Juuso Pakarinen

Eläinlääkäriohjelmisto mobiililaitteille

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

5.5.2013

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Juuso Pakarinen Eläinlääkäriohjelmisto mobiililaitteille 33 sivua 5.5.2013
Tutkinto	insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Lehtori Vesa Ollikainen Toimitusjohtaja Janne Huttunen

Insinöörityön tavoitteena oli esitellä Provet.mobi-eläinlääkäriohjelmisto mobiililaitteille. Tämä projekti sai alkunsa vuonna 2012 kesällä. Tarkoituksena oli luoda Finnish Net Solutions Oy:n Provet Win 2-sovellukselle mobiiliversio.

Tässä työssä yhdistyvät JQuery Mobile -kehys, CakePHP-sovelluskehys sekä SOAP-rajapinta. Näillä tekniikoilla saatiin toteutettua monipuolinen ja kattava mobiilisovellus eläinlääkäreille perinteisen tietokonesovelluksen rinnalle.

Opinnäytetyössä keskitytään eri tekniikoiden esille tuontiin sekä sovelluksen arkkitehtuuriin ja esittelyyn. Lopussa käydään läpi projektin eri vaiheita ja sitä, missä Provet.mobi on nykyään.

Avainsanat	JQuery Mobile, CakePHP, SOAP

Author(s) Title Number of Pages Date	Juuso Pakarinen Veterinarian software for mobile devices 33 pages 5 May 2013
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Vesa Ollikainen Janne Huttunen, CEO

The objective of this thesis was to introduce Provet.mobi, software for veterinarians that runs on mobile devices. This project was originally started in the summer of 2012. The purpose was to create a mobile version of Provet Win 2 software that Finnish Net Solutions Oy has.

This thesis combines JQuery Mobile framework, CakePHP framework and SOAP web service. With these technologies we were able to create a diverse and comprehensive mobile software, alongside with the traditional Provet Win 2 PC software.

This thesis focuses on these technologies and how they were used, the architecture of the software and how it works. The different stages of the project are introduced in the end of the thesis, and also a little bit where Provet.mobi is today.

Keywords	JQuery Mobile, CakePHP, SOAP

Sisällys

Lyhenteet

SOAP	SOAP on tietoliikenneprotokolla, joka on tarkoitettu kommunikaation eri sovelluksien välille. Kommunikaatio tapahtuu internetin yli. SOAP-syntaksi perustuu XML-kieleen.
WSDL	Web Service Description Language joka on syntaksiltaan XML-kieltä ja on myös XML-dokumentti. WSDL:llä kuvataan web-teknologioihin perustuva Web-palvelu.
MVC	Model-View-Controller eli malli-näkymä-ohjain on ohjelmistoarkkitehtuurityyli, jossa käyttöliittymä erotetaan sovelluksen logiikasta. Arkkitehtuurissa ohjelma jaetaan kolmeen osaan: malliin, näkymään ja ohjaimeen. Malli kuvaa sovelluksen logiikkaa ja tiedon tallentamista. Näkymä kuvaa sovelluksen ulkoasua ja grafiikkaa käyttöliittymässä. Ohjain on mallin sekä näkymän välissä ja ohjaa käyttäjän käskyjä näiden kahden komponentin välillä.
UDDI	Universal Description Discovery and Integration (UDDI) on palvelu, johon voidaan rekisteröidä web service -palveluita. Sitä voisi verrata puhelinluetteloksi, jonne on listattu mm. palvelun tarjoaja, kuvaus palvelusta sekä url-osoite palvelun WSDL-tiedostoon.

1 Johdanto

Nykypäivänä mobiilisovellukset ovat yleistymässä kovaa vauhtia. Tämä luo kuitenkin haasteita kehittäjille, koska älypuhelimia on monia markkinoilla ja niissä on eri käyttöjärjestelmät ja ne toimivat eri alustoilla. Jos haluaisi kehittää sovelluksen kaikkiin eri älypuhelimiin, pitäisi sovellus toteuttaa jokaiselle alustalle erikseen. Tämä tarkoittaisi suurta työtaakkaa kehittäjille sekä useiden sovelluskehitystyökalujen latausta ja asennusta. Tämän lisäksi ohjelmistokehittäjän pitäisi osata useita eri ohjelmointikieliä.

Tämän työn tarkoituksena on esitellä Provet.mobi-eläinlääkäriohjelmisto mobiililaitteille. Toinen tarkoitus tällä työllä on esitellä ja arvioida eri tekniikoita, joita projektissa on käytetty. Tämä työ toteutettiin Finnish Net Solutions Oy -nimiselle yritykselle, jossa olen töissä. Finnish Net Solutions on erikoistunut räätälöityihin selainkäyttöisiin verkkopalveluihin. Suurimpina asiakkaina meillä ovat eläinlääkärit sekä hoitoalan ammattilaiset.

Tämä projekti sai alkunsa kesällä 2012, kun halusimme luoda Provet Win 2 -nimisestä tuotteestamme mobiiliversion eläinlääkäreille. Tärkeimpänä kriteerinä oli luoda sovellus, joka olisi yhteensopiva kaikkien mobiililaitteiden kanssa. Siksi päätimme luoda selainpohjaisen sovelluksen hyödyntäen JQuery Mobile -nimistä sovelluskehystä. Tämä mahdollistaa sen, että kaikki älypuhelimet jotka on varustettu selaimella, voivat käyttää Provet.mobi-palvelua. Toinen erittäin suuri haaste tässä projektissa oli tiedon synkronointi kahden palvelun välissä. On tärkeätä, että Provet.mobin tiedot välittyvät Provet Win 2 -palveluun. Myös Provet Win 2 -palvelusta välitetään tiedot Provet.mobiin.

Provet Win 2 on eläinlääkäriohjelmisto, joka on suunniteltu eläinlääkäreille päivittäiseksi työkaluksi, jolla voi huolehtia tietojen tallentamisesta sekä organisoinnista. Se tallentaa tietoja eläimistä, eläinten omistajista sekä suoritetuista hoitotoimenpiteistä. Provet Win 2 hoitaa myös monipuoliset toiminnot liittyen laskutukseen; ohjelma luo käteiskuitit sekä laskut ja huomioi matkakorvaukset, korotukset sekä subventiot. [1.]

2 Tekniikat

Tässä luvussa käydään läpi tekniikat, joilla Provet.mobi on toteutettu. Esittelen CakePHP-sovelluskehyksen, JQuery Mobile -kehyksen sekä SOAP rajapinta -kehyksen. Nämä kaikki tekniikat olivat keskeisessä roolissa tässä työssä.

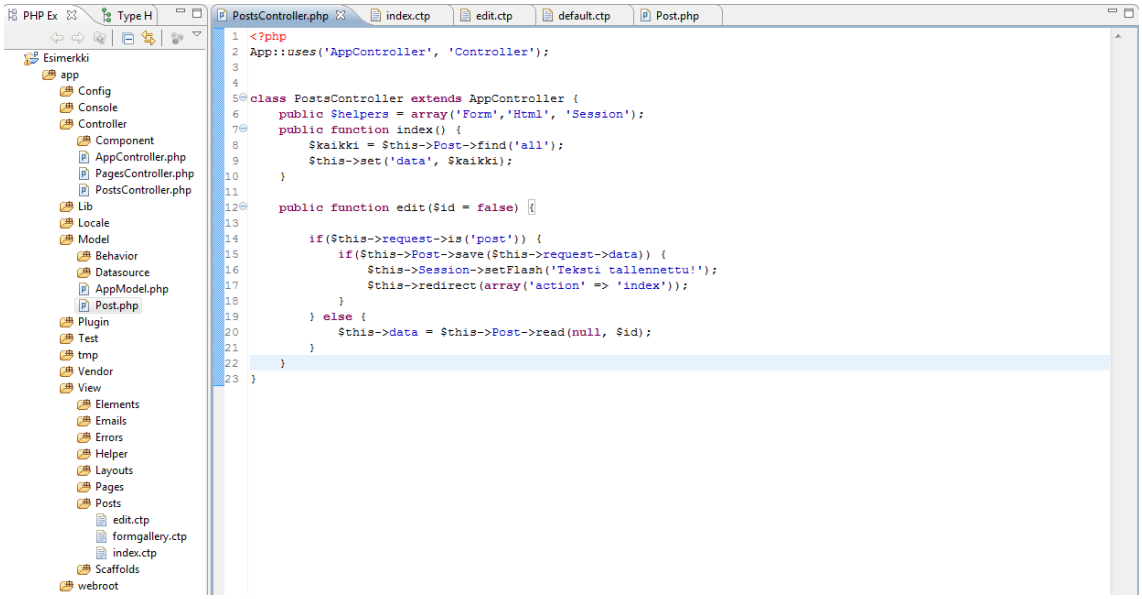
2.1 CakePHP

CakePHP on sovelluskehys, joka tarjoaa helpon lähestymistavan web-ohjelmointiin php-kielellä. Sovelluskehyksellä tarkoitetaan apuvälinettä, jonka tarkoituksena on nopeuttaa ohjelmistotuotteiden valmistusta. Sovelluskehys tarjoaa valmiiksi rakennettuja osia, joita ei tarvitse itse kirjoittaa uudelleen. Esimerkiksi CakePHP tarjoaa valmiiksi tietokantayhteyden luonnin. Käyttäjän tarvitsee vain määritellä muutama asetusta. CakePHP:n suosio onkin siinä, että se on helppo omaksua verrattuna muihin sovelluskehyksiin. CakePHP on helppo ottaa käyttöön: ei tarvitse kuin määrittää tietokanta määritystiedostoon, jolloin on valmis aloittamaan kehitystyön.

Sovelluskehys käyttää myös MVC-arkkitehtuuria, joka on hyvä ohjelmointikäytäntö. Model-View-Controller (MVC) eli malli-näkymä-ohjain on ohjelmistoarkkitehtuurityyli, jossa käyttöliittymä erotetaan sovelluksen logiikasta. Arkkitehtuurissa ohjelma jaetaan kolmeen osaan: malliin, näkymään ja ohjaimeen. Malli kuvaa sovelluksen logiikkaa ja tiedon tallentamista. Näkymä kuvaa sovelluksen ulkoasua ja grafiikkaa käyttöliittymässä. Ohjain on mallin sekä näkymän välissä ja ohjaa käyttäjän käskyjä näiden kahden komponentin välillä. CakePHP tarjoaa myös hyvät työkalut sähköpostien lähettämiseen, autentikointiin, pääsyn valvontaan, lokalisaatioon, tietoturvaan sekä istuntojen luomiseen ja hallitsemiseen. [2.]

Uusi Cakephp-projekti voidaan aloittaa lataamalla se CakePHP:een kotisivuilta osoitteesta http://cakephp.org/. Etusivulla on linkki uusimpaan ladattavaan versioon CakePHP:sta. CakePHP käyttää MIT-lisenssiä, joka tarkoittaa, että se sallii teoksen käytön kaupallisissa suljetun lähdekoodin projekteissa. Kun CakePHP on ladattu, täytyy se purkaa haluttuun kansioon. Tämän jälkeen kehitystyö voidaan aloittaa. Olen luonut esimerkkiprojektin, jossa esittelen erittäin yksinkertaisen CakePHP-sovelluksen.

Kuvassa 1 on luotu esimerkki CakePHP-projektista. Tämän esimerkin tarkoituksena on havainnollistaa kansiorakennetta CakePHP-projektissa sekä näyttää, kuinka yksinkertaisesti saadaan yhden taulun data haettua kannasta. Kyseinen esimerkki siis hakee tietokannan taulusta ”posts” kaikki tiedot, ja tulostaa ne sivulla. Riviä painamalla käyttäjä saa avattua kyseisen rivin tiedot. Tietoja voidaan muokata ja tallentaa takaisin tietokantaan.



Kuva 1. Esimerkki CakePHP-projektista.

Kun CakePHP otetaan käyttöön, niin täytyy tehdä muutamia määrityksiä. Seuraavassa esimerkissä käydään läpi, kuinka tietokanta määritetään. Tämä kyseinen määritys löytyy Config-kansion sisältä, ja tiedoston nimi on database.php.

```
class DATABASE_CONFIG {
	public $default = array(
		'datasource' => 'Database/Mysql',
		'persistent' => false,
		'host' => 'localhost',
		'port' => '',
		'login' => 'root',
		'password' => 'root1234',
		'database' => 'esimerkki_kanta',
		'schema' => '',
		'prefix' => '',
		'encoding' => ''
	);
```

Esimerkistä käy ilmi, että käytössä on MYSQL-kanta, ja se on käynnissä palvelimen localhostissa. Esimerkissä myös määritellään, millä käyttäjänimellä ja salasanalla kan-

taan kirjaudutaan sekä mitä tietokantaa käytetään. Oletuksena CakePHP käyttää tätä oletusmääritystä, joka löytyy valmiina määritystiedostosta, mutta sinne voidaan luoda muitakin tietokantayhteyksiä samalla tyylillä. Eri kanta voidaan ottaa käyttöön mallin sisällä komennolla $useDBConfig = '$maarityksen_nimi'.

