Tampereen ammattikorkeakoulu

Online-varausjärjestelmän jatkokehitys

Neea Metsola

OPINNÄYTETYÖ
Kesäkuu 2020

Tietojenkäsittely
Ohjelmistotuotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittely
Ohjelmistotuotanto

METSOLA, NEEA:
Online-varausjärjestelmän jatkokehitys

Opinnäytetyö 40 sivua
Kesäkuu 2020

Tämän opinnäytetyön aiheena oli Tampereen Tulli Business Parkissa käytettävän online-varausjärjestelmän jatkokehitys. Työn toimeksiantajana toimi ohjelmistotalo BearIT Oy. Opinnäytetyön tavoitteena oli jatkokehittää online-varausjärjestelmää vastaamaan paremmin sen loppukäyttäjien tarpeisiin. Työn tarkoituksena oli tutustua jatkokehityksessä käytettäviin tekniikoihin ja määritellä, suunnitella sekä toteuttaa järjestelmän jatkokehitysprojekti.

Opinnäytetyön tuloksena syntyi uusi järjestelmäversio, joka sisältää päivitettyjä ja uusia toiminnallisuuksia. Järjestelmän ohjelmistokoodia refaktoroitiin jatkokehityksen yhteydessä. Refaktoroiminen tarkoittaa ohjelmistokoodin rakenteen ja luettavuuden parantamista niin, ettei ohjelmiston ulkoinen toiminta muutu. Järjestelmän jatkokehitys edisti sen mahdollista laajempaa käyttöönottoa, sillä uusi versio lisäsi sen käyttömahdollisuuksia ja paransi sen rakennetta.

Refaktoroimista voidaan pitää tärkeänä osana ohjelmistokehitystä. Refaktoroimalla ohjelmistokoodia säännöllisesti sen rakenne pysyy selkeänä, jolloin sen ylläpitäminen on tehokkaampaa. Työssä käsitellyn online-varausjärjestelmän vielä laajempi refaktoroiminen olisi hyödyllistä sen ylläpidon ja laajemman käyttöönoton kannalta.

Asiasanat: jatkokehitys, refaktoroiminen

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Option of Software Development

METSOLA, NEEA
Further Development of an Online Booking System

Bachelor's thesis 40 pages
June 2020

The purpose of this thesis was to further develop an online booking system to make it achieve the new requirements set for it. The system was originally developed for Tulli Business Park by the commissioner of this thesis, BearIT Oy. The theoretical section explores software maintenance and refactoring techniques.

As a result of this thesis, a new system version of the online booking system was developed. The new version includes new and modified features. The software code was refactored during the project, which improved the design of the code.

Refactoring is an effective way to prevent the structure of software code from becoming too difficult to understand. Well-structured code is easier to maintain and modify.

Key words: further development, refactoring

SISÄLLYS

LYHENTEET JA TERMIT

AJAX	Asynchronous JavaScript and XML, hyödyntää useita teknologioita mahdollistaen käyttöliittymän osittaisen päivittämisen
AWS	Amazon Web Services, pilvilaskenta-alusta
EC2	Amazon Elastic Compute Cloud, palvelu, joka tarjoaa skaalautuvaa laskentakapasiteettia pilvessä
ORM	Object-Relational Mapping, tekniikka, jolla data muutetaan yhteensopivaksi eri järjestelmien välillä
RDS	Amazon Relational Database Service, relaatiotietokantojen hallintaan ja luomiseen käytettävä tietokantapilvipalvelu

1 JOHDANTO

Ohjelmistokehittäjät käyttävät uransa aikana enemmän aikaa koodin ylläpitoon kuin uuden koodin kirjoittamiseen. Usein kehittäjät ylläpitävät omaa koodiaan, eivätkä ainoastaan jonkun toisen kirjoittamaa vanhaa koodia. Tämän vuoksi on hyvä oppia jo ohjelmistokehitysuran alussa, miten koodia ylläpidetään ja miten kirjoitetaan koodia, jota on helppoa ylläpitää. (Sonmez 2017.)

Jatkokehittämisen kannalta molemmista taidoista on hyötyä, sillä hyvin ylläpidetyn koodin jatkokehittäminen on sujuvaa. Kun jatkokehityksen yhteydessä kirjoitettu uusi koodi on myös helposti ylläpidettävää, on ohjelmiston ylläpito jatkossakin selkeää. Aina näin ei kuitenkaan ole, vaan ohjelmiston rakenne on saattanut ajan kuluessa muuttua epäselkeäksi ja koodi vaikeasti ymmärrettäväksi. Tällöin voidaan käyttää apuna refaktoroimista, eli koodin rakenteen parantamista tekemällä siihen muutoksia, jotka eivät vaikuta ohjelmiston ulkoiseen toimintaan.

Tämän opinnäytetyön aiheena on Tampereen Tulli Business Parkin vastaanotossa käytössä olevan Aulapalvelu-järjestelmän jatkokehitys. Työn toimeksiantaja on Tampereella ja Helsingissä toimiva ohjelmistotalo BearIT Oy, joka on alun perin kehittänyt järjestelmän. Järjestelmää käytetään vierailijoiden sisään kirjaamiseen ja neuvotteluhuoneiden varausten ylläpitoon. Se kehitettiin korvaamaan paperiset lomakkeet ja kalenterit sekä helpottamaan ja automatisoimaan henkilökunnan työtehtäviä. Nyt järjestelmää halutaan jatkokehittää vastaanoton henkilökunnan käyttökokemusten pohjalta.

Työn tavoitteena on jatkokehittää Aulapalvelu-järjestelmää vastaamaan paremmin loppukäyttäjän tarpeisiin ja näin helpottaa heidän päivittäistä työtään. Työn tarkoituksena on tutustua jatkokehityksessä käytettäviin tekniikoihin ja määritellä Aulapalvelu-järjestelmän jatkokehitystarpeet sekä suunnitella ja toteuttaa ominaisuudet määrittelyn pohjalta. Opinnäytetyössä käsitellään järjestelmän jatkokehityksen lisäksi sen taustaa ja nykyistä toimintaa. Lopuksi esitellään jatkokehitysprojektin tulokset.

2 JATKOKEHITYS

2.1 Ylläpito ja jatkokehitys

Jotta käyttöön otettu järjestelmä pysyisi käyttökelpoisena, tulee sitä kehittää vastaamaan uusiin ja muuttuviin vaatimuksiin. Järjestelmää saatetaan joutua korjaamaan, jos sen käytön aikana on löytynyt virheitä tai muokkaamaan, mikäli sen toimintakykyä halutaan parantaa tai järjestelmä halutaan ottaa käyttöön eri alustalla kuin alun perin. Ohjelmiston kehitys ei siis pääty siihen, kun se otetaan käyttöön, vaan kehitys jatkuu läpi ohjelmiston elinajan. (Sommerville 2006, 489.)

Ohjelmiston kehitystä voidaan ajatella spiraalina, jossa vaatimukset, suunnittelu, toteutus ja testaus toistuvat koko järjestelmän elinajan. Käyttöönoton jälkeen kehittäminen ei liity pelkästään järjestelmän virheiden korjaamiseen, vaan suuri osa muutoksista johtuu uusista vaatimuksista, jotka syntyvät liiketoiminnan tai käyttäjien muuttuneiden tarpeiden pohjalta. (Sommerville 2006, 489.)

Järjestelmän ylläpidolla tarkoitetaan järjestelmän muokkaamista sen toimituksen jälkeen. Muutokset voivat korjata virheitä, tai ne voivat vastata uusiin vaatimuksiin. Muutokset toteutetaan muokkaamalla olemassa olevia järjestelmän osia ja tarpeen mukaan lisäämällä uusia osia siihen. Järjestelmän ylläpito voidaan jakaa kolmeen osaan: järjestelmää korjaavaan, mukauttavaan ja muokkaavaan ylläpitoon. Järjestelmää korjaava ylläpito voi koskea koodissa ilmenneitä virheitä, suunnitteluvirheitä tai väärin ymmärrettyihin vaatimuksiin liittyviä virheitä. Mukauttava ylläpito vastaa järjestelmän ympäristön, kuten käyttöjärjestelmän tai laitteiston vaihtumisesta johtuvista muutoksista. Muokkaava ylläpito vastaa järjestelmän uusiin ja muuttuneisiin vaatimuksiin, eli se muuttaa tai lisää järjestelmän toiminnallisuuksia. Muokkaavan ylläpidon vaatima laajuus on usein selkeästi suurempi kuin muiden ylläpitotyyppien. (Sommerville 2006, 492–493.) Muokkaavasta ylläpidosta voidaan puhua myös jatkokehityksenä.

Käytännössä ylläpidon jakaminen eri osiin ei ole näin selkeää, vaan erityyppisiä ylläpitotöitä voidaan tehdä samanaikaisesti, jos se koetaan hyödylliseksi. Lisäksi

eri tahot luokittelevat tai nimeävät ylläpitotyypit eri tavoin. Pääsääntöisesti korjaava, mukauttava ja muokkaava ylläpito kuitenkin tunnistetaan omiksi tyypeikseen. (Sommerville 2006, 493.) Yksi tapa luokitella ylläpito on jakaa se kahteen osaan: korjaavaan ja parantavaan ylläpitoon. Korjaava ylläpito huolehtii, että järjestelmä toimii halutulla tavalla. Parantava ylläpito muuttaa järjestelmän toimintaa tai toteutusta, ja se on jaettu vielä kolmeen alaluokkaan: ominaisuuksia muokkaavaan ja ominaisuuksia lisäävään ylläpitoon sekä toteutusta muokkaavaan ylläpitoon, joka ei kuitenkaan muuta järjestelmän toimintaa. (Tripathy & Naik 2014, 4–5.) Toteutusta muokkaavaa, mutta toiminnan ennallaan säilyttävää ylläpitoa voidaan kutsua myös refaktoroimiseksi.

2.2 Refaktoroiminen

Ohjelmistoa muokataan, parannetaan ja mukautetaan jatkuvasti vastaamaan uusiin vaatimuksiin. Ohjelmiston kehittyessä se usein menettää alkuperäisen mallinsa, jolloin sen ymmärrettävyys ja luettavuus heikkenevät samalla, kun ylläpidon aiheuttamat kustannukset nousevat. Tämä johtuu usein koodin monimutkaistumisesta ylläpidon yhteydessä, vanhentuneesta dokumentaatiosta ja koodin standardien puutteesta. (Tripathy & Naik 2014, 255–256.) Tämän estämiseksi tai tilanteen korjaamiseksi ohjelmistokoodia voidaan refaktoroida.

Refaktorointi tarkoittaa muutosta, joka tehdään ohjelmiston sisäiseen rakenteeseen, jotta siitä tulisi helpommin ymmärrettävää ja muokattavaa ilman, että ohjelmiston ulkoinen toiminta muuttuu. Refaktoroimisella tarkoitetaan kyseisten muutosten tekemistä ohjelmistoon. Refaktorointi on siis yksittäinen muutos, joita tehdään useita refaktoroimisprosessin aikana. Refaktoroimiseksi voidaan kutsua ainoastaan muutostyötä, joka ei vaikuta ohjelmiston toimintaan. Näin ollen esimerkiksi suorituskyvyn parantamiseen tähtäävät muutokset eivät ole refaktoroimiseen liittyviä, vaikkei ohjelmiston ulkoinen toiminta välttämättä muutu niiden myötä näkyvästi. (Fowler & Beck 2018.)

Refaktoroimisella pyritään siis pitämään ohjelmiston rakenne selkeänä ja koodi helppolukuisena. Kun koodi on selkeää, sitä on nopeampaa ja yksinkertaisempaa muokata. Koodin pitäminen helppolukuisena mahdollistaa myös sen tehokkaan

muokkaamisen siinäkin tapauksessa, että jatkokehitystä tekevä ohjelmoija ei ole aiemmin työskennellyt kyseisen ohjelmiston parissa. Toisaalta uuteen ohjelmistoon tutustuva ohjelmoija voi käyttää refaktoroimista apuna koodin ymmärtämisessä, jos koodi on vaikealukuista. Refaktoroinnin tekeminen auttaa ymmärtämään koodin toimintaa samalla, kun sen rakenne paranee. Helppolukuisesta koodista on myös helpompaa löytää mahdollisia virheitä, joten niiden etsimiseen kuluu vähemmän aikaa. (Fowler & Beck 2018.)

Fowler ja Beck (2018) jakavat jatkokehityksen refaktoroimiseen ja uusien ominaisuuksien lisäämiseen. Kun koodia refaktoroidaan, ei ole tarkoitus lisätä uusia toimintoja vaan saada olemassa olevasta koodista toimivampaa, helpommin ymmärrettävää tai paremmin jäsenneltyä. Uusia ominaisuuksia lisättäessä ei ole tarkoitus muuttaa aiempaa koodia vaan luoda uutta. Ohjelmistokehittäjä saattaa liikkua näiden kahden vaiheen välillä hyvinkin tiuhaan. Kun kehittäjä on lisäämässä uutta ominaisuutta, hän saattaa huomata, että olemassa olevan koodin refaktoroimisesta olisi hyötyä ja taas ominaisuuden valmistuttua, että juuri kirjoitettua koodia olisi hyvä vielä muokata.

Vaikka Fowler ja Beck puhuvat refaktoroimisen tärkeydestä lähinnä jatkokehityksen aikana, refaktoroimisesta on hyötyä myös ohjelmiston muissa ylläpitotehtävissä. Ilman jatkuvaa refaktoroimista ohjelmiston rakenne muuttuu usein ajan kuluessa vaikeasti ymmärrettäväksi. Säännöllinen refaktoroiminen auttaa säilyttämään ohjelmiston perusrakenteen myös ylläpidon yhteydessä. (Tripathy & Naik 2014, 13.)

Refaktoroiminen kannattaa Fowlerin ja Beckin (2018) mukaan ottaa osaksi jatkokehitystä sen sijaan, että sille varattaisiin oma aikansa. Yleensä yksittäinen refaktorointi on nopeaa tehdä, mutta joskus refaktoroiminen voi viedä jopa viikkoja. Tällöinkin he suosittelevat refaktoroimisen sisällyttämistä projektiin. Usein toimiva ratkaisu on sopia, että aina kun joku kehitystiimistä työstää refaktoroimista kaipaavaa koodia, hän refaktoroi sitä samalla kun toteuttaa muutoksia.

