Saimaan ammattikorkeakoulu
Tekniikka Lappeenranta
Tietotekniikan koulutusohjelma
Tietojärjestelmien kehitys

Tero Ranta

Parsonrussellinterrierit ry:n terveystietokannan suunnittelu

Opinnäytetyö 2015

Tiivistelmä

Tero Ranta
Parsonrussellinterrierit ry:n terveystietokannan suunnittelu, 53 sivua
Saimaan ammattikorkeakoulu
Tekniikka Lappeenranta
Tietotekniikan koulutusohjelma
Tietojärjestelmien kehitys
Opinnäytetyö 2015
Ohjaajat: lehtori Martti Ylä-Jussila, Saimaan ammattikorkeakoulu, Heli Korpinen, Parsonrussellinterrierit ry

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa Parsonrussellinterrierit ry:n terveystietokanta. Opinnäytetyön asiakkaana oli Parsonrussellinterrierit ry.

Opinnäytetyön teoriaosassa käsitellään tietokannan määrittelyä, suunnittelua ja dokumentointia.

Käyttöliittymän tietokantaan toteutti opinnäytetyönään Tuomas Porvali.

Tietokannan kehitys eteni ketterää menetelmää soveltaen. Asiakkaan edustaja arvioi työtä viikoittain.

Lopputuloksena saatiin toimiva ja vaatimusten mukainen tietokantapalvelu, joka otettiin käyttöön keväällä 2015.

Avainsanat: tietokanta, MySQL, määrittely, suunnittelu, toteutus, dokumentointi.

Abstract

Tero Ranta
Health Database for Parsonrussellinterrierit ry, 53 pages
Saimaa University of Applied Sciences
Technology Lappeenranta
Information Technology
Information Systems
Bachelor's Thesis 2015
Instructors: Senior Lecturer Martti Ylä-Jussila, Saimaa University of Applied Sciences, Heli Korpinen, Parsonrussellinterrierit ry

The goal of the thesis was to design and produce a health database for Parsonrussellinterrierit ry. The project was commissioned by Parsonrussellinterrierit ry.

The thesis deals with specification, design and implementation of a database.

The web user interface for the database was created by Tuomas Porvali.

The database was specified, designed and implemented using an agile software development methods. The customer evaluated the work weekly.

A fully functional database and a web user interface were created which were deployed in spring 2015.

Keywords: database, MySQL, specification, design, implementation, documentation.

Sisällys

Termit ja käsitteet

Apache	Verkkopalvelin.
CMS	Sisällönhallintajärjestelmä.
CSS	Cascading Style Sheets. Tyyliohje www-sivujen näyttämiseksi.
ER-kaavio	Entity-Relationsip-malli, jotka käytetään tietosisällön kuvaamiseen.
FTP	File Transfer Protocol. Tiedostonsiirtoprotokolla.
Gantt-kaavio	Projektinhallinnassa käytettävä janakaavio, jolla esitetään työvaiheiden edistymistä.
HTML	Hyper Text Markup Language. Kieli, jolla tehdään verkkosivuja.
Kenttä	Yksi rivi taulussa.
MySQL	Tietokantaohjelmisto.
OVH-Hosting	Webhotelleja tarjoava palvelu.
Palvelin	Palvelin on tietokone tai laite, jolla voidaan säilyttää verkkosivuja.
PHP	Hypertext Preprocessor, palvelinkieli.
phpMyAdmin	Tietokantasovellus, jotka käytetään selaimen avulla.
Pääavain	Yksilöivä avain, joka erottaa tietokannan käsitteet.
Relaatiotietokanta	Yleisin tietokantamuoto.
Root-tunnus	Pääkäyttäjän tunnus.
Scrum	Ketterä kehitysmenetelmä.
SQL	Structured Query Language, relaatiotietokannan ohjelmointikieli.
Taulu	Tietokannan käsitteet.
TCP	Transmission Control Protocol, tietoliikenneprotokolla.
Tietokanta	Koottu tietovarasto.
UML 2	Unified Modeling Language 2, mallinnuskieli.

Webhotelli	Vuokrattava tila verkossa, jota voi käyttää esimerkiksi kotisivujen säilytyspaikkana.
Viiteavain	Käytetään relaatiotietokannan aputauluissa purkamaan moni-moneen-yhteyksiä.
XAMPP	Ohjelmistokokoelma, jolla verkkosivuja ja palveluita voidaan ajaa Windows-laitteessa.

1 Johdanto

Opinnäytetyön tavoitteena on suunnitella ja toteuttaa terveystietokanta Parsonrussellinterrierit ry:n tarpeisiin. Tietokanta kokoaa yhdistyksen jäsenten omistamien koirien terveystiedot yhteen paikkaan, jolloin niitä on helppo muokata ja tarkastella keskitetysti.

Asiakas on Parsonrussellinterrierit ry, jonka jäsenkunnan muodostavat kyseisestä koirarodusta kiinnostuneet henkilöt. Asiakkaan edustajana toimii Heli Korpinen, joka on myös yhdistyksen sihteeri ja terveystietokannan pääasiallinen ylläpitäjä. Tietokanta määriteltiin asiakkaan toivomusten mukaan ja asiakas oli mukana koko projektin ajan varmistamassa, että järjestelmästä tulee tarkoituksenmukainen ja toimiva.

Parsonrussellinterrierit ry:n terveystietokantaa käytetään verkkopohjaisella käyttöliittymällä, jonka toteutuksesta vastasi Tuomas Porvali.

2 Ohjelmistotuotannon menetelmät

Ohjelmistotuotantoa voidaan lähestyä monella eri tavalla. Onnistuneen ohjelmiston tuottamiseksi kehitys voidaan jakaa osiin ja valittava ohjelmistotuotannon menetelmä riippuu ohjelmiston laajuudesta, työmäärästä ja laadusta. Tätä kutsutaan vaihejakomalliksi ja niihin lukeutuu sekä perinteisiä että ketteriä menetelmiä. Eri mallien erot ovat lähinnä siinä, missä vaiheessa eri vaiheet toteutetaan kehityksen edetessä. (Haikala & Mikkonen 2011.)

Monet ohjelmistot toteutetaan projektimallisena, joka vaatii suunnittelua, henkilökuntaa ja riittävää dokumentaatiota. Projektinmuotoisen toiminnan tarkoituksena on hallita laatua, aikataulua ja budjettia mahdollisimman tehokkaasti.

2.1 Projektinhallinta

Projektinhallinta on resurssien hallintaa niin, että ne käytetään tehokkaasti ja varmistetaan laadukas lopputulos. Projektilla on tavoite, jonka saavuttamiseksi se jaetaan pienempiin osakokonaisuuksiin. Osien työmäärää on helpompi arvioida sekä selvittää niiden välisiä riippuvuuksia. Jokin työvaihe ei ehkä voi alkaa ennen kuin edellinen on saatu valmiiksi. Nämä riippuvuudet ovat kriittisiä koko projektin aikataulun kannalta. (Haikala & Mikkonen 2011.)

Projektinhallintaa johtaa projektipäällikkö, jonka tehtävänä on varmistaa rajallisten resurssien riittävyys, joita ovat esimerkiksi raha, aika, työvoima ja työvälineet. Projektipäällikön pitää myös huolehtia tarpeellisesta informaation kulusta projektiryhmän, asiakkaan ja muiden sidosryhmien kesken.

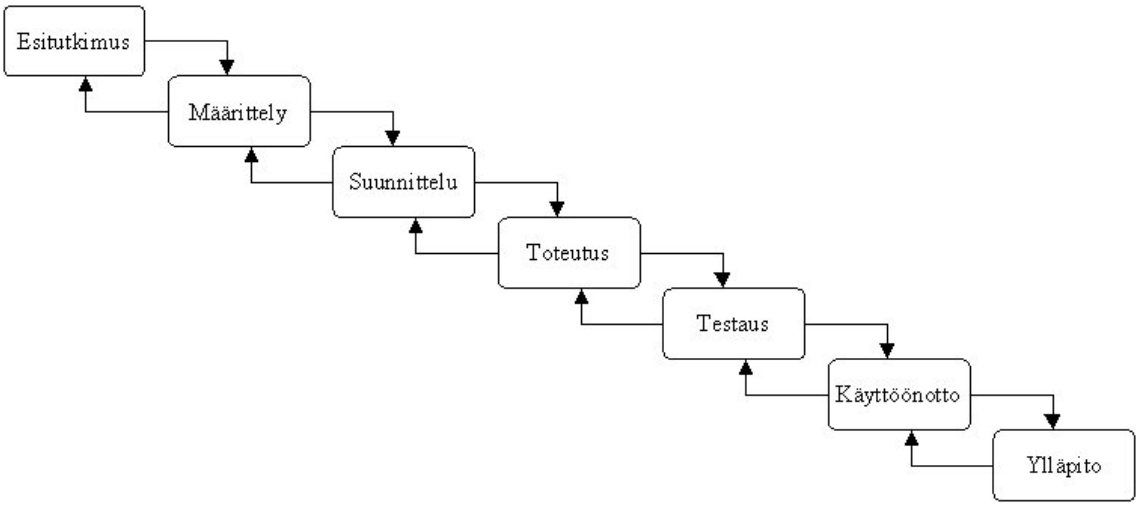
2.2 Vaihejakomalli

Vaihejakomalleja on kehitetty useita erilaisia, eikä voida sanoa, että jokin tietty olisi aina parempi kuin toinen. Perinteiset mallit ovat vakiinnuttaneet asemansa vuosien kuluessa ja niitä käytetään monesti juuri siksi, että niihin on totuttu. Uudet mallit haastavat vanhoja, mutta tärkeintä on, että henkilökunta on koulutettu käyttävään malliin. Mallia ei kannata vaihtaa vain muutoksen vuoksi, vaan siihen pitää olla konkreettinen syy.

2.2.1 Vesiputousmalli

Yksi perinteisistä vaihejakomalleista on vesiputousmalli. Siinä edetään ikään kuin ylemmältä tasolta alemmalle, suorittaen aina edellinen vaihe ennen seuraava. Olisi kuitenkin tärkeää myös iteroida edelliseen vaiheeseen, jotta vesiputousmenetelmä toimisi halutulla tavalla. Vesiputousmalli on perustana monille muillekin malleille mukaan lukien ketterät menetelmät. (Haikala & Mikkonen 2011, s. 37.)

Perinteisessä vesiputousmallissa on viisi vaihetta: määrittely, suunnittelu, toteutus, testaus ja käyttöönotto. Voidaan puhua myös seitsemästä erillisestä vaiheesta (Kuva 1), kun mukaan otetaan esitutkimus ja ylläpitovaiheet. (Hiltunen 2004)



Kuva 1. Vesiputousmalli (Hiltunen 2004)

Esitutkimusvaiheessa päätetään, onko järjestelmää kannattavaa alkaa kehittää vai ei. Tavoitteena on määrittää lähtökohdat kehittämishankkeelle ja raportoida päättäville tahoille esimerkiksi organisaation nykytilanne, ongelmat ja ratkaisut niihin. Kuvaillaan myös sidosryhmät, rajaukset, tavoitteet ja tehdään alustava suunnitelma ohjelmistohankkeen läpiviemiseksi. (Hiltunen 2004.)

Määrittelyvaiheessa päätetään, mitä ohjelmiston olisi tarkoitus tehdä valmistuttuaan. Tuotetaan selkeä toiminnallinen määrittelydokumentti, johon kuvataan jokainen ohjelmiston toiminto, tietokanta, tieto ja rajapinta yksityiskohtaisesti.

Suunnitteluvaiheessa toiminnallinen määrittely muutetaan tekniseksi määrittelyksi. Siinä suunnitellaan tietokannan arkkitehtuuri ja moduulit, jolloin ohjelmisto voidaan jakaa mahdollisimman itsenäisiksi moduuleiksi ja jakaa se työntekijöiden tehtäväksi.

Toteutusvaiheessa ohjelmoidaan ohjelmisto valitulla ohjelmointikielellä ja kootaan edellisen vaiheen moduulit toimivaksi kokonaisuudeksi. Toteutunutta järjestelmää verrataan asetettuihin vaatimuksiin ja suunnitteludokumentteihin.

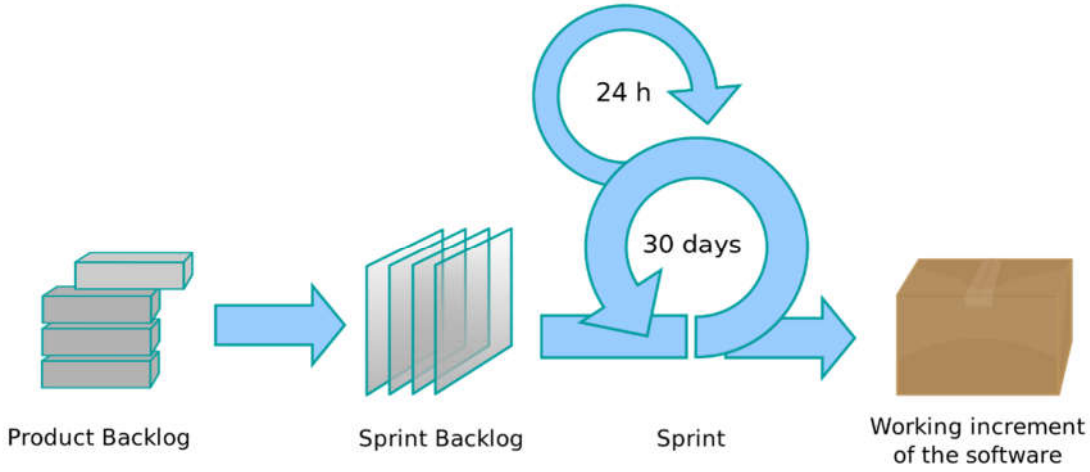
Testausvaiheessa etsitään virheitä eri menetelmillä. Näitä ovat esimerkiksi moduulitestaus, integrointitestaus ja systeemitestaus. Testausta varten laaditaan testaussuunnitelma.

Käyttöönottovaiheessa ohjelmisto tai tietokanta siirretään lopulliselle alustalleen esimerkiksi palvelimelle webhotelliin. Käyttäjälle annetaan koulutusta ja ohjeistusta järjestelmän käyttämiseksi itsenäisesti.

Ylläpitovaihe alkaa kun ohjelmisto on valmistunut ja luovutettu käyttäjälle ja voi kestää pitkäänkin. Sen onnistuminen riippuu dokumentoinnin laadusta ja määrästä.

2.2.2 Ketterät menetelmät

Ketterillä menetelmillä tavoitellaan ohjelmistokehityksessä riskien minimointia suorittamalla lyhyitä pyrähdyksiä. Pyrähdykset ovat usein noin viikon mittaisia, jonka jälkeen arvioidaan tehtyä työtä ja tulevia työtehtäviä. Tarkoituksena on ensisijaisesti tuottaa toimiva ohjelmisto, kun perinteisissä menetelmissä painotetaan laajaa ja huolellista dokumentointia. Ketterissä menetelmissä tärkeää on projektiryhmän välinen kommunikointi ja nopea reagointi muutoksiin. (Wikipedia: Ketterä ohjelmistokehitys.) Tässä työssä sovellettiin ketterää menetelmää Scurmia (Kuva 2).



Kuva 2. Scrum (Wikipedia: Scrum)

Scrumin ideana on iteratiivinen kehitystyö, jossa pidetään palaveri muutaman viikon välein. Palaverin aikana keskustellaan seuraavan pyrähdyksen sisällöstä (sprint backlog) ja arvioidaan edellisen jakson työtä. Iteraation tuloksena syntyvä ohjelmisto voidaan esitellä asiakkaalle, jolloin saadaan nopeasti palautetta ja voidaan suunnitella tarvittavia muutoksia.

