VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Mikko Ranta

Verkkoyhteydestä riippumaton web-sovellus

Case: Huoltovarma

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma

TIIVISTELMÄ

Tekijä	Mikko Ranta
Opinnäytetyön nimi	Verkkoyhteydestä riippumaton web-sovellus Case: Huoltovarma
Vuosi	2012
Kieli	suomi
Sivumäärä	36
Ohjaaja	Mika Tamminen

Opinnäytetyön tarkoituksena oli löytää ratkaisu siihen, miten tietokantaa käyttävä web-ohjelma saadaan toimimaan tehokkaasti myös ilman yhteyttä palvelimeen käyttämällä selainta väliaikaiseen tallennukseen.

Raportin alussa esitellään, mitä ohjelmointikieliä, tekniikoita ja työkaluja on käytetty sovelluksen toteuttamiseen. Sovellus esitellään aluksi yleisesti ja sen jälkeen käydään sen toimintaa läpi syvällisemmin. Koska itse raporttisovellus jäi kehitysvaiheeseen, raportissa on oma osio jatkokehitysehdotuksille.

Sovelluksen yhteydettömän tilan ratkaisun toteuttaminen oli haasteellinen, koska kyseisestä tekniikasta ei ole vielä syvällisempää tietoa.

Avainsanat Ajax, PHP, web-sovellus, verkkoyhteys, yhteydetön

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietojenkäsittelyn koulutusohjelma

ABSTRACT

Author	Mikko Ranta
Title	Offline Web Application
Year	2012
Language	Finnish
Pages	36
Name of Supervisor	Mika Tamminen

The goal of this thesis was to find and create a solution to make an database driven web-application work offline without a connection to the server.

The used languages, techniques and tools are introduced at the beginning of the thesis. The application itself is presented in two parts. The first part tells about the application in a more general manner and the second one dives deeper in how the application is built. Because the application is still in the development stage, there is a chapter telling about further development in the thesis.

Because there was not much information about this kind of a solution available, it was very challenging to create.

Keywords Ajax, PHP, Web Application, Offline

SISÄLLYS

KUVIO- JA TAULUKKOLUETTELO

LISTAUSLUETTELO

1 JOHDANTO

Opinnäytetyön alkuperäinen tavoite oli tehdä kiinteistöhuoltoyritykselle verkossa toimiva raporttisovellus, jolla voidaan kirjata muistiin esimerkiksi kiinteistön kunto- ja korjausehdotukset. Koska raporttisovellusta käytettäisiin suurimmaksi osaksi taulutietokoneilla mobiiliverkkoja hyödyntäen (ei aina saatavilla), sovelluksen täytyisi toimia myös yhteydettömässä tilassa.

Työn edetessä tuli kuitenkin ilmi, että yhteydettömässä tilassa toimivan web-sovelluksen rakentaminen ei ole helppoa, eikä aiheesta löytynyt työn tekemisen aikana vielä paljoakaan tietoa. Tästä syystä työ tulikin painottumaan enemmän yhteydettömän tilan ratkaisun tekemiseen. Itse raporttisovellus jäi kehitysvaiheeseen, koska työmäärä oli liikaa yhdelle ihmiselle.

Esittelen tässä raportissa kehitysvaiheeseen jääneen sovelluksen ja siihen rakennetun yhteydettömän tilan ratkaisuni sekä millä tekniikoilla se on rakennettu. Raportin lopussa ehdotuksia jatkokehityksestä.

Työn toimeksiantajana on Huoltovarma. Huoltovarma (Koko Scandinavia Oy) on vuonna 2011 perustettu yritys, jonka päätoimiala on kiinteistöhuolto.

2 TOTEUTUSTEKNIIKAT JA ASIASANASTO

Osio esittelee lyhyesti opinnäytetyössä käytettyjä työkaluja, ohjelmointi- ja merkkauskieliä, sekä avaa tarvittavien termien tarkoituksen.

2.1 JavaScript

JavaScript on Netscapen kehittämä oliopohjainen komentosarjakieli, jota käytetään suurimmaksi osaksi tuomaan dynaamisuutta web-sivustoille. Kielen suosio on kasvussa myös palvelinpuolella. JavaScript pohjautuu ECMAScript-standardiin.

Kieli julkaistiin ensimäisen kerran vuonna 1995 nimellä "LiveScript", mutta se vaihdettin myöhemmin JavaScriptiksi. Nimestään huolimatta kielellä ei ole juurikaan yhteyttä Javaan. (Mozilla 2012)

2.1.1 Ajax

Ajax on lyhennys termille "Asynchronous JavaScript + XML". Ajax ei itsessään ole uusi tekniikka, vaan monen eri tekniikan yhdistämistä. Termiä käytetään kuvaamaan asynkroonista ilman sivun uudelleenlatausta tapahtuvaa toimintaa.

Nimestään huolimatta se tukee myös muita tiedonsiirtotekniikoita (XML), kuten JSON, joka on nykypäivänä XML:ää suositumpi. (Mozilla 2012)

2.1.2 JSON

JSON (JavaScript Object Notation) on kevyt tiedonsiirtomuoto, joka nimensä mukaisesti perustuu JavaScript-objektiin. Yksinkertaisen ja tehokkaan muotonsa takia moni ohjelmointikieli osaa kääntää JSON:ia molempiin suuntiin. JSON on sen helppouden ansiosta noussut XML:ää suositummaksi tiedonsiirtomuodoksi. (JSON 2012)

2.1.3 jQuery

jQuery on JavaScript-ohjelmointikielelle tarkoittettu kirjasto. jQueryn olennaisin tarkoitus on helpottaa DOM:in (HTML Document Object Model) käsittelyä, tapahtumahallintaa, animointia sekä Ajaxin käyttöä. jQueryä käyttämällä ei myöskään tarvitse huolehtia selainten eroista, sillä jQueryn käyttö poistaa selainriippuvaiset ratkaisut.

Koska jQueryn omalaatuinen käyttötapa eroaa JavaScriptistä, niin se usein mainitaan ikäänkuin omana "kielenä". (jQuery 2012)

2.2 PHP

PHP (PHP: Hypertext Preprocessor) on oliopohjainen ohjelmointikieli, jota käytetää erityisesti web-palvelimilla. Verrattuna JavaScriptiin PHP toimii ainoastaan palvelinpuolella. PHP:n yleisin käyttötapa on liittää PHP-koodia HTML-merkkauksen sekaan, mutta erilaisia sovelluskehyksiä (esim. CakePHP) ja malleja (esim. MVC) käyttämällä voidaan nämä kaksi erottaa toisistaan, jolloin ylläpito helpottuu.

Vuonna 1995 PHP:n alkuperäinen kehittäjä Rasmus Lerdorf julkaisi ensimmäisen version. PHP:n kehityksestä ja ylläpidosta vastaa nykyisin The PHP Group. (PHP 2012)

2.2.1 CakePHP

CakePHP on ilmainen avoimen lähdekoodin web-sovelluksien luomiseen tarkoitettu ohjelmistokehys PHP:lle. CakePHP on luotu käyttämällä mallina Ruby on Rails -ohjelmistokehyksen konsepteja, mutta sen tarkoitus ei ole olla Railsin kopio. CakePHP:n pohjana on MVC (Model–view–controller) ohjelmistoarkkitehtuurityyli. (CakePHP 2012)

2.2.2 Web service

Web service on eräänlainen ohjelmointirajapinta kahden verkossa toimivan laitteen välillä. Kyseisellä termillä ei siis tässä tapauksessa tarkoiteta yleistä verkkopalvelua.

