Haaga-Helia
ammattikorkeakoulu Oy

Autopaikkojen vuokrasopimusten tulostusjärjestelmä

Mira Louhe

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2020

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä

3.4.2020

Tekijä(t)
Mira Louhe

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Autopaikkojen vuokrasopimusten tulostusjärjestelmä

Sivu- ja liitesivumäärä
24+9

Opinnäytetyön otsikko englanniksi
System for Printing Rental Agreements of Parking Spaces

Tämä on produktityyppinen opinnäytetyö, jossa lopputuotoksena on MVP-ajattelumallia noudattaen tehty sovellus, josta voidaan tulostaa autopaikkojen vuokrasopimuksia. MVP eli Minimun Viable Product tarkoittaa hyvin kevyttä ja minimalistista järjestelmää, jonka avulla saadaan selvitettyä tehokkaasti sekä mahdollisimman pienellä työn määrällä ratkaiseeko järjestelmä tarvittavan ongelman. Järjestelmä on rajattu siten, että siihen ei tallenneta vuokrasopimusten, työntekijöiden ja asiakkaiden tietoja.

Opinnäytetyössä esitellään web-ohjelmoinnin perusteet, johon sisältyvät Internet-sivujen tekeminen sekä erilaiset ohjelmointikielet ja tietoturvaan liittyvät perusasiat. Lisäksi opinnäytetyössä on selvitetty, miten lopputuotoksena tehty sovellus on toteutettu.

Opinnäytetyön tavoitteena on ollut selvittää, onko olemassa keinoa, jonka avulla saadaan ylläpidettyä autopaikkojen vuokrasopimusten yhdenmukaisuutta sekä vuokrasopimusten ehtojen ajantasaisuutta reaaliajassa kaikkien tarvittavien henkilöiden ulottuvilla.

Asiasanat
Web-ohjelmointi, järjestelmäsuunnittelu, web-sovelluskehitys

Sisällys

1 Johdanto

Tässä opinnäytetyössä kehitetään selainpohjainen sovellus autopaikkojen vuokrasopimusten tulostamista varten. Sovellus on tarkoitettu käytettäväksi kiinteistöalan yrityksessä, joka vuokraa asuntoja sekä autopaikkoja vuokralaisille.

Mahdollisimman pitkälle kehitetyt resurssit sekä muut liiketoiminnan kannattavuutta lisäävät tekijät aikaansaavat liiketoiminnalle tärkeän tuloksen kasvun. Kiinteistöalan palvelun-tarjoajalle on tärkeää tuntea tarjoamansa palvelut hyvin, ja pitää esimerkiksi autopaikkojen tai asuntojen vuokrauksessa käyttöaste mahdollisimman korkeana. Autopaikkoja saattaa vuokrata useampi erillinen toimija, joita ovat tässä opinnäytetyössä huoltoyhtiöt ja asuntovuokrauksen työntekijät. Eri toimijoilla on käytössään erilaiset vuokrasopimuspohjat ja vuokrasopimusten ehdot voivat vaihdella paljon, joten tässä opinnäytetyössä ratkaistavaksi ongelmaksi on nostettu vuokrasopimusten samankaltaisuus sekä sopimusehtojen päivittäminen siten, että ne ovat reaaliajassa kaikkien toimijoiden saatavilla.

Opinnäytetyön keskeisimpänä tavoitteena on tehdä kevyt järjestelmä autopaikkojen vuokrasopimusten tulostamiseen. Järjestelmään tallennetaan eri taloyhtiöiden osoitteet, erityisehdot sekä lisätiedot. Järjestelmään tehdään yksi yleinen käyttäjätunnus sekä salasana, joilla käyttäjät voivat tunnistautua palveluun. Järjestelmässä on yhdet yleiset sopimusehdot, joita voidaan tarvittaessa päivittää käyttäjien toimesta.

Järjestelmä on rajattu siten, että tietokantaan ei tallenneta vuokrasopimuksia eikä autopaikkoja vuokraavien asiakkaiden tietoja. Järjestelmä toteutetaan MVP-ajattelumallia (Minimum Viable Product) hyödyntämällä, joka tarkoittaa hyvin kevyttä ja minimalistista järjestelmää, jonka avulla saadaan selvitettyä tehokkaasti sekä mahdollisimman pienellä työn määrällä ratkaiseeko järjestelmä tarvittavan ongelman (Haapahovi 12.3.2017).

Tuotoksesta hyötyvät autopaikkoja vuokraavat työntekijät sekä huoltoyhtiössä että asuntovuokrauksessa, koska järjestelmän avulla saadaan tulostettua yhdenmukainen autopaikan vuokrasopimus. Lisäksi päivitetyt ehdot ovat kaikkien toimijoiden saatavilla reaaliajassa, ja ehtojen päivitys on helppoa järjestelmän avulla. MVP-ajattelumallin produkti mahdollistaa hyvin yksinkertaisen tietokannan sekä käyttöliittymän tekemisen, joten opinnäytetyössä kirjoitettavan ohjelmakoodin sekä tietoperustan avulla lukija saa hyvän johdatuksen järjestelmäkehitykseen, vaikka lukijoilla ei olisi aiempaa ohjelmointikokemusta.

Opinnäytetyön keskeisimpiä käsitteitä ovat web-ohjelmointi, järjestelmäsuunnittelu ja web-sovelluskehitys. Tärkeimmät käsitteet ovat web-ohjelmointi ja -sovelluskehitys, koska tar-

koituksena on toteuttaa selaimessa toimiva verkkosovellus, jota voivat käyttää fyysisesti eri paikoissa sijaitsevat toimijat siten, että järjestelmän tiedot ovat kaikkien käyttäjien saatavilla reaaliajassa. Web-sovellus tarkoittaa erilaisten ohjelmointi-, merkintä- ja tyylikielien avulla luotua käyttöliittymää, joka toimii tietokannan kanssa yhdessä tarjoten tietyn palvelun käyttäjälle (Gibb 2016).

Järjestelmäsuunnittelu käsitteenä kuuluu oleellisena osana opinnäytetyöhön, koska työssä toteutetaan uusi järjestelmä. Järjestelmäsuunnittelu tarkoittaa sitä prosessia, jonka avulla suunnitellaan järjestelmän rakennetta sekä elementtejä, joita ovat mm. järjestelmäarkkitehtuuri, erilaiset moduulit tai muut komponentit, rajapinnat sekä data, joka kulkee koko järjestelmän kautta (Didacus 2018).

2 Web-ohjelmoinnin perusteet

Web-ohjelmoinnilla tarkoitetaan selaimella suoritettavan sovelluksen tekemistä käyttäen apuna jotain ohjelmointikieltä. Sovelluskehityksessä käytettäviä ohjelmointikieliä on paljon ja niillä kaikilla on erilaisia ominaisuuksia. Ohjelmointikieliä valittaessa tulee ottaa huomioon sovelluksen ominaisuudet, ja valita käytettävät kielet siten, että niillä saadaan koodattua sovellukseen tarvittavat ominaisuudet. Lisäksi ohjelmoijan omat mieltymykset ovat suuressa roolissa ohjelmointikieliä valittaessa (Wodehouse a.) Web-ohjelmoinnissa front end -ohjelmointi tarkoittaa visuaalista, käyttäjille konkreettisesti näkyvää osuutta ja back end -ohjelmointi taas esimerkiksi tietokantojen luomista eli niin sanotusti käyttäjille näkymätöntä puolta ohjelmoinnista (Stewart 14.1.2019).

2.1 Internet-sivuston tekeminen

Internet-sivut voivat sisältää hyvinkin monipuolista toiminnallisuutta. Sivujen tekemisessä voidaan käyttää esimerkiksi erilaisten lomakkeiden luonnissa ohjelmointikieliä kuten PHP:tä ja JavaScriptiä. Internet-sivut ovat yleensä myös selaimella suoritettavan sovelluksen käyttöliittymä (Purely Branded).

Web-kehityksen opiskelu aloitetaan yleensä harjoittelemalla käyttämään HTML-kieltä. HTML eli Hypertext Markup Language ei ole varsinaisesti ohjelmointikieli, vaan merkintäkieli, jonka perusteella selain esittää Internet-sivut käyttäjälle. HTML kirjoitetaan tekstieditorissa, ja tässä voidaan käyttää editorina esim. Notepad++ -ohjelmaa (kuva 1) (W3Schools a).

```
<!DOCTYPE HTML>
    <head>
        <title>Otsikko näkyy välilehden nimenä</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    </head>
    <body style="background-color:black">
        <h1 style="color:white">Hello World!</h1>
    </body>
</html>
```

Kuva 1. Notepad++ -tekstieditori, johon on kirjoitettu yksinkertainen HTML-sivu.

Kun HTML-koodi on kirjoitettu tekstieditoriin, sitä voidaan testata avaamalla tiedosto verkkoselaimella (kuva 2).

Kuva 2. Verkkoselain muodostaa näytölle sisällön HTML-koodin rakenteen ja muotoilujen mukaisesti.

CSS (Cascading Style Sheets) on tyylikieli, jolla voidaan määrittää erilaisia muotoiluja sivuston elementeille (kuva 3). CSS-määritykset ovat yleensä omassa .css-päätteisessä tiedostossaan, mutta CSS voidaan kirjoittaa tarvittaessa kevyiden, hyvin suppeiden sivujen ollessa kyseessä myös HTML:n sekaan, kuten kuvassa 1. on tehty. CSS-tiedostossa käytettävät käsitteet ovat muuten melko samanlaiset kuin HTML:ssäkin, mutta muotoilut voidaan kohdistaa joko kehittäjän itse määrittelemiin luokkiin tai suoraan esimerkiksi kaikkiin otsikoihin (W3Schools b.)

```
1   body {
2     background-color: black;
3   }
4
5   h1 {
6     color: white;
7     text-align: center;
8   }
9
10  p {
11    font-family: verdana;
12    font-size: 20px;
13  }
```

Kuva 3. Esimerkki CSS-muotoilusta.

Tyylitiedostoja on tehty myös valmiiksi ns. kehyksiksi, joihin on valmiiksi luotu erilaisia asettelumalleja, käyttöliittymäelementtejä sekä mm. gallerioiden tyylejä, jotka ovat vapaasti yhdisteltävissä. Tällaiset kehykset nopeuttavat sovelluskehitystä, sillä Internet-sivujen suunnittelijan ei tarvitse itse koodata kaikkea, vaan tämä voi käyttää suoraan valmiita komponentteja ja asettelumalleja. Näitä tyylikehyksiä ovat mm. Bootstrap, Foundation, Bulma CSS Framework ja Semantic UI. Kaikilla tyylikehyksillä on omat, hieman erilaiset ominaisuutensa, ja kehykset valitaan Internet-sivuille vaaditun tyylin sekä tarvittavan sivustopohjaratkaisun mukaan (Kim 2015).

2.2 Ohjelmoinnin perusteet

Web-ohjelmointi käsittää sekä selainohjelmoinnin että palvelinohjelmoinnin. Ohjelmakoodi suoritetaan joko asiakkaan selaimella tai palvelimella. Web-ohjelmointiin sisältyy palvelimella suoritettava koodi, tietokantarajapinta sekä käyttäjälle näkyvä käyttöliittymä, johon koodataan valitulla ohjelmointikielellä erilaisia elementtejä, kuten esimerkiksi lomakkeita (Wodehouse b.)

PHP (kuva 4.) on yksi yleisimmistä web-ohjelmoinnissa käytetyistä ohjelmointikielistä. Itse koodi voidaan kirjoittaa joko suoraan HTML:n yläpuolelle, HTML:n sekaan tai eri tiedostoon kokonaan. PHP-koodilla voidaan tuottaa eli generoida HTML-merkkikieltä silloin, kun tuotetaan esimerkiksi lista silmukalla tietokannan sisällön perusteella. Käytännössä PHP-esikääntäjä suorittaa koodin palvelimella, kun käyttäjä lataa sivun. Käyttäjä näkee vain tulosteen (output), mutta ei itse koodia. PHP-koodia ei voi testata avaamalla tiedosto suoraan omalla selaimella, vaan se tulee ladata ensin palvelimelle ja suorittaa koodi suoraan palvelimelta käsin (php.net a). Tunnetuimpia PHP:llä toteutettuja sovelluksia ovat mm. Facebook ja WordPress (Kim 2015).

```php
<?php
    if (isset($_POST["submit"])) {
        $name = $_POST['name'];
        $email = $_POST['email'];
        $message = $_POST['message'];
        $human = intval($_POST['human']);
        $from = 'Yhteydenotto';
        $to = 'mira.louhe@louhedesign.com';
        $subject = 'Yhteydenottopyyntö Louhe Design ';

        $body ="From: $name\n E-Mail: $email\n Message:\n $message";
        // Check if name has been entered
        if (!$_POST['name']) {
            $errName = 'Kirjoita nimesi!';
        }

        // Check if email has been entered and is valid
        if (!$_POST['email'] || !filter_var($_POST['email'], FILTER_VALIDATE_EMAIL)) {
            $errEmail = 'Tarkista sähköpostiosoite!';
        }

        //Check if message has been entered
        if (!$_POST['message']) {
            $errMessage = 'Kirjoita viestikenttään lyhyt viesti!';
        }
        //Check if simple anti-bot test is correct
        if ($human !== 5) {
            $errHuman = 'Tarkista vastaus';
        }
// If there are no errors, send the email
if (!$errName && !$errEmail && !$errMessage && !$errHuman) {
    if (mail ($to, $subject, $body, $from)) {
        $result='<div class="alert alert-success">Thank You! I will be in touch</div>';
    } else {
        $result='<div class="alert alert-danger">Pahoittelut, viestin lähetyksessä oli ongelma. Kokeile myöhemmin uudestaan, kiitos!</div>';
    }
}
    }
?>
```

Kuva 4. Yllä esimerkki PHP:llä koodatusta yhteydenottolomakkeen käsittelijästä.

