VERKKOSIVUSTON TOTEUTTAMINEN UMBRACO-SISÄLLÖNHALLINTAJÄRJESTELMÄLLÄ

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietotekniikan koulutusohjelma

Syksy, 2021

Gia Matikainen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietotekniikan koulutusohjelma Tiivistelmä
Tekijä Gia Matikainen Vuosi 2021
Työn nimi Verkkosivuston toteuttaminen Umbraco-sisällönhallintajärjestelmällä
Ohjaaja Toni Laitinen

Riihisoft Oy on Microsoftin teknologioihin perustuviin ohjelmistokehitystyökaluihin erikoistunut ohjelmistopalveluiden tarjoaja. Opinnäytetyössä yritykselle suunniteltiin ja toteutettiin uusi verkkosivusto, jotta käyttöön saatiin uusi ulkoasu, uusia toiminnallisuuksia sekä tehokkaat sisällönhallinnan mahdollisuudet hyödyntäen Umbraco CMS -sisällönhallintajärjestelmää.

Opinnäytetyön teoriaosuudessa tutustutaan projektissa hyödynnettyihin teknologioihin, jotka vahvasti keskittyvät Microsoftin teknologioihin sekä MVC-sovellusarkkitehtuuriin. Toteutusta kuvaavassa osuudessa esitetään ASP.NET MVC -verkkosovelluskehyksessä kehitetyn Umbraco-pohjaisen verkkosivuston kehittämiselle ominaisia käsitteitä ja työvaiheita. Viimeisessä osassa pohditaan verkkosivuston toteutuksen onnistumista.

Verkkosivuston toteuttaminen syvensi osaamista verkkosivuston toteuttamisesta ja hyödynnetyistä teknologioista. Lopputuloksena on vaatimusmäärittelyt täyttävä ja aikataulussa valmistunut verkkosivusto, joka toimii yrityksen virtuaalisena näyteikkunana.

Avainsanat Sisällönhallintajärjestelmä, ohjelmistokehitys, ASP.NET MVC
Sivut 28 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Information Technology
Author Gia Matikainen
Subject Implementing a website with Umbraco Content Management System
Supervisors Toni Laitinen

Abstract
Year 2021

Riihisoft Ltd is an industry-independent software service provider that is specialized on software development tools based on Microsoft technologies. The aim of this thesis was to plan and implement a website for Riihisoft Ltd in order to provide a new layout, new functionalities and efficient content management possibilities by utilizing the Umbraco CMS content management system.

Main technologies used in the project are introduced in the theoretical part of this thesis. The technologies are mainly focused on Microsoft technologies and MVC architecture pattern. The development part of the thesis presents the concepts and incident operations of developing a website with ASP.NET MVC framework and Umbraco CMS. The succes of the implemented website is discussed in the last part.

Implementing the website deepened my expertise in development of a website and in technologies used for work. The result is a website that was published on time and meets the specification requirements. Now the website serves as the company's virtual showcase.

Keywords Content Management System, software development, ASP.NET MVC
Pages 28 pages

Sisällys

1 Johdanto 1

2 Teknologiat 2

2.1 MVC-malli 2

2.2 ASP.NET MVC 3

2.3 C# 3

2.4 Umbraco-sisällönhallintajärjestelmä 4

2.4.1 Dokumenttityypit 4

2.4.2 Näkymät 6

3 Verkkosivuston suunnittelu ja toteutus 7

3.1 Kehitysympäristö 7

3.2 Dokumenttityyppien luominen 7

3.3 Näkymien luominen 9

3.3.1 Osittaisten näkymien luominen 10

3.3.2 Makro-osat 13

3.4 Yhteydenottolomake 15

3.5 Ajankohtaista-sivu 20

3.6 Kustomoitu tapahtuma 22

3.7 Sisällön siirtäminen 24

3.8 Tuotantoympäristö 25

4 Yhteenveto 25

Lähteet 27

1 Johdanto

Riihisoft Oy on Microsoftin teknologioihin perustuviin ohjelmistokehitystyökaluihin erikoistunut ohjelmistopalveluiden tarjoaja. Verkkopalveluita, sovelluksia, ohjelmistoja ja integraatioita tarjotaan toimialariippumattomasti keskisuurille ja suurille yrityksille. Julkaisujärjestelmien osalta yrityksen erityisosaamista on ASP.NET-pohjainen Umbraco CMS -sisällönhallintajärjestelmä. Yritys on Certified Umbraco CMS Partner ja yrityksellä on sertifioituja Umbraco-kehittäjiä. Tämän toiminnallisen opinnäytetyön aiheena on yrityksen verkkosivuston toteuttaminen hyödyntäen Umbraco CMS -sisällönhallintajärjestelmää.

Uusi verkkosivusto toteutetaan, jotta käyttöön saadaan uusi ulkoasu, uusia toiminnallisuuksia sekä tehokkaat sisällönhallinnan mahdollisuudet hyödyntäen Umbracon kahdeksatta versiota. Olemassa olevan verkkosivuston Umbracon seitsemännen version ja käyttöönotettavan kahdeksannen version väliset muutokset estävät kahdeksannen version käyttöönoton päivityksenä, joten järjestelmälle tulee suorittaa migraatio. Umbracon verkkosivuilla ohjeistetaan suorittamaan migraatio asentamalla verkkosivuston projektiin uusi kahdeksannen version Umbraco-asennus, väliaikaisesti poistamaan käytöstä Umbracon kanssa toimivan koodin ja osa kerrallaan palauttamaan koodia muokaten sitä yhteensopivaksi kahdeksannen version kanssa. Verkkosivuston tapauksessa migraatio suoritetaan rakentamalla verkkosivusto uuden projektin, uuden Umbraco-asennuksen ja uuden tietokannan päälle. Verkkosivusto rakennetaan alusta asti, jotta kehittäjän osaaminen verkkosivuston kehittämisestä sisällönhallintajärjestelmää hyödyntäen ja Umbracon kahdeksannen version uusista ominaisuuksista syventyy.

Opinnäytetyössä vastataan kysymykseen ”Miten toteuttaa verkkosivusto Umbraco CMS -sisällönhallintajärjestelmää hyödyntäen?” esittämällä Umbraco-pohjaiselle verkkosivustolle ominaisia käsitteitä, käytänteitä ja työvaiheita. Tavoitteena ei ole kattaa verkkosivuston kehitysprosessia kokonaisuudessaan. Opinnäytetyössä esitettyjä ratkaisuja ja havaintoja voidaan hyödyntää Umbraco-pohjaisen verkkosivuston kehittämisessä sekä sisällönhallintajärjestelmän valintaprosessissa.

2 Teknologiat

Sisällönhallintajärjestelmällä mahdollistetaan verkkosivuston sisällön luominen, muuttaminen, tallentaminen ja julkaiseminen tehokkaasti intuitiivisen käyttöliittymän kautta. Marraskuussa 2021 suosituimpia sisällönhallintajärjestelmiä olivat WordPress, Shopify, Wix, Squarespace ja Joomla (W3Techs, 2021-a). WordPress on avoimeen lähdekoodiin perustuva PHP-ohjelmointikielellä kirjoitettu sisällönhallintajärjestelmä, joka on alun perin suunniteltu blogialustaksi. Joulukuussa 2021 WordPress-sisällönhallintajärjestelmällä oli 43,2 % käyttöosuus Umbracon käyttöosuuden ollessa alle 0,1 % (W3Techs, 2021-b).

Umbracon Microsoftin infrastruktuuriin perustuva ASP.NET-alusta oli sisällönhallintajärjestelmän valinnassa olennaisinta. Palvelinpuolen verkkosovelluskehyksistä marraskuussa 2021 käytetyin oli PHP 78,1 %:n käyttöosuudella, ASP.NET:in käyttöosuuden ollessa 8 % (W3Techs, 2021-c). Marraskuussa 2021 ASP.NET-alustalla oli toteutettu enemmän korkeamman liikenteen verkkosivustoja, kuin PHP:lla (W3Techs, 2021-d). ASP.NET-pohjaisen Umbracon joustavuus tekee siitä soveltuvan matalan ja korkean liikenteen sekä kompleksisuuden totetuksiin. Umbraco tarjoaa tyhjän alustan, jolle rakentaa sisällönhallintaa kustomoidusti. WordPress on puolestaan kerryttänyt suosionsa kehittäjien ja loppukäyttäjien keskuudessa helppokäyttöisyydellä. Helppokäyttöisyys kiristää puolestaan kustomoitavuutta. Umbraco palvelee teknistä osaamista omaavia kehittäjiä, jotka osaavat hyödyntää alustan tarjoamia mahdollisuuksia. Umbracon mainittavia etuja ovat mahdollisuus usean sivuston hallitsemiseen yhdestä Umbraco-instanssista, sisäänrakennettu hakukone ja käyttäjien sekä jäsenten hallinta.

2.1 MVC-malli

Model-View-Controller (MVC) on sovellusarkkitehtuurimalli, jonka pääkäsitteet ovat malli, näkymä ja käsittelijä. Malli hallitsee sovelluksen käsittelemiä tietoja. Yleisesti jokainen malli esittää yhtä tai useampaa tietokannan taulua. Näkymä sisältää visuaalisen esitysmuodon sovellukselle. Verkkosivustoilla näkymät toteutetaan yleisesti hyödyntäen HTML-merkintäkieltä, CSS-tyylejä ja JavaScript-ohjelmointikieltä. Mallin ja näkymän välissä työtä

tekee käsittelijä, joka pyytää tietoa mallilta ja välittää näkymälle, joka esittää tiedon. (Munro 2015, 1.)

2.2 ASP.NET MVC

ASP.NET on Microsoftin kehittämä verkkosovelluskehys, joka tarjoaa palveluita palvelinpuolen verkkosovellusten kehittämiseen. ASP.NET:in alustana toimii .NET Framework -ohjelmistokomponenttikirjasto. ASP.NET:lla voidaan kirjoittaa sovelluksia Common Language Runtime (CLR) -suoritusalustan kanssa yhteensopivilla ohjelmointikielillä, kuten C# ja Visual Basic. (Paz 2013, 1.)

Vuonna 2002 julkaistiin ensimmäinen ASP.NET 0.1. Seuraavina vuosina uudet julkaisut tarjosivat kehittäjille uusia ominaisuuksia dynaamisten verkkosivujen toteuttamiseen, mutta kehittäminen koettiin haastavaksi, kun käytössä oli vain sisäänrakennettu malli. Kehittämistä varjosti komplikaatiot, hämmentävät sivujen elinkaaret, matala kustomoitavuus sekä epäoptimaalinen HTML-merkintäkoodi. Haasteisiin vastattiin vuonna 2007 julkaisemalla MVC-mallia toteuttava ASP.NET-versio eli ASP.NET MVC. Samaan aikaan MVC-mallista oli tulossa yleisesti yksi suosituimmista tavoista rakentaa verkkosovelluskehyksiä. (Galloway, Wilson, Allen & Matson 2014, 2.)

