Verkkokaupan toivelistaominaisuuden kehitys Django-ohjelmistokehyksellä case Levykauppa Äx

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

HAMK Visamäki, Tietojenkäsittely

Kevät 2018

Veeti Liljedahl

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittely
HAMK Visamäki

Tekijä Veeti Liljedahl **Vuosi** 2018

Työn nimi Verkkokaupan toivelistaominaisuuden kehitys Django -ohjelmistokehyksellä

Työn ohjaaja Tommi Saksa

TIIVISTELMÄ

Opinnäytetyössä oli tarkoitus kehittää toivelistaominaisuus Levykauppa Äx:n jo valmiina olevaan verkkokauppaan. Levykauppa Äx:llä oli tarve parantaa verkkokauppaansa, ja tämä työ oli osana verkkokaupan kehitysohjelmaa.

Tässä opinnäytetyössä selvitettiin, miten valmiiseen verkkokauppaan toteutetaan toivelistaominaisuus, ja että miten kyseinen ominaisuus toteutetaan Django-ohjelmistokehyksellä ja Python-ohjelmointikielellä. Työssä käytiin läpi paikallisen kehitysympäristön asennus. Työssä käytiin myös läpi, miten neljä muuta verkkokauppaa oli toteuttanut toivelistaominaisuuden omissa verkkokaupoissaan ja poimittiin tärkeimmät ominaisuudet niistä ylös.

Opinnäytetyössä tehtiin suunnittelutyötä toteutuksen pohjalle, ja työssä onnistuttiin toteuttamaan toimiva toivelistaominaisuus annettujen vaatimusten mukaisesti. Toivelistaominaisuus saatiin toimimaan yhdessä valmiin verkkokaupan kanssa paikallisessa kehitysympäristössä.

Avainsanat Verkkokauppa, Django, Python

Sivut 25 sivua

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Business Information Technology
HAMK Visamäki

Author Veeti Liljedahl **Year** 2018

Subject Wish List Feature Development in Online Store with Django Framework

Supervisor Tommi Saksa

ABSTRACT

In this thesis, the objective was to develop a wish list feature into Levykauppa Äx's online store. Levykauppa Äx had a need to improve their online store and the wish list feature was part of that program.

This thesis figured how to develop a wish list feature into the previously built online store with Django framework and Python programming language. The thesis includes information on how the development environment was set up. In the thesis was also included how four other online stores have done their wish list features and what advantages they had that could be used in the developed feature.

Planning was made prior to programming and the wish list –feature was successfully finished with given requirements. The feature was also successfully integrated into the existing online store in the development environment.

Keywords Online store, Django, Python

Pages 25 pages

SISÄLLYS

1 JOHDANTO

Tässä opinnäytetyössä suunniteltiin ja toteutettiin toivelistaominaisuus Levykauppa Äx -verkkokauppaan käyttäen Django-ohjelmistokehystä ja Python-ohjelmointikieltä. Toivelistaominaisuus on osa Levykaupan verkkokauppauudistusta, jossa verkkokauppaa kehitetään. Tämä ei tullut täysin valmiiksi opinnäytetyön aikana, johtuen suuresta opeteltavien asioiden määrästä. Ominaisuuden kehitys jatkuu opinnäytetyön loppumisen jälkeen.

Opinnäytetyön asiakas on Levykauppa Äx Oy. Levykauppa Äx on vuonna 1997 perustettu musiikki- ja videotallenteiden vähittäiskauppa, jonka kotipaikka on Kuopio ja toimitusjohtaja Jyri Jukka Lipponen. Levykauppa myy musiikkitallenteiden ja elokuvien lisäksi artistien ja yhtyeiden paitoja sekä muita fanituotteita verkkokaupassaan.

Levykaupan verkkokaupassa ei ole olemassa olevaa toivelistaominaisuutta, joten ominaisuuden työstö aloitetaan tyhjästä. Asiakkaalta on saatu ominaisuudesta määrittely, jonka perusteella lähdetään suunnittelemaan ja toteuttamaan ominaisuutta. Ominaisuus on kaikessa yksinkertaisuudessaan sellainen, että käyttäjä voi valita ja lisätä toivelistalleen minkä tahansa tuotteen verkkokaupasta. Lisäyksen jälkeen käyttäjä voi siirtyä omalle toivelistalleen ja valita tuotteita ostoskoriin tai poistaa tuotteita listalta.

Ominaisuus kehitetään jo olemassa olevan verkkokaupan päälle. Ennen kuin varsinaisen ominaisuuden työstöä voi aloittaa, pitää luoda kehitysympäristö omalle työasemalle ja tutustua verkkokauppaan tarkemmin. Koska verkkokauppa on ohjelmoitu Django-ohjelmistokehyksellä, back-end -ohjelmointikielenä on Python.

Opinnäytetyön tavoite on kertoa, kuinka Djangolla ja Pythonilla kehitetään toivelistaominaisuus jo olemassa olevaan verkkokauppaan. Tavoitteena opinnäytetyön aikana on myös oppia käyttämään Pythonia ja Djangoa tehokkaasti. Opinnäytetyön tutkimuskysymykset ovat:

- Miten Djangolla ja Pythonilla kehitetään toivelistaominaisuus?
- Miten toivelistaominaisuus kehitetään jo olemassa olevaan verkkokauppaan?

2 VERKKOKAUPAT

Verkkokauppa on kaupankäyntiä verkossa. Verkkokaupassa pätevät erilaiset arvot kuin normaalissa kaupankäynnissä, koska asiakkaalle on siirretty osa kauppiaan ennen tekemistä töistä. Verkkokauppatoiminnassa on tärkeää, että tehdään asiakkaalle ostotapahtuma mahdollisimman helpoksi, jotta asiakas tekee kauppiaan kannalta suotuisan päätöksen. (Hallavo 2013, s. 19-21)

2.1 Verkkokauppatoiminta

Kaupankäynti on mullistunut viimeisen kuudenkymmenen vuoden aikana. Valta on siirtynyt valmistajilta yhä enemmän asiakkaalle, ja valmistajien tulee vastata asiakkaiden tarpeisiin. Valmistajien tulee koittaa kommunikoida enemmän henkilökohtaisella tasolla asiakkaiden kanssa. Viimeiset kymmenen vuotta kaupankäynti on vakiintunut asiakkaiden ajaksi. Valmistajien tulee miettiä, miten kehittää omaa verkkokauppaa, jotta pystytään pitämään kiinni omasta markkinaosuudesta. (Hallavo 2013, 19-21.)

Asiakkuusmarkkinointiliitto, Kaupan liitto ja TNS Gallup teettivät vuonna 2010 tutkimuksen, jossa mitattiin suomalaisen verkkokaupan arvoksi jopa 4,8 miljardia euroa pelkästään tammi-helmikuun ajalta. Tutkimuksessa ennustettiin, että arvo lähenisi kymmentä miljardia euroa vuoden loppuun mennessä. Suurin osa kokonaisarvosta oli palveluita. (Ruotsalainen, Närhi & Juntunen 2010, 6.)

Verkkokauppaohjelmisto ja sen toiminnot voivat tehostaa kaupan toimintaa ja synnyttää kustannussäästöjä. Asiakkaan ostaessa tuotteita verkosta suurin ero on se, että asiakas ei tapaa verkkokauppiasta henkilökohtaisella tasolla. Verkkokauppias voi kuitenkin oikeanlaisella toiminnalla ja ominaisuuksilla kasvattaa asiakkaan tietoisuutta tuotteista ja palveluista. (Ruotsalainen, Närhi, Juntunen 2010, 7.)

Verkkokaupassa on mahdollisuus tavoittaa suurempi asiakaskunta kuin normaalissa kaupankäynnissä. Vaikka asiakaskunta on suurempi, on kuitenkin mahdollista saavuttaa parempi asiakaspalvelu. Verkkokauppaan voi luoda esimerkiksi wikisivuja tai foorumeita jotka parantavat asiakkaan kosketusta tuotteisiin ja palveluihin. Verkkokauppa voi myös helpottaa asiakkaan kynnystä antaa palautetta kaupan toiminnasta. (Ruotsalainen, Närhi, Juntunen 2010, 7-8.)

