttö-Pvm-attribuutti on alle 2 kuukauden päässä, otetaan koko tuoterivi talteen ja siirrytään seuraavaa riviä tarkastamaan. Jos rivi läpäisee kummatkin ehdot, siirrytään seuraavaan riviin.

Kun viimeinen rivi on tarkastettu, prosessin käyttämä tiedosto nimetään päivämäärän perusteella. Tätä tiedostoa käytetään Sähköposti-toiminnossa.

6.2.2 Sähköposti-ilmoitukset

Yksi asiakkaan toivomuksista oli sähköpostiin saapuva ilmoitus varaston tilanteesta. Tämäkin on toteutettavissa palvelimella ajettavilla skripteillä.

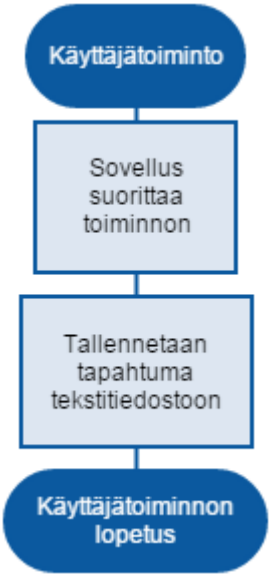


KUVA 23. Sähköpostin lähetys tuoteriveistä

Prosessi luottaa täysin aiempaan tietokannan tarkastukseen ja sen luomaan tiedostoon. Prosessi tarkastaa määrätystä sijainnista löytyvän tiedoston koon ja jos tiedostoa ei löydy, lähetetään määrätyille kontakteille sähköposti ”Varasto OK”. Jos tiedosto löytyy ja sillä on kokoa enemmän kuin 0 Kilobittiä, luetaan tiedostosta löytyvät varaston epäkohdat. Prosessi muotoilee sähköpostin tiedoston pohjalta ja lähettää sen määrätyille kontakteille. Prosessi tarkastaa vielä sähköpostin lähetyksen kuuntelemalla omaa Saapuneet-kansiotaan ja epäonnistuminen aloittaa prosessin alusta. Onnistunut lähetys taas poistaa aiemmin luodun tiedoston ja lopettaa prosessin.

6.2.3 Tietokannan tapahtumaloki

Tietokannalle tehtävät muutokset ovat helpointa toteuttaa lisäämällä käyttäjätoimintoihin koodirivi, joka tallentaa koko tapahtuman tekstitiedostoon.



KUVA 24. Yksinkertainen tapahtumaloki

Jokaisen toiminnon loppuun liitetään komento, joka kirjoittaa tapahtumalokiin tapahtuman tiedot. Wrapper-funktioon kerätyt tiedot lähetetään siis sekä tietokannan, että tapahtumalokin käsiteltäväksi ja tallennettavaksi. Näin saadaan varmistettua tapahtumat kahdella tapaa. Ensimmäinen varmistus tapahtuu tietokannan omilla toiminnoilla, toinen varmistus toimintoihin koodattuna.

Tapahtumalokin lukeminen tapahtuu suoraan palvelimella käyttämällä mitä tahansa tekstieditoria. Tiedoston muutosoikeudet pysyvät kuitenkin pelkästään palvelimella, eikä tiedostoa ole mahdollista muokata käsin ilman oikeuksien muuttamista. Tämä toteutetaan palvelintasolla.

7 PROTOTYYPPI

Prototyyppiä luodessa tulee ottaa seuraavaksi esitetyt asiat huomioon.

7.1 Toteutuksen vaatimukset

Sovelluksen toteutus asettaa tietyt vaatimukset sekä etälaitteille, palvelimelle että tietokannalle.