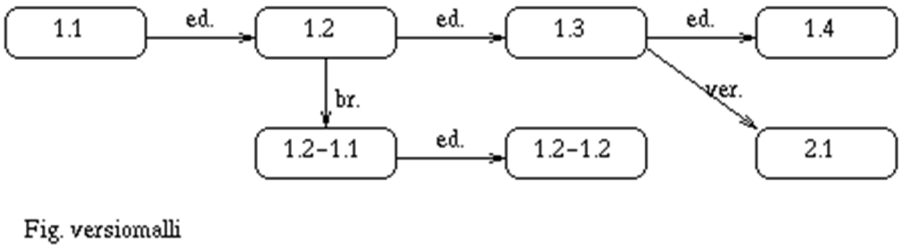
Normaalissa Scrum-ryhmässä on tuotteen omistaja, Scrum-mestari ja tiimin jäsenet. Tässä työssä sovellettiin menetelmää niin, että varsinaista Scrum-mestaria ei ollut vaan Scrum-ryhmän jäsenet tekivät päätökset ja pitivät huolta tehdyistä töistä ja menetelmistä itse. Tässä työssä ei myöskään tehty Scrum-menetelmään kuuluvia päiväpalavereita, mutta itse pyrähdysten tiheys oli noin viikon. (Haikala & Mikkonen 2011.)

2.3 Laadunhallinta

Ohjelmiston laatu voidaan varmistaa toimenpiteillä kuten testaus, tarkastus ja katselmointi. Ohjelmiston laatua on edullisinta valvoa jo kehitysvaiheessa käyttäen hyväksi syntyneitä dokumentteja tarkastuksissa. Katselmoinnissa asiakkaan edustaja on mukana, jolloin hän saa myös käsityksen ohjelmiston toiminnasta ja käytöstä. Testausta tulisi suorittaa usein ja aikaisessa vaiheessa, sillä myöhäisessä vaiheessa löydetyt viat ja puutteet ovat vaikeampia ja kalliimpia korjata. (Haikala & Mikkonen 2011, s. 197 - 200.)

2.4 Versionhallinta

Versionhallinnalla varmistetaan, että ohjelmiston eri versiot ovat käytettävissä, dokumentoituja ja niiden vastuuhenkilö on selvillä. Myös versionumero, version tila ja muutoksen ajankohta on hyvä dokumentoida. Versionhallinnassa käytetään versiopuuta (Kuva 3), jonka haarat kuvaavat kohoavia versionumeroita ja myös haarautuvia versioita samasta ohjelmistosta. Tätä tarvitaan, jos halutaan käyttää ohjelmaa esimerkiksi eri alustoilla tai laitteistoissa. Liiallinen versioiden haarautuminen johtaa työmäärän lisääntymiseen ylläpitovaiheessa. (Haikala & Mikkonen 2011. s. 172.)



Kuva 3. Versiopuu (Kokko 1996)

2.5 Dokumentointi

Perinteisen ohjelmistokehityksen aikana syntyy runsaasti dokumentteja, jotka tulee pitää ajan tasalla. Dokumentit on hyvä päivittää jo muutoksen syntyessä, viimeistään kuitenkin projektin päättyessä. On hyvä arvioida, mitä kannattaa dokumentoida, sillä ylimääräinen dokumentointi aiheuttaa paljon työtä ja mahdollisia ristiriitaisuuksia, mikäli samaa asiaa käsitellään useassa dokumentissa. (Haikala & Mikkonen 2011.)

Ensimmäinen laadittava dokumentti on projektisuunnitelma. Vaikka suunnitelma olisi vasta luonnos, se muodostaa kuitenkin perustan koko ohjelmistoprojektille ja määrittää muun muassa aikataulun, resurssit ja tavoitteet.

Vaatimusmäärittelyssä määritellään projektin tavoitteet ja kuinka niihin päästään. Vaatimuksiin kuuluvat sekä toiminnalliset, että ei-toiminnalliset vaatimukset ja sen tulee olla niin tarkka, että kehitystyö voidaan perustaa sen varaan. Toimin-

nallisiin vaatimuksiin kuuluvat tietosisältö ja toimintovaatimukset, yhteensopivuus, standardit, turvallisuus ja toimintaympäristö. Ei-toiminnallisia vaatimuksia ovat resurssit ja laadulliset vaatimukset. (Immonen 2002.)

Tietokannan suunnitteludokumentissa kuvataan järjestelmään liittyvät määritykset ja toiminnot loogisesti ja teknisellä kielellä. Sen tehtävänä on kertoa, miten järjestelmä toteutetaan. Suunnittelu jaetaan usein kahteen osaan eli arkkitehtuuri- ja moduulisuunnitteluun. Arkkitehtuurisuunnitelmassa järjestelmä jaetaan itsenäisiin moduuleihin ja kuvataan niiden rajapinnat. Moduulisuunnittelussa kunkin moduulin rakenne ja toiminta määritellään. (Immonen 2002.)

3 Toiminnallinen määrittely

Toiminnallisen määrittelydokumentin tarkoituksena on kuvata tietokannan sekä tietojärjestelmän pääpiirteet ja tärkeimmät toiminnot. Vaatimusmäärittelydokumentti toimii sopimuksena asiakkaan ja projektiryhmän välillä. Dokumentti toimii myös pohjana testaussuunnitelman laatimiselle.

3.1 Toiminnallisen määrittelyn sisältö

Toiminnallinen määrittelydokumentti sisältää jokaisen toiminnon yksityiskohtaisen kuvauksen. Siinä määritellään tietokannat ja niihin talletettavien tietojen tyyppi, laatu, määrä ja pituus. Myös järjestelmään liittyvät rajapinnat kuvataan toiminnallisessa määrittelyssä. (Hiltunen 2004.)

Järjestelmän toiminnot kuvataan käyttötapauskaaviossa, jonka perustella laaditaan erilliset käyttötapaukset. Käyttötapauksissa tulee ilmetä jokainen toiminto, joka järjestelmässä voidaan tehdä. Käyttötapaukseen sisältyy tapauksen yleiskuvaus, prosessi joka johtaa kyseiseen käyttötapauksen, käyttäjän rooli järjestelmässä, käyttötapauksen sanallinen ja tarkka kuvaus, mahdolliset poikkeukset käyttötapauksen aikana ja käyttötapauksen lopputuloksen (Kuva 4).

Koiran lisääminen

Yleiskuvaus: Käyttäjä lisää koiran
Laatija: Tuomas Porvali.
Prosessi: Koiranlisäys- ja muokkausprosessi

Käyttäjäroolit: Hallinnoija ja omistaja

Esitiedot/Ehdot: Oletetaan, että käyttäjä on kirjautunut sivustolle

Käyttötapauksen kuvaus

1. Käyttäjä klikkaa Lisää koira -painiketta.
2. Käyttäjä täyttää lomakkeen kentät.
3. Käyttäjä klikkaa lisää-painiketta.
4. Käyttäjä voi lähettää viestin jalostustoimikunnalle.
5. Koira on lisätty onnistuneesti sivustolle.

Poikkeukset

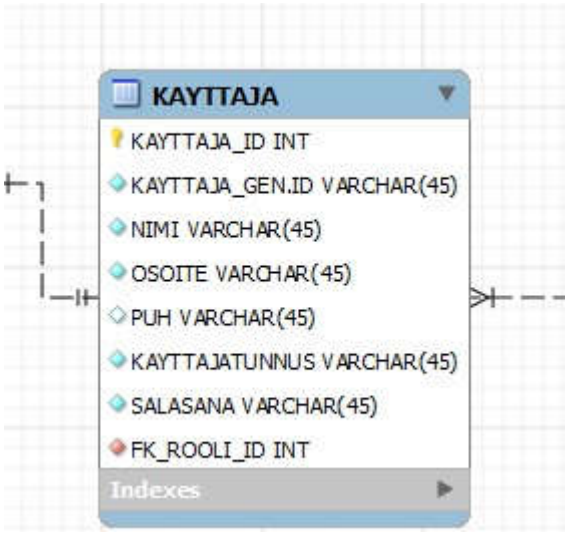
1. Käyttäjä jättää kentät tyhjiksi. Sivusto ilmoittaa tyhjistä kentistä.
2. Käyttäjä yrittää asettaa tiedoston, joka ei ole kuva. Sivusto antaa virheviestin.
3. Käyttäjä yrittää asettaa kuvan, joka on liian suuri. Sivusto antaa virheviestin.
4. Käyttäjä ei anna sähköpostiosoitetta tai jättää viestikentän tyhjäksi. Sivusto ilmoittaa, että viestiä ei lähetetä.
5. Käyttäjä yrittää lähettää haitallisen sähköpostiviestin. Viestiä ei lähetettä.

Lopputulos: Käyttäjä on lisännyt koiran.

Muut vaatimukset: Kuvan asettaminen ja jalostustoimikunnan viestin lähettäminen on vapaaehtoista.
Käyttötiheys: n. 50 päivässä
Muuta: Ei ole

Kuva 4. Tyypillinen käyttötapaus järjestelmässä

Ohjelmiston tietokannat kuvataan yksityiskohtaisesti toiminnallisessa määrittelyssä. Dokumentista tulee ilmetä, millaista tietoa tietokantaan tallennetaan ja missä muodossa (Kuva 5). Tietokantakaavio selventää visuaalisesti taulujen välisiä relaatioita ja tallennettavan tiedon laatua.



Kuva 5. Osa tietokannasta

Tietokannan tauluihin tallennetavat tiedot esitellään ja määritellään niin, että ne voidaan toteuttaa valitulla ohjelmointikielellä (Kuva 6). Määritellään myös, mitkä tiedot ovat pakollisia ja mitkä luodaan järjestelmän toimesta. Ylläolevassa esimerkissä automaattisesti luotavia tietoja ovat KAYTTAJA_ID ja FK_ROOLI_ID.

KOIRA	
KOIRA_ID	INT(32)
KOIRA_GEN.ID	VARCHAR(45)
REKNRO	VARCHAR(45)
NIMI	VARCHAR(45)
SYNTAIKA	DATE
KUOLLUT	BOOL
OMISTAJA	VARCHAR(45)
FK_TSP_ID	INT(32)
FK_SUKUPUOLI_ID	INT(32)
FK_ULKOMUOTO_ID	INT(32)
FK_KUVA_ID	INT(32)
FK_KASVATTAJA_ID	INT(32)

Kuva 6. Tietokannan eräs taulu

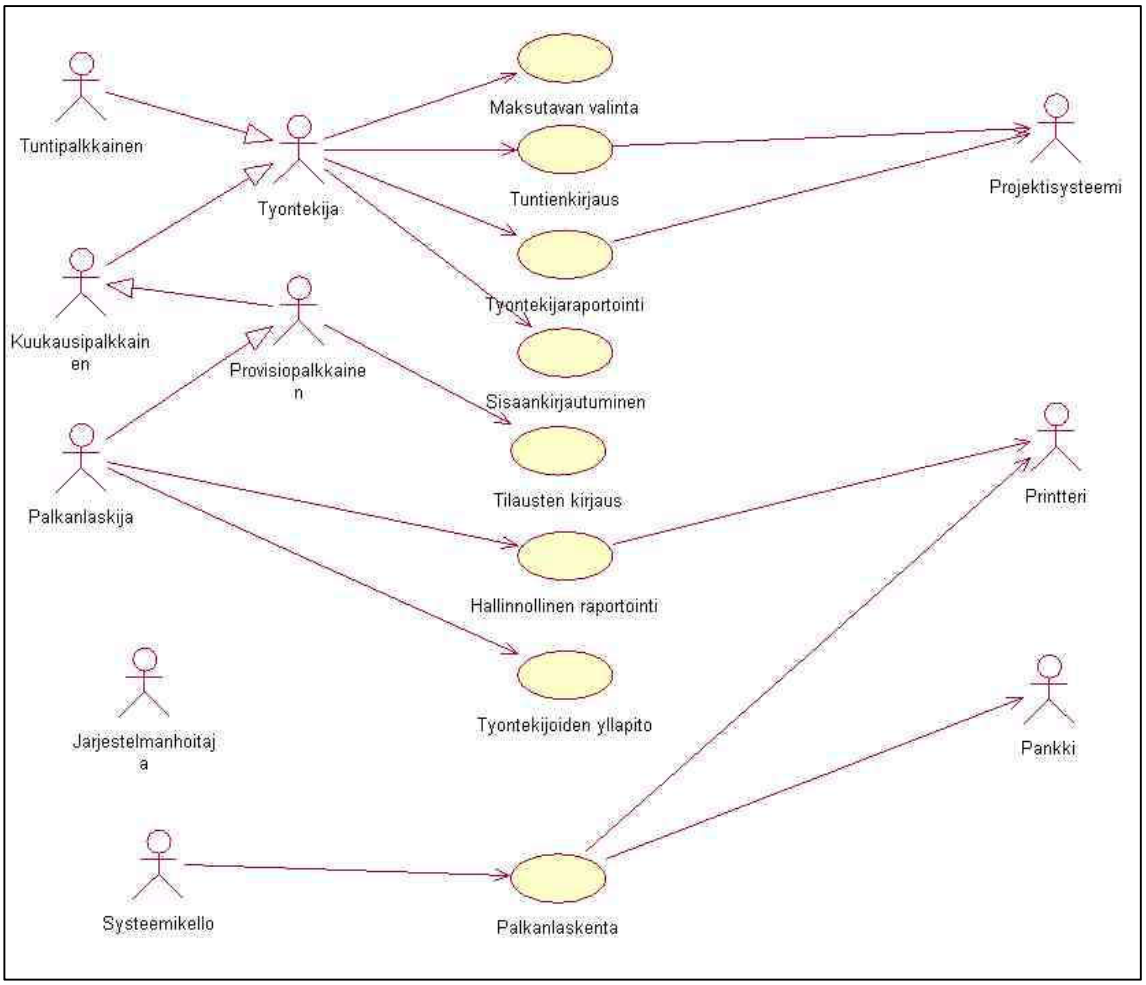
3.2 Toiminnallisen määrittelyn dokumentointimenetelmät

Toiminnallinen määrittelydokumentti sisältää sekä sanalliset selitykset että kaaviot. Kaavioilla voidaan näyttää kerralla esimerkiksi tietokannan rakenne ja järjestelmän käyttötapaukset, ja ne täydentävät sanallisia kuvauksia. Kaaviot laaditaan UML 2 -standardin mukaisesti, jottei niiden tulkinnasta synny epäselvyyttä. (Haikala & Mikkonen 2011, s. 79.)

UML 2

UML 2 on standardoitu mallinnuskieli, jossa ohjelmistoa kuvataan graafisesti. Se sisältää 13 erilaista kaaviota, joilla eri osa-alueet voidaan esittää yksiselitteisesti ja ymmärrettävästi. Kaavioilla voidaan kuvata ohjelmiston rakennetta, käyttäytymistä ja vuorovaikutusta sisäisesti tai ulkoisesti. (Haikala & Mikkonen 2011.)

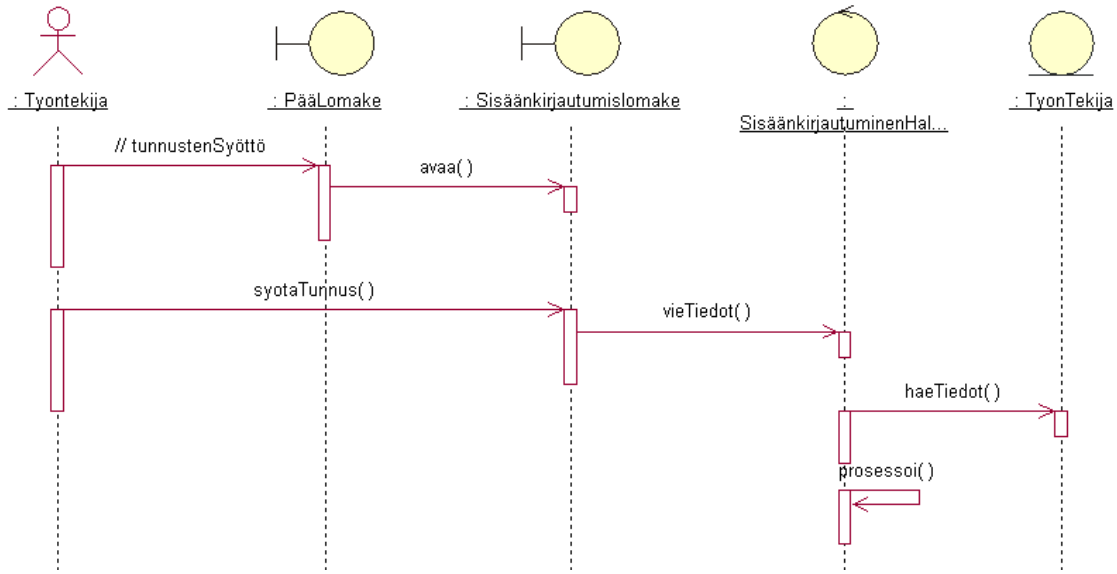
Käyttötapauskaaviolla (Kuva 7) kuvataan ohjelmiston toimijoiden (aktorit) käyttäytymistä ja toimenpiteitä järjestelmässä. Käyttötapauskaavio on hyvä pitää yksinkertaisena ja selkeänä, jotta erillisten käyttötapausten tunnistaminen on mahdollista. Toimijat voivat olla joko henkilöitä, laitteita tai toisia ohjelmistoja. (Haikala & Mikkonen 2011, s. 79.)



