VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Miika Mäkynen

RAPORTOINTIOHJELMISTON KEHITYS

Tekniikka
2021

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Miika Mäkynen
Opinnäytetyön nimi	Raportointiohjelmiston kehitys
Vuosi	2021
Kieli	suomi
Sivumäärä	33
Ohjaaja	Anna-Kaisa Saari

Opinnäytetyön tarkoituksena oli luoda kokonaisvaltainen ohjelmisto erilaisten raporttien tekoon. Ohjelmiston tuli mahdollistaa käyttäjälle raporttipohjien luominen websivustolla sekä näiden joustava käyttö hybridiapplikaatiossa. Kehityksessä tuli ottaa huomioon myös yrityksen jatkosuunnitelmat sovellukselle.

Työ toteutettiin vaasalaiselle it-yritykselle käyttäen heillä ennestään käytössä olevaa sisällönhallintajärjestelmää. Ohjelmistokehityksen ohessa työssä tutustuttiin yleisesti sisällönhallintajärjestelmiin sekä käytössä olleen järjestelmän eri ohjelmistokehyksiin ja kirjastoihin. Lisäksi kehitettävä sovellus suunniteltiin ohjelmistomallinnuksen eri kaavioilla ennen toteutusvaihetta.

Opinnäytetyön ohjelmisto kehitettiin yrityksen yhdessä rakennustarkastajan kanssa muodostamien vaatimusmäärittelyjen sekä suuntaa antavien ulkoasusuunnitelmien pohjalta.

Tuotettu ohjelmisto vastasi kaikilta osin vaatimusmäärittelyihin ja lopullinen sovellus pääsi käytännön testaukseen työn päätyttyä. Ohjelmiston helppo käytettävyys sekä rajattomat käyttömahdollisuudet saivat erityistä kiitosta yrityksen edustajalta.

Avainsanat	raportointi, sisällönhallintajärjestelmä, ohjelmistotuotanto, sovelluskehitys, web-ohjelmointi

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikka

ABSTRACT

Author	Miika Mäkynen
Title	Developing Reporting Software
Year	2021
Language	Finnish
Pages	33
Name of Supervisor	Anna-Kaisa Saari

The purpose of the thesis was to develop software for generating various reports. The thesis was ordered by an IT company from Vaasa, at the initiative of a building inspector. Although reporting applications can be found in the market, they are often only web based. The aim of the project was to make a hybrid application where the user can fill in reports.

The software was implemented with a content management system that enables the creation of a web interface and a hybrid application. The same content management system is used by the company in other projects. During this thesis content management systems in general were studied, as well as the software frameworks and libraries used in the used system. The software was also designed with UML diagrams, before implementation.

The result of the thesis was a web interface where the user can create report templates and a hybrid application to fill in the created templates. The software met all the requirements set by the company.

After the thesis, the software will be tested in practice and further developed based on the feedback received. During the thesis, preparations were also made for the software to integrate with the company's other systems and the electronic signature service.

Keywords	Reporting, content management system, software development, application and web development

SISÄLLYS

KUVALUETTELO

KÄSITE- JA LYHENNELUETTELO

Luokka: Laajennettava ohjelmakoodimalli, jonka pohjalta oliot luodaan /1/.

Olio: Luokasta luotu uusi ilmentymä /1/.

Attribuutti: Muuttuja, joka jokaisella luokasta muodostetulla oliolla on /1/.

Metodi: Luokan sisällä oleva itsenäinen toiminto.

Alaluokka: Luokka, joka saa toiselta luokalta ominaisuuksia.

Ohjelmistokehys: Tarjoaa rungon ja valmiita komponentteja ohjelmiston kehitykseen.

HTTP: Hypertext Transfer Protocol. protokolla, jolla siirretään tietoa verkossa.

HTML: Hypertext markup language. Verkkosivujen staattisen rakenteen merkintäkieli.

CSS: Cascading Style Sheets. Verkkosivujen muotoiluun käytetty tyyliohjeistuskieli.

JavaScript: Komentosarjakieli, joka mahdollistaa web-sivujen dynaamisen toiminnan.

PHP: PHP: Hypertext Preprocessor. Palvelinpuolella käytetty ohjelmointikieli.

Istunto: Pysyvä yhteys palvelimen ja päätelaitteen välillä.

MIT-lisenssi: Ohjelmistolisenssi, jolla sallitaan lisensoidun tuotteen käyttö, muokkaus ja kopiointi rajoituksetta, kunhan alkuperäinen lisenssiteksti säilytetään lähdekoodissa /2/.

Ajax-kutsu: Selainpuolelta kutsutaan palvelinpuolta taustalla ilman, että koko verkkosivua ladataan.

GitLab: Versionhallintajärjestelmä

Nimiavaruus: Tyyli ohjelmoinnissa kapseloida toisiinsa liittyviä asioita yhteen.

Factory: AngularJS-luokkaa muistuttava palvelu, joka voi säilyttää attribuutteja ja metodeja käytettäväksi kaikkialta sovelluksessa. /3/

1 JOHDANTO

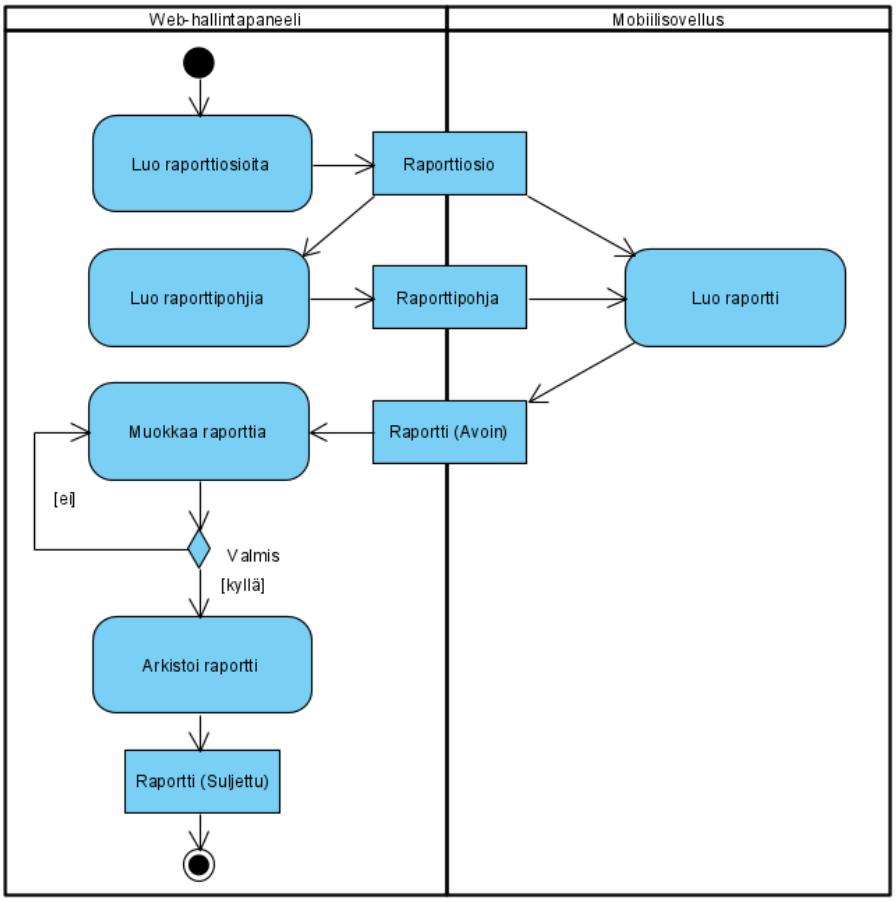
Opinnäytetyön tavoitteena on suunnitella ja toteuttaa raporttien luontiohjelmisto vaasalaiselle startup-yritykselle. Ohjelmiston pääasiallisena kohderyhmänä pidetään rakennustarkastajia, joita on myös ohjelmiston ideoinnissa mukana. Ohjelmiston tulee kuitenkin soveltua kaikenlaisten dokumenttien tekoon. Markkinoilta löytyy muutamia vastaavanlaisia ohjelmistoja, mutta yrityksen tarkoituksena on integroida työssä kehitetty ohjelmisto omaan kiinteistönhallintajärjestelmäänsä tulevaisuudessa.

