KARELIA-AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

Teemu Airaksinen

TUOTESIVUSTON KEHITTÄMINEN LVI AIRAKSINEN TMI:LLE

Opinnäytetyö
Toukokuu 2015

Karelia AMMATTIKORKEAKOULU	**OPINNÄYTETYÖ** **Toukokuu 2015** **Tietotekniikan koulutusohjelma** Karjalankatu 3 80200 JOENSUU +358 50 260 6800

Tekijä(t)
Teemu Airaksinen

Nimeke
Tuotesivuston kehittäminen LVI Airaksinen TMI:lle

Toimeksiantaja
LVI Airaksinen

Tiivistelmä

Tämän työn tarkoituksena oli suunnitella ja kehittää tuotesivusto sillä tavalla, että se on myöhemmässä vaiheessa mahdollista päivittää verkkokaupaksi. Tässä opinnäytetyössä käydään läpi sivuston toiminnalliset toteutusvaiheet ja niissä käytetyt tekniikat. Lopuksi tarkastellaan työn tulokset ja lisäksi mietitään jatkokehitysmenetelmiä, jossa tulee myös ilmi toimeksiantajalta tulleet ideat.

Opinnäytetyö tehtiin toimeksiantona Pieksämäellä toimivalle yksityiselle elinkeinonharjoittajalle nimeltä LVI Airaksinen. Yrittäjällä ei ole internetsivuja, joten opinnäytetyön tekijälle tuli tehtäväksi tehdä tuotesivuston lisäksi karkea malli etusivusta. Etusivun on tarkoitus kertoa yrityksen toiminnasta ja yleistä tietoa yrityksestä.

Opinnäytetyö on täysin toiminnallinen ja suoritettu ilman varsinaista aikataulua. Toimeksiantajalta ei tullut mitään varsinaista määritettyä takarajaa työn valmistumiselle. Työn haluttiin kuitenkin valmistuvan mahdollisimman nopeasti, että tuotesivustoa päästäisiin jatkokehittämään verkkokaupaksi.

Kieli Suomi	Sivuja 26

Asiasanat

tuotesivusto, data access layer, vaatimusmäärittely

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
May 2015
Degree Programme in Information Technology
Karjalankatu 3
80200 JOENSUU
FINLAND
+358 50 260 6800

Author (s)
Teemu Airaksinen

Title
Developing Product Site for LVI-Airaksinen

Commissioned by
LVI Airaksinen

Abstract

The purpose of this thesis was to design and develop a product website that can be updated at a later point to online store. This thesis introduces the functional implementation steps and the techniques which were used in the implementation. Finally, the results of the work and the further development of methods and the entrepreneur's ideas are considered.

This thesis was commissioned by LVI Airaksinen. The entrepreneur does not have internet pages so the author of this thesis was supposed to do the product site as well as a rough model for a front page. The front page is supposed to describe the company's operations and provide general information.

The thesis was made as functional study and completed without an actual timetable. The entrepreneur did not set a deadline for the completion of thesis. However, the entrepreneur wanted this thesis to be completed as soon as possible because they want soon to get to work with the online store.

Language
Finnish

Pages 26

Keywords

product site, data access layer, demand specification

Sisältö

1 Johdanto 5
2 Opinnäytetyön lähtökohdat 5
3 Verkkosivuston vaatimusmäärittely 6
3.1 Toiminnalliset vaatimukset 6
3.2 Sivuston ulkoasu 7
3.3 Tietokanta 8
4 Ohjelmistot ja tekniikat 8
4.1 Microsoft Visual Studio Professional 2013 9
4.2 ASP.NET Web Forms 9
4.3 HTML5 10
4.4 CSS3 11
4.5 C# 12
4.6 Entity Framework 12
4.7 SQL Server 2012 12
5 Verkkosivuston toteutus 13
5.1 Data Access Layer 13
5.1.1 Tuotteet 14
5.1.2 Kategoria 15
5.1.3 TuoteKonteksti 15
5.1.4 Tietokannan alustaminen 16
5.2 Etusivu 18
5.3 Tietojen näyttäminen tietokannasta 19
5.3.1 Tuotesivu 19
5.3.2 Kategoria 20
5.3.3 Yksittäisen tuotteen näkymä 21
5.4 Tavoitteet ja aikataulu 22
6 Tulokset 23
7 Pohdinta 24
7.1 Omia mietteitä 24
7.2 Jatkokehittämisideat 25
Lähteet 26

1 Johdanto

Opinnäytetyön tarkoituksena oli suunnitella ja kehittää ASP.NET Web Forms -tekniikan avulla tuotesivusto, joka on mahdollista päivittää myöhemmässä vaiheessa verkkokaupaksi. Toimeksiantajana toimi Pieksämäellä toimiva yksityinen elinkeinonharjoittaja nimeltä LVI Airaksinen.

Internet-sivusto oli yrityksen ensimmäinen, joten tuotesivuston lisäksi kehitettiin etusivu, jossa tarkoituksena oli kertoa yrityksestä perustietoja. Tuotesivuston tuli olla mahdollisimman helppokäyttöinen ja ymmärrettävä, koska myöhemmin jatkokehityksen tuloksena syntyvän verkkokaupan käyttäjäkunta tulee olemaan ikähaarukaltaan laaja. Lisäksi tuotesivustolle kehitettiin toiminnallisuus, jonka ansioista käyttäjä pystyy rajaamaan tuotteet tarvittaessa kategorian mukaan esimerkiksi kylpyhuonekalusteisiin.

Tässä opinnäytetyön raportissa käydään läpi työn toiminnalliset vaatimukset ja toteutuksen eri vaiheet sekä perehdytään opinnäytetyössä käytettyihin tekniikoihin.

