SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

TIETÄMYKSENHALLINTA-SOVELLUS

TEKIJÄ/T: Timitri Riikonen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma/Tutkinto-ohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Timitri Riikonen	
Työn nimi Tietomyksenhallintasovellus	
Päiväys 25.11.2018	Sivumäärä/Liitteet 32
Ohjaaja(t) lehtori Jussi Koistinen, lehtori Keijo Kuosmanen	
Toimeksiantaja/Yhteistyökumppani(t) Tauno Hyvärinen, Solteq Oyj	

Tiivistelmä

Tämän työn tavoitteena oli luoda Solteq Oyj:lle sovellus, jolla levittää yrityksen sisäistä taitoa ja tietämystä. Työssä luotasiin selainpohjainen sovellus, joka olisi kaikkien yrityksen työntekijöiden käytettävissä. Sovelluksen tulisi olla helppokäyttöinen, sekä työntekijöiden vapaasti käytettävissä ilman rajoituksia.

Tuloksena syntyi selainohjelma, jolla pystytään siirtämään tiedostoja varastoon, Azure Blob Storageen. Sovellus pitää kirjaa siitä, mitä tiedostoja on siirretty ja missä ne fyysisesti sijaitsevat. Samalla käyttäjä pystyy merkitsemään tiedostot ”tagein”. Tiedostonsiirron lisäksi sovelluksella pystytään etsimään tiedostoja tagien avulla, sekä myös hakusanoilla tekstitiedostojen sisältä. Tiedostot voidaan päivittää uudempaan versioon. Tiedostot ja niiden vanhat versiot ovat tallennettavissa sovelluksen avulla.

Sovellus luotiin käyttämällä pääosin ASP.NET-tekniikkaa. Lisäksi käytössä oli myös Cshtml, TypeScript ja jQuery -kielet käyttöliitymää varten, sekä EntityFramework-kehys tietokannan ylläpitoon ja käyttöön.

Avainsanat
Tietämyksenhallinta, ASP.NET, Azure Blob Storage

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Timitri Riikonen			
Title of Thesis Knowledge management software			
Date	25 November 2018	Pages/Appendices	32
Supervisor(s) M. Jussi Koistinen, Senior Lecturer, Mr. Keijo Kuosmanen, Senior Lecturer			
Client Organisation /Partners Tauno Hyvärinen, Solteq Oyj			

Abstract

The goal of this thesis was to create an application for Solteq Oyj, for spreading knowledge and skill within the company. The purpose was to make the application browser based and available for every employee of the company and it should also be easy to use by employees without any restrictions.

The application was created by using the ASP.NET-technology. Cshtml, TypeScript and jQuery languages were also used. EntityFramework was used to maintain the database.

As a result of the project, a browser base application was created. It can be used to transfer files into Azure Blob Storage. The application keeps track of the files and their location. Users can also tag the files with keywords. Files can be searched by using tags or using a text entry to search inside text files. The files can also be updated into a newer version. The files and their older version can be downloaded using the application.

Keywords
Knowledge management, ASP.NET, Azure Blob Storage

SISÄLTÖ

1 JOHDANTO

Nopeasti kasvavassa yrityksessä tietämyksenhallinta on haastavaa. Kun keskitytään vain uuden luomiseen, voi sekä vanhan että uuden opitun tiedon hallitseminen jäädä taka-alalle. Tämä voi luoda ongelmia henkilöstömuutosten myötä, jolloin hallittu tieto auttaisi uusia työntekijöitä integroitumaan yritykseen ja oppimaan työtehtävänsä nopeammin.

Opinnäytetyön alkaessa Inpulse Works Oy:llä ei ollut yhtenäistä linjaa tietämyksen varastoinnille. Yrityksessä on mukana kokeneita osaajia, mutta keskenäistä tiedon jakamista ei ole kehitetty. Syntyi tarve ratkaisulle, jolla tietoa saataisiin jaettua helposti.

Tämän työn tarkoituksena on kehittää alusta, jolla tietoa pystyttäisiin sekä jakamaan että etsimään tehokkaasti. Alustaa kehitetään jatkuvasti esille tulevien tarpeiden mukaan.

2 SOLTEQ

Työ aloitettiin tekemään inPulse Works Oy:lle keväällä 2017. Kesällä 2017 tapahtui yrityskauppa, jossa Solteq Oyj osti inPulsen.

Solteq on 1982 perustettu monitoimialan yritys. Solteqilla on toimipisteet Tampereella, Vantaalla, Jyväskylässä, Seinäjoella, Kuopiossa, Krakovassa, Oslo ja Karlstadissa. Yrityksessä työskentelee noin 550 työntekijää.

Tässä työssä esiteltävät tuotteet ovat Inpulse Worksin aikaisia tuotteita.

2.1 inWorks

inWorks on selainpohjainen asiakastietojärjestelmä energia-alan yrityksille. Sen ominaisuuksia ovat muun muassa:

- Asiakastietojen hallinta
- Käyttöpaikkatietojen hallinta
- Laskutus
- Reskontra
- Raportointi
- Mittalaitehallinta
- Mittarilukemien tuonti
- Sähkösanomien vienti, tuonti ja välitys

inWorksiin sisältyviin hyödykkeisiin kuuluu mm:

- Kaukolämpö ja -kylmä.
- Maakaasu
- Vesi, jätevesi ja hulevesi
- Sähkön siirto ja myynti
- Teollisuushyödyke
- Palveluhyödyke

Sovelluksen tavoitteena on massaoperoida isoja prosesseja, kuten laskutuskauden laskujen massaluonti mittarilukemat ja eri sopimukset huomioon ottaen. Sovellus hallitsee myös kumulatiivisten lukemien ja tuntisarjalukemien tuonnin ulkoisista järjestelmistä. Sähkön laskutusta varten sovellus osaa käsitellä sähköyhtiöiden välillä käytävän sanomien välityksen.

2.2 i4Utilities

i4u on selainpohjainen ratkaisu energiayhtöiden asiakaspalvelulle ja heidän loppuasiakkailleen. Palvelun kautta loppuasiakkaat pääsevät osallistumaan prosesseihin, sekä selaamaan sopimuksiaan ja laskujaan, sekä hoitamaan avoimia tapahtumia energiayhtiön kanssa.

Kuva 1 inpulse4Utilities

Asiakaspalvelu vuorostaan pystyy tehokkaasti hoitamaan prosessejaan, kuten asiakkaiden yhteydenotot, asiakastietojen muutokset ja sopimusmuutokset.

Lisäpalveluihin kuuluu mm. liittymispalvelut, eli LISA. Palvelu tarjoaa sähköyhtiöille sähköliittymienhallinta-sovelluksen, jonka avulla esimerkiksi vesiliittymien tilaus onnistuu samaan aikaan. Sovelluksella voi seurata tilausten etenemistä ja tarvittaessa puuttua niihin.

Häiriötiedotus, eli HÄTI mahdollistaa häiriötietojen massailmoitukset loppuasiakkaille. Sovelluksella voidaan karttanäkymän avulla valita alue, jonka sisällä kaikille yhtiön asiakkaille laitetaan haluttu viesti halutussa muodossa, kuten sms-viestinä, tai kirjeenä.

3 LÄHTÖKOHTA JA TAVOITTEET

Yrityksen henkilökunnalla on paljon tietoa, jota ei ole varastoitu mihinkään. Tämä aineeton pääoma tulisi saada kerättyä hallittavaksi kokonaisuudeksi. Tietoa voi olla myös dokumentteina tietokoneella, johon muilla ei ole pääsyä. Dokumenttien löytäminen voi olla vaikeaa myös oman koneen sisältä. On syntynyt tarve alustalle, jossa tällaisia tilanteita syntyisi vähemmän ja tietoa olisi helpompi levittää.

Tämä työ on jatkokehitystä aiempaan opinnäytetyöhön ”Tietämyksenhallinta, Myynnistä Ylläpitoon”. (Tauno Hyvärinen, 2017)

3.1 Tavoite

Tavoitteena oli luoda sovellus, jolla hallitaan yrityksen sisäistä tietoa tehokkaan tiedonhakuympäristön avulla. Sovellus tulisi olemaan helppokäyttöinen ja henkilöstön vapaasti käytettävissä, jotta tieto leviäisi luontevasti ja nopeasti.

3.2 Vaatimukset

Sovellus tuli tehdä Microsoftin tuotteilla, yrityksen lisensseillä. Sen tuli olla helppokäyttöinen, jotta jokainen työntekijä voi sitä luontevasti käyttää. Hankalasti käytettävän sovelluksen käyttämistä tultaisiin todennköisesti välttelemään. Käyttäjien tuli pystyä lisäämään, sekä muokkaamaan tiedostoja haluamallaan tavalla ilman erikseen määritettyjä rooleja.