2.3 C#

C# on moderni, objekti- ja komponenttiorientoitunut sekä tyyppiturvallinen ohjelmointikieli, jota ajetaan .NET-alustalla. Kielen juuret ovat C-kieliperheessä, joten se omaa yhteneväisiä piirteitä C-, C++ - ja Java-ohjelmointikielten kanssa. Objekti- ja komponenttiorientoituneisuus ilmenevät kielessä rakenteina, jotka soveltuvat erityisesti ohjelmistokomponenttien hyödyntämiseen. (Microsoft, 2021-a)

C#-ohjelmat suoritetaan virtuaalisella Common Language Runtime (CLR) -suoritusalustalla. CLR:n tehtävä on tulkita ohjelmointikielestä kääntäjän tuottama välikieli konekieleksi ajon aikana (Microsoft, 2021-b). CLR on Microsoftin implementaatio kansainvälisestä Common

Language Infrastructure (CLI) -standardista. CLI mahdollistaa ohjelmointikielten ja kirjastojen yhteensopivuuden suoritus- ja kehitysympäristöissä. (Microsoft, 2021-c)

2.4 Umbraco-sisällönhallintajärjestelmä

Umbraco on maksuton, avoimeen lähdekoodiin perustuva, .NET-pohjainen ja C#-ohjelmointikielellä kirjoitettu sisällönhallintajärjestelmä. Ensimmäinen versio Umbracosta julkaistiin vuonna 2003, mutta avoin lähdekoodi vasta vuonna 2005. (Umbraco, n.d.-a) Umbracoa ylläpitää Tanskalainen yritys nimeltään Umbraco HQ, jonka lisäksi avoimen lähdekoodin myötä kehitykseen osallistuu globaali 220 000 kehittäjän yhteisö (Umbraco, 2021-b).

Umbraco sai nimensä tanskan kielen puhekielessä käytettävän termin ”umbraconøgle” myötä, joka tarkoittaa kuusiokoloavainta. Umbracon luoja Niels Hartvig viittaa nimivalinnalla Umbracon olevan helppokäyttöinen työkalu, joka mahdollistaa upeiden ratkaisuiden toteuttamisen. (Umbraco, n.d.-c) Umbracolla toteutetaan kustomoituja ja intuitiivisesti käytettäviä käyttöliittymiä verkkosovellusten sisällönhallintaan. Umbraco soveltuu verkkokauppojen, intranet-sivustojen sekä headless-ratkaisujen eli palvelinpuolen kokonaisuuksien toteuttamiseen.

Umbracon alustana on aina toiminut .NET Framework -ohjelmistokomponenttikirjasto ja ASP.NET-verkkosovelluskehys. Syksyllä 2021 julkaistiin Umbracon yhdeksäs versio, joka on rakennettu ensimmäistä kertaa .NET 5 -ohjelmistokehitysalustan ja ASP.NET Core -verkkosovelluskehyksen päälle. Yhdeksäs julkaisu tekee Umbracosta alustariippumattoman eli yhteensopivan Windows-käyttöjärjestelmän lisäksi macOs- ja Linux-käyttöjärjestelmien kanssa. (Umbraco, n.d.-d)

2.4.1 Dokumenttityypit

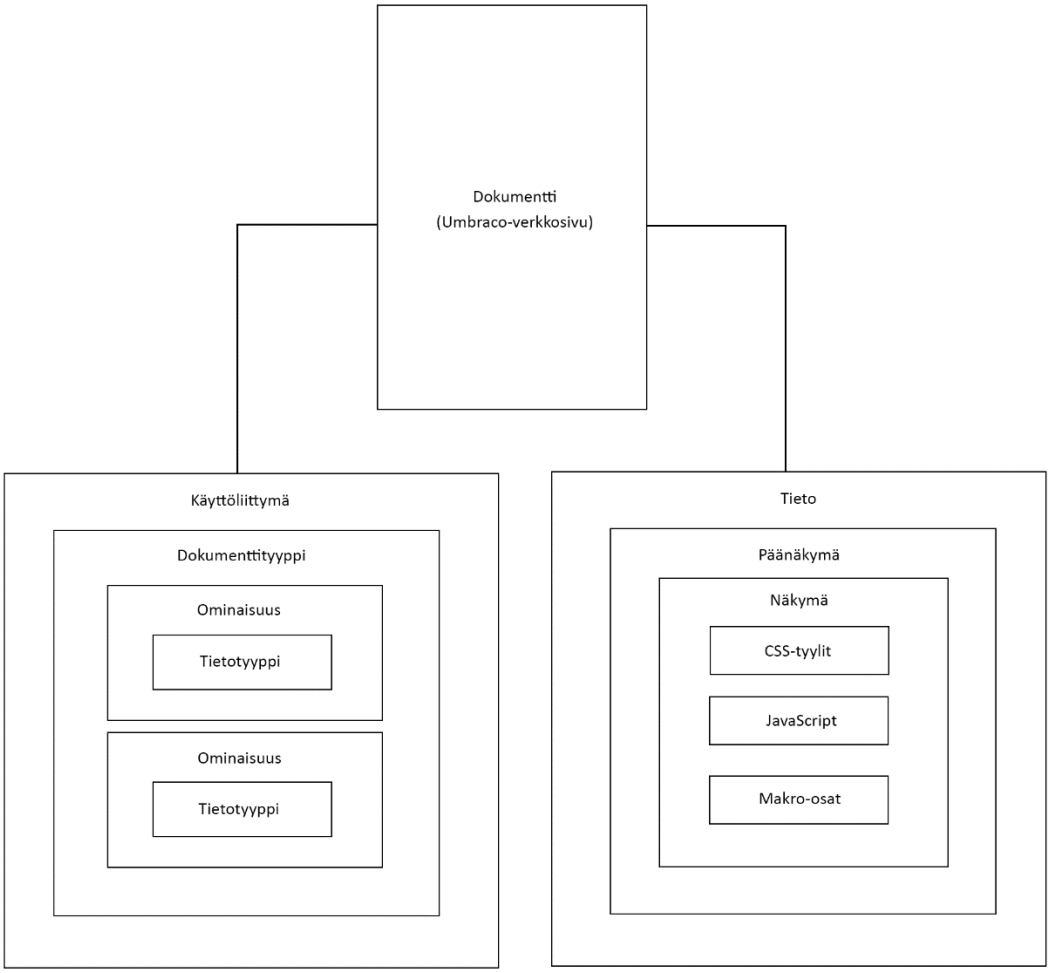
Sisällön syöttämiseksi verkkosivustolle, tulee Umbracoon määritellä tietovarastoja, joihin sisältöä tallennetaan ja joista sitä haetaan. Tietovarastot eli dokumenttityypit (Document Type) määrittelevät verkkosivuston sisällön nimien ja tyyppien avulla. Jokaisen

verkkosivuston sisältö on uniikkia, tehden niihin perustuvista dokumenttityypeistä uniikkeja. Analogiana dokumenttityypin luomiselle on elokuvakokoelman seuranta tallentamalla teoksen nimi, ohjaaja ja julkaisuvuosi. Dokumenttityyppien määrittelyssä hyödynnetään tietokannan taulujen muodostamiseen verrattavia prosesseja.

Verkkosivuston sisältö on monimuotoista ja sisältää lukuisia tietoja. Dokumenttityypit määritellään sisällöstä poimittujen tietojen perusteella, kuten yksittäinen sivu, sisällössä esiintyvä tieto tai tietojen yhdistelmä. Analogiana useiden erilaisten dokumenttityyppien luomiselle on elokuvakokoelman lisäksi musiikkikokelman seuranta. Musiikkikokoelmassa seurataan eri tietoja, joten musiikkikokoelmalle luodaan oma dokumenttityyppi, joka tallentaa teoksen nimen, esittäjän ja tyylilajin. Umbracon kustomoitavuus piilee dokumenttityypeissä, sillä niiden avulla voidaan määritellä mitä vain. (Umbraco, n.d.-e)

Dokumenttityypit koostetaan ominaisuuksista (Property), joilla on tietotyypit (Data Type), kuten kokonaisluku (Kuva 1). Ominaisuudet ovat kuin tietokenttiä tai attribuutteja, joihin syöttää tietoa, kuten sivun otsikko. Jokaisella ominaisuudella on tietotyypin mukainen ominaisuusmuokkaaja (Property Editor), jonka avulla sisältöä syötetään ominaisuuteen. Ominaisuusmuokkaajat ovat olennainen osa Umbracon kustomoitavuutta. Ominaisuusmuokkaajilla voidaan toteuttaa monimuotoisia työkaluja, kuten kuvien rajaamista tai sivun elementtien sommittelua. (Umbraco, n.d.-f)

Kuva 1. Umbraco-verkkosivun rakenne. (mukailtu Quick Start n.d.)



2.4.2 Näkymät

Verkkosivuston sisällön rakenteen ja dokumenttityyppien määrittelyn jälkeen määritellään tapa, jolla esittää dokumenttityyppien säilyttämää sisältöä. Luodaan näkymiä, jotka osoitetaan dokumenttityypeille. Umbraco hyödyntää .NET-alustan standardeja monessa toteutuksessa, kuten pääsivut (Master Pages), jotka ovat yksi Umbracon tärkeimmistä konsepteista. Pääsivut mahdollistavat muokattavien, hallittavien ja joustavien sivujen ulkoasujen toteuttamisen eriyttämällä sisällön rakenteesta. Lisäksi konsepti mahdollistaa sisällön ja tyylien jakamisen monien sivujen läpi sivujen jälkeläisiksi määritellyille sivuille. (Wahlberg & Sterling 2011, 73.)

Näkymillä hyödynnetään Razor-merkintäsyntaksia, jolla upotetaan dynaamista .NET-pohjaista koodia näkymille. Razor-syntaksi muodostuu Razor-merkinnästä, C#-

ohjelmointikielestä ja HTML-merkintäkielestä. Razor-syntaksia sisältävien näkymien tiedostot ovat .cshtml-päätteisiä. (Microsoft, 2021-d)

3 Verkkosivuston suunnittelu ja toteutus

Verkkosivuston sisällönhallintajärjestelmäksi määriteltiin ASP.NET-pohjainen Umbraco ja sovellusarkkitehtuurimalliksi MVC, joten verkkosivuston alustaksi määriteltiin ASP.NET MVC. ASP.NET-alustan myötä palvelinpuolen ohjelmointikielenä hyödynnettiin C#-ohjelmointikieltä. Graafisen käyttöliittymän HTML-elementit ja CSS-tyylit toimitti mainostoimisto. Käyttöliittymään interaktiivisuutta ja dynaamisuutta toteutettiin JavaScript-ohjelmointikielellä ja Razor-syntaksilla.

3.1 Kehitysympäristö

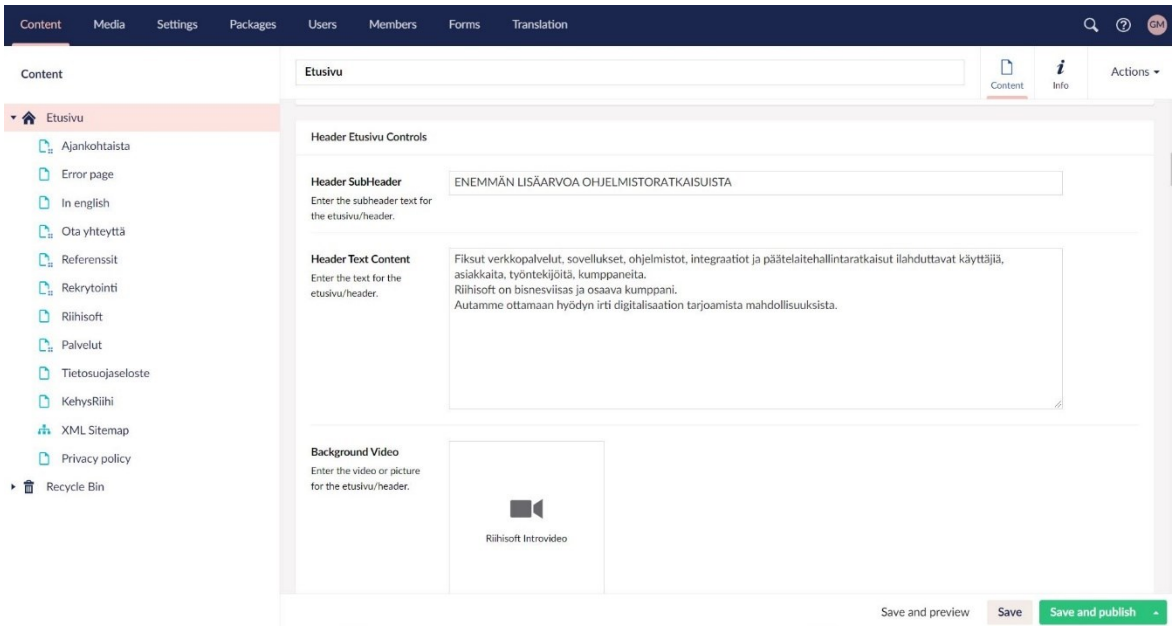
Kehitysympäristön Windows-käyttöjärjestelmän ominaisuuksissa käyttöönotettiin .NET Framework -ohjelmistokomponenttikirjasto, joka sisältää tietokoneen arkkitehtuurissa ja käyttöjärjestelmässä tarvittavat osat ASP.NET:in hyödyntämiseksi. Kehityksen aikaiseksi tietokannaksi asennetiin pienen mittakaavan ohjelmistokehittämiseen suunniteltu Microsoft SQL Server Express -relaatiotietokantajärjestelmä. Järjestelmän hallinnointityökaluksi asennettiin Microsoft SQL Server Management Studio, jolla luotiin SQL-tietokanta ja käyttäjä verkkosivuston tietokantayhteyttä varten. Ohjelmointiympäristöksi asennettiin Microsoft Visual Studio, joka tarjoaa koodin muokkaajan, työkaluja virheiden jäljittämiseen ja kääntäjän yhdestä käyttöliittymästä. Verkkosivusto toteutettiin Visual Studiossa luotuun ASP.NET Web Application -projektiin. Umbraco versiossa 8.15.0 asennettin Visual Studiossa projektiin NuGet-pakettina, joka on yleinen tapa jakaa koodia .NET-alustalla.

