|  |  |  |
| --- | --- | --- |
| VSDSL | junio 29  2012 | |
| PFC por Mario A. Corchero Jiménez | | VSDSL: un DSL para la generación automática de aplicaciones de video-vigilancia |

Contenido

[Introducción 3](#_Toc328499751)

[Aplicación Servidor 3](#_Toc328499752)

[Cliente 3](#_Toc328499753)

[Análisis de Tecnologías 4](#_Toc328499754)

[General 4](#_Toc328499755)

[Cadena de Eventos 9](#_Toc328499756)

[Servicios Web 9](#_Toc328499757)

[WebSite 11](#_Toc328499758)

[Comunicación Servidor-Cliente 12](#_Toc328499759)

[Modelado 20](#_Toc328499760)

[Decisiones de diseño 22](#_Toc328499761)

[Gestor de eventos 22](#_Toc328499762)

[Aplicación servidor 22](#_Toc328499763)

[Clientes 22](#_Toc328499764)

[Servidor web 22](#_Toc328499765)

[Modelado 23](#_Toc328499766)

[Gestor de eventos 29](#_Toc328499767)

[Visión general 30](#_Toc328499768)

[Ejemplo 31](#_Toc328499769)

[Casos de uso 31](#_Toc328499770)

[Diagrama de clases 32](#_Toc328499771)

[Jerarquía de eventos 33](#_Toc328499772)

[Servicios Disponibles 33](#_Toc328499773)

[Implementar un nuevo servicio 36](#_Toc328499774)

[Servidor 38](#_Toc328499775)

[Visión General 38](#_Toc328499776)

[Estructura del proyecto 39](#_Toc328499777)

[Servicios Web 40](#_Toc328499778)

[Base de datos 44](#_Toc328499779)

[Archivo de configuración 44](#_Toc328499780)

[Interfaz de usuario 45](#_Toc328499781)

[Usuarios y Roles 47](#_Toc328499782)

[Diseño 48](#_Toc328499783)

[Configuración de la base de datos 49](#_Toc328499784)

[Instalación 50](#_Toc328499785)

[Servidor Web 51](#_Toc328499786)

[Visión General 51](#_Toc328499787)

[Seguridad 51](#_Toc328499788)

[Estructura del proyecto 52](#_Toc328499789)

[Árbol de navegación 52](#_Toc328499790)

[Descripción de las páginas 53](#_Toc328499791)

[Script 56](#_Toc328499792)

[Notificaciones desde el servidor 56](#_Toc328499793)

[Conexión a las cámaras 56](#_Toc328499794)

[Soporte 56](#_Toc328499795)

[Instalación 57](#_Toc328499796)

[Referencias 60](#_Toc328499797)

# Introducción

Este documento presenta la documentación técnica relativa a al PFC “VSDSL: un DSL para la generación automática de aplicaciones de video-vigilancia”.

La idea general es el desarrollo de un sistema de video vigilancia que permita ver el estado de los dispositivos, y recibir y manejar eventos de estos a través de una serie de servicios configurables. La configuración se realizara principalmente a través de un DSL que debe permitir desde configurar un servicio en concreto a añadir servicios nuevos desarrollados por el usuario final, pasando por generar una cadena de eventos que proporcionen la funcionalidad deseada.

El proyecto se divide en:

* Modelado y generación del código.
* Aplicación servidor.
* Gestor de eventos.
* Cliente

El gestor de eventos forma parte de la aplicación servidor pero se trata como una parte debido a su importancia.

## Aplicación Servidor

Esta pretende ser una aplicación servidor que por una parte controle y procese los eventos recibidos según se ha indicado en el DSL con el que se generó la aplicación y por otra proporcione servicios para los clientes que se conecten. Es por ello que la aplicación se puede dividir en la parte relativa al trato de eventos y en el uso de clientes para conectarse al servidor.

La aplicación debe ser completamente escalable y flexible, de forma que sea más versátil a la hora de generar código para añadir nuevas funcionalidades.

## Cliente

El cliente es de tipo web, es decir, el sistema dispone de un servidor web montado sobre IIS, un WebSite que carga toda su configuración e información a través de los servicios web expuestos por el servidor implementado en C#. Este WebSite prácticamente no contiene ninguna información de forma que al generar el código solo será necesario modificar la configuración del servidor, la cual la será expuesta a través de los servicios web, siendo mostrada en el cliente web.

# Análisis de Tecnologías

## General

### Conexión a la base de datos

Nuestro sistema utiliza dos tipos de conectores de bases de datos actualmente, fácilmente ampliable a mas. Los conectores elegidos son:

#### Driver PostgreSQL

Este driver se ha elegido específicamente debido a que se recomienda el uso del gestor de bases de datos PostgreSQL, el cual se ha utilizado a lo largo de las pruebas. Asi, este driver ofrece una respuesta mucho más eficiente que el driver ODBC.

#### Windows ODBC

Windows Open Database Conectivity es una API de Windows que permite acceder a una base de datos utilizando los Orígenes de Datos de Windows. Esto añade una capa más de abstracción, definiendo la conexión a la base de datos desde una herramienta de Windows ODBCad. Así, en la aplicación solo se indica el nombre de la entrada mientras que en Windows se crea una entrada con todos los parámetros de conexión. Una de las ventajas de esta tecnología es que permite conectarse a prácticamente cualquier base de datos, puesto que además de las soportadas por defecto por Microsoft, podemos encontrar instaladores de drivers ODBC para el resto de gestores.

De esta forma, podremos separar totalmente la conexión de la base de datos de la aplicación, permitiendo que un cambio no solo de ubicación de la base de datos, sino del gestor de base de datos que se utilice repercuta únicamente en cambiar una línea desde un asistente en Windows.

### C#

El lenguaje C# es un leguaje relativamente moderno (2001) que en la plataforma .Net y la herramienta de desarrollo Visual Studio hacen de la codificación una tarea mas llevadera. La ultima versión de C#, 4.0, añade aun más facilidades para el programador y es que aun que sea un lenguaje Orientado a Objetos también es declarativo y funcional permitiendo declarar funciones lambda por ejemplo, muy útiles en determinados momentos, así como LINQ, un lenguaje de consulta nativo en el lenguaje. De esta forma, junto con otras funcionalidades como las Propiedades, delegados, sobrecarga de operadores, métodos agregados, etc… hacen a C# un lenguaje cómodo de utilizar a la vez que potente, aunque no este a la altura en eficiencia de C o C++[1]. Debido a esto se ha decido utilizar C# para la implementación del servidor del sistema puesto que además, tecnologías como WCF y WPF, explicadas a continuación, pueden ser utilizadas para flexibilizar la implementación de algunas características del servidor.

### WPF

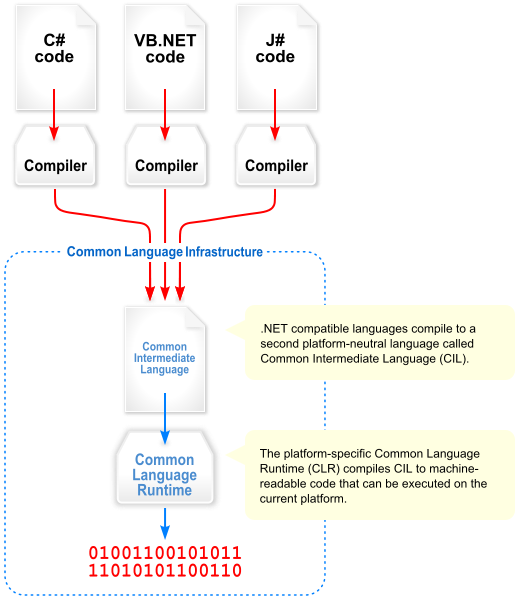
Windows Presentation Foundation es una tecnología incluida en .Net 3.0 conocida anteriormente como Avalon. Esta es una opción mas flexible y potente que Windows Forms[2] ya que permite separar totalmente la interfaz de usuario con la “lógica de negocio”, permitiendo que estas se definan de forma separada, una característica mas que deseable para nuestro proyecto, razón por la que se eligió esta tecnología de Microsoft.

### WCF

Windows Comunication Foundation, anteriormente conocido como Indigo, es una tecnología de Microsoft que proporciona de una manera rápida y solida las herramientas para el desarrollo de servicios web de diferentes tipos. Siendo especialmente útil en el proyecto a la hora de definir y desplegar los servicios web que el servidor expone.

