VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Mika Aho

WEB-SOVELLUS TUOTTEIDEN SUODATUKSEEN JA HINNOITTELUUN VERKKOKAUPPAJÄRJESTELMÄSSÄ

Tekniikka ja liikenne
2014

VAASAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

TIIVISTELMÄ

Tekijä	Mika Aho
Opinnäytetyön nimi	Web-sovellus tuotteiden suodatukseen ja hinnoitteluun verkkokauppajärjestelmässä
Vuosi	2014
Kieli	suomi
Sivumäärä	51
Ohjaaja	Martti Mustonen

Tämän opinnäytetyön tarkoituksena oli suunnitella ja kehittää sisäiseen käyttöön tarkoitettu web-sovellus Multitronic Oy:lle, joka on tietotekniikkatuotteita myyvä verkkokauppa. Web-sovellusta voidaan käyttää myytävien tuotteiden suodatukseen ja hinnoitteluun. Tuotteita voidaan suodattaa tuotteen kategorian ja ominaisuuksien perusteella ja hinnoitella joko tuotekohtaisesti tai määrittelemällä sääntöjä, jotka koskevat tiettyjä tuotekategorioita. Multitronicin järjestelmässä ylläpidetään useita erillisiä verkkokauppoja, ja tässä opinnäytetyössä kehitetty web-sovellus tarjoaa niille kaikille samat suodatus- ja hinnoitteluominaisuudet erikseen.

Web-sovelluksessa on kaksi osaa: asiakassovellus ja palvelinsovellus. Asiakassovellusta käytetään verkkoselaimen kautta, ja se on toteutettu käyttäen JavaScript-kieltä ja jQuery-kirjastoa. Se tarjoaa sovelluksen käyttöliittymän ja kommunikoi palvelinsovelluksen kanssa yksinkertaisen ohjelmointirajapinnan kautta. Loppukäyttäjälle näkymätön palvelinsovellus on toteutettu PHP-ohjelmointikielellä, ja se suorittaa varsinaiset käyttäjän pyytämät toimenpiteet luomalla tuotteiden suodatus- ja hinnoittelusääntöjä tietokantaan. Lisäksi kehitettiin automaattisia tietokantaproseduureja, jotka tulkitsevat näitä sääntöjä ja huolehtivat niiden täytäntöönpanosta käytännössä.

Opinnäytetyön tuloksena kehitettiin sovellus, joka vastaa tuotteiden suodatukselle ja hinnoittelulle asetettuja vaatimuksia. Tätä opinnäytetyötä kirjoitettaessa sovellus ei ollut vielä tuotantokäytössä, koska sitä kehitettiin osaksi uutta verkkokauppajärjestelmää, joka oli vielä kehitteillä toimeksiantajayrityksessä.

Avainsanat verkkokauppa, PHP, JavaScript, AJAX, SPA

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikan koulutusohjelma

ABSTRACT

Author	Mika Aho
Title	Web Application for Product Filtering and Pricing in an Online Store System
Year	2014
Language	Finnish
Pages	51
Name of Supervisor	Martti Mustonen

The purpose of this thesis was to design and develop a web application for internal use at Multitronic Oy, an online IT retailer. This web application can be used to filter and price the products to be sold. Products can be filtered based on product category and properties and priced either individually or by defining rules that apply to specific product categories. There are multiple different online stores maintained in Multitronic Oy's system and the web application developed for this thesis gives each of them the same product filtering and pricing capabilities separately.

The web application has two parts: a client-side application and a server-side application. The client-side application is used through a web browser, and it has been implemented using the JavaScript language and jQuery library. It provides the user interface for the application and communicates with the server-side application through a simple application programming interface (API). The server-side application, not visible to the users, has been implemented using the PHP programming language, and it executes the actual actions requested by the user by creating product filtering and pricing rules to the database. In addition, automatic stored procedures were developed in the database to interpret these rules and to put the filtering and pricing definitions into action.

As a result of this thesis, an application was developed that meets the requirements of filtering and pricing the products. The application was not yet in production at the time of writing this thesis because it was developed as part of a new e-commerce system which was still in development at the client company.

Keywords online store, PHP, JavaScript, AJAX, SPA

SISÄLLYS

KUVIOLUETTELO

TAULUKKOLUETTELO

TERMIT JA LYHENTEET

AJAX	Asynchronous JavaScript + XML, kommunikaatiomenetelmä interaktiivisille verkkosivuille
API	Application Programming Interface, ohjelmointirajapinta
CSS	Cascading Style Sheet, tyyliohje
DOM	Document Object Model, ohjelmointirajapinta HTML-dokumenttien sisällön muokkaamiseen
HTML	HyperText Markup Language, kuvauskieli
HTTP	Hypertext Transfer Protocol, hypertekstin siirtoprotokolla
JavaScript	Ohjelmointikieli
JSend	Määritelmä JSON-muotoisen AJAX-vastauksen rakenteelle
JSON	JavaScript Object Notation, tiedostomuoto tiedonsiirtoon
Nested Set	Menetelmä hierarkkisen tiedon tallentamiseen relaatiotietokannassa
PHP	PHP: Hypertext Preprocessor, ohjelmointikieli
PL/pgSQL	PostgreSQL-tietokannan sisäinen ohjelmointikieli
PostgreSQL	Relaatiotietokantapalvelinohjelmisto
SPA	Single-page application, yhden sivun web-sovellus
SQL	Structured Query Language, relaatiotietokannan kyselykieli
QFD	Quality Function Deployment, menetelmä vaatimusten priorisointiin

1 JOHDANTO

Tuotteiden, palvelujen ja sisältöjen ostaminen verkosta on yleistynyt viime vuosina suomalaisten kuluttajien keskuudessa. Erityisesti kasvua on nähty vähittäiskaupassa, jonka verkkomyynnin osuus on lisääntynyt neljän viime vuoden aikana lähes 30 prosenttia. Osa kasvusta selittyy sillä, että suomalaiset tekevät ostoksensa yhä useammin ulkomaisista verkkokaupoista. /1/ Vuonna 2013 jo reilu kolmannes suomalaisista teki verkko-ostoksia ulkomailta /2/. Kilpailu alalla on kovaa, ja eri asiakaskuntien palvelu parhaalla mahdollisella tavalla on tärkeää.

Opinnäytetyön toimeksiantajana toimii Multitronic Oy, joka harjoittaa erityisesti tietotekniikkatuotteiden vähittäismyyntiä verkossa. Multitronic Oy:n kehittämässä verkkokauppajärjestelmässä ylläpidetään useita erillisiä verkkokauppoja, jotka voivat kaupasta riippuen keskittyä eri tuoteryhmien myyntiin tai eri maiden asiakaskuntien palveluun. Järjestelmässä ylläpidettävien verkkokauppojen tarpeet, kuten tuotevalikoima tai tuotteiden hinnoitteluperiaatteet, voivat olla hyvinkin erilaiset kaupasta riippuen. Kaikkia tuotteita ei haluta välttämättä myyntiin kaikissa verkkokaupoissa ja tuotteet halutaan hinnoitella kussakin verkkokaupassa eri tavalla. Lisäksi verkkokaupoilla voi olla täysin omia räätälöityjä tuotteita tai palveluita, joiden myyminen muissa verkkokaupoissa ei olisi mahdollista.

Opinnäytetyön tarkoituksena on kehittää yrityksen sisäiseen käyttöön tarkoitettu web-sovellus osana yrityksen uutta verkkokauppajärjestelmää. Sovelluksen avulla kunkin verkkokaupan tuotevalikoimiin voidaan poimia helposti halutut tuotteet ja hinnoitella ne jokaiselle verkkokaupalle erikseen. Kullekin verkkokaupalle voidaan siis luoda muista verkkokaupoista riippumattomat säännöt tuotevalikoiman suodatukselle ja tuotteiden hinnoittelulle.

Aiemmassa järjestelmässä nämä verkkokaupat olivat lähemmin riippuvaisia toisistaan, mutta uuteen järjestelmään ne halutaan paremmin toisistaan riippumattomammiksi ja ominaisuuksiltaan mahdollisimman tasavertaisiksi tuotteiden suodatuksen ja hinnoittelun suhteen. Kehitettävä web-sovellus tulee näin tärkeään roo-

liin osaksi sitä ketjua, jolla tukkukauppojen tarjoamat tuotteet ja verkkokauppojen omat tuotteet saadaan myyntiin asiakkaille asti.

2 MULTITRONIC OY

Multitronic Oy on vuonna 1995 perustettu suomalainen tietotekniikkatuotteiden myyntiin keskittynyt yritys. Multitronic myy tuotteita sekä verkkokauppansa kautta etämyyntinä että suoraan Vaasassa ja Pietarsaaressa sijaitsevista liikkeistään. Lisäksi Multitronic tarjoaa asiakkailleen tietokoneiden, niiden oheislaitteiden ja ohjelmistojen huoltopalveluja sekä palvelintilaa verkkosivustojen pystyttämiseen ja tähän liittyviä palveluja. /3/

Multitronicin verkkokauppa ja monet muut yrityksen toimintaan liittyvät ohjelmistot ovat pääasiassa yrityksen omien työntekijöiden kehittämiä, eivätkä ne pohjaudu muihin olemassa oleviin järjestelmiin. Näin kaikki tekniset ratkaisut on voitu rakentaa alusta alkaen vastaamaan yrityksen tarpeita. /3/

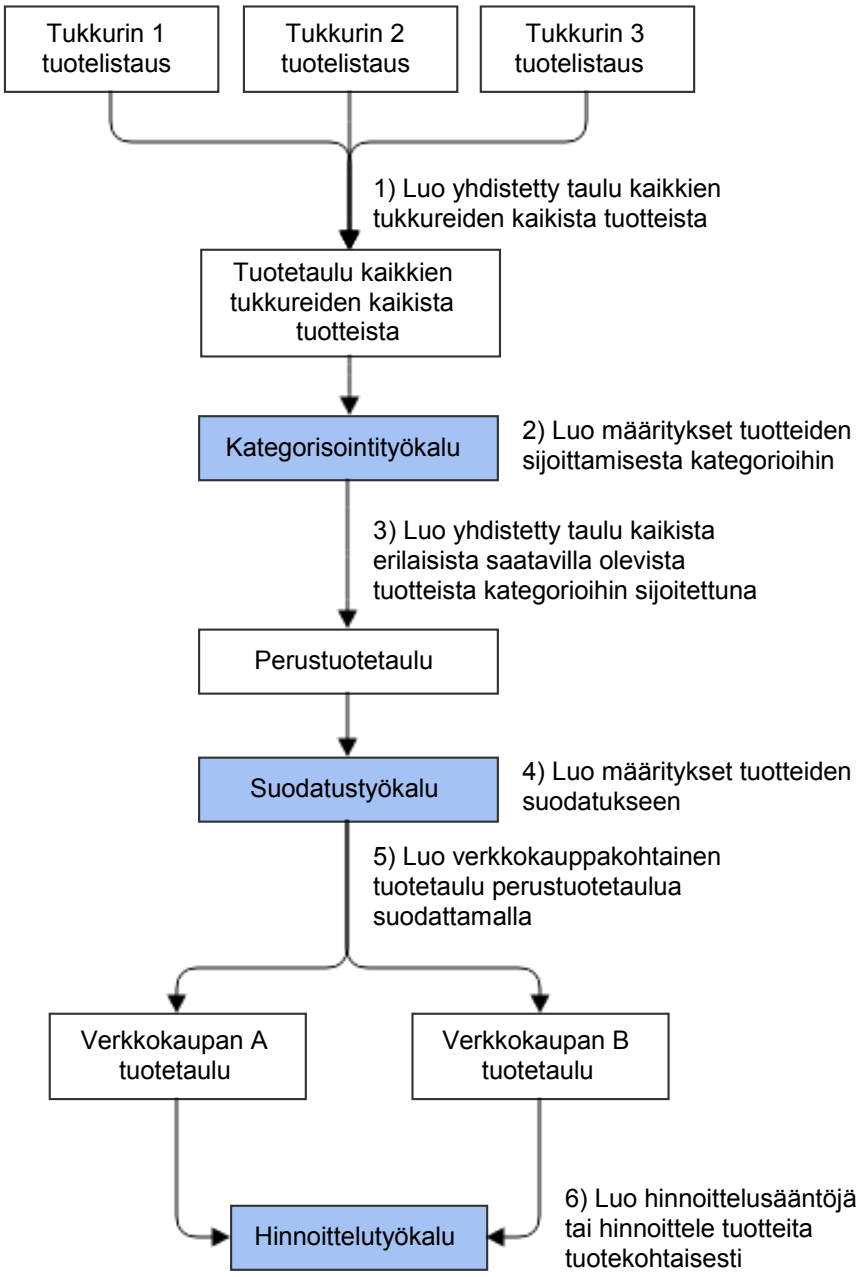
3 TYÖN TAUSTA JA LÄHTÖTILANNE

Tässä kappaleessa kuvataan opinnäytetyössä kehitetyn web-sovelluksen taustaa ja teknistä tilannetta, josta työtä lähdettiin tekemään.

3.1 Prosessi tuotteiden tuomiseksi verkkokauppajärjestelmään

Multitronic Oy:n tuotevalikoima koostuu pääasiallisesti tukkukaupoilta saatujen tuotelistojen tuotteista. Nämä tuotelistat sisältävät muun muassa tuotteen nimen, tuotekoodin, hinnan ja tukkukaupan itse määrittelemän kategorian tuotteelle. Tukkukaupoilta saatavat tuotelistat eivät ole yhtenäisessä formaatissa, vaan jokainen niistä on hieman erilainen. Näistä tuotelistoista luodaan Multitronic Oy:n järjestelmässä edelleen yksi yhdistetty tietokantataulu, joka sisältää kaikkien tukkureiden kaikki saatavilla olevat tuotteet yhtenäisessä, yritykselle sopivassa muodossa. Tästä tietokantataulusta löytyvien tietojen perusteella voidaan tehdä erityisellä kategorisointityökalulla määritykset kunkin tuotteen sijoittamiseen johonkin toimeksiantajayrityksen tuotekategoriaan. Samalla nähdään, miltä kaikilta tukkureilta mitkäkin tuotteet ovat saatavissa. Prosessin lopputuloksena saadaan tietokantaan perustuotetaulu, joka sisältää kaikki erilaiset myytävissä olevat tuotteet verkkokauppajärjestelmän yleiseen kategoriarakenteeseen sijoitettuna. Tähän tauluun voidaan myös lisätä yrityksen omia tuotteita.

