Eero Tukiainen

SOVELLUS KIINTEISTÖHUOLLON HUOLTOJEN HALLINTAAN

Tietojenkäsittelyn koulutusohjelma
2017

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

SOVELLUS KIINTEISTÖHUOLLON HUOLTOJEN HALLINTAAN

Tukiainen, Eero
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Huhtikuu 2017
Ohjaaja: Nieminen, Hans
Sivumäärä: 38
Liitteitä: 0

Asiasanat: WWW, HTML, web-ohjelmointi

Tämän opinnäytetyön aiheena oli selainpohjaisen huoltosovelluksen toteuttaminen Ruskatalojen palveluyhdistyksen kiinteistöhuollolle. Sovellus koostuu käyttöliittymästä selaimessa, tietokannasta sekä palvelinpuolen toiminnallisuudesta. Sovellus on tarkoitus ottaa käyttöön vuoden 2017 kesän aikana.

Sovellukselle määriteltiin aloituspalaverissa yksinkertaiset tavoitteet. Sen piti käsittää listat aktiivisista- ja käsitellyistä huoltoilmoituksista, sekä sivun huoltoilmoitusten luomiseen säilyttäen mahdollisimman yksinkertaisen käyttöliittymän. Työ osoittautui haastavaksi, ja se opetti paljon web-ympäristöstä sovellusalustana.

Sovelluksen on tarkoitus korvata perinteinen huoltoilmoitusten käsittely, joka sisältää paperityötä. Yhdistyksen työtekijät tulisivat käyttämään sovellusta pääosin tietokoneilla ja älypuhelimilla.

Opinnäytetyössä käydään lävitse mitä web-sovellukset ovat, jatkaen työssä käytettyihin tekniikkoihin sekä itse toteutettuun sovellukseen.

AN APPLICATION FOR HANDLING AND MANAGING PROPERTY MAINTENANCE NOTICES

Tukiainen, Eero
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Business Information Systems
April 2017
Supervisor: Nieminen, Hans
Number of pages: 38
Appendices: 0

Keywords: WWW, HTML, web-programming

This thesis' subject was to create a browser based application for Ruskatalojen Palveluyhdistys ry's property maintenance. The application consists of user interface in browser, database and server side logic. The application is scheduled to be operational by the summer of 2017.

The application was given simple objectives in the kick-off meeting. It had to comprise lists of active and processed maintenance notices, as well as an interface for creating notices while remaining simple to use. The task proved challenging, and it taught much of the web-environment as an application platform.

The application is intended to replace the traditional process of handling maintenance notices, which includes paperwork. The personnel of the association will be using the application mainly with computers and smartphones.

This thesis will go through the definition of web-application, continuing to technologies used in the making of the application and finally to the application itself.

SISÄLLYS

LYHENTEET JA TERMISTÖ

AJAX	*Asyncronous JavaScript And Xml*. Joukko web-sovelluskehityksen tekniikoita, joiden avulla web-sovelluksista voidaan tehdä vuorovaikutteisempia.
CSS	*Cascading Style Sheets*. Verkkosivujen tyylittämisen käytetty tietokonekieli.
HTML	*Hyper Text Markup Language*. Yleinen verkkosivujen luomiseen käytetty tietokonekieli.
HTTP	*Hypertext Transfer Protocol*. Protokolla, jota selaimet ja web-palvelimet käyttävät tiedonsiirtoon.
JSON	*JavaScript Object Notation*. Kevyt tiedostomuoto tiedonvälitykseen.
LDAP	*Lightweight Directory Access Protocol*. Protokolla verkkohakemistopalveluiden käyttämiseen.
PHP	*PHP: Hypertext Preprocessor*. Ohjelmointikieli, jota käytetään web-palvelinympäristössä dynaamisten web-sivujen luonnissa.
SQL	Kyselykieli, jolla relaatiotietokantaan voidaan tehdä erilaisia hakuja, muutoksia ja lisäyksiä.
URI	*Uniform Resource Identifier*. Merkkijono, jolla kerrotaan tietyn tiedon paikka (URL) tai yksikäsitteinen nimi (URN).
URL	*Uniform Resource Locator*. Käytetään osoittamaan WWW-sivuja.

Web-palvelin	Tietokone tai ohjelmisto, joka jakaa dokumentteja HTTP-protokollalla asiakasohjelmille ja koneille.
WWW	*World Wide Web*. Internet-verkossa toimiva hajautettu hypertekstijärjestelmä.

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena oli luoda Ruskatalojen Palveluyhdistyksen kiinteistöhuollolle selaimessa toimiva huoltojärjestelmä. Huoltojärjestelmän on tarkoitus sisältää listat sekä aktiivisista että käsitellyistä huoltoilmoituksista, kuten myös ominaisuus huoltoilmoitusten jättämiseen.

Ruskatalojen Palveluyhdistys ry on Porin kaupungissa ja sen lähiympäristössä toimiva vanhuspalveluita tarjoava organisaatio. Yhdistys on perustettu vuonna 1954 luomaan hyvinvointia porilaisille vanhuksille, jolloin yhdistys tunnettiin vielä nimellä Porin Vanhojen Huolto ry. Yhdistyksen toimintaa laajennettiin 1980-luvulla ateria- ja kotipalvelujen suuntaan. Yhdistys sai nykyisen nimensä vuonna 2009, kun toiminta laajentui useampaan palvelutaloon. Ensimmäinen palvelutalo rakentui vuonna 1995 Porin Pormestarinluotoon, jossa se on vaiheittain laajentunut kiinteistönä vuosina 1999 ja 2000. (Ruskatalojen palveluyhdistys ry a; Ruskatalojen palveluyhdistys ry b.)

Yhdistyksen tarve huoltojärjestelmälle on tullut kiinteistöjen kasvaneesta lukumäärästä. Yhdistyksellä on oma kiinteistöhuolto, jonka tehtävänä on huoltaa ja ylläpitää kaikkia yhdistyksen kiinteistöjä. Nykyinen vanhentunut huoltojen käsittelymenetelmä on alkanut hankaloittaa työn tekemistä, jonka vuoksi tarve nykyaikaiselle huoltojärjestelmälle on suuri. Vaatimuksena on, että tulevaa järjestelmää voidaan hyödyntää sekä tietokoneilla että älypuhelimilla.

Tämä opinnäytetyö aloitetaan tekemällä katsaus web-sovellusten määritelmiin, jatkaen teoriassa yleisimpien web-sovellustekniikoiden esittelyyn sekä käytännön esimerkkeihin. Myöhemmin työssä käydään lävitse toteutettavaa sovellusta projektina sekä sen lopputulosta.

2 WEB-SOVELLUKSET

Web-sovellus on sovellus, joka käyttää selainta toimintaympäristönä. Web-sovelluksen toteuttamiseen on erilaisia rakenteita ja ratkaisuja.

2.1 Verkkosivujen toimintaperiaate lyhyesti

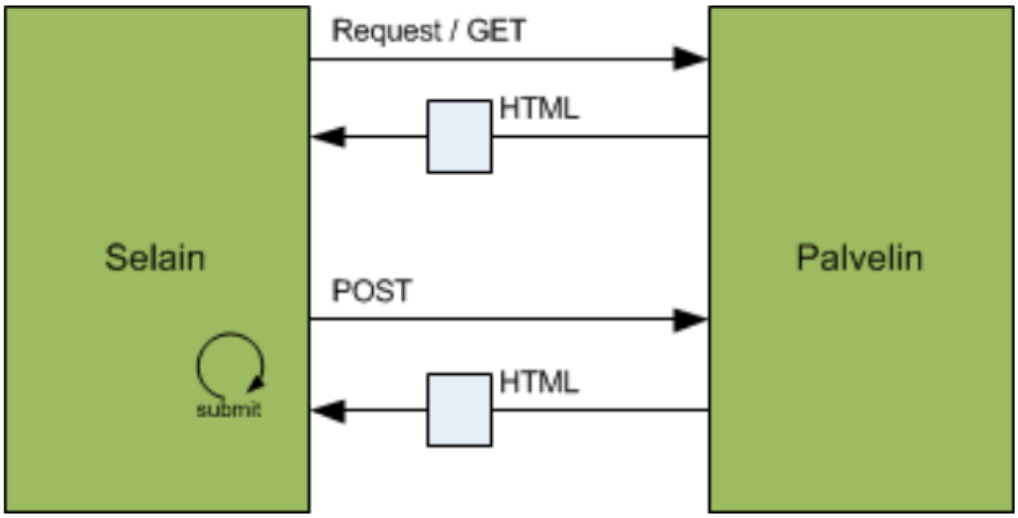
Verkkosivuja voidaan toteuttaa useiden eri ohjelmointikielien ja tekniikoiden avulla. Sivujen perustana toimii HTML-kieli ja usein http- tai https-tiedonsiirtoprotokolla. Selain lähettää pyynnön palvelimelle http-protokollaa käyttäen. Tämä pyyntö sisältää palvelimen verkko-osoitteen ja tekstiosuuden, josta palvelin päättelee, miten sen tulee vastata lähetettyyn pyyntöön. Palvelin käsittelee pyynnön ja se lähettää sen takaisin käyttäjän selaimelle, jos sivu on olemassa. Tarvittaessa sivua muokataan eri ohjelmointikielten avulla palvelimen päässä ja sivun sisältö saatetaan hakea esimerkiksi tietokannasta tai sivulla olevalla materiaalilla saatetaan tehdä laskutoimituksia. (Rajala 2009.)

2.2 Perinteiset web-sovellukset

Perinteiset web-sovellukset perustuvat kolmitasoarkkitehtuuriin. Nämä kolme tasoa ovat selain, web-palvelin ja tietokanta. Ensimmäinen, eli selaintaso, on käyttöliittymän toteutustaso sisältäen HTML-sivun esittämisen ja JavaScript-koodin suorittamisen. Toinen taso, eli web-palvelin, vuorostaan käsittää web-sovelluksen sovelluslogiikan, tietojen noutamisen tietokannasta ja selaimelle palautettavan HTML:n muodostamisen. Samassa tasossa käsitellään myös käyttäjään ja dataan liittyvät tunnistautumiset ja todennukset. Kolmannessa, eli tietokantatasossa, säilytetään sovelluksessa käytettävää dataa, kuten esimerkiksi käyttäjä-, asiakas-, tuote- ja tilaustietoja. Kuvassa 1 on hahmotettuna kolmitasoarkkitehtuurin rakenne.

Kuva 1. Kolmitasoarkkitehtuurin rakenne (Nieminen 2016).

