Lea Äyhynmäki

SOVELLUKSEN TOTEUTUS KOLMI-KERROSMALLILLA

Opinnäytetyö

Liiketalouden ammattikorkeakoulututkinto

Tietojenkäsittely

2020

Kaakkois-Suomen
ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tekijä/Tekijät	Tutkinto	Aika
Lea Äyhynmäki	Tradenomi (AMK)	Marraskuu 2020
Opinnäytetyön nimi Sovelluksen toteutus kolmikerrosmallilla		43 sivua 1 liitesivua
Toimeksiantaja InStore Media Oy		
Ohjaaja Arto Väätäinen		

Tiivistelmä

Tämän opinnäytteen tavoitteena oli perehtyä sovelluskehityksessä käytettävään käsitteeseen *kolmikerrosmalli* ja toteuttaa toimeksiantona tilityssovellus InStore Media Oy:lle. Yritys on erikoistunut myymälämainontaan, joten sovellus tulee käyttöön myymälämainonnan laskujen luonnissa, jotka asiakas toteuttaa käsin.

Kolmikerrosmallin mukainen sovellusarkkitehtuuri koostuu kolmesta tasosta. Kolmikerrosmallin rakenne erottaa tasot toisistansa ja sen takia mallilla toteutettuja ohjelmia on helppo kehittää erikseen ilman, että joutuu tekemään kahteen muuhun tasoon muutoksia. Kolmikerrosmallin kolme tasoa ovat esitys-, tehtävä- ja tietotaso.

Tilityssovellus toteutuksessa hyödynnettiin kolmikerrosmallia. Sovelluksen kehityksessä käytettiin kolmikerrosmallin tasojen toteutukseen eri frameworkejä, rajapintoja, ohjelmointikieliä ja lisäkirjastoja esim. Bootstrap, jQuery ja SQL.

Sovelluskehitys aloitettiin keräämällä tietokantaan tulevat tiedot ja suunnittelemalla tietokannan rakenne. Tämän jälkeen toteutettiin rautalankamalli sovelluksesta, jonka avulla ohjelmoitiin prototyyppi sovelluksesta. Prototyyppiin lisättiin toiminnallisuuksia, näkymiä ja tietokantakyselyt, joista muodostui lopullinen sovellus.

Työssä kehitettiin web-hotellissa toimiva sovellus, joka luo tulostettavan PDF-laskun kauppakohtaisesti toteutuneiden mainoskampanjoiden ja valitun ajanjakson mukaan. Laskua luodessa voi määrittää käytettävän laskupohjan ja kampanjat, jotka tilitetään. Sovelluksessa voi myös luoda ja hallita laskuun tulevia kauppoja ja kampanjoita.

Toimeksianto auttoi muodostamaan konkreettisen kuvan siitä, mitä kaikkea sovelluksen julkaisemiseen internetissä vaaditaan. Suurimpana yllätyksenä tuli, että suurin osa web-hotelleista tuki ainoastaan PHP-ohjelmointikieltä palvelin puolella. Työ korosti suuresti sitä, että vaikka suunnittelu vaiheessa kokee ottaneen huomioon kaiken ja rajaavansa työn, niin yllätyksiä ja lisävaatimuksia voi aina ilmentyä työn edetessä ja jopa viime metreillä.

Asiasanat

www-pohjainen palvelu, palvelin, rajapinnat, ohjelmointi, PHP, HTML, CSS, JavaScript, SQL

Kaakkois-Suomen ammattikorkeakoulu

Author (authors)	Degree	Time
Lea Äyhynmäki	Bachelor of Business Administration	November 2020

Thesis title

Application development
Using the 3-tier architecture

43 pages
1 pages of appendices

Commissioned by

InStore Media Oy

Supervisor

Arto Väätäinen

Abstract

The objective of the thesis was to become familiar with the concept of 3-tier architecture in application development and to develop a commissioned application for accounting. The thesis commissioner was InStore Media Oy, specializing in in-store advertising. The accounting application will be used for creating invoices for the advertising campaigns made by hand.

The 3-tier architecture is an application composed of three layers. This architecture separates the layers from each other, and because of this the layers can be developed without making any changes to the remaining two. These three layers are the presentation layer, the business layer and the data layer.

The 3-tier architecture was used in the development of the accounting application. Techniques such as frameworks, interfaces, programming languages and additional libraries were used in the development of the application, e.g. Bootstrap, jQuery and SQL.
The development of the application started with collecting all the necessary data for the database and planning the structure of the database. After that, wireframes were made and used to program a prototype of the application. The application was developed into its final version by adding functions, views and database queries to the prototype.

The resulting application is hosted by a web hosting service. The application generates a store specific PDF invoice which includes completed advertising campaigns during a chosen time period. While creating an invoice the base for the invoice can be determined and campaigns for accounting can be chosen. The application also gives the possibility to create and manage stores and advertising campaigns that appear in the invoice.

The commission gave a clear understanding of what was needed for hosting an application on the web. The biggest surprise was that most of the web hosting services only supported PHP as a programming language on the server side. The commission also emphasized the fact that even considering and defining everything coming to mind while planning, there could always be surprises and additional requirements emerge during development, even in the last stages.

Keywords

web-based service, server, interfaces, programming, PHP, HTML, CSS, JavaScript, SQL

SISÄLLYS

1 JOHDANTO

Tämän opinnäytteen tavoitteena on perehtyä sovelluskehityksessä käytettävään käsitteeseen *kolmikerrosmalli*. Työssä esitellään malliin kuuluvat vaiheet, tekniikat ja ohjelmointikielet. Näitä edellä mainittuja asioita hyödyntäen toteutetaan toimeksianto InStore Media Oy:lle, jonka eri työvaiheita käydään läpi kolmannessa luvussa.

Luvussa kaksi kerrotaan mikä on kolmikerrosmalli ja mistä se koostuu. Luvussa tutustutaan kolmikerrosmallin eri tasoihin ja niiden sisältöihin. Luvun aliluvuissa perehdytään tarkemmin tasojen sisältöön ja ohjelmointikieliin ja sen tekniikoihin mitä tasossa yleisesti käytetään. Ohjelmointikielissä ja tekniikoissa keskitytään esittelemään ne, joita käytetään toimeksiannossa.

Kolmannessa luvussa esitellään toimeksiantaja ja toimeksianto. Esittelyn jälkeen kerrotaan toimeksiantajan käyttötarpeesta sovellukselle ja esitellään sovelluksen toiminnallisuudet. Tämän jälkeen kerrotaan työn etenemisestä prosessikuvauksena, joka on jaoteltu kolmikerrosmallin tasojen mukaisesti. Kolmannen luvun viimeisessä aliluvussa esitellään kuvien kautta lopullinen sovellus, joka luovutettiin toimeksiantajalle. Lopullista tulosta verrataan luvussa aikaisemmin esiteltyihin suunnitelmiin ja versioihin, josta voi mahdollisesti nähdä sovelluksen kehityksen työnteon aikana.

Neljännessä luvussa käsitellään, miten oppinäytetyön tavoite saavutettiin, mitä työntekijä oppi toimeksiannosta ja mitä mahdollisia esteitä tai yllätyksiä toimeksiannon teon aikana tuli vastaan. Luvussa kerrotaan myös, miten työ onnistui ja käydään läpi ominaisuudet ja toiminnallisuudet, joita ei ehditty toteuttaa tai jotka jäivät vajaiksi.

Toimeksiannon teko aloitettiin tammikuussa 2020. Toimeksiantajan toive oli, että sovellus olisi valmis ja käytettävissä toukokuussa 2020. Toimeksianto oli alkuperäisesti erittäin laaja, jota sitten rajattiin toimeksiantajan tarpeen mukaan. Tässä työssä toteutettava työ on vain yksi pienempi osuus alkuperäisestä toimeksiannosta.

Toimeksiannon teon aikana vertailtiin eri web-hotelleja ja vastaavia vaihtoehtoja toimeksiantajalle, jota ei käsitellä tässä työssä.

Työn kolmannessa luvussa käytetään sovelluskehitykseen kuuluvia suunnittelu tekniikoita kuten rautalankamallit, prototyypit ja UML-kaavio, jota ei kuitenkaan tarkemmin käsitellä tässä työssä.

2 KOLMIKERROSMALLI

Kolmikerrosmallin mukainen sovellusarkkitehtuuri koostuu kolmesta kerroksesta tai tasosta. Tätä mallia käytetään sovelluksissa, jossa on tietyn tyyppinen asiakas – palvelin (eng. Client-server) rakenne. Tämä rakenne erottaa tietyllä tavalla kerrokset toisistaan ja niitä voi kehittää erikseen. Koska tasot ovat erillään niin se auttaa vähentämään suoritusongelmia. Nämä kolme kerrosta ovat esitystaso, tehtävätaso ja tietotaso (eng. Presentation layer, business layer and data layer).

Esitystaso kuuluu front end puoleen, joka sisältää graafisen käyttöliittymän, joka on saatavissa selaimen kautta. Taso esittää tarpeelliset tiedot käyttäjälle ja sen teossa käytetään teknologioita kuten HTML5, JavaScript ja CSS. Tehtävätaso tai sovellustaso sisältävät sovelluksen toimintalogiikan ja ylläpitävät sen ydinkyvyt. Tason teossa käytetään muun muassa Java, .NET, C#, Python tai C++.

Tietotaso sisältää saatavuuden tietokantaan tai tiedon tallennusjärjestelmän. Nämä järjestelmät voivat olla MySQL, Oracle, PostgreSQL, Microsoft SQL Server tai MongoDB. Tieto on saatavissa API kutsujen kautta, joka tehdään sovellus tasossa. (Logi Report s.a.)

Kolmikerrosmallin määrittää se, että esittelytaso ei voi ottaa suoraan yhteyttä tietotasoon, vaan sen pitää keskustella tietotasolle tehtävätason kautta. Tämän takia on helppoa korvata jokin osa yhdestä tasosta ilman, että tekee muutoksia kahteen muuhun. (Marston 2012.) Microsoftin (2018) mukaan kolmikerrosmallin vahvuuksia ovat, että sitä tai sen osia voi uudelleen käyttää, kasvattaa, sitä on helppo ylläpitää ja muokata ja se on hyvin joustava. Isoissa

monimutkaisissa projekteissa malli auttaa pilkkomaan projektia pienempiin osiin.

2.1 Esitystaso

Esitystaso on kolmikerrosmallin ylin taso. Ylin taso ohjelmasta on käyttöliittymä. Tason tehtävä on kääntää tehtävät ja tulokset sellaiseen muotoon, että käyttäjä ymmärtää ne. (Rishabh 2020.) Yleensä tätä tasoa myös märitellään front endiksi, joka keskittyy verkkosivun esillepanoon ja siihen, kuinka tieto esitetään sivulla selaimessa ja älylaitteilla. Kaikki mitä voit nähdä sivulla, esimerkiksi ulkonäkö ja sommittelu teksteistä, kuvista, väreistä, fonteista, napeista ja niin edelleen ovat asioita mitä front end kehittäjän pitää ottaa huomioon. (An 2020.) Esittelytaso usein sisältää usean sivun tai näkymän jossa käyttäjä voi vaikuttaa näkymään valinnoillaan. (Microsoft 2014.)

Seuraavissa aliluvuissa käsitellään toimeksiannossa käytettäviä front end tekniikoita ja esitellään mitä ne ovat, niiden taustoja, sisältöä ja miten niitä käytettään.

2.1.1 HTML

HTML (Hypertext Markup Language) on merkintäkieli, jota käytetään selittämään verkkosivun rakennetta. HTML:n tehtävä on selittää mikä tekstistä on otsikko, missä kappale alkaa ja missä se päättyy sekä mikä osa tekstistä pitäisi olla luettelossa. Tällä kielellä voi esittää linkkejä eri sivujen välillä, missä kuvat ja videotallenteet pitäisi näkyä, sekä lomakkeen täyttökohdat. (Larsen & Duckett 2013, xxxiii.)

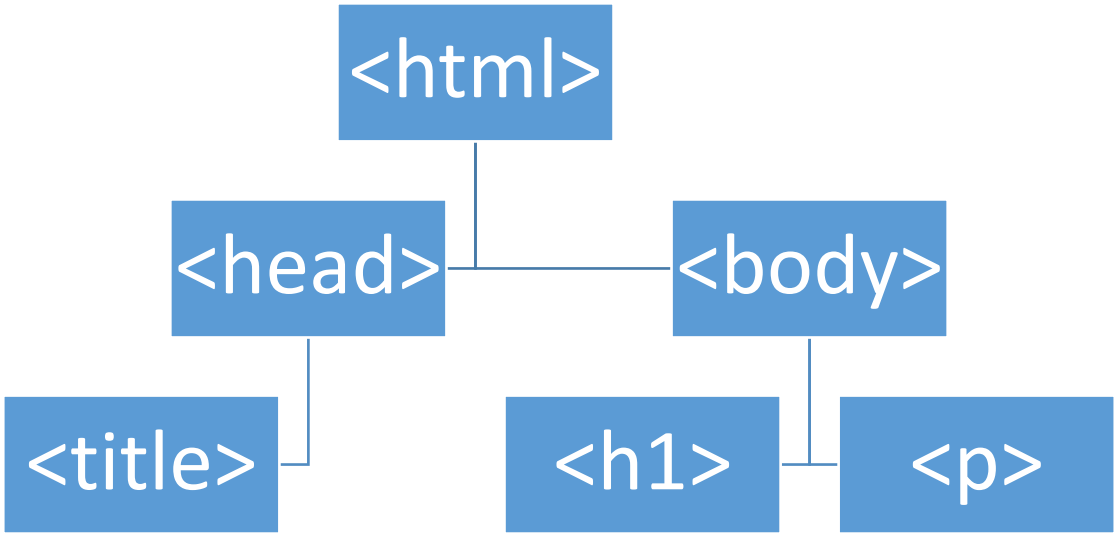
Herra Tim Berners-Lee loi internetin vuonna 1989 ja vuonna 1990 hän loi ensimmäisen verkkosivun, joka oli tarkoitettu jakamaan tiedostoja, jotka olivat enimmäkseen tieteellisiä artikkeleita. Hän toi tekstin paperi muotoon niin kutsutusta mustasta ruudusta. Hän kehitti kielen, joka ei ottanut kantaa fonttiin tai väreihin, jotta sitä voitaisi käyttää eri alustoilla. (Mozilla Developer 2019.) Tämän takia Larsenin ja Duckettin (2013, 2–3) mukaan on helpointa verrata paperisia tulosteiden rakennetta tai tekstinkäsittelyohjelman toimintoja HTML:ään. Esimerkiksi rivinvaihto korvataan tagilla (eng. Tag). Tärkeintä on muistaa, että tageilla määritellään, mikä kyseinen osio on.

```
1  <html>
2      <head>
3          <title>Esimerkki sivu</title>
4      </head>
5      <body>
6          <h1>Ensimmäinen HTML sivu</h1>
7          <p>Tervehdys maailma!</p>
8          <p>Tässä on ensimmäinen HTML sivuni</p>
9      </body>
10 </html>
```

Kuva 1. Kuvakaappaus HTML merkintäkielestä (Visual Studio Code 2020)

Kuvassa 1 on esitelty HTML koodin rakenne hyvin yksinkertaisesti. Merkit **<** ja **>** sisältävät englanninkielisen sanan, joka kertoo verkkosivulle sivun rakenteen. Nämä ovat nimeltään tageja. Kaikki kuvassa 1 esiintyvät tagit tulevat pareissa. Larsen ja Duckett (2013, 4) kuvailevat parillisia tageja siten, että ensimmäinen on avautuva ja toinen on sulkeva. Kokonaisuuttaa: avautuva tagi, sisältö ja sulkeutuva tagi, kutsutaan nimellä elementti. Kuva 1 koostuu HTML elementistä, jonka sisällä on otsikko ja vartalo elementti. Otsikko elementin sisällä on nimike ja vartalossa on otsikko ja kaksi paragrafia.

