**IndignadoFramework**

Gonzalo Castro

elgonzacastro@hotmail.com

Vanessa Revetria

lavarece@hotmail.com

Maximiliano Silvera

maxi.silveraf@gmail.com

Andrés Aldao

andrew5900@hotmail.com

Juan Miraballes

juanma.miraballes@gmail.com

**RESUMEN**

En los últimos años se ha incrementado la cantidad de grupos de protesta, basados inicialmente en el movimiento Occupy Wall Street, el cual, realiza actividades de protesta contra instituciones financieras en New York. Actualmente, estos grupos abarcan diferentes temáticas como por ejemplo protestas sobre el medio ambiente, reivindicaciones sociales y políticas, etc.; y a su vez, se han expandido por muchas partes del mundo.

En este artículo se presenta el diseño e implementación de una plataforma informática que de soporte para este tipo de movimientos donde se permita la gestión, coordinación y organización de las diferentes actividades que estos grupos pueden llevar a cabo.

Para ello, se realizará un sistema desarrollado con tecnologías .NET que constará de dos módulos. Uno de ellos front office, de acceso público y utilizando MVC 3 con Razor; el segundo, un back office para la administración del sistema, de acceso restringido y utilizando Windows Presentation Fundation (WPF).

**Palabras Clave**

SQL, .NET, Windows Comunication Fundation (WCF), Windows Presentation Fundation (WPF), MVC 3, Razor, Movimientos de protesta, Multi-Tenant, ADO (Data Access Object), Sitios (Grupos).

# INTRODUCCIÓN

Tal como se ha mencionado anteriormente, en el último tiempo se han incrementado la cantidad de grupos que sienten la necesidad de protestar sobre temas que resultan de importancia para la sociedad actual. Estos grupos han crecido en número y se han ido extendiendo a lo largo del planeta. Debido a esto, es de suma importancia generar un espacio en la web para todas aquellas personas interesadas, en donde se pueda compartir información de interés común y conocer las diferentes actividades que se organizan por parte de sus integrantes.

El objetivo es desarrollar una plataforma de soporte para estos movimientos de protesta, con la finalidad de centralizar toda la información relevante para los grupos. A su vez, se busca que el sistema pueda servir como infraestructura para la gestión y coordinación de varios de estos movimientos. Cada grupo tendrá a su disposición un único portal web, el cual deberá actuar como punto de integración de toda la información relevante al mismo. El contenido de esta información variará dependiendo del foco temático del grupo como por ejemplo protestas medio ambientales, reivindicaciones estudiantiles, lucha por la igualdad social, etc.

En cada uno de estos sitios web correspondiente a un movimiento de protesta en particular, se podrá compartir información proveniente de diferentes fuentes de datos, como puede ser YouTube, Twitter, BBC, CNN, Wikipedia, etc. Además, se podrá compartir diferentes tipos de recursos como links, imágenes y videos. Desde el portal web de un movimiento se dará la posibilidad de coordinar actividades en donde los integrantes serán informados e invitados a participar y se facilitará la comunicación entre ellos mediante un chat.

Se espera que los sitios web de los distintos movimientos de protesta, tengan una estética diferente uno de otro, contenidos específicos y url diferente. Es decir, a pesar de ser una plataforma común a todos los grupos y compartir una misma lógica, su estética deberá diferenciarse.

En el marco de la edición 2012 de la asignatura Taller de Sistemas de Información 1, se plantea implementar una solución para dicho interés.

Analizado el problema, surgió la necesidad de implementar dos sistemas, un sistema back office de tipo aplicación de escritorio desarrollada con WPF, donde se manejará la gestión del sistema web por parte de un administrador y un front office de tipo aplicación web que será de acceso público.

# MARCO CONCEPTUAL

En esta sección se describen varios conceptos sobre los cuales se deben tener conocimiento para la implementación del sistema. Entre estos conceptos podemos encontrar el de Multi-tenant que utilizará para resolver el problema de los multi-sitios para la aplicación web. Además se utilizarán WCF para la comunicación entre la capa de negocio y presentación (para comprender el concepto de WCF se recomendable tener una noción de web services); también debemos tener en cuenta como concepto importante el de Arquitectura en capas que es fundamental para el desarrollo escalable, desacoplable y mantenible de la plataforma.