CakePHP:ssa on myös tarkka nimeämisavaruus sekä kansiorakenne käytössä. Ohjaimien nimet ovat monikossa, ja niiden perään lisätään Controller ja nämä tiedostot sijaitsevat /app/Controller/ kansiossa. Esimerkiksi jos ohjaimen nimi olisi Products, niin tulisi siitä ProductsController. Mallien nimet ovat taas yksikössä, eli se olisi Product pelkästään, ja nämä tiedostot sijaitsevat /app/Model/-kansiossa. Näkymien nimet taas noudattavat ohjaimen funktioiden nimiä. Esimerkiksi jos ProductsController-luokassa on funktio nimeltä 'index()', niin näkymätiedoston nimeksi tulisi index.ctp ja se sijaitsisi /app/View/Product/-kansiossa.

Kuten kuvasta 1 huomataan, PostsController sijaitsee Controller-kansion sisällä. Vastaavasti Post sijaitsee Model-kansion sisällä, ja näkymät ovat View/Posts/-kansion alla. Jos kuvassa olevaa koodia tutkitaan, niin sieltä löytyy kaksi metodia: toisen nimi on index ja toisen edit. Vastaavan nimiset näkymätiedostot löytyvät myös, eli index.ctp sekä edit.ctp.

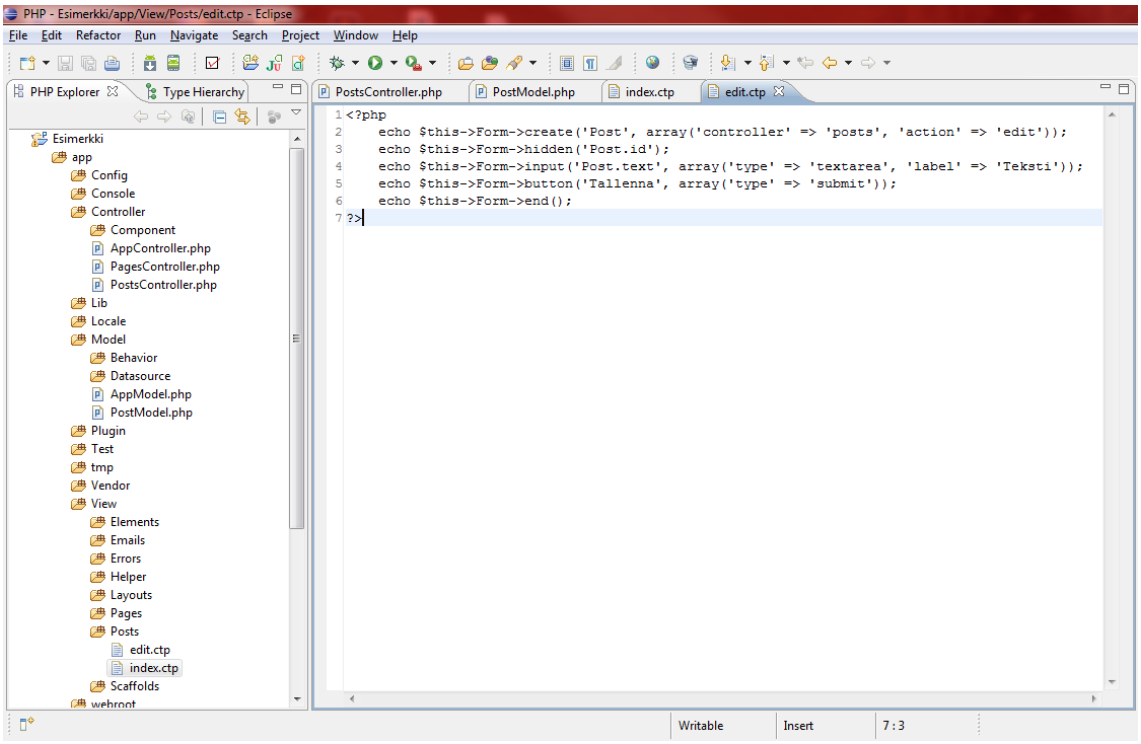
Jos tutkimme tarkemmin esimerkiksi index-nimisen funktion sisältöä, niin huomaamme, että siellä on haettu kaikki Post-taulusta oleva data yhdellä rivillä:

```
$kaikki = $this->Post->find('all');
```

CakePHP:ssa ei tarvitse itse kirjoittaa omia SQL-lauseita, koska CakePHP tarjoaa omat metodit datan hakua varten tietokannasta. Tämä helpottaa huomattavasti koodin kirjoittamista ja sen näyttämistä näkymäsivulla. Saman funktion viimeisellä rivillä asetamme kaikki haetut tiedot "data"-nimiseen muuttujaan, joka on käytössä index.ctp-tiedostossa.

Toinen funktio, joka kuvassa näkyy, on tarkoitettu datan hakemiseen kannasta, sen muokkaamiseen ja lopulta sen tallentamiseen takaisin. Funktio saa parametrikseen id:n jonka perusteella kannasta haetaan kyseinen rivi. Tämän rivin tiedot saadaan näkymäsivulle asettamalle ne "$this->data"-nimiseen muuttujaan. Kuvassa 2 on näkymäsivu, jossa on yksinkertainen lomake. Siinä on tekstikenttä, johon haettu rivi tulostetaan.

Tekstikenttää voidaan muokata ja lopuksi tallentaa. Kun tiedot on tallennettu, asetetaan flash-viesti ”Teksti tallennettu” -ruudulle ja ohjataan takaisin index.ctp-sivulle.



Kuva 2. Näkymäsivun esimerkki koodi (edit.ctp)

Kuvassa 2 luodaan lomake, johon aiemmin haettu data sijoitetaan. Lomake luodaan rivillä 2. Siinä määritetään, että ohjain on posts ja funktio edit, eli kun käyttäjä painaa tallenna, niin lomake ohjaa kuvan 1 mukaisesti post-ohjaimen funktioon edit. Kolmannella rivillä on kyseisen haetun rivin id, joka on käyttäjältä piilotettu kenttä. Rivillä 4 on tekstikenttä, johon sijoitetaan haettu data. Rivillä 5 luodaan tallenna painike, joka on tyypiltään submit, eli se lähettää lomakkeen eteenpäin.

2.2 JQuery Mobile

JQuery Mobile on kosketus optimoitu JavaScript-kirjasto mobiililaitteille. Kyseinen kirjasto on tehty erittäin selainyhteensopivaksi ja toimivaksi moniin mobiililaitteisiin ja siksi tässä työssä päädyttiin sen käyttöön. Kuvassa 3 nähdään, mitkä laitteet tukevat JQuery Mobilea. Vihreä väri (A) tarkoittaa parasta tukea, keltainen (B) keskitasoa ja punainen (C) matalaa tasoa.

Platform	Version	Native
iOS	v2.2.1	B
	v3.1.3, v3.2	A
	v4.0	A
Symbian S60	v3.1, v3.2	C
	v5.0	A
Symbian UIQ	v3.0, v3.1	
	v3.2	
Symbian Platform	v.3.0	A
BlackBerry OS	v4.5	C
	v4.6, v4.7	C
	v5.0	B
	v6.0	A
Android	v1.5, v1.6	A
	v2.1	A
	v2.2	A
Windows Mobile	v6.1	C
	v6.5.1	C
	v7.0	A
webOS	**1.4.1**	A
bada	**1.0**	A
Maemo	**5.0**	B
MeeGo	1.1*	A*

Kuva 3. JQuery Mobilen eri mobiililaitteiden tuki. [3.]

JQuery Mobilen -ominaisuuksiin kuuluu HTML 5 -tekniikat sekä AJAX-pohjainen sivunavigointi. Tämä tarkoittaa sitä, että sivujen vaihdokset on saatu näyttämään natiivisovelluksen tapaisilta. JQuery Mobile myös skaalautuu eri näyttökokoihin ilman erillistä ohjelmointia ja osaa kääntää näytön, kun mobiililaite käännetään. [3.]

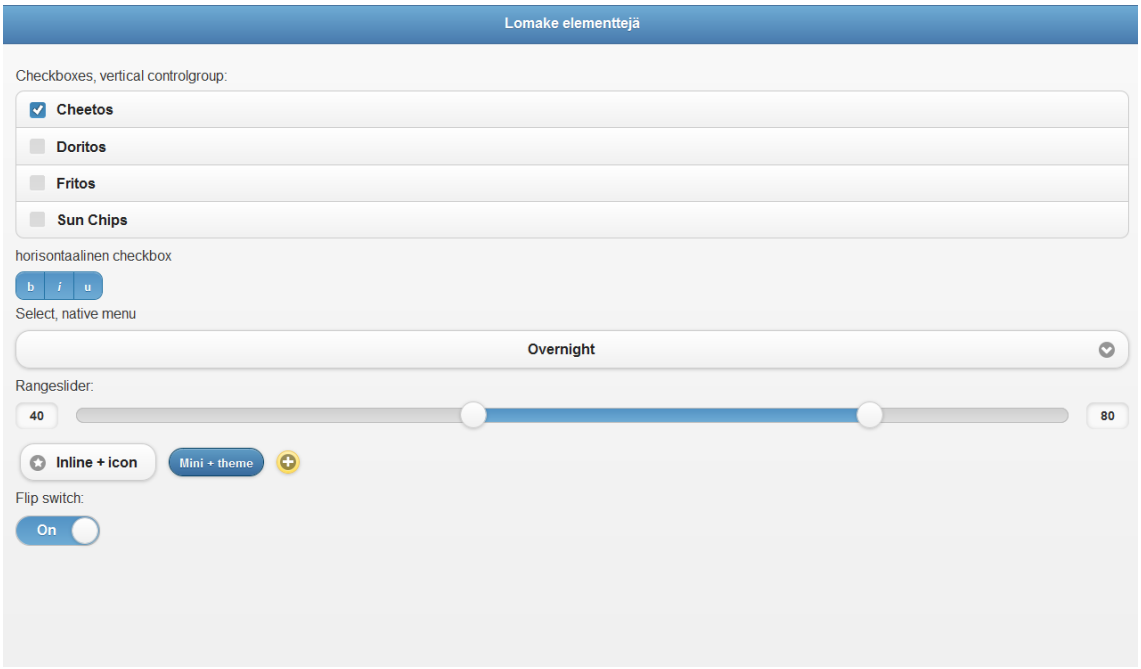
Kun JQuery, JQuery Mobile sekä tarvittavat JavaScript ja CSS-tiedostot on ladattu, voidaan ne ottaa käyttöön mobiilisivuilla määrittämällä ne html-dokumentin header-osioon. Seuraavassa koodiesimerkissä on tavallisen JQuery Mobilen -sivun rakenne.

```
<!DOCTYPE html>
<html>
<head>
   <title>Page Title</title>
        <meta name="viewport" content="width=device-width, initial-
scale=1">
        <script type="text/javascript"
src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
```

```
        <script type="text/javascript"
src="http://code.jquery.com/mobile/latest/jquery.mobile.min.js"></scri
pt>
        <link type="text/css"
href="http://code.jquery.com/mobile/latest/jquery.mobile.min.css"
rel="stylesheet" />
</head>
<body>

<div data-role="page" id="first" data-theme="a">

        <div data-role="header">
                <h1>Page Title1</h1>
        </div><!-- /header -->

        <div data-role="content">
                <p>Page content goes here.</p>
                <a href="#second">Go to second page</a>
        </div><!-- /content -->

        <div data-role="footer">
                <h4>Page Footer1</h4>
        </div><!-- /footer -->
</div><!-- /page -->

</body>
</html>
```

Kuten esimerkistä nähdään, niin HTML-dokumentti on HTML 5 -tekniikkaa ja JQuery sekä JQuery Mobile ovat header-osiossa. Header-osiossa on myös määritelty meta-elementtiin, että kyseessä on mobiililaite ja että sivu skaalataan laitteen leveyden mukaan. Itse sivun sisältö on body-osion sisällä. JQuery Mobilessa käytetään paljon ”data-role” attribuuttia div-elementeissä. Ensimmäisessä div-elementissä määritellään ”data-role”-attribuutilla, että kyseessä on sivu, jolla on JQuery Mobilen tarjoama teema nimeltä ”a”. Tämän jälkeen määritellään eri div-elementeillä header, content sekä footer osiot. JQuery Mobilen kaikki sivut käyttävät samanlaista rakennetta.

JQuery Mobilessa on myös paljon valmiita lomake-elementtejä valmiina käytettäväksi. Esimerkkiprojekti, jonka esittelin aiemmassa luvussa, sisältää näkymäsivun, jossa on esillä lomake elementtejä (kuva 4). Esittelen, kuinka näitä elementtejä saa käyttöön omassa koodissa.



