Lasse Kelottijärvi

SMARTPICKBOX EXTRANET -VERKKOSIVUN TOTEUTUS

SMARTPICKBOX EXTRANET -VERKKOSIVUN TOTEUTUS

Lasse Kelottijärvi
Opinnäytetyö
Kevät 2020
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehitys

Tekijä: Lasse Kelottijärvi
Opinnäytetyön nimi suomeksi: Smartpickbox Extranet -verkkosivun toteutus
Opinnäytetyön nimi englanniksi: Smartpickbox Extranet Website Implementation
Työn ohjaaja: Eino Niemi
Työn valmistumislukukausi ja -vuosi: Kevät 2020
Sivumäärä: 29

Opinnäytetyön aiheena oli noutoautomaatin lokeroiden varaussivuston toteutus. Toimeksiantajana työlle toimi SHJ Group Oy, joka valmistaa Smartpickbox-noutoautomatteja. Tavoitteena oli toteuttaa verkkosivut, joiden kautta asiakas voi varata itselleen noutoautomaatin lokeron käyttöönsä. Näitä varauksia pystyy muokkaamaan ja poistamaan tarvittaessa. Lisäksi ylläpitäjillä on mahdollisuus luoda uusia automaattinäkymiä sivustolle.

Sivusto toteutettiin CodeIgniter-sovelluskehyksellä ja rajapinta Lumen-mikro-sovelluskehyksellä. Rajapinta vastaa tässä työssä kaikista tietokantakyselyitä. Tietokantajärjestelmänä toimii MariaDB.

Sivusto saatiin tehtyä projektin tavoitteiden mukaisesti. Se sisältää sisäänkirjautumisen sivustolle, noutoautomaatin lokeroiden tarkastelunäkymän, vapaana olevan lokeron varaamisen, oman varauksen muokkaamisen ja poistamisen sekä ylläpitäjille uusien käyttäjien rekisteröimisen ja automaattinäkymien luonnin. Sivusto tarvitsisi enää yhteyden automaattiin, jotta tämä sivusto olisi valmis.

Asiasanat: verkko-ohjelmointi, verkkosivut, PHP

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Software Development

Author: Lasse Kelottijärvi
Title of thesis: Smartpickbox Extranet Website Implementation
Supervisor: Eino Niemi
Term and year when the thesis was submitted: Spring 2020
Pages: 29

This thesis is about pickup locker reservation site. Thesis was commissioned by SHJ Group Oy which makes Smartpickboxses, fully automated system for picking up or leaving deliveries. Goal was to make website where customers can reserve locker from cabinet. Reservations can be edited or removed. Administrators can make new cabinet views.

Website was created with CodeIgniter framework and API with Lumen microframework. In this project API is responsible for all database queries. MariaDB works for database server.

Objectives for this project were accomplished. Website has user logging, cabinet review view, locker reservation, own reservation edit and remove features and new users registration and cabinet view creator for administrators. Only connection between cabinet and website is needed until this website is ready.

Keywords: web development, website, PHP

ALKULAUSE

Haluan kiittää SHJ Group Oy:tä mahdollisuudesta toteuttaa opinnäytetyö heidän kanssaan. Lisäksi haluan kiittää Oulun ammattikorkeakoulun lehtori Eino Niemeä opinnäytetyön ohjauksesta.

Oulussa 23.3.2020

Lasse Kelottijärvi

SISÄLLYS

TIIVISTELMÄ 3
ABSTRACT 4
ALKULAUSE 5
SISÄLLYS 6
SANASTO 7
1 JOHDANTO 8
2 OHJELMISTOT 9
2.1 CodeIgniter 9
2.2 Lumen 10
2.3 Composer 10
2.4 MariaDB 11
3 SIVUSTON TOTEUTUS 12
3.1 Suunnittelu 12
3.2 Toteutus 15
3.3 Viimeistely 20
4 SMARTPICKBOX EXTRANET 21
4.1 Kirjautuminen 21
4.2 Lokeron varaus 22
4.3 Ylläpito 23
5 JATKOKEHITYS 27
6 YHTEENVETO 28
LÄHTEET 29

SANASTO

CSS	(Cascading Style Sheets) Tyylitiedostot ulkoasun määrittämiseen.
HTML	(Hypertext Markup Language) Kuvauskieli verkkosivujen tekoon.
JavaScript	Ohjelmointikieli verkkosivujen tekoon.
PHP	(Hypertext Preprocessor) Ohjelmointikieli verkkosivujen toiminnallisuuden tekoon.
Rajapinta	Mahdollistaa ohjelmistojen välisen kommunikoinnin.
Rajapintakysely	Hakee tai lähettää tietoja rajapintaan.
Sovelluskehys	Ohjelmiston runko, jonka päälle ohjelmisto rakennetaan.
Token	Yksilöllinen tunniste, jolla todennetaan käyttäjä.

1 JOHDANTO

Työssä tehdään verkkosivusto noutoautomaatin lokeroiden varaamiseen. Projekti sisältää verkkosivuston käyttöliittymän ja tietokannan sekä rajapinnan sivuston ja noutoautomaattien tietokannan väliseen kommunikointiin.

Työn tilaaja SHJ Group Oy on oululainen Smartpickbox-noutoautomaatteja valmistava yritys. Yrityksen noutoautomaatteja löytyy rautakaupoista, apteekeista ja autoliikkeistä.

Automaatti voi toimia esimerkiksi autoliikkeessä. Autoliikkeen asiakas varaa netistä autolleen huollon ja saa puhelimeen tekstiviestinä koodin, jolla hän pääsee jättämään auton avaimet automaattiin. Automaatti voi olla toimipisteen tuulikaapissa, jonne asiakas pääsee ympäri vuorokauden tekstiviestillä saamallaan koodilla. Kun asiakas syöttää automaattiin koodin, luukku avautuu, asiakas jättää avaimet lokeroon ja sulkee luukun. Lokerosta voi löytyä sijaisauton avaimet, mikäli asiakas on varannut sijaisauton huollon ajaksi. Kun huolto on valmis, asiakas saa tekstiviestillä noutokoodin, jolla saa automaatista oman autonsa avaimet. Maksaminen voi tapahtua automaatista löytyvällä maksupäätteellä.

Noutoautomaattien ideana on sujuvoittaa tuotteiden ja palvelujen toimitusta. Automaatit toimivat ympäri vuorokauden, jolloin sieltä voi noutaa tilauksen silloin kun itselle sopii. Esimerkiksi autoliikkeessä ei tarvitse jonottaa, vaan asioinnin voi hoitaa nopeasti automaatilla, myös aukioloaikojen ulkopuolella.

Työssä kehitetään järjestelmä, joka mahdollistaa yhden automaatin käytön useiden kumppanien kanssa. Yritykset voivat käyttää tarvittaessa noutoautomaattia tuotteensa toimitukseen asiakkaalle. Nykyinen järjestelmä mahdollistaa vain sellaisen noutoautomaatin käytön, jonka yritys on hankkinut itselleen.

