

TTMS0700 - Web-projekti 2

Dokumentaatio

4/2019

Tekijät:

Johanna Kaasalainen

Noora Ojanen

Laura Ala-Korte

Maria Salonen

Johdanto

Tässä dokumentissa kuvataan Web-projekti 2 –opintojakson harjoitustyötä ja arvioidaan työn onnistumista. Opintojakson tarkoituksena oli toteuttaa laaja web-sovellus, joka käyttää sekä asiakas- että palvelinpuolen tekniikoita. Työ toteutettiin neljän hengen ryhmässä.


Projektin aloitus

Projektityön toteutus aloitettiin jo aiemmalla opintojaksolla TTMS0600 Web-palvelun määrittely ja suunnittelu, jolloin olimme suunnitelleet verkkokaupan teoriassa. Web projektia varten karsimme laajasta suunnitelmasta palasia, joiden avulla lähdimme toteuttamaan verkkokauppaa käytännössä. Päädyimme toteutukseen, jossa aluksi pyrimme Laravellin avulla simppelein vaatekaupan. Valitsimme toiminnoiksi seuraavat: haku sekä suodatus, vaatteiden lajittelu (miehet ja naiset), ostoskori sekä tilaus.

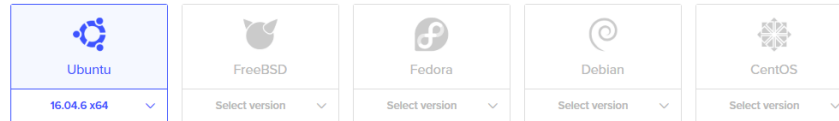
Pohja ja ympäristö

Työ aloitettiin päättämällä toteutusalue, jossa voisimme hyödyntää helposti tietokantoja sekä pääsisimme helposti käsiksi sivuihin. Tämän vuoksi paikallisesti virtuaalikoneelle rakennettavat nettisivut eivät käyneet. Valitsimme suosittelun ja kevyen tutustumisen jälkeen toteutuspaikaksi Digitaloceanin dropletin, joka maksaa 5€/kk. Droplettiin saamme tietokannasta lähtien kaikki sekä helposti yhteyden niin kotoa kuin koulussa. Tai niin luulimme, sillä valitettavasti koulun labranetin palomuuuri ei sallinut samban hyödyntämistä.

Create Droplets

Choose an image 

Distributions Container distributions Marketplace Custom images



Loimme dropletin Ubuntu 16.04.6 pohjalla, jonka päälle aloimme rakentamaan projektia. Droplet on netissä toimiva virtuaalikone.

Päätimme toteuttaa työn Laravellin avulla. Laravel on avoimen lähdekoodin PHP framework, jolla voidaan toteuttaa erilaisia nettisivustoja helposti. Tämän hetkinen stabiili versio Laravellista on 5.8. Meidän työssämme käytetään versiota 5.8.4.

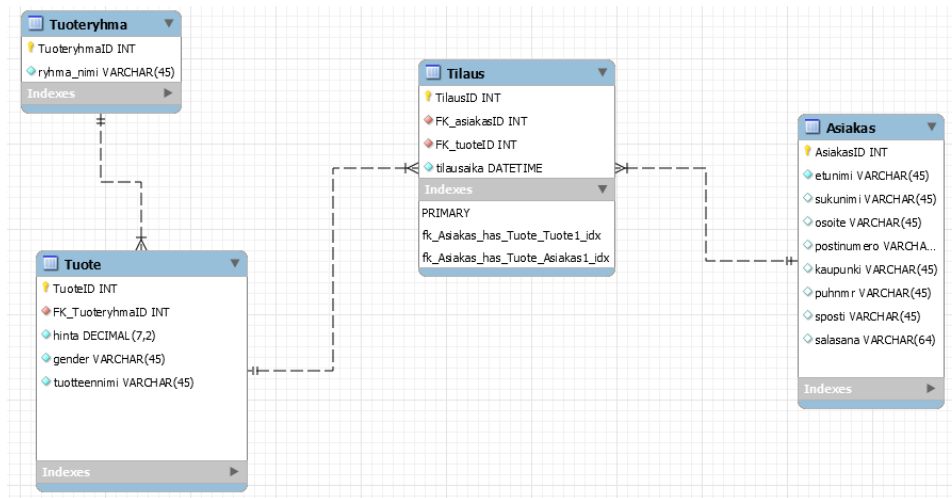
Asensimme Laravellin oppimateriaaleista löytyvän ohjeen mukaisesti. Ensimmäisenä ajettiin päivitykset juuri asennettuun droplettiin, minkä jälkeen asennettiin PHP. Lisäksi varmistimme, että apache2 ymmärtää PHP:ta erilaisten pakettien avulla. Loimme apacheen myös tavallisen käyttäjän public_html kansion, johon luomme itse Laravellin lopulta.

```
1 sudo apt update
2 sudo add-app-repository ppa:ondrej/php
3 sudo add-apt-repository ppa:ondrej/php
4 sudo apt-get update
5 sudo apt-get install zip apache2 libapache2-mod-php7.2 php7.2 php7.2-xml
php7.2-gd php7.2-openssl php7.2-mbstring
6 cd /tmp
7 curl -sS https://getcomposer.org/installer | php
8 sudo mv composer.phar /usr/local/bin/composer
9 sudo a2enmod userdir
10 sudo nano /etc/apache2/mods-enabled/php7.2.conf
11 sudo systemctl restart apache2.service
12 cd
13 mkdir public_html
14 cd public_html/
15 composer create-project laravel/laravel l5todo --prefer-dist
16 sudo nano /etc/apache2/sites-available/laravel.conf
17 sudo nano /etc/apache2/envvars
18 sudo a2dissite 000-default.conf
19 sudo a2ensite laravel.conf
20 sudo a2enmod rewrite
21 sudo systemctl restart apache2.service
```

Todettuamme Laravellin toimivaksi siirryimme asentamaan Samba-palvelinta, jonka avulla voisimme helposti päästä koneen sisältöön käsiksi ja sivuja olisi helpompi rakentaa. Samban asennuksessa ilmeni ongelmia .conf tiedostojen kanssa, jotka ratkaistiin opettajan avulla lopulta. Lisäksi ongelmakohtaksi nousi labranetin palomuuuri, joka ei sallinut Samban käyttämistä windowsin tiedostonhallinnassa. Kotona sivuja kuitenkin on pystynyt rakentamaan paikallisesti. Koululla käytimme WinSCP ohjelmaa.

Tietokanta

Verkkokaupan rakentaminen aloitettiin luomalla tietokanta. Aloitimme suunnittelun aluksi perinteisin menetelmin paperia ja kynää hyödyntäen. Pohdimme mitä ominaisuuksia tarvitsemme nimenomaan tähän työhön, jotta projekti ei laajenisi liian suureksi, mutta toimisi kurssivaatimusten puitteissa. Päätimme luoda neljä erilaista taulua: Asiakas, Tilaus, Tuote, Tuoteryhma.



Käytimme MySQL Workbenchii, jossa loimme aluksi labranetin palvelimelle demotietokannan. Testasimme tietokannan toimivuutta lisäämällä, muokkaamalla sekä yhdistelemällä dataa. Lisäksi konsultoimme opettajaa. Tietokantapohjan ollessa valmis, lähdimme siirtämään koko tietokantaa Digitaloceaniin.