Uutta ominaisuutta lisättäessä koodia refaktoroidaan usein, jotta ohjelmoija ymmärtää vanhaa koodia paremmin tai koska koodin rakenteen parantaminen yk-

sinkertaistaa uuden ominaisuuden lisäämistä. Refaktorointi virheiden korjaamisen yhteydessä on myös yleistä. Toisaalta virheraportin saaminen saattaa myös tarkoittaa sitä, että koodia olisi hyvä refaktoroida, jotta virheitä olisi helpompi välttää tai ainakin löytää ne. Joskus koodista löytyy huonosti toteutettu osa, jonka refaktoroiminen parantaisi rakennetta ja helpottaisi koodin ymmärtämistä jatkossa, joten ohjelmoija päättää refaktoroida, vaikkei se vaikuta työstettävän muutoksen tekemiseen. Refaktoroimista voidaan tehdä myös koodin katselmoinnin yhteydessä. Refaktoroiminen ei kuitenkaan aina ole kannattavaa. Jos huonosti toteutettua koodia ei ole tarvetta ymmärtää tai muokata, sen refaktoroiminen ei välttämättä ole hyödyllistä sillä hetkellä. Joskus taas koodi on helpompaa korvata kokonaan uudella, kuin käyttää aikaa sen refaktoroimiseen. (Fowler & Beck 2018.)

2.3 Refaktoroimistekniikat

Yksi tärkeimmistä asioista koodin selkeyden ja luettavuuden kannalta on metodien ja muuttujien nimeäminen selkeästi. Kuvaavan nimen avulla voidaan heti päätellä, mitä esimerkiksi jokin metodi tekee. Huono nimeäminen vaikeuttaa koodin ymmärtämistä, mutta valitettavan usein kuvaavan nimen keksiminen on hankalaa. Yleisimpiä refaktorointeja ovatkin muuttujan ja metodin nimeäminen uudelleen. Metodin kohdalla on hyvä miettiä, että myös sen vastaanottamien parametrien nimet ovat kuvaavia. (Fowler & Beck 2018.)

Metodin eristämistä käytetään pilkkomaan pitkiä koodiosioita pienempiin osiin. Jos koodista on vaikeaa saada selville, mitä se tekee, sen pilkkominen osiin on hyvä ratkaisu. Eristetty metodi nimetään kuvaavasti, jolloin nimi kertoo sen käyttötarkoituksen, eikä ohjelmoijan välttämättä tarvitse keskittyä itse metodin koodin toteutukseen selvittääkseen mitä se tekee. Tämä tekee koodista itsensä dokumentoivaa. Usein kommentteja sisältävä pitkä metodi kertoo siitä, että se olisi hyvä pilkkoa metodeihin, joiden nimet korvaavat samalla kommenttien tarpeen. Metodin eristämistä voidaan käyttää myös tapauksessa, jossa kaksi koodiosiota tekevät saman asian, eli koodi on monistettua. Silloin koodi voidaan eristää omaksi metodikseen, jota kutsutaan sitä käyttävissä paikoissa. Sama onnistuu,

jos monistetut koodiosiot ovat alaluokissa, jotka jakavat yhteisen yläluokan. Tällöin monistettu osa voidaan nostaa osaksi yläluokkaa ja alaluokista voidaan kutsua yläluokassa sijaitsevaa metodia. (Fowler & Beck 2018.)

Koodilausekkeista voi muodostua hyvin monimutkaisia, jolloin niitä on vaikeaa seurata ja lukea. Tällöin paikallisen muuttujan luominen auttaa pilkkomaan lauseketta ymmärrettävämpään muotoon. Esimerkiksi pitkässä laskukaavassa lausekkeen osat voidaan eristää muuttujiin. Muuttujien kuvaavat nimet helpottavat tällöin lausekkeen lukemista. Joskus paikalliset muuttujat kuitenkin vaikeuttavat refaktoroimista, eivätkä niiden tuomat nimet tuo lisäarvoa koodin ymmärtämiseen. Silloin lauseketta voidaan refaktoroida toisinpäin, eli muuttaa paikallinen muuttuja osaksi lauseketta. (Fowler & Beck 2018.)

Joskus metodille annetaan useita parametreja. Jos kyseiset parametrit ovat osa samaa objektia, voidaan ne korvata lähettämällä metodille kokonainen objekti. Tämä selkeyttää metodikutsujen tekemistä. Metodin sisällä tarvittavat parametrit haetaan käyttämällä sille lähetetyn objektin metodeja. Vaikka metodi ei käyttäisi kaikkia objektin sisältämiä arvoja, parametrilistan korvaaminen objektilla on usein hyvä ratkaisu. (Fowler & Beck 2018.)

Globaali, eli koko tiedostoa koskeva data aiheuttaa usein ongelmia. Globaalia dataa voidaan muokata mistä tahansa käsin, eikä ole keinoa selvittää mikä koodiosio sitä on muokannut. Tämä aiheuttaa virheitä, joiden syyn selvittäminen voi olla todella hankalaa. Globaalin datan eristäminen on usein paras ratkaisu. Eristämisellä tarkoitetaan globaalin muuttujan refaktoroimista niin, että sitä muokataan käyttäen metodeja. Tällöin muuttujan näkyvyyttä voidaan rajoittaa ja tehdä myös datan validointia sen asettamisen yhteydessä. (Fowler & Beck 2018.)

3 AULAPALVELU-JÄRJESTELMÄ

3.1 Järjestelmän tausta

Aulapalvelu-järjestelmän on alun perin kehittänyt Tampereella ja Helsingissä toimiva ohjelmistotalo BearIT Oy. Tampereen toimitilat sijaitsevat Tulli Business Parkissa, joka on Spondan omistama toimitilarakennus Tampereella. Rakennuksessa on useiden yritysten toimitilojen lisäksi neuvotteluhuoneita, joita vuokrataan sekä talon sisäisten että ulkoisten yritysten käyttöön. Tulli Business Parkin henkilökunta hoitaa neuvotteluhuoneiden varausten tekemisen ja yrityksiin saapuvien vierailijoiden sisään kirjaamisen vastaanotossa. Ennen Aulapalvelu-järjestelmää vierailijoiden sisään kirjaaminen tapahtui paperisilla lomakkeilla ja neuvotteluhuoneiden varaukset merkittiin paperisiin kalentereihin. Henkilökunta kaipasi parempaa tapaa hoitaa nämä tehtävät, jolloin BearIT Oy kehitti Aulapalvelu-järjestelmän Tulli Business Parkin käyttöön heinäkuussa 2016.

Nykyään järjestelmä on korvannut paperiset lomakkeet ja kalenterit. Vierailijat käyttävät järjestelmää tabletilla, jolla he täyttävät vierailijalomakkeen vastaanotossa saapuessaan Tulli Business Parkiin. Vastaanoton henkilökunta käyttää järjestelmää web-käyttöliittymällä, josta vierailijatietoja voidaan tarkastella ja ylläpitää. Vieraiden kirjaamisen lisäksi järjestelmää käytetään neuvotteluhuoneiden varausten tekemiseen ja ylläpitoon.

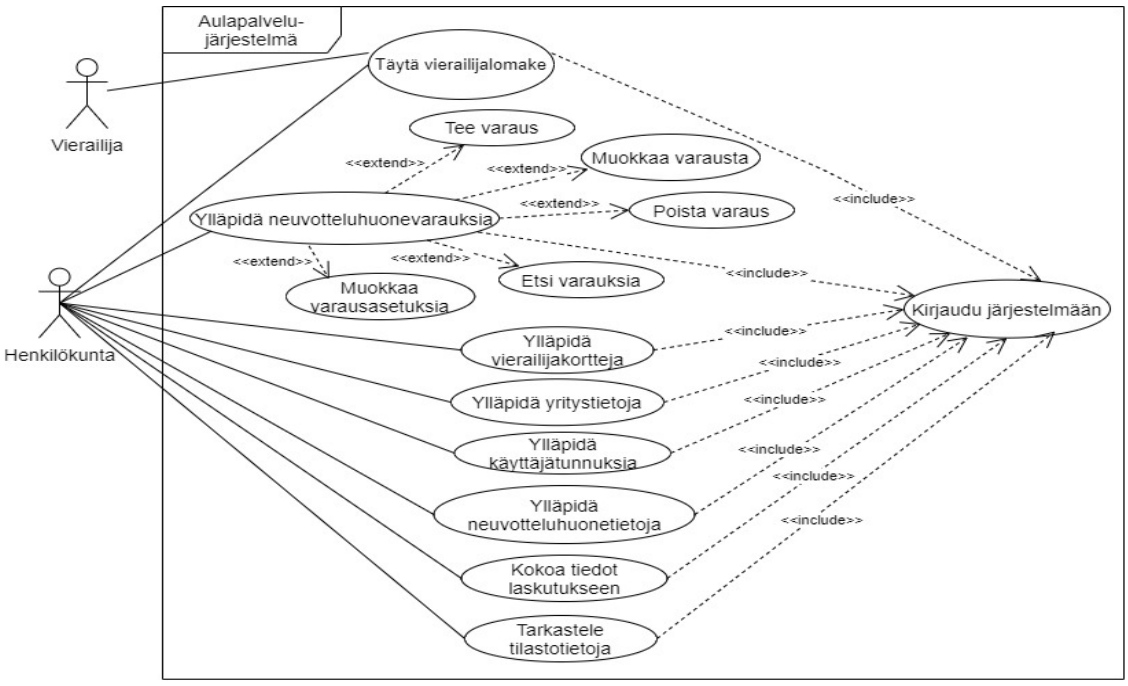
Aulapalvelu-järjestelmä on helpottanut ja nopeuttanut henkilökunnan päivittäisiä töitä paljon, mutta siitä puuttuu vielä joitain toivottuja ominaisuuksia. Lisäksi joihinkin olemassa oleviin ominaisuuksiin on kaivattu toiminnallisia muutoksia. Näistä syistä järjestelmää päätettiin jatkokehittää. Myös järjestelmän käytön yhteydessä esiin nousseiden virhetilanteiden korjaaminen sisältyy jatkokehitysprojektiin.

3.2 Järjestelmän toiminnallinen kuvaus

Aulapalvelu-järjestelmä sisältää kaksi roolia, *ADMIN* eli ylläpitäjä ja *USER* eli käyttäjä. Käyttäjätunnuksilla, joiden rooliksi on asetettu ylläpitäjä, kirjaudutaan järjestelmän henkilökunnan näkymään. Tunnuksilla, joiden rooli on käyttäjä, kirjaudutaan vierailijanäkymään. Järjestelmän käyttäjätunnuksia ja niiden rooleja hallinnoidaan ylläpitäjätunnuksilla.

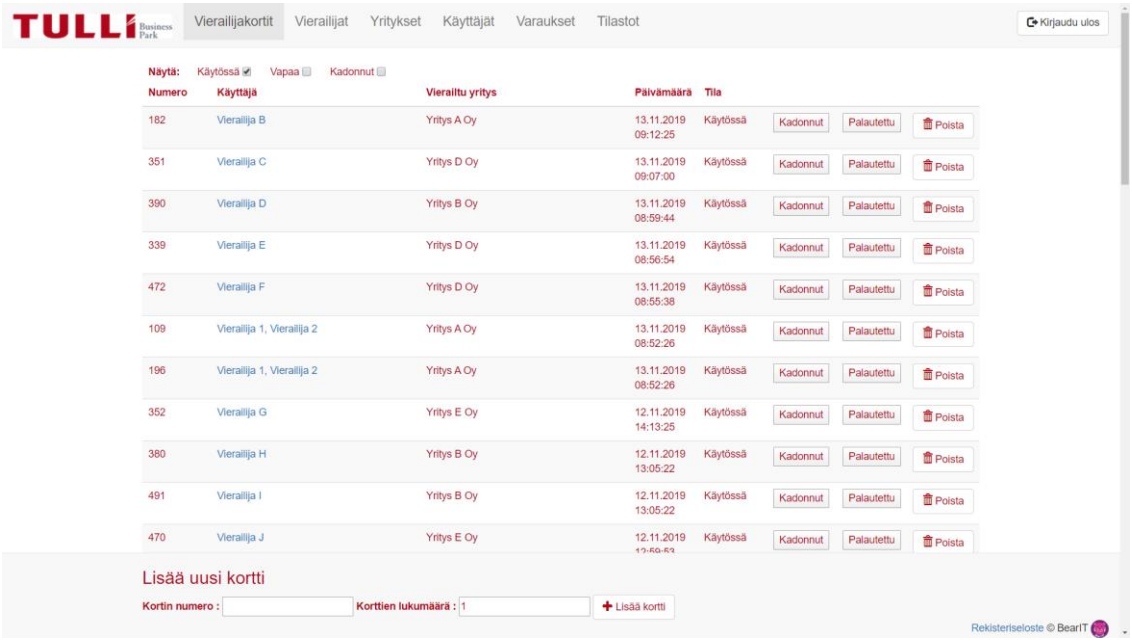
Henkilökunnan näkymää käytetään web-selaimella. Näkymässä voi tarkastella ja ylläpitää vierailijakorttien, vierailijoiden ja Tulli Business Parkissa sijaitsevien yritysten tietoja sekä tehdä neuvotteluhuonevarauksia. Näkymästä löytyy myös tilastot-välilehti, joka sisältää tilastotietoja kävijöiden määrästä ja neuvotteluhuoneiden käytöstä.

Kuviossa 1 on esitelty järjestelmän toimintaa käyttötapauskaavion avulla. Järjestelmän tärkeimmät käyttötapaukset ovat vierailijoiden sisään kirjaaminen ja neuvotteluhuoneiden varausten ylläpito. Käyttötapauskaavion toimijoita ovat vierailija, eli USER-tunnuksella järjestelmää käyttävä henkilö ja henkilökunta, joka käyttää järjestelmää ADMIN-tunnuksella.