Kuvassa 1 näkyvää koodin kokonaisuutta selain käsittelee rajapintana, jota kutsutaan DOM:iksi (Document Object Model). Tämä rajapinta määrittää dokumentin loogisen rakenteen ja millä tavalla dokumenttia voi manipuloida. Rajapinta kuvaa tageja objekteina. (Rajput 2019.) DOM:ia voidaan miettiä puuna tai metsänä, jos HTML koodin esittää hierarkia kuviona siitä tulee puumainen, katso kuva 2.



Kuva 2. Kuva 1 DOM rakenne kuvattu kuviona (Äyhynmäki 2020)

HTML:stä on viisi versiota. Vuotena 1997 julkaistiin HTML 4, ja merkintäkielen kehittäminen lopetettiin siihen ja W3C päätti keskittyä XML pohjaisen merkintäkielen kehittämiseen, nimeltään XHTML. Myöhemmin 2000-luvun alku puolella huomattiin tekniikan kehittyessä, että XML käyttöönotto verkkoteknologiana rajoittaisi uuttaa teknologiaa. W3C päättivät, että ennemmin kuin korvaisi aikaisemmin kehitettyä verkkoteknologiaa palattaisiin takaisin kehittämään HTML:ää. Teknologian vauhdikkaan kehityksen takia selain palveluntarjoajat ja W3C alkoivat tehdä yhteystyötä. Yhteistyön ydin oli, että verkkoteknologian pitää olla yhteensopivia vanhemman teknologian kanssa. Yhtiöt ja W3C määrittivät, työskentelivät ja kehittivät monta vuotta yhdessä HTML5, vuoteen 2011 asti. Nämä kaksi tahoa tavoittelivat eri asioita HTML5 suhteen, joten päättivät lopettaa yhteistyön ja jatkaa itsenäisesti. Mutta vuonna 2019 he allekirjoittivat sopimuksen siitä, että jatkossa kehittävät yhdessä yhtä ainutta HTML versiota. (WHATWG 2020.)

Gustafson (2013, 33) nostaa esille muutaman hyödyllisen tagin HTML5:ssä.

- <*article*>, määrittää artikkelin dokumentissa
- <*aside*>, määrittää sisällön, joka on muun sivun sisällöstä sivussa
- <*footer*>, määrittää alatunnisteen
- <*header*>, määrittä ylätunnisteen
- <*nav*>, Sisältää sivun navigaatio linkit
- <*section*>, määrittää osan dokumentissa

2.1.2 CSS

CSS eli Cascading Style Sheet, on tyylitiedosto, jota käytetään kuvailemaan HTML tai XML tiedoston esillepanoa. CSS kuvailee kuinka elementtien pitäisi näyttää näytöllä, paperilla tai muussa media muodossa.

CSS on yksi ydin kielestä avoimessa verkossa ja on standardi kaikissa internet selaimissa. CSS:stä on julkaistu kolme virallista versiota. Tyylitiedostoa käytettään muun muassa muuntamaan fonttia, väriä, kokoa, etäisyyttä ja väliä. Sillä voi myös luoda animaatioita tai muunlaisia koristeellisia ominaisuuksia. (MDN web docs 2020.)

Cascading style on kokoelma erilaisia sääntöjä, miten tyylit sopivat yhteen ja ylikirjoittavat toisiaan. Selain käsittelee nämä muuttujat, yhdistää ne sääntöjen mukaisesti ja esittää ne näytöllä. Tyylitiedostossa ehdotetaan selaimelle mitä halutaan saavuttaa. CCS on vain valinnainen taso, jonka voi halutessaan lisätä verkkosivulle. Verkkosivu ei tarvitse tyylitiedostoa toimiakseen, joten se on kehitetty niin että se ei riko sivun esittämistä, vaikka se otettaisiin pois käytöstä tai jokin osa ei siinä toimisi. Valinnaisuuden takia selain tekee niin sanotusti omat päätöksensä miten sivu näytettään eri tilanteissa ottaen huomioon tyylitiedostossa määritetyt asiat. (Mozilla Developer 2019.)

```
1  body {
2      background-color: #000;
3  }
4
5  h1 {
6      font-family: Impact;
7      padding-bottom: 3px;
8      color: red;
9  }
10
11 p {
12     text-align: right;
13 }
```

Kuva 3. Kuvakaappaus CSS tyylitiedostosta (Visual Studio Code 2020)

Kuvassa 3 on esitetty osa tyylitiedostoa. Keltaiseksi värjättyä tekstiä kutsutaan valitsijaksi (eng. Selector). Sulkujen sisällä on deklaraatio, jossa ensimmäinen on ominaisuus ja toinen on arvo. HTML tagit voivat myös sisältää tyylitietoa esimerkkinä kuvassa 3 p-tag HTML koodissa olisi: * … *. (Larsen & Duckett 2013, 192.) Erillisessä tyylitiedostossa voit määrätä tietyn tai tiettyjen elementtien ulkonäön kutsumalla valitsijan tiedostossa luokka-, id- tai tagilla.

CSS frameworkit ovat työkaluja, jotka on kehitetty käyttäjä rajapinnan kehittäjille helpottamaan heidän työtänsä. Frameworkeillä voi nopeasti toteuttaa rajapinnan ja isomassa työporukassa sillä on helppo kehittää yhtenäinen teema, joka on käytettävissä useammassa kuin yhdessä projektissa. Framework on kokoelma CSS-tyylitiedostoja, jotka ovat valmiita käyttöön ja jota on laajennettu JavaScript koodilla ja SASS (Syntactically Awesome Style Sheet) lisäkirjastolla. (Lawernce 2019.)

Bhardwajn (2020) mukaan kuusi suosituinta frameworkkiä ovat:

- Bootstrap
- Foundation
- Materialize
- Semanic UI
- Bulma
- Tailwind

Bootstrap on Twitterin kehittäjien kehittämä framework, vuodelta 2011. Bootstrapin käyttö on ilmaista ja se on vapaata lähdekoodia. Framework tarjoaa resposiivisuutta, sisään rakennettuja kirjastoja, ison yhteisön ja välttää selain yhteensopivuusongelmat.

Foundation on framework jota käytetään eniten yrityssivujen kehityksessä. Framework tarjoaa vahvan tuen sähköpostien muotoiluun. Foundationia on helppo muokata ja siinä on responsiivisiä HTML pohjia.

Materialize on Googlen esittämä framework, joka pohjautuu Material Designiin. Materializessa on sisään rakennettuja ominaisuuksia, laaja valinta lisäosista ja se on helppokäyttöinen. (Bahardwaj 2020.)

Tässä työssä tulemme hyödyntämään Bootstrap frameworkiä.

2.1.3 JavaScript

JavaScript on oliopohjainen ohjelmointikieli. Jokaisella oliolla (eng. Object) on ominaisuuksia, jotka kuvailevat siihen kuuluvia osia. Oliolla on myös metodeja. Nämä kuvaavat toimintoja, jota voi tehdä olioilla tai tehdä oliolle. (Larsen & Duckett 2013, 340–341.)

JavaScript on ohjelmointikieli, jolla voi:

- Lukea elementtejä dokumentista ja kirjoittaa uusia elementtejä ja tekstiä dokumenttiin
- Manipuloida tai siirtää tekstiä
- Suorittaa matemaattisia laskelmia
- Reagoida tapahtumiin, kuten käyttäjä painaa nappia
- Palauttaa käyttäjän tietokoneen tämänhetkisen päivämäärän ja kellon ajan tai ajankohdan, milloin dokumenttia on viimeksi muokattu
- Määrittää käyttäjän näytön koon, selainversion tai näytön resoluution
- Suorittaa tapahtumapohjaisia ehtoja, kuten ilmoittaa käyttäjälle, jos syötetty tieto lomakkeessa on viallinen

Olio pohjainen ohjelmointi tunnetaan lyhenteellä OOP (Object-Oriented Programmin) ja ohjelmaa on helppo kuvailla puhekielellä, kun kuvaillaan mitä olioilla halutaan tehdä. Stefanov ja Sharman (2018, 32-33) kuvailevat OOP:n lauseita kieliopin kautta. Heidän mukaansa oliot koostuvat substantiivista, adjektiivistä ja verbistä. Eli asiasta, sen ominaisuudesta ja toiminnasta.

JavaScriptissä alustetaan tai esitellään olio muuttujaan ohjelmointilauseella: *var olio = ”arvo”;*. Olio voi olla myös tyhjä, jolloin olio esitellään ohjelmointilauseella: *var olio;*.

Stefanov ja Sharman (2018, 53) kertovat että JavaScriptissä on viisi perustietotyyppiä.

1. Numerot. Sisältäen kokonaisluvut ja liukuluvut
2. Merkkijonot (eng. String). Nämä merkitään lainausmerkkien sisälle ohjelmointilauseessa
3. Totuusarvot. Nämä ovat true ja false
4. Määrittelemättömät (eng. Undefined). Tämä tietotyyppi on palautusarvo, jos ohjelmassa yrittää kutsua muuttujaa, jota ei ole esitelty tai sillä ei ole arvoa.
5. Tyhjä (eng. Null). Toisin kuin määrittelemätön, tyhjä on arvo.

Olio voi sisältää operaattorin. Nämä operaattorit ovat sellaisia, jotka sisältävät laskutoimituksen. Operaattoreita on myös loogisia, joita käytetään ehtolauseissa. Nämä ovat:

- ! , ei operaattori. Esim. !olio. Oliossa ei ole arvoa
- && , ja operaattori. Esim. olio > 2 && olio < 4. Olio on suurempi kuin numeerinen arvo kaksi ja pienempi kuin numeerinen arvo neljä
- || , tai operaattori. Esim. olio == 3 || olio == 6. Olio on yhtä kuin numeerinen arvo kolme tai kuusi

(Stefanov & Sharman 2018, 64.)

Yllä olevassa listauksessa on näkyvillä muutama vertailu operaattori. Kyseiset operaattorit ovat ==, > ja <. Muita vertailu operaattoreita ovat:

- === , yhtä kuin ja tarkistaa että oliot ovat samaa tietotyyppiä
- != , ei ole yhtä kuin
- !== , eivät ole yhtä kuin ja tarkistaa että oliot eivät ole samaa tietotyyppiä
- >= , isompi kuin tai yhtä kuin
- <=, pienempi kuin tai yhtä kuin

(Stefanov & Sharman 2018, 69.)

Vertailu- ja loogisia operaattoreita hyödynnetään usein ehtolauseissa. Ehtolauseiden lisäksi JavaScript koodissa on myös toistolauseita, jossa hyödynnetään edellä esitettyjä operaattoreita.

JavaScriptissä on olemassa monta kirjastoa kuten esimerkiksi Vue, React ja Angular. Mutta tässä toimeksiannossa tullaan käyttämään kirjastoa nimeltään jQuery. Tätä kirjastoa kutsutaan käyttämällä $-merkkiä kun ohjelmoi. jQuery lisää JavaScriptiin tapahtumakäsittelyä ja tukee AJAX:ia (Asynchronous JavaScript And XML). Näiden lisäksi se piilottaa eri selaimien erikoisuudet, joten ei tarvitse keskittyä siihen, että toimiiko koodi eri selaimissa. (Gustafson 2013, 40.)

JavaScriptissä voi valita yhden tai useamman elementin HTML:n DOM-puusta. Elementtejä voi etsiä id-, luokan- (eng. Class), nimi- (eng. Name) tai tag arvon mukaan.

```
1  var elementtiJS = document.getElementsByTagName("div");
2
3  var elementtiJQ = $("div");
```

Kuva 4. Kuvakaappaus div elementin valinnasta (Visual Studio Code 2020)

Esimerkiksi jos haluaa valita kaikki div-elementit HTML:n DOM-puusta. Kuvassa 4 rivillä yksi näkyy JavaScriptin ohjelmointilause kyseiseen valintaan, kun taas sama ohjelmointilause jQueryssä on kuvan (kuva 4) rivillä kolme. Gustafsonin (2013, 40) mielestä jQuery tekee JavaScriptin ohjelmien kirjoittamisesta kestettävämpää ja jopa hauskaa.