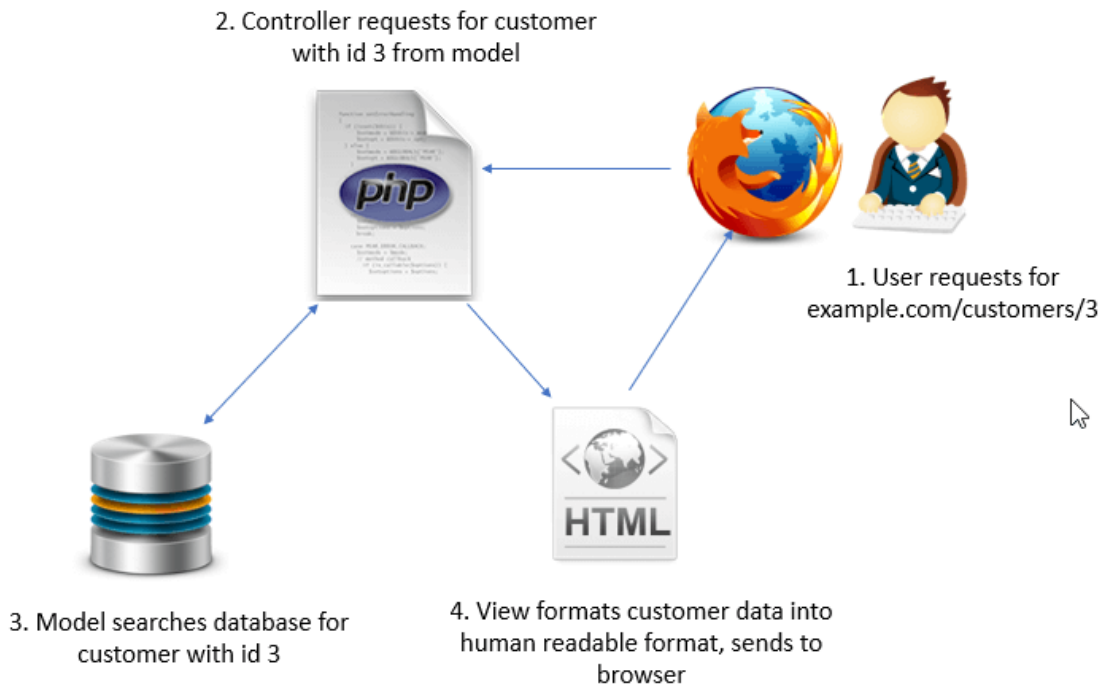
2 OHJELMISTOT

Tässä projektissa tarvitaan ohjelmistoja sivuston ja rajapinnan luomiseen ja ylläpitoon. Sivusto päädyttiin rakentamaan CodeIgniter-sovelluskehyksen avulla ja tietokanta MariaDB tietokantajärjestelmällä, koska ne ovat olleet muissakin yrityksen projekteissa käytössä. Rajapinnan luomiseen valikoitui Lumen-mikro-sovelluskehys, koska se soveltuu projektin tarpeisiin kevytrakenteisuudellaan ja nopeudellaan. Composer-hallintatyökalulla huolehditaan CodeIgniteriin ja Lumeniin lisätyistä PHP-kirjastoista.

2.1 CodeIgniter

CodeIgniter on ilmainen sovelluskehys web-sovellusten luomiseen PHP-ohjelmointikielellä. CodeIgniter sisältää monia kirjastoja, jotka helpottavat yleisten toimintojen luomisessa, kuten tietokantakyselyjen toteuttamisessa. Sovelluskehys sisältää myös yksinkertaisen käyttöliittymän ja loogisen rakenteen kirjastojen käyttöönotossa. (1.)

CodeIgniter käyttää MVC-kehitysmallia (kuva 1), jossa jaetaan sovellus Model-, View- ja Controller-luokkiin. Model-luokka vastaa tiedon käsittelystä, joko tietokantakomentojen tai jonkin palvelun ajamisella, joka käsittelee tiedot. Tässä projektissa Model-luokassa kutsutaan rajapinnan funktioita, jotka käsittelevät tiedon ja ovat sitä kautta yhteydessä tietokantaan. Controller-luokka siirtää tiedot Model- ja View-luokan välillä. View-luokka esittää tiedot käyttäjälle. Web-sovellukset yleensä sisältävät View-luokassa HTML:llä, CSS:llä ja JavaScriptillä tehtyjä toteutuksia. (2.)



KUVA 1. MVC-mallin toiminta (2)

2.2 Lumen

Lumen on Laravel-sovelluskehyksestä kehitetty kevyempi ja nopeampi versio. Se on tarkoitettu pääsääntöisesti mikropalveluiden ja rajapintojen toteuttamiseen, kun Laravel on tarkoitettu suuremmille sovelluksille. Mikropalvelut ovat pieniä, irrallisia prosesseja, joka kommunikoivat mahdollisesti suuremman ohjelmiston kanssa. (4; 5.) Rajapinta puolestaan on ohjelmiston välittäjä, joka mahdollistaa sovellusten välisen kommunikoinnin (3). Tässä projektissa Lumen toimii rajapintana, joka tekee kaikki tietokantapyynnöt. Tietokannan päivittäminen ja tiedon saaminen sivustolle tapahtuvat siis rajapinnan kautta. Tätä voi käyttää myös jatkokehityksessä, kun kommunikoidaan noutoautomaatin kanssa.

2.3 Composer

Composer on hallintatyökalu, joka huolehtii projektiin lisätyistä kirjastoista. Sen kautta voi helposti lisätä ja ylläpitää haluttuja PHP-ohjelmointikielellä tehtyjä kirjastoja. Composer valitsee projektiin sopivan kirjaston version ja lataa sen projektin kansioon, minne muutkin Composerin kautta ladatut kirjastot ladataan. Nämä kirjastot on helppo päivittää kerralla yhdellä komennolla Composerin

kautta. Tässä projektissa CodeIgniterin ja Lumenin kanssa on käytössä Composer, jonka kautta huolehditaan suurinta osa käytetyistä kirjastoista. (6.)

2.4 MariaDB

MariaDB on avoimeen lähdekoodiin perustuva tietokantajärjestelmä, joka toimii SQL-kyselykielen avulla (7). SQL-kyselyillä voidaan hakea, lisätä, muokata ja poistaa tietoa tietokannasta (8).