Edellä kuvatun prosessin tuloksena saatua perustuotetaulua ei voida sellaisenaan käyttää minkään yrityksen järjestelmässä ylläpidettävän verkkokaupan tuotevalikoimana vaan siitä täytyy suodattaa kullekin verkkokaupalle oma versio. Tämä johtuu siitä, että verkkokaupoilla on toisistaan eroavat tarpeet tuotevalikoiman ja tuotteiden hinnoittelun suhteen. Lisäksi tukkukaupoilta saadut tuotelistat saattavat sisältää tuotteita, joita ei muista syistä voida myydä. Prosessia on havainnollistettu kokonaisuudessaan seuraavassa kuviossa 1. Tässä listatut tukkukaupat ja verkkokaupat toimivat vain esimerkkeinä, eivätkä vastaa niiden todellista määrää.



Kuvio 1. Tuotteiden sisääntuontiprosessi

Toimeksiantajayrityksen aiemmassa verkkokauppajärjestelmässä on jo tehty jossain määrin vastaavanlaista tuotteiden suodatusta ja hinnoittelua, mutta nämä toimenpiteet ovat olleet hieman erilaisia verkkokaupasta riippuen. Lisäksi osa verkkokaupoista on ollut riippuvaisia toisista verkkokaupoista, jolloin esimerkiksi tietyn verkkokaupan tuotevalikoiman lähtökohtana on voinut olla toisen verkkokaupan tuotevalikoima. Yrityksen uudistetussa verkkokauppajärjestelmässä tämä haluttiin muuttaa niin, että kaikki verkkokaupat ovat tasavertaisessa asemassa.

3.2 Olemassa olevien toteutusten hyödyntäminen

Toimeksiantajayrityksessä oli aikaisemmin kehitetty uutta verkkokauppajärjestelmää varten tuotekategorioiden hallintaan tarkoitettu työkalu, joka sisälsi myös osan työssä vaadituista tuotteiden hinnoitteluominaisuuksista. Työn aluksi pohdittiin mahdollisuutta lisätä uudet hinnoitteluominaisuudet sekä tuotteiden suodatus tähän työkaluun, mutta suunnitelma osoittautui hankalaksi ratkaisuksi monestakin syystä.

Aiemmin kehitetyssä kategoriahallintatyökalussa ei ollut otettu erillisiä verkkokauppoja huomioon, ja kaikki sillä tehtävät toimenpiteet koskivat vain multitronic.fi -verkkokauppaa. Tämä tarkoitti sitä, että mahdollisuus verkkokaupan valintaan ja tuki muille verkkokaupoille olisi täytynyt lisätä sovellukseen jälkikäteen. Lisäksi sovelluksen käyttöliittymä oli rakennettu sillä olettamuksella, että kategorioiden hallinnan lisäksi sillä halutaan tehdä vain hinnoittelua, mikä olisi tehnyt suodatustoiminnallisuuden lisäämisestä sovellukseen hankalaa. Tekniseltä kannalta ongelma oli myös, että sovellus oli rakennettu käyttäen Yahoo! User Interface Library 2 -ohjelmistokehystä, joka toimii edelleen hyvin, mutta jota ei enää tueta. Merkittävien uusien ominaisuuksien kehittäminen tätä ohjelmistokehystä käyttäen ei siten olisi tulevaisuuden kannalta ollut hyvä ratkaisu. Koska tuotekategoriat ovat yhteisiä kaikille verkkokaupoille, mutta kategorioiden sisällön suodatus ja tuotteiden hinnoittelu on verkkokauppakohtaista, sovellus olisi näin tullut toimimaan samaan aikaan sekä yleisellä kaikkia verkkokauppoja koskevalla tasolla kategorianhallinnassa että verkkokauppakohtaisella tasolla tuotteiden suodatuksessa

ja hinnoittelussa. Näiden kahden eri tason hallinta samasta käyttöliittymästä ei olisi ollut kovin selkeää.

Edellä kuvatuista syistä johtuen katsottiin parhaaksi, että olemassa olevasta kategoriahallintatyökalusta poistetaan kaikki verkkokauppakohtainen hinnoittelutoiminnallisuus ja jätetään se kategorioiden hallintaa varten yleisen tason työkaluksi. Opinnäytetyössä kehitetään sen sijaan uusi työkalu, jossa on alusta lähtien otettu erilliset verkkokaupat huomioon ja johon toteutetaan verkkokauppakohtaiset suodatus- ja hinnoitteluominaisuudet sekä mahdollisuus muiden ominaisuuksien lisäämiseen myöhemmin. Uudessa työkalussa pyritään hyödyntämään aiemmin toteutettua hinnoittelutoiminnallisuutta niiltä osin kuin se on mahdollista.

3.3 Muuttuva tietokannan rakenne

Koska työtä tultiin tekemään osaksi kehitteillä olevaa uudistettua verkkokauppajärjestelmää oli tiedossa, että verkkokaupan tietokannan rakenteeseen voi tulla työn aikana merkittäviä muutoksia. Osittain nämä tulisivat olemaan odotettavissa olevia muutoksia mutta sellaisia, joita ei vielä työtä aloitettaessa oltu tehty. Lisäksi oli kuitenkin oletettavaa, että rakenteeseen tehdään myös erilaisia optimointeja sitä mukaa kun havaitaan asioita, joita voi tehdä tavalla tai toisella paremmin.

Nämä lisäsivät omalta osaltaan työn haastavuutta, kun oli tiedossa, että jotkin tietokantataulut, joihin kehitettävä työkalu tukeutuu, saattavat työn aikana vaihtaa paikkaa tai niiden sisäinen rakenne voi muuttua. Tämä pyrittiin ottamaan työkalun toteutuksessa huomioon niin, ettei muutoksiin reagoiminen työkalussa aiheuttaisi kohtuutonta vaivaa. Käytännössä tämä tarkoitti sitä, että suoria viittauksia tietokantatauluihin pyrittiin tekemään vain tilanteissa, joissa se oli ehdottoman tarpeellista. Tehokkaiden ja monimutkaisten tietokantahakujen onnistumiseksi suoria viittauksia ei kuitenkaan voitu aina välttää.

4 KÄYTETYT TEKNIIKAT JA OHJELMISTOT

Tässä kappaleessa kuvataan lyhyesti työssä käytetyt merkittävimmät tekniikat, kehitysympäristö ja ohjelmistot sekä niiden rooli opinnäytetyön kannalta. Osaa tekniikoista käsitellään syvemmin myöhemmissä osioissa, joissa kehitetyn web-sovelluksen toimintaa kuvataan tarkemmin.

4.1 Palvelinpään tekniikat

4.1.1 PHP

PHP on heikosti tyypitetty ohjelmointikieli /4/, jota käytetään erityisesti dynaamisten verkkosivujen rakentamiseen /5/. PHP tukee useita erilaisia ohjelmointiparadigmoja, joista työssä käytettiin olio-ohjelmoinnin lähestymistapaa /6/. PHP toimii työssä palvelinpään ohjelmointikielenä ja sillä suoritetaan kehitetyllä web-sovelluksella tehtävät toimenpiteet sekä kommunikointi tietokantapalvelimelle. PHP-kielellä on työssä rakennettu ohjelmointirajapinta eli API, jolle web-sovelluksesta voidaan lähettää erilaisia käskyjä ja pyyntöjä, ja joka palauttaa web-sovellukselle tietoa JSON-muodossa. PHP valittiin palvelinpään ohjelmointikieleksi erityisesti siitä syystä, että sillä on rakennettu lähes kaikki toimeksiantajayrityksen ohjelmistot ja sisäiset työkalut, ja näin ollen se mahdollisti sujuvimman integroinnin olemassa olevan järjestelmän kanssa.

4.1.2 PostgreSQL

PostgreSQL on tietokannan hallintajärjestelmä relaatiotietokannan pystyttämiseen ja hallintaan /7/. Relaatiotietokannassa tieto jaetaan nimettyihin tauluihin, jotka sisältävät sarakkeita ja rivejä tallennetulle tiedolle ja joiden välille luodaan yhteyksiä /8/. PostgreSQL-tietokannan kanssa kommunikoidaan SQL-kyselykielellä. Tietokantaan tallennettiin kaikki opinnäytetyössä kehitetyn web-sovelluksen käsittelemä muutettavissa oleva tieto koskien tuotteiden suodatusta ja hinnoittelua. Työssä käytettiin jo olemassa olevaa toimeksiantajayrityksen tietokantaa, jota laajennettiin web-sovelluksessa tarvittavilla tietokantatauluilla ja tallennetuilla proseduureilla.

4.1.3 PL/PgSQL

PL/PgSQL on PostgreSQL-tietokannassa käytettävä sisäinen ohjelmointikieli tallennetuille proseduureille. PL/PgSQL mahdollistaa suoraan tietokantaan tallennettavien ja tietokannasta ajettavien funktioiden kehittämisen. Nämä funktiot voivat edelleen sisältää ohjelmointikielille tyypillisiä muuttujia ja rakenteita, kuten ehtolauseita ja silmukoita. /9/ Näin tiettyjä tietokantaan liittyviä prosesseja voidaan ajaa suoraan tietokantapalvelimella, ilman tarvetta siirtää tietoa käsiteltäväksi palvelimen ulkopuolelle. Opinnäytetyössä PL/PgSQL-funktioita käytettiin web-sovelluksella määriteltyjen suodatus- ja hinnoittelusääntöjen täytäntöönpanoon.

4.1.4 Nested set

Nested set on menetelmä puumaisen hierarkkisen tiedon tallentamiseen relaatiotietokantaan. Nested set -menetelmässä hierarkkinen tieto tallennetaan tietokantaan antamalla kullekin hierarkiassa olevalle noodille kaksi lukuarvoa siten, että tiedon hakemiseen tarvittavien SQL-kyselyiden määrä vähenee huomattavasti. Esimerkiksi tiettyyn hierarkiassa olevaan noodiin johtavan polun hakeminen tietokannasta onnistuu yhdellä SQL-kyselyllä sen sijaan, että kysely pitäisi suorittaa tavalla tai toisella rekursiivisesti. Toimeksiantajayrityksen järjestelmässä hallittavien tuotteiden kategoriapuu on tallennettu tietokantaan nested set -menetelmää käyttäen, joten tämän menetelmän hyväksikäyttäminen oli tärkeässä roolissa opinnäytetyössä kehitetyllä web-sovelluksella hallittavien hierarkkisesti periytyvien suodatus- ja hinnoittelusääntöjen kanssa.

4.2 Asiakaspään tekniikat

4.2.1 HTML, CSS ja Handlebars

HTML on kuvauskieli, jolla voidaan kuvata hypertekstiä. HTML määrittelee verkkoselaimella esitettävän sivun rakenteen, ja se muodostuu erilaisista elementeistä, jotka voivat olla rakenteessa peräkkäin tai sisäkkäin, ja joilla voi olla edelleen esimerkiksi tekstisisältöä. Työssä kehitetyn web-sovelluksen käyttöliittymän rakenne on määritelty HTML-dokumenttina. Käyttöliittymän ulkoasu, eli tapa jolla HTML-dokumentti esitetään, /11/ on määritelty CSS -tyylitiedostoja käyttäen.

Handlebars on JavaScript-kielellä käytettävä asiakaspään sivupohjajärjestelmä, jolla HTML-dokumenttiin tai siihen liitettävään sektioon voidaan määritellä osia, joiden sisältöä täydennetään dynaamisesti halutulla tiedolla ennen dokumentin esittämistä. /12/ Tämä mahdollistaa työssä kehitetyssä web-sovelluksessa sen, että palvelinpäässä ei tarvitse ottaa millään tavalla kantaa siihen, kuinka tietoa asiakaspäässä esitetään. Palvelinpäästä ei näin tarvitse palauttaa HTML-dokumentteja, vaan pelkästään dataa, joka asiakaspäässä voidaan esittää halutulla tavalla.

4.2.2 JavaScript ja jQuery

JavaScript on heikosti tyypitetty ohjelmointikieli /13/, jota käytetään erityisesti asiakaspään ohjelmointikielenä verkkoselaimissa. JavaScript tukee useita erilaisia ohjelmointiparadigmoja /14/, joista opinnäytetyössä pyrittiin käyttämään lähinnä olio-ohjelmointia lähestymistapana. JavaScript toimii työssä asiakaspään ohjelmointikielenä verkkoselaimessa, ja sillä rakennetaan käyttäjälle näkyvän käyttöliittymän interaktiivinen toiminnallisuus sekä kommunikaatio palvelinpään ohjelmointirajapintaan.

JQuery on JavaScript-kirjasto, joka helpottaa HTML-dokumenttien sisällön muokkaamista ja käsittelyä DOM-rajapinnan kautta. JQuery yksinkertaistaa myös JavaScript-ohjelmakoodin kirjoittamista tarjoamalla funktioiden kautta rakenteita ja muita toiminnallisuuksia, joita verkkoselaimien JavaScript-tulkit eivät valmiiksi sisällä. /15/ Lisäksi se helpottaa AJAX-pyyntöjen hallintaa /16/. JQuery valittiin vastaavanlaisten kirjastojen joukosta, koska se on erittäin hyvin tuettu ja vakaa kirjasto, joka oli jo laajasti käytössä toimeksiantajayrityksen muissa työkaluissa.

4.2.3 AJAX ja JSON

AJAX on JavaScript-kielellä käytettävä menetelmä, jolla verkkoselaimesta voidaan lähettää palvelimelle HTTP-pyyntöjä ilman, että koko verkkosivua tarvitsee ladata uudelleen. Palvelin vastaa lähetettäviin pyyntöihin palauttamalla dataa, jota voidaan käsitellä edelleen halutulla tavalla JavaScript-ohjelmakoodissa - esittämällä se sivulla tai reagoimalla siihen jollain muulla tavalla. AJAX mahdollistaa näin viestien lähettämisen ja vastaanottamisen palvelimelta web-sovelluksessa

taustalla niin, ettei sovelluksen käyttö keskeydy /17/. Työssä AJAX-menetelmää käytetään web-sovelluksen ja palvelimen väliseen kommunikointiin.

JSON on erityisesti tiedonvälitykseen käytetty tiedostomuoto /18/, jota voidaan käyttää muun muassa AJAX-pyyntöjen tiedostomuotona. JSON valittiin työssä kehitetyn palvelinpään ohjelmointirajapinnan palauttamaksi tiedostomuodoksi, koska se on yksinkertainen ja sen käsittely JavaScript-kielellä on helppoa.

