Pysäköintisovelluksen kehittäminen Laurea-ammattikorkeakoululle

Pepita Kautto

2019 Laurea

Laurea-ammattikorkeakoulu

Pysäköintisovelluksen kehittäminen Laurea-ammattikorkeakoululle

Pepita Kautto
Tietojenkäsittely
Opinnäytetyö
Toukokuu, 2019

Laurea-ammattikorkeakoulu **Tiivistelmä**
Tietojenkäsittelyn koulutusohjelma
Tradenomi (AMK)

Pepita Kautto

Pysäköintisovelluksen kehittäminen Laurea-ammattikorkeakoululle

Vuosi 2019 Sivumäärä 43

Tässä opinnäytetyössä kehitettiin toimeksiantajalle, eli Laurea-ammattikorkeakoululle selaimessa käytettävä sovellus, jolla voidaan hallita korkeakoulun Tikkurilan kampuksen henkilökunnan pysäköintitiloja. Vaadittuja ominaisuuksia kehitetyltä sovellukselta olivat pysäköintiruutujen varaaminen sekä omien paikkojen vapauttaminen muiden sovelluksen käyttäjien käyttöön.

Kehittämistyössä perehdyttiin käytännön kautta selainpohjaisen sovelluksen rakentamiseen aina vaatimusmäärittelyn kokoamisesta julkaisuun asti. Tarkemmin avattuja aihealueita olivat yleisellä tasolla verkkosovelluksen tekeminen, sovelluksen visuaalisuuden kehittäminen sekä staattisen sivun kehittäminen dynaamiseksi hyödyntäen tallennustapana tietokantoja.

Kehittämistyön tuloksena syntyi toimiva, selaimessa käytettävä pysäköintisovellus, johon tunnistaudutaan Shibboleth –kertakirjautumisjärjestelmän avulla. Sovelluksen toiminnallisuuteen kuuluu pysäköintipaikkojen vapauttaminen ja varaaminen. Lisäksi sovelluksessa on hallinnointinäkymä, jonka kautta voidaan hallita paikkoja ja niille määriteltyjä käyttäjiä, lisätä uutisia sovelluksen käyttäjille, sekä lisätä ja poistaa sovelluksen ylläpitäjiä. Hallinnointinäkymän kautta voidaan myös varata pysäköintipaikkoja organisaation ulkopuolisille käyttäjille. Kaikille käyttäjille näkyvä karttanäkymä auttaa parkkihallin löytämisessä ja sovelluksesta löytyvä palautelomake auttaa loppukäyttäjien palautteiden keräämisessä. Sovelluksessa on myös kaksikielinen käyttöliittymä.

Valmis sovellus julkaistiin ammattikorkeakoulun henkilökunnan käyttöön helmikuussa 2019. Se sai heti lämpimän vastaanoton ja runsaasti positiivista palautetta.

Asiasanat: Verkkosovellus, kehitystyö, Hypertext Preprocessor (PHP), tietokannat

Laurea University of Applied Sciences
Degree Programme in Business Information Technology
Bachelor's thesis

Pepita Kautto

Abstract

Parking application for Laurea University of Applied Sciences

Year 2019 Pages 43

The aim of this Bachelor's thesis was to develop a browser-based web application for the employees of Laurea University of Applied Sciences. The main goal for the application was to optimize the usage of Laurea's parking garage located on Tikkurila campus. Required functions for the application were to release unused parking spaces and make bookings for the available ones.

The development work took a practical approach to developing a browser-based web application from requirement analysis to publishing the finalized product. More detailed topics were developing a web application, building a visual user interface, developing dynamic actions and storing data to databases.

The result of this thesis was a fully functioning parking application which uses Shibboleth Single-Sing-On service to authenticate the user. Main functions include booking and releasing parking spaces and the application includes admin view to manage the parking lot and its users. For admin users there is a possibility to write and manage news, add other admins and book spaces for users outside the organisation. The application has a map view to help locate the parking garage, and a feedback form to collect feedback from the users. It also has a bilingual user interface.

Finalized application was released to Laurea's employees on February 2019. It got a warm welcome and plenty of positive feedback.

Keywords: web application, development, Hypertext Preprocessor (PHP), databases

Sisällys

1 Johdanto

Laurea-ammattikorkeakoulun Tikkurilan toimipisteen kellarikerroksessa sijaitsee parkkihalli, josta korkeakoulun henkilökunta voi ostaa itselleen pysäköintiruudun käyttöoikeuden. Monet henkilökunnan jäsenet, jotka omistavat ruutuja kuitenkin työskentelevät myös toisilla kampuksilla tai tekevät etätöitä, joten paikkoja on usein vapaana. Näille vapaille paikoille ei kuitenkaan voi muut pysäköidä, sillä ne ovat jonkun toisen henkilön omia. Korkeakoulun pysäköintitiloissa on muutamia vieraspaikkoja, jotka ovat aktiivisessa käytössä.

Korkeakoululla on käytössään selainpohjainen sovellus, joka mahdollistaa kampuksen parkkihallissa olevien pysäköintipaikkojen hallinnan. Sovellus sisältää kuitenkin paljon ongelmia, ja on vaikeasti ylläpidettävä. Tämän kehitystyön tavoitteena on tuottaa versio tai vaihtoehto nykyiselle sovellukselle, joka ratkaisisi nämä ongelmat.

Toimivasta pysäköinninhallintasovelluksesta hyötyvät organisaation, eli Laurea-ammattikorkeakoulun sisällä useat tahot. Sovellusta ylläpitää tietohallinto, joka toivoo mahdollisimman automatisoitua ja helppoa ylläpitoa. Asiakasrajapinnassa käyttäjien ja sovelluksen välissä ovat korkeakouluisännät, jotka vastaavat tilojen, mukaan lukien parkkihallin, ylläpidosta. Isännät ovat saaneet jonkin verran huomautuksia vanhan sovelluksen toiminnallisuuden virheistä sekä ongelmista. Isännille toimiva sovellus siis mahdollistaisi helpotusta parkkihallin käytön hallinnointiin. Tilojen hallintaan liittyen, sovelluksesta olisi tarkoitus saada irti myös käyttäjädataa. Tarvittaessa olisi mahdollista seurata sitä, kuinka paljon käyttäjät vapauttavat ja varaavat paikkoja, tai mihin aikaan päivästä tai viikosta käyttö tapahtuu. Saatuja tilastoja seuraamalla voidaan nähdä esimerkiksi se, kuinka hyvin käytössä oleva laskutusmalli sopii kohderyhmälle.

Merkittävimmin sovelluksesta kuitenkin hyötyvät ne käyttäjät, jotka haluavat varata paikkoja. Siirtyminen päivän aikana kampukselta toiselle helpottuu, kun tietää, että auton saa pysäköityä helposti ja nopeasti myös seuraavalla kampuksella. Ylläpitävälle taholle on myös merkittävän hyödyllistä nähdä sovelluksen kautta kerättävää statistiikkaa, jotta voidaan helposti havainnoida sovelluksen käyttöastetta ja sitä myötä tarpeellisuutta ja kannattavuutta.

2 Työn lähtökohdat

Lähtökohtana kehitystyölle oli tarve jatkokehittää olemassa olevaa pysäköintisovellusta. Korkeakoululla oli kehitetty aikaisemman harjoittelijan toimesta pysäköintisovellus, jonka ideasta toimeksiantaja piti, mutta toteutuksen kanssa kaivattiin apua ja tekijää. Tämän työn tekijälle tarjottiin korkeakoululta määräaikainen työpaikka esimerkiksi opintoihin kuuluvan harjoittelujakson suorittamiseen, sekä mahdollinen aihe opinnäytetyöhön. Projekti alkoi marraskuussa 2018, ja jatkui kevättalvelle 2019. Valmis sovellus saatiin julkaistua helmikuussa 2019.

2.1 Kehittämiskohteen kuvaus ja kehittämistavoitteet

Kehittämiskohteena tässä opinnäytteessä on Laurea-ammattikorkeakoulun käyttöön tehty pysäköintisovellus. Sovellusta ylläpidetään Laurean tietohallinnon kautta ja sitä voivat käyttää kaikki ammattikorkeakoulun henkilökuntaan kuuluvat, kampuksesta riippumatta. Sovelluksen idea pähkinänkuoressa on tarjota käyttäjilleen mahdollisuus varata vapaaksi merkittyjä parkkipaikkoja ammattikorkeakoulun Tikkurilan kampuksen autohallista.

Sovelluksen ensimmäinen, kehitystä kaivannut versio sisälsi ongelmia; käyttäjät eivät tunnistautuneet sovellukseen, joten parkkipaikkojen tietoihin ei voitu lisätä tietoa siitä, kenelle se kuului. Tämä mahdollisti sen, että kuka vain pystyi vapauttamaan kenen tahansa parkkipaikan muiden varattavaksi. Sovellus ei myöskään ollut yhteensopiva Microsoft Edge eikä Internet Explorer –selainten kanssa, jotka ovat korkeakoulun henkilökunnan työkoneiden oletusselaimia. Edellä mainituilla selaimilla käytettäessä rajoitus, jolla käyttäjät voivat varata vain yhden paikan ei toiminut, vaan yksi käyttäjä pystyi halutessaan varaamaan kaikki parkkihallin pysäköintiruudut.

Vanhassa versiossa kirjauduttiin sisään Laurean tunnuksilla, mutta kirjautunutta käyttäjää ei tunnistettu millään tavalla. Myöskään uloskirjautumismahdollisuutta ei ollut. Jos sovellukseen kirjautui esimerkiksi opiskelijatunnuksilla, ei päässyt käyttämään sovellusta, sillä se oli rajattu ainoastaan henkilökunnalle, mutta ei myöskään päässyt kirjautumaan ulos. Edellä mainitussa tilanteessa jos olisi halunnut kirjautua uudelleen, esimerkiksi eri tunnuksilla, piti odottaa session vanhenemista tai tyhjentää selaimen välimuisti. Sovellukseen kaivattiin myös helpompaa ylläpitoa, lisää toiminnallisuuksia sekä kaksikielistä käyttöliittymää, joka tukisi suomen kielen lisäksi myös englantia.

2.2 Toimeksiantajan esittely

Laurea on pääkaupunkiseudulla ja sen lähiympäristössä toimiva ammattikorkeakoulu. Koululla on opetusta kuudella eri kampuksella; Tikkurilassa, Leppävaarassa, Otaniemessä, Lohjalla, Porvoossa sekä Hyvinkäällä. Laurean verkkosivuilla kerrotaan, että Laurea tarjoaa ammattikorkeakoulututkintoja, sekä ylempiä ammattikorkeakoulututkintoja liiketaloudesta, sosiaali- ja terveysalalta sekä ravitsemus- ja talousalalta.

Korkeakoulun verkkosivujen mukaan Laurea aloitti toimintansa vuonna 1992 nimellä Vantaan ammattikorkeakoulu, ja laajentuessaan vuonna 1997 koulu uudelleennimettiin Espoon-Vantaan ammattikorkeakouluksi. Nykyinen nimi valittiin valtionneuvoston vakinaistettua ammattikorkeakoulun toiminnan vuonna 2001. Organisaationa korkeakoululla on 7800 opiskelijan lisäksi viidensadan työntekijän henkilöstö. Valmistuneita opiskelijoita, eli Laurean alumneja on jo yli 24 000.

Laurean IT-hankinnoista, lähituesta ja –verkosta sekä IT-sopimuksista vastaa korkeakoulun tietohallinto. Käytännössä henkilökunta ja opiskelijat ovat yhteydessä tietohallintoon esimerkiksi, kun kohtaavat ongelmia päivittäisissä tietoteknisissä pulmissa. Tietohallinnon kautta tehdään myös kaikki korkeakoulun laitetilaukset. Tietohallinnon ylläpitämä lähituki on loppukäyttäjien tukipalvelua, jossa työskentelee pääasiassa Laurean tietojenkäsittelyn opiskelijoita harjoittelujaksoillaan pääasiassa Tikkurilan ja Leppävaaran kampuksilla. Tietohallinto voi rekrytoida myös projektityöntekijöitä ja harjoittelijoita muihin tehtäviin, kuten sovelluskehitysprojekteihin.

2.3 Tutkimuskysymykset ja aihealueen rajaus

Kehitystyön tulisi vastata kysymykseen: miten sovelluksesta saataisiin mahdollisimman käyttäjäystävällinen? Miten siitä voitaisiin saada mahdollisimman paljon hyötyä irti? Miten sovelluksesta saataisiin mahdollisimman helppo ylläpitää ja miten se voisi auttaa käyttäjiä; niin paikkaansa vapauttavia, kuin halukkaita varaajia? Miten verkkosovellus yleisesti kannattaa kehittää?

Tässä kehitystyössä aihealue on rajattu sovellustuotteen kehitysprojektiin. Rajauksen ulkopuolelle on jätetty kaikki sovelluksen toimintamalliin ja palvelumuotoiluun liittyvä problematiikka ja työssä keskitytään enemmän tekniseen toteutukseen.

Teoriaosuudessa perehdytään verkkosovelluksen kehittämisen teoriaan. Tarkemmin avattuja aihealueita ovat käyttöliittymän ja visuaalisuuden suunnittelu ja toteuttaminen, sekä toiminnallisuuden rakentaminen. Toteutustapoina opinnäytteessä tutustutaan käyttöliittymän rakentamisessa HTML:ään ja CSS:ään, toiminnallisuuden kehittämisessä käytettyyn ohjelmointikieleen nimeltä PHP, sekä relaatiotietokantoihin tapana tallentaa tietoa.

Opinnäytetyössä kuvataan pysäköintisovelluksen kehitystyötä aina vaatimusmäärittelystä julkaisuun asti. Keskeisiä aihealueita ovat vaatimusmäärittely, toiminnallisuuden rakentaminen, sovellukseen tunnistautuminen ja ulkoasun toteutus.

2.4 Keskeiset käsitteet

PHP	Suosittu, monikäyttöinen palvelinpohjainen ohjelmointikieli, jonka lyhenne tulee sanoista Hypertext Preprocessor.
SQL	Structured Query Language, jolla voidaan ottaa yhteys ja muokata tietokantoja.
HTML	Kieli, jolla useimmat verkkosivut ovat tehty. Sitä käytetään luomaan sivuja ja niille staattista sisältöä, kuten tekstiä ja linkkejä, sekä liittämään niihin kuvia.