2.2 **Verkkokaupparatkaisuja**

Verkkokauppaa perustaessa tulee miettiä, millä tavoin sen haluaa pystyttää. Pystyttämiseen on lähtökohtaisesti kolme eri tapaa: oman verkkokaupan luominen tyhjästä, avoimen lähdekoodin palvelut, kuten WordPress-alustan WooCommerce-palvelu sekä tilaustyönä tehtävän räätälöidyn verkkokaupan hankkiminen. Tässä opinnäytetyössä kehitettävä verkkokauppa on itse tehty, eikä pohjalla ole käytetty esimerkiksi avoimen lähdekoodin valmiita verkkokaupparatkaisuja.

Tilaustyönä tehtäviä verkkokaupparatkaisuja kutsutaan yleisesti SaaS-palveluiksi, eli ”Software as a Service” -palveluiksi. SaaS-palvelut ovat rajoittuneempia kuin palvelimelle suoraan asennettavat palvelut. SaaS-palvelut soveltuvat enemmän pienille verkkokaupoille ja näissä palveluissa ei ole niin paljon räätälöinti- ja muokkausmahdollisuuksia kuin avoimen lähdekoodin järjestelmissä. (Järvenpää n.d.)

Isommille verkkokaupoille tarkoitetut järjestelmät, kuten esimerkiksi Magento, omaavat jo paljon muokkaus- ja integraatiomahdollisuuksia. Siitä johtuen näiden järjestelmien käyttöönottokustannukset ovat myös varsin hintavia. Tällaisen järjestelmän käyttöönotto saattaa maksaa jopa 5 000 – 500 000 euroa. (Järvenpää n.d.)

3 PYTHON-OHJELMOINTIKIELI

Python on tällä hetkellä yksi maailman käytetyimmistä ohjelmointikielistä Java-, C- ja C++ -kielten ohella (TIOBE index 2018). Pythonin suosiota on kasvattanut muun muassa sen laaja standardikirjasto. Pythonin standardikirjastossa on yli 100 moduulia matemaattisista funktioista graafisen käyttöliittymän luomiseen asti. Tämän kirjaston lisäksi Pythoniin on saatavilla myös kolmannen osapuolen tekemiä lisämoduuleita. (van Rossum 2009.)

3.1 Pythonin historiaa

Python on dynaaminen ohjelmointikieli, jonka kehitystyö alkoi vuonna 1989. Python julkaistiin varsinaisesti helmikuun 20. päivänä 1991 versionumerolla 0.9.0. Version 1.0.0 julkaisupäivä oli 26. tammikuuta 1994. (van Rossum 2009.)

Pythonin kehityksen on aloittanut alankomaalainen Guido Van Rossum. Van Rossum työskenteli aikoinaan CWI-nimisessä tutkimuslaitoksessa, josta Pythonin kehitystyö sai alkunsa. Van Rossum siirrettiin ABC-projektin loppumisen jälkeen Amoeba-projektiryhmään. Amoeba-projekti tarvitsi korkeatasoisempaa ohjelmointikieltä, josta idea ohjelmointikielestä, joka ”luo sillan C-kielen ja komentorivin välille” lähti liikkeelle. Tämä oli pitkään Pythonin iskulauseena. (van Rossum 2009.)

3.2 Python ohjelmoinnissa

Python on ohjelmointikieli, joka tarjoaa helpon käytön ja syntaksin ohella eväät myös monimutkaiseen ohjelmointiin. Kieli on myös helppo oppia. (Chun 2000, 3.) Pythonia luullaan yleisesti pelkäksi skriptauskieleksi, mutta tosiasiassa se on hyvin yleispätevä kieli (van Rossum 2009). Python on laajasti käytetty esimerkiksi webkehityksessä, numeerisen datan käsittelyssä, opetuksessa, graafisten käyttöliittymien luomisessa ja sovelluskehityksessä (Python, 2018).

Pythonin standardikirjastossa on tuki esimerkiksi HTML, XML, JSON, FTP ja IMAP internet protokollille. Web kehityksessä auttaa Django-ohjelmistokehyksen lisäksi myös esimerkiksi Pyramid-ohjelmistokehys. (Python, 2018) Pyramid-ohjelmistokehyksen motto on suomennettuna: ”aloita pienestä, päätä isosti, pysy valmiina” (Pyramid, 2018). Pythoniin on saatavilla myös lukuisia kolmannen osapuolen tekemiä moduuleita, jotka helpottavat webkehitystä.

3.3 **Dynaamiset ohjelmointikielet**

Dynaamisten ohjelmointikielten ero staattisiin on se, että muuttujien tyyppiä ei tarvitse erikseen määrittää, vaan tulkki määrittää ne ajonaikaisesti. Staattisissa ohjelmointikielissä muuttujille pitää kertoa, ovatko ne esimerkiksi tyyppiä integer vai tyyppiä string, jotta kääntäjä osaa tulkita muuttujat oikein.

Ohjelmointikieliä, joissa voidaan kirjoittaa dynaamisesti, on esimerkiksi JavaScript, Perl, PHP, Python ja Ruby. Staattisesti kirjoitettavia kieliä ovat esimerkiksi C-, C++-, Java- ja Rust -ohjelmointikielet.

3.4 **Pythonin ja PHP:n vertailu**

Webkehityksessä on front-end ja back-end ohjelmointia. Front-end -ohjelmointi tarkoittaa sitä osaa ohjelmasta, minkä käyttäjä näkee, ja minkä kanssa käyttäjä on tekemisissä. Back-end on puolestaan se osa ohjelmasta, mikä toimii pellin alla. Käyttäjä lähettää nettisivulla esimerkiksi lomakkeen (front-end), jonka pythonskripti ottaa vastaan ja käsittelee (back-end). Back-end -ohjelmointikieliä ovat esimerkiksi Java, Python, PHP ja Ruby. (Codeup 2014)

Back-end -ohjelmoinnissa on kolme osaa: serveri, sovellus ja tietokanta. Serveri on tietokone, joka ajaa ohjelmistoa vastatakseen käyttäjien pyyntöihin. Websovellus on back-end -kielellä kirjoitettu ohjelma, joka esimerkiksi kirjoittaa, hakee tai muuttaa dataa tietokannasta. Tietokantoihin tallennetaan sovelluksessa käytettävä data. (Codeup 2014.)

3.4.1 Yhtäläisyydet

Python ja PHP ovat dynaamisia, korkean tason ohjelmointikieliä. Korkean tason ohjelmointikieli tarkoittaa, että kirjoitetaan ohjelmakoodia, jonka tulkki kääntää koneelle ymmärrettäväksi kieleksi. Molempia kieliä on helppo oppia ja niitä voi ajaa melkein millä laitteistolla tahansa. Molemmat kielet tukevat nimiavaruuksia (namespace), ja metodien ketjuttamista. (Python wiki n.d.)

3.4.2 Eroavaisuudet

PHP-kielessä webkehitys-ominaisuudet on rakennettu suoraan kielen ytimeen. Pythonissa samat ominaisuudet saavutetaan lisäosina saatavina moduuleina. Jotkut ominaisuudet, kuten esimerkiksi CGI-moduuli, ovat saatavilla Pythonin standardikirjastossa. (Python wiki n.d.) Django on hyvä esimerkki ei-sisäänrakennetusta moduulista, joka auttaa webkehityksessä.

Python websovelluksia voi ajaa omissa prosesseissaan, tai suoraan web serverillä. PHP on taas upotettuna web serveriin. (Python wiki n.d.)