Kuva 4. Lomake elementtien esittely

Kuten kuvasta 4 nähdään, nämä lomake elementit on suunniteltu sormella paineltaviksi. Ne ovat isoja komponentteja, joihin sormella osuu helposti. JQuery Mobile tarjoaa suoraan nämä komponentit käyttöön ilman, että tarvitsee itse säätää mitään. Seuraavassa koodiesimerkissä esittelen, kuinka se onnistuu.

```
<div data-role="header" data-theme="b">
    <h1>Lomake elementtejä</h1>
</div>

<div data-role="content">

        <fieldset data-role="controlgroup">
            <legend>Checkboxes, vertical controlgroup:</legend>
            <input name="checkbox-1a" id="checkbox-1a" checked=""
type="checkbox">
            <label for="checkbox-1a">Cheetos</label>
            <input name="checkbox-2a" id="checkbox-2a" type="checkbox">
            <label for="checkbox-2a">Doritos</label>
            <input name="checkbox-3a" id="checkbox-3a" type="checkbox">
            <label for="checkbox-3a">Fritos</label>
            <input name="checkbox-4a" id="checkbox-4a" type="checkbox">
            <label for="checkbox-4a">Sun Chips</label>
        </fieldset>

        <fieldset data-role="controlgroup" data-type="horizontal" data-
mini="true">
            <legend>horisontaalinen checkbox</legend>
            <input name="checkbox-6" id="checkbox-6" type="checkbox">
            <label for="checkbox-6">b</label>
```

```
            <input name="checkbox-7" id="checkbox-7" checked=""
type="checkbox">
            <label for="checkbox-7"><em>i</em></label>
            <input name="checkbox-8" id="checkbox-8" type="checkbox">
            <label for="checkbox-8">u</label>
        </fieldset>

        <label for="select-choice-1" class="select">Select, native
menu</label>
        <select name="select-choice-1" id="select-choice-1">
            <option value="standard">Standard: 7 day</option>
            <option value="rush">Rush: 3 days</option>
            <option value="express">Express: next day</option>
            <option value="overnight">Overnight</option>
        </select>

        <div data-role="rangeslider">
            <label for="range-1a">Rangeslider:</label>
            <input name="range-1a" id="range-1a" min="0" max="100" val-
ue="40" type="range">
            <label for="range-1b">Rangeslider:</label>
            <input name="range-1b" id="range-1b" min="0" max="100" val-
ue="80" type="range">
        </div>

        <a href="#" data-role="button" data-inline="true" data-
icon="star">Inline + icon</a>
        <a href="#" data-role="button" data-inline="true" data-
theme="b" data-mini="true">Mini + theme</a>
        <a href="#" data-role="button" data-inline="true" data-
icon="plus" data-iconpos="notext" data-theme="e" data-mini="true">icon
only button</a>

        <label for="slider2">Flip switch:</label>
        <select name="slider2" id="slider2" data-role="slider">
            <option value="off">Off</option>
            <option value="on">On</option>
        </select>

</div>
```

Kuten koodiesimerkistä nähdään, niin kaikki lomakedata on sen div-elementin sisällä, jonka data-role-attribuutilla on arvo content. Aluksi luon erityyppisiä valintaruutuja näytölle. Nämä luodaan fieldset-elementin sisälle ja niille annetaan data-role-attribuuttille arvo controlgroup. Alemmalle valintaruudulle annetaan vielä toiselle attribuutille data-type arvoksi horizontal, eli siis se on vakaatasossa, kuten kuvasta huomataan.

Tämän jälkeen luon sivulla alasvetovalikon. Seuraava erikoinen määritys on slider-elementissä. Se luodaan div-elementin sisään, jonka data-role-attribuutilla on arvona rangeslider. Elementin sisällä on kaksi input-elementtiä, jotka ovat tyypiltään ”range”. Ensimmäinen input-elementti on liukusäätimen vasemman puoleinen arvo. Toinen input-elementti on liukusäätimen oikean puoleinen arvo.

Painikkeet luodaan linkkeinä sivulle, eli a-elementeillä. Näille data-role-attribuutteille annetaan arvoksi button. JQuery Mobile osaa luoda näistä linkeistä painikkeita. Painikkeille voidaan antaa myös ikoneita, kuten ensimmäisessä on tähden kuva. Ikoni voidaan antaa painikkeelle data-icon-attribuutilla. On myös mahdollista olla painike ilman tekstiä, jolle määritetään data-iconpos=”notext” attribuutti.

Viimeiseksi luon on-off-kytkimen. Se luodaan select-elementillä, mutta sinne annetaan data-role=”slider”-attribuutti mukaan. Nämä kytkimet ovat erittäin hyviä ja käyttökelpoisia sivuilla, joissa on yleensä kyllä-ei-valintoja.

Erittäin tärkeitä ovat myös lista-elementit, joita voidaan luoda erittäin helposti JQuery Mobilessa. Seuraavassa koodiesimerkissä on yksinkertainen listanäkymä.

```
<ul data-role="listview" data-filter="true" data-filter-
placeholder="Search fruits..." data-inset="true">
                    <li><a href="#">Apple</a></li>
                    <li><a href="#">Banana</a></li>
                    <li><a href="#">Cherry</a></li>
                    <li><a href="#">Cranberry</a></li>
                    <li><a href="#">Grape</a></li>
            <li><a href="#">Orange</a></li>
```

Kun lista luodaan, annetaan sille data-role=”listview”-attribuutti -elementin sisälle. Tämä kyseinen esimerkki sisältää myös data-filter-attribuutin, joka tarkoittaa, että se luo listanäkymän yläpuolelle tekstikentän, johon voi kirjoittaa haun. Haku kohdistuu tähän listaan ja etsii listasta hakusanalla olevaa lista riviä. Se karsii muut rivit pois näkyvistä samalla. Tässä listassa listaelementit ovat samalla myös linkkejä.

2.3 Web Service

Web Services -termillä tarkoitetaan tietokoneiden välistä vuorovaikutusta tietoverkon yli. Tämä käsite saatetaan usein sekoittaa verkkopalveluiden kanssa. Verkkopalveluilla tarkoitetaan ihmisille tarkoitettuja palveluita, joita verkossa on. Tästä hyvä esimerkki olisi Wikipedia, joka on verkkopalvelu, mutta samalla se tarjoaa web services -rajapinnan, jolla käyttäjät voivat lukea ja muokata sitä.

2.3.1 SOAP ja WSDL

SOAP on tietoliikenneprotokolla, joka mahdollistaa datan lähettämisen internetin yli sovellusten välillä. SOAP on alustariippumaton, joten sitä voidaan hyvin käyttää esimerkiksi Windows-sovelluksien ja web-sovelluksien välillä. SOAP:iin liittyy läheisesti WSDL tiedoston määritys. Tässä tiedostossa määritellään XML-kielellä, mitä SOAP-rajapinta sisältää. Kun WSDL-tiedosto on kirjoitettu, niin SOAP osaa lukea WSDL-tiedoston ja lukea, mitä funktioita se sisältää. Tämän jälkeen niitä voidaan käyttää normaalin funktion tavoin sovelluksessa.

WSDL-tiedosto koostuu abstraktista sekä konkreettisesta osasta. Abstraktissa osiossa kuvataan palvelun rajapinta. Konkreettisessa osiossa kuvataan käytetty protokolla sekä määritetään palvelun osoite tietoverkossa. Seuraavassa esimerkissä on yksinkertainen Helloservice.wsdl tiedosto. [4.]

```
<definitions name="HelloService"
   targetNamespace="http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <message name="SayHelloRequest">
      <part name="firstName" type="xsd:string"/>
   </message>
   <message name="SayHelloResponse">
      <part name="greeting" type="xsd:string"/>
   </message>

   <portType name="Hello_PortType">
      <operation name="sayHello">
         <input message="tns:SayHelloRequest"/>
         <output message="tns:SayHelloResponse"/>
      </operation>
   </portType>

   <binding name="Hello_Binding" type="tns:Hello_PortType">
   <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>
   <operation name="sayHello">
      <soap:operation soapAction="sayHello"/>
      <input>
         <soap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:examples:helloservice"
            use="encoded"/>
      </input>
      <output>
         <soap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```

```
            namespace="urn:examples:helloservice"
            use="encoded"/>
      </output>
   </operation>
   </binding>

   <service name="Hello_Service">
      <documentation>WSDL File for HelloService</documentation>
      <port binding="tns:Hello_Binding" name="Hello_Port">
         <soap:address
            location="http://www.examples.com/SayHello/">
      </port>
   </service>
</definitions>
```

Esimerkistä nähdään, että definitions-elementin attribuutissa on määritetty palvelun osoite, joka on ”http://www.examples.com/wsdl/HelloService.wsdl”. Tämän jälkeen on määritetty ”sayHelloRequest”-niminen viesti, joka saa parametriksi nimen. Tämän alla on määritetty ”sayHelloResponse” eli se, mitä palautetaan takaisin. Operaatio luodaan portType-osiossa nimellä ”sayHello” ja siinä määritetään, että parametreina otetaan sisään ”sayHelloRequest”-osiossa määritetty string-tyyppinen nimi. Operaatio palauttaa taas vastaavasti ”sayHelloResponse”-osiossa määritetyn string-tyyppisen tekstin. Seuraavaksi wsdl:ssä määritetään, kuinka operaatiot sidotaan. Tästä esimerkistä käy ilmi, että operaatio sidotaan SOAP-kutsuksi, koska binding-elementin sisällä on seuraavanlainen rivi:

```
 <soap:binding style="rpc”
transport="http://schemas.xmlsoap.org/soap/http"/>
```

Tämän jälkeen WSDL:ssä määritetään varsin suoraviivaisesti, mikä operaatio on kyseessä. Lopuksi service-elementissä määritetään, mitä portteja tuetaan rajapinnassa sekä mistä osoitteesta se löytyy.

Kun WSDL on kirjoitettu, voidaan SOAP:lla tehdä palvelinpuolen funktiot, joita käytetään sitten asiakaspuolella. Seuraavassa esimerkissä on yksinkertaista php-kieltä, jossa luodaan uusi ilmentymä SOAP:sta ja lisätään siihen funktio. [5.]

```
<?php
if(!extension_loaded("soap")){
  dl("php_soap.dll");
}

ini_set("soap.wsdl_cache_enabled","0");
$server = new SoapServer("hello.wsdl");

function doHello($yourName){
```

```
  return "Hello, ".$yourName;
}

$server->AddFunction("doHello");
$server->handle();
?>
```

Esimerkistä nähdään, että kun luodaan uutta SOAP-ilmentymää nimeltä $server, niin annetaan sille parametriksi wsdl-tiedosto. Tämän jälkeen on kirjoitettu funktio ”doHello”, mikä palauttaa tekstin sekä nimen, joka sille annetaan parametriksi. Tämä lisätään sitten SOAP-funktioksi komennolla $server->AddFunction("doHello"). On tärkeä huomata, että tässä määritetty funktio ”doHello” pitää olla samanniminen kuin WSDL-tiedostossa määritetty operaatio.

Kun palvelinpuolen operaatiot on kirjoitettu, voidaan SOAP ottaa käyttöön asiakaspuolella. Tässä on esimerkki, kuinka se tapahtuu. [6.]

```
<?php
try{
  $sClient = new SoapClient('http://localhost/test/wsdl/hello.xml');

  $params = "Aqila";
  $response = $sClient->doHello($params);

  var_dump($response);