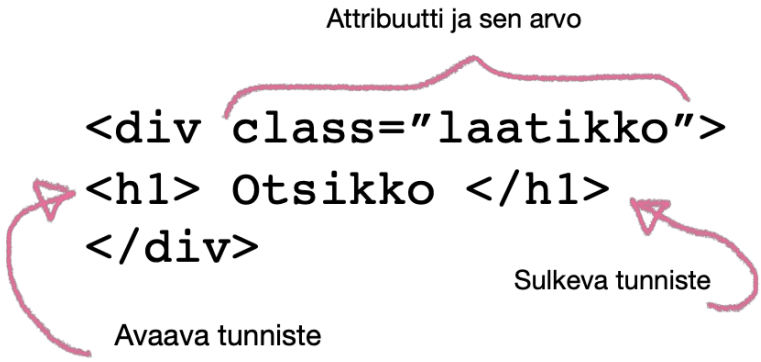
CSS	Kieli, jolla määritellään miten HTML–elementit esitetään. Kielen lyhenne tulee sanoista Cascading Style Sheets.
JavaScript	Ohjelmointikieli, jota käytetään verkko-ohjelmoinnissa. Sen avulla voidaan manipuloida HTML– ja CSS –koodia, laskea, luoda ehtolauseita ja funktiota. Javascriptin avulla voidaan luoda sivuille toiminnallista ja dynaamista sisältöä.
jQuery	JavaScript kirjasto, jolla voidaan yksinkertaistaa JavaSciptin syntaksia.
Shibboleth	Avoimen lähdekoodin kertakirjautumispalvelu, joka mahdollistaa käyttäjien tunnistautumisen. Koostuu kahdesta osasta; Service Providerista ja Identity Providerista.
Service Provider	Shibbolethin osa, joka on sovelluksen kanssa samalla palvelimella sijaitseva ohjelmisto. Service Providerin tehtävä on tarjota ja suojata siihen liitettyä palvelua.
Identity Provider	Shibbolethin osa, joka tarjoaa käyttäjien tiedot tunnistautumista varten ja lähettää ne eteenpäin.
Single Sign-on, SSO	Kertakirjautuminen, jolla tarkoitetaan sitä, että kotiorganisaation kirjautumista vaativiin palveluihin kirjaudutaan vain yhden kerran.
Cookie	Suomeksi tunnetaan myös evästeenä. Kooltaan pieni tekstitiedosto, jonka selain tallentaa verkkosivulta. Evästeisiin voidaan tallentaa esimerkiksi käyttäjän tietoja, sivuston asetuksia tai kieli- ja sijaintitietoja.
Grafana	Ohjelma, joka on tarkoitettu pääasiassa aikajanamuotoisen tiedon esittämiseen, mutta tukee myös muita muotoja. Käytännössä siihen voi yhdistää tietokantoja ja palveluita, ja esittää niihin tallennettua tietoa inforuutumaisilla näkymillä.
Font Awesome	Avoimen lähdekoodin verkkopalvelu, joka tarjoaa ikoneita ja kuvakkeita vapaaseen käyttöön fontti- tai vektorigrafiikkana.
PUTTY	Alun perin Simon Tathamin kehittämä, nykyisin avoimen lähdekoodin ohjelma, jolla voidaan ottaa etäyhteyksiä verkon yli käyttäen SSH, Telnet ja Rlogin protokollia konsolikäyttöliittymää käyttäen. (Tatham, S. 2019).

WINSCP	Ilmainen avoimen lähdekoodin client Windowsille, joka tukee SFTP, FTP, WebDAV, S3 ja SCP yhteyksiä. Ohjelman tarkoitus on helpottaa tiedonsiirtoa palvelimien välillä tarjoamalla graafisen käyttöliittymän. (Martin 2018).
ATOM	Avoimen lähdekoodin tekstieditori, joka on suunniteltu ohjelmointiin sekä koodin kirjoittamiseen. Atom on hyvin mukautuva, ja jokainen käyttäjä voi konfiguroida sen vastaamaan juuri omiin tarpeisiinsa esimerkiksi lataamalla lisäosia ja muokkaamalla asetuksia.

3 Verkkosovelluksen kehittämisprosessi

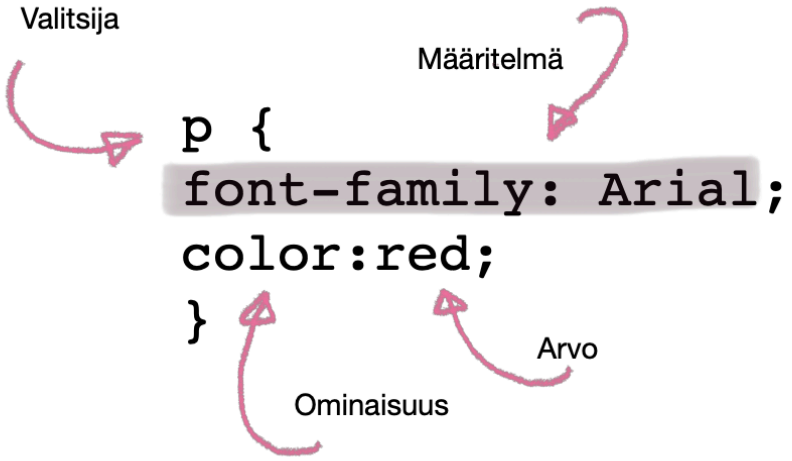
Verkkosivujen ja -sovelluksien kehittäminen, aloitetaan miettimällä sitä, mitä sivun tai sovelluksen on tarkoitus tehdä. Jos sivulla halutaan vain esitellä staattisia elementtejä, kuten kuvia ja tekstejä, riittää toteutustavaksi HTML ja visuaalisiin muutoksiin CSS. Edellä mainituilla kielillä pystyy toteuttamaan staattisen sivun tai sivuston, jota palvelin näyttää käyttäjälle verkkoselaimen kautta. (Beighley & Morrison 2009, 2).

HTML määrittelee sivun rakenteen ja sen avulla merkitään esimerkiksi mikä sivun tekstistä on otsikkoa, mikä toimii linkkinä ja mikä on leipätekstiä. Eri rakenteita varten käytetään eri tagejä, eli tunnisteita, jotka sijoittuvat hakasulkumerkkien väliin. Ilman attribuutteja avaava ja sulkeva tunniste ovat lähes samanlaisia, ainoastaan kauttaviiva sulkevassa tunnisteessa erottavat ne toisistaan. HTML-tunnisteille voidaan määritellä attribuutteja jotka sijoittuvat avaavaan tunnisteeseen. Attribuutit tarjoavat lisätietoja tunnisteen sisältämästä elementistä, kuten esimerkiksi luokan nimen. (Duckett 2011, 20-26). HTML-tunnisteiden rakenne on kuvattu kuviossa 1.



Kuvio 1: HTML-tunnisteen rakenne

CSS määrittelee sen, miten HTML:n avulla kirjoitetut elementit esitetään. Sen avulla voidaan luoda sääntöjä, joilla kerrotaan miltä eri elementit näyttävät. CSS voi määrittää esimerkiksi, että sivun taustaväri on vaaleanpunainen, kaikki otsikot ovat sinisiä ja että leipätekstin fontti on Arial. CSS:ää voi kirjoittaa HTML-tiedostoon tyylitunnisteiden sisälle, mutta suositeltavin tapa on kirjoittaa se kokonaan omaan tiedostoonsa, jota kutsutaan HTML-tiedostosta. Rakenteeltaan CSS sisältää valitsijan (*selector*) ja sille määritelmän (*declaration*). Määritelmiä voi olla useita ja ne sijoitetaan aaltosulkujen sisäpuolelle. Määritelmä koostuu kahdesta osasta, ominaisuudesta (*property*) ja sen arvosta (*value*). Ominaisuus kuvaa määriteltävän elementin ominaisuutta, joka voi olla esimerkiksi leveys, korkeus, väri, tai koko. Arvo määrää taas sen, miksi ominaisuus halutaan määritellä. Jos ominaisuudella määritellään väriä, tulee arvoksi täten jokin väri, esimerkiksi punainen. (Duckett 2011, 227-232). CSS-säännön rakennetta on avattu kuviossa 2.



Kuvio 2: CSS säännön rakenne

Yhdistämällä HTML:ää ja CSS:ää voidaan luoda yksinkertaisia, staattisia verkkosivuja, jotka esittävät sisältöä käyttäjilleen. Toteutustapoina sivuille voidaan käyttää esimerkiksi yhden sivun ulkoasua, jossa kaikki sisältö on yhdellä sivulla, tai luoda useita tiedostoja, jotka linkitetään toisiinsa esimerkiksi valikon kautta.

3.1 Kehitysmenetelmänä Scrum

Scrum-mallissa ohjelmistotuotteen toteutus jaetaan sprintteihin, joita työstetään yksi kerrallaan. Sprintit ovat usein pituudeltaan viikosta kuukauteen. Sprintteihin kuuluu myös tuotettujen ominaisuuksien testaus. Kun kaikki sprintit on tehty, ohjelmistotuotteen pitäisi olla valmis, toimiva ja siirrettävissä tuotantoon ja ylläpitovaiheeseen. Tärkeään rooliin menetelmässä nousee tiimin sisäinen kommunikointi, ja se sopiikin parhaiten tiivistä yhteistyötä tekevälle tiimille, jolla on mahdollisuus aktiiviseen yhteydenpitoon. Yksinkertaisuutensa ansiosta

sitä pystytään hyvin soveltamaan projekti- tai tiimikohtaisesti ja se on kerännyt suosiota yritysmaailmassa myös ohjelmistoalan ulkopuolella. (Kasurinen 2013).

Scrumissa työtiimit ovat itseohjautuvia, eikä tiimin jäsenille anneta erillisiä nimikkeitä scrummasterin ja tuoteomistajan lisäksi. Scrummasterin rooli on huolehtia siitä, että työtiimi työskentelee tehokkaasi ja menetelmän mukaisesti. Hän myös valmentaa itseohjautuvuuteen, fasilitoi sekä auttaa tekijöitä ymmärtämään miksi jotain asiaa tehdään. Tuoteomistaja puolestaan on vastuussa tuotteen kehitysjonosta eli back logista, sen selkeydestä sekä siitä, että kehitystiimi ymmärtää ja tietää, mitä seuraavaksi tulisi tehdä. (Schwaber & Sutherland 2017, 5-0).

Scrum-projektien taustalla on aina tuotteen kehitysjono, joka on lista kaikista ominaisuuksista, toiminnoista, vaatimuksista sekä parannuksista ja korjauksista, joita valmis tuote ja sen kehittäminen sisältävät. Kehitysjono muuttuu ja mukautuu projektin edetessä. Jonoon päivitetään kohtien etenemistä kirjaamalla, onko kohta tulossa seuraavaksi kehityksen alle, tällä hetkellä työn alla vai valmis. Tuotteen kehitysjonosta kootaan sprinttikohtaisesti pienempi kehitysjono, jonka kaikki kohdat käsitellään yhden sprintin aikana. (Schwaber & Sutherland 2017, 15).

3.2 Verkkosovelluksen ulkoasu

Verkkosivujen ulkonäön suunnittelu saavutti nykyisen muotonsa vuoden 2000 tienoilla, kun aikaisemmin suositun, mutta raskaan Flashin syrjäytti CSS, joka on lyhenne sanoista Cascading Style Sheets. CSS määrittelee, miten kirjoitettu HTML-koodi esitetään, ja sitä myötä mahdollistaa verkkosivujen ulkoasun määrittelemisen koskematta sivun muuhun sisältöön. Suunnittelijat käyttivät Flashia esimerkiksi liittämään sivuille animaatioita ja liikkuvaa kuvaa, kolmiulotteisilta vaikuttavia elementtejä sekä kustomoituja kirjaisintyylejä. Myöhemmin, viimeistään kun Apple päätti olla tukematta Flashia käyttöliittymissään vuonna 2010, sen suosio rupesi laskemaan merkittäväsi, samalla kun suunnittelijat rupesivat käyttämään CSS:ää samojen asioiden teknisenä toteutustapana. (Hong 2018, 11-13).

Kun verkon selaaminen mobiililaitteilla on lisääntynyt räjähdysmäisesti, täytyy myös verkkosuunnittelun vastata siihen. Mobiililaitteella selattaessa tietokoneella käytettäväksi tarkoitetut verkkosivut eivät ole kovin käyttäjäystävällisiä, vaan niiltä voi olla haastavaa löytää tietoa. Tästä syystä mobiiliversiot yleistyivät verkkosivujen suunnittelussa ja nykypäivänä niitä löytyy lähes jokaiselta verkkosivulta.

Ensimmäisiä ratkaisuja ongelmaan, oli ylläpitää kahta eri verkkosivua; toinen perinteiselle tietokonekäytölle ja toinen mobiilikäyttäjille. Kuten arvata saattaa, kahden erillisen verkkosivun ylläpitäminen sisältää tuplasti työtä, ja kun CSS3 -version mukana julkaistiin Media Queryt, eli ehdolliset CSS-määreet, suunnittelijat pystyivät rakentamaan samasta sivusta eri

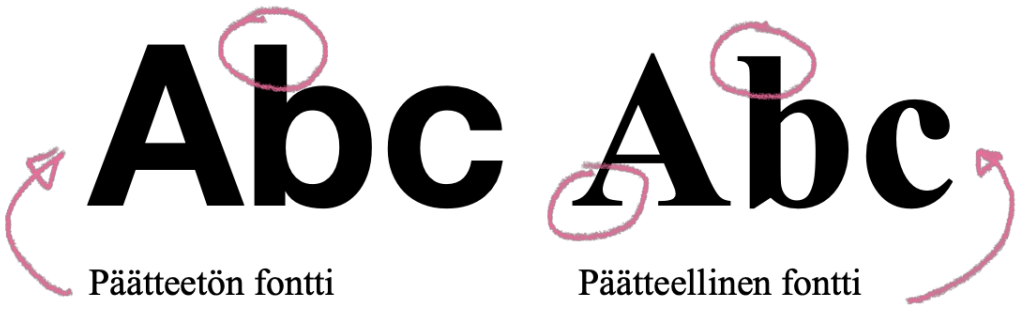
näkymiä riippuen laitteesta, jolla niitä käytetään. Ehdolliset CSS-määreiden avulla voidaan tarkastella esimerkiksi ruudun resoluutiota ja kokoa, ja määritellä tyylejä sen mukaan. (Hong 2018, 16-18)

Ehdollisten CSS-määreiden avulla voidaan siis määritellä oma käyttöliittymänsä mobiilinäkymälle, eli näytettäväksi kun sivun leveys vastaa mobiililaitteen näytön leveyttä. Samalla tavalla voidaan määritellä esimerkiksi tablettitietokoneelle tai muille pienemmille näytöille omansa.

3.2.1 Typografia ja ikonit

Verkkosivujen visuaalisuutta suunnitellessa voidaan käyttää ikoneita. Ikonien tarkoitus on helpottaa sivun sisällön hahmottamista, ja tekemään johtopäätöksiä. Ne ovat myös visuaalisesti sivun sisältöä rikastuttava elementti ja parhaimmillaan auttavat käyttäjää löytämään juuri sitä tietoa, mitä he etsivät. Ikoneilla voidaan korvata myös käyttöliittymän sanoja, esimerkiksi eteenpäin vievä linkki voi sisältää nuolen eteenpäin ja painike, jolla poistutaan näkymästä, sisältää usein ruksin. (Hong 2018, 36-37).

Tärkeä osa verkkosivun ulkonäköä on teksti. Kootessa typografista kokonaisuutta, tekstien ulkonäköä ja ominaisuuksia, on otettava huomioon muutama seikka. Typografia kannattaa valita niin, että se tukee sivujen brändiä. Vaikutteita voi ottaa esimerkiksi logosta, tai muuten brändin värivalikoimasta. Niinkin pieni asia, kuin fontin valinta, voi luoda käyttäjälle yllättävän paljon mielikuvia yrityksestä. Esimerkiksi päätteelliset fontit, jotka tunnetaan myös nimellä sherif –fontit koetaan usein klassisina, romanttisina, asiallisina sekä elegantteina. Päätteettömät, eli sans-serif –fontit puolestaan koetaan moderneina, minimalistisina ja ystävällisinä. Päätteellisten ja päätteettömien fonttien lisäksi on myös olemassa esimerkiksi käsinkirjoitettua jäljittelevii fontteja, joita voidaan käyttää tehosteena ja luomassa mielikuvia esimerkiksi otsikoissa. Tärkeä sääntö typografiaa suunniteltaessa on kuitenkin pysyä maltillisena; liian montaa fonttia yhdistämällä tuskin saa aikaan muuta kuin sekavaa ulkoasua. Yhdestä kolmeen erilaista fonttia pidetään hyvänä määränä. (Hong 2018, 40-48). Kuviossa 3 on esimerkit päätteellisestä ja päätteettömästä fontista.