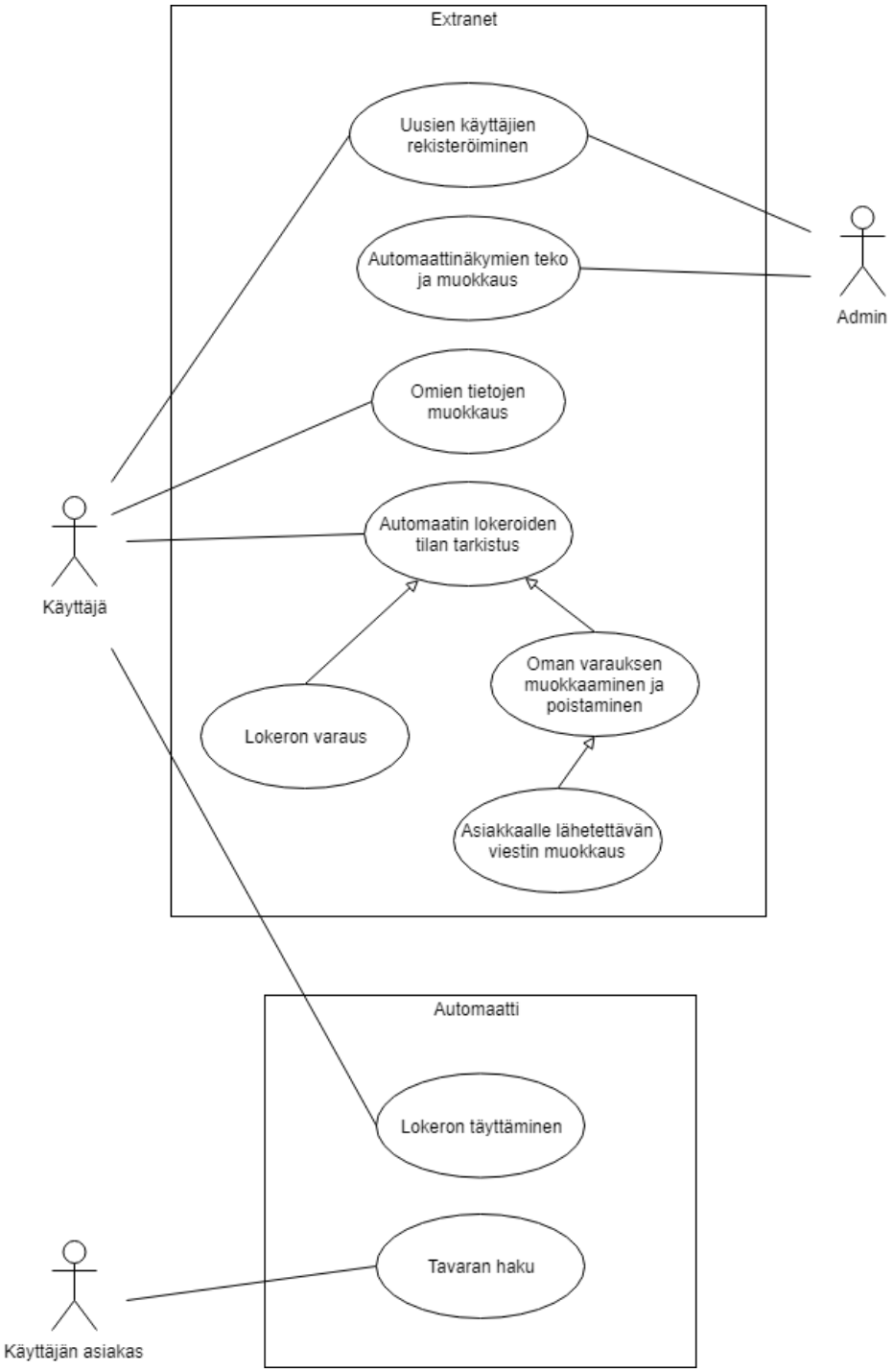
3 SIVUSTON TOTEUTUS

Opinnäytetyössä toteutettiin Smartpickbox Extranet -verkkosivusto. Sivustolla rekisteröidyt käyttäjät varaavat Smartpickbox-noutoautomaatista lokeroita omaan käyttöön ja ylläpitäjät luovat ja muokkaavat noutoautomaattinäkymiä sivustolle.

3.1 Suunnittelu

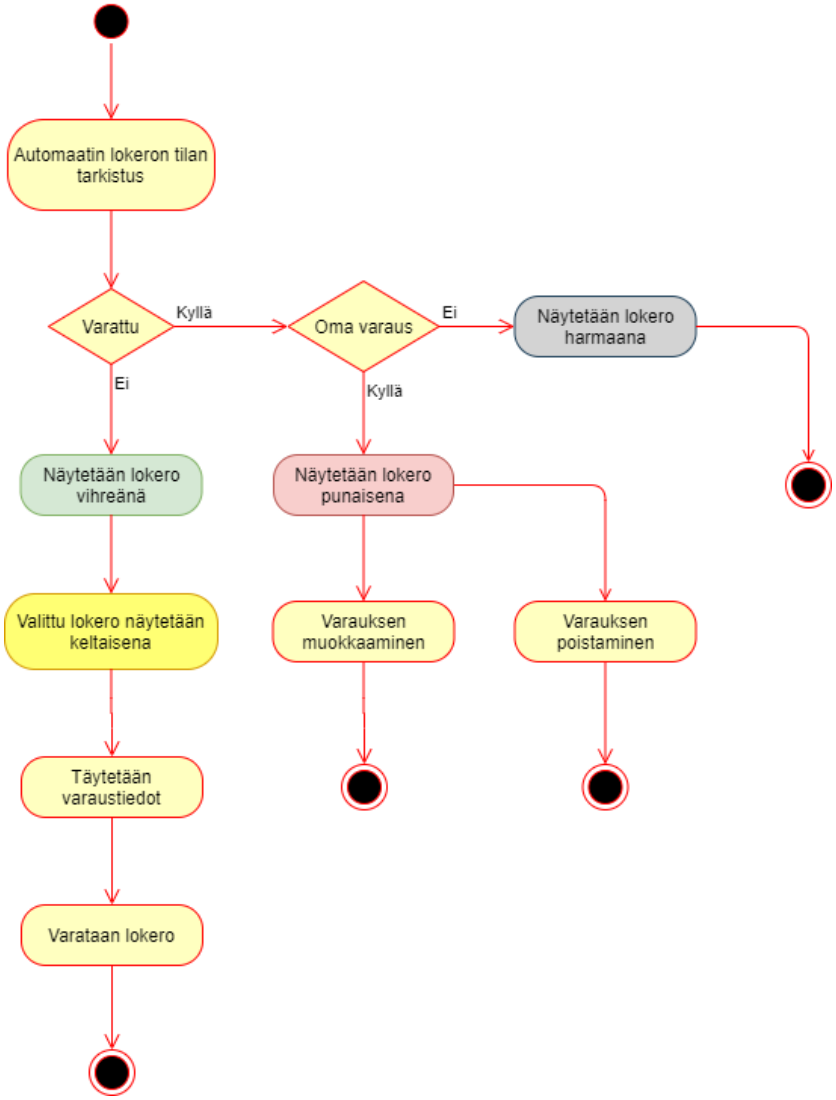
Työ aloitettiin suunnittelemalla projektin sisällön toteutus yrityksen työnantajan ja ohjelmistosuunnittelijan kanssa. Ominaisuuksia listattiin ranskalaisin viivoin työn määrittelyyn. Käyttöliittymää suunniteltiin tekemällä käyttötapauskaavio. Lokeroiden varausprosessin suunnittelussa tehtiin vuokaavio. Tietokantaa suunniteltiin tekemällä tietokantakaavio. Toteutukselle tehtiin myös alustava aikataulu, johon yritettiin hieman erotella työn eri vaiheita. Näihin kuuluivat suunnittelu, kirjautumisen toteutus, automaattinäkymän toteutus, lokeroiden tilan vaihto, rajapinnan toteutus, käyttäjän tietojen muokkaus ja työn viimeistely.

Käyttötapauskaaviossa (kuva 2) on esitetty sivuston toiminta yksinkertaistetusti liittäen noutoautomaatti mukaan. Siinä käyttäjä on sivustolle ylläpitäjän rekisteröimä noutoautomaatin lokeroita omille lähetyksilleen varaava henkilö. Extranetin puolella käyttäjä voi rekisteröidä uusia käyttäjiä, muokata omia tietojaan, tarkastella automaatin lokeroiden tilaa, varata lokeroita ja muokata ja poistaa omia varauksia. Automaatin puolella käyttäjä voi täyttää varaamansa lokeron, jonka käyttäjän asiakas hakee. Ylläpitäjä voi myös Extranetin puolella rekisteröidä uusia käyttäjiä sekä tehdä uusia automaattinäkymiä ja muokata niitä.



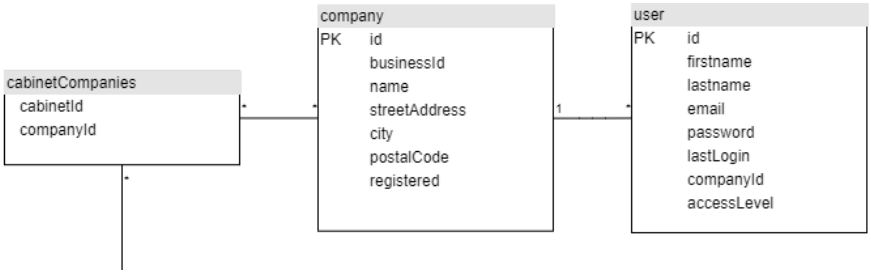
KUVA 2. Sivuston ja sen toimintaan liittyvän automaatin käyttötapauskaavio

Noutoautomaatin lokeron varausprosessin vuokaaviossa (kuva 3) käydään läpi, mitä lokeroille voi tehdä ja miltä ne näyttävät eri tilanteissa visuaalisesti. Varaamattomat lokerot ovat vihreitä, omat varatut lokerot punaisia ja muiden varaamat lokerot harmaita. Valittu vapaa lokero näytetään keltaisena.



KUVA 3. Vuokaavio lokeroiden varausprosessista

Tietokantakaaviosta (kuva 4) näkee taulujen nimet ja sarakkeet, joita ne sisältävät. Myös perusavain eli primary-key (PK) ilmoitetaan sarakkeessa. Tämä on taulukon yksilöivä avain, joka yksilöi esimerkiksi jokaisen käyttäjän omalla id-numerolla. Taulujen välinen suhde ilmoitetaan yhdistämällä taulut viivalla ja merkkaamalla viivan päihin taulujen rajoitteet toista taulua kohti. Jos taululla voi olla vain yksi toisen taulun välinen yhteys, merkataan numero yksi. Esimerkkinä käyttäjään (user) voidaan liittää vain yksi yritys (company). Mikäli taululla voi olla monta yhteyttä toiselle taululle, merkataan tähti. Esimerkkinä yrityksellä (company) voi olla monta käyttäjää (user). (8.)