Yritys on koostanut ohjelmiston vaatimusmäärittelyn yhdessä kohderyhmän edustajan kanssa, ja näiden vaatimusten pohjalta ohjelmisto suunnitellaan. Lisäksi valmiin ohjelmiston karkeat ulkoasusuunnitelmat ovat valmiina, mikä osaltaan auttavat myös teknisten ratkaisujen valintaa. Ohjelmistolla käyttäjän tulee voida luoda web-hallintapaneelissa raporttipohjia, joita käyttäjä voi myöhemmin täyttää joko verkkoselaimella tai mobiiliapplikaatiolla. Täytetyt raportit tulee lopuksi tallentaa PDF-muodossa.

Työ toteutetaan avoimen lähdekoodin sisällönhallintajärjestelmällä, joka mahdollistaa applikaation ja webportaalin teon. Sama järjestelmä on yrityksellä käytössä myös muissa projekteissa, joten sen käyttö myös tässä työssä helpottaa ohjelmiston ylläpitoa siinä vaiheessa, kun ohjelmisto on luovutettu yritykselle.

2 MÄÄRITTELY JA SUUNNITTELU

Ohjelmiston kehitys aloitetaan tapaamisella yrityksen edustajan kanssa. Heti ensimmäisessä tapaamisessa alkuperäinen suunnitelma kokonaisten raporttipohjien luomisesta muuttuu. Työtä lähdetään toteuttamaan siten, että käyttäjä luo hallintapaneelissaan raporteille valmiita osia, joita käyttäjä voi raporttia tehdessä yhdistellä tarpeen mukaan. Tällä saavutetaan huomattavasti joustavampi sekä käyttäjäystävällisempi lopputulos. Lisäksi käyttäjä voi tallentaa kokonaisia raporttipohjia tallentamistaan osista, jotta raportin teko olisi mahdollisimman helppoa. Tuotettava ohjelmisto koostuu kahdesta osasta, web-pohjaisesta hallintapaneelista sekä hybridiapplikaatiosta. Sovelluksen käyttäjälle luodaan ylläpitäjien toimesta käyttäjätilit molempiin ympäristöihin. Kuvassa 1 hahmotetaan ohjelmiston toimintaa loppukäyttäjän näkökulmasta aktiviteettikaavion avulla.



Kuva 1. Ohjelmiston aktiviteettikaavio

Käyttäjä voi luoda esimerkiksi rakennustarkastajan tarpeisiin erilaisia osioita erilaisten huoneiden raportointiin. Yhtenä esimerkkinä osio nimeltä Makuuhuone, joka sisältää vapaatekstikentän, kuvanlisäysosion sekä alasvetovalikon. Taulukossa 1 listataan eri syötetyypit, joista raportin osat voi koota.

Taulukko 1 Osioiden syötetyypit.

Tyyppi	**Selite**
Otsikko	Vakioteksti, joka tulostetaan raportille
Tekstialue	Vapaa tekstikenttä, jota voi muotoilla hallintapaneelista
Numero	Syöte, joka hyväksyy vain numeroita sekä desimaalierottimen
Alasvetovalikko (Select)	Valikko, josta voi valita yhden tai useamman ennalta määritellyn arvon.
Valintanappi (Radio)	Ryhmä valintanappeja, joista voi valita yhden
Asetusnappi (Checkbox)	Asetusnappi, jolle määritetään 2 arvoa.
Kuva	Kenttä, johon voi syöttää yhden tai useamman kuvan, joko laitteen muistista tai ottaa kuvan laitteen kameralla.
Päivä ja aika	Kenttä, josta saa valita joko päivämäärän, ajan tai molemmat käyttäjän valitsemassa muodossa.

Raporttiosioiden luonnin jälkeen käyttäjä voi yhdistellä osioita valmiiksi raporttipohjiksi. Esimerkiksi käyttäjä voi luoda rakennustarkastustyyppisen raporttipohjan nimellä 2h+k, johon hän lisää kaksi Makuuhuone-osiota sekä yhden Keittiö-osion. Raporttipohjien käyttö nopeuttaa entisestään käyttäjän työtä tarkastettavassa kohteessa.

Raportit luodaan hybridiapplikaatiossa, jossa käyttäjä voi vapaasti yhdistellä luomiaan osioita sekä pohjia. Ulkoasusuunnitelmien perusteella osiot käsitellään yksitellen omassa näkymässään, jossa täytetään osiolle valitut syötetyypit. Lisäksi käyttäjä voi syöttää raportille nimen, tyypin sekä sijainnin näkymässä, johon myös täytettävät osiot listataan.

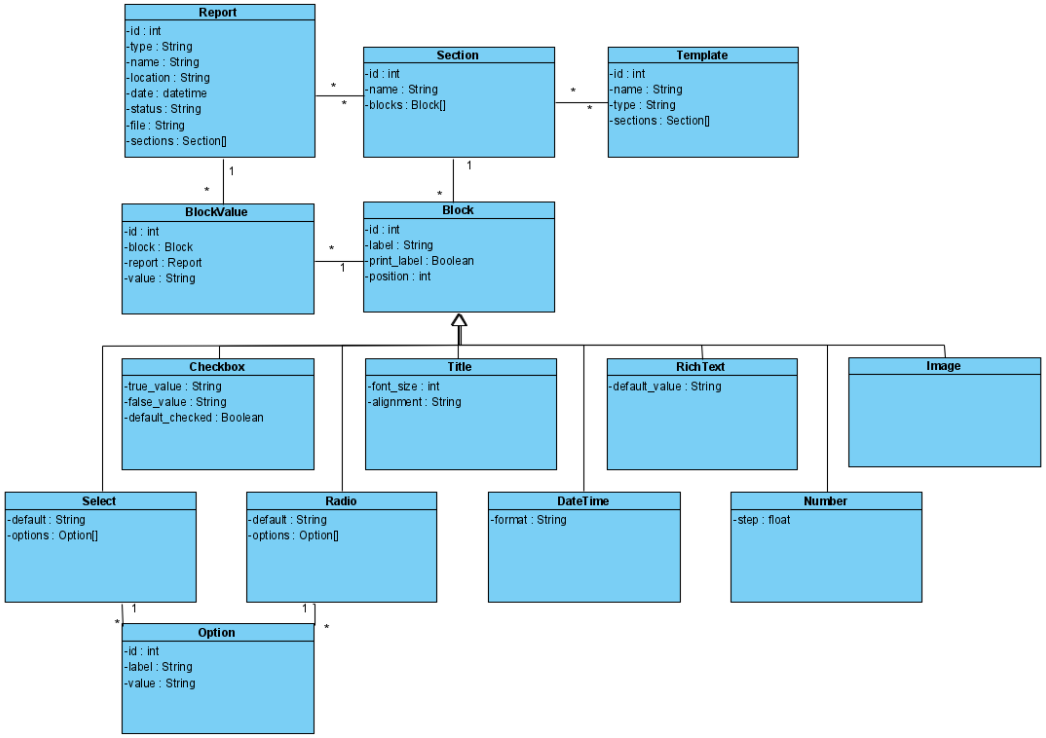
Raportti viimeistellään webportaalissa, jossa myös osiot ja pohjat on luotu. Raportin tekstikentille tehdään tekstieditori, jolla voidaan kevyesti muotoilla tekstejä. Lisäksi applikaatiossa otettuja kuvia täytyy pystyä käsittelemään. Viimeisten muokkausten jälkeen raportti arkistoidaan, jolloin raportti tallennetaan PDF-muodossa palvelimelle. Arkistoitua raporttia ei voi muokata, mutta se on tarkasteltavissa hallintapaneelissa.

