HAAGA-HELIA
University of Applied Sciences

Improving social collaboration possibilities by implementing a Social App for SharePoint 2013

Niko Lönnberg

13 November 2014

HAAGA-HELIA
University of Applied Sciences

Abstract

Date of presentation of thesis

Business Information Technology

Authors Niko Lönnberg	**Group** X
The title of your thesis Improving social collaboration possibilities by implementing a Social App for SharePoint 2013	**Number of pages and appendices** 42 + 2
Supervisors Juha Pispa, Niina Kinnunen	

This document will show how and why an App is implemented on top of SharePoint 2013.

The document will give a brief insight to what SharePoint 2013 is and used for. The document will present the historical background and the evolution that has led to the current functionalities of SharePoint 2013.

This document will reflect on why the social features are important in a modern intranet and what the app aims to do to improve them.

The document will show how to set up the development environments for developing with a connection to Microsoft Azure, the Microsoft cloud hosted service platform, or on premises development. The document will reflect on the advantages and disadvantages of both development models.

The logical and desired layout of the app is explained and ways of achieving the desired effects will be included in the document

The implementation section will cover how to access SharePoint with the app, explaining in detail the use of the built in artefacts and how they are accessed. The document will show what files the app solution consists of and how the different files in the solution are related to the final product and deployment. The document will demonstrate with code snippets and screen screenshots the main methods and give explanations on what is achieved with them. A brief troubleshooting section is included as a reference for the usual and critical errors that were encountered.

The document will reflect upon the outcome and the further development possibilities of the product.

In the summary part of the document the greatest obstacles for the project and suggestions for improvements will be enclosed.

Key words
JavaScript Object Model, App, SharePoint 2013, Social collaboration, Visual Studio

Table of contents

1 Introduction

“And between what you've seen in Windows 8, what you've seen in the new Windows 8 devices and are still to see, and what you will see in Office 15, this will be the best year ever, ever, ever to be a Microsoft partner.” (Balmer 2012)

In 2012 Microsoft announced that, in forth coming year most of the Microsoft products are going to be renewed. Announcements were made about Windows 8, Office15, Windows Server 2012, Azure, and Office 365. The emphasis was of course on Windows 8 with the Internet Explorer 10 but, it also meant the beginning of cloud services and the start of Platform as a Service (PaaS) era. Microsoft renewed all of its major products and was ready to take them to cloud.
With the PaaS solution end users get the most benefits from their applications as access has become the priority.

The reason why the service is offered from cloud is reliability and in some extent control. Compared to on-premises deployment Cloud is less likely to have system-down-time and other flaws, the service is available in chunks, so customers only pay for what they need and now also as a result of actions by the American Government it is very secure. In short Office 365 is easy to use with various devices and it’s available securely for the end users.

There are some not so well known challenges however. The development and administration for such applications has become harder as the old-fashion back-end development is no longer possible. With on-premises deployed solutions there is always the possibility to investigate the functionalities using tools like reflector, or if the database entries and commands need to be checked they can be opened with SQL Server Studio. The resources of the applications are visible and easy to use if a developer wants to make a customization or otherwise utilize the existing resources of a Microsoft application. The drawback with cloud deployed applications, such as Office 365, these are obscured. Finally, SharePoint online is maintained automatically by Microsoft. This can bring new unwanted features or remove old, liked, features literally over night as well as change the styles and other User Interface (UI) elements. (Korhonen 2014)

To make the resources of SharePoint and other web applications, which are deployed to cloud, accessible to the developers and administrators Microsoft had to come up with a new way of allowing these resources to be tapped. With no back-end coding possible the SharePoint resources had to be available from the front-end. This project aims at finding out how to develop an application using only these front-end resources.

1.1 SharePoint Intranet and competitors

SharePoint is designed mainly as an intranet/extranet platform although, if SharePoint 2013 licence is purchased for intranet, SharePoint 2013 can be used freely as internet platform for the enterprise.
The modern Intranet could be described with a couple of sentences. Employees need to be able to work anywhere without compromising security, in short, having a digital workspace. Documents and other file types need to be managed and shared properly, all kinds of news and other information is distributed as needed, and it is the company wide social media.
(Del Monte 2014, Matthews 2014)
Not all intranet software provide all these features but SharePoint 2013 comes with them all. The main competition have one more major disadvantage, they are not being a part of the Microsoft Office family, SharePoint is a part of the Office package thus interaction with Microsoft Excel and Microsoft Word and the rest of the Microsoft Office family are at the core of its functionalities.
A recent blog, based on a larger research, lists the following trending features to be of notice in intranets:

- Carousels, all award winning intranets have useful carousels that save space and give improved visual functionality. Some criticism exists on the usefulness of the carousel but with responsive pages and smaller devices they are invaluable.
- Persistent right rails, quick access to all intranet data via clear structural navigation
- Functional footers, with added navigation there is less need of scrolling to the main navigational elements, especially useful on responsive sites viewed on smaller devices.
- Local search, the better the scope for search the lighter it is on the system

- Mega menus with shrinking displays, where all needed tools can be added and used but without interference to the normal navigation and usability
- Filmstrip, embedded viewing experience for most important content and links
- The Metro look, boxier, flat and app like functionalities with minimal UI and live tiles. The Metro look is more than just looks, it is the way things are made to work. Content of the application is displayed in the tile instead of some Chrome and motion is made a part of the displayed content. (Turnbull 2013)
- Social, the intranets are becoming more like social networks

(Caya, Nielsen, Pernice & Schad 2014)

SharePoint is offering most of the list OOTB (Out Of The Box), making it highly competitive as the base for an intranet.
The main alternatives for SharePoint are Drupal, Joomla!, Liferay, Lotus Notes, IBM WebSphere and SAP Net Weaver Portal, however SharePoint has an estimated market share of more than 50%, making the competition solutions chosen because of more specific needs of the customer.

1.2 SharePoint history in brief

To understand how SharePoint evolved to be the intranet typified, a brief history summary will enlighten the matter. In 1996 Microsoft Site Server was released for easier management of all needed functionalities of a complex application, these included content management, authorization, indexing and search, document management and analytics. The features were not neatly wrapped as one product but were more like separate tools to choose from. In 1998 Microsoft announced Exchange 2000 Server with a new added element code named “Tahoe” which introduced a new technology called WebDAV. It introduced document handling with version control and authoring as well as better search/indexing features. In 1999 Microsoft wrapped this basis of their new technology as their first portal framework calling it Digital Dashboard. It also introduced content management from different sources in form of “Nuggets” which would later evolve in to Web Parts, the main content management feature of SharePoint.

Microsoft was running two database platforms side by side but in 2000 SQL Server was chosen as the further developed platform while Exchange Server evolved to be the modern mail server and contact management platform running on the SQL Server. The first product named SharePoint was announced, the SharePoint Portal Server 2001. The new product which included the Digital Dashboard was not supported by the core development platform by Microsoft, Visual Studio, so there was limited support possibilities. It was aimed at the growing portal market but did not perform as well as it should. In 2001 Microsoft also acquired the content management vendor nCompass and renamed it Microsoft Content Management Server 2001 (MCMS 2001). On top of these two Microsoft had also released an add on called SharePoint Team Services (STS) to Office as a team collaboration feature.

In 2002 the next version of Microsoft Content Management Server was released, the major difference was ASP.NET front end and thus better support from Visual Studio. The next year a new Microsoft Office 2003 was released along with a new server and the next generation of SharePoint. SharePoint Team Services was renamed Windows SharePoint Services (WSS) and the main SharePoint content management system was named SharePoint Portal Server 2003 (SPS 2003). The features remained pretty much the same as before but performed better.

It became pretty clear that the document management and portal features needed to be incorporated and with SharePoint 2005 Content Management Server was included into Windows SharePoint Services, this was a huge step forward for SharePoint, but yet Windows SharePoint Services lacked synchronization and other important features. The most notable improvement in the upgrade that became available in 2005 was the inclusion of Web Parts and workflows. The workflows really improved content and document management by adding approvals, comments and other features to the publishing process. In previous versions this was possible with heavy customisations which couldn’t be migrated to the newer versions. Because of this some the heavily customized MCMS portals are still in use today.
The last SharePoint with separate WSS was the 2007 SharePoint, Microsoft Office SharePoint Server (MOSS). In MOSS the content is now displayed mostly by web parts instead of regular .aspx pages. With the help of master pages the style- and content hierarchy are easily manageable and navigation

editing tools are available OOTB. MOSS still posted restrictions to the user by having some functionalities that only worked with Internet Explorer, such as document sharing and editing online.

In 2010 SharePoint had absorbed WSS and was divided in to three levels, the free one was called SharePoint Foundation and SharePoint Server could be bought on top of it. The full-scale platform was SharePoint Enterprise. The interface was changed drastically and support for browsers other than the current Internet Explorer was added. The API: s were introduced and Client Side Object Model (CSOM) was taken as optional approach to the traditional SharePoint API (SharePoint Server Object Model). (Crown Canyon Systems 2014, Khartikeyan 2013)

1.3 SharePoint 2013

The latest SharePoint version was introduced in 2012 and made commercially available in 2013. It boasted all the previous features and huge improvements to search and collaboration features. The social features are in a huge role in the new SharePoint. Users have by default their own MySite site where they can post blogs, follow people or documents and manage favourite contents when working in an authenticated mode. The documents do no longer need to be uploaded from the UI, drag and drop features are making the UX (User Experience) even better. The biggest difference though, was that the built in features are now apps, all libraries and lists and even sites are in displayed as apps.

The designing of pages has been made a lot more comprehensive, it is possible for end users to create style packages with their own master pages and deploy them across the web application. Further styling of the sites can for the first time be done by editing images which are already deployed, the size and dimensions can be pre-set or the images can even be cropped and filtered to fit the visual needs perfectly.
To enhance the possibility to create the perfect look, SharePoint 2013 comes with pre-set view states for mobile platforms such as tablets. The default viewing device and browser types can be defined to fit organizational needs.
There is now Search Engine Optimization (SEO) OOTB. The feature can be configured site specifically by the site administrator or owner.

1.4 The apps approach

What can the apps development model be like compared to the traditional solution development model? Is there a large contrast between these development models or can previous knowledge of the classic development model be used when developing apps?

Apps are not only significant parts in SharePoint but also in all the new Windows 8 and Office 15 Microsoft products. Windows 8 is running all features as apps this is including the desktop. Apps are downloaded from the App Store or installed straight to the Office program or operating system.

Figure 1 The App Store icon in Windows 8

Apps are not only deployed on the SharePoint host but can also execute in browsers or external web servers as cloud service.

Although these apps can now be developed using almost any programming language, the fact is that, SharePoint is a Microsoft product and thus the best available development platform is Visual Studio. The resources available for developing an app are in such extent available for Visual Studio and the new development tool called Napa, which is free and purpose specific for developing these apps.

Taking the apps approach thus makes the software much easier to discover, purchase and install, than other forms of retail software.

There is minimal or no need for marketing an app or being online sales services as the App Store provides for both.

The app has minimal effect on future upgrades of the platform, which is a huge bonus considering the problems faced with upgrading heavily customized solutions. With the flexibility of the app it can be deployed to a new platform and use the same hosting server when using the provider hosted deployment model.

For a developer who has little or no experience in developing SharePoint specific solutions the app development is much easier, as the programming can be done with more familiar languages and SharePoint internal information can be accessed through JavaScript. Additionally the app can use remote applications that can be built on any platform stack. These applications could be built using LAMP (Linux, Apache, MySQL, PHP) stack and displayed through an app IFrame window in an app-part. The apps are also able to use external, non-Microsoft platform stack, databases and storages increasing versatility.

The App uses SharePoint components, including customized parts such as content types and event handlers, even external event handlers can be used. The app cannot however use all SharePoint components making app incapable to do some functionalities that traditional solutions can.

The app can run on host machines or remote servers, the only place from the app cannot run is the SharePoint server itself, making the SharePoint server less vulnerable and the app easier to sell to potential customers who are concerned with safety of their application. (Microsoft 2014a)

2 The project staging

When this project was started SharePoint 2013 had just come out to the market and the first customer solution projects had just begun. The information relating to customization of SharePoint 2013 was scarse as most online-manuals that were published were focused on configuration through the UI. There were no books published, and still most of the books that could have been used as sources to this project are programming language related more than SharePoint related.

Fortunately SharePoint is, according to various surveys, the most used intranet platform in the Fortune 500 companies so there was genuine interest in the development community for building customized solutions, apps being an important new part of the solutions package (Sisler 2012).
Microsoft responded to the needs by offering a 30 day trial (extendable to 90 days) Office 365 Developer Site, a site that has been preconfigured for SharePoint apps development. In April 2014 the offering is still 30 days but for subscribers to MSDN (Microsoft Developer Network) there is a one year free subscription available. (Microsoft 2014b)

As SharePoint 2013 was published the software vendors had to get as much information about the new product as possible, in the shortest possible time. Conferences on SharePoint 2013 possibilities where held alongside with smaller workshops for Architects and Project Managers, everything that needed to be known had to be known fast. In the hectic environment some smaller issues were easier to explore through trial. Small projects were started and discarded alike, to learn more about the flexibility and possibilities SharePoint 2013 had to offer.