Kuvio 3: Päätteetön ja päätteellinen fontti

3.2.2 Värien rooli käyttöliittymissä

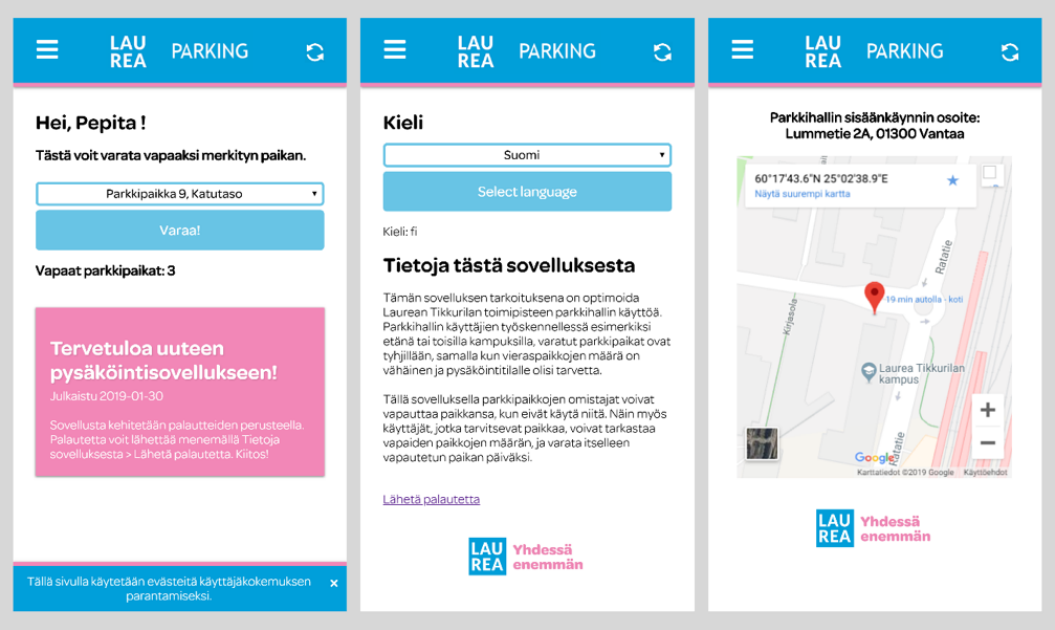
Käyttäjillä kestää noin 90 sekuntia luoda mielipide verkkosivusta, ja 62-90% tästä perustuu pelkästään värimaailmaan (Hong 2018, 49). Värit pelaavat siis suurta roolia koko visuaalisuuden rakentamisessa. Kuten typografiankin, myös värien valinnassa kannattaa huomioida brändi ja sen antamat mahdollisuudet sekä rajoitukset. Jokaisella värillä on assosiaatioita ja ne tuottavat käyttäjille erilaisia mielikuvia. Sosiaalisessa mediassa usein suositaan sinistä, kun taas ruokailuun ja syömiseen yhdistetään lämpimämpiä värejä. Lisää väreistä ja niiden merkityksistä kuviossa 4.

Keltainen	Oranssi	Punainen	Violetti	Musta
Lasten rakastama mutta etenkin miehet kokevat epämiellyttäväksi. Käytetään usein varoituksissa.	Lämmin mutta vaaraton. ”Uusi punainen”.	Symboloi rakkautta ja intohimoa. Liitetään usein rakkauteen, ruokaan sekä muotiin ja kosmetiikkaan.	Kuvastaa eleganssia, naisellisuutta ja luksusta. Ei aiheuta negatiivisia assosiaatiota.	Asiallinen, muodollinen, klassinen ja tavanomainen.
Vihreä	Sininen	Pinkki	Valkoinen	Ruskea
Yhdistetään luontoon ja hyvinvointiin. Vaaleat sävyt kuvaavat innovaatiota ja uusia ideoita.	Kuvastaa älykkyyttä ja tyyneyttä. Tummat sävyt luovat mielikuvaa luksuksesta.	Käytetään usein herättämään naisten huomiota. Mielletään tyttömäiseksi ja tuo mieleen makeat asiat ja vauvat.	Puhdas, viileä, moderni ja rauhallinen.	Yhdistetään luontoon. Useat ihmiset välttävät. Ilmaisee luotettavuutta.

Kuvio 4: Värejä ja niiden tarkoituksia (Hong 2018, 49-50)

Vaikuttavia käyttöliittymiä pystytään suunnittelemaan esimerkiksi monokromaattista väriskaalaa käyttäen, jolloin kaikki sivustolla käytössä olevat värit ovat päävärin eri sävyjä, joko tummempia tai vaaleampia tai vaihtoehtoisesti kirkkaampia tai haaleampia. Vaihtoehto monokromaattisuudelle on vastavärien hyödyntäminen. Vastavärejä ovat väriympyrän vastakkain olevat värit, ja ne luovat keskenään suurimman mahdollisen kontrastin. Vastavärejä ovat esimerkiksi sininen ja oranssi sekä vihreä ja punainen. Kolmas yleinen malli on käyttää analogista värimallia, jossa valitaan väriympyrässä vierekkäin olevat värit tukemaan toisiaan. Vierekkäin olevia värejä ovat esimerkiksi keltainen ja oranssi, sekä violetti ja sininen. (Beaird & George 2014, 62-72).

Pysäköintisovelluksen käyttöliittymä on esitelty kuviossa 5. Siinä on käytetty väreinä vaaleaa sinistä, joka kuvaa tuoreita ja raikkaita ideoita sekä yhdistetään älykkyyteen ja tyyneyteen. Sinistä tukee vaalea pinkki, joka puolestaan mielletään tyttömäiseksi ja makeanhimoa kasvattavaksi. Kumpikin väri on osa toimeksiantajan brändiä ja vahvassa käytössä yrityksen julkaisuissa niin verkossa, kuin painoviestinnässäkin. Käytetty päätteetön fontti tunnetaan minimalistisena, modernina ja puhtaana. Saman fontin eri leikkauksilla luodaan kontrastia otsikoiden ja leipätekstin välille. Logot ja ikonit luovat tekstin tueksi muita elementtejä, ettei lopputulos ole tylsä.



Kuvio 5: Pysäköintisovelluksen värimaailma ja ilme

3.3 Staattisesta sivusta dynaamiseksi sovellukseksi

Jotta sivulle saataisiin staattisuuden sijaan interaktiota sekä toiminnallisuutta, tulee sivulla käyttää jotain ohjelmointikieltä lisää, esimerkiksi PHP:ta. PHP on suosittu ohjelmointikieli, joka suoritetaan selaimen sijaan palvelimella. Se mahdollistaa esimerkiksi pääsyn tietokantoihin sekä palvelimen muihin tiedostoihin. (Beighley & Morrison 2009, 3).

PHP -tiedostot voivat sisältää sekä HTML-, että PHP-koodia, joista PHP-muotoinen osuus täytyy eritellä PHP-tunnisteiden sisälle. Tiedoston PHP-koodi määrittelee, miten HTML-koodia käsitellään. Tiedostot ja niiden sisältö prosessoidaan palvelimella ennen kuin ne lähetetään selaimelle HTML-muodossa. (Beighley & Morrison 2009, 3-24). Käytännössä siis voidaan ohjelmoida esimerkiksi aikamääritelty ehtolause; jos kello on vähemmän kuin 10, sivun otsikko voisi olla ”Hyvää huomenta!”, ja jos kello on enemmän, voisi otsikko olla ”Päivää!”. Tällöin sivun PHP käsiteltäisiin palvelimella niin, että selaimelle lähtee ainoastaan kellonaikaan sopiva otsikko.

PHP-ohjelmointikielestä julkaistaan edelleen tätä uusia versioita, ja tällä hetkellä sen 25:n olemassaolovuoden jälkeen tuorein versio on 7.3. Vaihtoehtoja PHP:lle löytyy useita, kuten JavaScriptiä hyödyntävä Node.js, joka on tällä hetkellä verkkokehitysmaailman kuumimpia teknologioita. Perinteisempiä vaihtoehtoja PHP:lle ovat esimerkiksi Java ja Python. (Nader 2019).

3.4 Tietokannat

Tietokantoja käytetään, kun jokin verkkosivulle syötetty tieto halutaan tallentaa, esimerkiksi lomakkeen sisältö. Tietokannat koostuvat tauluista, jotka sisältävät tallennettua dataa. Dataa voidaan jaotella tietokannan eri tauluihin rivit yhdistävällä avain-tiedolla, jonka avulla voidaan myös hakea ja yhdistellä eri tauluista rivejä. (Beighley & Morrison 2009, 109). Tietokantoja ja niiden sisältöä voidaan käsitellä ja kutsua joko tietokoneen komentorivityökalun kautta, tai graafisen käyttöliittymän kautta. Graafisen käyttöliittymän kautta tietokantojen ja taulujen hallinta voi olla hieman selkeämpää. (Beighley & Morrison 2009, 65-68).

Kun tietokantaan lisätään tietoa, tallentuu siitä rivi tietokannan tauluun. Saman taulun kaikilla riveillä tulee olla samat kentät. Tauluja luodessa määritellään skeema, eli rakenne, joka sisältää kenttien nimet ja tietotyypit sen mukaan, mitä niihin tullaan tallentamaan. Tietotyyppejä voivat olla esimerkiksi VARCHAR, johon voidaan tallentaa esimerkiksi tekstiä, numeroita tai merkkejä, DATE, johon tallennetaan päivämäärä tai DECIMAL, johon voidaan tallentaa numero -99.99 ja +99.99 väliltä. Jokaiseen tauluun tulisi myös lisätä uniikki perusavain –kenttä eli, *primary key*, jolla rivi, yksilöidään. (Darmawikarta 2014, 5-10).

Tietokannasta haetaan tietoa kyselemällä. Kyselyitä voi tehdä yhdelle tai useammalle taululle kerrallaan. Kyselyn vastauksena palautetaan tiedot, jotka vastaavat kyselyssä määriteltyjä ehtoja. Kyselyissä voidaan rajata hakua haluttuihin kenttiin, tauluihin ja muihin ehtoihin. Relaatiotietokannoissa kyselyitä kirjoitetaan SQL-kielellä, jonka lyhenne tulee sanoista Structured Query Language. Kyselyillä voidaan tietojen hakemisen lisäksi, päivittää, lisätä ja poistaa tietoa. (Darmawikarta 2014, 12-15). Taulukossa 1 esitellään erilaisia SQL-kyselyitä.

`SELECT id, State, User FROM Parkingslots WHERE State = ”Free”;`	`DELETE FROM Bookings WHERE id = $poistettava;`	`UPDATE Parkingslots SET State = 'In Use', User = '$kayttaja', date = now() WHERE id=$poistettava;`
Valitsee Parkingslots-taulusta rivit, joilla State -kentän arvo on ”Free”. Näyttää taulusta vain kentät id, State ja User.	Poistaa Bookings-taulusta rivin, jonka id vastaa PHP-muuttujan arvoa.	Päivittää Parkingslots-taulun riviä, jolla id vastaa PHP-muuttujan arvoa. Muuttaa State-kenttään arvon ”In Use”, User-kenttään PHP-muuttujan arvon, johon on tallennettu kirjautuneen käyttäjän käyttäjätunnus ja date-arvon kyselyn aikaleimalla.

Taulukko 1: SQL-kyselyitä

3.5 Ohjelmistotestaus

Testaamalla kehitettyä ohjelmistoa pyritään selvittämään, toimiiko kaikki ohjelmiston osa-alueet kuten pitää ja onko kehitetty ohjelmisto sellainen kuin sen halutaan olevan. Jussi Pekka Kasurinen tiivistää ohjelmistotestauksen kirjassaan Ohjelmistotestauksen käsikirja (2013) sanoin: ”Varmistetaan, että toimiiko tuote ja onko se tehty oikein.”

Ohjelmistokehitysprosessissa testausta voi tehdä usealla eri tavalla. Jos kehityksessä käytetään vesiputousmallia, jossa edetään työvaiheesta seuraavaan, suoritetaan kaikki testaus ohjelmistotuotteen toteutusvaiheen jälkeen, juuri ennen ylläpitovaiheeseen siirtymistä. Vesiputousmallissa siis testataan vain yhdessä vaiheessa, kun varsinainen kehitystyö ja suunnittelu on jo tehty. Tästä syystä suositeltavampi malli prosessille on esimerkiksi V-malli, jossa testaus ja ohjelmiston rakentaminen kulkevat käsikädessä koko prosessin ajan. Vaatimuksia testataan hyväksymistestauksella, määrittelyä järjestelmätestauksella, suunnittelua ja sen toteutumista integraatiotestauksella ja itse ohjelmointia yksikkötestauksella. (Kasurinen 2013).

4 Tutkimusmenetelmät

Laadullisessa tutkimuksessa tuloksia tarkastellaan niiden sisällön perusteella, kun taas kvantitatiivisessa, eli määrällisessä tutkimuksessa tarkastellaan enemmän lukuja ja määriä. Tiedonkeruu näissä menetelmissä eroaa jonkin verran; laadullisessa tutkimuksessa prosessi on usein joustavampi ja aineisto kerätään todellisissa tilanteissa. Laadullisen tutkimuksen yleisiä menetelmiä ovat erilaiset haastattelut, kuten ryhmä- ja teemahaastattelut, havainnointi sekä esimerkiksi kuvien, videoiden, pöytäkirjojen tai vuosikertomusten tulkitseminen. Laadullista aineistoa analysoidaan usein keräämisen kanssa käsi kädessä. Aineiston analysoinnilla pyritään jäsentämään kerättyä aineistoa, jotta tulkinta olisi selkeämpää. Vaiheina analysoinnissa usein aloitetaan tutustumalla aineistoon, ja lukemalla se läpi. Luettu aineisto luokitellaan tyypin, teeman tai sisällön perusteella. Viimeiseksi aineistosta löydetyt havainnot liitetään asiayhteyteen, eli kontekstiin. (Järvenpää 2006)

4.1 Haastattelu ja kysely

Haastattelussa haastattelija ja haastateltava kohtaavat joko kasvokkain tai esimerkiksi puhelimitse. Kyselyssä puolestaan vastaaja vastaa kysymykseen tai kysymyksiin itsenäisesti esimerkiksi lomakkeella tai kirjoittaen. Kummassakin menetelmässä kysymykset voivat olla suljettuja tai avoimia. Avoimessa kysymyksessä vastaaja itse päättää miten hän siihen vastaa, kun taas suljetussa kysymyksessä tarjotaan vaihtoehdot, joista vastaaja valitsee osuvimman. (Routio 2007)

Haastattelu vaatii kummaltakin osapuolelta enemmän aikaa, mutta menetelmänä se joustaa ja mukautuu tilanteessa kyselyä enemmän. Haastattelussa tärkeää on dokumentoida aineisto esimerkiksi nauhoittamalla se, tai tekemällä tarkat muistiinpanot. Haastattelu mahdollistaa

tarkentavien kysymysten esittämisen. Haastattelumalleina voidaan käyttää esimerkiksi teema- tai ryhmähaastattelua. Menetelmänä haastattelu sopii tilanteisiin, joissa tutkija ei osaa tai pysty etukäteen määrittelemään kysyttäviä kysymyksiä (Routio 2007).

Kyselyssä kysymykset ovat valmiiksi määriteltyjä, eli jos niitä halutaan muokata, voidaan joutua tekemään uusi kysely. Kysymysten täytyy olla mahdollisimman yksiselitteisiä ja ymmärrettäviä, sillä kyselyssä ei ole mukana haastattelijaa, joka voisi tarvittaessa selventää kysyttyjä kysymyksiä. Kysely sopii hyvin tilanteisiin, joissa tutkimuksen resurssit ovat pienemmät tai tutkimukseen osallistuu paljon vastaajia. (Routio 2007).

4.2 Sovelluskehitysprojektin tiedonkeruu

Projektissa käytettiin osana tiedonkeruuta epävirallisia asiantuntijahaastatteluja ja kyselyitä. Etenkin tunnistautumisen integraatiossa hyödynnettiin organisaation sisäistä asiantuntijaosaamista. Haastattelut sovittiin etukäteen, ja niille määriteltiin aihealue. Aihealueen perusteella pyydettiin asiantuntevaa osaajaa haastateltavaksi. Haastatteluissa käsiteltiin prosessissa ilmenneitä ongelmakohtia ja pohdittiin niihin ratkaisuja. Ympäristöksi haastattelulle valittiin työtila tietokoneiden ääressä, jotta käsiteltäviä aihealueita pystyttiin demonstroimaan ja lisätietoa hankkimaan haastattelun lomassa.