Ensimmäisen kokouksen tuomien muutosten sekä aikaisempien vaatimusmäärittelyjen perusteella työn teknisiä ominaisuuksia aletaan suunnitella. Ohjelmiston tekninen rakenne mallinnetaan Visual Paradigm -ohjelman luokka- sekä ER-kaavioilla.

2.1 Luokkakaavio

Ensimmäisenä työ suunnitellaan luokkakaavio tasolla. Luokkakaavio on graafinen UML-kaavio (Unified Modeling Language), jolla kuvataan ohjelmiston staattista rakennetta. Kaavion avulla visualisoidaan ohjelmiston luokkia sekä näiden attribuutteja, metodeja ja suhteita. Luokkakaaviolla saadaan kattava kuva siitä, mitä muuttujatyyppejä luokan metodit odottavat parametreina ja mitä tyyppiä ne palauttavat. Vastaavasti kaavio kertoo luokkien attribuuttien muuttujatyypit. /4/

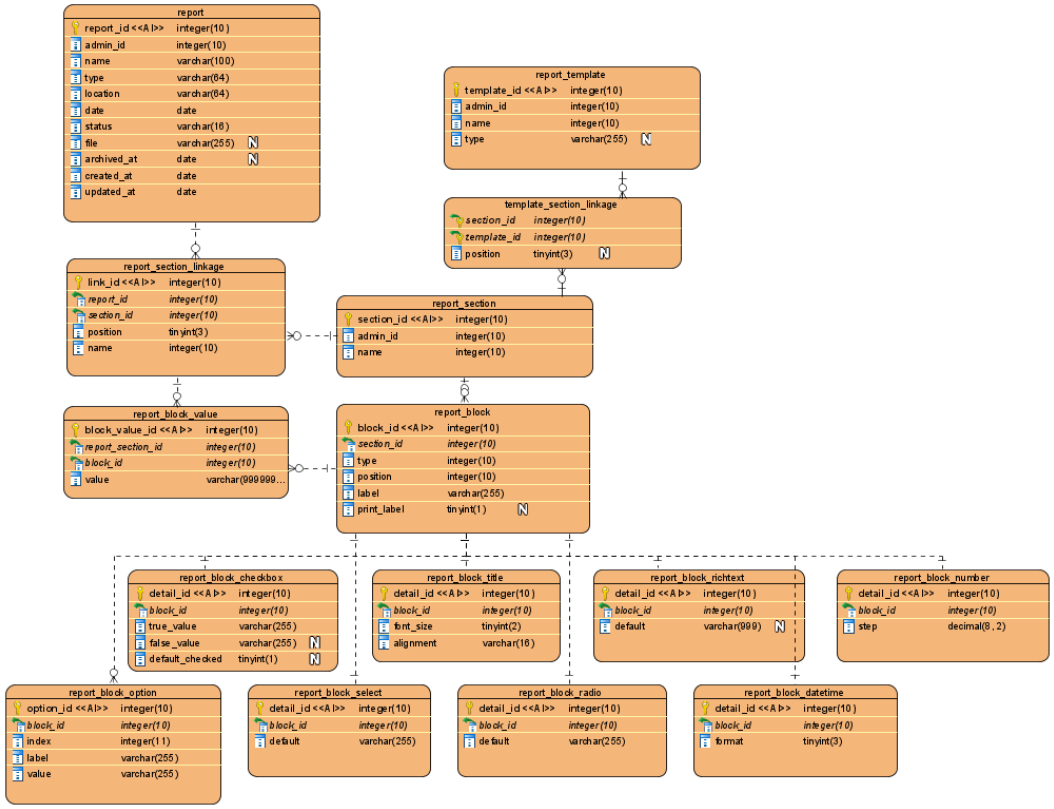
Toisin kuin normaalisti luokkakaavioilla, tässä vaiheessa ei mietitä luokkien toimintoja, mutta attribuutit sekä luokkien väliset suhteet kirjataan ylös. Lopputuloksena muodostuu viisi itsenäistä luokkaa, yksi yläluokka sekä kahdeksan raportin eri osia kuvaavaa alaluokkaa. (**Kuva 2.**)



Kuva 2. Ohjelmiston luokkakaavio

2.2 ER-kaavio

Toinen osa työn suunnitelmaa on ER-kaavio (Entity-Relationship Diagram). Sitä käytetään usein tietokantarakenteiden kuvaamiseen. Kaaviolla saadaan tarkemmin suunniteltua eri luokkien vaatimien tietokantataulujen tietotyypit sekä koot. Lisäksi ER-mallilla hahmotetaan tarvittavat linkkitaulut eri luokkien välille. Ohjelmiston alustava tietokanta koostuu yhteensä 15 taulusta, joista kaksi on linkkitauluja. (**Kuva 3.**)



Kuva 3. Ohjelmiston ER-kaavio

3 KÄYTETYT TEKNOLOGIAT

Yrityksen sisällönhallintajärjestelmä käyttää palvelinpuolella kustomoitua Zend Framework -ohjelmistokehyksen ensimmäistä versiota. Sekä web-hallintapaneeli että mobiiliapplikaatio käyttävät samaa palvelinta. Hallintapaneelin käyttöliittymä kehitetään HTML-merkintäkielellä, CSS-tyyliohjeilla sekä JavaScript-komentosarjakielellä. Nämä ovat yleisimmin käytössä olevat verkkosivujen ohjelmointikielet /5/. Järjestelmän selainpuolella on käytössä lukuisia ohjelmistokehyksiä ja kirjastoja, joista suurimpina CSS- ja JavaScript-pohjainen ohjelmistokehys Bootstrap sekä JavaScript-kirjasto jQuery.

Applikaatiossa on käytössä Ionic-ohjelmistokehyksen ensimmäinen versio, joka puolestaan perustuu AngularJS- sekä Apache Cordova -ohjelmistokehyksiin. Vaikka järjestelmässä on osittain sovelluskehysten vanhoja versioita käytössä vastaavat ne hyvin myös nykypäivän vaatimuksiin, sillä niitä kaikkia on kustomoitu runsaasti.

3.1 WWW-sisällönhallintajärjestelmä

WWW-sisällönhallintajärjestelmä eli WCMS (Web Content Management System) on websivujen hallintaan tarkoitettu sovellus. Nykyään yli 60 % kaikista internetin sivuista on rakennettu käyttäen sisällönhallintajärjestelmää ja osuus kasvaa jatkuvasti. Suosituin järjestelmistä on WordPress, jolla on hallussa 64 % markkinaosuus sisällönhallintajärjestelmistä. /6/

WWW-sisällönhallintajärjestelmä mahdollistaa verkkosivujen teon täysin ilman tuntemusta ohjelmoinnista. Ne tarjoavat valmiita osia ja toimintoja, joista käyttäjä voi rakentaa itselleen sivut graafisella käyttöliittymällä helposti. Tästä poiketen opinnäytetyön tilannut yritys ohjelmoi kaikki käyttämänsä ohjelmistot itse hyödyntäen sisällönhallintajärjestelmän tarjoamaa arkkitehtuuria ja käyttäjähallinnan raameja. Näistä on räätälöity omiin tarpeisiin soveltuva kokonaisuus.

3.2 Palvelinpuolen ohjelmointi

Palvelinpuolen eli back end -ohjelmoinnilla tarkoitetaan sovelluksen palvelimella toteutuvan osan kehitystä. Palvelinpuolella toteutetaan esimerkiksi käyttöoikeuksien tarkistus, kommunikointi tietokantaan sekä käyttäjän syötteiden tarkistus. Nämä on hyvä toteuttaa palvelimella, sillä käyttäjä ei näe tapahtuvia prosesseja eikä voi vaikuttaa niihin, toisin kuin toimintojen toteutuessa käyttäjän päätelaitteella. Kommunikaatio palvelimen ja päätelaitteen välillä toteutuu HTTP-protokollalla.