2.1 The Goal

One of the improved features in SharePoint 2013 compared to the previous versions are the social features. There was an expressed interest on how these social features could be tapped to full extent to allow better collaboration between employees by an app using the built in resources SharePoint. As a recent study shows happy workers are more productive (Oswald, Proto &

Sgroi 2014). Could an app improve on the happiness and productiveness of a worker? If co-workers appreciate an employee’s effort and can share that thought through the social hub it could increase the happiness of the company.

The goal was set to see if the social features can be improved with endorsing, praising and other Social Media kind of features with some gamification included. This was to be done by an app called the 9 to 5 app, symbolizing the work hours. Other specification was that deployment and use should not be cumbersome, the user’s credentials should be sufficient for the app to work. Furthermore the app should use only client side API: s for communicating with SharePoint resources. The outcome would be an easy to use interface with three different layouts to choose from, the personal view, endorsement view and admin view. As the project was started the ambitious goal on the views were to populate empty divs, a division or a section mark of the Hyper Text mark-up Language (HTML), with JavaScript according to the view.

2.2 Available resources

Soon after the release of SharePoint 2013 blogs on small projects and experiments and hints on connections and configurations started appearing, most were obviously small scale, but some larger tutorials were also published. At first there was scarce official documentation on the subject but over the course of two years the Microsoft MSDN documentation has grown quite substantially. The printed media is limited to Programming in HTML5 with JavaScript and CSS3 by Glenn Johnson, a book aimed at taking Microsoft Certification in programming for SharePoint. This book was published a year after initial development begun, and thus not available in the initial stages.

2.2.1 Distinguishing good resources

Distinguishing good resources is easiest done by debugging. There are now quite good references in the MSDN documentation, showing clearly the structure and properties of the social client class, but the only really good method is to implement the code and see if that does what is wanted. Many of the blogs were extremely helpful on troubleshooting issues that started immediately when setting up implementation environments and continued having excellent advices when error messages were encountered. The most important criteria for a good source was not the source itself but asking the right question, when the knowledge base of the technology is at novice level getting help with errors is fundamental. Because the author has familiarized himself to the SharePoint environment during this project, a good source is not necessarily giving the exact information but has specific information that points the author in the right direction.

2.3 Criteria of a successful project

For the project to be successful it must meet a few essential criteria. Firstly an App which uses JavaScript to connect to SharePoint resources and is able to add social functionalities for users is ready for use in a SharePoint 2013 environment.

Secondly, the resources that were used are well documented and the work is detailed and analysed.

Finally, the Thesis is well formed and has definite way points of the project.

3 Preparation for Implementation

There are a few things that must be taken into consideration when developing an app for SharePoint 2013. These choices have a huge impact on what technology should be used when developing the app as well as on the method of data transfer between SharePoint and the App.
The following questions should act as a basis when determining the development- and deployment methods of the app.

Where will the app be hosted? The implementation methods depend on choice of hosting, and where the app will be deployed. Further thought must be taken on if the users are navigating to the app web or will the app be launched from navigation or be displayed in an app part.
What is the scope of the app? The most basic question of what the app will do.
Will the App available through the App Store and can it be a universally deployable app?
What are the privileges the app must have to run? Will the app manipulate data in SharePoint or need to create artefacts and also where the app will do the manipulation, the permissions of SharePoint 2013 are fine grain and must be specified with precision.

When starting to break these points down the parameters for development clarify. The hosting will effectively dictate which programming languages are available for the app along with resources that can be used by the app. The hosting also set limitations on how the app can be deployed and thus the UX. (Microsoft 2014c)

There are four different API: s (Application Programming Interface) that can be chosen from depending on the approach and preference. These API: s limit the choices of the hosting if specific techniques are desired. These are the choices available:

.Net Framework client object model with LINQ (Language Integrated Query) syntax. The LINQ syntax allows for SQL-like queries to be made against SharePoint objects, such as lists and sites. When using **.Net Framework COM** SharePoint hosting is Autohosted.

Silverlight client object model gives more possibilities than .Net Framework COM and still allows the use of LINQ. With the code behind available this type of app is able to manipulate timer jobs and deploy features making it a lot more versatile than the other types but also needs to place code in the Global Assembly Cache (GAC) which makes it more hazardous to deploy than the client side apps.

JavaScript client object model (JSOM) offers the SharePoint functionalities as JavaScript libraries that can be accessed from the client side. This type is called SharePoint hosted App.

REST (Repetitional State Transfer**) endpoints** that can be accessed by any client. **REST** was initially described in context of HTTP but it is not as limited as a protocol, in addition of URI: s and other protocol like transfer objects, it has **Request Methods.** This is usually a Provider Hosted app, but also SharePoint hosted apps use REST.
(Microsoft 2014d)

3.1 Hosting the App

An Autohosted app are deployed to cloud where the hosting is in Azure and Azure SQL. The coding is done with traditional .aspx page in front and c# code in the code behind, for this purpose multitenancy is provided. There is no need for an app web, although it can be implemented with one. The possibility to deploy without the App Web makes it easier for customers to accept an app solution, after all the need for a custom web for app purposes only is not always appealing. The app is imported to the app gallery and deployed to App parts which resemble web parts in many aspects. The app can be deployed to any page layout through the app part. On May 16th 2014 Microsoft abruptly announced that the autohosted app development will be shut down starting June 30th (Microsoft 2014e).

The SharePoint-hosted app utilizes the CSOM (Client Side Object Model) which means the app runs in browser and calls SharePoint JavaScript libraries for SharePoint related information. This type of app is built from a minimum of three elements, the HTML5 page, the JavaScript code and the style sheet

CSS3. It is possible to include things such as SharePoint list definitions or libraries in the app definition.

Provider-hosted apps are installed within the organisations network on their own servers. The applications themselves do not have to be programmed using Microsoft Specific programming languages or even hosted on Microsoft SQL Servers. These apps will be relayed to SharePoint using App-Parts which are actual IFrames for the app to be displayed in. As the connection to SharePoint is done via JavaScript Object Model or JavaScript Object Notation (JSON) APIs, the code is universally deployable as long as the hosting is deemed secure. (Microsoft 2014c)

3.2 APIs

SharePoint 2013 has large number of different APIs. In web development APIs usually mean web service. The API should be chosen according to the skillsets of the implementer and as the APIs are in some cases overlapping so the implementation can be done in more than one way.

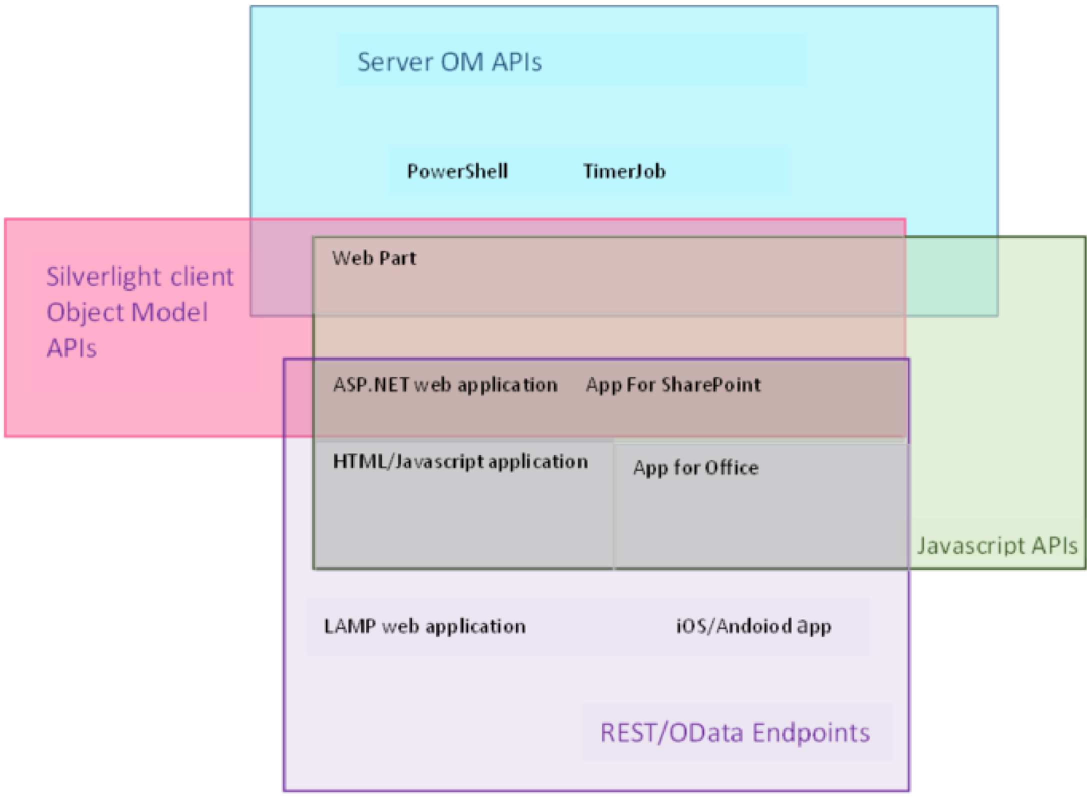


Figure 2 Overlapping APIs and choices on implementation (Microsoft 2013d)

The Server sandboxed solutions was the default way of development in previous SharePoint generations but has been depreciated in SharePoint 2013. Still the .NET framework and Silverlight object models have most features available to them. JavaScript may be used to utilize the resources available for server side code with REST web service with almost all API: s. The call is made an the answer is sent back either as JavaScript Object Notation (JSON) or Atom which is based on XML, JSON being the choice with AJAX applications. The Following table displays what features are available for different APIs.

Table 1 "Features of API: s" (Microsoft 2013d)

Feature	**.NET Framework or Silverlight object models**	**JavaScript object model**	**REST/OData endpoints called from a Windows platform or JavaScript**
Object-oriented programming	Yes	Yes	No
Batch processing	Yes	Yes	No
APIs for conditional processing and exception handling	Yes	No	No
Availability of LINQ syntax	Yes	No	No
Combining list data from different SharePoint web applications	Yes	No	Yes
Leveraging jQuery, Knockout, and other JavaScript libraries	No	Yes	No, from Windows platform Yes, from JavaScript

(Microsoft 2013d)

3.3 The Logic for the App

The final stage before the programming phase is to understand how the project could be implemented and what are the SharePoint resources that can be used. SharePoint can be used as a database as custom lists are available and are in fact stored in content databases on SQL servers.
The App should inspire people to do more socially within the boundaries of SharePoint. The way it would do it was by allowing endorsements and rewarding achievements for users, the latter is called Gamification.
The App itself should be a list of a User’s skills and endorsements along with a visual score of Social Activity.

The List of Endorsements should be a global list because the display should mimic LinkedIn, with one skill and many endorsements. The only reasonable way to implement this would be a hidden list in the root of the application. The endorser would be the “Owner” of the item created and it allowing for CRUD permissions. The App must have Full Control for the web so it can create and update the list. The Endorsee should be able to see objects concerning the Social Actor itself. The App is displayed in an app part on a MySite, the logic demands the use of MySite if it needs to be kept relatively simple. MySite URL has both the domain and username in the URL and context has the users information, allowing for the app with a simple check to determine whether this is a MySite of someone else than the user and enabling the Endorse button and listing the skillsets of someone else other than the user. The more cumbersome method would be to implement a People Picker and have the App available in other sites, like a workspace.
Later the project had to revert to people picker because the URL rewrites may make the URL unusable.
The Microsoft Enterprise Social Application Yammer has built in the ability to praise someone for a job well done. This is something the app will also do, because it is possible to post to a social feed using REST API. There should be a button and text field implemented for this purpose, in the text field the context of the praise is described and the post would always start with “User A PRAISED User B” and then the description. The Praise is rendered to both users’’ feeds and viewable by subscribers of both feeds.

Unfortunately because it is impossible to write to the user attributes with the app, therefore the number of praises cannot be used as a social marker. This could however be easily implemented with a feature with code behind and an event receiver as server side object model has far greater permissions to the attributes.(Microsoft 2013f)

The Gamification should use the SP.Social namespaces attributes for reference calculating a score based on social activities. The App should give hint son how to perform better in the social network of the application. This is implemented by getting the social feed of the user and scoring point son blogs, likes, tags etc. and informing the user on the possibility of doing these things. The Score leads to merits or achievements, i.e. a novice could be called a

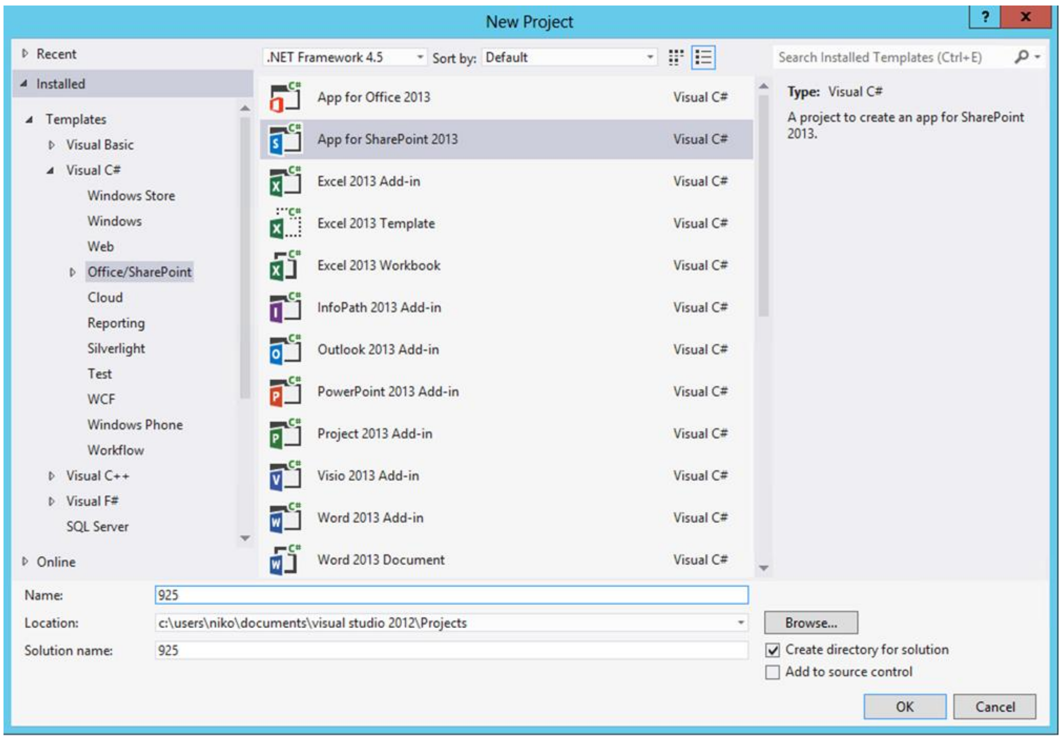
“Dormant blogger” while a person with more blogs a “Rising Star of Information”. More gamification could be added with providing a three star rating of the current user along with changing transparent background colour for the app. The highest merit could be for example three stars and a golden background.