} catch(SoapFault $e){
  var_dump($e);
}
?>
```

Tässä esimerkissä luodaan myös ilmentymä SOAP:sta. Tämän jälkeen voidaan ilmentymän avulla kutsua normaalisti ”doHello”-nimistä funktiota, joka määritettiin palvelinpuolella. Tälle funktiolle annetaan parametriksi nimi, ja se palauttaa takaisin tekstin sekä nimen.

2.3.2 UDDI

SOAP-tekniikoihin liittyy myös tekniikka nimeltä UDDI (Universal Description Discovery and Integration). Tämä on palvelu, johon voidaan rekisteröidä omia eri web services -

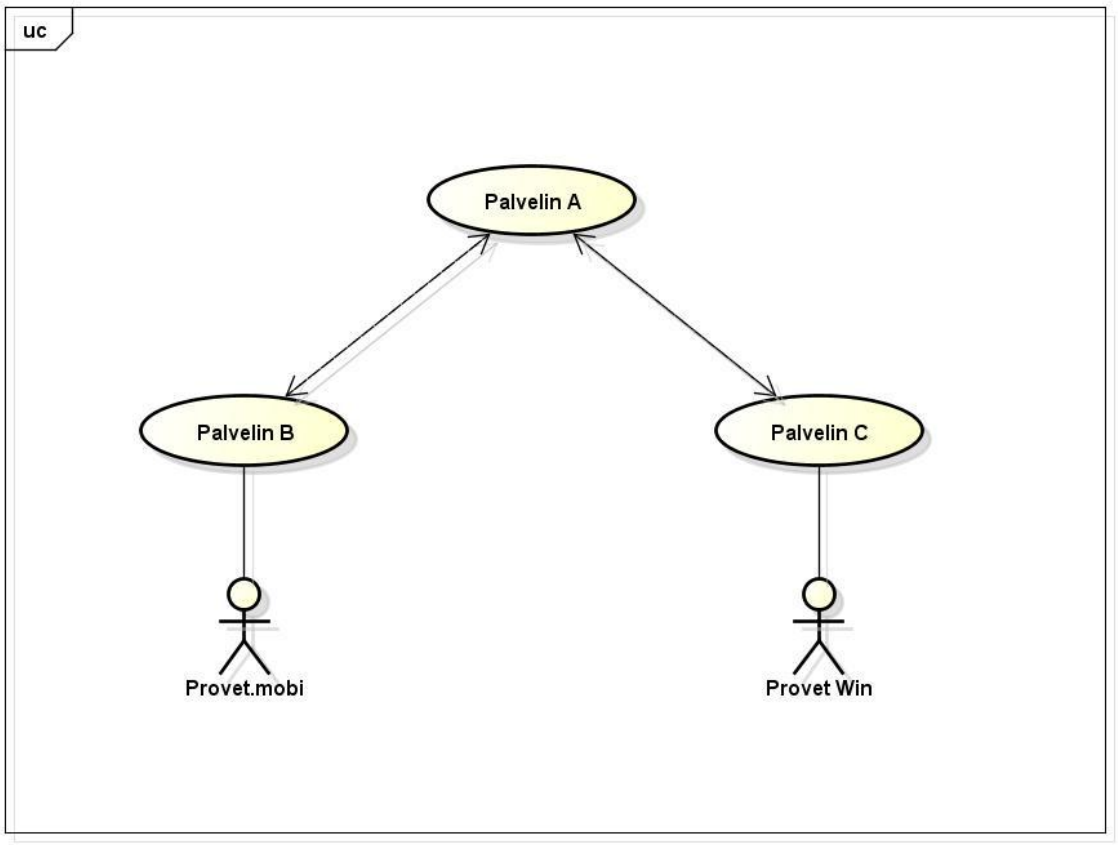
palveluita ja paikka, josta muut palvelut tai sovellukset voivat löytää sinne laitettuja web services -palveluita. Tämä mahdollistaa sen, että muut yritykset tai ryhmät voivat käyttää web services -palveluita, koska täältä löytyy url-osoite, jossa web services -palvelu on sekä kuvaus siitä, mitä se pitää sisällään. [7.]

3 Järjestelmän arkkitehtuuri

Tämä järjestelmä on hyvin perinteinen hajautettu järjestelmä. Provet.mobi-palvelu on käynnissä itsenäisesti omalla palvelimellaan ja samaan aikaan käyttäjät voivat käyttää Provet Win 2 -ohjelmaa. Kaikki valmis data, mitä ohjelmissa luodaan, kuitenkin siirretään yhteiselle keskuspalvelimelle. Palvelut siis toimivat itsenäisesti, mutta kommunikoivat keskuspalvelimelle. [8.]

3.1 Kahdensuuntainen tiedonvälitys

Yksi suurimmista haasteista varmaankin tässä työssä oli kahdensuuntainen tiedonvälitys Provet.mobista keskuspalvelimelle ja sieltä takaisin ja samalla myös keskuspalvelimelta Provet Win 2 -ohjelmaan ja sieltä takaisin. Haasteellista tässä oli myös se, että näiden kahden palvelun omat tietokannat ovat erilaisia, koska Provet.mobi pyörii MySQL-tietokannalla ja Provet Win 2 Microsoft SQL Serverillä. Kuvassa 5 on esitetty järjestelmän kahdensuuntainen tiedonvälitys.



Kuva 5. Järjestelmän kahdensuuntainen tiedonvälitys

Provet Win 2 -ohjelmassa on myös mahdollisuus toimia offline-tilassa. Tällöin eläinlääkärit voivat käydä tiloilla hoitamassa eläimiä ja kirjata hoidot ohjelmaan ilman, että tietokone olisi kytketty internetverkkoon. Jossain vaiheessa kuitenkin olisi suotavaa, että tietokone kytketään verkkoon ja valmiit käynnit synkronoidaan keskuspalvelimelle, jotta ne olisivat myös Provet.mobin käytettävissä.

Perinteisen järjestelmän arkkitehtuuri on varsin yksinkertainen: on esimerkiksi jokin palvelin, missä sivusto toimii. Sivustolla on oma tietokanta, missä kaikki data sijaitsee. Dataa voidaan hakea tietokannasta ja esittää sivustolla. Tietokannassa on eri tauluja, jossa data sijaitsee. Yhden taulun dataa voidaan hakea helposti uniikilla perusavaimella, eli id:llä. Jokaisessa taulussa jokaisella rivillä on oma uniikki perusavain, joka ei voi toistua millään muulla rivillä. Se on usein inkrementaalinen luku, joka alkaa numerosta yksi ja kasvaa aina yhdellä, kun uusi rivi lisätään tietokantaan.

Koska tämä on hajautettu järjestelmä, on tietojen yhdistäminen tuottanut haasteita. Jos esimerkiksi Provet.mobissa luodaan tilalle uusi eläin ja tämän tiedot tallennetaan ja

synkronoidaan keskuspalvelimelle. Provet Win 2 saa tämän eläimen tiedot myös käyttöönsä ja tallentaa sen omaan paikalliseen tietokantaansa. Nyt jos eläimen tietoja muokataan Provet.mobissa ja tallennetaan uudelleen ja synkronoidaan keskuspalvelimelle ja Provet Win 2 lataa uudet tiedot palvelimelta, niin täytyy Provet Win 2 ohjelman osata täsmäyttää uusi data eläimelle omaan paikalliseen tietokantaansa, jossa eläimen tiedot on tallennettu ihan eri id:llä kuin mitä keskuspalvelimelta tulee. Vastaavasti taas kun Provet Win 2 ohjelmassa muokataan kyseistä eläintä ja synkronoidaan se keskuspalvelimelle, niin täytyy siellä tehdä tarkastuksia siitä, onko eläin jo olemassa. Jos eläin on olemassa ja tässä tapauksessa se on, niin sitten täytyy päivittää tietokannan rivin tiedot. Sitä ei voida tutkia perusavaimella, koska eri tietokannoissa se on eri luku.

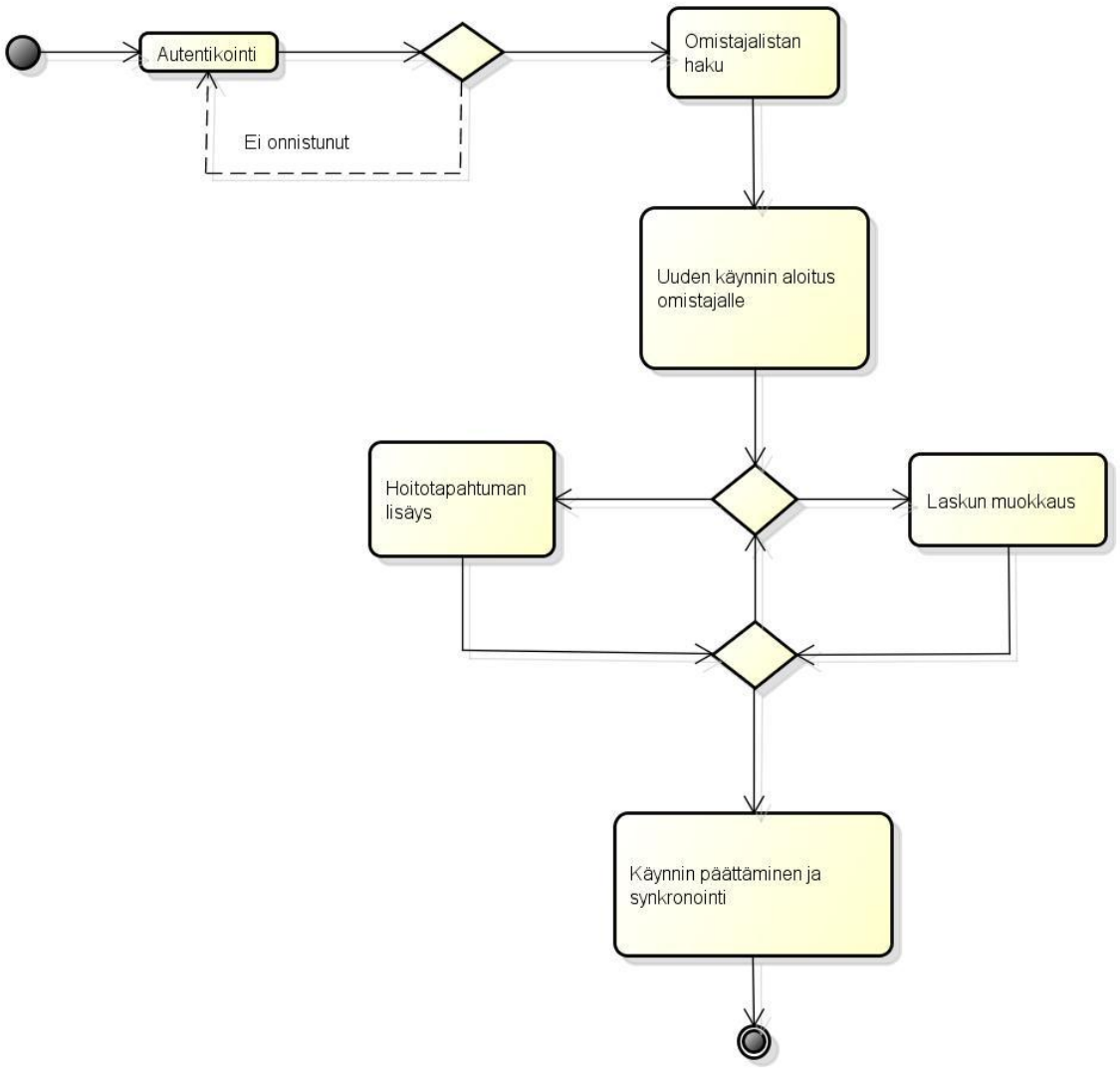
3.2 Tietokannan kuvaus

Provet.mobissa on käytössä MySQL-tietokanta, jossa on seitsemän eri taulua. Näistä keskeisin taulu on se, johon käynti tallennetaan. Tämä on keskeisin taulu, koska kaikki muut taulut viittaavat tähän tauluun. Tiedon synkronointikin liittyy aina käyntiin, käyntejä synkronoidaan, ei yksittäisiä lääkityksiä tai vaikkapa rokotuksia.

Provet.mobin tietokannassa ylläpidetään viite-eheyttä. Muissa tauluissa on viittaus, mihin käyntiin se aina liittyy. Provet.mobissa ei voi olla tilannetta, jossa luotaisiin jotain käyntiin liittyvä eikä sitä liitettäisi kyseiseen käyntiin. Tämä tulee myös vastaan, kun keskeneräistä käyntiä halutaan poistaa sovelluksesta. Käyntiä ei voida poistaa ennen kuin muut rivit tauluista, joissa siihen viitataan, on poistettu. Tällä varmistetaan se, että tietokantaan ei jää mitään haamurivejä, jotka eivät viittaa mihinkään ja joita ei käytetä missään.