KUVA 4. Osa tietokantakaaviosta

3.2 Toteutus

Työn toteutus aloitettiin kirjautumissivun tekemisellä. Myös tietokanta tehtiin senhetkisen suunnittelun pohjalta, jotta pystyttiin hakemaan käyttäjien ja yritysten tiedot kirjautumisen testausta varten. Kirjautuminen toimii siten, että kun sähköpostikenttään ja salasanakenttään syöttää tiedot ja painaa kirjaudu sisään -painiketta (kuva 5), kirjautumissivun Controller-luokan kirjautumisfunktiota (kuva 6) kutsutaan syötetyillä arvoilla. Tämä funktio puolestaan kyselee Model-luokan funktiota (kuva 7). Model-luokasta tehdään kaikki tietokantahaut rajapintakyselyillä. Mikäli annettu sähköpostiosoite löytyy tietokannasta ja sen salasanan tiiviste (hash) täsmää syötetyn salasanan kanssa, tallennetaan käyttäjän tiedot istuntoon (session) ja ladataan etusivu. Istunnon tarkoituksena on säilyttää tiedot turvallisesti käyttäjän koko istunnon ajan, sillä tietoja ei välitetä selaimelle, vaan ne säilytetään palvelimella (9). Jos syötetyt tiedot eivät täsmää, ladataan kirjautumissivu uudelleen.

```
<form method="post" action="<?php echo site_url('login/tryLogin');?>">
  <div class="w-50 mx-auto mb-2">
    <input name="email" type="email" class="form-control" placeholder="Sähköposti" onfocus="this.placeholder = ''" onblur="this.placeholder = 'Sähköposti'">
  </div>
  <div class="w-50 mx-auto mb-4">
    <input name="password" type="password" class="form-control" placeholder="Salasana" onfocus="this.placeholder = ''" onblur="this.placeholder = 'Salasana'">
  </div>
  <div class="w-50 mx-auto">
    <button type="submit" class="btn btn-block btn-success">Kirjaudu sisään</button>
  </div>
</form>
```

KUVA 5. Sivuston kirjautumiskentät View-luokassa

```
public function tryLogin()
{
  $data = $this->input->post();
  $res = json_decode($this->Loginmodel->checkLoginInfo($data));
  .
  .
  .
```

KUVA 6. Controller-luokan kirjautumisfunktio

```php
public function checkLoginInfo($reqdata)
{
  $email = $reqdata['email'];
  $password = $reqdata['password'];

  $data = array(
    'email' => $email,
    'password' => $password
  );

  return $this->APIPostRequest($data, 'api/checkLoginInfo');
}
```

KUVA 7. Model-luokan kirjautumisfuntio, joka kutsuu rajapintakyselyn tekevää funktiota

Seuraavana kehitettiin sivuston lähes jokaiselta sivulta löytyvä navigointipalkki. Toteutus on lista, johon voi helposti lisätä uusia sivuja. Tässä vaiheessa myös sivuston responsiivisuus otettiin huomioon. Silloin kun navigointipalkin sivustolinkit eivät enää mahdu jonossa näkymään, tilalle ilmestyy valikkopainike, jota klikkaamalla sivustolinkit aukeavat allekkain olevaan listaan. Käyttäjän käyttäjätason perusteella ladataan näkyviin vain sallitut sivustolinkit. Myös sivuston Controller-luokka ajaa sivun latautuessa funktion, joka tarkistaa, onko käyttäjällä oikeus päästä kyseiselle sivulle (kuva 8). Käyttäjä ohjataan etusivulle, mikäli hänellä ei ole tarpeellista käyttäjätasoa.

```php
    private function userPermission()
    {
      $level = $this->session->accessLevel;
      if($level > 2)
      {
        redirect('main');
      }
    }
```

KUVA 8. Käyttäjätason tarkastava funktio

Seuraavana siirryttiin tämän projektin varsinaisen tarkoituksen toteutukseen eli lokeronäkymän ja toiminnan tekoon. Lokeronäkymästä pyrittiin toteuttamaan mahdollisimman samantapainen kuin jo olemassa olevista noutoautomaateista löytyy. Sen pohjalta tietokantaan tallennettujen lokeron moduulin, numeron ja koon perusteella toteutetaan View-luokassa JavaScriptin puolella painikkeita lokeroiksi (kuva 9). Näille lokeroille tila saadaan myös tietokannasta ja sen perusteella värjätään vihreiksi vapaat lokerot, punaisiksi varatut lokerot, tässä tapauksessa omat varaukset, ja harmaaksi näyttölokerot, toisten käyttäjien varaamat

lokerot sekä pois kytketyt lokerot. Klikkaamalla vapaata tai omaa varattua lokeroa, eli valitsemalla lokeron, väri vaihtuu keltaiseksi. Vapaana olevan lokeron voi varata, kun klikkaa haluttua lokeroa ja kirjaa avautuvaan puhelinnumerokenttään numeron ja viestikenttään haluamansa tekstiviestin. Varauksen jälkeen sivu latautuu uudelleen ja näyttää varatun lokeron punaisena. Kun sitä klikkaa, näkee kirjatun puhelinnumeron ja viestin. Niitä voi muokata tai varauksen voi poistaa kokonaan. Varatun lokeroon voisi varauksen teon jälkeen käydä täyttämässä noutoautomaatilla. Varaukseen kirjattuun numeroon lähtisi kirjattu viesti noutokoodin kanssa, kun lokero suljettaisiin täyttämisen jälkeen. Tässä vaiheessa lisättiin toiminnot, joilla lokeron kokoa voi muokata sekä ottaa lokeron pois käytöstä. Nämä siirrettiin myöhemmin omalle automaatin muokkaussivulle.

```
for (var modul = 0; modul < moduleCount; modul++) {

  html += '<div id="moduleRow'+modul+'" class="moduleRow" style="width:'+viewSize+'%; margin-left:auto; margin-right:auto;">';
  for (i in json['cabinetLocker'])
  {
    var moduleNumber = json['cabinetLocker'][i].moduleNumber;
    var lockerId = json['cabinetLocker'][i].id;
    var lockerType = json['cabinetLocker'][i].type;
    var lockerNumber = json['cabinetLocker'][i].lockerNumber;

    //Get correct module lockers
    if(moduleNumber == modul)
    {
      //Do not show locker number if type is 0
      if(lockerType == 0)
      {
        var height = Number(json['cabinetLocker'][i].height);
        html += '<button id="lockerButton'+lockerId+'" data-module="'+moduleNumber+'" data-lockerid="'+lockerId+'"
                data-lockernumber="'+lockerNumber+'" data-lockerheight="'+height+'" data-lockertype="'+lockerType+'"
                name="lockerButton" class="lockerBox col-md-12 cabinetButtonNotInUse" style="height:'+ height +'px;">
                <div id="lockerNumber'+lockerId+'" class="lockerNumber"></div></button>';
      }
      else
      {
        var height = Number(json['cabinetLocker'][i].height);
        html += '<button id="lockerButton'+lockerId+'" data-module="'+moduleNumber+'" data-lockerid="'+lockerId+'"
                data-lockernumber="'+lockerNumber+'" data-lockerheight="'+height+'" data-lockertype="'+lockerType+'"
                name="lockerButton" class="lockerBox col-md-12" style="height:'+ height +'px;">
                <div id="lockerNumber'+lockerId+'" class="lockerNumber">'+lockerNumber+'</div></button>';
      }
    }
  }
  html += '</div>';
}
```

KUVA 9. Lokeroiden toteutus tietokantakyselyn tuloksella JavaScriptillä

Lokeroiden varaaminen ja varauksen poistaminen tallentuvat tietokantaan. Nämä tiedot näkyvät sivuston etusivulla sekä ylläpitäjän automaatin muokkaussivulla. Ylläpitäjä näkee kaikkien käyttäjien toiminnot, kun käyttäjä etusivullaan näkee vain omansa.