Tasojen suoritusjärjestys etenee selaintasosta tietokantatasolle ja sieltä takaisin selaimelle. Perinteisessä mallissa selain keskustelee web-palvelimen kanssa http:n GET- ja POST-komentoja käyttäen. Tällöin käyttöliittymän muutokset selaimessa tapahtuvat sivun uudelleenlatauksen myötä. Kuvassa 2 on hahmotettuna selaimen ja palvelimen välinen liikenne.

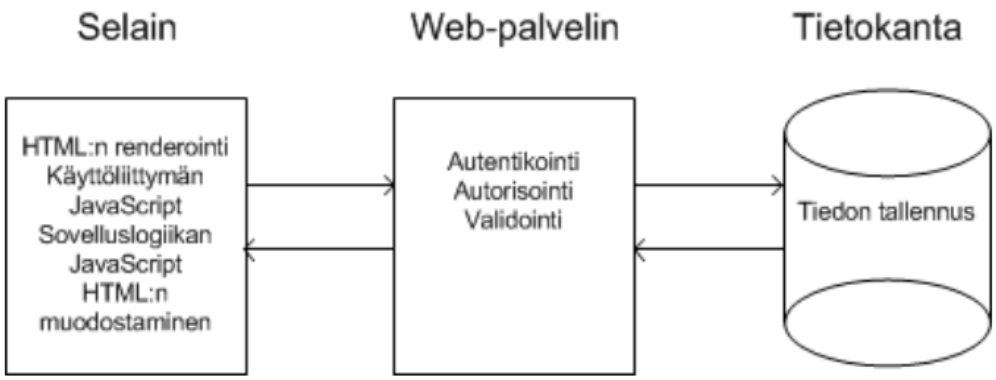


Kuva 2. Selaimen ja palvelimen välinen liikenne hahmotettuna (Nieminen 2016).

GET-komentoa käytetään HTML-sisällön ja datan noutamiseen määritetystä resurssista, kun taas POST-komentoa käytetään datan lähettämiseen määritettyyn resurssiin prosessoitavaksi. GET-pyyntöjen vastauksia voidaan tallentaa palvelimen välimuistiin tehostamaan toimintaa. Komentoa tulisi kuitenkin käyttää vain datan noutamiseen joka ei ole arkaluontoista. POST-pyyntöjä ei voida tallentaa välimuistiin, eikä tuloksista muodostu jaettavaa linkkiä. Tämän vuoksi komento onkin hieman tietoturvallisempi kuin GET ja sitä suositellaan käytettäväksi arkaluontoisemman datan käsittelyssä. Turvallisuus paranee, koska POST-komennossa voidaan URL-osoitteen yhteydessä annettavien parametrien asemesta käyttää komennon body-osaa. Tällöin parametrien arvot eivät tule tallennetuiksi selaimen selaushistoriaan eikä web-palvelimen lokitiedostoihin. Komento ei kuitenkaan salaa käsiteltävää dataa, joten data ei ole piilossa verkkoliikenteen seuraamisohjelmilta. (Rajala 2009; w3schools.com b.)

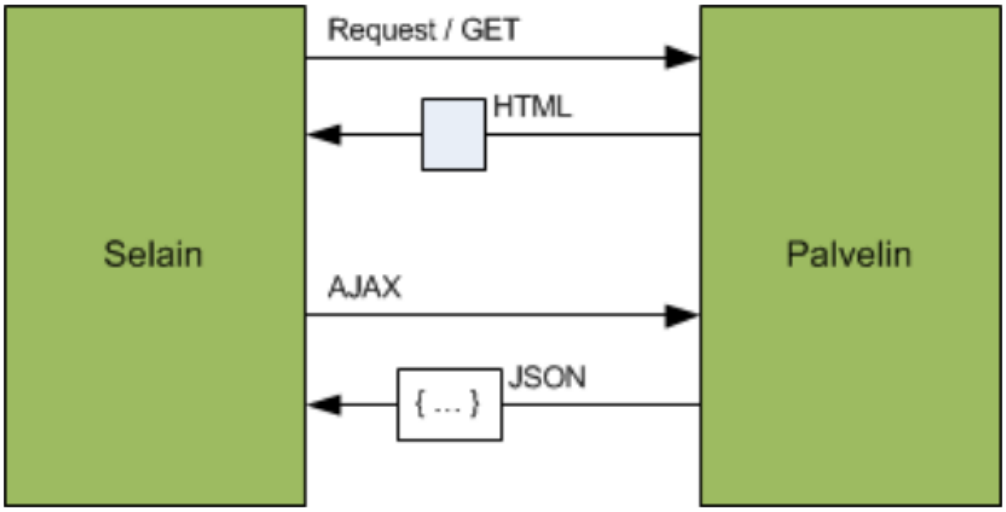
2.3 Vuorovaikutteiset web-sovellukset

Vuorovaikutteiset web-sovellukset omaavat perinteisten web-sovelluksien tapaan kolmitasoarkkitehtuurin, mutta toiminnallisuuden painottuminen on erilainen. Nykyaikaisissa web-sovelluksissa selaintasoon on sisällytetty enemmän toimintoja: aiemmin perinteisen web-sovelluksen palvelintaso käsitti sovelluslogiikan ja HTML:n muodostamisen, mutta nykyaikaisessa mallissa ne on siirretty selaintasolle. Käytännössä tämän kaltaista mallia kutsutaan nimellä SPA (Single Page Application), eli yhden sivun sovellukseksi. SPA perustuu vahvasti JavaScriptin käyttöön niin, että koko sovellus on käytännössä vain yksi HTML-sivu. Kuvassa 3 on hahmotettuna SPA:n kolmitasoarkkitehtuuri. (Nieminen 2016).



Kuva 3. SPA:n kolmitasoarkkitehtuurin rakenne (Nieminen 2016).

SPA on verkkosivu tai web-sovellus, jossa sivun sisältö rakentuu yhden sivun alle, tarjoten käyttäjälle työpöytäsovelluksen kaltaisen käyttökokemuksen. Koko sivun sisältö ladataan yhdellä kertaa ja sisällön muuttuessa vain muuttuneet tiedot päivitetään, joten koko verkkosivun laajuisia uudelleenlatauksia ei tarvita sisällön muuttuessa. Tämä tarjoaa käyttäjälle sulavamman ja nopeamman verkkovierailukokemuksen. Kuuluisimpia esimerkkejä tämän kaltaisista toteutuksista ovat esimerkiksi Googlen Gmail, Maps ja Drive sekä Applen iCloud. Mallista käytetään myös termiä SPI (Single Page Interface). Kuvassa 4 hahmotetaan SPA-sovelluksen selaimen ja palvelimen välistä yhteyttä. (4psa.com 2013.)



Kuva 4. SPA-sovelluksen selaimen ja palvelimen välinen liikenne hahmotettuna (Nieminen 2016).

Perinteisessä web-sovelluksessa käytetty POST-komento on korvattu SPA-sovelluksessa AJAX-kutsulla. AJAXin avulla web-palvelimelta voidaan pyytää ja vastaanottaa dataa verkkosivun latautumisen jälkeen ja web-palvelimelle voidaan lähettää dataa taustalla. AJAX ei ole ohjelmointikieli, vaan web-sovellustekniikoiden joukko. AJAXia hyödyntävät sovellukset saattavat käyttää tiedon välittämiseen nimessä esiintyvän termin mukaan XML-tekniikkaa, mutta ne voivat yhtä hyvin välittää tiedot JSON-muodossa tai puhtaana tekstinä. (w3schools.com c.)

AJAX-kutsussa luodaan XMLHttpRequest-objekti, joka lähetetään web-palvelimelle. Noudettava data voisi olla kuvan 4 mukaisesti JSON-tyyppistä. Web-palvelin käsittelee pyynnön ja luo vastauksen, jonka se lähettää takaisin pyynnön esittäjän selaimelle. Vastauksesta saadut tiedot jäsennetään verkkosivulle DOM-elementteihin JavaScriptin avulla. JSON-datan jäsentämiseen käytetään JavaScriptin funktiota JSON.parse(). (tutorialspoint; w3schools.com c.)

3 KÄYTETYT TEKNIIKAT

3.1 HTML5

HTML on standardoitu kieli verkkosivujen määrittelyyn ja sen tuorein versio on HTML5. HTML5 julkaistiin virallisesti 28. lokakuuta 2014 ja samalla siitä tuli myös W3C:n suositus. (World Wide Web Consortium a 2014.)

HTML-sivu koostuu koodista, joka sisältää HTML-kielen elementtejä ja niiden sisältöä. HTML-elementit ovat käyttäjälle piilotettuja avainsanoja, jotka määrittelevät miten selaimen tulisi järjestellä tarkasteltavana oleva verkkosivu. HTML-tiedostossa esiintyvillä elementeillä on alku- ja päätemerkinnät. Merkinnät ovat muuten samanlaisia, mutta päätemerkinnässä on kauttaviiva ennen elementin tunnusta. Näiden väliin sijoitettu teksti muotoillaan elementin ominaisuuksien mukaan. Elementtien sulkemisjärjestykseen tulee kiinnittää huomiota, sillä HTML-elementeillä voi olla sisäkkäisiä elementtejä, jolloin niiden sulkeminen tulee aloittaa sisimmästä uloimpaan. HTML-tiedoston tulee käsittää neljä pakollista elementtiä, jotka ovat HTML, HEAD, TITLE ja BODY. Pakolliset elementit ja niiden rakenne ovat esitettynä kuvassa 5. (Duncan 2015.)

```
1   <!DOCTYPE html>
2   <html>
3       <head>
4           <title>Otsikko</title>
5       </head>
6       <body>
7
8           <p>Hei maailma!</p>
9           <!-- Tämä on kommentti -->
10          <img src="helloworld.jpg" alt="kuva">
11
12      </body>
13  </html>
```

Kuva 5. Esimerkki HTML-koodista

HTML5 toi mukanaan nipun uusia elementtejä ja samalla joitain elementtejä poistettiin. Suurin osa poistetuista elementeistä liittyi tekstin muotoiluun, jotka toteutetaan nyt tyylitiedostojen (CSS3) avulla. Uudet merkittävimmät elementit liittyvät median esittämiseen sekä tekstisisällön järjestelemiseen verkkosivulla. (Kyrnin 2016.)