4 Implementation

As this project was started the clear need was for a JavaScript App for SharePoint, ignoring the choice of API which would be best suited for the job because the knowledge on implementing with the Server Object Model existed already. The clear goal was to see how Apps could be used instead of code behind web parts. When a project is set up, there has to be a connection to a working SharePoint 2013 environment. This can be either Office 365 or on premises SharePoint 2013.

4.1 Developing to cloud

Without a connection to SharePoint the development cannot be debugged, making the development nearly impossible. The are two different ways to start the development of any SharePoint app. First is to develop to cloud by signing up for Office 365 developer site (Microsoft 2014b, Fox 2013).
This way the developer does not need to setup a development environment and does not have to have an extreme computer to host SharePoint 2013.
The only procedures that need to be taken are setting up Visual Studio for SharePoint App development. It was initially chosen as the development method, as this seemed the to be the obvious way to develop mirroring the growing numbers of Office 365 based solutions.
The steps needed for setting up the development environment are listed in the appendix.



App for Office 2013	Visual C#
App for SharePoint 2013	Visual C#
Excel 2013 Add-in	Visual C#
Excel 2013 Template	Visual C#
Excel 2013 Workbook	Visual C#
InfoPath 2013 Add-in	Visual C#
Outlook 2013 Add-in	Visual C#
PowerPoint 2013 Add-in	Visual C#
Project 2013 Add-in	Visual C#
Visio 2013 Add-in	Visual C#
Word 2013 Add-in	Visual C#
Word 2013 Document	Visual C#

Type: Visual C#

A project to create an app for SharePoint 2013.

Name: 925

Location: c:\users\niko\documents\visual studio 2012\Projects

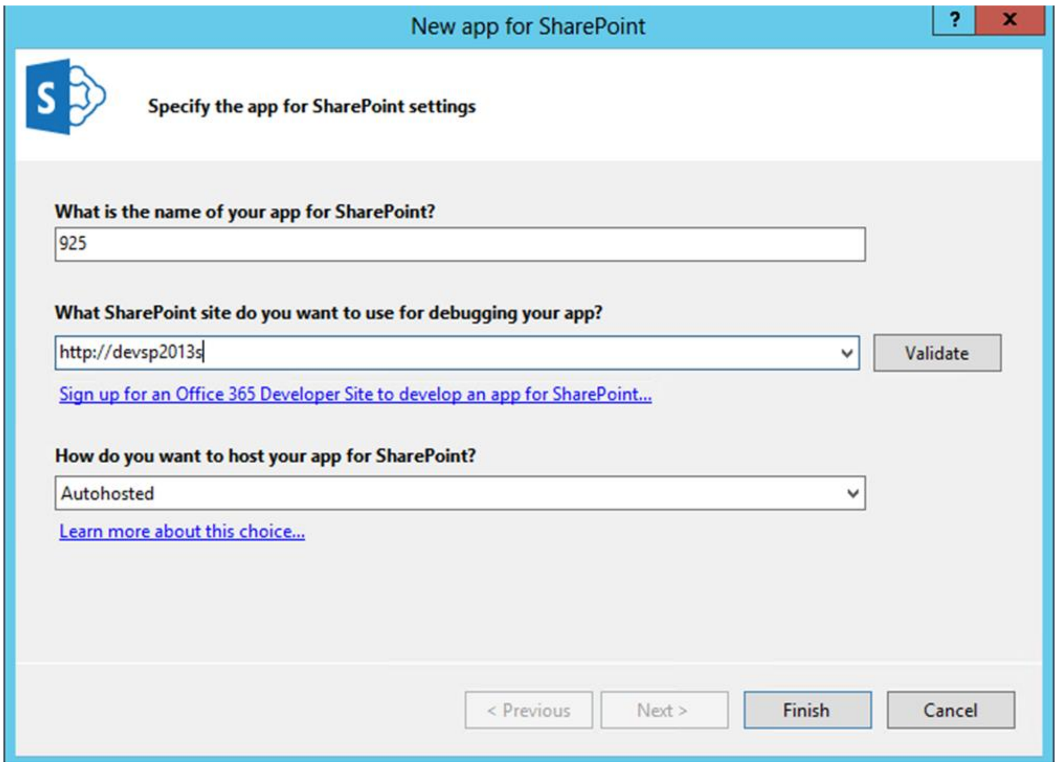
Solution name: 925

[✓] Create directory for solution

[ ] Add to source control

Figure 3 Initial choice of app Project creation

When the project is started the deployment-url is demanded.


New app for SharePoint
Validate
Sign up for an Office 365 Developer Site to develop an app for SharePoint...
Learn more about this choice...
< Previous
Next >
Finish
Cancel

Specify the app for SharePoint settings

What is the name of your app for SharePoint?

925

What SharePoint site do you want to use for debugging your app?

http://devsp2013s

How do you want to host your app for SharePoint?

Autohosted

Figure 4 Connecting to hosting environment

When developing to 365 Developer site the Domain is sharepoint.com, the site used at the beginning of the project was: "https://atconsultants.sharepoint.com/sites/at925/"

The URL is checked and permissions asked before the development can start. When everything is done correctly the deployment and retraction is done by building the project and then debugging it, Visual Studio uploads the project to the site and starts the default browser at the app site. It is even possible to make changes while debugging by just rebuilding the project at the go.

4.1.1 Limitations

The greatest limitation when developing to cloud/Azure is the lack of relevant tools and permissions. Firstly the complete lack of the scripting tool PowerShell is definitely a big problem. On premises the tool can be used to check status of attributes and the internal names of metadata etc. basically anything can be checked and changed with PowerShell. The other great limitation is time. You can only develop for a certain amount of time for free and when developing for social actors there must be social actors present in SharePoint thus increasing the costs. Before the app was finished the site was discontinued because of costs and the fact that the company for which the app was developed had purchased laptops that were capable of running SharePoint 2013.

4.2 Development on premises

When the app development was re-launched the development was transferred from cloud/azure to on premises solution. This meant that SharePoint that will host the development site for the app needs to be configured for App Development. One more reason not to start from scratch was made on the grounds that for a Social App, there has to be social activity and actual users who can test the functionalities and give feedback. The chosen environment is a SharePoint which is used as a Demo platform for Nintex workflows and other CRM related customizations. Unfortunately the Site Collection was not set up for apps development so the first step in the 3 week sprint was to set up the environment for the development. This solution for developing on premises is also perfectly suitable for virtual machines on personal computers, the only prerequisite being that SharePoint 2013 is installed and running, and for that is needed approximately 200GB of free disk space and 32GB of RAM which SharePoint 2013 needs to operate.

Firstly the correct services needed to be started from SharePoint Central Administration.

The Windows Server 2012 uses apps, so it is to open any known app by just typing the first letters in tile view:

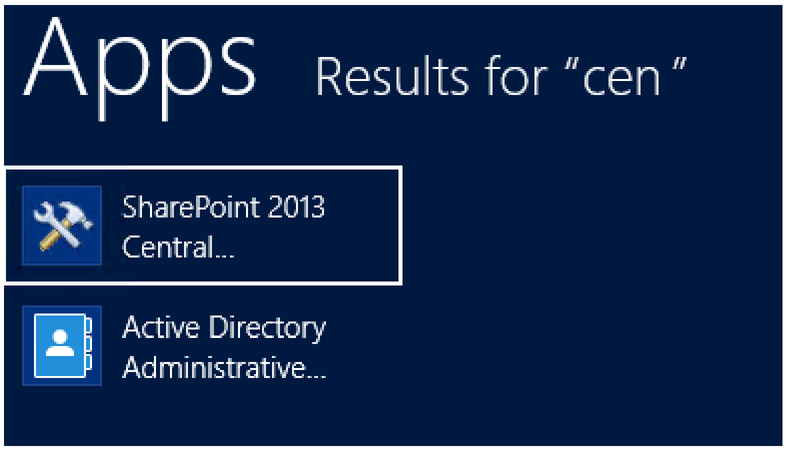

Apps Results for "cen"
SharePoint 2013 Central...
Active Directory Administrative...

Figure 5 Everything is an App

In Central Admin the following steps are made:

Click Application Management

Manage Service applications

Start the following Services if they are stopped

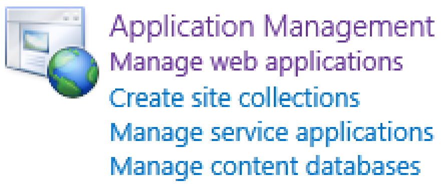
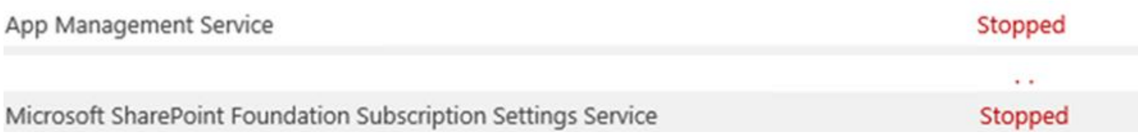


Figure 6 Central Administration Web Application management menu

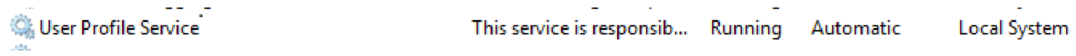


App Management Service Stopped

Microsoft SharePoint Foundation Subscription Settings Service Stopped

Figure 7 The Services which need to be started

After these are started go to services on server and check that the user profile service is also started. This should be the case as it is automatically on.

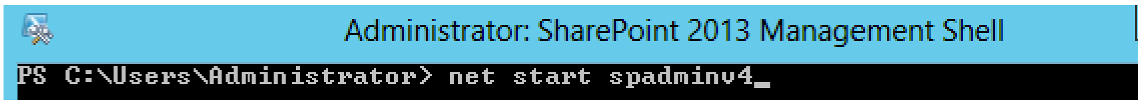


User Profile Service This service is responsib... Running Automatic Local System

Figure 8 Check that the User Profile Service is running

After this the Isolated app domain must be created, the creation can only be done by using PowerShell and the permissions of the Farm Account.

First the admin and timer services need to be started just to ensure that they are running



Administrator: SharePoint 2013 Management Shell

```
PS C:\Users\Administrator> net start spadminv4_
```

Figure 9 Run the SharePoint Management Shell as Administrator

The command should result in: The requested service has already been started. Do the command net start also for sptimerv4.

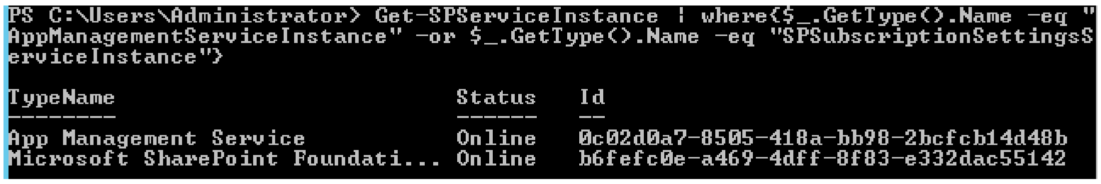
Next set the app domain with Set-SPAppDomain command, in this project the command looked like Set-SPAppDomain http://devsp2013.contoso.com/

Check that the services are running by running these commands Get-SPServiceInstance | where{$_.GetType().Name -eq “AppManagement-ServiceInstance” | Start-SPServiceInstance . Do the command also for “SPSubscriptionSettingsServiceInstance”.

The |(pipe) character invokes a method and $ (dollar) character is used to mark a variable.

The line is read as follows: Get the instance where the following is true name == AppManagementServiceInstance, and if found start the service.