2.2 Tehtävätaso

Tehtävätaso on kolmikerrosmallin keskimmäinen taso. Taso sisältää loogiset laskennat ja operaattorit. Taso myös toimii linkkinä esittely- ja tietotason välillä. Tason tärkeimpiä tehtäviä on kutsua toimintoja, lukea lomakkeita ja määrittää, onko niiden arvot oikein ennen kuin ne lähetetään eteenpäin. (Rishabh 2020.)

Tässä toimeksiannossa käytämme PHP ohjelmointikieltä tehtävä tasossa. Vuonna 1994 Rasmus Lerdorf kehitti ensimmäisen niin kutsutun PHP ohjelman C kielellä. Hän käytti sitä siihen, että pystyi seuramaan kävijämääriä hänen ansioluettelollaan verkossa. Lerdorf nimitti kyseisen ohjelman ”Personal Home Page Tools”, johon nykyään viitataan PHP Tools. Ajan myötä kaivattiin enemmän toiminallisuuksia ohjelmaan, joten Lendorf kirjoitti PHP Toolsin uudelleen. Tässä uudessa versiossa pystyttiin ottamaan yhteyttä tietokantaan ja tarjottiin mahdollisuus frameworkeihin jotta voi kehittää dynaamisen sovelluksen. Kesäkuussa 1995 hän julkaisi lähdekoodit PHP Toolsiin ja antoi luvan kehittäjille käyttää sitä mielensä mukaan. Tätä työkalua kehitettiin ja vuonna 1996 siihen lisättiin sisäänrakennettu tuki tietokannoille DBM, mSQL ja Postgres95. Siinä oli myös tuki evästeille (eng. Cookies. Selaimen välimuistissa tallennetut tiedot) ja käyttäjän määrittämille toiminnoille. Kun nämä tuet julkaistiin, niin PHP:tä alettiin käsittämään ohjelmointikieleksi.

PHP:n rakenne muistuttaa hyvin paljon C kieltä, mutta sen muuttujat muistuttavat enemmän Perl ohjelmointikieltä. PHP:ssä voi upottaa PHP koodia HTML

tiedostoon. Nykyisestä PHP:stä (Hypertext Preprocessor) on kolme versiota PHP 3, PHP 4 ja PHP 5. PHP 3 julkaistiin 1997 ja tärkeimpänä lisäyksenä siihen tuli API:t. PHP 4 julkaistiin 2000 ja siinä tärkeimpänä lisäyksenä tuli HTTP istunnot (eng. Session). PHP 5 julkaistiin 2004 ja siinä tuli tuki projekteille kuten PEAR (PHP Extension and Application Repository) ja PECL (PHP Extension Community Library). (The PHP Group s.a.)

PHP:n voi upottaa eri tiedostoihin ja tämän takia PHP koodin kyseisissä tiedostoissa on sijoitettu tagin sisälle joka merkitään: *<?php … ?>*. Kuten HTML:ssä siinä on avaava- ja sulkeva tagi.

```
1  <?php
2      $muuttuja= "maailma";
3
4      echo "Tervehdys $muuttuja! Tässä on ensimmäinen PHP ohjelmani";
5  ?>
```

Kuva 5. Kuvakaappaus PHP ohjelmasta (Visual Studio Code 2020)

Kuvassa 5 on esitelty PHP ohjelma kaikessa yksinkertaisuudessaan. Kuten aikaisemmin mainittu ohjelma aloitetaan avautuvalla tagilla jossa ilmoitetaan, että tämä on PHP kieltä. Rivillä kaksi esitellään PHP muuttuja (kuva 5). Muuttuja merkitään $-merkillä. Toisin kuten JavaScriptissä, jossa var voi jättää pois sitten kun muuttuja on esitelty, PHP:ssä pitää aina muistaa käyttää $-merkkiä käytäessä muuttujaa. $-merkki määrittää muuttujan (Carr & Gray 2018, 4). Rivillä kaksi oleva muuttuja on merkkijono, jonka voi merkitä ” tai ’ merkkien sisälle. Rivillä neljä tulostetaan muuttujan sisältämä merkkijono näytölle komennolla echo. (Kuva 5.) Kuten JavaScriptissä PHP:ssä ei alusteta tietotyyppejä. Kokonaisluvut esitellään muuttujassa *$muuttuja = 29;* ja liukuluvut *$muuttuja = 29.94;*. Totuusarvot ovat myös käytössä PHP kielessä.

2.3 Tietotaso

Tietotaso on mallin alin taso. Taso pitää huolen, että kaikki tietoon liittyvät toiminnat suoritettaan. (Managementmania 2015.) Nämä toiminnot voivat olla tietokantaan tehtäviä lisää (eng. Insert), poista (eng. Delete) ja päivitä (eng. Update) komentoja. (Rishabh 2020.) Aikoinaan tiedon saantiin kirjoitettiin

omat erilliset ohjelmat ja jos haluttiin muuta tietoa, jouduttiin pyytämään ohjelmoijaa kirjoittamaan erillisen ohjelman sitä varten. Koska työn määrä yhden ohjelman tekemiseen tiedon saantia varten oli suuri ja käyttäjät kaipasivat helposti saatavaa tietoa, alettiin kehittämään kieltä, jolla käyttäjä voisi esittää kyselyitä. (Microsoft 2017.) IBM kehitti kyselykielen SQL:n (Structured Query Language) vuonna 1970. Erilaisten tietokantojen takia SQL:ään on muodostunut murteita, kuten MySQL ja Oracle. SQL koostuu DDL (Data Definition Language) ja DML:stä (Data Manipulation Language). (Darmawikarta 2014, 1.) Tässä toimeksiannossa tulemme käyttämään MySQL tietokantaa.

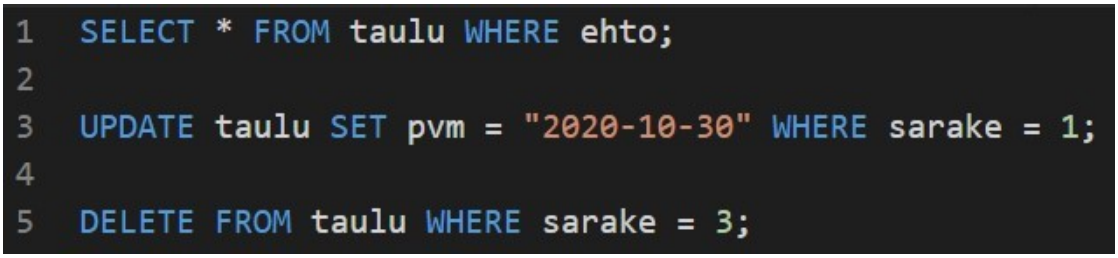
MySQL on vapaan lähdekoodin tietokanta, jonka Monty Widenius on kehittänyt (Converse ym. 2004, 4–5). MySQL on relaatio tietokanta jonne tietoa voi tallentaa tauluihin. Taulujen tieto lisätään riveittäin ja jokaisella erilliselle tiedolle rivissä on oma nimetty kenttänsä, jota kutsutaan sarakkeeksi. (Darmawikarta 2014, 5). Tietokannan toiminnallisuus toimii siten että käyttäjä lähettää pyynnön tietokannalle. Tätä kyseistä pyyntöä kutsutaan kyselyksi. (Microsoft 2017.) Kyselyiden lisäksi tietokantaan voidaan tallentaa ja ylläpitää tietoa, tallentaa menetelmiä ja sääntöjä tiedonhallintaa varten.

Converse ym. (2004, 247) esittävät relaation tietokannan perusajatuksen kysymyksellä: ”Miksi en vain voi kirjoittaa osoitettani kerran ja he voivat tarkistaa sen jostain tarpeen tullen?”.

```sql
CREATE DATABASE tietokanta;

CREATE TABLE taulu
    (
        sarake INT(10),
        nimi VARCHAR(50),
        pvm DATE
    );

INSERT INTO taulu (sarake, nimi, pvm)
VALUES (1, "Rivi 1", "2020-10-29");

INSERT INTO taulu (sarake, nimi, pvm)
VALUES (2, "Rivi 2", "2020-01-13"), (3, "Rivi 3", "2020-12-13");
```

Kuva 6. Kuvakaappaus SQL tietokannan luonnista (Visual Studio Code 2020)

Kuvassa 6 esitellään, kuinka SQL kielellä luodaan uusi tietokanta, lisätään tietokantaan taulu, jossa on kenttiä ja kuinka tauluun lisätään tietoa. Kuten ohjelmointikielissä, SQL:ssä on myös valmiiksi määriteltyjä tietotyyppejä. Kuten kuvassa 6, rivillä viisi, sarake niminen kenttä määriteetään tietotyypiksi integer, eli kokonaisluku. Darmawikarta (2014, 127) kirjoittaa että integer tallentaa kokonaisluvut -2,147,483,648 – 2,147,483,647 välillä. VARCHAR on tietotyyppi merkkijonolle, jossa sulkujen sisälle voi määrittää jonolle maksimi pituuden. DATE on tietotyyppi pätevälle päiväykselle 1000-01-01 – 9999-12-31 välillä. Päiväys MySQL:ssä on merkitty vuosi-kuukausi-päivä muodossa. CREATE ja INSERT käskyjen lisäksi SQL:ssa on käytettävissä SELECT, UPDATE ja DELETE.



Kuva 7. Kuvakaappaus SQL komennoista (Visual Studio Code 2020)

SELECT on komento, jolla saa haettua tietoa tietokannalta (Converse ym. 2004, 247). Kyselyssä pitää esittää mitkä sarakkeet halutaan, mistä taulusta (FROM) ja onko kyselylle joitain ehtoja (WHERE). Katso kuva 7, rivi yksi.

UPDATE on komento, jolla voi muuttaa tietoa tietokannassa ilman että poistaa riviä (Converse ym. 2004, 251). Kuten edellä mainitussa komennossa, tässäkin pitää määrittää mihin tehdään muutoksia. Asetetaan muutokset, jotka halutaan tehdä (SET) ja annetaan tarvittaessa ehdot (WHERE). Katso kuva 7, rivi kolme.

DELETE on komento, jolla poistetaan pysyvästi jokin rivi tai tieto taulusta (Converse ym. 2004, 252). Katso kuva 7, rivi viisi. Poisto komentoja on kaksi. Toinen niistä on DROP, jolla voi poistaa kokonaisen taulun tai jopa itse tietokannan (Converse ym. 2004, 254).

Yleensä sovelluksissa, johon kuuluu tietokanta, käytetään CRUD-toimintoja. CRUD-toiminoilla viitataan tietokannan käsittely toimintoihin. Nämä toiminnot ovat:

- Create. Luo uuden osion tietokantaan. Kuten tietokannan, taulun, taulun sisällön tai säännön.
- Read. Lukee tietoja tietokannasta.
- Update. Päivittää tai muokkaa jotain tietokannan osiota.
- Delete. Poistaa jonkun tietokannan osion.

(Carr & Gray 2018, 66.)

3 TOIMEKSIANTO JA SEN SUUNNITTELU SEKÄ TOTEUTUS

Toimeksiantaja työssä on InStore Media Oy, joka on myymälämainontaan erikoistunut mikkeliläinen yritys. He toteuttavat mainoskampanjoita sopimusmyymälöissään päivittäistavarakauppojen tavarantoimittajien toimeksiannosta. He toimittavat mainoksia ympäri Suomea.

Toimeksiantona on rakentaa selainpohjainen sovellus, jolla voi luoda myymälä kohtaisia tilityslaskelmia tietyllä ajanjaksolla toteutuneista mainoskampanjoista. Laskelman tueksi sovellukseen tarvitaan tietokanta, joka sisältää ainakin kolme taulua, jota voi hallinnoida CRUD toiminnolla.
Toimeksiantajalle tehty sovellus sisältää arkaluontoista tietoa. Tämän takia työn kuvissa ja muussa oikeata tietoa ei esitellä. Kaikki näkyvä tieto ovat sattuman varaisesti luotua tietoa.

Seuraavissa luvuissa käydään läpi sovelluksen käyttötarve, tietokannan taulut, niiden sisältö, kuinka ne ovat yhteydessä toisiinsa, sekä käsitellään sovelluksen teon eri vaiheet.

3.1 Käyttötarve

Myymälöiden kauppiaat saavat tietyn sovitun korvauksen jokaisesta toteutuneesta kampanjasta. Tilitys koostuu korvauksen saavan myymälän tiedoista, kyseisen myymälän kauppiaan yhteistiedoista, vapaavalinnaisesta ajanjaksosta kuukausi tasolla, jonka perusteella tuodaan tilitys laskelmaan yksi tai useampi kampanja, sekä korvauksen maksupäivästä ja korvauksen summasta verollisena ja verottomana. On yksi poikkeus, jolloin tilitys hieman muuttuu. Tietyn kauppaketjun tai ryhmän mediatilityslaskelmissa ei tuoda

kauppiaan tietoja laskuun. Kyseisissä kauppaketjuissa korvaukset menevät osuuskunnalle, eikä kauppiaalle. Tässä tapauksessa tilitys koostuu osuuskunnan yhteistiedoista, vapaavalinnaisesta ajanjaksosta kuukausi tasolla, jonka perusteella tuodaan tilitys laskelmaan yksi tai useampi kampanja, listaus kaikista osuuskuntaan kuuluvista kaupoista, jossa kampanja toteutui, korvauksen maksupäivä ja korvauksen suurus per tietty kauppatyyppi. Katso taulukko 1. Sekä verollinen ja veroton summa korvauksesta.

Taulukko 1. Osuuskunnan tilitys kampanjasta

Kampanja	Kampanjan kaupat	Korvaus per kauppatyyppi	Yhteensä
Elintarvike kampanja 13.1.-23.3.2020	3 x market	15 €	45 €
	4 x supermarket	23 €	92 €
	1 x hypermarket	40 €	40 €

Laskelman valmistuessa siitä muodostetaan PDF-tiedosto, joka tallennetaan kirjanpitoa varten ja lähetetään kauppiaalle tai osuuskunnalle.