3.4.3 Syntaksi

PHP ja Python ovat syntaksia katsottaessa hyvin erilaiset kielet. Alla yksinkertainen ohjelma kirjoitettuna Pythonilla:

```
# funktion määrittely
def looper(list):

    # tulostaa jokaisen listan alkion
    for item in list:
        print(item)


def collect():
    list = ('first', 'second', 'third')

    # toisen funktion kutsu
    looper(list)


collect()
```

Ohjelmakoodi 1. Yksinkertainen ohjelma Pythonilla

Sama kirjoitettuna PHP:lla:

```
<?php

collect();

// funktion määrittely
function collect() {
    $list_1 = array('first','second','third');

    // listan lähetys toiseen funktioon
    looper($list_1);
}

// funktio ottaa vastaan listan
function looper($list_1) {
    foreach($list_1 as $item) {
        echo $item . "<br>";
    }
}

?>
```

Ohjelmakoodi 2. Yksinkertainen ohjelma PHP:lla

Ohjelmassa määritellään kaksi funktiota. Ensimmäinen määrittelee listan, jonka se sitten lähettää toiselle funktiolle. Toinen funktio käy listan läpi for

-loopissa ja tulostaa jokaisen listan alkion. Pythonilla kirjoitettaessa tulee funktion määrittely tehdä ennen kuin funktiota kutsutaan, koska ohjelmaa tulkitaan ajon aikana. PHP kääntää ohjelman ennen ajoa, joten funktion voi kirjoittaa kutsun alle, mikäli niin haluaa.

Pythonissa funktiot rakennetaan eri tavalla, kuin PHP-kielessä. Pythonissa funktiota luodessa sitä ei tarvitse ympäröidä aaltosulkein, vaan funktion määrittelyn jälkeen tulee kirjoittaa kaksoispiste ja funktion sisältö sisentää neljällä välilyönnillä. Pythonia kirjoittaessa ei tarvitse myöskään käyttää puolipistettä rivin päätteeksi. Pythonia kirjoittaessa kommentointi tapahtuu ristikkomerkillä (#), PHP taas käyttää kommentointiin kahta kauttamerkkiä peräkkäin (//).

Jotta PHP-ohjelmakoodin voi ajaa, tarvitsee tietokoneelle jonkinlaisen ympäristön tätä varten. Esimerkiksi Xampp on tehty tähän tarkoitukseen ja sen avulla voi luoda paikallisen webpalvelimen, jolla voi ajaa myös PHP-kieltä.

Python-ohjelmakoodia ei voi myöskään ajaa ilman asiaankuuluvaa ohjelmistoa. Python IDE:n, eli Pythonin kehitysympäristön saa ladattua Pythonin kotisivuilta (Python n.d.)

4 DJANGO-OHJELMISTOKEHYS

Django on ilmainen ohjelmistokehys, jolla voidaan rakentaa nopeasti dynaamisia verkkosivuja. Django on julkaistu BSD-lisenssin alaisuudessa, eli se on vapaasti käytettävissä ja muokattavissa. (Dauzon 2014, 27.) Django on käytössä monella suurella verkkosivulla. Näihin sivuihin lukeutuu esimerkiksi Instagramin verkkosivu, BitBucket, NASA verkkosivut ja The Washington Postin verkkosivut (Bogdanov 2015).

Djangon tavoitteena on vaatimattomasti täydellisyys, ja Django soveltuukin kaikille kehittäjille, jotka haluavat selkeää ja helposti luettavaa koodia. Djangon motto onkin: ”The web framework for perfectionists with deadlines”, eli: ”webohjelmistokehys perfektionisteille, joilla on takarajoja.” (Dauzon 2014, 30.)

4.1 Djangon historiaa

Djangon tarina alkoi Kansasista Yhdysvalloista vuonna 2003, kun Lawrence Journal-World sanomalehden webkehittäjät alkoivat rakentaa Pythonilla sovelluksia. Tätä kehitystiimiä kutsuttiin nimellä The World Online -tiimiksi. Kehittäjät joutuivat ongelmiin, kun he eivät ehtineet kehittää tarpeeksi nopeasti uusia sovelluksia nettisivuille, joita he ylläpitivät.

The World Online -tiimi päätyi ajan ollessa kortilla kehittämään ohjelmistokehyksen, joka vastasi heidän tarpeisiinsa saada kehitettyä uusia sovelluksia nopeammin. Vuonna 2005, kun ohjelmistokehys oli käytössä suurimmalla osalla World Onlinen sivuilla, Django julkaistiin avoimen lähdekoodin alaisena ohjelmana, jazz kitaristi Django Reinhardtin mukaan. (Holovaty, Kaplan-Moss 2007.)

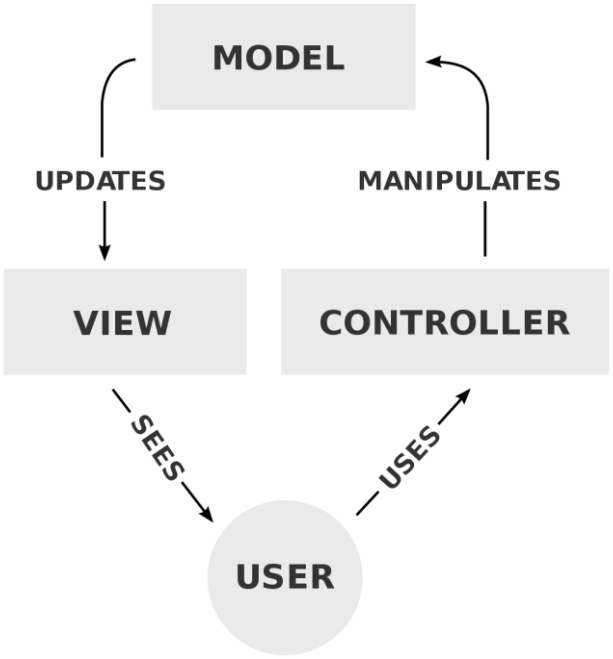
4.2 Mikä on ohjelmistokehys

American Heritage Dictionary kuvaa frameworkia eli kehystä näin: ”Rakennelma, joka tukee tai sisällyttää jotain muuta”. Ohjelmistokehys eli application framework on juuri tällainen kehys, joka tukee sen päälle rakennettavaa ohjelmistoa. Ohjelmistokehityksessä sanaa käytetään kuvaamaan uudelleenkäytettäviä ohjelman osia, jotka auttavat ja nopeuttavat kehitystä.

Ohjelmistokehykset tarjoavat sovellukseen järjestystä ja auttavat organisoimaan koodia. Kun käytetään ohjelmistokehystä, ei tarvitse pitää huolta niin monesta liikkuvasta osasta. Ohjelmistokehystä käytettäessä voidaan rakentaa ohjelma kehyksen päälle, jolloin sovelluksen kehittämiseen käytettävä aika lyhenee merkittävästi. (Chen 2004, 2.)

4.3 **Djangon ominaisuuksia**

Django tukee MVC-mallia, niin kuin useat modernit ohjelmistokehykset. MVC-malli tarkoittaa, model-view-controller -mallia ohjelmistosuunnittelussa. MVC-mallista puhutaan usein käyttöliittymän omaavien ohjelmistojen yhteydessä. MVC-mallin suhteet on havainnollistettu kuvassa 1.



Kuva 1. MVC-malli

Djangossa MVC-malli on nimeltään MVT-malli. Nimenmuutos johtuu siitä, että Django huolehtii kaikesta tiedonsiirrosta model- ja view -osioiden välillä. Sen sijaan Django käyttää template-ominaisuutta, eli ohjelmoija käyttää Pythonia HTML-ohjelmakoodin seassa, kuten ohjelmakoodi 3 esittää.

Templateissa käytetään kahta erilaista merkintätyyliä. Kahdet aaltosulkeet peräkkäin ”{{ muuttuja }}” sisältävät muuttujan. Muuttujien perässä voi olla putkimerkki ”|”, joka tarkoittaa jonkinlaista suodatinta. Prosenttimerkki aaltosulkeen perässä ”{% tag %}” tarkoittaa tagia. Tagiin voi kirjoittaa esimerkiksi loopin, tai tehdä monipuolisempaa funktionaalisuutta. (Tutorialspoint n.d.)

```
{% extends 'base.html' %}
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Product page</title>
  </head>

  <body>

    {% block pagecontent %}

    <div class="table-div">
      <table class="table">
        <thead>
          <tr>
            <th>Product Name</th>
            <th>Price</th>
          </tr>
        </thead>
        <tbody>

          {% for p in products %}
            <tr>
              <td>{{ p.name }}</td>
              <td>{{ p.price }}</td>
            </tr>
          {% endfor %}

        </tbody>
      </table>
    </div>

    {% endblock %}

  </body>
</html>
```

Ohjelmakoodi 3. Esimerkki Djangon templatesta

5 TOIVELISTAN SUUNNITTELU

Tässä opinnäytetyössä kehitettävä toivelistaominaisuus on idealtaan varsin yksinkertainen. Käyttäjä kirjautuu verkkokauppaan, selailee tuotteita ja voi halutessaan tuotesivulta lisätä tuotteen omalle toivelistalleen. Käyttäjä voi myös navigoida itsensä omalle toivelistalleen ja poistaa sieltä tuotteita, lisätä tuotteita ostoskoriin, tai mennä toivelistalla olevien tuotteiden tuotesivuille. Myös teknisesti ottaen idea on varsin yksinkertainen.