Sovelluksen tuli olla selainkäyttöinen. Se tuli toteuttaa ASP.NET-websovelluskehyksellä. Sillä piti pystyä viemään tiedostoja pilveen, sekä tuomaan niitä sieltä käyttäjän koneelle. Vietäviä tiedostoja piti pystyä merkitsemään tagein tiedostojen löytämisen helpottamikseksi. Tiedoston tageja oli pystyttävä muokkaamaan. Tageja piti voida merkitä keskenään synonyymeiksi, jotta tiedostoille ei tarvitsisi laittaa synonyymien takia useampia tageja.

Kun tiedostoa lisätään, käyttöliittymän pitää pystyä ennustamaan mitä tagia käyttäjä kirjoittaa. Jos tagia ei ole olemassa, se lisätään tietokantaan. Tiedostot pitää pystyä päivittämään uusiin versioihin, vanhojen versioiden jäädessä talteen. Dokumentteja tulee olle haettavissa tageilla, sekä tiedostoille liitetyillä vapaamuotoisella tekstirivillä. Tekstirivi on tarkoitettu käytettäväksi virheilmoituksille, jos ladattava tiedosto liittyy virheen ratkaisemiseen.

Lisäksi tutkittiin, pystytäänkö pilveen vietyjen dokumenttien sisällöstä etsimään hakusanoilla.

4 TIETÄMYKSENHALLINTA

Nykypäivän yrityksen arvo mitataan myös sen omistaman tietämyksen mukaan. Tästä puhutaan usein yrityksen aineettomana pääomana. Voidaan sanoa, että modernin yrityksen tärkein resurssi sijaitsee sen työntekijöiden ja asiakkaiden mielissä. Tämän resurssin hallinnasta on monta hyötyä, joita kaikkia ei välttämättä heti havaita (Fernandez;ym., 2010 s. 4).

Otetaan esimerkiksi kuvitteellinen ohjelmistoyritys. Yrityksen on perustanut muutama erittäinen kokenut ohjelmoija. Uusi ohjelmisto kehittyy ja yritys kasvaa. Tulee tarve uusille työntekijöille. Kaikki tieto ohjelmistosta, sen tavoitteista, kehityksesta, ongelmista ja yrityksen asiakkaista, on vielä yrityksen perustajilla. Jos tietämystä ei siirretä uusille työntekijöille tehokkaasti, työtunteja kuluu hukkaan sekä tehottomaan perehdyttämiseen, että perehtymiseen. Kasvavan yrityksen kannalta tietämyksen siirtämiseen on siis tärkeää kiinnittää huomiota.

Keith Bradley on kirjoittanut, että tietämys kaksinkertaistuu 18 kuukaudessa. Teknologian kehitys kiihtyy ideoiden ja tietämyksen tahdissa (Bradley, 1997 s. 54). Tietämyksen lisääntyminen johtaa myös siihen, että tarvittavan perustiedon määrä kasvaa (Fernandez;ym., 2010 s. 6). Voidaan päätellä, että tämä johtaa kasvavassa määrin tilanteisiin, jossa tietoa täytyy etsiä.

Tiedon löytämisen lisäksi on tärkeää käyttää tietoa viisaasti. Oikeanlainen tietämyksenhallinta (knowledge management) parantaa päätöksien tekoa. Kun viisautta pystytään jakamaan, tiedetään mihin tietoa käytetään. Kehityksen kannalta on myös tärkeää ymmärtää, miksi tietoa käytetään. (From Data to Wisdom, 1999)

Tämän työn tarkoituksena oli helpottaa yrityksen sisäisen tiedon jakamista.

4.1 Hiljainen ja eksplisiittinen tietämys

Hiljainen tietämys perustuu kokemukseen, intuitioon ja vaistoon. Sitä on vaikea ilmaista ja selittää, eli haastavaa jakaa eteenpäin. Tieto saattaa olla myös niin erikoistunutta, että sitä on liian hankala muuttaa eksplisiittiseksi ja täten on parempi jättää tieto asiantuntijoille. (Fernandez;ym., 2010 s. 26)

Eksplisiittinen tietämys taas päinvastoin tarkoittaa helposti selitettävää tietämystä. Tällaista tietämystä löytyy esimerkiksi ohjekirjoista. (Fernandez;ym., 2010 s. 25)

Työn tavoitteena oli auttaa hiljaisen tiedon muuttamista eksplisiittiseksi tietämykseksi.

4.2 Uuden tietämyksen synty

Uuden tietämyksen syntyä voidaan kuvata SECI-mallilla (Socialization, Externalization, Combination, Internalization). Sen kehittäjien, Ikujiro Nonakan ja Hirotaka Takeuchin, mukaan tietämyksen synty voidaan kuvata neljällä eri tyypillä.

Sosialisaatio (Socialization) liittyy hiljaisen tiedon keskenäiseen siirtämiseen. Hiljainen tieto vaihtuu ihmiskohtaamisilla, keskustelemalla, havainnoimalla, väittelemällä ja työskentelemällä. Yritysmaailmassa tällaisia kohtaamisia tapahtuu myös yrityksen ulkopuolella, esimerkiksi asiakkaiden kanssa.

Yrityksen kannalta on tärkeää pitää yllä hyvää keskusteluilmapiiriä henkilöstön sisällä. Huono työilmapiiri estää sosiaalisaation tapahtumisen, jolloin hiljainen tieto ei siirry aiheuttaen ongelmia tietämyksen leviämisen kanssa.

Ulkoistaminen (Externalization) pyrkii muuttamaan hiljaista tietoa eksplisiittiseksi tiedoksi, jotta tieto olisi helppoa jakaa isolle ryhmälle. Tätä varten yritys voisi käyttää esimerkiksi sisäistä tietosanakirjaa tai tässä työssä luodun sovelluksen kaltaista tietämyksenhallintasovellusta.

Yhdistäminen (Combination) muuttaa ekplisiittiä tietoa toisenlaiseksi. Eli siinä yhdistetään valmista tietoa laajemmaksi kokonaisuudeksi, tai korjataan vanhaa tietoa. Jotta yhdistämistä voi tapahtua, on vanhaa tietoa pystyttävä muuttamaan yrityksen sisällä.

Sisäistäminen (Internalization) on tiedon ymmärtämistä yrityksen sisällä. Vaikka tieto on saatavilla, sitä ei välttämättä ymmärretä, joten se täytyy sisäistää. Tätä varten voidaan pitää sisäisiä koulutuksia henkilökunnalle ja jakaa ohjeita. (Design as Learning--or "Knowledge Creation"--the SECI Model, 2011)

Sisäistämisen aikana on tärkeä analysoida sitä, miksi tieto on jäänyt ymmärtämättä, jotta tieto saadaan tehokkaasti jaettua.

4.3 Aineeton pääoma

Aineetonta pääomaa ei voi arvottaa kuten esimerkiksi yrityksen osakkeita, joilla on selvä aikaan sidottu hinta pörsseissä. Aineettoman pääoman johtamisella tarkoitetaan yrityksen ei-taloudellisten resurssien ja toimintatapojen johtamista. Aineettoman pääoman johtamisella tavoitellaan kokonaisvaltaista näkemystä yrityksen kriittisistä ei-taloudellisista resursseista (IC Partners, 2004 s. 4).

Gio Wiederhold (Wiederhold, 2014) kuvaa johdannossa aineetonta pääoma seuraavanlaisesti. Aineeton pääoma sisältää modernien ja innovatiivisten yritysten kyvykkäiden ja lahjakkaiden työntekijöiden henkisen pääoman ja tekijänoikeudet. Yritykset tarvitsevat toimiakseen ja kasvaakseen tätä henkistä pääomaa. Nykypäivänä henkisen pääoman merkitys talouteen ei ole hyvin tunnistettua, eikä ymmärrettyä.

Aineeton pääoma kattaa työvoiman aineettoman voimavaran sekä sen tuotokset. Immateriaalioikeudet on se osa aineettomasta pääomasta, jonka voi omistaa ja sen voi suojata. Aineetonta pääomaa voidaan suojata patenteilla ja tekijänoikeuksilla. (Wiederhold, 2014 s. 36)

Yritys luo aineettomalla pääomallaan tulevaisuuden toimintaedellytyksiä. Aineettoman pääoman strategisen johtamisen lähtökohtana on varmistaa yrityksen käytettävissä olevien resurssien soveltuvuus ja riittävyys tavoitteisiin nähden (IC Partners, 2004 s. 5).

Osaamisen ja henkilöstön kehittäminen on yksi yleisimmistä syistä yrityksille lähteä kartoittamaan aineetonta pääomaa. Yleensä motivaationa on ymmärtää mitkä ovat yrityksen kriittisiä ja kilpailukykyä luovia osaamisia ja miten osaamisten kehittäminen saataisiin kytkettyä paremmin strategiaprosessiin. Aineettoman pääoman tarkastelu nähdään siis työkaluna kytkeä henkilöstön kehittäminen tiukemmin yrityksen operatiiviseen ja strategiseen toimintaan (IC Partners, 2004 s. 7).