Tässä vaiheessa siirryttiin tutustumaan ja toteuttamaan rajapintaa Lumenilla. Sivuston Model-luokassa tehdään Guzzle-kirjastoa käyttäen kyselyjä rajapintaan. Kaikki aikaisemmin Model-luokassa tehdyt suorat tietokantakyselyt siirrettiin rajapinnan puolelle. Rajapinnalle otettiin käyttöön tokenit, jotka generoidaan kirjauduttaessa sivustolle. Token eli merkkijono lähetetään jokaisen rajapintakyselyn yhteydessä, sillä sen avulla tunnistetaan käyttäjä ja sallitaan rajapintakyselyjen teko. Näin varmistetaan, että kyselyt tekee sallittu käyttäjä. Token myös vanhenee ajan kuluessa. Sen voi asettaa vanhenemaan esimerkiksi tunnin kuluttua, jonka jälkeen sen avulla ei voi enää suorittaa kyselyjä. Rajapintakysely päivittää ja palauttaa tokenin sekä sen vanhenemisajan. Vanhenemisaika tallennetaan sessiotietoihin ja JavaScriptin funktio ilmoittaa ilmoituksena, kun aikaa on enää 10 minuuttia jäljellä ennen tokenin vanhenemista. Ilmoituksessa kerrotaan, että käyttäjä kirjataan ulos automaattisesti, jos ilmoituksen Jatka-painiketta ei paineta. Jatka-painike päivittää tokenin. Pois kirjauduttaessa tokenista tehdään kelvoton.

Noutoautomaatin muokkaussivu on ainoastaan ylläpitäjien käytössä. Sivulla näkee noutoautomaatin kaappinäkymän ja voi muokata automaatin nimeä sekä muita tietoja. Aikaisemmat toiminnot eli lokeron koon muokkaus, käyttöönotto ja poisto sisällytettiin sivulle aukeavaan muokkausikkunaan. Muokkausikkunassa lokeron numeroa voi vaihtaa ja lokerosta voi tehdä näyttölokeron. Ikkunassa voi myös lisätä ja poistaa lokeroita haluamaltaan moduulilta. On siis mahdollista muokata noutoautomaatin kaappinäkymää tarpeidensa mukaan. Muokkaaminen, esimerkiksi lokeron lisääminen, muodostaa JavaScriptissä (kuva 10) JSON-dataa, joka käsitellään rajapinnassa lisäten ja muokaten sen perusteella tietokannan arvoja. Poistettaessa lokeroa tarkistetaan JSON-datasta, onko kyseinen lokero lisätty add-riville eli ei vielä tallennettu tietokantaan. Jos näin on, poistetaan tämän lokeron add-rivi eikä lisätä delete-riville mitään.

```
function addLockerFunction(moduleNum)
{
  highestId++;

  //Add new locker button
  html = '<button id="lockerButtonEdit'+highestId+'" data-module="'+moduleNum+'" data-lockerid="'+highestId+'"
         data-lockernumber="'+-1+'" data-lockerheight="'+80+'" data-lockerwidth="'+500+'" data-lockertype="'+1+'"
         name="lockerButtonEdit" class="lockerBox col-12 cabinetButtonFree" style="height:'+80+'px;">
         <div id="lockerNumberEdit'+highestId+'" class="lockerNumber"></div>
         </button>';

  $('#moduleRowEdit'+moduleNum).append(html);

  //Add locker to the JSON
  editJSON['add'].push(JSON.parse('{"lockerId":'+highestId+', "height":80, "width":500, "type":1, "moduleNum":'+moduleNum+'}'));

  //Activate new locker button click event
  newLockerButtonEditFunction(highestId);
}
```

KUVA 10. Uuden lokeron tekevä funktio

Uuden käyttäjän rekisteröiminen tapahtuu ylläpitäjien uuden käyttäjän rekisteröimissivulla. Rekisteröitäessä tarvitaan uuden käyttäjän nimi, sukunimi ja sähköpostiosoite, jota käytetään kirjautumiseen ja jonne salasana lähetetään. Käyttäjälle tulee valita myös käyttäjätaso ja yritys. Käyttäjälle voi valita yrityksen, joka on jo rekisteröity, valitsemalla sen pudotusvalikosta. Valikkoon on listattu jokainen sivustolle rekisteröity yritys. Jos kuitenkin haluaa tehdä uuden yrityksen, kirjataan uuden yrityksen rekisteröimiskenttään yrityksen nimi, y-tunnus, osoite, postinumero ja postitoimipaikka. Yritykselle valitaan myös vähintään yksi noutoautomaatti käyttöön pudotusvalikosta, johon on listattu kaikki sivustolle tehdyt automaatit. Jos jokainen kohta on täytetty, lähetetään kirjatut tiedot uudesta käyttäjästä rajapintaan, missä tiedot tallennetaan tietokantaan. Sivuston Model-luokan funktio lähettää myös rekisteröitävän käyttäjän sähköpostiin tiedon rekisteröitymisestä, joka sisältää satunnaisesti generoidun salasanan.

Salasanan voi vaihtaa Asetukset-sivulla. Vaatimuksena salasanalle on vähintään viiden merkin pituus. Myös kirjautumissivulle lisättiin linkki Unohditko salasanasi? -sivulle, josta voi lähettää uuden satunnaisesti generoidun salasanan sähköpostiinsa, mikäli on unohtanut oman salasanansa. Rajapintakysely tarkistaa, onko sähköpostiosoitteelle olemassa tiliä. Mikäli tili löytyy, lähetetään uusi salasana.

3.3 Viimeistely

Viimeistelyvaiheeseen jäi ulkoasun korjailua, responsiivisuuden viimeistelyä ja palautesivun, materiaalisivun sekä sivuston alareunassa näkyvän alatunnisteen (footer) toteutus.

Palautesivulla voi lähettää sivustosta palautetta, joka tulee valittuihin sähköpostiosoitteisiin. Näitä sähköpostiosoitteita voi lisätä niin monta kuin haluaa, mutta opinnäytetyössä valmiiksi saadussa versiossa lisääminen onnistuu vain koodin puolella.

Materiaalisivulle lisäillään haluttuja pdf-tiedostoja, jotka voi ladata. Esimerkiksi sivuston käyttöopas löytyy materiaaleista. Näitä materiaaleja voi lisätä tällä hetkellä ainoastaan koodin puolelta.

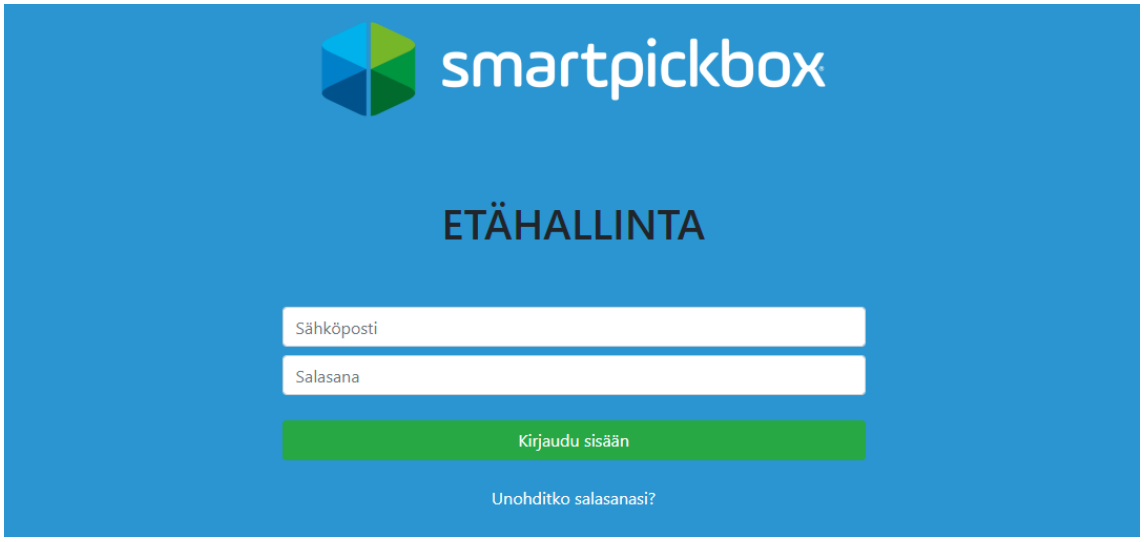
4 SMARTPICKBOX EXTRANET

Smartpickbox Extranet -verkkosivustolla käyttäjä voi nähdä noutoautomaatin lokeroiden varaustilanteen ja varata omaan käyttöönsä vapaana olevan lokeron. Omia varauksia pääsee muokkaamaan ja poistamaan. Verkkosivuilla voi myös vaihtaa asetuksista omaa salasanaansa ja lähettää palautetta sivustosta.