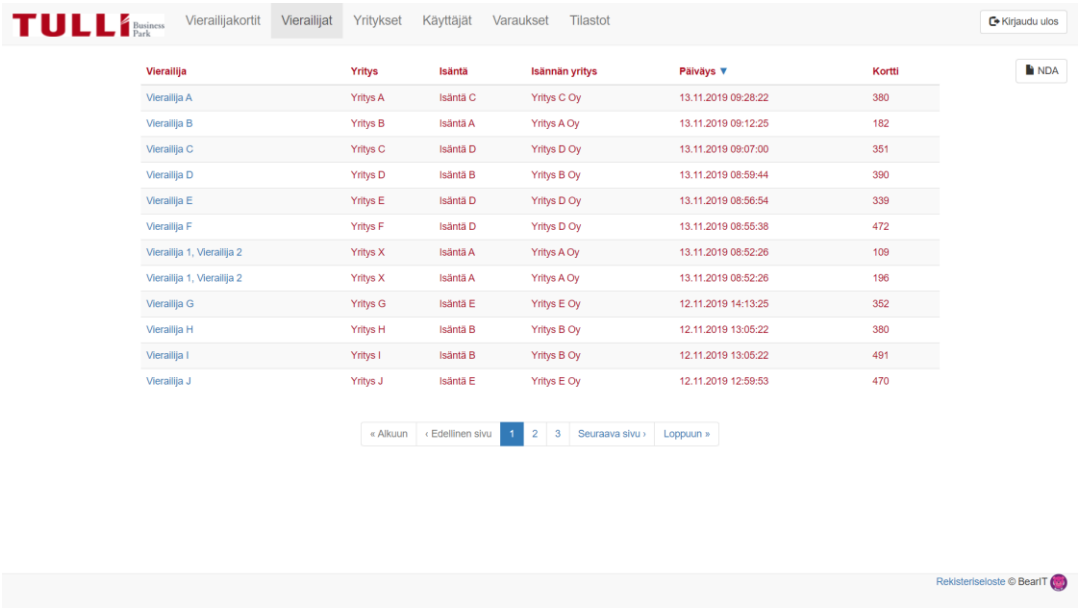
KUVIO 1. Käyttötapauskaavio Aulapalvelu-järjestelmästä

Tulli Business Parkiin saapuville vieraille annetaan vastaanotossa vierailijakortti sisään kirjautumisen yhteydessä. Kuvassa 1 esitetyssä vierailijakortit-näkymässä voi tarkastella käytössä olevia, vapaita ja kadonneita vierailijakortteja. Kortin tiedoissa näkyy sen viimeisin käyttäjä, vierailtu yritys, vierailun ajankohta sekä kortin tila. Näytettäviä kortteja voi suodattaa kortin tilan perusteella ja järjestää päivämäärän, numeron, tilan sekä käyttäjän tai vieraillun yrityksen mukaan. Vierailijan nimeä klikkaamalla pääsee tarkastelemaan ja muokkaamaan vierailijan täyttämää vierailijalomaketta. Näkymässä voi merkitä kortin kadonneeksi tai palautetuksi ja tarvittaessa poistaa kortin tiedot järjestelmästä. Näkymän alalaidassa olevalla lomakkeella voi lisätä uuden vierailijakortin järjestelmään.



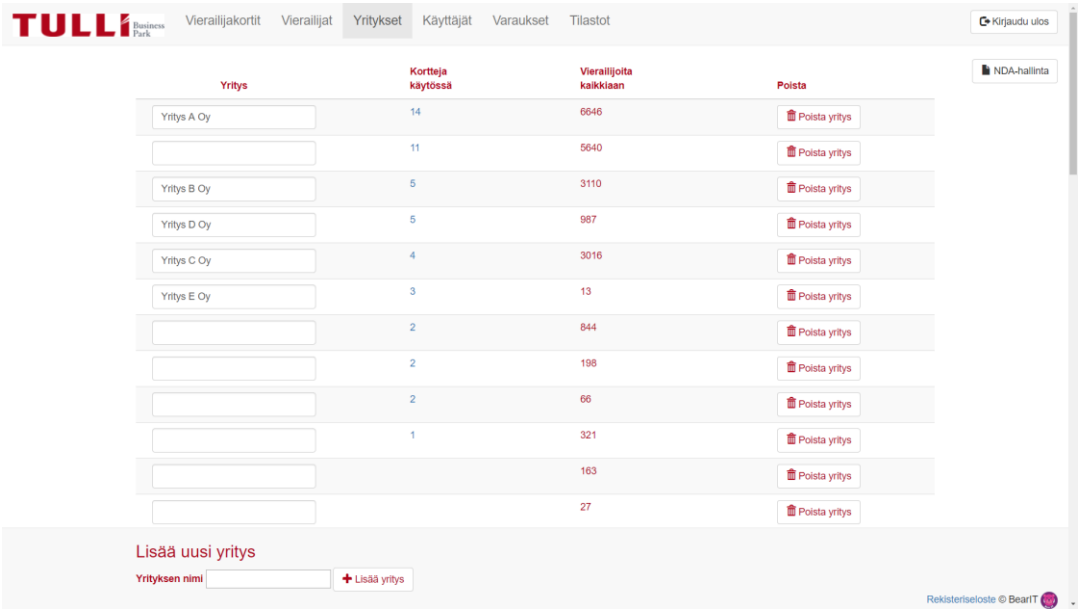
KUVA 1. Näyttökuva Aulapalvelu-järjestelmän vierailijakortit-näkymästä

Kuvassa 2 olevassa vierailijat-näkymässä on listattu Tulli Business Parkissa käyneet henkilöt. Listassa näkyy vierailija nimi, tämän edustama yritys, vierailijakortin numero, vierailun isäntä ja isännän yritys sekä vierailun ajankohta. Lista voidaan järjestää päivämäärän mukaan joko laskevasti tai nousevasti. Samoin kuin vierailijakortit-näkymässä, vierailijan nimeä klikkaamalla voi muokata ja tarkastella vierailijalomaketta. Vaikka vierailijan käyttämän kortin tiedot olisi poistettu järjestelmästä, vierailijat-näkymässä vierailijan tiedot ja käytössä olleen kortin numero säilyvät.



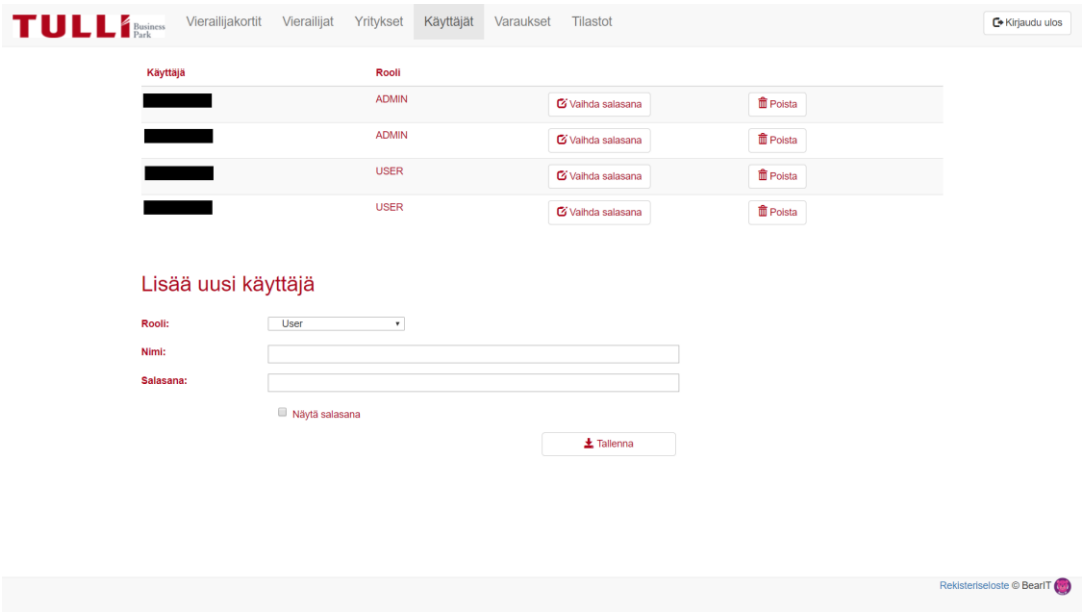
KUVA 2. Näyttökuva Aulapalvelu-järjestelmän vierailijat-näkymästä

Kuvassa 3 on yritykset-näkymä, jossa voidaan tarkastella ja ylläpitää Tulli Business Parkissa sijaitsevien yritysten tietoja. Yrityksen nimeä voidaan muokata tai tarvittaessa sen tiedot voidaan poistaa järjestelmästä. Näkymän alalaidassa on lomake, jolla lisätään uusia yrityksiä järjestelmään. Yrityksen tiedoissa näkyy sen käytössä olevien vierailijakorttien määrä sekä yrityksessä vierailleiden henkilöiden määrä. Korttien määrää klikatessa aukeaa ikkuna, johon on listattu yrityksen käytössä olevien korttien numerot, käyttäjät ja käytön ajankohta.



KUVA 3. Näyttökuva Aulapalvelu-järjestelmän yritykset-näkymästä

Kuvassa 4 esitetyssä käyttäjät-näkymässä on listattu järjestelmässä käytössä olevat käyttäjätunnukset ja niiden roolit. Tunnusten salasanoja voi vaihtaa ja tunnuksia voi poistaa näkymästä käsin. Näkymä sisältää lomakkeen, jolla järjestelmään voidaan lisätä uusia käyttäjätunnuksia. Kaikkia käyttäjätunnuksia voidaan hallinnoida kirjautumalla järjestelmään ylläpitäjäroolin tunnuksilla. Käyttäjätunnukset eivät ole henkilökohtaisia vaan kaikki vastaanottovirkailijat käyttävät samoja tunnuksia ja vierailijanäkymään kirjaudutaan käyttäen samaa käyttäjäroolin tunnusta. Järjestelmässä näkyvät tiedot eivät myöskään ole käyttäjäkohtaisia, vaan kaikilla ylläpitäjätunnuksilla pääsee tarkastelemaan samoja tietoja ja käyttämään järjestelmää samoin tavoin.

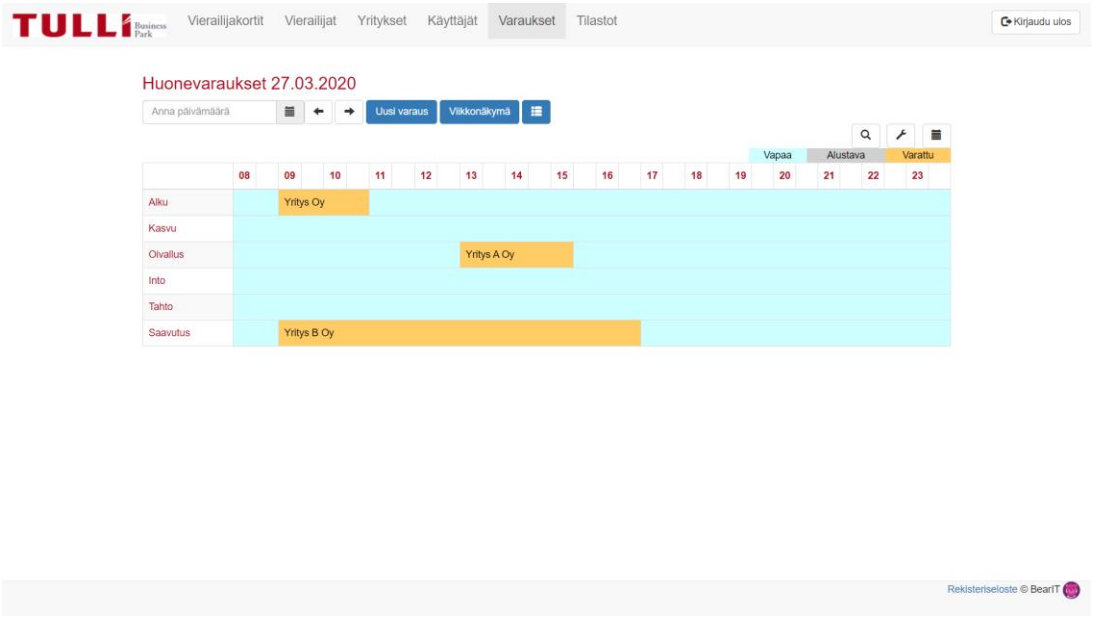


KUVA 4. Näyttökuva Aulapalvelu-järjestelmän käyttäjät-näkymästä

Kuvan 5 varaukset-näkymässä on varauskalenteri, jota käytetään Tulli Business Parkin neuvotteluhuoneiden varausten luomiseen ja ylläpitoon. Varauskalenteri on jaettu puolentunnin osioihin ja sen jokainen rivi sisältää yhden neuvotteluhuuoneen päivän varaukset. Uusi varaus luodaan klikkaamalla joko ”Uusi varaus” -painiketta tai varauskalenterin riviä varattavan neuvotteluhuoneen kohdalta. Tällöin avautuu varauslomake, johon varauksen tiedot syötetään. Varausta voi muokata klikkaamalla varausta kalenterista jolloin varauslomake aukeaa. Varaus poistetaan samaa lomaketta käyttäen.

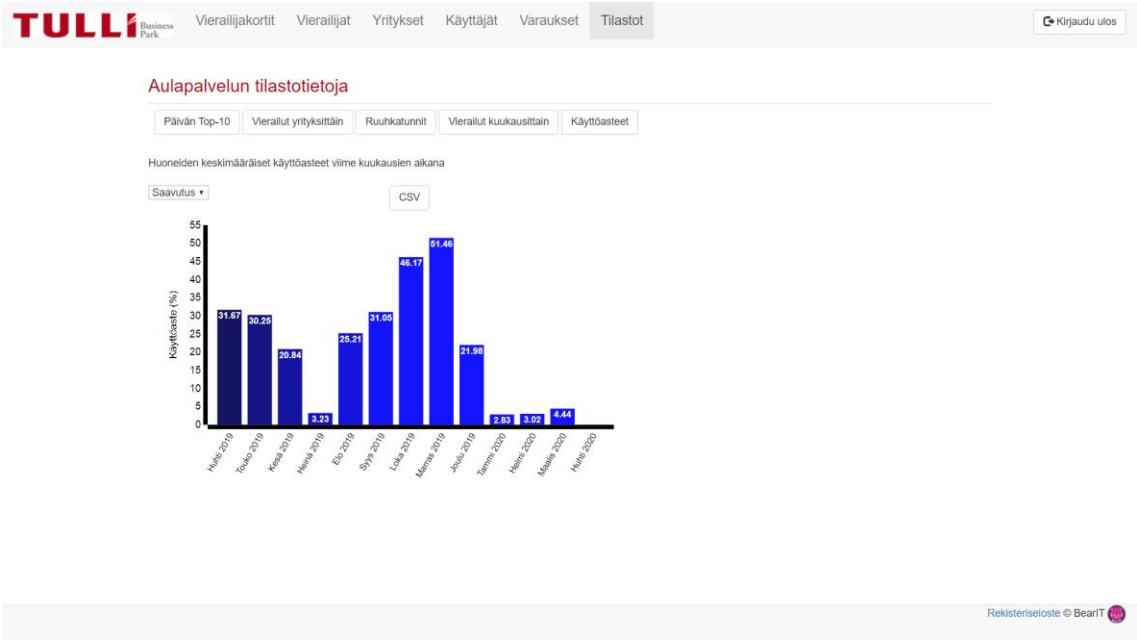
Vahvistetut varaukset näkyvät kalenterissa oranssilla pohjalla. Varaus täyttää eri kokoisen alueen kalenterista riippuen sen kestosta. Kuvassa 5 Yritys Oy on tehnyt Alku-neuvotteluhuoneeseen varauksen, joka alkaa kello yhdeksältä ja päättyy kello yksitoista. Varauskalenterin ylälaidasta näkyy, että alustavaksi merkityt varaukset näkyisivät kalenterissa harmaalla taustalla, mutta ominaisuutta ei ole toteutettu. Varauskalenterin näyttämää päivää voi vaihtaa sen yläpuolella olevista nuolipainikkeista tai valitsemalla haluamansa päivän kalenterista, joka aukeaa klikkaamalla ”Anna päivämäärä”-kenttää.