Yrityksen viestinnän tärkein tehtävä on välittää relevanttia ja ajantasaista tietoa yrityksestä eri sidosryhmille. Yrityksen kartoittaessa aineetonta pääomaansa syntyy tästä työstä runsaasti tietoa, jota voidaan hyödyntää sidosryhmäviestinnässä sekä myös yrityksen markkinoinnissa ja markkinointiviestinnässä. Argumentoitaessa toisaalta yrityksen tuotteen tai palvelun ainutlaatuisuutta ja asiakashyötyjä, on tärkeää korostaa esimerkiksi yrityksen erikoisosaamista ja kokemusta alalta, yhteistyösuhteita ja toimintatapoja yrityksen sisällä, jotka mahdollistavat asiakaslisäarvon luomisen. Näin pystytään luomaan uskottavuutta konkreettisesti ja faktoihin perustuen (IC Partners, 2004 s. 8).

Aineettoman pääoman selvitys voi palvella myös sellaisissa muutostilanteissa, jotka vaativat uudenlaista resursointia ja pohdintaa siitä, miten organisaation tulee uudessa tilanteessa toimina. Tämän kaltaisia tilanteita voivat olla esimerkiksi: yritys lähtee uudelle markkina-alueelle, lanseeraa markkinoille uuden tuotteen, laajentaa toimintaansa palveluihin ja organisaation rakenteissa tehtävät muutokset esim. siirryttäessä matriisiorganisaatioon (IC Partners, 2004 s. 10).

Tietämyksenhallinta on yksi aineettoman pääoman hallintakeino, johon keskityttiin tässä työssä.

5 KÄYTETYT TEKNIIKAT

Sovelluksesta tehtiin selaimella toimiva websovellus. Se tuli kehittää samoilla tekniikoilla, kuin yrityksen omat tuotteet. Sen lisäksi sovelluksen tuli käyttää valmiina olevia inWorks-luokkia työmäärän vähentämiseksi ja ylläpidettävyyden takaamiseksi. Sovellus pohjautuu ASP.NET-ojelmistokehykseen ja Model-View-Controller malliin.

Malli (model) on luokka, joka vastaa tiedonhallinnasta. Luokka hoitaa tiedon keruun tietokannasta, sekä muokkaa ja päivittää tietoa tarvittaessa. Tiedonhaun jälkeen malli lähettää käsitellyn tiedon käsittelijälle, joka vastaa tiedonvälityksestä eteenpäin.

```
    public class File
    {
        public int Id { get; set; }
       public int GroupId { get; set; }
        [ForeignKey("GroupId")]
        public virtual Group Group { get; set; }
        public int VersionNumber { get; set; }
        public string VersionComment { get; set; }
        public string Path { get; set; }
        public string Name { get; set; }
        public string Extension { get; set; }
        public string AzureGuid { get; set; }


    }
```

Kuva 2 File-luokan malli Entity Frameworkissä

```
export class File
    {
        constructor() { }
        Id: number;
        GroupId: number;
        Group: Models.Group;
        VersionNumber: number;
        VersionComment: string;
        Path: string;
        Name: string;
        Extension: string;
        AzureGuid: string;


    }
```

Kuva 3 File-luokan malli TypeScriptissä

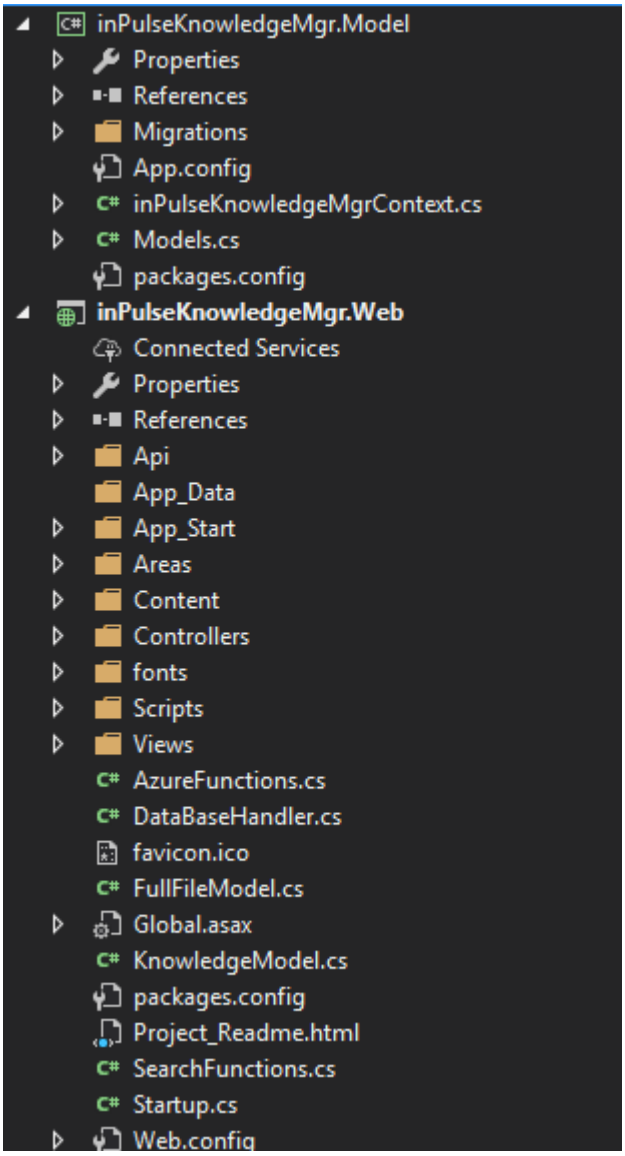
Näkymä (view) on käyttäjälle näkyvä komponentti. Tämän työn näkymät koostuvat cshtml-tiedostoista, joissa on mukana Java- ja TypeScriptiä. Näkymät saavat datan käsittelijältä, ja JavaScript hoitaa tiedon asettamisen käyttöliittymälle.

Käsittelijä (controller) vastaa tiedonvälityksestä tietokannan ja käyttöliittymän välillä. Asiakkaan lähettämät pyynnöt palvelimelle vastaanotetaan käsittelijäluokassa.

5.1 ASP.NET

ASP.NET on Microsoftin tuottama, avoimen lähdekoodin palvelinpuolen websovelluskehys. Sillä voidaan kehittää websovelluksia ja -palveluita, sekä dynaamisia verkkosivuja. ASP.NET-sovellus käsittelee asiakkaan pyytämän prosessin palvelimella ja palauttaa valmiin HTML-koodin asiakkaalle. JavaScript-tuki mahdollistaa epäsynkronoidun tiedonvälityksen asiakkaan kanssa. Sen suurimpina etuina on nopeus, olio-ohjelmointi, sekä .Net-luokkakirjasto. Sovelluksen koodi käännetään, eikä tulkita ajon aikana, kuten monilla vastaavilla websovelluskehyksillä. Tämä tekee sovelluksista erityisen nopeita. (Guru99)

.NET-kirjasto sisältää suuren määrän valmiita metodeja ja luokkia, joita ohjelmoijan ei tarvitse itse uudestaan ohjelmoida. Kirjaston metodit ovat lähinnä usein toistuvia tapahtamia, kuten matemaattiset laskut, tiedostoon kirjoittamiset ja muuttujien parsimiset. (Guru99)



Kuva 4 Projektin solution

Kuvassa 4 nähdään sovelluksen rakenne. Näkymät ja käsittelijät ovat omassa projektissaan erillään malleista. Web-projektilla ei ole yhteyttä tietokantaan, vaan tietokantayhteys hoidetaan mallin projektin kautta.

5.2 TypeScript

TypeScript on JavaScript-komentosarjakielen tyypityskirjasto. Se on luotu isoja sovelluksia varten paikkaamaan JavaScriptin ongelmia. Kun TypeScript-koodi käännetään, siitä syntyy JavaScript-tiedosto. JavaScript-koodi toimii suoraan TypeScript-tiedostossa. JavaScript on ohjelmointikieli, jonka avulla websivuilla voidaan tehdä pelkkää html-sivua monipuolisempia asioita, sekä sivut voidaan luoda dynaamisesti metodien sisällä.