2 Opinnäytetyön lähtökohdat

Toimeksiantajana on Pieksämäellä toimiva yksityinen elinkeinonharjoittaja LVI Airaksinen. Yritys perustettiin vuonna 2009 ja sen omistaa Timo Airaksinen. Yrityksellä ei ole tällä hetkellä työntekijöitä palkattuna. Toimiala käsittää kaikki LVI-alan työt: lämpö-, vesi- ja ilmastointityöt, jotka sisältävät kaikki uudet energiaratkaisut, kuten maa- ja aurinkolämpö sekä ilma- ja vesilämmitys. Yritys hoitaa myös tarvittaessa tarvikemyynnin rakennuskohteisiin. Tavoitteena on tulevaisuudessa työllistää 1-10 työntekijää ja saada toimiva verkkokauppa muun urakoinnin ohelle. Yrityksen strategia on pyrkiä kasvamaan työtilanteen parantuessa suuremmaksi yritykseksi ja pysyä alan kehityksessä mukana.

Toimeksiantaja oli miettinyt LVI-tarvikkeiden myyntiä internetin välityksellä jo pitemmän aikaa. Ennen opinnäytetyön aloittamista käytiin toimeksiantajan kanssa läpi, miltä sivuston pitäisi ulkoasullisesti näyttää. Opinnäytetyö tehtiin täysin toimintapainoitteisesti.

Työn tekemisessä käytettiin apuna Microsoftin teknisiä opetussivustoja ja hyväksi todettuja asianmukaisia ja olennaisia lähteitä. Opinnäytetyö pystyttiin tekemään opinnäytetyön tekijän omissa tiloissa, ja opinnäytetyölle aikataulua ei varsinaisesti määritetty toimeksiantajan puolesta, mutta oletuksena oli, että työhön asetetut tavoitteet täyttyisivät.

3 Verkkosivuston vaatimusmäärittely

Verkkosivuston vaatimusmäärittelyssä kuvataan verkkosivuston toteutuksen toiminnallisia vaatimuksia ja tavoitteita. Lisäksi käydään läpi sivuston ja tietokannan suunnittelu sekä perehdytään siihen, mitä niitten suunnittelussa täytyy ottaa huomioon.

3.1 Toiminnalliset vaatimukset

Toiminnallisilla vaatimuksilla kuvataan, miten verkkosivuston tulisi toimia käyttäjän näkökulmasta. Verkkosivuston toiminnalliset vaatimukset ovat seuraavat:

- Käyttäjä voi nähdä kaikki tuotteet kerralla.
- Käyttäjä voi valita tuoteryhmästä haluamansa tuotteet näkyville.
- Käyttäjä voi tarkastella haluamansa tuotteen tietoja.
- Käyttäjä voi rekisteröityä käyttäjäksi.
- Käyttäjä voi kirjautua sisään.
- Käyttäjä voi muuttaa käyttäjätilinsä salasanan.
- Käyttäjä voi etsiä tuotteita käyttäen hakutoimintoa.
- Käyttäjä voi lisätä tuotteita ostoskoriin.

- Käyttäjä voi nähdä ja valita yksittäisen tuotteen näkymässä siihen tuotteeseen liittyvät muut tuotteet.
- Ylläpitäjä voi hallita tuotteita käyttäjätilinsä kautta hallintapaneelissa.
- Käyttäjä voi lähettää tarjouspyynnön.

3.2 Sivuston ulkoasu

Sivuston ulkoasu suunniteltiin siten, että jokainen sivu näyttää mahdollisimman samanlaiselta. Tämän ansiosta sivustosta saatiin käyttäjän kannalta helposti ymmärrettävä ja selkeä, koska sivuston jokainen osa löytyy jokaiselta sivulta samasta kohdasta (kuva 1).

Kuva 1. Sivun ulkoasu.

Sivuston ulkoasuista voi suunitella erilaisia, mutta tässä työssä siitä suunniteltiin ulkoasultaan sellainen, josta opinnäytetyön tekijällä on aikaisempaa kokemusta. Tämä edesauttaa erityisesti sivuston vanhempaa käyttäjäkuntaa ymmärtämään nopeasti sivuston toiminnallisuuden, kun monilla verkkokaupoilla on samantyylinen ulkoasu.

Kuvasta 1 huomataan, että sivun ulkoasu sisältää ylänavigoinnin, ylätunnisteen (logo), sivunavigoinnin vasemmassa reunassa ja sisältöalueen sekä alatunnisteen.

Verkkosivuston ylänavigointiin sijoitettiin linkki tuotesivulle, jossa sijaitsevat projektissa syntyneet linkit. Ylätunnisteeseen asetettiin tässä tapauksessa yrityksen nimi, johon on mahdollista sijoitaa myöhemmin myös yrityksen logo. Sivunavigointiin sijoitettiin navigointilinkit eri tuotteisiin, joiden sisältö aukeaa sisältöalueeseen niin kuin kaikki sivuston sisällöt. Alatunnisteeseen laitettiin nyt aluksi yrityksen nimi ja vuosiluku.

3.3 Tietokanta

Tietokannan suunnittelussa on tärkeää, että tietokannasta pystyy vaivattomasti hakemaan sekä myös lisäämään että muokkaamaan tietoja. Tietokantaa alettiin suunnittelemaan ensimmäiseksi miettimällä, mitä kaikkea tietoja tietokannasta tullaan hakemaan. Tietokannan muutoksissa ainoastaan tietokannan taulujen rivit muuttuvat, joten esimerkiksi kun tauluun lisätään tietoja niin tauluun lisääntyy riviä yhtä tietoa kohden. (Laaksonen 2009)

Lisäksi kun tästä opinnäytetyöstä jatkokehitetään myöhemmässä vaiheessa verkkokauppa niin tietokannan suunnittelussa tärkeänä pidettiin sitä, että yksi tietokannan kenttä saa sisältää ainoastaan yhden tiedon. Yksi tietokannan kenttä kun saa sisältää ainoastaan yhden tiedon niin tällä tavoin saadaan ostoskorin sisällöstä järkevämmän näköinen ja helposti luettavampi koska jokainen tuote sijoitetaan omalle rivilleen (Laaksonen 2009).

4 Ohjelmistot ja tekniikat

Tuotesivuston kehittämisessä käytettiin Microsoftin tarjoamaa Visual Studio Professional 2013 -kehitysympäristöä, joka on aikaisempien kokemuksien

perusteella havaittu hyväksi. Kehittämisessä tarvittiin monia eri tekniikoita kuten HTML5, CSS3, C#, Entity Framework ja SQL Server 2012.

