VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Olli-Pekka Sirviö

Sharepoint Extranet AngularJS:llä

Case Visualweb Oy

Liiketalous
2016

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelmaa

TIIVISTELMÄ

Tekijä	Olli-Pekka Sirviö
Opinnäytetyön nimi	Sharepoint Extranet AngularJS
Vuosi	2016
Kieli	suomi
Sivumäärä	40
Ohjaaja	Raija Tuomaala

Tämä opinnäytetyö käsittelee Sharepoint-ratkaisua AngularJS-ohjelmistokehyksen avulla. Työn tarkoituksena oli toteuttaa uusi sivusto Sharepoint-dokumenttikirjastoja varten. Tämä työ tehtiin toimeksiantona Visualweb Oy:lle.

Tavoitteena oli tehdä Sharepoint-sivusto, joka toimisi varsinaisen Sharepoint-palvelimen etusivuna ja käyttöliittymänä. Sivusto toimii asiakkaalla käyttöliittymänä dokumenttikirjastoon. Käyn työssä läpi ne teknologiat, joita projektissa käytettiin. Työssä esitellään AngularJS,Sharepoint ja REST-rajapinnan teknologiat.

Työn tuloksena syntyi sivusto, joka tehtiin Visualweb Oy:n asiakkaan käyttöön. Sivuston käytettävyys todettiin paremmaksi kuin perinteiset Sharepoint-ratkaisut. Sivusto on toimitettu Visualweb Oy:n asiakkaalle, mutta käyttökokemuksia ei käsitellä työssä, koska asiakas ei halua jakaa kyseisiä tietoja.

Avainsanat Sharepoint, AngularJS, REST, AngularJS Datatables, AJAX

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Business Information Technology

ABSTRACT

Author	Olli-Pekka Sirviö
Title	Sharepoint Extranet AngularJS
Year	2016
Language	Finnish
Pages	40
Name of Supervisor	Raija Tuomaala

This thesis studies how to make a Sharepoint page solution by using the AngularJS framework. The objective of this solution is to make a new page solution for Sharepoint document libraries. This project was developed for Visualweb Oy.

The goal of the project is to make a Sharepoint page that would then replace the usual Sharepoint front page and user interface. The page works as a user interface for the customer's document library. This thesis covers the technologies that were used in the solution. These technologies are AngularJS,Sharepoint, REST API.

The solution was made for a customer of Visualweb Oy. The usability of the solution was confirmed to be better than a regular Sharepoint-solution. The page was delivered for the customer but as they do not wish their name to appear on this project, user experiences were not covered in this work.

Keywords Sharepoint, AngularJS, REST, AngularJS Datatables, AJAX

SISÄLLYS

TIIVISTELMÄ

ABSTRACT

1 JOHDANTO 5

1.1 Toimeksiantaja 5

1.2 Tavoitteet 5

1.3 AngularJS Sharepoint-sivuston räätälöimisessä 6

2 SHAREPOINT 7

2.1 Dokumenttikirjasto 7

2.1.1 Metatieto 9

2.1.2 Sisältötyypit 10

2.2 Hakutoiminnot 11

2.3 REST-hakurajapinta 12

2.3.1 Getbytitle 12

2.3.2 Hakukysely 13

2.3.3 Käyttäjähaku 13

3 ANGULAR.JS 14

3.1.1 Moduuli 14

3.1.2 Kontrolleri 15

3.1.3 Riippuvuusinjektio 16

3.1.4 Suodatin 16

3.1.5 Tiedon sitominen 17

3.1.6 AngularJS-toistin 18

3.1.7 Tapahtumat 19

3.1.8 Aikatoiminnot 20

3.1.9 Tiedon hakeminen palvelimelta 21

3.1.10 Datatables 22

4 SIVUSTON TOTEUTUS 25

4.1 Yleiskuvaus 25

4.2 Kehitysympäristö 25

4.3 Sivuston rakenne 26

4.3.1 Käyttöliittymä 27
4.4 Etusivu 27
4.5 Dokumenttinäkymät 29
4.5.1 Dokumenttinäkymien ohjaimet 30
4.6 Käyttäjätiedot 32
4.7 Palvelut 34
4.7.1 Suodattimet 34
4.7.2 Aikatoiminnot 35
5 YHTEENVETO 37
LÄHTEET 39
LIITTEET

KÄSITTEET

AJAX	Asynchronous Javascript and XML on kokoelma tekniikoita, jotka mahdollistavat sivuston päivittämisen osittain
Active Directory	Microsoftin käyttäjätietokanta ja hakemistopalvelu (pitää sisällään käyttäjätietoja)
ATON	Aton on opinnäytteessä käsitellyn asiakkaan dokumenttikirjasto
DOM	Document Object Model eli suomeksi dokumenttimalli on lyhenne, joka kuvaa HTML-sivun puurakennetta
Extranet	Suljettu verkkosivusto, jonka avulla yritys voi jakaa tietoja asiakkaan kanssa
JSON	Javascript Object Notation on avoimen standardin tiedostomuoto
MVC	MVC eli Model-View-Controller (Malli-Näkymä-Ohjain) on ohjelmistoarkkitehtuurityyli, joka erottelee ohjelman eri osiot kolmeen alueeseen.
REST	Represational State Transfer HTTP-protokollaan perustuva arkkitehtuurimalli
Sharepoint	Microsoftin Web-sovellusalusta
Sharepoint-lista	Sharepointin tapa tallettaa tietoa Sharepoint-ympäristössä. Käytätetään myös termiä dokumenttikirjasto.
XML	Extensible Markup Language on merkintäkielen yläkäsite ja tiedostomuoto

KUVIO- JA TAULUKKOLUETTELO

Kuvio 1. Sharepoint dokumenttikirjaston rakenne 8

Kuvio 2. Sharepointin listanäkymä asiakirjoista 9

Kuvio 3. Sharepointin dokumentin metatiedot 10

Kuvio 4. Sisältötyypit ja sen metatiedot 11

Kuvio 5. Valmis Datatables taulu 24

Kuvio 6. Sivuston etusivu 26

Kuvio 7. Uutiset etusivulla 27

Kuvio 8. Uutiset lukutila 28

Kuvio 9. Etusivun ”viimeisimmät päivitetyt tiedostot”-listaus 29

Kuvio 10. Table-näkymä Technical Documents-alasivustosta 30

Kuvio 11. Key Account Team-näkymä 32

Kuvio 12. Näkymä sivustolla olevasta listasta 33

1 JOHDANTO

Opinnäytetyönä Visualweb Oy:lle toteutetaan toimeksiantona Sharepoint-sivusto. Työ on toteutetaan AngularJS-ohjelmistokehystä käyttäen. Käytän opinnäytetyössä paljon esimerkkejä varsinaisesta sivustosta.

1.1 Toimeksiantaja

Toimeksiantaja on Visualweb Oy. Visualweb Oy on Sharepoint-ratkaisuja kehittävä ohjelmistotalo. Yrityksen toimialueena on koko maailma. Sivusto on toimitettu Visualweb Oy:n asiakkaalle ja heidän pyynnöstään en mainitse asiakkaan nimeä tässä työssä.

1.2 Tavoitteet

Työn tavoitteena on tehdä toimiva Extranet-ratkaisu Visualweb Oy:n asiakkaalle. Visualweb Oy:n asiakas on sulautettuja järjestelmiä rakentava yritys. Työn tarkoituksena on tehdä nopea ja toimiva sivusto, josta asiakkaan omat asiakkaat saisivat helposti ja nopeasti tarvittavat dokumentit laitteillensa. Työ käyttää hyväkseen Sharepointin listarakennetta. Tarkoituksena on, että sivustolta saataisiin nopeasti ja helposti ulos tarvittava tieto hakusanoilla tai listoja selaamalla. Asiakkaan muiden palveluiden yhdistäminen järjestelmään on myös tärkeä saada toimimaan. Extranet-sivua voidaan ajatella asiakasportaalina, joka pitää sisällään lehdistömateriaalin, promootiokuvien ja tiedostojen helpon jakamisen. (Roine 2015) Tämä sivusto pitää sisällään tuotteiden ohjekirjat, ohjelmistot ja mainosmateriaalit.

1.3 AngularJS Sharepoint-sivuston räätälöimisessä

Opinnäytetyö on toteutettu Sharepoint-sivustona Visualweb Oy:n käyttämälle Metronic-pohjalle. Metronic on Visualweb Oy:n ostama sivustopohja. Opinnäytetyö jakautuu niiden teknologioiden teoriaan, joita työssä olen käyttänyt ja soveltanut ja toiminnalliseen osaan, jossa esittelen opinnäytetyön tuloksia ja ratkaisuja.

Teoriaosuudessa käyn läpi Sharepointin, AngularJS:n ja sen liitännäisen AngularJS-Datatablesin, joka tuo jQueryn Datatables-liitännäisen AngularJS-ympäristöön. AngularJS valittiin projektin kieleksi, koska Microsoft osallistuu AngularJS 2.0 kehitykseen ja on vahvasti mukana AngularJS:n käyttämisessä Sharepoint-alustalla.

Aineistoa on rajattu vain niihin osioihin ja teknologioihin, joita työssä käytetään.

Opinnäytetyö on tehty projektista, jonka lopputulos on asiakkaalla jo käytössä. Opinnäytetyössä oli tarkoitus lähestyä Sharepoint-ratkaisuja AngularJS-näkökulmasta ja tehdä suorituskyvyltään nopeampia sivustoratkaisuja.

Opinnäytetyön tarkoituksena on tutkia onko mahdollista tehdä suorituskykyisempiä ratkaisuja Sharepoint Extranet-sovelluksiin käyttäen AngularJS:ää. Tämän vuoksi työ keskittyy AngularJS- ja AngularJS Datatables-ratkaisuihin, joita työssä käytin. Tarkoituksena oli rakentaa tehokkaampi tapa esittää Sharepointissa olevia dokumenttikirjastoja. Muina tarkoituksina oli tutustua miten AngularJS:ää voidaan käyttää tiedon hakemiseen Sharepointin dokumenttikirjastoista käyttäen AJAX:ia tiedon välitykseen ja hakemiseen. AJAX eli Asynchronous Javascript and XML on kokoelma tekniikoita, jotka mahdollistavat vuorovaikutteiset Web-sovellukset. Tähän ongelmaan saadaan vastaus käyttämällä REST-rajapintaa. Opinnäytetyön esimerkeissä näkyy oleellinen osa määrittelyn alusta.

