|  |
| --- |
|  |

17/3/2015

|  |
| --- |
|  |
| https://uknowit.uwgb.edu/images/group91/41011/Office365.jpg |
|  |



|  |  |
| --- | --- |
|  | Laboratorio 6. Uso de las API’s de Office 365 (I) |

**Indice**

[1 Introducción 2](#_Toc414306285)

[2 Ejercicio 1: Crear una Aplicación ASP.NET MVC 5 3](#_Toc414306286)

[2.1 Tarea 1 Crear una lista de Memorias de Projectos 3](#_Toc414306287)

[2.2 Tarea 2 Subir una Memoria de Proyecto a One Drive 5](#_Toc414306288)

[2.3 Tarea 3 : Crear algunos contactos 6](#_Toc414306289)

[2.4 Tarea 4: Crear el Proyecto en Visual Studio 8](#_Toc414306290)

[2.5 Tarea 5 Conectar los servicios 11](#_Toc414306291)

[2.6 Tarea 6 Añadir paquete ADAL 13](#_Toc414306292)

[2.7 Tarea 7: Añadir Ficheros auxiliaries al Proyecto 14](#_Toc414306293)

[2.8 Tarea 8: Modificar el código del controlador Home 14](#_Toc414306294)

[2.9 Tarea 9 Código del Discover Service 17](#_Toc414306295)

[2.10 Tarea 10: Obtener los Contactos 21](#_Toc414306296)

[2.11 Tarea 11 Obtener los Ficheros 25](#_Toc414306297)

[2.12 Crear Nuevo Proyecto 29](#_Toc414306298)

[3 Ejercicio 2: Crear Aplicación Apache Cordova 36](#_Toc414306299)

[3.1 Tarea 1: Crear el Proyecto en Visual Studio 36](#_Toc414306300)

[3.2 Conectamos los servicios de Office 365 37](#_Toc414306301)

[3.3 Agregamos Angular JS a la solución 37](#_Toc414306302)

[3.4 Creación del servicio de acceso a las API’s de Office 365 37](#_Toc414306303)

[3.5 Creación de la Pantalla de Login 37](#_Toc414306304)

[3.6 Creación de la Pantalla de mostrar los contactos 37](#_Toc414306305)

[3.7 Creación del Inicio de la App 37](#_Toc414306306)

[4 Ejercicio 3: Registrar Aplicación en AAD 37](#_Toc414306307)

[5 Ejercicio 4: Crear Liberia Portable 38](#_Toc414306308)

[5.1 Creación del Proyecto de Visual Studio 38](#_Toc414306309)

# Introducción

**Tiempo estimado para este laboratorio:**

45 a 60 minutos

**Objetivos:**

Después de completar este laboratorio usted será capaz de:

* Aprender a utilizar las API de Office 365 en una aplicación Web ASP NET. MVC 5
* Aprender a utilizar las API de Office 365 en una aplicación móvil

**Prerequisitos**

Visual Studio 2013 Update 3

Tools para Apache Cordova

Tools API Office 365

Debes de tener la cuenta de Office 365 vinculada con la subscripción de Microsoft Azure

**Descripción del laboratorio**

En este laboratorio vamos a introducir la manera de poder utilizar las API’s de Office 365. La finalidad de estas API’s es poder utilizar estos servicios en cualquier de nuestros desarrollos, independientemente de la plataforma en la que decidamos desarrollar, del dispositivo y de la plataforma.

Iniciaremos el laboratorio desarrollando una aplicación web que utiliza las API de Office 365. El laboratorio se creará una aplicación que le permite definir nuevos proyectos de investigación en una lista de SharePoint, asignar un propietario (perteneciente a nuestros contactos de Outlook), y crear una memoria del proyecto (perteneciente a documentos almacenados en nuestro OneDrive).

Continuaremos el laboratorio, desarrollando una aplicación móvil usando Apache Cordova, utilizando el framework JavaScript AngularJS consumiendo las API’s de Office 365.

# Ejercicio 1: Crear una Aplicación ASP.NET MVC 5

En este ejercicio añadiremos documentos y datos a Office 365 mediante esta solución

## Tarea 1 Crear una lista de Memorias de Projectos

1. Acceder a nuestra cuenta de SharePoint Online.
2. Click “Contenidos del sitio”.

3. Click “Añadir una“.

4. Click “Lista Personalizada”.

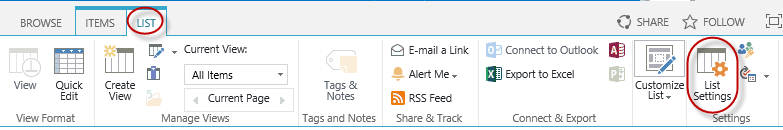
5. Le prondemos de nombre “Research Projects”.

6. Click “Crear”

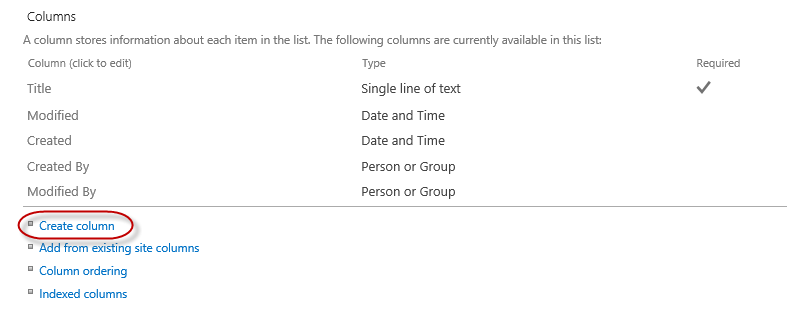


7. Click en la lista recién creada “Research Projects”

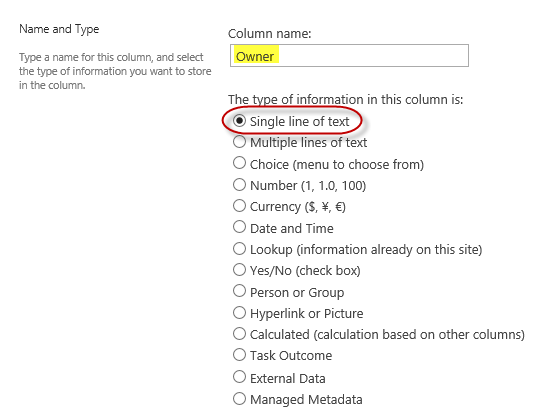
8. Click en la pestaña “Lista” y pulsar sobre “Características”



9. Click “Crear Columna”



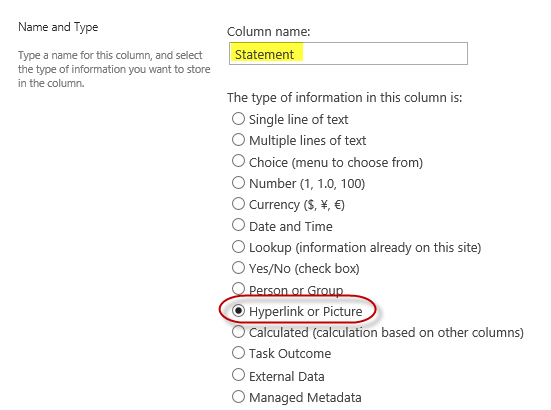
10. Crear nueva columna de tipo “Una línea de texto” y la llamaremos “Owner”