4.1 Microsoft Visual Studio Professional 2013

Visual Studio on monipuolinen Microsoftin kehittämä ohjelmankehitysympäristö, jolla voi kehittää sovelluksia esimerkiksi Windows-puhelimille, työpöytä-, web- tai pilviympäristöön. Visual Studiossa voi käyttää monia eri kieliä kuten esimerkiksi C#, VB, C++ ja sisältää myös täyden tuen JavaScriptille ja jQuerylle web-kehitystä varten. (Microsoft 2014)

4.2 ASP.NET Web Forms

ASP.NET Web Forms on osa Microsoftin kehittämää web-ohjelmistokehystä. Web Forms hyödyntää tapahtumapohjaista mallia käyttäen tutuksi tullutta vedä ja pudota (Drag and Drop) -tekniikkaa, jonka avulla voidaan rakentaa dynaamisia verkkosivustoja. (Microsoft 2015)

ASP.NET Web Forms sovellusta rakentaessa Data Access –luokat sijoitetaan projektissa olevaan Models kansioon. Verkkolomaketta lisättäessä (Web Forms) syntyy kaksi osaa: visuaaliseen puoleen perustuva ASPX-tiedosto ja ASPX.CS -tiedosto, joka hoitaa sovellukseen kuuluvan koodin. Eli tässä nähdään Web Forms arkkitehtuurin rakenne, kun visuaaliselle, sovellukselle ja tietokannalle on määritetty omat tiedostot/paikat koodia varten.

Web Formsin vahvuuksia:

- Yhteensopiva monien eri Microsoftin kehittämien kielin kanssa, kuten Visual Basic ja C#.
- Tapahtumapohjainen ohjelmointimalli on tuttu Microsofin Visual Basic ohjelmoijille.
- HTML rajapinta erotettu sovelluslogiikasta.
- Tarjoaa .NET Frameworkin kaikki hyödyt kuten turvallisuuden ja perintämahdollisuuden. (Microsoft 2015)

ASP .NET sovellukset ovat selainriippumattomia ja server-kontrollien käyttö sekä tekeminen mahdollistavat myös selaimen erityispiirteiden käyttämisen ilman että sovelluskoodia käsitellään erikseen. (Microsoft 2015)

4.3 HTML5

HTML tulee sanoista Hypertext Markup Language ja suomennettuna se tarkoittaa hypertekstin merkintäkieltä. Tällä hetkellä HTML-kielen uusin versio on HTML5 ja sen ansiosta lisäohjelmien tarve vähenee. (Marshall 2011)

HTML-dokumentti on tekstimuotoinen tiedosto, joka sisältää sivun rakenteen kertovan koodauksen, joita ovat esimerkiksi kappaleet, otsikot, hyperlinkit, kuvat ja listat. Jokaisessa HTML-dokumentissa käytetään HTML-kielen omia ohjausmerkkejä, joita kutsutaan tageiksi. Esimerkiksi koko dokumentin aloitustag on <html> ja samalla koko dokumentti lopetetaan lopetustagiin, joka on </html>. Kyseisten tagien sisälle tulee kaikki muut elementit (kuva 2).

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Dokumentit otsikko</title>
</head>

<body>
        Dokumentin sisältö..
</body>

</html>
```

Kuva 2. HTML5-dokumentin runko.

HTML5 on yhteensopiva vanhempien versioiden kanssa ja se sisältää monia ominaisuuksia helpottamaan sivustojen luomista ja parantamaan sivustojen käyttökokemusta.

Uusia semaattisia elementtejä ovat esimerkiksi:

- <article> määrittelee artikkelin dokumentissa.
- <section> määrittelee erilaisia osioita dokumenttiin.
- <header> määrittelee ylätunnisteen dokumenttiin tai sectioniin.
- <footer> määrittelee sivun alatunnisteen.
- <nav> määrittelee alueen, jossa sijaitsevat navigointilinkit. (Aitken 2015)

Myös uusia mediaelementtejä ja grafiikkaan liittyviä elementtejä tuli HTML5-kielen mukana, joita ovat esimerkiksi:

- <audio> määrittelee ääni tai musiikki sisällön.
- <video> määrittelee video tai elokuva sisällön.
- <canvas>, joka liittyy grafiikkapuoleen, määrittää graafisen piirtämisen käyttäen JavaScriptia. (Aitken 2015)

4.4 CSS3

CSS on lyhenne englannin kielen sanoista Cascading Style Sheets, joka on erityisesti web-sovelluskehityksessä käytetty kuvaamaan yleisemmin HTML tai XHTML sivustojen muotoiluja ja ulkoasuja. CSS3 tarkoittaa yksinkertaisesti sen olevan uusin versio. CSS3 tarjoaa mahdollisuuden rakentaa web-sivuista puhtaampia, joka latautuu aikaisempaa versiota nopeammin. CSS3 tarjoaa myös paljon sisältöä kuten esimerkiksi visuaalisia tehosteita ja graafisia elementtejä, joita ovat muun muassa varjot ja kaltevuudet. (Bookwalter 2011)

CSS3-tiedoston linkittäminen HTML-tiedostoon tapahtuu HTML-tiedoston head-elementin sisällä link-elementin avulla, jossa määritetään CSS3-tiedoston sijainti. CSS3-tiedostossa määritetään sen HTML-elementin ulkoasua, jolle on määritetty ID- tai luokka-attribuutti. ID-attribuuttiin viitataan CSS3-tiedostossa laittamalla #-merkki ja nimi, jonka on antanut HTML-tiedostossa, kun taas luokka-attribuuttiin viitataan laittamalla nimen eteen piste (kuva 3).