Median esittämiseen HTML5 toi VIDEO- ja AUDIO-elementit. Elementtien nimet kuvaavat niiden toiminnallisuutta jo paljon, eli VIDEO-elementtiä käytetään videotiedostojen esittämiseen, ja AUDIO-elementtiä ääniraitojen esittämiseen verkkosivulla. Materiaalia voidaan tuoda elementteihin paikallisesta tai verkossa sijaitsevasta kansiorakenteesta, kuten myös verkko-osoitteesta. Lähde sijoitetaan media-elementtien src-attribuuttiin. Ilman JavaScriptiä VIDEO-elementillä ei ole mahdollista toistaa videoita verkko-osoitteista, joiden osoite ei suoraan viittaa itse videotiedostoon (esimerkkinä youtube.com/watch?v=YE7VzlLtp-4). Tämä johtuu siitä, että VIDEO-elementin type-attribuuttiin pitää määritellä videolähteen tyyppi, jotta voidaan tarkistaa, onko näytettävän videon tyyppi tuettujen tyyppien listalla. Mikäli tyyppi ei ole tiedossa, sitä ei voida määritellä. Tämän johdosta videon toisto epäonnistuu. Media-elementtien tuettuja tyyppejä ovat WebM, Ogg, MP4, MP3, WAVE PCM, FLAC sekä MSE (Media Source Extensions). (Mozilla Developer Network 2017; World Wide Web Consortium c 2014.)

Tekstisisällön järjestelemiseen HTML5 esitteli elementit MAIN, ARTICLE, SECTION, ASIDE, NAV, HEADER sekä FOOTER. Näiden elementtien on tarkoitus selkeyttää sisällön jakautumista verkkosivulla. Koko sarjasta hyötyvät eniten tiedon esittämiseen perustuvat verkkosivut, kuten esimerkiksi blogit ja uutispalvelut. Elementeistä kaikki muut, paitsi HEADER ja FOOTER, on tarkoitus sisällyttää uuteen MAIN-elementtiin, joka siis käsittää kokoelman ARTICLE-elementtejä. ARTICLE-elementin tarkoitus on sisällyttää useita SECTION-elementtejä, jotka taas sisältävät itse tekstikappaleet. HEADER- ja FOOTER-elementit sijoitetaan sivun ylä- ja alalaitoihin ja ne omaavat omanlaisensa käyttötarkoituksen: HEADER sisältää verkkosivulle olennaisia linkkejä ja valikoita, kun FOOTER sisältää usein yhteystietoja sekä tekijänoikeusilmoituksien kaltaisia huomioita. NAV-elementillä luodaan ja hallinnoidaan linkkien kokoelmaa, ja ASIDE-elementtiä käytetään erilaisten tekstiin liittyvien huomioiden esittämiseen. (Kyrnin 2016; World Wide Web Consortium b 2014.)

3.2 CSS3

CSS-tekniikkaa käytetään HTML-verkkosivujen tyylittämiseen. Tyylittämisellä tarkoitetaan verkkosivujen esitystapaa, kuten graafista ulkoasua. Tyylikielen Cascading -sanalla viitataan mahdollisuuteen, jossa dokumentteihin voidaan vaikuttaa samanaikaisesti usean tyylisäännöstön ja -säännön välityksellä. CSS3 on CSS-tyylikielen uusin versio. (Saarikumpu 2016.)

CSS3 toi mukanaan useita uusia parannuksia ja muutoksia. CSS3:en ja CSS2:en suurin ero on CSS3:en jakautumisessa moduuleihin. CSS3:en on todettu olevan niin suuri ja laaja, että sen valmiiksi saaminen yhtenä kokonaisuutena kestäisi iäisyyden, jonka vuoksi se on jaettu osiin ja näitä osia voidaan kehittää ja hyväksyä omina kokonaisuuksinaan. (Lahtonen 2016.)

CSS3:n merkittävimpiin uusiin ominaisuuksiin lukeutuvat muun muassa

- pyöreät kulmat, varjostus ja kuvista muodostetut rajat
- esitystasot (joilla voi määrätä näytetäänkö enemmän vai vähemmän tietoja halutun esitystason mukaisesti, esimerkiksi WWW-sivu vs. kalvo)
- matemaattisten merkintöjen muotoilu
- uudet valitsimet
- tapahtumankäsittely. (Lahtonen 2016.)

CSS-tyylit otetaan käyttöön HTML-dokumentissa sisällyttämällä HEAD-elementin sisään linkkiviittaus käytetyn CSS-tiedoston sijaintiin. HTML-dokumenttiin on mahdollista sisällyttää sisäisiä tyylimäärityksiä tuomatta lainkaan tyylitiedostoa. Tässä tapauksessa tyylimääritykset tehtäisiin STYLE-elementtien sisään tai upotettuna tyyliteltävän elementin STYLE-attribuuttiin. HTML-dokumenttien sisäinen tyylimäärittelyjen syntaksi on samanlaista kuin CSS-tyylitiedostoissa. Mikäli HTML-dokumenttiin on määritelty sisäisiä tyylittelyjä, ne menevät CSS-tiedoston syntaksin edelle, tarkoittaen että sisäiset syntaksit luetaan ensisijaisesti.

CSS-tiedosto rakentuu valitsimista ja niitä seuraavista aaltosuluin erotelluista tyylimäärittelyistä. Valitsin kohdistuu HTML-elementteihin elementin nimen, attribuutin ja mahdollisesti sen arvon tai erityisesti class- ja/tai id-attribuuttien arvojen mukaan.

Aaltosulkujen lohkoon sijoitetaan tyylimääritykset, jotka valitsimeen valittu elementti halutaan sisällyttää. (Saarikumpu 2016.)

Kuvassa 6 on esimerkki CSS-koodista, jonka avulla tyylitetään pudotusvalikko verkkosivulla. Kuvassa näkyvät tyylimääritykset kohdistuvat pudotusvalikossa käytettäviin luokkiin.

```
.dropbtn:hover, .dropbtn:focus {
    background-color: #3e8e41;
}

.dropdown {
    position: relative;
    display: inline-block;
    float: right;
    margin-top: 60px;
    margin-right: -20px;
}

.dropdown-content {
    display: none;
    position: absolute;
    background-color: orange;
    color: white;
    min-width: 160px;
    box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
    z-index: 1;
    right: 0;
    border-radius: 6px;
}

.dropdown-content a, .dropdown-content p {
    color: white;
    padding: 10px;
    text-decoration: none;
    display: block;
    text-align: center;
    font-size: 12pt;
}

.dropdown-content a:hover {
    background-color: lightblue;
    border-radius: 0 0 6px 6px;
}

.show {
    display:block;
}
```

Kuva 6. Esimerkki CSS-koodista, jonka avulla luodaan määrittelyt verkkosivulla esiintyvälle pudotusvalikolle.

3.3 JavaScript

3.3.1 Mikä on JavaScript

JavaScript on lähinnä verkkosivun toimintojen toteuttamiseen käytettävä kieli. Sillä kirjoitetut ohjelmat eli skriptit suoritetaan käyttäjän selaimessa. JavaScriptiä voidaan käyttää esimerkiksi gallup-kyselyjen luomiseen. Käyttäjä syöttää verkkosivulla vastauksia osio kerrallaan ilman, että verkkosivua täytyy päivittää. Toisena esimerkkinä voisi olla verkkosivulla muotoaan muuttavat tekstilaatikot tai muut muunnokset. (Chapman 2017.)

Usein kysytään, mitä yhteistä on Javalla ja JavaScriptillä. Yhteistä kielillä on ainakin se, että molemmat kuuluvat C-kieliperheeseen, tarkoittaen, että ne omaavat yhteisen kieliopin useimmissa kohdissa. Muuten kielillä ei varsinaisesti ole tekemistä toistensa kanssa, vaikka nimi niin antaisi ymmärtää.

Vuosina 1996 ja 1997 JavaScript vietiin ECMA:iin (European Computer Manufacturers Association) standardoitavaksi. Tarkoituksena oli luoda standardoitu määrittely, joka voitaisiin sisällyttää selaimiin. Standardoitu toteutus sai nimen ECMAScript, jonka katsotaan olevan JavaScriptin virallinen nimi. Toinen tunnettu ECMAScriptiin perustuva toteutus on Adoben ActionScript 3. (Web Education Community Group 2012.)

JavaScript-tuki on käytössä oletuksena selaimessa, mutta käyttäjä voi halutessaan ottaa ominaisuuden pois käytöstä. Tämän päivän verkkosivut sisältävät todella paljon JavaScript-sisältöä, jolloin edellä mainittu toimenpide aiheuttaisi katseltavassa verkkosivussa todennäköisesti toimintahäiriöitä.

JavaScript on laajasti yhteisöjen kehittämä koodikieli, jonka ansiosta verkosta löytyy useita JavaScript-lisäosia (eli plugineja). Kehittäjän on sivua luodessaan käytettävä lisäosan vaatimia muuttujia ja funktiota, joiden avulla sisältöä voidaan luoda. Kuvassa 7 on esimerkki JavaScript-koodista HTML-dokumentissa, jossa sivun otsikkoa ja leipätekstiä muokataan.

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="tyyli.css">
        <title>Otsikko</title>
    </head>
    <body>
        <p>Hei maailma!</p>
        <!-- Tämä on kommentti -->
        <img src="helloworld.jpg" alt="kuva">