3.2 Toiminnallisuudet

Toimeksiantaja on esittänyt toimeksiannon aikana ja työn edetessä vaatimuksia minkälaisia toiminnallisuuksia sovelluksessa pitäisi olla. Toiminnallisuudet ovat jaettu tärkeyden mukaan. Sovelluksen tärkeimmät toiminnallisuudet kuuluvat mediatilitykseen. Sen jälkeen tulee toiminnallisuudet, jotka kuuluvat tietokannan hallintaan, uuden sisällön luomiseen ja sen näyttämiseen. Viimeisenä ovat pienemmät toiminnallisuudet, jotka eivät vaikuta suuresti sovelluksen toiminnallisuuteen. Nämä toiminnot toteutetaan seuraavassa versiossa. Mediatilityslaskelman toiminnallisuudet:

Automaatio

- Päiväys, jolloin tilitys on luotu, näkyy laskelmassa.
- Kun kauppa on valittu, ohjelma tuo automaattisesti tilitykseen tarvittavat tiedot. Esimerkiksi kaupan ja kauppiaan yhteystiedot.
- Kaupassa toteutuneet kampanjat tulevat automaattisesti laskelmaan, jos kampanjan aloituspäivä osuu ajanjaksolle, joka valitaan.

Muokattavuus

- Laskuun tulevia kampanjoita voi valita listasta. Listasta valitut kampanjat näkyvät laskelmassa.
- Tilityksen korvaus on oletuksella kauppias kohtainen korvaus. Korvauksen voi valita onko se kauppias kohtainen, kampanja kohtainen vai jokin muu summa, jonka voi kirjottaa sovellukseen.

Laskenta

- Ohjelma laskee kampanja kohtaisen verollisen osuuden arvolisäveroprosentin mukaan.
- Ohjelma laskee kampanja kohtaisen yhteissumman, johon on lisätty arvolisä.
- Ohjelma laskee kaikkien kampanjoiden verottoman yhteissumman.
- Ohjelma laskee kaikkien kampanjoiden verollisen osuuden.
- Ohjelma laskee kaikkien kampanjoiden yhteissumman, johon on lisätty verollinen osuus.

Poikkeus

- Muokkaa laskelman ulkonäköä.
- Hakee osuuskunnan tiedot laskutusta varten.
- Hakee kaikki kaupat, jotka kuuluvat osuuskuntaan, jossa valitut kampanja ovat toteutuneet valitulla ajanjaksolla.
- Lajittelee kaupat kauppatyypittäin.
- Laskee kuinka monta kauppaa jokaisessa kauppatyypissä on.
- Tuo kauppatyypin korvauksen ja kertoo sen kauppojen määrällä.

Tulostus

- Ohjelma luo PDF-laskun, joka sisältää kaikki valitut tiedot.

Mediatilityksen lisäksi sovellukseen tulee kolme muuta näkymää, jossa on toiminnallisuuksia. Nämä näkymät ovat: listaus kaikista kaupoista, uuden kaupan luonti ja uuden kampanjan luonti.

Kauppojen listaus näkymässä oletuksena haetaan postinumeron mukaan kymmenen ensimmäistä kauppaa ja esitetään ne listattuna. Näkymässä voi suodattaa kauppoja jakelualueen mukaan, jolloin ryhmitetään kaupat jakelualueisiin. Ensin esitettään mikä jakelualue on kyseessä ja sen alle on listattu kaikki jakelualueella olevat kaupat. Tässä suodatuksessa näytetään kaikki jakelualueet, jotka ovat järjestetty jakelukoodin mukaan. Näkymässä vois myös suodattaa kauppoja kauppaketjun ja jakelualueen mukaan. Näkymän muita toimintoja ovat kaupan poisto ja kaupan tarkempien tietojen katselu. Listauksessa näytetään kauppaketjun nimi, kaupan nimi, kaupan osoite ja kauppiaan nimi. Kun avaa kaupan tarkemmat tiedot siellä näkyvät edellä mainituiden tietojen lisäksi jakelualue johon kauppa kuulu, kauppiaan yhteystiedot, jakelualueen jakelijan nimi ja yhteystiedot, sekä kauppaan kiinnitetyt kampanjat.

Uuden kaupan luontinäkymässä oletuksena tuodaan lomakkeeseen kaikki tietokannassa olevat kauppaketjut, kauppiaat, jakelualueet ja kampanjat.

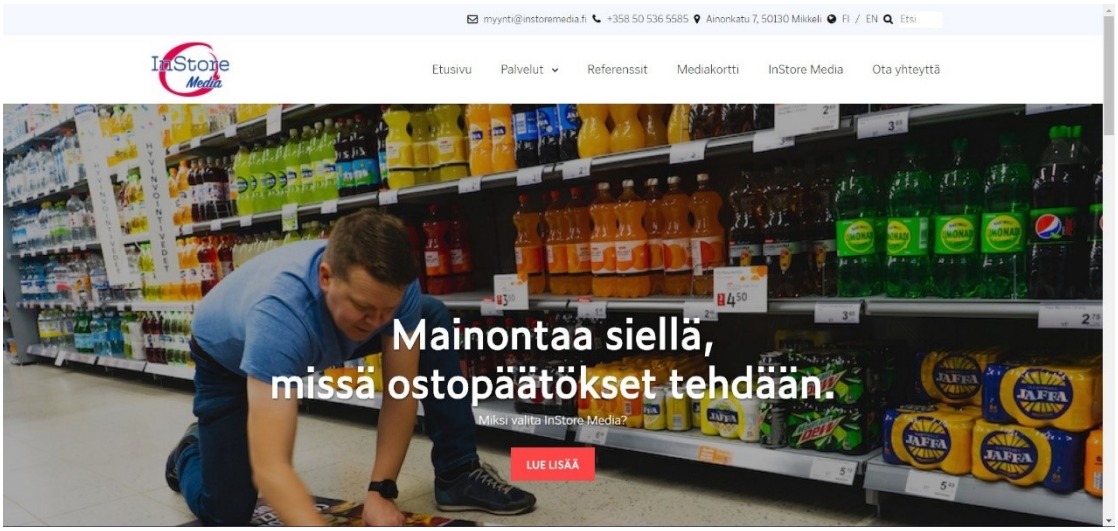
Uuden kampanjan luontinäkymässä oletuksena tuodaan lomakkeeseen kaikki tietokannassa olevat mediatoimistot ja asiakkaat. Näiden lisäksi näkymässä voi liittää uuteen kampanjaan olemassa olevia kauppoja. Kyseistä toimintoa varten näkymässä on alasvetovalikko, josta voi liittää kauppoja kampanjaan kauppaketjun tai osuuskunnan mukaan. Valintojen mukaan ohjelma hakee listaukseen kyseisen määritelmän täyttäneet kaupat, jotka ovat automaattisesti valittu. Valintoja voi poistaa, lisätä ja sitten tallentaa valinnat kampanjaan. Tallennuksen jälkeen tulee näkymään ruutu, jossa on yhteenveto siitä, kuinka monta kauppaa tullaan kiinnittämään kampanjaan. Yhteenvedossa lasketaan myös, että monessako kaupassa on banaanin malliset ostoskorit. Kaupat, joissa on näitä niin kutsuttuja banaanikoreja, on merkitty listauksessa banaania muistuttavalla kuvakkeella.

Sovelluksessa näiden edellä mainittujen toimintojen lisäksi pitää löytyä toiminnot, jotka vaihtavat näkymää tai palaavat takaisin tiettyyn näkymään.

3.3 Esitystaso

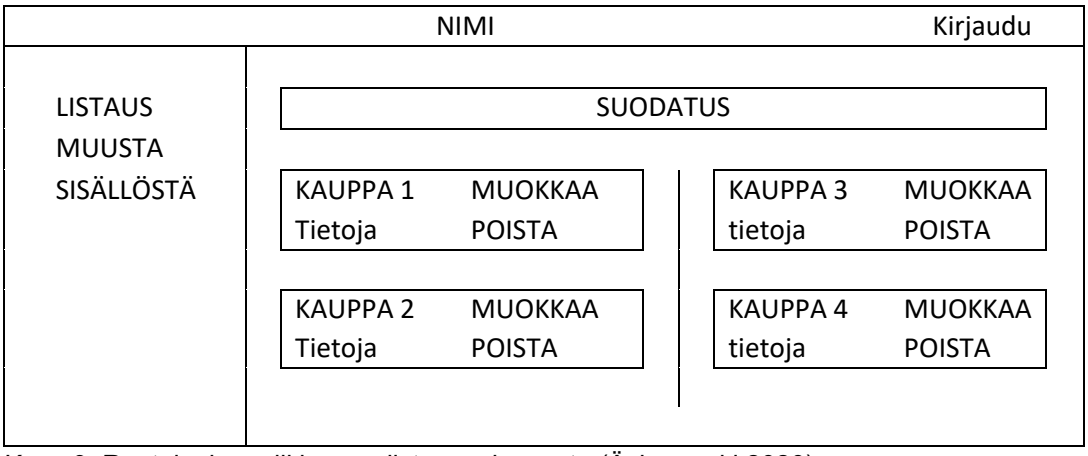
3.3.1 Ulkonäön suunnittelu

Toimeksiantajan toive oli, että sovelluksen ulkonäkö olisi moderni ja helppokäyttöinen. Värimaailmaksi toivottiin yrityksen värejä. Yrityksellä ei ollut graafisia ohjeita brändäystä varten, joten viitteitä haluttuun brändiin otettiin yrityksen markkinointi verkkosivuilta, katso kuva 8.



Kuva 8. Kuvakaappaus yrityksen verkkosivusta (InStore Media Oy. s.a.)

Ulkonäön suunnittelu alkoi sillä, että tutkittiin yrityksen verkkosivuja sekä olemassa olevaa Wordpress sivustoa, jossa ylläpidettiin tietoja. Kummastakin sivusta poimittiin ne asiat mistä toimeksiantaja piti ja mietittiin ulkonäköä niiden ja kappaleessa 3.2 esitettyjen toiminallisuuksien kautta.



Kuva 9. Rautalankamalli kauppalistausnäkymästä (Äyhynmäki 2020)

Kuvan 9 tyylin mukaisesti jokaisesta näkymästä, joka oli suunniteltu sovellukseen, tehtiin oma rautalankamalli. Rautalankamalleja hyödynnettiin sovelluksen prototyypin teon aikana. Toiminnallinen prototyyppi tehtiin, jotta toimeksiantaja saisi konkreettisen kuvan sovelluksesta ja että sovelluksesta voitaisiin helposti poistaa tai lisätä elementtejä toivomuksen mukaan.

Kuva 10. Kuvakaappaus kauppalistausnäkymästä prototyypissä (Äyhynmäki 2020)

Prototyypin ohjelmoinnin yhteydessä suunniteltiin sovelluksen värimaailmaa. Värimaailman kolmeksi pääväriksi valittiin yrityksen logosta löytyvät punainen,

tummansininen ja valkoinen. Värit sijoiteltiin eri elementteihin ja testattiin niiden yhteensopivuutta. Kuvassa 10 näkyy, kuinka edellä mainituista väreistä on luotu sovelluksen värimaailma.

Sovelluksen ulkonäön luomisessa, elementtien sijoittamisessa ja responsiivisen käyttöliittymän luomiseen hyödynnettiin Bootstrap frameworkia. Bootstrap liitetään koodiin tyylitiedostolinkkinä kuten CSS:än.

```
.suodatus {
    margin-top:2rem;
}

.suodatus .input-group-text {
    background-color: #284a8c;
    color: #FFF;
}

.suodatus .btn{
    background-color: #284a8c;
}

.suodatus .btn:hover{
    color: #f4484b;
}

.rekisteri{
    margin-top:2rem;
}

.listausV{
    float:left;
    width:49%;
}

.listausO{
    float:right;
    width:49%;
}

.laatikko{
    padding:1rem 1rem 0.1px 1rem;
    box-shadow:1px 1px 10px #284a8c;
    margin-bottom:1rem;
}
```

Kuva 11. Kuvakaappaus osasta prototyypin tyylitiedostosta (Visual Studio Code 2020)

Kuvassa 10 käytetty tyylitiedosto on nähtävillä kuvassa 11. Tyylitiedostossa määritellään elementtien värit, sijoitus sekä etäisyys reunaan. Suomenkieliset luokat ovat itse tehtyjä tyyliluokkia, jolla halutaan määrittää tietyt arvot, kun taas kuvassa 11 näkyvät luokat rivillä 171, 176 ja 180 ovat Bootstrapin valmiiksi määritettyjä luokkia, joilla on jo tiettyjä valmiiksi määriteltyjä arvoja, mutta tässä tapauksessa niille halutaan määrittää tietty väri. Bootsrapissä on käytössä monia valmiiksi määriteltyjä tyyliluokkia, joita voi hyödyntää suoraan HTML:ssä esimerkiksi mt-1. Mt-1 tarkoittaa että elementille määritetään sen yläpuolelle marginaalia yhden rem:in verran. Rem (root em) on yksikkö, jota käytettään CSS:ssä ilmaisemaan kokoa. Yksi rem vastaa 16 pikseliä. Koska Bootsrapin luokat käyttävät rem yksikköä niin muissa tyylitiedoston luokissa on vain helpompaa käyttää samaista yksikköä.

Prototyypin näkymien ja ulkonäön valmistuessa se esitettiin toimeksiantajalle ja hänen palautteidensa kautta aloitettiin rakentamaan sovellusta.