2.3 MySQL

MySQL on maailman eniten käytetty tehokas vapaan lähdekoodin tietokantaohjelmisto. MySQL käyttää nimensä mukaisesti SQL-kieltä (Structured Query Language) relaatiotietokantojen käsittelemiseen. Yleisin ohjelmistokokonaisuus, jota käytetään web-sovellusten kehityksessä, on "LAMP" eli Linux, Apache, MySQL ja PHP, mutta MySQL tukee myös monia muita kokonaisuuksia. (MySQL 2012)

2.4 Web storage

Web storage on eräänlainen tapa tallentaa tietoa web-selaimeen. Tieto tallennetaan assosiatiiviseen taulukkoon tekstimuodossa, eli tekstimuotoista avainta vastaa tekstimuotoinen arvo. Tallennustilaa on yleisimmissä selaimissa oletuksena 5–10 megabittiä, mutta moni selain antaa tallentaa lähes rajattomasti käyttäjän hyväksynnällä.

Tallennusmuotoja on kaksi, jotka ovat localStorage ja sessionStorage. Local storage on domain-kohtainen, eli tieto joka tallennetaan on saatavana kyseiselle domainille ja se säilyy, vaikka selain suljettaisiin. Session storage on ikkuna- tai välilehtikohtainen ja se tyhjenee, kun ikkuna tai välilehti suljetaan.

2.5 HTML5

HTML5 (HyperText Markup Language) on viides versio HTML-merkkauskielestä ja sen ensimäinen versio julkaistiin vuonna 2008. HTML5 sisältää uusia ominaisuuksia ja pyrkii selkeyttämään merkkauskieltä. Monet ominaisuudet on myös suunniteltu virrankulutusta silmälläpitäen, eli paremmin mobiililaitteille sopiviksi.

Joitain uusia ominaisuuksia ovat mm. tuki äänelle ja videolle, mahdollisuus käyttää prosessorin lisäsäikeitä, ohjelmoitava 2D-piirtoalusta ja paikallinen tallennus.

HTML5 ei ole vielä standardi.

2.6 Työkalut

Opinnäytetyössä käytettiin muutamia hyödyllisiä työkaluja helpottamaan koodin luontia, testausta, tietokannan käyttöä ja luomaan nopeasti pystytettävä kehityspalvelin.

2.6.1 Firebug

Firebug on ilmainen kehitystyökalu Firefox selaimelle, jolla voi mm. reaaliaikaisesti muokata, tarkkailla ja etsiä virheitä JavaScriptistä, CSS- ja HTML-merkkauksista sekä tarkkailla verkon aktiivisuutta.

Firebug toimii Firefoxissa lisäosana ja itse lisäosaan on myös saatavilla lisäosia. Muille suosituimmille selaimille on saatavilla Firebug lite -versio, joka on karsittu versio ohjelmasta. (Firebug 2012)

2.6.2 Aptana Studio

Aptana Studio on ilmainen ja tehokas avoimen lähdekoodin web-ohjelmien kehitystyökalu Windowsille, Linuxille ja Macille. Aptanaa voidaan käyttää joko itsenäisenä sovelluksena tai Eclipse-kehitysympäristön lisäosana. Aptana tukee monia eri kieliä kuten HTML5, CSS3, JavaScript, Ruby, Rails, PHP ja Python. (Aptana 2012)

2.6.3 XAMPP

XAMPP on alustariippumaton, ilmainen ja avoimen lähdekoodin ohjelmistopaketti ja se sisältää Apache-palvelinohjelmiston, johon on lisätty tuki MySQL-tietokantaohjelmistolle, PHP:lle sekä Perlille. XAMPP sisältää myös erinomaisen phpMyAdmin-sovelluksen, jolla on helppo tarkastella ja muokata tietokantaa.

XAMPP toimii asennettuna tai purettuna suoraan ilman konfigurointia ja sitä voidaan käyttää myös USB-tikulta. (Apachefriends 2012)

2.6.4 phpMyAdmin

phpMyAdmin on ilmainen selainpohjainen työkalu MySQL-tietokantaohjelmiston hallintaan ja se on nimensä mukaisesti luotu PHP-kielellä.

phpMyAdmin tukee monia ominaisuuksia kuten mm. tietokantojen, taulujen, kenttien, yhteyksien, käyttäjien ja lupien hallintaa. (phpMyAdmin 2012)

3 SOVELLUS

Tämän kappaleen tarkoituksena on antaa yleiskuva sovelluksesta.

Tekstissä käytetään usein sanoja "yhteydellinen tila" sekä "yhteydetön tila". Yhteydellisellä tilalla tarkoitetaan sitä, että verkkoyhteys palvelimeen on saatavilla ja yhteydetön tila tarkoittaa, että verkkoyhteyttä palvelimeen ei ole saatavilla.

3.1 Vaatimukset

Alla olevassa listassa lueteltuna sovelluksen keskeisimmät vaatimukset, jotka asetin itselleni. Listassa on siis vain sovelluksen vaatimuksia, jotka määrävät koko sovelluksen ydinrakennetta.

Kappale 5 antaa tietoa sovelluksen jatkokehityksestä. Yhdistämällä alla olevat vaatimukset kappaleen 5 kanssa saadaan parempi kuva siitä, mitä vaatimuksia valmis sovellus tulisi sisältämään.

1. Käyttäjän pitää kirjautua käyttääkseen ohjelmaa.
2. Käyttäjä voi hakea, luoda ja muokata asiakkaita.
3. Asiakkalla voi olla 0–n kappaletta kiinteistöjä, kiinteistössä voi olla 0–n kappaletta osioita, osioilla voi olla 0–n kappaletta ominaisuuksia ja ominaisuuksilla voi olla 0–n kappaletta kuvia.
4. Valitut asiakkaat latautuvat automaattisesti käytettäväksi myös yhteydettömään tilaan siltä varalta, että verkkoyhteys katkeaisi.
5. Asiakkaita voi ladata myös erillisesti käytettäväksi yhteydettömään tilaan.
6. Yhteydettömässä tilassa voi luoda ja muokata asiakkaita.
7. Yhteydettömässä tilassa olevien tietojen pitää säilyä, vaikka selain tai laite sammutettaisiin.

8. Ohjelma lähettää kaikki yhteydettömässä tilassa tehdyt päivitykset automaattiseti palvelimen päätietokantaan, kun verkkoyhteys on saatavilla.

3.2 Sovelluksen kehitystilanne

Sovellus on vielä kehitysvaiheessa, joten esimerkiksi käyttöliittymän käytettävyyteen ja tyyliin ei ole panostettu, vaan aika on käytetty ydintoimintojen, logiikan ja muiden korkeampien prioriteettien asioiden toteuttamiseen.

Sovelluksen hankalin ja suurimman osan ajasta vieneen yhteydettömän tilan rakenne, idea ja toteutus on valmiina.

Sovelluksen tietokannan rakenne ja taulut ovat kokonaisuudessa valmiina. Tauluihin ei ole lisätty vielä kehitysvaiheessa mitättömiä, mutta lopputuotteessa tärkeitä tietueita.

Käyttöliittymän toiminnot ulottuvat tällä hetkellä asiakkaisiin ja niiden rakennuksiin. Näille on siis toteutettu yhteydelliset "suorat" toiminnot sekä yhteydettömän tilan tarvittavat toiminnot.

Valmiina tai osittain valmiina ovat määritykset 2, 3 osittain, 4, 5, 6, 7 ja 8.

Jatkokehityksestä lisää kappaleessa 5.

3.3 Rakenne

Sovellus muodostuu kolmesta eri osasta, jotka ovat käyttöliittymä, web service -rajapinta ja tietokanta.