        <p id="teksti">Tekstiä</p>

        <script>
            document.getElementById("teksti").innerHTML = "Tämä teksti tulee skriptistä.";
            document.title = "JavaScript rulettaa";
        </script>
    </body>
</html>
```

Kuva 7. Esimerkki JavaScript-koodista HTML-dokumentissa.

3.3.2 JavaScript-kirjastot

JavaScript-kirjastot ovat kokoelmia erilaisista funktioista ja moduuleista, joiden tarkoitus on helpottaa JavaScriptin käyttöä verkkosivulla. Tyypillisesti kirjasto voisi sisältää funktion jollekin toiminnolle, joka vaatisi puhtaalta JavaScriptiltä useita rivejä koodia. Käytännössä kirjastojen käyttöperiaate asettuu aatteelle ”kirjoita vähemmän, tee enemmän”. JavaScript-kirjastoja on olemassa useita eri tarkoituksiin, joista todennäköisesti kuuluisin on HTML/DOM-käsittelyyn tarkoitettu kirjasto jQuery. (w3schools.com a.)

jQuery on kirjasto, jonka tarkoituksena on selkeyttää JavaScriptin kirjoittamista ja sisällyttämistä verkkosivuilla. Sen käyttämiseen ei vaadita syvällistä JavaScriptin tuntemusta, vaan sen käyttämistä voivat harjoittaa kielen aloittelijatkin. jQuerylla voidaan toteuttaa myös AJAX-kutsuja. (jquery-tutorial.net.)

3.4 PHP

PHP on web-sovellusten ohjelmointiin hyvin soveltuva ohjelmointikieli. PHP muistuttaa syntaksiltaan Perl- ja C-ohjelmointikieliä. PHP on avoimen lähdekoodin ohjelmisto. (PHP.net a.)

PHP on alusta alkaen kehitetty erityisesti web-ympäristön sovelluksia varten ja PHP-skriptit voidaankin upottaa suoraan HTML-tiedoston sekaan. Koodi käsitellään palvelimella sivun pyynnön yhteydessä ja käsittelyn suorittaa palvelimelle asennettu niin sanottu PHP-tulkki. Kaikki PHP-koodiin liittyvä prosessointi suoritetaan palvelimessa ja selaimelle lähetetään normaali HTML-sivu, joka sisältää pelkkää HTML-koodia, joten käyttäjän selaimessa ei tarvita erityislaajennuksia. PHP:n avulla verkkosivulle voidaan lisätä erilaisia toimintoja, tarkastaa ja lähettää lomaketietoja ja rakentaa monimutkaisiakin sovelluskokonaisuuksia. PHP on erittäin suosittu ja paljon käytetty, ja siitä on saatavilla useita oppaita, kirjallisuutta ja valmiita koodikappaleita. (Laaksonen 2011; PHP.net a.)

PHP-kieli tukee muuttujia, ehtolauseita, toistosilmukoita, funktioita ja sisäänrakennettuja luokkakirjastoja. PHP:n muuttujat ovat heikosti tyypitettyjä, eli ne voivat saada minkä tahansa tyyppisen arvon kontekstin mukaan. Muuttujatyyppejä ovat muun muassa merkkijono (string), kokonaisluku (int), liukuluku (float), totuusarvo (boolean), taulukko tai matriisi (array) ja olio (object). Muuttujat merkitään dollarimerkillä nimen alussa. Muuttujan nimi voi sisältää kirjaimia, numeroita tai alaviivoja, mutta se ei saa alkaa numerolla. (PHP.net b; Rajala 2009.)

Kuvassa 8 on esitettynä verkkosivulla esiintyvä PHP:lla toteutettava LDAP-autentikointi, eli käyttäjän tunnistaminen. Koodi aloitetaan alustamalla session-eväste ja tallentamalla LDAP-palvelin $adServer-muuttujaan myöhempää käyttöä varten. Ehtolause tarkistaa, onko käyttäjä määritellyt käyttäjätunnuksen ja onko se asetettu evästemuuttujaan. Jos tulos on tosi, koodi etenee LDAP-yhteyden muodostamiseen ja käyttäjätunnuksen testaamiseen palvelimella. LDAP_BINDillä testataan annettuja käyttäjätietoja LDAP-palvelinta vasten ja tuloksen mukaan käyttäjä ohjataan halutulle sivustolle.

```php
<?php
    session_start();

    $adServer = "ldap://DC1.oppi.samk.fi";

    if (isset($_SESSION['login_user'])){

        $ldap = ldap_connect($adServer);
        $ldaprdn = 'OPPI' . "\\" . $_SESSION['login_user'];

        ldap_set_option($ldap, LDAP_OPT_PROTOCOL_VERSION, 3);
        ldap_set_option($ldap, LDAP_OPT_REFERRALS, 0);

        $bind = @ldap_bind($ldap, $ldaprdn, $_SESSION['login_pass']);

        if (!$bind) {
            header('Location: logout.php');
            exit;
            @ldap_close($ldap);
        }

    } else {
        header('Location: login.php');
        exit;
    }
?>
```

Kuva 8. Esimerkki PHP-kielestä: LDAP-autentikointi.

3.5 SQL

SQL on standardoitu relaatiotietokantojen käyttämiseen, rakentamiseen ja hallintaan tarkoitettu kieli. SQL ei ole ohjelmointikieli, vaan lähinnä määrittelykieli. SQL poikkeaa muista kielistä siten, että sen avulla tehtävät määritellään kysymyksellä ”mitä” ennemminkin kuin ”miten”. SQL määrittelee, mitkä tiedot haetaan tai lisätään, ei niinkään miten toiminto tapahtuu. Pääpiirteisesti SQL:n avulla voidaan määritellä tietokannan rakenne, määritellä käyttäjäoikeuksia, lisätä, poistaa tai muuttaa tietoja sekä tehdä erilaisia tiedonhakukyselyjä. (KK Mediat; Oulun seudun ammattiopisto)

Relaatiotietokannoissa tallennetut tiedot (eli data) esitetään tauluina (table). Ne muodostuvat sarakkeista (column) ja riveistä (row) siten, että yksittäinen rivi on sarakkeiden yksittäisten arvojen muodostama joukko. Taulussa on oltava perusavain, jonka arvo on oltava jokaisella rivillä erilainen. Usein yksittäinen rivi vastaa jotakin reaalimaailman kohdetta. Kukin yksittäinen rivi voidaan hakea perusavaimen arvon avulla.

Relaatiotietokannasta tietoa haetaan vain tiedon nimien ja arvojen perusteella, ei koskaan tiedon järjestyksen tai sijainnin mukaan. (Jyväskylän yliopiston IT-tiedekunta ja avoin yliopisto a 2004.)

Relaatiotietokannan tauluissa sarakkeille määritetään aina nimi, tyyppi ja merkkien enimmäismäärä. Nimen ollessa omavalintainen, sarakkeen merkittävimmässä roolissa on sen tyyppi. SQL-kielessä sarakkeen tyyppejä ovat esimerkiksi CHAR (kiinteänmittainen merkkijono), VARCHAR (vaihtuvamittainen merkkijono), INT (kokonaisluku) ja DATE (päivämäärä).

Tyypit CHAR ja VARCHAR on suunniteltu sisältämään merkkijonoja eli mitä tahansa kirjaimia, numeroita ja erikoismerkkejä, jotka kuuluvat taululle määritettyyn merkistöstandardiin. Näiden kahden tyypin ero on niiden varaamassa tallennustilan määrässä. CHAR varaa aina kiinteän, merkkien enimmäismäärän verran tallennustilaa, kun VARCHAR taas varaa tilaa riippuen merkkien määrästä, mutta ei kuitenkaan koskaan enempää kuin enimmäismerkkien verran. Jokainen merkki varaa yhden tavun tallennustilaa sarakkeen tyypistä riippumatta. Tyypeistä INT puolestaan sisältää pelkästään kokonaislukuja ja DATE päivämääriä. (Jyväskylän yliopiston IT-tiedekunta ja avoin yliopisto b 2004.)

Vaikka kaikki käsiteltävä data voitaisiin tallentaa merkkijonoina, on järkevää käyttää tarkoitukselle sopivinta tyyppiä, sillä se helpottaa datan käsittelyä ja sen uudelleenjärjestämistä. Mikäli SQL-lauseessa sarakkeeseen yritetään syöttää enemmän merkkejä kuin mitä sen tyypissä on määritetty, operaatio tuottaa virheilmoituksen. Sama tapahtuu, jos sarakkeeseen yritetään syöttää merkkejä, jotka eivät sovellu sarakkeen tyypille, esimerkiksi INT-tyyppinen sarake ei hyväksy muita kuin kokonaislukuja.