4.3 Kehitysympäristö ja ohjelmistot

4.3.1 Netbeans

Netbeans on ohjelmointiympäristö, joka tukee useita eri ohjelmointikieliä ja kuvauskieliä. Netbeans hallitsee muun muassa ohjelmakoodin syntaksin värityksen, reaaliaikaisen syntaksivirheiden havaitsemisen, ohjelmointikielestä riippuen virheenkorjauksen virheenjäljittäjän avulla /19/ sekä kommunikoinnin useiden eri versionhallintajärjestelmien kanssa /20/. Työssä Netbeansia käytettiin PHP- ja JavaScript -ohjelmointikielien, HTML-kuvauskielen ja CSS-tyyliohjeiden kanssa. Näin se kattoi kaikki työssä tarvittavat ohjelmointitarpeet tietokantaproseduureja lukuun ottamatta. Netbeansin valintaa ohjelmointiympäristöksi puolusti erityisesti sen hyvä tuki työssä tärkeässä asemassa olevalle PHP-ohjelmointikielelle.

4.3.2 EMS SQL Manager for PostgreSQL

EMS SQL Manager for PostgreSQL on graafinen ylläpitotyökalu PostgreSQL-tietokantojen kehittämiseen ja hallintaan. Ohjelmalla voidaan ajaa SQL-kyselyitä tietokannassa, ja se hallitsee muun muassa SQL-kielen syntaksin värityksen ja syntaksin oikeellisuuden tarkastuksen. Lisäksi sen avulla voidaan helposti suunnitella ja muuttaa tietokannan rakennetta ja tietokannan tauluissa olevaa tietoa ilman monimutkaisten SQL-lauseiden kirjoittamista /21/. Työssä EMS SQL Manager for PostgreSQL oli käytössä lähinnä alustana SQL-kyselyiden sekä tietokantaan tallennettujen proseduurien kehittämiselle ja testaamiselle. Ohjelma valittiin työhön erityisesti sillä perusteella, että se oli toimeksiantajayrityksessä jo ennestään tuttu ja pitkään käytetty.

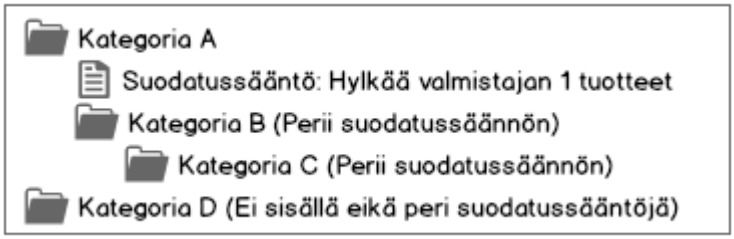
5 SOVELLUKSEN MÄÄRITTELY JA SUUNNITTELU

Tässä kappaleessa kuvataan toteutusta edeltänyttä määrittelyä ja suunnittelua.

5.1 Vaatimusmäärittely

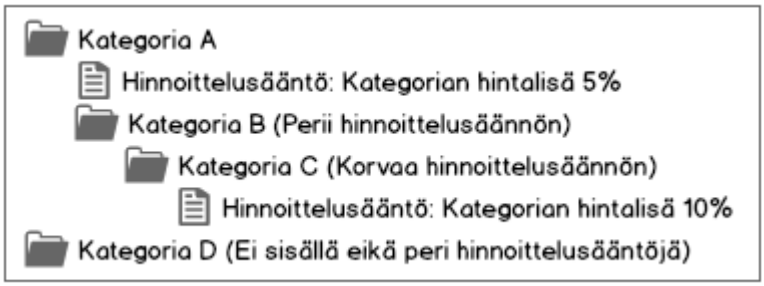
Kohdassa 3 kuvatun sovelluksen taustan perusteella lähdettiin toimeksiantajayrityksen kanssa tekemään sovelluksen vaatimusmäärittelyä. Peruslähtökohtana oli, että kaikille verkkokaupoille täytyy voida tehdä tuotteiden suodatus- ja hinnoittelumäärityksiä tasavertaisesti. Lisäksi molempien toimenpiteiden pitäisi onnistua saman sovelluksen kautta. Tämä siksi, ettei jokaisessa tuotteiden hallintaan liittyvässä toimenpiteessä tarvitse siirtyä käyttämään eri sovelluksia, joita oli jo useita.

Tuotteiden suodatuksen haluttiin onnistuvan kategoriakohtaisesti niin, että kokonaisen kategorian kaikki tuotteet voidaan kerralla sisällyttää valitun verkkokaupan tuotevalikoimaan, tai hylätä. Sisällytettyjen kategorioiden tuotevalikoimaa haluttiin edelleen rajata tuotteen valmistajan ja painon mukaan niin, että esimerkiksi tiettyjen valmistajien tuotteet voidaan pudottaa kategoriasta pois, tai vaihtoehtoisesti ottaa pelkästään mukaan. Lisäksi näiden kategorioihin liitettyjen lisämääritysten haluttiin periytyvän kategoriahierarkiassa alakategorioille niin, ettei jokaiselle kategorialle tarvitse tehdä määrityksiä erikseen. Alakategorioissa tehdyillä omilla määrityksillä täytyi kuitenkin voida syrjäyttää perittyjä määrityksiä. Viimeisenä tuotesuodatuksen osana yksittäisiä tuotteita haluttiin sisällyttää tai hylätä kaikista muista suodatussäännöistä riippumatta. Kuviossa 2 on havainnollistettu yksinkertaisella esimerkillä sitä kuinka suodatussääntöjen haluttiin periytyvän kategoriahierarkiassa.



Kuvio 2. Suodatussääntöjen periytyminen kategoriahierarkiassa

Tuotteiden hinnoittelun haluttiin onnistuvan myös kategoriakohtaisesti niin, että kategorioille voidaan määritellä prosentuaalinen hintalisä, jonka perusteella kategoriassa olevien tuotteiden hintaa kasvatetaan. Kategorioihin haluttiin myös määritellä tarvittaessa tietyille valmistajille oma prosentuaalinen hintalisä. Näin ollen kategorialle määritettyä hintalisää käytetään kaikille kategoriassa oleville tuotteille, paitsi jos tuotteen valmistajalle on määritelty kategoriassa erikseen hintalisä. Samoin kuin suodatusmääritysten, myös hinnoittelumääritysten tuli periytyä kategoriahierarkiassa alakategorioille, paitsi jos ne syrjäytetään kategorian omilla hinnoittelumäärityksillä. Kuviossa 3 on havainnollistettu yksinkertaistettuna hinnoittelusääntöjen periytymistä kategoriahierarkiassa.



Kuvio 3. Hinnoittelusääntöjen periytyminen kategoriahierarkiassa

Lisäksi sovelluksen tuli mahdollistaa yksittäisten tuotteiden erikoishintojen määrittely taulukkomuotoisen tuotelistauksen kautta niin, että hintoja voidaan muuttaa suoraan taulukon soluun. Erikoishintoja voivat olla esimerkiksi tiettyyn arvoon lukittu hinta tai aikarajan sisältävä kampanjahinta. Tuotteiden erikoishintojen tekninen toteutus oli jo olemassa, ja normaalisti niitä muokataan verkkokaupassa tuotesivujen kautta. Tässä työssä sovelluksen tuli tarjota vain vaihtoehtoinen tapa erikoishintojen muokkaamiseen. Kuviossa 4 on havainnollistettu yksinkertaisella esimerkillä tuotekohtaisten erikoishintojen hallintaa.

Tuotenumero	Tuotteen nimi	Lukittu hinta (€)	Kampanjahinta (€)	Alkupvm	Loppupvm
1	Tuote 1	290			
2	Tuote 2		449	2014-09-01	2014-09-07

Kuvio 4. Tuotekohtainen taulukkomuotoinen hinnoittelu

Suodatus- ja hinnoittelusäännöt itsessään ovat vain määrityksiä, jotka tulevat voimaan vasta kun niitä käyttävät prosessit ajetaan. Pelkkien sääntöjen lisäksi tarvittiinkin jotkin automaattisesti ja tarvittaessa myös manuaalisesti ajettavat prosessit, joilla säännöt voidaan panna täytäntöön. Suodatuksen kannalta tämä tarkoittaa sitä, että haluttiin prosessi, joka käy tietyin väliajoin tai tiettyjen tapahtumien yhteydessä koko perustuotetaulun läpi, ja suodattaa tuotteet verkkokauppakohtaisiin tuotetauluihin senhetkisiä verkkokauppojen suodatussääntöjä käyttäen. Verkkokauppakohtaisten hinnoittelusääntöjen tulkinta taas haluttiin osaksi sitä prosessia, joka määrää tuotteen lopullisen asiakkaalle esitettävän hinnan eri tilanteissa.

Edellä kuvattuja suodatuksen ja hinnoittelun vaatimusmäärityksiä sekä muita sovellukselle asetettuja vaatimuksia on esitetty taulukossa 1. Tässä vaatimukset on QFD-menetelmän mukaisesti jaoteltu kolmeen ryhmään niiden prioriteetin perusteella. Ensimmäisen ryhmän ominaisuudet täytyy olla toteutettuna, jotta työn voidaan katsoa olevan valmis. Toisessa ryhmässä on ominaisuudet, jotka ovat odotettuja, mutta eivät välttämättömiä. Kolmanteen ryhmään on kerätty hyödyllisiä lisäominaisuuksia, jotka parantavat sovelluksen käyttömahdollisuuksia ja laajennettavuutta tulevaisuudessa. Lisäksi tähän viimeiseen ryhmään on kerätty erityisesti sellaisia ominaisuuksia, joiden suuntaan sisäisten työkalujen on käytettävyyden osalta toivottu yrityksessä kehittyvän.

Taulukko 1: Projektin QFD-taulukko

Täytyy olla

- Verkkokaupan valinta: Sovelluksella tehtävät määritykset koskevat vain valittua verkkokauppaa. Jokaiselle verkkokaupalle täytyy voida tehdä tasavertaisesti omat määritykset.
- Määritysten kategorialähtöisyys ja periytyminen: Tehtävät määritykset (tuotekohtaisia määrityksiä lukuun ottamatta) liitetään haluttuun tuotekategoriaan, josta ne periytyvät alakategorioille. Alakategorioissa mahdollisuus perittyjen määritysten syrjäyttämiseen tai niihin lisäämiseen uusilla määrityksillä.
- Tuotteiden suodatus: Mahdollisuus tuotteiden suodatussääntöjen tekemiseen tuotteen kategorian, valmistajan, painon ja tuotekoodin perusteella. Suodatusmäärityksiä täytyy pystyä tekemään (tilanteesta riippuen) tavoilla:
  - Sisällytä sääntöä vastaavat tuotteet
  - Hylkää sääntöä vastaavat tuotteet
  - Sisällytä vain sääntöä vastaavat tuotteet
- Tuotteiden hinnoittelu: Mahdollisuus tuotteiden prosentuaalisen hintalisän määrittämiseen kategorian ja valmistajan perusteella. Valmistajakohtainen hintalisä pitää voida määritellä hierarkiassa pakotetusti periytyväksi. Tuotekohtainen erikoishintoja täytyy lisäksi voida hallita tuotekohtaisesti.
- Määritysten tarkastelu: Tehtyjä määrityksiä voidaan tarkastella sekä kokonaisuutena että kategoriakohtaisesti.
- Taustaprosessit: Automaattisesti ajettavat ja tietyissä tilanteissa kutsuttavat taustaprosessit hinnoittelu- ja suodatusmääritysten täytäntöönpanoon.

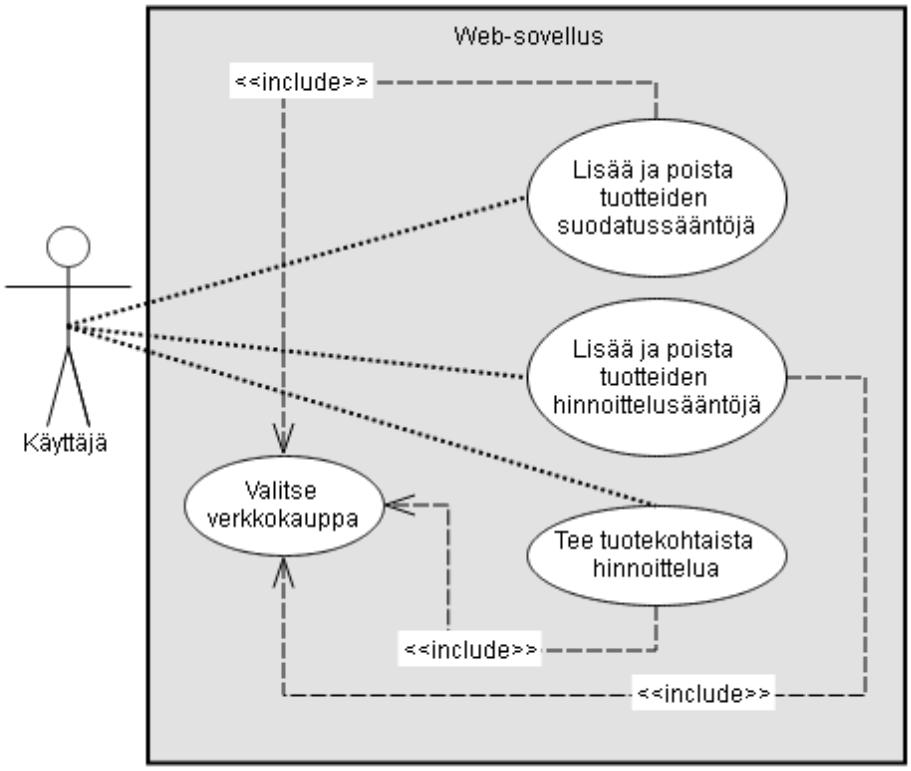
Pitäisi olla

- verkkokaupan valinta sekä suodatus- ja hinnoittelutyökalut samassa web-sovelluksessa.
- web-sovelluksen toiminta yleisimpien selainten nykyaikaisilla versioilla. Koska työkalu tulee yritykseen sisäiseen käyttöön, vanhojen selainversioiden tukeminen ei ole oleellista.
- selkeä ja helppokäyttöinen käyttöliittymä. Käyttäjälle tulee esittää selvästi mitkä määritykset ovat voimassa missäkin kategoriassa väärinymmärrysten välttämiseksi.

Hyvä olla

- mahdollisuus sovelluksessa käsiteltävien arvojen muuttamiseen paikoillaan ilman erillisiin muokkausnäkymiin siirtymistä, ja sovelluksen toiminta ilman koko sivun uudelleenlatausta.
- sovelluksen käyttömahdollisuus näppäimistöltä
- sovelluksen suorittamat operaatiot käytettävissä muista työkaluista.
- mahdollisuus lisätä sovellukseen uusia kategorioihin liittyviä työkaluja myöhemmin niin, ettei sovellus ole liian sidottu yhden työkalun käyttöön.