11. Click “OK”.

12. Click “Crear Columna”.

13. Crear nueva columna de tipo “Enlace o Imagen” llamada “Statement”



14. Click “OK”.

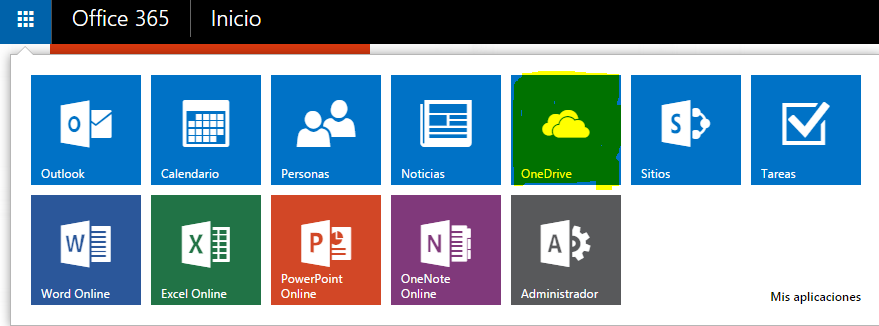
Ahora ya tenemos creada nuestra lista para poder utilizarla en nuestro proyecto

## Tarea 2 Subir una Memoria de Proyecto a One Drive

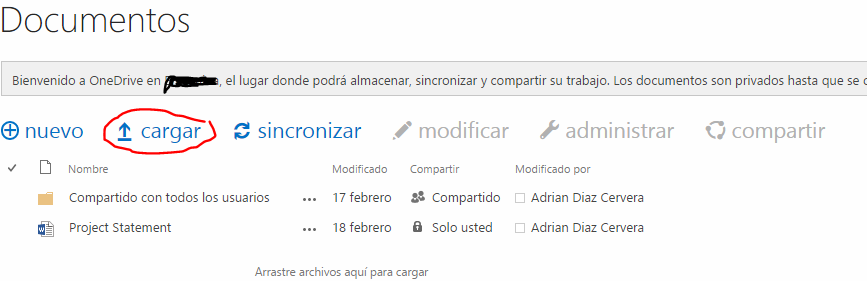
Seguiremos los siguientes pasos para subir una memoria de Proyecto dentro de nuestra libreria personal en One Drive for Bussines

1. Logarnos en nuestra cuenta professional o educativa de Office 365 accediendo a traves del siguiente enlace: http://portal.microsoftonline.com

2. Click “OneDrive”

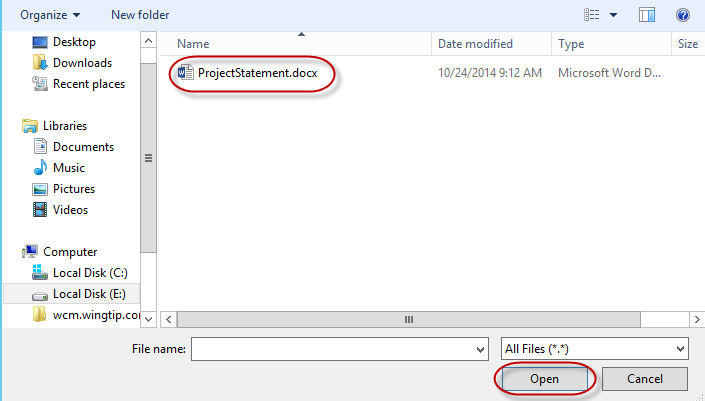


3. Click “Cargar”.



5. Localizar el documento “ProjectStatement.docx”, lo puedes encontrar dentro de la solución dentro de la siguiente carpeta “src/Lab Files/Docs”.

6. Selecciona el fichero y pulsa click en “Open”



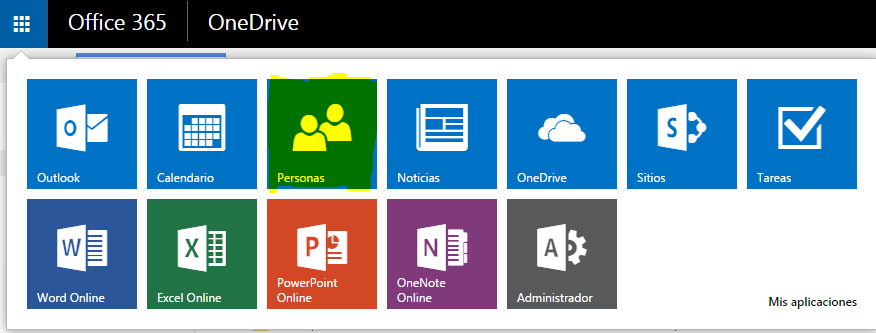
Ahora ya esta disponible el documento dentro de OneDrive for Business.

## Tarea 3 : Crear algunos contactos

Siguiendo estos pasos crearemos algunos contactos en Outlook

1. Logarnos en nuestra cuenta professional o educativa de Office 365 accediendo a través del siguiente enlace: http://portal.microsoftonline.com

2. Click “Personas”



3. Click “Nuevo”



1. Click “Crear Contacto”



4. Crearemos un Nuevo contacto asegurándonos de rellenar “ Nombre, “Apellidos” y “Email”

5. Opcionalmente puedes agregar más contactos

Ahora ya disponemos mínimo de un contacto para trabajar en el Proyecto.

## Tarea 4: Crear el Proyecto en Visual Studio

Siguiendo los siguientes pasos crearemos un Nuevo Proyecto MVC 5

1. Arrancamos “Visual Studio 2013”.

2. Seleccionamos “Archivo/Nuevo/Proyecto” del menu principal.

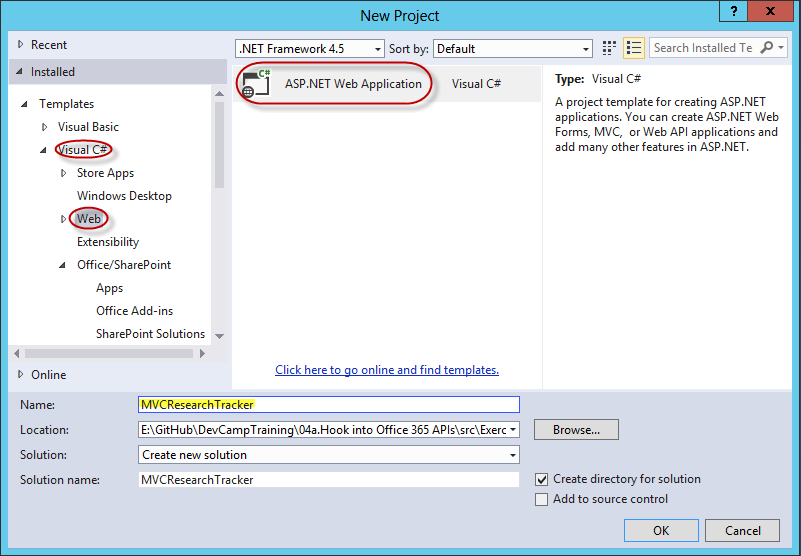
3. En el dialogo de “Nuevo Proyecto”:

1. Seleccionamos “C#/Web”.

2. Click “ASP.NET Web Application”.

3. Llamamos al proyecto “SUGESResearchTracker”.

4. Click “OK”.



4. En la pantalla de dialogo de “Nuevo Proyecto ASP.NET Project":

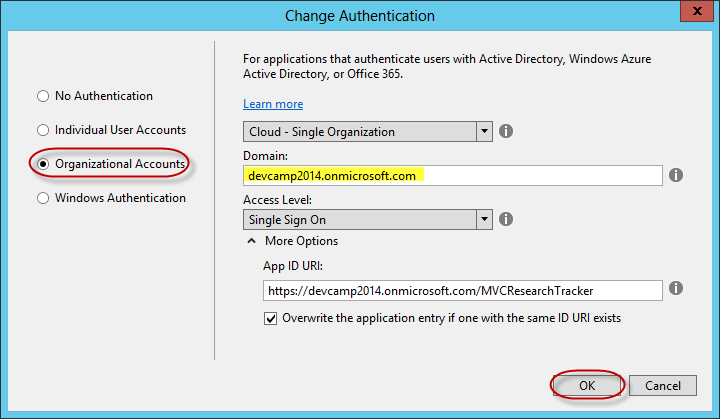
1. Click en la plantilla “MVC”.