Muita palvelimella suoritettavia ohjelmointikieliä ovat mm. Java, Python ja Ruby sekä Node.js ympäristö, joka sallii JavaScript-koodin suorittamisen palvelimella. Tällöin sekä selain- että palvelinohjelmointi voidaan tehdä samalla kielellä (Crampete 2020).

JavaScript on ohjelmointikieli, jonka avulla voidaan ohjata myös selaimen toimintaa. Koodin voi suorittaa avaamalla tiedosto suoraan selaimella. Tästä syystä koodin testaaminen on suoraviivaisempaa, kuin palvelimella suoritettavilla ohjelmointikielillä (Peltomäki & Nykänen 2006, 93-97). JavaScriptille on omat kirjastonsa, joita ovat mm. JQuery, React, Vue.js sekä Ember.js. Näiden kirjastojen avulla saadaan tehtyä helpommin erilaisia elementtejä Internet-sivuille (Goel 27.1.2020.)

2.3 Sovellusten tietoturva

Sovellusten suojaaminen tarkoittaa järjestelmäsuunnittelussa huomioon otettavia erilaisia turvallisuuteen liittyviä toimenpiteitä ja menetelmiä siten, että estetään sovelluksessa olevien tietojen kaappaaminen tai varastaminen ulkopuolisen tahon toimesta. Näitä toimenpiteitä voivat olla mm. sovelluspalomuuri ja laitteistotason suojauksena oleva reititin, joka estää tietokoneen IP-osoitteen näkemisen tai erilaisia protokollia sisältävä suojausrutiini, kuten säännöllinen sovelluksen testaaminen. Sovelluskehityksessä tulee ottaa huomioon tietoturvariskit sekä eri ohjelmistojen haavoittuvuudet, kun sovelluksia käytetään eri verkoissa ja ne on yleensä yhdistetty suoraan pilvipalveluun. Sovellusten suojaamiseen käytetään erityyppisiä suojausominaisuuksia, joita ovat todennus, valtuutus, salaus, lokitiedot sekä sovellusten suojauksen testaus (WMware 2020.)

Sovellusten suojaamista voidaankin kuvailla erityiseksi prosessiksi, jonka avulla etsitään, korjataan sekä parannetaan sovellusten tietoturvaominaisuuksia erilaisten salassapitotekniikoiden ja työkalujen avulla. Sovelluksiin voidaan jo kehittämisen aikana koodata erilaisia tietoturvaominaisuuksia lisääviä elementtejä, jotta saadaan testattua varhaisessa vaiheessa sovelluksen mahdolliset haavoittuvuudet (Storm 2019.)

Todennuksella varmistetaan, että vain valtuutetut käyttäjät pääsevät kirjautumaan sovellukseen sekä se, että käyttäjä todella on, kuka hän sanoo olevansa. Tämä voidaan toteuttaa siten, että käyttäjältä vaaditaan käyttäjätunnus ja salasana sisäänkirjautumisen yhteydessä. Monitekijäisellä todennuksella tarkoitetaan sellaista varmennusta, jossa vaaditaan useampi kuin yksi todennuksen muoto. Näitä muotoja voivat olla salasanan lisäksi mm. mobiililaitteen avulla tehty tunnistautuminen tai biometrinen tunnistautuminen, joka voi olla esimerkiksi kasvojen tai sormenjäljen avulla tapahtuva tunnistus. Valtuutuksella tarkoitetaan sitä, että todennuksen jälkeen järjestelmä vahvistaa käyttäjän oikeuden käyttää sovellusta vertaamalla käyttäjän henkilöllisyyttä järjestelmässä sijaitsevaan valtuutettujen henkilöiden luetteloon (WMware 2020.)

Kun käyttäjä on todennettu ja käyttää sovellusta, voidaan salauksen avulla suojata arkaluontoisia tietoja tietoverkkorikkomuksilta erilaisten tietoturvatoimenpiteiden avulla. Varsinkin pilvipalvelupohjaisia sovelluksia käytettäessä käyttäjän ja palvelun välisen datan siirrossa voidaan käyttää salausta siirtyvien tietojen tietoturvan varmistamiseksi (WMware 2020.)

Lokitiedot tarkoittavat sovelluksen muodostamia aikaleimattuja tapahtumamerkintöjä, joiden avulla voidaan mahdollisesti tunnistaa tietoturvarikkomuksien tekijä. Lokitiedostoista nähdään, kuka on päässyt tietoihin käsiksi, miten se on tapahtunut ja mihin sovelluksen osiin pääsy kohdistui (WMware 2020.)

Sovelluksen suojauksen testaus on välttämätöntä ohjelmistokehityksessä, koska sen avulla varmistetaan, että kaikki tietoturvan hallintaan liittyvät tekniikat toimivat oikein ja sovellus täyttää tietoturvalle asetetut kriteerit. Läpäisytestauksessa ohjelmoijat testaavat sovellusta samankaltaisilla tekniikoilla, kuin hakkeroijat käyttävät ja tällä tavoin pääsevät korjaamaan sovelluksessa esiintyvät suojausheikkoudet (VMware 2020.) Staattinen testaus tarkoittaa koodin analysointia kiinteissä pisteissä sen kehittämisen aikana, jolloin ohjelmoija voi tarkistaa koodinsa kirjoittaessaan sitä varmistaakseen tietoturvan käyttöönoton jo kehityksen aikana. Käynnissä olevaa koodia voidaan analysoida dynaamisella testauksella, jolloin voidaan simuloida järjestelmiin kohdistuvia hyökkäyksiä ja sitä kautta paljastaa monimutkaisempia hyökkäysmalleja. Interaktiivinen testaus tarkoittaa staattisen ja dynaamisen testauksen elementtien yhdistämistä (Storm 2019.)

Yleisin tietoturvauhka web-sovelluskehityksessä on SQL-injektio. Tällä tarkoitetaan hyökkäystä, joka mahdollistaa haitallisten SQL-lauseiden suorittamisen web-sovelluksen tietokantapalvelimelle. SQL-injektion avulla hyökkääjä saa ohitettua sovelluksen suojaustoimenpiteet ja kiertää verkkosivuston todennuksen sekä valtuutuksen. Injektion kautta hyökkääjän on mahdollista saada haltuunsa koko tietokannan sisältö sekä mahdollisuuden lisätä, poistaa tai muokata tietokannassa olevia tietueita. SQL-injektiolla saadaan haavoitettua mitä tahansa suojaamatonta verkkosivustoa tai -sovellusta, joka käyttää SQL-tietokantaa. Näitä tietokantoja ovat mm. MySQL, jota käytetään tässä opinnäytetyössä sekä Oracle ja SQL Server (Acunetix.)

Tehdäkseen SQL-injektion, hyökkääjän tulee löytää haavoittuva käyttäjäsyöte verkkosivulta tai -sovelluksesta. Mikäli haavoittuvuus löytyy, voi hyökkääjä syöttää suoraan sisältöä SQL-kyselyn kautta tietokantaan. Tätä kutsutaan haitalliseksi hyötykuormaksi, joka on keskeinen osa hyökkäystä. Kun sisältö on lähetetty, voi hyökkääjä suorittaa haitalliset SQL-komennot suoraan tietokantaan. Ainoa varma tapa estää SQL-injektiohyökkäyksiä

on validoida syötteet sekä käyttää kyselyitä, joihin on määritetty parametrit. Ohjelmiston kehittäjän tulee siistiä kaikki syötteet, kuten mm. kirjautumislomakkeet sekä muut vastaavat sekä poistaa potentiaaliset koodielementit, joiden avulla SQL-injektio voidaan tehdä (Acunetix.)

Tietoturvan lisäämiseksi voidaan myös asentaa tietokanta palomuurin taakse erilliselle tietokantapalvelimelle sen sijaan, että tietokanta olisi samassa paikassa kuin web-sovelluksen palvelin. Web-sovelluksesta säilytettävät tiedostot sisältävät yleensä tiedot myös tietokannasta, johon sovellus ottaa yhteyttä. Tämän vuoksi tulisi salata kaikki sovelluksessa tai tietokannassa säilytettävä tieto, joka on arvokasta sovelluksen käyttäjälle. Lisäksi voidaan käyttää verkkosovellukselle tarkoitettua palomuuria, jonka avulla voidaan suojata myös sivustojen välisten komentosarjojen haavoittuvuuksia sekä mm. SQL-injektiohyökkäyksiä (Matelski 2015.)

Web-sivuilla sekä web-sovelluksissa tietoturvauhkana on myös istuntokaappaus, jossa hyökkääjä saa käynnissä olevan istunnon hallintaansa. Istuntokaappauksessa hyökkääjä varastaa käynnissä olevan istunnon istuntotunnisteen, jolloin web-sivusto luulee hyökkääjän olevan luvallinen käyttäjä ja antaa hyökkääjälle oikeudet käyttää sivustoa. Istuntokaappausta voidaan ehkäistä salausmenetelmien avulla. Istunnon hallinta tarkoittaa käyttäjän, selaimen ja palvelimen välille muodostettua pidempikestoista yhteyttä. Istunto aloitetaan todennuksella ja istunnon elinkaari on se aika, jonka käyttäjä on kirjautuneena järjestelmään (Banach 22.8.2019.)

3 Autopaikkojen vuokrasopimusjärjestelmän toteuttaminen

Opinnäytetyön aiheena oleva sovellus kehitetään web-ohjelmoinnin keinoin, koska selainsovellus sopii hyvin ympäristöön, jossa käyttäjät sijaitsevat fyysisesti eri paikoissa. Etuna on myös, ettei käyttäjien koneille tarvitse asentaa mitään, vaan käytön voi aloittaa heti omalla Internet-selaimella.

Tavoitteena on toteuttaa toimiva, mahdollisimman kevyt ja helppokäyttöinen autopaikkojen vuokrasopimusten tulostusjärjestelmä, jota voidaan käyttää autopaikkojen vuokrasopimusten tulostamiseen niin asukkaille kuin ulkopuolisillekin vuokralaisille. Autopaikkasopimus on autopaikkakohtainen, ja sopimuksen muoto on samanlainen sekä asukkaille että ulkopuolisille autopaikan vuokraaville henkilöille.

3.1 Projektin työsuunnitelma

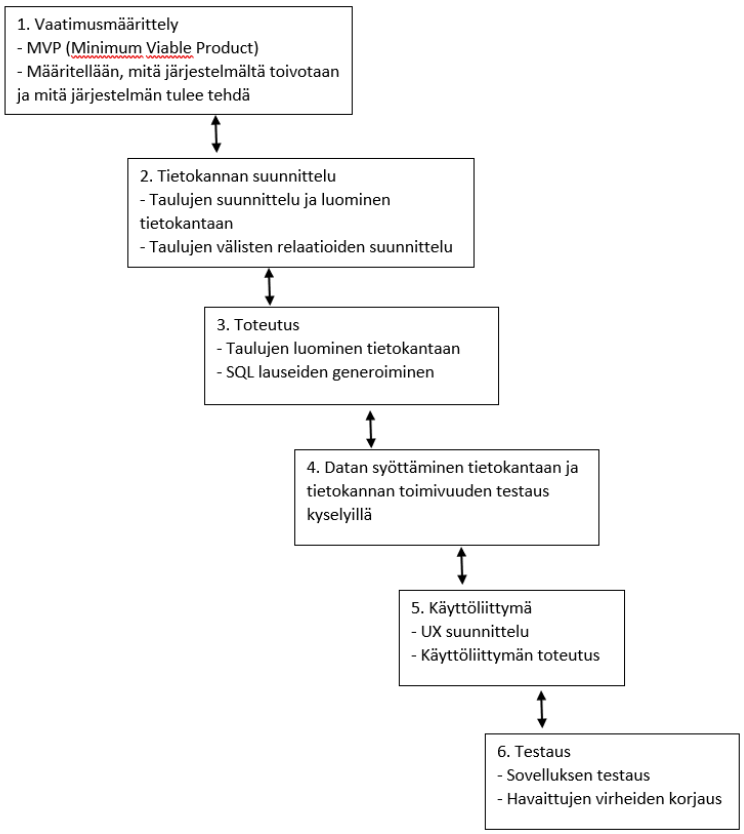
Koska järjestelmän tarve ja toimivuus halutaan testata pienimmällä mahdollisella työllä sekä mahdollisimman nopeasti, tehdään työ MVP-toteutuksena (Minimum Viable Product). Näin saadaan validoitua, ratkaiseeko järjestelmä olemassa olevan tarpeen riittävällä tavalla. Ennen muita vaiheita järjestelmästä tehdään vaatimusmäärittely dokumentti, joka kuvaa järjestelmään kohdistuvia vaatimuksia sekä tavoitteita. Vaatimusmäärittelyssä kuvataan erilaisin käyttötapauskaavioin sekä testitapauksin, miten järjestelmän tulisi toimia.

Toimenpiteet on jaettu kuuteen eri päävaiheeseen ja työskentely tapahtuu vesiputousmallin mukaisesti iteratiivisella periaatteella siten, että työvaiheiden sisällä testataan jatkuvasti tehdyn muutoksen toimivuus. Ensimmäisenä tehdään vaatimusmäärittely, jonka jälkeen suunnitellaan tietokanta. Suunnittelu aloitetaan tietokantaan lisättävien taulujen sekä taulujen suhteiden eli relaatioiden suunnittelulla. Tässä työvaiheessa hahmotellaan piirtämällä tai käyttämällä esimerkiksi Excel-työkirjaa, taulujen relaatiot. Samalla hahmotetaan, mikä tieto on tietokannan osalta tärkeää ja karsitaan pois se tieto, mikä ei ole oleellista MVP-ajattelumalliin perustuvan järjestelmän toimivuuden kannalta. Taulujen välisiä relaatioita luodessa merkitään myös taulujen pääavaimet, viiteavaimet sekä id, eli yksilöllinen tunnistenumero.