Saimme koko MySQL koodin irti Workbenchistä ja sen avulla loimme tietokannan projektiimme. Jotta tietokanta saatiin droplettiin, meidän tuli asentaa MySQL. Asennuksen jälkeen loimme tietokannan käyttämällä Workbenchistä valmiiksi saamaamme MySQL komento kokonaisuutta.

```
182 sudo apt update
183 sudo apt install mysql-server
184 sudo mysql -u root -p
185 sudo systemctl restart mysql.service
186 mysql -u root -p
```

```
mysql> show tables
-> ;
+-----+
| Tables_in_ttms0700 |
+-----+
| Asiakas             |
| Tilaus              |
| Tuote               |
| Tuoteryhma          |
+-----+
4 rows in set (0.00 sec)
```

Tietokannan luonnin jälkeen pääsimme tekemään varsinaista projektia. Projektin edetessä törmäsimme tilanteisiin, joissa tietokannan rakennetta oli päivitettävä. Niistä kuitenkin kerromme tarkemmin myöhemmissä osioissa.

Hakupalkki

Sivuston oikeasta ylälaidasta löytyy vapaa hakukenttä, jolla voi hakea tuotteen nimellä haluamaansa tuotetta, esim. Farkut. Hakupalkin tekemiseen tarvitsi neljä eri paikkaa: routesin web.php, controllersiin SearchController.php sekä views kansioon search.blade.php. Formi löytyy views/layouts app.blade.php:stä, jotta se näkyy jokaisella sivulla.

A screenshot of a search bar. It consists of a white rectangular input field with a thin grey border, followed by a grey button with the word "Etsi" in white text. The entire search bar is set against a dark background.

Aluksi loin formin, jonka avulla käyttäjä voisi hakea tuloksia. Formin toimintaa varten tehtiin controller. Controller nappaa käyttäjän inputin \$hakueto nimiseen muuttujaan. Jotta tieto saataisiin ulos tietokannasta, tehtiin seuraavanlainen SQL-lauseke:

```
$sql = <<<SQLEND
SELECT tuotteennimi, hinta, gender, TuoteID, tuotteenkuva
FROM Tuote WHERE tuotteennimi
LIKE :hakueto
```

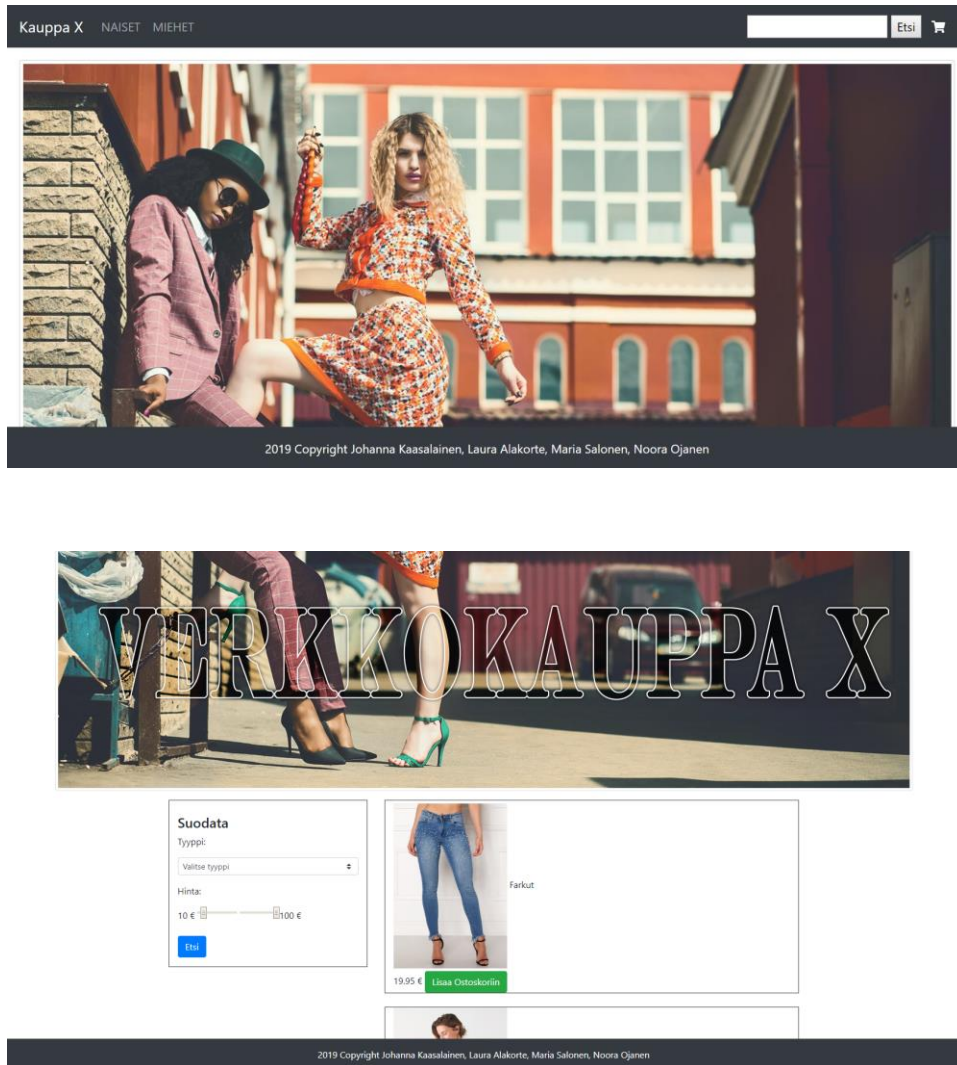
SQL-lausekkeen :hakueto kohta viittaa käyttäjän antamaan inputtiin, jolla käyttäjän tulos lopulta haetaan tietokannasta. :hakueto kohtaa täytyi hieman soveltaa, sillä pelkkä muuttujan käyttäminen ei tuonut toivottua tulosta. Pelkän muuttujan avulla koko hakusana olisi tullut löytyä sellaisenaan tietokannasta.

```
$stmt = $PDO->prepare($sql);
$h = "%$hakueto%";
$stmt->bindParam(':hakueto', $h, \PDO::PARAM_STR);
$stmt->execute();
$tulokset = $stmt->fetchAll(\PDO::FETCH_OBJ);
//return $tulokset;
return view('search')->with('tulokset', $tulokset);
```

Yllä näkyy soveltamistapa, jolla hakueto saatiin toimimaan oikein. \$hakueto muuttuja tuli asettaa uuteen muuttujaan, jossa sille annetaan %-merkkien avulla ilmaus "näytä kaikki jotka sisältävät seuraavan kirjainyhdistelmän". Muuttujaa kutsutaan tulokset-muuttujaan ja lopuksi returnataan tulokset. Search.blade.php:ssa sitten kutsutaan tulokset näkymään.

Etusivu -näkymä

Etusivulle listataan kaikkien kategorioiden kaikki tuotteet ja niitä voi suodattaa suodatuspalkin avulla. Etusivu hakee näkymän app.blade.php:sta ja muut tarvittavat elementit etusivu.blade.php:sta. Reitit löytyvät web.php:sta ja controller on nimeltään EtusivuController.php, jonka avulla haetaan tietoa tietokannasta.