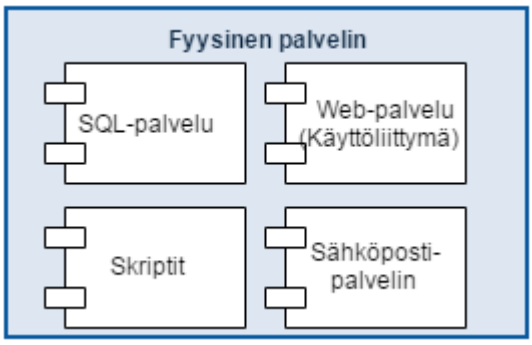
Etälaitteilla tulee olla :
Selain, joka tukee WebRTC-projektia (GetUserMedia-funktio)
Yhteys palvelimelle (Verkkoyhteys joko lähiverkossa tai ulkomaailmasta)
Laitekamera (Koodien keräämisen nopeuttamista varten)

Palvelimen tulee olla lähiverkossa, mutta etäkäytön vuoksi näkyvyys ulkoverkkoon asti on suositeltavaa. Muutoin sovelluksen käyttö rajoittuu asiakkaan lähiverkkoon. Lisäksi asiakkaan toiveet Sähköpostilla toimivasta raportoinnista rajoittuisivat samaiseen lähiverkkoon. Samalla palvelimella tulee olla Web-, Sähköposti- sekä SQL-palvelut, jotta sovelluksen nettisivun ylläpito saadaan pidettyä keskitettynä.
Tietokannan tulee tukea useita arvoja, jotka ilman optimointia toistuisivat useasti jokaisella rivillä. Jokainen tuote sisältää arviolta 9 attribuuttia, jotka voidaan lukea eri tietokannoista.

7.2 Yhteyden sekä palvelimen määrittely

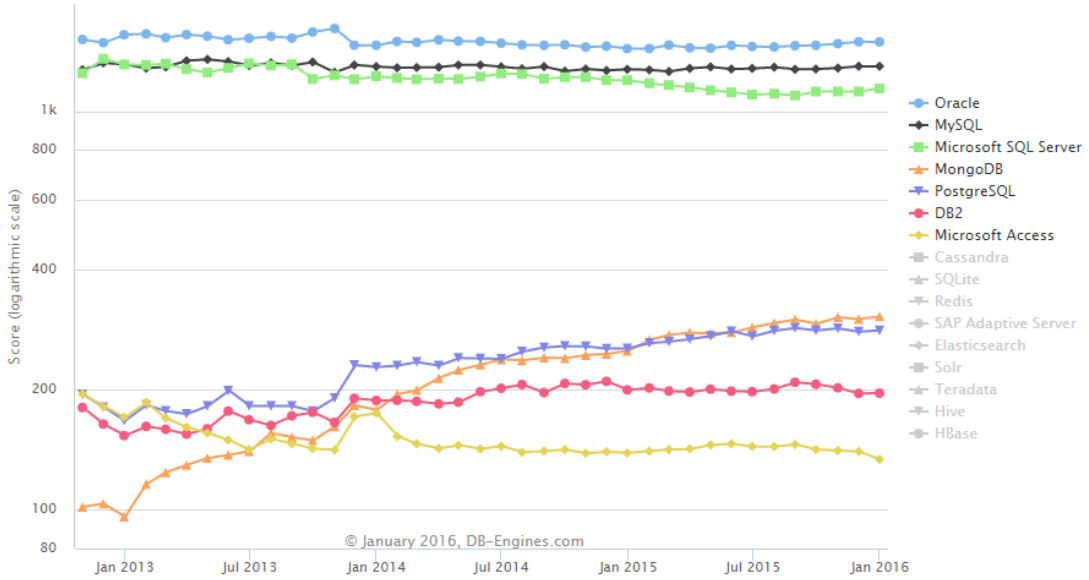
Käyttäjän laite ottaa yhteyden palvelimeen käyttäen HTTPS-protokollaa, jonka yli ladataan HTML5-pohjainen verkkosivu. Verkkosivut toimivat sovelluksen rajapintana käyttäjän ja palvelimen välillä ja nämä toteutetaan kielillä HTML5, PHP sekä JavaScript. Näitä käyttämällä saadaan luotua sovelluksen tärkeimmät toiminnot. Lisätoiminnot luodaan palvelimella ajettavilla skripteillä.