3.3.2 Käyttöliittymän rakentaminen

Sovelluksen rakennus on hyvin vaivatonta, jos prototyyppi on tehty. Prototyyppi on tyypillisesti toteutettu pelkästään HTML koodilla. Tämän takia prototyyppiä on helppo hyödyntää siten, että ohjelmoidaan isompia toiminnallisuuksia. Ensimmäiseksi toiminnallisuuksia lisättiin etusivuun ja näkymiin, joissa voi lisätä uuden kaupan tai kampanjan, koska näissä oli vähiten JavaScriptillä toteutettavia toiminnallisuuksia. Näiden näkymien toiminallisuuksien valmistumisen jälkeen alettiin lisäämään mediatilitysnäkymään toiminallisuuksia. Prototyypissä esikatselu laskusta oli pelkkä kuva, joten näkymän toiminnallisuuksien lisäys alkoi sillä, että kuvan kohtaan kirjoitettiin HTML:llä tulostettavaa laskua vastaava näkymä. Laskulle tehtiin oma osio, kuten kuvassa 12 näkyy. Tämä laatikko jaoteltiin kolmeen osaan: ylä, sisältö ja ala. Yläosiossa on yrityksen nimi, tieto siitä mikä tämä on ja päiväys, jolloin dokumentti on luotu. Sisältö osiossa löytyy pienempiä osioita tai lohkoja, kuten lohkot, jossa esitetään kaupan- ja kauppiaan tiedot. Sisältö osuus, jossa esitetään laskun sisältöä, eli mistä kampanjoista maksetaan, on toteutettu taulukkorakenteella. Rakenteen valmistumisen jälkeen laskulle määritettiin tyylit ja näkymään alettiin ohjelmoimaan toimintoja. Näkymän tärkeimmät toiminnot ovat ne, jotka tuovat tiettyihin kenttiin syötetyn sisällön avulla laskun esikatseluun ilman että käyttäjä tekee mitään.

```
<div id='lasku'>
    <div id='yla'>
        <img src='../img/ism-logo.png' alt='logo'/>
        <p class='float-left'>MEDIATILITYSLASKELMA</p>
        <p class='text-right' id='paiva'>PP.KK.VVVV</p>
    </div>
    <div id='sisalto'>
    <!---->
        <div id='muu'>
            <div id='kauppaInf'>
                <p>Kaupan nimi</p>
                <p>Osoite</p>
                <p>Postinumero</p>
                <p>Paikkakunta</p>
            </div>

            <div class='mt-4' id='kauppiasInf'>
                <p>Kauppiaan nimi</p>
                <p>Sähköposti</p>
                <p>Puhelinnumero</p>
                <p>Tilinumero</p>
            </div>

            <div class='mt-5'>
                <p class='korostus iso border-bottom border-dark pl-1 mb-2'>Yhteenveto maksettavista mediatilityksistä</p>
                <ul>
                    <li>Maksut perustuvat toteutuneisiin myymälämainoskampanjoihin</li>
                    <li id='jakso'>Maksut ajanjaksolla KK-KK/VVVV alkaneista kampanjoista</li>
                    <li id='maksuP'>Maksupäivä PP.KK.VVVV</li>
                </ul>
            </div>

            <table id='taulu' class='mt-4'>
                <thead>
                    <tr class='korostus'>
                        <th>Kampanja</th>
                        <th>Kampanjajakso</th>
                        <th>Mediaeurot</th>
                        <th id='oAlvTH'>ALV 24%</th>
                        <th>Tilitys €</th>
                    </tr>
                </thead>
                <tbody id='tauluB'>
                </tbody>
            </table>

            <table id='yhteensa'>
                <tbody>
```

Kuva 12. Kuvakaappaus laskun HTML:stä (Visual Studio Code 2020)

Yksi esimerkki näistä toiminnoista on laskutuksen ajanjakso. Näkymässä on kenttä, jonne voi valita kuukausitasolla laskutuksen ajanjakson alkamis- ja päättymiskuukauden. Kuvassa 13 näkyvä JavaScript ohjelma, kuuntelee alkamiskuukasi kenttää. Kun kentän sisältö vaihtuu, ohjelma tarkistaa, että kenttä ei ole tyhjä ja sen jälkeen ohjelma pilkkoo osiin saamansa arvon ja muuntaa sen osista taulukon (eng. Array). Jos ajankohtaan on myös määritetty loppumiskuukausi, niin se pilkkoo sen myös ja korvaa oletuksella tyhjänä olevan muuttujan.

Kaikki muuttujat kerätään yhteen taulukkoon ja kutsutaan ohjelmaa, joka luo laskuun tekstin, jossa ilmoitetaan laskun ajanjakso hyödyntäen sille lähetetty taulukon tietoja. Kyseinen ohjelma lisää luodun tekstin laskuun sille ositetulle paikalle. Edellä mainittujen toimintojen valmistumisen jälkeen ohjelmoitiin toiminnot, joissa voi valita mitä laskupohjaa käytetään.

Näkymään siirtyessä oletuksena on valittu yksittäisen kaupan laskupohja, mutta jos haluaa käyttää osuuskunnalle suunnattua laskupohjaa, niin näkymässä voi sen valita ja ohjelma piilottaa tarpeettomat kentät ja tuo esille kyseiseen laskelmaan oleelliset kentät, sekä muokkaa laskun esikatselua osuuskunnan laskutuspohjan mukaisesti.

```
$("#alku").change(function(){
    let res1 = $("#alku").val();
    if(res1){
        let res2 = $("#loppu").val();
        let aKoko = res1.split("-");
        let lKoko = null;
        let id = null;

        if(res2){
            lKoko = res2.split("-");
        }

        let tiedot = {
            "laskupohja":laskupohja,
            "alku":aKoko,
            "loppu":lKoko
        };

        luoTeksti(tiedot);
```

Kuva 13. Kuvakaappaus toiminnosta, jolla alku päivämäärä viedään laskuun (Visual Studio Code 2020)

Osuuskunnan laskupohjassa ei näytetä kaupan tai kauppiaan tietoja. Tässä tuodaan osuuskunnan tiedot ennen sisältöosiota, jossa esitetään mistä kaikesta maksetaan.

3.4 Tehtävätaso

Sovellus tulee sijaitsemaan web-hotellin tarjoamalla palvelimella. Web-hotellin tilaus kattaa yhden MySQL tietokannan, verkkotunnuksen, viiden gigatavun edestä levytilaa palvelimella sekä SSL-salauksen. Kaikki sovellukseen kuuluvat koodit lähetetään palvelimelle FTP-ohjelman kautta. Web-hotellin palvelin tukee PHP-ohjelmointikieltä, joten sen takia toimeksiannossa tietokantayhteys, käyttäjän näkymän ja palvelimen väliset toiminnot sekä yhteydet toteutetaan kyseisellä ohjelmointikielellä.

Sovelluksen arkaluontoisen sisällön takia on hyvin tärkeää, että kuka tahansa ei pääse näkemään sisältöä, joten sovelluksessa on kirjautuminen sekä istunto. Jokaisessa sovelluksen näkymässä tai sivussa aloitettaan istunto, joka luo ainutlaatuisen istuntoavaimen, jos sitä ei ole jo olemassa. Kuvassa 14 näkyvässä ehtolauseessa tutkitaan, onko istuntotunnusta määritetty istunnolle. Jos ehto täyttyy, niin sivu näytetään, muuten näytetään kirjautumissivu. Ilman tunnusta pääse ainoastaan kirjautumissivulle ja jos pyrkii jollekin muulle sovelluksen näkymistä ilman tunnusta, koodi ohjaa automaattisesti kirjautumiseen. Tunnus määritetään onnistuneen kirjautumisen yhteydessä.

```
if(isset($_SESSION["tunnus"])){
    echo "<a class='navbar-text' href='#'>"
        .$_SESSION["tunnus"].
        "<form method='GET' action='script/php/signOut.php'>
            <input type='submit' name='ulos' value='Kirjaudu ulos'/>
        </form>
    </a>
    </nav>
    <div class='container mt-3'>
        <div class='row text-center'>
            <div class='col'>
                X kampanjaa käynnissä
            </div>
            <div id='kaupat' class='col'>
            </div>
            <div id ='ketjut' class='col'>
            </div>
        </div>
    </div>
    <div class='container'>
        <div class='row kortti'>
            <div class='col'>
                <h4><a class='h' href='record/shops.php'>Kaupat</a></h4>
                <div class='ks'>
                    <a class='p' href='new/shop.php'><p class='m-0'>Uusi</p></a>
                    <p>Lista kauppiaista</p>
                </div>
            </div>
            <div class='col mx-3'>
                <h4>Kampanjat</h4>
                <div class='ks'>
                    <a class='p' href='new/campaign.php'>Uusi</a>
                </div>
            </div>
            <div class='col mr-3'>
                <h4>Mediatoimistot</h4>
                <div class='ks'>
                    <p>Uusi</p>
                </div>
```

Kuva 14. Kuvakaappaus etusivun näyttämisen ehtolauseesta (Visual Studio Code 2020)

Web-hotellin tarjoama verkkotunnus ei ole yksityinen. Tämä tarkoittaa sitä, että kuka tahansa, joka tietää sivun osoitteen pystyy luomaan tunnukset ja käyttämään niitä sisäänkirjautumiseen. Varmistaakseen sen, että näin ei pääse käymään, kirjautumisessa tarkistetaan salasanan ja käyttäjätunnuksen lisäksi, että käyttäjätunnus on vahvistettu. Rekisteröityessä käyttäjätunnus luodaan, mutta sitä ei ole vahvistettu. Jos kirjautuminen palauttaa, että tunnus ei ole vahvistettu, tässä tapauksessa 0, kirjautuminen ei onnistu, vaikka käyttäjätunnus ja salasana olisi oikein. (Kuva 15.)

```php
if(isset($_REQUEST["kirjaudu"])){
    include("script/php/connection.php");

    $tunnus = $_REQUEST["kTunnus"];
    $salasana = crypt($_REQUEST["kSala"], "rl");

    $kysely = $yhteys->prepare("SELECT tunnus, vahvistus FROM tunnukset WHERE tunnus = :tunnus AND
    salasana = :salasana");

    $kysely->bindParam(":tunnus", $tunnus);
    $kysely->bindParam(":salasana", $salasana);

    $kysely->execute();
    $tulos = $kysely->fetch();

    if($tulos[0] && $tulos[1] != 0){
        $_SESSION["tunnus"] = $tulos[0];
        header("Location: http://localhost/oppari/demo/index.php");
        die();
    }
    else if($tulos[0] && $tulos[1] == 0){
        echo "<div id='sHuom' class='alert alert-dismissible alert-danger' role='alert'>
            Kirjautuminen ei onnistunut. Käyttätunnustasi ei ole vielä vahvistettu. Pyydä
            vahvistusta ja yritä uudelleen<button type='button' class='close' data-dismiss='alert'
            aria-label='Close'><span aria-hidden='true'>&times;</span></button>
        </div>";
    }
    else{
        echo "<div id='sHuom' class='alert alert-dismissible alert-danger' role='alert'>
            Kirjautuminen ei onnistunut. Käyttätunnus tai salasana on väärin. Yritä
            uudelleen<button type='button' class='close' data-dismiss='alert'
            aria-label='Close'><span aria-hidden='true'>&times;</span></button>
```

Kuva 15. Kuvakaappaus ohjelmasta, jossa tarkistetaan, että käyttäjätunnus on vahvistettu (Visual Studio Code 2020)

Tärkein osa sovelluksen toimivuudessa on yhteys palvelimen ja tietokannan välillä. Kaikki sovelluksessa oleva tiedot haetaan eri vaiheissa tietokannasta erilaisilla kyselyillä tai tietokantaan tallennetaan jotain uutta, jota halutaan ohjelmaan näkyville. Ilman yhteyttä sovellus olisi kokonaan tyhjä.

```php
$yhteys = "mysql:host=localhost;dbname=oppari";
$kayttajatunnus = "root";
$salasana = "";

try{
    $yhteys = new PDO($yhteys, $kayttajatunnus, $salasana);
}
catch (PDOException $e){
    die("Tietokantaan ei saada yhteyttä. Virhe: " . $e);
}

$yhteys->exec("SET NAMES utf8");
```

Kuva 16. Kuvakaappaus yhteydestä tietokantaan (Visual Studio Code 2020)

Kuvassa 16 luodaan yhteys tietokantaa käyttäen PDO (PHP Data Objects) laajennusta. Koodissa alustetaan tarvittavat tiedot muuttujiin yhteyden luomista varten. Kuvassa olevat muuttujat luovat tietokoneessa paikallisesti olevaan tietokantaan yhteyden ja sitä käytettiin työssä ainoastaan ohjelman testaukseen, ennen kuin koodit lähetettiin verkkoon. Tiedot lähetetään laajennukselle, joka pyrkii luomaan yhteyden annettuun tietokantaan. Onnistuessa yhteysmuuttuja on käytettävissä koodissa ja sitä käsketään käyttämään UTF-8 merkistöä tietokannassa. Tämä merkistö sisältää muun muassa suomen kielessä käytettävät kirjaimet *å*, *ä* ja *ö*. Jos yhteyden luominen epäonnistuu, koodi nappaa virhekoodin muuttujaan ja tulostaa syyn näytölle. Nyt kun yhteys on kunnossa, voidaan tuoda tietokannassa olevaa tietoa näkymiin ja ohjelmoida toiminnallisuudet, jolla tallennetaan uusi kampanja ja kauppa.