Web-sovelluksen keskeisimpiä toimintoja on havainnollistettu käyttötapauskaaviona kuviossa 5.



Kuvio 5. Käyttötapauskaavio web-sovellukselle

5.2 Suunnittelu

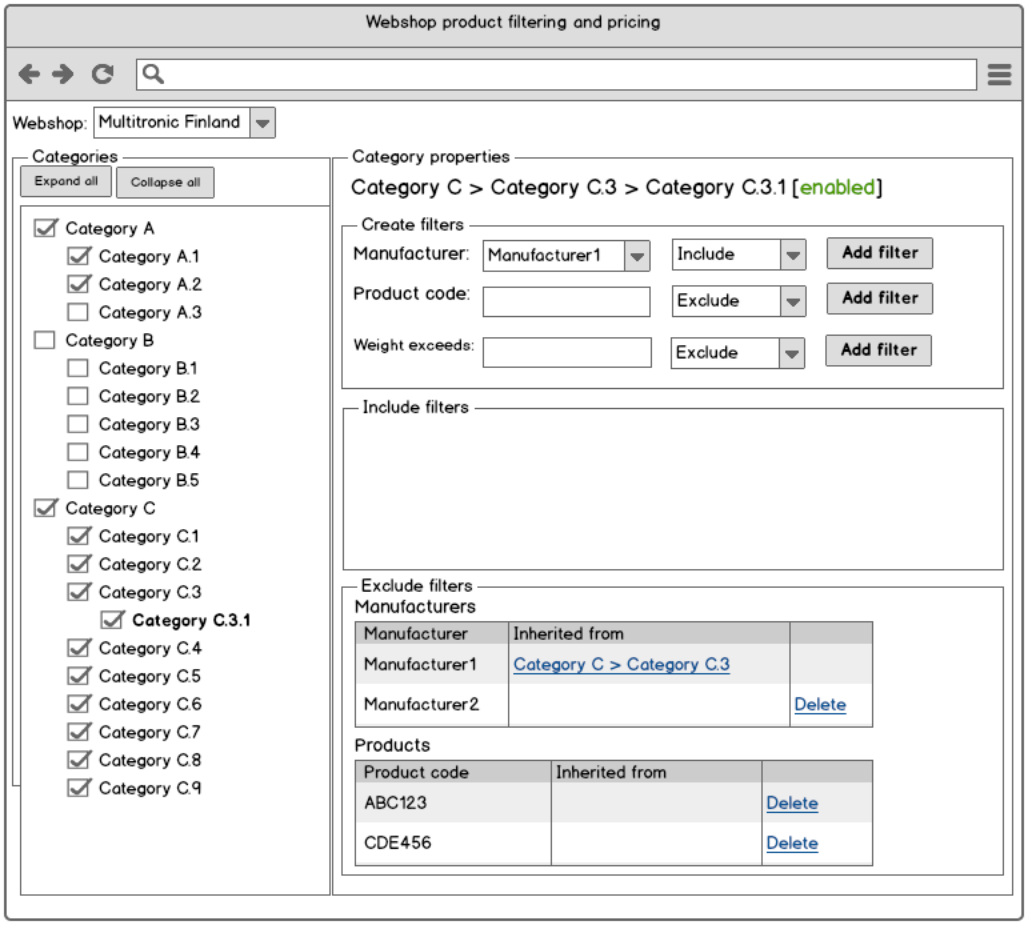
Tässä kappaleessa kuvataan ennen toteutusvaiheen aloittamista tehtyä sovelluksen suunnittelua. Suunnitteluvaiheessa pyrittiin miettimään sovelluksen eri osien rakennetta, toimintaa ja käytettävyyttä ottamatta kantaa kaikkien tekniikoiden valintoihin ja komponenttien yksityiskohtiin. Tekniikoita valittiin toteutusvaiheessa vastaamaan suunnittelussa ilmenneitä tarpeita.

5.2.1 Asiakassovellus ja käyttöliittymä

Sovelluksen käyttöliittymän ensimmäinen oleellinen asia oli käsiteltävän verkkokaupan valinta, vaihtaminen ja esittäminen käyttäjälle. Tämä toiminto suunniteltiin sovelluksen yläreunaan pudotusvalikkoon, missä se on jatkuvasti käyttäjälle näkyvissä ja mistä sen vaihtaminen on helppoa.

Koska useimpien sovelluksella tehtävien määritysten lähtökohtana toimivat tuotekategoriat, kategoriat päätettiin esittää jatkuvasti näkyvissä olevana puurakenteena. Puurakenne haluttiin interaktiiviseksi siten, että sen eri osia voidaan avata ja sulkea tarpeen mukaan. Siitä täytyi voida myös valita jokin kategorioista, jotta kategoriaan liittyviä tietoja päästään tarkastelemaan ja muokkaamaan. Koska tuotesuodatuksen yhtenä tasona oli suodattaminen kategorian perusteella, kategoriapuu tarjosi hyvän paikan tämän suodatuksen tekemiseen. Kategoriapuuhun suunniteltiin valintaruudut, joilla kokonaisten kategorioiden sisältöä voidaan suodattaa.

Viimeisenä suunniteltiin se käyttöliittymän osio, jossa puurakenteesta valitun kategorian tietoja voidaan muokata. Tämän osion sisään täytyi saada erilliset näkymät, joista yksi mahdollistaa tuotteiden suodatussääntöjen hallinnan, toinen hinnoittelusääntöjen hallinnan ja kolmas taulukkomuotoisen tuotekohtaisen hinnoittelun. Sääntönäkymät suunniteltiin listauksina ja pudotusvalikkoina, joihin yhdistetään JavaScript-ohjelmakoodia, mutta tuotekohtaiseen hinnoittelunäkymään haluttiin käyttöön jokin interaktiivinen JavaScript-taulukkokirjasto. Suunniteltua käyttöliittymää on esitetty kuviossa 6 olevassa hahmotelmassa. Hahmotelmassa näytetään kategoriapuusta valitun kategorian tuotesuodatustietoja.



Kuvio 6. Käyttöliittymän hahmotelma tuotesuodatusnäkymässä

Suunnitellun käyttöliittymän toteutus siten, että koko sivu ladataan uudelleen jokaisen käyttöliittymässä tehtävän toimenpiteen yhteydessä, olisi tehnyt sovelluksen käytöstä hyvin kankeaa. Suunnittelussa päädyttiinkin hyvin varhain siihen tulokseen että, käyttöliittymän toteutus täytyy tehdä yhden sivun sovelluksena (SPA) siten, että sivun eri osien sisältöä päivitetään dynaamisesti JavaScript-kielellä. Tämä tarkoitti myös sitä, että sovelluksessa tullaan käyttämään AJAX-menetelmää tiedon hakemiseen palvelimelta. Aluksi ajateltiin, että AJAX-pyyntöjen tuloksena palautetaan pääasiassa valmiiksi muotoiltua HTML-koodia, joka esitetään sellaisenaan käyttöliittymässä. Myöhemmin päädyttiin kuitenkin siihen tulokseen, että AJAX-pyyntöjen on parempi palauttaa dataa ja HTML-koodi tulisi luoda tämän datan perusteella vasta asiakassovelluksessa. Sovelluksen voitiin näin katsoa

jakautuvan kahteen erilliseen osaan - käyttöliittymästä huolehtivaan asiakassovellukseen sekä tietoa käsittelevään ja palauttavaan palvelinsovellukseen.

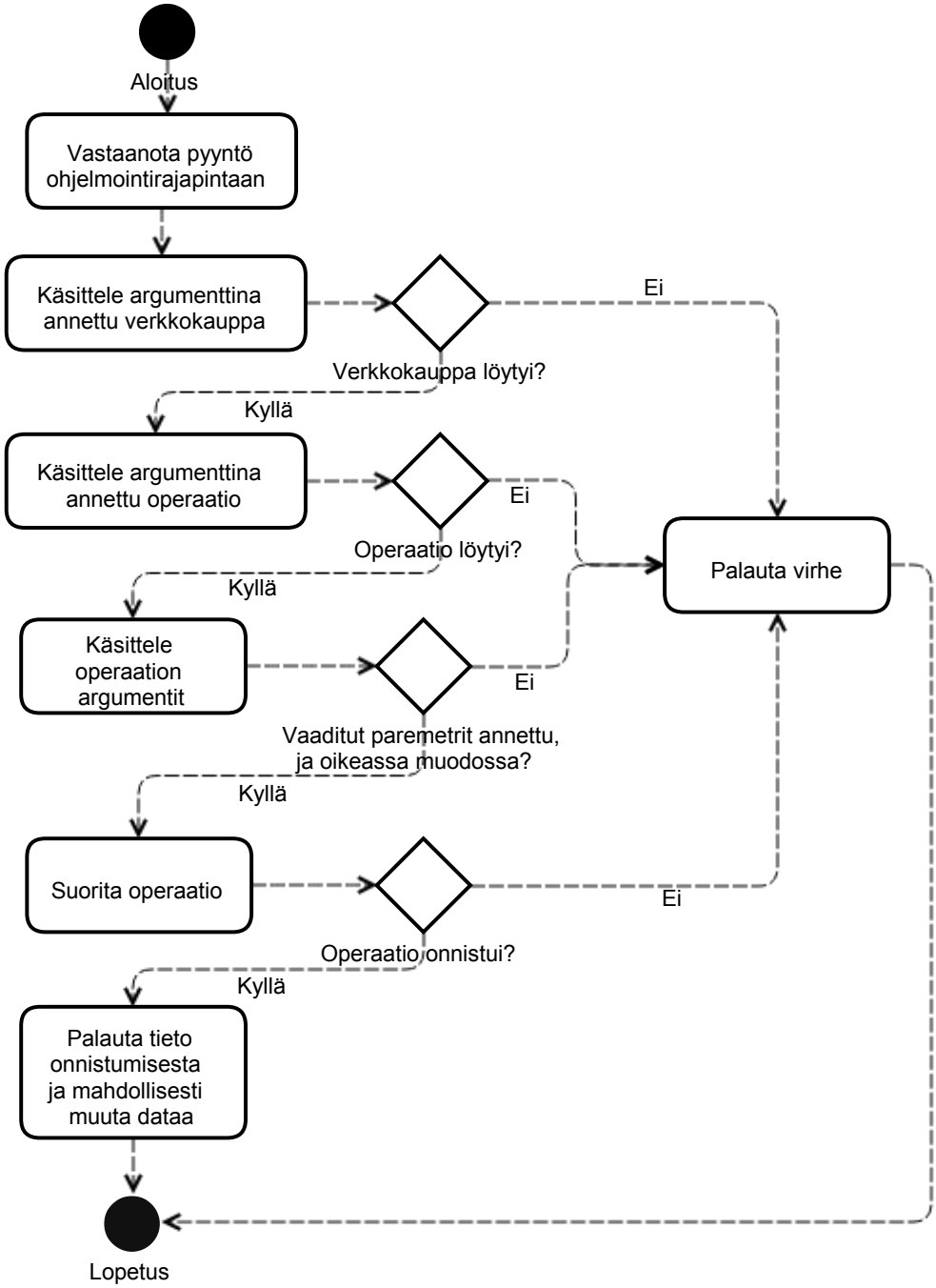
5.2.2 Palvelinsovellus

Palvelinsovelluksen täytyy vastaanottaa asiakassovellukselta pyyntöjä ja käskyjä, joiden perusteella se palauttaa asiakassovellukselle tietoa ja tarvittaessa muokkaa tuotteiden suodatus- ja hinnoittelumäärityksiä sekä tuotekohtaisia hintoja. Palvelinsovelluksen tuli näin tarjota asiakassovellukselle jonkinlainen ohjelmointirajapinta. Palvelinsovellus tuli myös kehittää PHP-ohjelmointikielellä, jotta se saatiin osaksi toimeksiantajayrityksen verkkokauppajärjestelmää. Verkkokauppajärjestelmä hoitaa sisäänkirjautumiseen ja käyttöoikeuksien hallintaan liittyvät toimenpiteet, joten nämä asiat voitiin sivuuttaa ohjelmointirajapinnan suunnittelussa.

Ensimmäinen tärkeä asia, joka palvelinsovelluksen täytyy tietää, on minkä verkkokaupan tietoja ollaan käsittelemässä. Tämän tiedon täytyi siis olla osa kaikkia asiakassovellukselta palvelinsovellukselle lähetettäviä pyyntöjä. Tämän jälkeen haluttiin tietää operaatio, joka palvelinsovelluksella pitäisi suorittaa, ja edelleen argumentit joita operaatiolle annetaan. Näiden tietojen avulla palvelinsovellus pystyy tekemään tarvittavat kyselyt ja muutokset tietokantaan sekä palauttamaan tietoa asiakassovellukselle. Näistä muodostui ohjelmointirajapinnan tarpeet pääpiirteissään.

Palvelinsovelluksen kaikkia yksittäisiä operaatioita ja niiden yksityiskohtaista sisältöä ei alettu vielä suunnitteluvaiheessa tarkasti määrittelemään, sillä toteutusvaiheessa ja asioita käytännössä testaamalla tarpeet vasta selviäisivät kunnolla. Ajatuksena oli kuitenkin, että edellä kuvatut vaaditut parametrit *verkkokauppa* ja *operaatio*, kulkevat HTTP GET -parametreina rajapintaan tulevissa pyynnöissä. Näille on lukitut parametrien nimet, joita ei voi käyttää operaatioiden sisäisten parametrien niminä. Operaatioiden parametrit kulkevat joko HTTP GET tai HTTP POST -parametreina riippuen siitä ollaanko hakemassa vai muuttamassa tietoa. Operaatioiden nimissä käytetään *get*, *add*, *delete* ja *update* -avainsanoja kuvaamaan sitä tullaanko operaatiolla hakemaan, lisäämään, poistamaan vai muutta-

maan tietoa. Näin saadaan esimerkiksi tuotteiden suodatussääntöjen hallintaan operaatiot *getFilters*, *addFilter*, *deleteFilter* ja *updateFilter*. Palvelinpään ohjelman toimintalogiikkaa on havainnollistettu kuviossa 7.