3.2.1 Zend Framework

Zend Framework on avoimen lähdekoodin web-sovelluskehys PHP-ohjelmointikielelle. Siinä on kymmeniä löyhästi kytkettyjä komponentteja aina istuntojen hallinnasta sähköpostien lähetykseen. Komponenttien ollessa löyhästi kytkettyjä, voidaan niitä käyttää myös osittain itsenäisesti, vaikka ohjelmistossa ei olisi käytössä kokonaan Zendin kehystä. Zend implementoi myös MVC-arkkitehtuuria (Model-View-Controller), jolla saadaan ohjelmistolle perusrakenne. /7/

MVC-arkkitehtuurissa ohjelmisto on jaettu kolmeen osaan: malliin (model), näkymään (view) ja käsittelijään (controller). Ohjelman näkymäosat sisältävät kaiken mitä käyttäjälle näytetään. Tähän Zendissä käytetään Zend_Layout-moduulin luokkia. Näille käsittelijäluokka lähettää tietoa, mitä käyttäen näkymä tulostetaan. Käsittelijät on toteutettu Zend_Controller-moduulin luokilla. Varsinaista mallien toteutukseen tarkoitettua luokkaa ei Zendissä ole, mutta Zend_Db_Table-luokalla voidaan sitoa tietyn luokan objektit tiettyyn tietokantatauluun. /7/

3.3 Verkkosivuohjelmointi

Toteutettaessa sovelluksen käyttäjälle näkyviä osia puhutaan selainpuolen eli front end -ohjelmoinnista. Se pitää sisällään palvelimella saatujen tietojen esityksen päätelaitteen näytölle, reagoinnin käyttäjän toimiin sekä tietojen lähetyksen ja pyytämisen palvelimelta.

3.3.1 Bootstrap

Maailman suosituin responsiivisten sivujen front end kirjasto, Bootstrap (alun perin Twitter Blueprint), luotiin vuonna 2010 Twitterin toimesta. Vuotta myöhemmin tämä mobile first -periaatteella rakennettu kirjasto julkaistiin avoimen lähdekoodin projektina MIT-lisenssin alla. /8/

Mobile first -suunnittelussa sivuston elementit muotoillaan ensin sopimaan pienelle näytölle ja tästä skaalataan sopiviksi näytön kasvaessa. Tämä on nykyään tärkeä osa verkkosuunnittelua, kun käyttäjä voi vierailla websivuilla 5 tuumaisella älypuhelimella tai 75 tuumaisella älytelevisiolla. /9/

Responsiivisten websivujen toteutukseen Bootstrap tarjoaa valmiita muotomääreitä HTML-elementeille, satoja CSS-luokkia ja ikoneita sivujen ulkoasun tekoon sekä uniikin 12-sarakkeisen ruudukkojärjestelmän sivujen asetteluun. Tässä järjestelmässä voi HTML-elementeille määrittää eri leveydet eri kokoisille näytöille eli kuinka monta saraketta ne käyttävät pienellä näytöllä (puhelin), keskisuurella näytöllä (tabletti), suurella näytöllä (tietokoneen näyttö) ja hyvin suurella näytöllä (suuret näytöt ja televisiot) /10/.

3.3.2 jQuery

Puhtaasti JavaScriptillä ohjelmoitaessa voivat yksinkertaisetkin funktiot vaatia useita rivejä vaikealukuista koodia. HTML-elementtien haku dokumentista on jäykkää ja lisäksi eri verkkoselaimet ymmärtävät eri termistöä. Näiden ongelmien ratkaisemiseksi on kehitetty jQuery, joka julkaistiin vuonna 2006. /11/

Esimerkiksi käyttäjien selaimessa tekemien tapahtumien kuuntelija (event listener) voidaan jQueryssa liittää HTML-elementtiin yhden rivin koodilla, jolloin jQuery sisäisesti tunnistaa käyttäjän selaimen ja käyttää oikeaa metodia kuuntelijan liittämiseen. Tämä tekee koodista helpommin ymmärrettävää, nopeampaa tuottaa sekä toimivampaa. Lisäksi JavaScriptilla on mahdollista hakea elementtejä

vain id:n tai luokan perusteella koko dokumentista kerralla. Tähän jQuery tuo joustavuutta mahdollistamalla monipuolisia hakulausekkeita, joilla valita elementtejä. /11/ Kuvassa 4 esimerkkikoodi, missä jQuerylla ja JavaScriptilla liitetään id:llä valittuun nappiin toiminto, jossa napin painalluksella vuorotellen piilotetaan ja näytetään HTML-elementti.

```
//jQuery
$("#test-button").click(function () {
    $("#test-div").toggle();
});

//JavaScript
document.getElementById("test-button").addEventListener("click", function () {
    let element = document.getElementById("test-div");
    if (element.style.display === 'none') {
        element.style.display = 'block';
    } else {
        element.style.display = 'none';
    }
});
```

Kuva 4. jQueryn ja JavaScriptin syntaksia.

Ajan saatossa jQueryyn on tullut muitakin kehittyneitä ominaisuuksia, kuten animaatiot ja Ajax-kutsut. JQueryn keveyden, nopeuden ja monipuolisuuden ansiosta se on pitänyt pintansa myös moderneja JavaScript ohjelmistokehyksiä vastaan ja sitä käytetään edelleen yli 77 % verkkosivuista. /12/

3.4 Sovellusohjelmointi

Puhuttaessa mobiiliapplikaatioista sovellukset ryhmitellään usein kolmeen luokkaan: web-sovellukset, hybridisovellukset sekä natiivisovellukset. Nämä eroavat toisistaan sekä kehityksen että käyttäjäkokemuksen osalta.

Web-sovellus on verkkosivu, joka on optimoitu mobiililaitteille ja ulkoasultaan vaikuttaa mobiilisovellukselta. Web-sovellukset kehitetään kuten normaalit verkkosivut, yleensä käyttäen HTML-merkintäkieltä, joten niiden tekeminen on helppoa ja nopeaa. Tämän vuoksi web-sovellusten kehitys ja ylläpito on kustannuksiltaan

halvinta näistä kolmesta sovellustyypistä. Nykyään web-sovelluksissa on käytettävissä myös joitakin puhelimen natiiviominaisuuksia kuten kamera, gps ja push-ilmoitukset. Web-sovelluksia ei jaeta puhelimen sovelluskaupoissa tai asenneta puhelimen muistiin, vaan niitä käytetään normaalilla verkkoselaimella ja sovellus ajetaan palvelimella. Tämän vuoksi sovellusten käyttö vaatii aina toimivan internetyhteyden. /13/

Arkikielessä mobiilisovelluksista puhuttaessa tarkoitetaan useimmiten natiivisovelluksia. Ne asennetaan laitteelle sovelluskaupasta, esimerkiksi Google Play tai Apple Store ja niissä on mahdollista käyttää kaikkia laitteen natiiviominaisuuksia. Natiivisovelluksilla saavutetaan nopein ja luotettavin vaste käyttäjälle. Ne tulee kuitenkin kehittää joka käyttöjärjestelmälle eri ohjelmointikielillä, mikä tekee niiden kehityksestä kallista. /14/

Viimeinen sovellustyyppi, hybridisovellus, on kahden edellä kuvatun yhdistelmä. Hybridisovellukset voidaan ladata sovelluskaupoista laitteen muistiin, mutta niiden käyttöliittymä voidaan kirjoittaa HTML-kielellä. Hybridisovelluksella on myös käytössä laajempi skaala puhelimen ominaisuuksia kuin web-sovelluksella, mutta silti rajatumpi saatavuus verrattuna natiivisovellukseen. Tärkeimpänä etuna natiivisovelluksiin verrattuna on saman lähdekoodin käyttö niin iOS kuin Android puhelimissa. Tämä mahdollistaa halvemman kehityksen sekä helpomman ylläpidon kuin natiivisovelluksissa. /15/ Hybridisovellusten kehitykseen on saatavilla useita ohjelmistokehyksiä, joista opinnäytetyön sisällönhallintajärjestelmässä käytetään Ionic-ohjelmistokehystä.