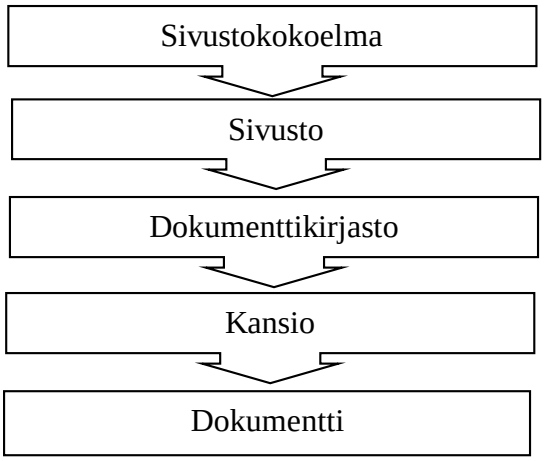
2 SHAREPOINT

Sharepoint on Microsoftin kehittämä ohjelmistokokonaisuus, jota käytetään yleisimmin Intranet-verkkopalveluiden alustana, ryhmätyöskentelyn tukena ja dokumenttien hallinnassa. Sharepoint tarjoaa myös toiminnallisuudet työnkulkujen, lomakkeiden ja asiakirjojen hallintaan. Sharepointin ensimmäinen versio julkaistiin vuonna 2001. Tässä työssä on käytetty Sharepoint 2013-versiota. (Roine 2015, 8)

Tässä työssä on tehty niin sanottu räätälöity sivusto. Työssä tämä on toteutettu tekemällä oma sivustopohja (Master Page), jonka avulla on voitu tuoda Sharepointiin oma tyylipohja. Sivusto rakennetaan tämän tyylipohjan päälle. Tästä huolimatta sivuston pohjalla on vahvasti Sharepoint (Roine 2015, 155). Muita vaihtoehtoja räätälöintiin on FTC (Full Trust Code) jossa sivusto toteutetaan ilman suojauksia , Hiekkalaatikko (Sandbox) jossa sivusto tehdään suojatussa tilassa ja App Model, joka on yhdistelmä näitä kahta (Roine 2015, 149).

2.1 Dokumenttikirjasto

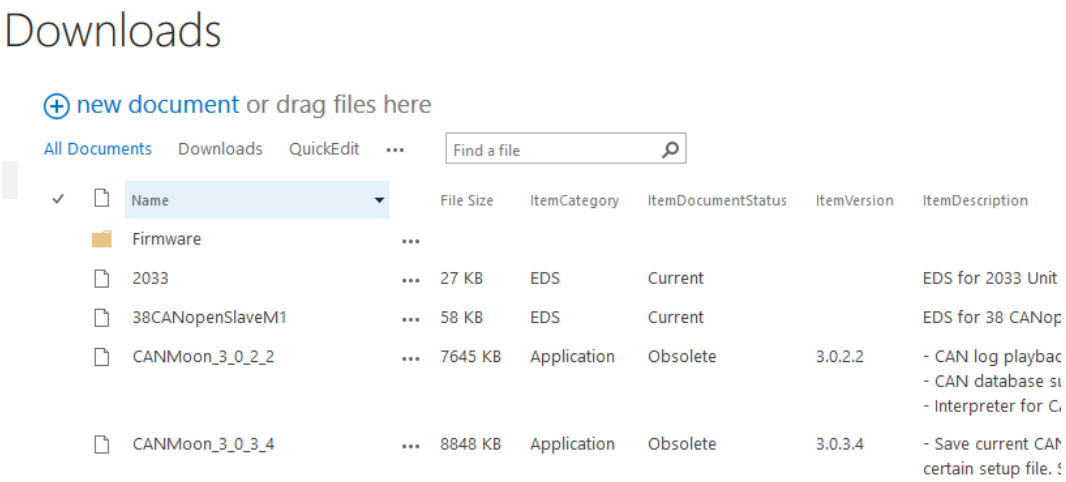
Sharepoint 2013-toteutuksissa on keskeisenä osana työtilat ja dokumenttien hallinta. Näitä työtiloja käytetään dokumenttien tallentamiseen. Sharepointissa nämä työtilat ovat Sharepoint-sivustoja ja alisivustoja, joista löytyy yksi tai useampi dokumenttikirjasto. Tämä helpottaa käyttöoikeuksilla kirjaston rajaamista, sillä yksittäisten dokumenttien sijaan voidaan rajata kokonaisia sivustoja käyttäjille. Sivustot ovat toisiinsa hierarkkisessa järjestyksessä, jossa pääsivustolla on alisivuja ja joilla voi olla omia alisivuja. Dokumenttikirjastoja voidaan vielä laajentaa ottamalla käyttöön kansioita, joihin dokumentteja voidaan tallentaa. (Roine 2015, 69)



Kuvio 1. Sharepointin rakenne (Roine 2015, 70)

Dokumenttikirjastot tarkoittavat samaa kuin Sharepointin listat. Näiden valmiiden listojen lisäksi voi sivuston ylläpitäjä määritellä mukautettuja listoja. Lista on tapa näyttää tietoja sarakkeittain ja riveittäin. (Roine 2015, 8). Käytän termiä ’lista’ opinnäytetyössä Sharepoint-listasta.

Listoilla olevat tiedot on tallennettu sarakkeisiin, joita voidaan hakea esimerkiksi REST-kyselyitä käyttäen. REST eli Represational State Transfer on HTTP-protokollaan perustuva arkkitehtuurimalli. Nämä tiedot pitävät sisällään valmiita toimintoja tai käyttäjän itse määrittelemiä tietoja.



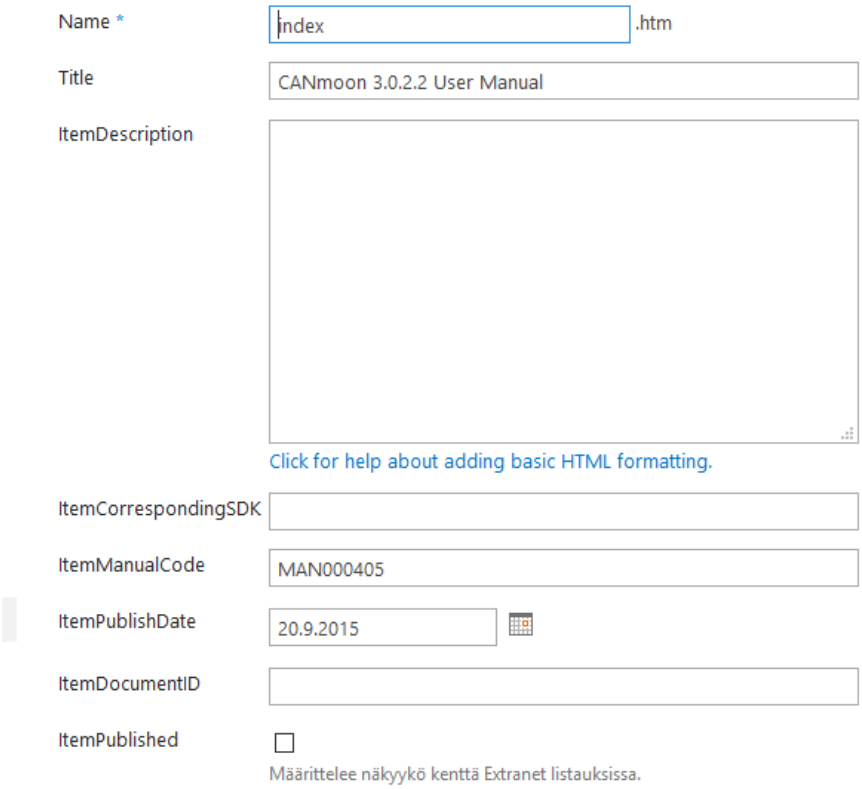
Kuvio 2. Sharepointin listanäkymä asiakirjoista

2.1.1 Metatieto

Metatiedot ovat dokumentteja kuvaavia tietoja. Näillä tiedoilla on merkitys dokumenttien hallinnassa ja informaatioarkkitehtuurissa. Näitä metatietoja ovat esimerkiksi

- haku
- ryhmittely
- toimintalogiikka ja säännöt
- autenttisuuden, luotettavuuden ja eheyden varmistaminen
- käyttöoikeudet

Näillä metatiedoilla on myös omat metatietotyypit, joista yleisimpiä ovat kuinka monta riviä tekstiä tieto pitää sisällään, päivämäärä ja käyttäjä tai ryhmä. (Roine 2015, 70)



Kuvio 3. Sharepointin dokumentin metatiedot

2.1.2 Sisältötyypit

Sisältötyyppi (Content Type) on keskeinen osa Sharepointia. Se määrittelee miten sivut, dokumentit ja muut listaobjektit erotellaan toisistaan. (Roine 2015, 73). Sisältötyyppiä voidaan käyttää myös Sharepointin hakutoiminnoissa rajaamaan haettavaa tietoa sen sisältötyypin perusteella. Sisältötyyppi ei ole sama kuin tiedostotyyppi, vaan se on tapa sisällyttää samanlaisia tiedostoja sisällön perusteella eikä tiedoston perusteella. Tämä tehdään määrittelemälle niille sisältötyyppi. (Roine 2015, 18-19)

Kuvio 4. Sisältötyypit ja sen metatiedot. Metatiedot pitävät sisällään li

2.2 Hakutoiminnot

Sharepointin hakutoiminnot saadaan päälle laittamalla hakupalvelu päälle. Sharepointissa hakutoiminnallisuus säädetään indeksoimaan tietoja yhdellä kolmesta tavasta. Indeksointi tarkoittaa kirjaston sisällön keräämistä tiedostoon. Indeksointi on se toimenpide milloin tiedot kerätään. (Roine 2015, 92-94)

Yksi tavoista on täysi indeksointi, jota ajetaan harvemmin. Tämä haku indeksoi kaikki tiedostot palvelimelta ja luo uuden indeksin. Tällä keinoin on mahdollista tehdä uusi indeksi, jos vanha pitää sisällään vanhentunutta tietoa. Tyypillisesti tämä indeksointi ajetaan kerran viikossa, koska se on aikaa vievä prosessi. (Roine 2015, 92-94)