```
#para {
     text-align: center;
     color: red;
}

.para1 {
     text-align: center;
     color: red;
}
```

Kuva 3. CSS3-tiedostossa tyylimäärittelyä.

4.5 C#

C#, joka lausutaan C Sharp, joka on suunniteltu olemaan yksinkertainen ja moderni olio-ohjelmointikieli Microsoftin .NET -alustalle. C#'n tarvoitteena on helpottaa tiedonvaihtoa ja palveluita internetin yli. Microsoftin kehittämä C# -kieli pyrkii yhdistämään C++ -kielen laskentatehon ja Visual Basicin helppouden. (Rouse 2007)

4.6 Entity Framework

Entity Framework on Microsoftin kehittämä avoimeen lähdekoodiin perustuva O/RM -työkalu (Object / Relational mapping), joka on tarkoitettu tietokantojen ohjelmointiin. Entity Framework mahdollistaa myös tietokantojen käsittelyn oliomaisesti. Työkalu helpottaa paljon, sillä sen avulla voidaan sekä hakea tietokannasta tietoja että tehdä muutoksia niihin tai jopa poistaa. (Entity Framework Tutorial 2015)

4.7 SQL Server 2012

Microsoftin kehittämä SQL Server 2012 on hallintajärjestelmä relaatiotietokannalla. Tässä opinnäytetyössä käytettiin SQL Serverin LocalDB ympäristöä, joka on SQL Server Express versiosta kevyempi käyttäen vähemmän muistia. LocalDB käyttää samoja ominaisuuksia kuin SQL Server Express. LocalDB on SQL Server Expressenin suoritustila. LocalDB tulee

Visual Studio 11 tai sitä uudemman mukana, joten sitä ei tarvitse erikseen asentaa. (Massi 2012)

5 Verkkosivuston toteutus

Tässä luvussa perehdytään siihen, miten verkkosivuston toiminnalliset vaatimukset toteutettiin teknisesti ja miten päädyttiin toteutuksessa käytettyihin ratkaisuihin. Lopuksi perehdytään aikatauluun ja siihen, mitkä olivat opinnäytetyölle asetetut tavoitteet.

5.1 Data Access Layer

Tässä alaluvussa käydään läpi, mikä on Data Access Layer, ja miten se toteutettiin.

Data Access Layer on sellaisen tietokoneohjelman osa, joka helpottaa tiedon pääsyä ohjelman ja pysyvän muistin eli esimerkiksi kiintolevyn välillä. Data Access Layer'n tehtävänä on varmistaa tiedon pääsy ohjelmalle heti, kun ohjelma sitä tarvitsee. (Adams 2015)

Nykyisen trendin mukaan tiedon pääsy (Data Access) –koodi on erotettu muusta koodista. Data Access'lla tarkoitetaan, että se voi päästä muistiin käsiksi eli voi kirjoittaa sekä lukea tietoa muistiin. (Patton 2006)

Data Access Layer kuuluu liiketoimintakerrokseen, joka toimii rajapintana tietokerroksen ja esityskerroksen välillä. Jokaisella kerroksella on omat päätoimintoalueensa:

- Tietokerrokselle kuuluu tietojen haun lisäksi myös fyysisen varastoinnin hallitseminen. Eli esimerkiksi tietokantojen taulukot.

- Liiketoimintakerroksen tarkoituksena on ylläpitää muun muassa liiketoiminnan sääntöjä, jotka pannaan täytäntöön ohjelmointilogiikan kautta, siitä miten näitä sääntöjä sovelletaan.
- Esityskerrokseen puolestaan kuuluu käyttöliittymä ja koodit, joita käyttäjä näkee. (Armstrong 2006)

5.1.1 Tuotteet

Ensialkuun aloitettiin lisäämällä yksinkertaiset luokat kuvaamaan skeemaa tuotteille. Tuoteluokka sisältää määritelmät jokaiselle tuotteelle. Kuten kuvasta 4 huomataan niin kyseiset määritelmät ovat TuoteID, TuoteNimi, Seloste, Kuva, Hinta, KategoriaID ja Kategoria. Tuoteluokkaan myös lisättiin attribuutteja, joilla määritettiin nimet näytettäville tiedoille ja myös merkkijonon pituudet sekä tietotyypiksi monirivinen teksti. Lisäksi määritettiin ScaffoldColumn(false), joka piilottaa TuoteID sarakkeen.

```
using System.ComponentModel.DataAnnotations;

namespace KodinPutki.Models
{
    public class Tuote
    {
        [ScaffoldColumn(false)]
        public int TuoteID { get; set; }

        [Required, StringLength(100), Display(Name = "Nimi")]
        public string TuoteNimi { get; set; }

        [Required, StringLength(10000), Display(Name = "Tuoteselostus"), DataType(DataType.MultilineText)]
        public string Seloste { get; set; }

        public string Kuva { get; set; }

        [Display(Name = "Hinta")]
        public double? Hinta { get; set; }

        public int? KategoriaID { get; set; }

        public virtual Kategoria Kategoria { get; set; }
    }
}
```

Kuva 4. Määritelmät tuotteille.

Tuoteluokka luo tietokantaan taulukon, jossa kyseiset määritelmät luovat taulukon sarakkeiden nimet.

5.1.2 Kategoria

Kategorialuokka toteutettiin lähes samalla tavalla kuin tuoteluokka. Kategorialuokka luo puolestaan tietokantaan kategorialle oman taulukon, johon luokan jokainen tieto määrittää taulukon sarakkeet.

Kuten kuvasta 5 huomataan, niin kategorialuokkaan määritettiin KategoriaID, jonka avulla tuotteet voitiin määritellä myöhemmässä vaiheessa omiin kategorioihin. Muita määritelmiä ovat KategoriaNimi ja Seloste. Näillä kahdelle määritelmälle määritettiin attribuutit, joilla asetettiin niille maksimi kirjainpituudet ja näytettävät nimet sekä selosteelle tietotyypiksi monirivinen teksti. Loppuun ICollection, joka lataa tuotteet kategorioihin.

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace KodinPutki.Models
{
    public class Kategoria
    {
        [ScaffoldColumn(false)]
        public int KategoriaID { get; set; }

        [Required, StringLength(100), Display(Name = "Nimi")]
        public string KategoriaNimi { get; set; }

        [Display(Name = "Tuoteselostus")]
        public string Seloste { get; set; }

        public virtual ICollection<Tuote> Tuotteet { get; set; }
    }
}
```

Kuva 5. Määritelmät kategorialle.