Kuva 7. Käyttötapauskaavio

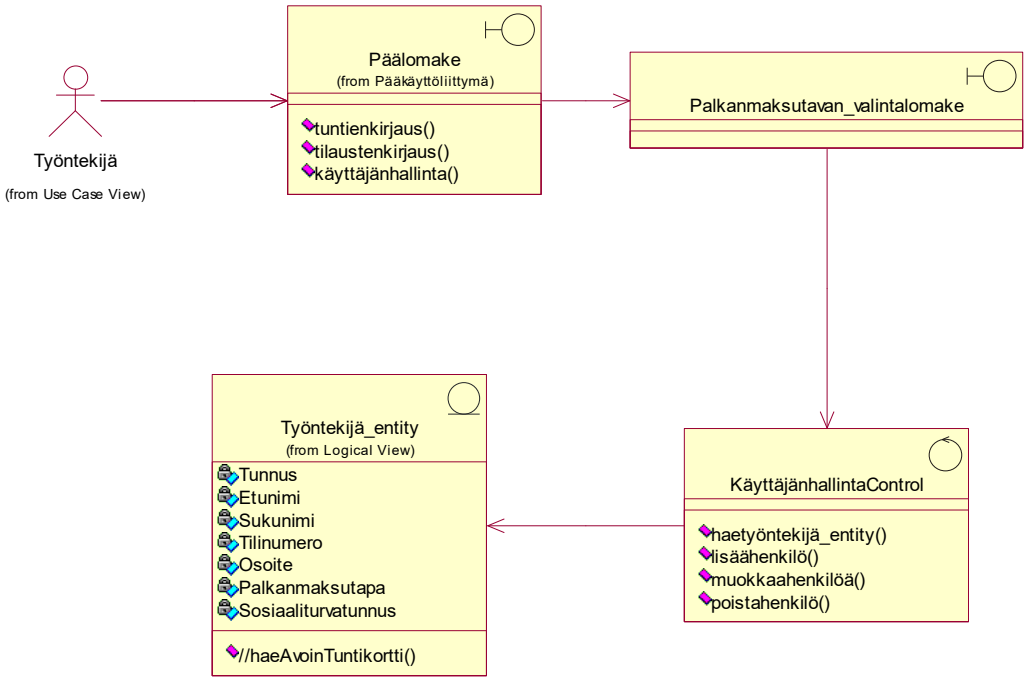
Käyttötapauksella kuvataan yksittäistä käyttötapausta mahdollisimman tarkasti ja yksiselitteisesti. Käyttötapaus voi olla esimerkiksi sisäänkirjautuminen järjestelmään, jolloin kuvataan käyttötapaus siitä hetkestä alkaen, kun toimija aloittaa sisäänkirjautumisen ja päättyy onnistuneeseen sisäänkirjautumiseen. Käyttötapauksessa esitellään siis alkutila, tapahtuman kulku, mahdolliset virheet toiminnan aikana ja lopputila sekä poikkeukset.

Sekvenssikaaviolla (Kuva 8) kuvataan yksittäisen käyttötapauksen eteneminen järjestelmässä. (Haikala, 2011.)



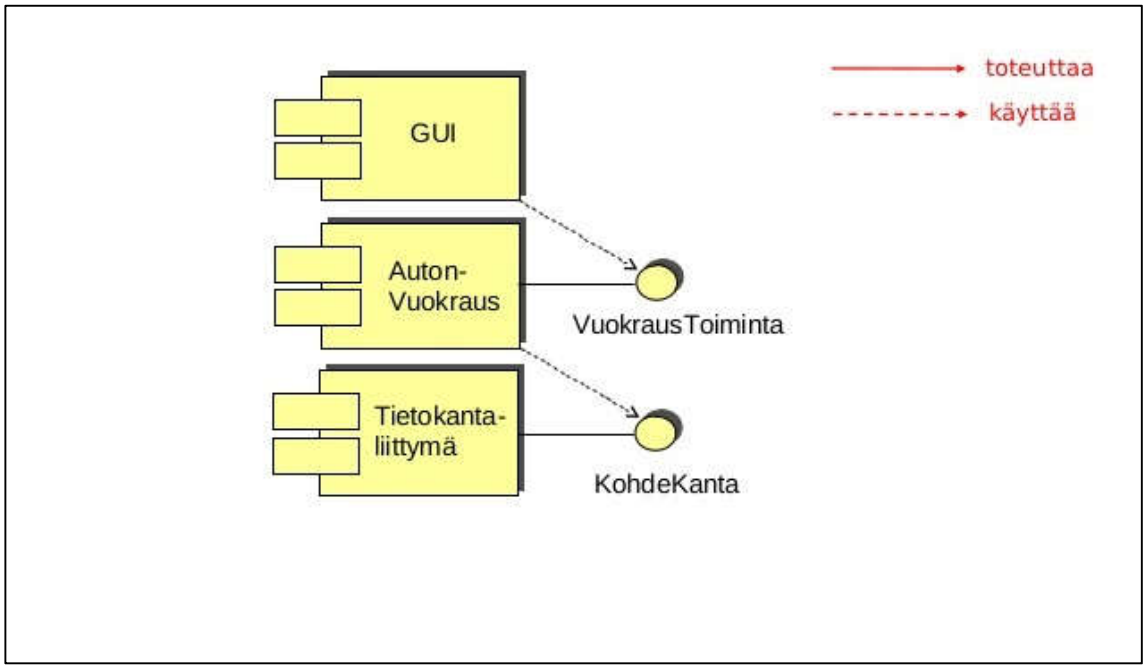
Kuva 8. Sekvenssikaavio

Olio-ohjelmoinnissa luokkakaaviolla (Kuva 9) kuvataan eri luokkien sisältö ja niiden suhteet toisiinsa. (Haikala 2011.)



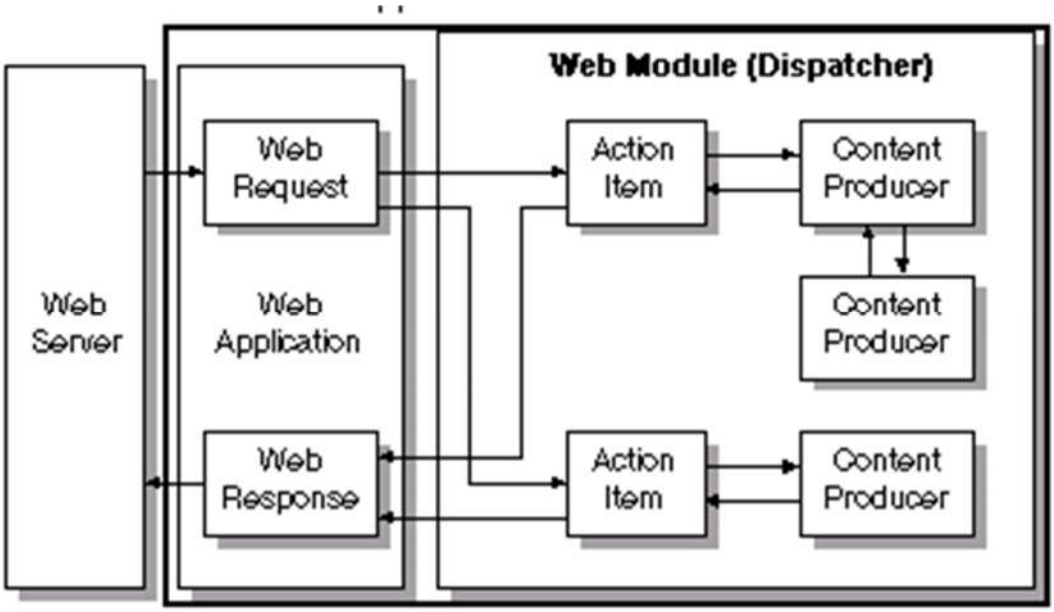
Kuva 9. Luokkakaavio

Komponenttikaaviolla (Kuva 10) kuvataan ohjelmiston eri komponenttien yhteyksiä ja sidoksia toisiinsa ja niiden välistä vuorovaikutusta. Kaaviolla voidaan kuvata myös komponenttien yhteyksiä muihin järjestelmiin (Haikala 2011.)



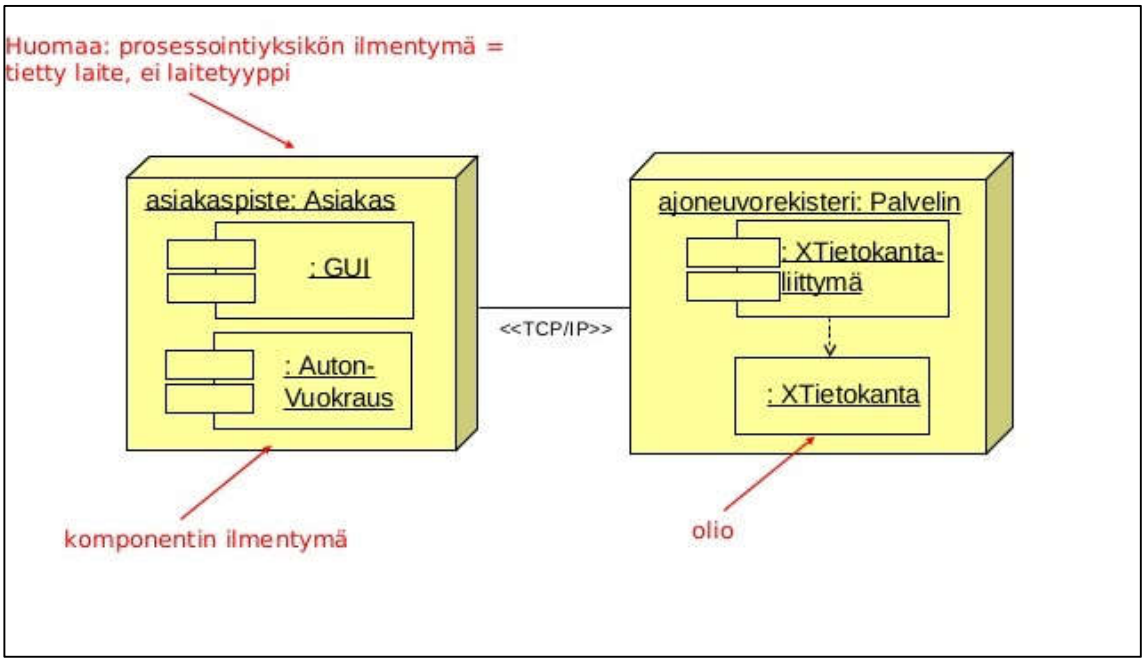
Kuva 10. Komponenttikaavio (Kellokoski 2013)

Arkkitehtuurikaaviolla (Kuva 11) kuvataan koko järjestelmän rakennetta (Haikala 2011).



Kuva 11. Arkkitehtuurikaavio (Kämäri 2011)

Sijoittelukaaviolla (Kuva 12) kuvataan ohjelman sijoittelua fyysisiin laitteisiin ja palvelimiin. Sen avulla kuvataan myös prosessorien väliset verkkoyhteydet ja protokollat.



Kuva 12. Sijoittelukaavio (Kellokoski 2013)

4 Tietokannan suunnittelu

Tietokantaan on koottu tietoa, jolla on jokin merkitys. Tietokantatyyppejä on erilaisia, mutta käytetyin on relaatiotietokanta. Tietokannan sisältämää tietoa käsitellään tietokantakielellä, kuten SQL (Structured Query Language). Relaatiotietokanta perustuu malliin, jossa tietorakenteita ja niiden välisiä relaatioita kuvataan. Mallia kutsutaan relaatiomalliksi. (Hovi 2003, s. 7 – 8; Lahtonen 2002, s. 2 - 4.)

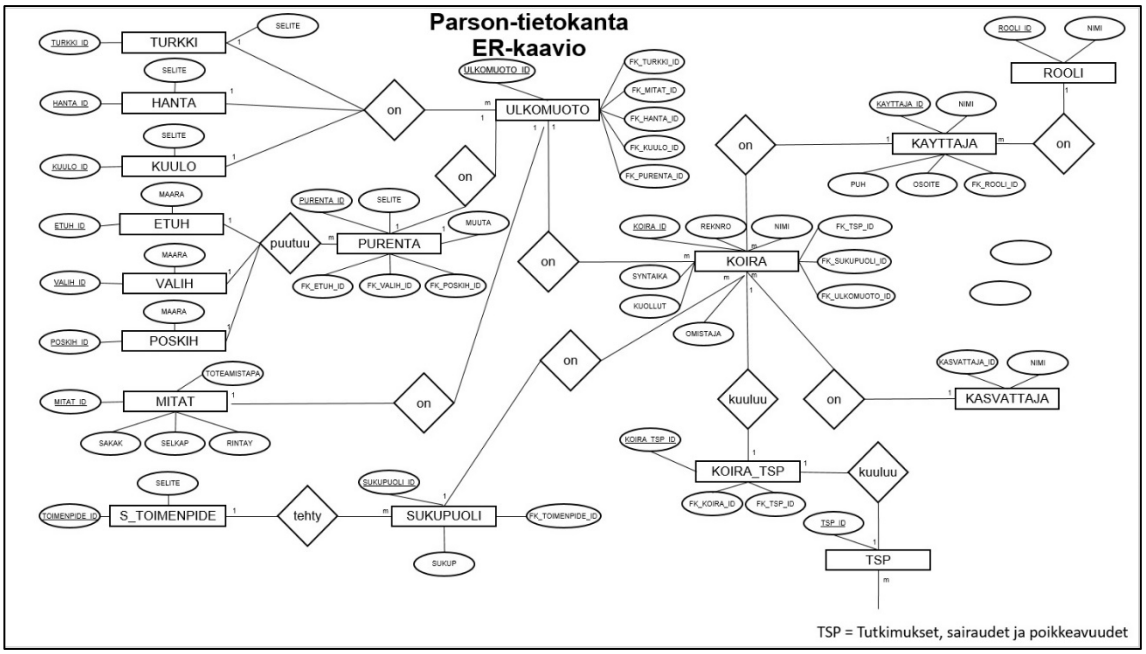
Tietokannan suunnittelu on laaja ja monivaiheinen prosessi. Sen merkitys on suuri koko järjestelmän kehityksen onnistumisen kannalta. Tietokannan suunnittelun aikana tietokanta mallinnetaan ja siihen kuulu myös sen fyysinen suunnittelu. Tietokannan suunnittelun vaiheet ovat: käsiteanalyysi, käsitemalli, tarveanalyysi, normalisointi, toteutus ja taulumääritykset. (Hovi 2003, s. 24.)

Tämän opinnäytetyön aikana toteutettu järjestelmä sisälsi melko laajan tietokannan. Suunnitteluvaiheet suoritettiin osittain ennen työn aloittamista, mutta ne viimeisteltiin vasta projektin loppusuoralla.