Varauskalenteria voi tarkastella myös viikkonäkymässä, jolloin se näyttää yhden neuvotteluhuoneen kokonaisen viikon varaukset. Viikkonäkymä-painikkeen vieressä olevaa lista-ikonia painamalla voi lähettää yhteenvedon valitsemansa aikavälin varauksista haluamaansa sähköpostiosoitteeseen. Varauksia voi etsiä joko varaajan tai yrityksen nimen perusteella klikkaamalla suurennuslasi-ikonia, jolloin siirrytään hakunäkymään. Jakoavain-ikonia klikkaamalla aukeaa näkymä, jossa voi muokata varauksiin liittyviä asetuksia, kuten peruutusehtoja, jotka liitetään varauksen vahvistussähköpostiin. Kahden aiemman ikonin vieressä olevaa kalenteri-ikonia klikkaamalla pääsee näkymään, jossa voi tarkastella kaikkien neuvotteluhuoneiden kokonaisen viikon varauksia yhtäaikaisesti.



KUVA 5. Näyttökuva Aulapalvelu-järjestelmän varaukset-näkymästä

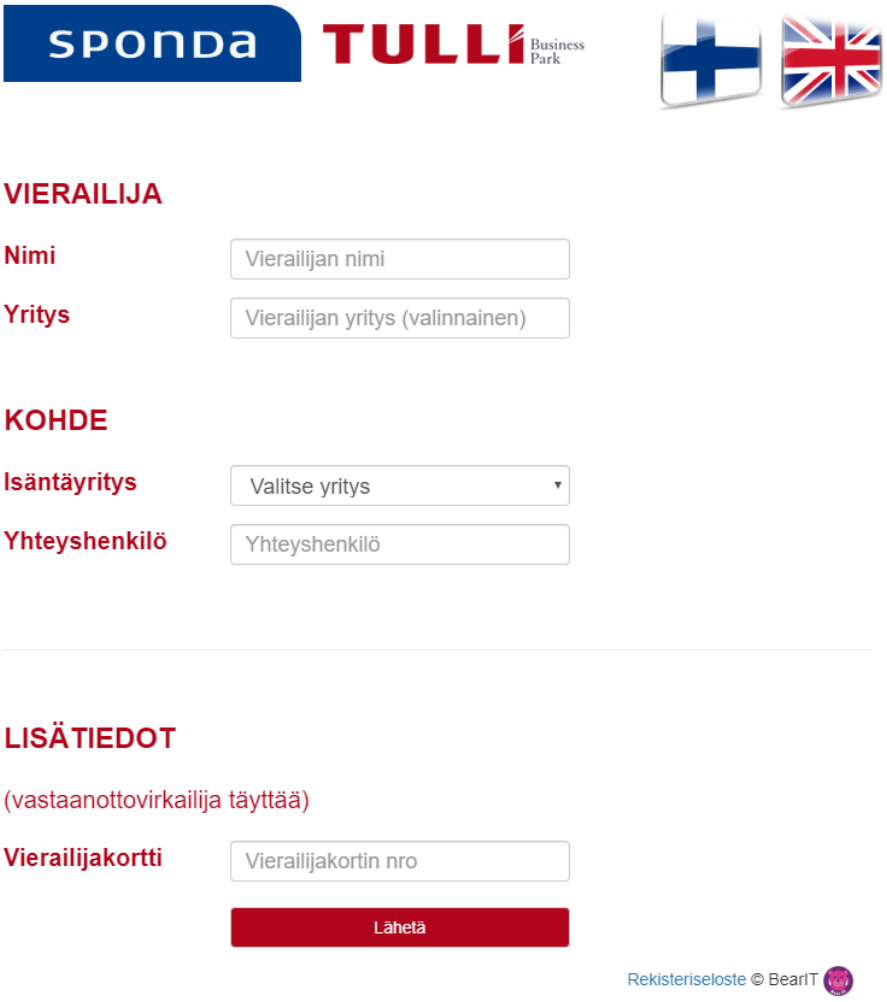
Tilastot-näkymä sisältää tilastotietoja Tulli Business Parkin yrityksiin tehdyistä vierailuista ja neuvotteluhuoneiden käytöstä. Näkymästä voi katsoa kuluvan päivän eniten vieraillut yritykset, vierailijoiden kokonaismäärän kuukaudessa joko yrityksittäin tai yhteensä sekä ruuhkatunnit, jolloin Tulli Business Parkissa vieraillaan eniten. Lisäksi näkymästä voi tarkastella neuvotteluhuoneiden käyttöasteita. Käyttöaste on prosenttiluku, joka muodostuu neuvotteluhuoneen käyttötunneista kuukauden aikana suhteessa käytettävissä olleisiin tunteihin. Kuvassa 6 on näkyvissä Saavutus-neuvotteluhuoneen käyttöasteet. Käyttöasteiden yhteenvedon voi lähettää haluamaansa sähköpostiosoitteeseen. Käyttöastetietoja tarvitaan neuvotteluhuonevarausten yhteenvedossa, jonka Tulli Business Parkin henkilökunta tekee kuukausittain.



KUVA 6. Näyttökuva Aulapalvelu-järjestelmän tilastot-näkymästä

Siirtyäkseen vierailijanäkymään Tulli Business Parkin vastaanoton virkailija kirjautuu Aulapalvelu-järjestelmään tunnuksella, jonka rooli on käyttäjä. Tällöin avautuu kuvassa 7 nähtävä näkymä, joka sisältää vierailijalomakkeen. Vierailijanäkymää käytetään vastaanotossa tabletilla, jolla vierailija täyttää lomakkeen

saapuessaan Tulli Business Parkiin. Vierailija täyttää lomakkeeseen omat tietonsa sekä kohdeyrityksen tiedot. Samaan lomakkeeseen voidaan kirjata useampi vierailija, joten jokaisen samassa ryhmässä saapuneen ei välttämättä tarvitse täyttää lomaketta erikseen. Vierailun kohdeyrityksen nimi valitaan pudotusvalikosta, joka sisältää järjestelmän yritykset-näkymässä listatut yritykset. Vastaanottovirkailija täyttää lomakkeeseen vierailijalle annetun vierailijakortin numeron ja lähettää lomakkeen, jolloin järjestelmään kirjataan tieto vierailijasta ja annetusta vierailijakortista.



KUVA 7. Näyttökuva Aulapalvelu-järjestelmän vierailijanäkymästä

3.3 Järjestelmän tekninen kuvaus

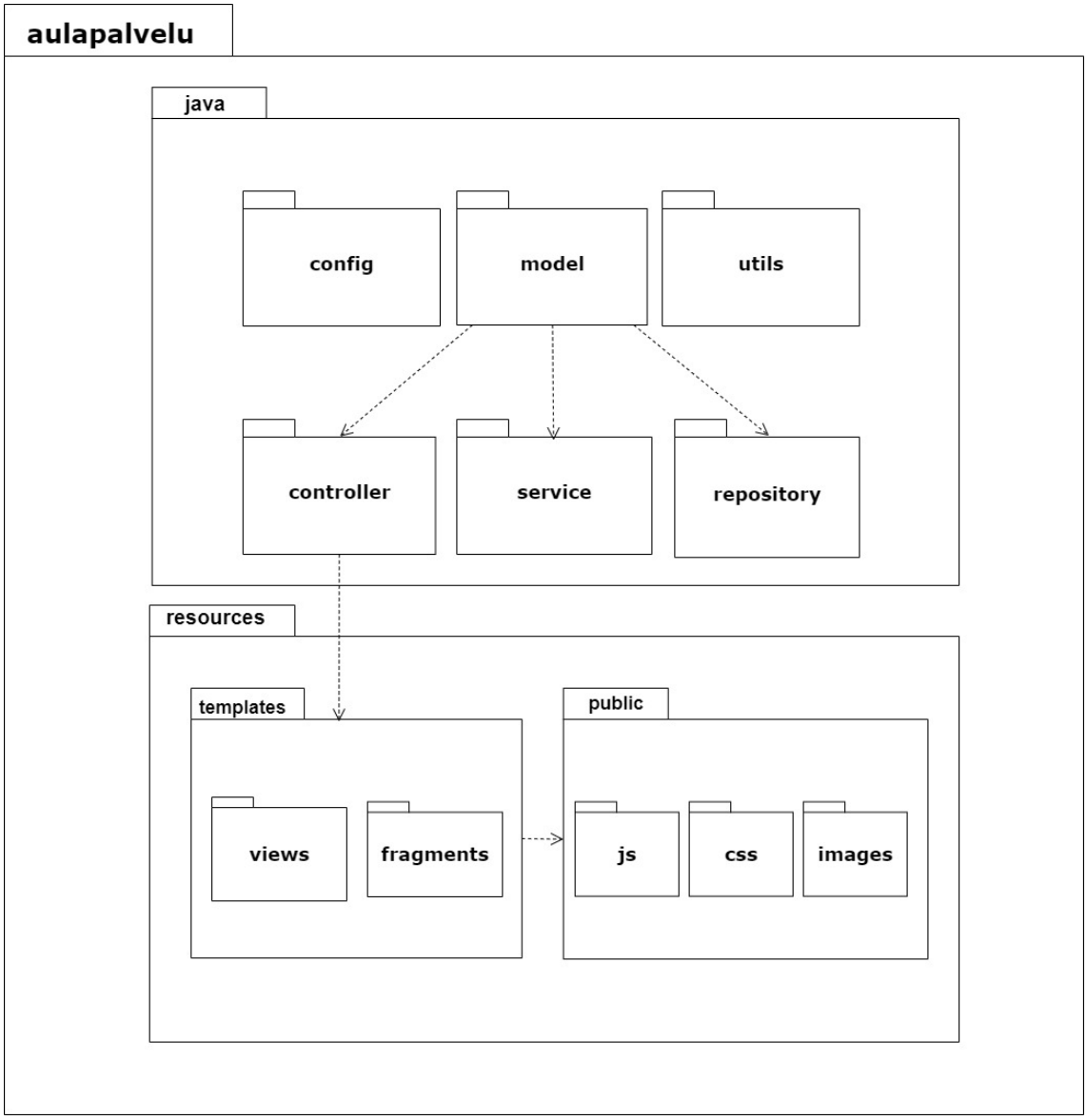
Aulapalvelu-järjestelmä on toteutettu Java-versiolla 1.8 käyttäen Spring Bootia. Spring Boot on rakennettu Spring-ohjelmistokehyksen päälle helpottamaan ja nopeuttamaan sovellusten alkutoimien tekemistä, konfigurointia sekä ajamista. Spring-ohjelmistokehystä käyttäessä konfiguraatio hoidetaan itse, mutta Spring Boot-projektissa konfiguraatio on automatisoitu. Spring Boot-sovellusta ei tarvitse asentaa erilliselle web-serverille tai muuhun ympäristöön, vaan se pyörii sisäänrakennetulla serverillä, jonka käynnistämisen ja konfiguroinnin Spring Boot hoitaa. (Raffai, 2018.)

Järjestelmä käyttää JPA-rajapinnan ja Spring Data JPA:n avulla MySQL-tietokantaa. Java Persistence API (JPA) hoitaa datan hallintaa ja määrittelee tavan Java-objektien muuntamiseen relaatiotietokantaan sopivaksi dataksi ja toisinpäin. Tätä kutsutaan termillä Object-Relational Mapping eli ORM. (Fadatare n.d). Spring Data JPA helpottaa datan käsittelyä entisestään mahdollistamalla JPA-pohjaisten repositorien luomisen vähällä vaivalla. Sen avulla CRUD-operaatioiden toteuttaminen onnistuu helposti luomalla oman repositorio-rajapinnan, joka laajentaa CrudRepository-rajapintaa. (Ugarte 2020). CRUD-operaatioilla tarkoitetaan tietojen luomiseen, lukemiseen, päivittämiseen ja poistamiseen tarkoitettuja tietokantatoimintoja (Stackify 2017).

Käyttäjän näkemät sivut palautetaan REST-rajapinnan yli. Useimmat näkymät muodostetaan käyttäen Thymeleaf-sivupohjamoottoria. Thymeleaf on kirjoitettu Javalla ja sen avulla voidaan tehdä XML-, XHTML- tai HTML5-sivupohjia (Anjana 2015). Järjestelmän näkymät ovat HTML-pohjaisia. Joidenkin sivujen sisältöä päivitetään käyttöliittymästä tehdyillä AJAX-kutsuilla. AJAX on malli, joka hyödyntää useita teknologioita mahdollistaen web-sovelluksen käyttöliittymän päivittämisen ilman koko selainsivun päivittämistä, tehden sovelluksesta nopeamman ja vastaanottavamman käyttäjän toimintaan (MDN web docs 2020). Järjestelmän käyttöliittymä on responsiivinen, eli se on käytettävissä eri kokoisilla näyttölaitteilla. Responsiivisuus on toteutettu Bootstrap-kirjastoa käyttäen.