Este proyecto en particular utiliza en concreto la utilidad WebServiceHost, el cual permite desplegar servicios web REST en una aplicación de carácter principal no web como en nuestro caso.

### .NET Framework 4

.Net framework es la plataforma sobre la que se desarrollan las aplicaciones en Visual Studio y que se ejecutan en un entorno conocido como CLR proporcionando un incremento en la seguridad, manejo de memoria y control de excepciones. Así, el CLR junto con las librerías de clases proporcionadas forman .NET Framework.

Todo el código compilado en .Net produce código CLR, de forma que una de las ventajas específica de esta tecnología para nuestro proyecto es la posibilidad de poder añadir una DLL al sistema para incluir un servicio sin necesidad de forzar al usuario en que sea en un lenguaje concreto.

### The Axis Communications AB logoCámaras Axis

Axis es una compañía sueca que desarrolla cámaras de red. Estas cámaras sobre IP ofrecen gran diversidad de funcionalidades interesantes para el proyecto, es por ello que se han elegido como principales dispositivos del sistema.

Estas cámaras se conectan a través de IP de forma que son fácilmente accesibles a través de la red.

Las cámaras Axis tienen un servidor web instalado que permite visualizarla cámara y cambiar su configuración desde el navegador de una forma sencilla.

#### Configuración

La configuración de la cámara se realiza a través de pestaña.

Las opciones más importantes a configurar son:

* Usuarios: Debemos añadir el usuario indicado al sistema para que pueda acceder
* TCP/IP->Advanced TCP/IP settings: para configurar el acceso a la cámara a través de la red.
* Focus: Permite enfocar correctamente la cámara.
* Video & Image: Opciones de video como resolución, tipo de video, etc…
* Event Config: Configuración para detección de eventos y control de estos.

#### Acceso a las cámaras

El acceso a la cámara desde la aplicación servidor se realiza a través del ActiveX Axis Media Control el cual proporciona acceso al stream de video y audio a la vez con métodos para realizar grabaciones, captura de imágenes, etc…

#### Notificaciones de la cámara al servidor

Las cámaras disponen de tres formas principales de comunicación con el exterior. Puertos binarios, envío de correos y mensajes TCP. Se ha escogido la última por sencillez, puesto que el puerto binario complicaría considerablemente la solución, y por velocidad de reacción, ya que la utilización de correos para recibir eventos de las cámaras introduciría un retraso más que considerable. De esta forma la comunicación se produce a través de un mensaje TCP, lo cual implica que el servidor tiene un cliente TCP a la espera de este tipo de mensajes. El mensaje puede contener la información que el usuario desee, pero para transmitir eventos de un tipo determinado como Deteccion de movimiento o cross line, se recomienda el formato: “:” Tipo de evento “:” Dispositivo “:” Informacion adicional “:”.

Los eventos actualmente reconocidos son:

* MDI: Detección de movimiento.
* CL: Cross line.

Cross line es una funcionalidad que puede instalarse de forma gratuita en las cámaras que permite producir un evento cuando un objeto cruza una línea dibujada en la cámara. Esto permite conteo de personas y otras funcionalidades.

#### Cámaras Actuales en el sistema

* [AXIS M5013 PTZ](http://www.axis.com/es/products/cam_m5013/) (2 unidades, de interior) sin carcasa
  + Movimiento horizontal/vertical y zoom en un diseño ultra discreto
  + SVGA y H.264
  + Protección IP51 frente al polvo y el goteo de agua
  + Alimentación a través de Ethernet (IEEE 802.3af)
  + Micrófono integrado y detección de audio
  + También soporta detección de movimiento.
  + Especificaciones:
    - <http://www.axis.com/files/manuals/um_m5013_42914_en_1104.pdf>
    - <http://www.axis.com/files/datasheet/ds_m5013_46851_es_1203_lo.pdf>
* [AXIS P3344](http://www.axis.com/es/products/cam_p3344/) (para el conteo de personas en edificio investigación, de interior)
  + Diseño discreto y compacto
  + Calidad HDTV
  + Fácil instalación con zoom y enfoque remotos
  + Múltiples secuencias H.264
  + Funcionalidad de visión día/noche
  + Alimentación a través de Ethernet
  + Especificaciones:
    - <http://www.axis.com/files/datasheet/ds_p33_43524_es_1111_lo.pdf>
  + Para colgarla en pared:
    - <http://www.axis.com/es/files/manuals/ig_pendant_kit_p33_V_40883_es_1011.pdf>
* [AXIS M3113-VE](http://www.axis.com/es/products/cam_m3113ve/) (puede ser de exterior para entrada o salida parking). No se necesita carcasa adicional.
  + Diseño plano y discreto
  + Excelente calidad de vídeo
  + A prueba de agresiones
  + Alarma anti manipulación
  + Alimentación a través de Ethernet
  + El objetivo se puede mover para centrar el ángulo de enfoque: de 0 a 90º en vertical y de 0 a 30º en horizontal (hacia derecha y hacia izquierda).
  + En la hoja de especificaciones aparece la funcionalidad de detección de movimiento (necesaria para poder realizar las fotos de las matrículas), aunque en la especificación de la aplicación de movimiento no aparece esta cámara como compatible. Deberíamos probarlo antes configurando la cámara.
  + Especificaciones en: <http://www.axis.com/es/files/manuals/ig_m31ve_42270_es_1105.pdf> <http://www.axis.com/files/manuals/um_M3113_3114_40826__en_1104.pdf>
* [AXIS 212 PTZ-V](http://www.axis.com/es/products/cam_212v/) (de interior, no hay carcasas)
  + Visión completa
  + A prueba de agresiones
  + ﻿Movimiento vertical/ horizontal y zoom instantáneo
  + Sin piezas móviles
  + Resolución constante
  + Especificaciones:
    - <http://www.axis.com/files/datasheet/ds_212ptz-v_34054_es_0812_lo.pdf>
* [**Axis 221**](http://www.axis.com/es/products/cam_221/)**:**
* Función día/noche automática con filtro IR extraíble
* Hasta 45 imágenes por segundo en todas las resoluciones, hasta 640x480 píxeles
* Soporta de forma simultanea secuencias de vídeo Motion JPEG y MPEG-4
* Detección de movimiento multi-ventana
* Tiene salida binaria para activar algo (e.g. la barrera).
* Especificaciones:
  + <http://www.axis.com/es/files/datasheet/ds_221_35210_es_0904_lo.pdf>
* [**Axis 211-W:**](http://www.axis.com/es/products/cam_211w/)
* Producto descontinuado → Producto de sustitución: [AXIS M1011-W](http://www.axis.com/es/products/cam_m1011w/index.htm) / [AXIS M1031-W](http://www.axis.com/es/products/cam_m1031w/index.htm)
* Excelente calidad de imagen
* Motion JPEG y MPEG-4 simultáneos
* Instalación flexible con capacidades inalámbricas
* Comunicación segura
* Paquete opcional para exteriores
* Especificaciones:[**http://www.axis.com/es/files/datasheet/ds\_211w\_42338\_es\_1103\_lo.pdf**](http://www.axis.com/es/files/datasheet/ds_211w_42338_es_1103_lo.pdf)

### SMS Esendex

Esendex es una empresa que proporciona una API que permite el envío de SMS desde código. Esto se realiza a través de un servicio web que proporciona Esendex al cual se accede a través de una cuenta creada en la propia web de la empresa.

Disponen de dos modalidades de pago:

* Prepago: Se compra una cantidad de SMS a enviar en menos de 12 meses.
* Contrato: Al final de cada mes se recibe una factura en función del número de mensajes enviados.
* Prueba: Se dispone también de una prueba de 25 mensajes para evaluar el producto.

Esta empresa además ofrece otro servicio, número móvil virtual, el cual puede ser más que interesante para implementar un servicio de recepción de eventos a través de mensajes SMS de forma que un usuario pueda lanzar eventos en el sistema desde su móvil.

### Envío de SMTP con puerto 25 abiertoServidor de Correo

Para el envío de correos desde el servidor se puede utilizar cualquier servidor de correo SMTP. Para ello el servicio utiliza la librería System.SMTP.Mail, enconcreto “SMTPClient” el cual indicándole el servidor SMTP a utilizar nos permite mantener una comunicación entre nuestro servidor y el servidor de correo para el envío de mails.