```
1 SELECT merkki, malli, vuosimalli, vari
2 FROM autot
3 WHERE vuosimalli > 2000
4 ORDER BY merkki ASC
```

Kuva 9. Esimerkki SQL-kyselystä.

Kuvassa 9 suoritetaan haku tauluun Autot. Taulusta haetaan sarakkeet merkki, malli, vuosimalli ja vari. Tuloksista näytetään vain ne, joiden vuosimallin arvo on 2000 tai suurempi. Tulokset järjestetään aakkosjärjestykseen merkki-sarakkeen mukaan.

4 TOIMEKSIANTAJAN NYKYINEN TILANNE

Ruskatalojen palveluyhdistys ei ole ajan saatossa muuttanut menettelyään huoltoilmoitusten käsittelyssä. Tarve järjestelmälle on ollut vähäinen, mutta kun kiinteistöjen lukumäärä on kasvanut, on se lisännyt myös huoltojen määrää. Tämä muutos on ajanut yhdistyksen siihen pisteeseen, missä järjestelmän tarve on kasvanut suureksi.

4.1 Huoltoilmoitusten käsittely

Tällä hetkellä yhdistyksen kiinteistöhuolto tulostaa työpäivän alussa paperisen huoltoilmoitusten listan yhdistyksen intrasta ja käy huollettavia kohteita läpi työpäivän aikana. Työpäivän lopuksi huolletut kohteet poistetaan intran huoltolistalta. Tehdyistä töistä ei jää historiaa, eikä tulostettuja papereita säilytetä. Tämä on suuri puute, jonka mahdollinen järjestelmä voisi paikata. Työpäivän aikana mahdolliset muutokset aamupalaverin suunnitelmiin tehdään puhelimen välityksellä tai tauoilla. Tavasta tai paikasta riippumatta asia vaatii aina ihmisten välistä kanssakäyntiä.

4.2 Vaikutus yhdistyksen toimintaan

Historian puute ja kiinteistöjen kasvanut määrä luovat tarpeet järjestelmälle. Nykyisellä menetelmällä ei ole mahdollista seurata tehtyjen huoltotöiden laatua, tekijöitä tai nopeutta. Joskus työpäivän aikana huoltosuunnitelmiin saattaa tulla muutoksia. Sellaisissa tilanteissa kiinteistöhuoltaja valitsee tulostetusta työlistasta uuden työn. Hänen tulisi kuitenkin ensin tarkastaa, onko joku muu jo tekemässä huoltoa. Päällekkäisyyksien välttäminen nykyisellä menetelmällä vaatisi esimerkiksi puhelinsoittoa. Kiinteistöhuoltajat voivat kuitenkin olla työn ääressä tai muuten kiireisiä, jolloin kysyjä jää vaille tietoa ja joutuu tekemään päätöksen ilman varmuutta. Mahdollinen päällekkäi-

syys luo turhaa liikkumista sekä kuluttaa työtunteja. Päivällä ilmaantuneet huollot jäävät työpäivän aikana täysin huomioimatta, koska huoltolistat tarkistetaan vain aamuisin. Listalla saattaa olla pieniä töitä, jotka voitaisiin hoitaa saman päivän aikana, mikäli huolto osuisi suunnitelmien reitille.

4.3 Sähköisen järjestelmän hyödyt

Järjestelmän avulla kiinteistöhuoltajat voivat tarkistaa reaaliajassa avoimia ja tehtyjä huoltoja, jolloin tarve ylimääräiselle puhelinsoitolle tai tapaamiselle kesken työpäivän voidaan välttää. Tehdyt huollot voidaan kuitata suoraan paikan päällä, jolloin muut järjestelmän käyttäjät näkevät huollon etenemisen. Näin voidaan välttyä päällekkäisyyksiltä, joissa kiinteistöhuoltaja siirtyisi huoltokohteeseen tietämättä, että huolto on jo tehty. Järjestelmää käyttäessään he saavat tiedon päivän aikana ilmaantuneista huoltotöistä, jolloin työpäivistä tulee tehokkaampia. Järjestelmän avulla voidaan päästä myös täysin eroon paperisista huoltoilmoituslistoista.

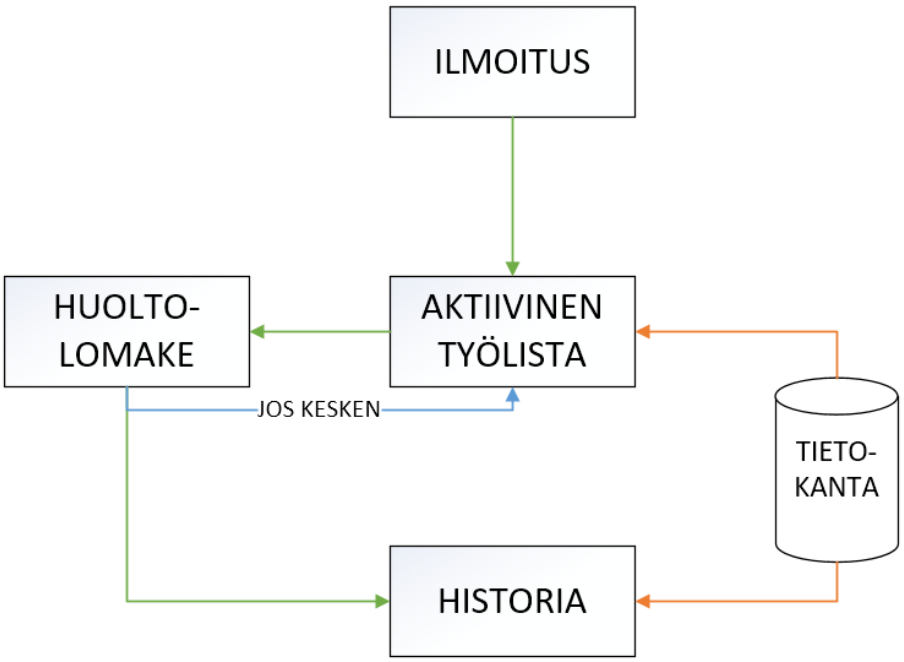
Järjestelmän avulla tehdyistä huoltotöistä jäisi myös historia. Historiasta voidaan tarkistaa huollon ajankohdat, kohde, suorittaja ja ilmoittaja sekä huollon syy. Huoltojen suorittajia voidaan myös tilastoida, joista nähdään esimerkiksi ketkä tekevät huoltoja eniten.

5 SOVELLUKSEN TOTEUTTAMINEN

Projekti aloitettiin pitämällä aloituspalaveri. Aloituspalaverissa määriteltiin sovellukselle tarvittavat ominaisuudet ja käyttäjäkunta. Toteutustapaa ei alleviivattu, mutta sovelluksen tulisi toimia usealla eri päätelaitteella.

5.1 Suunnittelu

Suunnittelupalaverissa käytiin keskustelua, mitä sovellukselta vaaditaan ja minkälainen sen ulkoasu tulisi olla. Sen käyttö tulisi olla ennen kaikkea helppoa ja suoraviivaista. Kuvassa 10 on esitettynä järjestelmän suunniteltu toimintaperiaate.



Kuva 10. Hahmottelu järjestelmän toimintaperiaatteesta.

Yhdistyksen hoitohenkilökunnalla on aktiivisessa käytössään tietokoneet, kun taas kiinteistöhuollon työntekijät käyttävät työnteossaan pääosin älypuhelimia. Tämän vuoksi katsottiin viisaaksi, että toteutettava sovellus voisi olla selainpohjainen. Tämä mahdollistaisi päätelaiteriippumattomuuden ja samalla sovelluksen ylläpitäminen ja kehittäminen olisi johdonmukaisempaa, kun olisi vain yksi järjestelmä mitä henkilökunta käyttää.

Toiminnallisina sovelluksen vaatimuksina olivat listat aktiivisista- ja käsitellyistä huoltoilmoituksista, sekä osio huoltoilmoitusten luomiseen. Laadullisesti sovelluksen toteutukselle asetettiin myös muutama vaatimus: sen käyttöliittymä tulisi olla yksinkertainen, käyttäminen tulisi olla helppoa ja ilmoituslomakkeen täyttäminen ei tulisi kestää minuuttia pidempään. Jotta kiinteistöhuoltajien ei tarvitsisi olla päivittämässä

sivua saapuvien ilmoitusten varalta, katsottiin sovellukselle järkeväksi sisällyttää jonkinlainen ilmoitusominaisuus uusista huoltoilmoituksista. Tämä voitaisiin toteuttaa esimerkiksi sähköpostilla, sillä postit liikkuisivat lähiverkossa palvelimelta toiselle, joten yhteys on viiveetön ja luotettava.

Kysymyksen herätti myös se, että miten käyttäjien autentikointi olisi järkevää toteuttaa? Yhdistyksellä oli jo olemassa oleva Windows-toimialue, joten sen hyödyntäminen sovelluksessa olisi mielekästä. Se jättäisi käyttäjätietokannan suunnittelun ja luomisen tarpeen kokonaan pois, ja käyttäjien hallinta tapahtuisi yhdestä ja samasta paikkaa. Käyttäjien ei myöskään tarvitsisi opetella uusia käyttäjätunnuksia ja salasanoja.

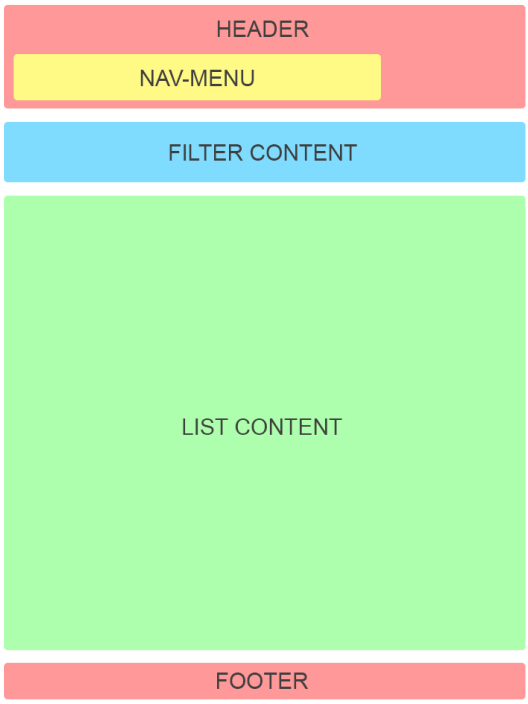
Tässä vaiheessa oli tiedossa, että sivusto tulisi sisältämään HTML-kieltä, palvelinpäässä toteutuvaa prosessointia sekä tietokannan käyttöä. Tekijällä ei tässä vaiheessa ollut paljoakaan kokemusta verkkosivujen luomisesta, joten oli tiedossa myös, että onnistunut toteutus vaatisi paljon asiaan perehtymistä ja uuden opettelemista. Ilman sen suurempia kysymyksiä, projekti laitettiin käyntiin ja sovelluksen toteuttaminen aloitettiin.

5.2 Luonti

Tekijä aloitti työn tekemisen vuoden 2016 kesällä kesäsijaisuuden aikana yhdistyksen työntekijänä. Myöhemmin syksyllä 2016 sovellus siirrettiin kehitettäväksi Satakunnan ammattikorkeakoulun virtuaalikoneympäristöön Bokxiin. Ratkaisu oli hyvä, sillä tekijän itse luodussa laboratorioympäristössä oli jo olemassa oleva Windows toimialue ja sen ympärille luotu palvelimien kokonaisuus, jossa sovellusta oli helppoa testata ja kehittää. Myöhemmin sovellusta päästiin myös testaamaan ulkoverkkoyhteydellä älypuhelimilla Bokxiin sisällytetyn ominaisuuden avulla, jossa jokaisen opiskelijan omassa laboratorioympäristössä on yksi sisäverkon osoite, josta on näkyvyys ulkoverkkoon.

5.2.1 Sivurakenteen hahmottaminen

Koko sovelluksen luonti aloitettiin tyhjästä tekstitiedostosta. Mitään aiempaa järjestelmää ei ollut olemassa, joten sovelluksen toteuttamiseen tekijällä oli vapaat kädet. Toteuttaminen aloitettiin pääsivun, eli aktiivisesta huoltolistan näkymästä. Järjestelmälle luotiin monta erilaista ilmettä, joita tekijä esitteli yhdistyksen työntekijöille ja keräsi palautetta. Eniten suosiota sai asettelumalli, joka on esillä kuvassa 11.



Kuva 11. Hahmotus verkkosivun sisällön asettelusta.

Sivuston toteuttamisen edetessä toiminnallisuuden merkitys korostui. Ominaisuuksille kehitettiin useita erilaisia lähestymis- ja toteuttamistapoja, ennen kuin toiminnallisuus saatiin halutulle tasolle. Jo tässä vaiheessa tekijä oli tutustunut PHP-kieleen aivan uudella tasolla, ja hän käyttikin runsaasti aikaa yksinkertaisen ongelman ratkaisemiseksi. Tämä sama kaava toistui useita kertoja eri ominaisuuksien käyttöönotossa.

Kuvan 12 mukainen, sivuston ensimmäinen vedos ei vielä sisältänyt minkäänlaista käyttäjien hallintaa tai muita lopulliselle järjestelmälle välttämättömiä ominaisuuksia. Suunniteltu toimintaperiaate sen sijaan oli toimintakuntoinen, eli huoltoilmoituksia voitiin laatia, selata huoltolistalta sekä siirtää historiaan. Yleisilme oli kuitenkin ankea, joten seuraavaksi oli aika siirtyä myös sivuston tyylittämisen pariin.

Kuva 12. Sivuston ensimmäinen vedos.

5.2.2 Sivun tyylittäminen

Verkkosivusto luotiin tyhjälle pohjalle, joten myös sen tyylittely tekijän tuli toteuttaa itse. Tämä oli vaativa prosessi, sillä jo pienetkin muutokset vaikuttavat sivuston ulkoasuun ja sisällön asetteluun, sekä itse sovelluksen käytettävyyteen. Oli myös otettava huomioon erikokoiset näytöt ja kuvasuhteet eri päätelaitteissa. Jotta sivusto olisi käytettävyydeltään samanlainen päätelaitteesta riippumatta, sivuston malli oli hyvä olla palstamainen.

Sivuston väritys päätettiin vastaamaan yhdistyksen muita värityksiä eli oranssia, sinistä ja valkoista. Värien haluttiin esiintyvän niin useassa paikkaa kuin mahdollista, jotta yleisvaikutus säilyisi muiden järjestelmien rinnalla tuttuna. Toteutuksessa kaikki linjat ja rajat saivat oranssin värin, siinä missä sivuston tausta, painikkeet ja valikot värjättiin sinisiksi.

5.2.3 Huoltolomake

Huoltoilmoitusten tekemiselle oli vaatimusmäärittelyssä asetettu tavoite, että ilmoituksen täyttäminen tulisi olla mahdollisimman helppoa, nopeaa ja vaivatonta. Tämän tavoitteen saavuttamiseksi huoltoilmoitukseen tulee täyttää etukäteen mahdollisimman paljon tietoa, jota on jo entuudestaan sivustolla saatavissa, kuten esimerkiksi käyttäjän nimi. Mikäli käyttäjä joutuu syöttämään kaiken tiedon aina omasta nimestä vian lähettämisen ajankohtaan saakka, ilmenee tiedoissa ajan saatossa inhimillisiä virheitä eikä kirjoitusasu välttämättä säily yhdenmukaisena.

Ilmoitusten täyttämiseen oli siis sisällytettävä mahdollisimman paljon automaattisia tietoja. Näitä voisivat olla esimerkiksi aika ja ilmoittajan nimi. Aika saadaan helposti PHP:n avulla hakemalla ilmoituksen lähetyksen yhteydessä sen hetkinen aika, käyttämällä PHP:n sisäänrakennettua funktiota date(). Nimi saadaan LDAP-autentikoinnin yhteydessä session-evästeeseen tallentuneesta muuttujasta. Lisäksi sijainnin määrittely ratkaistiin pudotusvalikoilla, joiden sisältö muuttuu dynaamisesti, riippuen aikaisemmasta valinnasta. Dynaamisuus on toteutettu jQueryn AJAX-kutsuilla, jossa onChange-tapahtumissa suoritetaan pudotusvalikkojen tuloksien tuottaminen. Varsinaiset tulokset tuotetaan PHP-tiedostossa, jossa suoritetaan kysely tietokantaan ja haetaan kaikki ne sijainnit, jotka sisältyvät aikaisempaan valintaan. Tulokset sittemmin tulostetaan toistorakenteella pudotusvalikon OPTION-elementteinä. Tämä ominaisuus vaati paljon selvitystä ja testailua ennen kuin toimivuus oli halutulla tasolla. Ominaisuus oli tärkeä, koska yhdistyksellä on useita kohteita ja listoista tulisi todella suuria, mikäli kaikki kiinteistöt, yksiköt ja asunnot tulostuisivat omille pudotusvalikoilleen samanaikaisesti. Tämä tekisi ilmoituksen täyttämisestä sekä vaivalloisempaa että hitaampaa. Kuvassa 13 on nähtävillä pudotusvalikkojen tuloksien tuottamisen JavaScript toteutus.