4.1 Käsiteanalyysi

Käsiteanalyysillä tarkoitetaan tietokannan havainnollistamista loogisesti sen sijaan, että sitä tarkasteltaisiin teknisellä tasolla. Tämän opinnäytetyön aikana tutustuttiin Parsonrussellinterrierit ry:n toimintaan asiakkaan edustajan avulla sekä hänen toimittamaansa ominaisuusluetteloon. Näistä selvisi tarvittavat tiedot koirista ja terveystietokantaan sisältyvistä tiedoista. Käsiteanalyysi on hyvä aloittaa karkealla tasolla, jota tarkennetaan kehitystyön edetessä. (Hovi 2003, s. 32 - 33.)

Käsiteanalyysin tuloksena syntyy käsitemalli, joka esitetään graafisesti ER-kaaviona (Kuva 13). Ensimmäiset versiot ER-kaaviosta ovat karkeita ja pelkistettyjä, lopulta päädytään normalisoituun ER-kaavioon, joka esitellään luvussa 7.2.



Kuva 13. Karkea ER-kaavio

Alustavassa käsiteanalyysissä tärkeimmiksi käsitteiksi tietokannassa ilmeni koira, käyttäjä, ulkomuoto ja terveydentila. Näiden lisäksi päädyttiin ottamaan mukaan käyttäjärooli, koiran sukupuoli ja kasvattaja. Koira on tässä tapauksessa vahvin käsite yhdessä käyttäjän kanssa, muodostaen koiran ja sen omistajan. Muut käsitteet riippuvat näistä kahdesta. Opinnäytetyön lopputuloksena syntyvän tietokannan tarkoituksenahan on kerätä tietoja koirien terveydentilasta, jotka koiranomistaja (käyttäjä) syöttää järjestelmään.

Käsitteisiin liittyy ominaisuuksia eli attribuutteja. Attribuutit kuvaavat käsitettä, ja voivat koostua yhdestä tai monesta osasta. Esimerkiksi käsitteen ”käyttäjä” attribuutteja ovat muun muassa nimi, osoite ja puhelinnumero. Käsitteelle määritellään pakollinen yksilöivä perusavain, jonka perusteella se erotetaan tietokannassa muista käyttäjistä. (Hovi 2003, s. 35; Lahtonen 2002, s. 19.)

Yksilöivän perusavaimen avulla käsitteestä voidaan luoda useita samanlaisia esiintymiä. Näihin esiintymiin tallennetaan varsinaiset arvot, esimerkiksi nimi tai osoite. Kuviossa 1 on esimerkki käsitteestä käyttäjä ja sen ensimmäiset esiintymät tietokannassa.

kayttajaid	nimi	osoite	puh	tunnus	salasana
1	Maija	Metsätie 24	0501234	maija1	maija2
2	Teemu	Kauppakatu 2	023333	temu	qwrty123
…	…	…	…	…	…

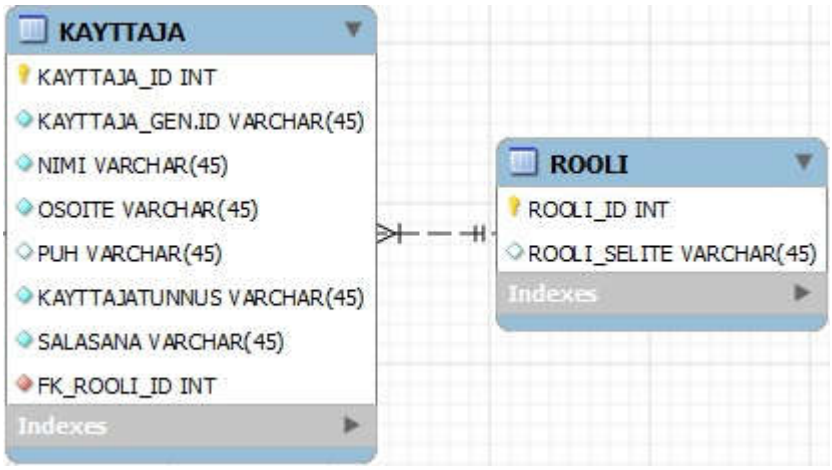
Kuvio 1. Käsite käyttäjä ja sen esiintymiä tietokannassa

Loogisesti toisiinsa liittyvät käsitteiden välillä on yhteyksiä, jotka ovat tärkeitä tiedon eheyden kannalta (Hernandez 2002). Yhteystyyppejä on yksi-moneen, yksi-yhteen tai moni-moneen.

Yksi-yhteen-yhteydet ovat harvinaisia, eikä niitä tässä opinnäytetyössä ollut lopulta jäljellä ainuttakaan. Tämä yhteystyyppi voidaan tulkita niin, että tietyllä käsitteellä voi olla vain yksi mahdollinen yhteys toiseen käsitteeseen, ja toisinpäin. Esimerkiksi koiralla voisi olla vain yksi omistaja ja omistajalla vain yksi koira.

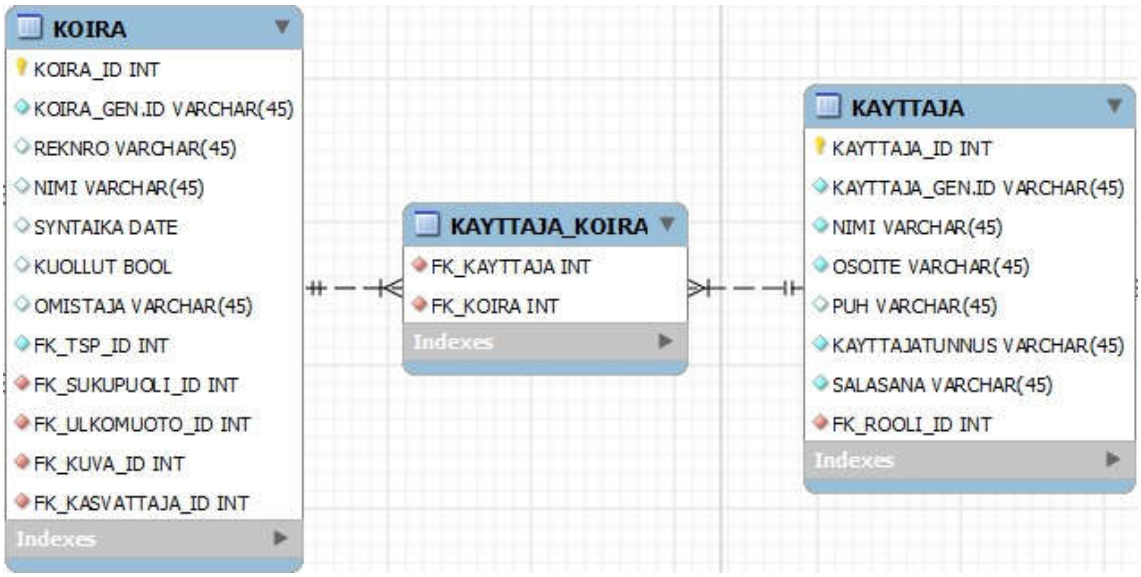
Yksi-moneen-yhteydessä yhdellä käsitteen rivillä voi olla yhteys moneen toisen käsitteen riviin. Toisen käsitteen yksi rivi voi liittyä vain yhteen ensimmäisen käsitteen riviin (Hernandez 2002, s. 48). Tämä yhteystyyppi on yleisin ja käytännössä kaikki yhteystyypit tässä opinnäytetyössä ovat tällaisia.

Kuviossa 2 on esitelty yksi-moneen-yhteys käyttäjän ja roolin välillä. Yhdellä käyttäjällä voi olla vain yksi rooli, mutta yksi rooli voi liittyä moneen käyttäjään.



Kuvio 2. Yksi-moneen-yhteys

Moni-moneen-yhteys syntyy, jos useampi käsitteen rivi voi liittyä useampaan toisen käsitteen riviin, ja päinvastoin. Tällaiset yhteydet ovat yleisiä suunnitteluvaiheen alkuvaiheessa, mutta ne on syytä purkaa. Purkaminen tapahtuu muodostamalla assosiatiivinen apukäsite (Kuvio 3), joka toimii moni-moneen-yhteyden välissä. Ensimmäinen ja toinen käsite muodostavat omat yksi-moneen-yhteytensä apukäsitteeseen. (Hovi 2003, s. 44.)



Kuvio 3. Moni-moneen-yhteyden purkaminen

4.2 Normalisointi

Kun käsitteet on selvitetty, tulee tietokanta normalisoida. Tällä vältetään tietojen toistamista ja päällekkäisyyttä. Näin tietoa tallennetaan vain yhteen paikkaan, jolloin tietokanta on tehokkaampi, joustavampi ja yhdenmukaisempi. (Hovi 2003, s 86.)

Normalisointia voidaan soveltaa jo käsiteanalyysin aikana, vaikka yleensä sillä tarkoitetaan taulujen normalisointia. Jos käsiteanalyysi on suoritettu huolellisesti, normalisointiprosessi onnistuu kivuttomasti tarkastamalla taulut lopussa. (Hovi 2003, s. 97.)

Opinnäytetyössäni käsitteet normalisoitiin osittain jo käsiteanalyysivaiheessa, jonka ansiosta tietokannasta saatiin pienellä vaivalla lopuksi normaalimuotoinen. Seuraavaksi esitellään yleisimmät normalisointimuodot.

Ensimmäinen normaalimuoto

Jokaisen taulun sarakkeen arvot ovat atomisia. Attribuuteilla on vain yksi arvo ja toistuvat ja moniarvoiset ryhmät on poistettu. Tämä onnistuu jakamalla sarakkeet useampaan osaan. (Hovi 2003, s. 89.)

Toinen normaalimuoto

Kun kaikki osittaiset funktionaaliset riippuvuudet on poistettu ja ensimmäisen normaalimuodon ehdot täyttyvät, relaatio on toisessa normaalimuodossa. Vain yhteen perusavaimen määrittämään asiaan liittyvää tietoa tallennetaan tauluun. (Lahtonen 2002, s. 33.)

Kolmas normaalimuoto

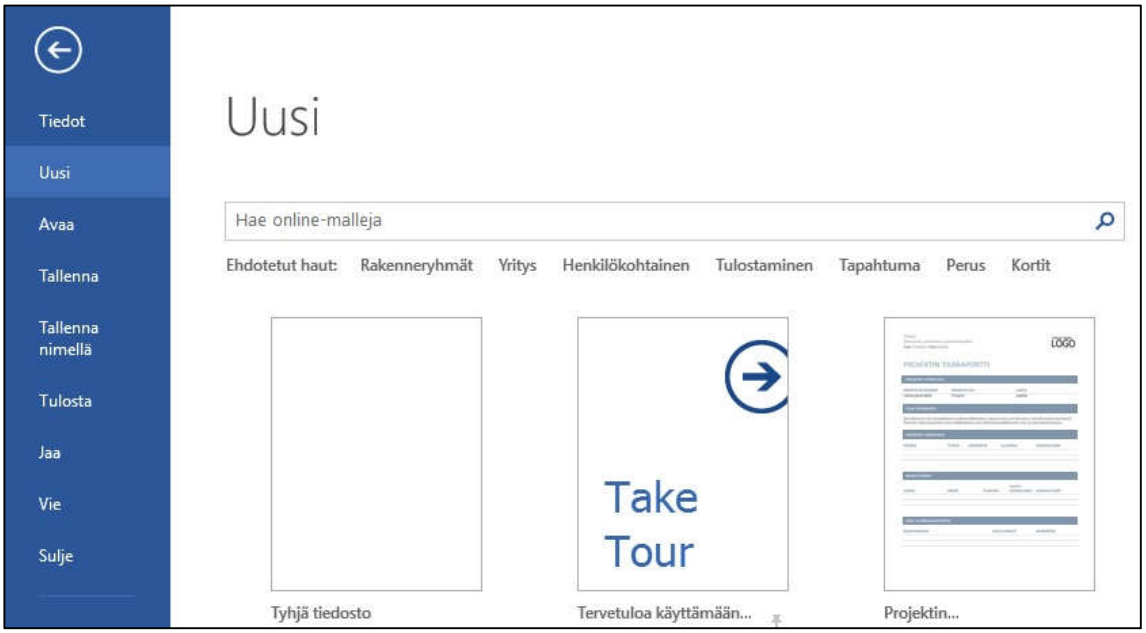
Kun ensimmäisen ja toisen normaalimuodon ehdot täyttyvät ja jokainen sarake on funktionaalisesti riippuvainen vain perusavaimesta, se on kolmannessa normaalimuodossa. Mikään sarake ei siten ole transitiivisesti riippuva muista avainehdokkaista. (Lahtonen 2002, s. 34.)

5 Käytetyt työvälineet

Tietokannan määrittelyyn, suunnitteluun ja toteutukseen käytettiin seuraavia työvälineitä: dokumentointi ohjelmalla Office 2013, tietokannan määrittely ohjelmalla StarUML, tietokannan suunnittelu ohjelmalla MySQL Workbench, tietokannan hallinta ohjelmalla phpMyAdmin ja koodin muokkaus ohjelmalla Notepad++.

5.1 Office 2013

Microsoft Office 2013 tuoteperheeseen kuuluu useita hyödyllisiä sovelluksia, joita käytetään laajasti dokumenttien, kaavioiden ja esitysten luomiseen ja käsittelyyn. Nämä ovat Word, Excel ja PowerPoint. Ohjelmat ovat maksullisia, mutta lähes pakollisia Windows-käytössä. Niille on olemassa ilmaisia vaihtoehtoja, jotka valitettavasti eivät ole aina yhteensopivia keskenään. Tämän vuoksi dokumentointi hoidettiin pääasiassa Microsoftin Word 2013 -ohjelmalla (Kuva 14), jolloin taattiin yhteensopivuus. Excelillä tehtiin kaaviot (Kuva 15) ja PowerPointilla eräät kuvat ja esitykset.



Kuva 14. Microsoft Word 2013 aloitusvalikko

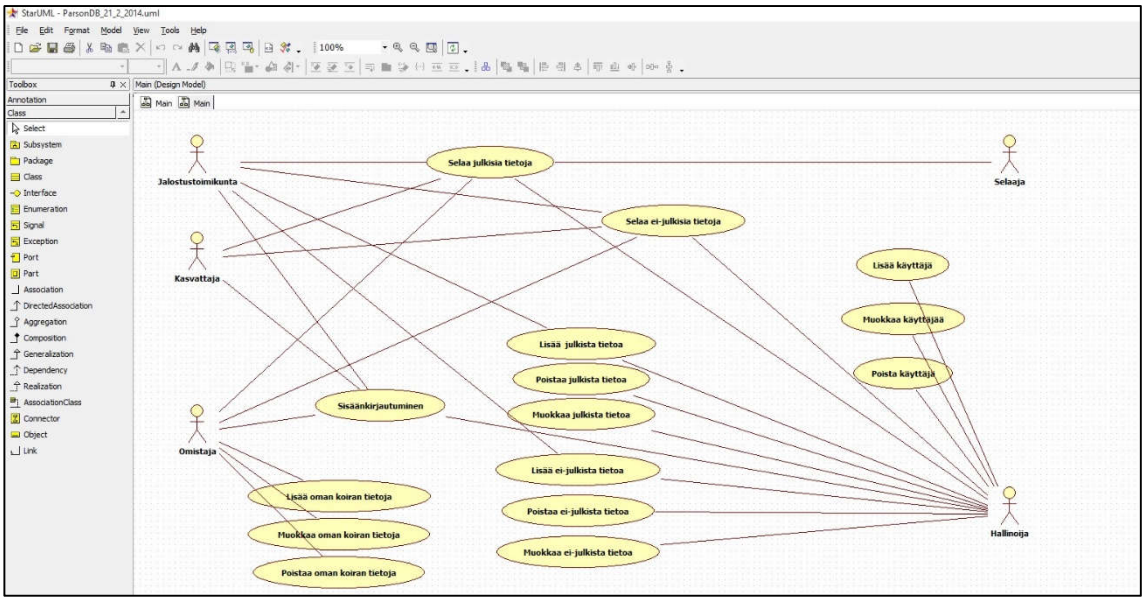
Projektin aikajana	viikko 7	viikko 8	viikko 9	viikko 10	viikko 11	viikko 12	viikko 13	viikko 14	viikko 15	viikko 16	viikko 17	viikko 18	Tuntimäärät
Projektisuunnitelma													20 h
Tietokannan määrittely													80 h
Tietokannan sunnittelu													60 h
Palvelimen pystytys													5 h
Tietokannan toteutus													60 h
Käyttöliittymän suunnittelu													80 h
Käyttöliittymän toteutus													80 h
Käyttöönotto													20 h
Testaus													20 h
Raportointi													10 h
Yht.													435 h

Tekemättä
Tekeillä
Tehty

Kuva 15. Microsoft Excel 2013, esimerkkinä Gantt-kaavio

5.2 StarUML

StarUML on UML-työkalu, jonka on kehittänyt MKLab. Se on suunniteltu korvaamaan kaupalliset ohjelmistot, kuten Rational Rose ja Borland Together. StarUML:llä voidaan luoda UML 2.0-standardin mukaisia kaavioita, esimerkiksi käyttötapauskaavio (Kuva 16).