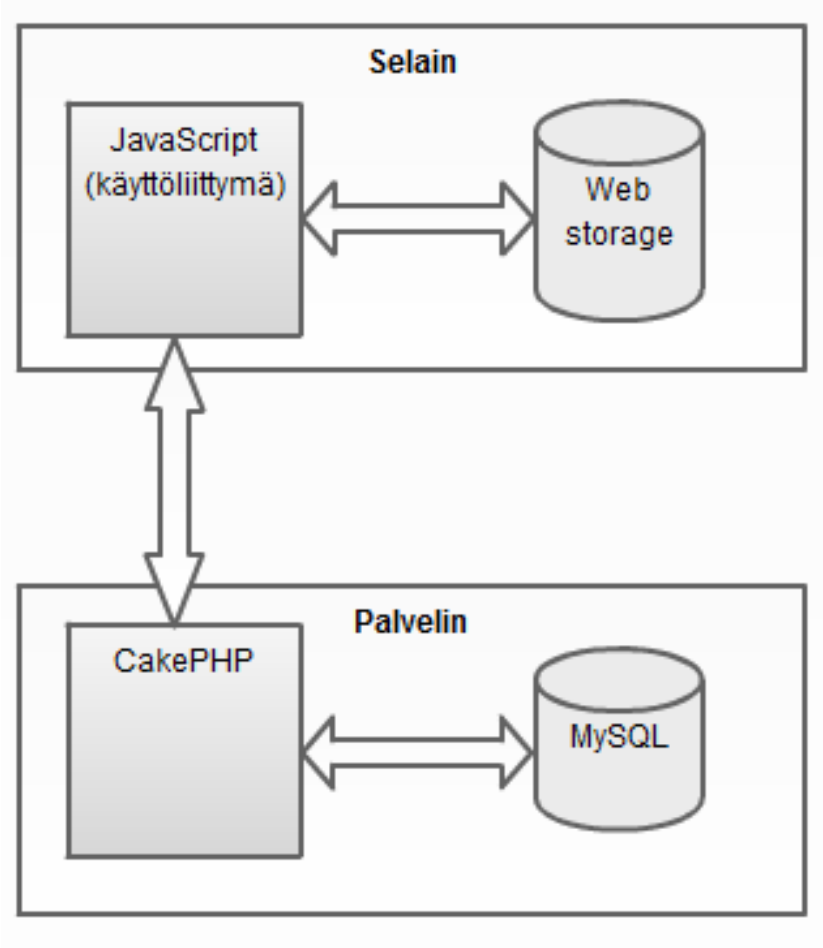
Käyttöliittymää ei voida rakentaa palvelinpuolen kielillä kuten PHP, koska ohjelman täytyy toimia myös yhteydettömässä tilassa. Tästä syystä käyttöliittymä rakennetaan kokonaan JavaScriptillä.

JavaScriptissä käytössä on jQuery-kirjasto, joka auttaa varsinkin tapahtumien käsittelyssä sekä poistaa huolet selainriippuvaisista ratkaisuista. jQueryn käyttö vä-

hentää huomattavasti koodin määrää tarjoamalla monimutkaisiakin ratkaisuja muutamalla rivillä koodia, jotka pelkällä JavaScriptillä veisivät kymmeniä rivejä koodia. jQueryyn on saatavilla myös todella paljon erilaisia lisäosia lisäämään sen toimintoja.

Sovellus tarvitsee web service -rajapinnan, koska JavaScript-pohjainen käyttöliittymä ei voi olla suoraan yhteydessä tietokantaan, kuten esimerkiksi PHP. Rajapinnan tarkoituksena on toimia käyttöliittymän ja tietokannan välissä ja hoitaa kaikki tietokannan muokkaukset. Rajapinta on rakennettu PHP-pohjaisella CakePHP-ohjelmistokehyksellä. Ohjelmistokehyksen tarkoituksena (tässä projektissa) on helpottaa rajapinnan rakentamista, tietokannan muokkaamista, tiedon validointia, tietoturvaa, käyttäjän todentamista sekä vähentämään huomattavasti koodauksen määrä.

Tietokantajärjestelmäksi on valittu MySQL, koska se on ilmainen, helposti saatavilla ja se kuuluu XAMPP-ohjelmistopakettiin, joka sisältää selaimessa käytettävän helppokäyttöisen phpMyAdmin-hallintaohjelmiston.



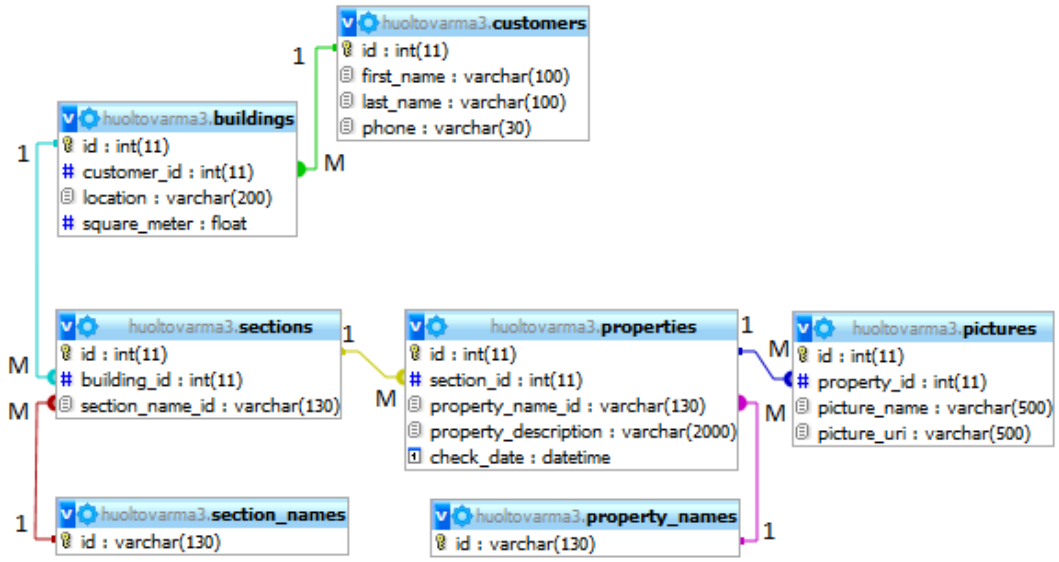
Kuvio 1. Rakenne.

3.4 Tietokanta

Tietokannan taulut ja tietueet ovat nimetty CakePHP:n nimeämistyylillä, jotta ne voidaan automaattisesti yhdistää oikein ilman konfiguraatiota. Taulut eivät sisällä vielä kaikkia tietueita, vaan ainoastaan koodin rakenteen kannalta tärkeät. Tietokanta on mahdollisimman pitkälle normalisoitu, jotta joustavuuden kanssa ei tulisi ongelmia.

"section_names" ja "property_names" -taulut eivät sisällä erillistä numeerista id:tä, koska nimet ovat aina uniikkeja. Tällä ratkaisulla voidaan vähentää myös web servicen ja käyttöliittymän koodin monimutkaisuutta.

Kuviossa 2 on tietokannan taulut ja relaatiot.



Kuvio 2. Tietokantakaavio.

"customers"-taulu sisältää asiakkaan henkilötiedot.

"buildings"-taulu sisältää rakennuksen tiedot.

"sections"-taulu sisältää rakennuksen osa-alueen, joka voi olla esimerkiksi huone tai putkisto. "section_names" sisältää osa-alueiden nimet. Itsenäisellä nimitaululla vähennetään tiedon toistumista, koska esimerkiksi "keittiö"-osio kuuluu varmasti moneen eri taloon.

"properties"-taulu sisältää osa-alueiden ominaisuudet, joita voi olla esimerkiksi vessan (osa-alue) seinä, lattia, kaapisto ym. Myös tällä taululla on tiedon toistoa vähentävä nimitaulu.

"pictures"-taulu sisältää ominaisuuksista otettuja kuvia.

3.5 Web service

Rajapinta sisältää tietokannan muokkaamiseen tarvittavat metodit, jotka ovat "search", "edit" ja "delete". Jokainen näistä metodeista toimii lähettämällä haluttuun osoitteeseen POST-pyyntö. Pyynnön mukana lähtevän tiedon pitää olla JSON-muodossa. Takaisin tuleva vastaus on myös JSON-muodossa ja se sisältää tilakoodin, löydetyt tiedot ym. riippuen, mihin osoitteeseen se lähetettiin.