TypeScriptin höyty on sen ohjelmoitavuus: sitä ohjelmoidessa voidaan käyttää esimerkiksi rajapintoja, enumeja, luokkia ja staattisia tyyppejä. Kääntäjä havaitsee virheet jo ennekuin itse ohjelma on käynnissä, tehden ongelmienratkaisusta paljon helpompaa. (Lauer, 2017)

5.3 jQuery

jQuery on JavaScript-kirjasto, jonka tarkoituksena on yksinkertaistaa JavaScript-koodia. Kirjaston sisällä on valmista koodia monimutkaisille asioille, kuten AJAX-kutsuille. Lisäksi usein toistuvat asiat, kuten tapahtumankäsittelijät, on yksinkertaistettu kirjastolla. Kirjastoa käyttämällä säästetään sekä ohjelmointiaikaa, että koodirivejä. Myös sovelluksen luotettavuus paranee, kun käytetään laajassa käytössä olevia valmiita ratkaisua oman ohjelman pohjana. (TutorialsTeacher.com)

```
$.ajax({
    method: "GET",
    url: _finalUrl,
    headers: {
        "Accept": "application/json",
        "api-key": queryKey,
        "Access-Control-Allow-Origin": "*"
    },
    contentType: "application/json",
    success: function (data) {
        console.log(data);
        var _tempTable: Models.File[] = [];

        $.each(data, function (index, value) {
```

Kuva 5 jQueryn AJAX-kutsu

AJAX-kutsut mahdollistavat epäsynkroonisen kommunikoinnin asiakkaan ja palvelimen välillä. Asiakas voi esimerkiksi lähettää hakupyynnön palvelimelle ja palvelimen vastausta odottaesssa sovellusta voi vielä käyttää, ja tietoa ladataan näytölle reaaliajassa. Lisäksi jQuery mahdollistaa myös animaatiot, jota hyödynnetään työssä latausikkunassa. (Refsnes Data)

5.4 LINQ

LINQ, eli Language Integrated Query, .Net-ympäristön syntaksi, jonka avulla tietokantakyselyitä voidaan kirjoittaa C#-kielellä. Työssä suoritetaan useita kyselyitä tietokantaan, joten on järkevää käyttää LINQ-syntaksia perinteisten SQL-kyselyiden sijaan. (TutorialsTeacher.com)

```
return _ctx.Tags.OrderBy(x => x.Name).Where(x => x.TagGroupId == tagGroupId).ToList();
```

Kuva 6 LINQ kysely

Kuvassa esimerkki yhdestä LINQ-kyselystä. Kyselyssä haetaan _ctx-kontekstista Tags-nimisiä olioita. Tags-olion avain TagGroupId pitää täsmätä paikalliseen muuttujaan tagGroupid. Lopuksi olioista muodostetaan lista joka järjestellään aakkosten mukaan.

5.5 Entity Framework

Entity Framework (EF) on .Net-ympäristön olioiden kartoitussyntaksi. Sen tarkoituksena on ylläpitää tietokannan mallia niin, että tietokanta ja sovelluksen käsitys tietokannasta ovat yhdenmukaiset. Tämä voidaan toteutta kahdella tavalla: *CodeFirst* mahdollistaa tietokannan muokkaamisen sovelluksen puolella. Sovellukseen luodaan mallit (model) ja EF:n työnä on varmistaa, että tietokanta vastaa malleja. Migraatioiden avulla tietokantaan voidaan lisätä esimerkiksi tauluja, sarakkeita ja relaatioavaimia, tai vaihtoehtoisesti poistaa niitä. Migraatioihin voidaan lisätä myös datankäsittelytoimia, jos niitä tarvitaan.

DatabaseFirst toimii toisinpäin, eli se varmistaa, että koodi vastaa tietokantaa. Tällöin halutut muutokset tehdään tietokantaan, ja EF varmistaa, että sovelluksen mallit vastaavat muutoksia. Työssä tullaan käyttämään *CodeFirst*-tapaa. (EntityFrameworkTutorial.net)

```
public DbSet<File> Files { get; set; }
public DbSet<Group> Groups { get; set; }
public DbSet<Tag> Tags { get; set; }
public DbSet<TagGroup> TagGroups { get; set; }
public DbSet<GroupTag> GroupTags { get; set; }
public DbSet<Error> Errors { get; set; }
```

Kuva 7 Työn EF mallit, CodeFirst

5.6 Azure Blob Storage

Azure on Microsoftin pilvipalveluhin keskittyvä tuoteperhe, jotka toimivat Microsoftin datakeskuksissä ympäri maailmaa.

Blob Storage on datan varastointipalvelu. Sinne pystytään tallentamaan suuria määriä tiedostoja, kuten kuvia, dokumentteja ja videoita. Pilveen ladattu tiedosto muutetaan binääridataksi ja se saa oman tunnisteen jolla se löydetään palvelimelta. (Microsoft)

5.7 Azure Search

Azure Search on hakukonepalvelu, jota voidaan käyttää pilvipalvelun sisällä. Työssä tätä käytettiin tekstitiedostojen sisältä hakemiseen hakusanoin.
Palvelua kutsutaan REST-Apin (Representational state transfer) avulla. Palveluun lähetetään hakusana HTML/Get-metodilla ja palvelu vastaa pisteyttämällä osumatarkkuuden ne tiedostot, jotka ovat lähellä hakusanoja. (Microsoft)

5.8 Kehityssovellukset

Sovellus kehitettiin Visual Studio 2015 -kehitysympäristössä. Ympäristö tukee monia eri ohjelmointikieliä, kuten työssä käytettäviä C#:ia, JavaScriptiä, TypeScriptiä ja LINQ:a. Sovelluksen NuGet Package Manager -ominaisuus on tehokas apu tarvittavien laajennusten etsimiseen ja hallinnointiin. Esimerkiksi Azure-pilvipalveluiden tuki on ladattu laajennuspaketteina.

Tietokantaa hallinnoidaan SQL Management Studio -sovelluksella. Sovelluksella pystytään ottamaan yhteys SQL Server -tietokantaan. Vaikka tietokanta luodaan käyttämällä Entity Frameworkia, käytetään Management Studiota kehityksessä tiedon tarkistamiseen ja korjaamiseen.

Azure Searchin REST Apia testattiin käyttämällä Postman-sovellusta. Postman pystyy lähettämään haluttuja REST-kutsuja ja vastaanottamaan vastauksia luettavassa muodossa. Kehitystyö nopeutuu huomattavasti, kun kutsuja pystyy nopeasti muokkaamaan ja testaamaan, ennen kuin niitä luodaan kehitettävässä sovelluksessa.

Azure Blob Storage:n sisältö hallinnoidaan Azure Storage Explorer -ohjelmalla. Vaikka tiedostoja pystyttäisiin selamaan myös REST-kutsuilla, on se helpompaa erillistä käyttöliittymää käyttämällä.

6 TYÖN TOTEUTUS

Työ aloitettiin keväällä 2017. Ensimmäisenä käytiin yhden päivän perehdytyskoulutus, jotta opittiin, minkälaisia tekniikoita yrityksessä käytetään ja minkälaista työn jälkeä odotetaan. Perehdytyksen jälkeen työtä tehtiin kevään aikana joko kotona, tai yrityksen toimistotiloissa.

Sovellusta aloitettiin kehittämään alkuperäisen suunnitelman mukaisesti. Työn edetessä sovellusta testattiin opinnäytetyön ohjaajan kanssa useaan kertaan. Testausten aikana nousi esille uusia tarpeita ja kehitysideoita, joiden mukaan sovellusta kehitettiin eteenpäin.

Sovellusta lähdettiin kehittämään keskittyen aluksi tiedostojen vientiin pilveen. Kun tiedostot saatiin luotettavasti siirrettyä, pystyttiin keskittymään hakutoimintoihin.

Ensimmäisissä versioissa tiedostot pystyttiin viemään pilveen ja tuomaan pilvestä. Niille voitiin antaa tageja ja vapaaehtoisesti myös tiedostoon liittyvän virheilmoituksen. Haku etsi tuloksia tageista ja/tai virheilmoituksesta. Testausten aikana havaittiin, että tiedostojen poistaminen tuotti ongelmia niihin liitettyjen tagien kanssa, sillä saman nimisiä tageja saattoi olla usealla tiedostolla. Ongelman korjaamiseksi kantarakennetta täytyi muuttaa. Lisäksi haluttiin, että tiedostoja on pystyttävä päivittämään uuteen versioon. Päivitystä varten tarvittiin versiohistoriointi ja -hallinta.

Seuraavien versioiden testausten aikana keksittiin, että tagit pitää pystyä ryhmittämään. Jotkut sanat ovat synonyymejä keskenään, tai niiden käyttötarkoitus on sama. Ryhmitystä varten haluttiin oma sivu, jolla voidaan hallita tagien linkittämistä toisiinsa. Linkityksen avulla haku löytää tiedostot helpommin.

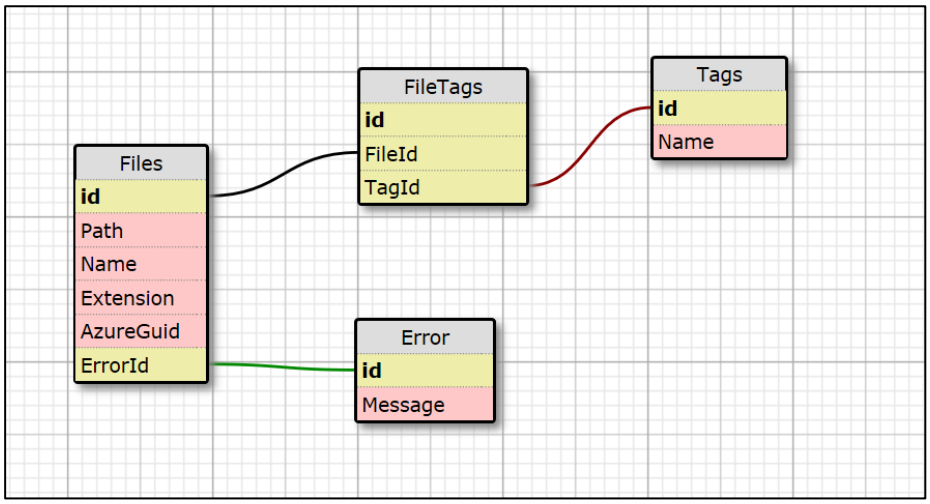
Työn loppuvaiheilla havaittiin, että pelkästään tageilla ja virheilmoituksella haku ei riitä. Oli etsittävä myös pilveen siirrettyjen tekstitiedostojen ja dokumenttien sisältä halutuin hakusanoin ja -lausein. Ominaisuus saatiin toteutettua.