Verkkokauppaan ei tehdä ominaisuuksia vain sen takia, että pelkkä verkkokauppa kehittyisi. Taustalla on yleensä asia, kuten vaikka tarve nostaa verkkokaupasta saatavaa liikevaihtoa. Sen takia toivelistan suunnittelua ei ohjaa pelkkä tekninen suunnittelu. Tässä työssä ei kuitenkaan syvennytä näihin liiketaloudellisiin seikkoihin, mitä taustalla on, vaan esitetään pelkästään tekninen toteutus.

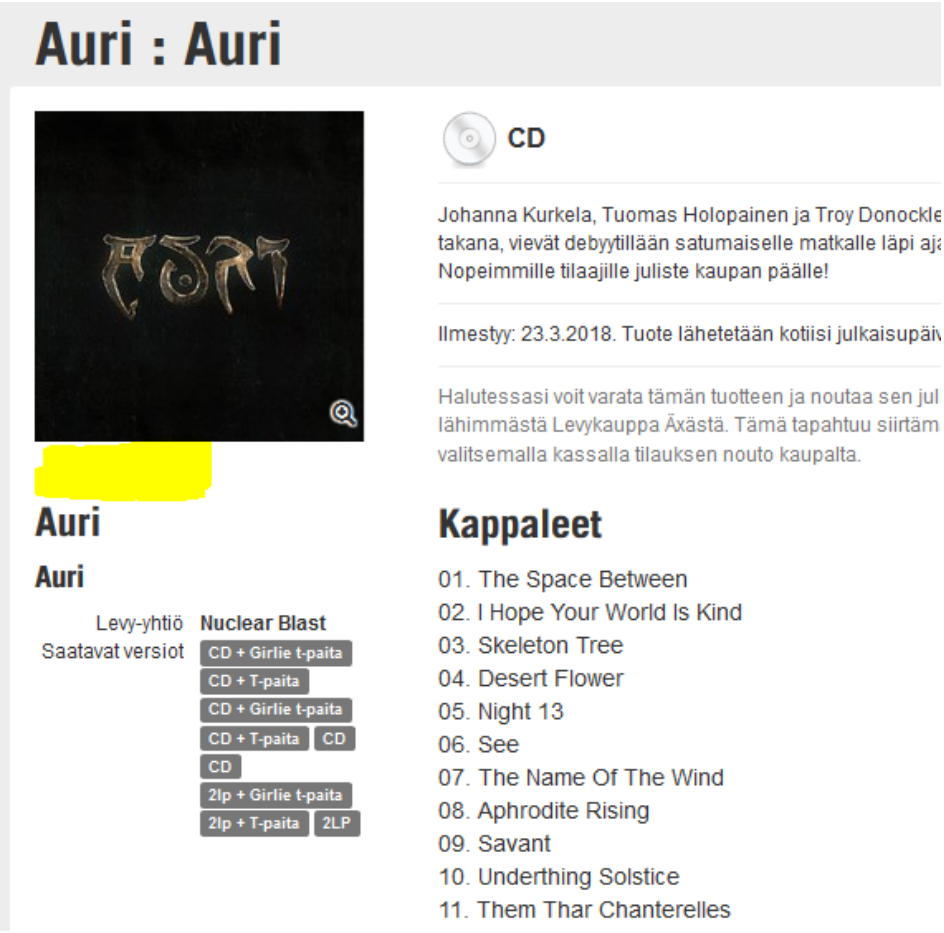
Toivelista suunniteltiin niin, että siinä on normaalit toivelistan ominaisuudet, kuten esimerkiksi tuotteen lisääminen toivelistalle ja tuotteen poisto toivelistalta. Näiden lisäksi piti myös suunnitella, miten toivelista soveltuu Levykaupan liiketoimintaan. Toivelistaominaisuus ei saanut olla sellainen, että se vaikka esimerkiksi vähentäisi myyntiä. Tämän takia piti tehdä benchmarkkausta eri verkkokauppojen toivelistoista, että miten muut ovat toteuttaneet ominaisuuden.

Toivelistan suunnittelussa piti ottaa myös huomioon, miten suunniteltu ominaisuus sijoittuu valmiiseen verkkokauppaan. Huomioon piti ottaa esimerkiksi tyylit, toivelistan listaus ja painikkeiden asettelu tuotesivulle. Tyyleissä piti käyttää jo valmiita tyylejä, jotta toivelista ei erottuisi sivulta erilaisena.

5.1 Tuotesivun suunnittelu

Tuotesivun suunnittelussa tuli ottaa huomioon, mihin toivelistalle-lisäys -painike pitää sijoittaa. Kyseinen painike on myös sama painike, mistä tuote poistetaan listalta.

Kuvassa 2 on suunniteltu sijainti lisäyspainikkeelle. Sijainti on merkattu keltaisella värillä tuotteen kuvan alle. Sijaintiin vaikuttaa se, että se ei saa vaikuttaa esimerkiksi asiakkaan ostopäätökseen. Tämän takia painike ei saisi olla osta-painikkeen vieressä. Painike ei saa myöskään olla liian piilossa, koska asiakkaiden on hyvä nähdä painike ilman isompaa etsimistä.



Kuva 2. Toivelistalle-lisäys -painikkeen suunniteltu lokaatio

5.2 **Käyttötapaukset**

Toivelistan suunnitteluun kuuluu myös käyttötapausten auki kirjoittaminen. Ne ovat muotoa kuka-mitä-miksi. Alla on luettelo alustavista käyttötapauksista.

- Kirjautuneena käyttäjänä minulla tulee olla mahdollisuus lisätä tuote toivelistalle tuotesivulta.
- Kirjautuneena käyttäjänä minulla tulee olla mahdollisuus poistaa tuote toivelistalta, kun olen tuotesivulla.
- Kirjautuneena käyttäjänä minulla tulee olla mahdollisuus poistaa tuote toivelistalta, kun tarkastelen omaa toivelistaani.
- Kirjautuneena käyttäjänä minun tulee voida navigoida toivelistalle.
- Kirjautuneena käyttäjänä minulla tulee olla mahdollisuus valita tuotteita ostoskoriin toivelistaltani.
- Kirjautuneena käyttäjänä minun tulee voida painaa toivelistalta tuotetta, ja päästä sen tuotesivulle.

5.3 Toivelistojen benchmarkkaus

Eri toivelistojen benchmarkkaukseen, eli analysointiin, valittiin neljä verkkokauppaa, joissa kyseinen ominaisuus on toteutettu tavalla tai toisella. Tällä analyysillä voitiin todeta, millä tavalla toivelista on parasta toteuttaa. Benchmarkkauksella haettiin vastauksia kysymyksiin, kuten: ”Mihin kannattaa sijoittaa toivelistalle lisäys -painike?” ja ”Miten toivelistaa käytetään verkkokaupassa?”. Verkkokaupoiksi valittiin Adlibris, Thomann, Prisma ja Zalando.

Verkkokaupat, joita käytettiin benchmarkkaukseen, valittiin satunnaisesti. Ennen benchmarkkausta tiedossa oli kuitenkin, että esimerkiksi Adlibriksen verkkokaupassa on toivelistaominaisuus käytössä ja se valittiin sen takia mukaan.