Toinen tapa on inkrementaalinen indeksointi, jolloin indeksointi suoritetaan esimerkiksi puolentunnin välein. Tämän inkrementaalisen indeksoinnin tarkoituksena on indeksoida tietoja, kun ne muuttuvat tai uusia tietoja tai tiedostoja on lisätty. (Roine 2015, 92-94)

Kolmas vaihtoehto on jatkuva indeksointi. Se on Sharepoint 2013:sta uusia ominaisuuksia, jonka ideana on täydentää täysindeksointia ja inkrementaalista indeksointia. Tämän indeksoinnin vahvuus on se, että useampaa indeksointia voidaan suorittaa päällekkäin (Roine 2015, 92-94). Näitä indeksejä käytetään käyttämällä REST-rajapintaa

2.3 REST-hakurajapinta

Työssä on käytetty paljon hyödyksi REST (REpresentational State Transfer)-rajapintaa. REST käyttää hyväkseen HTTP-protokollan toimintoja. Nämä toiminnot eli HTTP-metodit ovat GET, POST, PUT ja DELETE. Työssäni käytän vain GET- ja POST-metodeja. Ne tarkoittavat tiedon hakemista palvelimelta (GET) ja tiedon lähettämistä palvelimelle (POST). (Microsoft Dev Center 2015a)

Sharepointin REST-tuki mahdollistaa Sharepoint-objektien käyttämisen REST-kutsuja hyväksi käyttäen. Sharepointin REST-kutsut rakennetaan käyttämällä osoitetietoa ja määrittelemällä REST-kysely osoiteriville. Mahdollisia hakuvaihtoehtoja ovat listakohtainen haku ja sivuston kattava haku. Työssäni käytän koko sivuston kattavaa hakua. (Microsoft Dev Center 2015a)

```
http://server/site/_api/web
```

2.3.1 Getbytitle

Sharepointissa olevat REST-kyselyt voidaan ohjata hakemaan suoraan listalta käyttämällä ’getbytitle’-komentoa. Tällä komennolla pystytään hakemaan listalla olevia sarakkeita käyttämällä listan nimeä hakusanana. (Microsoft Dev Center 2015b)

```
http://server/site/_api/web/lists/getbytitle('listname')/items
```

Komento hakee kaikki tällä listalla olevat ”items” eli ne sarakkeet, joissa on tietoa. Tätä voidaan käyttää myös tietojen lähettämiseen palvelimelle, kun käytetään POST-metodia. Työssäni olen käyttänyt tätä keinoa vain POST-kyselyissä. (Microsoft Dev Center 2015b)

2.3.2 Hakukysely

Toinen tapa hakea tietoa on käyttää hyväkseen Sharepointin hakutoimintoja tekemällä kyselyitä sen indeksiin. Hakukyselyt indeksiin tehdään käyttämällä hakusanoja tai mahdollisia sisältötyyppejä, jolloin on mahdollista hakea koko palvelimella sijaitsevia tiedostoja eikä pelkästään listoilla olevia tietoja. Haku tapahtuu REST-kyselyn kautta komennolla, esimerkiksi (Microsoft Dev Center 2015a)

```
http://server/_api/search/query?querytext='sharepoint'
```

Tällä tavoin on mahdollista hakea kaikkia tietoja palvelimelta, joissa on se hakusana jolla tietoa haetaan. Tätä voidaan rajata määrittelemällä ne sarakkeet listalta, joita halutaan hakea käyttämällä selectproperties-parametriä, esimerkiksi (Microsoft Dev Center 2015a)

```
http://server/_api/search/query?querytext='sharepoint'&selectproperties='Title,Author
```

Tästä on vielä mahdollista rajata tuloksia vielä käyttämällä ”RefinementFilters”-parametriä, jolloin on mahdollista rajata tuloksia määrittelemällä esimerkiksi jokin haettava arvo. Tällöin palvelu palauttaa vain tämän parametrin kanssa yhteensopivia rivejä, esimerkiksi (Microsoft Dev Center 2015a)

```
/query?querytext='sharepoint'&refinementfilters='fileExtension:equals("docx")'
```

2.3.3 Käyttäjähaku

Kolmas REST-kyselyissä käyttämäni palvelu on PeopleManager. Tällä palvelulla voidaan hakea käyttäjän tietoja palvelimelta. Työssäni olen käyttänyt tätä palvelua vain tietoja hakemiseen nykyisestä käyttäjästä. Palvelussa käyttäjätiedot on tallennettu Active Directoryyn.

Nämä kyselyt tapahtuvat käyttämällä SP.UserProfiles.PeopleManager-toimintoa kyselyissä. Alla esimerkki siitä miten haetaan nykyisen käyttäjän tiedot. (Microsoft Dev Center 2015c)

```
http://server/_api/SP.UserProfiles.PeopleManager/GetMyProperties?$select=*”
```

3 ANGULAR.JS

AngularJS on avoimen lähdekoodin Javascript-ohjelmistokehys, joka mahdollistaa helppojen yhden sivun ratkaisujen kehittämisen. AngularJS:n kehitti alun perin Googlen työntekijät Miško Hevery ja Adam Abrons. Ensimmäinen versio AngularJS:stä julkaistiin vuonna 2012 ja seuraavan version (versio 2.0) kehitys on jatkunut jo usean vuoden ajan. Versio 2.0 kehitykseen on tullut mukaan myös Microsoft. AngularJS on saanut nimensä ns. aaltosulkeista ({ }).

AngularJS käyttää ns. MVC-arkkitehtuuria. MVC (Model-View-Controller) eli malli-näkymä-kontrolleri. AngularJS:n tarkoituksena on tuoda juuri tämä MVC-arkkitehtuuri selaimiin. MVC-arkkitehtuuri perustuu kolmeen osa-alueeseen. Model eli malli tallettaa tiedon kontrollerista saadun tiedon perusteella. Controller eli kontrolleri päivittää tietoja käyttäjän antamien komentojen perusteella. View eli näkymä näyttää sen, mitä käyttöliittymässä näytetään mallista saadulla tiedolla. AngularJS-applikaatioissa tämä näkymä on Document Object Model (DOM), jossa kontrollerit ovat JavaScriptin luokkia. Document Object Model on rajapinta HTML-dokumenttien rakenteen ja sisällön muokkaamiseen. (Green 2015).

3.1.1 Moduuli

Moduuli on se osa AngularJS:ää, jossa sovellus ja sen tiedot sijaitsevat. Moduuli kokoaa sovelluksen osat yhteen paikkaan. Moduuli pitää siis sisällään kontrollerit, suodattimet ja palvelut. Moduuli on siis pääasiallinen osa, jota kutsumalla sivuston alussa aktivoidaan kaikki sen sisällään pitämät toiminnot. Uusi moduuli luodaan angular.module-komennolla, esimerkiksi. (AngularJS dokumentaatio 2015a)

```
angular.module('metronicApp', []);.
```

Tässä vaiheessa annetaan kaksi parametriä; moduulin nimi ja sen riippuvuudet. Riippuvuudet ovat muita moduuleita – joko itse tehtyjä tai valmiita. Riippuvuudet

saadaan moduuliin mukaan käyttämällä riippuvuusinjektiota. Riippuvuusinjektiosta lisää kappaleessa 3.1.3.

```
var MetronicApp = angular.module("MetronicApp", [
    "datatables",
    "datatables.tabletools",
    "ui.router",
    "angular.filter",
    "ui.bootstrap",
    "oc.lazyLoad",
    "ngSanitize"
]);
```

Tämä kutsutaan HTML-koodissa ng-app-komennolla. Ng-app ja data-ng-app ovat toistensa kanssa vaihdettavia, joista data-ng-app on vanhempi tapa tehdä, esimerkiksi

```
<html ng-app="MetronicApp">
```

Tämän jälkeen ovat kaikki tämän moduulin tai sen riippuvaisuuksien sisällä olevat kontrollerit, suodattimet ja muut palvelut käytettävissä sen elementin sisällä, jossa moduulia kutsuttiin. (AngularJS dokumentaatio 2015b)

3.1.2 Kontrolleri

Kontrolleri on se osa, jolla voidaan rajata lähtöarvoja sovelluksin osille ja määritellä $scope-lisäominaisuuksia. Kontrolleri luodaan Angular.module controller()-metodilla. Controller-metodin luomisen yhteydessä on syytä katsoa myös riippuvuudet riippuvuusinjektiolla kappale 3.1.3. Alla on esimerkki siitä miten kontrolleri luodaan ja miten siihen saadaan riippuvuudet mukaan. (Green 2015)

```
MetronicApp.controller('Programming_ManualsController',
```

Tämä kontrolleri kiinnitetään dokumenttiin ng-controller-komennolla. Kontrollerit pitävät sisällään toimintoja, joita voidaan kutsua ng-controller-komennolla kiinnitetyn HTML-elementin sisältä. Useaa kontrolleria voidaan kutsua toistensa sisällä tarvittaessa, mutta niiden tulee olla eritelty omiin elementteihinsä. Esimerkki siitä

miten kontrolleri sidotaan elementtiin, joka pitää sisällään toisen kontrollerin. (AngularJS dokumentaatio 2015d)

```
<div ng-controller="Technical_DocumentsController as showCase" ng-init="init()">
```

3.1.3 Riippuvuusinjektio

AngularJS pitää sisällään komponentteja, jotka pitää määritellä kontrolleria tai moduulia tehdessä. Tähän käytetään riippuvuusinjektiota (Dependency Injection), jossa sovelluksen osia voidaan käyttää riippuvuuden määrittelyn jälkeen. Riippuvuusinjektiolla saadaan tehtyä luokkia, joissa ne itse pyytävät vain ne tarvitsevansa osat joita niissä käytetään. Näitä voivat olla palvelut, direktiivit, suodattimet ja animaatiot.

Riippuvuusinjektiota käytetään niin moduulin kuin kontrollerin luonnin yhteydessä. Moduulin riippuvuusinjektiolla voidaan kutsua muita moduuleita laajentamaan moduulin toiminnallisuuksia. Esimerkiksi projektissa käytettävä Datatables on kutsuttu tällä tavoin. (AngularJS dokumentaatio 2015e)

```
MetronicApp.controller('Programming_ManualsController', ['$scope','$http','DTOptionsBuilder',
```

AngularJS pitää sisällään valmiita palveluita, joita voidaan käyttää koodissa lisäämällä ne riippuvuusinjektioon. Näitä palveluita ovat esimerkiksi $scope, $http, $interval ja $timeout. Näitä palveluita käytän työssäni.