```
// ------ Näyttää yksiköt valitussa kiinteistössä ------
function naytaYksikko(kiinteisto) {
    $("#drop_asunto").empty();
    $("#asunto").css("pointer-events", "none");
    $("#asunto").fadeTo( "slow", 0.25 );
    $.ajax({
        type: "POST",
        url: "getyksikko",
        data:'kiinteisto='+kiinteisto,
        success: function(data){
            $("#drop_yksikko").html(data);
        }
    });
}

// ------ Näyttää asunnot valitussa yksikössä ------
function naytaAsunto(yksikko) {
    //Tarkistetaan onko "yksikko" valinta Maahovi tai tyhjä...
    if (yksikko != "Maahovi" && yksikkoyksikko != ""){
        $("#asunto").css("pointer-events", "auto");
        $("#asunto").fadeTo( "fast", 1.00 );
        $.ajax({
            type: "POST",
            url: "getasunto",
            data:'yksikko='+yksikko,
            success: function(data){
                $("#drop_asunto").html(data);
            }
        });
    }

    //... jos on, niin piilotetaan asunto valinta
    else {
        $("#drop_asunto").empty();
        $("#asunto").css("pointer-events", "none");
        $("#asunto").fadeTo( "slow", 0.25 );
    }
}
```

Kuva 13. Huoltoilmoituksen pudotusvalikkojen tuloksien tuottamisen JavaScript toteutus. Funktioissa suoritetaan myös valikkojen piilottamista valintojen perusteella.

Kohteiden valinnasta saadaan entistä vikasietoisempi peittämällä valinnat, joihin ei vielä ole tuotettu tuloksia. Pudotusvalikoita myös tyhjennetään aikaisempien valintojen muuttuessa, jotta valintoihin ei jää virheellisiä tietoja. Kuvassa 14 on nähtävillä

piilotustoiminnon JavaScript-toteutus. Sivun latautuessa kolmesta valikosta kaksi jälkimmäistä piilotetaan, ja ne tulevat käytettäväksi sitä mukaa, kun edeltävään valikkoon on tehty valinta.

```
$(document).ready(function piilotaValikot() {
    $("#yksikko").fadeTo( 0 , 0.25 );
    $("#yksikko").css("pointer-events", "none");
    $("#asunto").fadeTo( 0 , 0.25 );
    $("#asunto").css("pointer-events", "none");
});

$(document).ready(function(){
    $("#drop_kiinteisto").change(function() {
        if (($(this).val() != 'Muu') && ($(this).val() != 'Villaruska')) {
            $("#yksikko").css("pointer-events", "auto");
            $("#yksikko").fadeTo( "fast", 1.00 ); //pehmeä häivytys
        }
        else {
            $("#yksikko").css("pointer-events", "none");
            $("#yksikko").fadeTo( "slow", 0.25 );  //pehmeä häivytys
        }
    });
});
```

Kuva 14. Funktiot pudotusvalikkojen piilottamiseen ja esille tuomiseen sivulatauksen jälkeen. Asunto-pudotusvalikko tuodaan esiin myöhemmin, kun yksikkö-pudotusvalikkoon on tuotettu tuloksia.

Käytännössä siis itse ongelman kuvaus on ainoa kenttä, jonka käyttäjä joutuu itse täyttämään. Tähänkin kenttään on mahdollista tehdä suodatus, jossa teksti esimerkiksi käsiteltäisiin aina yhden ulkoasutyypin mukaiseksi. Tälle ominaisuudelle ei kuitenkaan tässä vaiheessa nähdä tarvetta, mutta se on myöhemmin lisättävissä tarpeen mukaan.

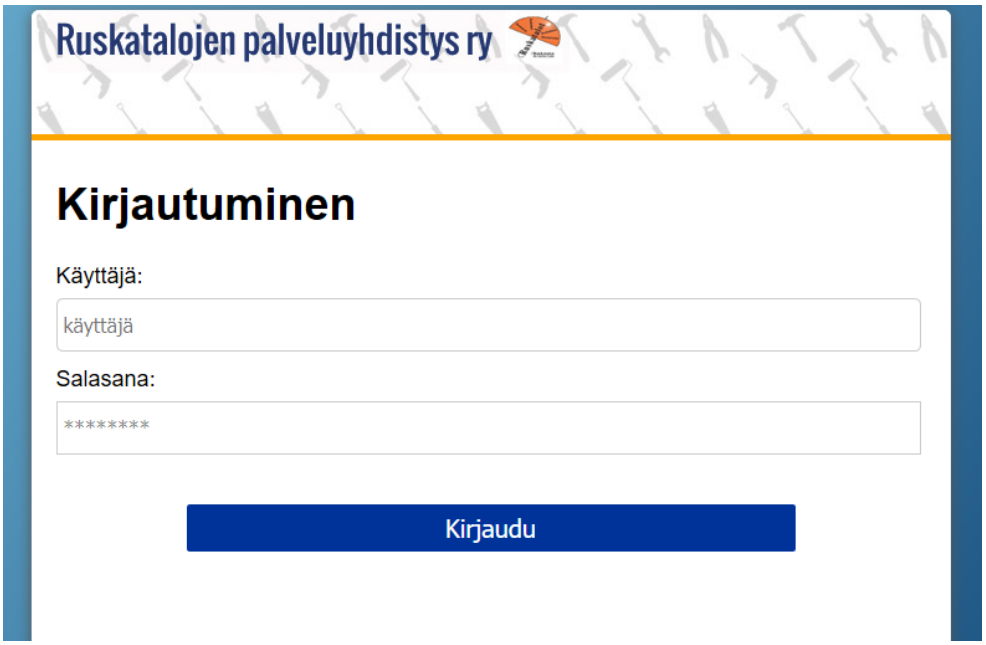
5.2.4 Käyttäjien hallinta

Sivuston toiminnallisuuden hahmottuessa tekijä siirtyi kirjautumisten käsittelyyn. Alkuperäisen suunnitelman mukaan sovellukseen tuli saada toimimaan toimialuepohjainen kirjautuminen, jolloin käyttäjät voisivat käyttää omia, toimialueeseen luotuja tunnuksia huoltojärjestelmän käyttämiseen. Ratkaisu tähän ongelmaan löytyi LDAP:sta. LDAP-verkkoprotokollalla voidaan testata käyttäjän syöttämiä tunnuksia toimialueen ohjainkonetta vasten.