3.2 Dokumenttityyppien luominen

Umbraco-asennuksen mukana toimitetaan selaimen kautta käytettävä sisällönhallintasovellus (Backoffice). Verkkosivuston sisältö määritellään, syötetään ja hallitaan sovelluksessa. Kehitysvaiheessa sovelluksen Settings-välilehdellä sisältö määritellään dokumenttityyppien, ominaisuuksien, näkymien, tietotyyppien ja

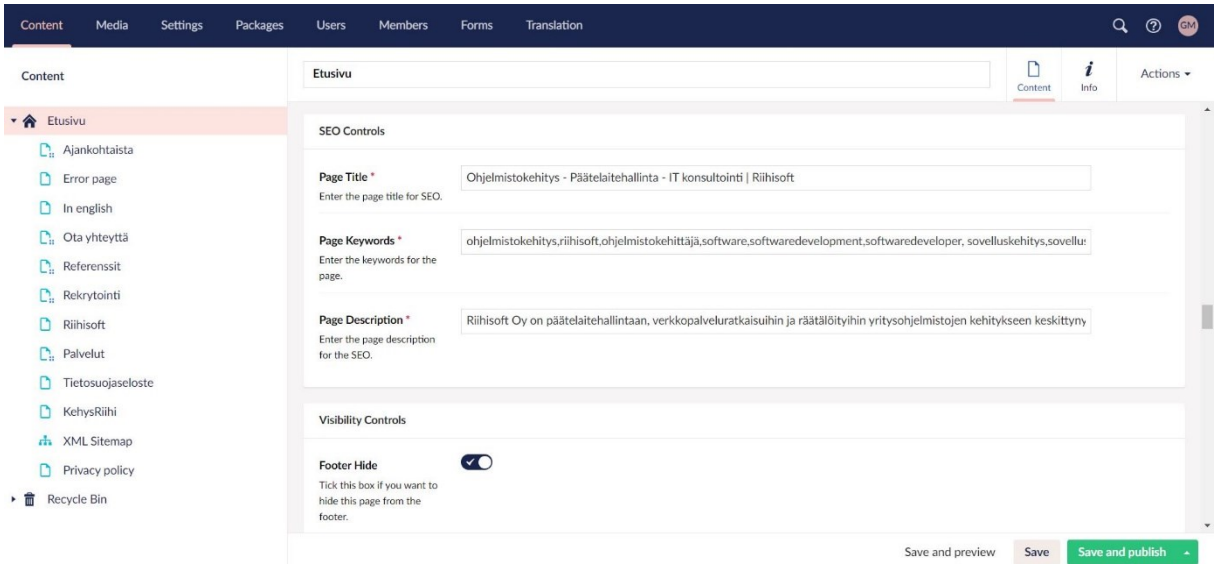
ominaisuusmuokkaajien kautta. Loppukäyttäjä hallitsee sisältöä sovelluksessa Content-välilehdellä sisältösolmujen muodostaman sisältöpuun kautta (Kuva 2). Verkkosivuston sisältöpuussa alisivut listautuvat etusivu-sisältösolmun alle, sillä alisivujen dokumenttityypit ovat määritelty etusivu-dokumenttityypin jälkeläisiksi. Määrittelemällä jälkeläisiä sisältöpuu järjestyy intuitiivisesti käytettäväksi loppukäyttäjälle sekä sisältöä voidaan kysellä puurakenteen perustella.

Kuva 2. Etusivu sisältöpuussa.



Dokumenttityypit jaettiin sivut-, elementit- ja kompositiot-hakemistoihin. Sivut-hakemiston dokumenttityypit edustavat verkkosivuston yksittäisiä sisällöltään uniikkeja sivuja, kuten etusivu. Sivut perivät kompositiot-hakemiston dokumenttityyppejä, jotka edustavat sivuille yhteisiä tietoja, kuten SEO Controls -dokumenttityyppi, joka sisältää sivukohtaisia asetuksia hakukoneoptimointia varten (Kuva 3). Kompositiot-hakemiston dokumenttityypit perivät dokumenttityyppejä elementit-hakemistosta, jotka muodostavat yksityiskohtaisimpia tietokokonaisuuksia. Luomalla perittäviä dokumenttityyppejä vähennettiin ominaisuuksien toistuvuutta dokumenttityypeissä suoraviivaistaen hallittavuutta.

Kuva 3. Etusivu-dokumenttityyppi.

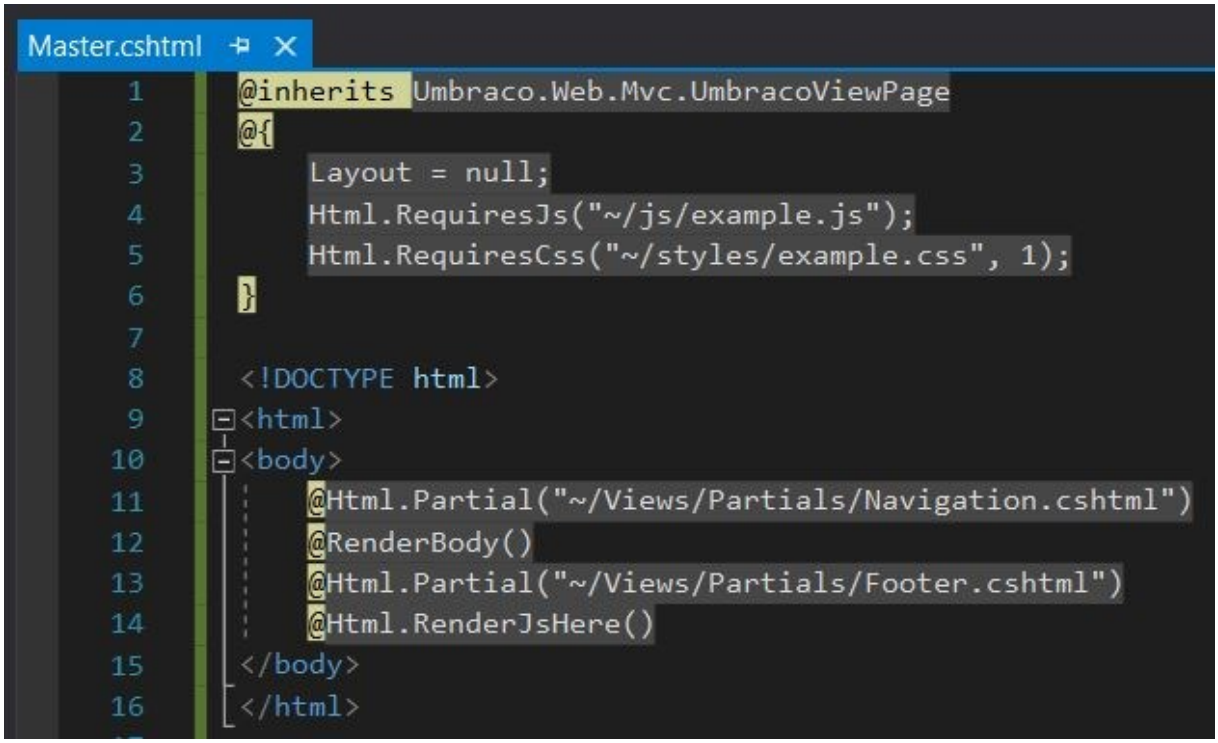


3.3 Näkymien luominen

Luodut dokumenttityypit tarvitsevat tavan esittää säilyttämäänsä tietoa. Sivut-hakemiston dokumenttityypeille luotiin yksilöidyt näkymät (Template), jotka osoitettiin dokumenttityyppien käytettäväksi. Dokumenttityypille voidaan osoittaa yksi tai useampi näkymä, jolloin loppukäyttäjä sisältöä luodessaan valitsee käytettävän näkymän.

Näkymille luotiin päänäkymä, joka sisältää sivuille yhteisiä tietoja ja elementtejä, kuten liitännäisiä, navigaation ja alatunnisteen. Päänäkymä peritään näkymässä Razor-syntaksissa Layout-määrittelyllä, joka osoittaa päänäkymän tiedostonimeen. Kuten kuvassa 4 esitetään päänäkymään määriteltiin JavaScript- ja CSS-tyylitiedostot sekä sisällön renderöintikomento, jolla alinäkymiä kutsuttaessa niiden sisältö injektoituu päänäkymän runkoon.

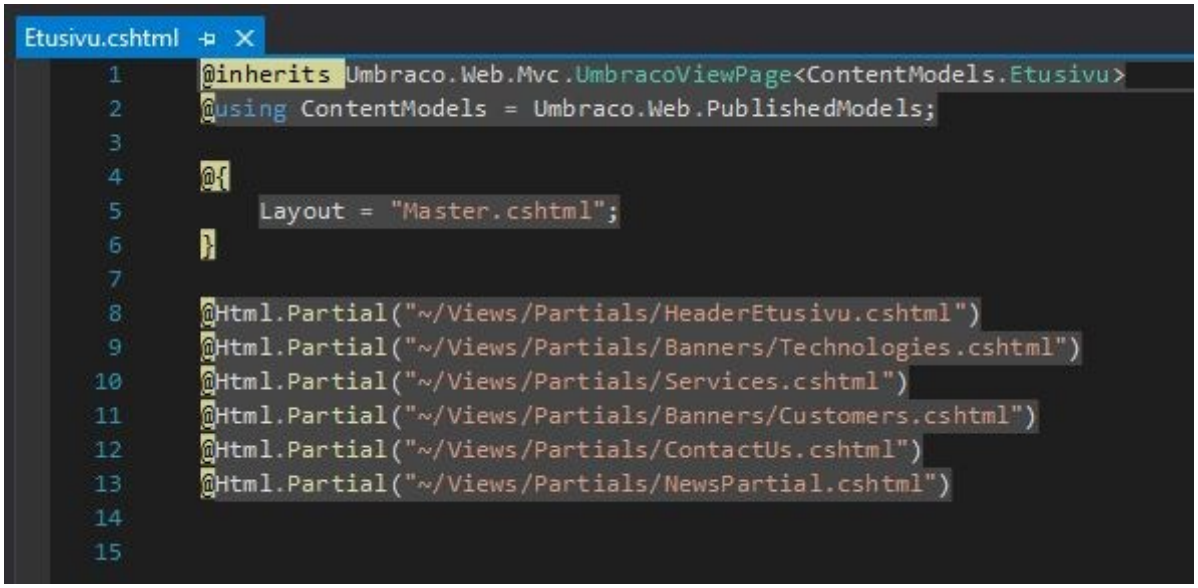
Kuva 4. Master-näkymä.