Palautteita ja kehitysideoita sovelluksen ensimmäisestä versiosta kerättiin kyselemällä käyttäjien kokemuksia ja mielipiteitä sisäisen viestintäkanavan Intran kautta. Intraan julkaistiin tiedote kehitystyön alkamisesta, johon liitettiin mukaan muutama avoin kysymys. Halukkaat käyttäjät saivat vastata kysymyksiin joko julkisesti tiedotteen kommentteihin tai sähköpostitse. Kerätty aineisto luokiteltiin aihealueittain ja käsiteltiin muotoon, jossa sitä voitiin hyödyntää vaatimusmäärittelyssä.

Aluksi luokat, joihin palautetta jaoteltiin, olivat sovelluksen ideaan liittyvät palautteet ja sovellukseen teknisesti liittyvät palautteet. Näistä tekniseen toteutukseen liittyvät jaoteltiin vielä sisältönsä mukaan toivottujen uusien ominaisuuksien- ja vanhojen toimintojen ja ongelmien kehittäviin palautteisiin. Koska sovelluksen idea perustuu täysin parkkipaikkojen laskutusmalliin, tallennettiin ne myöhempää tarkastelua varten erikseen, jos tulevaisuudessa koko pysäköintikonseptille tehdään palvelumuotoilua. Palautteiden luokkia kuvataan kuviossa 6.

Kuvio 6: Palautteiden luokittelu

4.3 Reliabiliteetti ja validiteetti

Tutkimuksen validiteetilla tarkoitetaan sitä, kuinka hyvin tutkimusmenetelmä soveltuu tutkittavan asian tutkimiseen. Tässä työssä käytetystä, avoimia kysymyksiä sisältäneestä, kyselystä saatiin hyödyllisiä vastauksia, mutta isommalla otannalla se olisi ollut varmasti kattavampi. Kyselyn kautta tuli suhteessa koko Laurean henkilöstöön pieni määrä vastauksia. Toisaalta koska alkuperäisen sovelluksen käyttöaste oli niin pieni, käyttäjillä olisi pitänyt olla parempi ymmärrys siitä, mihin sovellus oli tarkoitettu ja missä muodossa, jotta he olisivat osanneet antaa tarkempia kehitysideoita.

Uuden version julkaisun jälkeen saatiin kerättyä paremmin palautetta käyttäjiltä. Tämä johtui siitä, että käyttäjät näkivät sen, mihin sovelluksella pystytään ja mitä se mahdollistaa. Uuden sovellusversion myötä myös yhä useampi henkilökunnasta päätyi käyttämään ja kokeilemaan sitä. Käyttöasteen lisääntyessä myös kehitysideoita, ja muita palautteita saatiin sovelluksen palautelomakkeen sekä intran kautta.

Palautteita kerättäessä luotettavuus ei kärsinyt, sillä palautteita annettiin omalla nimellään. Näin myös nähtiin, keneltä vastaus oli tullut, ja olivatko käyttäjät jakaneet vastauksensa eri viesteihin. Avoimeen kyselyyn vastanneilla oli myös eri näkökulmia, sillä vastauksia kerättiin niin parkkipaikan omistavilta käyttäjiltä, kuin halukkailta varaajilta.

Mittauksessa reliabiliteetti osoittaa, että tutkimuksen tulos ei ole sattumanvarainen ja että mittaustulos olisi toistettavissa (Hiltunen 2009.) Kyselyssä kysymykset olivat mahdollisimman yksiselitteisiä ja helposti ymmärrettäviä, mutta vastausten toistettavuuteen vaikuttaa moni tekijä. Kun kysytään palautetta kokemuksista johonkin asiaan liittyen, vastaukseen vaikuttaa paljon esimerkiksi mielentila, käyttökokemukset ja -määrät. Esimerkiksi käyttäjä, joka on käyttänyt kerran sovellusta ja on ollut siihen erittäin tyytyväinen antaa varmasti parempaa

palautetta kuin käyttäjä, joka on käyttänyt samaa sovellusta kymmeniä kertoja, joista muutamalla on ilmennyt ongelmia. Jos ongelmat ovat ilmenneet lähipäivinä, saattaa palaute olla negatiivisempaa kuin tilanteessa, joissa viimeisimmästä ongelmasta on enemmän aikaa.

Ilmiön jatkuvuutta voidaan pitää jonkinlaisena keinona tarkastaa reliabiliteetti (Hiltunen 2009.) Vanhan sovelluksen kehitysideoita kerättäessä palautteiden sisällön reliabiliteettia nostaa ilmiön jatkuvuus, jolla tarkoitetaan sitä, että samankaltaisia palautteita heikoista käyttäjäkokemuksista ja ongelmista oli vastaanotettu jo pidemmän ajanjakson aikana. Todettavissa siis oli, etteivät vanhan sovelluksen ongelmat olleet ainakaan täysin ainutkertaisia tai käyttäjäkohtaisia.

5 Kehittämistyön toteutus

Kehitystyö alkoi tutkimalla alkuperäisen pysäköintisovelluksen toiminnallisuutta, käyttöä ja kehitysmahdollisuuksia. Tarkoituksena oli selvittää, mikä olisi kannattavin tapa jatkaa kehitystyötä. Alkuperäinen toteutus oli rakennettu sovelluksien tekoon tarkoitetulla ohjelmalla nimenomaan sovelluskaupasta ladattavaksi natiivisovellukseksi. Sovelluksen ylläpidosta vastaava taho kuitenkin päätti julkaista sen selainpohjaisena, jolloin useimmat ongelmista ilmenivät.

Sovelluksen tutkiminen tehtiin ottamalla käyttöön testipalvelin ja laittamalla sinne kopio julkaistusta versiosta. Testipalvelimeen otettiin yhteys WINSCP-ohjelmalla, joka tarjoaa graafisen käyttöliittymän palvelimelle, mikä auttaa kansiorakenteen hahmottamisessa. Tutkimalla kopiota, vältyttiin siltä, ettei sovellukseen tule tuotantopuolella käyttökatkoja. Ilmenneet ongelmat kirjattiin ylös, ja toteutustapoja niiden korjaamiseksi ja selvittämiseksi alettiin vertailla.

Toiminnallisuuden tutkimisen jälkeen aloitettiin kokoamaan käyttäjiltä saatujen palautteiden perusteella vaatimusmääritelmää sovelluksen uudelle versiolle. Kootut vaatimukset järjestettiin niille annettujen prioriteettien mukaan tärkeimmästä vähiten tärkeään. Vaatimusmäärittelmään ja sen kokoamisprosessiin palataan tarkemmin tässä opinnäytteessä omassa luvussaan.

Kun sovellukseen haluttu toiminnallisuus oli määritelty, lähdettiin rakentamaan itse sovellusta. Kehitystyö aloitettiin tärkeimmiksi määriteltyjen vaatimusten ohjelmoinnista ja kun ne oltiin saatu toimiviksi kokonaisuuksiksi, lähdettiin ympärille rakentamaan seuraavaksi tärkeimpiä ominaisuuksia ja toiminnallisuuksia. Edellä mainitulla menetelmällä käytiin läpi koko vaatimusmäärittelmä, kunnes kaikki vaaditut toiminnallisuudet oli tehty.

Tuotteen toimintaa testattiin koko kehityskaaren ajan. Testauksella varmistettiin se, että kaikki toiminnot tekevät juuri niin kuin pitääkin ja että ensimmäisinä rakennetut toiminnallisuudet eivät kärsi, kun lisätään uusia toimintoja. Testausta tehtiin eri laitteilla ja niiden verkkoselaimilla.

Aikataulullisesti kehittämistyö aloitettiin loppusyksystä 2018 ja se jatkui vuoden 2019 kevättalven puolelle. Valmis sovellus julkaistiin tuotantoon 20.2.2019.

Kehitystyötä aloitettaessa sille ei määritelty mitään tarkkaa kehitysmenetelmää, jota tultaisiin seuraamaan ja noudattamaan projektin ajan. Jälkeen päin tarkastellessa kehitystyön kulusta kuitenkin erottuu selkeitä toimintamalleja ja -tapoja, jotka ovat yhtenäisiä ketterien ohjelmistokehitysmenetelmien kanssa.

Tässä sovelluskehitysprojektissa yhtenäistä ketterään kehitysmenetelmään nimeltä scrum, oli ajatus kehitysjonosta. Listausta, johon oltiin koottu vaatimusmäärittelyssä haluttuja toimintoja, ominaisuuksia ja vaatimuksia käsiteltiin samaan tyyliin kuin scrum-projektissa käsitellään kehitysjonoa. Listaukseen oli pilkottu kohtia, joiden etenemistä seurattiin merkitsemällä niille tiloja. Varsinaisia erillisiä sprinttejä ei projektissa hyödynnetty, mutta joitain tavoitteita asetettiin viikko- ja päiväkohtaisesti. Itseohjautuvuus oli myös tärkeässä roolissa, vaikka projektissa ei ollut mukana varsinaista työtiimiä scrum-mastereineen ja omistajineen. Ketteryys projektissa ilmeni esimerkiksi niin, että vaatimuslistaus päivittyi projektin edetessä ja työskentelymalli oli hyvin itseohjautuva.

5.1 Vaatimusmäärittely

Alkuperäinen suunnitelma oli parannella olemassa olevaa pysäköintisovellusta ja tehdä siitä toimivampi ja käytettävämpi. Sovellukseen perehdyttäessä kuitenkin ilmeni, että alkuperäisen version koodi kaipaisi niin paljon uudelleenkirjoittamista, että olisi kannattavampaa tehdä kokonaan uusi versio, joka toiminnallisuudeltaan vastaisi edeltävää versiota. Lisäksi sovellukseen toivottiin uusia lisäominaisuuksia, jotka olisivat helpompia ja nopeampia rakentaa puhtaalta pöydältä.

Vaatimusmäärittely koostettiin keräämällä vanhan sovelluksen toiminnallisuus myös uuden sovelluksen vaatimusmäärittelyyn. Vanhojen toimintojen lisäksi käyttäjiltä kerättiin kommentteja ja palautteita sovelluksesta ja sen toiminnasta. Kommenttien ja palautteiden perusteella muodostettiin lisää vaatimuksia täydentämään määrittelyä. Projektin edetessä listaan lisättiin useita rivejä, kun uusia vaatimuksia ilmeni, mutta tärkeimmät vaatimukset pysyivät samoina. Vaatimusmäärittelyn ensimmäinen versio on nähtävissä taulukossa numero 2.

1	Oman parkkipaikan vapauttaminen
1	Vapaan parkkipaikan varaaminen
1	Selainpohjaisuus
1	Alustariippumattomuus
1	Varaaminen samalle päivälle
1	Vapauttaminen tälle päivälle
2	Tunnistautuminen
2	Parkkipaikan yhdistäminen tietoihin
2	Vapaiden paikkojen hallinnointi
2	Oman vapautuksen peruminen
2	Oman varaamisen peruminen
2	Vapauttaminen tulevaisuuteen
3	Varattujen paikkojen hallinnointisivu
3	Paikan vapauttaminen aamupäiväksi / iltapäiväksi
3	Tiedotteita -ikkuna
4	Uusien käyttäjä-parkkipaikka parien yhdistäminen
4	Hallinnointisivulle pääsy
5	Sensorit vieraspaikoille, jotta näkee onko vapaa / varattu

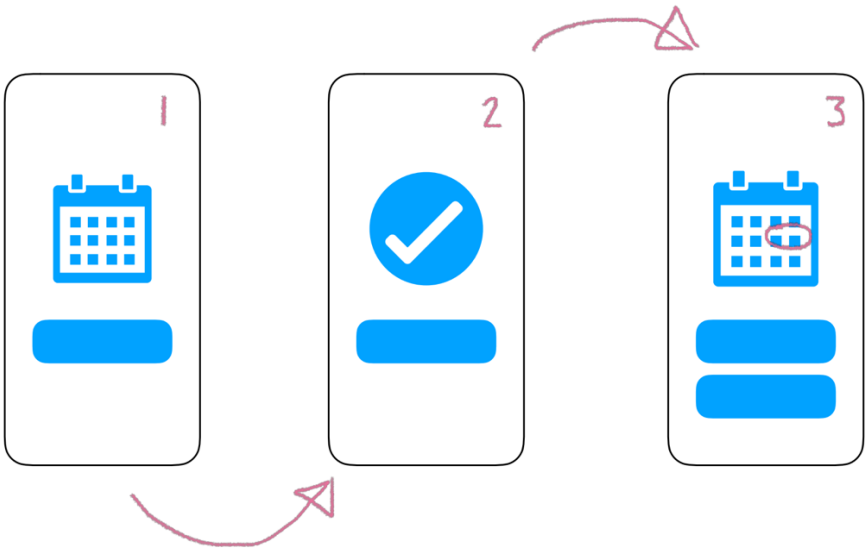
Taulukko 2: Ensimmäinen versio vaatimusmäärittelytaulukosta

Käyttäjiltä kyseltiin palautetta ja toiveita ammattikorkeakoulun henkilökunnan sisäisen kommunikaatiokanavan Intran kautta. Verkkokyselyiden lisäksi esimerkiksi Tikkurilan toimipisteen korkeakouluisänniltä kysyttiin heidän vastaanottamiaan palautteita, joita he olivat käyttäjiltä saaneet, sekä havaintoja sovelluksesta ja sen ympärillä olevista toiminnoista. Kokonaisuuden hallinnan helpottamiseksi kaikki kerätyt vaatimukset koottiin yhteen tiedostoon seurannan helpottamiseksi. Listatut vaatimukset järjestettiin prioriteettiarvonsa mukaan, ja uusia vaatimuksia lisätessä ne laitettiin paikoilleen prioriteetin mukaan, eikä esimerkiksi niiden vastaanottojärjestyksessä. Aina, kun jokin vaatimus oli täytetty ja valmis, merkittiin se listaan, jolloin projektin etenemistä oli helppoa seurata.

5.2 Sovelluksen näkymät

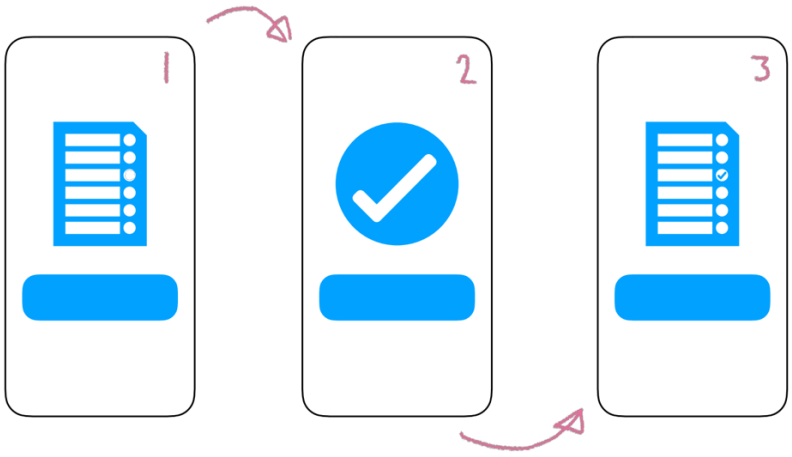
Sovelluksen rakenne sisältää erilaisia näkymiä, joista osa on kaikille käyttäjille samanlaisia ja osan sisältö riippuu siitä, tunnistetaanko käyttäjä parkkipaikan omistajana, eli vapauttavana käyttäjänä, varaavana käyttäjä tai ylläpitävänä käyttäjänä. Kirjautuneelle käyttäjälle, joka omistaa parkkipaikan, aloitusnäkymä sisältää paikan vapauttamiseen tarvittavat toiminnot. Suoraan etusivulla on tiedot käyttäjän parkkipaikasta (paikan numero, tila ja kerros) ja kentät vapautuksen aloitus- ja lopetuspäiville. Kenttiä valitessa aukeaa kalenterinäkymä, josta on helpompi valita vapautuksen päivämäärät. Vapautuksen aktivointia varten on vahvistuspainike. Kun päivämäärät on valittu ja painiketta painittu, käyttäjä ohjataan vahvistussivulle, jossa esitetään juuri kirjatun vapautuksen tiedot. Vahvistusnäkymästä pääsee takaisin aloitus-

näkymään painikkeella. Kun paikka on vapautettu, aloitusnäkymässä voi lisätä uuden vapautuksen eri päiville tai peruuttaa käyttäjän aiemmin kirjaamia vapautuksia. Vapauttavan käyttäjän aloitusnäkymää ja siinä etenemistä on kuvattu kuviossa 7.