2. Click “Cambiar Autenticación”.

3. Click “Cuenta Profesional o educativa”.

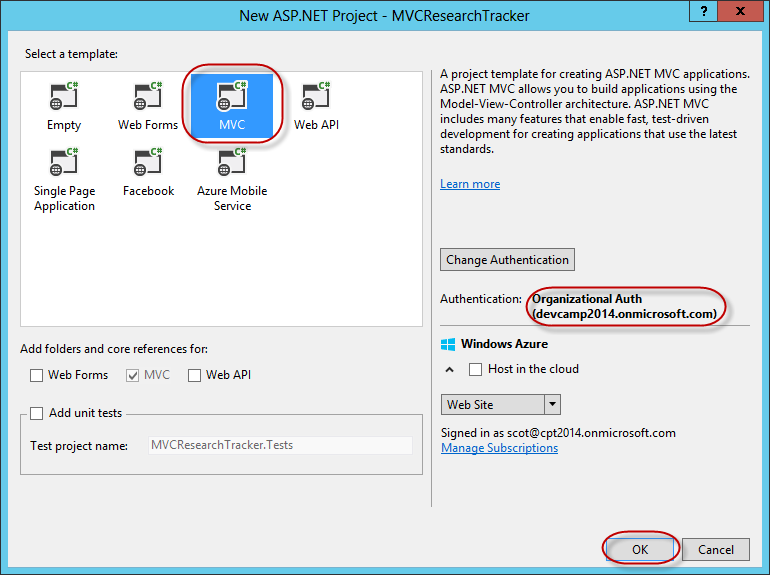
4. Introducimos el dominio de nuestro tenant de Office 365

5. Click “OK”



6. Cuando se muestro, introducer las crecedenciales del Azure Active Directory

7. Click “OK”

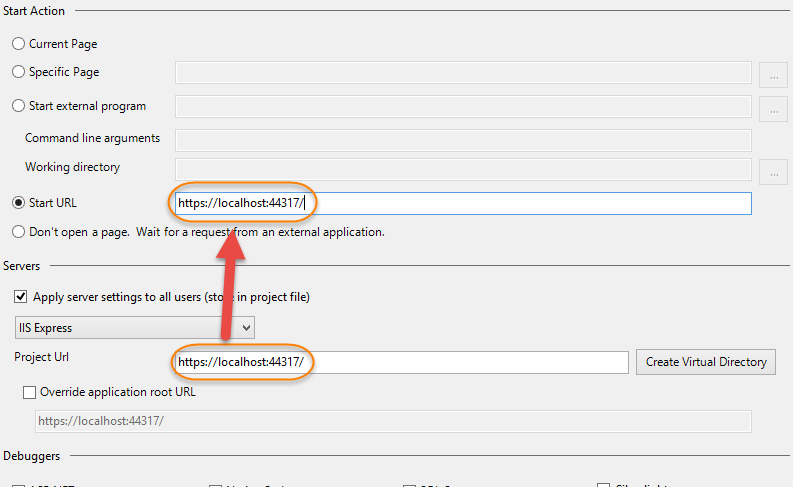


5. En el “Explorador de Soluciónes”, botón derecho en el Proyecto y seleccionamos “Propiedades”

6. Click “Web”.

7. Seleccionamos la opción “Start URL” dentro de “Acción de inicio”.

8. Copiamos el valor del campo “Projecto Url” en el campo “Start URL”. Este valor es el endpoint que usa “htttps”



9. Pulsar “F5” para testear la seguridad de tu apliación introduciendo las credeciales de tu “Cuenta Profesional o educativa”

Ahora ya tienes una aplicación web ASP.NET MVC securizada con la “Cuenta de Office 365”.

## Tarea 5 Conectar los servicios

Siguiendo los siguientes pasos añadiremos los servicios contectados a la aplicación MVC5

1. En el “Explorador de Soluciones”, botón derecho dentro de nuestro Proyecto y seleccionamos la opción “Añadir/Conectar Servicios”.

2. En el cuadro de dialógo de “Service Manager”:

1. Click “Registrar tu App”.

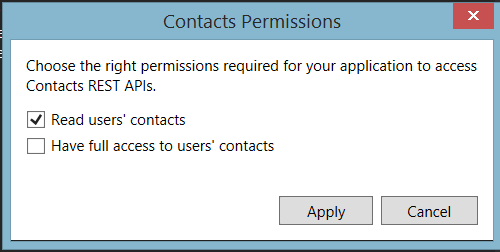
2. Cuando se muestre, introducer las credenciales de la “Cuenta de Office 365” para accede al Azure Active Directory .

3. Click “Contacts”.

4. Click “Permissions”.

5. Seleccionar “Read Users' Contacts”.

6. Click “Apply”.

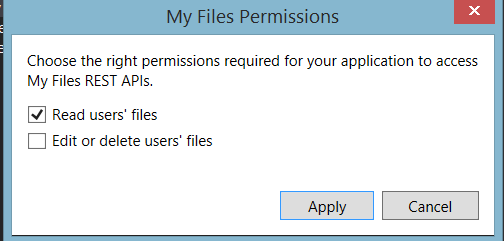


7. Click “My Files”.

8. Click “Permissions”.

9. Seleccionar “Read Users' Files”.

10. Click “Apply”



11. Click “Sites”.

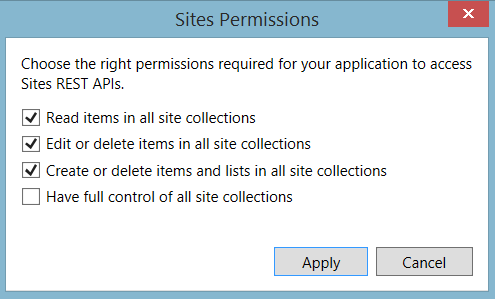
12. Click “Permissions”.

13. Seleccionar “Create or Delete Items and Lists in All Site Collections”.

14. Seleccionar “Edit or Delete Items in All Site Collections”.

15. Seleccionar “Read Items in All Site Collections”.

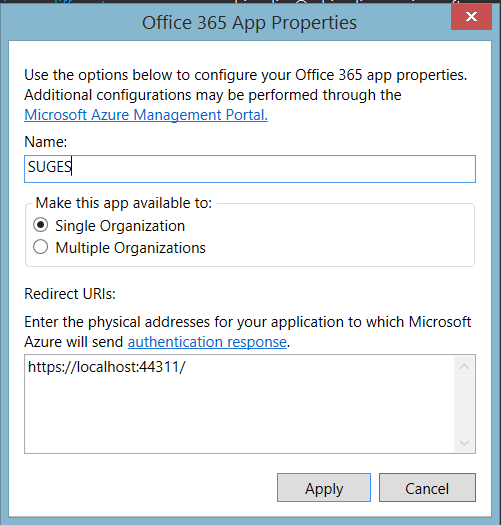
16. Click “Apply”.