EtusivuController.php:ssa tulostetaan sql lausekkeella etusivulle kaikki tuotteet tietokannasta.

```

11 class EtusivuController extends Controller
12 {
13
14     public function list_all()
15     {
16         $PDO = DB::connection('mysql')->getPdo();
17
18         $sql = <<< SQLEND
19         SELECT *
20         FROM Tuote
21     SQLEND;
22
23         $allsql = $PDO->prepare($sql);
24
25         $allsql->execute();
26
27         // Muista TÄMÄ FETCH_OBJ
28         $etusivu = $allsql->fetchAll(\PDO::FETCH_OBJ);
29         //return $etusivu;
30
31         return view('etusivu')->with('etusivu', $etusivu);
32     }
33 }

```

Routesiin tehtiin kaksi reittiä, joista pääsee verkkokaupan etusivulle. Nämä kaksi vaihtoehtoa ovat:

1. <http://159.65.232.228/etusivu>
2. <http://159.65.232.228/>

Koodi:

```

14 Route::get('/', 'EtusivuController@list_all',
15 function () {
16     return view('etusivu');
17 });
18 Route::get('etusivu', 'EtusivuController@list_all');

```

Etusivu.blade.phplla tulostetaan etusivun näkymä.

Tuotteita listataan muuttujalla \$etus:

```

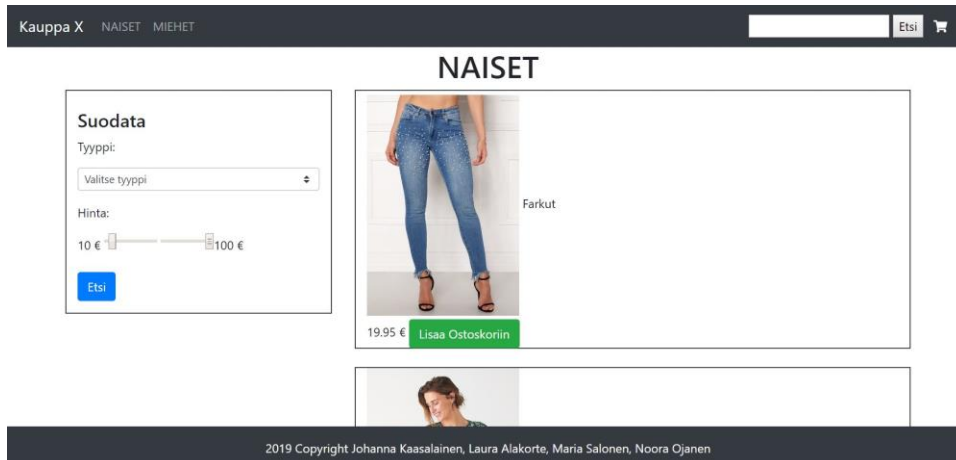
62 @foreach ($etusivu as $etus)
63
64     <div class="col-sm border border-dark">
65          {{ $etus->tuotteennimi }} <br>{{ $etus->hinta }} € <a href="#"
66     </div> <br>
67
68 @endforeach

```

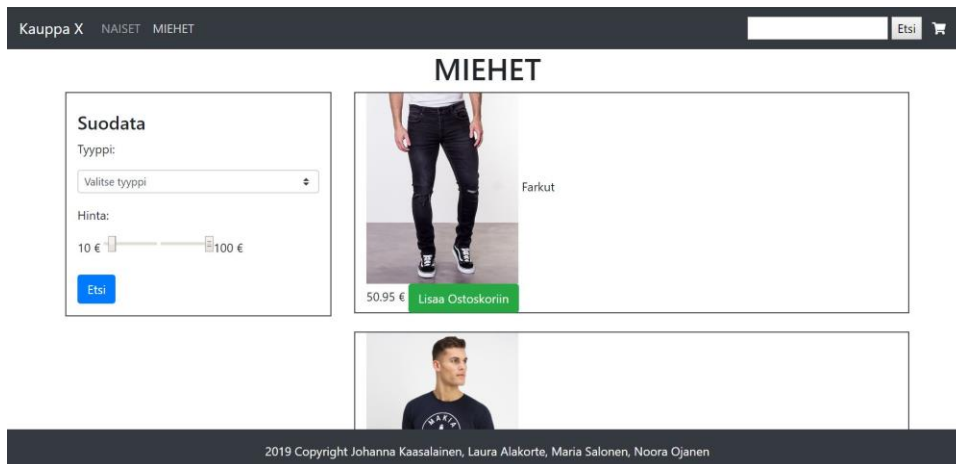
Naiset ja Miehet -näkymät

Naisilla ja miehillä on omat näkymät, joista käyttäjä voi helposti löytää itselle sopivia vaatteita. Naisten tiedostoja ovat naiset.blade.php ja NaisetController.php. Miehillä miehet.blade.php ja MiehetController.php.

Naiset -näkö:



Miehet -näkö:



naiset.blade.php

```

80     @foreach ($naiset as $nainen)
81
82         <div class="col-sm border border-dark">
83              {{ $nainen->tuotteennimi }} <br>{{ $nainen->hinta }} € <a href="#"
84         </div> <br>
85
86     @endforeach

```

miehet.blade.php

```

64     @foreach ($miehet as $mies)
65
66         <div class="col-sm border border-dark">
67              {{ $mies->tuotteennimi }} <br>{{ $mies->hinta }} € <a href="#"
68         </div> <br>
69
70     @endforeach

```

Reitit naisille ja miehille:

```

29     Route::get('naiset', 'NaisetController@list_all');
30     Route::get('miehet', 'MiehetController@list_all');

```

Suodatus –toiminto

Käyttäjä voi halutessaan suodattaa tuotteita suodatuspalkin avulla. Vaihtoehtoina ovat tyyppi ja hinta. Teimme erilliset suodattukset naisille naiset-sivulle, miehille miehet-sivulle ja etusivulle, jossa listataan kaikki tuotteet.

Tiedostot:

- web.php
- filter.blade.php
- filternaiset.blade.php
- filtermiehet.blade.php
- FilterController.php
- FilterNaisetController.php
- FilterMiehetController.php

Suodatus -näkyvä:

Suodata

Tyyppi:

Valitse tyyppi

Hinta:

10 €
100 €

Etsi

Suodata

Tyyppi:

Housut

Valitse tyyppi

Paita

Housut

Hinta:

10 €
100 €

Etsi

Formiin laitetaan action `"/filter"`, jolloin Etsi-painiketta painettaessa sivu siirtyy erilliselle suodatettujen sivulle. Oninput toiminto tarvitaan, että hintarullaa liikuttaessa sen vieressä näkyy valitun hinnan lukema.

```
34 <form action="/filter" method="get" oninput="z1.value=minhinta.value; z2.value=maxhinta.value">
```

On tärkeää nimetä kaikki kodat oikein. z1 etsii minhinta arvon ja z2 etsii maxhinta arvon. Minimihinnaksi on asetettu 0 € ja maksimiksi 100 €.

```

<label id='hinta'>
  <output id='z1' name="z1" for="minhinta">10</output> €
  <input class="text-center" style="width: 70px;" type="range" name="hintamin" id="minhinta" value="10" min="0" max="100">
  <input style="width: 70px;" type="range" name="hintamax" id="maxhinta" value="100" min="1" max="100"><output id='z2' name="z2"
</label>