Ylläpitäjä pääsee rekisteröimään uusia käyttäjätilejä sekä muokkaamaan ja poistamaan niitä. Ylläpitäjä voi myös luoda, muokata ja poistaa noutoautomaatteja.

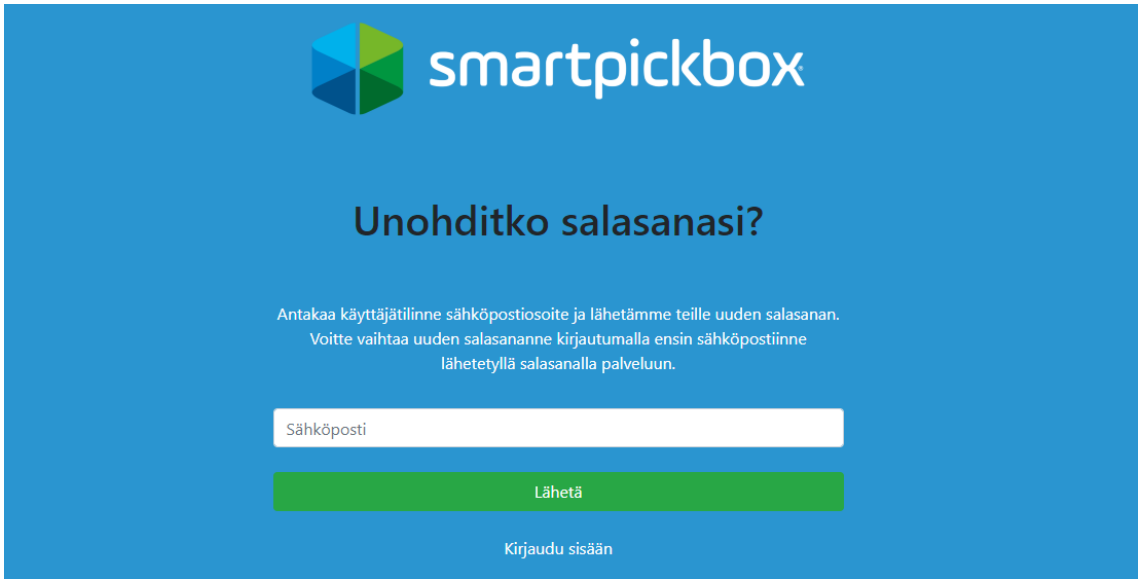
4.1 Kirjautuminen

Smartpickbox Extranet -sivusto vaatii sisäänkirjautumisen. Kirjautumatta pääsee vain kirjautumissivulle ja salasananpalautussivulle. Sisäänkirjautuminen tapahtuu syöttämällä rekisteröidyn käyttäjän sähköpostiosoitteen ja salasanan (kuva 11). Uuden käyttäjän pääsee tekemään vain ylläpitäjä. Halukas sivuston käyttäjä voi esimerkiksi ilmoittaa kiinnostuksensa palvelua kohtaan, jonka jälkeen ylläpitäjä luo hänelle käyttäjätilin. Käyttäjä saa ylläpitäjän rekisteröimään sähköpostiosoitteeseen satunnaisesti generoidun salasanan, jolla pääsee kirjautumaan sivustolle. Ensimmäisen kirjautumisen yhteydessä pyydetään käyttäjää vaihtamaan salasanansa omakseen. Salasanan tulee olla vähintään 5 merkkiä pitkä.



KUVA 11. Kirjautumissivu

Mikäli käyttäjä unohtaa oman salasanansa, pääsee kirjautumissivun kautta salasananpalautussivulle (kuva 12). Rajapintakysely tarkistaa tietokannasta, onko syötetylle sähköpostiosoitteelle luotu käyttäjää. Uusi satunnaisesti generoitu salasana lähetetään haluttuun sähköpostiosoitteeseen, johon tili on luotu.



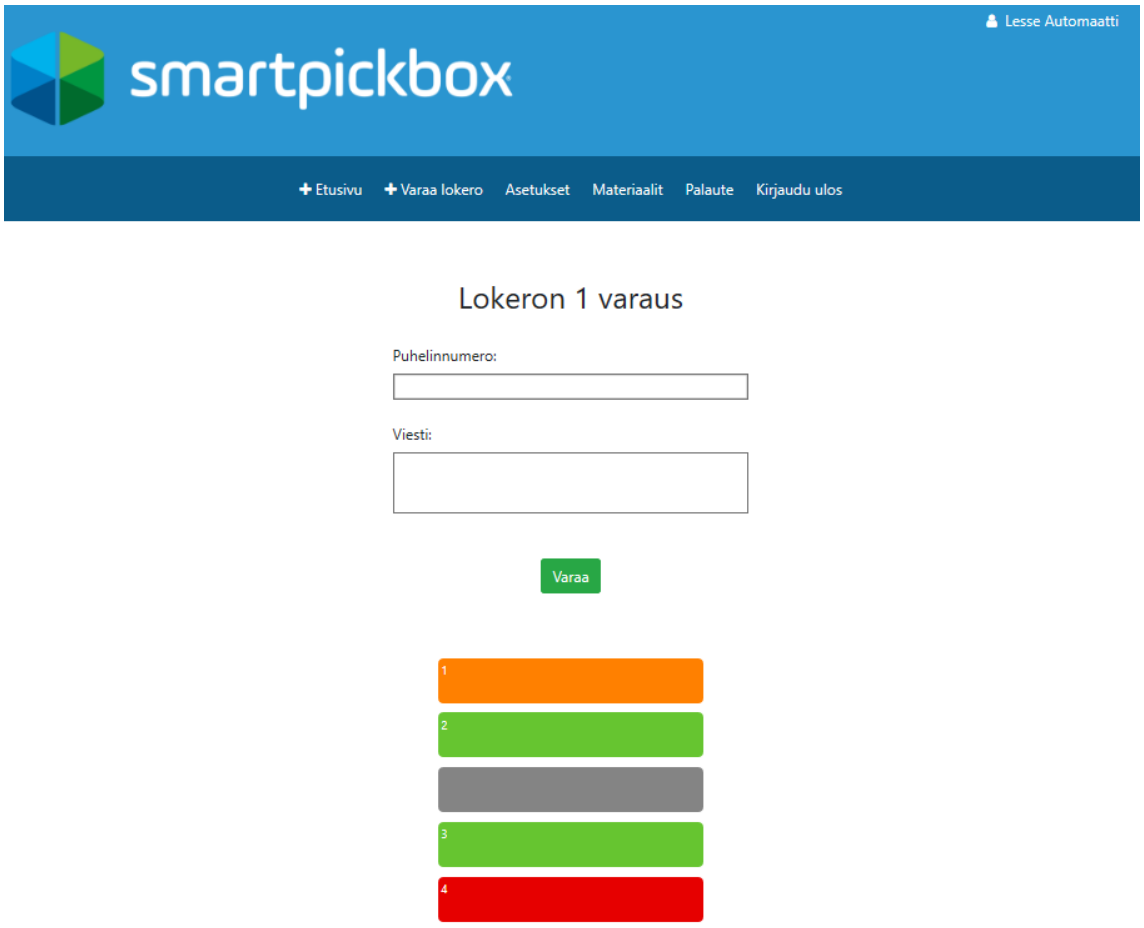
KUVA 12. Salasananpalautussivu

4.2 Lokeron varaus

Vapaana olevan noutoautomaatin lokeron voi varata omaan käyttöön. Vapaana olevat lokerot ovat vihreitä, punaiset ovat omia varattuja lokeroita ja harmaat ovat toisten varaamia, käytöstä poistettuja tai näyttölokeroita, joita ei pysty klikkaamaan.