Työ oli valmis kesällä 2017 ja kirjallinen osuus tehtiin keväällä 2018.

6.1 Tietokantarakenne

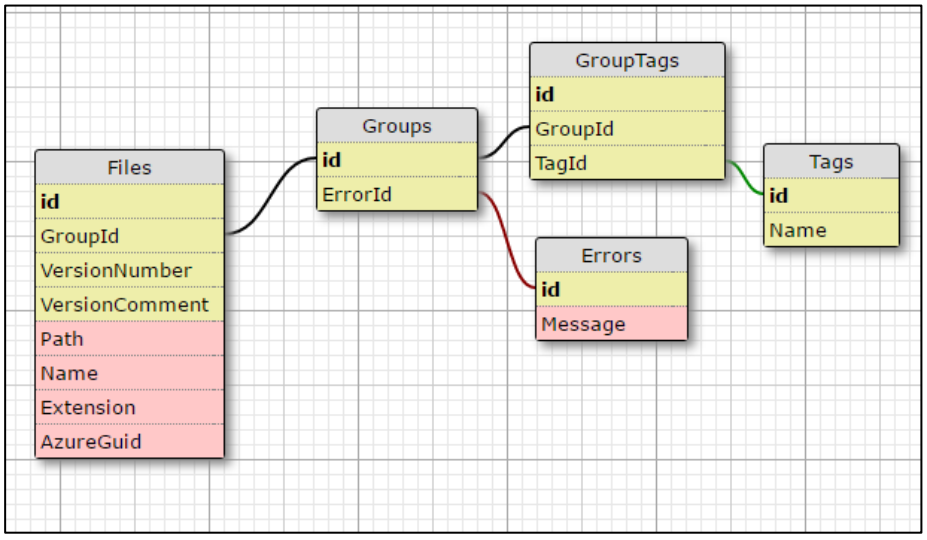
Työssä käytetään yrityksen linjan mukaisesti SQL Serveriä. Työtä varten luodaan uusi tietokanta, ilman valmista pohjaa. Tietokantarakenne muuttuu kehityksen tarpeiden mukaan, mutta kuitenkin relaatiotietokannan periaatteita noudattaen. Tieto jaetaan useaan tauluun, jotka viittaavat toisiinsa vierasavaimilla.

Tietokanta rakennetaan Entity Frameworkin avulla, CodeFirst-mallilla. Halutut taulurakenteet kirjoitetaan omaan malliluokkaan (Models), jonka EF tulkitsee ja suorittaa tarvittavat käskyt tietokantaan. Tulevat muutokset hallitaan migraatioiden avulla.



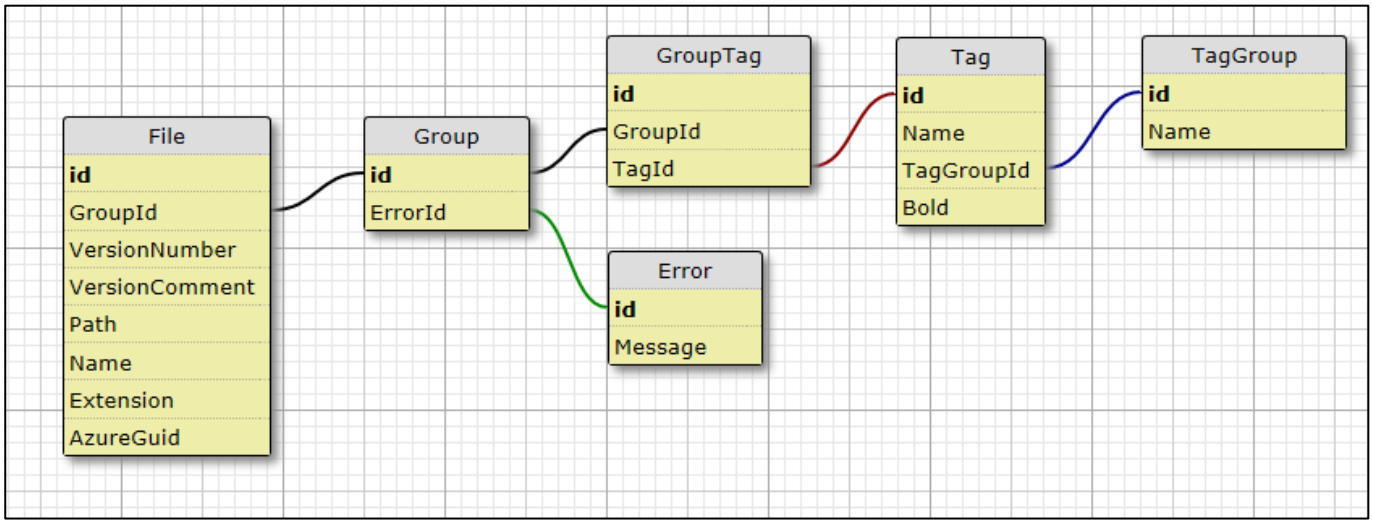
Kuva 8 Kantarakenteen ensimmäinen versio

Työn alussa tietokantarakenne oli yksinkertainen. Tiedostolla on tietokannassa tieto sen fyysisestä sijainnista Azure Blob Storagessa. FileTags-välitaulu linkittää Tag-taulun sisältämät tagit tiedostoon. Error-taulu sisältää virhetekstin. Lopullinen kantarakenne tarkempine selitykseen avataan myöhemmin tässä raportissa.



Kuva 9 Kantarakenteen yksi väliversioista

Ylläolevassa kuvassa (Kuva 9) sovellukseen on lisätty versionhallinta. Tätä varten ”Files”-tauluun on lisätty sarakkeet versionumeroa, sekä päivityksen syyksi sovelluksen kysymää kommenttia varten. Versiot on linkitetty toisiinsa ”Groups”-taulun avulla, ja yhteen tiedostoryhmään liittyy samat tagit ja virhetiedosto.

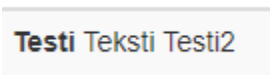


Kuva 10 Kantarakenteen viimeisin versio

Sovelluksen viimeisimmän version kantarakenne on seuraavanlainen:

- Files: Taulu sisältää yksittäisen tiedoston fyysisen paikan pilvessä, sekä versiotiedot
    - Id (int): Pääavain, jonka mukaan rivit indeksoidaan lisäämisjärjestyksessä
    - GroupId (int): Vierasavain, joka linkittää tiedoston yhden tiedostoryhmän versioksi. Käyttöliittymällä tiedostoryhmä näyttäytyy yhtenä tiedostona, jolla on monta eri versiota, vaikka todellisuudessa kyseessä on monta eri tiedostoa, jotka on linkitetty toisiinsa.
    - VersionNumber (int): Versionumero, jonka avulla käyttöliittymä pystyy listaamaan uusimman version hakutuloksiin, sekä järjestelmään vanhojen versioiden listan uudestaan.
    - VersionComment (nvarchar): Kun käyttöliittymällä päivitetään tiedosto uuteen, käyttäjältä pyydetään vapaaehtoista selitettä, miksi ja mitä tiedostosta on päivitetty.