```

Reitit suodatukselle:

```

43 Route::get('filter', array(
44     'as' => 'filter',
45     'uses' => 'FilterController@index'
46 ));
47
48 Route::get('filternaiset', array(
49     'as' => 'filternaiset',
50     'uses' => 'FilterNaisetController@index'
51 ));
52
53
54 Route::get('filtermiehet', array(
55     'as' => 'filtermiehet',
56     'uses' => 'FilterMiehetController@index'
57 ));

```

Sql-lauseke FilterController.php, jolla suodatetaan kaikkia tuotteita.

```

$sql = <<<SQLEND
SELECT tuotteennimi, hinta, gender, TuoteID, tuotteenkuva
FROM Tuote WHERE FK_TuoteryhmaID = :kategoria
AND hinta > :hintamin
AND hinta < :hintamax

SQLEND;

$stmt = $PDO->prepare($sql);
/*$h = "%$hakuehto%";*/
$stmt->bindParam(':kategoria', $kategoria, \PDO::PARAM_STR);
$stmt->bindParam(':hintamin', $hintamin, \PDO::PARAM_STR);
$stmt->bindParam(':hintamax', $hintamax, \PDO::PARAM_STR);
$stmt->execute();
$filter = $stmt->fetchAll((\PDO::FETCH_OBJ));
//return $filter;
return view('filter')->with('filter', $filter);

```

FilterNaisetController.php poikkeaa edellisestä siten, että nyt suodatetaan vain naisten vaatteita.

```
$sql = <<<SQLEND
SELECT tuotteennimi, hinta, gender, TuoteID, tuotteenkuva
FROM Tuote WHERE FK_TuoteryhmaID = :kategoria
AND gender = 'naiset'
AND hinta > :hintamin
AND hinta < :hintamax
```

FilterMiehetController.php:

```
$sql = <<<SQLEND
SELECT tuotteennimi, hinta, gender, TuoteID, tuotteenkuva
FROM Tuote WHERE FK_TuoteryhmaID = :kategoria
AND gender = 'miehet'
AND hinta > :hintamin
AND hinta < :hintamax
```

Ostoskori

Ensin piti miettiä kaikki perustoiminnot, mitä ostoskori yleensä tarvitsee:

1. Ostoskoriin täytyy pystyä lisäämään tuotteita.
2. Ostoskorissa pitää pystyä selaamaan ostoskorissa olevia tuotteita
3. Ostoskorissa olevien tuotteiden lukumäärää pitää pystyä muokkaamaan
4. Ostoskorista pitää pystyä poistamaan tuotteita
5. Yhteishinta ostoksille täytyy laskea

Ostoskorin pääsääntöiset toiminnot suoritetaan controllerissa AjaxController ja tulostakori.bladessa. Alhalla kuva ostoskorin ulkoasusta:

Ostoskori

	Farkut	19.95	1	poista korista
	Adidaksen verkkarit	19.95	1	poista korista
	T-paita	29.95	1	poista korista

Ostokset yhteensä: 69.85euroa

[Jatka tilauksen](#)

Tuotteiden lisäys ostoskoriin

Jos menee naiset.blade tai miehet.blade huomataan, että jokaisen tuotteen kohdalla on "lisää ostoskoriin" painike, kun sitä painetaan click funktio scriptissä aktivoituu (luotu jQueryn avulla). Painikkeissa on id:nä tuotteen tietokannassa oleva TuoteID ja click funktiossa haetaan tämä painikeen id ja ajaxin avulla lähetetään tämä tieto AjaxControllerille funktiolle tulosta.

```

12 ▾ public function tulosta(Request $request) {
13     $tuoteid = $request->id;
14     //echo $tuoteid;
15     session_start();
16     $_SESSION['kori'][]=$tuoteid;
17     $laskutoi = array_count_values($_SESSION['kori']);
18
19 ▾     if (in_array($tuoteid,$_SESSION['kori'])){
20 ▾         if($laskutoi[$tuoteid] > 5){
21
22
23             $nollaa = $tuoteid;
24 ▾             foreach (array_keys($_SESSION['kori'],$nollaa,true) as $key){
25                 unset($_SESSION['kori'][$key]);
26
27
28             }
29             $_SESSION['kori'] = array_values($_SESSION['kori']);
30             $laskuri = 0;
31 ▾             for($a=0; $a<$laskutoi[$tuoteid]; $a++){
32                 echo "lasketaan";
33 ▾                 if($a == 5){
34                     break;
35                 }
36                 $_SESSION['kori'][]=$tuoteid;
37             }
38
39         }
40         var_dump($_SESSION['kori']);
41         $laskutoi = array_count_values($_SESSION['kori']);
42         print_r($laskutoi);
43     }
44
45 }
46

```

Ensin laitetaan bladen puolelta saatu muuttuja toiseen muuttujaan ja laitetaan se \$_SESSION['kori'] taulukkoon. Samaa tuotetta pystyy tilaamaan korkeintaan viisi kappaletta kerralla, joten pitää tehdä tarkistuksia jos painiketta klikkaa enemmän, kuin viisi kertaa. Yläpuolella kuvassa rivi 17 lasketaan \$_SESSION['kori'] taulukon arvoja eli, kuinka monta kertaa sama arvo on olemassa taulukossa ja siirretään se muuttujaan, jotta tätä toimintoa on helpompi käsitellä.

(Ylhäällä kuvassa) Rivillä 20-39 suoritetaan toiminnot, jos tuotetta on korissa enemmän, kuin sallittu määrä tarkistamalla if-ehtolauseen avulla. Poistetaan foreach-loopin avulla \$_SESSION['kori']-taulukosta kaikki muuttujat, jotka ovat saman arvoisia kuin napista saatu tuoteid. Tämän jälkeen järjestellään taulukko uudelleen (alkiot laitetaan järjestykseen alkamaan nolasta). Seuraavaksi for-loopilla luodaan elementtejä niin kauan, kun niitä on luotu viisi kertaa.

Tein myös jQuery:n avulla animaation, jossa ilmoitetaan käyttäjälle, että tuote on lisätty ostoskoriin.

Tuotteiden tulostus ostoskori näkymään

Kun painetaan ostoskori kuvaketta oikeasta yläkulmasta saadaan näkymään tulostakori.blade.

AjaxControllerissa funktiossa naytakori() käsitellään dataa, riippuen siitä mitä halutaan sivustolla näkyvän.

Käsitellään ensin alapuolella olevasta kuvasta rivejä 49-86. Ihan ensimmäiseksi pitää if lauseen avulla tarkistaa onko \$_SESSION['kori'] olemassa ja jos on pitää tarkastaa onko se erisuuri kuin nolla, ettei anna erroria, jos esimerkiksi korista poistaa kaikki tuotteet.

Ensin luodaan taulukko tuotejson taulukko ja luodaan samanlainen muuttuja, joka laskee kuinka monta samaa TuoteID:tä \$_SESSION['kori']:ssa esiintyy. Tämän jälkeen tarvitaan foreach looppia antamaan jokainen arvo \$_SESSION['kori']:sta muuttujana osto.

Haemme myös tietokannassa olevat tiedot taulusta Tuote rivillä 56 Korissa.php:ltä. Käytetään jälleen foreach-looppia, mutta tälläkertaa tietokannasta haettuihin tietoihin, hakemaan jokainen elementti muuttujana item.