5.1.3 TuoteKonteksti

Tuotekontekstiluokka luotiin sen takia, koska se hallitsee kokonaisuudessaan tuote- ja kategorialuokkaa sekä antaa tiedoille pääsyn tietokantaan. Kuvassa 6 näkyvän System.Data.Entity nimiavaruuden avulla on pääsy kaikkiin päätoiminnallisuuksiin Entity Frameworkissa. Entity Framework sisältää kyvyn kysellä, lisätä, päivittää ja poistaa tietoja voimakkaasti merkityistä objekteista.

```
using System.Data.Entity;
namespace KodinPutki.Models
{
    public class TuoteKonteksti : DbContext
    {
        public TuoteKonteksti()
            : base("KodinPutki")
        {
        }
        public DbSet<Kategoria> Kategoriat { get; set; }
        public DbSet<Tuote> Tuotteet { get; set; }
    }
}
```

Kuva 6. TuoteKontekstin luonti.

5.1.4 Tietokannan alustaminen

Tietokannan alustamisen avulla saatiin lisättyä tuotteiden ja kategorioiden tiedot tietokantaan, jolloin tuotteet ja kategoriat on mahdollista saada näkyville. Kuvat 7 ja 8 kuuluvat samaan luokkaan, josta nähdään miten kategorioille ja tuotteille määritettiin tiedot. Alussa määritettiin GetKategoriat ja GetTuotteet, joita kutsutaan myöhemmässä vaiheessa, että saadaan kategoriat ja tuotteet.

```
using System.Collections.Generic;
using System.Data.Entity;

namespace KodinPutki.Models
{
    public class TuoteTietokantaAlustus : DropCreateDatabaseIfModelChanges<TuoteKonteksti>
    {
        protected override void Seed(TuoteKonteksti context)
        {
            GetKategoriat().ForEach(c => context.Kategoriat.Add(c));
            GetTuotteet().ForEach(p => context.Tuotteet.Add(p));
        }

        private static List<Kategoria> GetKategoriat()
        {
            var kategoriat = new List<Kategoria> {
                new Kategoria
                {
                    KategoriaID = 1,
                    KategoriaNimi = "Hanat"
                },
                new Kategoria
                {
                    KategoriaID = 2,
                    KategoriaNimi = "WC-Istuimet"
                },
                new Kategoria
                {
                    KategoriaID = 3,
                    KategoriaNimi = "Pesualtaat"
                },
        };
            return kategoriat;
        }
```

Kuva 7. Malli kategorioiden luonnista.

Kuten yllä olevasta kuvasta 7 huomataan, niin kategoriat luotiin tekemällä lista, jolle annettiin metodiksi GetKategoriat. Kyseiselle metodille lisättiin kategoriat,

ja jokaiselle kategorialle määritettiin ID ja nimi. Jokainen lisätty kategoria tulee tietokannassa omalle rivilleen ja sarakkeiden nimet ovat KategoriaID ja KategoriaNimi ja Seloste. Selosteessa on NULL-arvot, koska sille ei määritetty ollenkaan arvoja.

```
private static List<Tuote> GetTuotteet()
{
    var tuotteet = new List<Tuote> {
        new Tuote
        {
            TuoteID = 1,
            TuoteNimi = "Lämminhana",
            Seloste = "Tämä tuote on lämmin.",
            Kuva="hana.png",
            Hinta = 22.50,
            KategoriaID = 1
        },
    new Tuote
        {
            TuoteID = 2,
            TuoteNimi = "IDO WC-Istuin",
            Seloste = "Hyvä istuin.",
            Kuva ="wcpontto.png",
            Hinta = 15.95,
            KategoriaID = 2
        },
        new Tuote
        {
            TuoteID = 3,
            TuoteNimi = "Keltainen pesuallas",
            Seloste = "Iso pesuallas",
            Kuva="pesuallas.png",
            Hinta = 32.99,
            KategoriaID = 3
        },
};
    return   tuotteet;
}
```

Kuva 8. Malli tuotteiden luonnista.

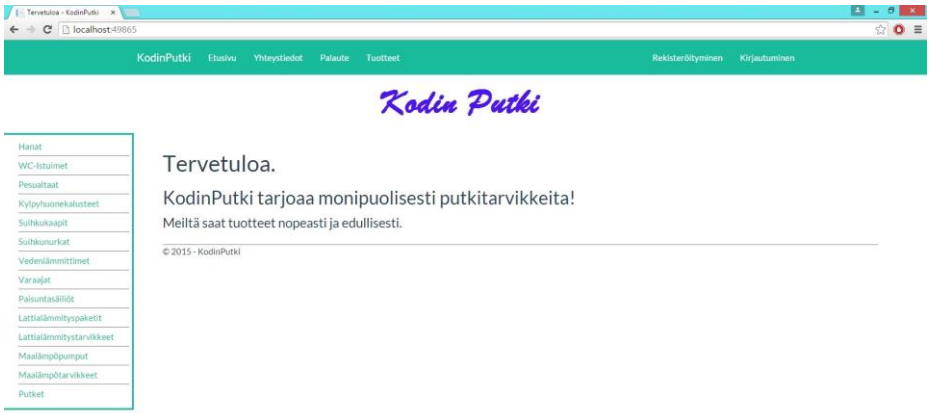
Tuotteille tehtiin samanlainen lista, johon määritettiin TuoteID, TuoteNimi, Seloste, Kuva ja Hinta sekä KategoriaID, jotka näkyvät yllä olevassa kuvassa 8. Projektiin luotiin kuvia varten oma kansio. Tuotteet listalla tehtiin samalla tavalla kuin kategorian kohdalla eli lisättiin metodiksi GetTuotteet.

Lopuksi tehtiin muutoksia Global.asax.cs ja Web.config nimisiin tiedostoihin. Global.asax.cs tiedostoon lisättiin tiedot tietokannan alustuksesta. Eli tiedoston alkuun lisättiin using System.Data.Entity ja using KodinPutki.Models nimiavaruudet sekä kohtaan Application_Start määritettiin Database.SetInitializer(new TuoteTietokantaAlustus());.

Web.config tiedoston tallennettiin tietokantayhteyden merkkijono ConnectionStrings tagien sisälle. Eli kun sovellus ensimmäisen kerran ajetaan niin se rakentaa tietokannan sijainnin määritetyllä yhteyden merkkijonolla.