Palvelimen tärkeimmät modulit ovat SQL, Sähköposti, sisäiset skriptit sekä Web (HTML5, PHP, JavaScript).



Kuva 25. Fyysisen palvelimen sisäiset modulit

Vaihtoehtoja SQL-palveluille on useita. Oracle, MySQL sekä MS SQL Server pitävät kolmen kärkeä (db-engines.com www-sivut 2016). Sivusto määrittelee järjestyksen usean eri vaatimuksen perusteella.

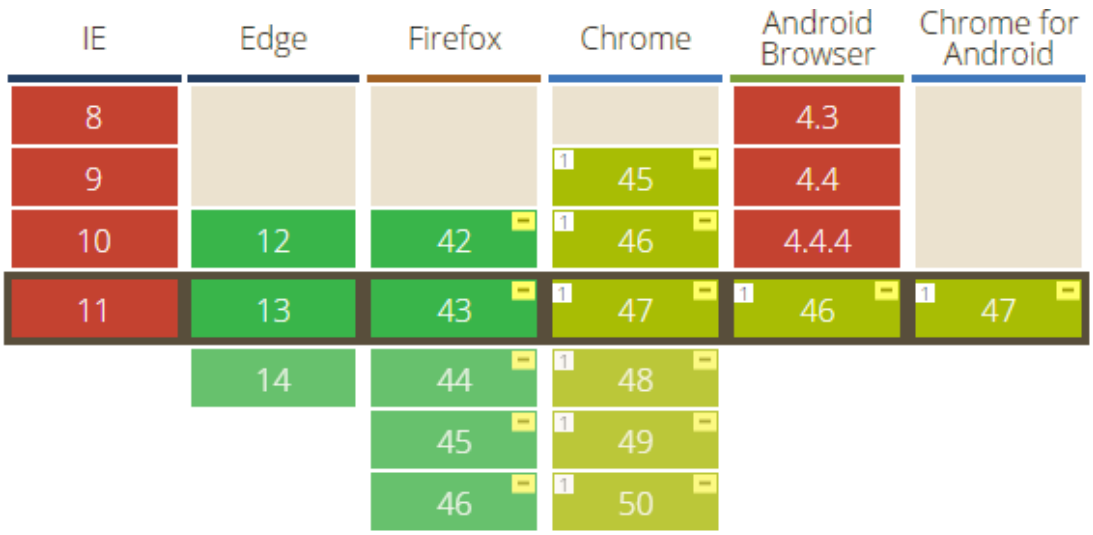


KUVA 26. Viisi suosituinta SQL-tietokantajärjestelmää

Web-palvelun ylläpitämiseksi sopii Apache, joka toimii Unix ja Windows-palvelimissa. Skriptien käytön vuoksi järjestelmäympäristön tulisi olla Unix, sillä skriptien ajo Windows-pohjalla ei ole yhtä suoraviivaista. Lisäksi Unixilla on suora tuki sähköpostin lähettämiselle ”mailx” komennolla. Windows-ympäristössä saman voi toteuttaa esimerkiksi PowerShellillä.

7.3 Laitteen yhteensopivuus ohjelmiston kanssa

Mobiililaitteella kamera, nettiyhteys sekä selain, jossa WebRTC-kapasiteetti.