A continuación detallamos dichos conceptos:

## Multi-Tenant

Multi-Tenant es un enfoque de un arquitectónico de sistemas de software. En un sistema donde se utilice este enfoque, los diferentes clientes consumen un servicio desde una misma plataforma, un mismo modelo de datos y mismos servidores. Este enfoque se refiere a un principio de arquitectura de software donde una misma instancia del software que se ejecuta en un servidor y da servicio a múltiples clientes simultáneamente, y cada cliente trabaja con una instancia de la aplicación virtual personalizada.

Este enfoque es muy útil para aplicaciones web en cuanto a eficiencia en costos y diseño.

## Web Services

Los web services son funcionalidades que pueden ser utilizadas por una aplicación en particular. Estas funcionalidades pueden ser descriptas, publicadas, localizadas e invocadas a través de una red (generalmente Internet). Una de las características principal de los servicios es que éstos se encuentran en una especie de “caja negra”, lo que nos permite reutilizarlos sin importarnos como fueron implementados. La forma de accederlos es utilizando protocolos web como HTTP y XML, esto permite la interoperabilidad entre las aplicaciones, por lo cual, los consumidores de los web services pueden ser implementados en cualquier plataforma y en cualquier lenguaje de programación.

## WCF

WCF es un conjunto de librerías que provee Microsoft en el framework .NET para aquellas aplicaciones orientadas a servicios. WCF es la evolución de los web services de Microsoft, por esta razón provee un rango de funcionalidades superior a su predecesor. Algunas de las ventajas frente a los web services, es que éstos, sólo pueden ser accedidos a través de HTTP en cambio un WCF se puede hospedar en un servidor web, puede ser un servicio de Windows y utiliza una variedad de protocolos más amplia.

## Inversión de Control (IoC)

La inversión de control es un patrón de programación donde se invierte la técnica tradicional del flujo de una aplicación. Comúnmente es que el código que se implementa el que llama a librerías para su ejecución, por el contrario en la inversión de control, son las librerías las que llaman a nuestro código. Con esto se logra delegar a otro componente (un framework por ejemplo) la responsabilidad de crear las instancias necesarias en lugar de crearlas nosotros mismos.

## Inyección de Dependencias

Este concepto está muy ligado al anterior debido a que en un escenario donde utilizamos IoC, delegamos a otro componente no sólo la creación de las instancias sino también la inyección de las dependencias, o sea, evitamos que nuestros objetos creen instancias nuevas de aquellos objetos de los cuales dependen, proveyéndoselos ya listos para operar (por ejemplo pasándoselos como parámetros al constructor).

## Arquitectura en capas

La arquitectura en capas es un estilo de programación donde su objetivo es separar los diferentes aspectos del desarrollo de una aplicación, como por ejemplo cuestiones de la presentación, lógica de negocio y mecanismos de almacenamiento. Comúnmente la comunicación entre las diferentes capas se da mediante interfaces bien definidas que nos permiten la correcta separación de las mismas. Además, este tipo de diseño permite la instalación distribuida de una aplicación, mejorando su escalabilidad, disponibilidad y utilización de recursos.

## MVC

MVC es un patrón de arquitectónico que separa los diferentes componentes de una aplicación web. Éste patrón divide la aplicacion en tres componentes: Modelo, Vista y Controlador. Esta separación ofrece más control sobre las partes lo cuál permite desarrollarlas, modificarlas y probarlas más fácilmente.

El Modelo contiene la información con la cual operará la aplicación.La Vista gestiona la presentación de la información al usuario. Y el Controlador nos da la respuesta a los eventos invocados desde la Vista comunicándose con la lógica de negocio para recibir una respuesta.

## ADO.NET

ADO.NET es una evolución de la tecnología ADO que provee servicios de acceso a datos relacionales, XML y de aplicaciones, mediante un conjunto de clases.