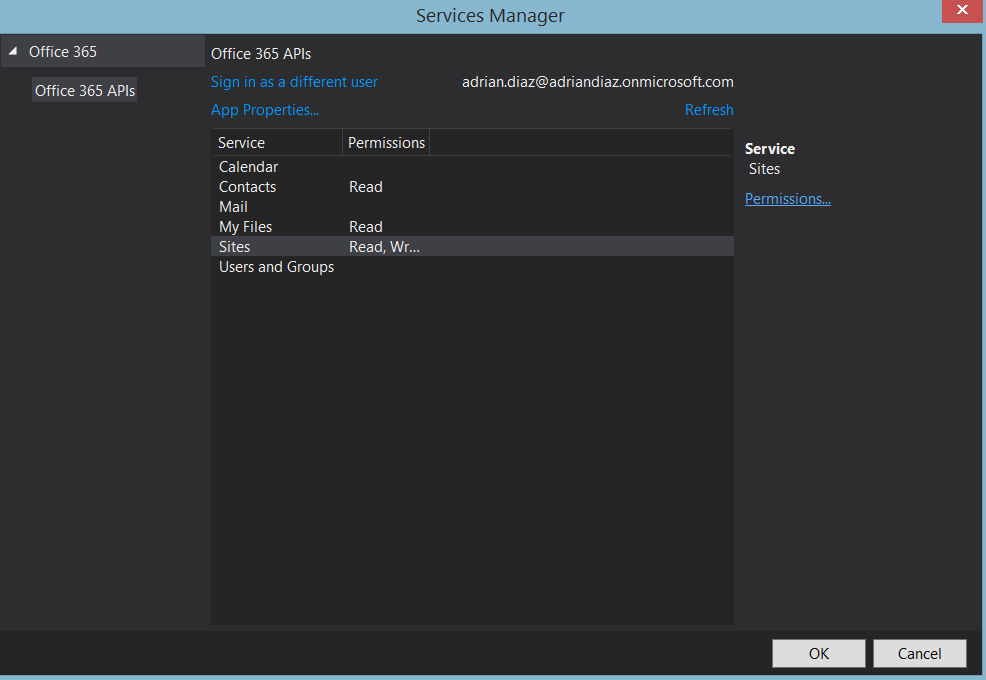
4. Click “App Properties”.

5. Eliminar las redirecciones que usen “http”, manteniendo en su lugar la que utilizan “https”.

6. Click “Apply”



7. Click “OK”



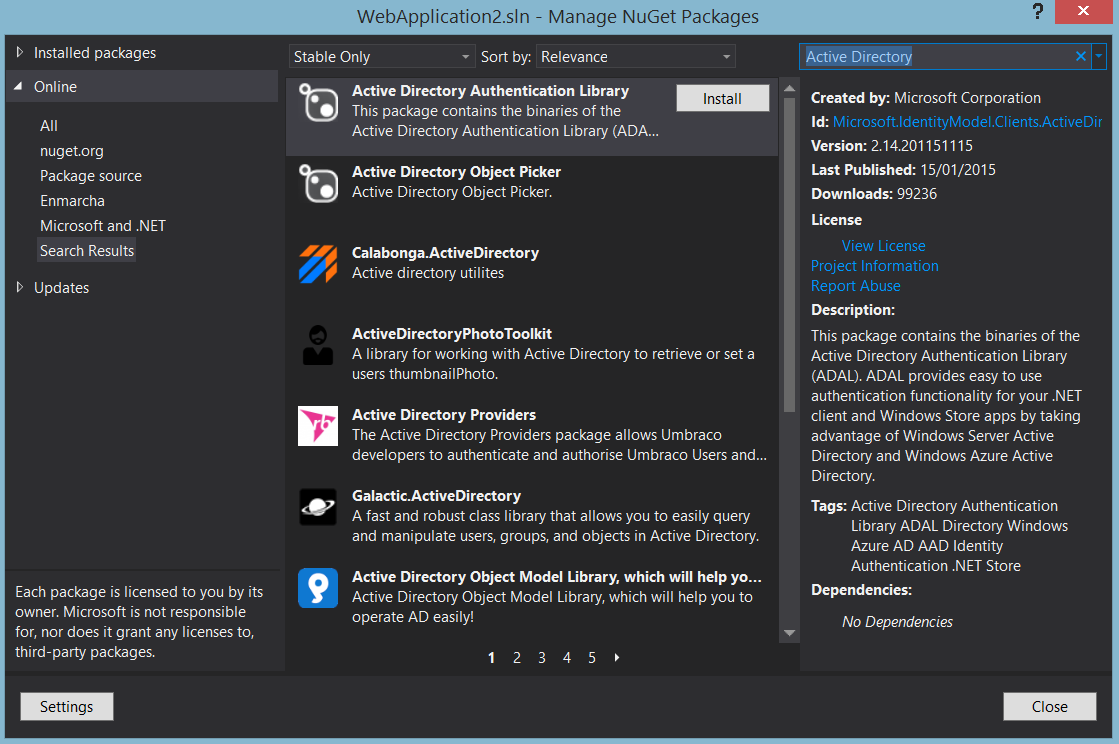
Ahora ya están conectados los servicios en tu aplicación Web.

## Tarea 6 Añadir paquete ADAL

Para realizar la autenticación contra Office 365 existe un paquete Nuget que nos abstree de conocer el mecanismo que se utiliza para realizar dicha autenticación.

Para añadir esta funcionalidad tendremos que realizar los siguientes pasos:

1. Ir a Tools-> Nuget Package Manager -> Manage Package For This Solution
2. Cuando se muestre la pantalla de Nuget, buscar “Active Directory Authentication Library”
3. Seleccionar el primer paquete
4. Pulsar “Install”



## Tarea 7: Añadir Ficheros auxiliaries al Proyecto

Siga los siguientes pasos para añadir unas clases con funciones auxiliares para simplificar el laboratorio.

1. En el “Explorador de Soluiones”, botón derecho sobre la carpeta “Models” y seleccionar “Añadir/Elemento Existente”.

2. Ir a la carpeta “src\Lab Files\Models”.

3. Seleccionar todos los ficheros de la carpeta y pulsar “Añadir”.

4. En el “Explorador de soluciones”, botón derecho sobre la carpeta “Utils”, y seleccionar “Añadir/Elemento Existente”.

5. Ir a la carpeta “src\Lab Files\Utils”.

6. Seleccionar el fichero “Helper.cs” dentro de la capeta y pulsar “Añadir”.

## Tarea 8: Modificar el código del controlador Home

Siguiendo estos pasos, añadirmos los métodos al controlador “Home” que utilizaremos a lo largo del laboratorio.

1. En el “Explorador de Solución”, expander la carpeta “Controladores, seleccionar el “HomeController para editarlo.

2. Reemplazaremos todo el código existente por el siguiente:

```C#

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Microsoft.IdentityModel.Clients.ActiveDirectory;

using Microsoft.Office365.Discovery;

using Microsoft.Office365.OutlookServices;

using System.Configuration;

using System.Threading.Tasks;

using MVCResearchTracker.Utils;

using MVCResearchTracker.Models;

using Microsoft.Office365.SharePoint.CoreServices;

using System.Text;

using System.Xml.Linq;

using System.Net.Http;

using System.Net.Http.Headers;

namespace MVCResearchTracker.Controllers

{

public class HomeController : Controller

{

private const string spSite = "https://[tenancy].sharepoint.com";

private const string discoResource = "https://api.office.com/discovery/";

private const string discoEndpoint = "https://api.office.com/discovery/v1.0/me/";

public async Task<ActionResult> Index(string code)

{

return View();

}

public async Task<ActionResult> Contacts(string code)

{

return View();

}

public async Task<ActionResult> Files(string code)

{

return View();

}

public async Task<ActionResult> Projects(ViewModel submitModel, string code)

{

return View();

}

public ActionResult Finished()

{

return View();

}

}

}

```

3. Reemplazaremos el valor “[tenancy]” con el nombre de nuestro Tenant de SharePoint

Ahora añadiremos el código necesario en los “placeholders” del controlador.