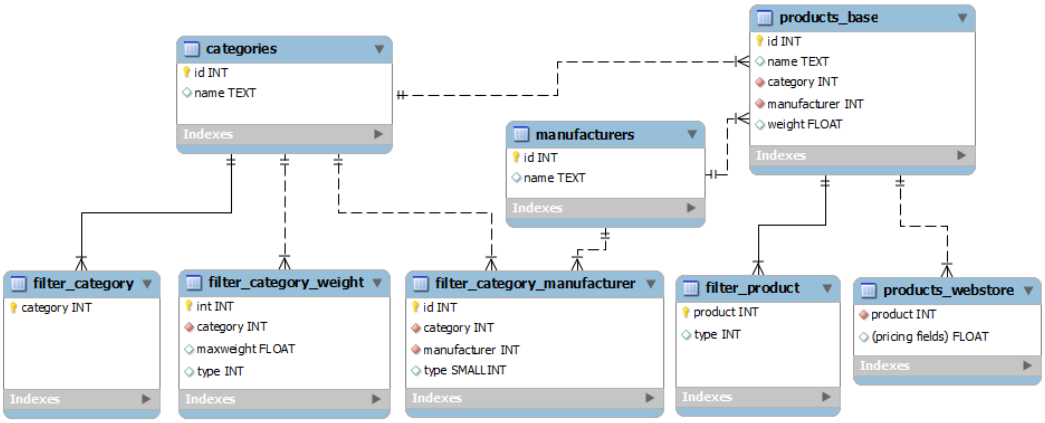
Kuvio 7. Palvelinsovelluksen toimintalogiikka

5.2.3 Tietokantataulut

Tietokantaan tarvittiin jokaiselle verkkokaupalle identtiset taulut tuotteiden suodatus- ja hinnoittelusäännöille. Tuotekohtaista hinnoittelua varten ei tarvinnut tämän työn puitteissa tehdä tietokantamuutoksia, sillä tuotekohtainen hinnoittelu määritellään kunkin verkkokaupan tuotetauluihin, joiden rakenne oli valmiina.

Tuotteiden suodatuksen vaatimukset olivat, että sääntöjä täytyy voida tehdä kategorian perusteella niin, että kategorian tuotteet joko otetaan mukaan tai hylätään ja kategorian tuotevalikoimaa täytyy voida edelleen rajata tuotteen valmistajan ja painon perusteella. Suodatus valmistajan ja painon perusteella on siis riippuvainen kategorian suodatuksesta. Kategoriaan liitetyt säännöt periytyvät hierarkiassa alakategorioille. Tuotekohtainen suodatus sen sijaan on muista säännöistä erillään, ja sillä joko lisätään yksittäisiä tuotteita tuotevalikoimaan, tai poistetaan niitä.

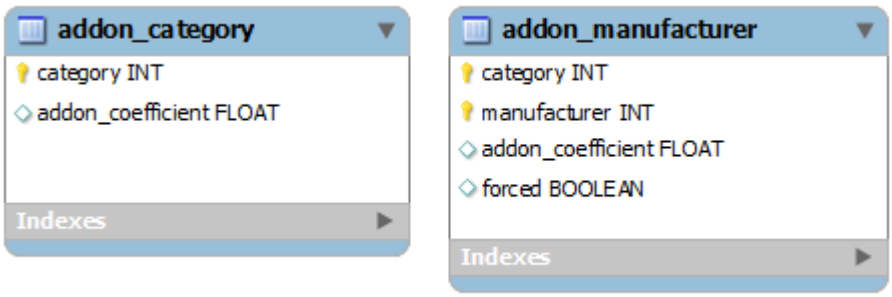
Näiden tietojen perusteella saatiin suunniteltua tietokantataulut, joita tarvittiin suodatukselle neljä. Kuviossa 8 on kuvattu suodatustauluja, niiden yhteyksiä yleisen tason kategoria-, valmistaja-, ja tuotetauluihin yksinkertaistettuna, sekä suodatuksen lopputuloksena saatavaa verkkokauppakohtaista tuotetaulua. Taulun *filter_category* perusteella kategorian tuotteet otetaan mukaan, mikäli kategorian tunniste on taulussa - muutoin tuotteet hylätään. Muissa suodatustauluissa esiintyvä *type* määrittelee, onko suodatussääntö tyyppiä *ota mukaan* (include), *hylkää* (exclude) vai *ota pelkästään mukaan* (include exclusively).



Kuvio 8. EER-kaavio tuotesuodatuksen tietokantatauluista

Tuotteiden hinnoittelusääntöjen vaatimuksena oli, että kategorioille voidaan määritellä sen tuotteita koskeva prosentuaalinen hintalisä. Lisäksi täytyi voida erikseen määritellä kategoriassa oleville tietyn valmistajan tuotteille erilainen prosentuaalinen hintalisä, joka menee prioriteetissa kategorian hintalisän edelle. Säännöt myös periytyvät kategoriahierarkiassa alakategorioille, ellei alakategorialle itselleen ole tehty hinnoittelumäärityksiä.

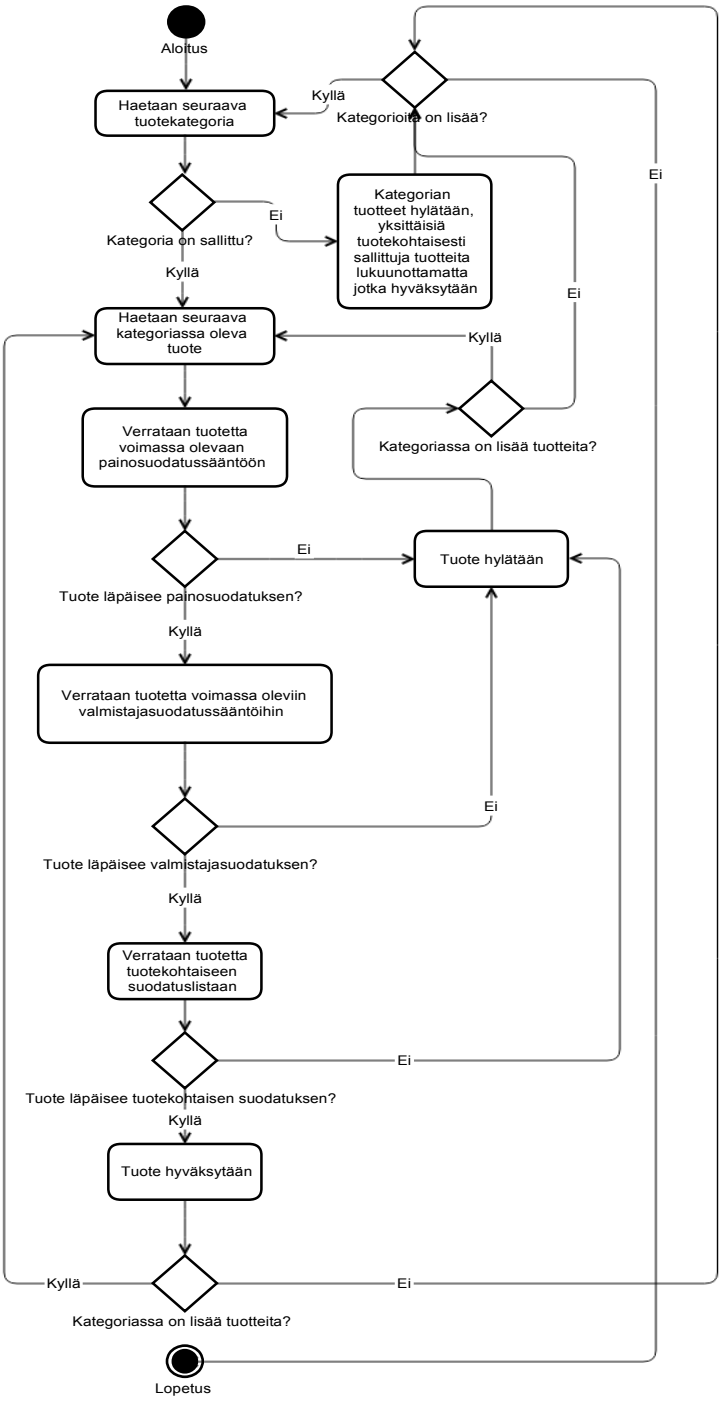
Kuten kohdassa 3.2 mainittiin, kategorioiden hinnoittelutoiminnallisuutta oli jo tehty osana aiemmin kehitettyä kategorioiden hallintaan tarkoitettua työkalua. Tämän työkalun rakentamisen yhteydessä oli luotu tietokantataulut hinnoittelusäännöille. Näitä voitiin siten käyttää työssä lähes suoraan, kunhan ne luotiin jokaiselle verkkokaupalle erikseen. Ainoa merkittävä muutos tauluihin oli, että valmistajakohtainen hintalisä täytyi voida halutessa määritellä pakotetuksi, jolloin se periytyy kategoriahierarkiassa alakategorioille, vaikkei kategorioilla itsellään olisi määriteltynä hinnoittelusääntöjä, jotka normaalisti kumoaisivat perityn säännön. Kuviossa 9 on kuvattu hinnoittelusääntöjen tietokantatauluja.



Kuvio 9. EER-kaavio tuotteiden hinnoittelusääntötauluista

5.2.4 Tallennetut tietokantaproseduurit

Tallennettujen tietokantaproseduurien tarkoituksena on suorittaa varsinainen tuotteiden suodatus ja hinnoittelu. Suodatuksen tietokantaproseduuri suunniteltiin sellaiseksi, että se käy jokaisen tuotekategorian läpi, laskee mitkä suodatussäännöt kategoriassa ovat voimassa, ja suodattaa kategorian tuotteet. Lopuksi tulokseen lisätään ja siitä poistetaan yksittäiset tuotteet, joille on määritelty suodatussääntö. Suodatuksen logiikkaa on havainnollistettu kuviossa 10.



Kuvio 10. Tuotteiden suodatusprosessin toimintalogiikka

Samoin kuin hinnoittelusääntöjen tietokantataulut, myös hinnoittelun tietokanta-proseduuri saatiin lähes sellaisenaan kohdassa 3.2 kuvatusta kategoriatyökalusta. Hinnoittelun tietokantaproseduuri ei siten edellyttänyt suunnittelua etukäteen.

6 SOVELLUKSEN TOTEUTUS

Tässä kappaleessa kuvataan työn tekninen toteutus. Toteutus jakautuu suunnitelman mukaisesti kolmeen osaan, joita ovat verkkoselaimella käytettävä asiakassovellus, palvelinsovellus, jolle asiakassovelluksesta lähetetään pyyntöjä ja käskyjä, sekä automaattisesti ajettavat tallennetut tietokantaproseduurit.

6.1 Asiakassovellus

Asiakassovelluksen tehtävänä on tarjota käyttöliittymä tuotteiden suodatukselle ja hinnoittelulle, ja kommunikoida AJAX-pyynnöillä palvelinsovelluksen kanssa.

6.1.1 Komponenttien valinta

Asiakassovelluksen toteutus aloitettiin valitsemalla komponentit joilla suunnitelman mukainen käyttöliittymä voidaan parhaiten toteuttaa. Tavoitteena oli valita sellaiset komponentit, jotka ovat vakaita ja hyvin tuettuja sekä ominaisuuksiltaan monipuolisia. Komponentteja joihin ei voi tukeutua vuosiksi eteenpäin pyrittiin välttämään.

Sekä suunniteltu kategoriapuu että taulukkomuotoinen tuotekohtainen hinnoittelu olivat käyttöliittymän osia, joihin haluttiin valmiit JavaScript-kirjastot. Erilaisia vaihtoehtoja löytyi useita, mutta näistä lupaavimmiksi valikoituivat hyvin nopeasti jsTree puurakenteeksi ja jqGrid interaktiiviseksi taulukkorakenteeksi. Asiakassovellus päätettiin rakentaa jQueryn päälle, koska sekä jsTree että jqGrid olivat jQuery-liitännäisiä, ja jQuery oli yrityksessä laajasti käytössä muutenkin. Käyttöliittymäelementtikirjastoksi valittiin jQuery UI, ja käyttöliittymän näkymien sivupohjajärjestelmäksi Handlebars.

Erilaisten JavaScript-ohjelmistokehysten käyttöä harkittiin, mutta näistä ei löydetty sopivaa vaihtoehtoa, koska oltiin rakentamassa enemmänkin työpöytäsovelluksen tyyppistä ohjelmaa kuin verkkosivua. Koska aiemmin kehitetyissä työkaluissa oli käytetty YUI2-kehystä, luonnollinen valinta olisi ollut uudempi YUI3. Työn aikana tämän kehyksen kehitys lopetettiin kuitenkin kokonaan. Toinen harkittu vaihtoehto oli AngularJS, josta taas ilmoitettiin työn aikana suunnitelmat uudesta

versiosta, joka on aiempien versioiden kanssa epäyhteensopiva. Asiakassovelluksen tulevaisuuden kannalta kumpikin näistä olisi ollut ongelmallinen valinta, ja tähän nähden vakaaseen jQueryyn tukeutuminen osoittautui hyväksi valinnaksi.

6.1.2 Sovelluksen rakenne

Asiakassovelluksen JavaScript-ohjelmakoodin pääosat on rakennettu käyttäen omaan nimiavaruuteen määriteltyjä olioliteraaleja, jotka muodostavat hierarkkisen kokonaisuuden. Olioliteraalit määrittelevät olion ilman erillistä luokan määritystä, ja ne rakentuvat aaltosulkeista, joiden väliin määritellään avain-arvopareja /22/. Arvoina voi olla muun muassa vakioita, muuttujia, funktioita tai toisia olioita. Olioliteraalien käyttöön päädyttiin, koska sovelluksen käyttöliittymän eri osista ei tarvitse olla useampia samanaikaisia instansseja. Olioliteraalien sisällä on määritelty sen käyttöliittymän osaan kuuluva toiminnallisuus. Taulukossa 2 on yksinkertainen esimerkki olioliteraalista, jolla on sen luomiseen ja tuhoamiseen käytetyt funktiot.

Taulukko 2: Esimerkki olioliteraalista

```
{
  create:  function() { /* luo käyttöliittymän osa  */ },
  destroy: function() { /* tuhoa käyttöliittymän osa*/ }
}
```

Sovellus koostuu kolmesta pääasiallisesta olioliteraalista, joita ovat MenuView, CategoryTree ja MainView. MenuView sisältää sovelluksen yleisiä valikoita ja toimintoja, CategoryTree kategoriapuun sekä siihen liittyviä toimintoja, ja MainView varsinaisen päänäkymän, jossa voidaan esittää erilaisia alinäkymiä välilehtinä. Jokainen näistä kolmesta pääosasta sisältää edelleen kaikki ne aliosat, jotka kuuluvat niiden muodostamaan käyttöliittymän alueeseen. Nämä pääosat ja niiden aliosat sisältävät kukin funktiot osan luomiseen ja tuhoamiseen, osaan liittyvät muuttujat ja muun toiminnallisuuden. Näiden lisäksi on määritelty olioliteraali Tools, joka sisältää erilaisia apufunktioita, ja Dialog, joka sisältää funktiot erilaisten valmiiksi konfiguroitujen jQuery UI -pohjaisten dialogien näyttämiseen. Yhdessä näistä muodostuu olioliteraalien rakenne, jota on kuvattu taulukossa 3.