Kun tietokanta on suunniteltu, siirrytään kolmanteen päävaiheeseen eli tietokannan toteutukseen, joka tarkoittaa taulujen ja relaatioiden luomista tietokantaan. Taulujen luominen tapahtuu manuaalisesti ensimmäisessä vaiheessa määritellyn suunnitelman perusteella. Taulujen luomisen jälkeen generoidaan SQL-lauseet käytettävässä editorissa.

Neljännessä vaiheessa syötetään tietokantaan dataa ja testataan kyselyillä tietokannan toimivuus. Tietokantaan syötettävä data on testidataa, joka ei sisällä henkilötietoja. Tässä vaiheessa laaditaan myös PHP:llä suoritettavat SQL-tietokantakyselyt.

Viidennessä vaiheessa suunnitellaan käyttöliittymän ulkoinen osa eli se osa, joka näkyy käyttäjille ja toteutetaan se. Viimeisessä vaiheessa testataan prototyyppiä ja korjataan mahdollisesti havaittavia virheitä.



Kuva 5. Työn suunnitelmakaavio.

3.2 Vaatimusmääritys

Vaatimusmäärityksessä kuvataan ensimmäisenä nykytila. Tällä hetkellä autopaikkasopimuksia tekevät eri huoltoyhtiöt sekä asuntovuokraus. Sopimuspohja on kaikilla toimijoilla erilainen ja autopaikkojen vuokrasopimusten sopimusehdot voivat olla hyvin erilaiset riippuen toimijasta. Erilaiset sopimusehdot voivat aiheuttaa autopaikkoja vuokraavilla vuokralaisilla sekaannusta, jos ehdot ovat hyvin erilaiset eri vuokrasopimuksissa.

Seuraavaksi mietitään, mikä on toiminnallinen tavoite ja mitä hyötyjä tavoitellaan. Tavoitteena on saada yhdenmukaiset sekä tarkempiehtoiset autopaikkojen vuokrasopimukset.

Lisäksi on tärkeää, että ehtoja voidaan päivittää helposti ja päivitetyt vuokrasopimukset ovat kaikkien käyttäjien ulottuvilla reaaliajassa. Vuokrasopimukset tulee voida tulostaa pdf-muodossa.

Tämän jälkeen päätetään, miten rajataan järjestelmä. Järjestelmässä ei tulla varastoimaan vuokrasopimuksia eikä asiakkaiden tietoja ja autopaikan vuokrasopimus poistuu järjestelmästä automaattisesti, kun käyttäjä on tulostanut vuokrasopimuksen ja poistuu käyttöliittymän tulostusnäkymästä. Järjestelmään tallennetaan yleiset sopimusehdot sekä kohteiden tiedot, joissa autopaikat sijaitsevat. Käyttäjä voi tarvittaessa muokata yleisiä, kaikkia autopaikkoja koskevia sopimusehtoja sekä kohteiden tietoja. Tämän lisäksi järjestelmään luodaan mahdollisuus lisätä eri kohteille erityisiä sopimusehtoja, jotka koskevat esimerkiksi tietyn taloyhtiön järjestyssääntöjä. Järjestelmään luodaan salasanan ja käyttäjätunnuksen avulla tehtävä tunnistautuminen. Järjestelmä toimii työaseman verkkoselaimella, ja sitä käyttävät eri huoltoyhtiöt ja asuntovuokraajat. Aikataulurajoitteena on produktin valmistuminen opinnäytetyön aikataulun puitteissa.

Vaatimusmäärittely jatkuu toiminnankulkujen kuvaamisella siten, kuinka oikeat toimijat kokisivat käyttötilanteen ja tarinassa kuvataan joko nykytilaa tai tehdyn järjestelmän käyttötilannetta. Näiden kuvausten perusteella laaditaan vaatimuslista, johon poimitaan arvoa tuottavat tavoitteet ja analysoidaan sekä jaetaan tavoitteet tyypin mukaan joko ei-toiminnallisiin tai toiminnallisiin tavoitteisiin. Laadulliset vaatimukset sekä resurssivaatimukset ovat ei-toiminnallisia vaatimuksia ja toiminnallisia vaatimuksia ovat käyttöön liittyvät vaatimukset. Tämän jälkeen vaatimukset priorisoidaan eli laitetaan tärkeysjärjestykseen, jonka jälkeen valitaan, mitkä vaatimukset voidaan toteuttaa MVP-ajattelumalliin perustuvassa järjestelmässä. Vaatimusluettelo on kuvattu liitteessä 1.

Seuraavaksi kuvataan valitut vaatimukset visuaalisesti siten, että toiminnalliset vaatimukset tehdään UML-käyttötapauskaavioina (liite 2) ja säilytettävät tiedot on kuvattu tietokannan relaatiomallissa (liite 3). Lisäksi piirretään käyttöliittymäluonnokset (liite 4) piirto-ohjelmalla. Lopuksi tehdään vaatimusten testaus, joka tarkoittaa käyttötapausten testaamista askel askeleelta siten, että testitapaukset kuvataan yksityiskohtaisesti (liite 5). Järjestelmän tietokantamallissa luodaan tietokantaan lisättävät taulut sekä tietokantataulujen relaatiot. Koko tietomallin luonti tapahtuu PhpMyAdminin avulla MySQL-tietokantaan.

3.3 Toteutus

Opinnäytetyön web-ohjelmointitekniikoiksi valittiin yleisimmin käytössä olevat ohjelmointi- ja merkintäkielet. Tyylien muotoilussa käytettiin myös CSS-tyylikehystä. Sivujen perusrakenteen luomisessa käytettiin HTML5-merkintäkieltä (kuva 6.), koska HTML5 on standardi, jonka etuja ovat sisältöä -elementtiä kuvaavat semanttiset elementit, modernit lomake-elementit ja hyvä tuki erilaisille mediasisällöille (Techark 15.4.2014). HTML5-kielellä määritettiin käyttöliittymädokumentin rakenne ja elementtien asettelu määräytyy eri laitenäkymissä CSS-määrittelyjen mukaan. HTML5 toimii siten, että sovellusta voidaan käyttää myös muilta laitteilta, kuin tietokoneella. Tämä tarkoittaa sitä, että kielen avulla saatiin sivut tehtyä responsiivisiksi ja sivut mukautuvat automaattisesti käyttäjän laitteen näytön kokoon (Medium.com 2018).

```
<nav class="navbar navbar-default">
    <div class="container">
      <ul class="nav navbar-nav navbar-right">
        <li><a href="etusivu.php">Etusivu</a></li>
        <li class="active"><a href="autopaikka.php">Tulosta vuokrasopimus</a></li>
        <li><a href="ehdot.php">Muokkaa sopimusehtoja</a></li>
        <li><a href="kohteet.php">Muokkaa kohteita</a></li>
        <li><a href="">Kirjaudu ulos</a></li>
      </ul>
    </div>
</nav>
```

Kuva 6. HTML5 <body>-elementin sisälle kirjoitetun navigointielementin koodi. Kuvassa on kirjoitettu lista navigointielementtiin, jossa aktiivisena on ”Tulosta vuokrasopimus” -sivu.

CSS-tyylikielen avulla määritettiin käyttöliittymän näkyvälle osalle erilaiset käytettävät fontit, värit sekä elementtien tyylit. CSS on riippuvainen HTML-merkintäkielestä, joten CSS-tiedosto linkitettiin HTML-tiedostoon, jotta muotoilut tulevat näkyviin sivuille (W3C 2016). Tyylikehyksenä opinnäytetyössä käytettiin Bootstrapia, jonka avulla saatiin luotua käyttöliittymälle visuaalisesti sopiva asettelu helpommin. Jotta Bootstrap-tyylikehyksen ominaisuudet saatiin käyttöön, linkitettiin se HTML-tiedostoon.

```
1  <!DOCTYPE html>
2
3  <html>
4
5  <head>
6    <title>Uusi vuokrasopimus</title>
7    <meta charset="utf-8">
8    <meta name="viewport" content="width=device-width, initial-scale=1">
9    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
10   <link rel="stylesheet" type="text/css" href="autopaikka.css">
11   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
12   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
13   <link href="https://fonts.googleapis.com/css?family=Allura|Arapey|Cardo|Covered+By+Your+Grace|Crimson+Text|Dancing+Script|Gentium+Basic|Great+Vibes|Libre+Ba
14 </head>
15
16 <body>
17
```

Kuva 7. CSS ja Bootstrap linkitettynä HTML5-merkintäkielen header-osioon opinnäytetyössä.

Näiden lisäksi lisättiin interaktiivisia elementtejä, kuten erilaisia painikkeita, käyttämällä JavaScript-ohjelmointikieltä (kuva 8.). JavaScript-koodi kirjoitetaan HTML5-kielessä <script>-elementin sisälle tai se ladataan erillisestä .js-tiedostosta (Markupbox 2017). Selain suorittaa JavaScript-koodia selaimen eri tapahtumien yhteydessä, esimerkiksi hiirellä klikattaessa tai dokumentin latauksen yhteydessä (JS).

```
<script type="text/javascript">
function haeRivi(rivit, id) {
    for (const rivi of rivit) {
        if (rivi["PysakointiAlue_id"]==id) {
            return rivi;
        }
    };
}
$(document).ready(function() {
    var data = <?=json_encode($arr)?>;
    var sopeht = <?=json_encode($sopimusehdot)?>;
    $('#pysalue').change(function() {
        var valittuPysalue = $(this).children("option:selected").val();
        var rivi = haeRivi(data, valittuPysalue, sopeht);
        $('#omistaja').val(rivi["Omistaja"]);
        $('#katuos').val(rivi["Katuosoite"]);
        $('#postinro').val(rivi["Postinumero"]);
        $('#ptoimipaikka').val(rivi["Postitoimipaikka"]);
        $('#tilinro').val(rivi["Laskutustili"]);
        $('#erehdot').val(rivi["EhdonSisalto"]);
        $('#sopimusehdot').val(sopeht[0]["EhdonSisalto"]);
    });
    $('#pysalue').change();
});

</script>
```

Kuva 8. Kuvassa näkyy opinnäytetyön Vuokrasopimuksen tulostus- sivun automaattisen täytön JavaScript-funktio, jossa on hyödynnetty jQueryä. Change ja ready ovat eventtejä, joiden tapahtuessa funktiot suoritetaan.

Sovelluksen tietokanta tehtiin SQL-kyselykielellä, jolla voidaan vaikuttaa myös tietokannan rakenteeseen. Tietokannan sisältöä voidaan mm. muuttaa, päivittää tai määritellä erilaisten komentojen muodossa sekä ohjata tapahtumien käsittelyä erilaisten DML- ja DDL-kyselyiden avulla (Hovi 2013, 14-16.) Tietokantakyselyt muodostettiin käytössä olevan tietokanta-alustan (MySQL) ohjeen mukaan.

```php
<?php

$config = parse_ini_file($_SERVER["CONTEXT_DOCUMENT_ROOT"] . "/../pdo.ini");

#$connection = mysqli_connect($config['server'], $config['username'], $config['password'], $config['dbname']);
$dsn = "mysql:host=" . $config['server'] . ";dbname=" . $config['dbname'] . ";charset=utf8mb4";
#$dsn = "mysql:host=localhost;dbname=myDatabase;charset=utf8mb4";
$options = [
  PDO::ATTR_EMULATE_PREPARES   => false, // turn off emulation mode for "real" prepared statements
  PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION, //turn on errors in the form of exceptions
  PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC, //make the default fetch be an associative array
];
try {
  $pdo = new PDO($dsn, $config['username'], $config['password'], $options);
  $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
} catch (Exception $e) {
  error_log($e->getMessage());
  exit('Database connection failed'); //connection failed
}
?>
```

Kuva 9. PHP- elementin sisällä ladataan ensin ($config) erillisestä tiedostosta käyttäjätunnus ja salasana. Sen jälkeen muodostetaan yhteys PDO:n avulla tietokantaan. PDO:n prepare-funktiota käyttäen tehdään parametroituja kyselyjä ja torjutaan tällä tavoin ainakin helpoimmin toteutettavat SQL-injektiohyökkäykset.

Tietokantapalvelimen pääsytiedot (käyttäjätunnus ja salasana) säilytetään erillisessä .ini-tiedostossa www-palvelimen public_html-hakemiston ulkopuolella, josta ne ladataan PHP:n avulla. Näin pääsytiedot eivät ole kenen tahansa ladattavissa palvelimelta, tai luettavissa ohjelmakoodista (Lionite.)