## Tarea 9 Código del Discover Service

Siguiendo los siguientes pasos utilizaremos el servicio de Discovery para mostrar los endpoints para los contactos de Exchange y los ficheros de OneDrive for Bussines.

1. Reemplazaremos el metodo “Index” del “HomeControler por el siguiente código:

```C#

public async Task<ActionResult> Index(string code)

{

AuthenticationContext authContext = new AuthenticationContext(

ConfigurationManager.AppSettings["ida:AuthorizationUri"] + "/common",

true);

ClientCredential creds = new ClientCredential(

ConfigurationManager.AppSettings["ida:ClientID"],

ConfigurationManager.AppSettings["ida:Password"]);

DiscoveryClient disco = Helpers.GetFromCache("DiscoveryClient") as DiscoveryClient;

//Redirect to login page if we do not have an

//authorization code for the Discovery service

if (disco == null && code == null)

{

Uri redirectUri = authContext.GetAuthorizationRequestURL(

discoResource,

creds.ClientId,

new Uri(Request.Url.AbsoluteUri.Split('?')[0]),

UserIdentifier.AnyUser,

string.Empty);

return Redirect(redirectUri.ToString());

}

//Create a DiscoveryClient using the authorization code

if (disco == null && code != null)

{

disco = new DiscoveryClient(new Uri(discoEndpoint), async () =>

{

var authResult = await authContext.AcquireTokenByAuthorizationCodeAsync(

code,

new Uri(Request.Url.AbsoluteUri.Split('?')[0]),

creds);

return authResult.AccessToken;

});

}

//Discover required capabilities

CapabilityDiscoveryResult contactsDisco = await disco.DiscoverCapabilityAsync("Contacts");

CapabilityDiscoveryResult filesDisco = await disco.DiscoverCapabilityAsync("MyFiles");

Helpers.SaveInCache("ContactsDiscoveryResult", contactsDisco);

Helpers.SaveInCache("FilesDiscoveryResult", filesDisco);

List<MyDiscovery> discoveries = new List<MyDiscovery>(){

new MyDiscovery(){

Capability = "Contacts",

EndpointUri = contactsDisco.ServiceEndpointUri.OriginalString,

ResourceId = contactsDisco.ServiceResourceId,

Version = contactsDisco.ServiceApiVersion

},

new MyDiscovery(){

Capability = "My Files",

EndpointUri = filesDisco.ServiceEndpointUri.OriginalString,

ResourceId = filesDisco.ServiceResourceId,

Version = filesDisco.ServiceApiVersion

}

};

return View(discoveries);

}

```

2. En el “Explorador de Soluciones”, expander la carpeta “Views\Home” y abrir el “Index.cshtml” para editarlo.

3. Reemplazar el contenido por el siguiente código :

```HTML

@model IEnumerable<MVCResearchTracker.Models.MyDiscovery>

@{

ViewBag.Title = "Discoveries";

}

<h2>Discoveries</h2>

<table class="table">

<tr>

<th>

@Html.DisplayNameFor(model => model.Capability)

</th>

<th>

@Html.DisplayNameFor(model => model.EndpointUri)

</th>

<th>

@Html.DisplayNameFor(model => model.ResourceId)

</th>

<th>

@Html.DisplayNameFor(model => model.Version)

</th>

<th></th>

</tr>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.Capability)

</td>

<td>

@Html.DisplayFor(modelItem => item.EndpointUri)

</td>

<td>

@Html.DisplayFor(modelItem => item.ResourceId)

</td>

<td>

@Html.DisplayFor(modelItem => item.Version)

</td>

</tr>

}

</table>

<div>

@Html.ActionLink("Get Contacts", "Contacts")

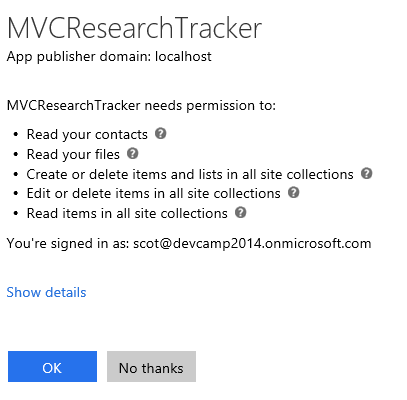
</div>

```

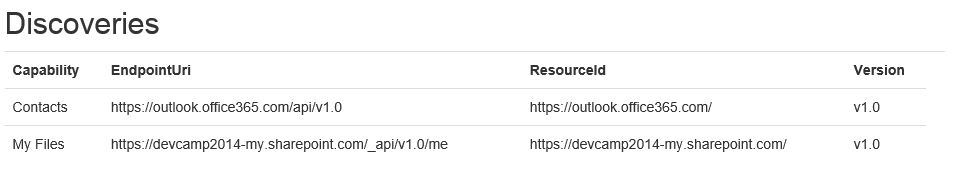
4. Pulsar “F5” para depurar el Proyecto

5. Cuando se muestre la pantalla, introducer las credenciales de Office 365

6. Cuando se muestre, click “Ok” en el siguiente diálogo



7. Se muestra los endpoints para “Contactos” y “Ficheros”



8. Paramos la depuración

Ahora seguiremos completando el código para el Servicio Discovery

## Tarea 10: Obtener los Contactos

Para obtener los contactos de Exchange deberemos de seguir los siguientes pasos

1. Reemplazar el contenido del método “Contactos” en el “HomeController” por el siguiente código

```C#

public async Task<ActionResult> Contacts(string code)

{

AuthenticationContext authContext = new AuthenticationContext(

ConfigurationManager.AppSettings["ida:AuthorizationUri"] + "/common",

true);

ClientCredential creds = new ClientCredential(

ConfigurationManager.AppSettings["ida:ClientID"],

ConfigurationManager.AppSettings["ida:Password"]);

//Get the discovery information that was saved earlier

CapabilityDiscoveryResult cdr = Helpers.GetFromCache("ContactsDiscoveryResult") as CapabilityDiscoveryResult;

//Get a client, if this page was already visited

OutlookServicesClient outlookClient = Helpers.GetFromCache("OutlookClient") as OutlookServicesClient;

//Get an authorization code if needed

if (outlookClient == null && cdr != null && code == null)

{

Uri redirectUri = authContext.GetAuthorizationRequestURL(

cdr.ServiceResourceId,

creds.ClientId,

new Uri(Request.Url.AbsoluteUri.Split('?')[0]),

UserIdentifier.AnyUser,

string.Empty);

return Redirect(redirectUri.ToString());

}

//Create the OutlookServicesClient

if (outlookClient == null && cdr != null && code != null)

{

outlookClient = new OutlookServicesClient(cdr.ServiceEndpointUri, async () =>

{

var authResult = await authContext.AcquireTokenByAuthorizationCodeAsync(

code,

new Uri(Request.Url.AbsoluteUri.Split('?')[0]),

creds);

return authResult.AccessToken;

});

Helpers.SaveInCache("OutlookClient", outlookClient);

}

//Get the contacts

var contactsResults = await outlookClient.Me.Contacts.ExecuteAsync();

List<MyContact> contactList = new List<MyContact>();

foreach (var contact in contactsResults.CurrentPage.OrderBy(c => c.Surname))

{

contactList.Add(new MyContact

{

Id = contact.Id,

GivenName = contact.GivenName,

Surname = contact.Surname,

DisplayName = contact.Surname + ", " + contact.GivenName,

CompanyName = contact.CompanyName,

EmailAddress1 = contact.EmailAddresses.FirstOrDefault().Address,

BusinessPhone1 = contact.BusinessPhones.FirstOrDefault(),

HomePhone1 = contact.HomePhones.FirstOrDefault()

});

}

//Save the contacts

Helpers.SaveInCache("ContactList", contactList);

//Show the contacts

return View(contactList);

}