Tiedonsiirtomuodoksi on valittu JSON, koska se on todella yksinkertainen, erittäin kevyt sekä helppo muodostaa ja sen kääntämistä tukee miltei kaikki kielet.

Kun CakePHP palauttaa tuloksia tietokannasta, ne ovat assosiatiivisessä talukossa, jonka voi kääntää suoraan JSON-muotoon ja se voidaan selainpuolella kääntää suoraan JavaScript-objektiksi. Tätä objektia voidaan muokata selainpuolella ja kun se pitää lähettää takaisin palvelimelle sisältäen päivitettyä tietoa, voidaan se suoraan kääntää JSON-muotoon. Palvelimella JSON voidaan kääntää helposti yhdellä PHP-metodilla assosiatiiviseksi taulukoksi, joka on suoraan yhteensopiva CakePHP:n tietokantakäsittelyssä.

Jos käytössä olisi esimerkiksi XML, pitäisi se erikseen muokata erilaisiin muotoihin ja tämä lisäisi paljon turhaa työtä, jota JSON-muodon kanssa ei tarvitse ollenkaan.

3.5.1 Haku

Listaus 1:ssä esimerkki objektista, joka lähetetään "search"-metodille JSON-muodossa.

Listaus 1. Haku-objekti.

```
{
 "conditions":{
               "Customer.first_name LIKE":"%Pertti%",
               "Customer.last_name":"Hirvilä"
              },
 "joins":"false",
 "recursiveness":-1
}
```

"conditions"-property sisältää hakuehdot. "joins"-property sisältää boolean-arvon, jonka avulla joko poistetaan tai lisätään taululiitokset. "recursiveness"-property sisältää kokonaisluvun, jolla määräytyy se, kuinka syvältä haun tulokset palautetaan. Esimerkiksi "-1" palauttaisi vain "customers"-taulun tiedot ja "4" palauttaisi kaikki (poislukien "section_names" ja "property_names"), eli "customers", "buildings", "sections", "properties" ja "pictuters" -taulujen tiedot.

Onnistuessaan listaus 1:ssä oleva esimerkki palauttaisi listaus 2:ssa olevan JSON-tekstin.

Listaus 2. Onnistunut palautus.

```
{
 "Customer":{
             "id":"2",
             "first_name":"Pertti",
             "last_name":"Hirvila",
             "phone":"0400000000"
            }
};
```

Jos asiakasta ei löydy tietokannasta, palautetaan virhekoodi ja teksti (listaus 3).

Listaus 3. Epäonnistunut palautus.

```
{
 "code":"404",
 "message":"Not found"
};
```

3.5.2 Lisäys ja muokkaus

"edit"-metodi hoitaa tiedon muokkauksen ja lisäämisen. Olemassa olevia rivejä voidaan muokataan lisäämällä mukaan halutut id:t. Kantaan voidaan lisätä uusia rivejä jättämällä id:t pois. Tietueita, joita ei muokata, ei tarvitse lähettää.

Yhteydettömässä tilassa luodut tiedot saavat väliaikaisen id:n, jonka edessä on teksti "OFFLINE" ja sen perässä JavaScriptin getTime()-funktiolla luotu aika millisekunneissa. Tämänlainen id poistetaan automaattisesti palvelinpuolella, jotta voidaan luoda uusi rivi oikeanmuotoisella id:llä.

Listaus 4:ssa esimerkki objektista, joka lähetetään "edit"-metodille JSON-muodossa. Objekti lisää uuden asiakkaan kahdella uudella rakennuksella. Alempi objekti muokkaa jo olemassa olevan asiakkaan tietoja (id:t lisätty).

Listaus 4. Lisäys- ja muokkaus-objektit.

```
{Customer: {
            first_name: 'Pertti',
            last_name: 'Keinonen',
            phone: '0402377264'
               },
 Building: [{location: "Kasarminkatu 88", square_meter: "300"},
            {location: "Koulukatu 28", square_meter: "100"}]};
```

```
{Customer: {
            customer_id: '3',
            last_name: 'Östermalm',
            phone: '0401231232'
               },
 Building: [{building_id: "3", customer_id: "3",
             square_meter:  "300"}]};
```

3.5.3 Poisto

"delete"-metodille lähetetään JSON, joka sisältää luokan nimen (CakePHP) ja id:n.

Listaus 5:ssa esimerkki kahdesta vaihtoehtoisesta objektista. Sovelluksessa ensimmäistä muotoa käytetään silloin, kun poistetaan uusi rakennus yhteydellisessä tilassa ja toista muotoa käytetään yhteydettömässä tilassa, koska siihen voidaan liittää useita rakennuksia kerralla.

Listaus 5. Poisto-objekti.

```
{Class: "Building", id: "76"};

{del:[{Class: "Building", id: "85"},
      {Class: "Customer", id: "2"}]};
```

3.6 Käyttötapaukset

Sovelluksen testikäyttöliittymän tämänhetkiset käyttötapaukset.

3.6.1 Haku

Kuviossa 3 on haun aktiviteettikaavio.

Kuvio 3. Haku.

Toimija: Käyttäjä

Tehtävä: Käyttäjä hakee asiakkaita tietokannasta.

Tapahtumat:

1. Käyttäjä valitsee hakuehdon.
2. Käyttäjä täyttää hakukentän.
3. Käyttäjä painaa hakunappia.

Lopputulos: Sovellus listaa hakuehtoa vastaavat tulokset.

Poikkeukset: Haku ei toimi ilman yhteyttä palvelimeen.

3.6.2 Asiakkaan valitseminen

Toimija: Käyttäjä

Tehtävä: Käyttäjä valitsee asiakkaan.

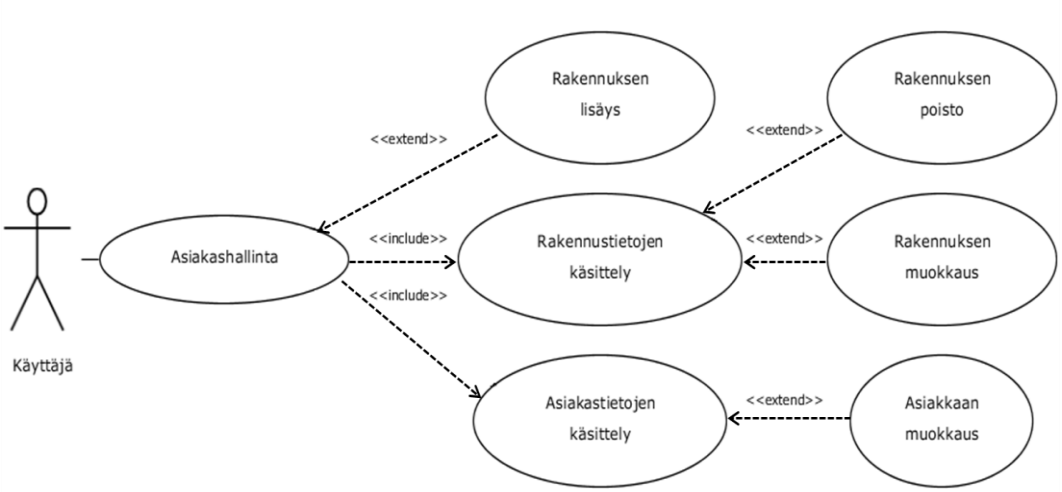
Tapahtumat:

1. Käyttäjä valitsee asikkaan linkistä / vetovalikosta.

Lopputulos: Käyttäjä näkee asiakkaan tiedot.

3.6.3 Asiakastietojen muokkaaminen

Kuviossa 4 on asiakashallinnan käyttötapauskaavio.



Kuvio 4. Asiakashallinta.

Toimija:Käyttäjä

Tehtävä: Käyttäjä muokkaa asiakastietoja ja painaa tallennusnappia.