Los puertos asociados con la comunicación con el servidor son el 25 y el 587(seguro).

La dirección del servidor puede encontrarse a través de internet o en la ayuda del servicio de correo que se desee usar, aunque este suela ser el nombre del dominio precedido de smtp.

Ej: “smtp.gmail.com”

## Cadena de Eventos

### Delegados y Sistema de notificación de eventos

Para implementar la configuración entre los servicios se busco un sistema de notificación que fuera adecuado para el sistema. Se pensó en CORBA, Servicios Web y otras librerías, pero se termino por utilizar el sistema asíncrono de delegados de C#. Un delegado es tipo cuya firma coincide con el tipo de métodos al que se le quiere asociar, de esta forma, a un atributo cuyo tipo es un delegado, se le pueden asociar un conjunto de métodos, todos con la misma firma y estos métodos pueden ser invocados posteriormente.

Esta invocación puede ser realizada de dos formas, síncrona y asíncrona. La primera produce un bloqueo en el momento de la llamada, es decir, hasta que no finalice el proceso no se recupera el control de la ejecución, mientras que de forma asíncrona, la ejecución sigue su curso sin esperar a que la llamada concluya. Esta propiedad es muy importante para el sistema puesto que de otra forma cada “pieza” quedaría bloqueada a la espera de que todas las piezas conectaras terminaran, provocando que si dos dispositivos lanzan un evento, estos no puedan ser atendidos concurrentemente, sino que habría que esperar a que uno terminara de utilizar un servicio para comenzar el otro.

Así, el sistema finalmente un servicio de notificación en el cual unos consumidores se subscriben a los productores de eventos para que sean notificados cuando estos ocurran sin bloqueo.

## Servicios Web

### RESTfull Web Services

“Los servicios web son un componente de software que toma una entrada y produce una salida”[3] estos suelen dividirse en dos tipos, servicios web tipo REST y servicios web comunes.

Estos dos tipos responden a los siguientes esquemas:

|  |  |
| --- | --- |
| File:Webservices.png |  |
| Arquitectura de un servicio web común | Arquitectura de un servicio web tipo REST |

Como podemos ver, los servicios web Comunes requieren de una mayor complejidad a la hora de desplegarse y tienen un protocolo mucho más estricto. Para los servicios web comunes, a la hora de establecer la comunicación es necesario acceder al dsl para obtener la dirección del proveedor del servicio, han de cumplirse ciertos protocolos, como que la información se transmita en xml, etc… haciendo que sea más tediosa la implantación de estos y el consumo por parte del navegador web. En cambio, al utilizar servicios web REST, el cliente conoce la ubicación y los métodos del proveedor de servicio directamente accediendo directamente a los servicios a través de una URL. Otra de las ventajas es la posibilidad de devolver los datos en formato JSON, un tipo mucho más amigable para JavaScript. Gracias a todo esto, el despliegue y uso de servicios REST se vuelve mucho más sencillo y manejable para nuestra situación en concreto.

Debido a que la seguridad es un aspecto crucial para nuestra aplicación, la implantación de un protocolo de seguridad era necesaria para los servicios web. Para realizar la comunicación de forma segura se decidió exponer los servicios web sobre HTTPS de forma segura. Para ello es necesario disponer de un certificado SSL que cifre la conexión, es posible obtener un certificado de entidades como VeriSign o firmar uno individualmente. Debido a que no se ata al usuario a utilizar un determinado certificado creado por Quercus, el certificado que el usuario elija usar deberá ser añadido a través de IIS (Esta herramienta permite también la creación de un certificado de forma sencilla).

### JSON

JavaScript Object Notation es un formato de intercambio de datos estándar independiente del lenguaje y basado en texto[4]. Este formato es muy conveniente para nuestra aplicación web que utiliza JavaScript ya que es soportado directamente por el lenguaje.

Los tipos básicos de JSON son:

* Números
* Cadenas
* Booleanos
* Arrays
* Objetos
* Null

JSON utiliza pares de valores “nombre”: “valor” separado por dos puntos y cada par entre coma. Así, un ejemplo de JSON sería:

{

"id": 1,

"name": "Foo",

"price": 123,

"tags": ["Bar","Eek"],

"stock": { "warehouse":300, "retail":20 }

}

## WebSite

### IIS

Internet Information Services es una aplicación de servidor web que permite alojar una web para que sea accedida desde internet. Por defecto acepta código ASP.Net y puede instalarse un plugin para que interprete también PHP. IIS 7.5 soporta HTTP, HTTPS, FTP, SMTP y NNTP. Es una parte integral de Windows Server y puede ser instalado en Windows 7. Sobre Windows XP y Vista pueden instalarse versiones más antiguas de IIS.

IIS puede utilizarse desde la librería WCF de forma que al exponer un servicio en HTTPS desde código, este buscará automáticamente los certificados en IIS.

A pesar de que la seguridad de los servidores de IIS en versiones anteriores a la 6 era considerada baja debido a que tenía diversas vulnerabilidades que hacían al sistema fácil de atacar o que los equipos no estaban debidamente actualizados ya que disponían de copias ilegales de Windows, pero esta situación ha cambiado para las versiones de IIS 6 y 7 ya que se ha aumentado mas que considerablemente la seguridad del sistema y ahora las actualizaciones de seguridad están disponibles para todos los usuarios.

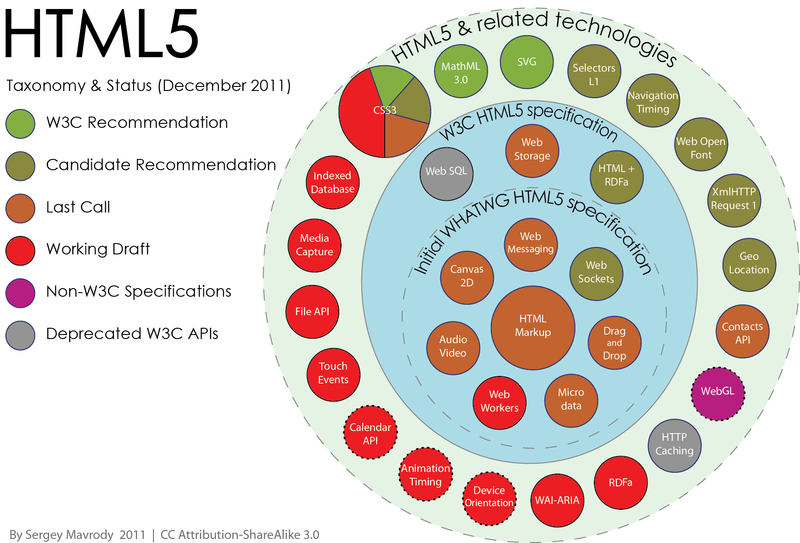
### HTTPS y SSL

HTTP Seguro es un protocolo de comunicación que se obtiene al colocar HTTP sobre SSL. La combinación de estos dos protocolos proporciona una conexión HTTP encriptado, de forma que la información intercambiada entre el servidor y el cliente no pueda ser leída por terceros.

Este protocolo de seguridad se utiliza tanto para el acceso al WebSite como para la comunicación con los servicios web.

### HTML5

HTML5 es un conglomerado de estándares aun no asentados totalmente aunque muchos han sido usados durante mucho tiempo y soportados por navegadores aunque no fueran parte de XHTML. HTML5 es muchas veces nombrado como “la tecnología de nueva generación para la escritura de páginas web”[5] y es que gracias a la gran cantidad de herramientas que ofrece lleva a la web aun más lejos sin la necesidad de añadir Plug-in como applets o código flash.

Debido a que gran cantidad de las herramientas que se incluyen en HTML5 son aun un borrador no todos los navegadores la soportan. Es por ello que se debe ser muy cauteloso a la hora de elegir qué usar de HTML5 puesto que es muy probable que tengamos que hacer diferentes versiones para los navegadores. Podemos consultar páginas como <http://caniuseit.com> o <http://html5please.com> para comprobar el soporte actual de una herramienta. HTML5 ha sido utilizado en el proyecto para la construcción del WebSite, es por ello que algunas funcionalidades solo funcionan en algunos navegadores tal y como se detalla en el apartado de compatibilidad entre navegadores.