5.3.1 Adlibris.fi

Adlibriksen verkkokaupassa tuotteen lisääminen toivelistalle on hyvin esillä tuotesivulla. Ostoskoriin lisäämispainikkeen viereen on tehty toinen erivärinen painike, mistä voi lisätä tuotteen toivelistalle. Jos käyttäjä ei ole kirjautuneena sisään, painiketta voi klikata, mutta se vie kirjautumissivulle. Kuva 3 on ruutukaappaus Adlibriksen verkkokaupan tuotesivulta.

Kuva 3. Tuotesivu Adlibriksen verkkokaupassa

Kun Adlibriksen verkkokaupassa lisää tuotteen toivelistalle, muuttuu painikkeessa oleva ”Lisää toivelistalle” -teksti, ”Lisätty toivelistallesi” -tekstiksi. Kyseistä tekstiä klikkaamalla pääsee verkkokaupassa omalle toivelistalle. Kun tuote on lisätty toivelistalle, voi tuotteen poistaa toivelistalta, tai siirtää sen eri toivelistaan.

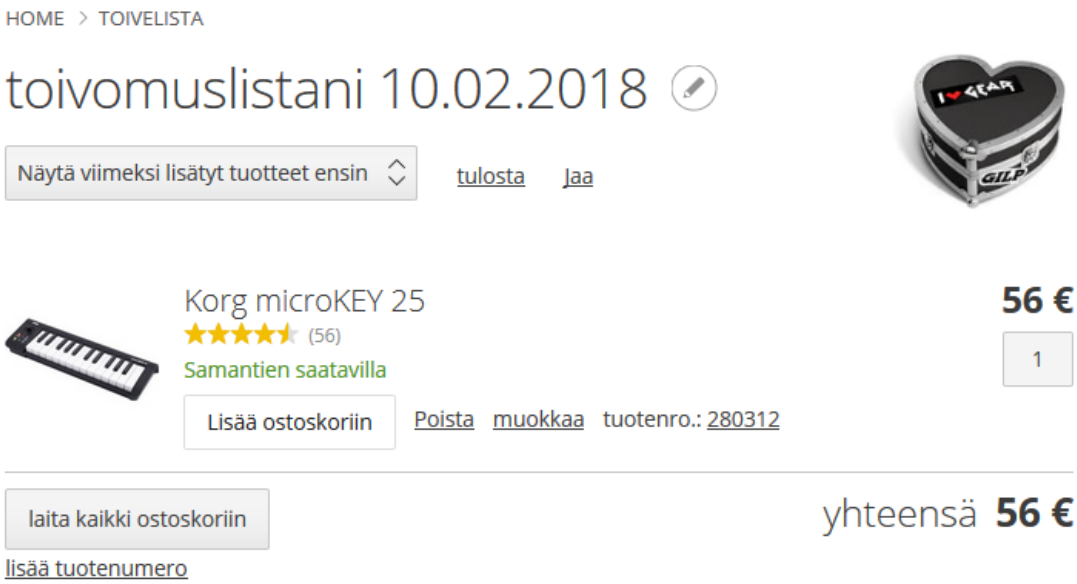
Toivelistalta on myös mahdollisuus lisätä tuotteita ostoskoriin. Tuotteet menevät automaattisesti listalle nimeltä ”Viimeksi lisätyt”, mutta verkkokauppaan voi tehdä muita toivelistoja ja nimetä niitä esimerkiksi eri kategorioihin. Tuote voi olla vain yhdellä toivelistalla samaan aikaan. Tuotetta

poistaessa verkkokauppa kysyy, haluaako tuotteen varmasti poistaa listalta. Verkkokaupassa navigoidaan toivelistalle ”oma tili” -painikkeen kautta.

5.3.2 Thomann.de

Thomann-musiikkiliikkeen verkkokaupassa toivelistalle lisäämispainike on paljon vaatimattomampi, kuin esimerkiksi Adlibriksen verkkokaupassa. Ostoskoriin lisäämispainikkeen alapuolella on tekstimuotoinen linkki, josta tuotteen voi lisätä toivelistalle. Linkin teksti muuttuu ”toivelistallasi” -tekstiksi lisäämisen jälkeen. Thomannin verkkokaupassa voi myös tehdä useampia toivelistoja ja nimetä ne, miten itse haluaa.

Thomannin verkkokaupan toivelista näyttää myös tuotteiden kokonaishinnan. Toivelistassa on myös mahdollisuus laittaa kaikki tuotteet yhdellä klikkauksella ostoskoriin. Toivelistan voi jakaa kenelle tahansa, vaikka Facebookin tai sähköpostin kautta. Toivelista löytyy myös pysyvän linkin takaa. Kuvassa 4 on esitetty Thomannin toivelista.



Kuva 4. Toivelista Thomannin -verkkokaupassa

5.3.3 Prisma.fi

Prisman verkkokaupassa toivelistalle lisäys onnistuu myös erillisen klikattavan linkin kautta ostoskoripainikkeen läheltä. Tuotteita ei voi lisätä toivelistalle, ellei ole kirjautuneena sisälle verkkokauppaan. Jos verkkokaupassa tuote on jo toivelistalla, ei tuotesivulla ole minkäänlaista indikaattoria tästä. Lisää toivelistalle -painike on samassa muodossa, on tuote toivelistalla, tai ei. Jos painiketta painaa uudestaan, tulee aina sama ilmoitus:

”Tuote on lisätty toivelistalle.” Tuote ei kuitenkaan ilmoituksesta huolimatta ilmesty toivelistalle kuin kerran.

Prisman verkkokaupassa omalle toivelistalle pääsee navigoimaan omista tiedoista sivun yläreunasta. Toivelistalta voi lisätä tuotteen ostoskoriin, poistaa sen tai lisätä kaikki tuotteet ostoskoriin. Toivelista kertoo myös kaikkien tuotteiden yhteenlasketun hinnan. Toivelistan tuotteissa näkyy myös päivämäärä, koska tuote on lisätty toivelistalle.

5.3.4 Zalando

Zalando-vaateliikkeen verkkokaupassa on myös käytössä toivelistaominaisuus. Toivelista on näkyvillä heti asiakastili-painikkeen vieressä sivuston yläpalkissa, kuten kuva 5 osoittaa. Toivelistalle lisäys tapahtuu verkkokaupassa tuotesivulta. Toivelistalle lisäys -painikkeen teksti muuttuu lisäyksen jälkeen ”Lisätty” -tekstiksi. Painiketta uudelleenklikkaamalla tuote poistuu toivelistalta ilman mitään erillistä ilmoitusta.

Kuva 5. Zalando-verkkokaupan yläpalkki

Omalle toivelistalle pääsee klikkaamalla yläpalkissa olevaa ”Toivelista”-painiketta. Toivelistassa on kategoriat: kaikki tuotteet, ale, vaatteet, kengät, urheilu, asusteet ja laukut, ja alusvaatteet ja uima-asut. Toivelista lajittelee tuotteet automaattisesti omiin kategorioihin, ja lisäksi tuotteet, jotka ovat alennuksessa menevät ale -kategoriaan. Toivelistan voi jakaa generoidulla linkillä ystäville tai suoraan Facebook, tai Pinterest -palveluihin.

5.3.5 Yhteenveto benchmarkkauksesta

Kaikki toivelistat olivat toimivia ja selkeitä, joskin esimerkiksi Zalandon toivelista oli kaikista selkein ja miellyttävin käyttää. Toivelistoja testatessa myös nousi ominaisuuksia esille, mitkä olivat käyttäjän näkökulmasta parhaita.

Tuotesivulta toivelistalle lisätessä painikkeen on hyvä kertoa, jos tuote on jo omalla toivelistalla. Näin ei esimerkiksi Prisman verkkokaupassa ollut. Parhaiten tämä ominaisuus oli esillä Zalandon verkkokaupassa.

Miellyttävää oli myös se, että toivelistalle pääsi helposti. Kaikissa verkkokaupoissa pääsi toivelistalle helposti suoraan yläpalkin kautta. Tällöin ei listaa tarvitse sen kummemmin etsiä.

Miellyttävää verkkokaupoissa oli myös nähdä toivelistan kokonaishinta ilman niiden lisäämistä ostoskoriin. Zalandossa tätä ominaisuutta ei ollut.