Tapahtumat:

A. Käyttäjä muokkaa asiakkaan henkilötietoja ja painaa tallennusnappia.

B. Käyttäjä muokkaa asiakkaan rakennuksen tietoja ja painaa kyseisen rakennuksen tallennusnappia.

C. Käyttäjä luo rakennuksen täyttämällä kentät ja painamalla luomisnäppäintä.

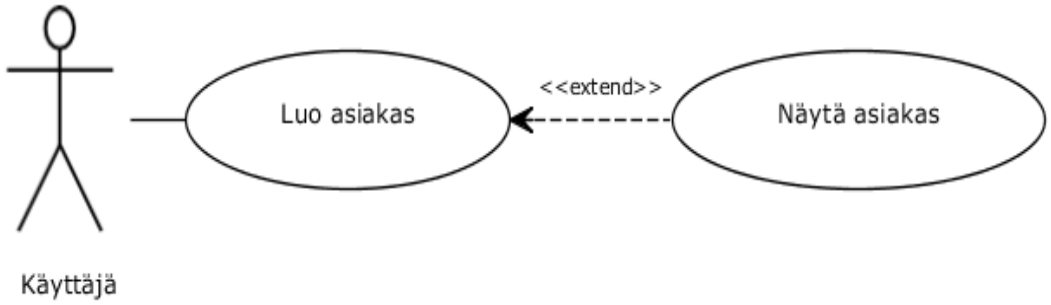
D. Käyttäjä poistaa rakennuksen painamalla rakennuksen poistonappia.

Lopputulos: Halutut muutokset tallentuvat asiakastietoihin.

Poikkeukset: Ilman yhteyttä palvelimeen asiakastiedot tallentuvat paikallisesti. Paikallisesti tallennetut muokkaukset lähetetään palvelimelle, kun yhteys on saatavilla.

3.6.4 Asiakkaan luominen

Kuviossa 5 on asiakkaan luomisen käyttötapauskaavio.



Kuvio 5. Asiakkaan luominen

Toimija:Käyttäjä

Tehtävä: Käyttäjä luo asiakkaan syöttämällä perustiedot ja painamalla luomisnappia.

Tapahtumat:

1. Käyttäjä syöttää asiakastiedot ja painaa luomisnappia.

Lopputulos: Käyttäjälle luodaan linkki asiakastietojen muokkaussivulle.

Poikkeukset: Yhteydettömässä tilassa tiedot tallennetaan paikallisesti. Yhteydettömässä tilassa luotujen asiakkaiden lähettäminen palvelimelle ei ole tehty valmiiksi, joten se toimii vain osittain.

4 RATKAISUT

Tämä kappale selostaa tärkeimmät tekniset ratkaisut, mitä käyttötapahtumien takana tapahtuu, eli esimerkiksi, miten tiedon tallennus väliaikaisesti tapahtuu ja miten se lähetetään palvelimelle.

4.1 Väliaikainen tallennus

Alla listattuna paikallisesti (localStorage) säilytettävät ja käytettävät tiedot.

Asiakastiedot esiintyvät paikallisesti tallennettuna erillisien avaimien takana, jotka ovat muotona "customer_" ja perään asiakkaan id, eli esimerkiksi "customer_4".

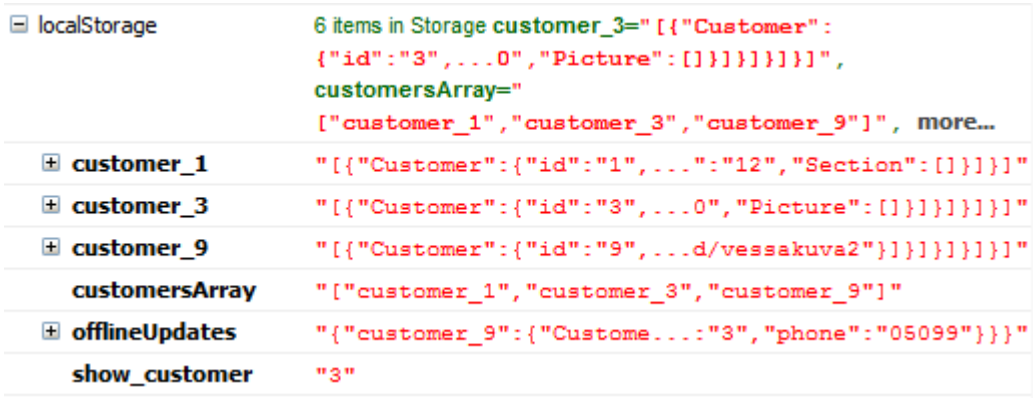
"customersArray" sisältää kaikkien asiakkaiden localStorage-avaimet (eli ylempänä mainitut avaimet). Tämän avulla voidaan helpottaa esimerkiksi asiakastietojen looppaamista localStoragesta.

"offlineUpdates" sisältää kaikki yhteydettömässä tilassa tehdyt muokkaukset. Nämä tiedot lähetetään palvelimelle silloin, kun yhteys on saatavilla.

"show_customer" sisältää asiakastietosivulla näytettävän asiakkaan id:n. Tämä siis vaihdetaan aina siihen asiakkaaseen, joka halutaan näyttää. Alunperin näytettävän asiakkaan id haettiin osoitekentän queryStringistä, mutta tämä ratkaisu ei toimi, jos yhteydettömässä tilassa luodaan uusi väliaikainen id.

Jälkimäisissä teksteissä "customersArray" sanalla tarkoitetaan sitä itseään, sekä localStorageen tallentuvia asiakastietoja, jotka ovat "customer_"-avaimien takana.

Kuviossa 3 esimerkki localStoragen sisällyksestä (Firebug).



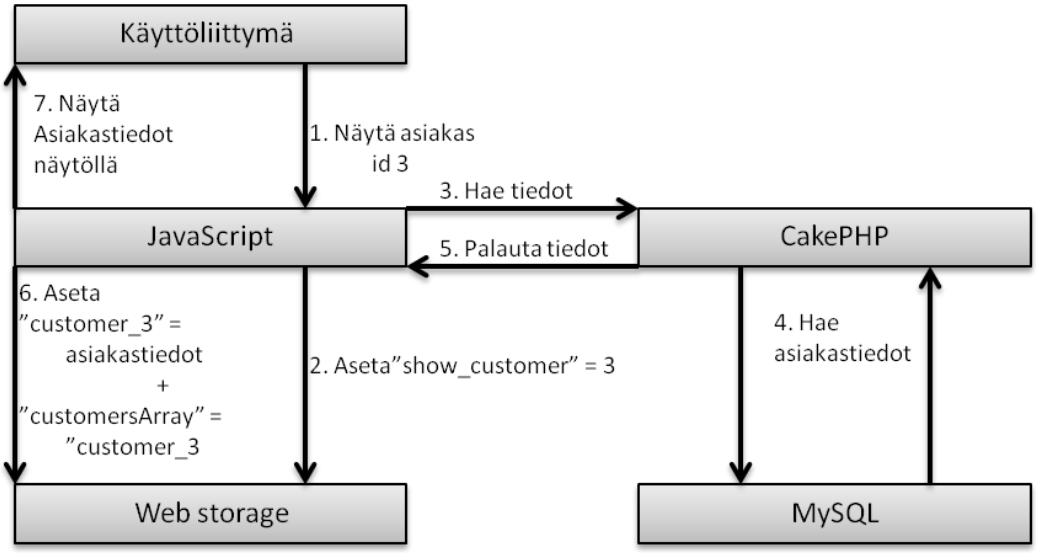
Kuvio 6. localStorage

4.2 Toiminnallinen kuvaus

Tässä kappaleessa esitellään tarkempia kuvauksia ohjelman toiminnasta. Tarkoituksena on antaa lukijalle tai mahdolliselle jatkokehittäjälle tarkempi kuva ohjelman toiminnasta.