3.3 Järjestelmäkuvaus

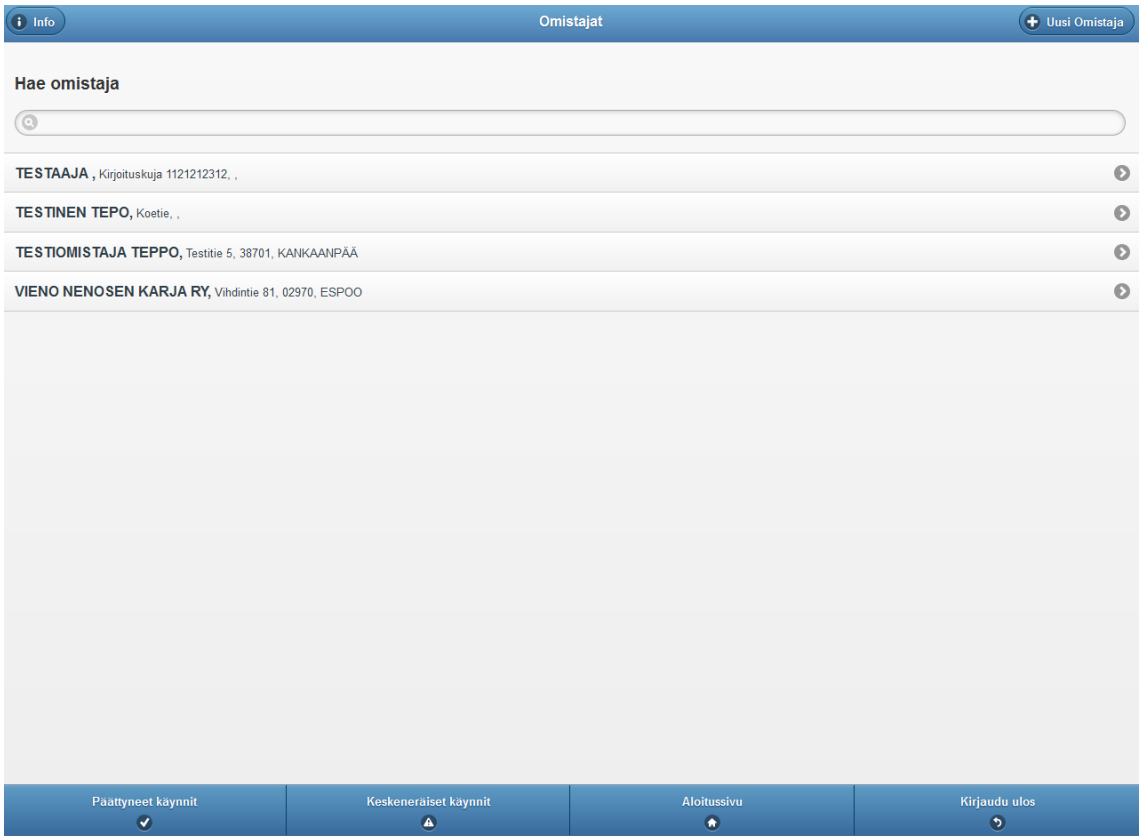
Hyvän mobiilisovelluksen perustana on yksinkertainen käyttöliittymä ja logiikan kulku sivulta toiselle. Kuvassa 6 esitän järjestelmän tilakaavion ja mitä tapahtuu missäkin vaiheessa.



Kuva 6. Järjestelmän tilakaavio

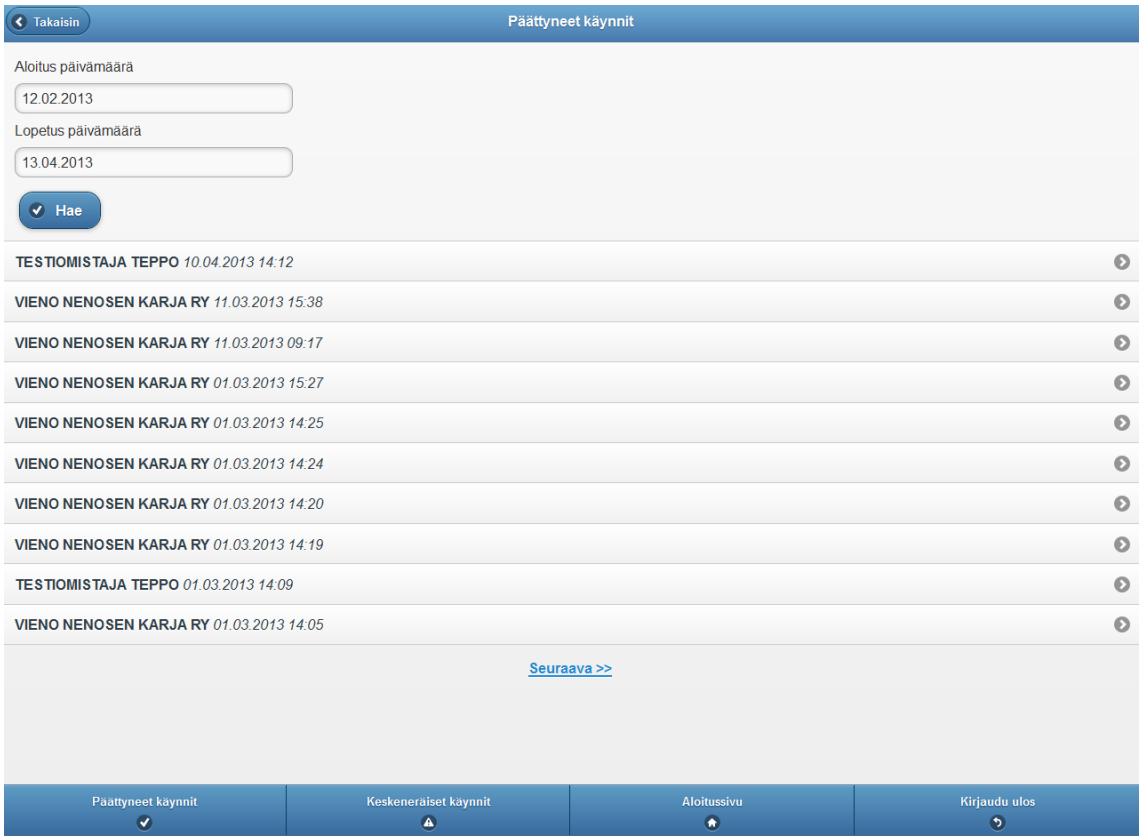
Kaikki alkaa käyttäjän autentikoinnista. Jos tämä epäonnistuu, käyttäjä ei pääse eteenpäin. Jos autentikointi onnistuu, taustalla ladataan Provet.mobin palvelimelle käyttäjän kaikki koodistot. Koodistot sisältävät xml-tiedostoja, jossa on hinnastoja lääkkeille ja toimenpiteille sekä muita asetustiedostoja. Käyttäjä voi muokata omia koodistojaan Provet Win 2 -ohjelmassa ja synkronoida ne sitten eteenpäin Provet.mobin käyttöön.

Kun käyttäjä on kirjautunut palveluun, hänet ohjataan ensimmäiseksi omistajalistaukseen. Tällä sivulla hän voi etsiä tilaomistajia ja tämän jälkeen avata omistajan kortin ja muokata tietoja. Omistajalistaussivulla on myös navigointinapit, joilla pääsee tutkimaan päättyneitä käyntejä sekä keskeneräisiä käyntejä. Kuvassa 7 on omistajalistausnäkymä.



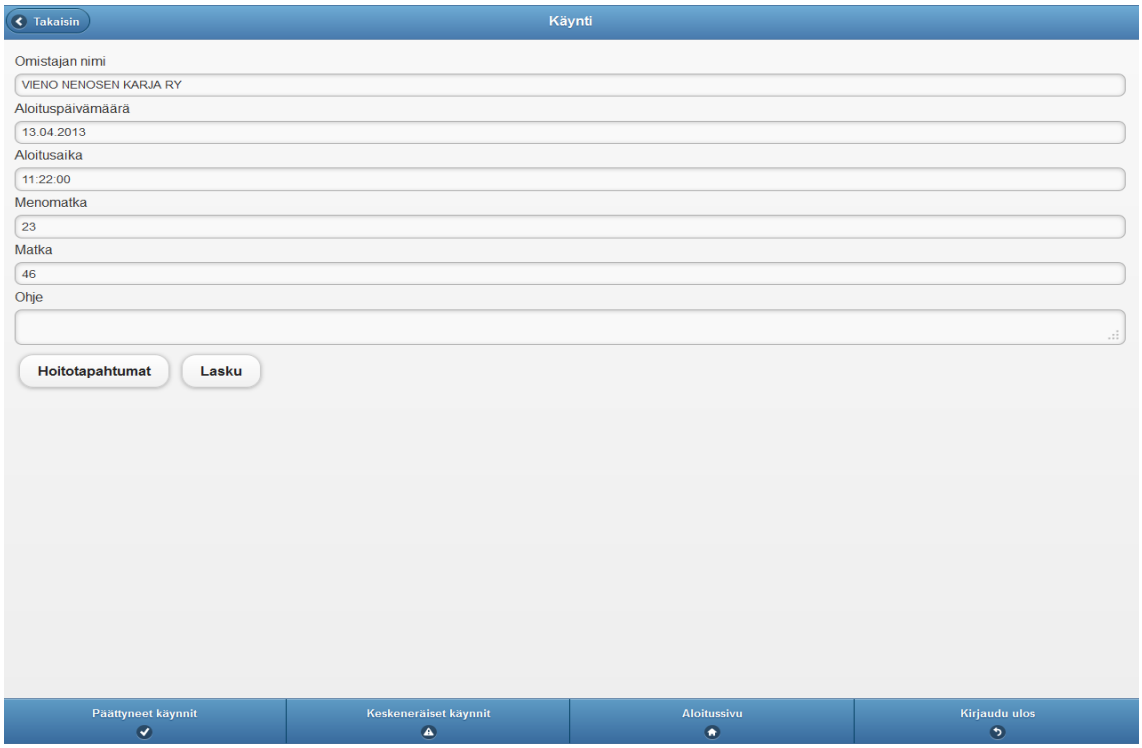
Kuva 7. Omistajalista näkymä

Päättyneet käynnit sisältävät kaikki käynnit, jotka eläinlääkäri on tehnyt joko Provet Win 2 -sovelluksella tai Provet.mobilla. Päättyneistä käynneistä voidaan tutkia käynnin perustietoja, jotka sisältävät tiedon siitä, kenelle käynti tehtiin, matkustuskilometrit sekä päivämäärän ja kellonajan. Kuvissa 8 ja 9 on esitetty päättyneiden käyntien listaus, josta voi valita käynnin tarkasteltavaksi ja tutkia käynnin perustiedot -sivua.



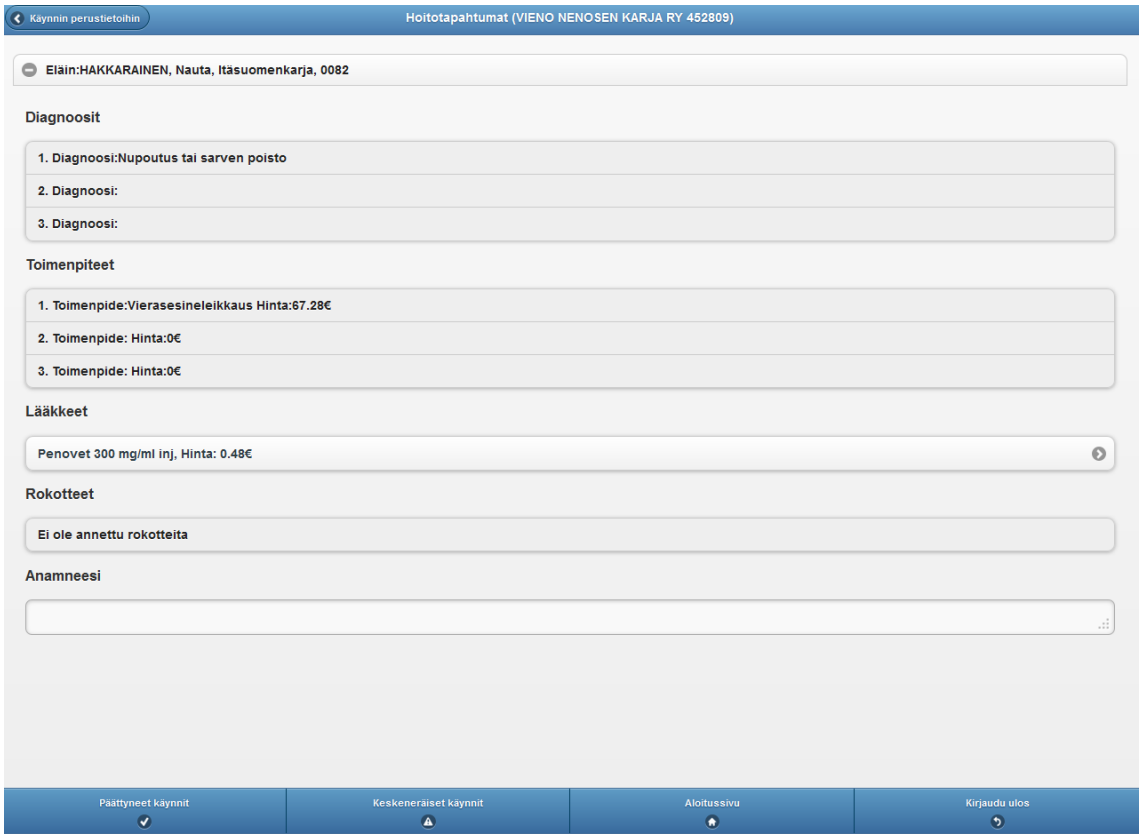
Kuva 8. Päättyneiden käyntien näkymä.

Kuvasta 8 nähdään, että käyntejä voidaan siis hakea tietyltä aikaväliltä. Niitä voidaan myös selata sivutuksella, jos käyntejä löytyy paljon haetulta aikaväliltä. Kun käynti, jota etsitään löytyy, niin kyseinen käynti avataan napsauttamalla sen riviä listanäkymästä. Kuvassa 9 on esitetty käynnin perustiedot. Kyseiseltä sivulta voidaan siirtyä tutkimaan joko käynnin hoitotapahtumia ”Hoitotapahtumat”-napista tai laskua ”Lasku”-napista.



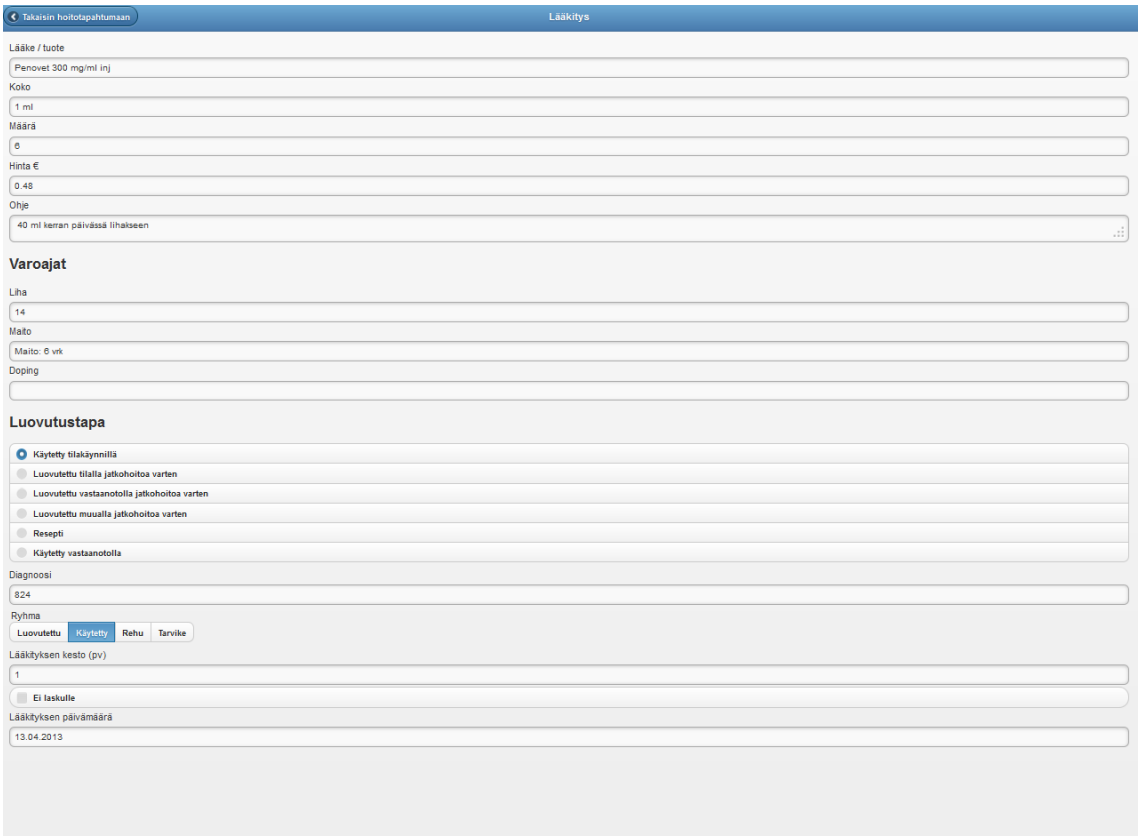
Kuva 9. Käynnin perustiedot -sivu.

Hoitotapahtumista voidaan vielä erikseen avata lääkkeiden tiedot, joissa on eritelty lääkkeen yksittäinen hinta, ohjeet, miten lääkettä on käytetty sekä luovutuspäivämäärä. Tämän lisäksi käynnistä voidaan tutkia laskua. Lasku-sivulla on eriteltynä käynnin eri hinnat, ja tästä voidaan myös tulostaa näytölle pdf-tiedosto. Tiedosto sisältää tilisiirto-lomakkeen, joten tämä tiedosto voidaan lähettää käyttäjälle sähköpostina tai postittaa meidän postituspalvelulla. Kuvissa 10 ja 11 on esitetty hoitotapahtuman sekä lääkkeen sivu.



Kuva 10. Päättyneen käynnin hoitotapahtuma-näkymä.

Kuvassa 10 on näkyvillä päättyneen käynnin hoitotapahtuma-sivu. Tällä sivulla on listattu mahdolliset kolme diagnoosia ja toimenpidettä sekä lääkkeet, joita eläimelle on annettu tai rokotukset. Eläimelle on voitu myös kirjoittaa oma anamneesi. Lääkkeitä sekä rokotuksia voidaan tutkia myös tarkemmin haluttaessa. Kuva 11 kuvaa lääkkeen sivua, jossa on tarkemmin määritetty lääkitys, joka määrättiin eläimelle hoitotapahtumassa.

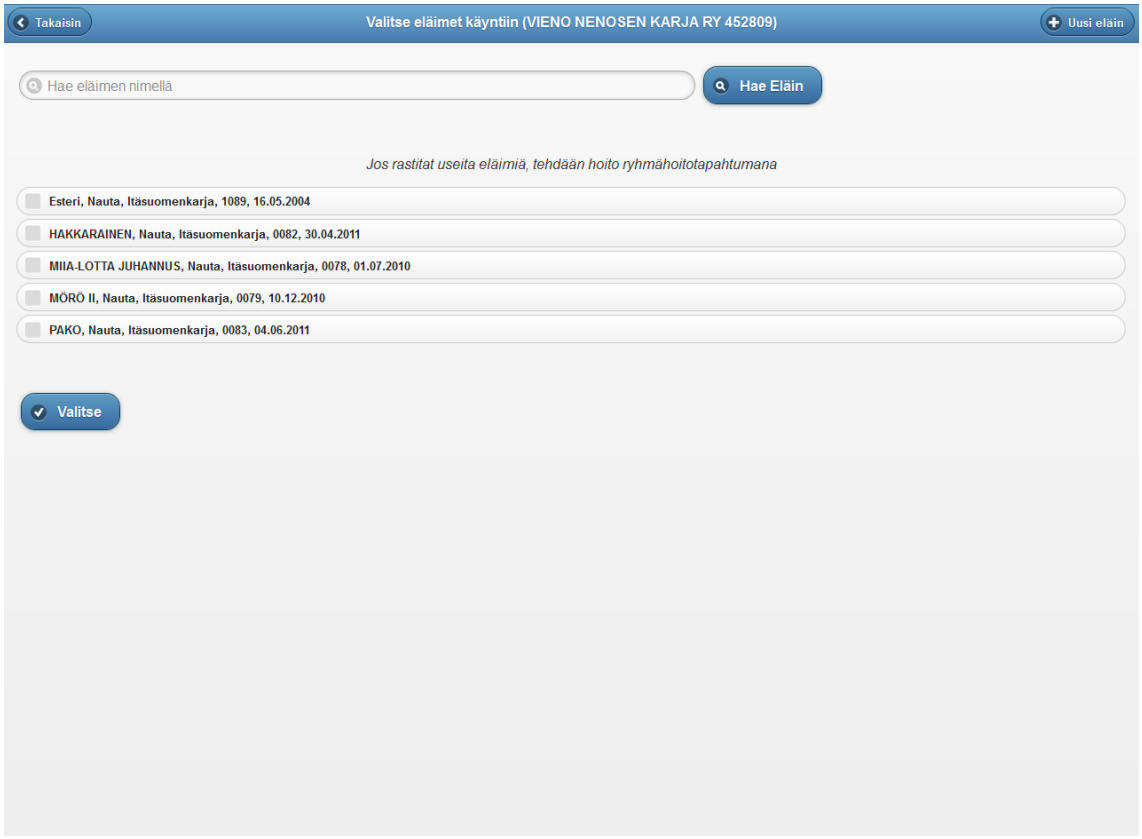


Kuva 11. Päättyneen käynnin lääkkeen katselu.

Käyttäjä voi myös omistajalistaus-sivulta valita keskeneräisten käyntien tutkimisen. Nämä ovat käyntejä, jotka on aloitettu Provet.mobi-palvelussa, mutta niitä ei ole vielä päätetty ja synkronoitu eteenpäin. Kun käynti päätetään ja synkronoidaan, niin se muuttuu päättyneeksi käynniksi palvelussa. Keskeneräistä käyntiä voidaan jatkaa siitä tilasta, mihin se on jäänyt. Siihen voidaan lisätä hoitotapahtumia tai muokata aiempia hoitotapahtumia. Laskun sivulla voidaan eriteltyjä hintoja vielä muokata käsin, jos niin tarvitsee tehdä. Laskun sivulla voidaan myös avata pdf-lasku esikatseltavaksi. Pdf-tiedosto sisältää myös lääkkeiden erittelysivun, jos käynnillä hoitotapahtumiin on eläimille annettu lääkkeitä ja valittu, että ne lääkkeet halutaan näyttää laskulla. Lääkkeiden erittelysivulla on eläimen nimi, rotu, lääke, joka annettiin, sen hinta ja varoajat.

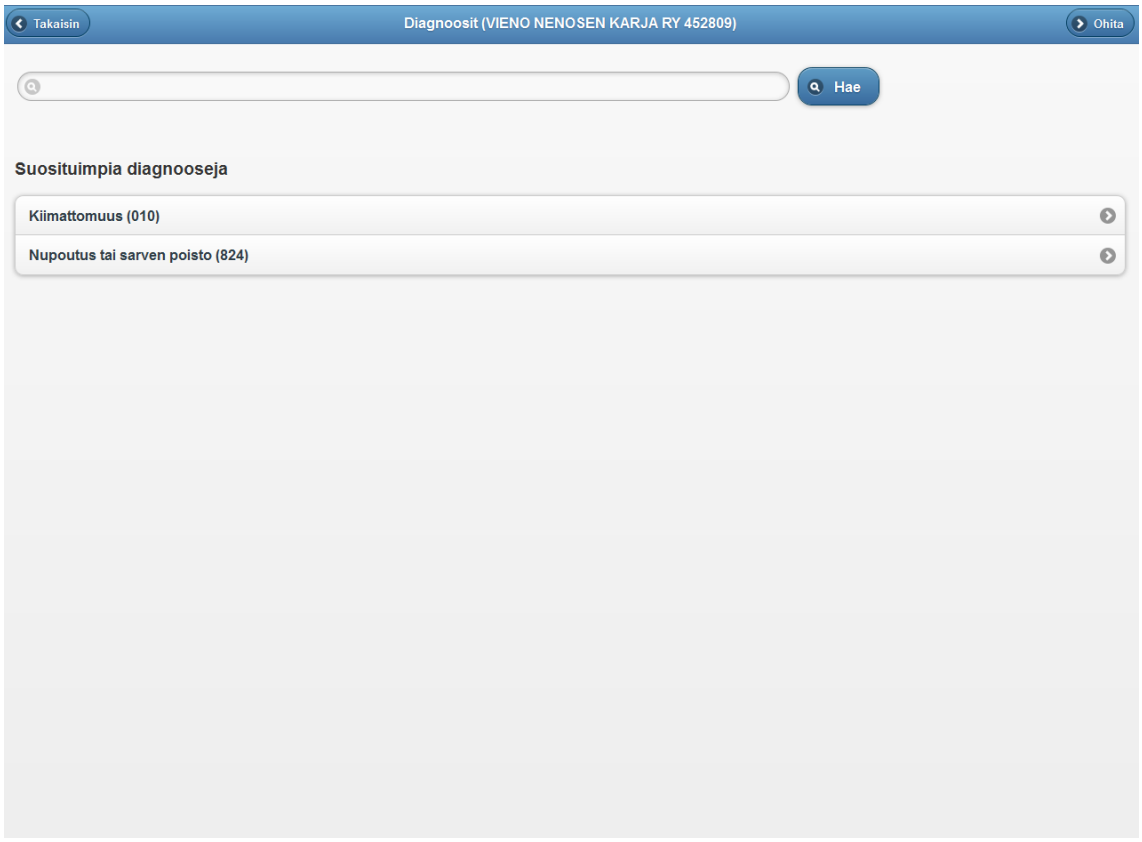
Kun omistajalistaus sivulta valitaan tilaomistaja, niin tämän kortin avauduttua voidaan aloittaa uusi käynti. Kun uusi käynti aloitetaan, aluksi täytetään käynnin perustiedot, eli tilaomistajan nimi, päivämäärä ja kellonaika, matkustuskilometrit sekä tieto siitä, onko kyseessä hyötyeläintila vai ei. Tämän jälkeen voidaan joko suoraan siirtyä käynnin laskulle tai valita hoitotapahtumien lisäys käyntiin. Jos käynnille valitaan hoitotapahtumia, täytyy ensiksi valita, mille eläimelle hoitotapahtumat lisätään. Kun eläin on valittu, niin