The Subscription service and AppManagement services also need to be checked



```
PS C:\Users\Administrator> Get-SPServiceInstance | where{$_.GetType().Name -eq "
AppManagementServiceInstance" -or $_.GetType().Name -eq "SPSubscriptionSettingsS
erviceInstance"}

TypeName                         Status   Id
--------                         ------   --
App Management Service           Online   0c02d0a7-8505-418a-bb98-2bcfcb14d48b
Microsoft SharePoint Foundati... Online   b6fefc0e-a469-4dff-8f83-e332dac55142
```

Figure 10 App Management Service and Subscription Services are running

The main thing in the returning information is that the Status is “Online” for both.

When these checks have been done the application pool, database settings and account for AppManagementInstance and SPSubscriptionService can be defined with code that is provided in MSDN. In the PowerShell code below the red highlight is a variable and can be named as wished and the green highlight is the actual Managed Account that was used when the app development domain was created.

```
$account = New-SPManagedAccount
$account = Get-SPManagedAccount "contoso0\SP_PortalAppPool"
$appPoolSubSvc = New-SPServiceApplicationPool -Name SettingsServiceAppPool -Account $account
$appPoolAppSvc = New-SPServiceApplicationPool -Name AppServiceAppPool -Account $account
$appSubSvc = New-SPSubscriptionSettingsServiceApplication –ApplicationPool $appPoolSubSvc –Name SettingsServiceApp –Database-
Name SettingsServiceDB
$proxySubSvc = New-SPSubscriptionSettingsServiceApplicationProxy –ServiceApplication $appSubSvc
$appAppSvc = New-SPAppManagementServiceApplication -ApplicationPool $appPoolAppSvc -Name AppServiceApp -DatabaseName AppS-
erviceDB
$proxyAppSvc = New-SPAppManagementServiceApplicationProxy -ServiceApplication $appAppSvc
Set-SPAppSiteSubscriptionName -Name "app" -Confirm:$false
```

After these steps everything should be ready for deploying apps to the development environment. The way to test this by starting the app project which should now be deployed to your destination app development site if it is already configured. (Microsoft 2013c)

If there are problems with deploying the app after the management service application has been successfully created the SharePoint environment must be configured to use apps from the apps domain.

4.2.1 Configuring the environment for apps

To able user to use apps in an environment the following steps must also be taken:

1. Create forward lookup zone for apps in DNS by right clickin Forward lookup zone and choosing new zone and keep given values until the app domain name is asked. Choose a name for the zone, apps was used in the project.
2. Create a new CNAME alias for the new zone by right clicking on the newly created forward lookup zone and select New Alias. Browse until the server name is found and double click that, double click Forward Lookup zone, double click domain, Select same as parent folder and OK, OK.
3. Create the app domain as the Forward Lookup zone was created but fill *.app as Alias name before continuing to the double clicking sequence

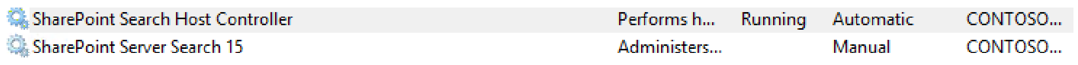
(Microsoft 2014d, van Olst 2012)

4.2.2 Investigative development

When developing for Apps for SharePoint 2013 there is a flood of unexpected events ranging from permission issues to other unexpected failures. In this part this document will go through some events that raised issues throughout the environment setup and programing phases in the order in which they impacted the project.

First issue was insufficient permissions when trying to stage the Subscription Settings Service Application with PowerShell. The Program returned Farm is unavailable error even when given Site Collection administrator privileges to the account in Central Admin and Local Server administrator privileges on the server. The privileges that were missing were for configuration database not SharePoint, adding db_owner and db_securityadmin roles to the account solved the issue and script could be run without errors.

Second issue was when trying to start the program for the first time in debug mode in on premises environment running on the virtual machine. The Program returned error “SharePoint” and as I wrote an alert message the program was trying to return an object but somehow could not do it correctly. This is due to insufficient memory to run Noderunner.exe and when checking the memory usage it was at a steady 94% even if there were no major programs running. The solution is to shut down some services that are not needed, in this case there is no need to run search on development farm where there is only the development site. This is done from Services and disabling the following services:



SharePoint Search Host Controller	Performs h...	Running	Automatic	CONTOSO...
SharePoint Server Search 15	Administers...		Manual	CONTOSO...

Figure 11 Services that are unnecessary can be shut down in development environments. If in a live environment these must not be touched

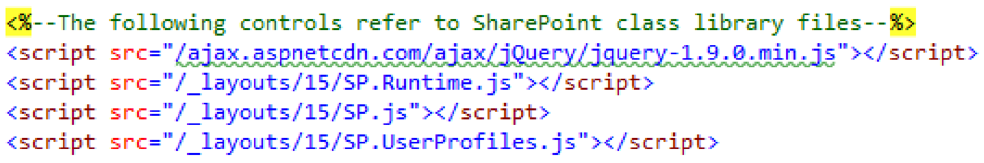
After shutting down the services the Memory was down to 75% usage, which proved to be insufficient still, so more memory had to be assigned for the virtual machine. The problem still presisted so some further error handlin was done to load the script only after some code had been transferred. Finally the solution to the problem was forcing the context before running any other script:

```
SP.SOD.executeFunc("SP.js", "SP.ClientContext", function (){ Code Here}
```

Third issue was losing the current context again. The error message stated that
0x800a138f - JavaScript runtime error: Unable to get property 'SPClientPeoplePicker_InitStandaloneControlWrapper' of undefined or null reference. This issue never got solved because of hard disk failure which wiped to computer clean and the project had to be restarted.

4.3 Connecting to SharePoint

As the JavaScript Object Model was the chosen implementation, a brief insight on how it connects to SharePoint is needed.
Visual Studio 2012 caters for the needed end points when starting an App project. The first four lines of the Default.aspx page of the App contains registerings of SharePoint Dynamic Link Libraries (.dll) which are deployed to the GAC which is found in %systemdrive%\Windows\assembly. This is what gives the App access to the SharePoint resources.
Furthermore the JavaScript libraries are also embedded in SharePoint and registered in the Default.aspx page. This means that the App does not have to include these libraries but can call the by Uniform Resource Identifier (URI) components, basically by adding them to the URL in the browser.



```
<%--The following controls refer to SharePoint class library files--%>
<script src="/ajax.aspnetcdn.com/ajax/jQuery/jquery-1.9.0.min.js"></script>
<script src="/_layouts/15/SP.Runtime.js"></script>
<script src="/_layouts/15/SP.js"></script>
<script src="/_layouts/15/SP.UserProfiles.js"></script>
```

Figure 12 SharePoint class library controls

4.4 SharePoint hosted libraries

SharePoint JavaScript object libraries are the asset which all code must relate to. The App uses mostly the sp.userprofiles.js library from where the SPSocial namespace and SPUserProfiles namespace can be accessed. These namespaces contain Objects which must be first retrieved and the object has members; Constructor, Methods and Properties. Each of these have their unique set of retrieval and posting values that must be in correct form. The Visual Studio autocomplete function does not know these getters and setters, which in turn complicate the coding as constant lookup for the correct Methods and properties must be researched.

4.5 Programming waypoints and resources available

Implementing user specific controls starts with acquiring the user data. This is done by calling the Sp.UserProfiles.Js and with the Context retrieving the SP.UserProfiles.PeopleManager object. After this the personal properties can be extracted with a call to the people manager object with specific user parameters domain\user.

```
personProperties = peopleManager.getPropertiesFor(targetUser);
```

If the call is successful the personProperties object can be used in the following manner:

```
personProperties.get_displayName();
personProperties.get_userProfileProperties()['Department'];
```

The displayName can be changed to any of the following attributes:

Table 2 Attributes of SP.UserProfiles.peoplemanager Object

Attribute	Outcome
accountName	Gets the user's account name.
directReports	Gets the account names of the user's direct reports.
displayName	Gets the user's display name.
email	Gets the user's email address.
extendedManagers	Gets the account names of the user's manager hierarchy.
extendedReports	Gets the account names of the user's extended reports.
isFollowed	Gets a Boolean value that indicates whether the user is being followed by the current user.
latestPost	Gets the user's latest microblog post.
peers	Gets the account names of the user's peers.
personalUrl	Gets the absolute URL of the user's personal site.
pictureUrl	Gets the URL of the user's profile picture.
title	Gets the user's title.
userProfileProperties	Gets user profile properties for the user.
userUrl	Gets the URL of the user's profile page.

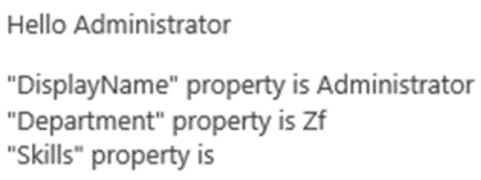
And the Key `['Department']` can be changed to any of the following

Table 3 Additional properties of userProfileProperties Attribute

AboutMe	SPS-AdjustHijriDays	SPS-LastKeywordAdded	SPS-ResourceSID
AccountName	SPS-AltCalendarType	SPS-Locale	SPS-Responsibility
ADGuid	SPS-Birthday	SPS-Location	SPS-SavedAccountName
Assistant	SPS-CalendarType	SPS-MasterAccountName	SPS-SavedSID
CellPhone	SPS-ClaimID	SPS-MemberOf	SPS-School
Department	SPS-ClaimProviderID	SPS-MUILanguages	SPS-ShowWeeks
EduExternalSyncState	SPS-ClaimProviderType	SPS-MySiteUpgrade	SPS-SipAddress
EduOAuthTokenProviders	SPS-ContentLanguages	SPS-O15FirstRunExperienc	SPS-Skills
EduPersonalSiteState	SPS-DataSource	SPS-ObjectExists	SPS-SourceObjectDN
EduUserRole	SPS-Department	SPS-OWAUrl	SPS-StatusNotes
Fax	SPS-DisplayOrder	SPS-PastProjects	SPS-Time24
FirstName	SPS-DistinguishedName	SPS-Peers	SPS-TimeZone
HomePhone	SPS-DontSuggestList	SPS-PersonalSiteCapabilitie	SPS-UserPrincipalName
LastName	SPS-Dotted-line	SPS-PersonalSiteInstantiati	SPS-WorkDayEndHour
Manager	SPS-EmailOptin	SPS-PhoneticDisplayName	SPS-WorkDayStartHour
Office	SPS-FeedIdentifier	SPS-PhoneticFirstName	SPS-WorkDays
PersonalSpace	SPS-FirstDayOfWeek	SPS-PhoneticLastName	Title
PictureURL	SPS-FirstWeekOfYear	SPS-PrivacyActivity	UserName
PreferredName	SPS-HashTags	SPS-PrivacyPeople	UserProfile_GUID
PublicSiteRedirect	SPS-HireDate	SPS-ProxyAddresses	WebSite
QuickLinks	SPS-Interests	SPS-RegionalSettings-Follo	WorkEmail
SID	SPS-JobTitle	SPS-RegionalSettings-Initia	WorkPhone
SISUserId	SPS-LastColleagueAdded	SPS-ResourceAccountName	

The output can be assigned to a variable to display if it has value

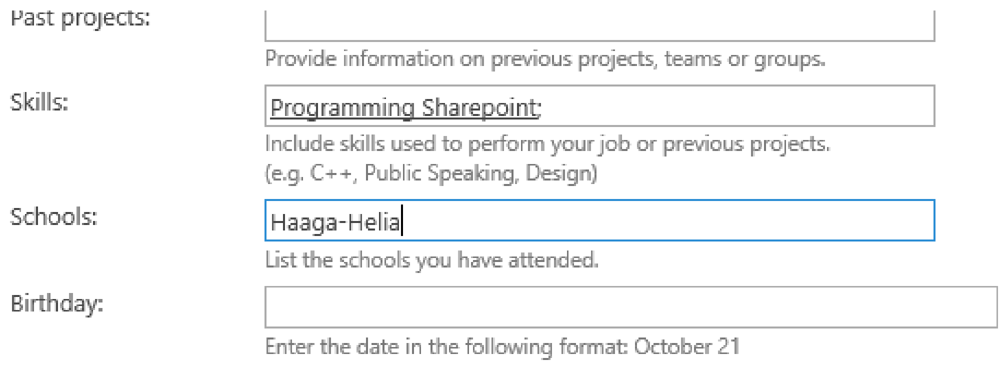
```
// Get a property directly from the PersonProperties object.
            var message = " \"DisplayName\" property is "
        + personProperties.get_displayName();

// Get a property from the UserProfileProperties property.
            message += "<br />\"Department\" property is "
        + personProperties.get_userProfileProperties()['Department'];

// Get another property from the UserProfileProperties property.
            message += "<br />\"Skills\" property is "
        + personProperties.get_userProfileProperties()['SPS-Skills'];
            $get("results").innerHTML = message;
```


Hello Administrator

"DisplayName" property is Administrator
"Department" property is Zf
"Skills" property is

Figure 13 results span after code is run

By adding values in Sharepoint Central Administration where all properties can be edited or in MySite where some properties can be edited by the user:



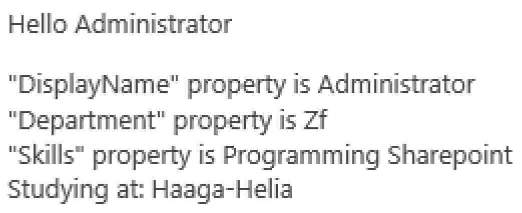
Past projects:	
	Provide information on previous projects, teams or groups.
Skills:	Programming Sharepoint;
	Include skills used to perform your job or previous projects. (e.g. C++, Public Speaking, Design)
Schools:	Haaga-Helia
	List the schools you have attended.
Birthday:	
	Enter the date in the following format: October 21

Figure 14 Central admin view of managing user profiles

When the property Schools has been assigned the schools value is retrievable.

```
message += "<br />Studying at: "
        + personProperties.get_userProfileProperties()['SPS-School'];
```

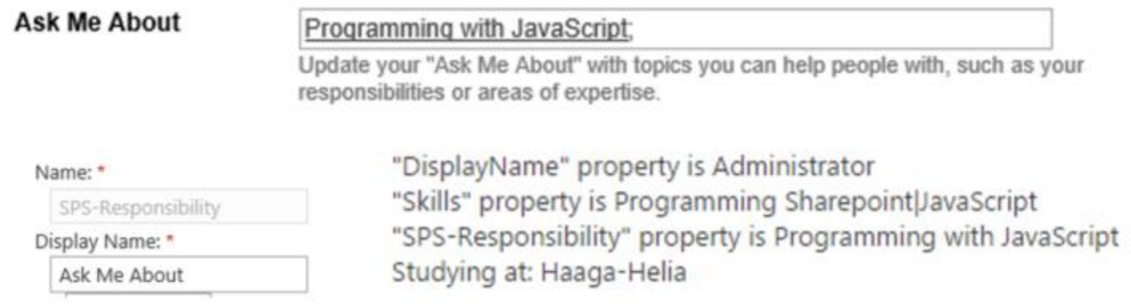
The outcome shows that there was immediate effect.