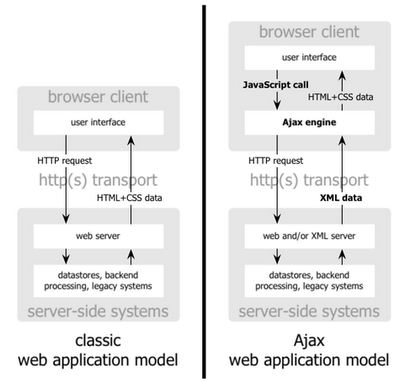
### JavaScript y JQuery

“JavaScript permite a una página reaccionar de forma inteligente, agregar contenido, hacer que desaparezcan elementos, emitir mensajes de información, incluso acceder a un servidor web para actualizar la información de la página sin necesidad de recargarla. En pocas palabras, JavaScript permite crear paginas web mas efectivas e interactivas.”[6]

Además, para incrementar la compatibilidad entre los navegadores y realizar la codificación más agradable existen librerías como JQuery que permiten manejar JavaScript desde un nivel más alto.

Así, utilizando JavaScript y en concreto la librería JQuery (en nuestro caso particular) nuestro WebSite se vuelve dinámico, permitiéndonos añadir, eliminar o modificar contenido en función de parámetros en tiempo de ejecución sin necesidad de recargar la pagina.

### AJAX

“AJAX (Asynchronous JavaScript and XML) es un grupo interrelacionado de técnicas de desarrollo web usados en el lado del cliente para crear aplicaciones web asíncronas”.[7] A pesar de que AJAX suele involucrar XML este no es necesario y puede ser sustituido por JSON como en nuestro caso.

Esta agrupación de tecnologías es de vital importancia para nuestra aplicación web ya que de otro modo no podría actualizarse de forma asíncrona.

Con AJAX se añade una capa en el lado del cliente que esta en comunicación con el servidor y actualiza la página dinámicamente sin necesidad de pedir al servidor que genere la página de nuevo. En nuestro caso utilizamos AJAX para comunicarnos con el servidor C# del sistema a través de los servicios web y realizar el logeo del usuario, obtención de los flujos de video y demás operaciones que soporta este. Sin esta tecnología no seria viable hacer este tipo de aplicación.

Es importante destacar que la comunicación AJAX se inicia en el lado del cliente, es decir, el cliente solicita unos datos al servidor y no al revés, por lo que para la comunicación inversa (servidor a cliente) es necesario el uso de otras tecnologías tal y como se indica a continuación.

## Comunicación Servidor-Cliente

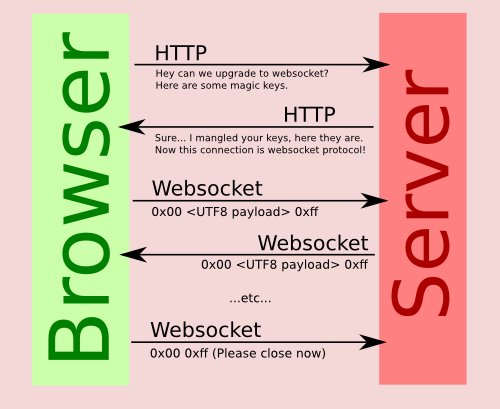
### Tecnologías Push

Push describe un estilo de tecnología en la cual la solicitud para una determinada transacción comienza en el servidor en lugar del cliente. Es lo contrario a las tecnologías tipo Pull en las cuales el cliente solicita la información al servidor. Este tipo de tecnología suele seguir el modelo “publish/subscribe”.

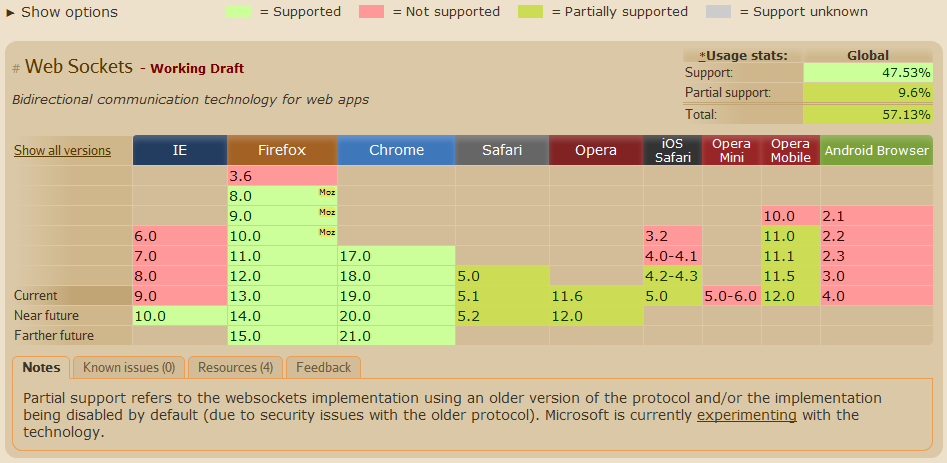
Este tipo de tecnología es fundamental para el desarrollo de aplicaciones web en tiempo real, debido a que la comunicación la inicia el servidor en el instante que ocurre un evento, en lugar de tener el cliente que consultar al servidor periódicamente produciendo un retraso inaceptable en ciertas situaciones.

El principal problema de las tecnologías Push en la web es que la web en si no fue diseñada para esto. En un principio la web estaba concebida para realizar una única petición al servidor y mostrar todo el contenido, estableciendo una conexión única en la dirección cliente-servidor. Aun así, actualmente este tipo de tecnología esta en demanda para gran cantidad de aplicaciones, y ya en HTML5 donde se encuentran varias soluciones.

### WebSockets

El protocolo de WebSockets posibilita una comunicación dúplex entre el servidor y el cliente.[8] Este es aun una proposición de estándar, pero todavía no está consolidado como tal. Una vez abierto un WebSockets tanto el cliente como el servidor pueden iniciar una request, lo cual permite ofrecer aplicaciones en tiempo real e incrementa notablemente la capacidad de reacción de un cliente web al tener un protocolo de comunicación dúplex.

Uno de los problemas es que existen diferentes protocolos de WebSockets, siendo el propuesto como estándar soportado por Internet Explorer 10, Firefox 11 y Chrome 16 lo cual corresponde al 45% de la cuota de mercado de navegadores. Pero por el contrario si utilizáramos como hybi-00 o hybi-10 nos encontraríamos con un soporte muy pobre (Solo Firefox y Chrome). Es por ello que se ha decidido utilizar otro tipo de servicio para este tipo de comunicación como alternativa a los WebSockets.



La implementación de WebSockets se divide en dos partes.

#### Cliente

En el cliente podemos utilizar el estándar de la forma:

|  |
| --- |
| Var ws = new WebSocket(direcionWS, 'post'); |

Con esta línea creamos una variable que almacenará el WebSocket indicándole la url y opcionalmente el protocolo a utilizar. A través de esta variable podemos acceder al estado de la conexión y al número de bytes en el buffer para ser enviados al utilizar send.

Además tenemos una serie de manejadores a los que deberemos de pasar una función que se encargue de gestionar ciertos eventos que son lanzados en el WebSocket. Para indicar la función basta con asignar al manejador una función tal y como se muestra a continuación.

|  |
| --- |
| ws.onmessage = function (evt) {  alertaEnCamara(evt.data);  }; |

El evento tiene dos atributos, data y type. El primero contiene la información recibida y el segundo el tipo de evento recibido.