ADO.NET separa el acceso a datos de la manipulación de los mismos, y crea componentes discretos que se pueden utilizar por separado o conjuntamente. Esta tecnología provee comandos para conectarse a una base de datos y recuperar resultados por medio de clases (Connection, Command, DataAdapter, DataSet, DataReader, entre otros). Además, el objeto DataSet puede utilizarse independientemente de un proveedor de datos .NET para administrar datos locales de una aplicación o que proceden de un origen XML.

## Internet Information Services (IIS)

IIS es un servidor web y un conjunto de servicios de Windows que nos ayuda a desplegar nuestras aplicaciones web a internet. Entre los servicios y protocolos que brinda IIS tenemos: FTP, SMTP, NNTP, HTTP/HTTPS, etc.

Algunas de las ventajas que podemos encontras frente a un servidor Apache es su fácil configuración y administración debido a una muy buena interfaz que permite una excelente experiencia de usuario.

# DESCRIPCIÓN DEL PROBLEMA

La motivación para la creación de esta plataforma reside en el diseño y construcción del un sistema para aquellos movimientos de protesta que desean tener un espacio para compartir información y organizar sus actividades desde un portal web. Se desea generar un framework para gestionar a todos estos grupos con el fin de centralizar una lógica de negocio que pueda servir de soporte para los todos los sitios de protesta.

Cada movimiento tendrá su propio portal web los cuales serán completamente diferentes desde el punto de vista del diseño, tendrán url distinta y mostrarán contenidos específicos dependiendo de la temática a la cual estén relacionados.

El sistema deberá contar con dos módulos: uno de front office de acceso público que será el encargado de mostrar los sitios al usuario y otro de back office de acceso privado encargado de la administración de los datos de la plataforma. Entre las funcionalidades que debe brindar el back se encuentra, gestión de usuarios, gestión de los grupos de protesta, gestión de variables del sistema (más adelante se explicará con más detalle esta funcionalidad), y creación de reportes. El front deberá permitir el registro de usuarios, la creación de convocatorias (actividades que el movimiento realizará) y visualización de recursos compartidos como videos, imágenes, links y novedades. Este módulo también contará con un chat que permitirá la comunicación entre los usuarios de un grupo.

Además, se plantean las siguientes funcionalidades opcionales: integración con Facebook y Twitter, integración entre servidores, desarrollo de un cliente Windows Phone, detección de dispositivos móviles y deploy en Windows Azure.

Para el desarrollo de la plataforma se exige el framework .NET 4.0, con todas sus tecnologías integradas.

# SOLUCIÓN PLANTEADA

La solución al problema plateado se ha llevado a cabo durante el proceso de análisis al comienzo del proyecto, y se ha ido modificando en el correr del desarrollo debido a que se ha ido adaptando a las diferentes necesidades y a los problemas encontrados durante la implementación.

Además de los ya conocidos módulos a implementar (back office y front office), en esta sección se explicará las decisiones que se han tomado para desarrollar el sistema, los casos de uso que se necesitan para cubrir todos los requerimientos y las soluciones que se han tomado al momento de enfrentarse a los diferentes problemas de implementación.

Para desarrollar el back office se ha utilizado la tecnología WPF. Se ha elegido utilizar una aplicación de escritorio bajo el argumento de manejar otro tipo de tecnología diferente a la que se va a utilizar en el front. El objetivo de esto, es adquirir un conocimiento más amplio del framework .NET ya que de la forma planteada inicialmente no se llegaría ver esta tecnología. Desde el punto de vista funcional el back office consta de un menú el cual tiene habilitado las siguientes funcionalidades para usuario específicos (administradores) previo inicio de sesión: Gestión de Administradores, Gestión de Usuarios, Gestión de Grupos, Gestión de Variables, Gestión de Reportes y el Watchdog.

En el caso del módulo front office se ha desarrollado una aplicación web para cumplir con todos los requerimientos. Entre ellos se encuentra el Home del sitio, aquí el usuario no registrado podrá ver información pública, pero para tener acceso a las demás funcionalidades, deberá registrarse para después loguearse y desde ese punto sí, poder acceder a otras secciones de la web y ser parte del movimiento de protesta.