hoitotapahtuman lisäys voidaan aloittaa. Aluksi valitaan diagnoosi ja toimenpide eläimelle. Tämän jälkeen voidaan lisätä lääke sekä rokotus. Kaikkia näistä tiedoista ei ole pakko antaa, vaan eläinlääkäri voi ohittaa esimerkiksi lääkkeiden valinnan, jos eläimelle ei lääkkeitä tule. Tämän jälkeen hoitotapahtuma on valmis ja sitä voidaan tutkia hoitotapahtumat sivulla. Tällä sivulla voidaan vielä muokata hoitotapahtumia. Niihin voidaan lisätä lääkkeitä, rokotuksia, toimenpiteitä tai diagnooseja tai vastaavasti poistaa niitä. Tällä sivulla voidaan myös poistaa kokonaisia hoitotapahtumia käynniltä. Kuvissa 12 ja 13 on esitetty eläimen valinta hoitotapahtumaan sekä diagnoosin ja toimenpiteen valinta.



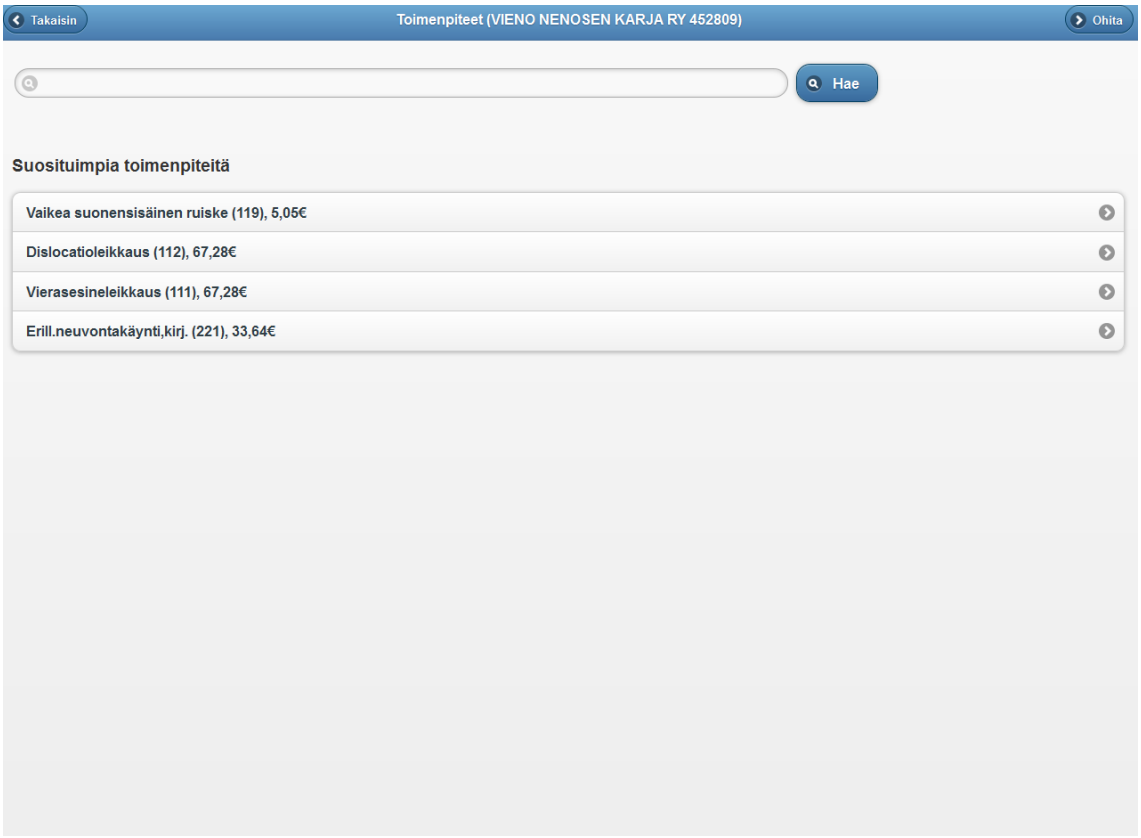
Kuva 12. Eläimen valintasivu

Kuvassa 12 on eläimen valinta hoitotapahtumalle. Jos eläimiä olisi useita kymmeniä, niin tuloksia voitaisiin selata sivutuksella. Jos lista on kovin suuri, voidaan eläintä myös hakea nimellä. Jos tältä sivulta valitsee useampia eläimiä, niin silloin aloitetaan ryhmähoito valituille eläimille, jolloin kaikille annetaan samat diagnoosit, toimenpiteet, lääkkeet sekä rokotukset.



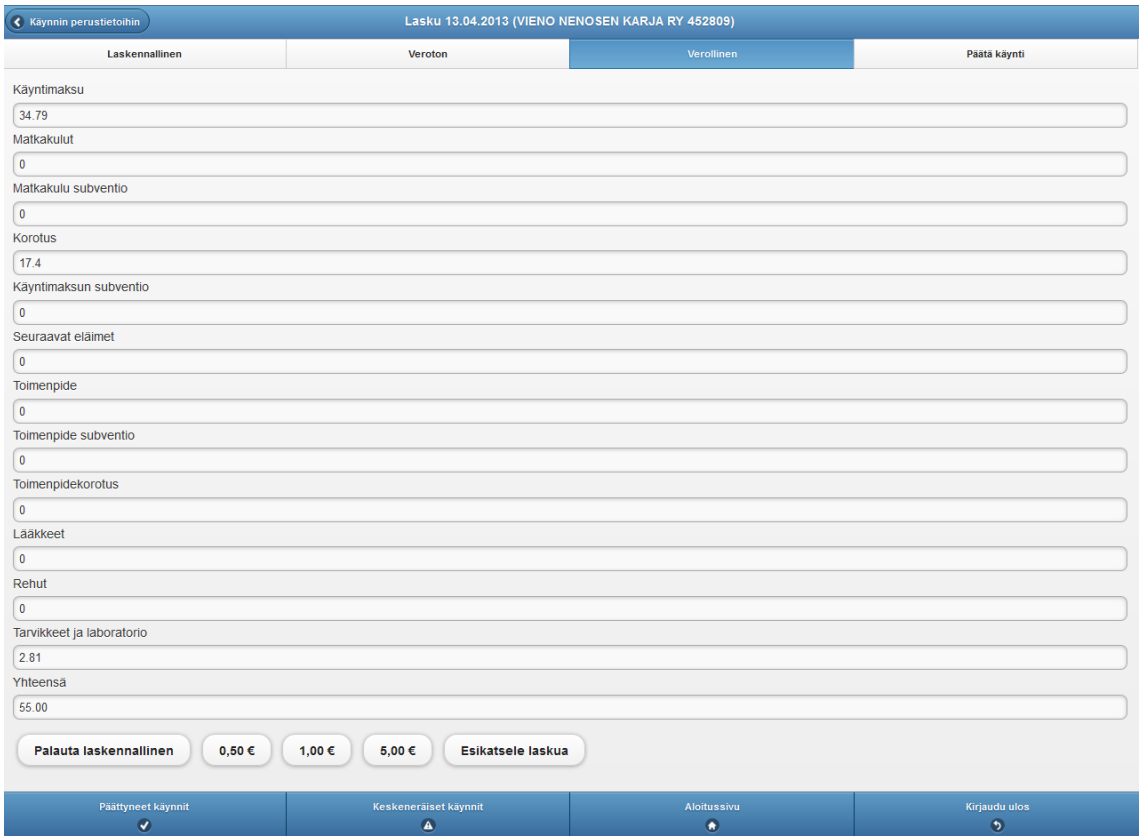
Kuva 13. Diagnoosin valinta hoitotapahtumalle

Kuva 13 sekä 14 esittävät diagnoosin sekä toimenpiteen valintaa hoitotapahtuman eläimelle tai eläimille. Listassa on heti näkyvillä eläinlääkärin suosituimmat diagnoosit / toimenpiteet kyseiselle lajille tai lajeille viimeiseltä kolmelta kuukaudelta. Jos sopivaa diagnoosia tai toimenpidettä ei löydy suosituimmista listoista, voidaan sitä hakea hakupalkista joko nimen tai sen koodin perusteella. Nämä sivut voidaan myös ohittaa oikealla ylänurkassa olevalla ohita-painikkeella.



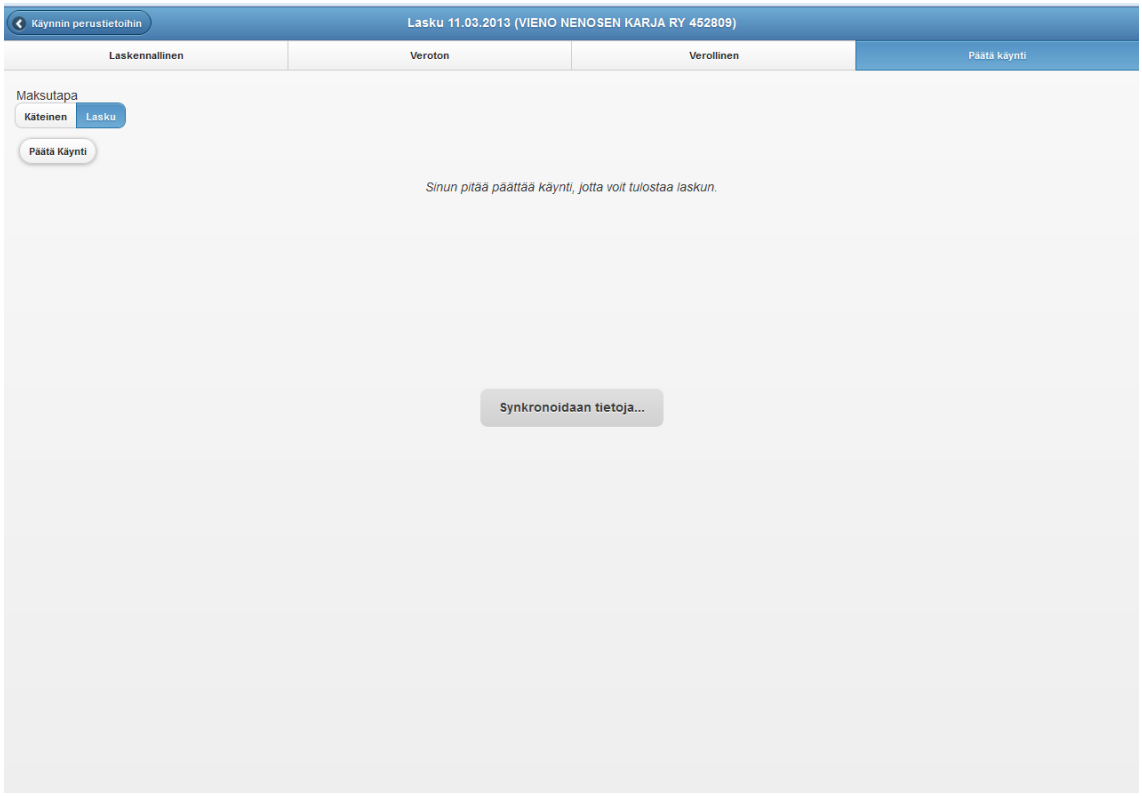
Kuva 14. Toimenpiteen valinta hoitotapahtumalle.

Kun käynnille on tehty haluttu määrä hoitotapahtumia, täytyy käyttäjän päättää ja synkronoida käynti, jotta se olisi valmis. Tämä onnistuu käynnin lasku -sivulta. Tällä sivulla on eriteltynä käynnin eri hinnat verottomina sekä verollisina. Hintoja voidaan käsin vielä muokata tai loppusummaa voidaan pyöristää lähimpään viiteenkymmeneen senttiin, euroon tai viiteen euroon. Tämä on suosittu toimintatapa eläinlääkäreillä; siksi se toiminto toteutettiin Provet.mobiin. Tällä sivulla voidaan myös avata näytölle esikatseltavaksi pdf-tiedosto laskusta. Kun hinnat ovat kohdillaan, voi eläinlääkäri vielä muuttaa laskun maksutapaa joko laskuksi tai käteiseksi. Tämän jälkeen käynti voidaan päättää ja synkronoida. Kuvissa 15 ja 16 on esitetty käynnin lasku-sivu sekä miten käynti synkronoidaan.



Kuva 15. Käynnin lasku -sivu, tästä voidaan vielä muokata laskun summia.

Kuvassa 15 ollaan verollisen laskutietojen muokkauskentissä. Tästä voidaan joko käsin syöttää summia kenttiin, jolloin verottomat hinnat myös päivittyvät kuten myös laskun loppusumma. Sivulla voidaan myös käyttää pyöristyspainikkeita, jolloin loppusuma aina pyöristetään lähimpään puoleen euroon, yhteen euroon tai viiteen euroon. Kun eläinlääkäri on tyytyväinen laskun summiin, pitää käynti päättää. Silloin siirrytään ”Päätä käynti”-välilehdelle kuten kuva 16 esittää. Tästä voidaan vielä vaihtaa laskun maksutapaa joko käteiseksi tai laskuksi. Kun maksutapa on valittu, painetaan ”Päätä käynti” -painiketta, ja sovellus aloittaa käynnin synkronoinnin keskuspalvelimelle.



Kuva 16. Käynnin synkronointi keskuspalvelimelle.

Näin luotiin uusi käynti Provet.mobi-sovelluksella ja synkronoitiin se. Uusi käynti löytyy nyt myös Provet Win 2 -ohjelmasta, kun se avataan ensikerran yhdistettynä verkkoon. Se yrittää automaattisesti ladata uudet käynnit Provet.mobi-palvelusta. Jos käyttäjä muokkaa päättyneitä käyntejä Provet Win 2 -ohjelmassa ja synkronoi tiedot, niin muutokset näkyvät myös Provet.mobissa.

4 Projektin kuvaus

Tässä luvussa esittelen projektin etenemistä eri vaiheissa. Käyn läpi työn aloituksen ja siihen liittyvät toimenpiteet, kehitystyön, järjestelmän testauksen sekä tuotantoon siirtymisen ja mitä se tarkoittaa Provet.mobin nykytilanteen näkökulmasta. Tähän projektiin kehitystyötä on omalta osaltani mennyt aikaa täysiä työpäiviä kolme ja puoli kuukautta.

4.1 Aloitus

Tämä projekti aloitettiin kesällä 2012 kesäkuun lopussa. Yrityksessämme oli jo palvelu nimeltään Provet Win 2, joka oli tietokonesovellus eläinlääkäreille. Halusimme tuoda samat palvelut, kuin Provet Win 2 tarjosi, mutta mobiililaitteille. Olimme kuitenkin huolissamme, siitä millä tekniikoilla tämä toteutettaisiin. Pienen tutkimustyön jälkeen päädyimme JQuery Mobilen käyttöön, koska se tarjosi niin suuren yhteensopivuuden eri laitteilla.