  - Path (nvarchar): Tíedoston fyysinen osoite Azure Blob Storagessa. Tämän avulla tiedostot tallennetaan pilvestä asiakkaalle oikeasta paikasta.
  - Name (nvarchar): Tiedoston nimi, joka näytetään käyttöliittymällä.
  - Extension (nvarchar): Tiedostomuoto, esimerkiksi ”.txt”, jota tarvitaan tiedostojen tallennusvaiheessa.
  - AzureGuid (nvarchar): Azure Blob Storagea varten tiedostolle luodaan uniikki GUID (Globally Unique Identifier), jotta tiedostonimet eivät sekoitu keskenään.
- Group: Välitaulu, joka pitää tiedoston eri versiot koossa. Kun käyttöliittymällä päivitetään tiedosto uuteen, tallentuu tiedosto normaalisti uutena tallennuksena, mutta sen GroupId pysyy samana, kuin edellisen version.
  - Id (int): Pääavain
  - ErrorId (int): Tiedostoryhmään liitetty virheteksti
- Error: Jos tiedosto liittyy jonkun virheen ratkaisemiseen, voidaan sille liittää virhetekstä, millä tiedostoa voidaan hakea.
  - Id (int): Pääavain
  - Message (nvarchar): Teksti, josta tiedostoa voidaan hakea hakusanoin.
- GroupTag: Välitaulu, joka yhdistää tiedostoryhmän (Group) tagiin. Aiemmissa sovellusversioissa tagit liittyivät tiedostoon, mutta kehityksen myötä logiikka muuttui niin, että tagit liittyvät ryhmään, ja hakutoiminto palauttaa palvelimelle tiedon ryhmästä, eikä yksittäisistä tiedostoista.
  - Id (int): Pääavain
  - GroupId (int): Vierasavain, joka kertoo mihin tiedostoryhmään tagi kuuluu.
  - TagId (int): Vierasvain, joka linkittyy tagiin.
- Tag: Avainsanataulu. Hakutoiminto perustuu ensisijaisesti näihin.
  - Id (int): Pääavain
  - Name (nvarchar): Tagin nimi, eli se sana joka tiedostoon liittyy.
  - TagGroupId (int): Vierasavain, joka liittää tagin tagiryhmään.
  - Bold (boolean): Käyttöliittymää varten tietokannalta piiloitettu kenttä. Sarake näkyy oikeasti vain Entity Frameworkin mallissa, mutta ei SQL Server-kannassa. Tietoa käytetään hakutuloslistauksessa tagisanan lihavoimiseen tarvittaessa. Alla olevassa kuvassa (Kuva 11) tiedostoa on haettu sanalla ”Testi”, jolloin palvelin kertoo käyttöliittymälle, että haku on onnistunut tuolla sanalla ja se on lihavoitu. Tagit ”Teksti” ja ”Testi2” ovat tiedoston tageja, mutta niitä ei ole käytetty tässä haussa.



Kuva 11 Tagien lihavointi

- TagGroup: Taulu, joka yhdistää synonyymit, tai toisiinsa muuten liittyvät sanat, toisiinsa.
  - Id (int): Pääavain
  - Name (nvarchar): Ryhmän nimi.

6.2 Käyttöliittymä

Tässä luvussa kerrotaan sovelluksen kolmesta eri osioista ja niiden tarvittaessa avautuvista dialogeista, jotka ovat seuraavat:

- Tiedostojen hakusivu
  - Tiedoston päivitysmodaali
  - Vanhojen versioiden listaus/latausmodaali
  - Tagien muokkausmodaali
- Tiedostojen lisäyssivu
- Tagien ryhmittämissivu

6.3 Tiedostojen varastointi

Kuva 12 Ote käyttöliittymästä

Tiedostot voidaan joko vetää käyttöjärjestelmän kautta harmaaseen laatikkoon, jolloin sovellus saa tiedon tiedoston nimestä ja polusta. Vaihtoehtoisesti käyttäjä voi painaa ”Selaa”-nappia.

Kuva 12 Tagien lisäysvalikko

Tagien syöttö ennustaa jo tietokannasta löytyviä tageja. Jos tagia ei ole jo olemassa, se lisätään tietokantaan. Select2-kirjasto sallii sanojen muodostamisen omiksi objekteikseen, jolloin käyttäjä havaitsee sanan olevan ”hyväksytty” ja sen pystyy poistamaan. Kirjoittamisen aikana sovellus ehdottaa tietokannasta löytyviä tageja, joissa on samat kirjaimet kuin sillä hetkellä on syöttökentässä. Ennustaminen helpottaa tiedonhakua sekä antaa tiedonhakijalle idoita, miten oikean tiedon löytäisi.

Tallennuksen aikana logiikka muuttaa tagit muotoon: Iso alkukirjain, loput pienellä. Jos kaikki tagin kirjaimet ovat isolla, oletetaan että niiden kuuluukin olla ja ne jätetään isoiksi.

6.3.1 Vienti pilveen

Tiedoston vienti pilveen alkaa tiedoston valitsemisella. Kun tiedosto valitaan joko vetämällä se selaimen ikkunaan tai selaa-valikolla, kutsutaan alla näkyvää tapahtumankäsittelijää. (Kuva 13). Kun kentän arvo muuttuu (on('change')), ladataan tiedoston sisältö muuttujaan.

```
$('input[type=file]').on('change', function (ev) {
    files = (<any>ev).target.files;
    $("#fileText").val(files[0].name);
});
```

Kuva 14 Esimerkki tapahtumankäsittelijästä

Kun käyttäjä painaa ”Lisää”-nappia, tiedosto aletaan siirtämään Azure-palvelimelle. Tiedoston tiedot kasataan ”data”-muuttujan sisälle jota käytetään alla olevassa AJAX-kutsussa. (Kuva 15)

```
Utils.blockUI("Tallennetaan tiedostoa " + $("#fileText").val());
var ajaxRequest = $.ajax({
    url: '/Api/File/Upload/',
    type: 'POST',
    data: data,
    cache: false,
    dataType: 'json',
    processData: false,
    contentType: false,
});

ajaxRequest.done(function (xhr, textStatus) {
    Utils.blockUI_success("Tiedosto tallennettu!");

    $('#blockUI_buttonOK').click(function () {
        location.reload();
    });
```

Kuva 15 AJAX-kutsu

Utils.blockUI on blockUI-JavaScript-kirjaston metodi, joka estää käyttäjää tekemästä sivulla mitään. Tällä annetaan palvelimelle rauha siirtää dataa eteenpäin. Ajax-kutsu lähettää tiedon ASP.NET-käsittelijälle. Sivu päivittyy, kun AJAX-kutsu saa tiedon siitä, että tiedoston vienti on onnistunut.

```
public void UploadFile()
        {
            if (HttpContext.Current.Request.Files.AllKeys.Any())
            {
                var HttpPostedFile = HttpContext.Current.Request.Files["UploadedFile"];
                string TagString = HttpContext.Current.Request["tagString"];
                string ErrorText = HttpContext.Current.Request["errorText"];
                string[] Tags = TagString.Split(',');

                KnowledgeModel KmFile = KnowledgeModel.createFile(HttpPostedFile);
                DataBaseHandler Dbh = new DataBaseHandler();

                //Jos tiedot menee tietokantaan, niin vasta sitten tallennetaan Azureen:
                if (Dbh.SaveUploadFileData(KmFile, Tags, ErrorText))
                {
                    AttachmentsAzureStorageController Azure = new AttachmentsAzureStor-
ageController();
                    Azure.AddFile(KmFile);
                    AzureFunctions.RunAzureIndexer();
                }
            }
        }
```

Kuva 16 Controller

Käsittelijä ottaa tiedoston vastaan ja muuttaa sen Azurelle sopivaksi malliksi (Kuva 16). Kun DataBaseHandler-luokka sallii tallennuksen, tiedosto viedään palvelimelle. Näin ollen tiedosto käy ensin yrityksen palvelimella, ennen kuin se siirretään Azureen. Azuren kanssa kommunikointiin käytetään yrityksen valmista, AttachmentsAzureStorageController -luokkaa.

```
            byte[] fileContent = null;
            using (var binaryReader = new BinaryReader(postedFile.InputStream))
            {
                fileContent = binaryReader.ReadBytes(postedFile.ContentLength);
            }
```

Kuva 17 Datan konvertointi binääriksi

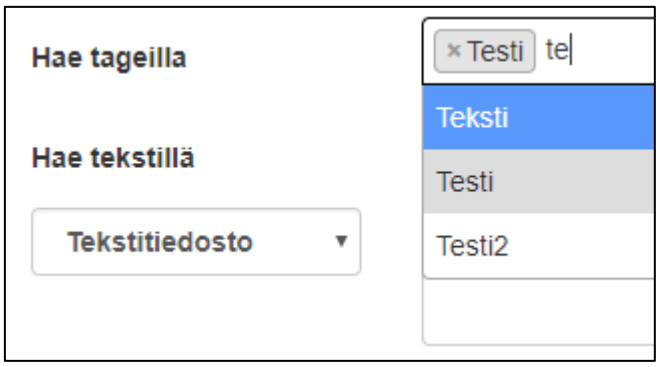
Tiedostojen data tallennetaan Azureen binäärimuodossa, joten kun KnowledgeModel-tyyppinen olio luodaan, data muutetaan binääriksi (Kuva 17).

6.3.2 Tuonti pilvestä

Tietoturvasyistä tiedoston dataa ei voida tuoda jQueryn käsiteltäväksi, niin kuin tiedostoa viedessä. Tiedostoa viedessä asiakaspään koodi käsittelee tiedoston, joka lähetetään palvelimelle. Tuontivaiheessa selainohjelma lähettää tiedon palvelimelle mikä tiedosto ladataan ja palvelin lähettää tiedoston suoraan asiakkaalle ohittamalla käyttöliittymän.

Kun tiedostoa tuodaan Azuresta, binääridata muutetaan takaisin oikeaan muotoon yrityksen palvelimella. Sen jälkeen se lähetetään käyttäjälle tallennettavaksi.

6.4 Tiedostojen haku



Kuva 18 Tagien ennustus tekstinsyöttörivillä

Kuten tiedoston viennissä, myös haussa käytetään tagien ennustustamista. Kun käyttäjä on syöttänyt kirjaimet ”te”, haetaan kannasta sanat, joissa esiintyy nuo kirjaimet ja ehdotetaan lähimpiä.