Tämän jälkeen katsotaan if-ehdolla onko item:in TuoteID yhtäsuuri, kuin muuttujan osto oleva luku ja luodaan muuttujia sen verta mitä tarvitsemme eli nimi, hinta, kuvatus(eli kuva) ja tunniste(eli tuoteid). Näihin muuttujiin laitetaan kyseisen itemin eli taulukon rivin arvoja (kuvassa rivit 60-63). Nämä muuttujat laitetaan sen jälkeen taulukkoon nimeltä uusitieto. Kun kaikki tarvittavat tiedot ovat uusitieto-taulukossa käytetään array_push:ia siirtämään ne tuotejson:in.

Rivillä 83 järjestellään taulukko, siten että taulukossa voi olla olemassa vain yksi saman arvoinen "tieto", muuten ostoskoriin tulostuisi sama tieto useampaan kertaan, jos samaa tuotetta \$_SESSION['kori']:ssa on enemmän kuin yksi. Palautetaan lopuksi tiedot tulostakori.blade:n return view:lla.

Jos \$_SESSION['kori'] on tyhjä tai ei ole olemassa on siihen tarvittavat muuttujat käsitelty alapuolen kuvassa rivillä 87-95.

```

47 ▾ public function naytakori() {
48     session_start();
49 ▾     if (isset($_SESSION['kori'])){
50
51 ▾         if (count($_SESSION['kori'])!=0){
52             $tuotejson=array();
53             $montako = 0;
54             $laske = array_count_values($_SESSION['kori']);
55 ▾             foreach ($_SESSION['kori'] as $osto) {
56                 $ostokset = Korissa::all();
57                 $json = $ostokset;
58 ▾                 foreach ($json as $item) {
59 ▾                     if ($item["TuoteID"] == $osto) {
60                         $nimi = $item["tuotteennimi"];
61                         $hinta = $item["hinta"];
62                         $tunniste = $item["TuoteID"];
63                         $kuvatus = $item["tuotteenkuva"];
64                         $uusitieto['tuotenimi'] = $nimi;
65                         $uusitieto['tuotteenkuva'] = $kuvatus;
66 ▾                         if($laske[$item["TuoteID"]] >= 1){
67                             $uusitieto['hinta'] = $hinta*$laske[$item["TuoteID"]];
68                         }
69                         $uusitieto['tunniste'] = $tunniste;
70 ▾                         if ($laske[$item["TuoteID"]] <= 5){
71                             $uusitieto['lkm'] = $laske[$item["TuoteID"]];
72                         }
73 ▾                         else {
74                             $uusitieto['lkm'] = 5;
75                         }
76                         array_push($tuotejson,$uusitieto);
77                         break;
78                     }
79                 }
80             }
81         }
82         //var_dump($_SESSION['kori']);
83         $unique = array_map('unserialize', array_unique(array_map('serialize', $tuotejson)));
84         $kaikki = json_encode($unique);
85         return view('tulostakori', ['kaikki' => $kaikki]);
86     }
87 ▾     else {
88         $tyhja = "korisi on tyhjä";
89         return view('tulostakori')->with('tyhja',$tyhja);
90     }
91 }
92 ▾ if (empty($_SESSION['kori'])){
93     $tyhja = "korisi on tyhjä";
94     return view('tulostakori')->with('tyhja',$tyhja);
95 }
96 }

```

Tulostakori.bladessa onload funktion avulla tutkitaan, mitä palautetaan näkyviin, joten tarvitaan erilaisia ehtoa.

@isset(\$_SESSION['kori']) eli jos sessio on olemassa ja @if (count(\$_SESSION['kori'])!= 0) eli sessio ei ole tyhjä (ei anna virhe ilmoitusta jos korista, vaikka poistaa kaikki tuotteet). Kun nämä ehdot täyttyvät laitetaan voidaan tulostaa tiedot javascriptin avulla. Tuotteet tulostetaan näkyviin createElement, setAttribute ja append menetelmien avulla. Myös yhteishinta ostoksille haetaan javascriptissä funktiolla haehinta(). Päätetään isset ja if-ehto @endif ja @endisset.

```

84 @isset($_SESSION['kori'])
85     @if (count($_SESSION['kori'])!= 0)
86     var something = <?php echo json_encode($kaikki);?>;
87     console.log(something);
88     //var div1 = document.createElement('div');
89     //div1.setAttribute("class","row");
90     var div2 = document.createElement('div');
91     div2.setAttribute("class","validys1 container");
92     //div2.append(div1);
93     $("#korissa").append(div2);
94     var korissa = jQuery.parseJSON(something);
95     $.each(korissa, function(key, value){
96         var div = document.createElement('div');
97         div.setAttribute("id",value.tunniste + "b");
98         div.setAttribute("class","Shoppingcart row align-items-center");
99         var header = document.createElement('p');
100         $(header).append(value.tuotenimi);
101         header.setAttribute("class", "header");
102         var headerdiv = document.createElement('div');
103         headerdiv.setAttribute("class", "headerdiv col");
104         $(headerdiv).append(header);
105         var hinta = document.createElement('p');
106         $(hinta).append(parseFloat(Math.round(value.hinta * 100) / 100).toFixed(2)+"€");
107         hinta.setAttribute("class","hinta");

```

Luodaan vielä näiden jälkeen tulostakori.blade:n @empty(\$_SESSION['kori']) ja päätetään sekin samaan tapaan kuin edelliset ehdot @endempty.

```

@empty($_SESSION['kori'])
var tyhja = "<?php echo $tyhja;?>";
document.getElementById("summa").innerHTML = tyhja;
var keski = document.getElementById("summa");
keski.setAttribute('class','centerkeski');
console.log("Onnistuין?");
var poistakor = document.getElementById("korissa");
poistakor.parentNode.removeChild(poistakor);
@endempty

```

Tuotteen poisto korista

Luodessani elementtejä loin poistopainikkeen, jossa on id:nä TuoteID ja jotain muita kirjaimia, jota painamalla taas aktivoidaan onclick funktio tulostakori.bladessa. Funktiossa muutetaan id kokonaisluvuksi ja lähetetään ajaxin avulla id AjaxControllerille funktioon poistakorista.

```

public function poistakorista(Request $request){
    $poistoid = $request->id;
    session_start();

    echo "poistetaan :";
    foreach (array_keys($_SESSION['kori'],$poistoid,true) as $key){
        unset($_SESSION['kori'][$key]);
    }

    $_SESSION['kori'] = array_values($_SESSION['kori']);
    var_dump($_SESSION['kori']);
    echo $poistoid;
}

```


Funktiossa on sama periaate, kuin tulosta funktiossa AjaxControllerissa. Eli foreach loopissa valitaan käsiteltäväksi \$_SESSION['kori']:ssa olevat arvot, jotka ovat saman arvoisia, kuin ajaxilla lähetetty id ja poistetaan ne taulukosta lopuksi järjestellään taulukon alkiot alkamaan nolasta array_valuesilla.