Como anexo a la plataforma se han incluido algunos casos opcionales como la integración entre grupos (en nuestro caso lograda con otra instancia de nuestra misma aplicación), un cliente Windows Phone y Login con Facebook.

***Figura 1:*** *Detalle de casos de uso (Ver Archivo Adjunto).*

## Front Office

***Home del sitio***: Desplegará todas las opciones detalladas a continuación aunque algunas de las opciones sólo están habilitadas para usuarios registrados en el sitio. Existe una vista pública la cual un usuario no registrado tiene acceso. En esta vista el usuario (NO registrado) podrá registrarse y ver aquellos contenidos que fueron publicados previamente y catalogados como contenido de acceso público.

***Login***: Esta funcionalidad está habilitada de manera pública y sólo los usuarios registrados previamente podrán acceder al sitio y visualizar todas las funcionalidades. Desde esta vista los usuarios se podrán registrar debido a un botón existente que lo redirecciona a la página de registro correspondiente.

***Chat*:** Funcionalidad que permite a los usuarios regsitrados chatear siempre y cuando esten conectados y activos.

***Crear Convocatoria***: En esta sección se podrá ingresar todos los datos necesarios para crear una convocatoria. La convocatoria será visible para todos los usuarios registrados, donde ellos podrán marcar su asistencia. Se deriva caso de uso Enviar notificaciones.

***Novedades y recursos compartidos***: Se detallarán todos los recursos compartidos por los usuarios registrados al sitio filtrados por cantidad de recursos más rankeados. El recurso puede tener la visibilidad habilitada para todos los usuarios o sólo para usuarios registrados (PUBLICO, PRIVADO). Los tipos de contenido que pueden compartir son Imágenes, Videos y Links.

***Aceptar Convocatoria***: El usuario podrá marcar que asisitira a las convocatorias deseadas. Quedará registrado que este asistira.

***Etiquetar Contenido***: Toda información publicada podrá ser etiquetada, de manera positiva, marcando “Me gusta” o de manera negativo, marcando “Inadecuado”. Estos valores son guardados, de tal manera de poder llevar un conteo de cuantas marcas recibio cada contenido.

***Ver información pública del sitio***: Se listará a los usuarios no registrados la información del sitio que fue publicada con visibilidad habilitada para todos los usuarios. Para hacer esta diferencia se guarda en la tabla de los contenidos un valor PUBLICO o PRIVADO que el usuario marcará al publicar el contenido.

## Back Office

***Crear nuevo sitio de protesta***: Se creará un nuevo sitio de protesta, donde se deberán ingresar campos obligatorios como el Nombre, descripción, url, cooredenadas del sitio, la tematica y por último se define el layout para seleccionar el diseño de cada sitio. Opcionalmente se podrá seleccionar las fuentes de datos habilitadas del sitio y el id para poder loguearse con facebook.

***Gestión de sitios***: Se listarán todos los sitios, donde se podrá navegar a cada sitio y editarlo en caso de desearlo.

***Crear* *Reportes***: Se podrán crear dos reportes para obtener diferentes estadísticas. Estos son: Reporte de usuario registrados en el tiempo, donde listara usuarios registrados por sitio y por mes. Y reporte de contenidos inadecuados donde listara por sitio la cantidad de contenidos que tiene con marcas de inadecuado.

***Gestión de variables***: Habran 4 variables que controlarán el sitio. Las variables podrán ser editadas aquí. Estas variables son: La cantidad de contenidos que se deberán listar en la home del sitio, la cantidad de contenidos mas rankeados que se muestran en la home del sitio, a partir de cuantas marcas de inadecuado se da de baja un contenido y por último, a partir de cuantos contenidos marcados como inadecuados se da de baja un usuario.

***Alta Administrado***r: Se permitirá gestionar administradores, dando de alta los administradores deseados.

***Gestión de usuarios***: Se listarán los usuarios por sitio.

***Watchdog***: Conta de dos funcionalidades: mostrar todos los contenidos a borrar y mostrar todos los usuarios a borrar. Estas listas se forman a partir de la gestión de variables.