Kesäkuussa juhannuksen jälkeen varsinainen työ aloitettiin. Aluksi suunniteltiin, mitä eri tekniikoita tässä työssä yhdistyy. Oli selvää, että tarvitsimme rajapinnan näiden kahden palvelun väliin, ja Finnish Net Solutions (FNS) oli aiemmin jo käyttänyt muissa projekteissaan SOAP:ia. Tämän jälkeen piti valita, millä ohjelmointikielellä Provet.mobia lähdetään toteuttamaan. Koska päätimme toteuttaa Provet.mobin verkkosovelluksena, niin ohjelmointikieleksi tuli luonnollisesti PHP ja sovelluskehykseksi CakePHP, koska tämä on käytössä monessa eri projektissa FNS:ssä.

4.2 Kehitystyö

Varsinainen kehitystyö alkoi heinäkuun alussa vuonna 2012. Aluksi valittiin sopivat kehitystyökalut, joilla töitä tehtäisiin. Kehitystyökaluksi valittiin Eclipse-kehitin ja versionhallinta meillä oli jo valmiina työpaikalla. Se oli Subversion.

Ensimmäisenä työn alla oli selvittää ja suunnitella Provet.mobin oma tietokanta, johon keskeneräiset käynnit tallennetaan. Tietokannaksi valittiin alussa MongoDB, mutta hyvin nopeasti huomattiin, että sen opettelu ja käyttöönotto olisi vienyt aikaa. MongoDB on avoimen lähdekoodin projekti. Se eroaa perinteisestä tietokannasta siten, että tietoa ei tallenneta tauluihin vaan JSON-tyyppisiin dokumentteihin. [9.] Tämän takia päädyimme hyvin nopeasti MySQL-tietokantaan, eli perinteiseen taulurakenteeseen. Yksi syy tähän päätökseen oli myös se, että keskuspalvelimella oleva data on MySQL-tietokannassa.

Seuraavana piti saada rajapinta valmiiksi Provet.mobin ja keskuspalvelimen välille, koska siellä sijaitsi kaikki data. Vaikka Provet.mobin käyttöliittymän rakentelu ja suunnittelu oli jo aloitettu, niin kaikkia rajapintametodeja ei ollut vielä toteutettu ja niitä tehtiin

samaan aikaan Provet.mobin kehitystyön kanssa. Ne onneksi valmistuivat samaan tahtiin kuin niitä tarvittiinkin.

4.3 Testaus

Testaaminen aloitettiin heti kun Provet.mobilla voitiin tehdä käyntejä. Ensimmäiset testit suoritettiin oman työryhmän sisällä. Meillä oli päiviä, jolloin kävimme läpi eri testitapauksia ja kirjasimme ylös, mitä puutteita tai korjausehdotuksia tuli. Korjaukset toteutettiin ja testitapaukset suoritettiin uudestaan, kunnes niissä ei ilmennyt ongelmia. Tämä testaaminen oli pakollista ja tärkeää, koska oli erityisen tärkeää, että tiedot siirtyivät oikein keskuspalvelimelle. Oli myös tärkeää, että Provet Win 2 sai tiedot käyttöönsä ilman, että se rikkoi siellä jo olemassa olevaa dataa.

Kun Provet.mobi oli saatu siihen kuntoon, että sillä pystyi jo tuottamaan kokonaisia käyntejä ja tutkimaan päättyneitä käyntejä, niin silloin aloitettiin yrityksen sisäinen testaaminen. Ensimmäistä kertaa Provet.mobi oli esillä yrityksen sisällä syyskuussa, meidän omassa akatemiapäivässämme, jossa esiteltiin toisillemme, mitä projekteja on ollut käynnissä kesän sekä alkusyksyn aikana. Samana päivänä jokainen sai omalla ajallaan testata Provet.mobia ja tehdä käyntejä. Tästä päivästä saimme ensimmäiset testitulokset, joiden pohjalta jatkoimme kehitystä.

Toisen kerran Provet.mobi oli esillä eläinlääkäripäivillä, jotka järjestettiin marraskuun lopulla Helsingin messukeskuksessa. Meidän yrityksellä oli siellä oma piste, jossa oli esillä meidän Provet-tuotteitamme ja -palveluitamme. Näillä messuilla eläinlääkärit saivat tulla testaamaan meidän testilaitteillamme (Ipad, Ipad Mini, Samsung Galaxy Tab 2) Provet.mobin käyttöä. Eläinlääkärit olivat erittäin kiinnostuneita uudesta tuotteesta, joka oli tulossa.

Messun jälkeen valitsimme kolme eläinlääkäriä, jotka olivat olleet eniten kiinnostuneita Provet.mobista ja pyysimme heitä toimimaan betatestaajinamme. He saivat meiltä testilaitteet omaan käyttöönsä (Ipad Mini, Iphone 5 sekä Samsung Galaxy Tab 2), joilla he pystyivät käyttämään Provet.mobia. Betatesti alkoi joulukuun alkupuolella, eli melkein heti eläinlääkäripäivien jälkeen ja loppui maaliskuun alussa. Betatestin jälkeen nämä betatestaajat saivat lunastaa testilaitteet meiltä edulliseen hintaan.

Nämä betatestaajat olivat erittäin tärkeitä meille, koska vaikka kuinka me itse yritämme testata järjestelmää, niin siellä on kuitenkin aina asioita, joita me emme ymmärrä huomioida. Eläinlääkärit myös saattavat käyttää sovellusta erilailla, joten saamme heiltä kehitysideoita, miten asiat voitaisiin toteuttaa eri tavalla.

4.4 Tuotanto

Tuotantoon siirtyminen tapahtui noin helmikuun alussa. Meillä oli vielä kolme betatestaajaa, mutta halusimme saada tuotteen jo tuotantoon. Niinpä avasimme tuotantopalvelimen, jonne Provet.mobi siirrettiin ja lähetimme sähköpostitse sekä kirjeitse eläinlääkäreille ilmoituksen uudesta tuotteesta.

Provet.mobin käyttöönotto tapahtui siten, että eläinlääkärit lähettivät meille sähköpostitse viestin, jossa he ilmoittivat halukkuutensa Provet.mobi-palvelulle ja me tarkistimme, että heidän Provet Win 2 sovelluksensa oli ajan tasalla ja valmis Provet.mobilisäosaa varten. Jos kaikki oli kunnossa, lähetimme heille tunnukset, joilla he pääsivät kirjautumaan Provet.mobiin ja aloittamaan sen käytön.