Taulukko 3: Asiakassovelluksen olioliteraalien muodostama hierarkia

```
├── MenuView
├── CategoryTree
│   ├── CatAddons
│   ├── ManAddons
│   └── Filters
├── MainView
│   └── Tab
│       ├── Category
│       ├── Category_Pricing
│       ├── Category_Products
│       ├── Category_Filters
│       └── ProductFilters
├── Tools
└── Dialog
```

CategoryTree sisältää olioliteraalit kategorian ja valmistajien hintalisälle, sekä kategorian suodatukselle. Näistä löytyvää toiminnallisuutta käytetään hintalisäsääntöjen ja suodatussääntöjen hakemiseen ja esittämiseen suoraan kategoriapuussa, jotta niitä voidaan tarkastella kokonaisuutena. MainView sisältää välilehdet kategorialle, sen hinnoittelusäännöille, tuotehinnoittelulle ja suodatussäännöille, sekä erillisien näkymän kaikkien tuotekohtaisten suodatussääntöjen hallintaan kerralla. Jokainen välilehdistä sisältää kaiken siihen liittyvän toiminnallisuuden pyyntöjen ja käskyjen lähettämiseen palvelimelle. Kukin välilehdistä muodostaa näin oman työkalunsa, ja uusia työkaluja voidaan myöhemmin lisätä uusina välilehtinä.

JavaScript-olioiden lisäksi asiakassovellus sisältää Handlebars-sivupohjat kaikille käyttöliittymän välilehdille. Näissä on määritelty HTML-koodilla välilehtien perusrakenne, ja siihen Handlebarsin syntaksilla osia joita täydennetään dynaamisesti ennen HTML-koodin esittämistä. Näin välilehtien sisältö rakennetaan dynaamisesti täydentämällä Handlebars-sivupohjaa palvelimelta saadulla tiedolla.

6.1.3 Välilehtien rakenne

Välilehdet hakevat tarvitsemansa tiedon palvelimelta ennen kuin niiden sisältö esitetään, ja ne reagoivat käyttäjän toimenpiteisiin kommunikoimalla palvelinsovelluksen kanssa. Taulukossa 4 on kuvattu yksinkertaistettuna välilehden tyypillis-

tä rakennetta. Välilehden luomiseen tarkoitetussa metodissa *create()* haetaan ensin dataa palvelimelta funktiolla *getData*, joka lähettää AJAX-pyynnön palvelimelle. Palvelinsovellus palauttaa asiakassovellukselle tietoa JSON-muodossa. JSON-data otetaan talteen, ja annetaan edelleen Handlebars-sivupohjan parametriksi funktiossa *render*. Funktio täydentää sivupohjan lopulliseksi käyttöliittymänäkymäksi. Lopuksi sivuun liitetään interaktiivista toiminnallisuutta funktiolla *init*. Jos välilehden sisältö käyttöliittymässä halutaan tyhjentää, ajetaan metodi *destroy()*.

Taulukko 4: Asiakassovelluksen välilehden rakenne

```
Tab_Name = {
  id       : 'element_id',
  template : 'template_filename',
  data     : {},

  create : function() {
    var tab = this;
    getData(render);

    function getData(callback) {
      $.get({ Tools.buildRequestUrl('getData', {param: 'abc'})
      }).done(function(data) {
        var response = Tools.parseJSendResponse(data, 'errMsg');

        if (response) {
          tab.data = response.data;
          callback();
        }
      });
    };

    function render() {
      $('#' + tab.id).render(tab.template, {
        data : tab.data
      }, function() {
        init();
      });
    };

    function init() {
      $("#" + tab.id + " .btn").click(function(e) {
        e.preventDefault();
        alert('Data: ' + tab.data);
      });
    };
  },
  destroy : function() {
    this.data = {};
    $('#' + this.id).empty();
  }};
```

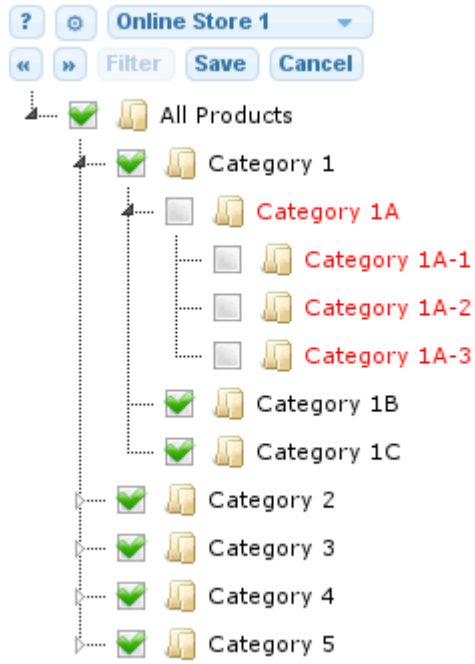
6.1.4 Sovelluksen käynnistyminen ja perusnäkymä

Sovelluksen käynnistyessä ladataan ensin sen pohjana toimiva staattinen HTML-dokumentti, jossa on määritelty sivun perusrakenne. Tämän jälkeen käynnistyy JavaScript-funktio, joka aloittaa käyttöliittymän osien lataamisen ja estää käyttäjää koskemasta käyttöliittymään kun sen lataus on kesken. Tässä vaiheessa pyydetään palvelinsovellukselta tiedot aktiivisesta verkkokaupasta, sekä lista valmistajista. Nämä tiedot pidetään tallessa asiakassovelluksessa sen käytön ajan, jotta niihin voidaan viitata tarvittaessa. Tämän jälkeen luodaan kohdassa 6.1.2 kuvatut sovelluksen pääosat *MenuView*, *CategoryTree* ja *MainView*, jotka hakevat tarvittaessa tiedoa palvelinsovellukselta, ja lopulta sallitaan sovelluksen käyttö. Tämän käynnistymisprosessin lopputuloksena saadaan kuviossa 11 kuvattu asiakassovelluksen perusnäkymä.

Kuvio 11: Asiakassovelluksen perusnäkymä

Tässä *MenuView* koostuu vasemmassa ylänurkassa olevasta sovelluksen käyttöohjeen avaavasta napista, konfiguraatiovalikosta ja verkkokaupan valintaan tarkoitetusta pudotusvalikosta. *CategoryTree* muodostuu jsTree-kirjastolla rakennetusta kategoriapuusta, ja sen yläpuolella olevista puuhun liittyvistä napeista. Napeilla voidaan siirtyä puussa kategorioiden suodatusnäkymään, ja näyttää tai piilottaa suodatus- ja hinnoittelusääntöjä suoraan puurakenteessa. *MainView* muodostuu oikealla puolella olevasta osiosta, joka sisältää eri työkalujen näkymät välilehtinä. Kun kategoriapuusta valitaan jokin kategoria, avautuu *Category* -välilehden alle uusi rivi välilehtiä, jotka sisältävät työkalut kategorian hinnoittelusääntöjen tekemiseen, tuotekohtaiseen hinnoitteluun ja suodatussääntöjen tekemiseen.

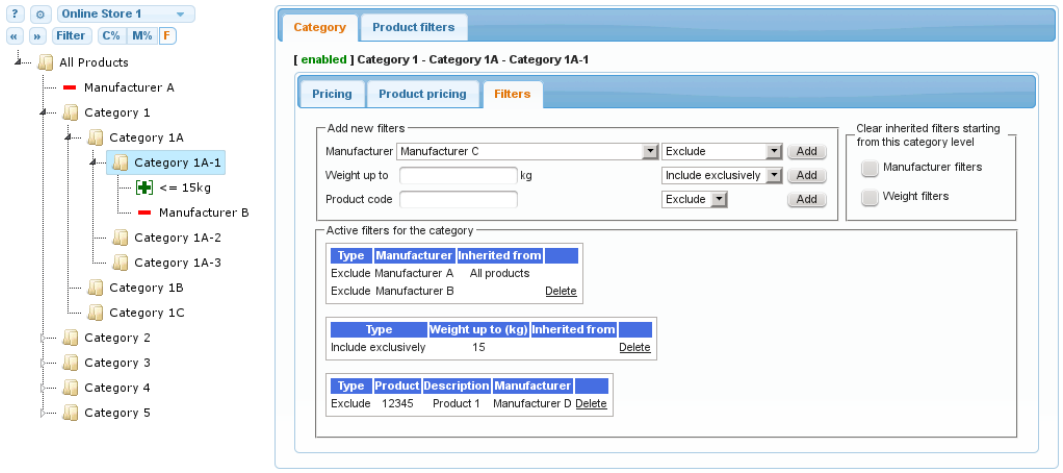
6.1.5 Kategoriasuodatuksen hallinta



Kuvio 12: Kategoriasuodatuksen editointi kategoriapuussa

Kuviossa 12 on kuvattu suoraan kategoriapuussa tapahtuvaa kategorioiden suodatusta verkkokaupalle. Valituista kategorioista otetaan tuotteet mukaan tuotevalikoimaan. Kategorioita voidaan valita puusta yksitellen tai kokonaisina sektioina. Ennen tallennusta ohjelma varmistaa käyttäjältä että muutokset ovat haluttuja.

6.1.6 Suodatussääntöjen hallinta

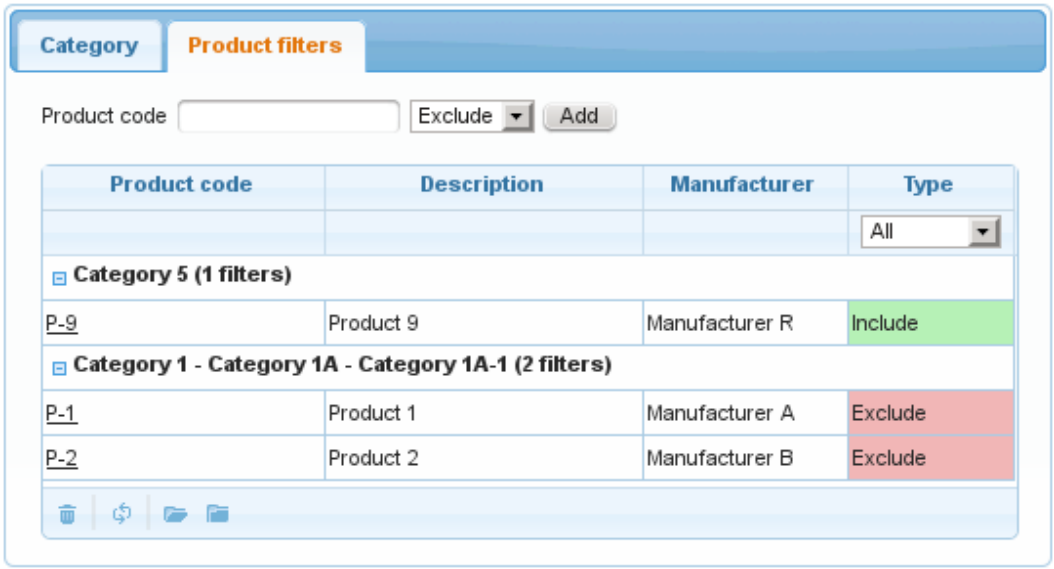


Kuvio 13: Suodatussääntöjen hallinta käyttöliittymässä

Kuviossa 13 on kuvattu kategorian suodatussääntöjen hallintaa. Tästä näkymästä nähdään valmistajille ja painolle annetut suodatussäännöt, jotka ovat voimassa puurakenteesta valitussa kategoriassa. Tämä käsittää sekä suoraan valitulle kategorialle määritellyt säännöt että säännöt, jotka on peritty ylemmiltä kategoriatasoilta. Kategorian suodatussääntöjen hallinnassa näytetään lisäksi niiden tuotteiden tuotekohtaiset säännöt, jotka kuuluvat sillä hetkellä valittuun kategoriaan. Valmistajille ja painolle määritellyt säännöt voidaan myös halutessa näyttää suoraan kategoriapuussa, jolloin niitä voidaan tarkastella kokonaisuutena.

Suodatusnäkymästä voidaan lisätä ja poistaa suodatussääntöjä kategorialle, mikä mahdollistaa myös perityn säännön syrjäyttämisen uudella säännöllä. Työn toteutusvaiheessa suodatustoimintoon lisättiin myös mahdollisuus tyhjentää perityt säännöt niin, että suodatus alkaa halutulla kategoriatasolla puhtaalta pöydältä.

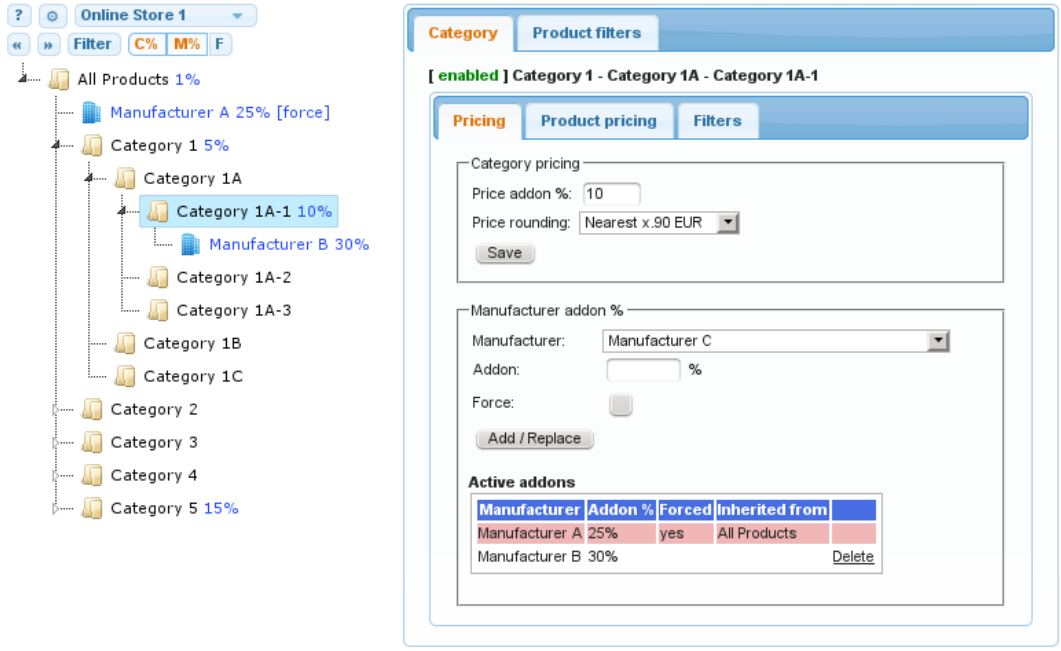
6.1.7 Tuotekohtaisen suodatuksen hallinta



Kuvio 14: Tuotekohtaisen suodatuksen hallinta käyttöliittymässä

Kuviossa 14 on kuvattu tuotekohtaiselle suodatukselle luotua erillistä näkymää, josta voidaan hallinnoida kerralla kaikkia tuotekohtaisia suodatussääntöjä. Säännöt on ryhmitelty kategorioittain. Näkymän tarkoituksena on antaa käyttäjälle yleiskuva voimassaolevista säännöistä, koska näitä sääntöjä olisi vaikea esittää sovelluksen kategoriapuussa.