3.3.1 Osittaisten näkymien luominen

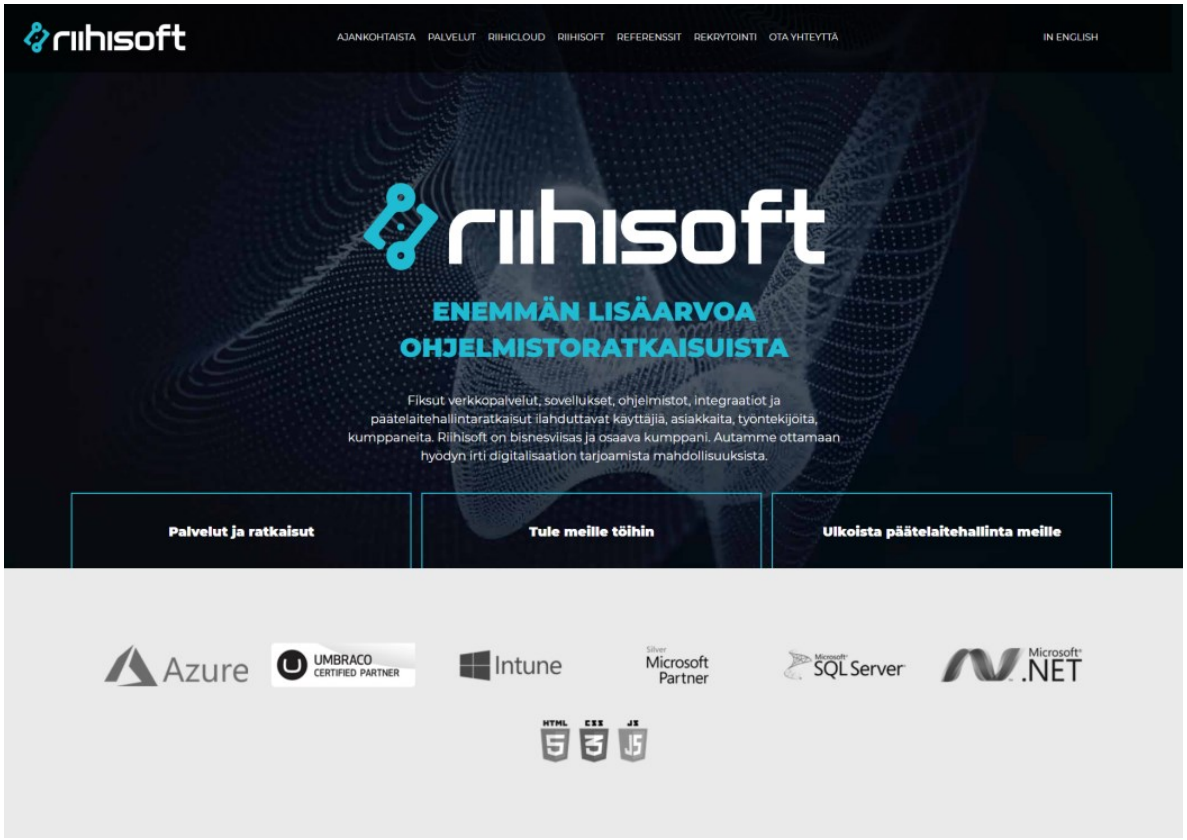
ASP.NET MVC:n osittaiset näkymät (Partial View) ovat näkymiä, joihin sijoitettiin näkymien uudelleen käytettäviä osia, kuten sivun otsake. Osittaiset näkymät vähensivät koodin toistuvuutta ja suoraviivaistivat hallintaa, sillä osiota ei muodostettu HTML-merkintäkoodissa uudelleen jokaisella näkymällä. Osittainen näkymä renderöityy Html.Partial-komennolla. Komento on yksi ASP.NET MVC:n tarjoamista HtmlHelper-luokan metodeista, joilla käsitellä HTML-merkkijonoja (Kuva 5).

Kuva 5. Etusivu-näkymä.



Kuvassa 5 esitettävällä etusivu-dokumenttityypin näkymällä renderöidään otsakkeen jälkeen Technologies-elementin osittainen näkymä esittämään yrityksen hyödyntämiä teknologioita (Kuva 6). Etusivun näkymällä elementin sisältö kysellään Umbracossa julkaistusta sisällöstä hyödyntäen Razor-syntaksia (Kuva 7). Sisältö saadaan content-muuttujaan kyselemällä yksittäisen sisältösolmun sisältöä tunnisteen avulla. Umbracossa uuden sisältösolmun julkaiseminen luo sisältösolmulle yksilöivän tunnisteen. Technologies-elementin sisältö voidaan kysellään tunnisteella, sillä elementin sisältö hallitaan etusivu-dokumenttityypissä ja käyttötapauksessa ollaan varmoja että loppukäyttäjä ei tule lisäämään tai poistamaan etusivu-dokumenttityyppistä sisältösolmua verkksivustolta. Etusivun lisäksi muut sivut, jotka esittävät kyseistä elementtiä hakevat elementtiin sisällön etusivun sisältösolmulta, joten elementin sisältö on identtinen jokaisella sivulla.

Kuva 6. Etusivu.



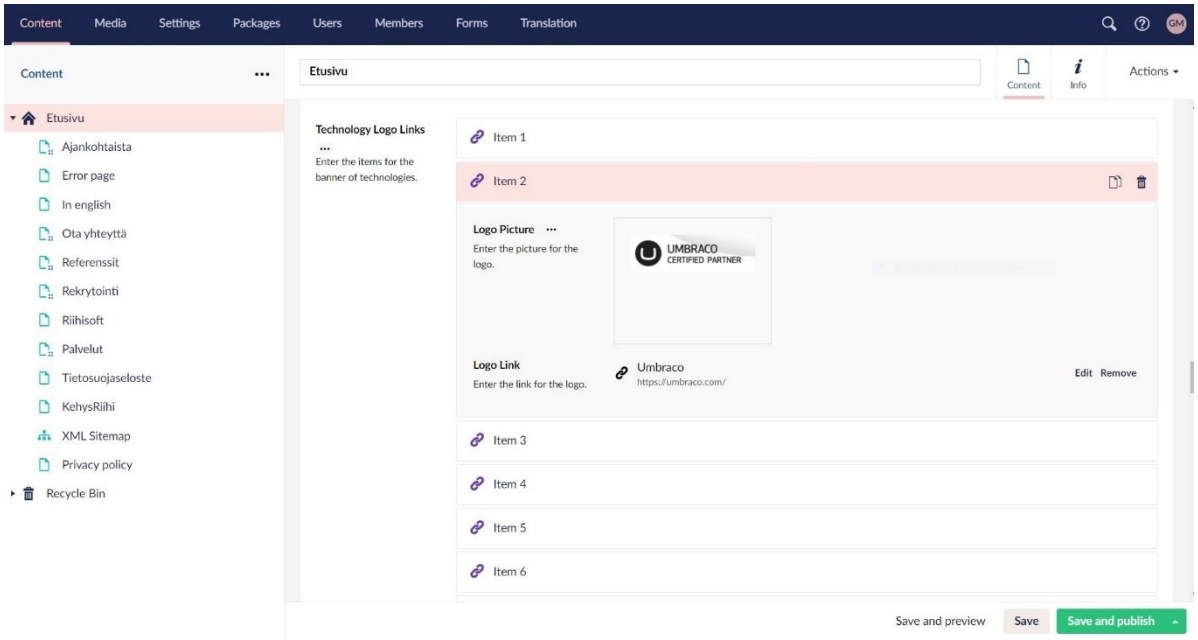
Kuva 7. Technologies- osittainen näkymä.

Technologies.cshtml

```
@inherits Umbraco.Web.Mvc.UmbracoViewPage
@using Umbraco.Web.Models

@{
    var content = Umbraco.Content(1067);
    var logoLinks = content.Value<IEnumerable<IPublishedElement>>("logoLinks");
}

<section id="teknologiat">
    <div class="grid-container">
        <div class="grid-x grid-margin-x small-up-2 medium-up-4 large-up-6 align-center">

            @if (logoLinks != null && logoLinks.Any())
            {
                foreach (var item in logoLinks)
                {
                    var link = item.Value<Link>("logoLink");
                    var linkUrl = link.Url;
                    var picture = item.Value<IPublishedContent>("logoPicture")?.Url();

                    <div class="cell">
                        <a href="@(!string.IsNullOrWhiteSpace(linkUrl) ? linkUrl : null)" target="@link.Target">
                            <img src="@picture" alt="@link.Name" />
                        </a>
                    </div>
                }
            }
        </div>
    </div>
</section>
```

Kuten kuvan 7 koodissa Umbracossa julkaistua sisältöä voidaan kysellä UmbracoHelper-määrittelyllä IPublishedContent-muotoisena. IPublishedContent on vahvasti tyypitetty malli, jolla renderöidä Umbracossa julkaistua sisältöä. Content-muuttujaan kysellystä sisällöstä haetaan Nested Content -muotoista arvoa logoLinks-aliaksen perusteella. Dokumenttityyppien ominaisuuksille määritellään luomisvaiheessa yksilöivät aliakset. Nested Content on Umbracon listaominaisuusmuokkaaja, joka käyttää dokumenttityyppejä määrittääkseen listan kohteen skeeman. Kysellyt elementit ovat elementit-hakemiston dokumenttityyppiä ja sisältävät kuvan ja linkin, edustaen osittaisella näkymällä esitettäviä yksittäisiä teknologia-elementtejä (Kuva 8).

Kuva 8. Technologies-elementti Umbracossa.