5.2 Etusivu

Kuten kuvasta 9 huomataan, niin etusivusta luotiin karkea versio, koska pääpaino oli toteuttaa tuotesivusto verkkokauppaa varten. Default.aspx sivusta määritettiin asetuksien kautta etusivu, ja kyseiselle sivulle lisättiin asp:Content tagin sisälle eri otsikkotasoja ja tekstiä varten kappale-tagi.



Kuva 9. Etusivun näkymä.

Etusivun Kodin Putki –logo lisättiin projektissa olevaan kansioon, johon viitattiin Site.Master sivulla. Site.Master sivulle määritettiin div-elementtiin ID ylätunnistetta varten, johon sijoitettiin logo viittaamalla ImageUrl-parametrin avulla. Lisäksi logolle lisättiin linkitys etusivulle, eli sitä klikkaamalla käyttäjä ohjataan takaisin etusivulle.

Sivustolle valittiin teema ilmaiselta www.bootswatch.com –sivustolta. Sivustolta ladattiin bootstrap.css ja bootstrap.min.css, jotka sijoitettiin projektin Content kansioon alkuperäisten bootstrap-tiedostojen lisäksi. Alkuperäisten bootstrap-tiedostojen nimet muutettiin, ettei alkuperäisiä tiedostoja korvata

uusilla. Bootstrap käyttää CSS3, joka tarjoaa responsiivisen suunnittelun, mikä tarkoittaa sitä, että se mukautuu dynaamisesti eri selainikkunoiden kokoon.

5.3 Tietojen näyttäminen tietokannasta

5.3.1 Tuotesivu

Jotta käyttäjä voi tutkia kaikkia tuotteita tuotesivulla niin projektiin lisättiin verkkolomake master sivun kanssa (englanniksi Web Form with Master Page), jolle määritettiin nimeksi TuoteLista.

Tuotteiden saamiseksi näkyville tietokannasta lisättiin TuoteLista sivulle ItemTemplate ListView'n sisälle. ListView'lle määritettiin muun muassa GroupItemCount ja SelectMethod. GroupItemCount'lle määritettiinkokonaisluku arvo, joka kertoo tuotteiden määrän yhtä riviä kohden. SelectMethod-kontrollille lisättiin arvoksi GetTuotteet, joka kutsuu saman nimistä kyselykoodia C# -tiedostosta.

ListView'n sisälle määritettyyn ItemTemplate –elementtiin määritettiin taulukko, jonka esimmäiselle riville sijoitettiin tuotekuva ja alapuolelle tuotenimi sekä hinta. Tuotekuvaa tai tuotenimeä klikkaamalla käyttäjä ohjataan kyseisen tuotteen tietoihin.

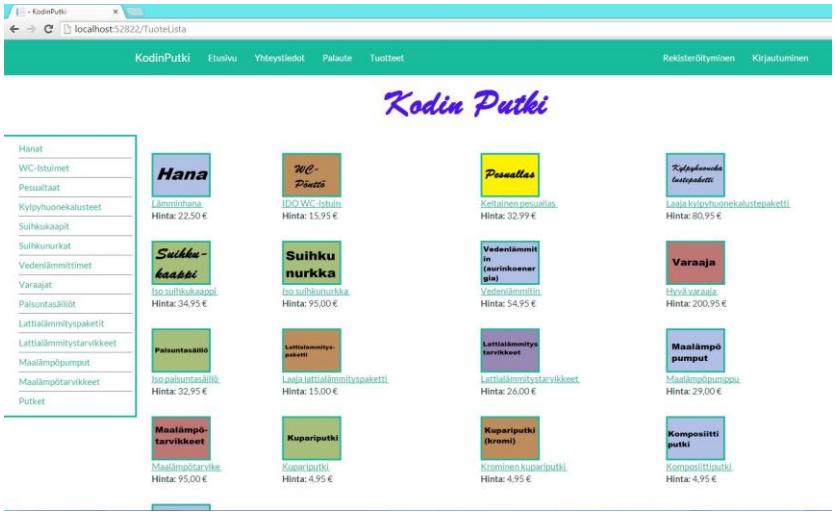
Tuotekuvan, tuotenimen ja hinnan hakemisessa käytettiin <%# %> -syntaxia apuna. Kyseinen syntax hakee tietokannasta tarvitsemansa tiedon. Tuotekuvan ja tuotenimen kohdalla syntaxia käytettiin sijainnin määrityksessä. Esimerkki syntaxin käytöstä tuotetietoihin linkityksessä kuvan yhteydessä:

```
<a href="TuoteTiedot.aspx?tuoteID=<%#:Item.TuoteID%>">
<img src="/Kuvat/<%#:Item.Kuva%>"/></a>
```

Kyseisessä esimerkissä määritettiin ensin linkitys tuotesivulle ja sen sisälle tuotekuva, jotta tuotekuvalle saatiin myös linkitys tuotesivulle. Tuotenimi toteutettiin lähes samalla tavalla kuin aikaisemmassa esimerkissä, mutta kuva tagin tilalle määritettiin tuotenimi span-elementin sisälle. Hinta puolestaan toteutettiin lisäämällä syntax span-elementin sisälle. Syntaxin sisälle määritettiin String.Format(”{0:c}”, Item.Hinta), joka tuo tietokannasta hinnan kahden desimaalin tarkkuudella.

5.3.2 Kategoria

SelectMethod’n arvo GetTuotteet lisättiin C#-tiedostoon, jolle määritettiin koodi, jossa tarkistettiin kyselyn avulla tietokannassa olevien tuotteiden kategoriaID-parametrit. KategoriaID-parametrien perusteella tuotteet saatiin jaettua oikeisiin kategorioihin.