3.1.4 Suodatin

Suodatin mahdollistaa AngularJS:ssä esitettävän tietosisällön suodatuksen. Näitä suodattimia voidaan käyttää AngularJS:n eri toiminnoissa, kuten tiedon sitomisessa, mutta myös kontrollereissa voi käyttää suodattimia. AngularJS:ssä tämä suodatin tehdään antamalla sille nimi moduulin kanssa. (AngularJS dokumentaatio 2015f)

Tämän jälkeen voidaan suodattimelle määritellä toimintoja. Näitä toimintoja voivat olla esimerkiksi järjestyksen muuttaminen tai sisällön muuttaminen. Alla on esimerkki suodattimesta, joka korvaa viivat pisteillä.

```
MetronicApp.filter('viivat', function () {
  return function (input) {
    if (input) {
      return input.replace(/-/g, '.');
    }
  };
});
```

AngularJS-suodatus tapahtuu käyttäen pystyviivaa tietueessa. Näitä suodattimia voi olla useampi sidottua tietoa kohden.

```
{{ teksti | suodatin}}
```

Valmiita ja itse tehtyjä suodattimia voidaan käyttää sekaisin. Niiden järjestys on sen mukainen, missä järjestyksessä suodattimet ovat tiedossa. Alla on esimerkki päivätiedon lyhentämisestä ja viivojen korvaamisesta pisteillä:

```
<td>
    {{document.ItemPublishDate | limitTo:10 | viivat}}
</td>
```

AngularJS:ssä on myös sisäänrakennettuja suodattimia, jotka tekevät erilaisia toimintoja, kuten tietojen järjestys ng-repeatin yhteydessä. AngularJS:ään toistorakenne eli ng-repeat esitellään kappaleessa 3.1.6. Näitä valmiita suodattimia ovat esim. orderBy, jolla voi määritellä, minkä mukaan tieto järjestetään. (AngularJS dokumentaatio 2015g). Alla on esimerkki siitä miten ng-repeat sisällä olevista useammasta suodattimesta

```
ng-repeat="document in documents | omit: newsFilter | omit: newsEndFilter
```

3.1.5 Tiedon sitominen

AngularJS $scope on se osa, joka kommunikoi näkymän ja kontrollerin välillä. Scope toimii siis liimana näiden kahden osan välillä ja muokkaamalla scopea voidaan muokata näkymää sivustolla. AngularJS-näkymällä ja -kontrollerilla on yhteys scopeen mutta ei toisiinsa. Tämä helpottaa testaamista. Scopea kutsutaan

kontrollerin alussa käyttäen riippuvuusinjektiota, kuten kappaleessa 3.1.2 on kerrottu. (AngularJS dokumentaatio 2015h). Nämä tiedot sidotaan HTML-dokumenttiin.

Perinteisessä HTML-sivussa tieto on tallennettu yksisuuntaisesti eli tieto liikkuu vain, kun koko sivu lähetetään asiakkaalle. Tästä termistä käytetään kahdenlaista nimitystä eli ’tiedon sitominen’ tai ’tiedon kytkentä’. Tässä työssä käytän termiä ’tiedon sitominen’. Kahden suuntainen tiedon sitominen (data-binding) on AngularJS:n ominaisuus, joka mahdollistaa yhteyden käyttöliittymän ja tietorakenteen välille, jossa tapahtuvat muutokset päivittyvät käyttäjälle jatkuvasti.

Kaksisuuntainen tiedonsidonta mahdollistaa nopean tiedon näyttämisen, koska kehittäjän ei tarvitse tehdä monimutkaisia tietosidontoja, vaan pystyy sen helposti yhdistämään sivustoon. Tämä aiheuttaa ongelmia, jos käytettävät tietomäärät ovat suuria.

Tiedon sitominen HTML-dokumenttiin voidaan toteuttaa helposti antamalla näytettävän tiedon sisältö aaltosulkeista, esimerkiksi

```
{{ Tieto }}
```

Tällä tavoin sidottuna tieto päivittyy, jos se muuttuu ohjelmassa. Nämä lausekkeet voivat pitää sisällään myös Javascript-tyylistä koodia tai laskutoimituksia (AngularJS dokumentaatio 2015c), esimerkiksi

```
{{ 1 + 2 }}
```

3.1.6 AngularJS-toistin

Ng-repeat on toiminto, jolla voidaan luoda elementtejä tieto kerrallaan HTML-koodiin esimerkiksi <table>-elementin sisällä

```
<td ng-repeat="document in documents">
```

Ng-repeatin sisälle on mahdollista sisällyttää myös toimintoja ja filttereitä, joiden avulla voidaan pienentää tulosjoukkoa käyttäen AngularJS:n valmiita tai itsetehtyjä filttereitä.

Tämän toiminnon avulla on mahdollista toistaa tapahtumia niin pitkään kunnes esimerkiksi kokoelmasta saadut tiedot ovat toistettu HTML-elementtiin. (AngularJS dokumentaatio 2015j)

3.1.7 Tapahtumat

Kontrollerin sisällä voi olla ominaisuuksia, jotka aktivoidaan ng-init-komennolla. Tällä tavoin voidaan kutsua kontrollerin toiminnallisuuksia vain silloin kuin ng-init on ajettu. Nämä tapahtumat ajetaan, kun niitä kutsutaan HTML-koodista.

Kun ng-init-tapahtuma tulee vastaan HTML-koodissa sivunlatauksen yhteydessä, ajaa se ne komennot, jotka löytyvät koodin sisältä. Tässä voidaan antaa ng-init-komennolle myös lähtöarvoja, joita on esimerkiksi aikaisemmin tehty muuttuja. (AngularJS dokumentaatio 2015k)

```
<div class="nav-justified" ng-init="init();init2()">
```

Toinen tapahtumatoiminto, jota projektissa käytetään, on ng-click. Tämä on tapahtuma, jolla voidaan kutsua tapahtumaa ohjelmasta, kun elementtiä painetaan. Tällä metodilla on mahdollista antaa lähtötietoja, joita käyttämällä voidaan ohjelmalle antaa tietoja HTML-koodista. (AngularJS dokumentaatio 2015l). Alla on esimerkki tapahtumasta jossa elementin painaminen laukaisee.

```
<li class="media" ng-href='#here' ng-click="ItemClick(sidebarUser.AccountName);"
```

Tämän jälkeen voidaan ajaa Javascript-koodia näitä lähtötietoja käyttäen.

```
$scope.ItemClick = function (Chosen) {

    $scope.filtersname = Chosen;

}
```

3.1.8 Aikatoiminnot

AngularJS pitää sisällään palveluita Javascriptin perustoiminnoille interval ja timeout. Nämä palvelut toimivat nimellä $timeout ja $interval.

Timeout eli odotus on palvelu, jonka avulla voidaan viivästyttää tapahtumien suorittamista koodissa. Tämä on hyödyllistä, koska AngularJS ajaa koko koodin yhtä aikaisesti, joka voi aiheuttaa ongelmia AJAX-kutsujen kanssa. (AngularJS dokumentaatio 2015m)

```
$timeout(function(){
    $http({
    method: 'GET',
    url: "/_api/SP.UserProfiles.PeopleManager/getuserprofilepropertyfor(accountName=@v,propertyname
    headers: { "Accept": "application/json;odata=verbose"  }
}).
success(function (data, status, headers, config) {
    $scope.homeOffice = data.d;
```

Interval eli aikaväli suorittaa pätkän koodia aikavälin sisällä. Tätä hyödyntämällä voidaan kutsua koodia esimerkiksi jatkuvaa AJAX-päivitystä ennalta määrätyin aikavälein. (AngularJS dokumentaatio 2015n). Alla esimerkki sivuston otsakkeen vaihtuminen aikavälein.

```
var titleset = $interval(function() {

    document.title = "Extranet Epec";

}, 100);
```

Sivustolla on myös käytetty Moment.js:ää, joka mahdollistaa aikavälien mittaamisen päivämäärittäin. Moment.js on tehty toimimaan niin selaimen kuin Node.js:n avulla. Sivustolla tätä toimintoa käytetään vain aikavälien mittaamiseen. (MomentJS dokumentaatio 2015)

```
$scope.paiva = moment().subtract(30, 'days').format('YYYY-MM-DD');
```

3.1.9 Tiedon hakeminen palvelimelta

AngularJS:n $http on keskeinen AngularJS-palvelu. Se mahdollistaa kommunikaation palvelimelle käyttäen XMLHttpRequest-objektia (Lähde: AngularJS Docs $http). XMLHttpRequest on objekti, joka mahdollistaa tiedon siirtämisen asiakkaan ja palvelimen välillä. Se mahdollistaa osan sivun sisällön päivittämisen ilman, että koko sivua tarvitsee päivittää tiedon hakemiseen. Tämä on keskeinen osa AJAX-ohjelmointia. (Mozilla.org 2015).

AngularJS AJAX-kutsut suoritetaan $http-palvelulla, jonka avulla voidaan hakea tietoja palvelimelta käyttäen REST-rajapinnan komentoja. AngularJS tukee myös $resource-palvelua (AngularJS dokumentaatio 2015p), jonka avulla on mahdollista toteuttaa monimutkaisia toimintoja mutta sitä ei tässä projektissa käytetä.

Kun $http-palvelua käytetään, on sille syötettävä ainakin kaksi parametriä. Nämä parametrit on se osoite josta tietoa haetaan ja se mitä metodia käytetään kyselyssä palvelimelle, näitä voivat olla tiedon haku (GET) tai tiedon tallennus (POST). Muita metodeja ovat esimerkiksi tiedon päivitys (UPDATE) tai tiedon poistaminen (DELETE), mutta niitä ei tässä työssä käytetä.