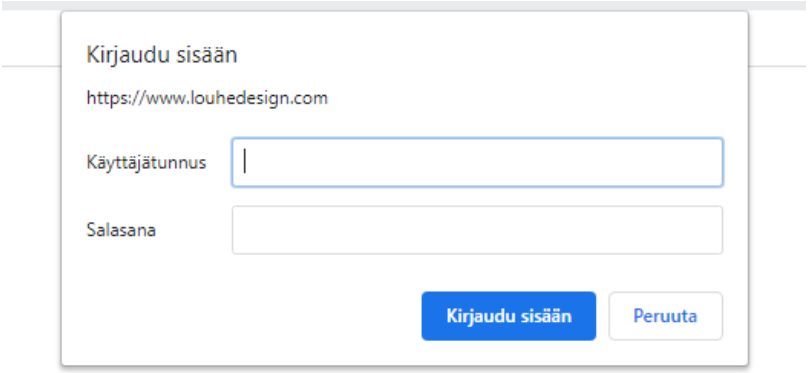
Opinnäytetyön palvelinkoodi toteutettiin PHP-kielellä, koska PHP:n kanssa voitiin käyttää helposti kaikkia muita työhön valittuja kieliä (php.net, B). PHP toimii hyvin MySQL:n kanssa yhdessä mm. sen vuoksi, että siinä on hyvät tukikirjastot MySQL:lle (McLaughlin 2020). Lisäksi PHP:llä on hyvä saatavuus, koska se löytyy käytännössä kaikista hosting-palveluista. Opinnäytetyössä toteutettavan järjestelmän teossa käytettävä hosting-palvelu tukee PHP:ta, joka on myös yksi syy, miksi palvelinkoodiksi valikoitui PHP-kieli. Tietokannan ja PHP:n välisenä rajapintana käytetään yleensä kirjastoa, jonka avulla rakennetaan SQL-kyselyt ja otetaan yhteys tietokantaan. Tällainen kirjasto on PHP:n dataobjekti PDO, jonka avulla saadaan myös tietoturvaominaisuuksia vaivattomammin käyttöön, kuin esimerkiksi sillä, että SQL-kyselyt rakennettaisiin käsin (php.net c.)

3.4 Tuotos

Tässä MVP-ajattelumallin vuokrasopimusjärjestelmässä ei ole tarvetta käyttäjän tunnistautumiselle, mutta luvaton käyttö tulee sen sijaan estää. Tämän toteuttamiseksi riittää Apache HTTP-palvelimen yksinkertainen autentikaatio (Basic HTTP autenthication) yhteiskäyttöisellä käyttäjätunnuksella (kuva 10.). Tämä onnistuu luomalla .htpasswd-

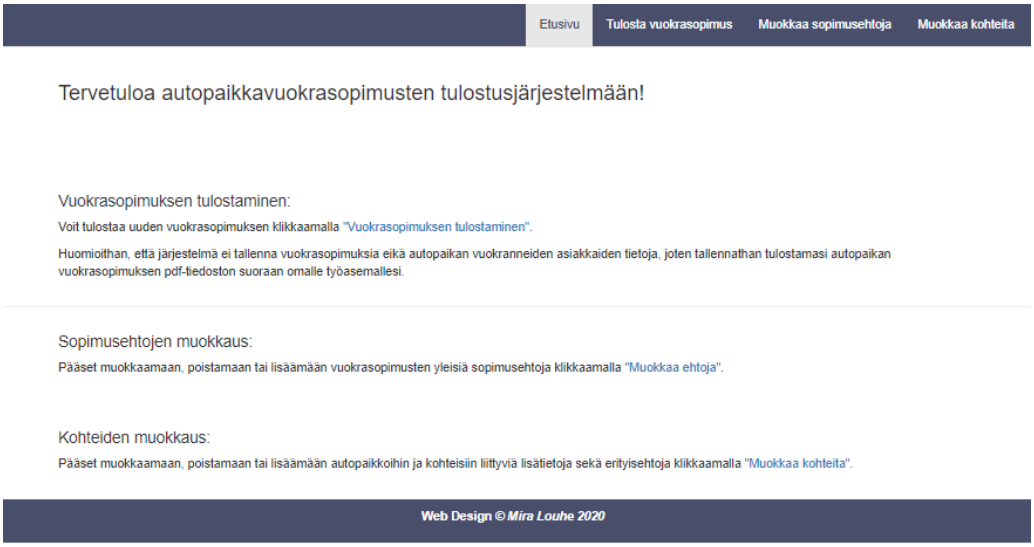
tiedosto, joka sisältää käyttäjätunnukset ja salasanat salatussa muodossa, sekä .htaccess-tiedosto, jossa määritellään Apachelle hakemiston suojausasetukset. Salasanatiivisteet sisältävä .htpasswd-tiedosto sijoitetaan julkisen www-dokumenttihakemiston (publick_html) ulkopuolelle, jotta se ei joutuisi vääriin käsiin.

Koska kyseessä on MVP-ajattelumalliin perustuva minimalistinen sovellus, ei sovellukselle tehty uloskirjautumista. Apachen .htaccess pitää kirjautumisen yllä siihen asti, että selain suljetaan. Kun käyttäjä sulkee käytössään olevan selaimen, uloskirjautuminen tapahtuu automaattisesti.



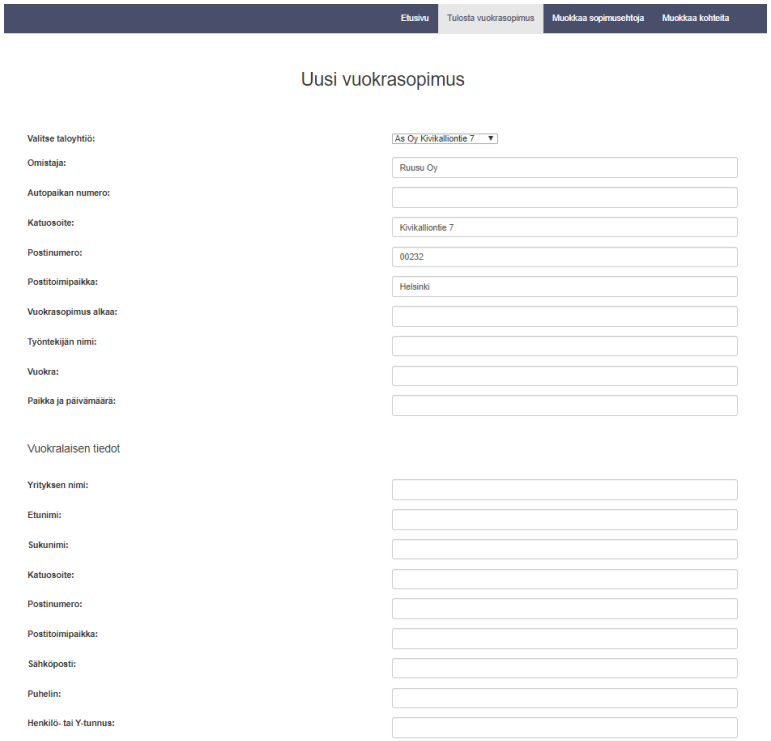
Kuva 10. Näkymä kirjautumisesta, kun järjestelmä pyytää käyttäjätunnusta sekä salasanaa.

Kun käyttäjä on kirjautunut järjestelmään, aukeaa käyttöliittymän etusivu (kuva 11.). Käyttäjä voi valita haluamansa toimenpiteen yläpalkin navigoinnista tai sivun sisällön sinisillä merkityistä linkeistä. Kutakin linkkiä painamalla käyttäjä siirtyy linkin otsikkoa vastaavalle sivulle.



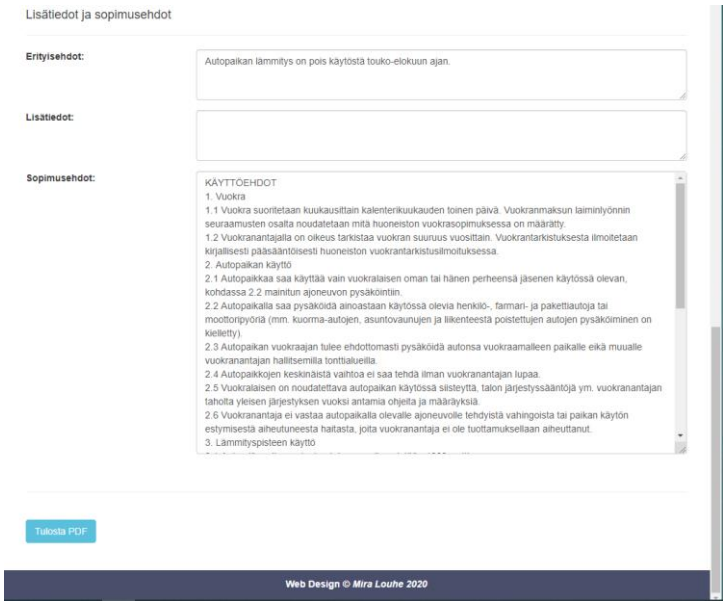
Kuva 11. Sovelluksen etusivu.

Klikkaamalla Tulosta vuokrasopimus -painiketta, aukeaa näkymä vuokrasopimuslomakkeelle. Kun käyttäjä valitsee sivun, käyttöliittymä hakee automaattisesti tietokannasta kohteiden taulusta ensimmäisenä olevan kohteen tiedot. Käyttäjä voi valita haluamansa taloyhtiön Taloyhtiö- kohdan pudotusvalikosta. Kun käyttäjä vaihtaa taloyhtiön, tietokannasta tulee automaattisesti halutun taloyhtiön tiedot vuokrasopimuslomakkeelle. Kun taloyhtiö on valittu ja autopaikkaa koskevat muut tiedot on valittu, syötetään lomakkeelle autopaikan vuokraavan henkilön tai yrityksen tiedot.



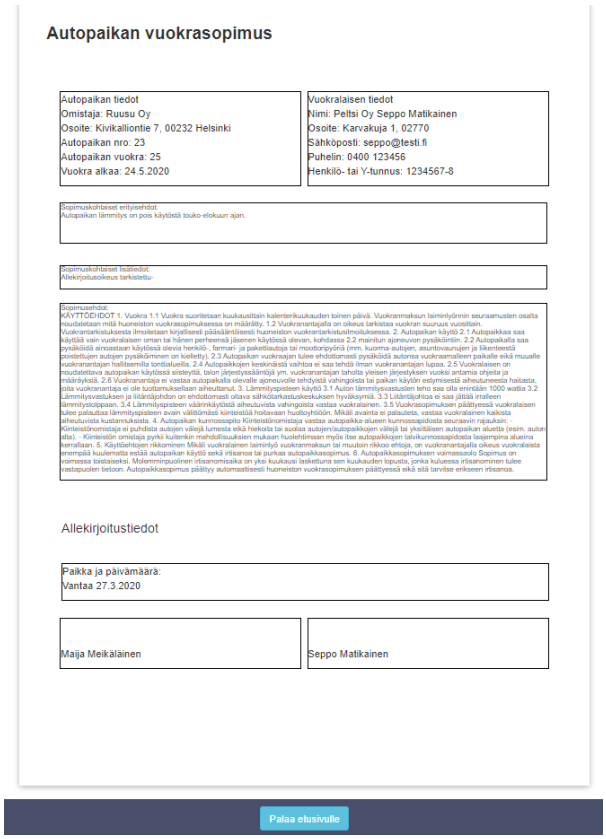
Kuva 12. Vuokrasopimuslomakkeelle tulee automaattisesti kohteen tiedot.

Vuokralaisen tietojen syöttämisen jälkeen voidaan lisätä manuaalisesti Lisätiedot -kohtaan, jos sopimukseen kohdistuu erityisiä lisätietoja. Vuokrasopimuksen erityisehdot tulevat lomakkeelle automaattisesti, kun taloyhtiö valitaan, jos taloyhtiöön kohdistuu erityisehtoja. Mikäli taloyhtiöön ei kohdistu erityisehtoja, jää Erityisehdot -kenttä automaattisesti tyhjäksi. Tämän lisäksi vuokrasopimuslomakkeelle tulostuu automaattisesti yleiset sopimusehdot, kun sivu aukeaa.



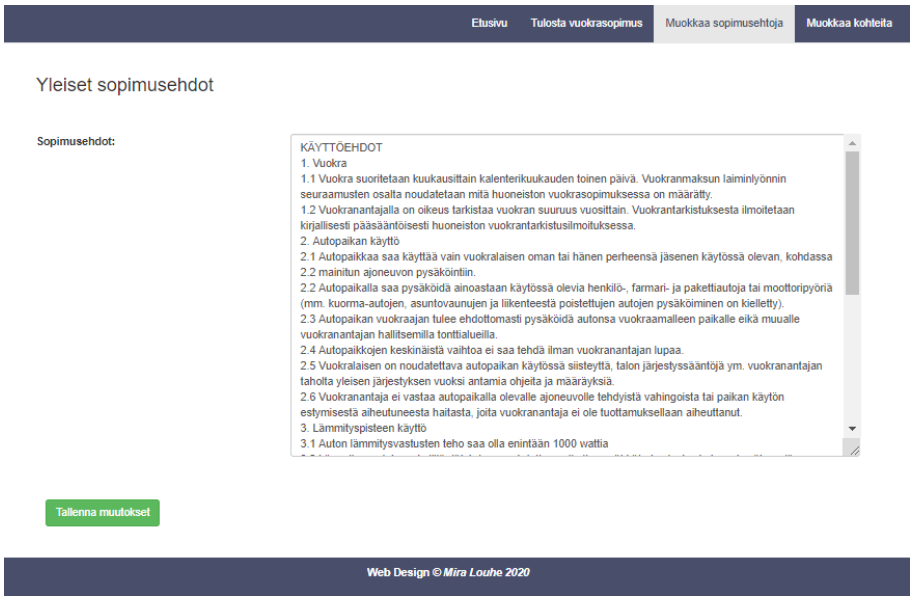
Kuva 13. Lisätiedot ja sopimusehdot.

Vuokrasopimustuloste avautuu omaan näkymään, johon on jaettu vuokrasopimuslomakkeen tiedot omille paikoilleen. Tulostuksessa käytetään yleisimpien selainten tukemaa PDF-tulostusta ja käyttäjä voi palata etusivulle sivun alalaidassa olevan painikkeen avulla. Vuokrasopimus ei tallennu järjestelmään ja sopimus katoaa, kun käyttäjä palaa etusivulle tai istunto vanhenee.