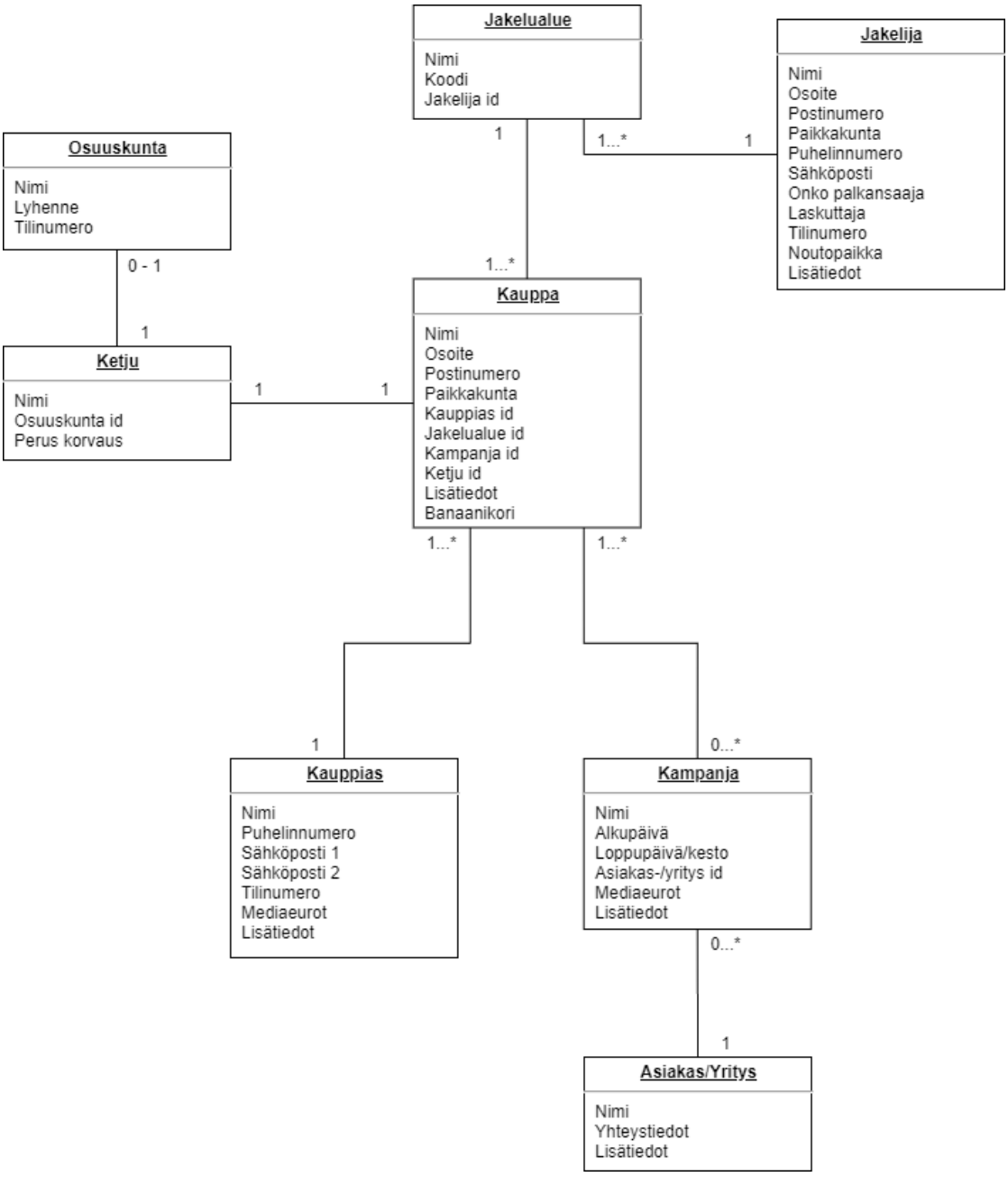
```php
include("connection.php");

$sql = "";

if(isset($_REQUEST["alku"])){
    $loppu = "";

    if(isset($_REQUEST["loppu"]) && $_REQUEST["loppu"] != null){
        $loppu = " AND (kampanjat.loppu <= :loppu OR kampanjat.jatkuva = 1)";
        $loppuPVM = $_REQUEST["loppu"]. "-31";
    }

    $alkuPVM = $_REQUEST["alku"];
    $alku = $alkuPVM. "-01";

    $sql = "SELECT kampanjat.id, kampanjat.nimi, kampanjat.alku, kampanjat.loppu, kampanjat.mediaeurot";

    if($_REQUEST["pohja"] == "muu"){
        $sql = $sql. ", kauppaKamppis.kauppaId FROM kampanjat
        JOIN kauppaKamppis ON kampanjat.id = kauppaKamppis.kamppisId
        WHERE kampanjat.alku >= :alku$loppu AND kauppaKamppis.kauppaId = :kauppa
        ORDER BY kampanjat.alku";
    }
    else{
        $sql = $sql. ", osuusKamppis.osuusId FROM kampanjat
        JOIN osuusKamppis ON kampanjat.id = osuusKamppis.kamppisId
        WHERE kampanjat.alku >= :alku$loppu AND osuusKamppis.osuusId = :kauppa
        ORDER BY kampanjat.alku";
    }
}

$kysely = $yhteys->prepare($sql);

$kysely->bindParam(":alku", $alku);
$kysely->bindParam(":kauppa", $_REQUEST["kauppa"]);

if(isset($_REQUEST["loppu"]) && $_REQUEST["loppu"] != null){
    $kysely->bindParam(":loppu", $loppuPVM);
}

$kysely->execute();

header("Content-type: application/json");
print json_encode($kysely->fetchAll(PDO::FETCH_ASSOC), JSON_PRETTY_PRINT);
```

Kuva 17. Kuvakaappaus ohjelmasta, joka hakee tietokannasta kampanjat, jotka sijoittuvat ajanjaksolle (Visual Studio Code 2020)

Luvussa 3.3.2 esiteltiin toiminnallisuus kuvassa 13, jossa muokattiin käyttöliittymän tietoja. Tässä samaisessa toiminnallisuudessa lähetetään kerätyt tiedot JavaScriptistä AJAX:ia hyödyntäen PHP:lle, jossa sitten haetaan kaikki kampanjat, jotka on liitetty kauppaan valitulta aikajaksolta. Kuten kuvassa 17 näkyy ohjelma aloittaa siitä, että se tarkistaa onko se saanut alkamiskuukauden ja sen jälkeen ohjelma tarkistaa onko loppukuukautta annettu. Ohjelma muodostaa SQL kyselyn riippuen siitä kumpi laskupohja on käytössä. SQL kyselyssä valitaan kaikki tarvittavat tiedot kampanjataulusta. Kampanjoihin haetaan niihin liitetyt kaupat ja sitten karsitaan listasta kampanjat, jotka ovat alkaneet ennen annettua ajankohtaa ja ne kampanjat, jota ei liitetty kauppaan, jota on haettu. Kyselyyn sidotaan tarvittavat parametrit ja suoritetaan. Tulos näytetään JSON muodossa, jota AJAX osaa lukea ja tuo sen käytettäväksi JavaScriptille. JavaScriptissä kampanjat listataan mediatilitysnäkymän omassa kentässä ja kaikki löytyneet kampanjat tuodaan laskun tauluun.

3.5 Tietotaso

Tietokannan taulujen sisältö pohjautuu toimeksiantajan olemassa oleviin Excel-taulukoihin, jota hän hyödyntää työssään. Toimeksiantajalta saatiin kuusi eri Excel-tiedostoa ja sen lisäksi tietoa oli Wordpress alustassa. Kaikki tieto on kerätty yhteen ja tehty siitä yksi luettava tiedosto, josta tietoa on alettu jakamaan osiin ja sen jälkeen jaettu omiin tietokanta tauluihin. Liitteessä 1 esitellään esimerkki Excel, jonne on koottu kaikki toimeksiantajan antamat tiedot.



Kuva 18. UML-kaavio tietokannan rakenteesta (Äyhynmäki 2020)

Tietokannassa käyttäjätunnusten lisäksi tauluja luotiin yhteensä kahdeksan, joista tärkein on kauppa. Kauppataulu sisältää kaikki myymälät, jonne kiinnitetään eri kampanjoiden mainoksia. Kuvassa 18 esitetään UML-kaaviolla tietokannan taulujen sisältö ja kuinka taulut ovat relaatiossa toisiinsa. Kuten kuvassa 18 näkyy, jokainen kauppa kuulu yhteen kauppaketjuun, mutta yhdellä kauppaketjulla on yksi tai useampi kauppa. Ketju voi kuulua osuuskuntaan. Osuuskunnalle voi kuulua useampi kuin yksi ketju. Kukin kauppa kuuluu tiettyyn jakelualueeseen. Yhdellä jakelualueella sijaitsee useampi kauppa, jota tietty jakelija hoitaa. Jakelija voi hoitaa useampaa kuin yhtä jakelualuetta. Jokaisella kaupalla on kauppias, mutta yhdellä kauppiaalla voi olla useampi kauppa. Kaupoissa voi olla käynnissä tai tulossa yksi tai useampi kampanja. Jokaisella kampanjalla on toimeksianto yritykseltä ja/tai mediatoimistolta. Yksi kampanja voi olla käynnissä useamassa myymälässä ja yhdessä myymälässä voi olla käynnissä samanaikaisesti useampi kampanja.

MySQL-tietokantaan luotiin ensin kaikki kahdeksan taulua. Tauluihin määriteltiin mitkä kentät ovat pakollisia ja mitkä saavat jäädä tyhjäksi, kuten esimerkki taulukossa 2. Tämän jälkeen taulujen välille luotiin tarvittavat relaatiot. Esimerkiksi kauppataulun ketju id-sarake on relaatiossa ketjutaulun id-sarakkeeseen.

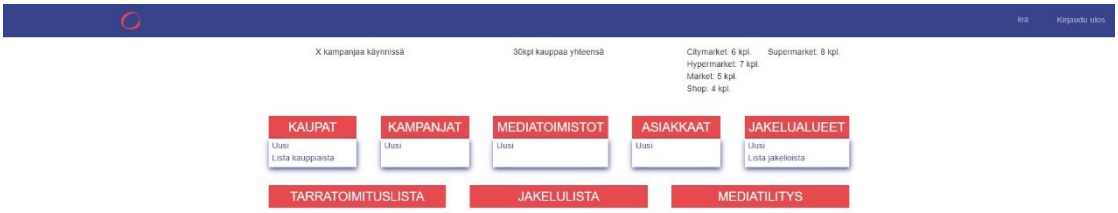
Taulukko 2. Kampanja taulun rakenne

Sarakkeen nimi	Tyyppi	Oletus arvo	Saako olla tyhjä?
Id	Numero		Ei
Nimi	Teksti		Ei
Asiakas id	Numero		Ei
Toimisto id	Numero	Null	Kyllä
Alku	Päivämäärä (PP.KK.VVVV)		Ei
Loppu	Päivämäärä (PP.KK.VVVV)	Null	Kyllä
Jatkuva	Numero (1 tai 0)	0	Ei
Korvaus	Desimaali numero		Kyllä
Info	Teksti	Null	Kyllä

Relaatioiden määrittelyn jälkeen syötettiin Exceliin kerätyt tiedot ja lisättiin ne omiin tauluihinsa. Ainoa taulu, joka jäi ilman sisältöä tiedon lisäyksen jälkeen, oli kampanjataulu.

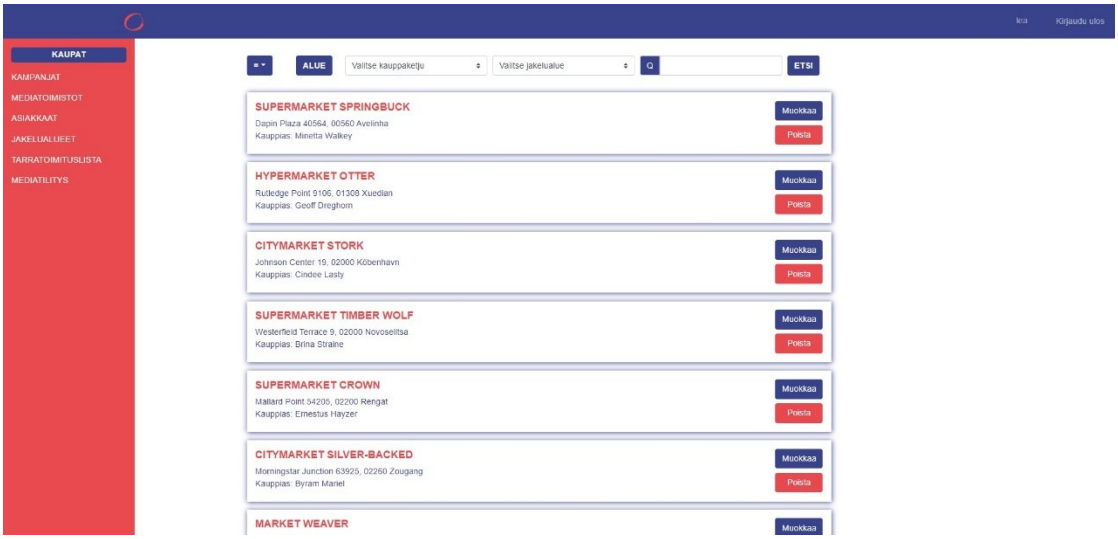
3.6 Toimeksiantajalle luovutettu sovellus

Toimeksianto luovutettiin asiakkaalle toukokuussa 2020. Luovutuksen yhteydessä pidettiin palaveri, jossa sovellus esiteltiin ja käytiin läpi kaikki toiminnot, sekä mistä muutokset tehdään. Asiakas vaikutti tyytyväiseltä lopputulokseen, vaikka tekijää jäi hieman harmittamaan, että ei ollut ehtinyt panostamaan kauheasti responsiivisuuteen, kun itse toimintojen ohjelmointi oli syönyt suuriman osan ajasta.