Toivelistoja testatessa myös näki, että joissain verkkokaupoissa on mahdollisuus luoda useampi toivelista, ja järjestää tuotteita esimerkiksi eri kategorioihin. Zalandon verkkokaupassa asia oli toteutettu niin, että on vain yksi toivelista, mutta sitä pystyi suodattamaan esimerkiksi näyttämään pelkästään kenkiä, tai vaikka alennuksessa olevia tuotteita. Alennuksessa olevien tuotteiden esiin nosto toivelistalta on ominaisuus, joka tulisi olla jokaisessa verkkokaupassa.

Toivelistan jako on myös ominaisuus, mikä tulisi toivelistassa olla. Silloin voi esimerkiksi helposti kertoa kavereille tai perheenjäsenille, mitä haluaa lahjaksi.

6 TOIVELISTAN TOTEUTUS

Toivelistan toteutus aloitettiin pystyttämällä kehitysympäristö. Kehitysympäristön pystytys oli vaiheittain raskasta joidenkin sovellusten yhteensopivuusongelmien kanssa. Kehitysympäristöä ei ollut kertaakaan pystytetty Windows työasemalle. Django ympäristöön luotiin myös ensin tyhjä ”wishlist”-sovellus. Sovelluksen luomisen jälkeen jatkettiin verkko-osoitteiden ohjaamisella, eli urls.py tiedoston muokkauksella. Näiden vaiheiden jälkeen oli valmista aloittaa varsinainen ohjelmointi.

Toivelistan ohjelmoinnissa koitettiin pitää mielessä ne raamit, mitä suunnitteluvaiheessa tuli tehtyä. Lisäominaisuutena toivelistalle tehtiin käyttäjälle mahdollisuus nähdä koko toivelistan yhteishinnan.

6.1 Kehitysympäristön pystytys

Ominaisuuden kehitystyö tapahtui Windows-työasemalla ja jotta kaikki ominaisuudet olivat mahdollisia, piti kehitysympäristön kokoonpano aloittaa asentamalla WSL, eli Windows Subsystem for Linux. Työasemalle piti myös asentaa Ubuntu-virtuaalikone Windowsin kaupasta. WSL mahdollistaa Linux-koneissa käytettävän bash-komentorivin käytön, jota vaadittiin kehitysympäristön asennuksessa. Ubuntu-virtuaalikoneen sai käyttöön sen asennuksen jälkeen avaamalla Windowsin komentorivin (cmd) tai PowerShellin ja kirjoittamalla komennon ”bash”. Tätä virtuaalikonetta käytettiin ajamaan Vagrant-järjestelmää, jonka sisällä pyöri toinen virtuaalikone.

Vagrant on tarkoitettu juuri virtuaalikoneiden luontiin kehitysympäristöiksi, ja Vagrantin nettisivuilla kuvataankin: ”Development Environments Made Easy” (Vagrant 2018), eli helpoksi tehdyt kehitysympäristöt. Asennuksen jälkeen Vagrant tarvitsi konfiguraation, jossa määriteltiin esimerkiksi, mikä Linux-version ajetaan, ja mitä kansioita omalta työasemalta sisällytetään virtuaalikoneeseen. Tämä konfiguraatio on nimeltään vagrantfile. Kun virtuaalikoneen konfiguraatio oli valmis, avattiin WSL bash-komentokehote, ja jotta Vagrant voitiin ajaa, piti antaa komento ”export VAGRANT_WSL_ENABLE_WINDOWS_ACCESS="1"”, joka antoi oikeudet Vagrantille käyttää Windowsin ominaisuuksia. Tämän jälkeen navigoitiin kansioon, missä vagrantfile sijaitsee ja annettiin komento ”vagrant up”, joka käynnistää koneen.

Tämän kehitysympäristön asennuksessa automatisoitiin kaikkien tärkeiden sovellusten asennus Ansible-järjestelmällä. Ensimmäistä kertaa ajettaessa ”vagrant up” -komento, Ansible luki asennettavat sovellukset sille tarjotusta tiedostosta ja asensi ne automaattisesti. Ansiblen asennettua sovellukset, Ubuntu kehitysympäristö oli valmis tulevia konfiguraatiota varten. Ansible myös tarkistaa jokaisella virtuaaliympäristön ajokerralla, että tarvittavat sovellukset ovat asennettuina.

Kehitysympäristön ollessa pystyssä, se piti vielä konfiguroida, jotta esimerkiksi pyynnöt selaimelta ohjataan oikeisiin portteihin, ja että tietokanta on oikea ja toimii. Kaikkia konfiguraatioita ei itse kirjoitettu, tai muokattu, vaan jotkut määritykset tarjottiin suoraan tiedostoina. Esimerkiksi Djangon konfiguraatioihin ei tarvinnut käsin koskea, vaan saadut tiedostot voitiin vain kopioida oletuskonfiguraatioiden tilalle.

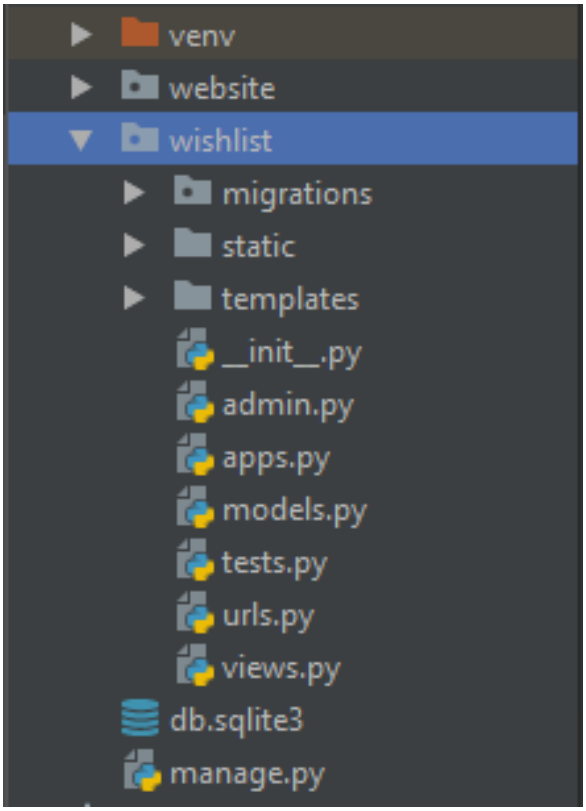
Jotta sivustolle saatiin sisältöä kehitysympäristössä, piti tietokanta laittaa kuntoon palvelimelle. Tietokannan käsittelyssä käytetään PostgreSQL -tietokantajärjestelmää. PostgreSQL on Kalifornian yliopistossa kehitetty avoimen lähdekoodin relaatiotietokantojen hallintajärjestelmä (PostgreSQL 2018). Serverille laitettava tietokanta sai olla isäntäkoneella, eli omalla Windows-työasemalla ja se palautettiin tietokannaksi serverille käyttämällä psql-komentoa, mille annettiin serverikoneen IP-osoite, PostgreSQL:n käyttämä portti, ja sql-tiedoston sijainti isäntäkoneella. Tietokanta oli hyvin suuri ja sen palauttaminen kesti hyvin pitkän ajan.

Http-palvelimena kehitysympäristössä toimii NGINX, joka on ilmainen avoimen lähdekoodin palvelin (NGINX 2018). Http-palvelinta ei tarvinnut muuten konfiguroida, kuin kopioida etukäteen luotu konfiguraatio oletuskonfiguraation päälle. Kopioitu konfiguraatio esimerkiksi määritti, että portista 8000 tulevat kutsut ohjataan porttiin 8001 Djangon käsiteltäviksi.

Lopuksi kehitysympäristöön asennettiin myös Sphinx-hakupalvelin. Sphinx on avoimen lähdekoodin tekstinhakupalvelin, jolla voi hakea dataa sql-tietokannasta. Sphinx hoitaa kaikki käyttäjien tekemät haut. Esimerkiksi, kun asiakas kirjoittaa hakukenttään artistin nimeä, osaa Sphinx jo kirjottaessa hakea tietoa palvelimelta, ja antaa erilaisia ennustavia vaihtoehtoja. Sphinx-palvelimen konfiguraatio hoidettiin myös kopioimalla jo ennalta tehty konfiguraatio palvelimelle, joka määritti mitä tietoja indeksoidaan tietokannasta. Konfiguraation jälkeen piti tiedot vielä indeksoida Python indeksointiskriptillä. Hakukoneen indeksoinnissa kului myös varteenotettavan pitkä aika.