```php
$adServer = "ldap://DC1.oppi.samk.fi";

$ldap = ldap_connect($adServer);
$username = $_POST['username'];
$password = $_POST['password'];

$ldaprdn = 'OPPI' . "\\" . $username;

ldap_set_option($ldap, LDAP_OPT_PROTOCOL_VERSION, 3);
ldap_set_option($ldap, LDAP_OPT_REFERRALS, 0);

$bind = @ldap_bind($ldap, $ldaprdn, $password);


if ($bind) {
    $base_dn = "DC=OPPI,DC=SAMK,DC=FI";
    $filter = "(&(objectClass=user)(sAMAccountName=$username)
            (memberof=CN=Kiinteistohuolto,OU=Kiinteisto,DC=OPPI,DC=SAMK,DC=FI))";
    $attr = array("memberof", "displayname");
    $search_result = ldap_search($ldap, $base_dn, $filter, $attr);
    $entries = ldap_get_entries($ldap, $search_result);
    $huoltaja = $entries["count"] > 0;
    if ($huoltaja){
        $fullname = $entries[0]['displayname'][0];
    }
    else {
        $filter = "(&(sAMAccountName=$username))";
        $attr = array("memberof", "displayname");
        $search_result = ldap_search($ldap, $base_dn, $filter, $attr);
        $entries2 = ldap_get_entries($ldap, $search_result);
        $henkilo = $entries2["count"] > 0;
        if ($henkilo){
            $fullname = $entries2[0]['displayname'][0];
        }
    }
}
else {
        $error = "Käyttäjä tai salasana on virheellinen";
}
```

Kuva 15. PHP:lla toteutettu LDAP-tunnistautumisen käsittely.

Kuvassa 15 suoritetaan LDAP-autentikointi käyttäjän syöttämien kirjautumistietojen perusteella. Aluksi koodissa alustetaan tarpeelliset muuttujat, jonka jälkeen näillä tiedoilla suoritetaan yhteyskysely toimialueen ohjainkoneeseen, jossa käyttäjäkanta sijaitsee. Kirjautuminen epäonnistuu, jos tunnukset eivät täsmää yhteenkään toimialueen käyttäjäkannan käyttäjään ja sovelluksen käyttäjä ohjataan virheilmoituksen kera takaisin kirjautumissivulle. Muuten koodia suoritetaan pykälää eteenpäin, jossa käyttäjästä haetaan olennaisia tietoja, kuten mihin organisaatioryhmiin käyttäjä kuuluu ja mikä käyttäjän nimi on. Näitä tietoja käytetään sivustolla myöhemmin muun muassa oikeuksien määrittelyssä sekä lomakkeiden täyttämisessä. Kuvassa 16 on nähtävillä kirjautumissivun lopullinen ulkonäkö.



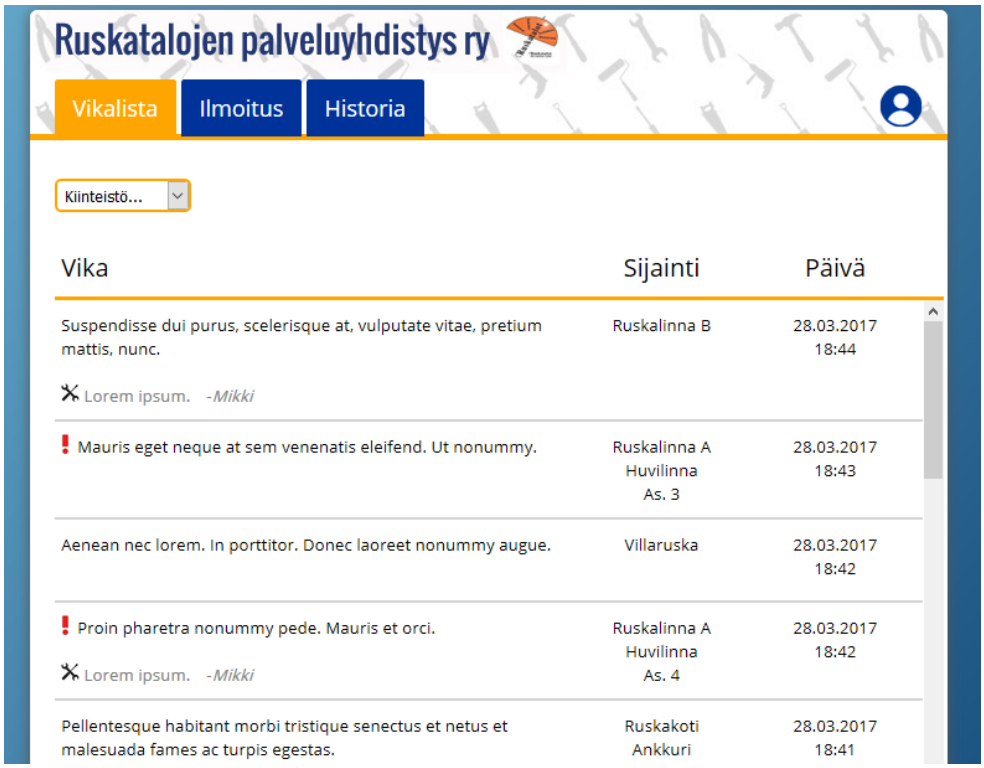
Kuva 16. Sovelluksen kirjautumissivun näkymä.

5.3 Toiminnallisuuden esittely

Sivusto rakentuu käytännössä huoltolistasta, huoltoilmoituslomakkeesta ja huoltohistoriasta. Käyttäjä kirjautuu sivustolle toimialuetunnuksien avulla, jonka jälkeen tunnuksien perusteella toimialueelta noudetaan käyttäjän tietoja sivuston eri toimintoja varten. Kaksi tärkeintä tietoa ovat käyttäjän nimi ja käyttötaso.

5.3.1 Päänäkymä

Kuvasta 17 on nähtävissä sivuston yleisilme ja eri välilehdet kullekin ominaisuudelle. Alempana välilehdistä on pudotusvalikko, jonka takaa aukeaa kiinteistöjen sijainnit, joiden avulla huoltolistaa on mahdollista suodattaa sijainnin perusteella. Kutakin huoltoilmoitusta painamalla aukeaa isompi näkymä, josta selviää esimerkiksi ilmoituksen jättäjä ja muita olennaisia tietoja.



Kuva 17. Sovelluksen aktiivinen huoltolista sekä lopullinen yleisilme.

Aktiivisien huoltoilmoituksien lista tuotetaan PHP-tiedostossa toistolauseen avulla samalla, kun näytettävät olennaisimmat tiedot noudetaan tietokannasta. Huoltoilmoituksia suodatetaan JavaScript -funktion avulla, jonka avulla listalta piilotetaan kaikki muut tulokset, paitsi ne, joiden sijainti vastaa suodattimen arvoa. Sivulla ei käytetä sivutusta, vaan rivit kasaantuvat pitkäksi listaksi. Kohteita ei tule olemaan mitä todennäköisimmin samanaikaisesti niin paljon, että siitä koituisi suorituskyvyllisesti ongelmia.

5.3.2 Huoltolomake

Huoltolistasta riviä painamalla ilmoituksesta avautuu uusi, isompi näkymä, jossa ilmoitus voidaan siirtää historiaan, asettaa keskeneräiseksi tai poistaa.

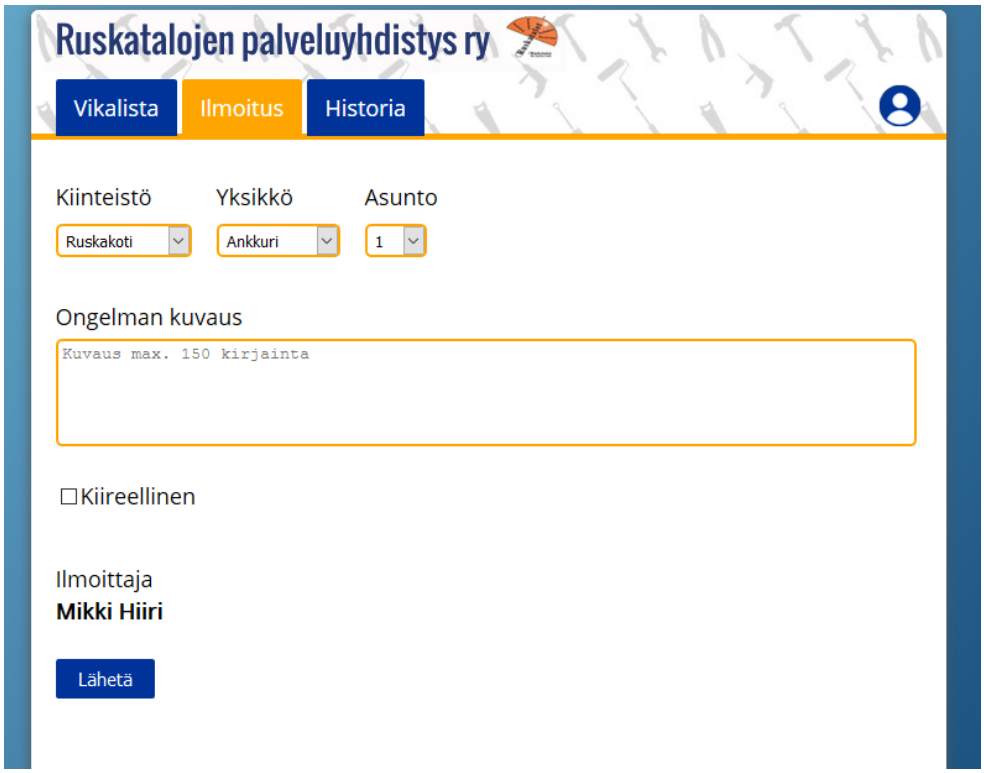
Kuva 18. Huoltoilmoituksen täyttölomake.

Kiinteistöhuoltajat merkitsevät työn tehdyksi, kun ilmoituksen työ on suoritettu ja ongelma korjattu, jolloin se siirtyy historiaan. Kuvan 18 näkymä on erilainen riippuen käyttäjän käyttötasosta. Käytännössä siis hoitohenkilökunta ja muut huoltotiimiin kuulumattomat työntekijät eivät pysty kuittaamaan töitä valmiiksi tai poistamaan ilmoituksia. He voivat kuitenkin seurata ilmoitusten olennaisia tietoja.

Huolto voi kuitenkin jäädä keskeneräiseksi monesta eri syystä. Tällöin huoltoilmoitus voidaan asettaa keskeneräiseksi, jonka yhteyteen on liitettävä selitys huollon keskeytymisestä. Työ voidaan kuitenkin kuitata tehdyksi ja siirtää historiaan ilman pakollista selitettä.

5.3.3 Huoltoilmoituksien luonti

Ilmoitukseen liitetään automaattisesti ilmoittaja ja aika. Työ on myös mahdollista asettaa kiireelliseksi, jolloin huoltolistalla ilmoitukseen liitetään punainen huutomerkki. Lomakkeen lähetyksen yhteydessä käyttäjää pyydetään tarkistamaan ilmoituksen tiedot. Kuvassa 19 on nähtävillä lomakkeiden lähettämisen käyttöliittymä.



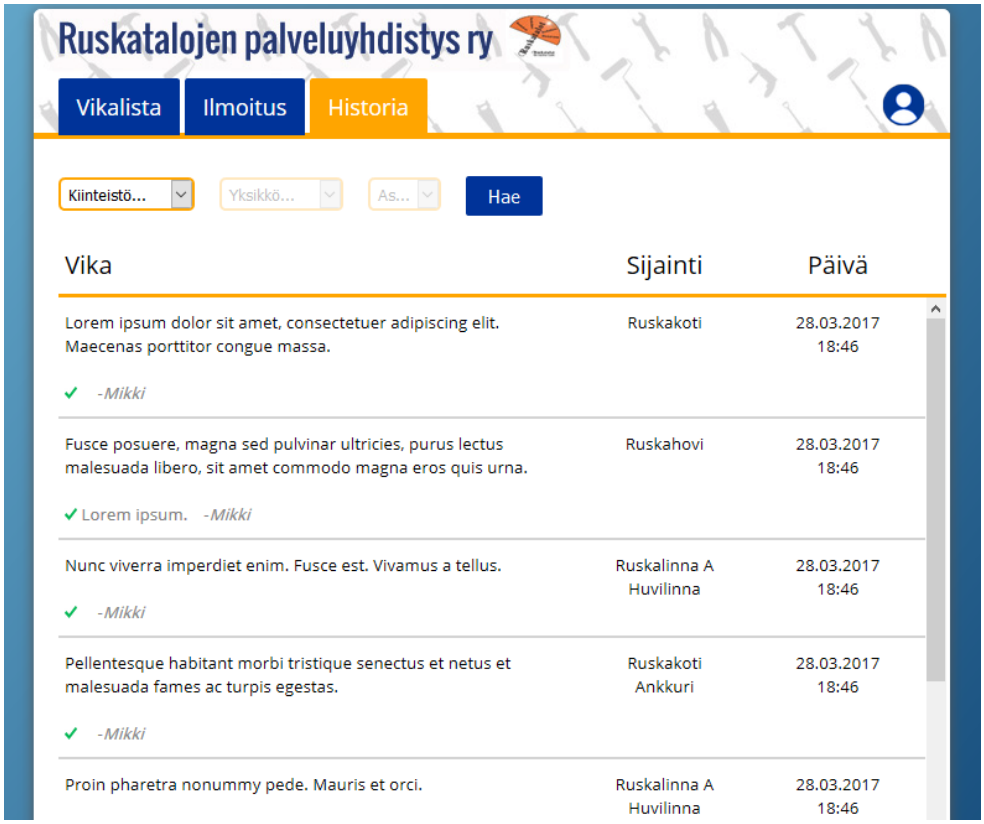
Kuva 19. Huoltoilmoituksen luominen.

Huoltoilmoituksen lähettämisen yhteydessä lähetetään myös sähköposti kaikille huoltoryhmään kuuluville jäsenille. Tämän ominaisuuden avulla kiinteistöhuoltajat pysyvät ajan tasalla saapuvista ilmoituksista, eikä heidän tarvitse käydä manuaalisesti tarkistamassa aktiivisia huoltoilmoituksia.

5.3.4 Historia

Historiasivu vastaa näkymältään ja toiminnaltaan pitkälti aktiivisen huoltolistan näkymää, ainoan käytännön eron ollessa historian kohteiden hakemisessa. Historian puolella listalla näytetään sivun latautuessa vain uusimmat 50 kohdetta. Tämän vuoksi

suodatus tapahtuu uudella tietokantakyselyllä, jossa arvoina toimivat pudotusvalikkoihin määritetyt arvot. Tässä pudotusvalikkojen toiminta on sama, kuin mikä aikaisemmin huoltoilmoituksen luonnissa esitelty toiminta. Historian kohteista aukeaa myös huoltolistan kaltaisesti oma lomake, josta voidaan tarkemmin tarkastella huollon tietoja. Kuvassa 20 on vielä esitettynä huoltohistorian näkymä.