Hello Administrator

"DisplayName" property is Administrator
"Department" property is Zf
"Skills" property is Programming Sharepoint
Studying at: Haaga-Helia

Figure 15 results span displaying changes when code is run

However these are built in Get-Only Properties, the code cannot write to these properties or add new ones. With the only interesting property “Skills” we can fortunately append all skills separately as the return string uses |(pipe) as the separator. Furthermore after researching the person profile properties from Central Administration => Manage Service Applications => User Profile Service Application => Manage User Profile Properties, we can also use SPS-Responsibility. It is available for all users in their personal MySites.


Ask Me About

Programming with JavaScript;

Update your "Ask Me About" with topics you can help people with, such as your responsibilities or areas of expertise.

Name: *

SPS-Responsibility

Display Name: *

Ask Me About

"DisplayName" property is Administrator
"Skills" property is Programming Sharepoint|JavaScript
"SPS-Responsibility" property is Programming with JavaScript
Studying at: Haaga-Helia

Figure 16 The results of changes shown when code was run

Now the app has found these two different Skillsets to use as a base for the Endorsement list
The PeopleManager is not the only feed about users available in SharePoint, the feed that could have more use for the App is the SocialFeedManager. From SocialFeedManager the following Methods are available

Table 4 SocialFeedManager Methods

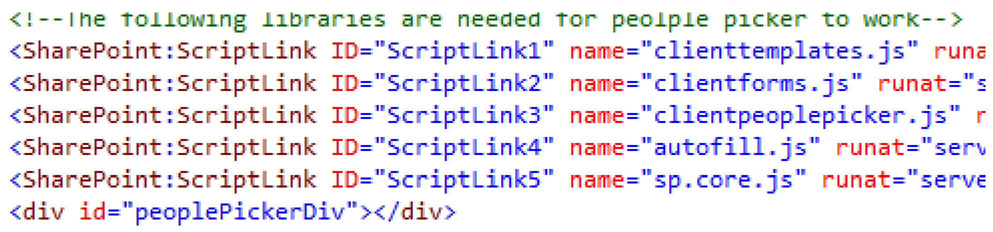
Method	Description
follow	Adds the specified actor to the current user's list of followed actors.
getFollowed	Retrieves the actors that the current user is following.
getFollowedCount	Retrieves a count of actors who the current user is following.
getFollowers	Retrieves the users who are following the current user.
getSuggestions	Returns Users who the current user might want to follow, as an array of SocialActor objects.
isFollowed	Determines whether the current user is following the specified actor.
stopFollowing	Removes the specified actor from the current user's list of followed actors.

4.5.1 Implementing People Picker

The App was initially supposed to get the users and domain from URL parameters of a MySite, but the idea was discarded. Instead the user can search for any user to look at their social rankings and endorse them. The initial pick will of a user will render the users Skillset to a custom control where the skills and endorsers of the skills. The SPClientPeoplePicker has the following components

- An input text box to enter the query
- A span control that shows the names, groups and claims
- A hidden element which is populated by the results
- An autofill control

The picker is dependent on server side JavaScript libraries that must be registered before the actual JavaScript code can be executed, these codes are executed after initial page load which means that the picker must first be rendered before the calls can be made.



```
<!--The following libraries are needed for peolple picker to work-->
<SharePoint:ScriptLink ID="ScriptLink1" name="clienttemplates.js" runa
<SharePoint:ScriptLink ID="ScriptLink2" name="clientforms.js" runat="s
<SharePoint:ScriptLink ID="ScriptLink3" name="clientpeoplepicker.js" r
<SharePoint:ScriptLink ID="ScriptLink4" name="autofill.js" runat="serv
<SharePoint:ScriptLink ID="ScriptLink5" name="sp.core.js" runat="serve
<div id="peoplePickerDiv"></div>
```

Figure 17 Linked JavaScript libraries to apprehend by the people picker

From the App.js code the div is initiated:

```
initializePeoplePicker('peoplePickerDiv');
```

The initialization of the picker is based on a Schema in the App.js file

```
            var schema = {};
            schema['PrincipalAccountType'] = 'User'; // Changed
            schema['SearchPrincipalSource'] = 15;
            schema['ResolvePrincipalSource'] = 15;
            schema['AllowMultipleValues'] = false; // Changed
            schema['MaximumEntitySuggestions'] = 50;
            schema['Width'] = '280px'; // CSS file will overwrite this
```

The Microsoft documentation does not say anything about the properties in the schema. Initially the picker takes multiple values and returns everything, the form asked to enter names or addresses, not a name and address. So by using people picker script libraries the look is kept OOTB. By removing unwanted Group data and setting AllowMultipleValues to false the schema is working for this purpose, and the picker is now nicely asking for one name or e-mail address to look for.

Then the schema is passed to the control:

```
this.SPClientPeoplePicker_InitStandaloneControlWrapper(peoplePickerElementId, null, schema);
```

Now the picker is ready for use, and auto populates the search box with AJAX elements, so the whole user name or e-mail address does not have to be typed. The call is initiated by a input.

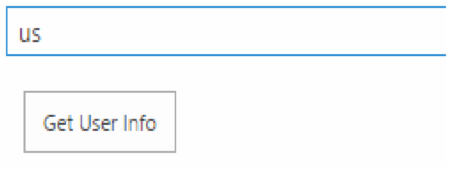


Figure 18 Start inserting name

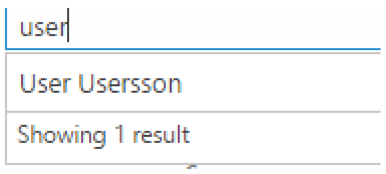


Figure 19 The AJAX will start suggestions

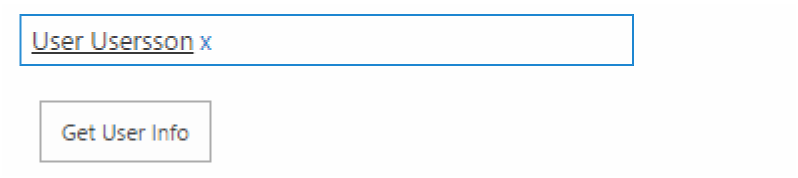


Figure 20 A SharePoint user profile was chosen

When the desired user has been selected the App.js retrieves the PeoplePicker object from the page:

```
var peoplePicker = this.SPClientPeoplePicker.SPClientPeoplePick-
erDict.peoplePickerDiv_TopSpan;
```

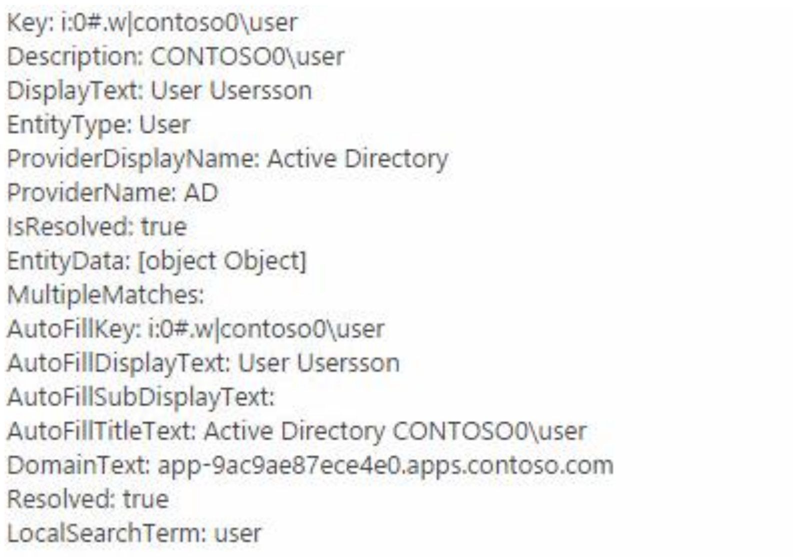
And from people picker the user information can be extracted with

```
peoplePicker.GetAllUserInfo();
```

The GetAllUserInfo returns a lot of unnecessary values for our purpose. The one thing that is of interest is the user Key which can be passed as Social Actor. It is retrieved with

```
keys = peoplePicker.GetAllUserKeys();
```

Because the people picker is not allowing multiple values, the returned value is the domain\userName of the searched account. (Microsoft 2013e)



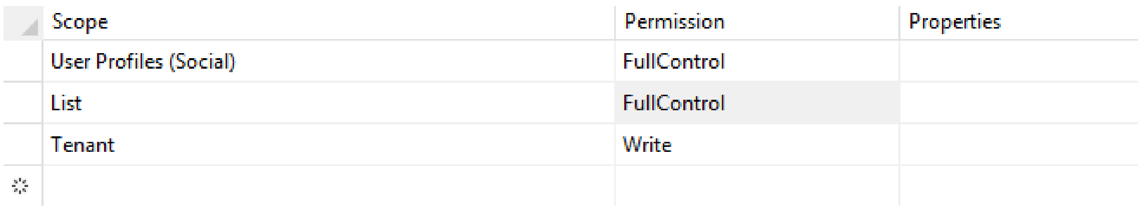
```
Key: i:0#.w|contoso0\user
Description: CONTOSO0\user
DisplayText: User Usersson
EntityType: User
ProviderDisplayName: Active Directory
ProviderName: AD
IsResolved: true
EntityData: [object Object]
MultipleMatches:
AutoFillKey: i:0#.w|contoso0\user
AutoFillDisplayText: User Usersson
AutoFillSubDisplayText:
AutoFillTitleText: Active Directory CONTOSO0\user
DomainText: app-9ac9ae87ece4e0.apps.contoso.com
Resolved: true
LocalSearchTerm: user
```

Figure 21 All the user metadata that the picker returns

With the use of the people picker the needed sets of metadata from user profiles can be extracted for any user. And by assigning the picked user as the social actor it is possible to directly access their feed and comment on it.

4.5.2 Implementing the PRAISE to the social feed

To POST to a social feed the app needs Write permissions to the Tenant scope. By default the permissions are Read, in the App Manifest the Tenant Scope is assigned Write permissions which will be granted when the administrator of the Site Collection approves the app.