Varaaminen tapahtuu valitsemalla halutun vapaan lokeron ja kirjoittamalla oman asiakkaansa puhelinnumeron ja noutoviestin (kuva 13). Noutoviesti lähetetään kirjattuun numeroon silloin, kun tavara on viety noutoautomaatin varattuun lokeroon.

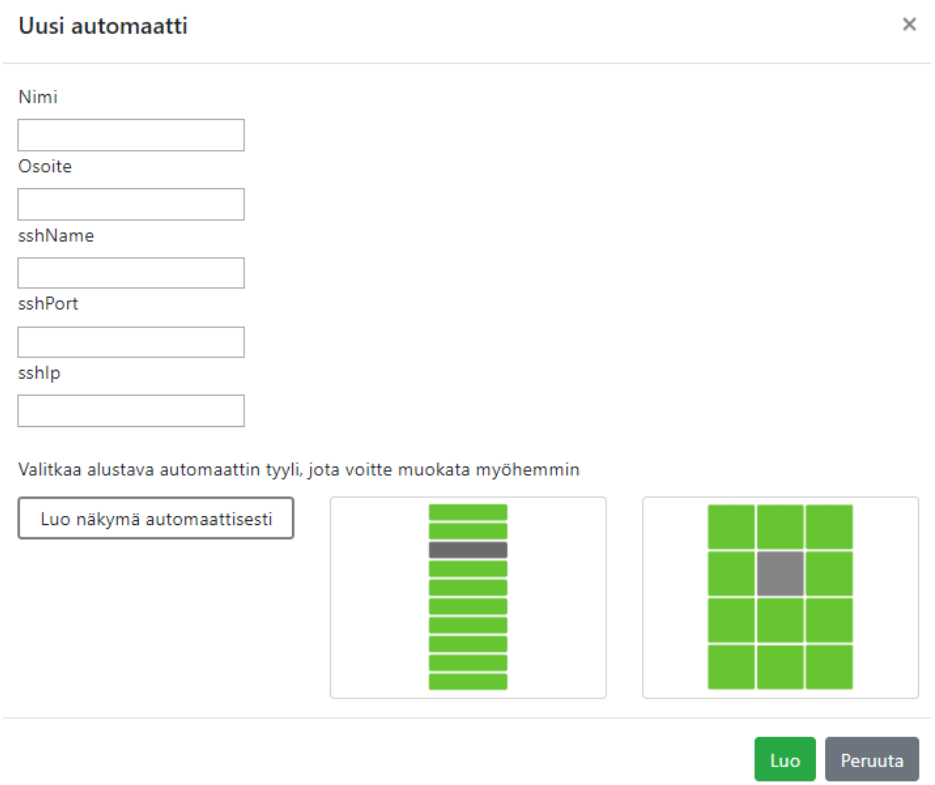
Klikkaamalla omaa varattua lokeroa voi muokata puhelinnumeroa ja tekstiviestiä tai poistaa varauksen.



KUVA 13. Noutoautomaatin vapaana olevan lokeron varaaminen

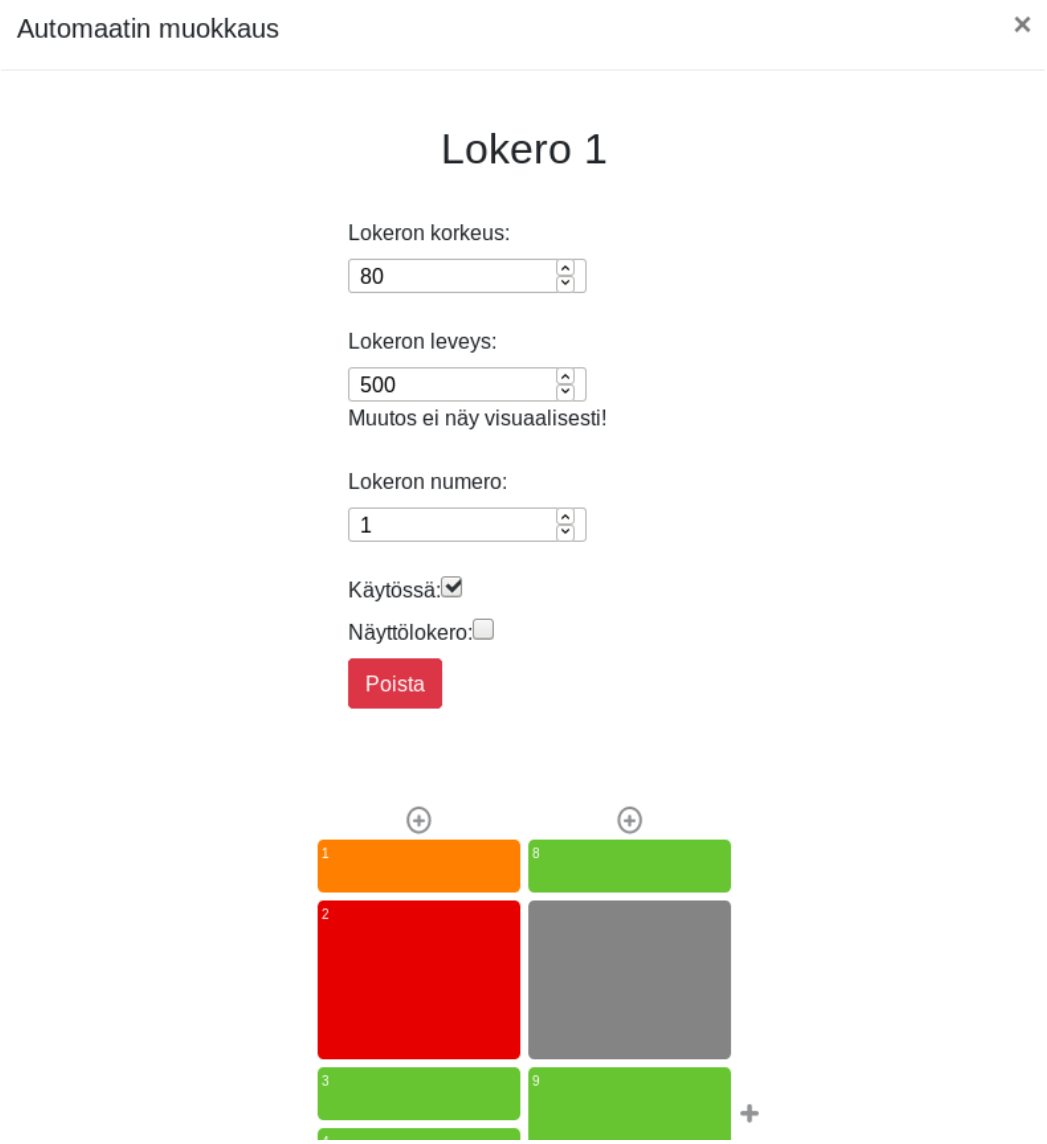
4.3 Ylläpito

Sivuston ylläpitäjä voi luoda noutoautomaattinäkymiä syöttämällä tarvitut tiedot ja valitsemalla kolmesta vaihtoehdosta halutun aloituskokoonpanon lokeroille (kuva 14). Ensimmäinen vaihtoehto muodostaa yhteyden noutoautomaattiin annetuilla tiedoilla ja saa tätä kautta muodostettua oikean määrän lokeroita. Tällöin mahdolliset lokeroiden senhetkiset varaukset tallentuvat sivuston tietokantaan ja ovat heti näkyvissä noutoautomaattinäkymässä. Muut vaihtoehdot luovat esimerkkikuvassa näkyvän kokoonpanon.