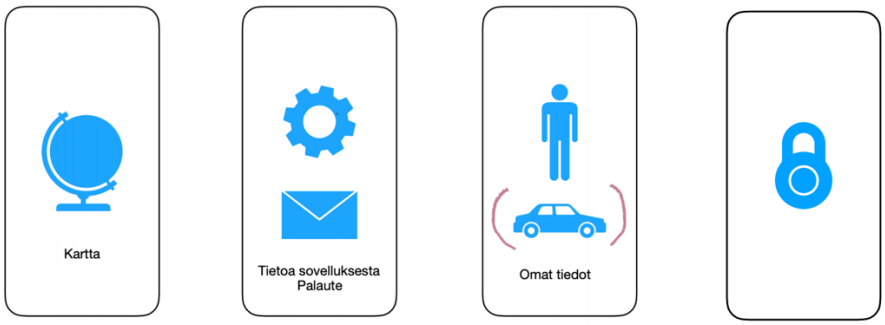
Kuvio 7: Vapauttavan käyttäjän aloitusnäkymän toimintoja

Varaavalle käyttäjälle aloitusnäkymä on hieman yksinkertaistetumpi. Toimintoina käyttäjälle on valita alasvetovalikosta vapautettu paikka, jonka hän haluaa varata, ja vahvistaa varaus vahvistuspainikkeella. Jos vapautettuja paikkoja ei ole, vahvistuspainiketta ei näytetä ja valikossa on ilmoitus tilanteesta. Vapautetun paikan varaaminen vahvistetaan painikkeella, josta ohjataan vahvistussivulle, joka esittää varauksen tiedot. Vahvistusnäkymästä palataan painikkeella aloitusnäkymään, josta näkee varatun paikan tiedot ja voi halutessaan poistaa varauksen sitten, kun paikkaa ei enää tarvitse. Kaikki varaukset poistuvat viimeistään vuorokauden vaihtuessa. Varaavan käyttäjän aloitusnäkymää ja sen toimintoja kuvataan kuviossa 8.



Kuvio 8: Varaavan käyttäjän aloitusnäkymä ja sen toiminnot

Aloitusnäkymien lisäksi sovelluksessa on karttanäkymä, josta näkee parkkihallin tarkan sijainnin sekä ajo-ohjeet. Tietoja sovelluksesta -näkymässä esitellään lyhyesti sovellusta ja sen käyttötarkoitusta. Samassa näkymässä on myös sovelluksen kielen valinta sekä linkki palautelomakkeeseen. Koska näkymän toimintoina on myös sovelluksen asetusten muuttamista, kuvataan sitä sovelluksen navigaatiossa akseli-ikonilla, jolla usein kuvataan asetusnäkymiä. Omat tiedot -näkymässä näytetään käyttäjälle hänen omat tietonsa. Jos käyttäjällä on parkkipaikka, näytetään myös sen tietoja, kuten kerros ja parkkiruudun numero. Viimeinen kaikille yhteinen näkymä on uloskirjautumisnäkymä. Edellä esiteltyjä näkymiä esitetään kuviossa 9.



Kuvio 9: Muita sovelluksen näkymiä

Ylläpitäville käyttäjille näytetään myös ylläpitonäkymä, josta voidaan muokata parkkipaikkojen ja käyttäjien tietoja, lisätä ja poistaa uutisia ja hallinnoida ylläpitäviä käyttäjiä. Ylläpitonäkymästä voi myös lisätä ja poistaa vapautuksia kaikilta peruskäyttäjiltä, ja varata paikkoja myös organisaation ulkopuolisille käyttäjille sähköpostiosoitteen perusteella. Ylläpitäjän tekemästä paikkavarauksesta lähtee automatisoitu vahvistussähköposti käyttäjälle, jolle paikka on varattu.

5.3 Sovelluksen toiminnallisuus

Laurea-ammattikorkeakoulun Tikkurilan toimipisteen kellarikerroksessa sijaitsee parkkihalli, josta korkeakoulun henkilökunta voi ostaa itselleen pysäköintiruudun käyttöoikeuden. Monet henkilökunnan jäsenet, jotka omistavat ruutuja kuitenkin työskentelevät myös toisilla kampuksilla tai tekevät etätöitä, joten paikkoja on usein vapaana. Näille vapaille paikoille ei kuitenkaan voi muut pysäköidä, sillä ne ovat jonkun toisen henkilön omia. Korkeakoulun pysäköintitiloissa on muutamia vieraspaikkoja, jotka ovat aktiivisessa käytössä.

Toteutustavaksi sovellukselle valittiin PHP-ohjelmointikieli sekä tietojen tallentamiseen SQL -relaatiotietokanta. Käyttöliittymän rakentamisessa on käytetty HTML- ja CSS-kieliä. Valintaa perustelee osaaminen toimeksiantajan suunnalta, sillä jos sovellusta halutaan jatkossa kehittää lisää, tai siinä ilmenee ongelmia, voidaan niihin puuttua varsinaisen kehitystyön jälkeen. Lisäksi PHP on kielenä vakaa, sillä on kattava dokumentaatio ja siitä on paljon tietoa ja esi-

merkkejä saatavilla. Joitain toiminnallisuuksia sovellukseen tehtiin myös hyödyntäen JavaScriptiä. Tällaisia osia olivat esimerkiksi vapautusnäkymän kalenteri ja informatiiviset pop-up –viestit. Koodieditorina työssä käytettiin avoimen lähdekoodin sovellusta nimeltä ATOM, joka ajoi asiansa erinomaisesti.

5.3.1 Käyttötapaukset

Päätoiminnallisuus ja idea sovelluksessa on tarjota käyttäjille mahdollisuus vapauttaa oma parkkiruutunsa tarjolle silloin, kun ei sitä itse tarvitse. Vapaaksi merkityn paikan voi sovelluksen toinen käyttäjä (joka ei omista parkkiruutua), varata ja pysäköidä siihen enintään yhdeksi päiväksi. Sovellus mahdollistaa myös pidempien ajanjaksojen vapauttamisen, esimerkiksi loman ajaksi. Edellä mainittuja käyttötapauksia avataan lisää taulukoissa 3 ja 4.

Nimi:	Parkkipaikan vapauttaminen
Suorittajat:	Käyttäjä, jolla on maksettu paikka
Esiehdot:	Käyttäjän on oltava tunnistettu (sisäänkirjautuminen)
Kuvaus:	Käyttäjä vapauttaa parkkipaikkansa, valitsemalla vapautuksen aloitus- ja lopetuspäivät näkymän kalenterista ja painamalla painiketta, jossa lukee "Merkkaa vapaaksi". Käyttäjälle näytetään vahvistusikkuna, jossa lukee vahvistus paikasta ja päivämääristä. Samalla tietokantaan päivittyy, että paikan tila on vapaa valittuina päivinä.
Poikkeukset:	Jos vapautus tehdään samana päivänä vain kyseiselle päivälle, päivämääriä ei tarvitse valita, vaan sovellus antaa automaattisesti meneillään olevan päivän aloitus- ja lopetuspäiväksi.
Lopputulos:	Tietokantaan päivittyy, että paikka on valittuina päivinä vapaa.

Taulukko 3: Käyttötapaus: Parkkipaikan vapauttaminen

Varausten ja vapautusten lisäksi sovelluksessa voidaan jakaa käyttäjille tiedotteita. Ylläpitävä käyttäjä voi hallinnointinäkymässä julkaista ja poistaa tiedotteita, joista aina uusin näkyy peruskäyttäjille sovelluksen etusivulla. Käyttäjille julkaistavat uutiset voivat koskea esimerkiksi parkkihallin käyttöön liittyviä poikkeustiloja tai huoltotöitä, kuten hallin pesua tai sovelluksen omia asioita, kuten käyttäjiä aktivoivia kampanjoita.

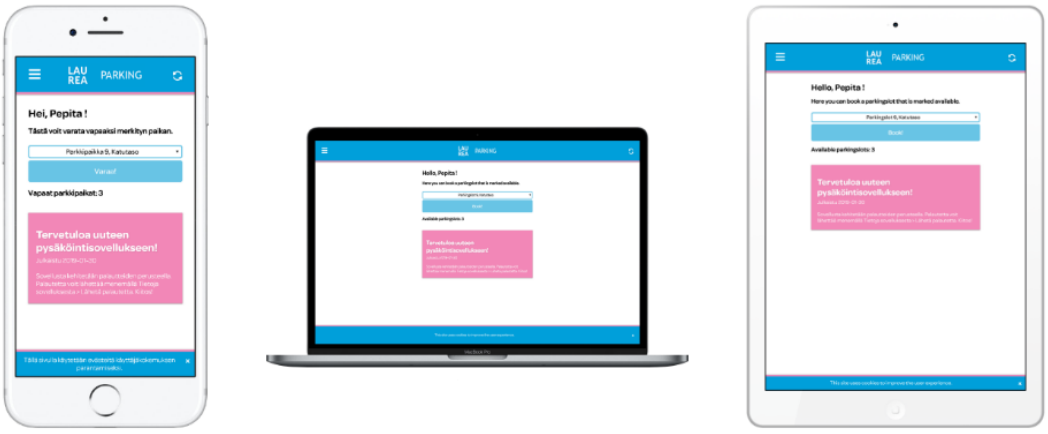
Nimi:	Parkkipaikan varaaminen
Suorittajat:	Käyttäjä, jolla ei ole maksettua paikkaa
Esiehdot:	Käyttäjän on oltava tunnistettu (sisäänkirjautuminen)
Kuvaus:	Käyttäjä näkee aloitusnäkymässä vapaat parkkipaikat ja valitsee alasvetovalikosta haluamansa paikan. Paikan valittuaan käyttäjä painaa "Varaa paikka" -painiketta ja hänelle näytetään vahvistus varauksesta ja paikan tiedoista. Tietokantaan päivittyy tieto siitä, että vapaaksi merkattu paikka on varattu toiselle käyttäjälle.
Poikkeukset:	Jos vapaita paikkoja ei ole, käyttäjälle näytetään "Ei vapaita paikkoja" -ilmoitus ja näkymästä piilotetaan "Varaa paikka" -painike.
Lopputulos:	Tietokantaan päivittyy, että käyttäjälle on varattu valittu paikka.

Taulukko 4: Käyttötapaus: Parkkipaikan varaaminen

5.3.2 Responsiivisuus

Koska sovellusta halutaan käyttää sekä älypuhelimella, että tietokoneella, täytyy sen olla responsiivinen. Responsiivisuus verkkosovelluksissa tai –sivuilla tarkoittaa sitä, että näytettävä näkymä mukautuu esimerkiksi laitteen näytön koon mukaan.

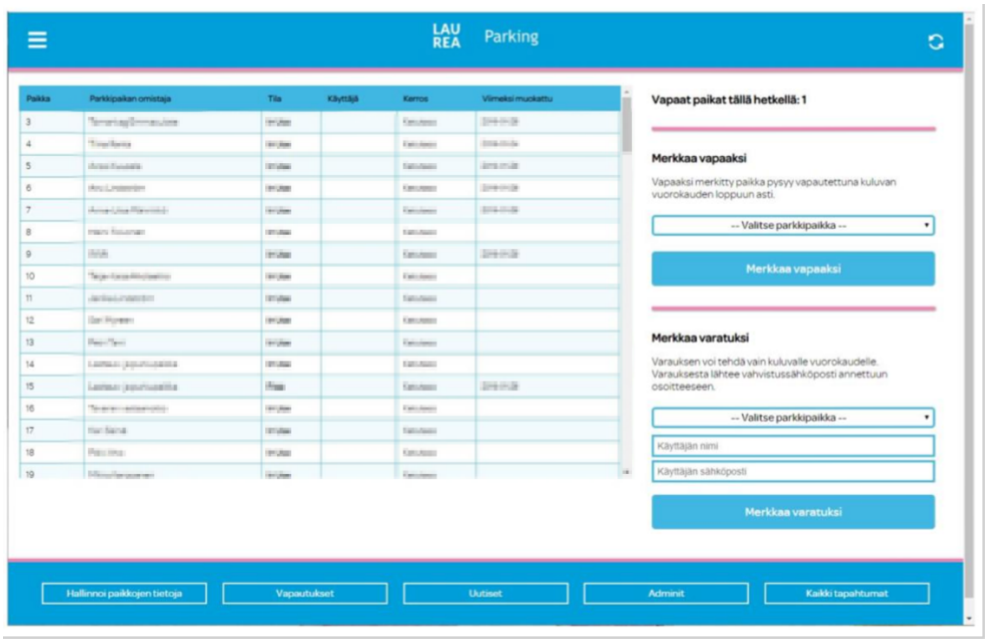
Peruskäyttäjät voivat varata ja vapauttaa paikkojaan kätevästi älypuhelimella, tai omalta työpisteeltään tietokoneella. Etenkin hallinnointia ja hallinnointisivua varten on tärkeää, että sovelluksesta on olemassa myös tietokoneella käytettävä työpöytäkäyttöliittymä. Näkymässä on esillä paljon informaatiota ja toimintoja, joten käyttö on helpompaa ja selkeämpää tietokoneella. Täyskokoisessa käyttöliittymässä kaikki elementit mahtuvat näytölle oikeassa koossaan, mikä helpottaa tekstisisällön lukemista, toimintojen löytämistä sekä yleistä käytön sujuvuutta. Peruskäyttäjälle sen sijaan se, että sovellus toimivuus ja helppokäyttöisyys älypuhelimella on tärkeää, sillä älypuhelin on se laite, jolla vapautuksia ja varauksia useimmiten tehdään. Kuviossa 10 on nähtävillä pysäköintisovelluksen aloitusnäkymä erikokoisilla näytöillä varustetuilla laitteilla.



Kuvio 10: Sovelluksen aloitusnäkymä eri laitteilla

Sovelluksen ylläpitäjät toivoivat sovelluksesta selaimessa toimivaa kokonaisuutta, eikä sovelluskaupasta ladattavaa natiivisovellusta. Selainpohjainen sovellus toimii laitteen verkkoselaimessa, joita ovat esimerkiksi Chrome, Safari, Edge ja Mozilla. Selainpohjaisuus mahdollistaa sovelluksen käytön kaikilla laitteilla, missä on verkkoselain. Se ei siis rajoita käyttömahdollisuuksia esimerkiksi käyttöliittymän tai puhelimen merkin tai mallin takia. Samaa sovellusta voi täten käyttää niin älypuhelimella, tabletilla kuin perinteisellä tietokoneellakin.