Kuviossa 2 on esitelty Aulapalvelu-järjestelmän korkeamman tason arkkitehtuurin rakenne ja elementtien välisiä riippuvuuksia pakkauskaavion avulla. Järjestelmä

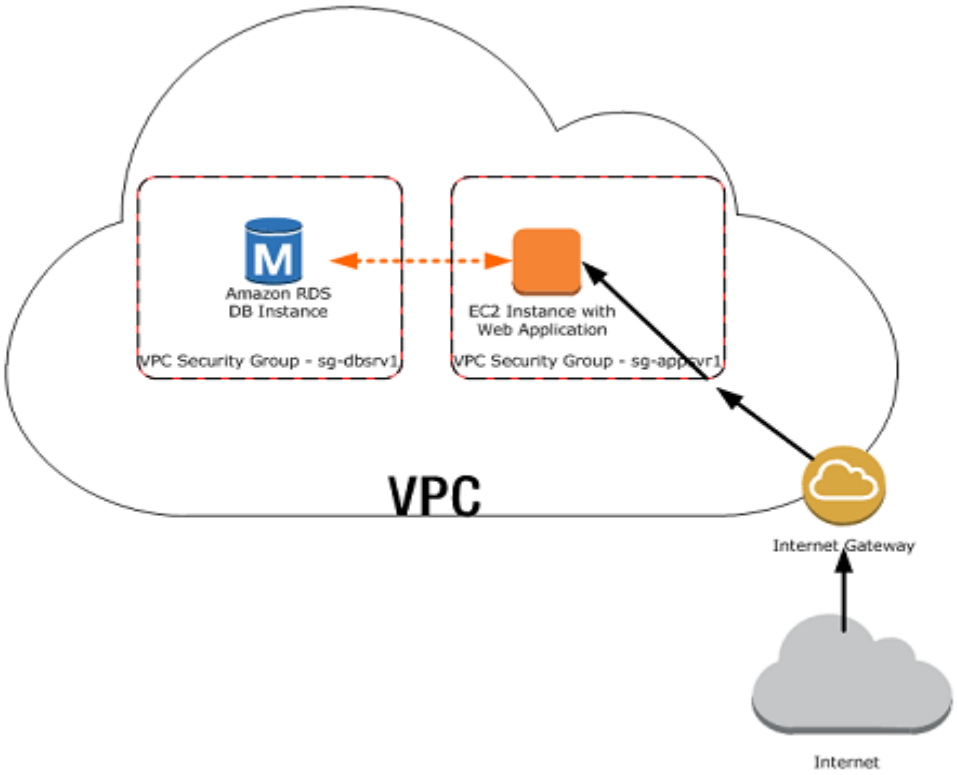
on jaettu kahteen pakettiin, joista toinen sisältää backend-rakenteen ja toinen käyttöliittymän komponentit. Backend-paketin sisällä olevat paketit sisältävät Java-objektit, kontrollerit, palvelut, repositoriot, konfiguraation ja muut hyödyksi käytettävät komponentit. Käyttöliittymäpaketti sisältää kaksi pakettia, joista toinen sisältää järjestelmän näkymät ja toinen JavaScript- ja CSS-tiedostot sekä kuvat.



KUVIO 2. Pakkauskaavio Aulapalvelu-järjestelmän rakenteesta

Järjestelmän tuotantopalvelin ja tietokanta sijaitsevat Amazon Web Services (AWS) pilvilaskenta-alustalla. Järjestelmä on asennettu Amazon Elastic Compute Cloud-serverille eli lyhyemmin EC2-serverille. EC2 on palvelu, joka tarjoaa skaalautuvaa laskentakapasiteettia pilvessä (Amazon Web Services. n.d.a). Järjestelmän tuotantotietokanta on AWS:n RDS-palvelussa. Amazon Relational Database

Service (RDS) on tietokantapilvipalvelu, jonka avulla voidaan luoda ja käyttää relaatiotietokantoja (Tutorialspoint. n.d.). Kuviossa 3 on esitelty järjestelmän AWS-konfiguraatiota vastaava malli.



KUVIO 3. Aulapalvelu-järjestelmän AWS-konfiguraatiota vastaava malli (Amazon Web Services. n.d.b)

4 JÄRJESTELMÄN JATKOKEHITYS

4.1 Määrittely ja suunnittelu

Aulapalvelu-järjestelmän jatkokehitysprojekti aloitettiin määrittelypalaverilla, johon osallistui toimeksiantajan edustajan lisäksi järjestelmän loppukäyttäjiä, eli Tulli Business Parkin vastaanoton henkilökuntaa. Määrittelypalaverin tavoitteena oli kartoittaa loppukäyttäjien jatkokehitystoiveita. Palaverissa myös toimeksiantaja kertoi näkemyksiään siitä, millaisia muutoksia ja lisäominaisuuksia järjestelmään voitaisiin toteuttaa.

Määrittelypalaverissa loppukäyttäjät näyttivät, miten ja mihin tarkoituksiin he käyttävät järjestelmää. Vaikka järjestelmä on alun perin kehitetty vastaanoton henkilökunnan kanssa yhteistyössä, on heille vakiintuneet heidän parhaiksi kokemansa käyttötavat järjestelmän päivittäisen käytön myötä. Näiden käyttötapojen pohjalta mietittiin, miten järjestelmää tulisi jatkokehittää, jotta se vastaisi paremmin käyttötarkoitustaan. Loppukäyttäjät kertoivat myös työtehtävistä, joiden he toivoivat hoituvan järjestelmän avulla ja näyttivät, miten kyseiset tehtävät sillä hetkellä toteutettiin järjestelmän ulkopuolella.

Palaverissa huomattiin, että järjestelmän suurimmat ongelmat ja puutteet liittyivät neuvotteluhuoneiden varauksia koskeviin toimintoihin. Lisäksi loppukäyttäjät toivoivat, että järjestelmän avulla voitaisiin hoitaa neuvotteluhuoneiden laskutukseen tarvittavat yhteenvedot huoneiden kuukausittaisista tuotoista ja käyttöasteista. Toivottuihin muutoksiin ja ominaisuuksiin tutustutaan tarkemmin tämän työn luvussa viisi, jossa käsitellään jatkokehityksen tuloksia.

Määrittelypalaverissa esiin nousseet asiat ja jatkokehitystoiveet käytiin myöhemmin läpi suunnittelupalaverissa toimeksiantajan kanssa. Järjestelmän toiminnallisuuksia tarkasteltiin vielä uudestaan ja mietittiin millaisia muutostöitä kunkin toivotun ominaisuuden tai muutoksen toteutus vaatii käytännössä. Loppukäyttäjien kertomien asioiden pohjalta suunniteltiin, miten ominaisuudet toteutetaan käyttöliittymään. Varsinaista käyttöliittymäsuunnittelua ei tehty, sillä muutokset ja uudet

toiminnallisuudet saatiin hyvin suunniteltua niin, että ne noudattivat olemassa olevan käyttöliittymän toimintatapoja. Uusien elementtien asettelua käyttöliittymässä suunniteltiin toimeksiantajan kanssa, mutta niiden ulkoasu määräytyi olemassa olevan käyttöliittymän tyylien perusteella.

Aulapalvelu-järjestelmän laajempaa käyttöönottoa muissakin toimitilarakennuksissa kuin Tulli Business Park on suunniteltu alustavasti. Järjestelmä on toteutettu Tulli Business Parkin tarpeiden pohjalta, joten se ei kuitenkaan nykyisen rakenteensa ja toimintatapojensa puolesta ole suoraan käyttöönotettavissa muualla. Toimeksiantajan toiveena oli, että jatkokehitysprojektin yhteydessä järjestelmän rakennetta kartoitettaisiin ja mahdollisuuksien mukaan refaktoroitaisiin kovakoodattuja ratkaisuja, jotta järjestelmä olisi mukautuvampi muihinkin käyttötarkoituksiin.

4.2 Projektin toteutus

Suunnittelupalaverin jälkeen jatkokehitysprojektin työtehtävät kirjattiin projektinhallintatyökalun Kanban-tauluun, joka toimi projektin kehitysjonona eli backlogina. Taulun avulla seurattiin, mitkä tehtävät olivat työn alla, testauksessa ja valmistuneet. Lisäksi projektin etenemistä seurattiin toimeksiantajan kanssa viikoittaisissa palavereissa, joissa käytiin läpi projektin tilanne ja tarpeen mukaan tarkennettiin työtehtäviä. Myös järjestelmän loppukäyttäjiin oltiin tarvittaessa yhteydessä koko projektin ajan. Heiltä pyydettiin tarkennuksia ominaisuuksien toimintaan liittyen ja tarvittavia lisämateriaaleja, kuten neuvotteluhuoneiden hintatietoja.

Jatkokehitys tapahtui omassa kehitysympäristössään ja käytössä oli lokaali tietokanta. Järjestelmän ohjelmistokoodi haettiin versionhallinnasta ja tietokanta luotiin kopioimalla järjestelmän tuotantokannan rakenne. Testidata lokaaliin tietokantaan haettiin järjestelmän testausta varten olevasta tietokannasta. Tietokannan luomiseen ja hallintaan käytettiin MySQL Workbenchia, joka on visuaalinen työkalu MySQL-tietokantojen käsittelyyn.

Ohjelmointiympäristöksi valittiin JetBrainsin IntelliJ IDEA. Se on integroitu ohjelmistoympäristö, eli IDE. Integroidussa ohjelmistoympäristössä ohjelmistokehityksessä tarvittavat työkalut on yhdistetty yhdeksi sovellukseksi, jolla ohjelmoija voi kirjoittaa, kääntää ja testata koodia helposti (codecademy n.d.). IntelliJ IDEA helpottaa lisäksi koodin refaktoroimista. Se sisältää useita automatisoituja refaktoroimiseen tarkoitettuja työkaluja, joiden avulla voidaan esimerkiksi nimetä metodeja uudestaan tai erottaa osa metodista omaksi kokonaisuudekseen. Työkalu hoitaa refaktoroimisen, joten kaiken voi luottaa toimivan halutulla tavalla muutosten jälkeenkin (Fowler & Beck 2018).

Uusien ja päivitettyjen ominaisuuksien valmistuttua ne siirrettiin testipalvelimelle, jossa niiden toimivuus varmistettiin ennen tuotantopalvelimelle siirtoa. Testipalvelin vastaa tuotantopalvelinta ja se pyörii samoin AWS-pilvilaskenta-alustalla. Valmistuneita ominaisuuksia ei siirretty testipalvelimelle yksi kerrallaan, vaan järjestelmäversioittain kahdessa osassa. Ensimmäinen versio sisälsi lähinnä virheiden korjauksia, joiden saaminen nopeasti tuotantoon oli tärkeää järjestelmän käytön kannalta. Toinen versio sisälsi uusia ominaisuuksia ja isompaa päivitystyötä vaatineet ominaisuudet. Toista versiota ei aikataulullisista syistä siirretty tuotantoon varsinaisen jatkokehitysprojektin aikana, mutta se testattiin ja todettiin toimivaksi.

4.3 Käytetyt menetelmät

Jatkokehitysprojektin varsinainen toteutus alkoi järjestelmän tekniseen toimintaan ja olemassa olevaan koodin tutustumisella. Tutustumisessa apuna käytettiin refaktoroimista, koska järjestelmän koodia oli välillä hankala ymmärtää. Järjestelmän jatkokehitys jakaantui pääosin ominaisuuksien lisäämiseen ja muokkaamiseen sekä koodin refaktoroimiseen. Refaktoroiminen keskittyi kuitenkin lähinnä niihin järjestelmän osiin, joita jatkokehitettiin, joten suurin osa refaktoroinneista koski frontend-koodia. Jatkokehitysprojektin yhteydessä toteutettiin myös joitain virheenkorjauksia.

Järjestelmän backend-koodin jatkokehitys keskittyi lähinnä uusia ominaisuuksia varten tarvittavien metodien ja luokkien lisäämiseen. Vaikka backend-koodin rakenne ei kaikkialta ole kovin selkeä ja koodi on paikoittain vaikealukuista, sen refaktoroimiseen ei käytetty kovin paljoa aikaa. Järjestelmän jatkokehitys koski pääosin sen frontend-koodiin tehtäviä lisäyksiä ja muutoksia, joten backend-koodin refaktoroimiseen liiallisen ajan käyttäminen ei ollut kannattavaa.

Tietokantaan luotiin uusia tauluja ja näkymiä, joita tarvittiin uusien ominaisuuksien toteuttamiseen. Myös joihinkin olemassa olleiden taulujen rakenteisiin ja taulujen välisiin yhteyksiin tehtiin muutoksia, jotka mahdollistivat toiminnallisuuksien lisäämisen. Suurin tietokantaa koskenut muutos oli järjestelmän tilastoja käsittelevien näkymien uudelleen koostaminen. Tilastojen laskutavoissa oli ongelmia ja näkymien muodostamisessa käytetyt lausekkeet sisälsivät kovakoodattuja tietoja, joten luontilausekkeet kirjoitettiin uudelleen. Kovakoodattujen tietojen poistaminen mahdollistaa näkymien käytön sellaisenaan, jos järjestelmä otetaan muissa toimitilarakennuksissa käyttöön.

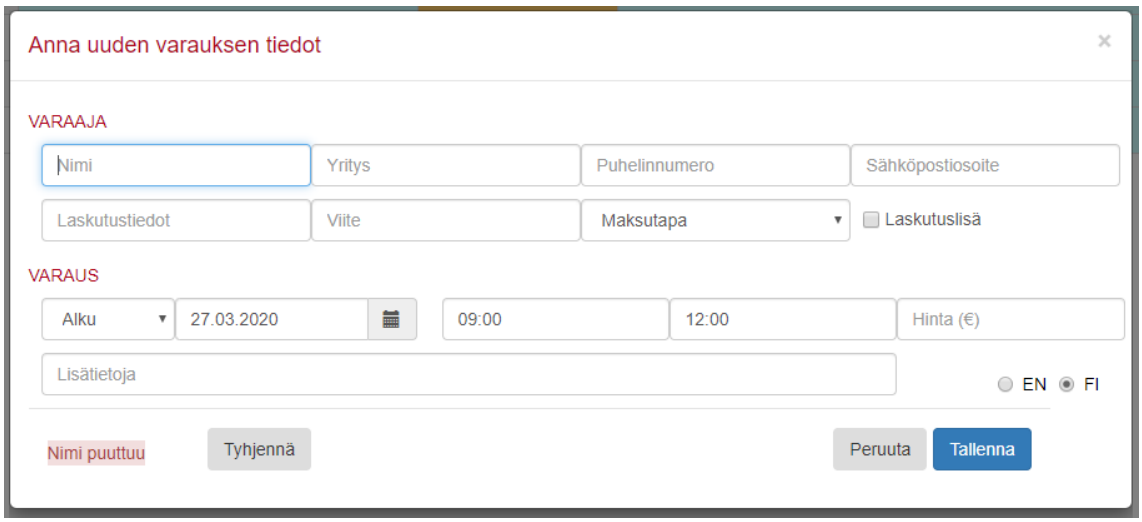
Frontend-koodin refaktoroinnit olivat pääsääntöisesti pieniä luettavuutta tai rakennetta parantavia muutoksia, kuten muuttujien ja metodien uudelleen nimeämistä kuvaavammin tai metodin pilkkomista pienempiin osiin. Yksi laajemmin refaktoroitu osa oli neuvotteluhuoneiden varauslomaketta koskeva koodi. Samaa lomaketta käytetään järjestelmän varaukset-näkymässä sekä varausten haku näkymässä ja jatkokehityksen yhteydessä huomattiin lomaketta koskevan koodin toistuvan molempien näkymien HTML-tiedostoissa. Näkymien toimintaa koskevat JavaScript-tiedostot sisälsivät myös keskenään monistettua koodia, sillä lomakkeen käsittelyä koskevat metodit löytyivät molempien näkymien hallintaa käsittelevistä tiedostoista.