Tässä työssä on myös määritelty se, missä muodossa tieto esitetään. Tähän on käytetty http-headeriä. Header määrittelee sen, että tieto jota pyydetään palvelimelta, on json-muodossa. (AngularJS dokumentaatio 2015o), esimerkiksi

```
$http({
   method: 'GET', url: $scope.query + "&rowsperpage=0",
   headers: { "Accept": "application/json;odata=verbose" }
}).
```

Tämän jälkeen, kun pyyntö on onnistunut, käsitellään sen tieto määrittelemällä, mitä tietoja tästä json-tiedostosta halutaan. Virhetilanteita varten on olemassa error-funktio, jossa voidaan määritellä virheviesti, jos AJAX-pyyntö epäonnistui.

Tieto tallennetaan ohjelmassa $scope-metodin kanssa omiin muuttujiinsa. Tämä tieto tallentuu array- eli taulukkomuodossa, jolloin jokaiselle tietueelle on oma alue, esimerkiksi (AngularJS dokumentaatio 2015o).

```
success(function (data, status, headers, config) {
    $scope.documents = [];
    //Results
    angular.forEach(data.d.query.PrimaryQueryResult.RelevantResults.Table.Ro

        angular.forEach(document.Cells, function (documentRow) {
            var obj = {};

            angular.forEach(documentRow, function (documentData) {

                obj[documentData.Key] = documentData.Value;
            });
```

3.1.10 Datatables

AngularJS-Datatables on valinnainen AngularJS-laajennus, joka mahdollistaa jQuery Datatables-liitännäisen käytön AngularJS-ympäristössä. Datatables on keino esittää dataa siististi tauluissa tuoden uusia ominaisuuksia, kuten tietojen suodatuksen hakusanalla. Datatables laajentaa ja tuo lisäominaisuuksia HTML-<table> elementteihin. Datatables on alun perin jQuery-liitännäiseksi MIT-lisenssin alle. Tämä liitännäinen ei suoraan tue AngularJS-tiedon sitomista. Sitä varten on tehty AngularJS-Datatables lisäosa, joka tuo Datatables-ominaisuuden ja mahdollistaa kahden suuntaisen tiedon sitomisen Datatables-tauluissa. (Datatables Manual 2015a)

AngularJS Datatables tuodaan mukaan sivuun lisäämällä sen Javascript-tiedosto ensiksi <script> tageilla työhön, jonka jälkeen sille pitää luoda riippuvuusinjektio moduulin alussa. Kun riippuvuusinjektio on luotu, voidaan näitä toimintoja käyttää kontrollereissa.

Datatables pitää sisällään valmiita toimintoja, joiden avulla voidaan esimerkiksi tuoda lisäominaisuuksia projektiin. Näillä toiminnoilla voidaan määritellä se, miten taulu aakkostaa tai miten monta tietuetta näytetään alkunäkymässä. Tässä projektissa on käytetty Tabletools-lisäosaa, joka tuo dokumentin tulostukseen ja selaamiseen tarvittavat työkalut mukaan tauluihin.

```
        var vm = this;
vm.dtOptions = DTOptionsBuilder.newOptions()
    .withDisplayLength(25)
    .withOption('oLanguage', {"sEmptyTable": "Waiting for Data" })
    // Add Table tools compatibility
    .withTableTools('../../_layouts/15/assets/js/scripts/TableTools/
    .withTableToolsButtons([
        'copy',
```

Kun Datatables-toiminnallisuuksia määritellään, voidaan niille antaa erilaisia lähtöarvoja, jotka voidaan määritellä sarakkeittain. Näitä määrittelyvaihtoehtoja ovat DTOptions, jolla määritellään taulukkokohtaisia määrityksiä ja DTColumndef, jolla määritellään sarakekohtaisia asetuksia esimerkiksi,

- .withDisplaylength(25)

  Tämän avulla voidaan määritellä se miten monta arvoa sivusto näyttää oletusarvoisesti

- .withOption(’order’) [2, ’desc’]”

  Määrityksellä voidaan määritellä, miten tiedot järjestellään taulussa sen sarakkeen perusteella mikä tässä määritellään (Angular Datatables 2015c)

- .withTableTools

  Määrityksellä voidaan tuoda ohjelmaan taulun hallintaan liittyviä toiminnallisuuksia, kuten tulostus ja tallennus esimerkiksi PDF-muodossa. (Angular Datatables 2015b)

Sarakkeiden määrityksellä voidaan esimerkiksi piilottaa tiettyjä sarakkeita.

```
    vm.dtColumnDefs = [
        DTColumnDefBuilder.newColumnDef(1).notVisible(),
        DTColumnDefBuilder.newColumnDef(2).notVisible()
];
```

Kun Datatables halutaan sitoa table-elementtiin HTML-koodissa, on sille annettava arvo käyttämällä datatable-komentoa.

Määrittelemällä datatable=”ng” saadaan AngularJS Datatables-toiminnallisuudet mukaan elementtiin. Tässä on syytä myös sitoa mahdolliset toiminnallisuudet taulun ominaisuuksista ja sarakkeiden ominaisuuksista. (Angular Datatables 2015a)

```
<table datatable="ng" dt-options="showCase.dtOptions" dt-column-defs="showCase.dtColumnDefs"
```

Kun datatable-määritykset on annettu, voidaan tieto sitoa tauluun käyttämällä ng-repeat-komentoa. Kappale 3.1.6.

Kuvio 5. Valmis Datatables taulu

4 SIVUSTON TOTEUTUS

Tämä luku keskittyy kuvaamaan käytettyjä ratkaisuja Sharepoint-projektissa. Toteutusta tarkastellaan asiakkaan näkymästä. Koska osa sivuista pitää sisällään samanlaisia ratkaisuja, joitakin sivuista ei käydä läpi ollenkaan.

4.1 Yleiskuvaus

Projektissa rakennettiin Sharepointin päälle AngularJS:llä toteutettu Extranet-ratkaisu. Extranetin avulla voidaan jakaa tarvittavat dokumentit ja materiaalit asiakkaiden ja työntekijöiden kanssa.

Extranet koostuu sivuista, joissa Extranet-dokumenttikirjastoihin tallennetut tiedot esitetään asiakkaalle tauluissa. Extranet pitää myös sisällään yhteydet joissakin sivuissa asiakkaan toiseen palveluun Atoniin. Extranetin tarkoituksena on näyttää asiakkaalle ne tiedostot, jotka ovat saatavilla niillä tunnuksilla, jotka asiakas on saanut. Koska Extranet on asiakaskohtainen, sen tarkoitus on näyttää vain ne tiedostot, joita sivuston omistaja tahtoo asiakkaan näkevän.

Sivustolla on myös kaikille näkyviä toimintoja ja sivuja, joiden tietosisältö on aina sama. Sivustolla on useita alasivuja ja kontrollereita.

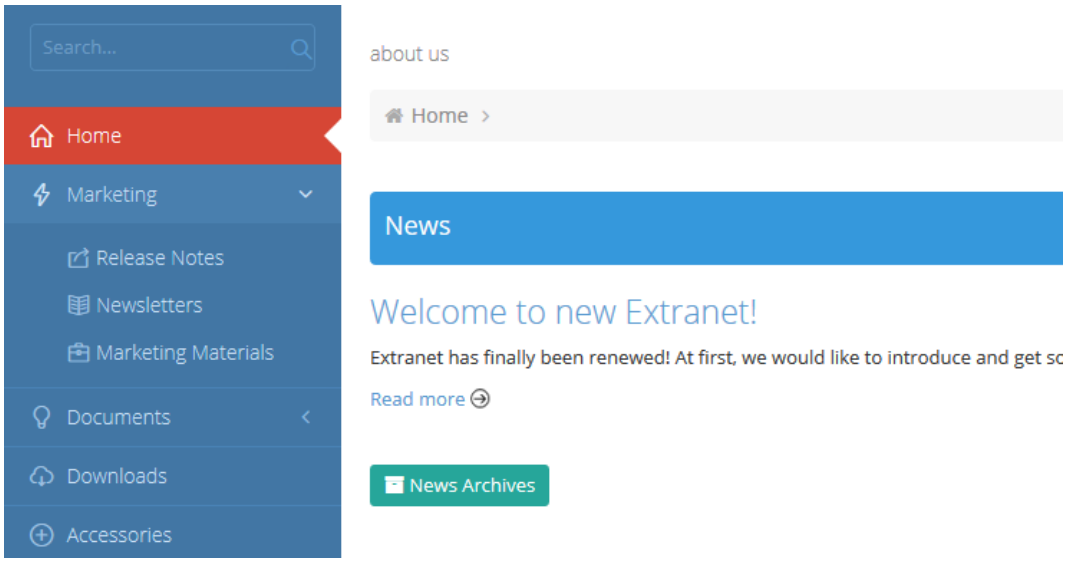
4.2 Kehitysympäristö

Sivuston kehitys tapahtui käyttäen Windows Server 2012-palvelinta, johon oli asennettu Sharepoint 2013. Sharepoint 2013-sivuston päälle oli luotu Metronic-pohjaa käyttäen sivusto, jota muokattiin vastaamana asiakkaan tarpeita.

Kehitys tapahtui etäyhteydellä Windowsin ”Remote Desktop Connection”-sovellusta käyttäen. Täten kehitysympäristö pysyi aina samana ja pystyin tekemään työtä missä tahansa. Koodieditorina käytettiin Notepad++-ohjelmaa.

4.3 Sivuston rakenne

Sivusto koostuu Sharepointin pohjan päälle asennetuista sivuista. Koska Sharepoint käyttää samaa sivupohjaa, joka sivulla oli sitä varten luotu Extranet-Page.master niminen tiedosto, jonka avulla sivuston pohja luotiin. Tälle pohjalle ladataan käyttäjän pyytämiä alasivuja. Sivusto koostuu siis alasivuista ja niitä vastaavista kontrollereista.



Kuvio 6. Sivuston etusivu

Sivusto lataa tarvittavat osat sivukohtaisesti käyttäen $ocLazyload-toimintoa. Koska kontrollerit ladataan sivukohtaisesti, sivun käyttö nopeutuu.

```
// Contacts
.state('contacts', {
    url: "/    /contacts.html",
    templateUrl: "../../_layouts/15/assets/views/epec/contacts.html",
    data: {pageTitle: 'Contacts'},
    controller: "ContactsController",
    resolve: {
        deps: ['$ocLazyLoad', function($ocLazyLoad) {
            return $ocLazyLoad.load([{
```

Jokaisella sivulla on oma kontrollerinsa, jonka lataamalla saadaan AngularJS antamat toiminnot sivulle. Koska sivulla on vain yksi moduuli, tapahtuu Javascript-koodit kontrollerikohtaisesti. Sivuston tiedot haetaan sivukohtaisesti käyttäen Sharepointin REST-rajapintaa.