6.1.8 Hinnoittelusääntöjen hallinta

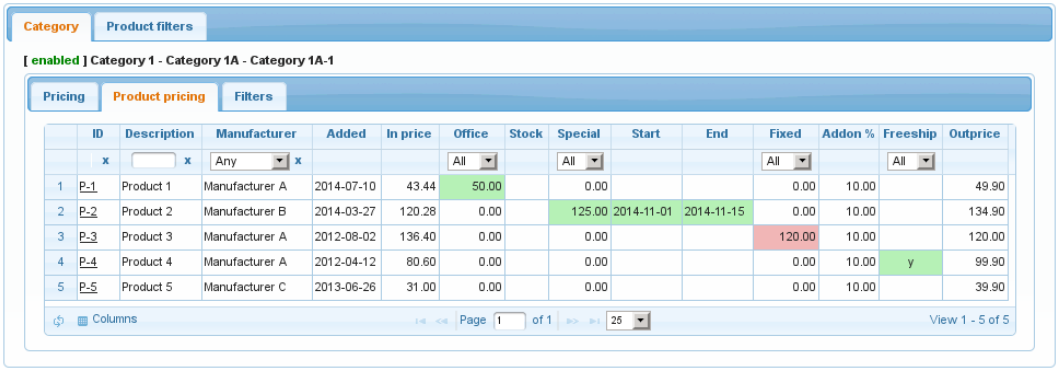


Kuvio 15: Hinnoittelusääntöjen hallinta käyttöliittymässä

Kuviossa 15 on kuvattu kategorioiden hinnoittelusääntöjen hallintaa. Tästä näkymästä nähdään kategoriapuusta valitulle kategorialle ja siinä oleville valmistajille määritellyt prosentuaaliset hintalisät. Tämä sisältää myös perityt säännöt. Näkymästä voidaan lisätä ja poistaa hinnoittelusääntöjä. Halutessa säännöt voidaan laittaa näkyviin myös kategoriapuuhun, jolloin niitä voidaan tarkastella kokonaisuutena.

Kategorialle määritelty hintalisä kumoaa normaalisti kaikki hinnoittelusäännöt, joita kategoria olisi muuten perinyt, jolloin kategoriassa on voimassa vain sen oma hinnoittelusääntö. Tämä johtuu siitä, että jokaisen tuotteen lisäprosentiksi valitaan normaalisti lähin mahdollinen kategoria- tai valmistajaprosentti, kun kategoriahierarkiaa kuljetaan ylöspäin alkaen tuotteen omasta kategoriasta. Halutessa voidaan kuitenkin määrittää valmistajakohtainen prosentuaalinen hintalisä pakotetuksi, jolloin se periytyy alakategorioille, vaikka ne sisältäisivät sääntöjä jotka normaalisti kumoaisivat perityn säännön. Tätä mahdollisuutta on ilmaistu käyttöliittymässä valinnalla *force*.

6.1.9 Tuotekohtaisten erikoishintojen hallinta



Kuvio 16: Tuotekohtainen erikoishintojen hallinta käyttöliittymässä

Kuviossa 16 on esitetty tuotekohtaisten erikoishintojen taulukkomuotoista hallintaa. Taulukko on rakennettu käyttäen jqGrid-kirjastoa, ja siinä näytetään kategoriapuusta valitun kategorian tuotteet, jotka ovat läpäisseet verkkokaupan suodatuksen. Taulukko on tarkoitettu tuotteiden erikoishintojen ja ilmaisen toimituksen hallintaan - muut kentät ovat näkyvillä vain lisäinformaationa. Taulukon sisältöä voidaan rajata sen sarakkaiden yläpuolella olevilla hakukentillä, ja järjestää sarakkeiden arvojen perusteella. Kaikki taulukon sisältö haetaan palvelinsovellukselta.

Taulukosta voidaan muuttaa tuotteiden erikoishintoja, kuten tiettyyn arvoon lukittua hintaa tai aikavälin sisältävää kampanjahintaa muokkaamalla taulukon solua suoraan. Kun muokkaus hyväksytään, palvelinsovellukselle lähetetään AJAX-pyyntö muutoksesta, johon palvelinsovellus vastaa ilmoittamalla onnistuiko muokkaus. Lisäksi palvelinsovellus palauttaa tiedon siitä mikä oli tuotteen voimassaoleva myyntihinta muokkauksen jälkeen, ja tämä tieto päivitetään dynaamisesti taulukon myyntihinnan sarakkeeseen. Asiakassovellus myös vertaa muutettua hintaa tuotteen sisäänostohintaan, ja värjää muutetun solun joko vihreäksi tai punaiseksi sen mukaan onko muutettu arvo sisäänostohintaa korkeampi vai alhaisempi, eli myytäisiinkö tuotetta sillä hinnalla voitolla vai tappiolla.

Taulukossa kiinnitettiin erityistä huomiota käytettävyyteen näppäimistöllä. Taulukossa liikkuminen ja editointi onnistui näppäimistöltä valmiiksi, mutta sivujen selaaminen näppäimistöltä lisättiin uutena ominaisuutena jqGrid-kirjaston koodiin.

6.2 Palvelinsovellus

Palvelinsovelluksen tehtävänä on vastaanottaa asiakassovellukselta pyyntöjä ja käskyjä, joiden perusteella muokataan tietokannassa olevia tuotteiden suodatus- ja hinnoittelumäärityksiä sekä tuotekohtaisia hintoja.

6.2.1 Komponenttien valinta

Palvelinsovelluksesta päätettiin suunnitteluvaiheessa, että se tulee toteuttaa PHP-ohjelmointikielellä, ja tietokantana käytetään olemassa olevaa PostgreSQL-tietokantaa. Kommunikointi tietokantaan tapahtuu PEAR DB -kirjastolla. Muita varsinaisia komponentteja ei tarvittu, jotta palvelinpuolelle saatiin kehitettyä sovellus, joka vastaanottaa HTTP GET ja POST -pyyntöjä, suorittaa tarvittavat operaatiot tietokantaan ja palauttaa dataa.

Palautettavan datan formaatiksi valittiin JSON, koska sen luonti PHP-kielellä on helppoa ja käsittely JavaScript-kielellä asiakassovelluksessa yksinkertaista. Palautettavalle JSON-datalle tarvittiin kuitenkin jokin yhtenäinen muoto, johon saadaan mukaan tieto operaation onnistumisesta sekä mahdollinen virheilmoitusviesti. Erilaisten vaihtoehtojen tutkimisen jälkeen löydettiin yksinkertaiselta vaikuttanut JSend. JSend on spesifikaatio, joka määrittelee säännöt palvelimen palauttaman JSON-datan muotoilulle /23/. Taulukossa 5 on esimerkki JSend-vastauksesta. Tässä *status* sisältää tiedon operaation onnistumisesta ja *data* varsinaisen palautettavan datan. Operaation epäonnistuessa voidaan mukaan laittaa myös *message*, joka sisältää virheilmoituksen.

Taulukko 5: Esimerkki JSend-vastauksesta

```
{
    status : "success",
    data : {
        "entry" : { "id" : 1, "name" : "example" }
     }
}
```

6.2.2 Ohjelmointirajapinta ja operaatiot

Palvelinsovelluksen tarjoama ohjelmointirajapinta toimii HTTP GET ja POST -pyynnöillä. HTTP GET -pyynnössä lähetetään palvelinsovellukselle aina kaksi parametria, joiden nimiä ei voida käyttää muissa operaatioissa GET -parametreina. Näistä ensimmäinen on käsiteltävän verkkokaupan tunniste, ja jälkimmäinen rajapinnasta kutsuttavan operaation nimi.

Varsinaiset operaation parametrit lähetetään joko HTTP GET tai POST -pyyntöjen osana, riippuen siitä, ollaanko palvelinsovellukselta hakemassa tietoa vai muuttamassa sitä. Tarvittavat parametrit riippuvat operaatiosta, ja joillekin operaatioille ei tarvita parametreja lainkaan.

Palvelinsovelluksen operaatiot pyrittiin rakentamaan pääsääntöisesti siten, että ne eivät ole sidoksissa pelkästään tässä työssä kehitettyyn asiakassovellukseen. Tämä tarkoittaa sitä, että operaatioita voidaan tarvittaessa kutsua myös muista sovelluksista. Toisaalta tämä mahdollistaa myös asiakassovelluksen rakentamisen uudelleen tai muuttamisen käyttämään toista tekniikkaa ilman, että palvelinsovellusta täytyy muuttaa. Operaatioiden nimissä on käytetty sanoja *get*, *set*, *add*, *delete* ja *update* kuvaamaan sitä, mitä ne tekevät. Taulukossa 6 on listattu merkittävimmät palvelinsovelluksen tarjoamat operaatiot.

Taulukko 6: Palvelinsovelluksen merkittävimmät operaatiot

`getWebshop`
`getManufacturers`
`getCategory`
 	
`getFilters`
`addFilter`
`deleteFilter`
`setFilterClear`
	`getCategoryAddons`
	`updateCategoryPricing`
	`addManufacturerAddon`
	`deleteManufacturerAddon`
	
	`getProducts`
	`updateProduct`
	
	`getCategoryTree`
	`getCategoryTreeCategoryAddons`
	`getCategoryTreeManufacturerAddons`
	`getCategoryTreeFilters`
	`updateCategoryFilters`

Operaatiot palauttavat JSON-dataa JSend-formaatin mukaisessa muodossa.

6.2.3 Sovelluksen rakenne ja toiminta

Palvelinsovellus on pyritty rakentamaan olio-ohjelmoinnin periaatteita noudattaen. Sovelluksen käyttö lähtee rajapinnan tarjoavasta PHP-tiedostosta, jota asiakassovellus kutsuu. Kun tiedostoa kutsutaan, se tarkastaa että parametrina annetut verkkokauppa ja operaatio löytyvät. Tämän jälkeen luodaan instanssi erityisestä controller-luokasta, josta kutsutaan operaatiota vastaavaa metodia. Metodissa luodaan operaatiossa tarvittavat oliot sekä varsinainen instanssi operaatiosta, suoritetaan operaatio ja palautetaan JSend-formaatissa oleva JSON-data asiakassovellukselle. Taulukossa 7 on esitetty yksinkertaistettuna operaatiota joka palauttaa asiakassovellukselle kategoriapuusta valitun kategorian tiedot.

Taulukko 7: Palvelinsovelluksen operaatio getCategory

```
class API_getCategory {
  private $category;

  public function __construct(Category $category) {
    $this->category = $category;
  }

  public function render($param) {
    if (isset($param['cat']) and ctype_digit($param['cat'])) {
      $cat_id = intval($param['cat']);

      try {
        $this->category->set($cat_id);
      } catch (Exception $ex) {
        return Ajax::returnErrorJSON($ex->getMessage());
      }

      $data = array('category' => $this->category);
      return Ajax::returnSuccessJSON($data);
    }
    else {
      return Ajax::returnErrorJSON('Invalid category id');
    }
  }
}
```

Palvelinsovelluksessa hyödynnetään PHP:n funktiota *json_encode*, jolla olio voidaan muuntaa suoraan JSON-dataksi. Näin palvelinsovelluksessa saadaan helposti muunnettua tietoa oliomuodosta jossa sitä käsitellään sellaiseen muotoon joka voidaan lähettää asiakassovellukselle.

6.3 Tallennetut tietokantaproseduurit

Tallennettujen tietokantaproseduurien tehtävänä on suodatus- ja hinnoittelusääntöjen täytäntöönpano. Nämä luotiin PL/pgSQL-kielellä PostgreSQL-tietokantaan.

6.3.1 Nested set -menetelmän hyödyntäminen

Sekä suodatus- että hinnoittelusääntöjen haluttiin periytyvän alakategorioille kategoriahierarkiassa. Jotta tämä onnistuisi käytännössä tietokantaproseduurien avulla, täytyy kussakin kategoriassa voimassaolevien sääntöjen laskemiseksi selvittää koko se kategoriapolku joka johtaa tutkittavasta kategoriasta ylös juurikategoriaan, ja kaikki ne säännöt jotka tällä polulla oleville kategorioille on määritelty. Polku voidaan selvittää rekursiivisesti useilla SQL-kyselyillä selvittämällä kunkin kategorian yläkategoria yksi kerrallaan. Koska kategoriat on tallennettu tietokantaan nested set -menetelmää käyttäen, polku saadaan kuitenkin selville yhdellä SQL-kyselyllä. Tällaisen SQL-kyselyn rakennetta on kuvattu taulukossa 8.

Taulukko 8: Kategorian polun selvittäminen nested set -rakenteesta /10/

```
SELECT parent.id, parent.name
FROM  nested_category AS node,
      nested_category AS parent
WHERE node.lft BETWEEN parent.lft AND parent.rgt
      AND node.id = (tutkittavan kategorian tunniste)
ORDER BY parent.lft DESC;
```

Yhdistämällä tähän kyselyyn suodatus- ja hinnoittelusääntöjen tietokantatauluja, saadaan selvitettyä säännöt jotka ovat voimassa kussakin kategoriassa. Taulukossa 9 on havainnollistettu kategoriapolun selvittämistä, tutkimalla kategoriaa *Category 1A-1* joka voidaan nähdä kategoriapuussa kuviossa 12.

Taulukko 9: Esimerkkikategorian polku juurikategoriaan

id	**name**
1154	Category 1A-1
1140	Category 1A
1139	Category 1
1	All Products

6.3.2 Tuotteiden suodatus ja synkronointi

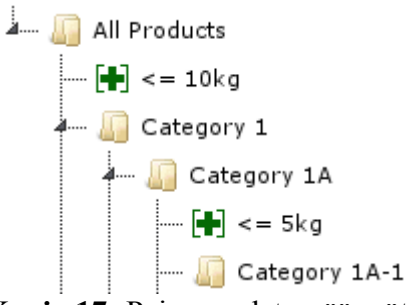
Tuotteiden suodatusta varten kehitettiin tallennettu tietokantaproseduuri, joka koostuu kahdesta vaiheesta.