```

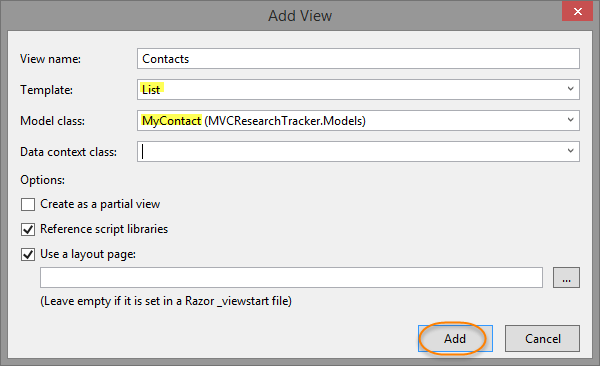
2. Botón derecho, sobre el Objetoo View existente en el método “Contactos” y “Añadir Vista” en el menu contextual.

3. En el Dialogo de “Añadir Vista”:

1. Seleccionar “Lista” como “Template”.

2. Seleccionar “MyContact” como “Model Class”.

3. Click “Añadir”



4. Buscar “Create New” en un “ActionLink” como el siguiente

```HTML

<p>

@Html.ActionLink("Create New", "Create")

</p>

```

5. Modificar el “ActionLink” por el siguiente código:

```HTML

<div>

@Html.ActionLink("Get Files", "Files")

</div>

```

6. Eliminar el siguiente código de la vista:

```HTML

<td>

@Html.ActionLink("Edit", "Edit", new { id=item.Id }) |

@Html.ActionLink("Details", "Details", new { id=item.Id }) |

@Html.ActionLink("Delete", "Delete", new { id=item.Id })

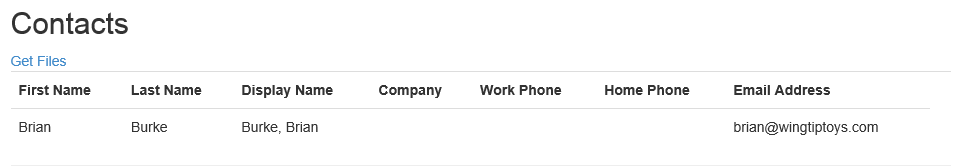
</td>

```

7. Pulsar “F5” para empezar la depuracion.

8. Cuando aparezcan los endpoints, pulsar sobre “Get Contacts”.

9. Visualizaremos los contactos de nuestro Exchange



10. Parar la depuracion

## Tarea 11 Obtener los Ficheros

Para obtener los ficheros existentes en nuestro OneDrive for Bussiness seguiresmos los siguientes pasos:

1. Substituir el código del método “Files” por el siguiente :

```C#

public async Task<ActionResult> Files(string code)

{

AuthenticationContext authContext = new AuthenticationContext(

ConfigurationManager.AppSettings["ida:AuthorizationUri"] + "/common",

true);

ClientCredential creds = new ClientCredential(

ConfigurationManager.AppSettings["ida:ClientID"],

ConfigurationManager.AppSettings["ida:Password"]);

//Get the discovery information that was saved earlier

CapabilityDiscoveryResult cdr = Helpers.GetFromCache("FilesDiscoveryResult") as CapabilityDiscoveryResult;

//Get a client, if this page was already visited

SharePointClient sharepointClient = Helpers.GetFromCache("SharePointClient") as SharePointClient;

//Get an authorization code, if needed

if (sharepointClient == null && cdr != null && code == null)

{

Uri redirectUri = authContext.GetAuthorizationRequestURL(

cdr.ServiceResourceId,

creds.ClientId,

new Uri(Request.Url.AbsoluteUri.Split('?')[0]),

UserIdentifier.AnyUser,

string.Empty);

return Redirect(redirectUri.ToString());

}

//Create the SharePointClient

if (sharepointClient == null && cdr != null && code != null)

{

sharepointClient = new SharePointClient(cdr.ServiceEndpointUri, async () =>

{

var authResult = await authContext.AcquireTokenByAuthorizationCodeAsync(

code,

new Uri(Request.Url.AbsoluteUri.Split('?')[0]),

creds);

return authResult.AccessToken;

});

Helpers.SaveInCache("SharePointClient", sharepointClient);

}

//Get the files

var filesResults = await sharepointClient.Files.ExecuteAsync();

var fileList = new List<MyFile>();

foreach (var file in filesResults.CurrentPage.Where(f => f.Name != "Shared with Everyone").OrderBy(e => e.Name))

{

fileList.Add(new MyFile

{

Id = file.Id,

Name = file.Name,

Url = file.WebUrl

});

}

Helpers.SaveInCache("FileList", fileList);

return View(fileList);

}

```

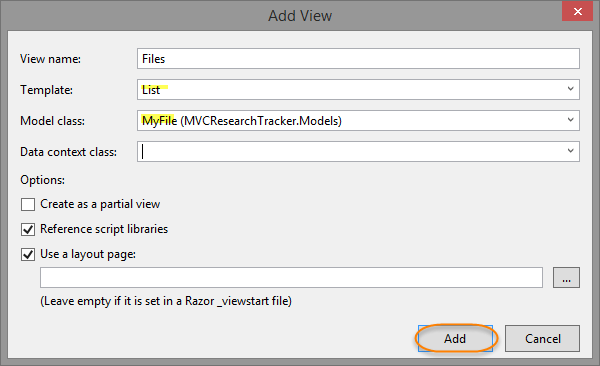
2. Botón derecho sobre el objeto “View” existente en el cuerpo del Método “Files” y seleccionar “Añadir Vista” desde el menu contextual.