3.4.1 Ionic

Vuonna 2013 julkaistu Ionic-ohjelmistokehyksen ensimmäinen versio on rakennettu AngularJS- sekä PhoneGap Cordova (nykyään Apache Cordova) -ohjelmistokehysten päälle. Vaikka Ionic sisältää CSS-tyylittelyjen lisäksi JavaScript-kom-

ponentteja, sen ei ole tarkoitus korvata AngularJS- tai Cordova-kehyksiä, vaan pikemminkin laajentaa niitä. Uudemmat Ionic versiot toimivat myös muiden JavaScrip- kirjastojen, kuten Reactin ja Vue.js kanssa. /16/

3.4.2 Cordova

Ionicin ydin, joka mahdollistaa hybridisovelluksen teon tulee Cordovasta. Cordova toteuttaa rajapinnat sovelluksen ja puhelimen natiiviominaisuuksien, kuten kameran välillä ja mahdollistaa käyttöliittymän merkinnän HTML-kielellä. Todellisuudessa Cordova-pohjainen sovellus pyörii eräänlaisessa verkkoselaimessa, joka on koko näytön kokoinen. /17/

3.4.3 AngularJS

Googlen kehittämän JavaScript-kehyksen AngularJS:n tärkein ominaisuus on HTML-dokumentin dynaamisuuden helpottaminen sekä MVC-arkkitehtuurin toteuttaminen. Sen kaksisuuntaisen datasidoksen ansiosta muuttaessa muuttujien arvoja kontrollerissa, päivittyvät ne automaattisesti myös käyttäjän näkymään. /18/ AngularJS on Angular sarjan ensimmäinen versio vuodelta 2010. Toisesta versiostaan eteenpäin ohjelmistokehys on pohjautunut Typescriptiin, joten AngularJS nimityksellä viitataan juuri ensimmäiseen versioon. /19/

4 TOTEUTUS

Kaikki kehitys toteutetaan paikallisesti (localhost), missä palvelimena toimii Windows tietokoneeseen asennettu alajärjestelmä Ubuntu 20.04, jossa on Apache 2 -palvelinohjelma. Ohjelmointiympäristönä käytetään IntelliJ IDEAa ja versionhallintaa varten perustetaan projektille oma tietovarasto (repository) yrityksen GitLabiin.

4.1 Palvelinpuoli

Ohjelmiston kehitys aloitetaan palvelinpuolelta, missä ensimmäisenä luodaan tietokantaan taulut sovellusta varten. Lisäksi palvelimelle toteutetaan suunnitellut luokat. Nämä yhdistetään Zend Frameworkin komponenttien avulla luokkia vastaaviin tietokantatauluihin. Luokat luodaan yhteiseen nimiavaruuteen (namespace), jotta näiden käsittely olisi helppoa. Toteutus on alkuun kaikilla luokilla lähes identtinen (**Kuva 5.**), mutta näihin lisätään omia metodeja tarpeen vaatiessa.

```php
<?php

namespace Report\Model;

use ...

/** Class Report\Model\Section ...*/
class Section extends Base
{

    /** Array of section blocks ...*/
    private $_blocks;

    public function __construct($params = [])
    {
        parent::__construct($params);
        $this->_db_table = Db\Table\Section::class;
    }
```

Kuva 5. Raportin osiota kuvaava luokka.

4.2 Hallintapaneeli

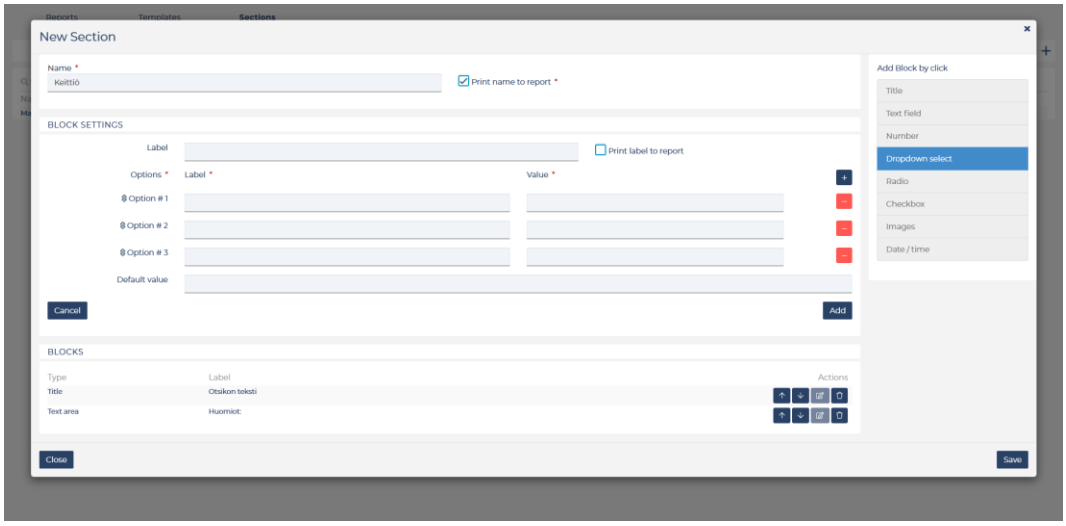
Sivusto toteutetaan yksisivuisen sovelluksen (single-page application) periaatteella. Selaimella navigoidaan sivuston osoitteeseen, jonka jälkeen sivua piirretään dynaamisesti uudelleen käyttäjän toimintojen mukaan. Näin saavutetaan sulavampi käyttäjäkokemus kuin perinteisellä verkkosivulta toiselle navigoinnissa.

Ensimmäisenä luodaan aloitussivu, jossa on täytettyjä raportteja sekä navigointipainikkeet luotujen raporttipohjien ja osioiden sivuille. Tässä vaiheessa aloitussivulle tehdään ainoastaan navigointinapit muille sivuille. Navigointi tapahtuu piilottamalla nykyinen ja näyttämällä käyttäjän valitsema sivu jQuery-animaatioiden avulla.

4.2.1 Osioiden hallinta

Osioiden välilehdelle tehdään taulukko luotuja osioita varten sekä ikoni, jota painamalla voi luoda uuden osion. Osion luonti suoritetaan muun sivun päälle tulevassa ikkunassa (Bootstrap modal). Uuden luonti -ikonia tai aiemmin luodun osion riviä klikkaamalla haetaan ikkunan tiedot jQueryn ajax-kutsulla. Palvelimen palauttama html kirjoitetaan ikkunalle varattuun elementtiin ja se näytetään käyttäjälle.

Avautuvassa ikkunassa käyttäjän tulee syöttää osiolle nimi sekä lisätä osiolle vähintään yksi syöte-elementti. Elementin valinnalla käyttäjälle näytetään kyseisen syötteen vaatimat asetukset. Asetusten tallennuksen jälkeen lisätty syöte listataan ikkunaan näkyville tulevaan taulukkoon. Lisättyjä elementtejä voi taulukossa uudelleen järjestellä, valita muokattavaksi tai poistaa. (**Kuva 6.**)



Kuva 6. Osion luonti-ikkuna. Valittuna alasvetovalikko elementti, jonka vaihtoehtoja voi uudelleenjärjestää osoittimella vetämällä (drag-and-drop).

Tallenna-painikkeella lisätyt tiedot tarkistetaan ensiksi jQueryn komentosarjalla. Tarkistuksen jälkeen tiedot lähetetään palvelimelle. Tiedot tarkistetaan sekä päätelaitteella että palvelinpäässä käyttäjäkokemuksen sujuvoittamiseksi. Päätelaitteella tehtävän tietojen tarkistuksen avulla käyttäjä saa ilmoituksen epähuomiossa jääneistä virheistä ilman http-siirron aiheuttamaa viivettä. Tarkistus toistetaan palvelinpuolella, sillä asiakaspään tarkistuksen voi käyttäjä sivuuttaa. Käytetty sisällönhallintajärjestelmä sekä Zend Framework tarkistavat saapuvat tiedot automaattisesti myös sql-injektioiden sekä muiden hyökkäysten varalta.