3.3.2 Makro-osat

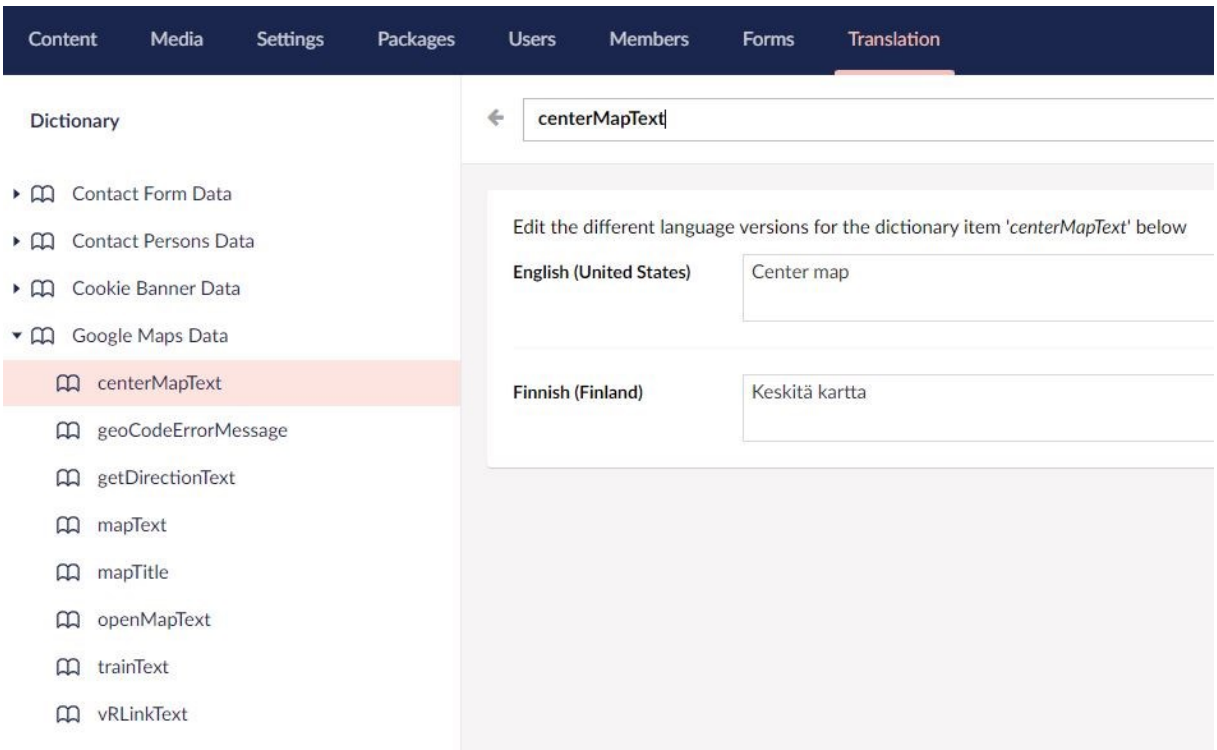
Osittaisten näkymien tapaan makro-osat (Macro) sisältävät uudelleen käytettäviä elementtejä. Erona osittaisiin näkymiin makro-osia voidaan upottaa sivuille Umbracossa grid- ja rich text area -ominaisuusmuokkaajien kautta. Makro-osille voidaan määritellä välimuistin käyttöä ja loppukäyttäjän hallitsemia parametreja, jotka edustavat makron asetuksia tai esitettävää sisältöä. Verkkosivustolle toteutettiin makro-osana Google Maps -kartta, joka esittää yrityksen toimipisteen sijainnin (Kuva 9). Loppukäyttäjä voi upottaa

kartan sivuille, joiden dokumenttityypit tarjoavat grid- ja rich text area -ominaisuusmuokkaajia hyödyntäviä osioita.

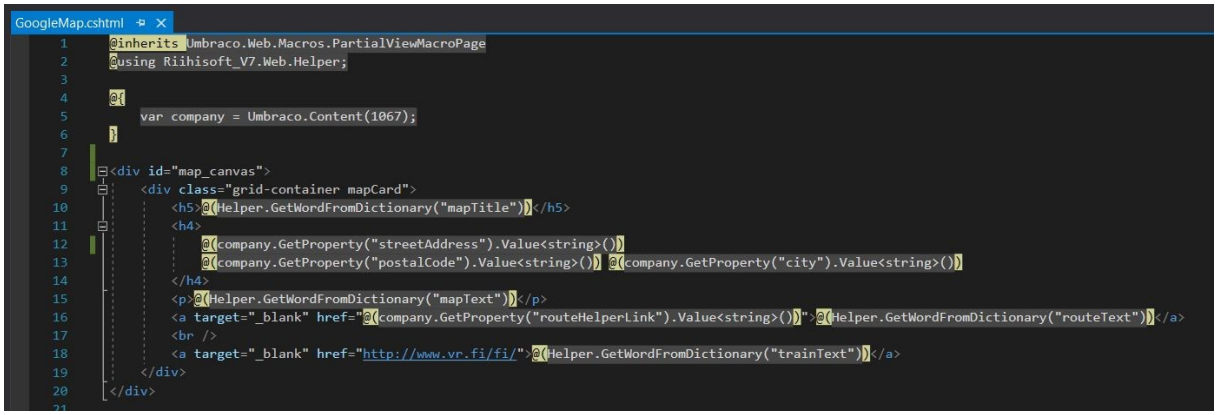
Kuva 9. Google Maps -kartta.

Kartan sijaintitiedot ja ulkoasun hallinta määriteltiin etusivu-dokumenttityyppiin. Kartta esitetään verkkosivustolla suomen- ja englanninkielisillä sivuilla, joten kartan tekstit määriteltiin Umbracon sanakirjaan, jolloin sanakirjan avainsanan perusteella karttaan renderöityy sisältö sivun kielen perusteella (Kuva 10). Kartan näkymällä Umbracon sanakirjasta sanoja kysellään GetWordFromDictionary-metodin avulla (Kuva 11), joka luotiin Helpers-hakemistoon (Kuva 12).

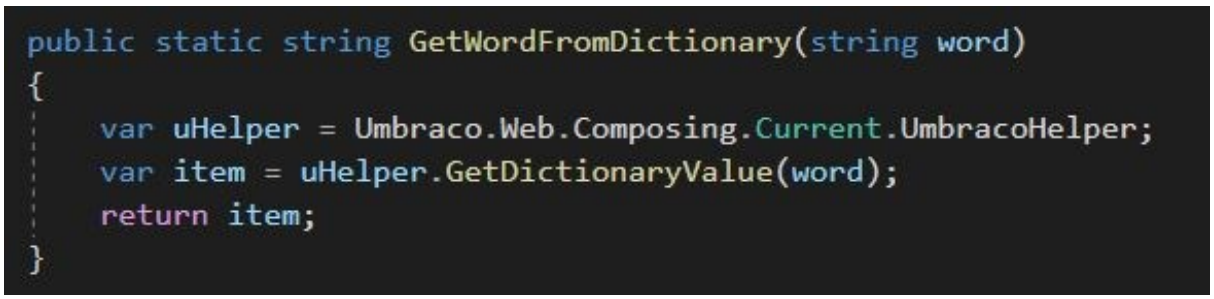
Kuva 10. Umbracon sanakirja.



Kuva 11. GoogleMap-näkymä.



Kuva 12. GetWordFromDictionary-metodi.



3.4 Yhteydenottolomake

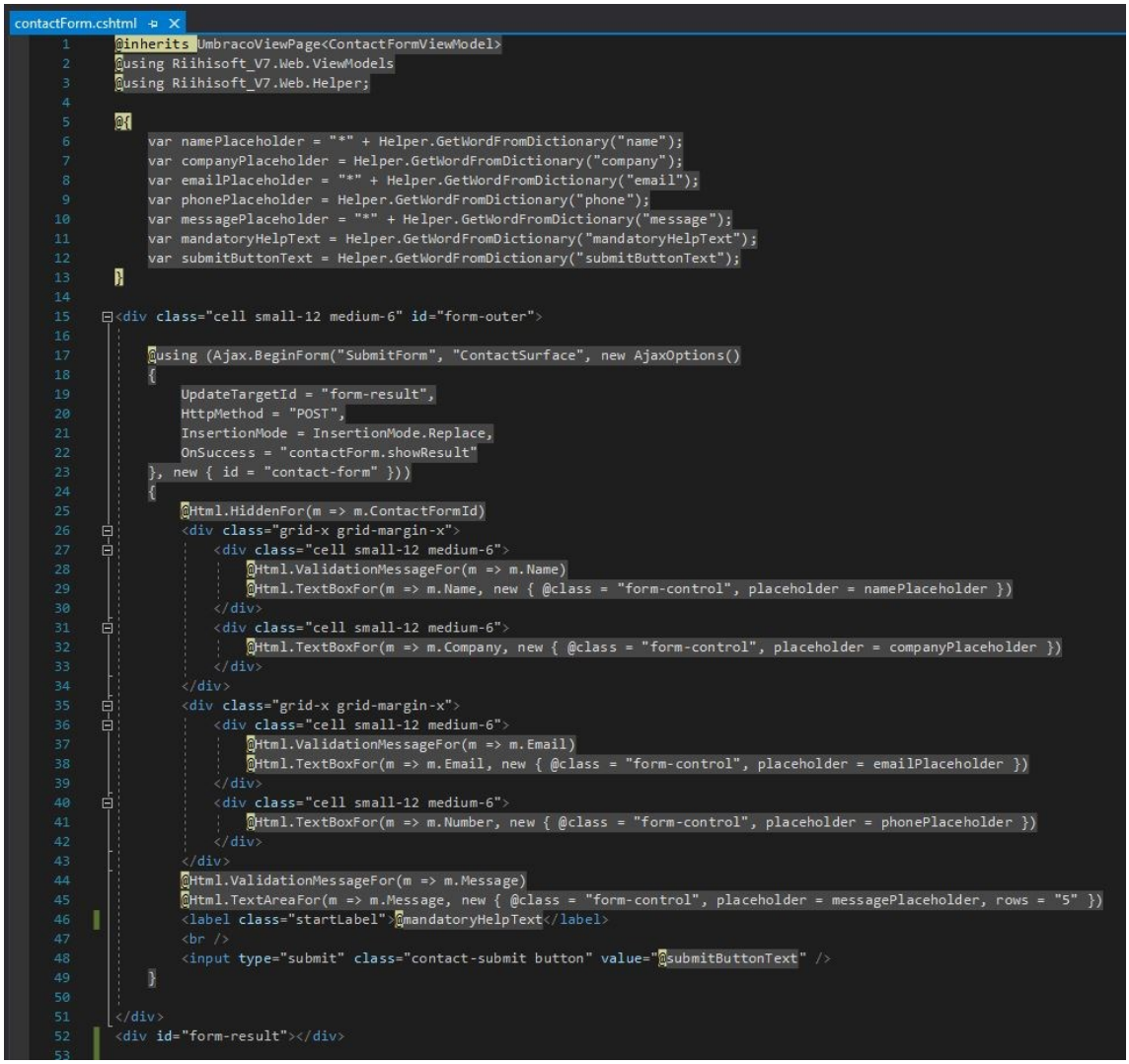
Verkkosivustolle toteutettiin yhteydenottolomake madaltamaan asiakkaan kynnystä ottaa yhteyttä yritykseen. Lomake toteutettiin MVC-mallille ominaisesti käsittelijän, mallin ja näkymän avulla. Lomakkeen tietojen kuvaamiseksi ja validoimiseksi luotiin näkymämalli (View Model). Näkymämalli eroaa tavallisesta mallista kuvaamalla tiedot vain näkymää varten. Näkymämallissa arvoille osoitetaan validointiattribuutteja ilmaisemaan arvon syöttämisen välttämättömyyttä, syötetyn sähköpostiosoitteen muodon tarkistamista sekä epäonnistuneisiin validointeihin palautettavia virheviestejä (Kuva 13). Lomakkeen lähetyksessä MVC kytkee näkymämallista validointiattribuutit käyttöliittymään käynnistäen lomakkeelle JavaScript-validaation perustuen attribuutteihin. JavaScript-validaation onnistuessa lomake lähetetään palvelimelle.

Kuva 13. Yhteydenottolomakkeen näkymämalli.

```
public class ContactFormViewModel
{
    [Required(ErrorMessage = "Syötä nimi.")]
    7 references | giamatikainen, 127 days ago | 1 author, 1 change
    public string Name { get; set; }

    4 references | giamatikainen, 127 days ago | 1 author, 1 change
    public string Company { get; set; }

    [Required(ErrorMessage = "Syötä sähköpostiosoite.")]
    [EmailAddress(ErrorMessage = "Syötä sähköposti oikeassa muodossa.")]
    6 references | giamatikainen, 127 days ago | 1 author, 1 change
    public string Email { get; set; }

    4 references | giamatikainen, 113 days ago | 1 author, 2 changes
    public string Number { get; set; }

    [Required(ErrorMessage = "Kerro meille, miten voimme auttaa.")]
    5 references | giamatikainen, 127 days ago | 1 author, 1 change
    public string Message { get; set; }

    4 references | giamatikainen, 127 days ago | 1 author, 1 change
    public int ContactFormId { get; set; }
}
```

Lomakkeen kentät luotiin HtmlHelper-luokan Html.TextBoxFor-metodilla, jolle määriteltiin käytettävä malli, luokka sekä paikanpitäjäteksti (Kuva 14). Lomakkeen lähetys suoritetaan ASP.NET MVC:n tarjoaman Ajax-luokan Ajax.BeginForm-metodilla, joka mahdollistaa lomakkeen lähettämisen palvelimelle ilman koko sivun palauttamista. Lomakkeen AjaxOptions-parametriin määritelty OnSuccess-ominaisuus kutsuu contactForm-mallin showResult-nimistä JavaScript-funktiota, joka piilottaa lomakkeen ja esittää viestin käyttäjälle lomakkeen lähettämisen onnistumisesta. InsertionMode-ominaisuus kertoo, että sisältö korvataan ja UpdateTargetId-ominaisuus määrittää tunnisteella elementin, jonka sisältö korvataan lomakkeen lähetyksen jälkeen.