3. En el dialogo de “Añadir vista”:

1. Seleccionar “Lista” como “Template”

2. Seleccionar “MyFile” como “Model Class”.

3. Click “Add”.



4. \*Localizar el “ActionLink” de “Create New”:

```HTML

<p>

@Html.ActionLink("Create New", "Create")

</p>

5. Modificar el “ActionLink” :

```HTML

<div>

@Html.ActionLink("Create New Project", "Projects")

</div>

```

6. Eliminar el siguiente código de la Vista:

```HTML

<td>

@Html.ActionLink("Edit", "Edit", new { id=item.Id }) |

@Html.ActionLink("Details", "Details", new { id=item.Id }) |

@Html.ActionLink("Delete", "Delete", new { id=item.Id })

</td>

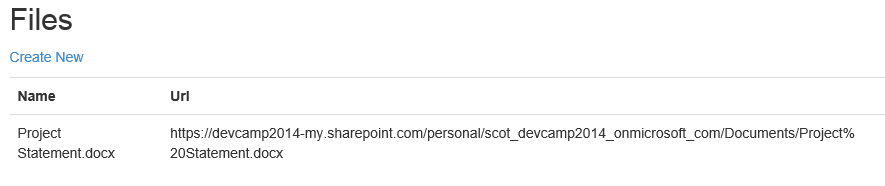
```

7. Pulsar “F5” para empezar la depuración.

8. Cuando aparezcan los endpoint, pulsar “Get Contacts”.

9. Cuando se muestren los contactos, pulsar sobre “Get Files”.

9.Ahora visualizamos los ficheros almacenados en OneDrive form Bussiness



10. Paramos la depuración.

## Crear Nuevo Proyecto

Crearemos un Nuevo Proyecto usando los contactos y ficheros seleccionados, para ello realizaremos los siguienes pasos :

1. Reemplazar el código del método “Projects del “HomeController” por el siguiente código:

```C#

public async Task<ActionResult> Projects(ViewModel submitModel, string code)

{

//If the New Project form needs to be displayed

if (submitModel.Project == null && code == null)

{

ViewModel formModel = new ViewModel();

formModel.Contacts = Helpers.GetFromCache("ContactList") as List<MyContact>;

formModel.Files = Helpers.GetFromCache("FileList") as List<MyFile>;

return View(formModel);

}

// A new project was submitted

else

{

if (submitModel.Project != null)

{

Helpers.SaveInCache("SubmitModel", submitModel);

}

AuthenticationContext authContext = new AuthenticationContext(

ConfigurationManager.AppSettings["ida:AuthorizationUri"] + "/common",

true);

ClientCredential creds = new ClientCredential(

ConfigurationManager.AppSettings["ida:ClientID"],

ConfigurationManager.AppSettings["ida:Password"]);

//Get an authorization code, if necessary

if (code == null)

{

Uri redirectUri = authContext.GetAuthorizationRequestURL(

spSite,

creds.ClientId,

new Uri(Request.Url.AbsoluteUri.Split('?')[0]),

UserIdentifier.AnyUser,

string.Empty);

return Redirect(redirectUri.ToString());

}

else

{

//Get the access token

var authResult = await authContext.AcquireTokenByAuthorizationCodeAsync(

code,

new Uri(Request.Url.AbsoluteUri.Split('?')[0]),

creds);

string accessToken = authResult.AccessToken;

//Build SharePoint RESTful API endpoint for the list items

StringBuilder requestUri = new StringBuilder()

.Append(spSite)

.Append("/\_api/web/lists/getbyTitle('Research Projects')/items");

//Create an XML message with the new project data

//This message will be POSTED to the SharePoint API endpoint

XNamespace atom = "http://www.w3.org/2005/Atom";

XNamespace d = "http://schemas.microsoft.com/ado/2007/08/dataservices";

XNamespace m = "http://schemas.microsoft.com/ado/2007/08/dataservices/metadata";

submitModel = Helpers.GetFromCache("SubmitModel") as ViewModel;

string description = (Helpers.GetFromCache("FileList") as List<MyFile>).Where(f => f.Url == submitModel.Project.DocumentLink).First().Name;

string url = submitModel.Project.DocumentLink;

string title = submitModel.Project.Title;

string owner = submitModel.Project.Owner;

XElement message = new XElement(atom + "entry",

new XAttribute(XNamespace.Xmlns + "d", d),

new XAttribute(XNamespace.Xmlns + "m", m),

new XElement(atom + "category", new XAttribute("term", "SP.Data.Research\_x0020\_ProjectsListItem"), new XAttribute("scheme", "http://schemas.microsoft.com/ado/2007/08/dataservices/scheme")),

new XElement(atom + "content", new XAttribute("type", "application/xml"),

new XElement(m + "properties",

new XElement(d + "Statement", new XAttribute(m + "type", "SP.FieldUrlValue"),

new XElement(d + "Description", description),

new XElement(d + "Url", url)),

new XElement(d + "Title", title),

new XElement(d + "Owner", owner))));

StringContent requestData = new StringContent(message.ToString());

//POST the data to the endpoint

HttpClient client = new HttpClient();

HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Post, requestUri.ToString());

request.Headers.Accept.Add(new MediaTypeWithQualityHeaderValue("application/xml"));

request.Headers.Authorization = new AuthenticationHeaderValue("Bearer", accessToken);

requestData.Headers.ContentType = System.Net.Http.Headers.MediaTypeHeaderValue.Parse("application/atom+xml");

request.Content = requestData;

HttpResponseMessage response = await client.SendAsync(request);

//Show the Finished screen

return RedirectToAction("Finished");

}

}

}

```

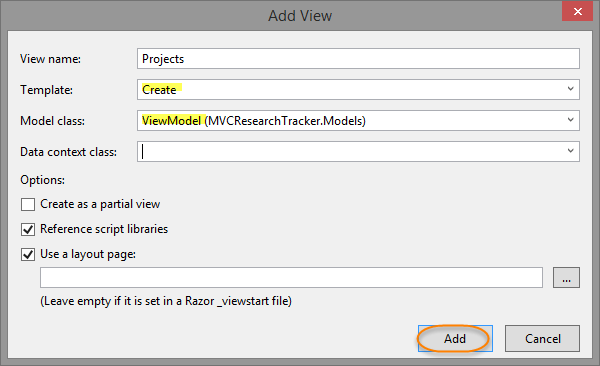
2. Botón derecho dentro del objeto View del cuerpo del método “Projects”, y seleccionamos “Añadir Vista” del menu contextual.

3. En el dialo de “Añadir Vista”:

1. Seleccionar “Crear” como “Template”.

2. Seleccionar “ViewModel” como “Model Class”.

3. Click “Add”



4. Reemplzar todo el código generado por el siguiente código:

```HTML

@model MVCResearchTracker.Models.ViewModel

@{

ViewBag.Title = "Add Project";

}

<h2>Add Project</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>Add Project</h4>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.LabelFor(model => model.Project.Title, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Project.Title, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Project.Title, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(m => m.Project.Owner, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownListFor(m => m.Project.Owner, new SelectList(Model.Contacts, "DisplayName", "DisplayName"), new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(m => m.Project.Owner, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(m => m.Project.DocumentName, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownListFor(m => m.Project.DocumentLink, new SelectList(Model.Files, "Url", "Name"), new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(m => m.Project.DocumentName, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Create" class="btn btn-default" />

</div>

</div>

</div>

}

<div>

@Html.ActionLink("Cancel", "Index")

</div>

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}

```

5. Botón derecho dentro del objeto View dentro del cuerpo del método “Finished” y seleccionar “Add View” del menu contextual.

6. Click “Add”.

7. Reemplazar todo el código de la vista por el siguiente código:

```HTML

@{

ViewBag.Title = "Finished";

}

<h2>Project Added</h2>

<div>

@Html.ActionLink("Start Over", "Index")