##### Atributos:

|  |  |
| --- | --- |
| Socket.readyState | Atributo de solo lectura, representa el estado actual de la conexión y puede tener los siguiente valores:   1. CONECTING: El valor 0 indica que la conexión aun no a sido establecida. 2. OPEN: El valor 1 representa que la conexión ha sido correctamente establecida y que la comunicación con el otro extremo es posible. 3. CLOSING: El valor 2 se utiliza para indicar que la conexión esta utilizando la primitiva de cerrado de conexión. 4. CLOSED: Con el valor 3 se indica que la conexión ya ha sido cerrada o no fue posible abrirla. |
| Socket.bufferedAmount | The readonly attribute **bufferedAmount** represents the number of bytes of UTF-8 text that have been queued using send() method. |

##### Eventos

|  |  |  |
| --- | --- | --- |
| **Evento** | **Manejador** | **Descripción** |
| open | Socket.onopen | Evento lanzado cuando la conexión se establece. |
| message | Socket.onmessage | Este evento ocurre cuando se recibe un mensaje desde el otro punto. |
| error | Socket.onerror | Evento lanzado al ocurrir un error en la comunicación. |
| close | Socket.onclose | Evento lanzado al cerrarse la conexión con el otro extremo. |

##### Métodos

|  |  |
| --- | --- |
| Socket.send() | El método send(data) transmite data utilizando la conexión abierta. |
| Socket.close() | Este método cierra la conexión del WebSocket. |

#### Servidor

En el lado servidor tenemos varias opciones:

* Implementar nuestro propio WebSocket siguiendo el protocolo borrador estándar RFC6455.
* Esperar al soporte nativo de Microsoft en .Net framework 4.5.
* Utilizar la API de una de las diversas implementaciones que se encuentran en la red.
  + NuGet: Este proyecto es muy interesante ya que hace realmente simple como se implementan los WebSockets pero tras implementarlo no fue posible hacerlo funcionar debido a que implementa el protocolo hybi en lugar del RFC por lo cual fue descartado como opción. Podemos encontrar el código en “<http://nuget.codeplex.com/>”.
  + SuperWebSocket: Aun siendo este proyecto muy interesante, basado en SuperSocket, esta mas enfocado a montar el servidor en una aplicación web por lo que se hace “farragosa” la implementación al no haber además ejemplo de como utilizarlo en código c# (los servidores se levantan con código JavaScript en los ejemplos). De forma que tras varios intentos se descartó igualmente. Podemos encontrar el código en: “<http://superwebsocket.codeplex.com/>”.
  + Alchemy: API que utiliza la RFC6455 y que proporciona al programador todo el control sobre el control de los usuarios conectados al WebSocket. Esta ha sido la opción utilizada. Se puede encontrar el código en “<http://alchemywebsockets.net/>”.

Esta API nos permite crear tanto servidores como clientes a través de código C#.

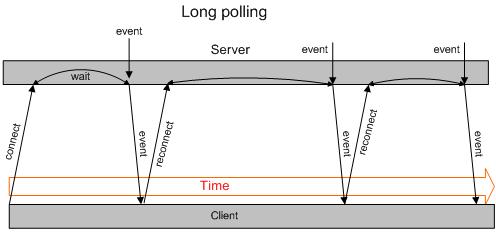
Al crear un servidor debemos indicarle los métodos a utilizar para los diferentes eventos que el WebSocket lanza. Es importante que utilicemos los eventos “OnConnect” y “OnDisconnect” para registrar a los usuarios y poder enviar mensajes a estos. Y “OnReceive” si queremos una comunicación dúplex.

### Long Polling / Comet

Long polling, también conocido como Comet, es una emulación de la tecnología Push utilizando tecnología Pull. Long polling envía una request al servidor y el servidor, en lugar de responder inmediatamente a la request espera a tener datos a transmitir, si la conexión es cerrada (los navegadores suelen cerrar la conexión a los quince segundos), el cliente vuelve a emitir otra request. De esta forma el servidor dispone de una conexión abierta en el momento que desea emitir información, emulando un servicio Push.[9]

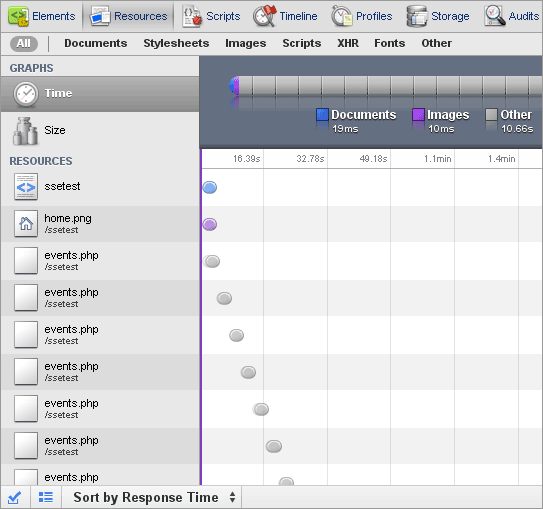
Esto tiene grandes ventajas sobre realizar polling (encuesta cada X tiempo con respuesta inmediata)[10]:

* El mensaje desde el servidor se envía en el instante deseado.
* Ofrece información en tiempo real. (Debido a el punto anterior)
* Produce menor carga de la red.



Aun así, debido a que esta tecnología no es realmente Push, sino una emulación de este y que aunque tenga menos carga que polling sigue sin ser comparable a una tecnología como WebSockets, por ello finalmente no se implementó como principal mecanismo de comunicación entre el servidor y el cliente.

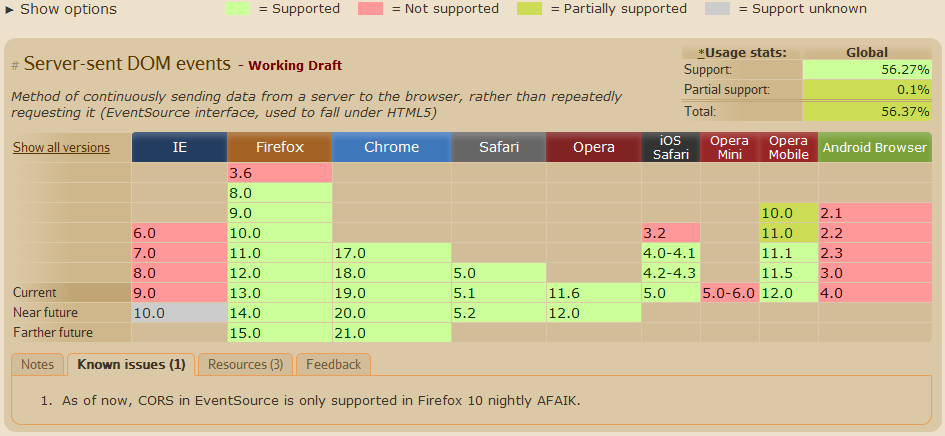
### Server Side Events

Server side events (o EventSource) representa un método para enviar de forma continua datos desde un servidor a un cliente (Push) en lugar de realizar peticiones continuas desde el cliente al servidor (polling).

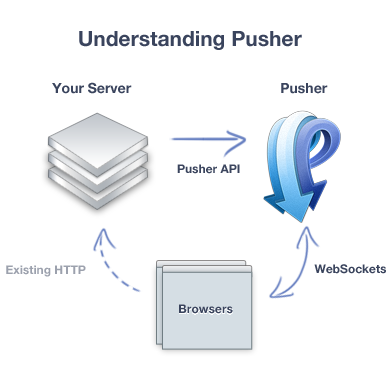
Es importante tener en cuenta que aunque esta tecnología pretenda ser de tipo Push lo que esta haciendo realmente es una especie de polling pero abstrayendo al programador de esta. Es decir, todo el control se hace a través de la API de event source siendo notificado el código JavaScript solo cuando se produce un evento nuevo, pero la aplicación web esta realizando request al servidor cada tres segundos (el intervalo es configurable) tal y como se muestra en la imagen.

La implementación de esta tecnología es sencilla, y se encuentra código comentado tanto en el servidor C# como en el cliente web por si se desea volver a activar. Podemos encontrar un buen ejemplo de cómo implementar esta tecnología en: <http://dsheiko.com/weblog/html5-and-server-sent-events>. Finalmente no se utilizo esta tecnología debido a que no era ralamente una tecnología tipo Push como tal.

Este es actualmente soportado por la mayoría de navegadores salvo Internet Explorer.