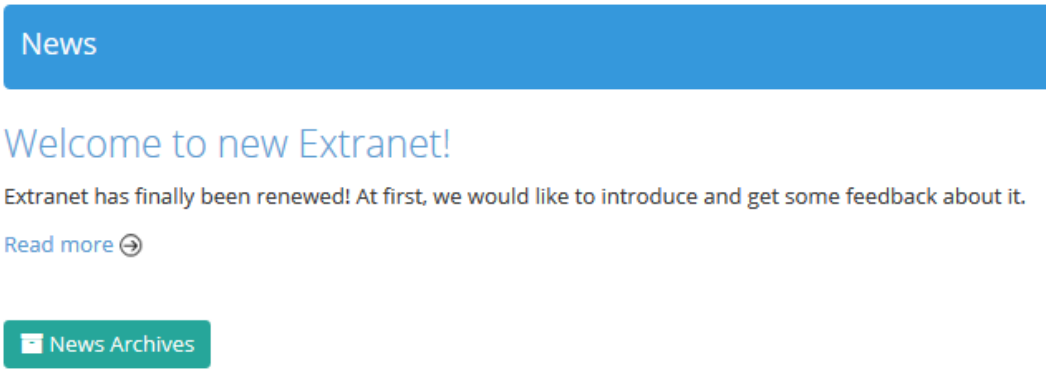
4.3.1 Käyttöliittymä

Sivuston käyttöliittymä toteutettiin käyttäen Visualwebin ostamaa Metronic-sivustopohjaa. Metronic tarjoaa valmiiksi monia toiminnallisuuksia, jotka helpottivat kehitystyötä. Metronicin valmiit teemat mahdollistivat siistin lopputuloksen. Käyttöliittymän tarkoituksena oli näyttää tarvittavat tiedot listoilta nopeasti ja helposti.

4.4 Etusivu

Ensimmäinen näkymä, joka ladataan, on Contacts.HTML-dokumentti. Tässä HTML-dokumentissa on uutiset, kymmenen uusinta tiedostoa ja Key Account Team-listaus. Key Account-listaus käydään tarkemmin läpi kappaleessa 4.7. Etusivun eri osiot on eritelty omiin kontrollereihin, jotka ladataan jokaista elementtiä varten erikseen.

Etusivun vasen sarake on tehty uutisia varten, jotka haetaan Sharepoint-palvelusta. Nämä uutiset on tallennettu listalle palvelimella.



Kuvio 7. Uutiset etusivulla

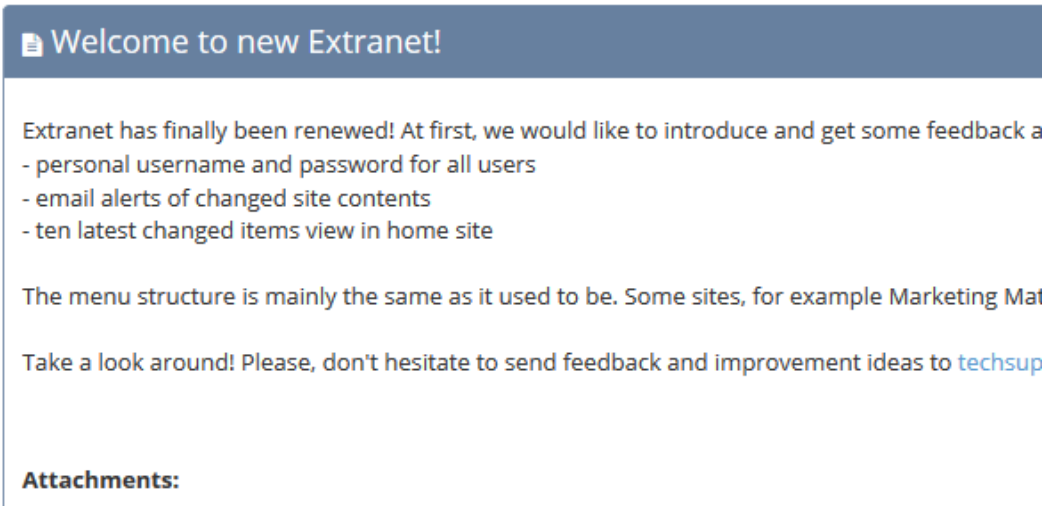
Tässä näkymässä näytetään kolme uusinta uutista. Tiedot tähän haetaan Sharepointista käyttäen kutsuja. Uutisissa voidaan myös listoilla määritellä niiden loppumispäivämäärä, jolloin uutinen lakkaa näkymästä etusivulla. Muuten kontrolleri on hyvin samanlainen muiden kontrollereiden kanssa.

```
$scope.newsFilter = function (document) {
return document.ItemPublishDate >= $scope.nyt;
}

$scope.newsEndFilter = function (document) {
return document.ItemPublishedEndDate <= $scope.nyt2;
}


$scope.init2 = function () {
    $scope.callService();
};
```

Painamalla Read More-kohtaa avautuu uutinen lukusivulle, josta voidaan lukea koko uutinen. Kun tätä Read More-nappia painetaan, haetaan uutinen käyttäen ListItemId-tietuetta.

```
<a ng-click="newsclick(document.ListItemId);" class="news-block-btn">
Read more <i class="m-icon-swapright m-icon-black"></i>
```

Tässä lukutilassa otetaan uutisen teksti kokonaan ja sidotaan sen mukana tulevat HTML-tagit myös dokumenttiin, jolloin rivin vaihdot ja vahvennukset tulevat näkymään.



Kuvio 8. Uutisten lukutila

Uutisten mahdolliset liitetiedostot haetaan tapauskohtaisesti käyttäen ”getbytitle” REST-hakuominaisuutta, jolloin uutiseen liittyvät liitetiedostot haetaan palvelimelta. Alla esimerkki liitetiedoston hakemisesta palvelimelta.

```
$scope.attachmentquery ="/docs/_api/web/lists/getbytitle('ExtranetNews')/items(" + NewsSelected
```

Uutisten Archive-listaus on yhdistelmä perinteistä dokumenttinäkymää, jossa lukutila toteutetaan käyttäen samaa toimintoa kuin tässä.

Etusivulla on ”Latest updated items”, jossa esitetään kymmenen uusinta tiedostoa Sharepoint-palvelusta. Atonista tulevia tietoja ei tässä näytetä. Koska taulussa ei tarvitse olla taulunhallintaan olevia Datatables-toimintoja, ei sitä ole käytetty tässä.

Latest updated items

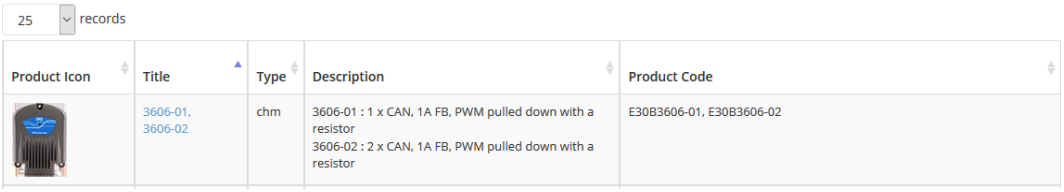
Title	Item Location	Modified
CANmoonLite 2.1.6.0	Downloads	2015.10.12
Newsletter 2/2015	Newsletters	2015.10.07
Newsletter 4_14	Newsletters	2015.09.24
Newsletter 4_13	Newsletters	2015.09.24
Newsletter 3/2012	Newsletters	2015.09.24
Newsletter 3/2011	Newsletters	2015.09.24

Kuvio 9. Etusivun ”viimeisimmät päivitetyt tiedostot”-listaus

4.5 Dokumenttinäkymät

Sivuston alasivuissa tärkein ominaisuus oli saada dokumenttikirjastoissa olevat tiedot näkymään helposti ja siististi. Tähän käytettiin Datatables-liitännäistä, jonka avulla tiedon esittäminen HTML <table>-elementteihin saatiin mukaan sivutus- ja hakutoiminnot taulukohtaisesti. Sivustolla olevat alasivut hakivat tiedot Sharepoint-palvelimilta dokumenttikirjastokohtaisesti ja jotkin sivustoista hakivat tietoa

myös asiakkaan muista palveluista. Sivustolla olevat table-elementit pitivät sisällään monia erilaisia tietoja ja tiedostoja.



Kuvio 10. Table-näkymä Technical Documents-alasivustosta

Määrittelemällä Datatables-muuttujia voidaan muokata varsinaista näkymää. Nämä muuttujat vaihtelevat sivustokohtaisesti ja suurimmat muutokset ovat ne miten listoissa olevat tiedot näytetään; esimerkiksi järjestys ja määrä. Alla on esimerkki näistä määrittelyistä.

```
    var vm = this;
vm.dtOptions = DTOptionsBuilder.newOptions()
    .withDisplayLength(25)
    .withOption('order', [1, 'asc'])
    .withOption('oLanguage', {"sEmptyTable": "Waiting for Data" })
    // Add Table tools compatibility
    .withTableTools('../../_layouts/15/assets/js/scripts/TableTools/swf/copy_csv_xls_pdf.swf')
    .withTableToolsButtons([
        'copy',
        'print', {
            'sExtends': 'collection',
            'sButtonText': 'Save',
            'aButtons': ['csv', 'xls', 'pdf']
        }
]);
```

4.5.1 Dokumenttinäkymien ohjaimet

Jokaiselle alasivulle on tehty oma kontrolleri, joka ladataan käyttämällä Lazyload-järjestelmää. Kontrollerit hakevat tiedot Sharepointista ja joillakin sivuilla myös asiakkaan toisesta tietojärjestelmästä Atonista. Aton on PHP-käyttöliittymä dokumenttien hallintaan. Tähän käytetään REST-kyselyitä. Käytettävät kyselyt on laitettu omiin muuttujiinsa käyttäen $scope-toimintoa. Koska Sharepointtiin menevät kyselyt tapahtuvat palvelimella, ei niiden määrittelyssä tarvitse määritellä palvelinta niin kuin Aton-kyselyissä. Alla on esimerkki $scope.query ja atonquery Accesories-kontrollerista.

```
$scope.atonquery = "https://extranet.    .fi:8443/awp/wsapi.php?cmd=wsalldocuments&wsid=

$scope.query = "/_api/search/query?querytext='ContentTypeId:0x0101007F904B7F780B1F4DBD76
```