***Figura 2:*** *Vista de SubSistemas(Ver Archivo Adjunto).*

***Soluciones Especiales***

**Multi-Tenant**

La implementación del patrón multitenant se realizó utilizando inversión de control e inyección de dependencias. Este patrón arquitectónico nos permitió resolver el problema de los multisitios de manera muy eficiente y performante.

**Integración con Grupos**

Debio a cierto contexto dado durante el proceso de desarrollo, y en especial con los otros grupos, se decidió hacer la integración entre servidores tomando otra instancia de nuestra misma aplicación. Para esto se generó una nueva base de datos para el servidor ha integrar, así como tambien se generó una nueva intancia de la aplicación con su correspondiente configuración en el IIS.

El funcionamiento de la integración está dado por un servcio de una aplicación que expone operaciones (ServiceExpose) y un módulo de la otra aplicación que consume los servicios expuestos por la primera (InG4Agent). De esta forma la aplicación que consume los servicios expuestos tiene a su disposición los datos para manipularlos y mostrarlos al usuario.

Estos datos obtenidos desde la otra aplicación, serán recuperados y “mergeados” por los managers y enviados a la capa de presentación mediante servicios web (WCF).

**Filtro de Busqueda de Convocatorias**

Una funcionalidad que se agregó al FrontOffice para comodidad de los usuario, fue un filtro de búsqueda de convocatorias en donde se pueden buscar las convocatorias de ese movimiento de protesta. Este filtro consta de varios campos por los que se puede buscar, alguno de ellos son Titulo de convocatorias, Descripción, Fecha Inicio, Fecha Fin, etc.

El filtro se desarrolló mediante una búsqueda dinámica en la capa de acceso a datos con la ayuda de una clase filtroBúsqueda que contiene los datos ingresados por el usuario en el filtro.

**Notificaciones**

Uno de los detalles que aportan al diseño y a la experiencia de usuario, es la notificación a un usuario de que se ha creado una convocatoria a menos de 50 kilómentros. Esta notificación es un pequeño número que aparece de manera intermitente en rojo, indicando las convocatorias pendientes o no leídas (siempre las que están a un radio menor a 50 km). El usuario posteriormente podrá marcar aquellas que ha leído.

**Ver Mapa**

Para darle una mayor comodidad al usuario, se creó un tab con un mapa WideScreen para ver todas las convocatorias que existen de ese movimiento. Este mapa muestra no sólo las convocatorias a menos de 50 kilómentros de usuario sino todas las que existan en el sitio, por esto último fue la decisión de hacerlo de esa manera.

**Editar Perfil**

Unas de las funcionalidades agregadas a la página fue la edición de los datos del usuario. Esto da la posibilidad a un usuario logueado y registrado poder editar sus datos en un futuro, como por ejemplo su contraseña.

**Encriptación de contraseñas**

La seguridad es algo muy importante en un sitio web, por esta razón las contraseñas de los usuarios se guardan encriptadas en la base de datos. Pero un punto muy importante es que viajan encriptadas desde capa de presentación, o sea que pasan por los servicios web de esta forma y no en su forma original.

# ARQUITECTURA DEL SISTEMA

***Figura 3:*** *Arquitectura del sistema (Ver Archivo Adjunto).*

La arquitectura utilizada para el desarrollo del sistema es una arquitectura clásica de 3 capas, dónde las capas superiores consumen de las de más abajo. Además se ha desarrollado una capa lateral que será referenciada en todas las capas horizontales.

Debido al diseño de la arquitectura se ha logrado un sistema mantenible que da la posibilidad de agregar nuevas funcionalidades sin tener que modificar el código ya escrito.

## PRESENTATION LAYER

Capa encargada de mostrar los datos al usuario final. Esta capa consta de otros módulos con el fin de obtener un sistema desacoplado, escalable y que cumpla con las funcionalidades requeridas.

### IndignaFwk.UI.FrontOffice

Es el módulo encargado de mostrar los datos al usuario final y se comunica directamente con el UI.Process. Se ha creado un proyecto web para implementarlo.

### IndignaFwk.UI.BackOffice

Aquí se implementa el sistema para que los administradores gestionen los movimientos de protesta utilizando la tecnología WPF. Este módulo también se comunica directamente con el UI.Process.