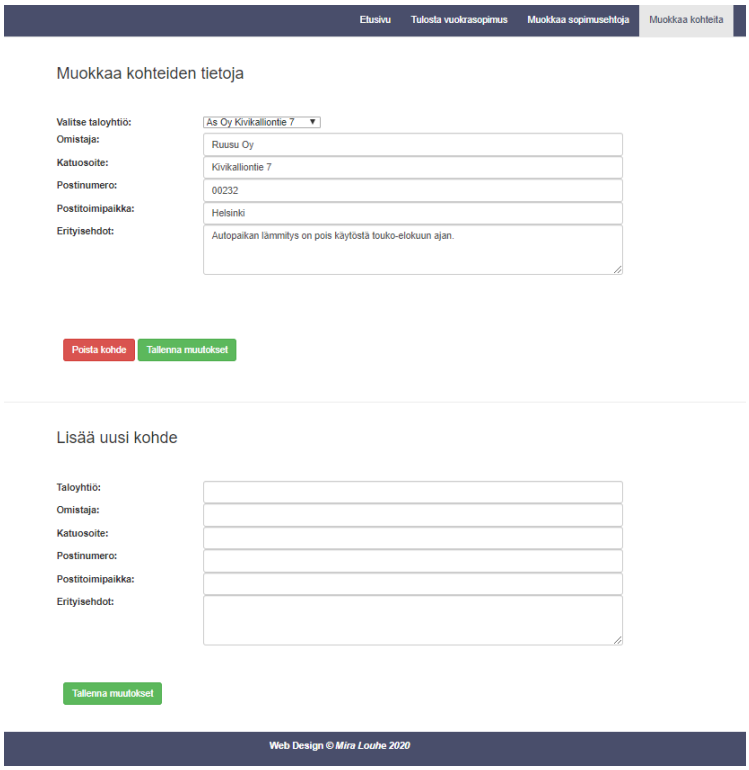
Kuva 14. Kuvassa autopaikan vuokrasopimuksen tulostenäkymä.

Käyttäjä voi tarvittaessa muokata vuokrasopimuksen yleisiä sopimusehtoja Muokkaa sopimusehtoja -sivun kautta. Kun sivu aukeaa, sopimusehdot latautuvat automaattisesti tietokannasta käyttöliittymän Sopimusehdot-kohtaan. Sopimusehtoja ei voi poistaa kokonaan, koska vuokrasopimuksia tehdessä tulee olla käytössä ennalta määritellyt vuokrasopimusehdot, jotka ovat yhtenäiset kaikissa vuokrasopimuksissa.



Kuva 15. Yleisten sopimusehtojen muokkaaminen.

Käyttäjä voi tarvittaessa muokata kohteen tietoja tai poistaa kohteen Muokkaa kohteita-sivun kautta. Kun käyttäjä avaa sivun, latautuu sivulle suoraan tietokannasta kohteiden taulusta ensimmäisenä olevan taloyhtiön tiedot. Käyttäjä voi valita toisen taloyhtiön pudotusvalikon kautta ja valitun taloyhtiön tiedot avautuvat automaattisesti tekstikenttiin. Käyttäjä voi lisätä järjestelmään uuden kohteen saman sivun kautta, mistä käyttäjä saa muokattua kohteiden tietoja tai poistaa kohteen. Käyttäjä syöttää uuden kohteen tiedot tekstikenttiin ja tallentaa tiedot, jolloin uusi kohde tallentuu tietokantaan.



Kuva 16. Kohteen tietojen lisäys, muokkaus ja poisto.

Ohjelmakoodi on tämän opinnäytetyön julkaisuhetkellä tarkasteltavissa osoitteessa https://github.com/Zippura/autopaikka.

4 Pohdinta

Opinnäytetyön aiheen valitseminen oli yllättävän helppoa, mutta itse työn aloittaminen oli vaikeampaa, kuin olin kuvitellut. Työskentelen tällä hetkellä kiinteistöalalla, joten oli luontevaa valita aiheeksi järjestelmä, joka liittyy omaan alaani. Lisäksi halusin opiskelujeni päätteeksi koota yhteen opiskeluaikanani saavuttamani uudet taidot. Aloittaessani opiskeluni hieman yli kolme vuotta sitten minulla ei ollut mitään ohjelmointikokemusta enkä ollut kirjoittanut riviäkään koodia. Tässä opinnäytetyössä tehty järjestelmä on ensimmäinen laatuaan, jonka olen koskaan tehnyt alusta loppuun asti itse. Tämän vuoksi oli haasteellista hahmottaa opinnäytetyön tehtävät oikeisiin ajanjaksoihin ja opinnäytetyösuunnitelma oli hieman puutteellinen aikataulullisesti. Opiskelun aikana käymäni kurssit olivat onneksi valmistaneet minua opinnäytetyötä varten, joten sain luotua itseäni varten suunnitelman sekä kokonaisuuden, joiden avulla pääsin aloittamaan myös produktin tekemisen varsinaisen opinnäytetyön kirjoittamisen lomassa.

Tietoperustassa päädyin ohjelmoinnin perusteiden avaamiseen mahdollisimman yksinkertaisten esimerkkien avulla. Tarkoituksena oli ensin selventää lukijoita varten erilaisten ohjelmointikielien ominaisuuksia, jonka jälkeen opinnäytetyössä siirryttiin sovellukseen valittuihin ohjelmointikieliin sekä ominaisuuksiin. Tietoperustat oli tarkoitus pitää mahdollisimman yksinkertaisina ajatellen myös sellaista lukijaa, jolla ei ole ohjelmointikokemusta siten, että lukija pystyisi kuitenkin ymmärtämään mahdollisimman pitkälle ohjelmoinnin perusteet sekä miten käytännössä sovellus tehdään.

Vaatimusmäärittelyn tekeminen sovelluksesta oli välttämätöntä, että saatiin määritettyä selkeämmin sovelluksen käyttöarvoa lisäävät ominaisuudet. Vaatimusmäärityksessä tehtyjen käyttötapausten perusteella suoritettu sovelluksen testaaminen oli suhteellisen nopeaa ja helppoa. Tehdystä vaatimusmäärittelystä huolimatta työskentelyn edetessä produkti muuttui sitä mukaa, kun järjestelmän rakenne hahmottui. Alkuperäisen suunnitelman mukaan käyttäjille tehtiin oma taulu tietokantaan, mutta työskennellessä huomasin, ettei käyttäjien tietoja välttämättä tarvita MVP-ajattelumalliin perustuvassa produktissa. Tämän vuoksi käyttäjien tietojen muokkaaminen sekä tallennus jätettiin kokonaan pois järjestelmän ominaisuuksista. Sovelluksen käyttäjille tehtiin tähän opinnäytetyön versioon yhteinen käyttäjätunnus ja salasana. Jos sovellus otetaan käyttöön, tulisi sovellukseen kuitenkin lisätä kirjautuminen sekä käyttäjien tunnistus tietoturvasyistä. Salasanan ja käyttäjätunnuksen omaava ilkeämielinen käyttäjä saattaa käydä turmelemassa tahallisesti tietokannan sisällön. Tätä voidaan ehkäistä sovellusta kehitettäessä siten, että asetetaan käyttäjille erilaiset oikeustasot sekä varmuuskopioimalla tietokanta säännöllisesti.

Sovellusta testattaessa kävi hyvin nopeasti ilmi, että sovellus edesauttaisi yhdenmukaisten autopaikkavuokrasopimusten tekemistä ja sopimusten ehtojen reaaliajassa tehtyä päivittämistä. Vuokrasopimuksen tulostaminen pdf-versioksi oli testattaessa huomattavasti helpompaa sovelluksesta, kuin nykyisen käytännön perusteella Excel-versiosta. Edellä mainittujen tulosten puitteissa voidaan todeta, että sovelluksella voisi olla myös tuotannollista arvoa, mutta tämänhetkinen testiversio vaatii vielä hieman jatkokehitystä. Mikäli sovellusta jatkojalostetaan, tulee kuitenkin pohtia täyttääkö sovelluksen käyttöarvo todelliset kehityskustannukset. Vaikka sovelluksella todettiin olevan käyttöarvoa, saattaa sovelluskehitys vaatia niin isot investoinnit, että tällä hetkellä sovelluksen jatkokehitys ei olisikaan taloudellisesti kannattavaa siitä saatuun hyötyyn nähden.

Bisneksen kannalta sovelluksella olisi kuitenkin entistäkin enemmän tällä hetkellä käyttöarvoa, kun otetaan huomioon vallitseva COVID-19-kriisi. Etätyössä usein käytetyt VPN-yhteydet voivat olla vallitsevassa kriisitilanteessa hitaat käyttäjäkuormituksen vuoksi, joten käyttöpaikasta riippumaton selaimella suoritettava sovellus on tällä hetkellä arvokas työkalu, jonka avulla pidetään tärkeät tiedot reaaliajassa kaikkien niitä tarvitsevien henkilöiden ulottuvilla.

Jatkokehitystä varten tulisi huomioida käyttäjien osalta käyttäjät-tietokantataulun käyttöönotto siten, että saataisiin luotua kaikille käyttäjille omat tunnukset ja mahdollisuus rajoittaa käyttäjäoikeuksia tarvittaessa. Sovelluksen käyttöoikeudet olisi järkevintä asettaa edellä mainitussa tapauksessa siten, että vain esimerkiksi yksi käyttäjä olisi pääkäyttäjä, jolla olisi oikeus lisätä, poistaa sekä muokata sovelluksessa olevia tietoja (käyttäjät, kohteet yms.). Sovelluksen poisto-ominaisuus kannattaa kehittäessä muuttaa sellaiseksi, että käyttäjän tekemä poisto asettaa tietueeseen vain esimerkiksi poistettu=True- merkinnän ja piilottaa poistetun tiedon näkyvistä, mutta ei kokonaan tietokannasta. Tällöin vahingossa poistettu tieto on helposti palautettavissa takaisin käyttöön. Lisäksi jatkokehitystä ajatellen voisi sovellukseen lisätä myös mahdollisuuden tallentaa vuokrasopimukset ja esimerkiksi rajoittaa käyttäjien oikeuksia nähdä muiden tekemiä vuokrasopimuksia.

Sovellusta voisi myös jatkojalostaa siten, että sen saisi liitettyä jo tälläkin hetkellä käytössä olevaan järjestelmään. Vaihtoehtoina voisi tässä tapauksessa olla integrointi tai sitten sovelluksen koodeja voisi muokata toiseen järjestelmään sopivaksi, jolloin sovellus sulautuisi toisen järjestelmän sisällä olevaksi ominaisuudeksi. Tällainen järjestelmä voisi olla esimerkiksi FimX, josta löytyy jo työpaikallani mm. autopaikkojen varaukset sekä muut kohteisiin liittyvät asiat. FimX:n ominaisuuksiin ei kuulu tällä hetkellä vuokrasopimusten tallentaminen tai tulostaminen, mutta käytännössä se voisi olla mahdollista.

Lähteet

Acunetix. What is SQL Injection (SQLi) and How to Prevent it. Luettavissa: https://www.acunetix.com/websitesecurity/sql-injection/. Luettu: 10.2.2020.

Banach, Zbigniew. 22.8.2019. What Is Session Hijacking: Your Quick Guide to Session Hijacking Attacks. Luettavissa: https://www.netsparker.com/blog/web-security/session-hijacking/. Luettu: 15.2.2020.

Crampete. 2020. Introduction To Server-Side Programming Languages. Luettavissa:https://www.crampete.com/resources/blogs/introduction-to-server-side-programming-languages/. Luettu: 15.10.2019.

Didacus, O. 2018. System Design in Software Development. Luettavissa: https://medium.com/the-andela-way/system-design-in-software-development-f360ce6fcbb9. Luettu: 30.9.2019.

Gibb, R. 2016. What is a Web Application? Luettavissa: https://blog.stackpath.com/web-application/. Luettu: 20.1.2020.

Goel, A. 27.1.2020. 10 Best JavaScript Frameworks to Use in 2020. Luettavissa: https://hackr.io/blog/best-javascript-frameworks. Luettu: 10.2.2020.

Haapahovi, M. 12.3.2017. Mikä on MVP eli Minimum Viable Product? Luettavissa: https://www.haapahovi.fi/mika-on-mvp-eli-minimum-viable-product/. Luettu: 27.1.2020.

Hovi, A. 2013. SQL-opas. 13. painos. Docendo Oy. Jyväskylä

JS. Introduction to browser events. Luettavissa: https://javascript.info/introduction-browser-events. Luettu: 16.2.2020.

Kim, L. 2015. 10 Most Popular Programming Languages Today. Luettavissa: https://www.inc.com/larry-kim/10-most-popular-programming-languages-today.html. Luettu: 30.9.2019.

Lionite. Using PHP with MySQL – The Right Way. Luettavissa: https://www.binpress.com/using-php-with-mysql/. Luettu: 16.2.2020.

Markupbox. 2017. Top 9 Advantages of JavaScript. Luettavissa: https://www.markupbox.com/blog/advantages-of-javascript/amp/. Luettu: 15.11.2019.

Matelski, J. 2015. IOUGHT Insight: 5 Best Practices for Securing Databases. Luettavissa: http://www.dbta.com/Editorial/News-Flashes/IOUG-Insight-5-Best-Practices-for-Securing-Databases-101930.aspx. Luettu: 10.2.2020.

McLaughlin B. 2020. PHP&MySQL: The Missing Manual. Luettavissa: https://www.oreilly.com/library/view/php-mysql/9781449318857/ch04.html. Luettu: 20.9.2019.

Medium.com. 2018. What is HTML5 and what can I do with it? Luettavissa: https://medium.com/adalab/what-is-html5-and-what-can-i-do-with-it-d6bc85eb8af9. Luettu: 30.9.2019.

php.net a. What is PHP? Luettavissa: https://www.php.net/manual/en/intro-whatis.php. Luettu: 15.11.2019.

php.net b. What can PHP do? Luettavissa: https://www.php.net/manual/en/intro-whatcando.php. Luettu: 15.11.2019.

php.net c. Introduction. Luettavissa: https://www.php.net/manual/en/intro.pdo.php. Luettu: 15.11.2019.