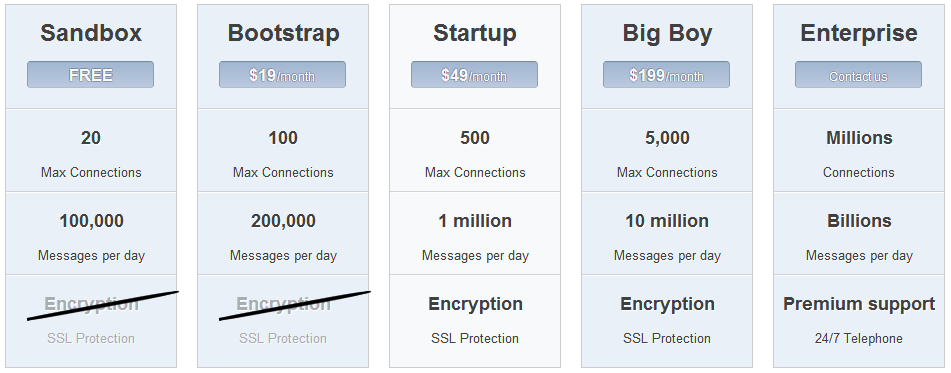


### Pusher.com

Pusher.com define su producto como “Pusher es un API simple hosteada en la web para añadir de una forma fácil y segura comunicación bidireccional a través de WebSockets”. Pusher utiliza el futuro estándar de WebSockets más una solución flash para aquellos navegadores que no soporten WebSockets.

El funcionamiento de Pusher es sencillo, basta incluir la API en nuestro código servidor para realizar la comunicación entre el servidor y Pusher y añadir un código JavaScript en el WebSite para establecer el otro punto de la comunicación. Así se establece una comunicación entre el servidor y Pusher y otra entre Pusher y cada cliente. La primera se hace a través de la API de este y la segunda a través de WebSockets como ya se ha explicado anteriormente.

Pusher requiere que el usuario se cree una cuenta para poder acceder al servicio. Hay diferentes tipos de cuentas, en función de la necesidad del sistema. En nuestro caso, es más que suficiente con coger la cuenta gratuita, la cual proporciona cien mil mensajes diarios y hasta veinte conexiones simultaneas.

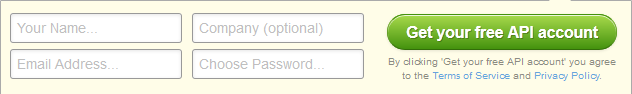


Tras ser probado el servicio muestra una respuesta rápida y es independiente del estándar de WebSockets (puesto que de esto se encarga Pusher) siendo una solución más que aceptable para el envío de eventos desde el servidor a los clientes. Además, con vistas de futuro, Pusher permite también la comunicación con móviles y otros dispositivos, lo cual puede resultar interesante para próximos desarrollos.

Aun así, es importante tener en cuenta que se esta dependiendo de un servicio externo, por lo cual seria más recomendable utilizar WebSockets una vez que el estándar se formalice.

En Docs podemos encontrar todo el código necesario para poner en funcionamiento Pusher.

#### Crear una cuenta en Pusher

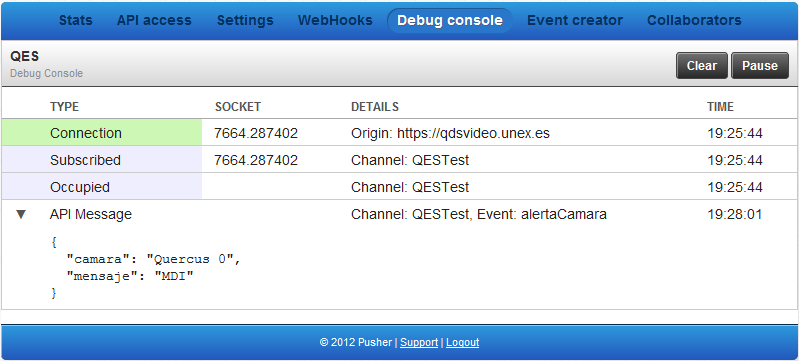
Para crear una cuenta en Pusher basta con acceder a <http://pusher.com> hacer click en , introducir los datos requeridos y pinchar en “Createyour free API account”. Tras esto recibiremos un correo con todos los datos para activar nuestra cuenta.

A partir de este momento podemos acceder al panel de la aplicación haciendo click en el tercer botón del panel tras habernos logeado. Aquí disponemos de los siguientes menús:

* Stats: Ratio de conexiones y mensajes diarios, para tener un control.
* API Access: Una página para obtener los datos de configuración necesarios para nuestro sistema.
* Settings: Algunas características se pueden configurar desde este sistema, como el nombre o la activación de SSL.
* WebHooks: Emitir información al servidor cuando eventos ocurren en Pusher. (conexiones, desconexiones, etc…)
* Debug Console: Consola de depuración.
* Event Creator: Herramienta para producir eventos.
* Collaborators: Pagina para proporcionar a otras personas la habilidad de depurar nuestra aplicación, producir eventos y acceder a los datos de esta.

#### Depurar una aplicación Pusher

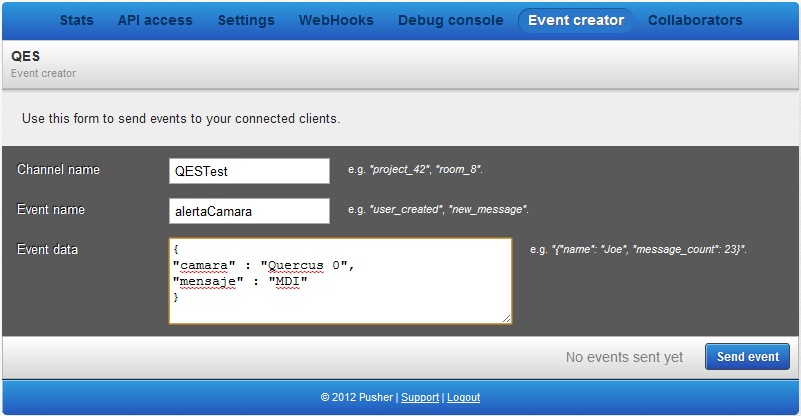
La opción “Debug console” nos permite depurar el sistema de Pusher ya que registra todas las comunicaciones que se producen. Conexiones de clientes, envío de mensajes, etc…



#### Enviar eventos desde Pusher

Al pinchar en “Event creator” accedemos a un menú que nos permite crear y enviar eventos desde la web, para realizar pruebas, una utilidad muy interesante.

En este panel de configuración deberemos indicar el nombre del canal, el nombre del evento y los datos que contiene el evento en formato JSON tal y como se puede ver en la imagen.



#### Código de Pusher

A continuación se expone el código necesario para producir un evento

Código para .Net:

|  |
| --- |
| var request = new ObjectPusherRequest("test\_channel",  "my\_event",  new { hello = "world" });  var provider = new PusherProvider(appId, appKey, appSecret);  provider.Trigger(request); |

Código JavaScript para recibir eventos de un determinado tipo lanzado por el servidor:

|  |
| --- |
| var pusher = new Pusher(pusherKey);  var channel = pusher.subscribe(pusherChannel);  channel.bind('alertaCamara', function(data) {  Control del evento  } |

Además de los eventos lanzados por el servidor es posible también registrar para manejar eventos del tipo nuevo conexión, desconexión, etc… También se dispone de una variable para obtener el estado actual de la conexión.

Podemos encontrar toda la información acerca de como utilizar la API de Pusher (tanto del lado servidor como desde la aplicación web) en la documentación disponible en la web.

## Modelado

Para la configuración del sistema se ha decidido utilizar desarrollo dirigido por modelos de forma que el usuario obtenga un entorno y funcionalidad totalmente adaptado a sus necesidades. Para ello se utilizan las tecnologías que se muestran a continuación.

### EMF

Eclipse Modeling Framework es una herramienta incorporada en eclipse para la definición de modelos y meta modelos. Es la base que utilizaremos para el DSDM sobre la que se construyen todas las demás tecnologías.[11]

En este proyecto se utiliza para la creación de los modelos y meta modelos.

### GMF

Graphycal Modeling Framework es un plugin de eclipse que permite la generación de editores gráficos para un meta modelo en particular. Este nos permite definir un modelo en base a un meta modelo “con el ratón”, es decir, dibujando los componentes como si se tratara de un diagrama UML.

### OCL

Object Constraint Language es un lenguaje que es utilizado para añadir restricciones a diagramas UML. Así, existe OCLinEcore que permite añadir restricciones a meta modelos (ficheros .ecore). Esta herramienta nos permite enriquecer el modelo con restricciones y operaciones que no se pueden hacer directamente con MOF.

A través de OCLinEcore se han definido diversas restricciones en el meta modelo del proyecto.