Ennen varauslomakkeen päivittämistä sen HTML-koodi eristettiin omaan fragmenttiinsa, joka haetaan näkymään, jossa lomaketta halutaan käyttää. Näin tehtiin, jotta jatkossa lomaketta koskevat muutokset voidaan tehdä vain yhteen paikkaan ja että lomakkeen käyttäminen järjestelmän muissa osissa on tarpeen vaatiessa yksinkertaisempaa. Lomakkeita käsitelleet monistetut metodit yhdistettiin ja nostettiin hierarkiassa ylöspäin niin, että molemmissa näkymissä voidaan käyttää samoja metodeja lomakkeen hallintaan.

5 JATKOKEHITYKSEN TULOKSET

5.1 Päivitetyt ominaisuudet

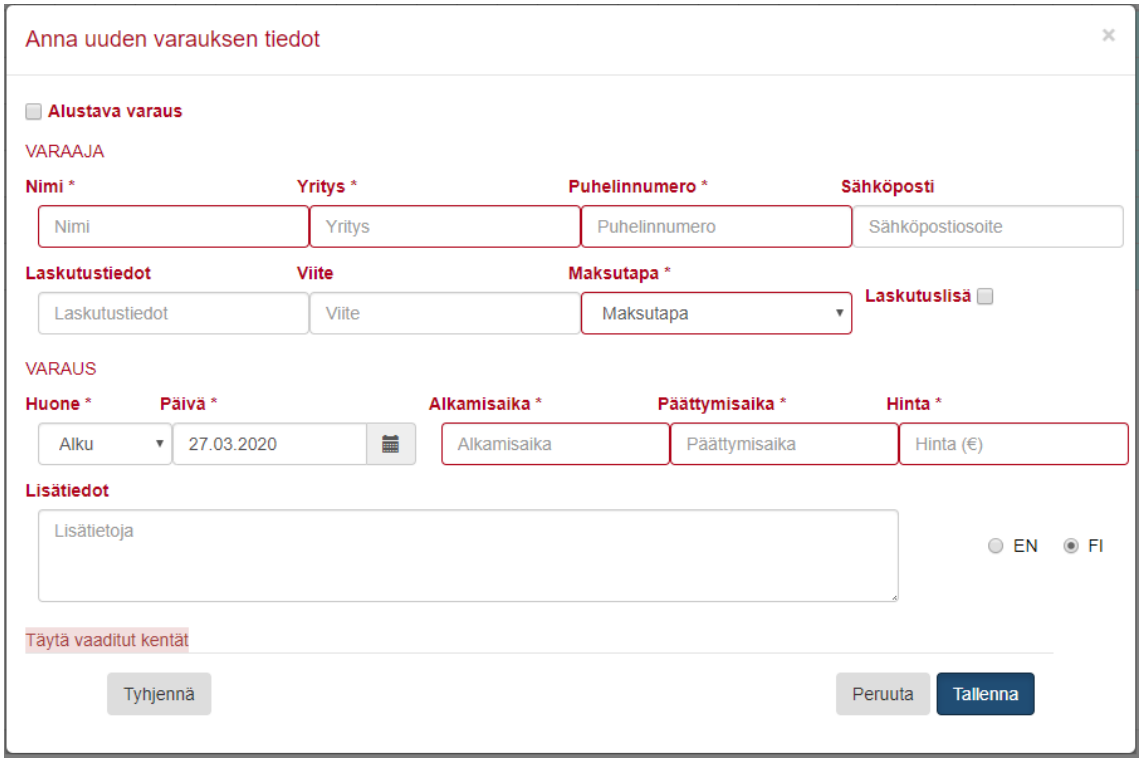
Neuvotteluhuoneiden varauslomakkeeseen toivottiin päivitystä, sillä virhetilanteissa käyttäjälle ei aina annettu selkeää palautetta. Lomake antoi virheilmoituksen ainoastaan varaajan nimen ja varauksen aloitus- ja lopetusajan puuttumisesta. Jos nämä tiedot oli syötetty lomakkeelle, mutta muita pakollisia tietoja puuttui, lomake suljettiin tallennuspainiketta painaessa ja varauskalenterin alla näytettiin virheilmoitus epäonnistuneesta tallennuksesta. Kun lomakkeen avasi uudelleen, ei käyttäjä kuitenkaan nähnyt mistään tallennuksen epäonnistumisen syytä, sillä virheilmoitukset puuttuivat eikä lomakkeessa ollut merkintöjä pakollisista kentistä. Kuvassa 8 on nähtävissä virheilmoitus, joka esitettiin, kun varaajan nimi puuttui. Vaikka muitakin pakollisia kenttiä on kuvan lomakkeella täyttämättä, ei niiden puuttumisesta ilmoitettu käyttäjälle.



KUVA 8. Näyttökuva Aulapalvelu-järjestelmän neuvotteluhuoneiden varauslomakkeesta, jossa virheilmoitus puuttuvasta nimestä

Päivitetyssä lomakkeessa vaaditut kentät on merkitty tähdellä. Kuten kuvasta 9 voidaan nähdä, tallennuspainiketta painaessa virheelliset kentät reunustetaan punaisella eikä lomake sulkeudu tallennuksen epäonnistuessa. Päivitetty virheil-

moitus lomakkeen alalaidassa huomauttaa puuttuvista tiedoista ja kertoo, jos varaus on päällekkäinen aiemman varauksen kanssa tai jos valittu varauksen alkamisaika on ennen sen valittua päättymisaikaa.

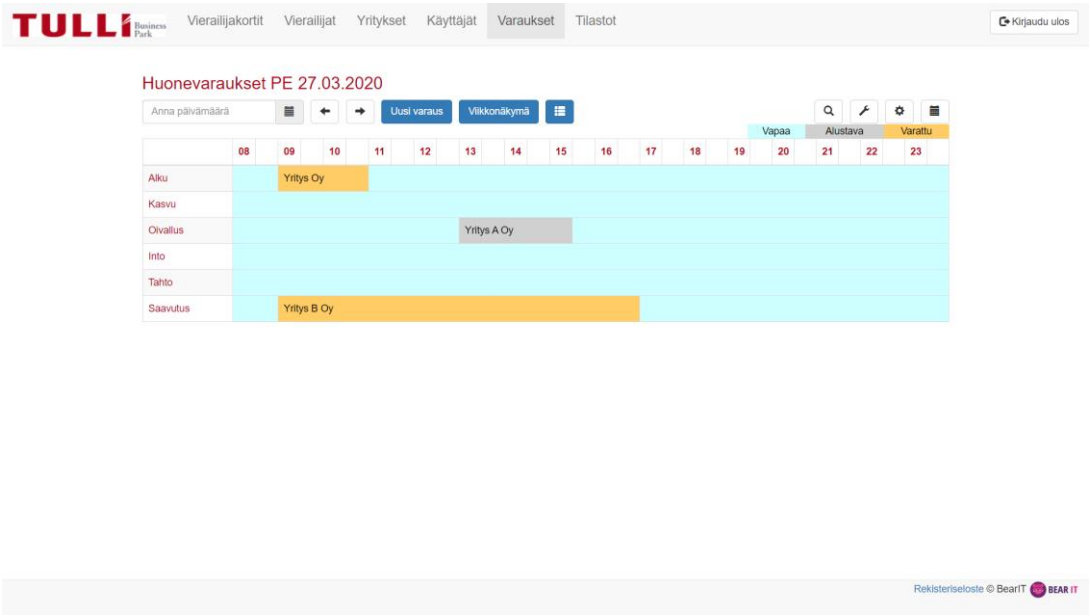


KUVA 9. Näyttökuva Aulapalvelu-järjestelmän päivitetystä neuvotteluhuoneiden varauslomakkeesta ja virheilmoituksista

Järjestelmään lisättiin mahdollisuus tehdä alustavia neuvotteluhuonevarauksia. Vastaanoton henkilökunta oli aiemmin merkinnyt varauksen alustavaksi kirjoittamalla varauksen lisätietoihin kyseessä olevan alustava varaus. Järjestelmä kuitenkin vaati tallennukseen laskutustiedot, joita ei alustavien varausten kohdalla välttämättä ollut tiedossa. Koska kenttiä ei voinut jättää tyhjäksi, niihin täytyi kirjoittaa jotain, mikä ei käytettävyyden kannalta ollut toimiva ratkaisu. Alustavat varaukset eivät ulkoisesti eronneet varauskalenterissa mitenkään vahvistetuista varauksista, joten henkilökunnan täytyi tarkistaa varauksen tila avaamalla sen tiedot.

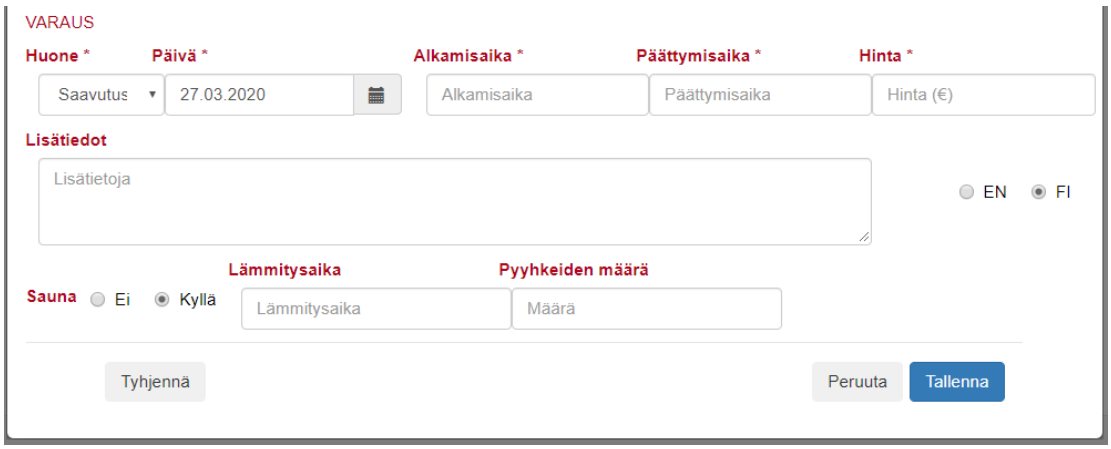
Nyt alustava varaus tehdään merkitsemällä kuvassa 9 näkyvän päivitetyn varauslomakkeen ylälaidassa oleva valintaruutu valituksi. Tällöin varauksen vaadittavien kenttien merkinnät päivittyvät, sillä alustavan varauksen tallennukseen ei

vaadita kaikkia tietoja. Alustavat varaukset näkyvät varauskalenterissa harmaalla pohjalla, joten ne erottuvat vahvistetutuista varauksista selkeästi. Kuvassa 10 näkyy Oivallus-neuvotteluhuoneeseen tehty alustava varaus.



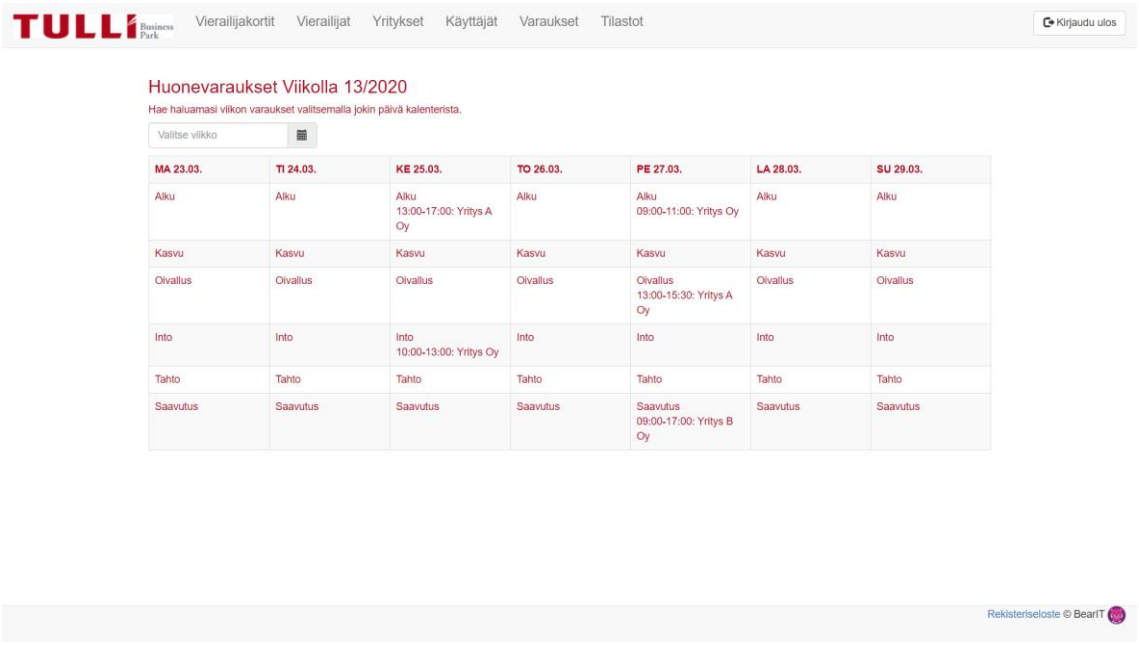
KUVA 10. Näyttökuva Aulapalvelu-järjestelmän varauskalenterista, jossa on alustava varaus

Saunallisen neuvotteluhuoneen varauslomakkeeseen haluttiin mahdollisuus merkitä, aiotaanko saunaa käyttää. Lisäksi toivottiin, että saunan lämmitysajan sekä pyyhkeiden määrän saisi kirjattua varaukseen, jos sauna on merkitty käytettäväksi. Päivitetyssä lomakkeessa on saunan käyttöä koskevat valintapainikkeet. Painikkeet näkyvät ainoastaan silloin, kun varausta ollaan tekemässä saunalliseen neuvotteluhuoneeseen. Jos sauna valitaan käytettäväksi, näytetään kentät, joihin saunan lämmitysaika ja pyyhkeiden määrä voidaan syöttää, kuten kuvassa 11 on esitetty.