Kuva 20. Historiasivu, jossa on lista tehdyistä huolloista.

5.4 Sovelluksen kehitys

Sovellusta tullaan tulevaisuudessa ylläpitämisen lisäksi kehittämään paremmin palvelevaksi. Sovelluksen oltua tuotantokäytössä sivustoon tullaan lisäämään ominaisuuksia ja toimintamenetelmiä tullaan parantelemaan käyttäjien palautteen mukaisesti.

Yksi jo valmiiksi ehdotettu kehitysaskel olisi sivu, joka käsittää kiinteistöjen ja asuntojen pohjapiirustukset. Ominaisuuden avulla voitaisiin ensin hakea esimerkiksi tietyn kiinteistön asunto, jonka jälkeen sovellus palauttaa asunnosta pohjapiirustuksen, huoneistotiedot sekä viimeisimmät huoltotapahtumat. Huoltotapahtumiin voitaisiin myös

manuaalisesti lisätä ylläpidollisia huoltoja, kuten seinien maalaus. Vastaavasti koko kiinteistön osiota voitaisiin käyttää vuosihuoltojen aputyökaluna.

6 YHTEENVETO

Tämän työn tekeminen vaati paljon selvittelyä ja oman osaamisen soveltamista, ja kokonaisuudessaan projekti on kestänyt lähes vuoden. Sovellus kokonaisuudessaan on käyttövalmis, mutta siitä tulee aina löytymään parantamisen varaa.

Projektin aikana tekijä koki monia ylä- sekä alamäkiä. Oli hetkiä, kun ominaisuuksien käyttöönotto ja luonti olivat sujuvaa, kun taas yhden yksinkertaisen oloisen toiminnon toteuttaminen kesti viikkoja. Tekijä koki sovelluksen kehittämisen palkitsevaksi, kun sovellus alkoi ajan saatossa hahmottua oikeaan suuntaan ja toiminnallisuudet paranivat.

Näin jälkiviisaana voi todeta, että projekti olisi varmasti ollut huomattavasti sujuvampi, mikäli alussa olisi toteutettu syvempi suunnittelu- ja määrittelyvaihe. Koodirivien määrä kasvoi projektin edetessä huomattavasti suuremmaksi kuin mitä oli odotettu, ja koodin hallinta lähtikin nopeasti käsistä. Sovelluksen ominaisuuksien nimeämispolitiikka koodissa on osittain sekavaa ja se vaatiikin tulevaisuudessa siivoustyötä.

Tämä projekti ja sen tuotoksen lopputulos olivat kuitenkin ammatillisen kehittymisen kannalta tärkeitä. Koulussa opitut menetelmät sekä omakohtaisen osaamisen ja opettelun yhdistäminen luovat lopussa kokonaisuuden, jonka avulla tämän työn kaltaisia projekteja saatetaan valmiiksi. Rohkeasti vastaanotettu työ aiheutti kauhunhetkiä, jolloin projektin valmistumisen ajankohdasta ei ollut varmuutta. Valmiiksi saatettu työ kuitenkin nostaa mielialaa ja antaa paremmat eväät seuraaviin haasteisiin.

LÄHTEET

Chapman, S. 2017. What is JavaScript? Viitattu 26.2.2017. http://javascript.about.com/od/reference/p/javascript.htm

Duncan, R. 2015. What is HTML? Viitattu 21.2.2017. http://www.simplehtml-guide.com/whatishtml.php

jquery-tutorial.net. What is jQuery? Viitattu 25.3.2017. http://www.jquery-tutorial.net/introduction/what-is-jquery/

Jyväskylän yliopiston IT-tiedekunta ja avoin yliopisto a. 2004. Relaatiotietokannat. Viitattu 4.3.2017. http://appro.mit.jyu.fi/doc/tiedonhallinta/tietokannat/index1.html

Jyväskylän yliopiston IT-tiedekunta ja avoin yliopisto b. 2004. Relaatiotietokantojen peruskäsitteet. Viitattu 4.3.2017. http://appro.mit.jyu.fi/doc/tiedonhallinta/tietokannat/index2.html

KK Mediat. Johdatus SQL:n maailmaan. Viitattu 4.3.2017. http://www.2kmediat.com/sql/alkeet.asp

Kyrnin, J. 2016. What's New in HTML5. Viitattu 3.4.2017. https://www.thoughtco.com/whats-new-in-html5-3467974

Laaksonen, A. 2011. Oppaat: PHP-ohjelmointi: Osa 1 – Johdanto. Viitattu 6.3.2017. http://www.ohjelmointiputka.net/oppaat/opas.php?tunnus=php_01

Lahtonen, T. 2016. CSS3. Viitattu 25.3.2017. http://appro.mit.jyu.fi/web-sovellukset/luennot/css3/

Mozilla Developer Network. 2017. Media formats supported by the HTML audio and video elements. Viitattu 3.4.2017. https://developer.mozilla.org/en-US/docs/Web/HTML/Supported_media_formats

Nieminen, H. 2016. Luentomateriaali: Palvelinohjelmointi Osa 2.

Oulun seudun ammattiopisto. MySQL-tietokannan käyttö ja SQL-kielen perusteet. Viitattu 4.3.2017. http://www.okol.org/verkkokurssit/datanomi/tietojarjestelmien_kehittaminen/tiedonhallintajarjestelmat/sqlkieli.htm

PHP.net a. What is PHP? Viitattu 21.2.2017. http://php.net/manual/en/intro-whatis.php

PHP.net b. Language Types Introduction. Viitattu 10.4.2017. http://php.net/manual/en/language.types.intro.php

Rajala, J. 2009. Ohjelmointi: PHP & MySQL perusteet. Viitattu 6.3.2017. http://www.php-perusteet.com/

Ruskatalojen palveluyhdistys ry a. Yhdistys. Viitattu 3.4.2017. http://ruskatalot.fi/yhdistys2

Ruskatalojen palveluyhdistys ry b. Ruskakoti. Viitattu 3.4.2017. http://ruskatalot.fi/ruskakoti2

Saarikumpu, O. 2016. Johdanto CSS-tyyliehdotuksien käyttöön Web-sivuilla. Viitattu 21.2.2017. http://weppipakki.com/css/tekstit/cssintro.htm

tutorialspoint. JSON with Ajax. Viitattu 5.4.2017. https://www.tutorialspoint.com/json/json_ajax_example.htm

Web Education Community Group. 2012. A Short History of JavaScript. Viitattu 5.4.2017. https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript

World Wide Web Consortium a. 2014. HTML5: 1 Introduction. Viitattu 21.2.2017. https://www.w3.org/TR/html5/introduction.html

World Wide Web Consortium b. 2014. HTML5: 4.3 Sections. Viitattu 3.4.2017. https://www.w3.org/TR/html5/sections.html

World Wide Web Consortium c. 2014. HTML5: 4.7 Embedded content. Viitattu 3.4.2017. https://www.w3.org/TR/html5/embedded-content-0.html

w3schools.com a. jQuery Introduction. Viitattu 26.2.2017. https://www.w3schools.com/jquery/jquery_intro.asp

w3schools.com b. HTTP Methods: GET vs. POST. Viitattu 16.3.2017. https://www.w3schools.com/tags/ref_httpmethods.asp

w3schools.com c. AJAX Introduction. Viitattu 16.3.2017. https://www.w3schools.com/xml/ajax_intro.asp

4psa.com. 2013. An Intro Into Single Page Applications (SPA). Viitattu 16.3.2017. https://blog.4psa.com/an-intro-into-single-page-applications-spa/