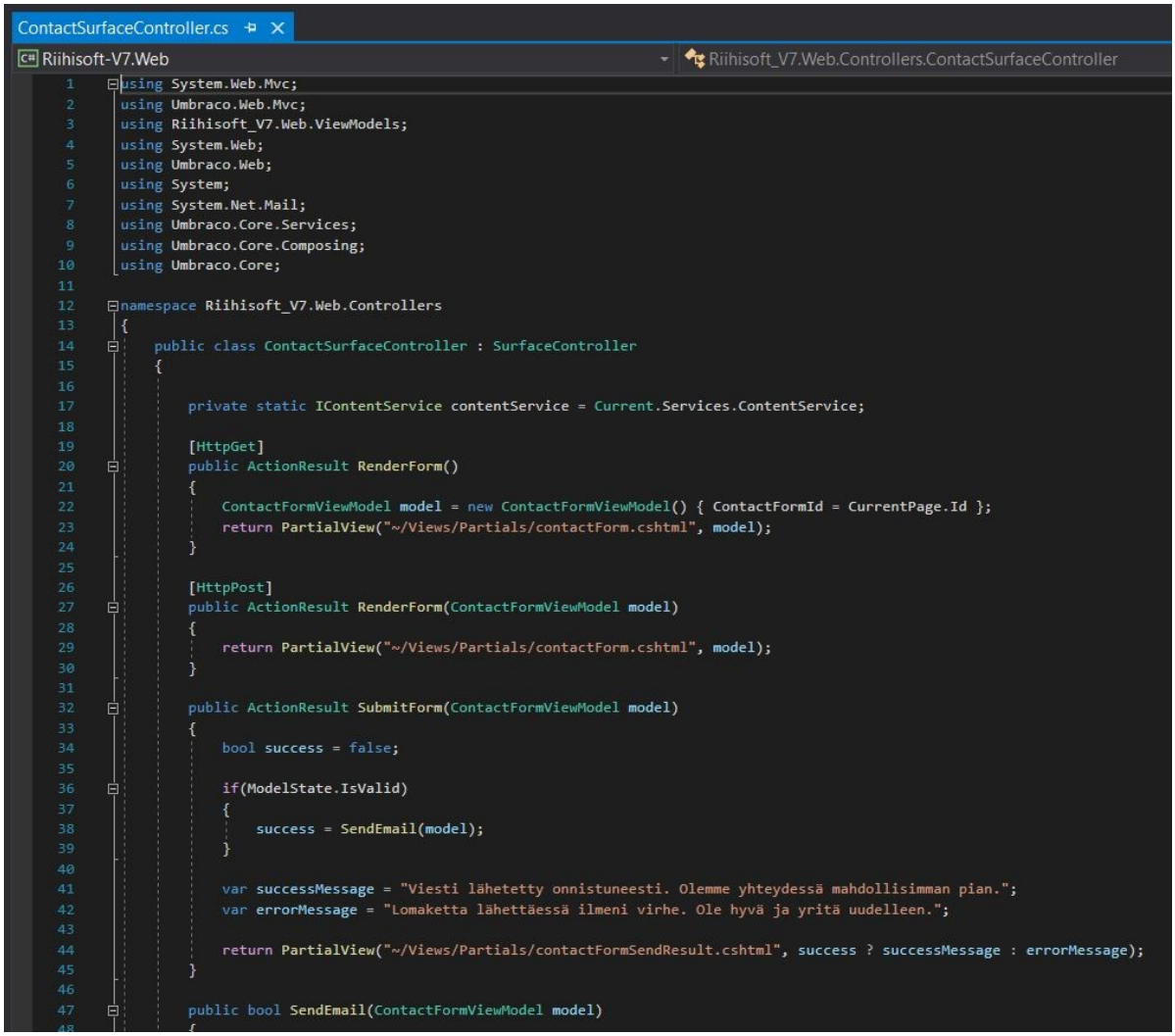
Kuva 14. Yhteydenottolomakkeen näkymä.



```
contactForm.cshtml
@inherits UmbracoViewPage<ContactFormViewModel>
@using Riihisoft_V7.Web.ViewModels
@using Riihisoft_V7.Web.Helper;

@{
    var namePlaceholder = "*" + Helper.GetWordFromDictionary("name");
    var companyPlaceholder = Helper.GetWordFromDictionary("company");
    var emailPlaceholder = "*" + Helper.GetWordFromDictionary("email");
    var phonePlaceholder = Helper.GetWordFromDictionary("phone");
    var messagePlaceholder = "*" + Helper.GetWordFromDictionary("message");
    var mandatoryHelpText = Helper.GetWordFromDictionary("mandatoryHelpText");
    var submitButtonText = Helper.GetWordFromDictionary("submitButtonText");
}

<div class="cell small-12 medium-6" id="form-outer">

    @using (Ajax.BeginForm("SubmitForm", "ContactSurface", new AjaxOptions()
    {
        UpdateTargetId = "form-result",
        HttpMethod = "POST",
        InsertionMode = InsertionMode.Replace,
        OnSuccess = "contactForm.showResult"
    }, new { id = "contact-form" }))
    {
        @Html.HiddenFor(m => m.ContactFormId)
        <div class="grid-x grid-margin-x">
            <div class="cell small-12 medium-6">
                @Html.ValidationMessageFor(m => m.Name)
                @Html.TextBoxFor(m => m.Name, new { @class = "form-control", placeholder = namePlaceholder })
            </div>
            <div class="cell small-12 medium-6">
                @Html.TextBoxFor(m => m.Company, new { @class = "form-control", placeholder = companyPlaceholder })
            </div>
        </div>
        <div class="grid-x grid-margin-x">
            <div class="cell small-12 medium-6">
                @Html.ValidationMessageFor(m => m.Email)
                @Html.TextBoxFor(m => m.Email, new { @class = "form-control", placeholder = emailPlaceholder })
            </div>
            <div class="cell small-12 medium-6">
                @Html.TextBoxFor(m => m.Number, new { @class = "form-control", placeholder = phonePlaceholder })
            </div>
        </div>
        @Html.ValidationMessageFor(m => m.Message)
        @Html.TextAreaFor(m => m.Message, new { @class = "form-control", placeholder = messagePlaceholder, rows = "5" })
        <label class="startLabel">@mandatoryHelpText</label>
        <br />
        <input type="submit" class="contact-submit button" value="@submitButtonText" />
    }

</div>
<div id="form-result"></div>
```

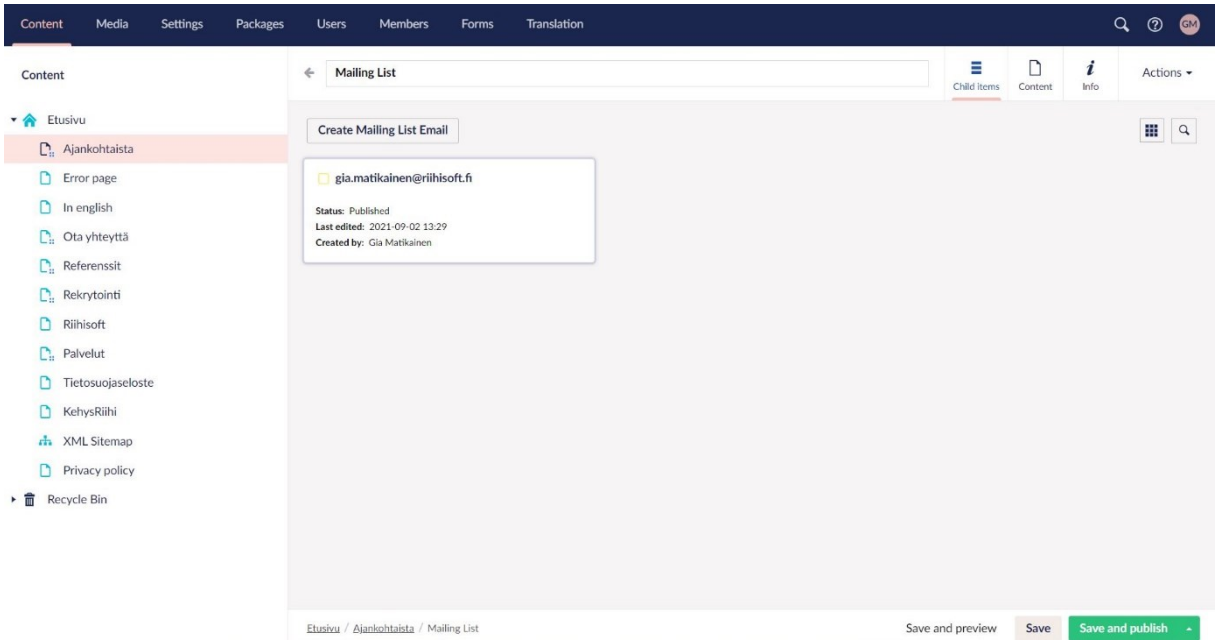
Lomakkeen käsittelijäksi luotiin Umbracon pintakäsittelijä (Surface Controller), joka tavallisen MVC-käsittelijän ominaisuuksien lisäksi tarjoaa Umbracolle hyödyllisiä ominaisuuksia, kuten automaattisen reitityksen (Kuva 15). Pintakäsittelijä on interaktiossa sivun käyttöliittymän renderöinnin kanssa eli se palvelee lomakkeen käyttötarkoitusta hyvin. Käsittelijään luotiin metodit lomakkeen renderöimiselle ja lähettämiselle. Onnistuneesta lomakkeen lähettämisestä lähetetään sähköposti yhteydenottajalle sekä yrityksen edustajalle, joka vastaa yhteydenottopyynnöistä. Sähköpostin lähettämiselle luotiin metodi, jota kutsutaan lomakkeen lähettämisestä vastaavassa metodissa lomakkeen validoinnin ja lähetyksen onnistuttua.

Kuva 15. Yhteydenottolomakkeen käsittelijä.



Yhteydenottopyynnön tietojen säilyvyyden varmistamiseksi tallennetaan tiedot Umbracoon dokumenttityyppeihin, jotka luotiin yhteydenottopyyntöjen tallentamiselle (Kuva 16). Lomakkeen käsittelijään luodaan metodi, joka hyödyntää contentService-palvelua, jolla käsitellä Umbracossa julkaistua sisältöä käsittelijässä IContentService-rajapinnan kautta (Kuva 17). ContactEmailContent-muuttujassa hakeudutaan sisältösolmuun, johon yhteydenottopyynnön tiedot halutaan tallentaa (Kuva 18). Lomakkeen mallia hyödyntäen ContentService-palvelu tarjoaa toiminnallisuudet sisällön luomiselle, dokumenttityypin ominaisuuksien arvojen asettamiselle sekä sisällön tallentamisen.

Kuva 16. Umbracoon tallentunut sisältö.



Kuva 17. ContentService-palvelu.