6.2 **Sovelluksen luonti**

Toivelista aloitettiin tekemällä uusi sovellus verkkokauppaan. Djangossa kaikki nettisivun ominaisuudet ovat omissa sovelluksissaan. Sovellukset luodaan käyttäen Djangon manage.py-tiedostoa. Wishlist-sovellus luotiin käyttäen komentorivikomentoa: ”python manage.py startapp wishlist”. Komento annettiin kansion sisällä, missä manage.py -tiedosto sijaitsee. Manage.py-tiedosto on Djangossa sellainen, että sen kautta voi luoda uusia sovelluksia ja hallita nettisivua. Komennon antamisen jälkeen sovelluksesta tuli nettisivun juureen uusi kansio ja sen sisälle Django loi automaattisesti tiedostot: __init__.py, models.py, tests.py ja views.py. Kuva 6 näyttää Djangon kansiorakenteen.



Kuva 6. Djangon kansiorakenne

Sovelluksen luomisen jälkeen sovellus lisättiin Djangon tarjoamaan settings.py -tiedostoon ”INSTALLED_APPS” -kohdan alle. Tämä lisäys kertoo Djangolle, että nettisivulla käytetään wishlist-sovellusta ja nettisivu osaa täten ottaa sovelluksen huomioon muissa sovelluksissa.

Sovellusta pitää voida testata, ja sen takia pitää olla jonkinlainen verkko-osoite sivulla, millä sovellukseen pääsee käsiksi. Djangossa nettisivun juurikansiossa sijaitsee tiedosto *urls.py*. Urls.py tiedostossa määritellään, mihin sovellukseen ohjataan, kun selaimeen kirjoitetaan tietty verkko-osoite. Testitarkoituksessa, tiedostoon lisättiin yksinkertaisesti rivit:

```
# Wishlist
(r'^wishlist/$', include('*.wishlist.urls')),
```

Ensimmäinen rivi on vain kommentti, ja toisella rivillä määritetään, että osoitteeseen *kehitysympäristön-osoite*/wishlist/ tulevat pyynnöt ohjataan wishlist-sovelluksen urls.py -tiedostoon Djangossa. Wishlist-sovelluksen urls.py -tiedosto taas määrittää sen, mihin viewin funktioon pyyntö kulkee. Alla wishlist-sovelluksen urls.py -tiedoston rivit:

```
# Wishlist index
(r'^', '*.wishlist.views.index'),
```

Kommentin jälkeiselle riville kirjoitettiin, että pyynnön tullessa, se ohjataan funktioon ”index”, jossa pyyntö käsitellään.

6.3 **Views**

View on Djangossa se, mikä määrittää mitä dataa lähetetään templatelle. View hoitaa myös kaiken sovelluksen taustalla hoituvan logiikan, pois lukien tietokantojen hallinnan, jotta templatelle jää pelkästään tietojen näyttö käyttäjälle.

Toivelistasovelluksen views.py -tiedostoon rakennettiin neljä eri funktiota. Ensimmäinen rakennettu funktio oli nimeltään index. Index on funktio joka palauttaa tietokannasta kaikki toivelistan rivit kirjautuneelle käyttäjälle ja se toimii silloin kun käyttäjä navigoi omalle toivelistalleen. Index-funktio myös tehtiin laskemaan toivelistan yhteenlaskettu hinta.

Toinen funktio on nimeltään wishlist_add. Nimensä mukaisesti sitä käytetään silloin, kun halutaan lisätä dataa toivelistalle. Kyseinen funktio saa datansa GETin avulla tuotesivulta.

Kolmas funktio mikä rakennettiin, on nimeltään wishlist_remove. Tätä funktiota käytetään myös tuotteiden omilta sivuilta ja se toimii samalla tavalla kuin toivelistalle lisäys, paitsi että se poistaa halutun tuotteen toivelistalta. Funktiota kutsutaan myös, kun käyttäjä klikkaa toivelistalla tuotteen poistoa.

Viimeinen funktio viewseissä on nimeltään is_product_in_wishlist. Funktio ottaa vastaan request-objektin ja tuotteen id:n. Request-objektissa kulkee mukana käyttäjän tiedot ja tuotteiden hakemisessa toivelistalta tarvitsee tietää kuka on kirjautuneena. Funktiota kutsutaan vain toivelistan views.py tiedoston sisällä ja se palauttaa true tai false -arvon riippuen, onko käyttäjällä jo kyseinen tuote toivelistalla.

6.4 **Tietokanta ja models.py**

Jotta Wishlist-sovellukseen saatiin omat tietokantataulut, piti models.py -tiedostoon luoda Wishlist-luokka. Wishlist-sovelluksen tietokanta suunniteltiin niin, että se pitää sisällään rivit yksilöivän id:n lisäksi: käyttäjän oman tunnisteavaimen, wishlistin id:n sekä viiteavaimen tuotteen tauluun.

Käyttäjän id:lle määriteltiin oma kenttä tietokantaan, koska tietoja haetaan käyttäjän perusteella. Jokaisella kirjautuneella käyttäjällä on oma toivelista, ja jokainen rivi tietokannassa vastaa yhtä tuotetta yhdellä käyttäjällä.

Tietokannan toinen sarake varattiin listan id:lle. Tietokantaa suunnitellessa otettiin huomioon, että tulevaisuudessa käyttäjälle annetaan oikeus luoda useampia toivelistoja. Listan id määritettiin luvuksi, jonka voi myöhemmin sitoa kiinni esimerkiksi käyttäjän itse nimeämään toivelistaan.

Tuote merkittiin viiteavaimeksi, koska Djangossa on ominaisuus hakea relaatioiden avulla tietoa toisista tauluista. Viiteavain mahdollisti tuotteen tietojen haun toisesta taulusta, joka pitää sisällään kaikki tiedot tuotteesta. Tietoja, mitä tarvittiin tuotteen näyttämiseen toivelistassa, oli esimerkiksi: tuotteen nimi, artisti, hinta ja tallenteen tyyppi.

Tietoja tallennettaessa viiteavaimella, tulee kenttään tietokannassa viitatun datan id. Kun tietoa tallennetaan, Django tekee automaattisesti viitetarkistuksen, onko tietoa olemassa id:llä.

6.5 **Templatet**

Templateen tulee Djangossa kaikki se, minkä käyttäjä nettisivulla näkee. Template saa kaiken datansa viewistä.

Toivelistan template suunniteltiin niin, että tulokset toivelistalla on muotoiltu samalla tavalla, kuten hakutulokset verkkokaupassa. Tällä menettelytavalla ei tarvitse kirjoittaa turhaa ohjelmakoodia toivelistan omaan templateen, vaan voi käyttää jo valmiina olevia templateja. Valmiina olevien templateiden käyttö luo myös selkeämpää ja yhtenäisempää ohjelmakoodia.

Jotta toivelistan oma sivu saatiin näyttämään yhtenäiseltä muun verkkokaupan kanssa, käytettiin tähän tarkoitukseen verkkokaupan vakiotemplatea, joka sisällyttää esimerkiksi headerin ja footerin. Tällä tavoin voitiin kirjoittaa templateen vain pelkästään toivelistaan tarvittavat ominaisuudet, eikä toistoa syntynyt.

Toivelistan template sisältää for-loopin, joka käy kaikki toivelistan tuotteet läpi ja tulostaa ne selkeästi käyttäjälle. Tuotteissa näkyy ovatko ne esimerkiksi CD levyjä, LP levyjä tai vaikka paitoja. Tuotteissa näkyy myös hinnat, ja jos tuotteen hinta on alennettu, näkyy alennettu hinta yliviivattuna.

6.6 **Tuotesivu**

Verkkokaupan tuotesivulle piti tehdä muutoksia, jotta toivelistalle lisäys -painike näkyisi. Tuotesivulla oleva painike tehtiin näkyväksi vain kirjautuneille käyttäjille ja vain kirjautuneet käyttäjät voivat lisätä tuotteita toivelistalle.