### IndignaFwk.UI.Process

Este módulo consume los servicios expuestos en la capa de negocio y es referenciado por IndignaFwk.UI.BackOffice e IndignaFwk.UI.FrontOffice.

## BUSINESS LAYER

Capa que contiene las reglas de negocio de la aplicación. Se comunica con la capa de persistencia mediante una interfaz definida en los ADO.

### IndignaFwk.Business.Services

Aquí están ubicados los servicios, para los cuales se ha utilizado WCF, que exponen funcionalidades desde la capa de negocio hacia el exterior de la capa. Estos servicios serán consumidos por la capa de presentación, en particular por el por el módulo UI.Process.

### IndignaFwk.Business.Managers

En este módulo se resuelve la parte de la lógica de negocio del framework. Se han creado diferentes managers para una correcta separación de las funcionalidades y una factoryManager para un mejor manejo de los mismos.

### IndignaFwk.Business.Agents

Este módulo se utilizará para la integración con otros grupos y para la integración de fuentes de datos.

# PERSISTENCE LAYER

Capa de acceso a datos. Esta capa se conecta directamente con la base de datos y es la encargada de crear, editar, borrar y obtener elementos de la base mediate objetos ADO (Data Access Object).

### IndignaFwk.Persistence.DataAccess

Módulo dentro de la capa de acceso a datos con los ADOs y sus correspondientes interfaces para la correcta comunicación con la capa de negocios; y por supuesto con la base de datos.

### Common Layer (capa vertical)

Esta es una capa lateral que será utilizada por todas las demás capas.

### IndignaFwk.Common

Módulo de la capa lateral que contiene las entidades de negocio, enumerados y funcionalidades utilitarias y de seguridad.

# IMPLEMENTACIÓN

- Se utiliza el IDE de Microsoft Visual Studio 2010 profesional SP1, integrado con el sistema de repositorio de versionado *SVN* Ankhsvn.

- Para la administración de la base de datos se utiliza Microsoft SQL Server 2008, con el IDE de SQL Server Management Studio.

- Para el acceso a datos se utilizó ADO.NET.

- Para la comunicación entre la capa de presentación y la lógica de negocio se utilizó WCF.

- En el módulo FrontOffice se utilizó el patrón MVC con tecnologías ASP MVC 3 y las vistas Razor. Además, para resolver diferentes problemas y desarrollar funcionalidades específicas en la presentación, se utilizó ajax, jquery, estilos CSS, APIs de GoogleMaps, Youtube y Wikipedia.

- Se utilizó el servidor Web IIS 7.5 para hacer el deploy de la aplicación web y los servicios.

- Para el módulo BackOffice se utilizó Windows Presentation Fundation (WPF) realizando una aplicación de escritorio para la administración de los movimientos de protesta.

- Se utilizó el cliente de repositorio *SVN* Tortoise para la documentación y material en general.