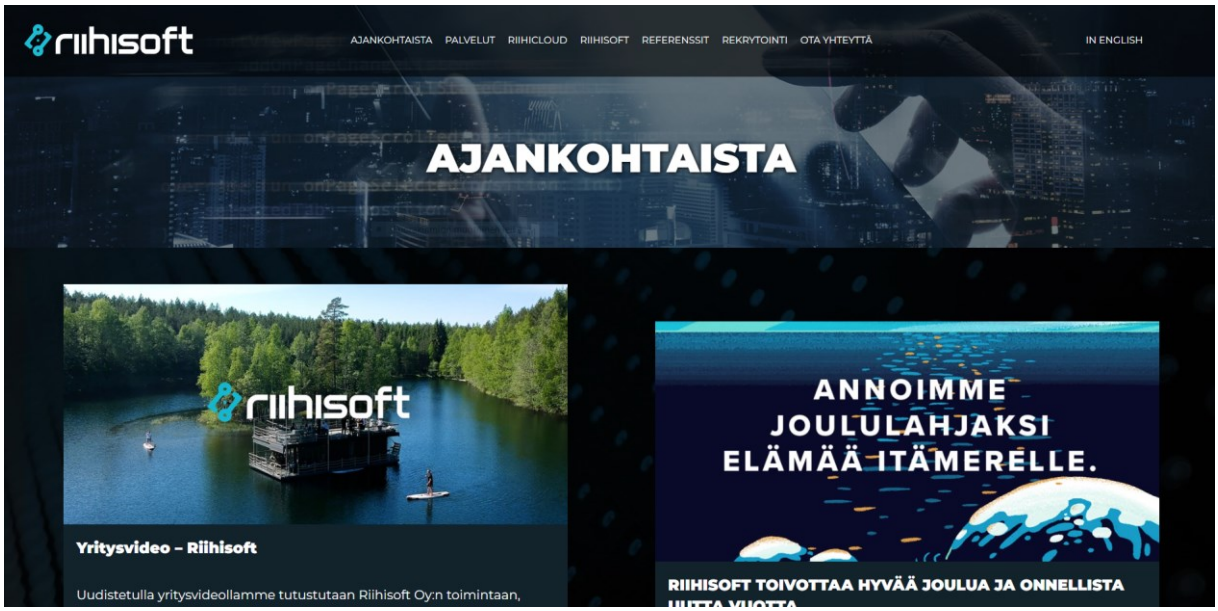
```
private static IContentService contentService = Current.Services.ContentService;
```

Kuva 18. Sisällön tallentaminen Umbracoon käsittelijän metodilla.

```
var ContactEmailContent = contentService.GetById(1087);
var content = contentService.CreateContent(model.Name + " " + DateTime.Now.ToString("dd.MM.yyyy HH:mm:ss"), ContactEmailContent.GetUdi(), "contactEmail");
content.SetValue("MessageName", model.Name);
content.SetValue("messageCompany", model.Company);
content.SetValue("messageEmail", model.Email);
content.SetValue("messagePhone", model.Number);
content.SetValue("message", model.Message);
var nodeName = contentService.GetById(model.ContactFormId).Name;
content.SetValue("formId", nodeName);
contentService.Save(content);
```

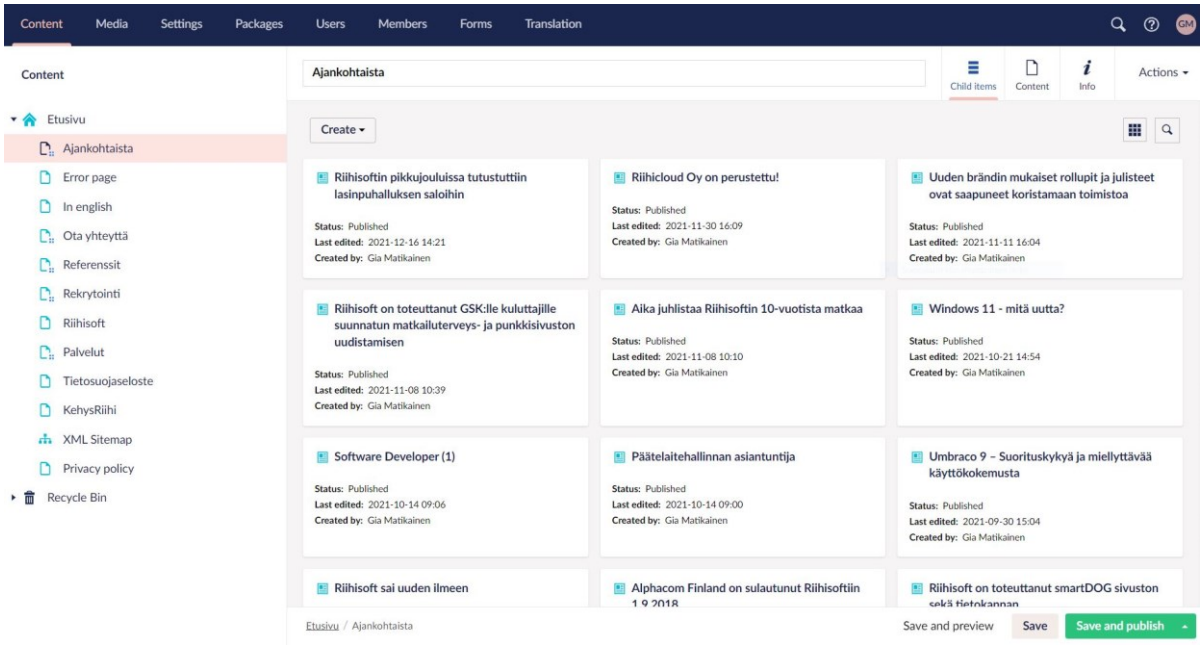
3.5 Ajankohtaista-sivu

Kuva 19. Ajankohtaista-sivu.



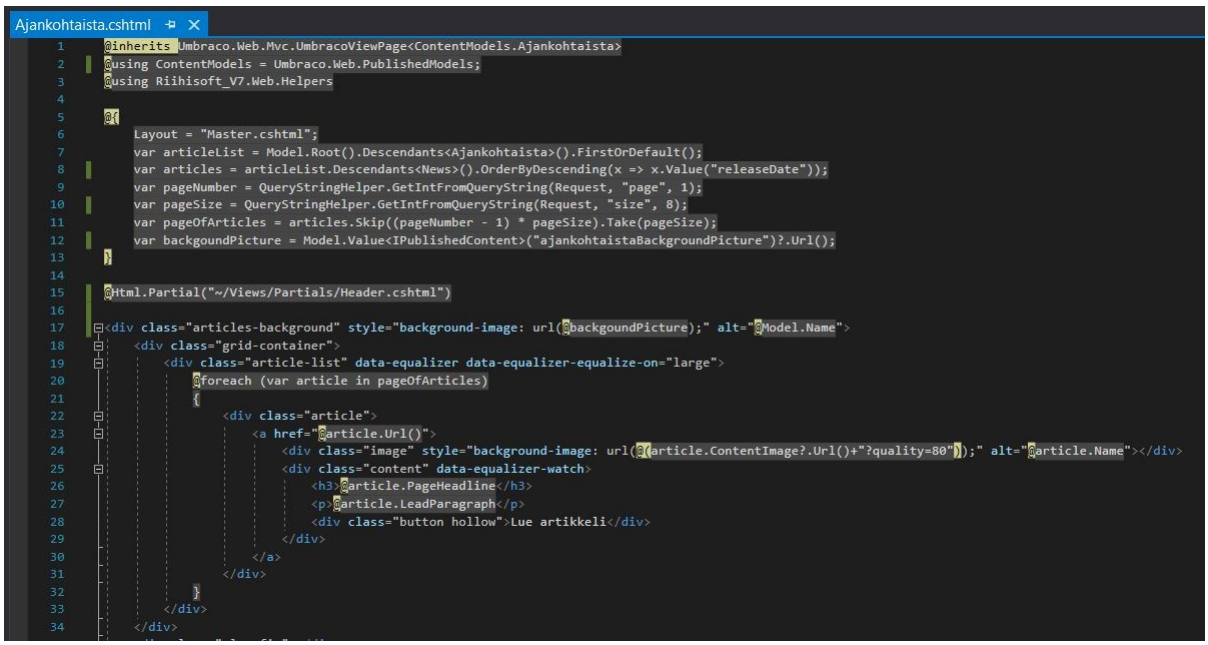
Ajankohtaista-sivulle julkaistaan artikkeleita yrityksen ajankohtaisista aiheista, uusista teknologioista sekä avoimista työpaikoista (Kuva 19). Sivu hyödyntää ajankohtaista-dokumenttityyppiä sekä yksittäistä artikkelia edustavaa news-dokumenttityyppiä. News-dokumenttityyppi on määritelty Umbracossa ajankohtaista-dokumenttityypin jälkeläiseksi, joten sisältöpuussa artikkelit listautuvat ajankohtaista-sivun alle (Kuva 20).

Kuva 20. Ajankohtaista-sivun artikkelit.



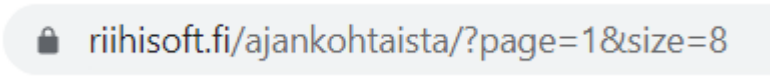
Ajankohtaista-sivulla esitetään kahdeksan viimeisintä artikkelia päiväys-ominaisuuden arvon perusteella (Kuva 21). Sivun näkymässä articleList-muuttujaan kysellään sisältöpuun juuren jälkeläisiä, jotka toteuttavat ajankohtaista-mallia eli ajankohtaista-sivun sisältösolmun sisältöä. Articles-muuttujaan poimitaan articleList-muuttujasta päiväykseen perustuen laskevasti jälkeläiset, jotka toteuttavat news-mallia eli artikkelit.

Kuva 21. Ajankohtaista-näkymä.



Artikkeleiden järjestäminen sivuille toteutettiin kuvassa 22 esitettäviä url-osoitteen parametreja hyödyntäen. PageNumber-muuttujan arvo edustaa url-osoitteen page-parametria eli esitettävän sivun numeroa. PageSize-muuttujan arvo edustaa url-osoitteen size-parametria, joka kertoo sivukohtaisen artikkeleiden kappalemäärän. Url-parametreja käsitellään QueryStringHelper-luokan GetIntFromQueryString-metodin avulla, joka luotiin projektin Helpers-hakemistoon. Url-osoitteen parametrien muuttuessa metodi palauttaa url-osoitteesta parametrin arvon kokonaislukuna. PageOfArticles-muuttuja poimii kaikista artikkeleista sivunumeron ja sivukohtaisen artikkeleiden kappalemäärän perusteella artikkelit esitettäväksi sivuille. Artikkeleiden selauspainikkeille määritellyt toiminnallisuudet muuttavat url-parametrien arvoja, joten artikkeleita voidaan selata sivukohtaisesti.