Responsiivinen ulkoasu on toteutettu sovellukseen suunnittelemalla käyttöliittymä mobile-first -ajatusmallilla, eli priorisoimalla käyttöliittymän mobiiliversio tärkeimmäksi. Työpöytäkäyttöliittymässä käytettiin mobiilista tuttuja elementtejä, kuten burgerivalikkoa ja keskitettyä sisältöä. Sivun elementtien leveyksissä hyödynnettiin toisistaan riippuvaisia kokoja, jolloin sivu skaalautuu kauniisti eri levyisillä näytöillä. Admin-näkymässä toimintoja sijoitettiin isommalle alueelle, sillä se optimoitiin pääasiassa perinteisen kokoisille tietokoneiden näytöille. Admin-näkymä toimii myös mobiililaitteella, mutta koska toimintoja on paljon ja ne sijoittuvat allekkain samalle sivulle, on käyttö himan hitaampaa. Admin-näkymän mobiilikäyttöä ei kuitenkaan priorisoitu niin tärkeäksi, että sille oltaisiin rakennettu mobiiliin selkeästi omaa versiotaan.



Kuvio 11: Admin-näkymä

5.3.3 Monikielisyyden toteutus

Sovellukseen toivottu kaksikielinen käyttöliittymä toteutettiin hyödyntämällä sessiokohtaisia evästeitä. Kaikki sovelluksen tekstit ovat tallennettu erillisiin kielitiedostoihin, joiden valittuja rivejä kutsutaan niissä paikoissa, joihin tekstiosuus halutaan. Sovelluksen aloitusnäkymässä määritellään evästeen arvoksi automaattisesti ”FIN” eli suomen kieli. Sovelluksen asetuksien kautta eväste voidaan kuitenkin tarvittaessa korvata uudella arvolla, kuten ”ENG”, jolloin ohjelmistokoodista luetaan, että käytetään englannin kielistä kielitiedostoa. Toteutetun kaksikielisen käyttöliittymän kielivaihtoehdot ovat suomi ja englanti, mutta lisää kieliä on helppo lisätä tulevaisuudessa, jos niin halutaan.

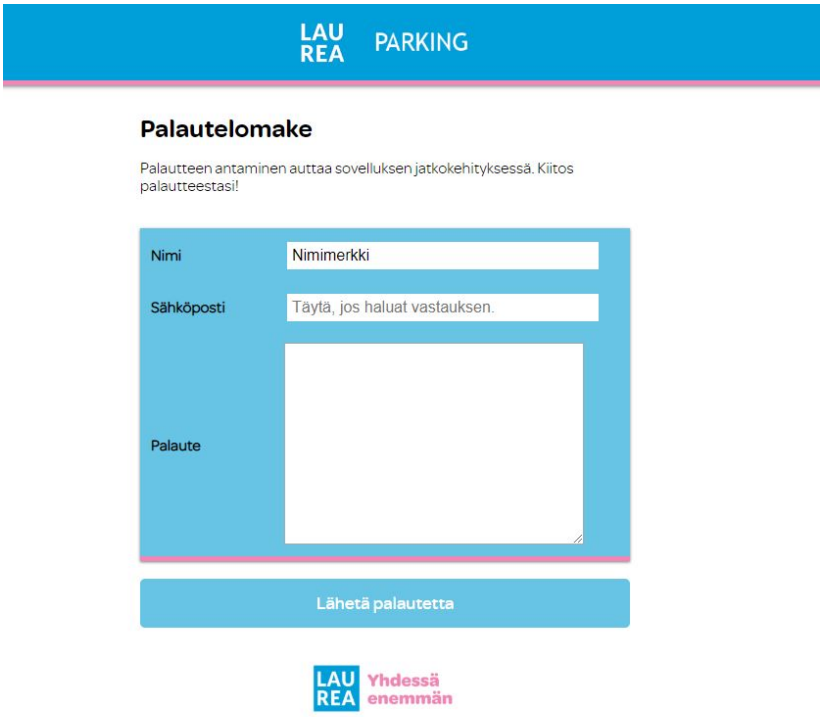
Rakenteellisesti siis luodaan PHP-muuttujia ja annetaan niille arvot, jotka sisältävät halutun käännöksen. Evästeisiin tallennettu tieto valitusta kielestä määrittelee sen, kumpaa tiedostoa käytetään. Ohjelmistokoodin kohdissa, joihin käännös halutaan sijoittaa, kutsutaan muuttujaa, jonka arvona on kyseisen kohdan haluttu käännös. Kuviossa 12 esitetään kuvakaappaus kielitiedostojen valikoiden käännöksistä.

Kuvio 12: Kuvakaappaus kielitiedostoiden valikon käännöksistä

5.3.4 Palautteen kerääminen

Palautteen keräämisen helpottamiseksi sovellukseen rakennettiin palautelomake, jolla käyttäjä voi lähettää palautetta suoraan sovelluksesta. Lomakkeeseen voi jättää nimen, sähköpostiosoitteen sekä itse palautteen. Näistä pakollisia kenttiä ovat kuitenkin vain palaute ja nimi, mutta nimen voi halutessaan vaihtaa esimerkiksi nimimerkiksi, jos haluaa pysyä anonyyminä. Lomakkeeseen on kirjoitettu huomautus siitä, että jos palautteelleen haluaa vastauksen, pitäisi sähköpostiosoitekin kirjoittaa mukaan sille varattuun kohtaan.

Lomakkeella lähetetyt sähköpostit määriteltiin lähteväksi Laurean tietohallinnon ylläpitämän Service Deskin yleiseen sähköpostiin, josta ne voidaan toimittaa eteenpäin sisältönsä perusteella. Palautelomakkeesta on nähtävillä kuvakaappaus kuviossa 13.



Kuvio 13: Pysäköintisovelluksen palautelomake

5.4 Tietokantarakenne

Pähkinänkuoressa sovelluksen koko toiminnallisuus tiivistyy dataan, joka on tallennettu tietokantaan, ja sen arvojen päivittämiseen ja esittämiseen graafisen käyttöliittymän kautta. Kun parkkipaikka vapautetaan tai varataan, muuttuu tietokannassa paikan tilan arvo, jonka perusteella käyttöliittymässä paikka näkyy joko varattuna tai vapaana. Sovelluksen tietokannassa on taulut paikkojen tiedoille, sovelluksen tapahtumille, tuleville vapautuksille, sovelluksessa julkaistaville ja julkaistuille uutisille sekä ylläpitäville käyttäjille.

Kaikkien pysäköintiruutujen tiedot tallennettiin Parkingslots nimiseen tauluun, joka sisälsi kentät paikan numerolle, tilalle, lisäyspäivälle, tämänhetkiselle käyttäjälle ja hänen tiedoilleen. Lisäksi taulussa oli paikat tiedolle siitä, missä kerroksessa paikka sijaitsee ja kohta, johon voidaan laittaa lisätietoja paikasta tai sen omistajasta.

Pysäköintipaikkojen tietoja sisältävän taulun lisäksi kannasta löytyy Bookings -taulu, jonka rooli on tallentaa paikkojen vapautuksiin liittyvää tietoa. Jotta vapautuksia voidaan tehdä myös tulevaisuuteen, täytyy tiedot vapautuksen alku- ja loppupäivästä olla erillisessä taulussa. Yhdistävänä avainkenttänä Bookings ja Parkingslots tauluilla on Parkingslots -taulun id, joka vastaa Bookings-taulun Parkingslot-kenttää. Bookings –taulussa jokaiselle vapautukselle annetaan myös yksilöidä id-tunniste.

Statistiikan keräämiseksi tietokannassa on Events-taulu, joka kerää kaikki sovelluksen tapahtumat yhteen. Sovellusta kehitettäessä todettiin, että koska sovelluksella on niin pieni kohderyhmä, näin voidaan toimia. Suuremmassa organisaatiossa taulu saattaisi täyttyä liian nopeasti, joka taas voisi olla raskasta palvelimelle ja ylläpidolle. Taulua voi toki tarvittaessa tyhjentää, jos koetaan että siinä on liikaa rivejä. Events-taulun avainkenttänä toimii parkkipaikan numeron sisältävä Parkingslot-kenttä, samalla tavalla kuten Bookings-taulussa.

Sovelluksen ylläpitäjiä hallitsemaan on tietokantarakenteeseen luotu taulu nimeltä Admin. Taulu sisältää yksilöivän tunnisteen lisäksi ainoastaan Admin-käyttäjien käyttäjätunnukset, jotka sovellus lukee sisäänkirjautumisen yhteydessä. Kirjautuessa sovellus ohjaa Admin-käyttäjät ylläpitonäkymään, jota he pääsevät käyttämään, kun taas peruskäyttäjille ylläpitonäkymää ei näy. Jos peruskäyttäjä kuitenkin jostain syystä päätyisi ylläpitonäkymän osoitteeseen, ohjataan hänet automaattisesti sieltä ulos takaisin etusivulle.

Taulukossa 5 esitellään Parkingslot-, Bookings-, Events- ja Admin-taulut.

Parkinslots	
id	INT
Owner	char(250)
State	char(250)
date	DATETIME
User	char(250)
Email	char(250)
Username	char(250)
Floor	char(250)
Adress	char(250)
PostCodeCity	char(250)
Info	char(250)

Bookings	
id	INT AUTOINC.
Parkinslot	int
User	char(250)
Start_date	date
End_date	date

Events	
ID	INT AUTOINC.
User	char(250)
Action	char(250)
Parkingslot	INT
Time	char(250)

Admin	
ID	INT
Käyttäjä	VAR(250)

Taulukko 5: Tietokannan tauluja

Edellä mainittujen taulujen lisäksi tietokannasta löytyy taulu sovelluksessa näkyville uutisille. News-taulu sisältää kentät otsikolle, sisällölle ja julkaisuajalle ja –päivälle. Julkaisun aika- ja päivämäärätiedot lisätään automaattisesti uutisen tallennuksen yhteydessä, jolloin uutisiin liitetään myös uniikki id-arvo.

Tietokantarakenteita luotiin komentorivin avulla ottamalla sovelluksen palvelimeen yhteys PUTTY–sovelluksella. Aikaisemmassa sovelluksessa komentorivi oli ainoa keino hallita tietokantaa, jonka takia sovelluksen uuteen versioon rakennettiin admin-näkymä ja sen toiminnot. Admin-näkymän graafisen käyttöliittymän kautta onnistuu tietokannan sisältöjen hallinta, kuten rivien lisääminen, päivittäminen ja poistaminen.

5.5 Tunnistautuminen

Tunnistautuminen tarkoittaa sitä, että sovelluksen käyttäjä tunnistetaan esimerkiksi käyttäjätunnuksen ja salasanan avulla. Moniin Laurean verkkopalveluihin tunnistaudutaan Shibboleth SSO-palvelun, eli kertakirjautumispalvelun avulla. Shibboleth koostuu kahdesta pääelementistä, jotka ovat Service Provider ja Identity Provider.

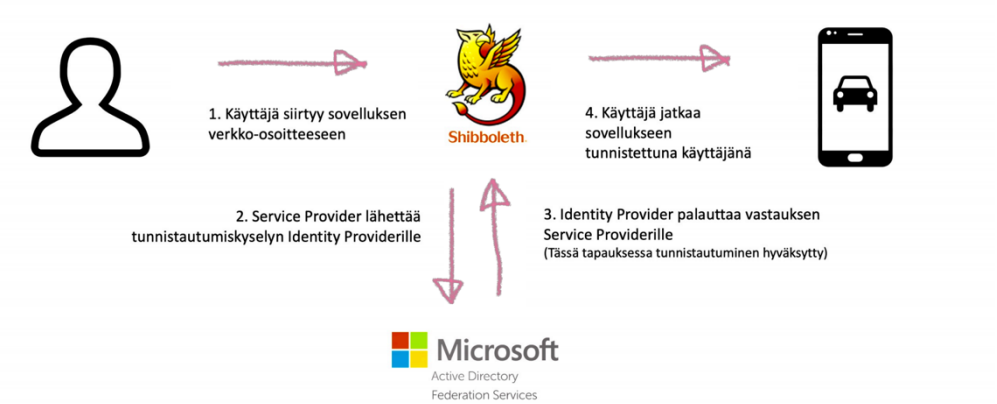
Identity Provider tarjoaa käyttäjien tiedot tunnistautumista varten ja lähettää ne eteenpäin. Sen sijaan, että jokaiseen palveluun olisi omat käyttäjätunnukset ja salasanansa jokaisella käyttäjällä, voidaan ne kaikki yhdistää yhteen palveluun, johon liitetään muut tunnistautumista kaipaavat palvelut.

Shibbolethin oman Identity Providerin sijaan ammattikorkeakoululla on käytössään Microsoftin AD FS, joka on määritelty toimittamaan samaa virkaa. AD FS –lyhenne tulee sanoista Active Directory Federation Services, jolla tarkoitetaan järjestelmää, jolla voidaan hallita organisaatioiden käyttäjien tunnuksia sekä käyttöoikeuksia liitettyihin palveluihin. Käytännössä tämä voi näkyä niin, että samalla salasanalla ja käyttäjätunnuksella kirjaudutaan sisään tietokoneelle, sähköpostiin sekä sisäisiin viestintäkanaviin tai tuntikirjausjärjestelmiin.

Service Provider on sovelluksen kanssa samalla palvelimella sijaitseva ohjelmisto, jonka tehtävä on tarjota ja suojata siihen liitettyä palvelua. Service Provider voi myös luovuttaa Identity Providerin tietoja sovelluksen käyttöön. Näin esimerkiksi sovelluksessa voidaan hyödyntää tietoa siitä, kuka sitä parhaillaan käyttää. Jos halutaan, että Shibbolethin osat luovuttavat tietoja toisilleen, täytyy se erikseen määritellä sekä Service Provideriin, että Identity Provideriin.

Käyttäjän tunnistaminen Shibbolethin kautta tapahtuu, kun käyttäjä avaa verkkopalvelun sivun, jolle tunnistautuminen on määritelty. Sivulta käyttäjä ohjataan kirjautumaan, jolloin Service Provider lähettää kirjautumisen tiedot Identity Providerille tarkastettavaksi. Jos kir-

jautuminen on hyväksytty, pääsee käyttäjä jatkamaan haluamalleen sivulle. Jos kirjautuminen ei mene läpi, antaa kirjautumissivu virheilmoituksen, kuten *”väärä käyttäjätunnus tai salasana.”* Kuviossa 14 on esimerkki, jossa esitetään Shibbolethin toimintaa.



Kuvio 14: Esimerkki Shibbolethin toiminnasta

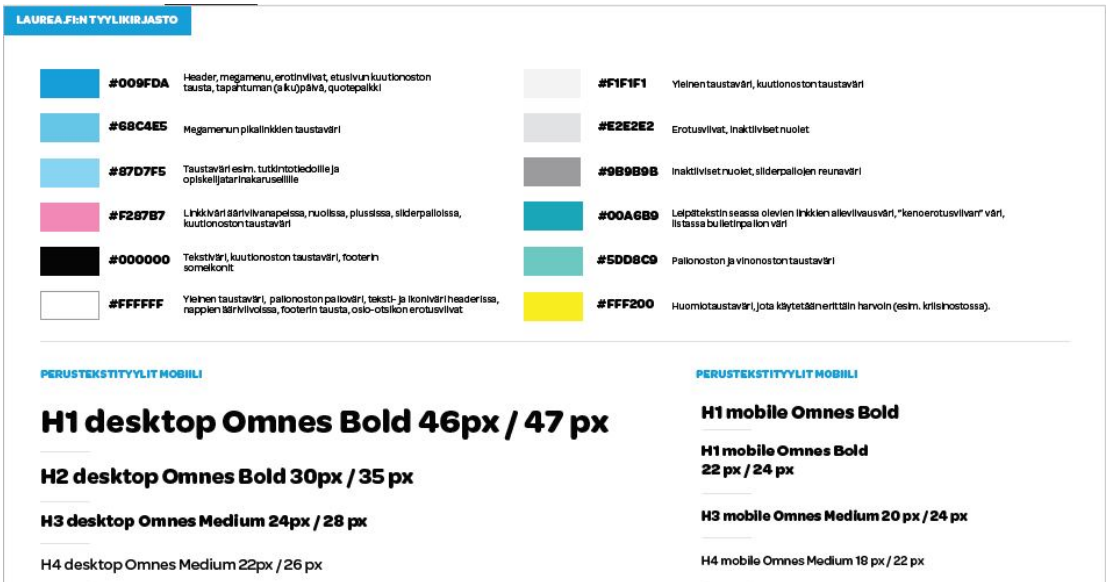
Pysäköintisovelluksessa tunnistautuminen mahdollistaa käyttäjien ja parkkipaikkojen yhdistämisen toisiinsa, joten käyttäjä voi ainoastaan vapauttaa omakseen määriteltyä paikkaansa. Edellä mainitun lisäksi tunnistautumalla rajataan sovelluksen käyttö ainoastaan korkeakoulun henkilökunnalle, eikä esimerkiksi opiskelijoille. Sovelluksen hallinnointimahdollisuuksiin kuitenkin kuuluu toiminto, jolla myös organisaation ulkopuoliselle käyttäjälle voidaan varata vapaaksi merkitty pysäköintipaikka. Tällainen käyttäjä voisi olla esimerkiksi vieraileva luennoitsija.