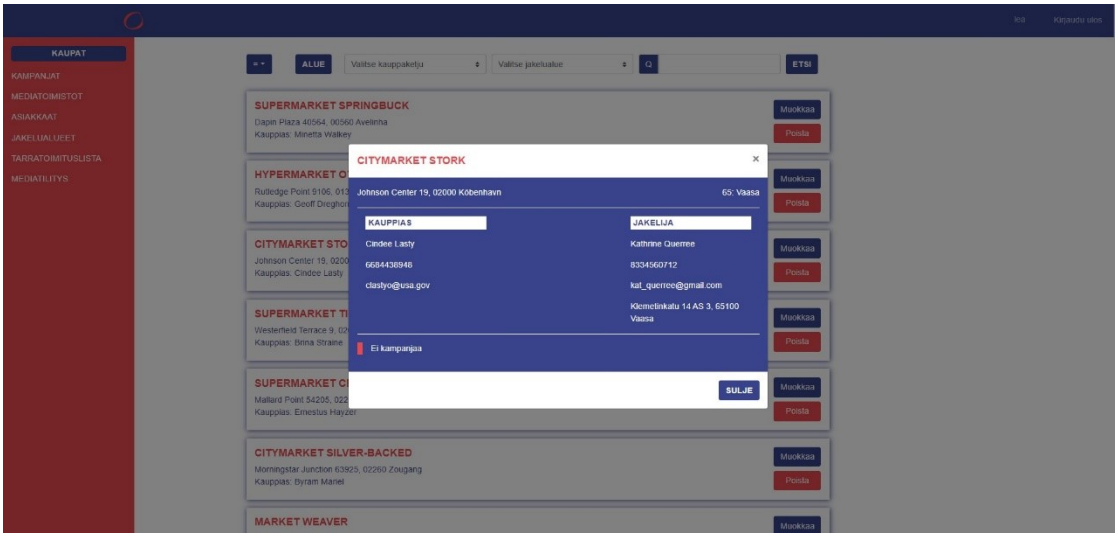
Kuva 19. Kuvakaappaus sovelluksen etusivusta (Äyhynmäki 2020)

Kuvassa 19 esitellään sovelluksen etusivu, joka avautuu onnistuneen kirjautumisen jälkeen. Etusivu ei muuttunut prototyypistä, mutta siitä jäi pois toiminnallisuus, joka oli prioriteetti listalla hyvin alhaalla. Tämä olisi ollut tarkoitus toteuttaa vain, jos aikaa olisi jäänyt yli. Etusivun alaosaan oli suunniteltu varoituslaatikko, joka ilmaantuu näkymään, jos jostain kaupasta puuttuu jokin tärkeä yhteistieto tai kauppias. Etusivussa näkyy myös muitakin osioita, jota ei ole käsitelty toimeksiannossa. Nämä osiot eivät vie näkymään. Ne ovat siinä niin sanotusti koristeena, koska on tarkoitus, että jossain vaiheessa nämäkin toteutetaan sovellukseen.



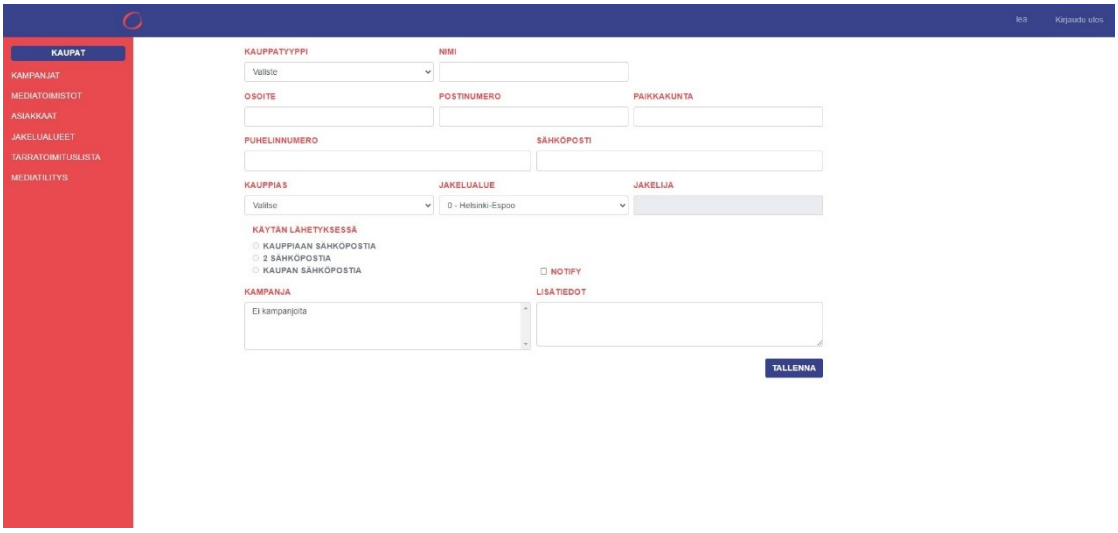
Kuva 20. Kuvakaappaus sovelluksen kauppalistauksesta (Äyhynmäki 2020)

Verratessa kuvaa 20 ja kuvaa 10, kappaleessa 3.3.1. voi nähdä kuinka kauppalistaus on kehittynyt kaksisarakkeisesta listasta yksisarakkeiseksi helppoluettavuuden vuoksi. Muuten kauppalistaus ei kauheasti muuttunut prototyypistä. Kuvassa 21 esitellään lisätietoikkuna, joka avautuu painaessa kauppalistauksessa kauppaa.



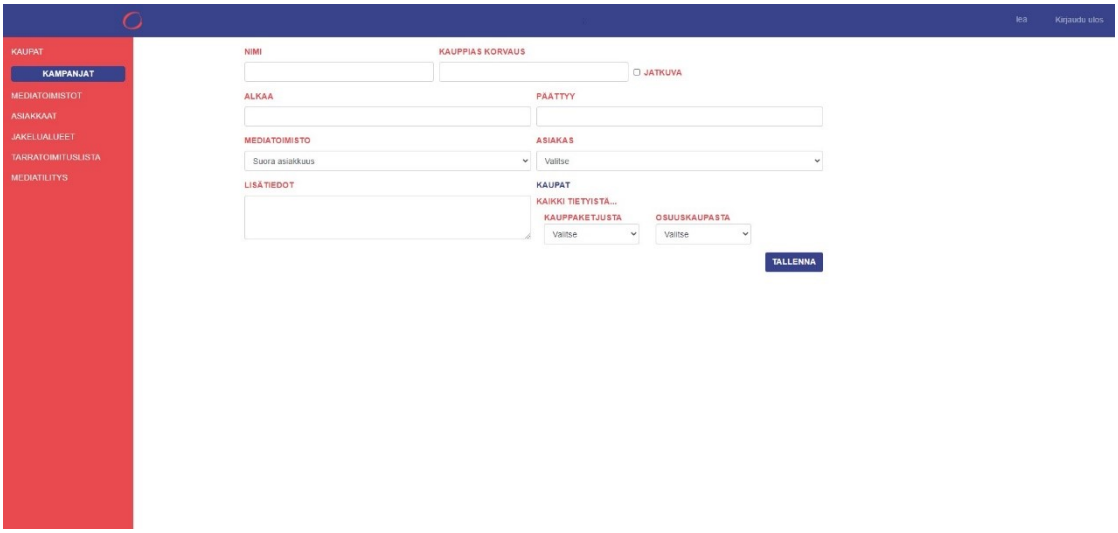
Kuva 21. Kuvakaappaus kaupan lisätiedoista (Äyhynmäki 2020)

Tietojen lisäys näkymät kuvissa 22 ja 23, ovat hyvin samanlaisia kuin prototyypissä. Joitain lomakkeiden nimiä vaihdettiin ja joitain valintoja lisättiin näkymään toimeksiantajan pyynnöstä.



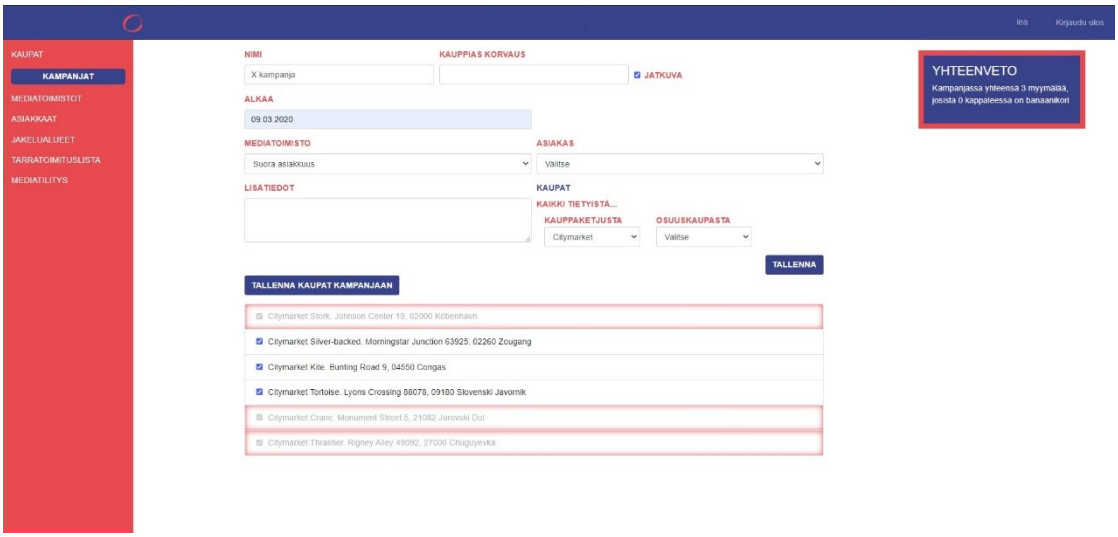
Kuva 22. Kuvakaappaus uuden kaupan lisäyksestä (Äyhynmäki 2020)

Esimerkiksi kuvassa 22, toiseksi viimeisellä rivillä kysytään mitä sähköpostia halutaan käyttää mediatilityksessä. Vaihtoehtoina ovat kauppiaan sähköposti, kauppiaan vaihtoehtoinen sähköposti ja kaupan sähköposti, joista kauppiaan sähköposti on oletuksena valittu, jos kauppiaalla on kirjattu tietoihin sähköpostiosoite. Tämä valinta lisättiin siksi, koska kauppiailla voi olla useampi kuin yksi kauppa ja mahdollisesti haluavat tilitykset eri sähköpostiin, taas kun toiset voivat haluta sen samaan. Lisäys vaati ohjelmointia mediatilitysnäkymään, jossa koodi tarkistaa, että mikä sähköposteista, jotka löytyvät tietokannassa, tuodaan laskuun.

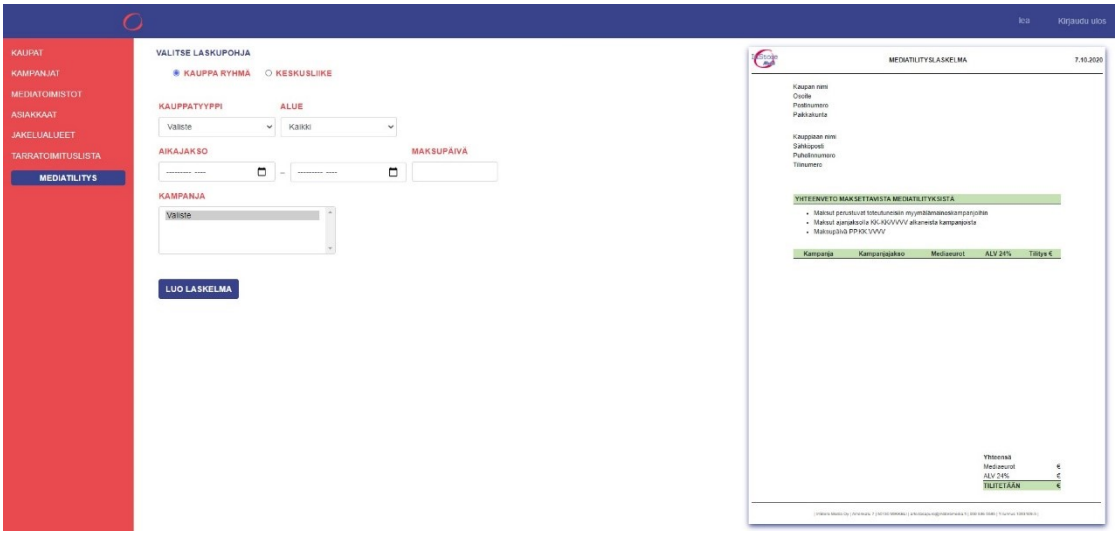


Kuva 23. Kuvakaappaus uuden kampanjan lisäyksestä (Äyhynmäki 2020)

Luodessaan uuden kampanjan voi siihen kiinnittää kaupat. Kuvassa 24 on näkymä, joka tulee näkyville, kun kiinnittää kauppoja kampanjaan. Näkymän oikeassa reunassa on yhteenveto, joka kertoo, kuinka monta kauppaa on kiinnitettävänä. Kauppalistauksessa punaisella ympäröidyt kaupat esittävät kauppoja, jotka ovat jo kiinnitetty.



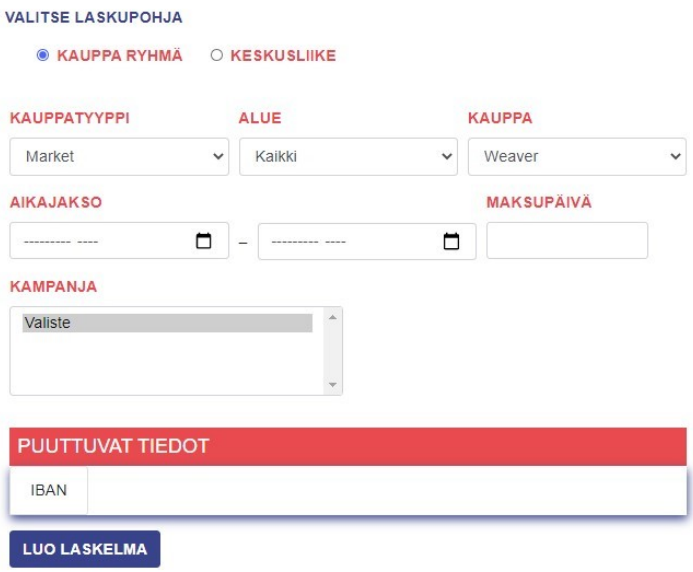
Kuva 24. Kuvakaappaus kauppojen kiinnityksestä uuteen kampanjaan (Äyhynmäki 2020)



Kuva 25. Kuvakaappaus sovelluksen mediatilityksestä (Äyhynmäki 2020)

Mediatilitysnäkymä on esitetty kuvassa 25. Sekään ei ole kauheasti muuttunut prototyypistä. Lisävalintana siihen lisätiin, että kauppoja voi rajata alueen mukaan, koska useammalla kauppaketjulla on hyvin monta kauppaa ympäri maata. Aiemmin kappaleessa mainittiin varoitusikkuna, jossa ilmoitetaan puuttuvista tiedoista, tämä toiminto toteutettiin pienemässä skaalassa mediatilitysnäkymässä. Kun tilitykseen valitsee kaupan, koodi tarkistaa, että puuttuuko

kaupantiedoista tärkeitä tietoja ja ilmoittaa sen näkymässä, kuten kuvassa 26 näkyy. Laskun luoja voi sitten helposti reagoida puuttuvaan tietoon ja mahdollisesti lisätä sen ennen kuin luo laskua.