## Productos y Herramientas

Tabla 1. Evaluación de Productos y Herramientas

|  |  |  |  |
| --- | --- | --- | --- |
| **Producto** | **Puntos Fuertes** | **Puntos Débiles** | **Evaluación General** |
| ADO.NET Core | Mayor control de datos.  Facilidad en la manipulación de datos. | Reducción de la reusabilidad.  La modificación de la base de datos implica la creación de un nuevo modelo. | Aceptable |
| WCF | Utilización de variedad de protocolos. | No es viable para aplicaciones sencillas, requiere mayor potencia de hw. | Muy buena |
| MVC 3 | Diseño claro. Facilita la implementación. | Ceñirse a una estructura predefinida, a veces puede incrementar la complejidad del sistema..  Curva de aprendizaje elevada para los nuevos desarrolladores.  Distribución de componentes obliga a crear y mantener un mayor número de ficheros. | Buena |
| Razor | Sintaxis compacta, fácil de escribir y minimiza el tipeo de código. | Esta tecnología utilizada por si sola no es suficientemente potencial. | Aceptable |
| Api Facebook | Disponible públicamente, fácil acceso a material. | Se precisa un id por cada sitio que vaya a utilizar el servicio de login. | Buena |
| API Google Maps | Disponible públicamente.  API muy completa . | Con bajos requerimientos de hw se vuelve un poco lento el desplazamiento y visualización. | Excelente |
| WPF | Mejor experiencia de usuario que Windows Forms. | Se debe ejecutar un instalador | Buena |
| CSS | Diseño desacoplado del resto del código.  Posibilidad de armar diferentes presentaciones parauna misma página. | Dependiendo del navegador utilizado, la página maquetada con CSS puede verse distinta e incluso verse mal. | Muy Buena |
| JQuery | Genera contenido dinámico a nuestras páginas de manera sencilla.  Ahorra líneas de código. | Incompatibilidad con algunos Navegadores. | Excelente |
| Ajax | Mejor experiencia de usuario.  Reducción de tráfico de datos al servidor. | Incompatibilidad con algunos Navegadores. | Muy Buena |
| IIS 7.5 | Pools independientes.  Interfaz de administración.  Fácil configuración.  Facilita el debug al  “atachearse” al proceso. | No es multiplataforma. | Muy Buena |
| SQL Server 2008 | Seguridad en el manejo y almacenamiento de datos. | Licenciado,nescesita de altos requerimientos de hw comparado con otros gestores de BD. | Buena |
| SQL Managment | Interfaz que facilita la administración de datos. | Elevado costo de licencia. | Buena |
| Windows Phone | Fácil de desarrollar debido a controles predeterminados (Pivot y Panorama). | Altos requerimientos de hw. | Aceptable. |
| AnkhSvn | Integrado a Visual Studio. |  | Buena |
| Tortoise | Fácil manejo, intuitivo.  Posibilidad de subir cualquier tipo de archivo.  Usuado para subir todos los docuementos del laboratorio. | No nos fue útil como cliente de SVN para nuestro código. | Buena |
| Visual Studio 2010 | Rico en la experiencia de usuario, fácil integración con tecnologías compatibles. | Licenciado,al desarrollar una aplicación de gran porte se precisan buenos requerimientos de hw para lograr trabajar fluidamente | Muy Buena |

***Calificaciones de la evaluación general por orden ascendente en lo que respecta a su valor:*** *Mala, Aceptable, Buena, Muy Buena, Excelente.*

## Problemas Encontrados

En el desarrollo de la solución nos presentamos con varias incognitas y varios problemas. Por no tener conocimientos sobre las tecnologías a usar, primero tuvimos que dedicarle tiempo a la investigación. Luego de la investigación es donde surgen los problemas, a continuación los detallamos:

- Login con facebook: el problema que encontramos con respecto a esta funcionalidad es que las políticas de Facebook requieren de un id por cada url de una web que utilice el servicio de login. Para esto, en el BackOffice se implementó un campo que guarda el id que brinda Facebook, permitiendo que el administrador del sitio elija si quiere que ese grupo se logue con esta aplicación o no.

- Otro de los problemas encontrados fue que al cerrar la sesión de forma abrupta un usuario conectado no se deslogueaba. Para resolver esto se utilizó el timeout de la sesión para que después de cierto tiempo el usuario no esté más conectado.

- Debido a la cantidad de tecnologías que se debian investigar se generó un atraso en la implementación de las mismas.

# EVALUACIÓN DE LA SOLUCIÓN

## Puntos Fuertes

**-** Arquitectura diseñada evaluando futuros riesgos permitiendonos desarrollar un sistema mantenible y escalable.

**-** Resolución de multitenant con el patrón Inversion de control e Inyección de dependencias permitiendo multiplicidad de vistas para los sitios web.

- Integración con otros grupos. Debido a la arquitectura diseñada, la implementación de esta funcionalidad no implicó grandes costos ni grandes cambios en el sistema.

- Integración con fuentes de datos (YouTube y Wikipedia).

- Encriptación de contraseñas (Algoritmo Rijndae). Permite que las contraseñas de los usuarios viajen por los WCF encriptadas. Aumentando la seguridad de la información de los usuarios.

-Comunicación entre capa de presentación y capa de negocio mediante WCF.