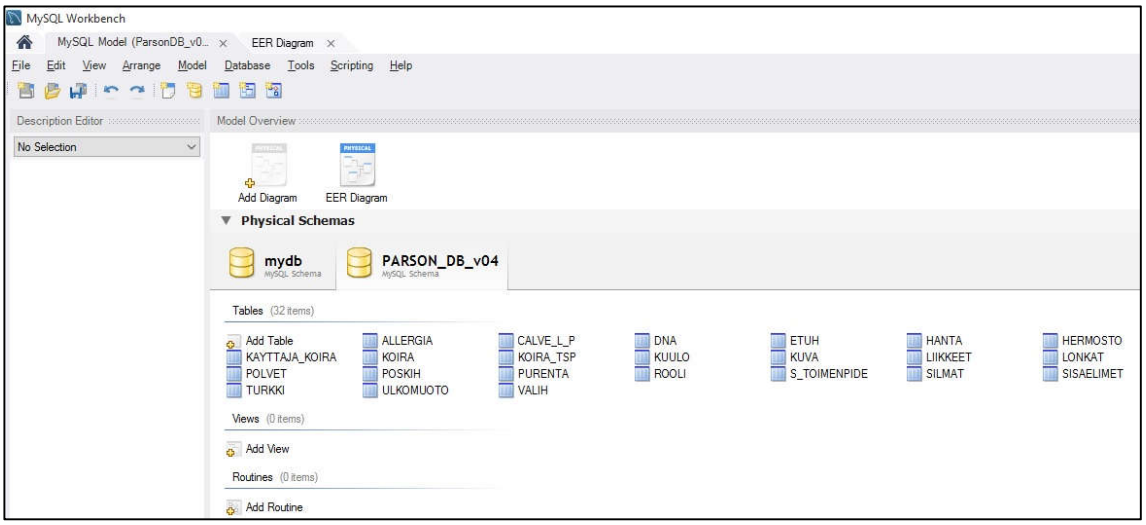


Kuva 16. StarUML:llä luotu käyttötapauskaavio

Ohjelma on erittäin laaja ja jo jonkin verran vanhentunut, mutta sillä voi yhä tehdä tarvittavat kaaviot tietokannan vaatimusmäärittelyyn. Ohjelmasta on saatavilla ilmainen vapaaseen lähdekoodiin perustuva versio, mutta sen kohtalo jatkossa on toistaiseksi auki. Viimeisin vakaa versio on julkaistu heinäkuussa 2015.

5.3 MySQL Workbench

MySQL Workbench on erittäin monipuolinen ilmainen työkalu tietokantojen suunnitteluun ja toteutukseen. Samalla työvälineellä relaatiotietokanta voidaan suunnitella määrittelyn perusteella ja luoda toimiva MySQL-pohjainen tietokanta (Kuva 17). Tämä on suoraan käytettävissä esimerkiksi XAMPP-virtuaalipalvelimella tai webhotellissa.



Kuva 17. MySQL Workbenchillä luotu tietokanta

Ohjelma tarjoaa työkalut tietokannan visuaaliseen suunniteluun, mallinnukseen, muokkaukseen ja toteutukseen. Ennen tietokannan muokkausta tai luomista pitää se perustaa esimerkiksi XAMPPilla paikallisesti tietokoneelle. MySQL Workbench:llä muodostetaan yhteys palvelimeen ja annetaan root-tunnukset. Tämän jälkeen sitä voidaan käsitellä huomattavasti helpommin, kuin vaihtoehtoisilla työkaluilla kuten phpMyAdminilla.

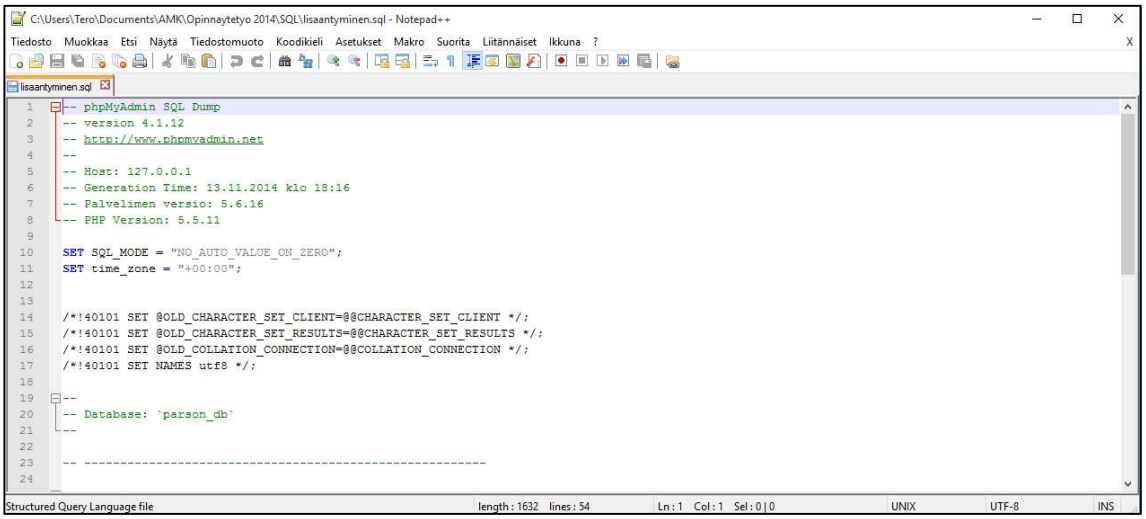
Työkalun forward engineer -toiminnoilla suunnitellusta relaatiotietokannasta saadaan suoraan tehtyä SQL-kielen mukaista, tai päinvastoin luoda jo olemassa tietokannasta visuaalinen mallinnus reverse engineer -toiminnolla. Tämä helpottaa esimerkiksi ER-kaavion luomista.

5.4 PhpMyadmin

PhpMyAdmin on tarkoitettu Linux-pohjaiseen MySQL-tietokannan hallintaan. Se ei ole yhtä tehokas ja helppokäyttöinen kuin eräät muut ohjelmat, mutta tietokannan hallinnan perustoimet onnistuvat sillä parhaiten. Näitä ovat tietokannan perustaminen, siirtäminen järjestelmästä toiseen, käyttöönottovaiheen tehtävät ja tietokannan varmuuskopiointi.

5.5 Notepad++

Notepad++ on Windows-pohjainen teksti- ja koodieditori. Se on hyvin kevyt ja pelkistetty mutta soveltuu esimerkiksi yksinkertaisten SQL-kyselyiden laatimiseen ja tarkasteluun (Kuva 18). Ohjelma kykenee tunnistamaan useita erilaisia ohjelmointikieliä, jonka vuoksi se on hyvä apuväline myös tietokannan toteutusvaiheessa.



Kuva 18. Notepad++

6 Opinnäytetyöprojekti

Tietokannan määrittely alkoi alkuvuodesta 2014 perustuen laadittuun projektisuunnitelmaan ja asiakkaan antamiin vaatimuksiin. Toimin aluksi yksin projektin parissa, helmikuussa 2014 mukaan tuli käyttöliittymän ohjelmoinnista vastannut Tuomas Porvali. Opinnäytetyön ohjaajana toimi Martti Ylä-Jussila ja asiakkaan edustajana Heli Korpinen.

6.1 Projektin suunnittelu

Projektisuunnitelma laadittiin projektin alkuvaiheessa helmikuussa 2014. Projektisuunnitelmassa otettiin kantaa projektin etenemiseen, aikatauluun ja tavoitteisiin. Projektisuunnitelma tietokannan toteuttamisesta laadittiin yhdessä asiakkaan edustajan kanssa ja sitä laajennettiin myöhemmin koskemaan myös järjestelmän käyttöliittymää Tuomas Porvalin tullessa mukaan projektiin.

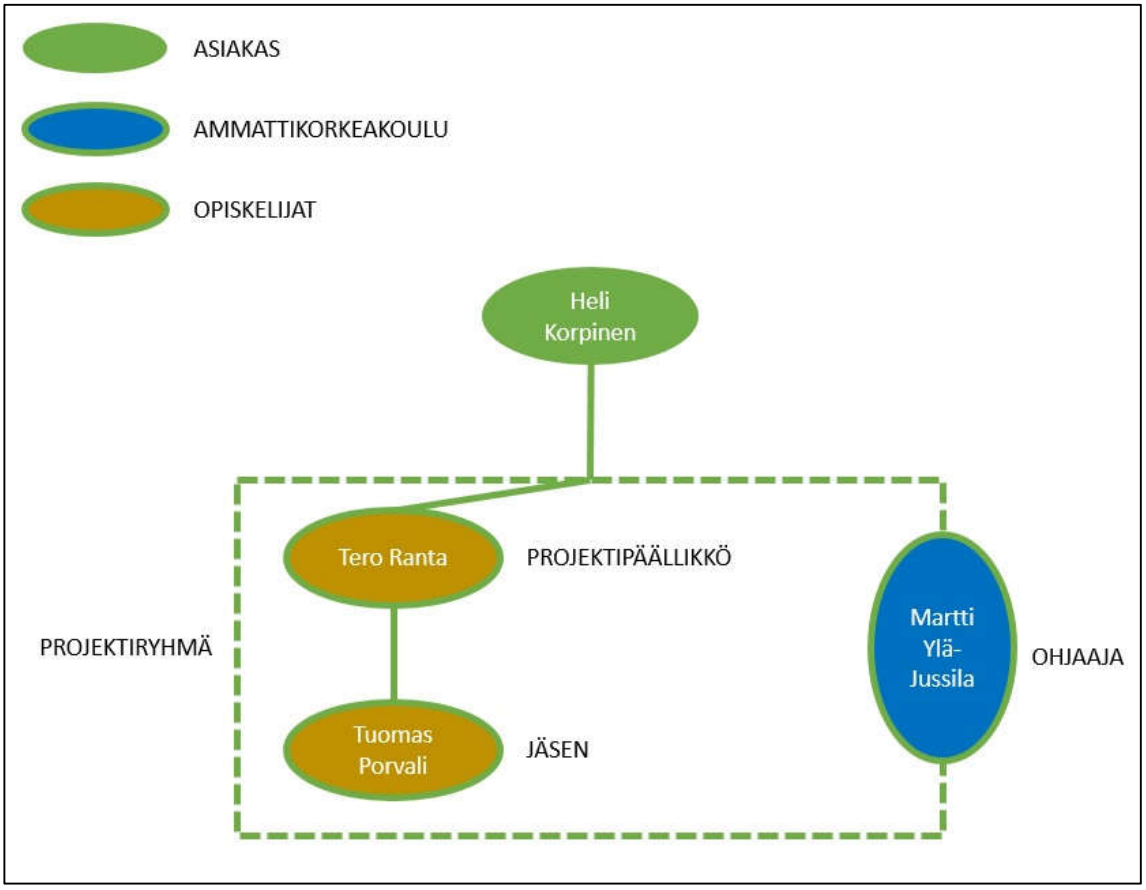
Projektiorganisaatio koostui aluksi asiakkaan edustajasta Heli Korpisesta, projektin ohjaajasta Martti Ylä-Jussilasta ja minusta. Toimin alkuosan projektia yksin projektipäällikkönä ja tietokannan toteuttajana. Helmikuussa mukaan tullut Tuomas Porvali toimi käyttöliittymän ohjelmoijana ja myöhemmin myös projektipäällikkönä.

Työnjako jakautui seuraavasti:

- projektin suunnittelu/projektipäällikkö: Tero Ranta, myöhemmin Tuomas Porvali
- määrittely: Tero Ranta ja Tuomas Porvali
- tietokannan suunnittelu: Tero Ranta
- käyttöliittymän suunnittelu ja ohjelmointi: Tuomas Porvali
- testaus: Tero Ranta, Tuomas Porvali ja Heli Korpinen
- käyttöönotto Tuomas Porvali ja Tero Ranta

Projektia toteutettiin käyttäen sovellettua ketterää menetelmää nimeltä Scrum. Viikoittaiset palaverit ja iteraatiot mahdollistivat nopeat muutokset tarvittaessa joko tietokantaan tai käyttöliittymään. Jokaisen viikottaisen iteraation jälkeen voitiin kysyä asiakkaan edustajan mielipidettä ja laatia lista uusista tehtävistä.

Seurantakokouksia pidettiin lähes joka viikko, jolloin kirjattiin ylös projektin edistyminen, mitä on tekemättä, muutokset, riskit ja aikataulun paikkansapitävyys. Seurantakokoukset kestivät noin 15 - 30 minuuttia. Kokouksesta laadittiin muistio, joka jaettiin osallistujien kesken ja toimitettiin myös projektin ohjaajalle ja asiakkaan edustajalle tiedoksi. Seurantakokouksiin osallistuivat asiakkaan edustaja Heli Korpinen, ohjaaja Martti Ylä-Jussila ja opiskelijajäsenet Tuomas Porvali ja Tero Ranta (Kuva 19).



Kuva 19. Organisaatiokaavio

6.2 Projektisuunnitelma

Laadin projektisuunnitelman heti projektin alussa yhdessä asiakkaan kanssa. Asiakas oli aiemmin toimittanut vaatimukset tietokannan ja järjestelmän toiminnalle, joiden pohjalta laadittiin projektisuunnitelma. Suunnitelma piti sisällään projektin tavoitteet, organisaation, vastuut, riskit, aikataulun sekä projektin eri vaiheet ja tehtävät ja niiden arvioidut työmäärät.

Projektisuunnitelman alkuvaiheessa ei otettu vielä kantaa käyttöliittymään ja toteutukseen vaan ainoastaan tietokannan määrittelyyn, suunniteluun ja toteutukseen. Tuomas Porvalin tullessa mukaan projektin jäseneksi lisättiin projektisuunnitelmaan hänelle määritetyt tehtävät, jotka koskivat verkkopohjaisen käyttöliittymän määrittelyä, suunnittelua, toteutusta, käyttöönottoa ja testausta.

6.3 Vaatimusmäärittely

Vaatimusmäärittelyä ennen oli asiakkaan kanssa yhteistyössä päästy yhteisymmärrykseen tietokannan rakenteesta ja sisällöstä. Tein aluksi tietokannasta luonnoksen, jota voitiin vertailla asiakkaan tarpeisiin ja näin muodostettiin pohja vaatimusmäärittelylle. Itse vaatimusmäärittelyä aloitettiin kirjoittamaan vasta myöhemmässä vaiheessa projektin kulkua, mutta sen sisältö oli melko selvä ja dokumentointi muutoin kunnossa. Itse dokumentti valmistui marraskuussa 2014.

Vaatimusmäärittely dokumentoitiin IEEE830-standardin mukaisesti valmiille pohjalle. Vaatimusmäärittelyssä kuvailtiin tietokannan ja järjestelmän pääpiirteet ja tärkeimmät toiminnot. Siinä määriteltiin tietokannan arkkitehtuuria, fyysistä rakennetta, käyttötapauksia ja mahdollisia rajoituksia. Dokumentissa otettiin myös kantaa asiakkaan tarpeisiin, talletettaviin tietoihin ja niiden muotoon sekä käyttöympäristöön, käyttäjiin ja toiminnallisiin vaatimuksiin. Dokumentti toimi myös pohjana testaussuunnitelman laatimiselle.