Kuva 26. Kuvakaappaus mediatilityksen huomautuksesta puuttuvasta tiedosta (Äyhynmäki 2020)

4 PÄÄTÄNTÖ

Työssä perehdyttiin sovelluskehityksessä käytettyyn käsitteeseen kolmikerrosmalli. Mallia käytetään usein laajan sovelluksen toteutuksessa, jossa sovelluksen ydin on tietokannan tieto. Usein MVC (Model-View-Controller) ja kolmikerros sekoitetaan keskenään tai oletetaan, että nämä kaksi sovellusarkkitehtuuria ovat sama asia. Marston (2012) esittää selkeästi erovaisuuden näiden kahden arkkitehtuurin välillä. MVC:n ohjain (eng. Controller) ja näkymä (eng. View) sijoittuvat kolmikerrosmallissa esitystasoon. Malli (eng. Model) sisältää toimintoja, jotka kolmikerrosmallissa sijoittuvat tehtävä- ja tietotasoon. Työ esittää selkeästi mistä kolmikerrosmalli koostuu ja miten se toteutetaan. Toimeksianto antaa hyvän esimerkin siitä minkälaisen sovelluksen kolmikerrosmallilla voi toteuttaa. Toimeksianto korostaa kolmikerrosmallin vahvuuksia siitä, että sovellusta voi laajentaa ja muokata, ilman että joutuu kirjoittamaan kaikkea koodia uudelleen, kun muutoksia tulee.

Toimeksianto antoi paljon realistisemman ja konkreettisemman kuvan siihen, mitä kaikkea vaaditaan ja mitä vaihtoehtoja on, että sovelluksen voi julkaista

internetissä. Suurimpana yllätyksenä tuli, että jos halusi keskittää ja käyttää web-hotellia niin suurin osa niistä tuki ainoastaan PHP-kieltä palvelin puolella. PHP ohjelmointikieli oli kaikista viimeisin kieli minkä tekijä oli oppinut. Jos web-hotellit olisivat tukeneet, tekijä olisi toteuttanut toimeksiannon Angularilla. Toiveena olisi, että web-hotellit päivittyisivät ohjelmointi kielien mukaan ja niitä pidettäisiin ajan tasalla.

Yllätyksenä tuli myös sovelluksen laajuus, vaikka työn tavoitteita oli reippaasti karsittu ja arvioitu sopivaksi määräksi opinnäytetyön työosuuden laajuuteen katsottuna. Sovelluksen ohjelmoinnin edetessä tuli yllätyksellisiä muutoksia ja lisää vaatimuksia, jotka eivät olleet tulleet esille sovelluksen suunnitteluvaiheessa. Tämän takia sovelluksesta jäi kokonaan pois tietojen muokattavuus. Asiasta sovittiin toimeksiantajan kanssa. Hän on tietoinen mistä ja miten tietoja muokataan tarpeen tullen, koska sovelluksen tärkein osa ja isoin prioriteetti on laskujen automaattinen luonti.

Toimeksianto oli opettavainen ja avasi silmät sille, että vaikka kokee esittävän ja rajaavan sovelluksen toiminnallisuudet tarkasti suunnittelu vaiheessa, voi asiakas silti yllättää, vaikka luulet ottaneen huomioon kaiken. Se myös lisäsi varmuutta uskaltaa ohjelmoida kielellä, joka ei ole vahvasti hallussa. Vaikka miettiikin monessa vaiheessa, että onko tämä oikeaoppisesti tehty.

Vaikka rajat olivatkin selvät, kun asiakas alkaa nähdä sovellusta kokonaisuudessa, heiltä yleensä tulee lisää vaatimuksia tai suuria muutoksia. Yleensä on se ikävä tosiasia, että asiakkaat uskovat, että ohjelmointi on hyvin yksinkertaista ja sitä tehdään tuosta noin vain ilman mitään ajatustyötä. Ohjelmoijana saa usein kuulla, miten yksinkertaista on kirjoittaa pari riviä tai että siihen käytettävä aika on korkeintaan muutama tunti. Ohjelmointi voi olla hyvin yksinkertaista jossain tapauksessa. Esimerkiksi pelkkä HTML sivu, joka sisältää otsikoita ja sisältö tekstiä, ilman mitään toimintoja taikka tyylejä. Toiminnallisuudessa pitää miettiä miten sivu käyttäytyy, mitä sinne tulee, missä järjestyksessä ja miten ohjelman kokoaa kaikista järkevimmällä tavalla. Sitten pitää ottaa huomioon virheet ja niiden käsittely, laitteet, jolla sitä käytetään, selaimet ja näyttöjen koot. Se ei ole niin yksinkertaista kuin voisi uskoa.

LÄHTEET

3-Tier Architecture: A complete overview. s.a. Logi Report. WWW-dokumentti. Saatavissa: https://www.jinfonet.com/resources/bi-defined/3-tier-architecture-complete-overview/ [viitattu 22.10.2020].

An, E. 2020. What exactly is a full-stack web developer? CareerFoundry. Blogi. Päivitetty: 6.1.2020. Saatavissa: https://careerfoundry.com/en/blog/web-development/what-is-a-full-stack-web-developer/ [viitattu 4.3.2020].

Bhardwaj, K. 2020. Top 6 CSS Frameworks for Web Developers. Tekki Web Solutions. Blogi. Päivitetty: 16.7.2020. Saatavissa: https://tekkiwebsolutions.com/blog/best-css-frameworks-for-web-developers/ [viitattu 21.10.2020].

Carr, D. & Gray, M. 2018. Beginning PHP. E-kirja. Packt Publishing. Saatavissa: https://kaakkuri.finna.fi [viitattu 27.10.2020].

Converse, T., Morgan, C. & Park, J. 2004. PHP5 and MySQL Bible. E-kirja. Hoboken: John Wiley & Sons, Inc. Saatavissa: https://kaakkuri.finna.fi [viitattu 29.10.2020].

CSS: Cascading Style Sheets. 2020. MDN web docs. WWW-dokumentti. Päivitetty: 30.9.2020. Saatavissa: https://wiki.developer.mozilla.org/en-US/docs/Web/CSS [viitattu 1.10.2020].

Darmawikarta, D. 2014. SQL for MySQL: A Beginner's Tutorial. E-kirja. Brainy Software. Saatavissa: https://kaakkuri.finna.fi [viitattu 29.10.2020]

Gustafson, J. M. 2013. HTML5 Web Application Development By Example Beginner's guide. E-kirja. Packt Publishing. Saatavissa: https://kaakkuri.finna.fi [viitattu 19.10.2020].

History of PHP. s.a. The PHP Group. WWW-dokumentti. Saatavissa: https://www.php.net/manual/en/history.php.php [viitattu: 27.10.2020].

HTML, Living Standard. 2020. WHATWG. WWW-dokumentti. Päivitetty: 14.10.2020. Saatavissa: https://html.spec.whatwg.org/multipage/introduction.html#history-2 [viitattu 16.10.2020].

Larsen, R. & Duckett, J. 2013. Beginning HTML and CSS. E-kirja. Wrox. Saatavissa: https://kaakkuri.finna.fi [viitattu 14.10.2020].

Lawrence, M. 2019. What is a CSS Framework? Medium. Blogi. Päivitetty: 3.5.2019. Saatavissa: https://medium.com/html-all-the-things/what-is-a-css-framework-f758ef0b1a11 [viitattu 21.10.2020].

Marston, T. 2012. What is the 3-Tier Architecture? Blogi. Päivitetty: 28.4.2020. Saatavissa: http://tonymarston.net/php-mysql/3-tier-architecture.html [viitattu 28.10.2020].

Rajput, A. DOM (Document Object Model). 2019. GeeksForGeeks. WWW-dokumentti. Päivitetty: 19.2.2019. Saatavissa: https://www.geeksforgeeks.org/dom-document-object-model/ [viitattu: 20.10.2020].

Rishabh. 2020. What is .NET 3-Tier Architecture? GeeksforGeeks. WWW-dokumentti. Päivitetty: 14.1.2020. Saatavissa: https://www.geeksforgeeks.org/what-is-net-3-tier-architecture/ [viitattu 28.10.2020].

Stefanov, S. & Sharma, K. C. 2013. Object-Oriented JavaScript - Second Edition. E-kirja. Birmingham: Packt Publishing. Saatavissa: https://kaakkuri.finna.fi [viitattu 12.10.2020].

Structured Query Language (SQL). 2017. Microsoft. WWW-dokumentti. Päivitetty: 19.1.2017. Saatavissa: https://docs.microsoft.com/en-us/sql/odbc/reference/structured-query-language-sql?redirectedfrom=MSDN&view=sql-server-ver15 [viitattu 29.20.2020].

Three-Layered Services Application. 2014. Microsoft. WWW-dokumentti. Päivitetty: 17.3.2014. Saatavissa: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff648105(v=pandp.10) [viitattu 28.10.2020].

Three-tier architecture. 2015. Managementmania. Blogi. Päivitetty: 5.12.2015. Saatavissa: https://managementmania.com/en/three-tier-architecture [viitattu 28.10.2020].

Using a Three-Tier Architecture Model. 2018. Microsoft. WWW-dokumentti. Päivitetty: 31.5.2018. Saatavissa: https://docs.microsoft.com/en-us/windows/win32/cossdk/using-a-three-tier-architecture-model [viitattu 28.10.2020].

Why is CSS so weird? 2019. Mozilla Developer. Videotallenne. Saatavissa: https://www.youtube.com/watch?v=aHUtMbJw8iA [viitattu 1.10.2020].

KUVALUETTELO

TAULUKKOLUETTELO

Liite 1

Ketju	Kaupan nimi	Kaupan osoite	Postinumero	Kauppias	Puhelin	Sähköposti	Tilinumero	Jakelukoodi
Hypermarket	Otter	Rutledge Point 9106	1308	Geoff Dreghorn	1274135352	gdreghorna@kickstarter.com	*NULL*	00
Market	Heron	Lake View Place 52	7067	Turner Rushmere	5281695431	trushmere3@europa.eu	GE29 UW63 8235 6082 79	40
Shop	Emu	Hollow Ridge Alley 8	4860	Quintana Ox	6486972169	qox8@engadget.com	*NULL*	40
Market	Weaver	Claremont Way 46	3880	Dur Sermin	6227324282	dserminc@chron.com	*NULL*	40
Citymarket	Thrasher	Rigney Alley 49092	27000	Milty Hilbourne	4967630612	mhilbournei@phpbb.com	FR54 0837 5121 72PY FO	40
Supermarket	Coatimundi	Steensland Way 951	98420	Rupert Crum	9927907202	rcrumt@telegraph.co.uk	*NULL*	40
Citymarket	Crane	Monument Street 5	21082	Sly Coen	2675646245	scoen5@desdev.cn	*NULL*	50
Hypermarket	Vine snake	Anniversary Alley 41429	48955	Gerianne Maffiotti	9799769811	gmaffiottie@hubpages.com	DE67 0705 2892 0771 92	50
Supermarket	Goose	Pond Center 3	6316	Auberon Thewles	2377609133	athewlesm@cnet.com	GT47 01ST UEDW FIRA XM	50
Market	Cormorant	Northridge Lane 8	25200	Abraham Swepson	3079059345	aswepsonn@a8.net	SM87 T098 4503 953P SX	50
Shop	Gull	Walton Trail 7849	6000	Emeline Thebeau	9595538172	ethebeauq@gnu.org	LT25 4385 4192 9257 82	50

Jakelualue	Jakelija	Jakelijan osoite	Jakelijan puh.	Jakelijan s.posti	Jakelijan IBAN
Helsinki-Espoo	Florri Hurdwell	Lontoonkatu 9, 00550 Helsinki	6823391234	hurdwell@outlook.com	NO45 3411 4622 392
Jyväskylä	Martina Goodie	Helokantie 1 B 10, 40640 Jyväskylä	4021127738	martina.goodie@outlook.com	KZ24 514A DPGX MKNE I
Jyväskylä	Martina Goodie	Helokantie 1 B 10, 40640 Jyväskylä	4021127738	martina.goodie@outlook.com	KZ24 514A DPGX MKNE I
Jyväskylä	Martina Goodie	Helokantie 1 B 10, 40640 Jyväskylä	4021127738	martina.goodie@outlook.com	KZ24 514A DPGX MKNE I
Jyväskylä	Martina Goodie	Helokantie 1 B 10, 40640 Jyväskylä	4021127738	martina.goodie@outlook.com	KZ24 514A DPGX MKNE I
Jyväskylä	Martina Goodie	Helokantie 1 B 10, 40640 Jyväskylä	4021127738	martina.goodie@outlook.com	KZ24 514A DPGX MKNE I
Mikkeli	Gabriell Kilian	Maahisentaival 1 B 27, 50970 Mikkeli	1556123282	gabriell.kilian@gmail.com	LV75 TDBW AUVT PTT7 PU
Mikkeli	Gabriell Kilian	Maahisentaival 1 B 27, 50970 Mikkeli	1556123282	gabriell.kilian@gmail.com	LV75 TDBW AUVT PTT7 PU
Mikkeli	Gabriell Kilian	Maahisentaival 1 B 27, 50970 Mikkeli	1556123282	gabriell.kilian@gmail.com	LV75 TDBW AUVT PTT7 PU
Mikkeli	Gabriell Kilian	Maahisentaival 1 B 27, 50970 Mikkeli	1556123282	gabriell.kilian@gmail.com	LV75 TDBW AUVT PTT7 PU
Mikkeli	Gabriell Kilian	Maahisentaival 1 B 27, 50970 Mikkeli	1556123282	gabriell.kilian@gmail.com	LV75 TDBW AUVT PTT7 PU