Kuva 19 Hakutoimintavalikko

”Hae tekstillä”-kenttää voidaan käyttää joko Azure Searchilla tekstitiedoston sisäiseen hakuun, tai tiedokannasta löytyvän virhetekstin avulla.

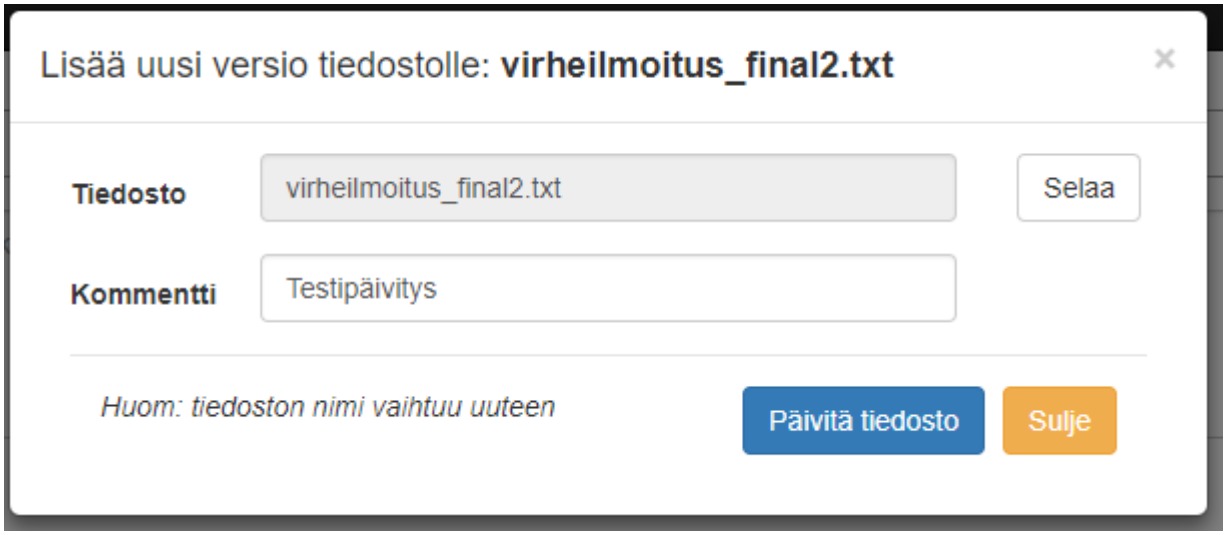
Kuva 20 Haun tulokset

Sovellus etsii tiedostot niin, että niillä täytyy olla kaikki ne tagit, jotka hakuehtoihin annettiin. Nämä tagit lihavoidaan hakutuloksissa ja muut tagit jätetään tavallisiksi. Tiedostot järjestetään niin, että ylimmäksi tulee

tiedostot joilla tagit täsmäävät suoraan hakuehtoihin, ja sen jälkeen ne, jotka täsmäävät tagiryhmien kautta. Hakutulosriville on liitetty myös tiedostoon liittyviä toiminnallisuuksia:

Kuva 21 Tiedoston tallennus, päivitys, sekä vanhojen versioiden listaus

Tallennus-nappi aloittaa tiedoston lataamisen Azuresta yrityksen palvelimelle, ja sieltä käyttäjälle. Päivitys-nappi avaa uuden modaalin.



Kuva 22 Päivitysmodaali

Vanhat versiot -modaali on valittavana, jos tiedostolla on aiempia versioita (Kuva 22).

Kuva 23 Versionhallintamodaali

Modaali näyttää vanhat versiot ja antaa myös tallentaa niitä (Kuva 23).

Kuva 23 Virheilmoitus sekä tagien muokkauslomake

Virheilmoitus-nappi on näkyvillä, jos tiedostolle on liitetty virheilmoitus. Koko virheilmoitus näytetään, kun hiiri viedään napin päälle. Painaminen ei tee mitään.

Tagien muokkausmodaali antaa lisätä ja poistaa tiedostoryhmän tageja (Kuva 24).

Kuva 24 Tagien muokkausmodaali

6.4.1 Haku tietokannasta

Kun hakukenttään kirjoitetaan, sovellus ennustaa mitä tagia käyttäjä aikoo kirjoittaa. Tällä myös estetään haut, jossa tageja ei löytyisi tietokannasta, koska haku suoritetaan vain, jos syötekentässä on olemassa olevia tageja.

Tagihaku etsii ensin tagiryhmät johon annetut tagit kuuluvat. Sen jälkeen etsitään tiedostoryhmät, joihin kuuluu kaikki löydetyt tagiryhmät. Samalla annetaan tagille tieto lihavoinnista, jos se on täysin sama, kuin hakuehdossa.

Virheilmoitushaku etsii suoraan tietokannasta käyttämällä SQL-kielen Difference-funktiota. Funktio vertailee hakuehtoa ja tietokannan tekstikenttiä ja pyrkii löytämään lähelle osuvat. Testien aikana kuitenkin havaittiin, että tämä toimii luotettavasti lähinnä pitkien virheviestien kanssa.

6.4.2 Haku tiedostojen sisältä

Azure Search mahdollistaa pilveen varastoitujen tekstitiedostojen ja dokumenttien sisältä etsimiseen. Search vertailee hakuehtoja tiedostojen sisältöön ja palauttaa palvelimelle pisteytyksen, kuinka lähelle hakuehdot osuvat.

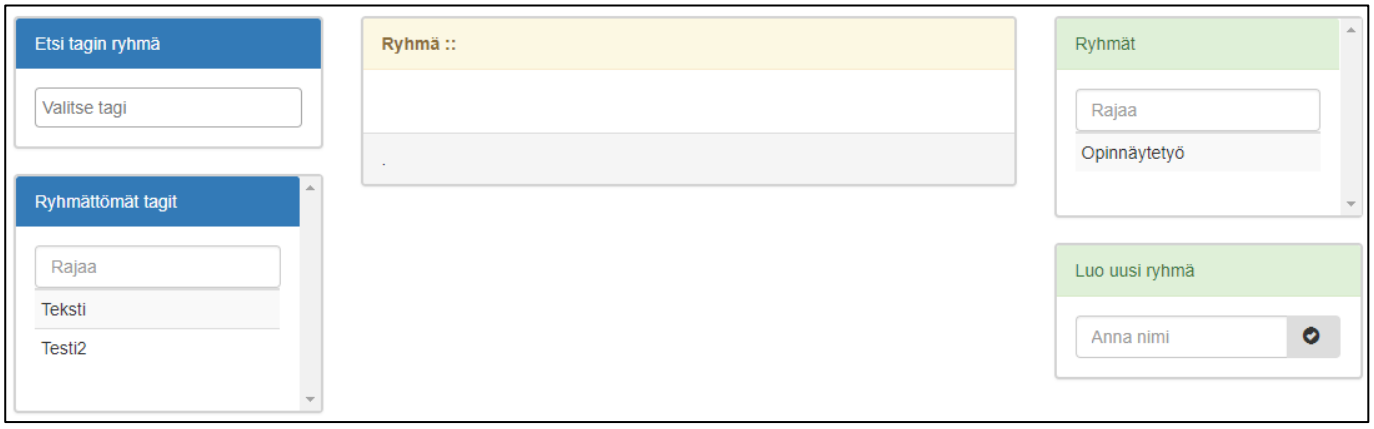
```
HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(uri);

            request.Method = "POST";
            request.ContentType = "application/json";
            request.Headers = new WebHeaderCollection();
            request.Headers.Add("api-key: X");
```

Kuva 26 Hakupyyntö

Palvelua käskytetään REST-protokollalla (Kuva 26). Hakuehto on liitetty uri-muuttujaan.

6.5 Tagien ryhmitys

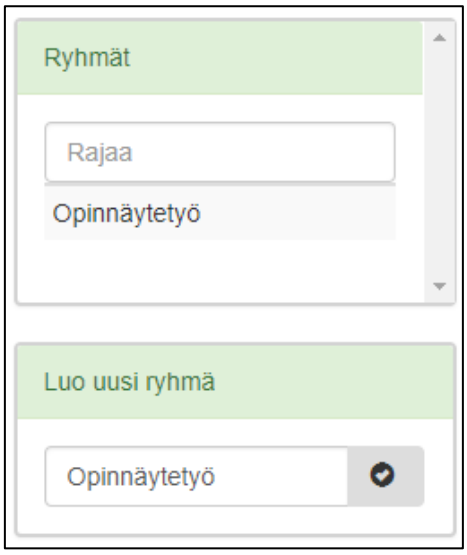
Synonyymien vuoksi ohjelmaan luotiin toiminto, jolla tagit pystytään ryhmittämään.