Koska sivusto pyytää sivunlatauksen yhteydessä niitä osia joita sillä sivustolla tarvitaan, toteutetaan varsinaiset kutsut ”init”-tapahtuman aikana. Tämän jälkeen koodissa ajetaan ne kyselyt, mistä tietoa haetaan. Sharepoint-kyselyissä ovat tiedot useamman askeleen takana, jolloin pitää määritellä se mistä tieto haetaan. Määrittelemällä ”data.d.query.PrimaryQueryResult.RelevantResults.Table.Rows.results” voidaan rajata kyselystä tuleva tieto vain niille riveille josta tietoa löytyy. Aton-kyselyissä tätä ei tarvitse tehdä, koska tieto tulee valmiiksi json-muodossa. Tämän jälkeen, kun tieto on haettu, päivittyy se automaattisesti HTML-dokumenttiin.

```
$scope.callService = function () {
    $http({
        method: 'GET', url: $scope.atonquery + "&rowsperpage=0",
        headers: { "Accept": "application/json" }
    }).
    success(function (data, status, headers, config) {
        $scope.atondocuments = data;
    })
```

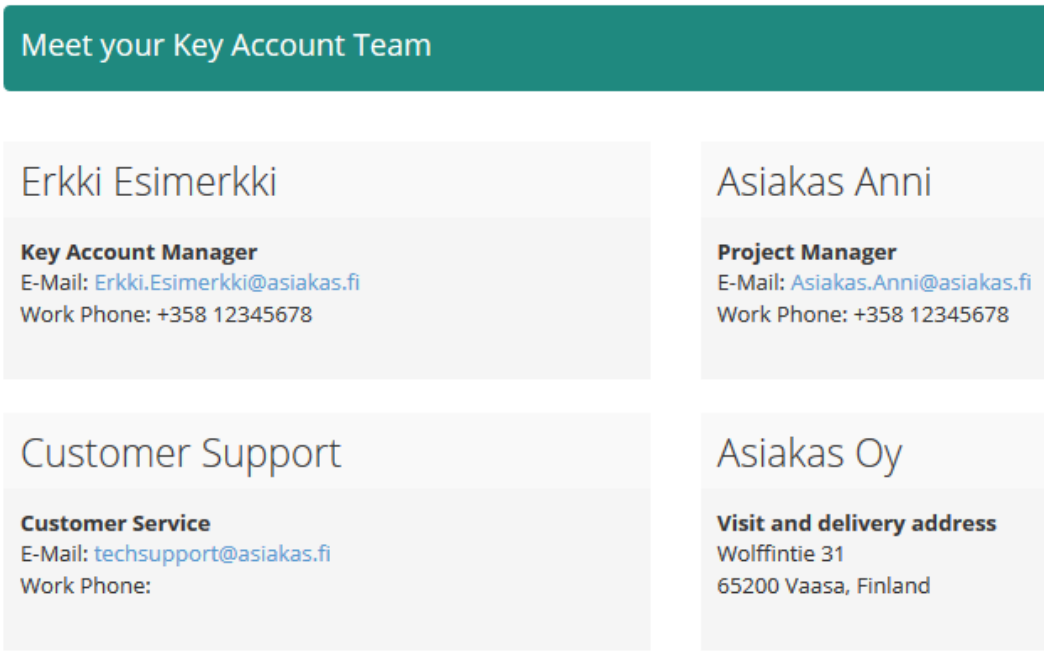
HTML-dokumentissa olevat tiedot on sidottu käyttäen kahden suuntaista tiedon sitomista, jolloin tauluun päivittyvä tieto on jatkuvasti päivittyvää. Vaikka sivustolla ei käytetä jatkuvasti päivittyviä tauluja, on tämä nopein tapa näyttää tietoa. Tieto sidotaan tauluun rivikohtaisesti käyttämällä AngularJS:n toistinta. Tällöin ensiksi tauluun sidotaan kaikki Atonista tulevat tiedot, jonka jälkeen rivikohtaisesti sidotaan Sharepoint-kyselyn tulokset. Tämä ei tapahdu yhtäaikaisesti palvelinten välisten viiveiden vuoksi. Alla on esimerkki toistimen sitomisesta table elementtiin.

```
<tr dt-rows="" ng-repeat="atondocument in atondocuments">

    <td>
        <a target="_blank" href="{{atondocument.link}}">{{atondocument.Desc2}}
    </td>
    <td>
        {{atondocument.Desc3}}
```

4.6 Käyttäjätiedot

Sivustolla on käyttäjän kirjautumistietojen perusteella vaihtuvia osioita, jotka vaihtelevat käyttäjätunnusten perusteella. Tämä vaikuttaa sivuston perustoimintoihin, jossa vain osa dokumenttikirjaston sisällöstä näytetään asiakkaalle sen tunnisteen perusteella, mistä asiakas on. Tämä toiminto toteutetaan käyttämällä Sharepointin valmista toimintoa, jolla voidaan rajata vain osa listan sisällöstä näkymään sille asiakastunnukselle.

Sivustolla on myös muita alueita, jotka vaihtelevat sen mukaan, mistä asiakas on. Etusivulla oleva ”Key Account Team” on tällainen toiminto, jossa vain sivuston omistajan yhteystiedot ovat staattisia ja muuten tiedot vaihtuvat käyttäjätiedon perusteella.

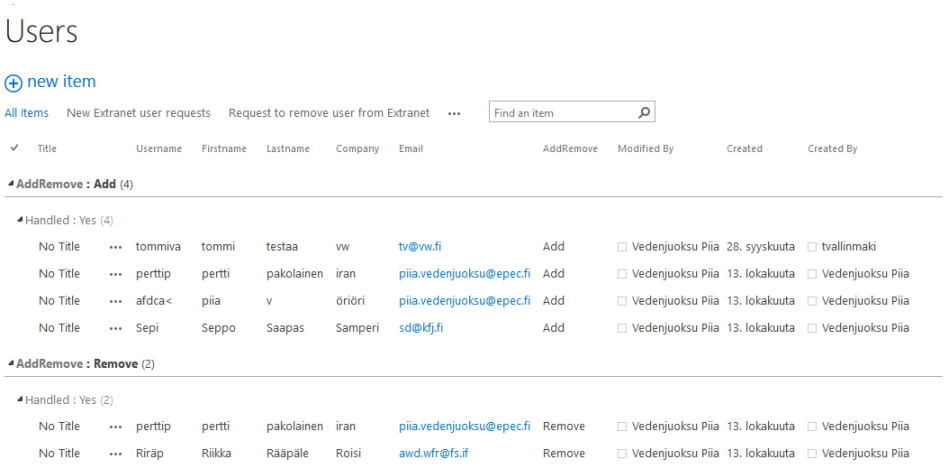


Kuvio 11. Key Account Team-näkymä

Toinen käyttäjätunnuksien mukaan muuttuva osa sivustossa on oikean puolen sivupalkki. Sivupalkki on piilossa, mutta saadaan esille painamalla nuolta. Tässä sivupalkissa on kaksi toimintoa, joista toinen päivittyy käyttäjätietojen perusteella mistä asiakas on. Käyttäjätiedot haetaan palvelimelta ja suodatetaan sen toimiston tai asiakkaan perusteella, mikä tieto sille tilille on määritetty. Palkkiin haetaan

kaikki käyttäjätiedot, joista suodatetaan pois ne, joilla ei ole samaa toimistoa ja ne käyttäjät, jotka on Sharepointin automaattisesti luotuja tilejä. Tällä tavoin asiakas näkee vain ne tilit, joiden käyttäjätiedot he pystyvät näkemään.

Toinen osa pitää sisällään myös lomakkeen, jonka avulla asiakas voi pyytää uusia tunnuksia. Tämä kysely lähetetään REST-kyselyllä palvelimelle käyttäen POST-metodia, joka tallennetaan palvelimella sijaitsevaan listaan. Tätä listaa käyttäen voi sivuston hallinnoija nähdä mitkä asiakkaat tarvitsevat uusia tunnuksia.



Kuvio 12. Näkymä sivustolla olevasta listasta

Määrittelemällä onko kyseessä lisäys vai poisto, voidaan myös pyytää käyttäjätunnuksien poistoa palvelimelta. Tunnusten tekemisessä on validointi, jonka avulla tarkastetaan onko kaikki tiedot annettu. Tähän käytetään AngularJS:n omaa validointi-ominaisuutta.

```
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', function () {


        var siteUrl = "/Epec/";
        var clientContext2 = new SP.ClientContext(siteUrl);
        var web = clientContext2.get_web();
        var list = web.get_lists().getByTitle('Users');
        var tietoListItemInfo = new SP.ListItemCreationInformation();
        var listItem = list.addItem(tietoListItemInfo);
        listItem.set_item('Username', c.Username);
        listItem.set_item('Firstname', c.Firstname);
        listItem.set_item('Lastname', c.Lastname);
        listItem.set_item('Company', c.Company);
        listItem.set_item('Email', c.Email);
        listItem.set_item('AddRemove', AR);
        listItem.update();
        clientContext2.executeQueryAsync();

        $scope.syotto = { Username: "" , Firstname: "", Lastname: "", Company: "", Email: "" };
        AR = "Add";



    })});


$scope.requestClick = function () {
            $scope.requestSent=true;
     $timeout(function () { $scope.requestSent = true; }, 3000);
      $timeout(function () { $scope.requestSent = false; }, 6000);
};
```

4.7 Palvelut

Sivustolla on käytetty vain muutamia palveluita. Nämä AngularJS-palvelut ovat käytössä yleisesti ohjelman sisällä ja pitävät sisällään suodattamia ja aikatoimintoja.