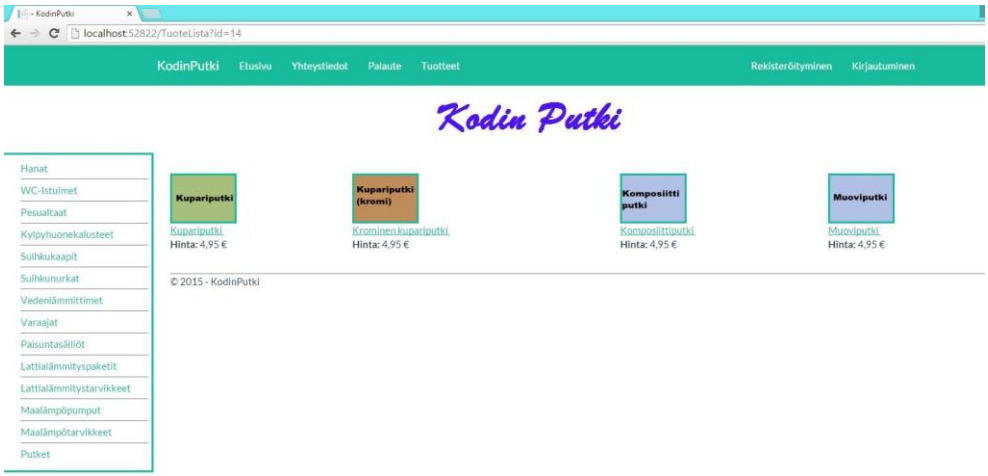
Kuva 10. Tuotesivun ja tuotevalikon näkymä.

Kategorioiden nimet eli toisin sanoen tuoteryhmät sijoitettiin sivun vasempaan laitaan, jotta sivustosta saataisiin ulkoasullisesti mahdollisimman selkeä (kuva 10). Site.Master sivulle määritettiin navigoinneille oma div-elementti, jonka sisälle määritettiin linkitykset jokaiseen kategoriaan ja jokaisen linkityksen jälkeen laitettiin rivinvaihto ja väliviivat hr-tagia käyttäen. Linkitys toteutettiin HTML href-attribuutin avulla, johon sijoitettiin linkki TuoteListaan ja loppuun ?id=

ja kategoriaID haluamaan kategoriaan. Eli esimerkkinä linkitys Hana kategoriaan:

Hanat

Sivuston käyttäjät pystyvät tarkastelemaan tiettyjä tuotteita valitsemalla tuoteryhmästä haluamansa tuotteet. Kuvassa 11 nähdään esimerkkinä ainoastaan putket valittuna.



Kuva 11. Rajattu tiettyihin tuotteisiin.

5.3.3 Yksittäisen tuotteen näkymä

Tuotetiedoille määritettiin oma sivu nimeltä TuoteTiedot.aspx, johon määritettiin content alueen sisälle FormView-kontrolli, joka näyttää yksittäisen tuotteen tiedot. FormView’n sisälle puolestaan määritettiin ItemTemplate-kontrolli, jonka sisälle määritettiin div-elementti otsikkoa varten, ja taulukko tuotekuvaa sekä tuotetietoja varten.

Kuten kuvasta 12 huomataan lopputulos kun sijoitettiin taulukossa esimmäiseen sarakkeeseen tuotekuva ja seuraavaan sarakkeeseen tuoteselostus, hinta ja tuotenumero. Tuotekuva, kuten myös tuoteselostus, hinta ja tuotenumero määritettiin aikaisemmassa vaiheessa jokaiselle tuotteelle erikseen. Tässä

vaiheessa käytettiin apuna <%# &> -syntaxia, joitten sisälle määritettiin tieto mitä haluttiin saada näkyville.

Kuva 12. Yksittäisen tuotteen näkymä.

Ennen kuin tuotetiedot saatiin näkyville, jouduttiin määrittämään TuoteTiedot.aspx.cs tiedostoon tietokantaan yhdistäminen. Kyseiseen tiedostoon määritettiin koodi, joka tarkistaa TuoteID kyselyarvon. Mikäli löytyy kelvollinen kyselyarvo niin näytetään tuote, joka vastaa tuoteID'n arvoa.

5.4 Tavoitteet ja aikataulu

Opinnäytetyön tavoitteena oli kehittää sivusto, joka sisältää kaiken kaikkiaan tietoa itse yrityksestä ja tuotesivuston. Tuotesivulla tarkoituksena oli, että tuotteen pystyy rajaamaan haluamaansa tuoteryhmään ja katsomaan yksittäisen tuotteen tietoja. Eli mikäli haluaa nähdä ainoastaan tietyt tuotteet niin sen tuotteen rajaaminen onnistuu valitsemalla sivunaginoinnista haluamansa tuoteryhmän. Tuotetietoja pystyy katsomaan klikkaamalla tuotetta.

Tavoitteena oli toteuttaa sivusto ilman responsiivisuutta, mutta osa sivustosta on responsiivinen johtuen Bootstrapista. Koska sivusto ei ole lopullinen versio niin sivustosta voidaan myöhemmässä vaiheessa tehdä täysin responsiivinen.

Toimeksiantaja ei varsinaisesti määrittänyt työlle tiettyä päivämäärää takarajaksi, että milloin työn pitäisi olla valmis. Halusi kuitenkin työn olevan kesään mennessä valmis.

6 Tulokset

Tämän opinnäytetyön tarkoituksena oli kehittää tuotesivusto verkkokauppaa varten. Tässä osiossa käydään läpi vaatimusmäärittelyn perusteella toteutuneet toiminnallisuudet sekä perehdytään toiminnallisuuksiin, joita ei toteutettu tässä opinnäytetyössä.

Toteutuneet toiminnallisuudet ovat juuri tämän tuotesivuston kannalta tärkeitä, eli kyseiset toiminnallisuudet ovat:

- Käyttäjä voi nähdä kaikki tuotteet kerralla.
- Käyttäjä voi valita tuoteryhmästä haluamansa tuotteet näkyville.
- Käyttäjä voi tarkastella haluamansa tuotteen tietoja.

Lisäksi projektin mukana tuli valmiina seuraavat toiminnallisuudet:

- Käyttäjä voi rekisteröityä käyttäjäksi.
- Käyttäjä voi kirjautua sisään.
- Käyttäjä voi muuttaa käyttäjätilinsä salasanan.

Työn toiminnalliset vaatimukset, joita ei toteutettu:

- Käyttäjä voi etsiä tuotteita käyttäen hakutoimintoa.
- Käyttäjä voi lisätä tuotteita ostoskoriin.
- Käyttäjä voi nähdä ja valita yksittäisen tuotteen näkymässä siihen tuotteeseen liittyvät muut tuotteet.
- Ylläpitäjä voi hallita tuotteita käyttäjätilinsä kautta hallintapaneelissa.
- Käyttäjä voi lähettää tarjouspyynnön.