Peltomäki, J. & Nykänen O. 2006. Web-selainohjelmointi. Docendo.

Purely Branded. Web Design or Development, What's the Difference? Luettavissa: https://www.purelybranded.com/insights/web-design-or-web-development-whats-the-difference/. Luettu: 30.9.2019.

Storm, D. 2019. What is application security? A process and tools for securing software. Luettavissa: https://www.csoonline.com/article/3315700/what-is-application-security-a-process-and-tools-for-securing-software.html. Luettu: 10.2.2020

Stewart, L. 14.1.2019. Front End vs Back End Development. Luettavissa: https://www.coursereport.com/blog/front-end-development-vs-back-end-development-where-to-start. Luettu: 30.9.2019.

Techark. 15.4.2014. Advantages of HTML5. Luettavissa: https://gotechark.com/blog/advantages-html5/. Luettu: 30.9.2019.

W3C. 2016. HTML & CSS. Luettavissa: https://www.w3.org/standards/webdesign/htmlcss. Luettu: 15.11.2019.

WMware. 2020. Application security. Luettavissa: https://www.vmware.com/topics/glossary/content/application-security. Luettu: 10.2.2020.

Wodehouse, C a. 2016. Web Development Languages 101. Luettavissa: https://www.upwork.com/hiring/development/web-development-languages-101/. Luettu: 8.10.2019.

Wodehouse, C b. 2015. Front-End Web Development: Client-Side Scripting & User Experience. Luettavissa: https://www.upwork.com/hiring/development/how-scripting-languages-work/. Luettu: 15.10.2019.

W3Schools a. What is HTML? Luettavissa: https://www.w3schools.com/whatis/whatis_html.asp. Luettu: 30.9.2019.

W3Schools b. What is CSS? Luettavissa: https://www.w3schools.com/whatis/whatis_css.asp. Luettu: 30.9.2019.

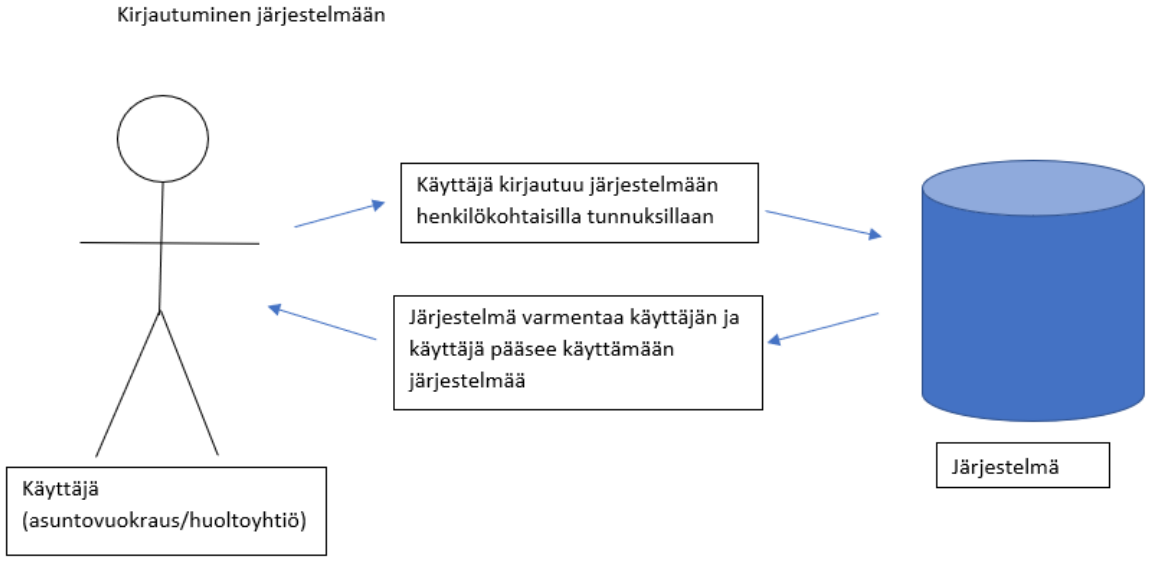
Liitteet

Liite 1. Vaatimusluettelo

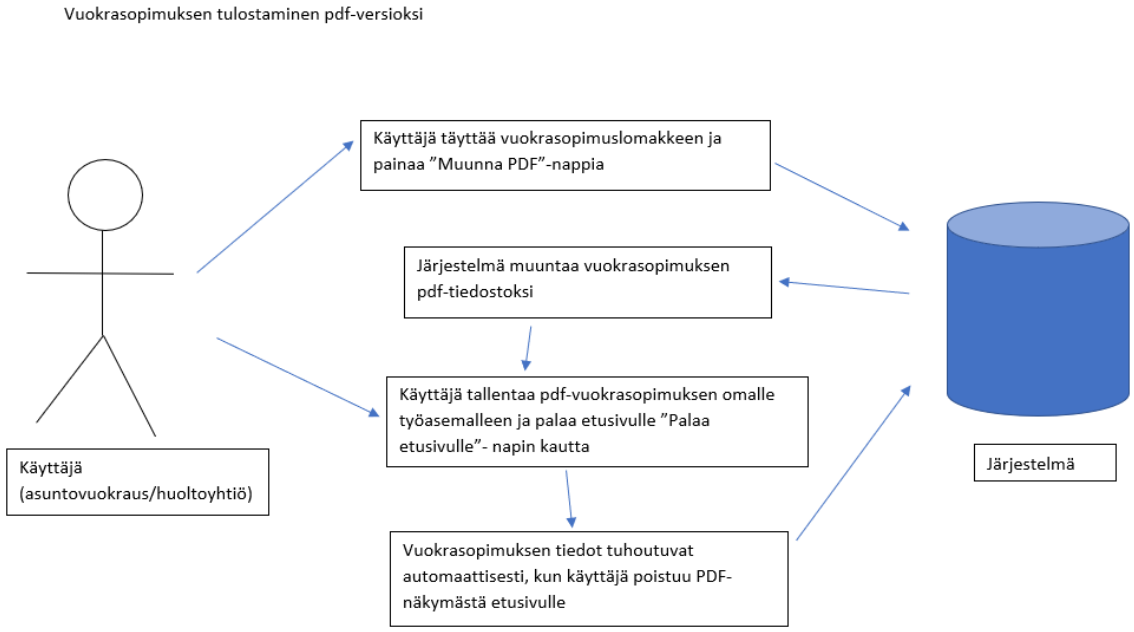
Tärkeys	Tunnus (ID)	Vaatimus	Vaatimuksen esittäjä	Päivämäärä	Tärkeys	Perustelu	Toimittajan kommentit
1.	TT2, NT2	Samat ehdot vuokrasopimuksissa (Toiminnallinen vaatimus)		15.1.2020	Pakollinen Priority !!!	Vuokrasopimusten ehdot tulee olla samansisältöiset kaikkien autopaikkoja vuokraavien tahojen osalta.	
2.	TT2, NT2	Erityisehtojen liittäminen (Toiminnallinen vaatimus)		15.1.2020	Pakollinen Priority !!!	Mikäli taloyhtiöön kohdistuu erityisehtoja, tulee ne voida liittää osaksi vuokrasopimusta.	
3.	NT2, TT2	Tietoturva (Ei-toiminnallinen vaatimus)		15.1.2020	Pakollinen Priority !!!	Koska järjestelmä käsittää käyttäjiin sekä taloyhtiöön kohdistuvia tietoja, tulee järjestelmän tietoturvaominaisuudet varmistaa.	
4.	TT2	Järjestelmän käyttö (Ei-toiminnallinen vaatimus)		15.1.2020	Pakollinen Priority !!!	Järjestelmää tulee voida käyttää suosituimmilla selaimilla käyttäjän laitteesta riippumatta, poislukien mobiililaitteet.	
5.	NT1, TT1	Päivitetyt sopimusehdot (Toiminnallinen vaatimus)		15.1.2020	Pakollinen Priority !!	Vuokrasopimusten ehdot tulee olla päivitetty ja reaaliajassa kaikkien autopaikkoja vuokraavien tahojen ulottuvilla.	
6.	TT2	Lisätiedot kohteesta järjestelmässä (Toiminnallinen vaatimus)		15.1.2020	Pakollinen Priority !!	Taloyhtiön kohdalle tulee voida lisätä tärkeää tietoa esim. tekstikentäksi.	
7.	TT1, TT2	Vuokrasopimuksen tulostaminen pdf-versioksi (Toiminnallinen vaatimus)		15.1.2020	Pakollinen Priority !!	Vuokrasopimus tulee voida tulostaa järjestelmästä pdf-versioksi siten, että vuokrasopimus voidaan lähettää allekirjoitettavaksi sekä sähköisesti että tulostaa paperiversioksi autopaikan vuokraajan toimesta.	

Liite 2. UML-käyttötapauskaaviot

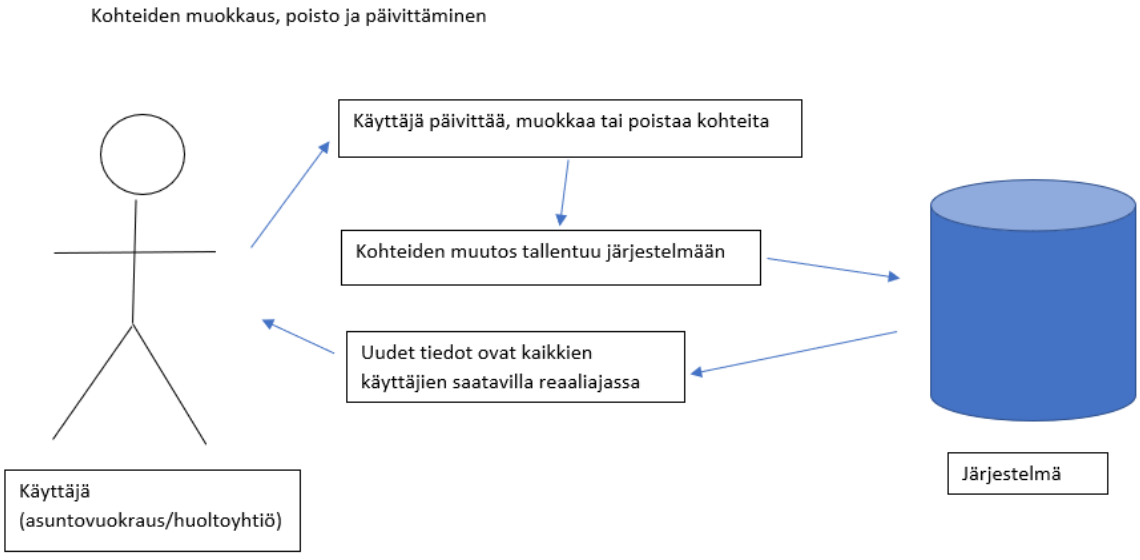
Käyttötapauskaavio 1. Käyttäjä kirjautuu järjestelmään henkilökohtaisilla tunnuksillaan. Järjestelmä varmentaa käyttäjän ja käyttäjä pääse käyttämään järjestelmää.



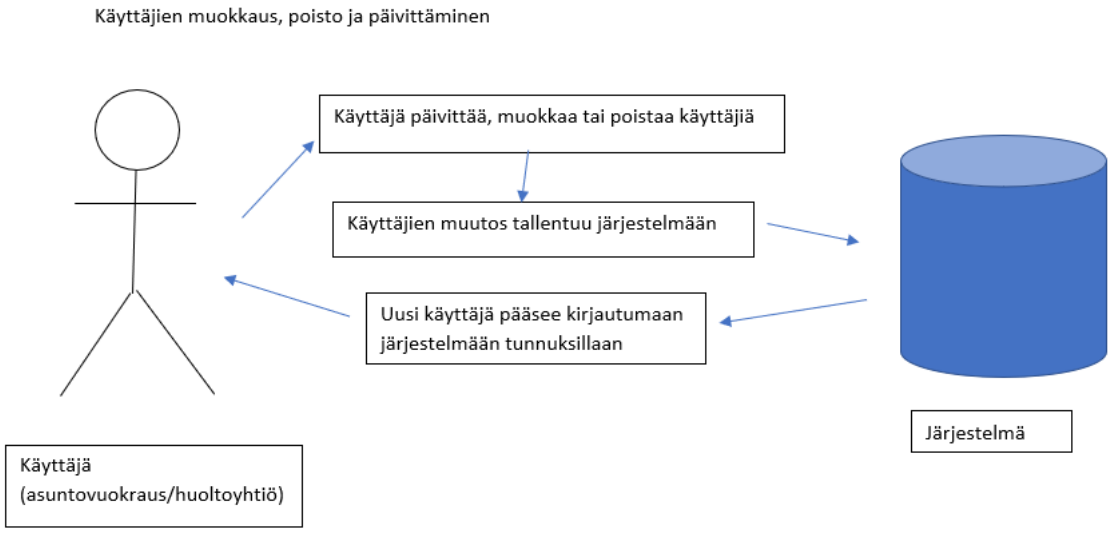
Käyttötapauskaavio 2. Käyttäjä täyttää vuokrasopimuslomakkeen, muuntaa sen PDF-tiedostoksi ja tallentaa pdf-vuokrasopimuksen omalle työasemalleen. Vuokrasopimus ja sen tiedot tuhoutuvat automaattisesti, kun käyttäjä poistuu pdf-tiedoston tulostamissivulta.