Tuotemäärän muokkaus

Bladen puolella toimii samoin tavoin scripti, kuin edellisessä eli onchange:illä tehdään funktio, joka hakee id:n ja myös muokatun arvon ja lähettää nämä ajaxilla AjaxControlleriin muutalkm funktioon.

```
public function muutalkm(Request $request){

    $varmistaid = $request->varmistaid;
    $muutalkm = $request->muuta;
    echo $varmistaid;
    $korjattu = (int)$muutalkm;
    echo $korjattu;
    session_start();
    $vaihto = array_count_values($_SESSION['kori']);
    if (in_array($varmistaid,$_SESSION['kori'])){
        $nollaa = $varmistaid;
        foreach (array_keys($_SESSION['kori'],$nollaa,true) as $key){
            unset($_SESSION['kori'][$key]);
        }

        $_SESSION['kori'] = array_values($_SESSION['kori']);
        for($i=0; $i<$korjattu; $i++){

            if($i == $korjattu){

                break;

            }
            $_SESSION['kori'][]=$varmistaid;

        }

        //var_dump($_SESSION['kori']);
        var_dump($_SESSION['kori']);
        $vaihto = array_count_values($_SESSION['kori']);
        print_r($vaihto);

        echo "toimii";
    }
}
```

Periaatteessa funktio toimii samoin kuin aiemmin ollut tulosta funktio AjaxControllerissa. Eli poistetaan kaikki tuotteet \$_SESSION['kori']:sta, jotka ovat yhtäkuin id ja luodaan niitä for-loopilla tietty määrä uudelleen. Ainoat muutokset ovat vain se, että for looppia suoritetaan niin kauan, kunnes muuttuja \$i on yhtäkuin tulostakori.bladesta saatu lukumäärän arvo.

Asiakastietolomake

Tilauksen yhteenveto näytetään samalla periaatteella, kuin aikaisemmin AjaxControllerin naytakori funktiolla, mutta tällä kertaa ne tuodaan controllerin vieyhteen funktiolla.

Asiakas syöttää omat tietonsa lomakkeeseen. Lomakkeen Valmis-painike luo tietokantaan uuden rivin Asiakas-tauluun sekä Tilaus-tauluun. Asiakas-tauluun lomakkeelta viedään kaikki tiedot, Tilaus-tauluun pelkästään aikaleima, tuoteID ja asiakasID.

<http://159.65.232.228/lomake>

Asiakkaan tiedot

Etunimi	Sukunimi
<input type="text" value="Etunimi"/>	<input type="text" value="Sukunimi"/>
Katuosoite	
<input type="text" value="Katuosoite"/>	
Postinumero	Kaupunki
<input type="text" value="Postinumero"/>	<input type="text" value="Kaupunki"/>
Puhelin	
<input type="text" value="Puhelinnumero"/>	
Email	
<input type="text" value="Email"/>	
Salasana	
<input type="text" value="Salasana"/>	
<input type="checkbox"/> Hyväksyn toimitusehdot, uutiskirjeen tilauksen ja myyn sieluni Kauppa X:lle	
<input type="button" value="Valmis"/>	

2019 Copyright Johanna Kaasalainen, Laura Alakorte, Maria Salonen, Noora Ojanen

LomakeController käyttää funktiota store, joka luo uuden rivin Asiakas-tauluun, hakee taulusta uusimman asiakasID:n ja lisää Tilaus-tauluun tarvittavat tiedot.

```
public function store(Request $request) {  
    Lomake::create($request->all());  
    return redirect('/yhteenvedo');  
    session_start();  
    var_dump($_SESSION['kori']);  
}
```

```
$PDO = DB::connection('mysql')->getPdo();  
$aika = date("Y-m-d h:i:s");
```

```
$sql = <<< SQLEND  
    SELECT MAX(AsiakasID) AS maxid from Asiakas  
SQLEND;
```

```
$newestCustomer = $PDO->prepare($sql);  
$newestCustomer->execute();  
$nc = $newestCustomer->fetchAll(PDO::FETCH_OBJ); // Muista TÄMÄ FETCH_OBJ
```

```
//return $nc[0]->maxid;
```

```
$tilaajaID = $nc[0]->maxid;  
//echo $tilaajaID;
```

```
var_dump($_SESSION['kori']);  
foreach ($_SESSION['kori'] as $tilattu) {  
    $sql = "INSERT INTO Tilaus (FK_asiakasID, FK_tuoteID, tilausaika) VALUES(:f1, :f2, :f3)";  
    echo $sql;  
    $insertsql = $PDO->prepare($sql);  
    $insertsql->execute(array(':f1' => $tilaajaID, ':f2' => $tilattu, ':f3' => $aika));  
}  
}
```

Lomaketta tehdessä vastaan tuli ongelma; Laravel tekee automaattisesti lomakkeen lähetyksen yhteydessä updated_at ja created_at – leimat tietokantaan, ja meidän tietokannastamme ne puuttuivat. Kolumnit lisättiin allaolevilla käskyillä:

```
mysql> ALTER TABLE Asiakas ADD updated_at TIMESTAMP AFTER salasana;  
Query OK, 0 rows affected (0.06 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

Asiakas-taulun kuvaus:

```
mysql> ALTER TABLE Asiakas ADD updated_at TIMESTAMP AFTER salasana;  
ERROR 1060 (42S21): Duplicate column name 'updated_at'  
mysql> ALTER TABLE Asiakas ADD created_at TIMESTAMP AFTER updated_at;  
ERROR 1067 (42000): Invalid default value for 'created_at'  
mysql> ALTER TABLE Asiakas ADD created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP AFTER updated_at;  
Query OK, 0 rows affected (0.04 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> desc Asiakas;  
+-----+-----+-----+-----+-----+-----+  
| Field | Type | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+-----+  
| AsiakasID | int(11) | NO | PRI | NULL | auto_increment |  
| etunimi | varchar(45) | NO | | NULL | |  
| sukunimi | varchar(45) | YES | | NULL | |  
| osoite | varchar(45) | YES | | NULL | |  
| postinumbero | varchar(40) | YES | | NULL | |  
| kaupunki | varchar(45) | YES | | NULL | |  
| puhnmr | varchar(45) | YES | | NULL | |  
| sposti | varchar(45) | YES | | NULL | |  
| salasana | varchar(64) | YES | | NULL | |  
| updated_at | timestamp | NO | | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |  
| created_at | timestamp | NO | | CURRENT_TIMESTAMP | |  
+-----+-----+-----+-----+-----+-----+  
11 rows in set (0.00 sec)
```

```
mysql> SELECT * from Asiakas;  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| AsiakasID | etunimi | sukunimi | osoite | postinumbero | kaupunki | puhnmr | sposti | salasana | updated_at | created_at |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 1 | Johanna | Kaasalainen | Piippukatu 2 | 40100 | Jyväskylä | 0441234321 | 14062@student.jamk.fi | NULL | 2019-04-03 07:20:00 | 2019-04-03 07:37:12 |  
| 2 | Laura | Ala-Korte | Valjankatu 5 | 40600 | Jyväskylä | +358451348730 | L4507@student.jamk.fi | NULL | 2019-04-03 07:20:00 | 2019-04-03 07:37:12 |  
| 3 | Maria | Salonen | Aapelinpolku 2 B 4 | 40270 | Palokka | +358504110663 | L4644@student.jamk.fi | NULL | 2019-04-03 07:20:00 | 2019-04-03 07:37:12 |  
| 4 | Noora | Ojanen | Puistokatu 18 | 40100 | Jyväskylä | +358443060366 | M0313@student.jamk.fi | NULL | 2019-04-03 07:20:00 | 2019-04-03 07:37:12 |  
| 5 | Teppo | Testiasiakas | Testitie 1 | 01000 | Testimaa | 0401234567 | teppo@testiasiakas.fi | salasana | 2019-04-03 07:38:01 | 2019-04-03 07:38:01 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
5 rows in set (0.00 sec)
```