Suodatuksen ensimmäisessä vaiheessa suoritetaan suunnitteluvaiheen mukainen tuotteiden suodatus. Suodatuksessa käydään jokainen käsiteltävään verkkokauppaan valittu tuotekategoria erikseen läpi. Kategoriasta selvitetään ensin onko sille voimassa jokin painosuodatussääntö, ja tämän jälkeen selvitetään voimassaolevat valmistajakohtaiset suodatussäännöt. Sitten suoritetaan kategorian suodatus ajamalla kategorian tuotteet suodatuksen läpi. Kun kaikkien kategorioiden suodatussäännöt on käyty läpi, lisätään lopputulokseen ja siitä poistetaan tuotekohtaisilla suodatussäännöillä määritellyt tuotteet. Suodatuksen tuloksena saadaan lista kaikista suodatuksen läpäisseistä tuotekoodeista.

Esimerkkikategorian *Category 1A-1* aktiivisen painosuodatussäännön selvittämistä on havainnollistettu taulukon 10 ja kuvion 17 avulla. Kuviosta 17 nähdään, että juurikategorialle *All products* on määritelty painosuodatussääntö joka edellyttää, että tuotteiden paino on korkeintaan 10 kg. Kategoriassa *Category 1A* on kuitenkin määritelty, että paino saa olla korkeintaan 5 kg. Yhdistämällä kappaleessa 6.3.1 esitettyyn kategoriapolun hakuun painosuodatussääntöjen tietokantataulu, löydetään taulukon 10 tuloksen mukaisesti lähin polussa oleva painosuodatussääntö arvolla 5, joka näin valitaan kategorian *Category 1A-1* perimäksi säännöksi.

Taulukko 10: Painosuodatussääntöjen periytyminen

id	name	maxweight
1140	Category 1A	5
1	All Products	10



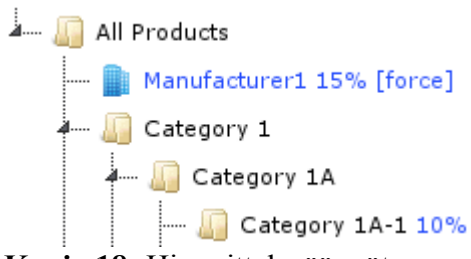
Kuvio 17: Painosuodatussäännöt

Suodatuksen toisessa vaiheessa synkronoidaan verkkokaupan tuotetaulun sisältö poistamalla siitä tuotteet jotka eivät enää läpäisseet suodatusta, ja lisäämällä siihen uudet suodatuksen läpäisseet tuotteet. Synkronointivaiheessa pyritään kuitenkin suojaamaan sellaiset tuotteet poistamiselta joista tavalla tai toisella nähdään, että niitä on tarkoitus myydä verkkokaupassa suodatussäännöistä huolimatta. Synkronointi tehdään vasta varsinaisen suodatusprosessin jälkeen siitä syystä, ettei se hidastaisi tuotetaulun muuta käsittelyä.

6.3.3 Tuotteiden hinnoittelu

Tallennettu tietokantaproseduuri tuotteiden hinnoittelusääntöjen valinnalle saatiin lähes sellaisenaan kappaleessa 3.2 kuvatusta aiemmin kehitetystä kategoriatyökalusta. Uutena ominaisuutena tähän proseduuriin lisättiin kuitenkin mahdollisuus pakottaa jonkin valmistajakohtaisen prosentuaalisen hintalisän periytyminen kategoriahierarkiassa. Jotta pakotettu valmistajakohtainen hintalisä saadaan huomioitua hinnoittelusäännön valinnassa, täytyy hinnoitteluproseduurin tulokset priorisoida ensisijaisesti pakotettujen valmistajakohtaisten hintalisien perusteella, ja toissijaisesti tutkittavasta kategoriasta juurikategoriaan johtavan polun perusteella.

Esimerkkikategorialle *Category 1A-1* periytyvää pakotettua valmistajakohtaista prosentuaalista hintalisää on havainnollistettu kuviossa 18 ja taulukossa 11. Kuvion 18 perusteella nähdään, että juurikategoriaan *All products* on määritelty pakotettu 15% hintalisä valmistajalle *Manufacturer1*, ja kategorialle *Category 1A-1* on määritelty 10% hintalisä.



Kuvio 18: Hinnoittelusäännöt

Normaalitilanteessa kategoriassa olevalle tuotteelle valitaan lähin saatavilla oleva kategoriahintalisä tai valmistajahintalisä. Tällöin kaikilla kategoriassa *Category*

1A-1 olevilla tuotteilla olisi hintalisänä 10%. Valmistajakohtaisen hintalisän asettaminen pakotetuksi muuttaa tilanteen kuitenkin niin, että pakotettu hintalisä valitaan käyttöön kaikille niille kategorian *Category1A-1* tuotteille joiden valmistaja on *Manufacturer1*. Taulukosta 11 nähdään sääntöjen järjestäytyminen tässä tilanteessa, jolloin tuotteen perityksi hintalisäksi valitaan 15%. Kuten taulukosta nähdään, kategoriat eivät enää järjestäydy pelkästään kategoriapolun mukaan, vaan valmistajakohtaisen hinnoittelusäännön pakotus on nostettu prioriteetissa korkeammalle.

Taulukko 11: Hinnoittelusääntöjen periytyminen pakotettuna

id	**name**	**man_coefficient**	**cat_coefficient**	**force**
1	All Products	1.15		x
1154	Category 1A-1		1.1	

7 SOVELLUKSEN TESTAUS

Sovellusta testattiin pääasiallisesti osana kehitysprosessia manuaalisena testauksena. Käytännössä tämä tarkoitti sovelluksen käyttämistä suorittamalla operaatioita, joita loppukäyttäjä tulee sovelluksella tekemään. Tämä sisälsi suodatus- ja hinnoittelusääntöjen lisäämistä ja poistamista ja jokaisen sääntömuutoksen jälkeen sen varmistamista, että sääntöjen täytäntöönpano tuotekohtaisesti ja tallennettujen tietokantaproseduurien osalta toimii edelleen tarkoituksenmukaisesti. Toisena tärkeänä testauksen osana oli virheellisten ja kiellettyjen tietojen syöttäminen ohjelmaan sen varmistamiseksi, että virhetilanteisiin on varauduttu asianmukaisesti, erityisesti palvelinpäässä, ja että käyttäjä saa virhetilanteissa mahdollisimman selkeän virheilmoituksen. Osana tätä prosessia asiakaspään käyttöliittymää pyrittiin myös tarkoituksellisesti rikkomaan sekä normaalilla käytöllä että asiakaspään ohjelmakoodia muokkaamalla. Näillä menetelmillä löytyi kehitystyön aikana tehokkaasti erilaisia ongelmakohtia jotka saatiin korjattua.

Testausprosessina voidaan pitää myös erilaisia kokouksia, joissa sovelluksen sen hetkistä tilaa arvioitiin toimeksiantajayrityksen kanssa. Näissä kokouksissa tuli hyvin esille käyttötilanteita tai muita ongelmakohtia, joita sovelluksessa ei oltu otettu parhaalla tavalla huomioon. Eräs tällaisessa kokouksessa muutettu ominaisuus oli yksittäisten tuotteiden suodatus, joka päätettiin suodatusprosessissa viedä kaikkien muiden suodatussääntöjen edelle prioriteetiltaan, jotta yksittäisiä tuotteita voidaan muista säännöistä riippumatta aina ottaa mukaan tai pudottaa pois.

Kun sovellus oli saavuttanut tilan, jossa sille asetetut vaatimukset oli pääasiassa toteutettu ja se oli normaalisti käytettävissä, sovellus siirrettiin yhteiselle testipalvelimelle. Täällä sovelluksen loppukäyttäjät pääsivät testaamaan sovellusta tuotantokäyttöä vastaavassa ympäristössä. Tässä testausvaiheessa käyttäjät pyrkivät tekemään todellisia hinnoittelu- ja suodatusmäärityksiä, jotka voidaan myöhemmin ottaa suoraan tuotantokäyttöön. Tätä kirjoittaessa testipalvelimella tapahtuva testaus jatkuu edelleen aina siihen asti, kunnes yrityksen uudistunut järjestelmä, ja sen myötä myös tässä työssä kehitetty sovellus saadaan tuotantokäyttöön.

8 YHTEENVETO JA JOHTOPÄÄTÖKSET

Opinnäytetyön tuloksena saatiin toimeksiantajayrityksen järjestelmään web-sovellus, joka täyttää sille asetetut vaatimukset ja vastaa yrityksen tarpeita, mahdollistaen verkkokauppakohtaisen tuotteiden suodatukseen ja hinnoitteluun. Lisäksi saatiin kehitettyä tarvittavat tallennetut tietokantaproseduurit, joilla varsinainen tuotteiden suodatus voidaan suorittaa web-sovelluksella tehtyjen määritysten mukaisesti, ja joilla tuotteille saadaan oikea lopullinen hinta hinnoittelusääntöjen perusteella.

Koska sovelluksen käyttöliittymän näkymät pohjautuvat välilehtiin, erityisesti kategorialähtöisiä ominaisuuksia voidaan tarvittaessa helposti lisätä sovellukseen uusina välilehtinä. Sovelluksen asiakaspään ja palvelinpään toiminnallisuuksien erottaminen toisistaan mahdollistaa myös palvelinpään operaatioiden käyttämisen muista sovelluksista tarpeen mukaan, josta voidaan hyötyä erilaisissa verkkokaupan ylläpitonäkymissä. Nämä seikat helpottavat sovelluksen jatkokehitystä ja parantavat sen joustavuutta.

Jatkokehityksen kannalta tärkeäksi kysymykseksi nousee varmasti jossain vaiheessa JavaScript-ohjelmistokehyksen valinta selaimella käytettävässä asiakassovelluksessa. JQuery-kirjasto soveltui hyvin asiakassovelluksen perustaksi, koska sovellus ei kasvanut suurikokoiseksi ja sovelluksessa käytetyt komponentit olivat kaikki jQuery-pohjaisia. Ohjelmistokehyksellä sovellukselle saataisiin kuitenkin selkeä ja hallittu rakenne. Työn alkuvaiheessa parhaina vaihtoehtoina pidetyt ohjelmistokehykset olisivat kuitenkin osoittautuneet työn aikana ongelmallisiksi valinnoiksi, sillä näissä nähtiin muun muassa yhden ohjelmistokehyksen kehityksen päättyminen kokonaan, ja toisen ohjelmistokehyksen tulevaisuudensuunnitelmat, jotka ovat sen nykyisen version kanssa epäyhteensopivia. Tällaisiin muutoksiin nähden vakaa jQuery-kirjasto osoittautui työn aikana turvalliseksi valinnaksi. Tulevaisuudessa tehtävässä ohjelmistokehyksen valinnassa tulee myös miettiä mikä eri vaihtoehdoista palvelee yrityksen tarpeita yleisesti parhaiten, jotta samaa ohjelmistokehystä voidaan hyödyntää mahdollisimman monessa eri työkalussa.

LÄHTEET

/1/ Asiakasmarkkinointiliitto, Kaupan liitto, TNS Gallup. 2014. Verkkokauppatilasto 2014. Viitattu 25.10.2014. http://www.tns-gallup.fi/doc/digi/Verkkokauppatilasto_2014H1.pdf

/2/ PostNord Oy. 2014. Verkkokauppa Pohjoismaissa 2014. Viitattu 25.10.2014. http://www.postnordlogistics.fi/fi/Documents/Raportit/Verkkokauppa-Pohjoismaissa-2014.pdf

/3/ Multitronic Oy. 2014. Multitronic Oy:stä. Viitattu 25.10.214. http://multitronic.fi/main.php?id=multi

/4/ The PHP Group. Introduction to Variables. Viitattu 26.11.2014. http://php.net/manual/en/internals2.variables.intro.php

/5/ The PHP Group. PHP: Hypertext Processor. Viitattu 26.11.2014. http://php.net/

/6/ The PHP Group. What can PHP do? Viitattu 26.11.2014. http://php.net/manual/en/intro-whatcando.php

/7/ The PostgreSQL Global Development Group. What is PostgreSQL? Viitattu 26.11.2014. http://www.postgresql.org/docs/9.3/interactive/intro-whatis.html

/8/ Oracle and/or its affiliates. A Relational Database Overview. Viitattu 26.11.2014. https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html

/9/ The PostgreSQL Global Development Group. 38.1. Overview. Viitattu 26.11.2014. http://www.postgresql.org/docs/8.3/static/plpgsql-overview.html

/10/ Hillyer, M. 2012. Managing Hierarchical Data in MySQL. Viitattu 26.11.2014. http://mikehillyer.com/articles/managing-hierarchical-data-in-mysql/

/11/ W3C. HTML & CSS. Viitattu 26.11.2014. http://www.w3.org/standards/webdesign/htmlcss.html

/12/ Handlebars. Viitattu 26.11.2014. http://handlebarsjs.com/

/13/ Microsoft. JavaScript Fundamentals. Viitattu 26.11.2014. http://msdn.microsoft.com/en-us/library/ie/6974wx4d(v=vs.94).aspx

/14/ Easttom, C. 2008. Advanced JavaScript. 3. painos. Plano, Texas. Wordware Publishing, Inc.

/15/ The jQuery Foundation. jQuery. Viitattu 26.11.2014. http://jquery.com/

/16/ The jQuery Foundation. Category: Ajax. Viitattu 26.11.2014. http://api.jquery.com/category/ajax/

/17/ Refsnes Data. AJAX Introduction. Viitattu 26.11.2014. http://www.w3schools.com/ajax/ajax_intro.asp

/18/ Introducing JSON. Viitattu 26.11.2014. http://www.json.org/

/19/ Oracle Corporation and/or its affiliates. Netbeans IDE – The Smarter and Faster Way to Code. Viitattu 26.11.2014. https://netbeans.org/features/index.html

/20/ Oracle Corporation and/or its affiliates. Versioning. Viitattu 26.11.2014. https://netbeans.org/features/ide/versioning.html

/21/ EMS Database Management Solutions, Inc. EMS SQL Manager for PostgreSQL. Viitattu 26.11.2014. http://www.sqlmanager.net/products/postgresql/manager

/22/ Object literals. Mozilla Developer Network and individual contributors. Viitattu 26.11.2014. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Values,_variables,_and_literals#Object_literals

/23/ JSend. OmniTI. Viitattu 26.11.2014. http://labs.omniti.com/labs/jsend