Tuotesivulle piti tehdä kysely toivelistan funktioon, joka kertoo, onko kyseinen tuote jo toivelistalla. Mikäli tuote on jo toivelistalla, näyttää tuotesivu eri tekstin painikkeessa.

Toivelistalle lisäys -painike tehtiin lisäämään tuote toivelistalle tai poistamaan tuote toivelistalta, riippuen onko tuote jo toivelistalla. Painikkeen tyylit lainattiin vahvasti ostoskoripainikkeesta.

6.7 **Toivelistan logiikka**

Toivelistan logiikka on lähtökohtaisesti yksinkertainen: käyttäjä valitsee tuotteen, lisää sen toivelistalleen yhdellä klikkauksella, navigoi toivelistalle, valitsee tuotteen ostoskoriin, tai poistaa tuotteen toivelistalta. Kaikki tämä logiikka toteutettiin views.py -tiedostoon.

Tuotetta lisätessä toivelistalle, painikkeen klikkaus lähettää pyynnön lisätä kyseinen tuote toivelistalle. Toivelistalle lisäys on kirjoitettu wishlist_add-funktioon. Funktio ottaa vastaan GET-metodilla tuotteen id:n ja verkko-osoitteen, johon käyttäjä ohjataan takaisin. Tuotteen id ja verkko-osoite saadaan valmiissa muodossaan edelliseltä templatelta, johon ne on lähetetty aikaisemmin. Jotta toivelistalle lisäys onnistuu, tulee verkko-osoitteen olla muotoa: ”https://*verkko-osoit.e*/wishlist/add/?product=**tuotenro**?back=**paluuverkko-osoite**”, tai vaihtoehtoisesti ilman verkko-osoitetta, johon palata. Mikäli paluuosoitetta ei ole annettu, ohjaa toivelistalle lisäys käyttäjän etusivulle. Wishlist_add-funktio tarkistaa ennen lisäystä, että onko lisättävä tuote jo toivelistalla, jotta toistuvia tuotteita ei ilmaannu. Jos tarkistukset menevät läpi, niin tuote lisätään toivelistalle ja funktio ohjaa käyttäjän takaisin tuotesivulle, tai etusivulle.

Toivelistalle navigoidessa käyttäjälle aukeaa toivelistan etusivu, eli kaikki tuotteet, mitkä toivelistalla ovat. Tuotteiden näyttämisen hoitaa index-funktio. Index-funktio luo listan, johon se iteroi toivelistan datan tietokannasta, jotta turhista objekteista päästään eroon ennen datan lähettämistä templatelle. Alkuperäinen objekteja sisältävä lista saadaan tietokantahaulla, joka etsii vierasavainten perusteella tuotteiden tiedot kannasta. Tämä on mahdollista, koska toivelistan models.py -tiedostossa on määritelty tuote toivelistan vierasavaimeksi. Haku myös suodatetaan käyttäjän id:llä, jotta saadaan vain kirjautuneen käyttäjän data kannasta. Ennen kuin toivelista lähetetään templatelle, lasketaan vielä funktiossa toivelistan kokonaishinta, jotta sitä ei tarvitse tehdä templatessa, pitäen ohjelmakoodin selkeänä.

7 YHTEENVETO

Levykauppa Äx on yksi Suomen suurimpia levykauppoja ja heillä on tarkoitus kehittää verkkokauppatoimintaa paremmaksi. Verkkokaupan kehitys ei ole pelkkää ohjelmointia. Kehitykseen kuuluu paljon suunnittelua ja liiketoiminnallisesta näkökulmasta ajattelua.

Opinnäytetyössä oli tarkoituksena näyttää ja toteuttaa toivelistaominaisuus Levykauppa Äxän jo valmiiseen verkkokauppaan. Tarkoituksena oli myös näyttää, miten Djangossa luodaan sovellus ja sen lisäys verkkosivuille.

Toivelista saatiin onnistuneesti valmiiksi niiden vaatimusten mukaisesti, mitä etukäteen oli annettu. Toivelistan oli tarkoitus olla teknisesti toimiva. Toivelistaominaisuuteen ei vielä opinnäytetyössä tehty esimerkiksi ominaisuutta, jossa asiakkaaseen otettaisiin yhteyttä sähköpostilla koskien toivelistan tuotteita.

Opinnäytetyössä oli haastavaa päästä mukaan vanhaan verkkokauppaan ja sen vanhan ohjelmakoodin tulkitseminen oli välillä haastavaa. Varsinainen ohjelmoiminen sujui pienistä logiikkaongelmista huolimatta varsin mallikkaasti.

Opinnäytetyössä tuli opittua esimerkiksi, miten toivelistaominaisuus toteutetaan, Django ja Python tulivat myös enemmän tutuiksi. Työn aikana opin myös kommunikaatiota ryhmän kanssa.

LÄHTEET

Bogdanov, V. (2015). Top 10 sites build with Django framework. Haettu 27.1.2018 osoitteesta https://www.linkedin.com/pulse/top-10-sites-built-django-framework-vladimir-bogdanov

Chen, X. (2004). *Application Frameworks in .NET.* Berkley: aPress.

Chun, W. (2000). *Core Python Programming.* 1. painos. Prentice Hall.

Codeup. (2014) Front-End Vs. Back-End | Codeup Career Accelerator. Haettu 5.2.2018 osoitteesta https://codeup.com/front-end-vs-back-end/

Dauzon, S. (2014). *Django Essentials.* Birmingham: Packt Publishing.

Hallavo, J. (2013). *Verkkokaupan rautaisannos.* Helsinki: Talentum.

Holovaty, Kaplan-Moss. (2009). *The Definitive Guide to Django: Web Development Done Right.* Apress.

Järvenpää, L. (n.d.) Verkkokaupparatkaisut – Digitalisoinnin opas. Haettu 5.2.2018 osoitteesta https://www.itewiki.fi/opas/verkkokaupparatkaisut/

NGINX. (2018). Welcome to NGINX wiki! Haettu 25.1.2018 osoitteesta https://www.nginx.com/resources/wiki/

PostgreSQL. (2018). PostgreSQL: Documentation: 10: 1. What is PostgreSQL? Haettu 25.1.2018 osoitteesta https://www.postgresql.org/docs/10/static/intro-whatis.html

Pyramid. (2018). Welcome to Pyramid, a Python Web Framework. Haettu 29.1.2018 osoitteesta https://trypyramid.com/

Python. (2018). Applications for Python. Haettu 29.1.2018 osoitteesta https://www.python.org/about/apps/

Python wiki. (n.d.). PythonVsPhp – Python wiki. Haettu 5.2.2018 osoitteesta https://wiki.python.org/moin/PythonVsPhp

van Rossum, G. (2009). Introduction and overview. Python history blogspot. Haettu 18.1.2018 osoitteesta http://python-history.blogspot.fi/2009/01/introduction-and-overview.html

van Rossum, G. (2009). Personal history part 1. Python history blogspot. Haettu 19.1.2018 osoitteesta http://python-history.blogspot.fi/2009/01/personal-history-part-1-cwi.html

van Rossum, G. (2009). Brief timeline of Python. Python history blogspot. Haettu 19.1.2018 osoitteesta http://python-history.blogspot.fi/2009/01/brief-timeline-of-python.html

Ruotsalainen, I., Närhi, M. & Juntunen, P. (2010). *Johdanto verkkokaupaan.* Haettu 5.2.2018 osoitteesta http://www.hameenuusyrityskeskus.fi/img/file.php?id=20698

Sphinx. (2018). About | Sphinx. Haettu 25.1.2018 osoitteesta http://sphinxsearch.com/about/sphinx/

TIOBE. (2018). TIOBE index. Haettu 19.1.2018 osoitteesta https://www.tiobe.com/tiobe-index/

Tutorialspoint. Django Overview. Haettu 5.2.2018 osoitteesta https://www.tutorialspoint.com/django/django_overview.htm

Vagrant. (2018). Vagrant by HashiCorp. Haettu 24.1.2018 osoitteesta https://www.vagrantup.com/