Kertakirjautumisen kautta määriteltiin, että sovellukseen välittyvät kirjautumisen yhteydessä attribuutit, jotka kertovat kirjautuvan käyttäjän käyttäjätunnuksen, etunimen sekä sukunimen. Laureassa käyttäjätunnustenhallinta on rakennettu niin, että käyttäjätunnus toimii myös sähköpostiosoitteen toisena versiona. Käyttäjätunnusta voidaan siis käyttää myös sähköpostiosoitteena, ja se on määritelty niin, että jokaisen käyttäjän käyttäjätunnus on erilainen.

Kun käyttäjä kirjautuu sovellukseen tällä omalla käyttäjätunnuksellaan, sovellus lukee tietokannasta, onko käyttäjätunnukselle määritelty omaa parkkipaikkaa. Jos tunnuksella on paikka, käyttäjälle näkyy paikan vapautusnäkymä, ja jos tunnukselle ei ole paikkaa, näytetään aloitusnäkymänä varausnäkymä. Kirjautumisen yhteydessä sovellus myös katsoo tietokannan Admin –taulusta, kuuluuko kirjautuneelle käyttäjälle ylläpitäjän oikeudet, vai kohdellaanko häntä peruskäyttäjän roolissa.

5.6 Ulkoasu

Pysäköintisovelluksen ulkoasu pohjautuu Laurealle tehtyyn visuaaliseen ilmeeseen, jonka on suunnitellut korkeakoululle palvelusuunnittelija Jaana Waari keväällä 2017 alkaneen sivustouudistuksen yhteydessä. Sivustouudistuksen yhteydessä julkaistiin päivitetty tyylikirjasto verkkojulkaisujen ulkoasujen yhtenäistämiseksi. Uudistuksen yhteydessä myös korkeakoulun logoa päivitettiin. (Burton 2018) Kuvakaappaus Laurean päivitetystä visuaalisesta ilmeestä löytyy kuviosta 15.

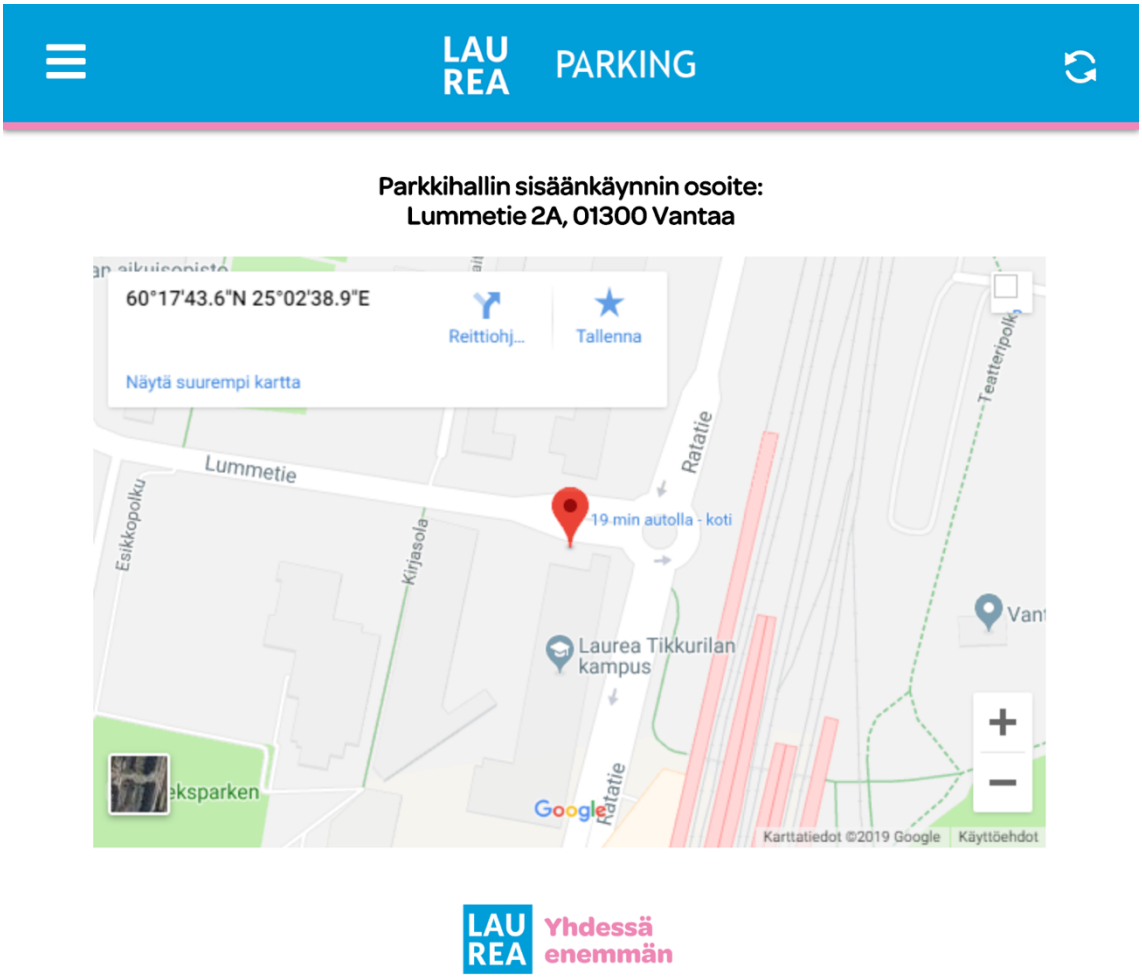


Kuvio 15: Kuvakaappaus Laurean päivitetystä tyylikirjastosta

Pysäköintisovelluksen värimaailman pääväri on ohjeistuksen sävyinen vaalean sininen, jota korostamaan ja raikastamaan on valittu kirjaston vaaleanpunainen. Sovelluksen painikkeissa on käytetty hieman vaaleampaa sävyä pääväristä. Näitä kolmea väriä lukuun ottamatta sovelluksen väriteema on harmaasävyinen, eikä mukaan ole otettu visuaalisen ohjeen lisävärejä, jotka ovat eri sävyjä kahdesta pääväristä. Koska sovellus ei juuri sisällä kuvia, ja siitä haluttiin mahdollisimman selkeä ja helppokäyttöinen, jätettiin kaikki ylimääräinen koristelu tietoisesti pois.

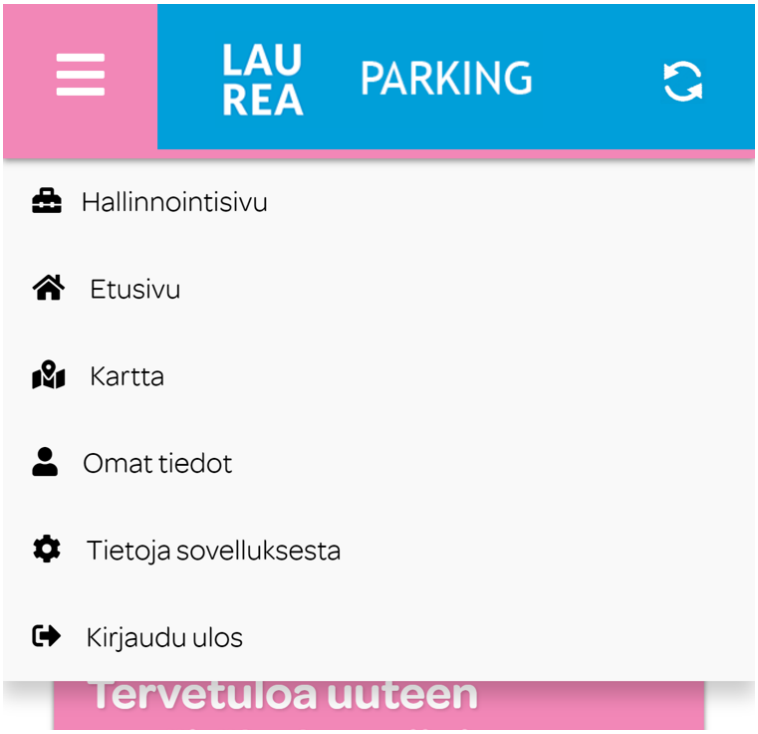
Sovelluksessa käytetty fontti on ohjeistuksen mukainen kirjaisin nimeltä Omnes. Fontista on kolme eri leikkausta, eli paksuutta: regular, medium ja semibold. Pääotsikoissa on käytetty fontin paksuinta leikkausta, painikkeissa ja pienemmissä otsikoissa keskikokoista, ja leipätekstissä normaalia leikkausta. Fonttien koot eivät ole täsmälleen samat kuin ohjeistuksessa, jotta esimerkiksi mobiilinäkymässä otsikot ja sisältö mahtuvat samaan näkymään. Lisäksi sovelluksessa ei ole käytössä yhtä montaa eri otsikkokokoa, sillä siinä ei ole painikkeiden ja otsikoiden lisäksi kovin paljon tekstisisältöä.

Sovelluksen karttanäkymän kartassa on esillä värejä päävärien ulkopuolelta. Koska kartta tulee suoraan Google Maps -palvelusta, tulee värimäärittely heidän valmiista kartoistaan. Kartan värit ovat kuitenkin informatiiviset, selkeät ja sopivat sovelluksen muihin väreihin, joten vaihtoehtoisia toteutuksia kartan esittämiseen ei tarvittu. Kuviossa 16 on esillä kuvakaappaus sovelluksen karttanäkymästä tabletilla käytettynä.



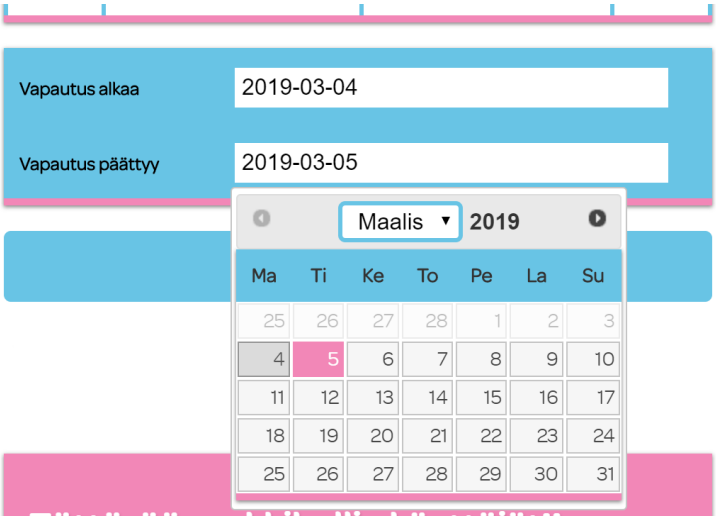
Kuvio 16: Karttanäkymä tablettitietokoneella

Käyttöliittymässä on mukana pieniä ikoneita selkeyttämään linkkien takaa löytyviä toiminnallisuuksia. Ikonit ovat FontAwesome -palvelusta. FontAwesome on avoimen lähdekoodin palvelu, joka tarjoaa kattavan valikoiman erilaisia ikoneita joko fontti-, tai vektorigrafiikkana. Perusvalikoima ikoneista on ilmainen käyttää missä vain projektissa, ja halutessaan lisää voi ostaa käyttöönsä vielä laajemman valikoiman erilaisia ikoneita ja versioita ilmaisista ikoneista. Pysäköintisovelluksen käyttöön riittivät ilmaisessa tarjonnassa olevat ikonit. Ikoneita sijoitettiin pääasiassa valikkonäkymän vasempaan reunaan, jokaiselle linkille omansa. Ikoneita valitessa yritettiin löytää mahdollisimman hyvin linkin toimintoa kuvaava ikoni. Kaikki valikkonäkymän eri ikonit ovat nähtävissä kuviossa 17.



Kuvio 17: Valikon FontAwesome -ikonit

Sovelluksen vapautusnäkymässä olevassa kalenterissa on hyödynnetty jQuery -kirjastoa, sekä jQuery UI -kirjaston valmiiksi rakennettua Datepicker -kalenteria ja päivämääränvalintaa. jQuery UI on jQuery-kirjaston päälle rakennettu paketti, joka sisältää valmiiksi määriteltyjä CSS-komentoja määrittelemään käytettyjen elementtien visuaalisuutta, kuten kokoa ja väriä. Valmiiksi määriteltyä visuaalisuutta voi kuitenkin muokata, joten siitä tehtiin sovelluksen yleisilmeeseen paremmin sopiva vaihtamalla väreiksi muualla sovelluksessa käytettyjä värejä. Tällaisia olivat kuviosta 18 havaittavissa olevat valitun päivän pinkki taustaväri, sekä viikonpäivien taustalla näkyvä sininen.



Kuvio 18: Kalenterinäkymä

5.7 Sovelluksen testaus

Kehitysvaiheessa sovellus sijaitsi projektia varten pystytetyllä testipalvelimella, jolle oli pääsy ainoastaan Laurean Tikkurilan kampuksen sisäisellä verkkoyhteydellä. Ennen sovelluksen julkaisua se siirrettiin testipalvelimelta tuotantopalvelimelle alakansioon niin, että sitä pystytään käyttämään sisäisen verkon ulkopuolelta. Testausta varten valittiin joukko mahdollisia käyttäjiä, joita pyydettiin suorittamaan ennalta määriteltyjä tehtäviä, kuten paikan varaamista ja sovelluksen kielen vaihtamista.

Sovelluksen toiminallisuutta testattiin pilotilla, jossa valitut testikäyttäjät varasivat ja vapauttivat heille määriteltyjä paikkoja, ja testasivat toimintoja käytännössä. Sovellusta testattiin myös erilaisilla laitteilla ja selaimilla, jotta voidaan vahvistaa yhteensopivuus. Sovelluksen kaikki tekstit oikoluettiin kirjoitusvirheiden löytämiseksi ja korjaamiseksi, sekä kaikki linkit ja painikkeet testattiin, että ne vievät oikeaan paikkaan.

Testeissä ilmeni joitain korjauksen kohteita. Yksi näistä ongelmista oli vapauttaessa parkkipaikkaa mobiililaitteella. Kun vapautetaan paikkaa ja valitaan kalenterista vapautukselle alku- ja loppupäivä, ilmestyi mobiililaitteen kosketusnäytölle näppäimistönäkymä. Näkymä saattaa olla hämmentävä, sillä kyseiseen kohtaan ei kuulu kirjoittaa mitään, vaan ainoastaan valita kalenterista haluttu päivämäärä. Näppäimistö mahdollisti myös yhdellä selaimella käytettäessä kenttään kirjoittamisen, mikä ei ollut toivottavaa, vaikka kirjoitettu teksti ei mennytkään vapautusta vahvistaessa läpi. Ongelma korjattiin asettamalla tekstikentälle ReadOnly -arvo, jolloin mobiililaitteiden selaimet eivät ehdota automaattisesti näppäimistöä. Lomakkeen ReadOnly -arvo tarkoittaa sitä, että kenttää pystytään ainoastaan lukemaan eikä siihen voi kirjoittaa.