Palvelinpuolen tallennettua tiedot tietokantaan palautuu selainpuolelle sanomassa lisätyn osion tiedot. Näillä tiedoilla lisätään uusi rivi taustalla olevaan osioiden taulukkoon ja luonti-ikkuna suljetaan. Käyttäjä voi tämän jälkeen luoda uusia osioita, muokata edellä luomaansa osiota tai käyttää osiota mobiilipuolella raportin luonnissa.

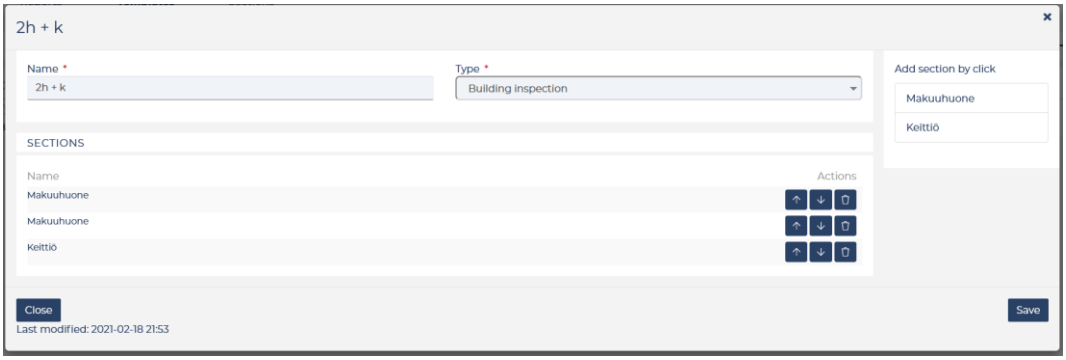
Osion luontia testataan kehityksen yhteydessä muun muassa tietokantakenttiä pidemmillä syötteillä, tyhjillä syötteillä sekä syöttämällä tekstiä kenttiin, joihin odo-

tetaan vain numeroita. Palvelinpäässä toteutettavat tietojen tarkistukset ohjelmoidaan havaitsemaan epäsopivat syötteet ja palauttamaan virheilmoitus käyttäjälle tallentamatta tietoja tietokantaan.

4.2.2 Raporttipohjien hallinta

Nopeuttaakseen työskentelyä raporttia täyttäessään, käyttäjä voi luoda valmiita malleja käytettäväkseen applikaatiossa. Pohjat kasataan edellä luoduista osioista ja ne voidaan nimetä vapaasti sekä valita raportille oletuksena tuleva tyyppi. Sovelluksen yhdenmukaisuuden vuoksi raporttipohjien välilehti tehdään vastaavasti kuin osioiden välilehti ja pohjien luonti sekä muokkaus tapahtuu vastaavasti Bootstrapin ikkunassa. Sovelluksen käytön oppiminen on käyttäjälle nopeaa ja ylläpitäjien päivitystyö on helppoa, kun näkymät ja koodi ovat mahdollisimman yhdenmukaisia.

Raporttipohjan hallintaikkunassa (**Kuva 7.**) käyttäjän oikeasta palkista valitsemat osiot listataan taulukkoon jQueryn avulla. Taulukossa näitä voidaan jälleen uudelleenjärjestellä tai poistaa. Raporttia luotaessa tulee raportille määrittää tyyppi, esimerkiksi kosteuskartoitus tai rakennustarkistus. Tallennettu tyyppi tulee oletuksena raportille pohjaa käytettäessä. Tyyppi-valikossa on muutamia valmiita vaihtoehtoja, minkä lisäksi käyttäjällä on mahdollisuus syöttää oma tyyppi vapaatekstikenttään. Valittu tyyppi voidaan korvata vielä raporttia täytettäessä.



Kuva 7. Raporttipohjien hallintaikkuna

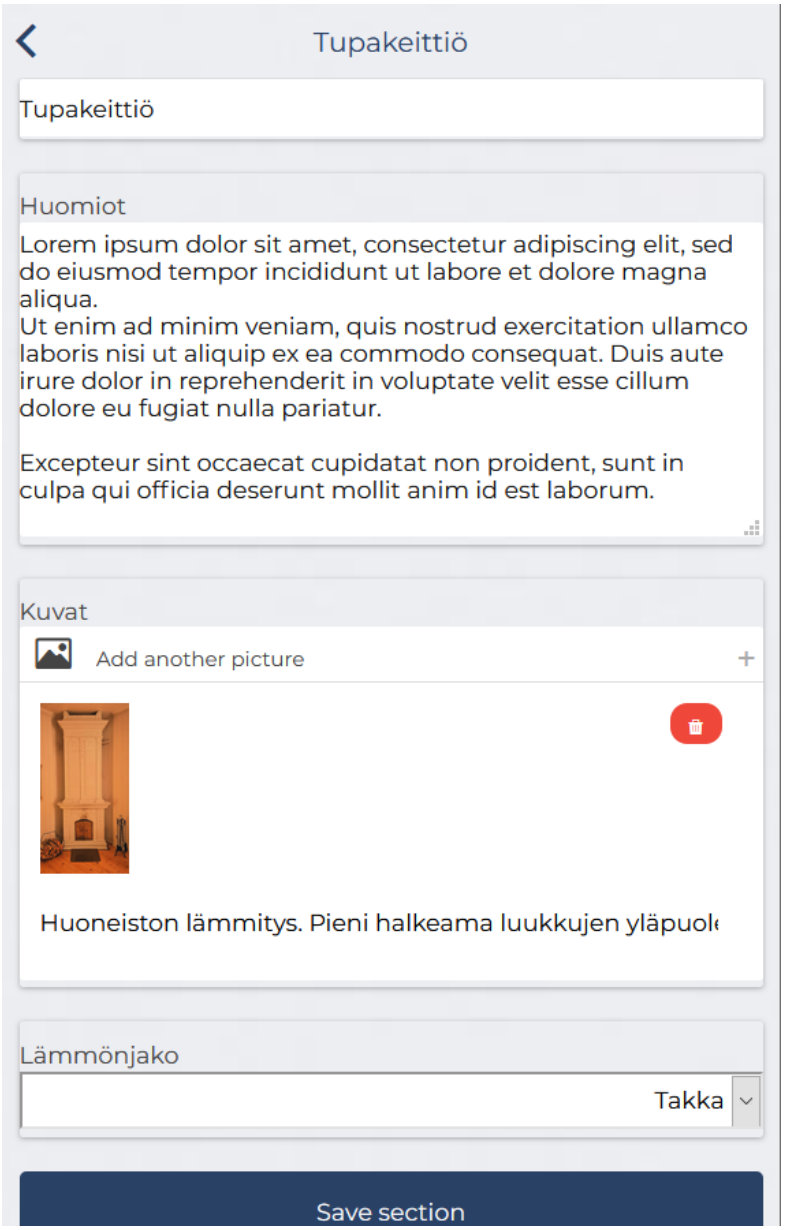
Raporttipohjan tallennuksessa käydään läpi vastaavat prosessit kuin osiota tallentaessa, eli tiedot tarkistetaan sekä asiakas- että palvelinpuolella ja tallennetaan tietokantaan. Tallennuksen onnistuttua pohja on välittömästi käyttäjän saatavilla sekä applikaatiossa että hallintapaneelin listauksessa.

4.3 Hybridisovellus

Käyttäjän kirjauduttua sisään hän saapuu uuden raportin luontinäkymään (**Kuva 8.**). Yläpalkin (Ionic header) painikkeilla voidaan lisätä joko yksittäisiä osioita raportille tai valita raporttipohja käytettäväksi, jolloin mallin osiot lisätään automaattisesti raportille ja tyyppikenttään asetetaan arvo valitun pohjan tiedoilla.

Kuva 8. Sovelluksen raportin luontinäkymä

Osion riviä klikkaamalla siirrytään AngularJS:n uuteen tilaan, jossa osion tiedot voidaan täyttää. Täytettävän raportin tiedot säilytetään Raportti factoryssa. Raportti factoryn avulla käyttäjän syötteet säilyvät eri tilojen välissä tallessa. Tässä uudessa tilassa osion sisältämät syötteet listataan yhteen näkymään ja käyttäjä voi vapaasti täyttää kentät. Osiolle voidaan antaa myös oma nimi kyseisessä raportissa. (**Kuva 9.**)



Kuva 9. Osion täyttönäkymä.