7 Pohdinta

Opinnäytetyö aloitettiin tammikuussa 2015 ja valmistui huhtikuussa. Työn valmistuessa tarkastelimme toimeksiantajan kanssa yhdessä millainen lopputulos suoritetusta työstä loppujen lopuksi tuli. Toimeksiantajan mielestä opinnäytetyö vastaa sitä mitä työltä ensisijaisesti haluttiin, vaikka mainitsi, että tuoteikkunat voisi olla vähän modernisempia.

Opinnäytetyön toteutuksessa käytin erilaisia tekniikoita, joista osa oli jo ennestään tuttuja ja osasta oli taas vähäistä kokemusta. Muun muassa data access layerin luontiin minun tuli tutustua entistäkin tarkemmin.

7.1 Omia mietteitä

Omasta mielestä pääsin opinnäytetyön aikana määritettyihin tavoitteisiin, ja samalla opin erityisesti edellä mainitusta data access layeristä. Opinnäytetyön aihe oli mielestäni mielenkiintoinen ja sopivan kokoinen sekä samalla sopivan haastava.

Lisäksi opinnäytetyössä suoritetut vaatimukset vastasivat sitä, mitä opinnäytetyöltä odotettiin. Toteutumatta jääneet vaatimukset olivat niitä, joita ei ensisijaisesti odotettu valmistuvan tämän työn puitteissa. Jätin toteuttamatta kyseiset lisätoiminnallisuudet ajan puutteen takia, koska opinnäytetyöraportin kirjoittamiseen täytyi myös varata aikaa.

Työ eteni pienissä osissa eteenpäin. Paremmalla tutustumisella Data Access Layer'iä käsitteleviin materiaaleihin ennen varsinaista työn aloitusta olisi luultavasti jäänyt aikaa perehtyä lisätoiminnallisuuksiin. Eniten aikaa vei sovelluskoodien toteuttaminen, varsinkin tuotteiden hakuun tarkoitettu koodi. Puolestaan mikä liittyi sivuston ulkoasuun ei vienyt paljoakaan aikaa. Ulkoasuun liittvät asiat oli kategoria ja tuotesivun tuoteikkunoiden toteuttaminen sekä tuotetiedoille tarkoitettu sivu.

7.2 Jatkokehittämisideat

Toimeksiantajalta tuli ideoita jatkokehittämistä varten samalla kun käytiin opinnäytetyön lopputulokset yhdessä läpi. Ensimmäinen idea oli tuoteikkunan muuttamisen lisäksi se, että etusivulle määritettäisiin ikkuna, jossa ilmoitettaisiin tuote, joka olisi juuri silloin alennuksessa.

Toinen idea oli, että määritettäisiin etusivulle toinen ikkuna, jossa asiakkaat voisivat ottaa yhteyttä kysyäkseen tuotetta tai lähettämällä kuva jostakin tuotteesta. Tämä mahdollistaisi asiakaspalveluhenkisen verkkokaupan, ja helpottaisi asiakkaiden ostohetkiä verkkokaupassa.

Omasta mielestä ikkuna, jossa voi ottaa yhteyttä myyjään, olisi parempi sijoittaa omalle sivulle. Toinen kehittämisidea olisi, että myyjälle voisi lähettää listan haluamistaan tuotteista ja kysyä näistä pakettihintaa. Tämän voisi toteuttaa ostoskorin sisälle, tai erilliselle sivulle.

Lähteet

1. Adams, T.S. 2015. What Is a Data Access Layer?. WiseGEEK. http://www.wisegeek.com/what-is-a-data-access-layer.htm. [Viitattu 8.5.2015.]
2. Aitken, D. 2015. All HTML5 Tags. HTML-5-tutorial.com. http://www.html-5-tutorial.com/all-html-tags.htm. [Viitattu 12.3.2015.]
3. Armstrong, D. 2006. .NET Application Architecture: the Data Access Layer. Redgate. https://www.simple-talk.com/dotnet/.net-framework/.net-application-architecture-the-data-access-layer/. [Viitattu 30.4.2015.]
4. Massi, B. 2012. Working with SQL Server LocalDB in LightSwitch Projects in Visual Studio 2012. http://blogs.msdn.com/b/bethmassi/archive/2012/04/19/working-with-sql-server-localdb-in-lightswitch-projects-in-visual-studio-11.aspx. [Viitattu 13.4.2015.]
5. Bookwalter, J.R. 2011. What is CSS3?. Mac|Life. http://www.maclife.com/article/features/what_css3. [Viitattu 16.3.2015.]
6. Entity Framework Tutorial. 2015. What is Entity Framework?. http://www.entityframeworktutorial.net/what-is-entityframework.aspx. [Viitattu 16.3.2015.]
7. Laaksonen, A. 2009. MySQL ja PHP: Osa 8 – Tietokannan suunnittelu. http://www.ohjelmointiputka.net/oppaat/opas.php?tunnus=mysqlphp08. [Viitattu 24.3.2015.]
8. Marshall, G. 2011. HTML5: what is it?. TechRadar. http://www.techradar.com/news/internet/web/html5-what-is-it-1047393. [Viitattu 12.3.2015.]
9. Microsoft. 2014. Visual Studio Professional with MSDN. Microsoft. https://www.visualstudio.com/en-us/products/visual-studio-professional-with-msdn-vs.aspx. [Viitattu 15.3.2015.]
10. Microsoft. 2015. Introduction to ASP.NET Web Forms. Microsoft. http://www.asp.net/web-forms/what-is-web-forms. [Viitattu 13.4.2015.]
11. Patton, T. 2006. Use the data access layer to simplify architecture. TechRepublic. http://www.techrepublic.com/article/use-the-data-access-layer-to-simplify-architecture/. [Viitattu 9.5.2015]
12. Rouse, M. 2007. C#. TechTarget. http://searchwindevelopment.techtarget.com/definition/C. [Viitattu 16.3.2015.]