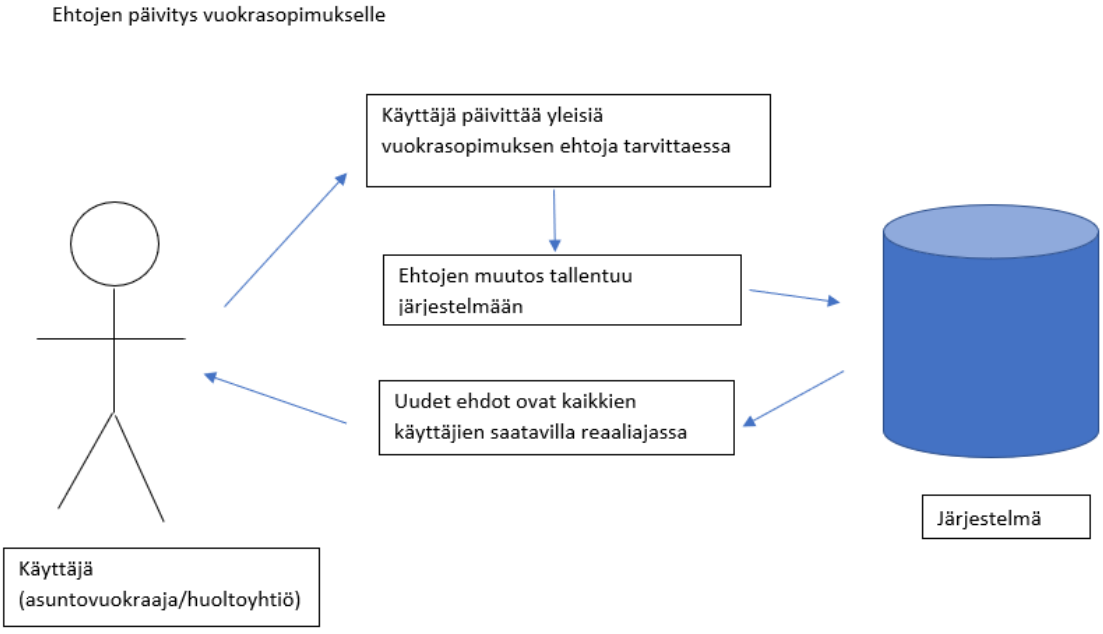
Käyttötapauskaavio 3. Käyttäjä voi päivittää, muokata tai poistaa kohteita tarvittaessa käyttöliittymän kautta. Muutetut tai poistetut tiedot tallentuvat automaattisesti järjestelmään.



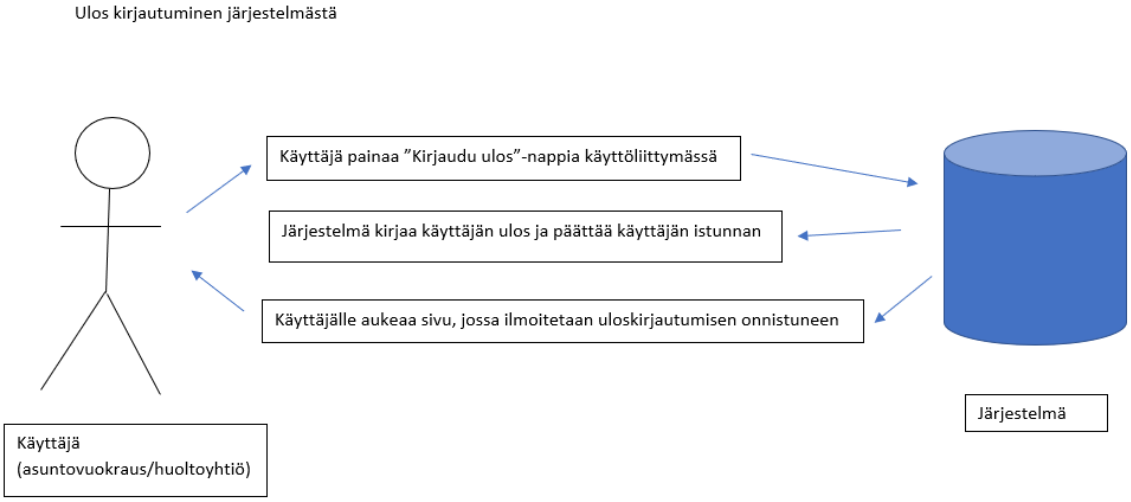
Käyttötapauskaavio 4. Käyttäjä voi päivittää, muokata tai poistaa käyttäjien tietoja tarvittaessa käyttöliittymän kautta. Muutetut tai poistetut tiedot tallentuvat automaattisesti järjestelmään.



Käyttötapauskaavio 5. Käyttäjä voi muokata yleisiä sopimusehtoja tarvittaessa käyttöliittymän kautta. Muutetut ehdot tallentuvat automaattisesti järjestelmään.

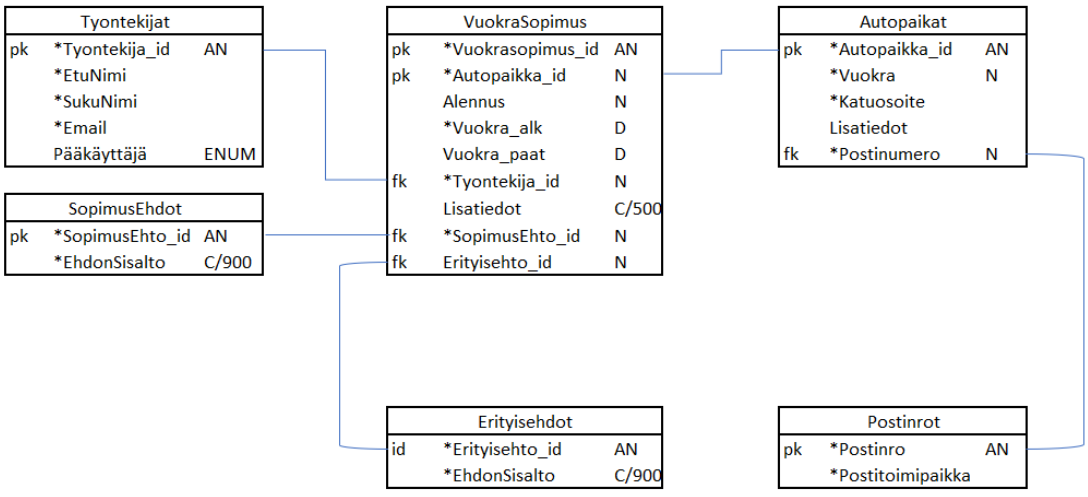


Käyttötapauskaavio 6. Käyttäjä kirjautuu ulos järjestelmästä Kirjaudu ulos-napin kautta. Järjestelmä vahvistaa uloskirjautumisen, lopettaa käyttäjän istunnan ja avaa uloskirjautumissivun, jossa on vahvistusviesti onnistuneesta uloskirjautumisesta.



Liite 3. Tietokannan relaatiomalli

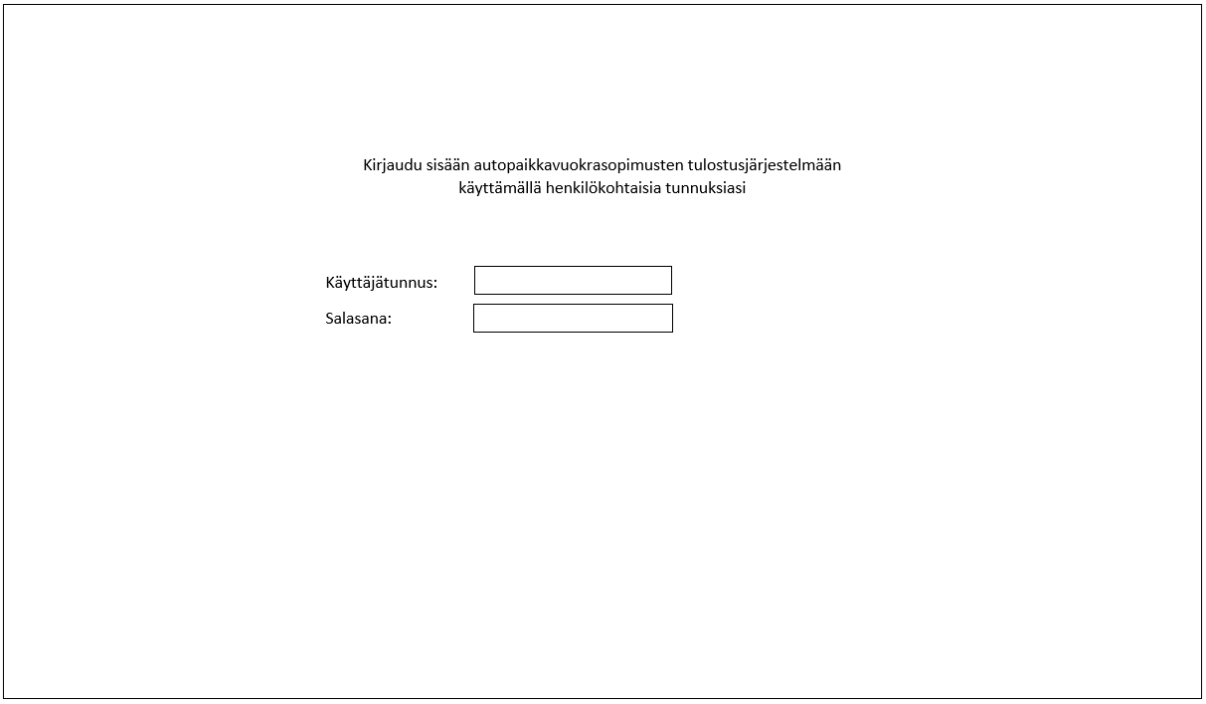
Kuvassa näkyvä relaatiomalli on tehty Excelin avulla. Tähdellä merkityt tiedot ovat pakollisia tietoja, jotka tulee merkitä kohteita, sopimusehtoja tai käyttäjiä muokatessa. Kuvassa näkyy myös tietokantaulujen väliset relaatiot.



*Pakollinen tieto

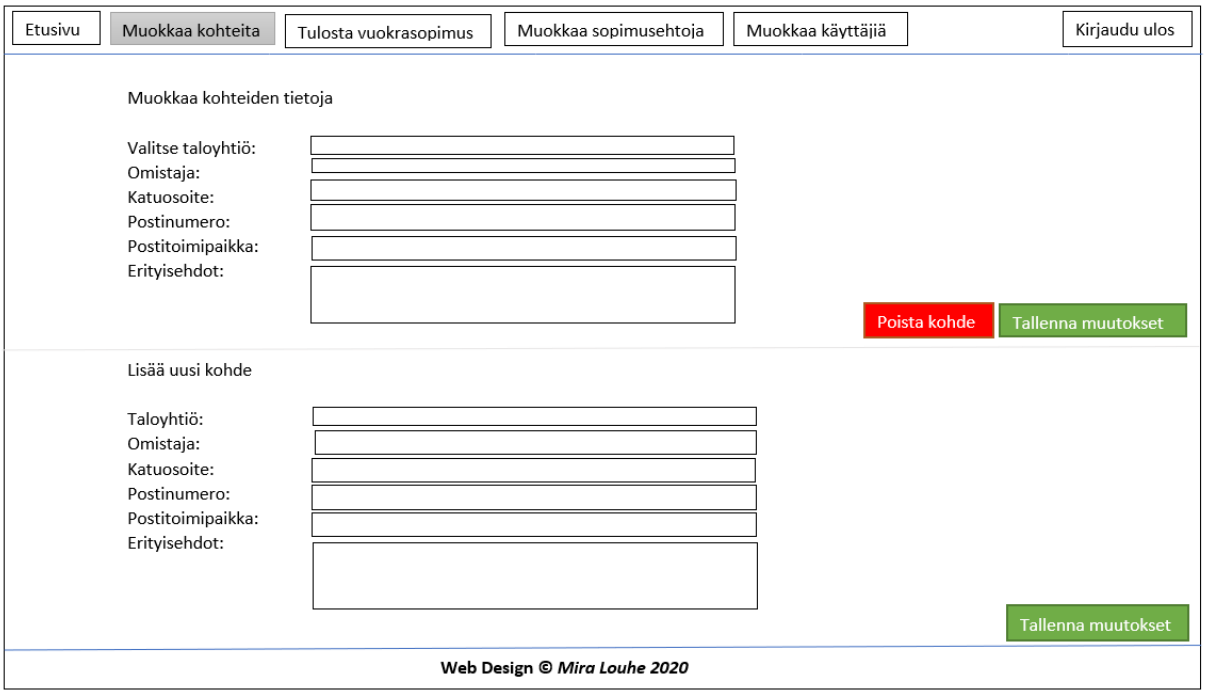
Liite 4. Käyttöliittymäluonnokset

Luonnos 1. Käyttäjä kirjautuu sisään järjestelmään omilla henkilökohtaisilla tunnuksillaan.