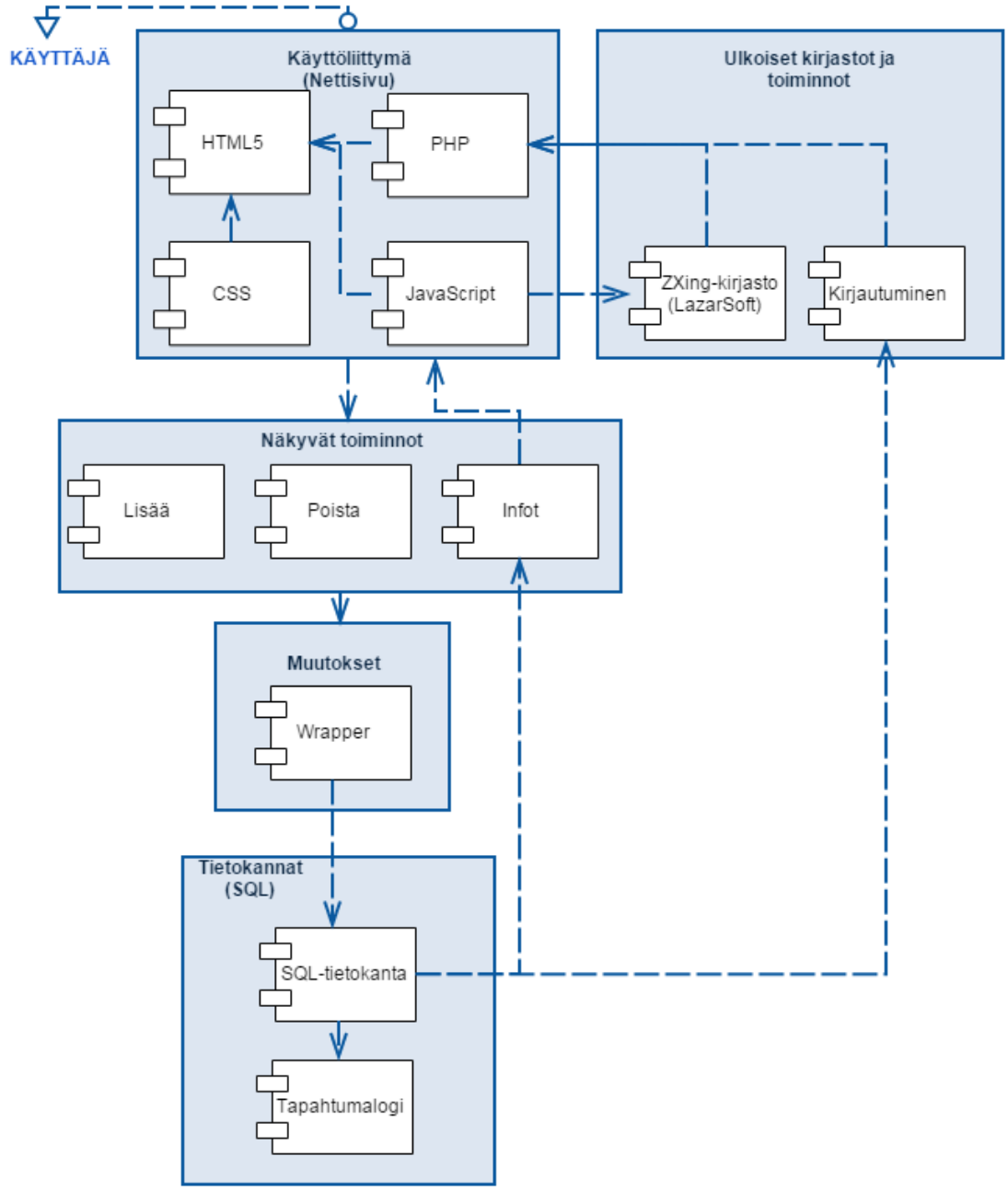
KUVA 27. Suosituimmat selaimet ja GetUserMedia-tuki.

Kuvasta käy ilmi, kuinka yksikään Internet Explorer (IE) versio ei tue GetUserMediaa. Myöskään Android Browserin versiot ennen v46 eivät tue kyseistä rajapintaa. Myrkyn vihreät pohjat tarkoittavat osittaista tukea, tumman vihreä täyttä tukea. Tummalla reunalla merkityt versiot ovat työn kirjoittamisen aikana olleet suosituimpia selainversioita.

Tärkeimmät selaimet työn suhteen ovat Edge, Firefox, Chrome, Android Browser sekä Chrome for Android. Listasta puuttuvat Safari, Opera, iOS Safari sekä Opera Mini, sillä nämä selaimet eivät olleet merkittävässä roolissa työtä tehdessä.