## Puntos Débiles

**-** El diseño de las intefaces en WPF no fue el deseado, luciendo unas intefaces poco amigables con el usuario.

- Poca funcionalidades desarrolladas en Windows Phone de nuestro sistema.

- Faltó hacer el deploy en Azure.

- Otras de las debilidades que tiene la aplicación es en el tema de la seguridad, ya que no se ha implementado ningun módulo para manejar este aspecto.

# DESARROLLO DEL PROYECTO

El desarrollo del proyecto se inicio el 21 de abril de 2012. Se arranco diseñando los casos de uso, arquitectura, un modelo relacional y diagrama de clases para definir el sistema que se implementaria.

Luego de tener los documentos aprobados empezamos a desarrollar por un lado las entidades, los Managers, los ADOs de cada entidad, la bases de datos, los servicios, en definitiva todo el backend. Mientras que en paralelo se investigaba la implementación de back office con el uso de WPF y la implementación del front office con el uso de MVC 3 con vistas razor. Luego de tener el codigo base, se avanzo en paralelo en todas los sectores detallados. Por último se desarrollo la aplicaciónde Windows Phone

Semana del 20 de abril: Estudio de requerimientos, diseño casos de uso y arquitectura (2 Semanas).

Semana del 7 de mayo: Se inicia el desarrollo de backend, creación del primer schema de base de datos y diseño de pantallas de back office (3 Semanas).

Semana del 15 de mayo: Implementación de multitenant, desarrollo del prototipo y comunicación con WCF (1 Semana).

Semana del 29 de mayo : Se comienza a escribir el artículo (Proceso hasta el final).

Semana del 6 de junio: Desarrollo de caso de usos obligatorios.

Semana de 14 de junio: Configuración de Servidor IIS (1 Semana).

Semana del 21 de junio: Login con Facebook, integración con grupos, integración con fuentes de datos y desarrollo de Windows Phone (2 Semanas).

Semana del 27 de junio: Testing, mantenimiento correctivo y perfectivo (1 semanas).

# CONCLUSIONES Y TRABAJO A FUTURO

Las primeras conclusiones que se puden sacar son muy positivas, ya que se pudo desarrollar una plataforma pontente y muy util para crear sitios web en un tiempo muy reducido. En tan sólo 2 meses y medio se logró la creación de 12 sitios con diferentes estilos que posteriormente se bajaron a 4 para profundizar en el diseño de los mismos.

Debido a las caracteríticas y el objetivo de los sitos de los movimientos de protesta, el trabajo a futuro esta dado una una mayor y mejor integración con otros grupos y fuentes de datos, ya que unos de los objetivos principales de estos grupos es poder compartir información. Esto brindará más riqueza a los portales webs y será posible lograrlo sin mayores dificultades gracias a la arquitectura planteada desde el comienzo.

A su vez, se deberán fortalecer los puntos débiles que actualmente tiene la aplicación nombrados anteriormente, especialmente se deberá mejorar el back office desde el punto de vista del diseño para lograr una mejor experiencia de usuario.

En general se deberá hacer mantenimiento correctivo, y más que nada perfectivo para lograr páginas más dinámicas y con mayor interacción por parte de los usuarios.

Una de las funcionalidades muy importate a implementar es la de los comentarios de los usuarios en los contenidos compartidos y las convocatorias. De esta forma lograremos una integración más rica entre los usuarios.

En general las mejoras de la aplicación están dadas en su mayoría por mejoras de funcionalidades ya terminadas y no tanto por agregar nuevas.

# REFERENCIAS

1. Laboratorio de Integración de Sistemas. Taller de Sistemas de Información 1 – Trabajo Obligatorio. Año 2011. <http://www.fing.edu.uy/inco/cursos/tsi/TSI1/2011/laboratorio/Obligatorio2011.pdf> [Consulta: Mayo 2011]
2. Microsoft .NET Framework

<http://www.microsoft.com/net/> - [Consulta: Mayo 2010]

1. ACM Proceedings Templates  
   <http://www.acm.org/chapters/policy/toolkit/template.html>   
   [Consulta: Mayo 2010]