Kuva 27 Tagien ryhmityssivu

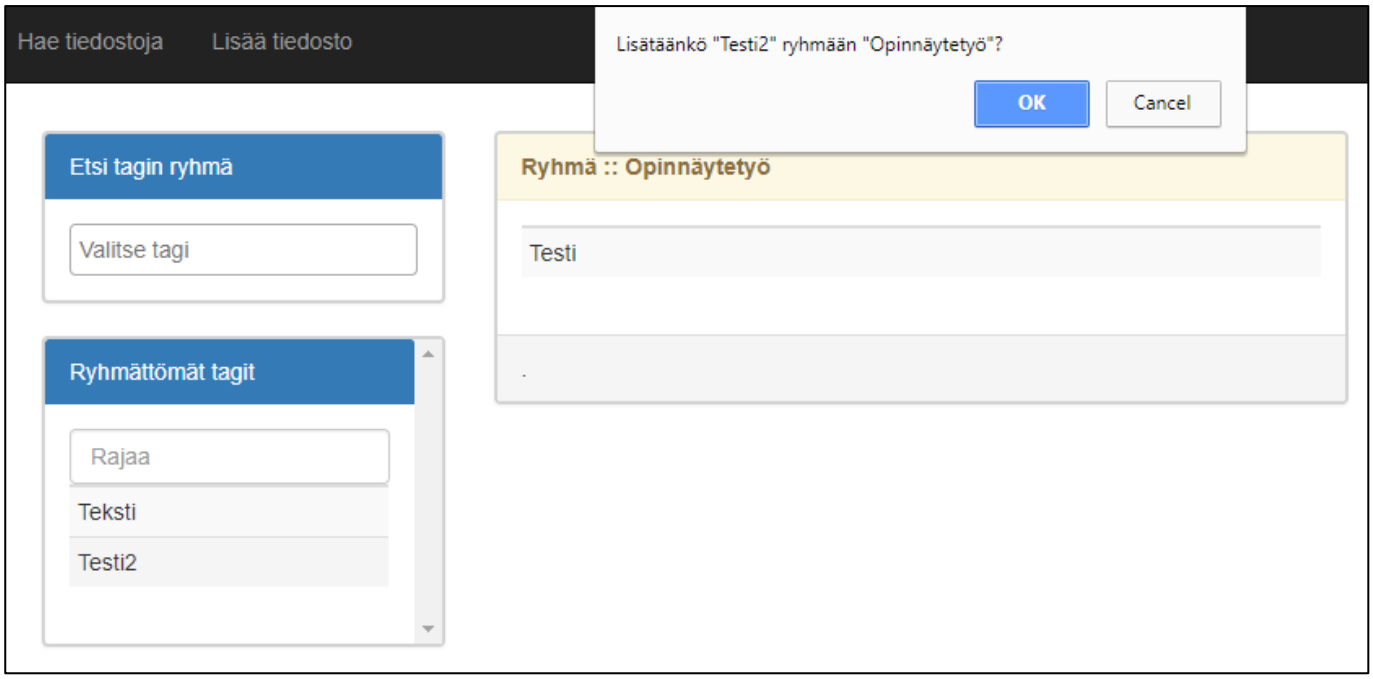
Tagien linkitystä varten luotiin oma sivu. Sivulla pystyy:

- Etsimään tiettyyn tagiin liittyvä ryhmä
- Etsimään tagit, joilla ei ole ryhmää
- Luomaan uusia ryhmiä
- Etsimään ryhmiä
- Liittämään tageja ryhmiin
- Poistamaan tageja ryhmistä



Kuva 28 Ryhmien luonti-ikkunat

Käyttäjä voi vapaasti luoda ja poistaa ryhmiä. Luomisvaiheessa käyttäjän täytyy nimetä ryhmä. Ryhmiä pystytään nimien perusteella etsimään rajaustoiminnolla.



Kuva 29 Tagin liittäminen ryhmään

Kun ryhmä on luotu, siihen voidaan lisätä tageja, tai poistamaan liitoksia. Yksittäinen tagi voi kuulua useaan ryhmään.

7 YHTEENVETO

Työn tavoitteena oli luoda alusta tietämyksen jakamiselle yrityksen sisällä. Erityistä huomiota kiinnitettiin alustan helppoon ja mahdollisimman vapaaseen käyttöön.

Tuloksena syntyi sovellus, jonka avulla pystytään siirtämään tiedostoja Azure Blob Storage -pilvipalveluun. Sen lisäksi sovelluksella pystytään merkkaamaan tiedostot tagein. Sovelluksella pystytään hakemaan tiedostoja tagien avulla, tiedostoihin liitettyjen virheilmoitusten avulla, tai tekstitiedostojen sisältä etsimällä.

Sovellus listaa tiedostot hakutulosten perusteella, jonka jälkeen käyttäjä pystyy lataamaan tiedoston, muokkaamaan tiedoston tageja, poistamaan tiedoston, tai päivittämään tiedoston uuteen versioon. Myös tiedostojen vanhat versiot jäävät talteen ja vanhoja tiedostoversioita pystyy selaamaan.

8 JATKOKEHITYSIDEOITA

Sovellus toimii halutun mukaisesti. Jäljelle jäi kuitenkin asioita, joita voisi sovelluksessa kehittää.

Sovelluksessa olisi hyvä käyttää laajemmin yhteysmäärittelytiedostoja (ConnectionStrings tai AppSettings). Tällä hetkellä osa verkko-osoitteista on asetettu suoraan koodiin. Mahdollisten muutosten tekeminen on siis huomattavasti hankalampaa, kuin jos yhteystiedot olisivat erillisessä tiedostossa. Tähän ei kiinnitetty työn aikana suurempaa huomiota, koska sovellusta ei ole tarkoitettu asennettavan kuin yhteen paikkaan, yhdelle yritykselle.

Sovelluksen virheidenkäsittely on jäänyt hieman vajaaksi. Kaikista mahdollisista virheistä, joita käytön aikana voi tapahtua, ei välttämättä informoida käyttäjälle. Olisi siis hyvä lisätä jonkinlainen virheilmoituslaajennus sovellukseen, jotta käyttäjät tietävät virheistä. Esimerkiksi yhteysongelmat Azureen voivat tallennusvaiheessa aiheuttaa ongelmia, eikä niistä ilmoiteta käyttäjälle.

Käyttäjä pystyy sovelluksella muokkaamaan tiedoston nimeä päivittämällä tiedoston uuteen versioon, sekä lisäämään ja poistamaan tageja. Tiedostoon liitettyä virhetekstiä ei kuitenkaan käyttöliittymällä pysty muokkaamaan, vaan muutos täytyy tehdä suoraan tietokantaa muuttamalla. Virhetekstin muokkaamiselle voisi tehdä tuen, mutta on pohdittava, onko muutos kannattava, ettei tiedoston alkuperäinen tarkoitus katoa.

Koodia olisi hyvä siistiä. Metodeja voisi muuttaa rajapintojen taakse, jotta jatkokehitys olisi helpompaa, sekä koodi luettavampaa. Lisäksi sovellus pohjautuu suureksi osaksi Azuren pilvipalveluun. Jos sovellusta haluattaisiin levittää laajempaan käyttöön, pitäisi yhteydetkin rajapinnoittaa niin, että pilvipalvelun voisi vaihtaa joksikin muuksi. Dokumenttien sisällöstä hakeminen pitäisi tällöin muuttaa pois asetettavaksi asetukseksi, koska muissa pilvipalveluissa ei tällaista ominaisuutta välttämättä tueta.

9 LÄHDELUETTELO

Bradley, Keith. 1997. *Intellectual Capital and the New Wealth of Nations.* 1997.

Design as Learning--or "Knowledge Creation"--the SECI Model. **Shelley, Evenson ja Dubberly, Hugh. 2011.** 2011.

EntityFrameworkTutorial.net. http://www.entityframeworktutorial.net/what-is-entityframework.aspx. *http://www.entityframeworktutorial.net/.* [Online]

Fernandez, Irma Becerra ja Sabherwal, Rajiv. 2010. *Knowledge Management: Systems and Processes.* 2010.

From Data to Wisdom. **Ackoff, Russell. 1999.** 1999.

Guru99. https://www.guru99.com/what-is-asp-dot-net.html. *https://www.guru99.com.* [Online]

IC Partners. 2004. *Aineettoman pääoman johtaminen TYÖKIRJA.* 2004.

Lauer, Rob. 2017. https://developer.telerik.com/topics/web-development/what-is-typescript/. *https://developer.telerik.com.* [Online] 22. February 2017.

Microsoft. https://docs.microsoft.com/en-us/azure/search/search-what-is-azure-search. *https://docs.microsoft.com/.* [Online]

—. https://docs.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction. *https://docs.microsoft.com/.* [Online]

Refsnes Data. https://www.w3schools.com/xml/ajax_intro.asp. *https://www.w3schools.com.* [Online]

TutorialsTeacher.com. http://www.tutorialsteacher.com/jquery/what-is-jquery. *http://www.tutorialsteacher.com/.* [Online]

—. http://www.tutorialsteacher.com/linq/what-is-linq. *http://www.tutorialsteacher.com/.* [Online]

Wiederhold, Gio. 2014. *Valuing Intellectual Capital.* 2014.