Scope	Permission	Properties
User Profiles (Social)	FullControl	
List	FullControl	
Tenant	Write	

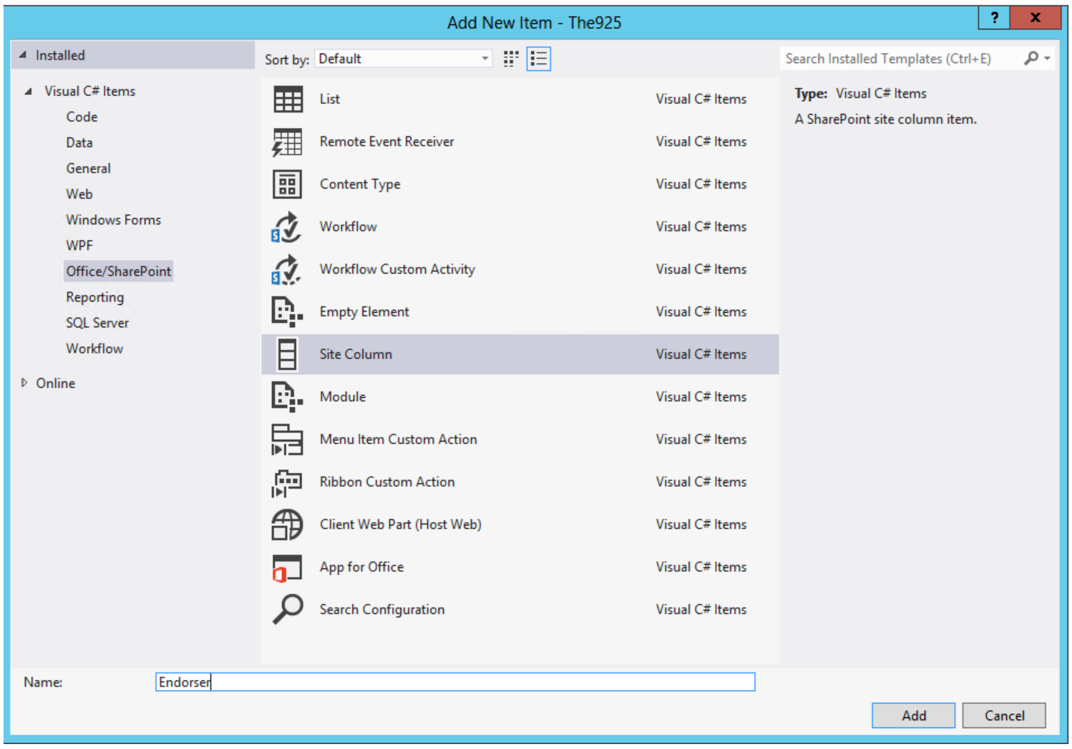
Figure 22 App has been given write permissions to tenant scope

By creating two different sets of messages to different SP.ClientContexts, one for the Social Actor giving the Praise and the other for the target(s) we can create two slightly different POSTs to respective feeds. The Feed is created as SP.Social.SocialFeedManager.createPost(targetId, creationData), the target ID stands for where in the post hierarchy the POST is written and the creation data is the whole object passed to the feed. When creating the Praise as a new POST the target ID is null, otherwise it is the ID of the POST where the current POST is added as a child posting.
We set the SP.Social.SocialDataItemType to user by assigning the value 0 to the DataItem, the values are: user = 0, document = 1, site = 2, tag = 3, link = 4. The tag would be great, but, once again a proper Tag for PRAISE with a Global Unique Identifier (GUID) cannot be created, that could be referenced as the object. A text can be set with set_Text() method to the SP.Social.SocialDataItem() to allow the PRAISE to be consistent even with user added data. For the feed to be noticed better an image of a star or something similar should be added, but again this is only supported with server side programming. (Microsoft 2013g & Narasiman 2014)

4.6 SharePoint Artefacts

For the app to work some SharePoint artefacts are needed to store data as the App only runs from the browser. One of these artefacts is a custom hidden list where the actual endorsements will be stored. To create such a list first the project is chosen, and a custom list is added. The list contains three columns Endorser, endorsee and Skill.

The first thing is to set up the list columns. Thi sis done from Visual Studio by clicking the project and add button

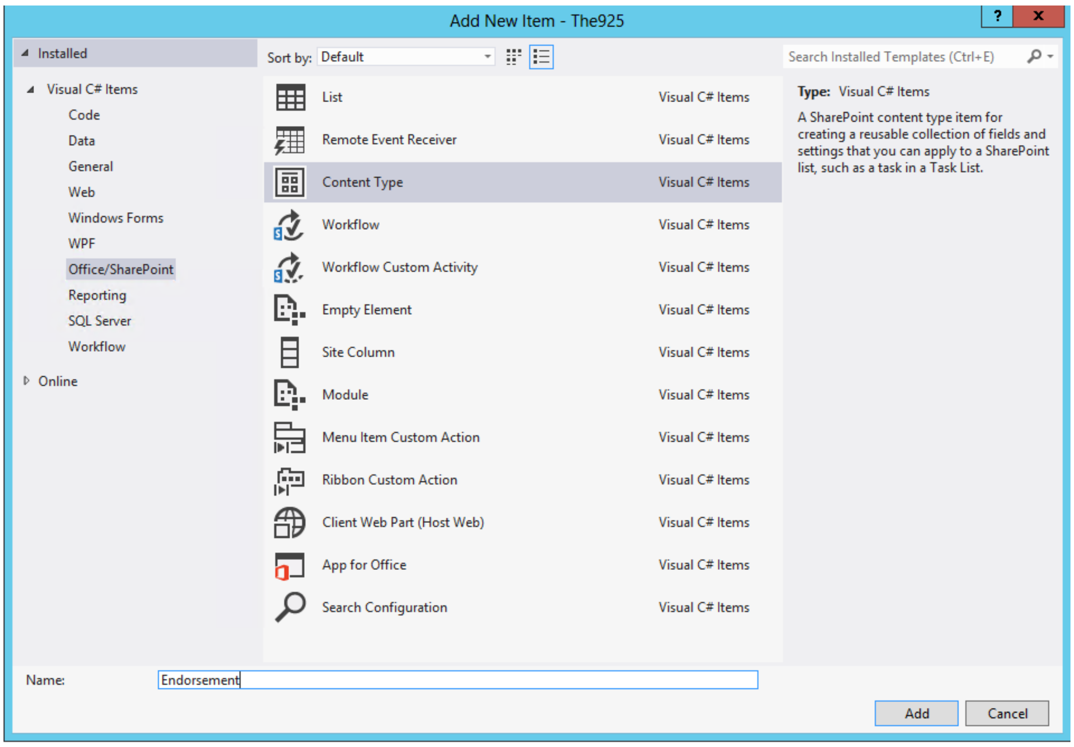


Type: Visual C# Items
A SharePoint site column item.

Name: Endorser

Figure 23 Adding custom Site Column to be used with the list

After all columns are created they must be assigned to a content type to be able to use them in a SharePoint list. For the app a new content type is created by adding it from the project menu.



Add New Item - The925

Installed

Sort by: Default

Search Installed Templates (Ctrl+E)

Visual C# Items
- Code
- Data
- General
- Web
- Windows Forms
- WPF
- Office/SharePoint
- Reporting
- SQL Server
- Workflow

Online

List	Visual C# Items
Remote Event Receiver	Visual C# Items
Content Type	Visual C# Items
Workflow	Visual C# Items
Workflow Custom Activity	Visual C# Items
Empty Element	Visual C# Items
Site Column	Visual C# Items
Module	Visual C# Items
Menu Item Custom Action	Visual C# Items
Ribbon Custom Action	Visual C# Items
Client Web Part (Host Web)	Visual C# Items
App for Office	Visual C# Items
Search Configuration	Visual C# Items

Type: Visual C# Items

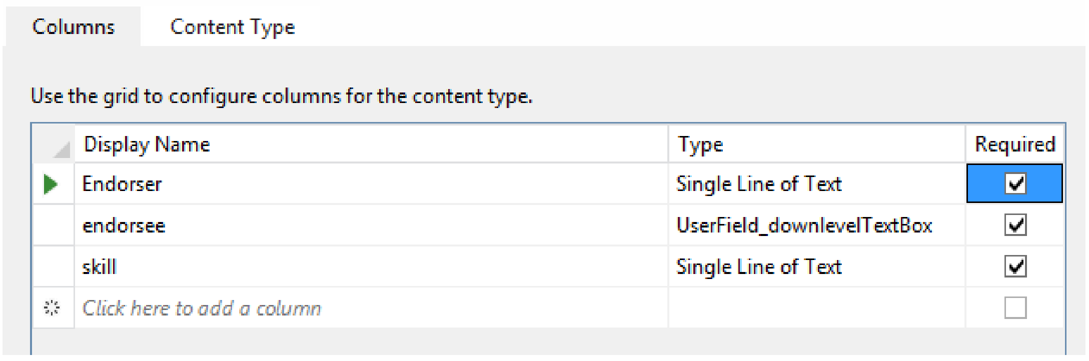
A SharePoint content type item for creating a reusable collection of fields and settings that you can apply to a SharePoint list, such as a task in a Task List.

Name: Endorsement

Add Cancel

Figure 24 Creating a content type to use in the list

There are numerous choises from where to choose the inheritance of the content type. For simplicity the Item base content type was chosen. Next the columns are assigned to the content type and given field type values. For simpicity the people columns are kept as text, this should be changed later to display the name and picture.



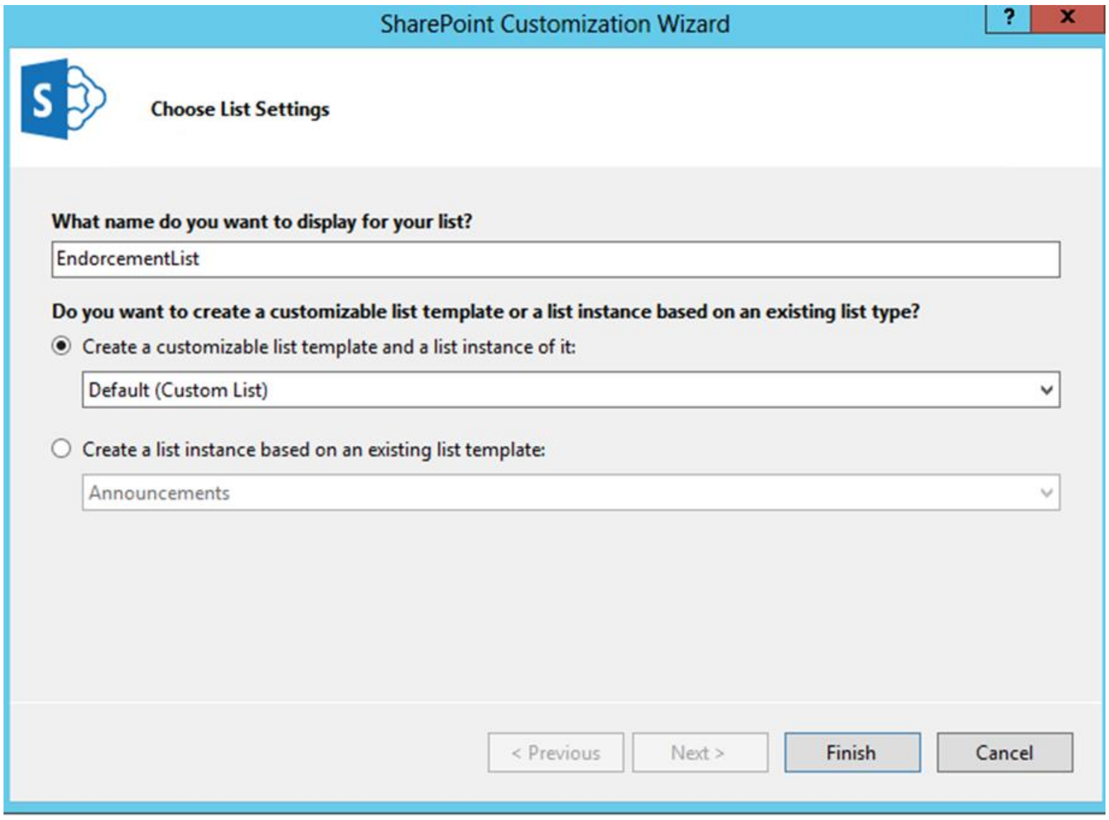
Columns Content Type

Use the grid to configure columns for the content type.

Display Name	Type	Required
Endorser	Single Line of Text	✓
endorsee	UserField_downlevelTextBox	✓
skill	Single Line of Text	✓
Click here to add a column		☐

Figure 25 The columns and their types, Userfield_downleveltTB does not work

The list is created in the same manner, the initial choises are for the type of list, any type of SharePoint list can be added including announcements and wikis.


SharePoint Customization Wizard

Choose List Settings

What name do you want to display for your list?

EndorcementList

Do you want to create a customizable list template or a list instance based on an existing list type?

[✓ Create a customizable list template and a list instance of it:]

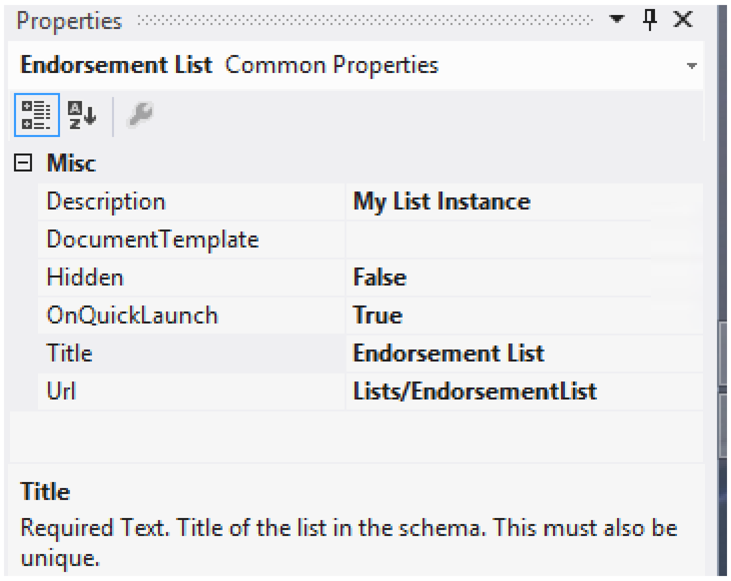
Default (Custom List)

Create a list instance based on an existing list template:

Announcements

Figure 26 Creating the list

The List cannot unfortunately be added to the root web without a Feature, the only option is to save it to the app web and hide it by setting the hodden attribute to true. For the purpose of debugging the list the option is left to false for the time of implementation.