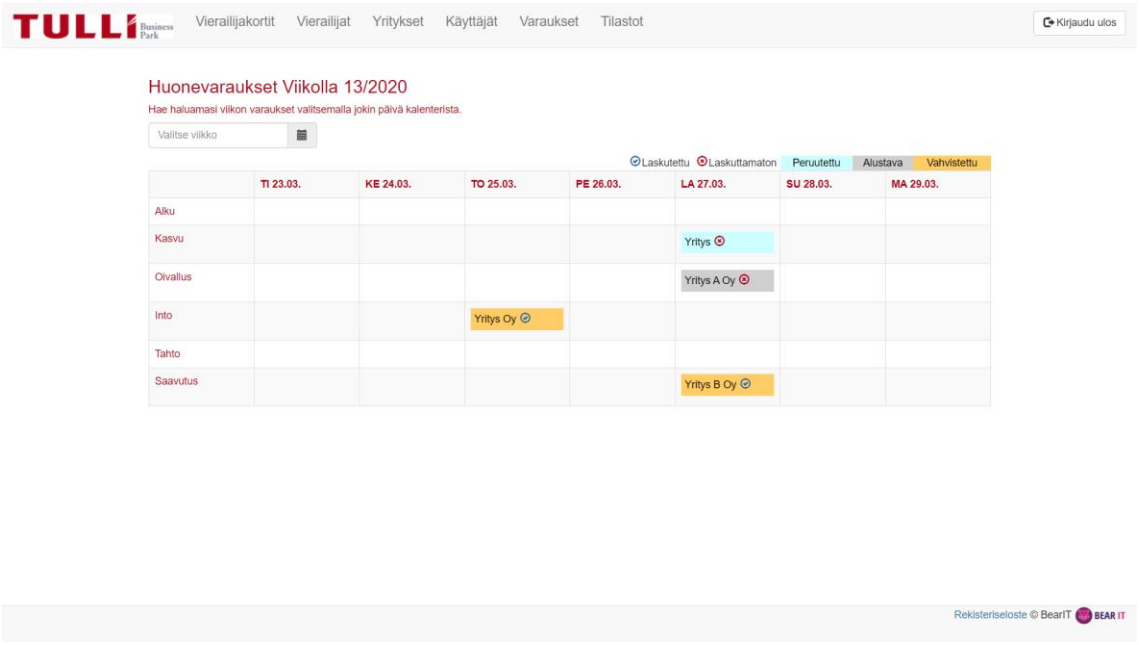
KUVA 11. Näyttökuva Aulapalvelu-järjestelmän saunallisen neuvotteluhuoneen varauslomakkeesta

Järjestelmän varaukset-näkymästä pääsi siirtymään kuvassa 12 olevaan viikkonäkymään, jossa pystyi tarkastelemaan kaikkien neuvotteluhuoneiden kokonaisen viikon varauksia kerralla. Varauksesta näytettiin yritys, jolle neuvotteluhuone oli varattu sekä varauksen ajankohta. Aulan henkilökunta ei kokenut näkymää tarpeelliseksi sellaisenaan, sillä nämä tiedot he pystyivät hakemaan myös muualta ja he kokivat neuvotteluhuonekohtaisen viikkonäkymän käytettävämmäksi. Henkilökunta toivoi, että näkymässä voisi tarkastella ja ylläpitää neuvotteluhuonevarausten laskutustilannetta. Laskutus on aiemmin hoidettu täysin järjestelmän ulkopuolella, eli varausten laskutustilannetta ei ole merkitty järjestelmään.

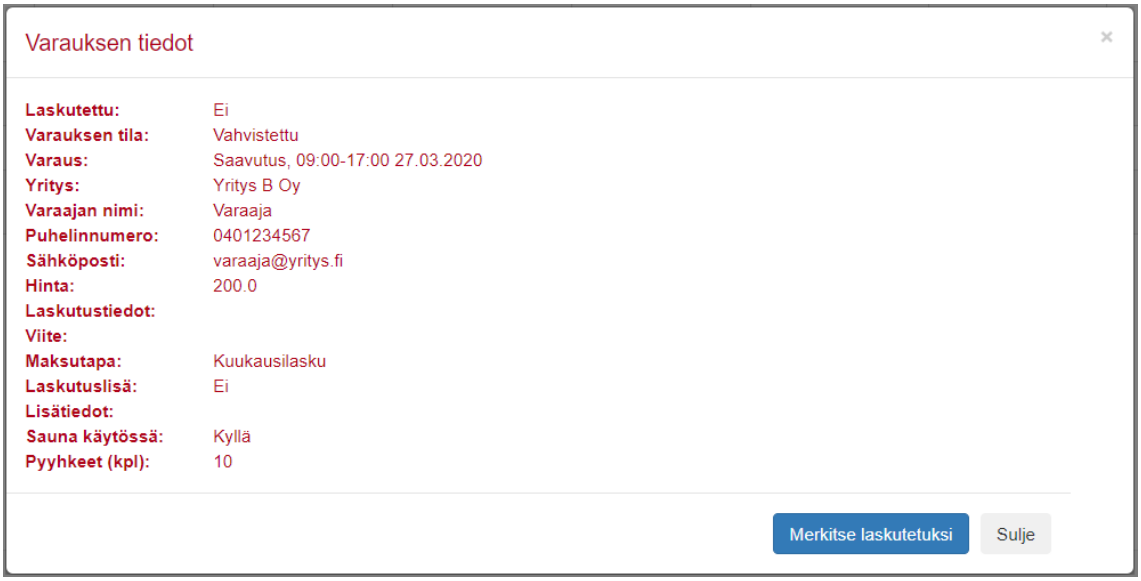


KUVA 12. Näyttökuva Aulapalvelu-järjestelmän viikon neuvotteluhuonevaraukset -näkymästä ennen päivitystä

Kuvassa 13 olevasta päivitetystä näkymästä tehtiin ulkoasultaan selkeämpi ja muita varauskalenterinäkymiä mukaileva. Varauksen taustaväristä näkee, onko kyseessä vahvistettu, alustava vai peruttu varaus ja varaajan nimen perässä olevasta ikonista, onko varaus laskutettu vai ei. Varausta klikkaamalla aukeaa kuvassa 14 näkyvä yhteenveto varauksen tiedoista, jonka yhteydessä varaus on mahdollista merkitä laskutetuksi tai poistaa laskutetuista, jos se on tarpeen. Varausten laskutusta ja laskujen lähettämistä ei hoideta järjestelmässä, mutta näkymän muutosten avulla on helpompi seurata laskutustilannetta ja koota laskutusten yhteenvedot. Laskutuksia hoitaessa näkymästä voi lisäksi helposti tarkastaa tarvittavat tiedot, kuten varauksen maksutavan, viitteen, hinnan tai mahdolliset lisämaksut.



KUVA 13. Näyttökuva Aulapalvelu-järjestelmän viikon neuvotteluhuonevaraukset -näkymästä päivityksen jälkeen



KUVA 14. Näyttökuva Aulapalvelu-järjestelmän neuvotteluhuoneen varauksen yhteenvedosta, jossa varaus merkitään laskutetuksi

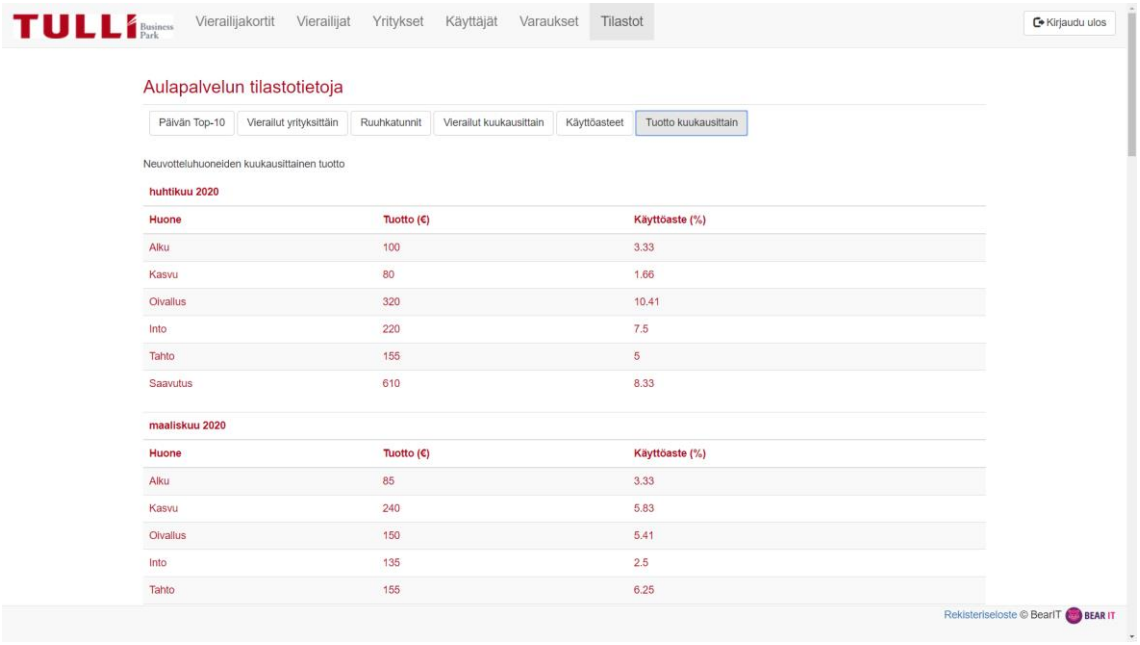
Neuvotteluhuoneiden kuukausittaista yhteenvetoa varten vastaanoton henkilökunta laskee huoneiden käyttöasteet. Käyttöasteprosentit olivat jo ennen jatkokehitysprojektia nähtävissä järjestelmän tilastot-näkymässä. Käyttöasteiden laskutapa oli kuitenkin virheellinen, jonka vuoksi näkymän antamia arvoja ei voitu

käyttää yhteenvedossa. Laskutapa päivitettiin vastaaman henkilökunnan käyttämää kaavaa.

Jatkokehitysprojektin yhteydessä tehtiin myös pienempiä päivityksiä ja korjauksia, sillä vastaanoton henkilökunta oli huomannut joitain ongelmia käyttäessään järjestelmää. Neuvotteluhuoneiden varauslomakkeen lisätietokenttään ei ollut asetettu rajoitusta merkkimäärästä, vaikka tietokannassa maksimi merkkimäärä oli asetettu. Tästä syystä varauksen tallennus epäonnistui, jos merkkimäärä ylittyi, vaikka varauslomake oli täytetty oikein. Lisäksi varauslomakkeen tyhjennyspainike ei tyhjentänyt kaikkia lomakkeen kenttiä. Varauskalenterin näyttämän päivän valintaan käytettävän pienoiskalenterin toiminnassa oli ongelmia, jonka vuoksi varausnäkymä saattoi jäädä jumiin. Nämä ongelmat korjattiin ensimmäisessä tuotantoon viedyssä versiossa.

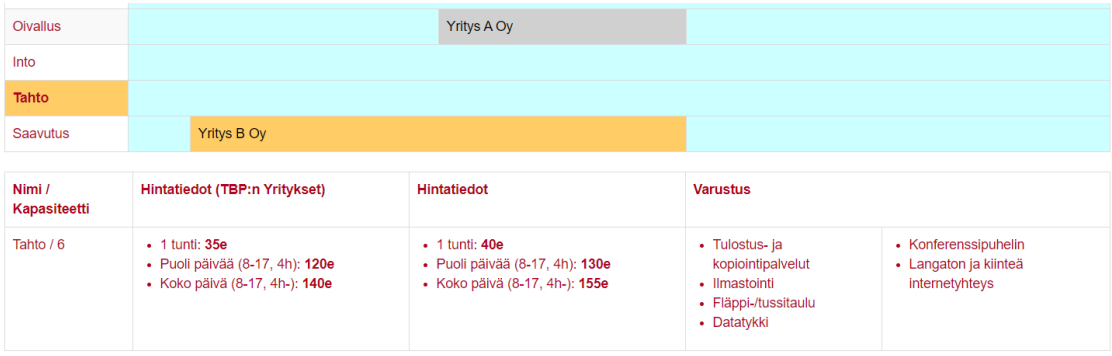
5.2 Uudet ominaisuudet

Neuvotteluhuoneiden tuottolaskelmaa varten vastaanoton henkilökunta kokoaa yhteenvedon koko kuukauden varauksista. Yhteenvetoon tarvitaan tieto jokaisen neuvotteluhuoneen tuotoista kyseisen kuukauden ajalta. Aiemmin vastaanoton henkilökunta käytti järjestelmän varauskalenteria apuna yhteenvedon koostamisessa, mutta koko kuukauden varausten tiedot käytiin läpi manuaalisesti ja koottiin taulukkoon, jonka pohjalta laskutus tapahtui. Tätä manuaalista ja aikaa vievää työvaihetta helpotettiin luomalla kuukausittaisille tuotoille kuvassa 15 esitetty oma osio järjestelmän tilastot-näkymään. Samasta näkymästä näkee kuukauden tuottojen lisäksi neuvotteluhuoneen käyttöasteen, sillä molempia tietoja tarvitaan yhteenvedon ja tuottolaskelman kokoamiseen.