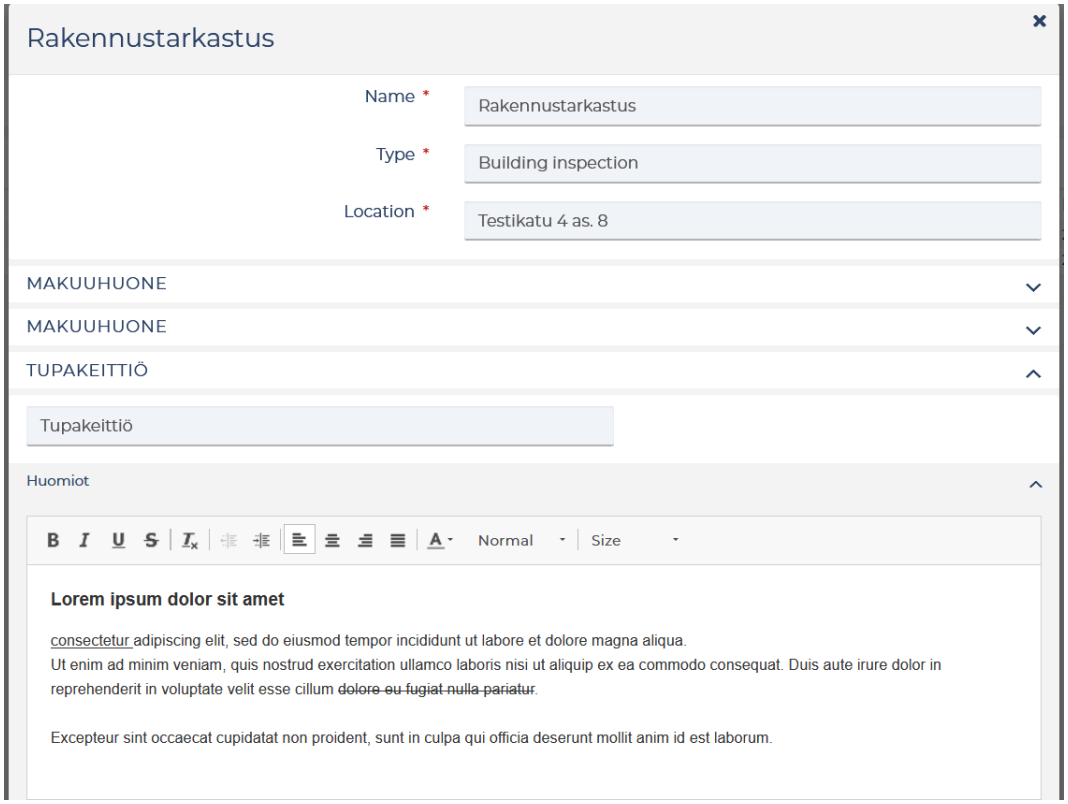
Osion tallennuksen jälkeen näkymä palautuu edelliseen tilaan ja osio merkataan täytetyksi. Täytettyä osiota voidaan edelleen muokata, uudelleen järjestellä muiden osioiden kanssa tai poistaa raportilta.

Kun käyttäjä on saanut tiedot täytettyä, raportti tallennetaan tietokantaan. Käyttäjä saa palvelimelta ilmoituksen onnistuneesta tallennuksesta Ionic dialog -ikkunaan ja raportin täyttönäkymä resetoidaan. Tämän jälkeen raporttia voi jatkojalostaa ja uudelleen muokata hallintapaneelissa.

Manuaalisella testauksella varmistetaan, että epäsopivat syötteet jälleen hylätään. Erityistä huomiota vaatii palvelimelle lähetetyn tiedoston tiedostomuoto, jonka tulee olla järjestelmän hyväksymässä kuvan tallennusformaatissa (.jpg, .png, .jpeg).

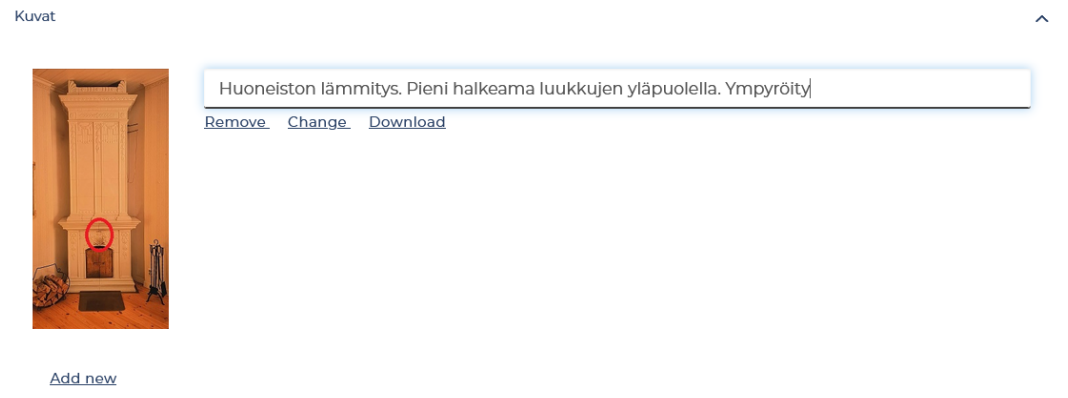
4.4 Raportin muokkaus

Tärkein ominaisuus raportin muokkauksessa on tekstialue syötteiden muotoilu. Kun raportin tiedot on sovelluksessa täytetty yksinkertaiseen tekstialueeseen, voidaan teksti tarvittaessa muotoilla kehittyneemmällä tekstieditorilla (**Kuva 10.**). Käytettävissä on tärkeimmät muotoilun välineet, kuten fonttikoon valinta, tekstin jäsentely, lihavointi, kursivointi ja alleviivaus. Ohjelmistossa käytetään avoimen lähdekoodin CKEditor-tekstieditoria.



Kuva 10. Raportin tekstialue syötteiden muotoilu

Myös muiden syötteiden arvoja voidaan vielä muokata vapaasti. Toinen tärkeä mahdollisuus muokkausvaiheessa on kuvien käsittely. Ohjelmistoon ei kehitetä omaa kuvienkäsittelyä, mutta käyttäjälle annetaan mahdollisuus ladata tallennetut kuvat omalle laitteelleen ja käsitellä kuvat haluamallaan tavalla. Kuvia voidaan latauksen lisäksi korvata, poistaa ja lisätä. (**Kuva 11.**)



Kuva 11. Raporttiin lisättyjen kuvien käsittely.

Käyttäjä voi useita kertoja generoida raportista PDF-tiedoston tallentamatta muutoksia tietokantaan. Tällöin raportin tiedot lähetetään palvelimelle, jossa rakennetaan väliaikainen PDF. Palvelimelta palautetaan käyttäjäpäähän url-osoite tiedostoon ja vastaanotettu url-osoite avataan JavaScript-komentosarjalla käyttäjän selaimessa uuteen ikkunaan. Tämän jälkeen luotu tiedosto poistetaan palvelimen kovalevyltä. Ollessaan tyytyväinen muutoksiin käyttäjä voi tallentaa raportin myöhempää tarkastelua tai arkistointia varten.