4.7.1 Suodattimet

Sivustolla on muutama itse tehty suodatin, jotka on tehty sivustolla näytettävien tietojen siistimiseen ja muokkaamiseen. Nämä suodattimet muokkaavat esimerkiksi päivämääristä väliviivat pisteiksi. Alla on esimerkki viivojen korvaamisesta pisteillä.

```
MetronicApp.filter('viivat', function () {
  return function (input) {
    if (input) {
      return input.replace(/-/g, '.');
    }
  };
});
```

Sivustolla olevissa tauluissa on tietoja, joiden aakkostaminen on haastavaa niiden tietosisällön vuoksi. Tätä varten on sivustolla suodattimia, jotka poistavat tietueesta numerot tai kirjaimet. Tämän jälkeen tieto lajitellaan ensiksi kirjainten perusteella ja sitten numeroiden perusteella. Alla on esimerkki numeroiden poistamisesta tietueesta.

```
MetronicApp.filter('PoistaTeksti', function () {
    return function (text) {
        if (text == '' || text == null)
        {
            text == "";

        }
        else {


        text = text.replace(/[^0-9]/g, '');

        return text;
        }

    };
});
```

Nämä tiedot eivät näy HTML-dokumentissa, koska ne on piilotettu käyttämällä Datatables ”columnDefs”-metodia.

```
        vm.dtColumnDefs = [
            DTColumnDefBuilder.newColumnDef(1).notVisible(),
            DTColumnDefBuilder.newColumnDef(2).notVisible()
    ];
```

4.7.2 Aikatoiminnot

Yleisin aikatoiminto, jota sivustolla käytetään on MomentJS:än avulla toteutettu ”new”-lippu uusien rivien vieressä. Tämä vertaa julkaisupäivämäärää dokumentista nykyiseen päivämäärään. Jos se on alle 30 päivää, tulee näkymiin ”new”-lippu. Alla esimerkki tästä toiminnosta ja sen HTML-elementtiin sitomisesta.

```
$scope.paiva = moment().subtract(30, 'days').format('YYYY-MM-DD');
```

```
ng-show="document.ItemPublishDate >= paiva" class="badge badge-roundless badge-danger">New</span>
```

Sivustolla on käytetty myös paljon $timeout-komentoa. Osa Sharepoint-kyselyistä tarvitsee tietoa käyttäjästä ja koska koko AngularJS-koodi ajetaan yhtäaikaisesti, olisi palvelimelta tuleva tieto myöhässä, jolloin tietoa ei saada. Näitä tietoja ovat esimerkiksi Office-tieto, jota käytetään oikean sivupalkin tiedon suodattamiseen. Tämä tieto haetaan käyttäjäkohtaisesti palvelimelta olevalta listalta. Tällä tavoin viivästetty tieto ei näy loppukäyttäjälle varsinaisesti viiveenä ja kaikki tarvittava tieto tulee näkyviin varmasti riippumatta palvelimen viiveestä. Alla on esimerkki toiminnosta, joka viivästyttää tiedon hakua 2 sekuntia (2000 ms).

```
$timeout(function(){
    $http({
    method: 'GET',
    url: "/_api/SP.UserProfiles.PeopleManager/getuserprofileprope
    headers: { "Accept": "application/json;odata=verbose"  }
}).
success(function (data, status, headers, config) {
    $scope.homeOffice = data.d;
}).
error(function (data, status, headers, config) {

});

    }, 2000)
```

5 YHTEENVETO

Projektin tarkoituksena oli rakentaa nopea käyttöliittymä asiakkaan Extranet-tarpeisiin käyttäen Sharepointin päälle rakennettua AngularJS-ratkaisua. Opinnäytetyössä olen käynyt läpi ne ominaisuudet Sharepointista ja AngularJS:stä, joita työssä käytin. Tämä on vain pieni osa sitä laajaa kokonaisuutta, joita nämä Sharepoint-alustana ja AngularJS-ohjelmistokehyksenä tarjoavat. Tämä työ tarjoaa ne perusteet, joiden avulla voidaan rakentaa yksinkertaiset ja nopeat, mutta varsin kattavat Extranet-sivut Sharepoint-ympäristöön. Kaikki esimerkit, joita työssäni käytin, ovat sivustolla käytössä ja niiden toiminnallisuus on todettu hyväksi.

AngularJS kehittyy jatkuvasti, joten on hyvä huomata, että vaikka sivuston ratkaisut ovat toimivia, voidaan ne toteuttaa eri tavalla, kuin sen miten ne tässä projektissa toteutettiin. Huomasin opinnäytetyötä tehdessäni, että esimerkiksi AJAX-hakujen tekeminen on mahdollista käyttäen paljon tehokkaampaa $resource-palvelua $http-palvelun sijaan. On siis olemassa tehokkaampia keinoja toteuttaa ratkaisuja kuin nämä mitä työssäni käytin. Tähän asiaan tulee vaikuttamaan myös tuleva AngularJS 2.0-julkaisu, jonka mukana tulee myös Microsoftin täysi tuki tälle ohjelmistokehykselle, koska he ovat olleet sitä mukana kehittämässä.

Opinnäytetyö oli aiheena haastava ja mielenkiintoinen. Opettelin AngularJS-ohjelmistokehyksen käytön projektia varten. Työssä ongelmia aiheutti erinäköisten versioiden vaihtuvuus AngularJS-versioiden välillä ja se, mitkä toiminnot voidaan tehdä eri tavalla kuin uusimissa versioissa. Tämä aiheutti lisätutkimusta työtä tehdessä. En mainitse näistä vanhentuneista teknologioista mitään paitsi tämän $resource-vaihtoehdon AJAX-kyselyissä. Sharepoint-osio oli helpompi toteuttaa Visualweb Oy:n vankan osaamisen vuoksi, josta sain tarvittavat neuvot ja ohjeet näihin ongelmiin.

Tekemällä sivustolle uusi sivustopohja käyttämällä AngularJS- ja REST-ominaisuuksia, on mahdollista rakentaa käytettävämpi sivusto sen sijaan, kuin että käytettäisiin Sharepointin valmista sivustorakennetta.

Sivusto on ollut aktiivisesti asiakkaan käytössä jo jonkin aikaa ja tulokset ovat olleet positiivisia. Projekti onnistui hyvin koska sain uutta osaamista joka mahdollistaa paremman työllisyyden jatkossa.

Jatkossa kehitykseen tulisi ottaa mahdollisuus tehdä sivustoista yhden sivun ratkaisuja, jolloin vain sivuston tieto vaihtuisi ja muuten rakenne pysyisi samana. Tällöin ei tarvitsisi määritellä erikseen jokaista haettavaa tietoa vaan haettaisiin kaikki tieto, jonka asiakas voisi itse suodattaa näyttämään ne tiedot joita haluttaisiin esittää.

LÄHTEET

AngularJS.org 2015a. Developer guide: What is Module?. Viitattu 20.11.2015. https://docs.AngularJS.org/guide/module

AngularJS.org 2015b. Developer guide: angular.module Viitattu 20.11.2015. https://docs.AngularJS.org/api/ng/function/angular.module

AngularJS.org 2015c. Developer Guide: Data Binding.. Viitattu 20.11.2015 https://docs.AngularJS.org/guide/databinding

AngularJS.org 2015d. Developer Guide: Understanding Controllers Viitattu 20.11.2015. https://docs.AngularJS.org/guide/controller

AngularJS.org 2015e. Developer guide: Depedency Injection. Viitattu 20.11.2015. https://docs.AngularJS.org/guide/di

AngularJS.org 2015f. Developer guide: Filters. Viitattu 20.11.2015. https://docs.AngularJS.org/guide/filter

AngularJS.org 2015g. Developer Guide: orderBy. Viitattu 20.11.2015. https://docs.AngularJS.org/api/ng/filter/orderBy

AngularJS.org 2015h. Developer Guide:What are Scopes?. Viitattu 20.11.2015. https://docs.AngularJS.org/guide/scope

AngularJS.org 2015j. Developer Guide: ng-repeat. Viitattu 20.11.2015. https://docs.AngularJS.org/api/ng/directive/ngRepeat

AngularJS.org 2015k. Developer Guide: ng-init . Viitattu 20.11.2015. https://docs.AngularJS.org/api/ng/directive/ngInit

AngularJS.org 2015l. Developer Guide: ng-click. Viitattu 20.11.2015. https://docs.AngularJS.org/api/ng/directive/ngClick

AngularJS.org 2015m. Developer Guide: $timeout . Viitattu 20.11.2015. https://docs.AngularJS.org/api/ng/service/$timeout

AngularJS.org 2015n. Developer Guide: $interval . Viitattu 20.11.2015. https://docs.AngularJS.org/api/ng/service/$interval

AngularJS.org 2015o. Developer Guide: $http. Viitattu 20.11.2015. https://docs.AngularJS.org/api/ng/service/$http

AngularJS.org 2015p. Developer Guide. $resource . Viitattu 20.11.2015. https://docs.AngularJS.org/api/ngResource/service/$resource

Angular Datatables 2015a. API. Viitattu 20.11.2015. https://l-lin.github.io/angular-datatables/#/api

Angular Datatables 2015b. WithTableTools. Viitattu 20.11.2015. https://l-lin.github.io/angular-datatables/#/withTableTools

Angular Datatables 2015c. WithOptions. Viitattu 20.11.2015. https://l-lin.github.io/angular-datatables/#/withOptions

Green B, Seshadri S. 2013. AngularJS . O'Reilly.

jQuery 2015. Datatables Manual. Viitattu 20.11.2015 https://datatables.net/manual/index

Green B, Seshadri S. 2013. AngularJS . O'Reilly.

Microsoft Dev Center 2015a. REST API Overview. Viitattu 20.11.2015 https://msdn.microsoft.com/en-us/library/office/jj163876.aspx

Microsoft Dev Center 2015b. ListCollection.GetByTitle method.Viitattu 20.11.2015 https://msdn.microsoft.com/en-us/library/microsoft.sharepoint.client.listcollection.getbytitle.aspx

Microsoft Dev Center 2015c. User Profiles REST API Reference. Viitattu 20.11.2015 https://msdn.microsoft.com/en-us/library/office/dn790354.aspx

MomentJS Documents. Viitattu 28.11.2015 http://momentjs.com/

Mozilla.org 2015. Mozilla Developer Network : WebAPIs XMLHttpRequest. Viitattu 20.11.2015. https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

Roine J,Anttila J 2015. Sharepoint Hyvät Pahat ja Rumat. Sharepoint Office 365 HPR.