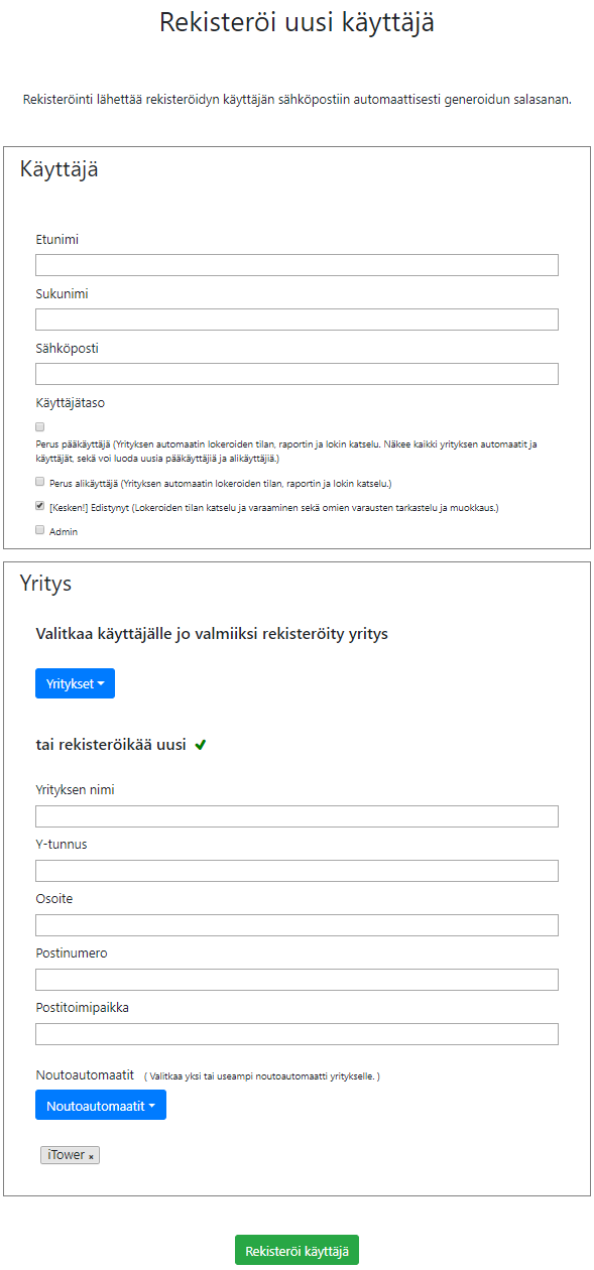
KUVA 14. Uuden noutoautomaatin luominen

Noutoautomaatteja voi muokata (kuva 15). Lokeroiden korkeutta ja leveyttä voi vaihtaa sekä lokeron numeron voi itse määrittää. Lokeron voi poistaa käytöstä ja lokerosta voi tehdä näyttölokeron. Lokeroita voi poistaa ja lisätä uudelle tai vanhalle riville eli moduulille.



KUVA 15. Noutoautomaatin muokkaaminen

Ylläpitäjä voi luoda uusia ylläpitäjiä tai muita käyttäjiä sivustolle (kuva 16). Käyttäjälle voi valita jo rekisteröidyn yrityksen tai tehdä uuden yrityksen syöttämällä tarvitut tiedot yrityksestä. Uudelle yritykselle myös valitaan sen käyttöön tulevat noutoautomaatit listasta. Käyttäjän rekisteröinti lähettää rekisteröidyn käyttäjän sähköpostiin satunnaisesti generoidun salasanan, jolla pääsee kirjautumaan sivustolle. Ylläpitäjä näkee kaikki sivustolle rekisteröidyt käyttäjät sekä yritykset ja voi muokata ja poistaa näitä käyttäjähallinnan kautta (kuva 17).



KUVA 16. Ylläpitäjän uuden käyttäjän rekisteröiminen

KUVA 17. Käyttäjähallinta

5 JATKOKEHITYS

Jotta sivustoa pystytään käyttämään siihen tarkoitetulla tavalla, tarvitaan yhteys noutoautomaattiin. Noutoautomaatin lokeron varaaminen ja varauksen muokkaamisen tiedot tulee päivittää noutoautomaattiin. Myös varatun lokeron täyttämisestä lähtisi tieto rajapintaan ja lähetettäisiin automaatin kautta tekstiviesti sivustolla kirjattuun numeroon. Kun tavara käydään noutamassa, varaus lähtisi pois, jolloin lokeron voisi varata uudelleen. Nämä vaatisivat muutoksia automaatin ohjelmistoon sekä automaatin oman tietokannan päivittämiseen soveltuvan ohjelmiston.

Opinnäytetyössä valmiiksi saadussa versiossa noutoautomaatin muokkaus sallii vain yhdellä moduulilla olevat lokerot, mutta mahdollisuus monen moduulin lokeroon pitää lisätä.

Noutoautomaattiin voi kirjautua tagin avulla, mikäli automaatissa on taginlukija. Sivustolle pitää lisätä sivu, jota kautta käyttäjät voisivat lisätä tageja itselleen. Tällöin käyttäjä, joka on varannut lokeron ja on tullut täyttämään tai poistamaan tuotettaan lokerosta, pääsisi kirjautumaan automaatille tagin avulla ilman salasanan syöttämistä.

Ylläpitäjille tarvitsee lisätä mahdollisuus lisätä ja poistaa palautesivulta lähetettyjen palautteiden vastaanottavia sähköpostiosoitteita. Myös materiaalisivun materiaalien lisäys- ja poistomahdollisuus olisi hyödyllinen, jotta sitäkään ei ohjelmoijan tarvitsisi lisätä suoraan koodiin.

6 YHTEENVETO

Työn tarkoituksena oli luoda verkkosivusto, jota kautta asiakas voi varata itselleen noutoautomaatista omaan käyttöönsä lokeron ja tarvittaessa muokata ja poistaa tämän varauksen. Lisäksi uusien automaattinäkymien luominen ylläpidon puolelle kuului työhön. Nämä saatiin tehdyksi sisällyttäen tähän toteutukseen tarvittavan käyttäjän rekisteröimisen ja kirjautumisen.

Viimeistelyosuus venyi hieman alkusuunnittelua pidemmäksi, sillä jatkoin yrityksessä työskentelyä ja sinä aikana parantelin ja lisäilin ominaisuuksia sivustolle. Sisällytin niistä osan myös tähän työhön kuuluviksi, kuten parannuksia käyttäjähallintaan sekä materiaalisivun ja palautesivun. Myös automaattinäkymän luominen automaattisesti ottaen yhteys noutoautomaattiin lisättiin.

Jotta sivusto toimisi sille tarkoitetulla tavalla, tarvittaisiin yhteys noutoautomaattiin. Noutoautomaatin tietoja tarvitsisi muokata sivustolta päin sekä päinvastoin. Tätä toimintoa ei sisällytetty työn tavoitteeseen, jotta projekti ei paisuisi liikaa.

LÄHTEET

1. Codeigniter at a Glance. 2019. British Columbia Institute of Technology. Saatavissa: https://codeigniter.com/user_guide/overview/at_a_glance.html. Hakupäivä 29.2.2020.

2. Ezell, Lonnie 2016. Practical Codeigniter 3. Lean Publishing. Saatavissa: https://api.grave-design.com/practicalcodeigniter3.pdf. Hakupäivä 29.2.2020.

3. What is an API? (Application Programming Interface). MuleSoft LLC. Saatavissa: https://www.mulesoft.com/resources/api/what-is-an-api. Hakupäivä 2.3.2020.

4. Stauffer, Matt 2015. Introducing Lumen from Laravel. Saatavissa: https://mattstauffer.com/blog/introducing-lumen-from-laravel/. Hakupäivä 2.3.2020.

5. Otemuyiwa, Prosper 2019. Developing RESTful APIs with Lumen (A PHP Micro-framework). Auth0 Inc. Saatavissa: https://auth0.com/blog/developing-restful-apis-with-lumen/. Hakupäivä 2.3.2020.

6. Introduction. Composer. Saatavissa: https://getcomposer.org/doc/00-intro.md. Hakupäivä 6.3.2020.

7. About MariaDB Server. MariaDB Foundation. Saatavissa: https://mariadb.org/about/. Hakupäivä 6.3.2020.

8. Vihavainen, Arto – Luukkainen, Matti 2017. Tietokantojen perusteet. University of Helsinki, Department of computer science. Saatavissa: https://tietokantojen-perusteet.github.io/. Hakupäivä 24.2.2020.

9. Lahtonen, Tommi 2016. Evästeet ja sessiot. Jyväskylän yliopiston informaatioteknologian tiedekunta. Saatavissa: http://appro.mit.jyu.fi/tiea2080/luennot/cookies/#TOC3. Hakupäivä 1.3.2020.