Properties

Endorsement List Common Properties

Misc

Description	My List Instance
DocumentTemplate	
Hidden	False
OnQuickLaunch	True
Title	Endorsement List
Url	Lists/EndorsementList

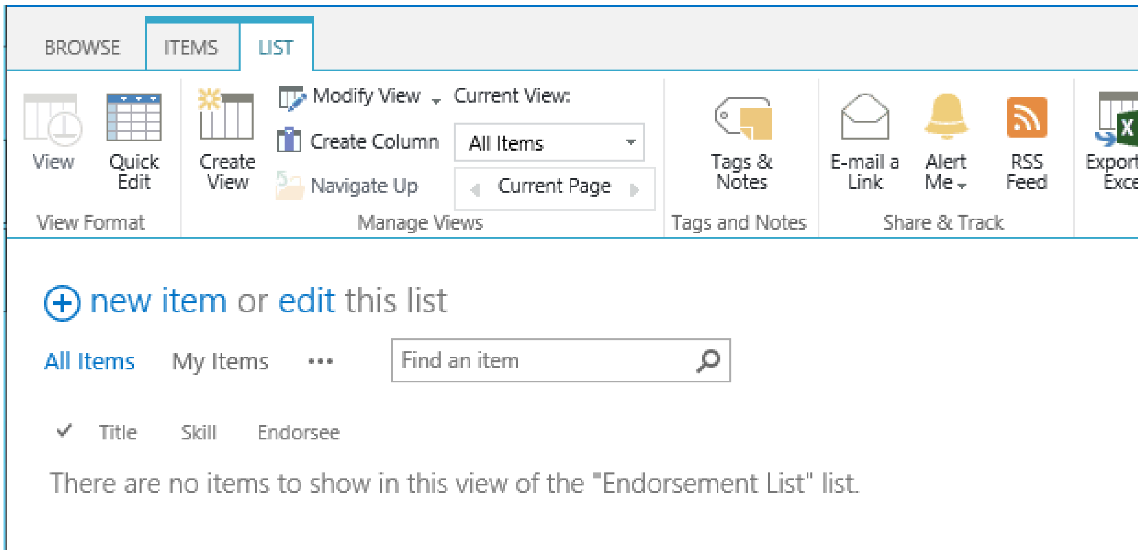
Title

Required Text. Title of the list in the schema. This must also be unique.

Figure 27 The List properties in Visual Studio properties window

When deploying the app the list is created under the app web url:

http://app-9ac9ae87ece4e9.apps.contoso.com/The925/Lists/EndorsementList/AllItems.aspx



There are no items to show in this view of the "Endorsement List" list.

Figure 28 The List in the App domain, still without entries

Because the app will be displayed in an app part of some other web than the app web the app web must be properly refernced when making calls to the app resources, or in this case the Endorcement list. For this the cross domain request calls have to be implemented and authorized. (Mircosoft 2014f)

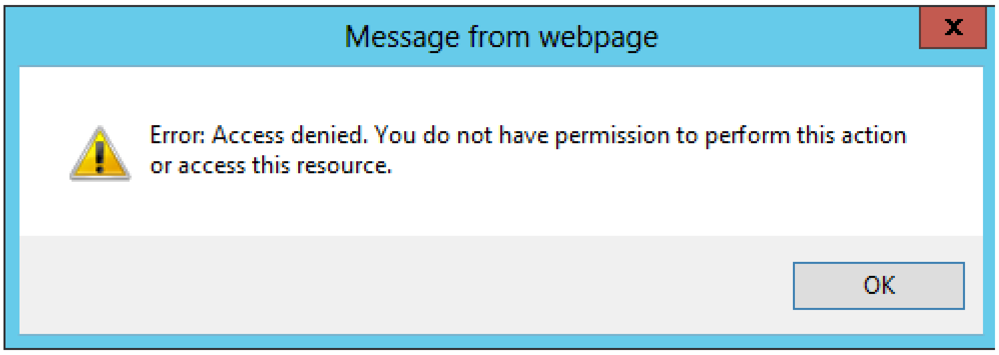
4.6.1 Updates on the list items

Usually if something changes that should affect the list a custom event receiver has to be implemented. The event receiver can be primed to monitor any changes to a list in the web application. Should the user leave the company or change the skills this should be registered on the Endorsement list, however this had to be discarded as the app cannot gain access to any Service Application, such as user profiles. On the server side it is possible to access the Microsoft.Office.Server.UserProfiles.dll assembly and through code implement monitoring on this library. This is accessed through UserProfileManager.GetChanges() method. (Lapointe & McDermott 2011) Unfortunately there is no equivalent on client side leaving no alternative but to iterate each user and look for changes. This can be made by checking each

user against Last Keyword Added Property (SPS-LastKeyWordAdded) and by doing so only in rendering the UI for specific users the app should be able to get the information without too long delays.
A JQuery working on it effect should be implemented at later stages.
At the moment the current user is looked for from the Endorsement List and with each row returned the check is made before rendering the Endorsements. For this purpose a new last modified column has been added to the list, and if the endorsee has modified the user profile since this time the whole list item is updated.

4.7 Permissions

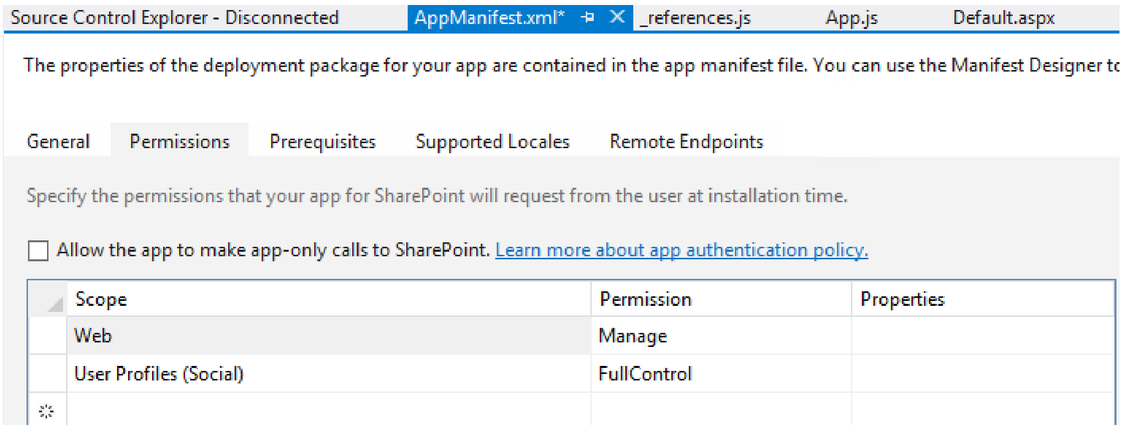
Permissions of the app are as important as the code itself. App permissions are set from the AppManifest.xml where the metadata of the app is configured. Without proper permissions the following error will appear on the App page:



Error: Access denied. You do not have permission to perform this action or access this resource.

Figure 29 Error message for insufficient rights

On the other hand Administrators will not install apps which have too much power over SharePoint. The App should be given the right set of Permissions to do what its designed to do.Here are the permissions of the SocialApp before the list had been implemented:


Source Control Explorer - Disconnected | AppManifest.xml* | _references.js | App.js | Default.aspx

The properties of the deployment package for your app are contained in the app manifest file. You can use the Manifest Designer to

General | Permissions | Prerequisites | Supported Locales | Remote Endpoints

Specify the permissions that your app for SharePoint will request from the user at installation time.

☐ Allow the app to make app-only calls to SharePoint. Learn more about app authentication policy.

Scope	Permission	Properties
Web	Manage	
User Profiles (Social)	FullControl	

Figure 30 App manifest XML in editor, can be also edited as the original XML file

4.8 Deploying the the App

When the app is deployed it creates its own app web under the root web. The URL can be broken down in the following manner:

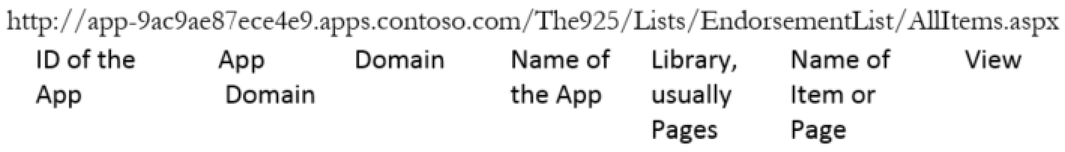

http://app-9ac9ae87ece4e9.apps.contoso.com/The925/Lists/EndorsementList/AllItems.aspx

Figure 31 The URL components

The endorsement list will be hidden and the The925/pages/default.aspx will projected to a web part once the app is installed to the app catalog of the host system.

4.9 The user interface

Given all the obstacles and limited time, the user interface was decided to be left outside the scope. Because the application logic is working the UI can be implemented as a separate project to provide for the much needed gamification and good UX aspects.

4.10 Troubleshooting errors

During the coding phase numerous errors will be encountered during debugging. Here are listed the ones that were critical and had to be fixed.

Table 5 The Errors and solutions

Error	**Solution**
Unable to get property SP.UserProfiles.PeopleManager of undefined or null reference	Register SP.UserProfiles before any actions: `$.getScript(scriptbase + "SP.UserProfiles.js", getUserProperties);` getUserProfiles is the method where Sp.UserProfiles.js is used
JavaScript runtime error: The property or field 'PictureUrl' has not been initialized. It has not been requested or the request has not been executed. It may need to be explicitly requested.	This turned out to be an issue with SP.UserProfiles.js script not loading even with `SP.SOD.executeOrDelayUntilScriptLoaded(getUserProperties, 'SP.UserProfiles.js');` declared in the App.js Fix: Declare the script location in Html `<script src="/_layouts/15/SP.UserProfiles.js"></script>`
Request failed: An app requested information that is not available from this location.	No Feed was available, wait if feed has been creatred. Implement null check.
For some strange reason the Endorser column has been changed to Title	Create a new view and make that view default, the Endorser is displayed
The property or field 'Id' has not been initialized. It has not been requested	This had similarities with the previous error. Then initializing the endorser field the program throws this error because it's expecting Title. FIX: `clientContext.load(collListItem, 'Include(Title, endorsee, skill)');`
A potentially dangerous request. Path value was detected from the client when using REST to upgrade list	Normally a `<%@ Page ValidateRequest="false"` would fix this but it is not allowed in SharePoint Hosted Apps The error was not fixed and the Post was done by CAML query instead.

5 Conclusions of the project

The world of intranets has changed. Only three years ago the balance of work was in customizing intranets for the customer because the platform did not offer all the capabilities desired. There were numerous systems that were placed on top of SharePoint for added value. Today SharePoint 2013 is a huge application with extremely vast capabilities OOTB, improving on these capabilities is not something that the customizations usually target. This has led to the shifting from coding projects to consultancy and Support projects. More work is also done on ensuring the communication between systems. Microsoft's vison is to offer SharePoint from the cloud only without the possibility to on premises support has led to the Apps approach. The App is a convenient way of doing small tasks, but cannot replace the Server Side Object Model with its limited capabilities and resources. At the moment the documentation for the JavaScript API is far from complete, the links library in Microsoft office Dev Center called JavaScript API reference for SharePoint 2013 is still leading mostly to empty pages, you can assure yourself of this by visiting their site at http://msdn.microsoft.com/en-us/library/office/jj193034.
Customers are also aware of the dangers with huge data in cloud, there have been many breaches to these Data Stores even this year.
The social features for SharePoint are good and with the possibility to get Yammer embedded in SharePoint makes the App nearly obsolete, but with some further effort it could be an affordable improvement to an existing intranet.
Working with SharePoint was initially chosen as a natural way to display the skills learned at the University and at Work. Previously the code used was C# along with html and CAML query, the SharePoint artefacts were accessed by referencing SharePoint .ddl: s and the use of SPList, SPIList item, SPWeb and SPSite had become familiar. This project was not using the familiar commands that could be used with code behind solutions and therefore the learning curve was steeper than expected.

5.1 Achievements

The project was successful in retrieving the needed data from SharePoint and in finding ways to manipulate the data in desired ways to get a functioning App. The insight on how the code should work and how to setup the programming environments was well learned. Upon discovering that the properties cannot be set from client side, the re-evaluation and assessment produced an idea for getting the working with other tools. Although installing a list to the root web of an application can be hard on the system if there are many users who start using endorsements, but with the idea of raising the ListViewTresh-old for the application root web this problem can be overcome. The logic of the App is working and without unnecessary hardcoding and string manipulations, although some string manipulation has to be used to get the desired part of the huge mass of data returned by calls to the JavaScript API. The major ob-stacles that were listed in the implementation phase were overcome by re-thinking in terms of “what can be done” instead of “can’t do it”.

5.2 Obstacles

At the best of times SharePoint 2013 in is somewhat volatile, something that is known by all the users. Error messages are not self-explanatory and some-times quite impossible to understand. When dealing with a rather new technol-ogy the Online sources are not as abundant on the matter as with some older versions, thankfully some advice can be retrieved and with a little thought ap-plied to the latest version as well.

The project was started over three times because of limitations on Office 365 development site lease times and the unstableness of the Hyper-V virtual ma-chine system introduced with Windows 8. The code was lost once because of corruption of the virtual machine and once with the corruption of the work-station Windows 8.

With scrapping of the original idea in 2013, the project was restarted in early summer 2014 and by then some of the original code no longer was usable be-cause of new restrictions. The setbacks were cumbersome and had effect on the motivations on completing the project.

When the running into the fact that Person properties cannot be set or, as hoped for, created with Key and value multivalued property, the application seemed to be unable to do what it was intended to do.