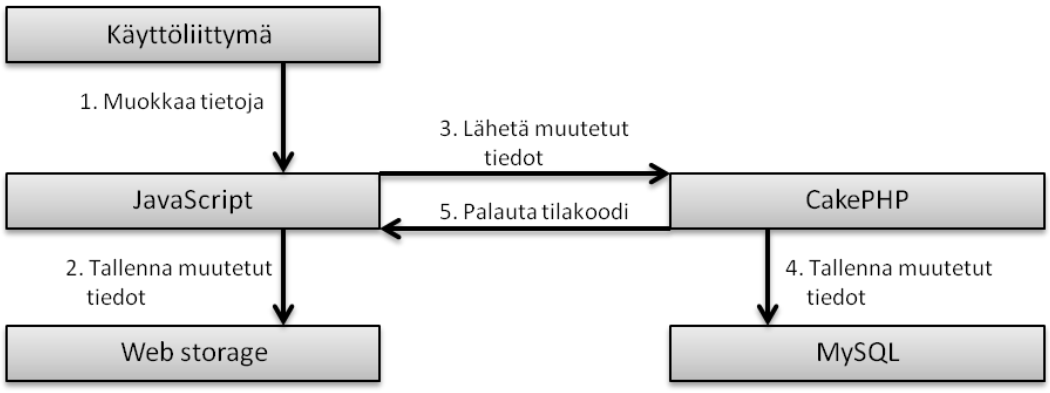
4.2.1 Yhteydellinen tila

Kun asiakastietoja haetaan palvelimelta käyttöliittymälle, tallentuvat ne aina paikallisesti (customersArray), vaikka verkkoyhteys olisi saatavilla. Tämä sen takia, että jos verkkoyhteys katkeaisi, niin käsiteltyjen asiakkaiden käsittelyä voidaan jatkaa normaalisti ja tiedot tallentuvat paikallisesti. Asiakkaita voidaan siis ladata etukäteen yhteydettömään tilaan.



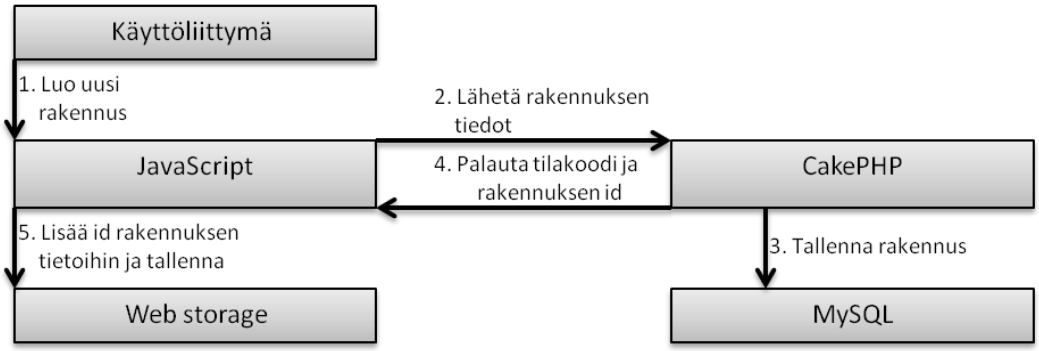
Kuvio 7. Asiakastietojen tuominen

Kun asiakkaan ja rakennuksien tietoja muokataan yhteydellisessä tilassa ja tallennetaan, niin syöttökenttien tietoja verrataan paikallisesti tallennettuun tietoon ja päätellään, mitkä tiedot ovat muuttuneet. Ainoastaan muuttuneet tiedot lähetetään palvelimelle sekä päivitetään paikallisesti tallennetut tiedot. Näin vähennetään myös liikennettä palvelimen ja käyttöliittymän välillä, koska palvelimen ei tarvitse lähettää tietoja uudestaan, vaan ainoastaan tilakoodi.



Kuvio 8. Tallennus

Jos yhteydellisessä tilassa luodaan uusi rakennus, lähetetään tallennetut tiedot palvelimelle ja sen jälkeen palvelin vastaa tilaviestillä, joka sisältää myös luodun rakennuksen id:n. Tämän jälkeen otetaan rakennuksen id viestistä ja tallennetaan tiedot paikallisesti.



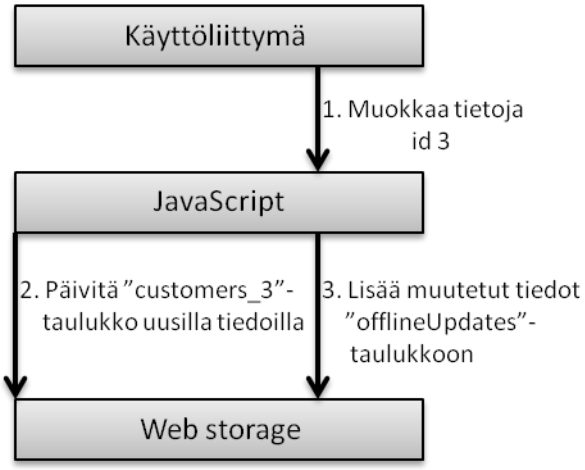
Kuvio 9. Rakennus

Kun rakennus poistetaan, lähetetään palvelimelle poisto-objekti ja poistetaan rakennus näytöltä, palvelimen tietokannasta sekä paikallisesti tallennetuista tiedoista.

4.2.2 Yhteydetön tila

Yhteydettömässä tilassa asiakastiedot tuodaan näytölle paikallisesti tallennetuista tiedoista. Kaikki paikallisesti tallentuneet asiakkaat ovat muokattavissa normaalisti.

Kun tietoja muokataan yhteydettömässä tilassa, ne tallentuvat paikallisesti customersArray-taulukkoon sekä offlineUpdates-taulukkoon. offlineUpdates-taulukkoon tallennetaan vain muutetut tiedot ja täten vähennetään palvelimelle lähetettävän tiedon määrää. Kun yhteys palvelimeen on saatavilla, lähetetään offlineUpdates-taulukon sisältö palvelimelle ja poistetaan se.



Kuvio 10. Tallennus (yhteydetön)

Kun yhteydettömässä tilassa lisätään uusi rakennus, se saa uniikin väliaikaisen id:n. Väliaikaista id:tä tarvitaan, jotta oikeaan rakennukseen voidaan viitata esimerkiksi muokkauksen tai poiston yhteydessä. Kun palvelimelle lähetetään päivityksiä, jotka sisältävät väliaikaisia id:itä, niin ne poistetaan ja tilalle lisätään oikeanlainen id MySQL:n toimesta.

Jos yhteydettömässä tilassa poistetaan rakennus, jolla on "oikea" id (eli luotu yhteydellisessä tilassa), lisätään offlineUpdates-taulukkoon osiossa 3.4.3 mainittu poisto-objekti ja poistetaan rakennus käyttöliittymästä sekä paikallisesti tallennetuista asiakastiedoista. Rakennus, jolla on väliaikainen id poistetaan suoraan paikallisesti tallennetuista tiedoista ja käyttöliittymästä ilman poisto-objektin lisäämistä, koska rakennus ei ole vielä palvelimen tietokannassa.

4.3 Idea

Sovelluksen koodeja olisi melko epäkäytännöllistä listata ja selittää raportissa, mutta listauksessa 6 on yksi tärkeä koodinpätkä selainpuolelta. Kyseessä on jQuery-kirjaston tapa hoitaa POST-lähetys. Koodilla hoidetaan siis Ajaxin käyttöä, mutta käytännöllisemmässä muodossa.