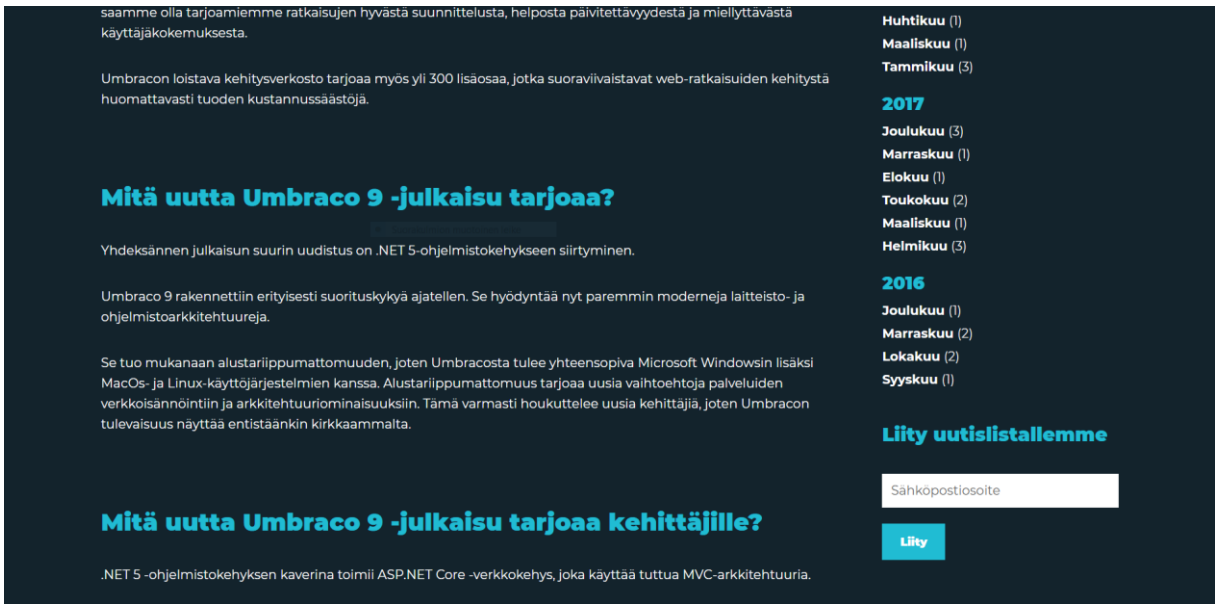
Kuva 22. Url-parametrit.



3.6 Kustomoitu tapahtuma

Artikkelin sivulla on mahdollista liittyä sähköpostiosoitelistalle syöttämällä sähköpostiosoite kenttään (Kuva 23). Listan sähköpostiosoitteisiin toimitetaan ilmoituksia uusista verkkosivustolle julkaistuista artikkeleista. Umbracossa voidaan seurata tapahtumia ja lisätä kustomoitua koodia tapahtumaan ennen ja jälkeen suorittamisen hyödyntäen komponentteja (Component). Komponentilla kustomoidaan Umbraco-sovelluksen käyttäytymistä muodostumisvaiheessa (Start up) muuttamalla Umbracon ydintoiminnallisuuksia tai rekisteröimällä kustomoitua koodia tapahtumien seuraamiseksi.

Kuva 23. Sähköpostiosoitelistalle liittymisen kenttä.



Uuden artikkelin julkaisun tapahtuman seuraamiseksi projektiin luodaan Composers-hakemisto, johon luodaan luokka, joka tunnistautuu koodikomponentiksi toteuttamalla Umbracon IComponent-rajapintaa (Kuva 24). Umbracoon luodulle komponentille tulee määritellä metodit, jotka suorittaa Umbraco-sovelluksen alustuksessa (Initialize) sekä suorituksen päättyessä (Terminate). Alustuksessa hyödynnettävä Initialize-metodi määrittelee että komponentti seuraa Umbracon julkaisutapahtumaa. Metodissa määritellään julkaisutapahtuman yhteydessä suoritettava ContentServce_Published-metodi, johon kirjoitetaan kustomoitua koodia. Metodissa kysellään Umbracosta julkaistun artikkelin sisältö ja muodostetaan HTML-merkintäkielellä sivu sisältöön perustuen. Sivu lähetetään sähköpostilla Umbracosta haetun listan sähköpostiosoitteisiin.

Kuva 24. Kustomoidun tapahtuman komponentti.

```
namespace Riihisoft_V7.Web.Composers
{
    [RuntimeLevel(MinLevel = RuntimeLevel.Run)]
    public class CustomEventComposer : IUserComposer
    {
        public void Compose(Composition composition)
        {
            composition.Components().Append<CustomEventComponent>();
        }
    }

    public class CustomEventComponent : IComponent
    {
        private static IContentService contentService = Current.Services.ContentService;

        public void Initialize()
        {
            ContentService.Published += ContentService_Published;
        }

        public void Terminate(){}

        private void ContentService_Published(IContentService sender, ContentPublishedEventArgs e)
        {
            //custom functionality here
        }
    }
}
```

3.7 Sisällön siirtäminen

Umbraco säilyttää tiedostoja, sisältöä ja mediaa levyllä sekä tietokannassa. Karkeasti jaettuna tietokannassa säilytetään Umbracoon ja sen rakenteeseen liittyviä tietoja sekä sisältö, kun levyllä säilytetään tiedostoja, konfiguraatioita ja kustomoitua koodia. Umbracon sisällön siirtämiseen Umbraco-instanssien välillä tarjotaan uSync-lisäosa. Lisäosalla voidaan siirtää tietokannan sisältö levylle, jolloin Umbraco-pohjaista verkkosivustoa voidaan versiohallita, kopioida ja siirtää.

Uudelle verkkosivustolle tuotiin uSync-lisäosalla olemassa olevan verkkosivuston Umbracosta artikkeleiden sisällöt. Sisällön siirtäminen manuaalisesti olisi todella työlästä, joten siirtäminen tehtiin uSync-työkalulla. Complete-versio uSync-työkalusta asennettaan Umbracoihin, joiden välillä dokumenttityypit ja sisältöä halutaan siirtää. Työkalulla valitaan siirrettäväksi Umbracon sisältöpuusta ajankohtaista-dokumenttityypit sekä media. Työkalu luo paketin XML-tiedostoja, jotka siirretään vie-toiminnolla kehitettävän verkkosivuston

Umbracossa. Näin saadaan uudelle verkkosivustolle tehokkaasti käyttöön ajankohtaista-artikkeleiden dokumenttityypit sisältöineen.

3.8 Tuotantoympäristö

Verkkosivusto isännöidään IIS-palvelinohjelmistolla virtuaalikoneella. Palvelimelle luotiin tietokanta, johon kopioitiin kehitystietokannan sisältö sekä luotiin uusi hakemisto, johon vietiin verkkosivuston tarvitsemat tiedostot. Verkkosivustoa testattiin yrityksen henkilöstön toimesta IIS:n isännöimän verkkosivuston testiversion avulla. Korjauksia tehtiin liittyen responsiivisuuteen ja latausnopeuksia parannettiin lisäämällä verkkosivuston mediaan laatua ja kokoa rajoittavia parametreja. Käyttöönotossa IIS:ssä olemassa olevan yrityksen verkkosivuston hakemisto osoitettiin uuden verkkosivuston hakemistoon.

4 Yhteenveto

Verkkosivuston toteuttaminen syvensi merkittävästi osaamista sisällönhallintajärjestelmien mahdollisuuksista, web-teknologioista ja Microsoftin ohjelmistokehitystyökaluista. Osaamisen syventyessä oivalluksia syntyi enenevissä määrin sovellusmahdollisuuksista lisäten mielenkiintoa entisestään hyödynnettyjä teknologioita kohtaan.

Verkkosivuston toteuttamisen ja opinnäytetyön kirjoittamisen välissä karttunut osaaminen sai oivaltamaan kehityskohtia. Verkkosivuston tyyli- ja JavaScript-tiedostot tulisi minimoida. Visual Studiossa tarjotaan työkalu, jolla tiedostoista poistetaan tarpeettomat tiedot vaikuttamatta selaimen tapaan käsitellä resurssia. Minimoiminen parantaa resurssien latausnopeutta. Osaamisen karttuminen Umbracon osalta lisäsi oivalluksia sisällönhallinnan mahdollisuuksien toteuttamisesta. Verkkosivuston ylläpitäjät tulevat olemaan ohjelmistokehitystä omaavat kehittäjät. Toisessa tapauksessa olisi ollut tarpeen toteuttaa monipuolisemmin hallittavia komponentteja, kuten visuaalisen ilmeen ja elementtien paikkojen hallintaa.

Verkkosivuston suunnittelu ja toteuttaminen etenivät aikataulussa tuottaen lopputulokseksi tavoitteet täyttävän verkkosivuston. Toimeksiantajan palautteen sekä osaamisen syvenemisen perusteella verkkosivuston toteutus onnistui.

Lähteet

Galloway, J., Wilson, B., Allen, K. & Matson, D. (2014). How ASP.NET MVC Fits in with ASP.NET. Teoksessa *Professional ASP.NET MVC 5* (s. 2). https://ebookcentral-proquest-com.ezproxy.hamk.fi/lib/hamk-ebooks/reader.action?docID=1744259

Microsoft. (30.11.2021-a). A tour of the C# language. Haettu 25.12.2021 osoitteesta https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

Microsoft. (11.12.2021-b). Common Language Runtime (CLR) overview. Haettu 25.12.2021 osoitteesta https://docs.microsoft.com/en-us/dotnet/standard/clr

Microsoft. (30.11.2021-c). A tour of the C# language. Haettu 15.12.2021 osoitteesta https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

Microsoft. (11.12.2021-d). Razor syntax reference for ASP.NET Core. Haettu 06.12.2021 osoitteesta https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-6.0

Munro, J. (2015). Introduction to MVC. Teoksessa M. Foley (toim.), *ASP.NET MVC 5 with Bootstrap and Knockout.js: Building Dynamic, Responsive Web Applications* (s. 1). O'Reilly Media, Inc. https://books.google.fi/books?id=DdtvCQAAQBAJ&dq=asp.net+mvc+5&lr=&hl=fi&source=gbs_navlinks_s

Paz, J. (2013). What Is ASP.NET?. Teoksessa E. Buckingham (toim.), Beginning ASP.NET MVC 4 Expert's voice in .NET (s. 1). Apress. https://books.google.fi/books?id=PHzbbSojPYIC&hl=fi&source=gbs_navlinks_s

Quick Start. (n.d.) Structure of an Umbraco page. Haettu 06.12.2021 osoitteesta https://www.uquickstart.com/umbraco-tips-and-tools/structure-of-an-umbraco-page

Umbraco. (n.d.-a). The History of Umbraco. Haettu 06.12.2021 osoitteesta https://umbraco.com/about-us/the-history-of-umbraco/

Umbraco. (n.d.-b). What is Umbraco? Haettu 06.12.2021 osoitteesta https://umbraco.com/knowledge-base/umbraco/

Umbraco. (n.d.-c). Why are we called Umbraco? Haettu 10.12.2021 osoitteesta https://umbraco.com/about-us/why-are-we-called-umbraco/

Umbraco. (n.d.-d). Minimum System Requirements. Haettu 25.12.2021 osoitteesta https://our.umbraco.com/documentation/Fundamentals/Setup/Requirements/

Umbraco. (n.d.-e). What is a Document Type. Haettu 10.12.2021 osoitteesta https://umbraco.tv/videos/umbraco-v7/implementor/fundamentals/document-types/what-is-a-document-type/documentation

Umbraco. (n.d.-f). Document Types. Haettu 01.12.2021 osoitteesta https://our.umbraco.com/documentation/tutorials/creating-basic-site/document-types/

Wahlberg, N. & Sterling, P. (2011). Templates, Markup, and Master Pages. Teoksessa *Umbraco User's Guide* (s. 73). John Wiley & Sons, Incorporated. https://ebookcentral-proquest-com.ezproxy.hamk.fi/lib/hamk-ebooks/reader.action?docID=693228

W3Techs. (26.12.2021-a). Content Management Systems. Haettu 26.12.2021 osoitteesta https://w3techs.com/

W3Techs. (26.12.2021-b). Comparison of the usage statistics of WordPress vs. Umbraco for websites. Haettu 26.12.2021 osoitteesta https://w3techs.com/technologies/comparison/cm-umbraco,cm-wordpress

W3Techs. (26.12.2021-c). Server-side Programming Languages. Haettu 26.12.2021 osoitteesta https://w3techs.com/

W3Techs. (26.12.2021-d). Comparison of the usage statistics of PHP vs. ASP.NET for websites. Haettu 26.12.2021 osoitteesta https://w3techs.com/technologies/comparison/pl-aspnet,pl-php