5.3 Successes

By far the hardest part and the biggest success was with the setup of the vir-tual machine environment. With the use of server side tools such as Central Administration and PowerShell command script the internal names and prop-erties could be resolved making the client side coding much easier. Unfortu-nately this was also very time consuming and took up all of the time primarily reserved for implementing the code. All error messages that were thrown were resolved and the important ones also documented.

5.4 Misleadings

The one big misleading was the initial assumption that the user properties could be manipulated with an app, this lead to the whole project being laid down for a while. At the restart the project needed to be rethought and an alternative solution had to be researched. As the author has gained a lot of knowledge about SharePoint 2013 from working with in various customer environments, there was a lot of ideas that had to be scrapped as the client side object model did not support the actions that were initially thought possible.

5.5 Areas for Improvement

The area of improvement that is critical for a project of this nature is the choosing of a scope that is narrow enough but has the challenge that a thesis must have. This project had far too many big obstacles to overcome to have had any chance to succeed within the given timeline and effort.
The App could be completed with the gamification and UX in mind, making it a neat addition to some intranet. By using JQuery effects the and deploying the app to some landing page it could be something users would adopt.
The project grew not only in scale but the timetable was impossible to keep as well, due to inexperience in implementing in client side and very limited time resources of the author.

6 Summary

SharePoint offers many interfaces for programming on the client side. The documentation on how to access these interfaces is quite good and examples are abundant. Unfortunately the steps needed to be taken before these sources are useful are not as straight forward as could be desired. Setting up the development environment for Apps is a rocky road at its best.

When the using the Azure developer site the developer does not have all the resources of SharePoint available and thus somewhat more challenging than the working on premises. The App that is developed for Social data use has limited resources of social data available unless more users are invited to the site in the first place to create the social data.
On premise development is on the other hand harder to set up, the isolated app web needs to be configured exactly right and the developer needs to have knowledge not only about programming, but also about the infrastructure and SharePoint Specific server side scripting. The App web needs to be added to DNS forward lookup zones to be accessed, the apps also need their own managed accounts to perform correctly in the host environment or even to deploy.

When the environment finally was successfully configured and the App deployed to the host environment, the scale of the product began to grow. When finding out that an App running on the client side interface could only execute get commands against user profile properties the logic had to be rethought. With a new perspective came new obstacles, the app needed to access its own web from host webs, something that the built in security of web browsers and SharePoint will try to prevent. One of the most difficult things to implement is the Cross Domain Request, a ratified way of Cross Site Scripting. The problems with the ratification could not be solved by the normal declarations in the page head because the App does not allow these changes.

The last major force to affect this project was volatile software. Not only is SharePoint 2013 sensitive to slightest changes and may stop working for no apparent reason, but the Windows 8 operating system and Hyper-V Manager also behaved eccentrically. This project had to deal with properties being dropped as services shut down unexpectedly on SharePoint, the complete crash of the operating system, broken network adapters and broken hard drives. This could possibly have been avoided if SharePoint would not have been on a Virtual Machine designed for development purposes. But, if the development would have been directed on a live environment the risks would have been far greater. Some problems were waved by installing Windows 8.1 operating system, which has improved stability compared to Windows 8.

The App Model itself is an easy on the environment and ingenious way of adding some extra features to SharePoint 2013. When the demands for safety have been met the App can, by using SharePoint specific resources, do a lot and bring more out of the social features of SharePoint 2013. However with the next generation of SharePoint it will be obsolete since the general consensus is that Yammer will be integrated fully to the SharePoint package, it is already available to be embedded to the existing environments and by itself can be used as an intranet.

7 References

Balmer, S. July 9, 2012. Microsoft Worldwide Partner Conference 2012 Day 1 Keynote. Can be read: http://www.microsoft.com/en-us/news/speeches/2012/07-09wpcday1ballmer.aspx retrieved 30.3.2013

Caya, P. Nielsen, J. Pernice, K. & Schad, A. January 5, 2014. Can be read: http://www.nngroup.com/articles/intranet-design/ retrieved 24.10.2014

Crow Canyon Systems 2014. The Evolution of SharePoint: a Brief History. Can be read: http://sharepoint-applications.com/sharepoint/the-evolution-of-sharepoint-a-brief-history/ retrieved 22.10.2014

Del Monte, M. August 11, 2014. Top 6 Other Names For Intranet Software. Can be read: http://www.intranetconnections.com/blog/other-names-intranet-software/ retrieved 20.10.2014

Fox, S. February 3, 2013. Steve Fox's OBA Ramblings, Setting up your Development Environment for O365 & Azure Development. Can be read: http://blogs.msdn.com/b/steve_fox/archive/2013/02/03/setting-up-your-development-environment-for-o365-amp-azure-development.aspx retrieved 30.3.2013

Khartikeyan, E. March 3, 2013. SharePoint Evolution. Can be read: http://karthikeyane63.wordpress.com/category/sharepoint-evolution/ retrieved 30.03.2013

Korhonen, H October 21, 2014. Intranet-ostajan opas, Intranet pilvestä – Kolme kokemusta SharePoint Onlinesta. Can be read: http://intranet-ostajanopas.fi/2014/10/21/intranet-pilvesta-kolme-kokemusta-sharepoint-onlinesta/ retrieved 10.11.2014

Lapointe,G & McDermott, M. Apr 20, 2011. Monitor SharePoint User Profile Changes. Can be read: http://sharepointpromag.com/sharepoint-2010/monitor-sharepoint-user-profile-changes retrieved 27.10.2014

Matthews, P. January 28, 2014. 8 Features Your Intranet Must Have In 2014. Can be read: http://www.claromentis.com/blog/8-features-your-intranet-must-have-in-2014/ retrieved 20.10.2014

Microsoft 2014a. May 29, 2014. Dev Center, Apps for Office and SharePoint. Can be read: http://msdn.microsoft.com/en-us/library/office/fp161507(v=office.15).aspx retrieved 05.06.2014

Microsoft 2014b. August 25, 2014. Dev Center, Sign up for an Office 365 Developer Subscription, set up your tools and environment, and start deploying apps. Can be read: http://msdn.microsoft.com/library/office/fp179924(v=office.15) retrieved 24.10.2014

Microsoft 2014c. August 27, 2014. Dev Center, Choose patterns for developing and hosting your app for SharePoint Can be read: http://msdn.microsoft.com/en-us/library/office/fp179887 retrieved 30.3.2013

Microsoft 2014d. March 3, 2014. TechNet Library, Configure an environment for apps for SharePoint (SharePoint 2013). Can be read: http://technet.microsoft.com/en-us/library/fp161236(v=office.15) retrieved 5.9.2013

Microsoft 2014e. May 16, 2014. Office Blog, Update on Autohosted Apps Preview program Can be read: http://blogs.office.com/2014/05/16/update-on-autohosted-apps-preview-program/ retrieved 6.11.2014

Microsoft 2014f. May 11, 2014. Developer Network, How to: Access SharePoint 2013 data from apps using the cross-domain library can be read: http://msdn.microsoft.com/en-us/library/fp179927.aspx retrieved 24.10.2014

Microsoft 2013a. July 16, 2013. Dev Center, Choose the right API set in SharePoint 2013. Can be read: http://msdn.microsoft.com/en-us/library/office/jj164060(v=office.15).aspx#Factors retrieved 30.03.2013

Microsoft 2013b. August 25, 2013. Dev Center, Sign up for an Office 365 Developer Subscription, set up your tools and environment, and start deploying apps. Can be read: http://msdn.microsoft.com/library/office/fp179924(v=office.15) retrieved 30.03.2013

Microsoft 2013c. August 30, 2013. Dev Center, How to: Set up an on-premises development environment for apps for SharePoint. Can be read: http://msdn.microsoft.com/en-us/library/office/fp179923(v=office.15).aspx retrieved 5.9.2013

Microsoft 2013d. July 16, 2013. Dev Center, Choose the right API set in SharePoint 2013. Can be read: http://msdn.microsoft.com/en-us/library/office/jj164060(v=office.15).aspx retrieved 20.10.2014

Microsoft 2013e. July 2, 2013. Dev Center, SharePoint 2013: Add the client-side People Picker control to apps. Can be read: https://code.msdn.microsoft.com/office/SharePoint-2013-Add-the-900e0742 retrieved 20.10.2014

Microsoft 2013f. July 1, 2013. Dev Center, How to: Work with user profiles and organization profiles by using the server object model in SharePoint 2013. Can be read: http://msdn.microsoft.com/en-us/library/office/jj163142(v=office.15).aspx#bkmk_CreateUPProp retrieved 20.10.2014

Microsoft 2013g. August 28, 2013. Dev Center, SP.Social.SocialDataItem object (sp.userprofiles). Can be read: http://msdn.microsoft.com/en-us/library/office/jj679811(v=office.15).aspx#methods retrieved 27.10.2014

Narasiman, S. February 2, 2014. How to publish a post to SharePoint Social Feed using SharePoint 2013 JSOM. Can be read: http://sundarnarasiman.net/?p=134 retrieved 27.10.2014

Oswald, A. Proto, E, & Sgroi, D. Thesis: Happiness and Productivity. Can be read: http://www2.warwick.ac.uk/fac/soc/economics/staff/eproto/workingpapers/happinessproductivity.pdf retrieved 9.11.2014

Sisler, L. August 29, 2012. SharePoint Adoptation and Use. Can be read: http://blogs.perficient.com/microsoft/2012/08/sharepoint-adoption-and-use-in-fographic/ retrieved 24.10.2014

Turnbull, C. August 27, 2013. Flat Design, iOS 7, Skeuomorphism and All That. Can be read: http://webdesign.tutsplus.com/articles/flat-design-ios-7-skeuomorphism-and-all-that--webdesign-14335 retrieved 9.11.2014

van Olst, M. July 29, 2012. Mirjam's thoughts on SharePoint, Setting up your App domain for SharePoint 2013. Can be read: http://sharepointchick.com/ar-chive/2012/07/29/setting-up-your-app-domain-for-sharepoint-2013.aspx re-trieved 10.10.2013

8 Appendices

8.1 Glossary

AJAX	Asynchrous JavaScript And XML, a modern way for the application to communicate with the server without having to reload the whole page
API	Application Programming Interface, a connection point to the application
BIN	Application specific library for compiled code, for universal library see GAC
CAML	Collaborative Application Mark-up Language, a SharePoint specific query syntax based on XML, used with SPQuery class
CMS	Content Management System, software for publishing information
CSOM	Client Side Object Model
CRUD	Create, Read, Update and Delete, abbreviation of full privileges
CSS	Cascading Style Sheet, style syntax readable by all browsers
DNS	Domain Name System, applies the name and address to the application
GAC	Global Assembly Cache, the server side system library for all code that is not application restricted. Code that is application specific see BIN
HTML	Hypertext Mark-up Language, structure syntax readable by all browsers
JSOM	JavaScript Object Model, client side object model using JavaScript
JSON	JavaScript Object Notation, easy to understand and lightweight data transfer format
LAMP	Linux, Apache, MySQL and PHP used as a platform for software
LINQ	Language Integrated Query, Microsoft syntax for querying data from application
MOSS	Microsoft Office SharePoint Server. Usually MOSS 2007 and WSS together
MSDN	Microsoft Developer Network, The library and discussion forum on Microsoft platforms and development.
OOTB	Out Of The Box, unmodified software
REST	Repetitional State Transfer, a dynamic connection endpoint

SDK	Software Development Kit
SEO	Search Engine Optimisation
UI	User Interface, the way the application is displayed to the user
UX	User Experience, how the functionalities are portrayed and made accessible to the user
WSS	Windows SharePoint Services, The Server services of Share-Point 2007

8.2 About the author

The Author of this thesis has worked for two years as Software Developer in a Maintenance Team, up keeping various customer environments based on SharePoint 2007, SharePoint 2010, SharePoint 2013 and Microsoft CMS 2002. During this time the author has learned the skills needed for developing SharePoint solutions in the previous versions of SharePoint, administration of all versions of SharePoint and troubleshooting a myriad of errors in different scenarios. The author has made custom web parts, features and hidden scripts for customer solutions, corrected databases and has worked with the tools provided for SharePoint maintenance such as STSADM.exe and Pow-erShell. This project will strengthen the skills in SharePoint 2013 development but could not be completed without the previous knowledge about the struc-ture and capabilities of SharePoint 2013 and its predecessors.

8.3 Appendix 1 How to set up the development environment

1. Get the Microsoft developer site started and familiarize yourself with SharePoint 2013.
2. Install Visual Studio 2012 or later
3. Get the following tools for Visual Studio:
    a. Office Developer Tools for Visual Studio 2012
    b. SharePoint Client Components (containing the client assem-blies)
    c. Windows Identity Foundation (WIF) SDK
    d. Workflow Tools SDK and Workflow Client SDK
    e. Windows Identity Foundation SDK and Windows Identity Foun-dation Extensions.

The correct way to install the needed updates to Visual Studio are as follows:

In Control Panel, on the Programs and Features page, choose the product edition to which you want to add one or more components, and then choose Change.

In the Setup wizard, choose Modify, and then choose the components that you want to install.

Choose “Next”, and then follow the remaining instructions. (Fox 2013)

8.4 References of appendices

Fox, S. February 3, 2013. Steve Fox's OBA Ramblings, Setting up your Development Environment for O365 & Azure Development. Can be read: http://blogs.msdn.com/b/steve_fox/archive/2013/02/03/setting-up-your-development-environment-for-o365-amp-azure-development.aspx retrieved 30.3.2013