```
mysql>
```

```
mysql> select * from Asiakas
-> ;
```

AsiakasID	etunimi	sukunimi	osoite	postinumero	kaupunki	puhnr	sposti	salasana	updated_at
1	Johanna	Kaasalainen	Piippukatu 2	40100	Jyväskylä	0441234321	l4062@student.jamk.fi	NULL	2019-04-03 07:20:00
2	Laura	Ala-Korte	Valjankatu 5	40600	Jyväskylä	+358451348730	L4507@student.jamk.fi	NULL	2019-04-03 07:20:00
3	Maria	Salonen	Aapelinpolku 2 B 4	40270	Palokka	+358504110663	L4644@student.jamk.fi	NULL	2019-04-03 07:20:00
4	Noora	Ojanen	Puistokatu 18	40100	Jyväskylä	+358443060366	M0313@student.jamk.fi	NULL	2019-04-03 07:20:00
5	Teppo	Testiasiakas	Testitie 1	01000	Testimaa	0401234567	teppo@testiasiakas.fi	salasana	2019-04-03 07:38:01
6	Teppo	Testiasiakas	Testitie 1	01000	Testimaa	0401234567	teppo@testiasiakas.fi	salasana	2019-04-10 05:20:39
7	Teppo	Testiasiakas	Testitie 1	01000	Testimaa	0401234567	teppo@testiasiakas.fi	salasana	2019-04-10 05:36:02

Tilaus-taulun kuvaus:

```
mysql> select * from Tilaus;
```

TilausID	FK_asiakasID	FK_tuoteID	tilausaika
3	1	2	2019-03-27 06:50:05
4	2	1	2019-03-27 06:55:18
5	1	1	2019-03-27 06:55:26
6	1	1	2019-04-10 06:34:49
7	1	1	2019-04-10 06:35:59
8	1	2	2019-04-10 06:35:59
9	1	1	2019-04-10 06:59:00
10	1	2	2019-04-10 06:59:00
11	1	1	2019-04-15 12:26:33
12	1	2	2019-04-15 12:26:33
13	39	1	2019-04-15 12:27:38
14	39	2	2019-04-15 12:27:38
15	40	1	2019-04-15 12:33:58
16	40	2	2019-04-15 12:33:58
17	42	1	2019-04-15 12:37:57
18	42	2	2019-04-15 12:37:57

16 rows in set (0.00 sec)

Tilauksen yhteenveto

Yhteenveto-sivu on asiakastietolomaketta seuraava näkymä tilausprosessissa. Näkymään tulostetaan tilauksen aikaleima, asiakkaan tiedot sekä tilatut tuotteet. Näkymää tehdessä pohdimme erilaisia vaihtoehtoja tietojen tulostukseen. Ensimmäinen vaihtoehto olisi ollut tulostaa tilauksen tiedot Tilaus-taulusta, mutta kysely olisi vaatinut niin paljon liitoksia ja ollut turhan monimutkainen. Päädyimme hakemaan asiakas-taulusta isoimman asiakasID:n ja tulostamaan sen perusteella id:tä vastaavan asiakkaan nimi-, osoite-, puhelinnumero- ja sähköpostirivit sekä created_at-aikaleiman json-taulukkoon. Ostokset tulostimme suoraan ostoskorin sessio-muuttujasta samalla periaatteella, kuin aikasemmin naytakori ja vieyhteen funktioilla AjaxControllerissa. Sessio tuhoetaan lopussa(session_destroy()), jotta saadaan ostoskori tyhjäksi.

Tulostukset toteutimme yhdellä yhteenveto-nimisellä funktiolla:

```

public function yhteenveto() {
//haetaan viimeksi lisätty asiakasID
$PDO = DB::connection('mysql')->getPdo();
$aika = date("Y-m-d h:i:s");

$sql = <<< SLEND
    SELECT MAX(AsiakasID) AS maxid from Asiakas
SLEND;

$newestCustomer = $PDO->prepare($sql);
$newestCustomer->execute();
$nc = $newestCustomer->fetchAll((\PDO::FETCH_OBJ)); // Muista TÄMÄ FETCH_OBJ

$tilaajaID = $nc[0]->maxid;

$sql = <<< SLEND
    SELECT * from Asiakas WHERE asiakasID = :tilaajaID
SLEND;
    $sqlresult = $PDO->prepare($sql);
    $sqlresult->execute(array(':tilaajaID' => $tilaajaID));
    $tilaaja = $sqlresult->fetchAll((\PDO::FETCH_OBJ));

$ostaja = $tilaaja[0];
$ostaja = json_decode(json_encode($ostaja), true);

/*$id = DB::getPdo()->lastInsertId();;
$tilaukset = Tilaus::All();
return view('tilaus')->with('tilaukset', $tilaukset);*/

session_start();

if (isset($_SESSION['kori'])){

    if (count($_SESSION['kori'])!=0){

        $tuotejson=array();
        $montako = 0;
        $laske = array_count_values($_SESSION['kori']);

        foreach ($_SESSION['kori'] as $osto) {
            $ostokset = Korissa::all();
            $json = $ostokset;
            foreach ($json as $item) {
                if ($item["TuoteID"] == $osto) {

```

```

        $nimi = $item["tuotteennimi"];
        $hinta = $item["hinta"];
        $tunniste = $item["TuoteID"];
        $kuvatus = $item["tuotteenkuva"];
        $uusitieto['tuotenimi'] = $nimi;
        $uusitieto['tuotteenkuva'] = $kuvatus;
        if($laske[$item["TuoteID"]] >= 1){
            $uusitieto['hinta'] = $hinta*$laske[$item["TuoteID"]];

        }

        $uusitieto['tunniste'] = $tunniste;
        if ($laske[$item["TuoteID"]] <= 5){
            $uusitieto['lkm'] = $laske[$item["TuoteID"]];
        }
        else {
            $uusitieto['lkm'] = 5;
        }
        array_push($tuotejson,$uusitieto);
        break;
    }
}
//var_dump($_SESSION['kori']);
$asiakasjson=array();
$etunimi = $ostaja["etunimi"];
$sukunimi = $ostaja["sukunimi"];
$uusiostaja['etunimi'] = $etunimi;
$uusiostaja['sukunimi'] = $sukunimi;

array_push($asiakasjson,$ostaja);
$ostaja = json_encode($asiakasjson);
$unique = array_map('unserialize', array_unique(array_map('serialize', $tuotejson)));
$kaikki = json_encode($unique);

session_destroy();
return view('yhteenveto', ['kaikki' => $kaikki], ['ostaja' => $ostaja]);
    }
    else {
        $tyhja = "korisi on tyhja";
        return view('yhteenveto')->with('tyhja',$tyhja);
    }
}

```