Podemos encontrar información interesante sobre como utilizarlo en los siguientes enlaces:

* “<http://help.eclipse.org/indigo/index.jsp?topic=%2Forg.eclipse.ocl.doc%2Fhelp%2Findex.html>”
* “<http://www.csci.csusb.edu/dick/samples/ocl.html>”.
* “<http://www.slideshare.net/EdWillink/enrich-your-models-with-ocl>”.

### XText

Xtext es un framework que facilita el desarrollo de DSLs textuales ya que automatiza la generación de herramientas de soporte del lenguaje. Es un proyecto Open-Source y se incluye en forma de plug-in a eclipse que forma parte de Eclipse Modeling Project.[12]

En el proyecto se pretende usar para la generación de un editor en modo textual para generar modelos del sistema.

### JET

Java Emiter Templates es una herramienta integrada en EMF inspirada en JSP, basado en tags y que permite realizar transformaciones modelo a texto[12]. Una de sus características más interesantes es que define interfaces de extensión del plugin, permitiendo ir aun más allá con la funcionalidad de la herramienta, como abrir o ejecutar un fichero directamente desde JET. Uno de sus inconvenientes es que solo permite transformaciones SIMO (Una entrada múltiples salidas), pero para nuestro objetivo, que tiene un modelo de entrada y uno de salida es mas que suficiente.

Esta herramienta se utiliza para transformar el modelo del sistema en código C#, BATCH, HTML y JavaScript para realizar la inicialización y configuración del sistema.

# Decisiones de diseño

## Gestor de eventos

El gestor de eventos se ha implementado con el servicio de notificación a través de eventos y delegados proporcionados por .Net. Esta fue una decisión tecnológica importante, debido a que era importante la velocidad de respuesta del sistema así como la reusabilidad de los servicios. Tras un amplio estudio esta fue la tecnología escogida debido a que cumplía adecuadamente todos los requisitos, permitiendo que los servicios no permanecieran bloqueados mientras propagaban los eventos.

## Aplicación servidor

La aplicación servidor ha sido implementada en la plataforma .Net, en el lenguaje C# en concreto. Se escogió esta plataforma debido a la gran versatilidad que proporcionaba .net y todo el soporte que se encuentra online. Además, .Net aportaba tecnologías como WPF y WCF que cubrían de una forma muy elegante los requisitos que esta aplicación tenía.

Para la interfaz elegimos WPF debido a que es una tecnología que separa completamente la interfaz del código y nos permite que nuevas “piezas” sean diseñadas fácilmente sin necesidad de saber programar a través de ciertas herramientas de Microsoft.

Por último, para exponer las funcionalidades necesarias de la aplicación servidor para los clientes, se decidió usar RESTfull webservices debido a que presentan una mayor facilidad de integración para los clientes al ser expuestos como URLs.

Tras un largo debate se decidió utilizar una base de datos para almacenar los usuarios, la configuración de algunos parámetros y el log, se estudió detalladamente que parámetros deben establecerse en el DSL y cuales son configurables desde la aplicación ya que lo configurado en el dsl va escrito en código por lo que no es necesario almacenarlo en la base de datos.

## Clientes

Tras valorar detenidamente si era necesario un cliente nativo para PC, se decidió optar por no desarrollarlo en favor de un cliente web sólido, permitiendo una mayor compatibilidad tanto para PCs como para dispositivos móviles. De esta forma, el cliente será el navegador de cada usuario, a través del cual se conectara contra el servidor, no descartando completamente la posibilidad de desarrollar aplicaciones nativas en el futuro para móviles o tablets.

## Servidor web

Se ha decidido desarrollar una página web en HTML5 y JavaScript debido a ciertas opcionalidades interesantes que ofrece este como Server Side Events o la posibilidad de integrar video que aunque no se hayan implementado actualmente son características importantes para el proyecto de cara al futuro. Otro punto de interés que decanto la decisión de adoptar HTML5 es CSS4 ya que puede usarse para que una misma web este optimizada para dispositivos tan dispares como móviles y PCs.

En cuanto a la comunicación entre el servidor y los clientes web, se estudiaron las siguientes tecnologías:

* Server side events.
* Comet
* WebSockets
* Pusher.com

Se descarto la primera puesto que no era una tecnología realmente push, la simulaba a través de Long Polling. Las bibliotecas de Comet para .Net eran de pago por lo que se descartaron también. WebSockets fue una de las principales opciones, pero debido a problemas de compatibilidad y a que su estándar es aun es un borrador se decidió utilizar Pusher.com como primera opción, que se encarga de todos los problemas de WebSockets sin necesidad de que el usuario sepa nada, pero aun así, en una ultima instancia a través de una API de WebSockets llamada Alchemy se realizo la implementación de WebScokets ya que de esta forma no se depende de un servicio externo. De este modo el usuario es libre de elegir que opción prefiere.

## Modelado

A la hora de desarrollar el modelo se decide incluir detalles como usuarios y otros parámetros que también son configurables desde la aplicación servidor para proporcionar mayor comodidad al usuario, pero por el contrario no se incluyen parámetros como correos o números de teléfono para servicios específicos ya que al modificarlos en la aplicación quedarían incoherentes.

### Servicios

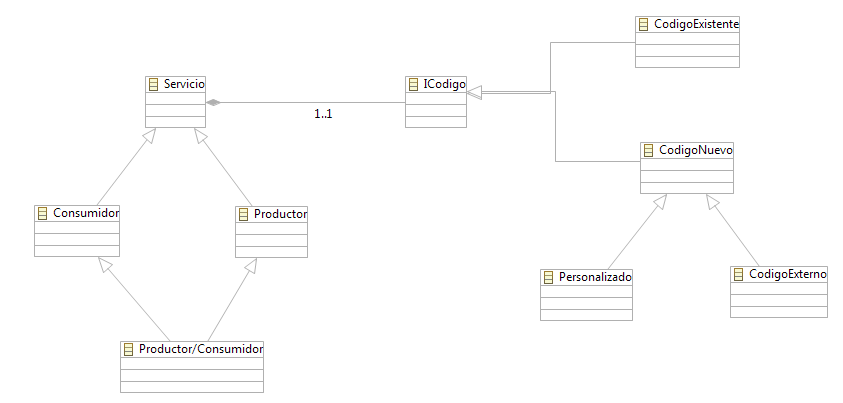
La definición del meta-modelo para la inclusión de servicios definidos por el usuario fue uno de los puntos más complicados y es que hay dos formas de categorizar los servicios:

* Consumidores.
* Productores.
* Productores/Consumidores.

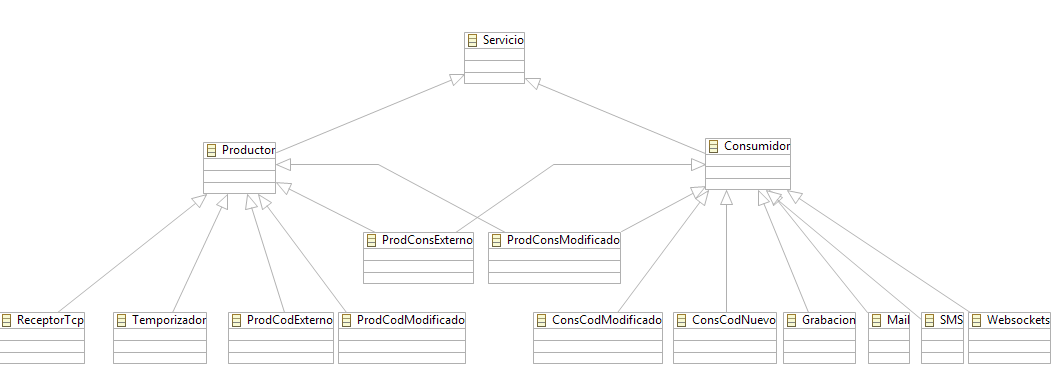
Y

* Servicios ya existentes.
* Servicios nuevos a partir de una personalización de un servicio existente.
* Servicios totalmente nuevos.

Como solución se propusieron los siguientes meta-modelos:

En este modelo se definen en un principio todos los servicios por igual, indicando solo si es Productor, Consumidor o Productor/Consumidor e indicando después el código que el servicio implementa. Es en este paso donde se descubre si el servicio utiliza código existente o crea un nuevo servicio.