6.4 Tietokannan suunnittelu

Tietokanta suunniteltiin alustavan vaatimusmäärittelyn ja teknisten tietojen perustalta. Myös tietokannan luonnosta, joka oli hyväksytetty asiakkaalla, käytettiin varsinaisen tietokannan suunniteluun ja toteutukseen.

Tietokannan suunnittelu eteni vaiheittain ja eli koko projektin ajan. Vaikka alun perin sovittu järjestelmä pysyi pääpiirteittäin muuttumattomana, tuli asiakkaan pyynnöstä tarve muuttaa useita yksityiskohtia. Tämä osaltaan hidasti tietokannan toteutusvaihetta, mutta muutokset olivat melko yksinkertaisia toteuttaa.

Aluksi suunniteltiin tietokannan sisältämät taulut ja niiden väliset yhteydet. Kun nämä oli päätetty, voitiin suunnitella kunkin taulun sisältämät kentät ja niihin talletettavien tietojen muoto, laatu ja koko. Jo alkuvaiheessa kävi ilmeiseksi, että

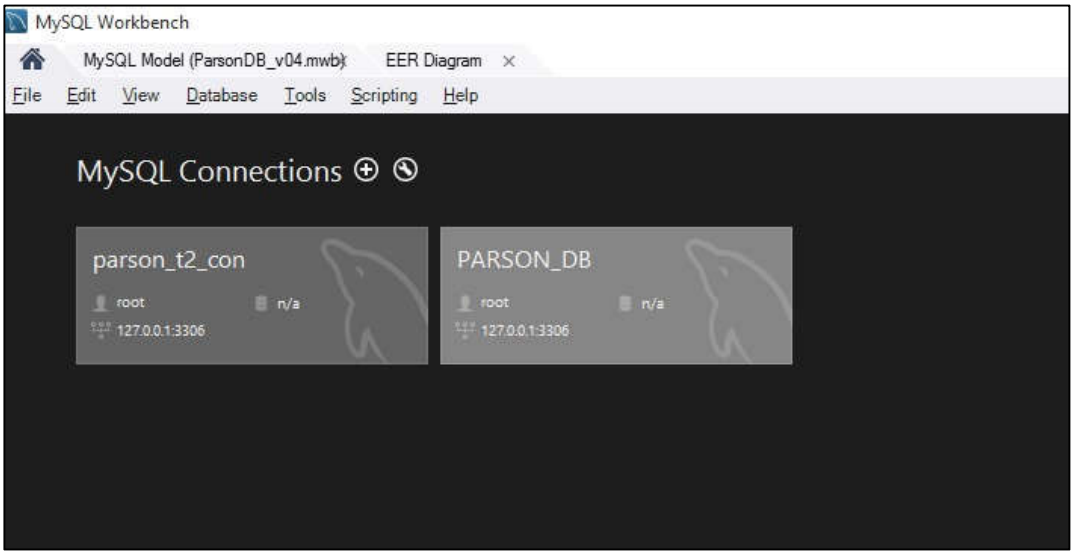
tietokantaan tulisi lukuisia tauluja ja tarvittavia aputauluja purkamaan moni-moneen relaatioita. Tämän vuoksi tietokanta jaettiin pienempiin kokonaisuuksiin, kuten käyttäjien tiedot, koirien tiedot ja tiedot eri sairauksista ja terveyteen liittyvistä asioista. Näin voitiin suunnitella yksi osa-alue kerralla, jotka lopuksi yhdistettiin yhdeksi kokonaisuudeksi eli lopulliseksi tietokannaksi.

Tietokannan suunnitteluun käytettiin UML-kieltä, joka on alalla standardi tietokantojen kuvaamiseen käytetty kieli. Tällä luotiin käyttötapausmallit ja tietokantakaaviot, joita voitiin myöhemmin käyttää itse tietokannan toteutukseen. UML-kielellä kuvattuja suunnitelmia voitiin myös jakaa muun työryhmän kesken sekä esitellä asiakkaalle projektipalavereissa.

Tietokantasuunnitelmassa päätettiin myös käytettävistä työvälineistä ja ohjelmointikielistä sekä käyttöönottovaiheessa käytettävästä laitteisto- ja ohjelmistoalustasta. Järjestelmän verkkokäyttöliittymä toteutettiin näiden pohjalta myöhemmässä vaiheessa, joten niiden dokumentointi oli tärkeää.

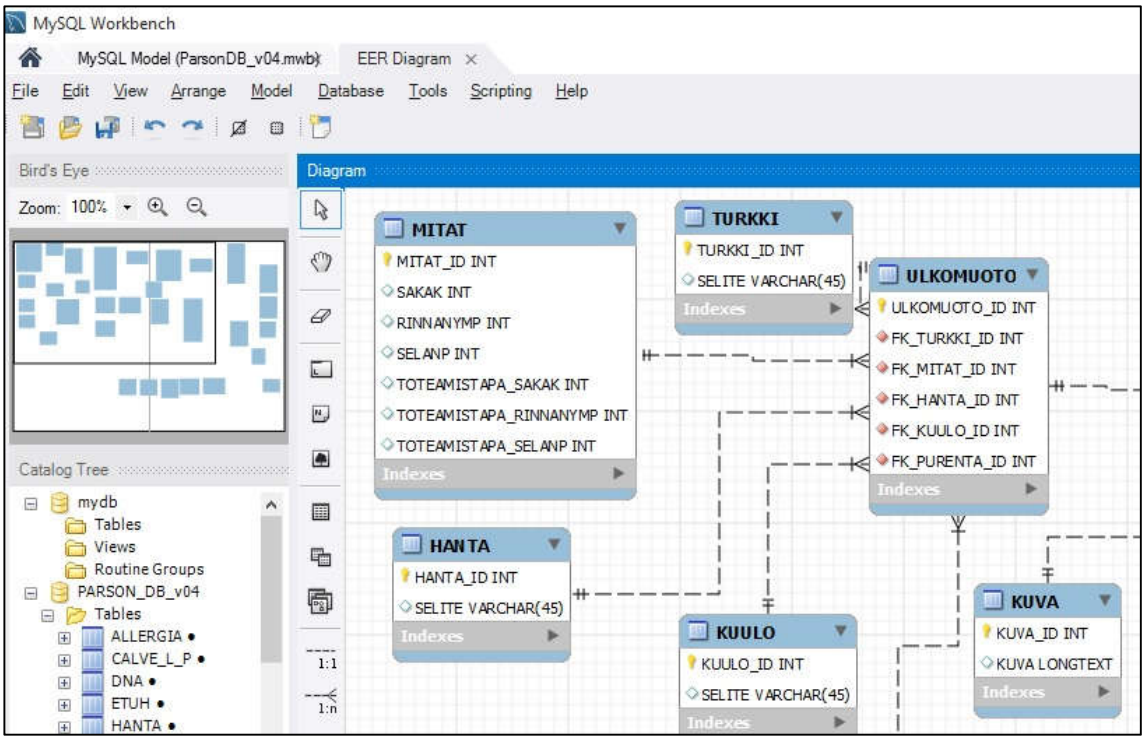
6.5 Tietokannan toteutus

Tietokanta toteutettiin suunnitelman mukaisesti, vaikka pieniä muutoksia siihen tulikin projektin edetessä. Toteutusta helpotti tehokkaiden ja helppokäyttöisten työkalujen käyttö, pääasiassa näitä olivat MySQL Workbench (Kuva 20) ja phpMyAdmin.



Kuva 20. MySQL Workbench

Toteutus aloitettiin MySQL Workbenchillä luomalla tietokanta ja ottamalla yhteys siihen. Tietokantaa säilytettiin tässä vaiheessa paikallisesti tietokoneella, jolloin sitä oli helppo muokata ja siirtää muille alustoille. Tietokanta vaati toimiakseen virtuaalipalvelimen, joka pystytettiin käyttäen XAMPPia. Tietokannasta luotiin ER-kaavio (Kuva 21).



Kuva 21. Otos terveystietokannan ER-kaaviosta

ER-kaavion avulla voitiin laatia tietokannan yksityiskohdat ja taulujen väliset relaatiot. Yllä olevassa kuvassa on osa tietokannasta, joka käsittelee koirien fyysisiä ominaisuuksia, kuten kokoa ja turkkia.

Kuten tietokannan suunnittelukin, toteutus eteni vaiheittain osa kerrallaan. Kun taulujen sisältämät kentät ja yhteydet oli toteutettu MySQL Workbenchillä, voitiin paneutua yksittäisten tietueiden tyyppiin, laatuun ja kokoon. Tässä vaiheessa piti jo tietää tarkasti, minkälaista tietoa kyseiseen tietueeseen talletetaan, eli onko tieto esimerkiksi numeerista vai tekstiä.

Kun ER-kaavio on valmistunut, voidaan MySQL Workbenchilla luoda tietokanta. Tietokanta on MySQL-muodossa, joten se on suoraan yhteensopiva muiden työkalujen kuten phpMyAdminin kanssa. Luodussa relaatiotietokannassa ovat kaikki

samat yhteydet, taulut ja kentät kuin ER-mallissa, joten sitä voidaan samantien testata ja huomata mahdolliset virheet. Nämä voidaan korjata ER-malliin ja luoda uusi versio tietokannasta. Nämä toteutuksen ja testauksen nopeat iteraatiot helpottavat lopullisen tietokannan valmistumista.

6.6 Tietokannan testaus

Tietokantaa voidaan testata phpMyAdmin-ohjelmalla, joka on valmiiksi sisäänrakennettu XAMPPin virtuaalipalvelimeen. Kenttiin voidaan syöttää tietoa ja testata erilaisia raportteja, jolloin nähdään, onko tieto talletettu kantaan oikeassa muodossa ja toimivatko relaatiot tarkoitetulla tavalla. Jos havaitaan, että esimerkiksi koiran MITAT-taulussa kenttään SAKAK (säkäkorkeus) on syötetty vahingossa tekstiä, vaikka pitäisi olla kokonaisluku senttimetreissä, voidaan päätellä vian paikka ja pakottaa käyttäjä syöttämään numeerista tietoa tai tehdä muutoksia tietokantaan.

Testauksen aikana tarkistetaan myös taulujen väliset relaatiot ja kenttäavaimet. Koska monta-moneen-yhteydet on purettu aputaulujen avulla, on tärkeää, että jokaiselle pääavaimelle löytyy viiteavainpari aputaulusta. Tälle pitää taas löytyä pari toisesta taulusta, jolloin muodostuu yksiselitteinen relaatio.

Projektin loppuvaiheessa tietokantaa testattiin myös Tuomas Porvalin tekemällä verkkokäyttöliittymällä, joka vastaa tilannetta loppukäyttäjän kannalta. Käyttöliittymä ei saa sallia väärän muotoisen tiedon syöttämistä, jotta tietokanta toimii oletetusti. Toiminnallisesta määrittelydokumentista voidaan tarkistaa, onko vika tällöin tietokannassa vai käyttöliittymässä, ja tehdä tarvittavat korjaukset.

Tietokantaa ja järjestelmää testattiin myös Saimaan ammattikorkeakoulun opiskelijoiden toimesta. Testauskurssin opiskelijat testasivat järjestelmän toimintaa kuormituksen alla sekä etsivät virheitä ja raportoivat niistä työryhmälle. Testaajat testasivat järjestelmää eri selaimilla pareittain Tuomas Porvalin tekemien testaustapausten perustella. Varsinaista testaussuunnitelmaa ei ollut kuin aivan lopussa, mutta testaajat löysivät silti virheitä ja suurin osa saatiin korjattua.

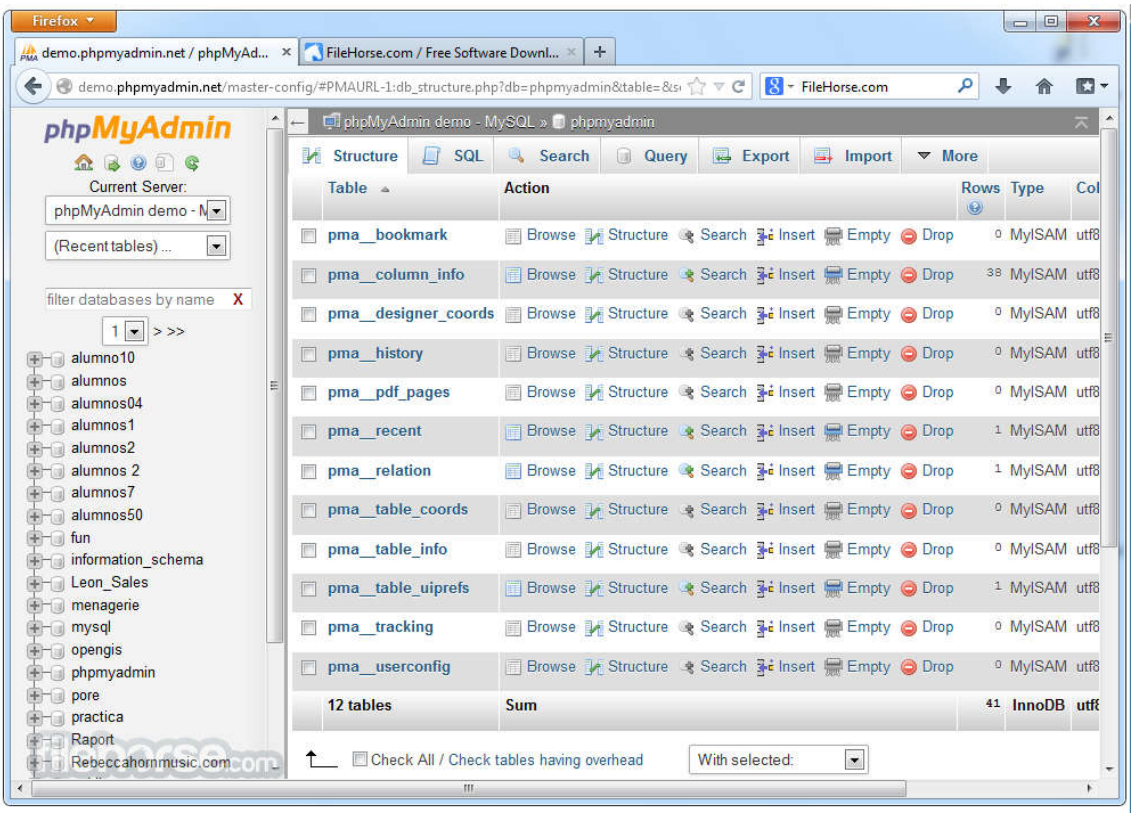
6.7 Webhotelli

Järjestelmä sijoitettiin asiakkaan toivomuksesta webhotelliin, jonka toimitti OVH-hosting (Kuva 22). Kyseinen palveluntarjoaja tarjoaa useita eritasoisia palveluita, jotka on hinnoiteltu niiden ominaisuuksien ja laadun mukaan. Tälle järjestelmälle riitti alhaisen tason webhotellitili, joka tarjosi kuitenkin tarvittavat ominaisuudet ja työkalut sen käyttämiseen. Tiliä voi jatkossa laajentaa maksua vastaan, jolloin esimerkiksi tallennustila ja muut ominaisuudet saadaan vastamaan tarpeita.

Kuva 22. OVH-hosting webhotellin kotisivu

Asiakkaalle luotiin asiakastili ja verkkotunnus. Verkkotunnus on WWW-osoite, jonka käyttäjä syöttää verkkoselaimen osoitekenttään. Verkkotunnukseksi valittiin parsonterveys.fi, joka rekisteröitiin Parsonrussellinterrierit ry:lle.

Tietokannan siirrossa webhotelliin oli aluksi vaikeuksia osin käyttäjien kokemattomuuden vuoksi. Kun oikea tapa löytyi, onnistui siirto suuremmitta ongelmitta. Tietokantaa hallitaan webhotellissa phpMyAdmin-ohjelmalla (Kuva 23), joka oli tuttu jo suunnittelu ja toteutusvaiheista.