</div>

```

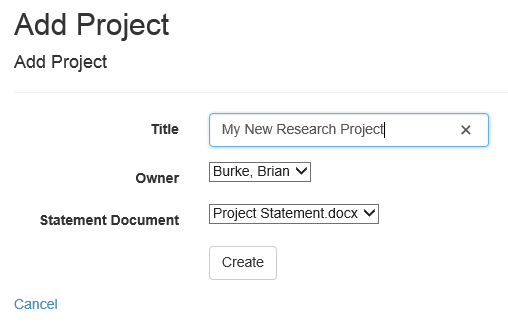
8. Pulsar “F5” para empezar la depuración.

9. Cuando se muestren los endpoint, click “Get Contacts”.

10. Cuando se muestren los contactos, click “Get Files”.

11. Cuando se muestren los ficheros, click “Create New Project”.

12. Releenar todo el formulario de Nuevo Proyecto y pulsar click en “Create”.



13. Ahora se muestra un mensaje indicando que el Proyecto ha sido añadido

14. Parar la depuración.

# Ejercicio 2: Crear Aplicación Apache Cordova

Apache Cordova es un framework para crear apliaciones moviles multiplataforma. Para su desarrollo se basa en la utilización de los estandares Web HTML5, JavaScript y CSS. Lo que proporiciona aplicaciones WebBrowser. La principal ventaja de este tipo de aplicaciones es que todo el código es 100 reutilizable, por contra el rendimiento de estas aplicaciones es menor cuando se quieres aprovechar las capacidades del dispositivo donde reside.

En este ejercicio vamos a utilizer Cordova, y para ello vamos a utilizer uno de los frameworks más populares de JavaScript: AngularJS. AngularJS es un Proyecto realizado por Google cuya principal caracteristica es la extension que realiza del HTML.

## Tarea 1: Crear el Proyecto en Visual Studio

Siguiendo los siguientes pasos crearemos un Nuevo Proyecto Apache Cordova Apps

1. Arrancamos “Visual Studio 2013”.

2. Seleccionamos “Archivo/Nuevo/Proyecto” del menu principal.

3. En el dialogo de “Nuevo Proyecto”:

1. Seleccionamos “JavaScript/ Apache Cordova Apps”.

2. Click “Blank App”.

3. Llamamos al proyecto “SUGESCordova”.

4. Click “OK”.

## Conectamos los servicios de Office 365

1. En el “Explorador de Soluciones”, botón derecho dentro de nuestro Proyecto y seleccionamos la opción “Añadir/Conectar Servicios”.

2. En el cuadro de diálogo de “Service Manager”:

1. Click “Registrar tu App”.

2. Cuando se muestre el cuadro de dialogo, introducir las credenciales de Office 365 para accede al Azure Active Directory .

3. Click “Contacts”.

4. Click “Permissions”.

5. Seleccionar “Read Users' Contacts”.

6. Click “Apply”.

## Agregamos Angular JS a la solución

Para añadir esta funcionalidad tendremos que realizar los siguientes pasos:

1. Ir a Tools-> Nuget Package Manager -> Manage Package For This Solution
2. Cuando se muestre la pantalla de Nuget, buscar “AngularJS”
3. Seleccionar el primer paquete
4. Pulsar “Install”

## Creación del servicio de acceso a las API’s de Office 365

1.- Dentro de Scripts creamos un fichero JS llamado serviceO365.js

2.- Añadiremos el siguiente código:

﻿(function () {

'use strict';

angular.module('app365').factory('app365api', [app365api]);

function app365api() {

var authContext;

var authtoken;

var outlookClient;

var userName;

// Login to O365

function login(callback) {

if (!authContext) {

authContext = new O365Auth.Context();

}

authContext.getIdToken("https://outlook.office365.com/")

.then((function (token) {

// Get auth token.

authtoken = token;

// Get user name from token object.

userName = token.givenName + " " + token.familyName;

// Create Outlook client object.

outlookClient = new Microsoft.OutlookServices.Client('https://outlook.office365.com/api/v1.0', authtoken.getAccessTokenFn('https://outlook.office365.com'));

// Callback without parameter to indicate successful sign-in.

callback();

}).bind(this), function (reason) {

// Log sign-in error message.

console.log('Failed to login. Error = ' + reason.message);

callback(reason.message);

});

};

// Logout

function logout() {

if (!authContext) {

authContext = new O365Auth.Context();

}

authContext.logOut();

};

// Get signed-in user name

function getUserName() {

return userName;

};

return {

login: login,

logout: logout,

getUserName: getUserName,

outlookClientObj: function () { return outlookClient; }

};

};

})();

## Creación de la Pantalla de Login

Toda pantalla en AngularJS esta separada en dos partes, por un lado el renderizado del HTML y por otro lado del HTML, para ello seguriemos los siguientes pasos:

1.- Dentro de la carpeta de JS añadiremos otro fichero llamado sign-in-ctrl.js

2.- Añadiremos el siguiente código:

﻿(function () {

'use strict';

angular.module('app365').controller('signInCtrl', ['$scope', '$location', 'app365api', signInCtrl]);

function signInCtrl($scope, $location , app365api) {

$scope.signIn = function () {

app365api.login(onlogin);

};

var onlogin = function (reason) {

$location.path("/home");

};

}

})();

A continuación creamos el siguiente template html, para ello dentro de la solución nos creamos una carpeta partials, agregamos un fichero html llamado sign-in.html, con el siguiente contenido:

﻿<div class="row row-center" style="height:450px">

<div class="col col-center">

<button class="button button-block button-assertive" ng-click="signIn()">Login</button>

</div>

</div>

## Creación de la Pantalla de mostrar los contactos

Por un lado añadiremos la funcionalidad al controlador de AngularJS para ello seguirmos los siguientes pasos:

1.- Dentro de la carpeta de JS añadiremos otro fichero llamado contact-list-ctrl.js

2.- Añadiremos el siguiente código:

﻿(function () {

'use strict';

angular.module('app365').controller('contactCtrl', ['$scope', 'app365api', contactCtrl]);

function contactCtrl($scope, app365api) {

var vm = this;

var outlookClient;

function getContacts() {

// Fetch all the contacts.

outlookClient.me.contacts.getContacts().fetch()

.then(function (contacts) {

// Get the current page. Use getNextPage() to fetch next set of contacts.

vm.contacts = contacts.currentPage;

$scope.$apply();

});

};

vm.loadList = function () {

// Get the Outlook client object.

outlookClient = app365api.outlookClientObj();

// Get contacts.

getContacts();

};

vm.loadList();

}

})();

A continuación creamos el siguiente template html, para ello dentro de la carpeta “Partials”, agregamos un fichero html llamado contact-list.html, con el siguiente contenido:

﻿<div class="has-header">

<div>

<div ng-repeat="contact in contacts | orderBy: 'displayName'" item-type="item item-icon-left">

{{ contact.displayName }}

</div>

</div>

</div>

## Creación del Inicio de la App

El primer paso crearemos un fichero JavaScript, le llamaremos app.js.

Introduciremos el siguiente código

var app365 = angular.module('app365', [ 'ngRoute']);

app365.config(['$routeProvider',

function ($routeProvider) {

$routeProvider.

when('/', {

templateUrl: 'partials/sign-in.html',

controller: 'signInCtrl'

}).

when('/home', {

templateUrl: 'partials/contact-list.html',

controller: 'contactCtrl'

})

}

]);

A continuación modificaremos la página de inicio, para ello iremos a la página index.html y lo reemplazaremos por el siguiente código:

<!DOCTYPE html>

<html ng-app="app365">

<head>

<meta charset="utf-8" />

<title>Suges </title>

<script src="services/office365/scripts/o365loader.js"></script>

<script src="services/office365/settings/settings.js"></script>

<script src="scripts/index.js"></script>

<script src="scripts/angular.js"></script>

<script src="scripts/angular-route.js"></script>

<script src="scripts/js/app.js"></script>

<script src="scripts/js/serviceO365.js"></script>

<script src="scripts/js/sign-in-ctrl.js"></script>

<script src="scripts/js/contact-list-ctrl.js"></script>

<link href="css/index.css" rel="stylesheet" />

</head>

<body>

<div ng-view></div>

</body>

</html>