Listaus 6. jQuery post

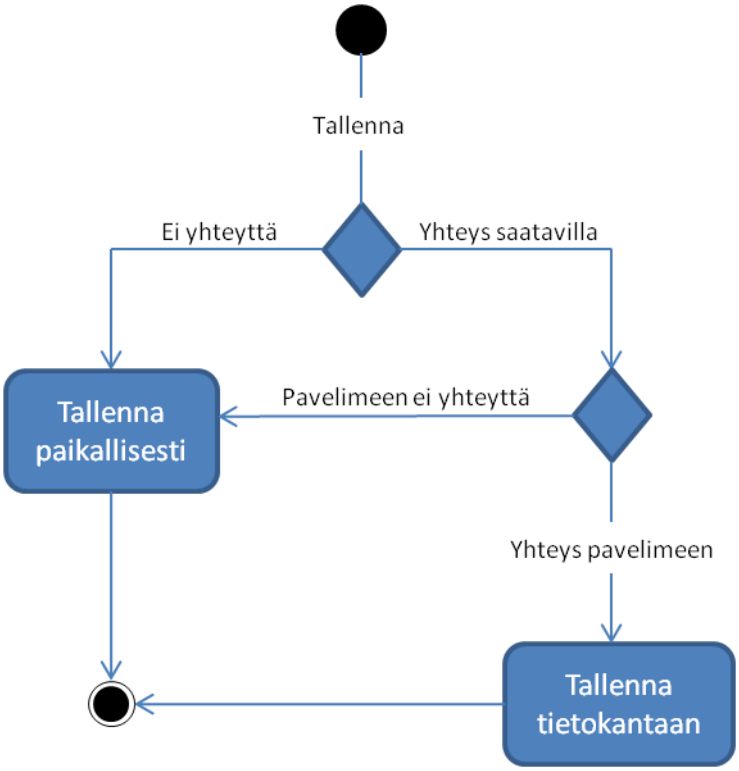
```
$.post(url, post, function(data){
      // Lohko 1
},"json").error(function(data){
      // Lohko 2
});
```

Lohko 1 suoritetaan, jos palvelimeen saadaan yhteys. Lohko 2 suoritetaan, kun palvelimeen ei saada yhteyttä. Tällä idealla siis toteutetaan väliaikainen tallennus.

Esimerkki 1

Jos asiakkaan tietoja muokataan ja tallennetaan, niin ensimmäiseksi suoritetaan lohko 1 ja yritetään lähettää tiedot palvelimelle, mutta jos se epäonnistuu, niin lohko 2 suoritetaan ja tiedot tallennetaan väliaikaisesti selaimen web storageen.

Kuviossa 11 on tallennuksen aktiviteettikaavio.

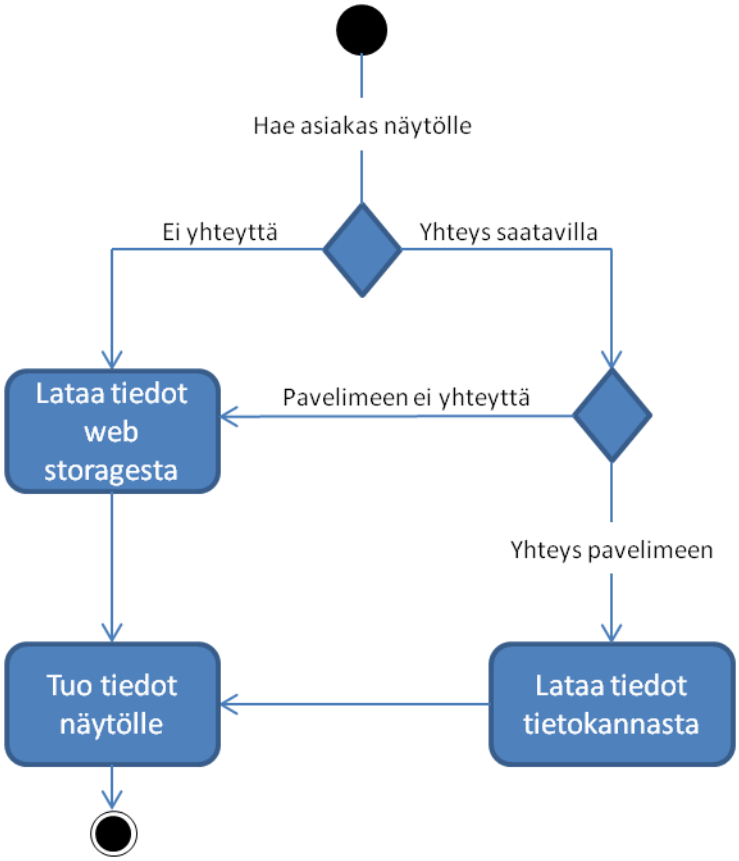


Kuvio 11. Tallennusaktiviteetti.

Esimerkki 2

Jos asiakkaan tietoja haetaan näytölle, niin ensisijaisesti suoritetaan lohko 1 ja yritetään tuoda tiedot palvelimelta, mutta jos se epäonnistuu, niin suoritetaan lohko 2 ja haetaan tiedot web storagesta (oletuksena tietenkin, että asiakas on ladattu sinne etukäteen).

Kuviossa 12 on asiakastietojen näytölle tuomisen aktiviteettikaavio.



Kuvio 12. Tiedon tuomisen aktiviteetti.

5 JATKOKEHITYS

Ohjelmassa on vielä paljon tekemistä ja jatkokehitys vaatii tietämystä useammasta eri tekniikasta. Tämä kappale ei vältämättä anna kaikkia tarvittavia tietoja jatkokehityksestä, vaan ainoastaan kokonaiskuvan, mitä pitäisi vielä tehdä sekä ideoita ja ehdotuksia.

5.1 Vaatimukset

Käyttöliittymän kehitys vaatii tietämystä HTML:stä, CSS:stä, JavaScriptista ja sille luodusta jQuery-kirjastosta. Tärkeää on tietää, miten jQuery hoitaa Ajaxin käytön, tapahtumat ja tapahtumien lisäämisen dynaamisesti luoduille HTML-elementeille.

Palvelinpuolen kehitys tarvii tietämystä PHP:sta, CakePHP-ohjelmistokehyksestä ja MySQL:stä. CakePHP:ta on käytetty ainoastaan web service -rajapintana, mutta jatkossa sillä pitäisi luoda myös esimerkiksi tiedon validointi, autentikointi ja käyttäjähallinta. CakePHP:n omilta sivuilta on saatavilla ohjeet kaikkiin yllä mainittuihin asioihin.

5.2 Kehitettävät kohteet

Alla ehdotuksia ja ideoita jatkokehitykseen.

Käyttöliittymät

Asiakashallinta-käyttöliittymä pitäisi rakentaa kattamaan koko tietokannan muokkaus. Alla on luettelo isoimmista asioista, mitä käyttöliittymästä puuttuu. Luettelo on myös mahdollisessa kehitysjärjestyksessä. Tämä luettelo koskee käyttöliittymää, joka tukee yhteydetöntä tilaa.

1. Asiakkaiden luominen yhteydettömässä tilassa (kehitys aloitettu, mutta vaiheessa) ja asiakkaiden poistaminen.

2. Rakennusten osioiden (sections) lisääminen, muokkaus ja poistaminen.

3. Osioiden ominaisuuksien (properties) lisääminen, muokkaus ja poistaminen.

4. Ominaisuuksille kuvien lisääminen ja poistaminen. Ominaisuudet pitää pystyä ryhmittämään järkevästi tarkastuspäivän (check_date) mukaan, eli esimerkiksi Vessalla (Osio) voi olla monta lattiaa (ominaisuus), jotka tarkoittavat sitä samaa lattiaa, mutta eri tarkistuspäivinä.

5. Paremmat ja laajemmat hakuominaisuudet.

6. Käyttäjän syötteen validointi.

7. Käyttöliittymän internalisaatio ja mahdollisuus helposti lisätä tuki uudelle kielelle.

8. Sovelluksen grafiikat ja käytettävyys.

9. Käyttöliittymän käyttäjän autentikointi.

Kuvien lisääminen ominaisuuksille kannattaa luultavasti toteuttaa erillisillä skripteillä ilman CakePHP:ta. jQuery-kirjastolle löytyy lisäosia, jolla voidaan tiedostojen liittäminen hoitaa helposti. Nämä lisäosat tarvitsevat tietenkin palvelimelle skriptit, jotka hoitavat kuvien käsittelyn.