Tällä hetkellä Provet.mobi on ollut tuotannossa jo pari kuukautta ja kehitystyötä tehdään koko ajan lisää. Saamme palautetta eläinlääkäreiltä, jonka pohjalta tiedämme, mihin suuntaan palvelua tulee viedä ja mitä pitää kehittää.

5 Tekniikoiden arviointia

Käyn läpi tässä kappaleessa eri tekniikoiden vahvuuksia sekä heikkouksia. Tässä työssä käytetyt tekniikat olivat hyvä valinta tähän kyseiseen projektiin, koska ne vastasivat tarpeitamme.

CakePHP:n vahvuudet ovat selvästi sen helpon käyttöönoton sekä MVC-arkkitehtuurin käytössä. Kyseisellä sovelluskehyksellä on helppo luoda uusia näkymiä ja hakea niihin dataa tietokannasta, hallita sessiomuuttujia sekä käsitellä suuria määriä dataa lomakkeissa. Kuten aiemmista käyttöliittymäkuvista huomattiin, niin Provet.mobissa on paljon lomakkeita käytössä käyttöliittymissä. CakePHP:lla on helppo luoda lomakkeita ja välittää niihin dataa esitäytettynä.

JQuery Mobile -kehyksen vahvuudet ovat sen laajassa yhteensopivuudessa eri laitteissa. Toinen suuri vahvuus tässä kehyksessä ovat sen käyttöliittymäkomponentit. JQuery Mobilessa tulee mukana suuri määrä eri lomakekomponentteja, listakomponentteja sekä eri teemoja, jotka on suunniteltu mobiililaitteille ja sormella paineltaviksi. Tämä mahdollistaa nopean kehitystyön sekä datan näyttämisen järkevästi käyttäjälle.

SOAP-rajapinnan käytön vahvuutena oli datan välitys palvelimelta toiselle sekä sen välittäminen Provet Win 2 -sovellukselle, joka toimii eri alustalla. Ilman tämänkaltaista tekniikkaa olisi tämän kaltaisen palvelun tuottaminen ollut vaikeaa, koska emme olisi saaneet molemmista palveluista dataa keskuspalvelimelle ja keskuspalvelimelta dataa palveluihin. Sen käyttö on myös hyvin helppoa Provet.mobin päässä, koska rajapintakutsut ovat kuin normaalit metodikutsut.

Näissä tekniikoissa oli myös heikkouksia. CakePHP:n sekä JQuery Mobilen yhteensovittaminen on hieman vaikeaa siinä mielessä, että CakePHP:ssa luotaessa uusi näkymä, niin se on aina oma sivunsa omalla URL-osoitteellaan. Kun taas JQuery Mobile mahdollistaisi sen, että koko sovellus olisi yhdessä HTML-tiedostossa. Tämä saatiin kuitenkin ratkaistua sillä, että sivut ladataan normaalisti, mutta JQuery Mobile saa sen näyttämään siltä kuin sivua ei ladattaisi, vaan siihen siirrytään, kuten mobiililaitteiden natiivisovelluksilla on tapana.

6 Yhteenveto

Tässä projektissa kehitettiin Finnish Net Solutions Oy:n Provet Win 2 -sovellukselle mobiiliversio, jota eläinlääkärit voivat hyödyntää. Kyseinen projekti aloitettiin kesällä 2012 ja se jatkui vuoden loppuun asti. Alkuvuoden se on ollut betatestauksessa ja helmikuusta lähtien tuotannossa.

Projektissa hyödynnettiin CakePHP-sovelluskehystä, JQuery Mobile -kehystä sekä SOAP-rajapintaa. Näillä tekniikoilla saatiin toteutettua varsin monipuolinen mobiilisovellus.

Mikään ohjelma ei ole koskaan valmis, eikä Provet.mobikaan ole sitä vielä. Saamme jatkuvasti eläinlääkäreiltä kehitysideoita, joita tulemme toteuttamaan ajan kanssa. Tällä hetkellä suurimpia kehitystarpeita Provet.mobissa on tilaomistajien tuonti Pro-

vet.mobiin, jotka on luotu Provet Win 2 sovelluksessa, mutta joille ei ole luotu käyntejä, ja jotka eivät ole tämän johdosta siirtyneet keskuspalvelimelle. Toinen suuri kehitystarve on Provet Win 2 -sovelluksen päiväkirjatoiminnon toteutus Provet.mobin ja Provet Win 2:n välille.

Provet.mobista tuli selkeä ja käyttökelpoinen työkalu Provet Win 2 -ohjelman rinnalle. Se mahdollistaa tilakäynnin hoitotapahtumien sekä laskutuksen kirjaamisen mobiililaitteilla. Täten eläinlääkäreiden ei tarvitse kantaa mukanaan eläinlääkärisalkkua, jossa oli kannettava tietokone mukana.

Lähteet

1. Provet Win -kunnaneläinlääkärin työtoveri. Verkkodokumentti. http://www.provet.fi/win. Luettu 11.3.2013.

2. JQuery Mobile. Verkkodokumentti. http://en.wikipedia.org/wiki/JQuery_Mobile. Luettu 11.3.2013.

3. Chan Kai, Omokore John, K. M. Richard Practical CakePHP Projects. 2009. Berkeley, CA : Apress.

4. WSDL Document Example. Verkkodokumentti. http://www.tutorialspoint.com/wsdl/wsdl_example.htm. Luettu 11.3.2013.

5. Web Services – WSDL: Creating SOAP Server. Verkkodokumentti. http://www.phpeveryday.com/articles/Web-Services-WSDL-Creating-SOAP-Server-P484.html. Luettu 11.3.2013.

6. Web Services – WSDL: Testing with SOAP Client. Verkkodokumentti. http://www.phpeveryday.com/articles/Web-Services-WSDL-Creating-SOAP-Server-P485.html. Luettu 11.3.2013.

7. Introduction to Web Services Technologies: SOA, SOAP, WSDL and UDDI. 2004. Verkkodokumentti. http://www.informit.com/articles/article.aspx?p=336265&seqNum=4. Luettu 21.4.2013.

8. Hajautetut järjestelmät. Verkkodokumentti. http://fi.wikipedia.org/wiki/Hajautetut_j%C3%A4rjestelm%C3%A4t. Luettu 12.4.2013.

9. MongoDB. Verkkodokumentti. http://en.wikipedia.org/wiki/MongoDB. Luettu 11.4.2013.