7.4 Toteutuksen menetelmät, rajapinnat

Aiemmin opinnäytetyössä jaoteltiin sovellus pienempiin paloihin. Sovelluksen palat kommunikoivat keskenään rajapintojen avulla. Tärkeimmät rajapinnat ovat HTML5 ja PHP:n välinen tiedonsiirto käyttöliittymään sekä PHP:n ja tietokannan välinen toimintojen suorittaminen.



KUVA 28. Rajapinnat ja niiden väliset relaatiot

7.5 Modulit, Kirjastot, Rajapinnat

Sovellus käyttää valmiita rajapintoja muunmuuassa nettisivun sekä tietokannan välille. Tämä välimaasto toteutetaan PHP:sta löytyvien valmiiden komentojen avulla.

7.5.1 Yhteyden luominen

Yhteyden luominen PHP:n ja tietokannan välille tapahtuu määrittelemällä palvelimen IP-osoitteen, käyttäjänimen sekä salasanan. Yhteys avataan käyttämällä ”new mysqli($osoite, $käyttäjänimi, $salasana). Jos yhteys katkeaa tai epäonnistuu, annetaan siitä ilmoitus.

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

KUVA 29. PHP ja SQL yhteys.

7.5.2 Tiedon lisääminen

Tiedon lisääminen noudattelee aiempaa kaavaa. Ensin määritellään yhteyteen vaadittavat tiedot, jonka jälkeen muokattava tietokanta. Yhteyden avaamisen jälkeen lähetetään tietokannalle tapahtumakomento $sql, jota seuraa tapahtuman tiedot. Alla olevassa esimerkissä annetaan komento ”Lisää tietokantaan, hyllykoodiin attribuutti1, attribuutti2, attribuutti3, arvoilla 00000, 11111, 22222”.
Jos palvelin palauttaa tiedon onnistuneesta tapahtumasta, luodaan siitä ilmoitus. Jos tapahtuma epäonnistuu, annetaan virhekoodi käyttäjälle.

```php
<?php
$servername = "localhost";
$username = "Kayttajanimi";
$password = "Salasana";
$dbname = "Varasto_nimi";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO Hylly_Koodi (Tuote_attribuutti, Tuote_attribuutti, Tuote_attribuutti)
VALUES ('00000', '11111', '22222')";

if ($conn->query($sql) === TRUE) {
    echo "Lisätty onnistuneesti";
} else {
    echo "Virhe: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

KUVA 30. Tuotteen lisäys tietokantaan käyttäen PHP:ta

7.5.3 Tiedon poistaminen

Kokonaisten rivien poistaminen tapahtuu kuten aiemmissa esimerkeissä. Tällä kertaa annetaan komento $sql = DELETE FROM, jota seuraa määritelmät, mitä poistetaan. Esimerkissä poistetaan Hylly_Koodi-nimisestä tietokannasta koko rivi, jos Tuote_Attribuutti on 00000.

```php
<?php
$servername = "localhost";
$username = "Kayttajanimi";
$password = "Salasana";
$dbname = "Varasto_nimi";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// sql to delete a record
$sql = "DELETE FROM Hylly_Koodi WHERE Tuote_Attribuutti=00000";

if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>
```

KUVA 31. Tuotteen poistaminen tietokannasta käyttäen PHP:ta

7.5.4 Tiedon poistaminen

Aiempien esimerkkien lisäksi käytössä on UPDATE-komento, jolla päivitetään määrätylle riville määrätty arvo. Tässä esimerkissä $sql-komennolla annetaan käsky UPDATE (Päivitä), jota seuraa SET (Aseta) ja asetettava arvo. Viimeisenä määritellään mille riville arvo muutetaan.
UPDATE-toiminnolla vältytään poistamasta ja lisäämästä kokonaisia rivejä uudelleen ja säästetään palvelintehoa.

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
}

$conn->close();
?>
```

KUVA 32. Tiedon päivittäminen

7.5.5 Tiedon kerääminen

Tietojen kerääminen tapahtuu valitsemalla tietyt attribuutit tietokannasta. Tietojen valitsemisen jälkeen käyttäjälle näytetään kaikki kerätty data.

```php
<?php
$servername = "localhost";
$username = "Kayttajanimi";
$password = "salasana";
$dbname = "Varasto_nimi";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT Tuote_attribuutti, Tuote_attribuutti, Tuote_attribuutti FROM Hylly_Koodi";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "Tuote_attribuutti: " . $row["Tuote_attribuutti"]. " " . $row["Tuote_attribuutti"]. "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
```

KUVA 33. Infojen kerääminen Hyllyltä.