Käyttöliittymässä voisi olla myös ominaisuutena automaattinen kenttien täyttäminen ja ehdotuksien antaminen, koska tämä nopeuttaisi käyttöä taulutietokoneilla. Nämä ominaisuudet olisivat käytössä osioiden ja ominaisuuksien nimissä.

Muita erillisiä käyttöliittymiä (joiden ei tarvitse toimia yhteydettömässä tilassa, joten voidaan rakentaa esim. PHP:lla) ovat mm. käyttäjähallinta ja raporttien luominen.

Asiakassovelluksen käyttäjien hallinnalle tarvitaan hallintasivu, jossa voi luoda, poistaa ja muokata tunnuksia.

Raporttien luomisella tarkoitetaan, että MySQL-tietokannasta muodostetaan erilaisia raportteja halutuin määritelmin.

Web service

Palvelinpuolen tiedon validointia ja autentikointia ei ole toteutettu, mutta koska käytössä on ohjelmistokehys, niin niiden toteuttaminen on helppoa. Varsinkin muokkaamis- ja haku-metodi tarvitsevat lisäkehitystä, jotta ne saadan vastaamaan yllä olevia vaatimuksia.

Tietokanta

Tietokannan taulut sisältävät tällä hetkellä vain pienen määrän kenttiä, jotta ohjelman kehittäminen olisi helpompaa. Toimeksiantajan kanssa täytyy keskustella ja selvittää kaikki tarvittavat kentät.

Kokonaisraportti

Asiakkaalle luotu raportti rakennuksen tiedoista ja mahdollisista kehitysehdotuksista. Tämä raportti lähetetään esimerkiksi asiakkaan sähköpostiin. Raportti voitaisiin tulostaa esimerkiksi HTML-sivuksi ja antaa sähköpostissa linkki tai mahdollisesti PDF-tiedoston luominen ja sen liittäminen sähköpostiin.

Jos raportti luodaan verkkosivuna, pitäisi sen olla salauksen takana. Salasana voitaisiin luoda etukäteen ja antaa vaikka sähköpostin mukana, jotta asiakkaan ei tarvitsisi itse keksiä ja muistaa sitä. Verkkosivuna toteutettu ratkaisu olisi luultavammin helpompi ja käytännöllisempi kuin PDF:n luominen.

PDF-tiedoston luominen tarvitsisi luultavasti huomattavasti enemmän aikaa ja syvällisempää paneutumista johonkin PDF:n luontiin tarkoitettuun ohjelmistokirjastoon.

5.3 Kehitysympäristö

Kehitysympäristönä on XAMPP, joten se on helppo asentaa. Sovelluksen pitäisi toimia ilman konfiguraatiota. CakePHP toimii myös normaalisti ilman konfigu-

raatiota, mutta joissain tapauksissa jotkin tiedostot voivat tarvita pieniä muutoksia.

Sovellus koostuu kahdesta pääkansiosta, jotka ovat käyttöliittymä ja CakePHP. Nämä kansiot pitää siis pistää XAMPPin htdocs-kansioon. Sovellus toimii tietenkin muillakin ohjelmistokokonaisuuksilla kuin XAMPP, mutta sen käyttäminen tähän olisi kaikista helpoin tapa.

6 JOHTOPÄÄTELMÄT

Toimeksiannon saadessani ajattelin, että työ olisi melko helppo tehdä ja saisin koko sovelluksen tehtyä valmiiksi. Kaikista muista vaatimuksista paitsi "yhteydettömässä tilassa toimiminen" minulla oli melko hyvä käsitys siitä, miten vaativia ne ovat toteuttaa. Työn aloittamista hankaloitti se, että en ollut vuoteen miettinyt ollenkaan ohjelmointia, joten jouduin syntaksista lähtien alottamaan eri kielien muistelua.

Kun alkukankeuden jälkeen pääsin miettimään ja aloittamaan itse sovellusta, tuli melko nopeasti selville, että vaatimuksen "yhteydettömässä tilassa toiminen" toteuttaminen oli kaikkea muuta kuin helppoa. Kyseisestä asiasta ei löytynyt juuri mitään kunnon tietoa internetistä tai kirjoista, ainoastaan yksinkertaisia esimerkkejä, joista ei juuri apua ollut.

Sovelluksen yhteydettömän osan toteuttaminen oli vaikea siksi, että tiedon muokkaus, poistaminen ja luominen piti olla mahdollista ilman yhteyttä palvelimeen. Koska selaimessa ei ole SQL-tietokantaa tai vastaavaa, vaan ainoastaan assosiatiivisen taulukon tapaan toimiva tallennuspaikka, niin näiden toimintojen toteuttaminen vaati paljon aikaa ja miettimistä. Tämän takia oli lähes mahdotonta suunnitella rakenteita etukäteen, joten sovellus eteni suurimmaksi osaksi kokeilemalla erilaisia toteutuksia ja valitsemalla niistä järkevin. Tämän takia opinnäytetyö tuli keskittymään enemmänkin kyseisen vaatimuksen toteuttamiseen.

Opinnäytetyön tekeminen vaati yhteydettömän tilan toteuttamisen lisäksi myös tutustumista muihin asioihin, joista minulla ei aiemmin ollut paljoa tai yhtään kokemusta. Esimerkiksi web servicen toteuttamisesta ei ollut pieniä testailuja lukuunottamatta juuri kokemusta, joten sain paremman käsityksen, mitä se vaatii ja miten semmoinen toteutetaan. JavaScriptin jQuery-kirjastoon tutustuminen ja sen käyttö oli myös erittäin hyödyllinen ratkaisu.

Sovelluksen ratkaisut, varsinkin yhteydettömän tilan toteutetuksen ratkaisut, eivät välttämättä ole aina ne kaikista järkevimmät. Suurin vaikuttaja tähän on se, että tekniikasta on vielä melko vähän tietoa. Joitain ei-niin-hyviä ratkaisuja oli yksin-

kertaisesti pakko hyväksyä, koska aikaa ei ollut loputtomasti ja sovelluksen kehitystä piti saada eteenpäin.

Kokonaisuudessaan opinnäytetyö oli varsin opettava ja erittäin haasteellinen kokemus.

LÄHTEET

Mozilla (2012). [Viitattu 7.4.2012]. Saatavilla internetissä:
<URL: https://developer.mozilla.org/en/JavaScript/About_JavaScript>

Mozilla (2012). [Viitattu 7.4.2012]. Saatavilla internetissä:
<URL: https://developer.mozilla.org/en/AJAX>

JSON (2012). [Viitattu 7.4.2012]. Saatavilla internetissä:
<URL: http://www.json.org/>

jQuery (2012). [Viitattu 7.4.2012]. Saatavilla internetissä:
<URL: http://jquery.com/>

PHP (2012). [Viitattu 7.4.2012]. Saatavilla internetissä:
<URL: http://www.php.net/>

CakePHP (2012). [Viitattu 7.4.2012]. Saatavilla internetissä:
<URL: http://cakephp.org/>

MySQL (2012). [Viitattu 7.4.2012]. Saatavilla internetissä:
<URL: http://www.mysql.com/>

Firebug (2012). [Viitattu 7.4.2012]. Saatavilla internetissä:
<URL: http://getfirebug.com/>

Aptana (2012). [Viitattu 7.4.2012]. Saatavilla internetissä:
<URL: http://aptana.com>

Apachefriends (2012). [Viitattu 7.4.2012]. Saatavilla internetissä:
<URL: http://www.apachefriends.org/en/xampp.html>

phpMyAdmin (2012). [Viitattu 27.4.2012]. Saatavilla internetissä:
<URL: www.phpmyadmin.net/>