Myös kalenterinäkymään tehtiin hieman korjauksia ja suljettiin pois esimerkiksi mahdollisuus tehdä merkintöjä menneisiin päivämääriin. Kalenterin tyylejä muutettiin myös hieman selkeämmiksi, jotta menneet päivät erottuvat valittavista päivistä.

5.8 Refaktorointi ja dokumentointi

Ennen julkaisua sovelluksen tietokantaan vietiin parkkipaikkojen tietoihin niiden omistajat, sekä omistajien perustietoja, kuten sähköposti. Tiedot saatiin ammattikorkeakoulun korkeakouluisännältä. Sovelluksen hallinnointinäkymässä on mahdollista muokata näitä syötettyjä tietoja, esimerkiksi parkkipaikan omistajan vaihtuessa.

Kirjoitettu koodi tarkastettiin ja siitä poistettiin tarpeettomia, ulos kommentoituja osia. Koodi myös formatoitiin siistimmäksi ja helpommin luettavaksi. Ohjelmistokoodin tulee olla mahdollisimman selkeää ja kommentoitua, jos siihen tulevaisuudessa halutaan tehdä muutoksia. Tärkeimpiä kohtia koodista nostettiin kommenttien avulla vielä helpommin löydettäviksi mahdollisia muutoksia varten.

Dokumentointia tehtiin kolmeen eri muotoon. Muotoja olivat:

- Excel-taulukkotiedosto, joka toimi projektin päivittäisenä dokumentointityökaluna.
- Tekstidokumentti, joka sisälsi saman kuin taulukkotiedosto, mutta helpommin haettavassa ja siistimmässä muodossa.
- Laurean käyttämässä eduuni-wiki –palvelussa sijaitseva tiedosto, jota koko tietohallinto pystyy tarkastelemaan ja muokkaamaan.

Taulukkotiedoston vahvuutena oli ketteryys, tekstitiedoston järjestelmällisyys ja eduuni-wikin saavutettavuus. Taulukkotiedoston toimiessa ikään kuin projektin backlogina, kaikkea, mitä taulukkotiedostoon dokumentoitiin, ei julkaistu muihin dokumentteihin. Näin kävi esimerkiksi tilanteissa, joissa dokumentoinnin kohteeseen tuli muutoksia tai se jäi kokonaan pois sovelluksesta. Tekstidokumentteihin sisällytettiin ainoastaan oleellisimmat ja ajankohtaisimmat tiedot ja dokumentaatio.

6 Tulokset ja löydökset

Tuloksena opinnäytetyöstä syntyi uusi versio pysäköintisovelluksesta, sekä käyttöohje ylläpitävän käyttäjän näkymästä ja toiminnoista. Ohjeistuksessa kerrotaan yksityiskohtaisesti mitä kukin toiminto tekee, mistä ne voi löytää ja missä tilanteissa niitä käytetään. Ohjeistus julkaistiin Laurean käyttämään eduuni-wiki –palveluun, jonne löytyy linkki hallinnointinäkymästä. Sama ohjeistus myös tulostettiin aulan isäntien käyttöön ja laitettiin PDF–muodossa sovelluksen hallinnointinäkymän kautta luettavaksi.

Ohjeistuksen ja itse sovelluksen lisäksi työn ohessa tuotettiin kattavaa dokumentaatiota sovelluksen lähdekoodista sekä kertakirjautumisjärjestelmästä ja sen käyttöönotosta. Sovelluksen mahdollista jatkokehitystä silmällä pitäen dokumentaatioon avattiin koodin toiminnallisuutta ja rakennetta, sekä kerrottiin tarkasti mistä mikäkin toiminnallisuus löytyy. Jokainen tiedosto ja sen tehtävä avattiin omaan taulukkoonsa, tietokanta ja sen rakenne kerrottiin omassaan sekä tärkeitä toiminnollisuuksia, kuten kaksikielinen käyttöliittymä ja sen toteutus, nostettiin vielä erikseen esille.

6.1 Sovelluksen julkaisu

Sovelluksen paranneltu, uusi versio julkaistiin koko korkeakoulun henkilökunnan käyttöön 20. päivä helmikuuta 2019. Sovelluksen tietokantaan tehtiin yhteys Grafana-palveluun, jolla pystytään seuraamaan sovelluksen käyttöä komentokeskus-tyylisellä infotaululla. Infotaululle määriteltiin ympyrädiagrammeja kuvaamaan käyttötapausten jakautumista ja vapaiden ja varattujen paikkojen suhdetta. Lisäksi näkymässä on aikajana, josta nähdään mihin aikaan käyttötapaukset ovat kirjattu. Näkymästä pystytään myös havainnoimaan ketkä käyttäjät käyttävät sovellusta aktiivisimmin.

Julkaisun yhteydessä sovellusta mainostettiin käyttäjille intratiedotteen muodossa. Julkaisuissa muistutettiin sovelluksen olemassaolosta ja kannustettiin käyttämään sitä. Vaikka julkaisuviikko sattui olemaan pääkaupunkiseudun koulujen hiihtolomaviikko, oli julkaisupäivänä ja -viikkona sovelluksella jo enemmän käyttäjiä, kuin edellisellä versiolla koko kuussa. Julkaisusta seuraava viikko oli vieläkin menestyksekkäämpi, sen näkymästä on nähtävillä kuvakaappaus kuviossa 19.

Kuvio 19: Grafana -näkymä sovelluksen julkaisua seuranneesta viikosta

Käyttöasteen nostamiseksi sovellukselle järjestettiin pari viikkoa julkaisun jälkeen kahden viikon mittainen kampanja, ja sen yhteyteen arvonta. Arvonnassa kaikki sovellusta käyttäneet tai palautetta jättäneet osallistuivat elokuvalippujen arvontaan. Kampanjan toteuttivat yhteistyössä korkeakoulun markkinointi ja tietohallinto. Kampanjaa mainostettiin levittämällä julisteita henkilökunnan tiloihin eri kampuksilla, sekä tiedottamalla Intran kautta. Sovelluksen uutisiin lisättiin myös tiedote kampanjasta.

6.2 Jatkokehitys

Nykymallissaan sovellus ja oikeastaan koko konsepti sisältää hieman problematiikkaa. Palautteiden perusteella monet käyttäjät kokevat, että jos he omistavat parkkipaikan, ja maksavat siitä kuukausittain, miksi he vapauttaisivat sen, että joku muu voisi käyttää sitä ilmaiseksi? Käyttäjiä laskutetaan paikoista kuukausittain, ja vaikka summa ei ole mikään kovin suuri, he silti maksavat siitä, että heillä on varattu pysäköintiruutu.

Jatkokehitysideana koko sovelluksen toimintamallia ja konseptia voisi kehittää. Sen sijaan, että käyttäjät vapauttaisivat omistamiaan paikkoja, voisi paikat alun perin olla vapaita, ja niitä varattaisiin sovelluksen kautta. Tällöin kukaan ei varsinaisesti omistaisi yhtään parkkipaikkaa, vaan kuka vain voisi varata paikan, kun sitä tarvitsee.

Toisaalta, jos joka aamu pitäisi kiireessä varata itselleen paikka, voisi sekin olla käyttäjille hankalaa. Ongelmallista olisi myös se, että jos käyttäjän työpäivä alkaakin vasta myöhemmin, saattavat kaikki paikat olla jo varattuja. Osalla ammattikorkeakoulun työntekijöistä on pitkä työmatka, jonka he kulkevat lähes aina autolla. Tällaisille käyttäjillä, jotka ovat aina samalla kampuksella ja tarvitsevat paikkaa päivittäin, kuukausittainen laskutus ja paikan omistaminen sopivat paremmin.

Teknisesti nykymallissaan olevaan sovellukseen voisi rakentaa mahdollisuuden tehdä varauksia myös muillekin, kuin kuluvalle päivälle. Etenkin seuraavaksi päiväksi varaaminen olisi hyödyllinen ominaisuus. Mahdollisuus varata paikkoja pidemmältä aikaväliltä saattaisi aiheuttaa myös paikkojen hamstraamista; käyttäjät, jotka saattavat tulla kampukselle seuraavalla viikolla voisivat varata paikan, vaikka eivät välttämättä sitä tarvitsisi. Pitkän aikavälin varaaminen tarkoittaisi myös sitä, että jos vapauttajan suunnitelmat muuttuvatkin, hän ei voisi enää peruuttaa vapautustaan, jos se on ehditty jo varata.

7 Yhteenveto ja johtopäätökset

Pysäköintisovelluksesta julkaistiin tuotantoon uusi monipuolisempi ja käyttäjäystävällisempi versio Laurea-ammattikorkeakoulun henkilökunnan käyttöön. Sovellukseen tunnistaudutaan kertakirjautumisjärjestelmän kautta, jonka ohessa käyttäjien tiedot lähetettiin sovelluksen käyttöön. Koska käyttäjät ovat tunnistettuja, käyttöliittymästä voitiin selkeyttää ylimääräi-

set, muiden käyttäjien, parkkipaikkojen numerot ja tiedot pois. Paikan voi vapauttaa tai varata vain kahdella klikkauksella. Sovelluksen käyttö on siis nopeaa ja kynnys käyttää sitä pienempi.

Sovelluksen ylläpidon helpottamiseksi se optimoitiin selainpohjaiseksi, ja sille luotiin erillinen graafinen ylläpitäjän näkymä. Ylläpitonäkymästä onnistuu parkkipaikkojen tietojen muokkaaminen ja päivittäminen, uutisten lisääminen ja poisto sekä kaikkien paikkojen varaaminen ja vapauttaminen.

Sovelluksen käyttö ja olemassaolo näkyvät loppukäyttäjille usealla tavalla. Mitä enemmän paikkojaan vapauttavia käyttäjiä sovelluksella on, sitä enemmän varattavia paikkoja on tarjolla. Vaikka kaikille halukkaille ei olisikaan sovelluksessa käyttöhetkellä varattavia paikkoja, näkyy sovelluksen olemassaolo vieraspaikoissa. Mitä enemmän sovelluksen kautta varataan paikkoja, sitä enemmän vieraspaikkoja jää vapaaksi muille käyttäjille. Näin ollen siis pysäköintitiloja optimoidaan paremmin, ja parkkihallin yleinen käyttöaste nousee.

Kokonaisuutena sovelluskehitysprojekti eteni tahdikkaasti eteenpäin koko projektin ajan. Kaikki eteen tulleet ongelmat saatiin ratkaistua pikaisesti, eikä projektissa tullut viivästyksiä. Loppukäyttäjät ottivat uuden version hyvin vastaan, ja sovelluksen käyttöaste lisääntyi heti julkaisun yhteydessä ja jatkui tasaisena sen jälkeenkin.

8 Oman oppimisen arviointi

Projektin edetessä opin paljon asioita. Projekti eteni osaltani hyvin itseohjautuvasti, sillä tein kehitystyötä pääasiassa yksin ja sain itse määrittää aikataulutukseni sekä jaotella työkuormani työpäivien kesken. Itsenäisesti työskennellessä täytyy olla hyvät taidot itsensä johtamiseen, ja uskallusta kysyä apua, jos sitä tarvitsee. Opin, että itselleni sopi työtapa, jossa asetetaan sopivan kokoisia tavoitteita, esimerkiksi viikko tai päivätasolla, ja lähdetään tavoittelemaan niiden täyttämistä. Näin kokonaisuutena suuri ohjelmisto ja sen rakenne pysyivät helpommin hallittavissa ja hahmotettavissa.

Pääsin työskentelemään ohjelmistokehitysprosessin useissa eri rooleissa, mikä lisäsi ymmärrystäni siitä, miksi yleensä projekteissa on mukana omat ammattilaisensa esimerkiksi määrittelyyn, suunnitteluun, kehitykseen ja testaukseen. Ymmärsin myös miksi projekteilla yleensä on kehitysmenetelmiä. Yksin työskennellessä projekti pysyi hyvin kasassa, mutta jos tekijöitä olisi ollut enemmän, oltaisiin ehdottomasti tarvittu jotain menetelmää apuna hallinnoimaan projektia kokonaisuutena.

Opin myös miten tärkeää on hyvä dokumentointi. Varsinaisten erillisten dokumentointien lisäksi kommentoin aina kirjoittamani koodin, jotta osasin palata siihen itse myöhemmin. Samalla kommentointi mahdollistaa myös sen, että joku muu voi tarvittaessa helpommin viilata

tekemääni työtä myöhemmässä vaiheessa. Sovelluksen vanhaa versiota pystyttäessä testipalvelimelle kattavammasta dokumentaatiosta olisi ollut myös apua.

Sovellusten kehittämisessä sain lisää varmuutta tekemiseeni. Opintojen yhteydessä tehdyissä projekteissa ei tarvitse ottaa niin montaa asiaa huomioon, kuin aidoissa työelämän töissä, joissa on mukana oikea toimeksiantaja ja tarkoituksena julkaisu tietylle kohderyhmälle. Tämä projekti kuitenkin todisti sen, että pystyn tällaisen projektin toteuttamaan. Olen todella kiitollinen, että tällainen mahdollisuus tuli eteeni, ja että uskalsin siihen tarttua. Toimeksiantajan ja käyttäjien palautteiden perusteella onnistuin projektissa hyvin, ja täytyy myöntää, että olen itsekin ihan tyytyväinen lopputulokseen.

Lähteet

Painetut

Beaird, J. & George J. (2014). The Principles of Beautiful Web Design. USA: SitePoint.

Beighley, L. & Morrison, M. (2009). Head first PHP & MySQL. Sebastopol (CA): O'Reilly.

Darmawikarta, D. (2014). SQL for MySQL: A beginner's tutorial. Montreal: Brainy Software.

Duckett, J. (2011). HTML and CSS: Design and build websites (1st ed.). Hoboken: Wiley.

Hong, P. (2018). Practical web design: Learn the fundamentals of web design with HTML5, CSS3, bootstrap, jQuery, and vue.js. Birmingham: Packt Publishing.

Kasurinen, J. P. (2013). Ohjelmistotestauksen käsikirja. Jyväskylä: Docendo.

Sähköiset

Burton, M. (2018). Laurean uudet verkkosivut avattu. Viitattu 23.1.2019. https://www.laurea.fi/ajankohtaista/uutiset/uudet-verkkosivut/

Hiltunen, L. (2009). Validiteetti ja reliabiliteetti. Jyväskylän Yliopisto. Viitattu 20.4.2019. http://www.mit.jyu.fi/OPE/kurssit/Graduryhma/PDFt/validius_ja_reliabiliteetti.pdf

Järvenpää, E. (2006). Laadullinen tutkimus. Viitattu 11.3.2019. http://www.cs.tut.fi/~ihtesem/k2007/materiaali/luento4.pdf

Martin. (2017). Introducing WinSCP. Viitattu 4.3.2019. https://winscp.net/eng/docs/introduction

Routio, P. (2007). Tuote ja tieto: Kyselevät tutkimustavat. Viitattu 11.3.2019. http://www2.uiah.fi/projects/metodi/064.htm

Schwaber, K., & Sutherland, J. (2017). Scrum-opas, scrumin määritelmä ja pelisäännöt. Viitattu 20.4.2019. https://scrumwell.files.wordpress.com/2018/03/2017-scrum-guide-fi-v1-02.pdf

Tatham, S. (2019). PuTTY: A free SSH and telnet client. Viitattu 4.3.2019. https://www.chiark.greenend.org.uk/~sgtatham/putty/

Youssef, N. (2019). Top 10 alternatives to PHP in 2019. Viitattu 19.2.2019. https://hackr.io/blog/top-10-alternatives-to-php-in-2019

Kuviot

Taulukot