4.5 PDF-generointi

Palvelimen Zend-kehys sisältää luokkia PDF-tiedostojen käsittelyyn, mutta työssä tutkitaan PHP-kirjastoja, jotka mahdollistavat PDF-tulostuksen HTML-pohjalta. Tarjolla on niin kaupallisia kuin avoimen lähdekoodin kirjastoja, joista opinnäytetyössä keskitytään avoimen lähdekoodin ratkaisuihin. Parhaiksi vaihtoehdoiksi osoittautuivat kirjastot FPDF, Snappy sekä TCPDF. Kirjastojen välillä erot ovat lopulta pieniä. Kaikki sisältävät tärkeimmät haetut ominaisuudet, kuten automaattisen sivunumeroinnin, kaikille sivuille tulevan ylä- ja alatunnisteen sekä tuen kuvien tulostukseen. Snappy sisältää kirjastoista laajimman HTML- ja CSS-tuen, mutta vaatii toimiakseen palvelimelle asennettavan Wkhtmltopdf-binääritiedoston. Tästä syystä Snappy jätetään vaihtoehdoista pois. Lopulta työssä päädytään käyttämään TCPDF-kirjastoa, sen lukuisien lisäominaisuuksien kuten viivakoodituen ansiosta.

Jotta TCPDF:n generoimien tiedostojen ylä- ja alatunnisteita voidaan muokata, täytyy kirjoittaa uusi luokka, joka periytyy TCPDF-luokasta. Tässä luokassa ylikirjoitetaan alkuperäiset Header- ja Footer-metodit. Järjestelmän muodostamien tiedostojen ylätunnisteeseen tulee raportin tehneen käyttäjän yrityksen logo sekä raportin perustiedot. Alatunnisteeseen tuodaan sivunumerointi, käyttäjän yhteystiedot sekä ohjelmiston kehittäneen yhtiön logo.

Itse raportti muodostetaan osio kerrallaan. Tietokannasta haetaan raportin osiot ja näiden sisältämien syötteiden arvot, jotka tulostetaan syötetyyppien mukaan

käyttäjän määrittelemään järjestykseen allekkain PDF-tiedostoon. Jokaisen osion jälkeen lisätään sivunvaihto, jotta rakenne pysyisi mahdollisimman selkeänä.

Käyttäjän saatua raportin valmiiksi ja muokatuksi, on viimeinen työvaihe raportille sen arkistointi. Tällöin raportista muodostetaan lopullinen PDF-tiedosto eikä sitä voida enää muokata. Samalla arkistoitu raportti poistetaan hallintapaneelin päänäkymän raporttilistalta, ja siirretään tarkasteltavaksi erilliseen arkistoikkunaan hallintapaneelissa. Arkistoituja raportteja tarkastellessa käyttäjälle näytetään levylle tallennettu raportti Bootstrap-ikkunassa.

5 POHDINTA

Työn aikana toteutettu ohjelmisto antoi hyvän käsityksen suuremman sovelluskokonaisuuden suunnittelun ja toteutuksen eri vaiheista. Viikoittaiset työn etenemisen seurantatapaamiset yrityksen edustajan kanssa tarjosivat kosketuspintaa ohjelmistokehityksen todelliseen luonteeseen. Työstä saatu palaute ja vaatimusten tarkentuminen valmensivat teknisten ratkaisujen kriittiseen arviointiin ja toivat näkemystä, miten luoda mahdollisimman helposti muokattava ohjelmisto. Näin tulevaisuuden muutostarpeisiin on helppo reagoida ja lisäominaisuuksien tuottaminen on mahdollista.

Tutustuminen eri sisällönhallintajärjestelmiin, ohjelmistokehyksiin sekä kirjastoihin opetti ohjelmistotuotannon helppouden nykypäivänä. Harvaa toimintoa tarvitsee rakentaa perustuksista lähtien itse, kun lukuisat avoimen lähdekoodin ratkaisut tarjoavat valmiita työkaluja ohjelmistojen rakentamiseen. Avoimen lähdekoodin ratkaisut jakavat myös ylläpidollista taakkaa, kun mahdolliset ohjelmointivirheet saatetaan korjata suuren yhteisön toimesta.

Tuhansien koodirivien tuottaminen yritykselle, jossa myös muut kehittäjät osallistuvat jatkossa ohjelmiston päivittämiseen, muovasi omaa koodaustyyliäni yhä selkeämmäksi. Pitkien metodien pilkkominen pienempiin osiin sai koodista huomattavasti helpompi lukuista sekä teki virheiden löytämisestä nopeampaa. Lisäksi käytetyn sisällönhallintajärjestelmän tiedostoarkkitehtuuri opetti erottelemaan ohjelmiston kokonaisuudet tiettyyn kansiorakenteeseen.

Ohjelmisto vastasi kaikkiin yrityksen asettamiin vaatimuksiin. Vaikka käyttöönotto saattaa olla työläs erilaisia raportin osioita ja pohjia luotaessa, on sen käyttö sujuvaa hyvän pohjatyön jälkeen. Jatkokehityksenä onkin tarkoitus antaa valmiita osioita ja pohjia käyttäjälle, jotta ensimmäinen raportti saataisiin tehtyä minuuteissa ohjelmiston hankinnasta.

Opinnäytetyön aikana toteutettu sovellus tullaan toimittamaan kehityksessä mukana olleelle rakennustarkastajalle käytännön testaukseen. Ohjelmiston kehitystä jatketaan hänen palautteensa perusteella mahdollisimman käyttäjäystävälliseksi. Lisäksi sovelluksen kehityksessä on valmistauduttu tuleviin rajapinta integraatioihin yrityksen muihin järjestelmiin sekä sähköisen allekirjoituksen palveluihin.

LÄHTEET

/1/ Luokka ja olio - Ohjelmoinnin MOOC 2019. Viitattu 18.1.2021. https://ohjelmointi-19.mooc.fi/osa-4/3-luokka-ja-olio

/2/ The MIT License | Open Source Initiative. Viitattu 22.1.2021. https://opensource.org/licenses/MIT

/3/ AngularJS | Factory Method. Viitattu 18.2.2021. https://www.geeksforgeeks.org/angularjs-factory-method/

/4/ UML Class Diagram Tutorial. Viitattu 17.1.2021. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/

/5/ Common Web Design Languages, What They Do and Why You Need Them. Viitattu 19.1.2021. https://www.spinxdigital.com/blog/common-web-design-languages-what-they-do-and-why-you-need-them/

/6/ Usage statistics of content management systems. Viitattu 3.3.2021. https://w3techs.com/technologies/overview/content_management

/7/ Zend Framework & MVC Introduction. Viitattu 19.1.2021. https://framework.zend.com/manual/1.12/en/learning.quickstart.intro.html

/8/ About – Bootstrap. Viitattu 22.1.2021. https://getbootstrap.com/docs/4.1/about/overview/

/9/ What is Mobile First Design? Why It's Important & How To Make It? | by Vincent Xia | Medium. Viitattu 23.1.2021. https://medium.com/@Vincentxia77/what-is-mobile-first-design-why-its-important-how-to-make-it-7d3cf2e29d00

/10/ Overview – Bootstrap. Viitattu 22.1.2021. https://getbootstrap.com/docs/4.1/layout/overview/

/11/ The history and legacy of jQuery. Viitattu 28.1.2021. https://blog.lo-grocket.com/the-history-and-legacy-of-jquery/

/12/ Usage statistics of JavaScript libraries for websites. Viitattu 28.1.2021. https://w3techs.com/technologies/overview/javascript_library

/13/ Web application (Web app). Viitattu 30.1.2021. https://searchsoftwarequal-ity.techtarget.com/definition/Web-application-Web-app

/14/ Native Apps, Web Apps or Hybrid Apps? What's the Difference?. Viitattu 30.1.2021. https://www.mobiloud.com/blog/native-web-or-hybrid-apps#5

/15/ What is Hybrid App Development? Viitattu 30.1.2021. https://ionic.io/re-sources/articles/what-is-hybrid-app-development

/16/ Ionic Documentation Overview. Viitattu 3.3.2021. https://ionicframe-work.com/docs/v1/overview/

/17/ PhoneGap Explained Visually. Viitattu 30.1.2021. https://phon-egap.com/blog/2012/05/02/phonegap-explained-visually/

/18/ AngularJS. Viitattu 31.10.2021. https://angularjs.org/

/19/ AngularJS and Angular 2+: a Detailed Comparison. Viitattu 31.10.2021 https://www.sitepoint.com/angularjs-vs-angular/