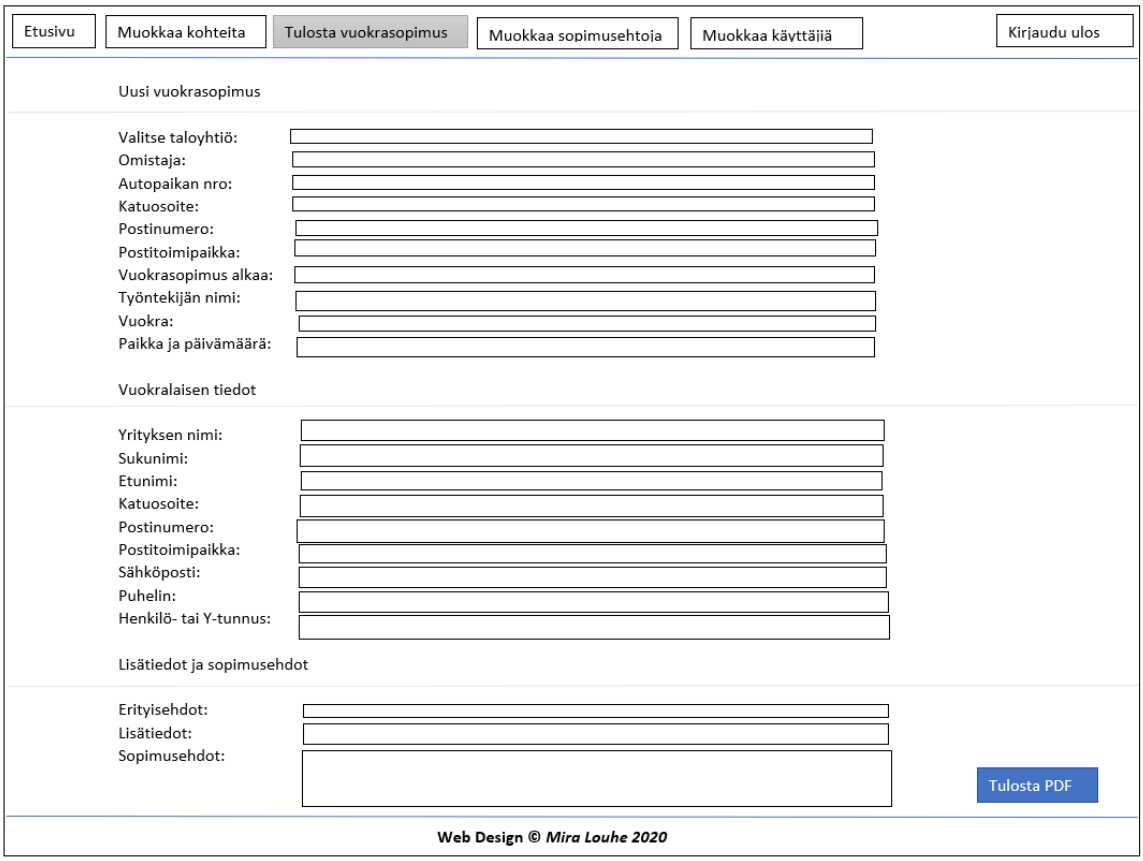


Luonnos 2. Sisäänkirjautumisen jälkeen käyttäjä siirtyy etusivulle. Käyttäjä voi valita haluamansa toiminnon yläpalkin linkeistä tai etusivun linkeistä, jotka näkyvät kuvassa sinisellä.

Luonnos 3. Käyttäjä voi muokata, lisätä tai poistaa kohteita tarvittaessa.

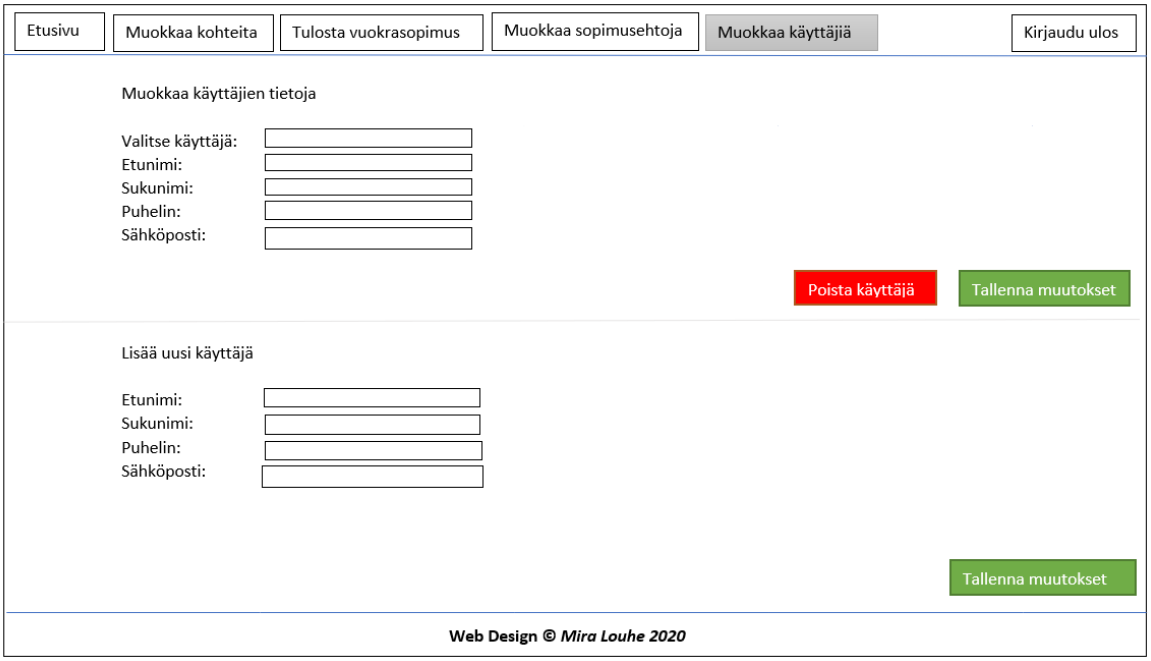


Luonnos 4. Luonnos vuokrasopimuslomakkeesta.

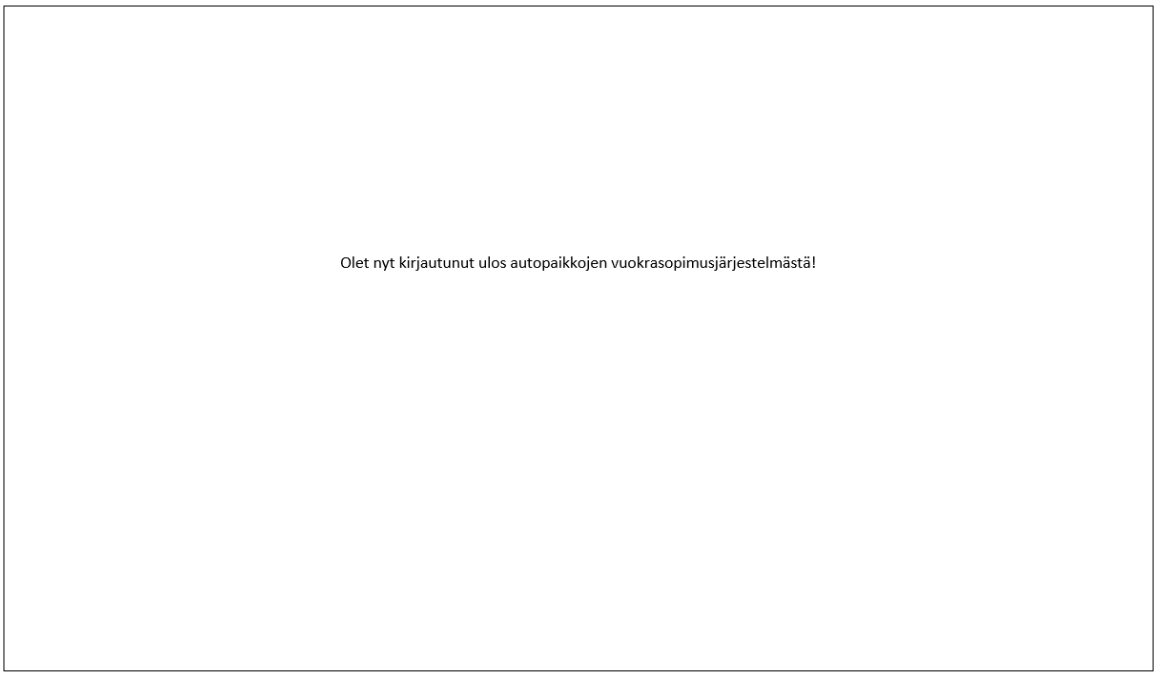


Luonnos 5. Käyttäjä voi tarvittaessa muokata yleisiä sopimusehtoja, jotka tulostetaan automaattisesti kaikille vuokrasopimuksille.

Luonnos 6. Käyttäjä voi tarvittaessa muokata käyttäjien tietoja tai poistaa sekä lisätä käyttäjiä.



Luonnos 7. Järjestelmä ilmoittaa käyttäjälle uloskirjautumisen onnistuneen, kun käyttäjä kirjautuu ulos järjestelmästä.



Liite 5. Testitapausten kuvaukset

Testitapaus 1. Yleisten sopimusehtojen päivitys.

Testitapauslomake 1

Testitapauksen tunnus: TC1 Testaaja: Mira Louhe
Pvm: 1.4.2020

Käyttötapaus (tai -tapaukset): Yleisten sopimusehtojen päivitys

Askel	Kuvaus	Odotettu tulos	Poikkeamat	OK/Hylätty
1	Käyttäjä kirjautuu järjestelmään.	Käyttäjä syöttää käyttäjätunnuksen ja salasanan. Siirrytään yleisten käyttäjätunnusten avulla järjestelmään.	Poikkeus 1: Käyttäjä syöttää väärän salasanan ja/tai väärän käyttäjätunnuksen. Käyttäjä ei pääse kirjautumaan järjestelmään.	OK
2	Käyttäjä klikkaa Muokkaa sopimusehtoja-linkkiä.	Siirrytään etusivulta yleisten sopimusehtojen muokkauksen sivulle. Avautuu Yleiset sopimusehdot- näkymä, jossa on tekstikentässä sopimusehdot.		OK
3	Käyttäjä muokkaa sopimusehtoa.	Käyttäjä muokkaa valitsemaansa kohtaa sopimusehdoissa.		OK
4	Käyttäjä tallentaa muutokset.	Käyttäjä klikkaa vihreää ”Tallenna muutokset” painiketta sivun vasemmasta alalaidasta. Tämän jälkeen tulee viesti sivun ylälaitaan, jossa lukee ”Muutokset tallennettu!”		OK
6	Uudet sopimusehdot.	Päivitetyt sopimusehdot ovat kaikkien käyttäjien saatavilla reaaliajassa.		OK

Vaatimukset: TT1 ja NT1 – Päivitetyt sopimusehdot

Testitapaus 2. Vuokrasopimuksen tulostaminen.

Testitapauslomake 2

Testitapauksen tunnus: TC2 Testaaja: Mira Louhe
Pvm: 1.4.2020

Käyttötapaus (tai -tapaukset): Vuokrasopimuksen tulostaminen

Askel	Kuvaus	Odotettu tulos	Poikkeamat	OK/Hylätty
1	Käyttäjä siirtyy vuokrasopimuksen tulostamiseen.	Klikataan sivun ylälaidassa olevasta navigaatiosta kohtaa ”Tulosta vuokrasopimus”.		OK
2	Käyttäjä täyttää vuokrasopimuksen.	Siirrytään vuokrasopimuksen sivulle. Valitaan taloyhtiö, johon vuokrasopimusta tehdään. Muuttumattomat taloyhtiön tiedot sekä vuokrasopimuksen yleiset sopimusehdot avautuvat syöttökenttiin. Täytetään manuaalisesti muut tiedot sopimukselle.		OK
3	Käyttäjä tulostaa vuokrasopimuksen.	Käyttäjä klikkaa sivun vasemmasta alalaidasta sinistä ”Tulosta PDF”-painiketta. Järjestelmä avaa automaattisesti uuden sivun, jossa on PDF-näkymässä vuokrasopimus. Käyttäjä tallentaa vuokrasopimuksen omalle käyttöasemalleen selaimen omaa PDF-tulostusta hyödyntäen.		OK
4	Tulostus valmis.	Käyttäjä klikkaa sivun alalaidassa keskellä olevaa sinistä ”Palaa etusivulle”-painiketta ja siirtyy takaisin etusivulle.		OK

Vaatimukset:
TT1, TT2: Vuokrasopimuksen tulostaminen PDF-versioksi

Testitapaus 3. Kohteiden muokkaus.

Testitapauslomake 3

Testitapauksen tunnus: TC3 Testaaja: Mira Louhe
Pvm: 1.4.2020

Käyttötapaus (tai -tapaukset): Kohteiden muokkaus

Askel	Kuvaus	Odotettu tulos	Poikkeamat	OK/Hylätty
1	Käyttäjä siirtyy kohteiden muokkaukseen.	Klikataan sivun ylälaidassa olevasta navigaatiosta kohtaa ”Muokkaa kohteita”.		OK
2	Käyttäjä muokkaa kohteita.	Muokkaa kohteiden tietoja-näkymässä avautuu automaattisesti kohteen tiedot syöttökenttiin. Käyttäjä valitsee alasvetovalikosta kohteen ja erityisehdot yhdelle taloyhtiölle. Käyttäjä painaa vihreää painiketta ”Tallenna muutokset”. Sivun ylälaitaan ilmestyy teksti ”Muutokset tallennettu!”. Muuttuneet tiedot löytyvät tarkistettaessa alasvetovalikosta. Käyttäjä valitsee kohteen alasvetovalikosta ja klikkaa punaista painiketta ”Poista kohde”. Sivun ylälaitaan ilmestyy teksti ”Kohde poistettu!”. Poistettua kohdetta ei löydy taloyhtiöiden alasvetovalikosta.		OK
3	Käyttäjä lisää kohteen.	Käyttäjä täyttää kohtaan ”Lisää uusi kohde” uuden taloyhtiön tiedot ja klikkaa sivun alalaidassa olevaa painiketta ”Tallenna muutokset”. Sivun ylälaitaan tulee viesti ”Muutokset tallennettu!”. Uuden taloyhtiön tiedot löytyvät taloyhtiön alasvetovalikosta.		OK

Vaatimukset
NT2, TT2: Erityisehtojen liittäminen ja Lisätiedot kohteesta järjestelmässä