Kuvassa sovellus valitsee Tuote_attribuuttien perusteella rivin Hylly_Koodin tietuetaulusta ja lähettää tuloksen yhteyden käyttöliittymään sekä tulostaa sen käyttäjälle näkyviin.

7.6 Skanneri

Sovelluksen tärkeimpiin ominaisuuksiin kuuluu laitekameralla käytettävä skanneri. Työ suunniteltiin käyttämään ZXing- sekä LazarSoft-kirjastoja.

7.6.1 Skannerin kirjastot

Toimiva skanneri vaatii seuraavat kirjastot:

ZXing (https://github.com/zxing/zxing)
QR Code scanner (https://github.com/LazarSoft/jsqrcode)
(Lazar Laszlo, www-sivut 2016)

Kirjastot ovat avointa lähdekoodia, tarkoittaen sitä, että kuka tahansa voi käyttää ja muokata koodia tarpeensa mukaan. Skannerista on olemassa valmis esimerkki, joka tukee Firefox-, Chrome- sekä Opera-selaimia (Lazar Laszlo, www-sivut 2016).

LazarSoftin kirjastoa käyttäen on helpointa toteuttaa sovelluksen kamerapohjainen skanneri käytön nopeuttamiseksi. Kirjaston omilla Github-sivuilta löytyvät valmiit ohjeet käyttöönottoa varten, joten niiden kopioiminen työhön ei ole tarpeellista. Prototyyppiä toteuttaessa on parempi tukeutua päivitettyihin tietoihin ja ohjeisiin.

7.6.2 Kirjastojen koodituki

ZXing-kirjasto tukee alla olevia 1D- ja 2D-koodiformaatteja.

1D product	1D industrial	2D
UPC-A	Code 39	QR Code
UPC-E	Code 93	Data Matrix
EAN-8	Code 128	Aztec (beta)
EAN-13	Codabar	PDF 417 (beta)
	ITF	
	RSS-14	
	RSS-Expanded	

KUVA 34. ZXing-kirjaston tuetut koodityypit (ZXing Github www-sivut, 2016)

8 YHTEENVETO

Opinnäytetyön tavoitteena oli suunnitella tuotteistettavissa oleva sovellus eri kokoisten varastojen hallintaan ja delegoituun ylläpitoon, jonka oli tarkoitus olla etäkäytettävä usealla eri päätelaitteella samaan aikaan. Lisäksi sovelluksen tuli olla mahdollisimman helppokäyttöinen ja nopea ja automatisoida suurin osa toiminnoista, joita normaalin varaston ylläpitämiseen kuuluu.

Suurin ongelma toteutuksessa oli eri tuotteiden viivakoodit, joista oli mahdotonta päätellä tuotteen sisältö automatisoinnin kannalta. Eri valmistajien tuotteiden viivakoodit vaihtelevat turhan paljon, jotta niitä kannattaisi opettaa sovellukselle ulkoa. Tämä lopulta johtaisi uuteen tietokantaan eri valmistajista ja tuotteista, jotka käytäisiin läpi aina tuotteen viivakoodia skannatessa. Tämä taas lisäisi tapahtumia palvelimella, joka hidastaisi koko sovelluksen toimintaa.

Tätä työtä käyttäen tulisi olla mahdollista toteuttaa koko kuvailtu Sovellus. Rajoitteita ja mahdollisuuksia työn toteuttajalle asettavat tämän hetkiset ulkoiset kirjastot ja rajapinnat muunmuuassa laitekameran lukuun sekä yleisellä tasolla tehdyt moduuli- ja rajapintakuvaukset.
Lisäksi SDD:n (Software Design Document) avulla Sovelluksen toteuttaminen isolla projektiryhmällä käy helpoksi. Tässä työssä ei SDD:tä raapaista tämän enempää.