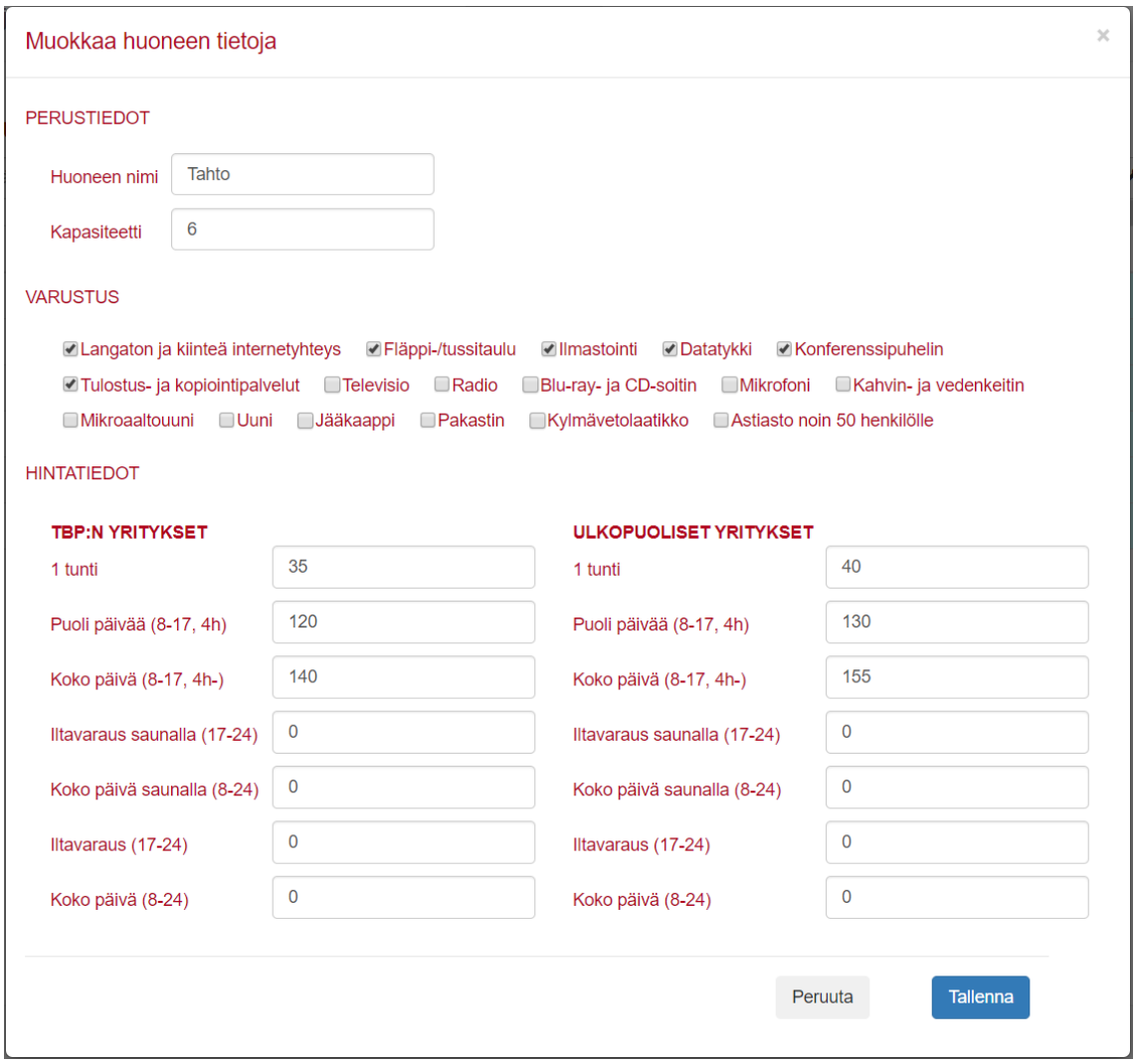


KUVA 15. Näyttökuva Aulapalvelu-järjestelmän tuotto kuukausittain -näkymästä

Neuvotteluhuoneilla on tietyt vuokrahinnat riippuen huoneesta, varauksen ajankohdasta ja siitä, sijaitsevatko varauksen tehneen yrityksen toimitilat Tulli Business Parkissa vai eivät. Hintatiedot haluttiin näkyville ja muokattaviksi järjestelmään, jotta hinnat voisi tarkistaa helposti varauksen teon yhteydessä. Hinnat lisättiin neuvotteluhuoneiden tietoihin, jotka näytetään varauskalenterin alapuolella, kun osoittimen vie neuvotteluhuoneen nimen päälle. Kuvassa 16 on näkyvissä Tahto-neuvotteluhuoneen hinta- ja varustetiedot. Hintoja voi muokata kuvan 17 asetuslomakkeella, joka aukeaa huoneen nimeä klikattaessa.



KUVA 16. Näyttökuva Aulapalvelu-järjestelmän neuvotteluhuonetiedoista

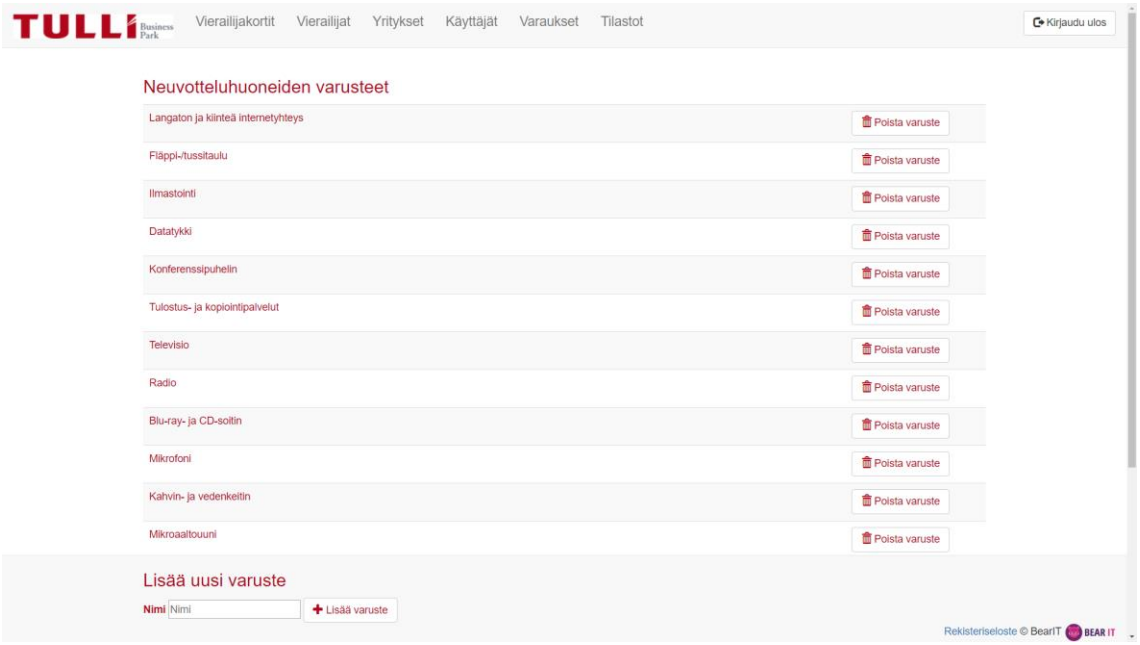


KUVA 17. Näyttökuva Aulapalvelu-järjestelmän neuvotteluhuoneen asetuksista

Hintatiedot ovat tällä hetkellä nähtävissä ja ylläpidettävissä järjestelmässä, mutta hintoja ei yhdistetty neuvotteluhuoneen varauslomakkeeseen. Lomakkeeseen voisi lisätä pudotusvalikon, josta käyttäjä voisi valita hinnan tai mahdollisesti päivittää järjestelmää niin, että varauksen hinta valittaisiin automaattisesti varatun huoneen, varaajan yrityksen ja varauksen ajankohdan perusteella.

Neuvotteluhuoneiden sisältämät varusteet olivat nähtävissä järjestelmässä jo ennen jatkokehitysprojektia, mutta varustetietoja ei voinut muokata kuin suoraan tietokantaa päivittämällä. Varustetiedot näkyvät hintatietojen vieressä, kun osoittimen vie neuvotteluhuoneen nimen päälle, kuten kuvassa 16. Järjestelmän tietokannassa oli valmiiksi taulu, joka sisälsi varustetiedot. Toisin kuin hintatiedot, varusteet eivät ole huonekohtaisia vaan eri neuvotteluhuoneissa voi olla käytössä

samoja varusteita. Tästä syystä järjestelmään lisättiin erillinen, kuvassa 18 esitetty asetusnäkymä, jossa varustetietoja voi lisätä ja poistaa. Asetusnäkymään siirrytään klikkaamalla ratasikonia varauskalenterin oikeasta yläreunasta. Huonekohtaiset varusteet valitaan kuvassa 17 näkyvällä neuvotteluhuoneen asetuslomakkeelta, jossa kaikki varustetiedot ovat listattuna.



KUVA 18. Näyttökuva Aulapalvelu-järjestelmän neuvotteluhuoneiden varusteiden asetuksista

5.3 Toimeksiantajan palaute

Toimeksiantaja on tyytyväinen jatkokehitysprojektin tuloksiin. Tehdyt muutokset parantavat toimeksiantajan mukaan sovelluksen käytettävyyttä huomattavasti. Erityisesti neuvotteluhuoneiden varauslomakkeiden uudistetut virheilmoitukset ja pakollisten kenttien selkeämmät merkinnät parantavat käytettävyyttä. Myös käyttöliittymään tehdyt muutokset tekevät toimeksiantajan mielestä käyttöliittymästä modernimman näköisen ja tuntuisen.

Toimeksiantaja on tyytyväinen jatkokehitysprojektin myötä vähentyneeseen manuaalisen työn tarpeeseen. Käsin tehtävä työ väheni, sillä nyt järjestelmästä saadaan raportointiin ja laskutukseen tarvittavat tiedot suoraan. Toimeksiantajan mukaan tehdyt muutokset myös mahdollistavat järjestelmän laajempaa käyttöönottoa muissa kohteissa. Neuvotteluhuoneiden varustetietojen muokkausmahdollisuus järjestelmästä käsin on yksi laajempaa käyttöönottoa mahdollistava uusi tekijä. Lisäksi toimeksiantaja uskoo järjestelmän koodiin tehdyn refaktoroimisen helpottavan järjestelmän ylläpitotöitä tulevaisuudessa.

6 POHDINTA

Rakenteeltaan selkeää ja helppolukuista ohjelmistokoodia on sujuvampaa ylläpitää ja jatkokehittää. Jotta ohjelmistokoodi säilyy helposti ylläpidettävänä, sitä tulee refaktoroida säännöllisesti niin jatkokehityksen kuin muunkin ylläpidon yhteydessä. Lyhyet metodit ja kuvaavat nimitykset ovat tekijöitä, jotka vaikuttavat koodin rakenteen selkeyteen ja ymmärrettävyyteen. Refaktoroiminen on tehokas työkalu, jota tulisi käyttää kaikessa ohjelmistokehityksessä.

Tämän opinnäytetyön tavoitteena oli jatkokehittää Aulapalvelu-järjestelmää sen loppukäyttäjien toiveiden pohjalta. Tuotoksena syntyi järjestelmän päivitetty versio, joka sisältää uusia ja päivitettyjä toiminnallisuuksia. Lisäksi järjestelmän koodia refaktoroitiin jatkokehityksen yhteydessä. Työn toimeksiantaja on tyytyväinen projektin tuloksiin ja loppukäyttäjiltä saatiin positiivista palautetta jo tuotantoon siirretyistä päivityksistä.

Järjestelmän kehitys jatkuu edelleen. Yhtenä jatkokehitystoiveena on näyttää neuvotteluhuoneiden varaustilanteet huoneiden ulkopuolella olevissa näytöissä, joista tällä hetkellä näkee vain neuvotteluhuoneen nimen. Ideana on, että näytöltä näkisi, onko neuvotteluhuone sillä hetkellä vapaa vai varattu ja seuraavan varauksen ajankohdan sekä huoneen varanneen yrityksen nimen. Toiveena olisi myös mahdollistaa useiden neuvotteluhuonevarausten tekeminen yhtä aikaa käyttäen samoja laskutustietoja. Tämä helpottaisi jatkuvien varausten tekemistä, kun ne voitaisiin luoda samalla kertaa yhdellä lomakkeella.

Aulapalvelu-järjestelmä on tällä hetkellä toteutettu Tulli Business Parkin tarpeiden mukaan. Jos se halutaan ottaa laajemmin käyttöön, tulisi sen ohjelmistokoodi ja tietokantarakenne käydä vielä kokonaisuudessaan läpi, sillä ne sisältävät jonkin verran kovakoodattuja tietoja. Yhtenä vaihtoehtona järjestelmän laajempaa käyttöönottoa ajatellen voisi olla sen toiminnallisuuksien pilkkominen muokattavissa oleviin komponentteihin. Kaikki toimitilarakennukset eivät välttämättä sisällä esimerkiksi neuvotteluhuoneita, jolloin niitä koskevat järjestelmän toiminnot eivät ole tarpeellisia. Jos järjestelmä olisi jaettu komponentteihin, asiakkaalle voi-

taisiin koota heidän tarvitsemansa paketti toiminnallisuuksista ja järjestelmän yksilöiminen eri asiakkaiden käyttöön olisi joustavampaa. Tämä vaatisi laajaa päivitystä ja hallinnointiominaisuuden lisäämistä järjestelmään.

Tässä työssä toteutettuun jatkokehitysprojektiin ei sisällytetty käyttöliittymän päivitystä, joten myös se olisi hyvä toteuttaa varsinkin ennen mahdollista laajempaa käyttöönottoa. Ulkoasun päivittämistä on suunniteltu, mutta on vielä epäselvää, tehdäänkö se järjestelmässä nykyisin käytetyillä tekniikoilla vai toteutetaanko mahdollisesti isompi päivitys, jossa frontend-teknologia vaihdetaan johonkin tuoreempaan.

LÄHTEET

Amazon Web Services. n.d.a. Amazon EC2. Luettu 24.4.2020. https://aws.amazon.com/ec2/

Amazon Web Services. n.d.b. Tutorial: Create a Web Server and an Amazon RDS Database. Luettu 24.4.2020. https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/TUT_WebAppWithRDS.html

Anjana, M. 2015. Mastering Spring Application Development. Birmingham: Packt Publishing.

Codecademy. n.d. What Is an IDE? Luettu 28.4.2020. https://www.codecademy.com/articles/what-is-an-ide

Fadatare, R. n.d. JPA Tutorial - Java Persistence API. Luettu 26.4.2020. https://www.javaguides.net/p/jpa-tutorial-java-persistence-api.html

Fowler, M. & Beck, K. 2018. Refactoring: improving the design of existing code. Boston: Addison-Wesley.

MDN web docs. 2020. Ajax. Päivitetty 17.2.2020. Luettu 24.4.2020. https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX

Raffai, Z. 2018. What is Spring boot? Julkaistu 27.7.2018. Luettu 22.4.2020. https://www.zoltanraffai.com/blog/what-is-spring-boot/

Sommerville, I. 2006. Software engineering. 8. painos. Harlow: Addison-Wesley.

Sonmez, J. 2017. A Software Developer's Guide to Maintaining Code. Julkaistu 13.2.2017. Luettu 11.2.2020. https://simpleprogrammer.com/maintaining-code/

Stackify. 2017. What are CRUD Operations: How CRUD Operations Work, Examples, Tutorials & More. Luettu 26.4.2020. https://stackify.com/what-are-crud-operations/

Tripathy, P. & Naik, K. 2014. Software Evolution and Maintenance: A Practitioner's Approach. New Jersey: John Wiley & Sons.

Tutorialspoint. n.d. AWS - Relational Database Service. Luettu 26.4.2020. https://www.tutorialspoint.com/amazon_web_services/amazon_web_services_relational_database_service.htm

Ugarte, A. 2020. Spring Boot CRUD Application with Thymeleaf. Päivitetty 21.3.2020. Luettu 26.4.2020. https://www.baeldung.com/spring-boot-crud-thymeleaf