Kuva 23. phpMyAdmin-ohjelmalla hallitaan tietokantaa webhotellissa

6.8 Käyttöönotto

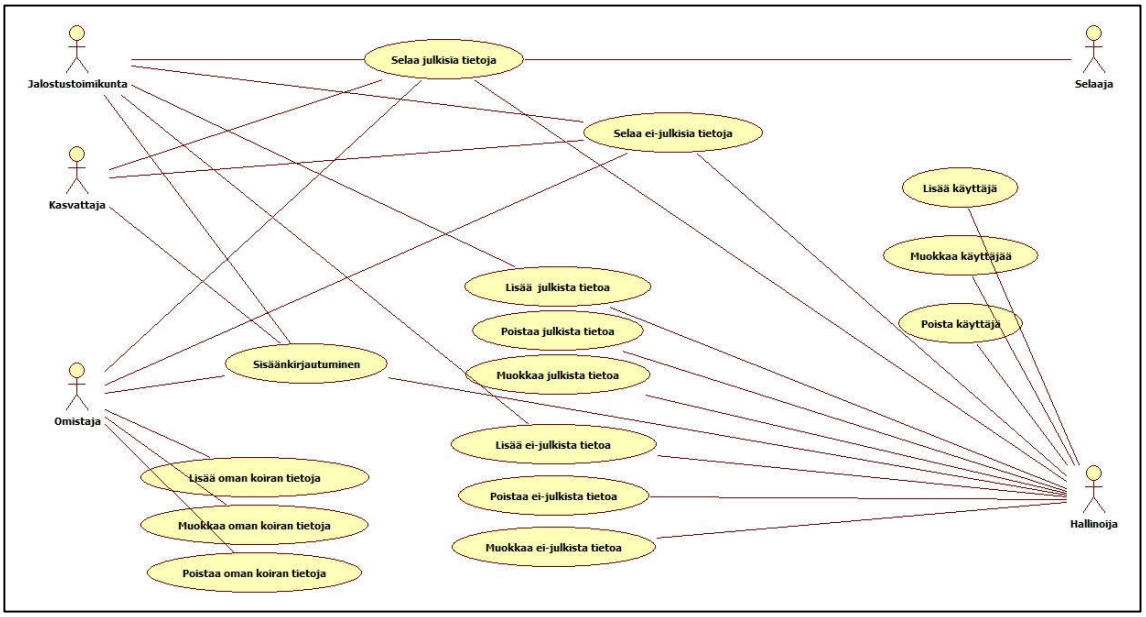
Tietokanta otettiin käyttöön samanaikaisesti käyttöliittymän kanssa keväällä 2015. Asiakas hyväksyi järjestelmän ja aloitti sen ylläpitämisen omalla palvelimellaan. Asiakkaalle kirjoitettiin ja annettiin myös järjestelmän käyttöohje, joka palvelee sekä ylläpitäjiä että loppukäyttäjiä.

7 Lopputuloksen kuvaus

Opinnäytetyöprojektissa luotiin verkkopohjainen tietokanta ja käyttöliittymä, joilla hallitaan Parsonrussellinterierit ry:n terveystietokantaa. Järjestelmän ylläpidosta ja oikeuksien hallinasta vastaa pääkäyttäjä. Järjestelmää käytetään verkkoselaimella, jolla käyttäjä kirjautuu sivuille ja voi tarkastella, lisätä, muuttaa tai poistaa omistamiensa koirien tietoja tietokannassa. Osaa järjestelmän tiedoista voivat tarkastella myös muut kuin omistajat, esimerkiksi eläinlääkärit tai jalostustoimikunnan edustajat. Heille annetaan asiaankuuluvat oikeudet järjestelmään tehtävänsä hoitamiseen. Järjestelmä sijaitsee OVH-hosting palveluntarjoajan webhotellissa.

7.1 Käyttötapauskaavio

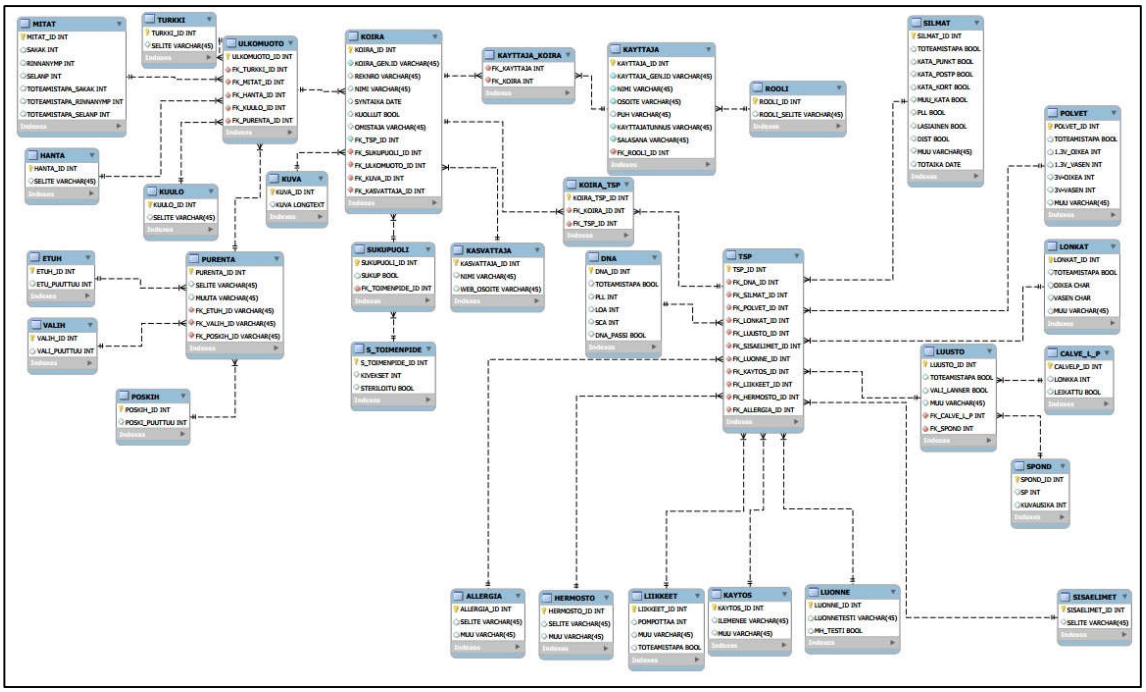
Järjestelmästä tehtiin käyttötapauskaavio (Kuva 24) käyttäen UML-standardin mukaista kaaviota. Kaaviosta selviävät kaikki käyttötapaukset ja järjestelmän käyttäjät. Käyttötapauskaavion avulla suunniteltiin itse tietokanta.



Kuva 24. Käyttötapauskaavio

7.2 Tietokantakaavio

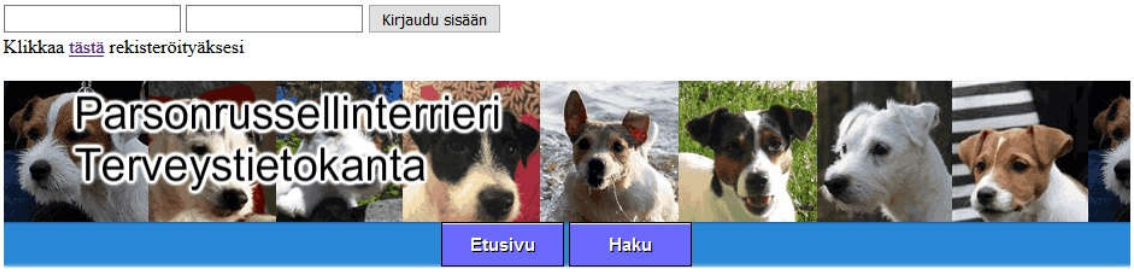
Tietokantakaaviossa kuvataan tietokannan rakenne, relaatiot, taulut ja kentät (Kuva 25).



Kuva 25. Tietokantakaavio

7.3 Etusivu

Käyttäjän tullessa palveluun, näkee hän ensimmäiseksi pääsivun eli etusivun (Kuva 26). Jos hän ei ole rekisteröitynyt käyttäjä, voi hän tehdä uuden käyttäjätilin tai selata koirien julkisia tietoja Hakuvalikon avulla.

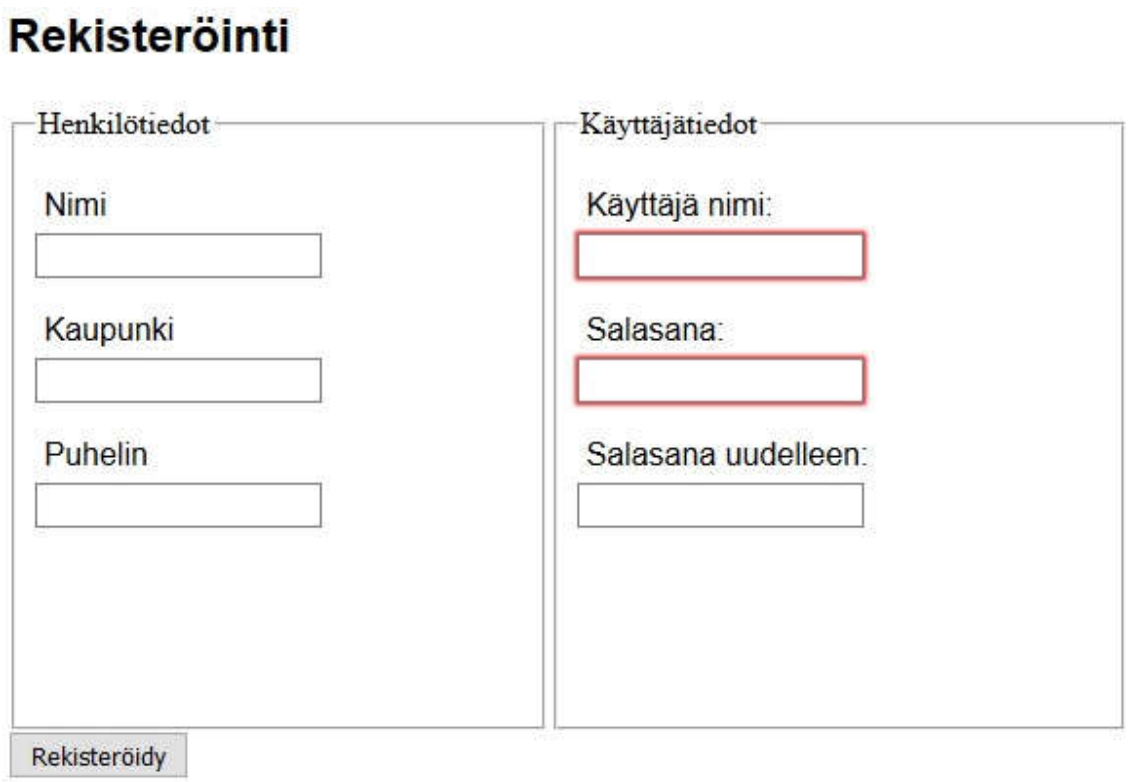


Kuva 26. Etusivu

Jos käyttäjä on jo rekisteröitynyt, hän voi kirjautua sivulle luomallaan tunnuksella ja salasanalla.

7.4 Rekisteröinti

Rekisteröinti tapahtuu etusivulla olevasta linkistä. Tämä vie käyttäjän rekisteröintisivulle, jossa häntä pyydetään syöttämään nimensä, osoitteensa, puhelinnumeronsa ja valitsemaan itselleen käyttäjätunnus ja salasana (Kuva 27).



Kuva 27. Rekisteröinti-sivu

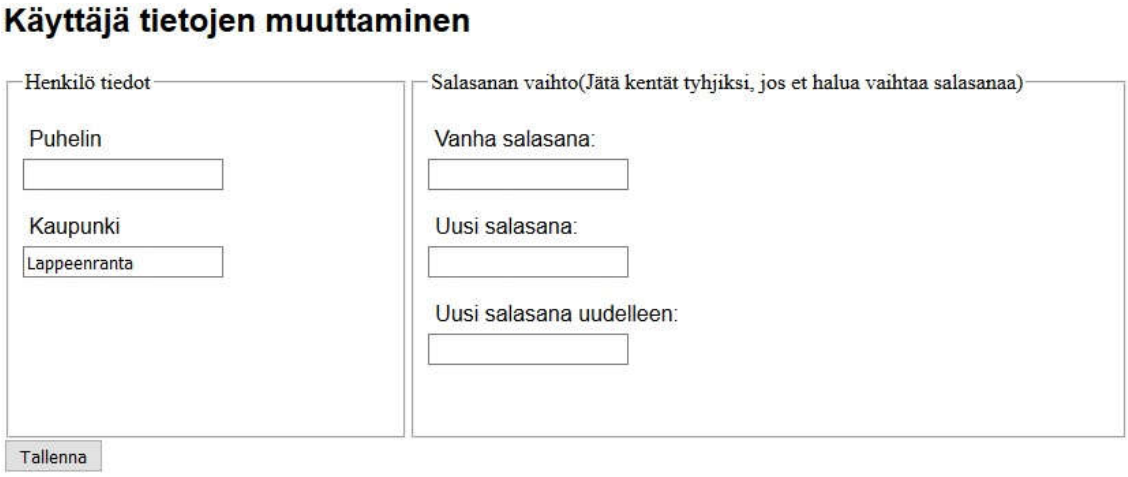
Kun käyttäjä on syöttänyt pyydetyt tiedot ja ne ovat oikeassa muodossa, hän painaa **Rekisteröidy**-painiketta. Tällöin tiedot siirtyvät tietokantaan ja ovat sen jälkeen liitetty kyseiseen käyttäjätunnukseen. Käyttäjä voi myöhemmin kirjautua ulos sivulta ja tiedot pysyvät järjestelmässä.

Uusille rekisteröityneille käyttäjille annetaan automaattisesti koiranomistaja rooli käyttöliittymän toimesta. Näin he voivat myöhemmin lisätä koiran järjestelmään ja muokata sen tietoja. Muut roolit annetaan tarpeen vaatiessa pääkäyttäjän toimesta manuaalisesti.

7.5 Käyttäjän tiedot

Käyttäjä voi katsella tietojaan (Kuva 28) ja muokata osaa niistä (Kuva 29), kun hän on kirjautunut sisään järjestelmään. Käyttäjätunnusta ja nimeä ei voi vaihtaa, muut tiedot ja salasanan voi. Käyttäjätietojen muokkauksen jälkeen hän painaa **Tallenna**-painiketta ja uudet tiedot tallentuvat tietokantaan.