```

if (empty($_SESSION['kori'])){
    $tyhja = "korisi on tyhja";
    return view('yhteenveto')->with('tyhja',$tyhja);
}
return view('yhteenveto');
}

```

Kuvakaappaus yhteenveto-näkymästä:

Kauppa X NAiset MIEHET

Tilauksen yhteenveto

Tilaus tehty : 2019-04-17 06:59:00

Omat tietosi:



ewgt Päivä

Umpikuja 5 15644 Kaupunki

asdasdasda

4564856

Tilaamasi tuotteet:

	Adidaksen verkkarit	39.90	2
	Pusero	20.95	1

	Farkut	50.95	1
	T-paita	29.95	1
ostokset yhteensä: 242.86euroa			

[Palaa etusivulle](#)

Lista Laravelin muokatuista tiedostoista

 [app/Korissa.php](#)

 [app/Lomake.php](#)

 [app/controllers/AjaxController.php](#)

 [app/controllers/EtusivuController.php](#)


 [app/controllers/FilterController.php](#)


 [app/controllers/FilterMiehetController.php](#)

 [app/controllers/FilterNaisetController.php](#)

 [app/controllers/LomakeController.php](#)

 [app/controllers/MiehetController.php](#)

 [app/controllers/NaisetController.php](#)

 [app/controllers/SearchController.php](#)

 [app/controllers/TilausController.php](#)

 [app/controllers/YhteenvetoController.php](#)

- + [resources/view/etusivu.blade.php](#)
- + [resources/view/filter.blade.php](#)
- + [resources/view/filtermiehet.blade.php](#)
- + [resources/view/filternaiset.blade.php](#)
- + [resources/view/layouts/app.blade.php](#)
- + [resources/view/lomake.blade.php](#)
- + [resources/view/miehet.blade.php](#)
- + [resources/view/naiset.blade.php](#)
- + [resources/view/search.blade.php](#)
- + [resources/view/tilaus.blade.php](#)
- + [resources/view/tulostakori.blade.php](#)
- + [resources/view/yhteenveto.blade.php](#)
- + [routes/web.php](#)

Itsearviot

Oma arvio työstä muutamalla virkkeellä. Listaus projektin osista, jotka tehnyt. Arvosanaehdotus/tekijä. Onnistumiset, puutteet, huomioita.

Johanna

Meillä jakaantui todella onnistuneesti osaamisalueet ja mielenkiinnon kohteet tämän työn saralla. Olen itse ollut entistä kiinnostuneempi back endistä tämän vuoden puolella ja sain toteuttaa ensimmäisen isomman projektin aiheeseen liittyen. Tietokantojen pyörittely ja Linuxin komentokehotteessa pyöriminen ovat olleet mielekästä puuhaa. Siksi pääsinkin toteuttamaan meidän työn ei näkyvän puolen lähes täysin kokonaisuutena.

Opin hallitsemaan suurempia kokonaisuuksia, pyörimään Linuxin sisällä sekä käyttämään Laravellia. Pyörin avustamassa vähän jokaisessa vaiheessa oman osuuteni loputtua. Tein grafiikkaa, päivittelin tietokantaa tarvittaessa ja yritin olla joka paikan höylänä loppuajasta.

Alun perin minun oli tarkoitus tehdä myös toivelista työtämme varten, mutta totesimme yhdessä ettei sille jää enää tarpeeksi aikaa. Sen sijaan tein hakupalkin navigointipalkkiin.

Ryhmämme toimi mielestäni aktiivisesti työn tavoitteiden eteen, ei hoppuillut turhia ja osasi karsia tarvittaessa työn laajuudesta.

Arvosanaehdotus: 4

Toteutetut projektin osat:

- Tietokannan suunnittelu (kaikki yhdessä)
- Tietokannan toteutus käytännössä ensin workbenchissä, sitten Digital oceanissa
- Digital Ocean dropletin tilaus ja koneen esiasennus
- Samban asennus sekä käyttöönotto
- Laravellin asennus sekä käyttöönotto
- Päälayoutin asentaminen (Noora teki ulkoasun rungon perinteiseen tyyliin ensin)
- Hakupalkki
- Kuvat sivustolle

Noora

Olen tyytyväinen siihen mitä saimme ryhmänä aikaan. Meillä olisi ollut vielä monta ideaa verkkokaupan laajentamiseksi (esim. toivelista, tuotearviointi ja Admin-näkymä), mutta emme ehtineet toteuttaa niitä rajallisen ajan takia.

Olin iloinen siitä, että sain tehdä näkymiä, koska tyylittely ja esillepano on minulle mieleistä puuhaa. Suodatuspalkin teko oli mielenkiintoista, koska en aiemmin tiennyt miten sellainen toimii. Opin käyttämään DigitalOceania, Laravellia ja php:tä ja osaan nyt käyttää oikeanlaisia sql –lausekkeita. Ymmärrän mitä tarvitaan, kun halutaan tehdä suurempi verkkosivusto.

Olisin halunnut kokeilla tehdä suodatusta Reactilla, mutta se jää minulle itselle opeteltavaksi. Hinnan suodatuspalkki olisi voinut olla yhtenäinen.

Arvosanaehdotus: 4

Toteutetut projektin osat:

- Tietokannan suunnittelu (kaikki yhdessä)
- Etusivu-näkymä
- Naiset-näkymä
- Miehet-näkymä
- Suodatus
- Suodatuksen jälkeiset näkymät (kaikki, naiset, miehet)

Laura

Omasta mielestäni työ onnistui hyvin ja kehitti hyvin myös omaa osaamista. Opin käyttämään Laravellia sekä koodaamaan php:llä paremmin. Etenkin tässä työssä oppi myös miettimään paremmin etukäteen, mitkä funktiot ovat tarpeellisia, jotta käyttäjällä olisi tarvittavat työkalut sivustolla.

Toki, kun katsoo omaa koodia jälkikäteen tuntuu siltä, että olisi voinut tehdä asioita eri tavoin esimerkiksi painikkeista, joista tuote lisätään ostoskoriin olisi voinut tehdä button elementtejä eikä a elementtejä, jolloin oltaisiin välttytty siltä, että sivu pomppii aina, kun niitä painaa(tämä on kuitenkin vain "kauneus virhe").

Myös session taulukon käsittelyn olisi varmasti voinut tehdä paremmin. Esimerkiksi jos ostoskorissa olevan päällimmäisen tuotteen lukumäärää muokkaa(korissa oltava siis enemmän tuotteita kuin yksi), niin siirtyy se alimmaiseksi, mikä ei ole omaan silmään mikään kaunis näky. Tältä oltaisiin voitu välttyä varmaankin sillä, että oltaisiin muokattu session taulukon ulkoasua erilaiseksi esimerkiksi siten, että oltaisiin tallennettu session taulukkoon TuoteID:n lisäksi myös lkm, eikä monta kertaa samaa TuoteID:tä. Mutta pääasialhan oli, että tuli opittua jotakin ja se kyllä tuli tehtyä.

Arvosanaehdotus 4

Toteutetut projektin osat:

- Tietokannan suunnittelu (kaikki yhdessä)
- Kaikki ostoskoriin liittyvät asiat (paitsi navigointi palkissa oleva ostoskori-nappula)
- Lomake-sivulla ostoskorin yhteenveto
- Yhteenveto-sivu yhdessä Marian kanssa

Maria

Työ onnistui mielestäni hyvin; sain kaikki suunnittelemani ominaisuudet toimimaan. Pääsin hyvin sisään Laravel-sovelluskehiksen toimintaan ja rakenteeseen sekä opin paljon uutta.

Arvosanaehdotus 3

Toteutetut projektin osat:

- Tietokannan suunnittelu (kaikki yhdessä)
- Lomake-sivu
- Yhteenveto-sivu yhdessä Lauran kanssa