Työssä ei käyty läpi Sovelluksen toteuttamista kooditasolla, sillä työn tarkoituksena oli suunnitella, eikä ohjelmoida. Lisäksi vaihtoehtoisia ratkaisuja tietokannan tarkastamiselle tai sähköpostitoiminnolle ei tutkittu tarpeeksi. Käyttäjän antamat tiedot tulisi myös tarkastaa ennen tietokannalle siirtoa, sillä SQL-tietokantoihin keskittyviä hyökkäyksiä on useita erilaisia. Esimerkiksi käyttäjä voisi skannata QR- tai viivakoodin, joka sisältää valmiiksi tietokannalle suoritettavia toimintoja. Näin olisi mahdollista väärentää tietokantaa, tai pyyhkiä se pois kokonaan.

Suunnittelu ei kuitenkaan edennyt täysin ilman tietoturvaa. Käyttäjien autentikaatio, varmuuskopiointi sekä tapahtumien tallentaminen erikseen varmistavat varastotilanteen jatkuvuuden sekä mahdollisten väärinkäyttäjien kiinnijäämisen.

9 POHDINTA

Työssä ilmikäyneet puutteet voisi korjata laajentamalla opinnäytetyötä koskemaan sovelluksen suunnittelua kooditasolle asti. Tämä kuitenkin rajoittaisi niitä, jotka kiinnostuvat sovelluksen omasta toteutuksesta.

Sovelluksen tietoturvaa tullaan käymään tulevaisuudessa läpi yliopistossa tapahtuvan jatkotyön kautta. Vahvempi käyttäjien autentikaatio, sisäisten muuttujien tarkastus ja mahdollinen estäminen sekä palvelimen oman tietoturvan parannus ovat prioriteettien kärjessä.

Valmiin Sovelluksen tulisi olla työntekijän jatke eikä korvike. Nopeakäyttöinen, aina mukana ja käytettävissä. Näihin vaatimuksiin päädyttiin kesällä töissä ollessani Satakunnan Keskussairaalan Lääkintähuollon osastolla. Mielestäni työni sovelluksen suunnittelusta onnistui odotetusti, vaikka alkuperäisiä suunnitelmia päätyikin romukoppaan mittavien työtuntien jälkeen.

LÄHTEET

DB-Engines www-sivut. Viitattu 17.2.2016
http://db-engines.com/en/ranking/relational+dbms

Haikala, I & Märijärvi, J. 2006. Ohjelmistotuotanto. Helsinki: Talentum

Kilpeläinen, A. Homes.jamk.fi www-sivut. Viitattu 7.3.2016
http://homes.jamk.fi/~kivni/http0140/material/relaatiotietokanta.html

Lazar Laszlon www-sivut. Viitattu 17.2.2016
https://webqr.com/about.html

Microsoft-yrityksen www-sivut. Viitattu 21.2.2016
https://support.microsoft.com/fi-fi/kb/283878

Ohjelmistotuotanto. 2000. Vaatimusanalyysi. Viitattu 17.2.2016
http://www.cs.helsinki.fi/u/taina/ohtu/k-2001/luennot/vaatimus/

Tauriainen, S, 2005: OHJELMISTOTESTAUKSEN KEHITTÄMINEN.
Kajaanin ammattikorkeakoulu. Opinnäytetyö.

Tietosuoja.fi www-sivut. Viitattu 17.2.2016
http://www.tietosuoja.fi/text/fi/index/rekisterinpitajalle/kayttotarkoituksenmaarittely-jakasittelynsuunnittelu.html

W3-organisaation www-sivut. Viitattu 17.2.2016
https://www.w3.org/TR/html5/

ZXing Github www-sivut. Viitattu 17.2.2016
https://github.com/zxing/zxing