Kuva 28. Käyttäjätietojen katselu



Kuva 29. Käyttäjätietojen muokkaus

7.6 Koiran lisäys

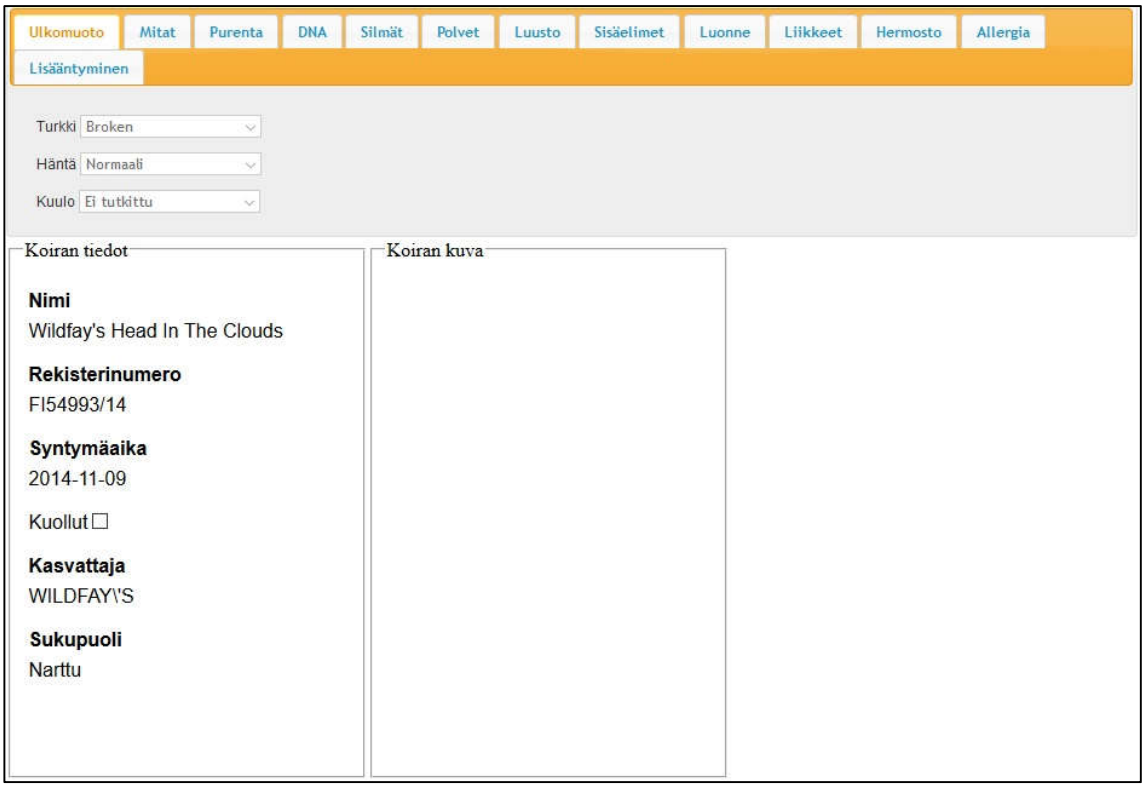
Rekisteröitynyt käyttäjä voi lisätä koiria käyttäjätililleen (Kuva 30). Lisääminen tapahtuu Lisää koira -valikosta, joka avautuu etusivun painikkeesta. Koiraa lisätessä pitää syöttää sen nimi, rekisterinumero, syntymäaika, kasvattaja ja suku-

puoli. Halutessaan omistaja voi myös lisätä pienen kuvan koirastaa, joka helpottaa tunnistamista. Hän voi myös jättää viesti jalostustoimikunnalle samalla sivulla. Kun tiedot on syötetty, käyttäjä painaa Lisää koira -painiketta.

Kuva 30. Lisää koira

7.7 Koiran tietojen muokkaus

Omistaja voi muokata lisäämänsä koiran tietoja sille tarkoitetulla sivulla (Kuva 31). Muokkaus tapahtuu valitsemalla koira joko omalta käyttäjäsivulta tai hakemalla sitä hakusivulla. Koirista on talletettu lukuisia eri tietoja koskien sen fyysistä ulkonäköä ja terveystietoja. Koska järjestelmän tarkoitus on nimenomaan terveystietojen hallinta, ovat niiden muokkausmahdollisuudet erittäin yksityiskohtaisia.

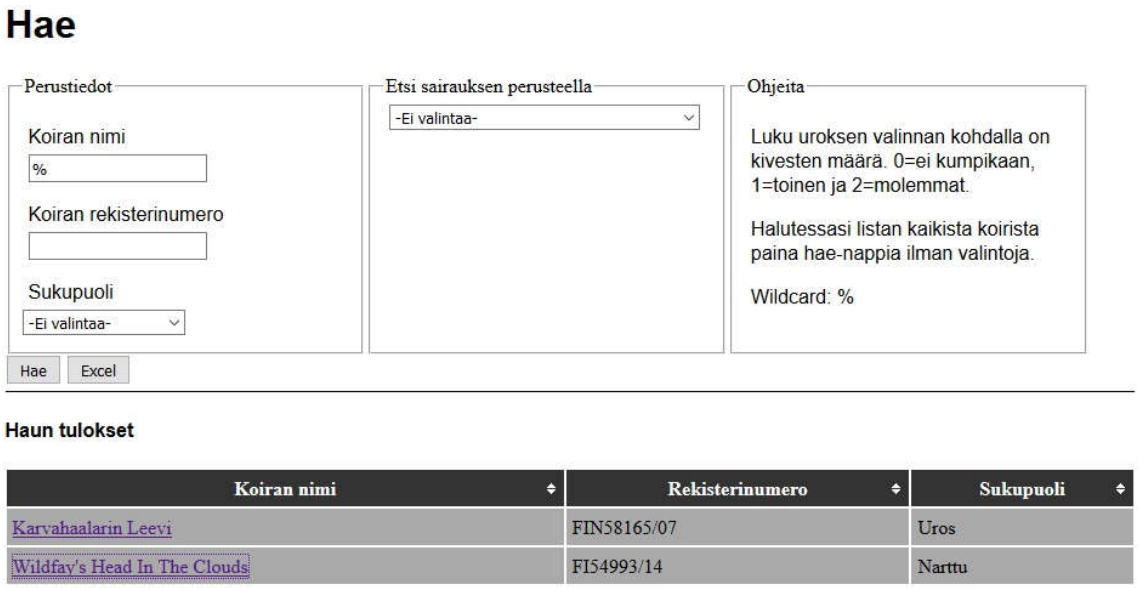


Kuva 31. Koiran tietojen muokkaus

Muokattavia osa-alueita ovat: ulkonäkö, mitat, purenta, DNA, silmät, polvet, luusto, sisäelimet, luonne, liikkeet, hermosto, allergia ja lisääntyminen. Näiden alla on yksityiskohtaisia valintoja ja kenttiä, joissa koiran ulkomuotoa ja terveyttä voidaan selventää. Myös koiran perustietoja voidaan muokata tällä sivulla.

7.8 Koiran haku

Koiria voidaan hakea ilman rekisteröitymistä järjestelmään. Vain jotkut tiedot ovat kuitenkin julkisia ja niiden tarkastelu vaatii omistajan, jalostustoimikunnan tai kasvattajan roolin. Koiran haku tapahtuu etusivun haku-valikosta (Kuva 32).

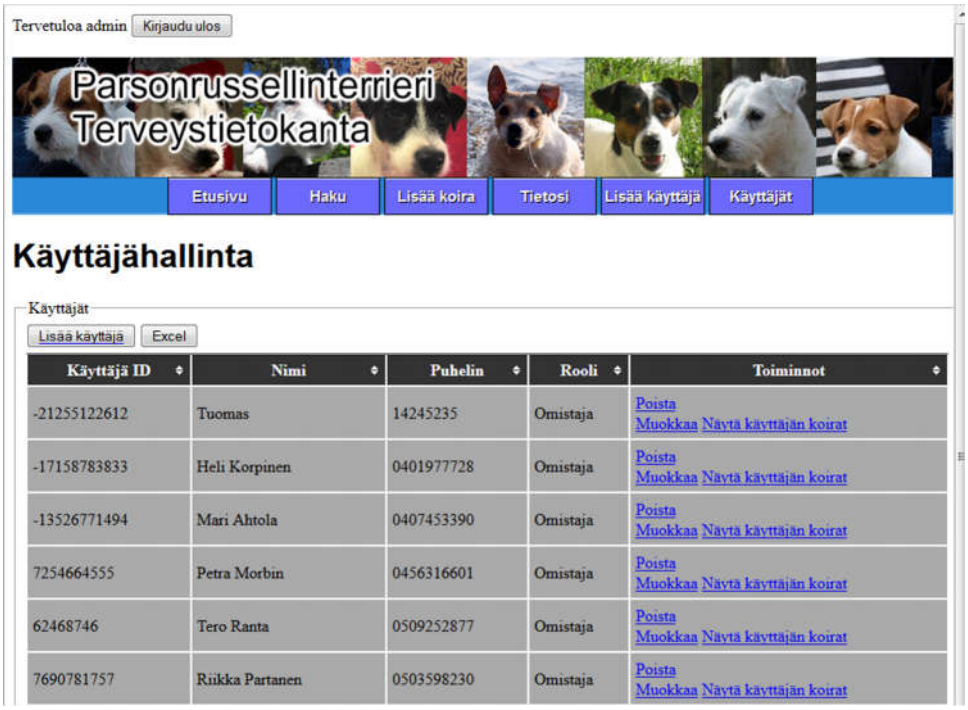


Kuva 32. Koiran haku

Hakuehtoina voi käyttää koiran nimeä, rekisterinumeroa tai sukupuolta. Syöttämällä kenttään jokerimerkin %, voidaan hakea kaikkia koiria, joita kyseinen kenttä koskee.

7.9 Käyttäjien hallinta

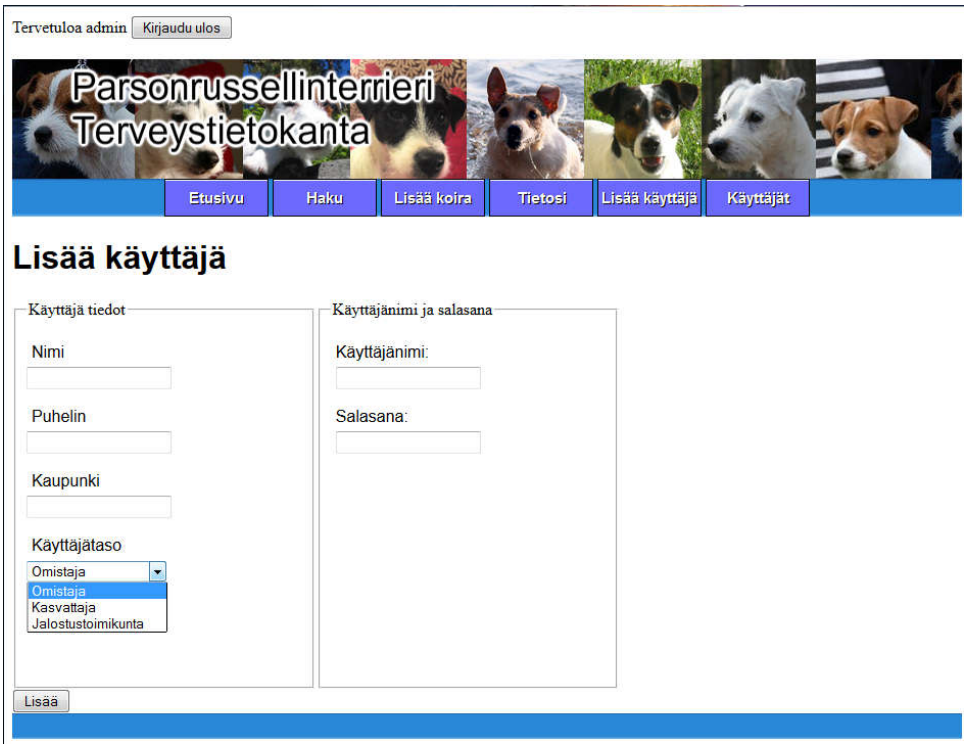
Käyttäjiä hallitsee pääkäyttäjä (Kuva 33). Hänen tehtävänään on jakaa käyttäjille rooleja tarpeen mukaan, mikäli se on joku muu kuin oletusarvo eli koiran omistaja. Käyttäjien hallinta tapahtuu samankaltaisesti kuin koirien hallinta. Jos pääkäyttäjä poistaa käyttäjän, poistuvat tietokannasta myös kaikki hänen tiliinsä liitetyt koirat.



Kuva 33. Käyttäjien hallinta

7.10 Käyttäjän lisäys

Koirien omistajia ei tarvitse erikseen lisätä, koska jokainen rekisteröitynyt käyttäjä saa automaattisesti tämän käyttäjätason. Pääkäyttäjän on kuitenkin lisättävä kasvattajat ja jalostustoimikunnan jäsenet erikseen (Kuva 34) manuaalisesti, antaen heille samalla vastaavan roolin järjestelmään. Rooleilla on erilaiset oikeudet koirien tietojen tarkasteluun ja muokkaukseen.



Kuva 34. Käyttäjän lisäys

8 Yhteenveto

Asiakkaan kannalta projektin tavoite saavutettiin, koska hänelle pystyttiin toimittamaan toimiva järjestelmä. Järjestelmään kuuluivat tietokanta ja verkkopohjainen käyttöliittymä, jotka saatiin toimimaan hyvin ennen luovutusta loppukäyttäjälle. Muutamia pienempiä vikoja jäi järjestelmään, mutta ne eivät haittaa järjestelmän käyttöä sanottavasti. Parsonrussellinterrierit ry:n terveystietokanta tarjoaa nyt alustan, johon kyseisen koirarodun terveystiedot kootaan ja niiden tarkastelu ja muokkaus onnistuu keskitetysti ja turvallisesti.

Ennen työn aloittamista olin ollut tekemisissä tietokantojen kanssa lähinnä Saimaan ammattikorkeakoulun järjestämillä kursseilla. Kokemusta tietokannan määrittelystä, suunnittelusta, toteutuksesta ja testauksesta ei ollut aikaisemmista työpaikoista tai harrastuksista. Sen sijaan verkko-ohjelmoinnista oli hieman aikaisempaa kokemusta. Tämän projektin aikana tärkeimmät opitut asiat olivat projektin hallinta, työtehtävien suoritukseen tarvittavan aikamäärän arviointi ja itse tietokantaohjelmoinnin teoria ja käytäntö.

Projekti sujui hyvin joistain ongelmista huolimatta. Itse suunnittelua ja toteutusta hidasti jonkin verran vaatimusten ja haluttujen ominaisuuksien muuttuminen ja lisääntyminen projektin edetessä. Suurin osa pystyttiin toteuttamaan, mutta aikataulu venyi melko paljon alkuperäisestä tavoitteesta. Yhteistyö projektiryhmän kesken oli hyvää, varsinkin Tuomas Porvalin panoksesta käyttöliittymän parissa oli suurta apua.

Jälkikäteen arvioituna aikataulu olisi pitänyt alussa tehdä paljon väljemmäksi, sillä nyt eräiden työtehtävien tekemiseen kului jopa kolminkertainen määrä tunteja verrattuna suunniteltuun. Myös vaatimusmäärittely olisi kannattanut tehdä heti aluksi, nyt se valmistui lopulliseen muotoonsa vasta melko loppuvaiheessa projektia.

Kuvat

Kuviot

Lähteet

Haikala, I & Mikkonen, T. 2011. Ohjelmistotuotannon käytännöt. Helsinki: Talentum

Hernandez, M. 2000. Tietokannat – suunnittelu ja toteutus. 1. painos. Jyväskylä: Gummerus Kirjapaino Oy

Hiltunen, M. 2004. Vesiputouksen malli. Oulun seudun ammattiopisto, Kaukovainion liiketalouden yksikkö. http://www.okol.org/verkkokurssit/datanomi/tietojarjestelmien_kaytto_ja_kehittaminen/johdatus_tietojarjestelmiin/kehittamistyon_vaiheet_ja_elikaarimallit/kehittamistyon_vaiheet_ja_elinkaarimallit_asia.htm. Luettu 30.11.2015

Hovi, A.; Huotari, J. & Lahdenmäki, T. 2003. Tietokantojen suunnittelu & indeksointi. 1. painos. Jyväskylä: Docendo Finland Oy

Immonen, 2002. Johdatus ohjelmistotuotantoon. http://cs.joensuu.fi/~jimmonen/jot_moniste/jot_moniste_121.html. Luettu 30.11.2015.

Kellokoski, P. 2013. UML perusteet. http://www.slideshare.net/PassoK/uml-perusteet. Luettu 1.12.2015

Kokko, J. 1996. Vaatimusmäärittely. http://www.soberit.hut.fi/tik-76.115/96-97/palautukset/groups/apina/su/documents/vm.html. Luettu 2.12.2015.

Kämäri, J-P. 2011. Tiedonkeruu (ISAPI dll). http://cs.stadia.fi/~kamaj/jp/tiedonke.htm. Luettu 1.12.2015

Lahtonen, T. 2002. SQL. 1. painos. Jyväskylä: Docendo Finland Oy

Mustonen-Ollila, E. 2006. Relaatiotietokanta. http://www2.it.lut.fi/kurssit/05-06/Ti5214300/kalvot.pdf. Luettu 1.12.2015.

Wikipedia: Scrum. https://fi.wikipedia.org/wiki/Scrum. Luettu 2.12.2015.

Wikipedia: Ketterä ohjelmistokehitys. http://fi.wikipedia.org/wiki/Ketterä_ohjelmistokehitys. Luettu 30.11.2015.