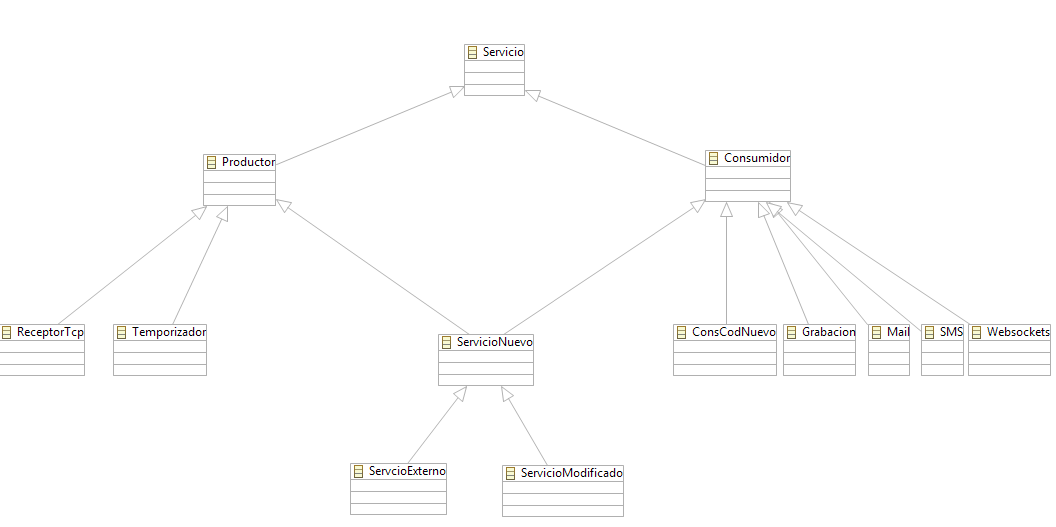
Las principales ventajas de este modelo es la simplicidad en la generación de código ya que todos los servicios se definen por igual. Pero hay ciertas desventajas como que la definición de tipos por atributos no permite la una diferenciación correcta en MDT para poner diferentes iconos por ejemplo o que el usuario no sabría que parámetros son obligatorios de este modo.

Este modelo diferencia en un primer lugar los servicios en Consumidores y Productores y después en los diferentes tipos de servicio que el sistema ya posee. Además, añade dos subtipos a consumidor, dos subtipos a productor y dos subtipos comunes a ambos para añadir nuevos servicios.

Este modelo es más sucio debido a la cantidad de clases necesarias para definir nuevos servicios pero es más fácil de usar para el usuario además de más claro.

Otra opción barajada fue añadir a cada servicio soportado por defecto una opción de configuración, pero añade complejidad innecesaria además de que implica que solo se podrían configurar los servicios que existan en el sistema actualmente y no se podrían configurar servicios que se creen únicamente para ser configurados.

Finalmente se decidió un término medio, la segunda opción en la que se clasifican los servicios con herencia pero añadiendo un único tipo “Nuevo Servicio” que hereda de productor y de consumidor y contiene un atributo para indicar cual de los dos implementa. De esta clase heredan dos clases para definir un servicio a través de personalizar otro o para definir un servicio totalmente nuevo.

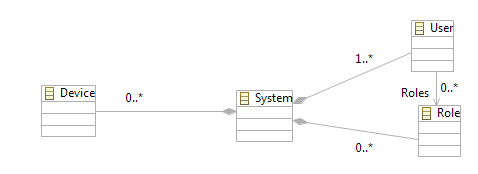


### Roles

La definición de los roles y los usuarios a sido complicado en la definición del meta modelo.

Esto se ha debido a diferentes problemas:

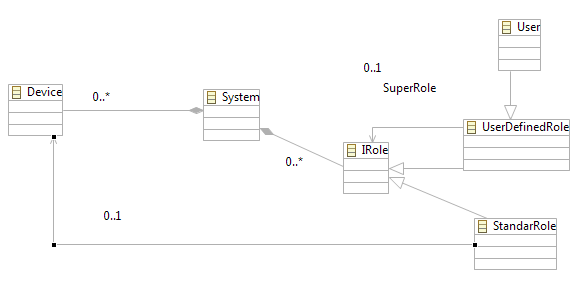
* Inclusión de los dispositivos en los roles.
* Inclusión de una serie de roles “estándar”: Admin y Viewer.
* Proporcionar comodidad a la hora de generar modelos.



En un primer lugar se adopto por un diseño “convencional” en el que el sistema tenia roles y usuarios(los cuales tenían roles asociados como privilegios). En esta opción debían crearse una serie de roles de forma obligatoria (a mano por el usuario): Admin, Viewer y uno por cada dispositivo en el sistema. Al validar el modelo se verificaba que estos modelos existían comprobando el nombre de cada rol (Es decir, se buscaba un rol con nombre “Admin”, otro con nombre “Viewer” y uno con cada nombre de cada dispositivo). Este modelo tenia multitud de problemas y poca flexibilidad.

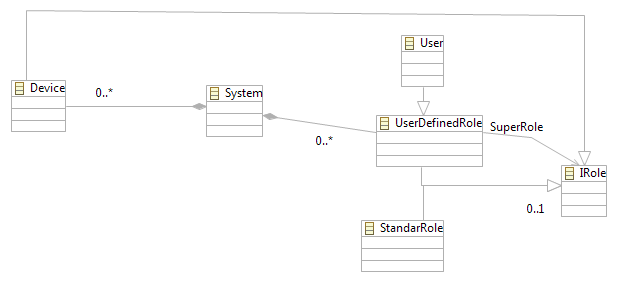
De forma que el primer cambio fue una pequeña mejora al sistema de usuarios. Los usuarios no contienen roles, los usuarios son roles en si con una contraseña y los roles tienen “super roles” asociados de los que heredan todos sus privilegios.

Como segundo paso se separa los roles definidos por el usuario de los roles “estándares” del sistema, pudiendo tener “super roles” únicamente los roles definidos por el usuario. Además, hay tres tipos de roles estándar: Admin, Viewer y Device Role. De este último debe haber tantos como dispositivos haya definidos en el sistema, asociando a cada dispositivo un rol y comprobando que todas estas condiciones se cumplían a través de restricciones OCL.

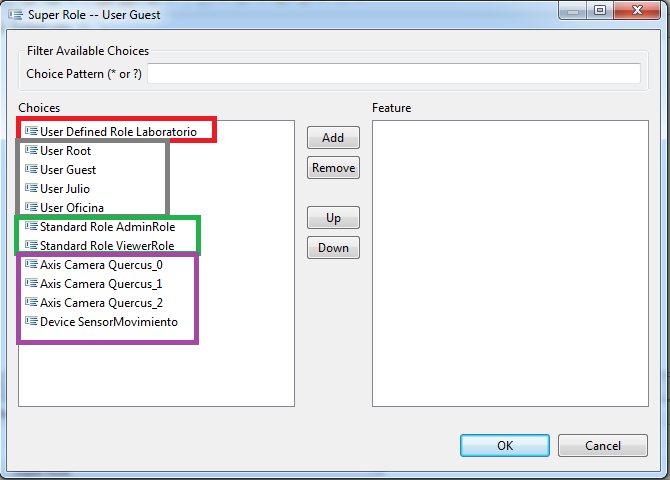


Pero esta solución aun tenia un problema, se obligaba al usuario a crear los roles del sistema, lo cual carece de sentido, puesto que son role que siempre van a existir, en todos los modelos que hagamos. De esta forma, se transforma el meta modelo primero haciendo que el sistema contenga únicamente los roles definidos por el usuario, eliminando así la capacidad de crear roles estándar al usuario, y en segunda lugar se elimina el tipo de rol estándar asociado a dispositivo y se hace heredar a dispositivo de Rol, de forma que los dispositivos representen roles también. Esto tiene tres consecuencias.

1. La buscada, el usuario no tiene que definir ninguno de los roles estándares del sistema.
2. Debe cargarse un recurso externo, un modelo que contiene los dos roles estándar.
3. Todos los roles asociados a los dispositivos se “generan automáticamente” es decir, al crear un dispositivo estamos creando también un rol.



La utilidad de este diseño se ve clara cuando se define un modelo en base a el meta modelo con este formato y es que de esta forma al crear un usuario (o un rol definido por el usuario) y pinchar en “super roles” podemos ver la siguiente ventana:



Tal y como se ve podemos añadir diferentes super roles:

* Roles definidos por el usuario. Marcado en rojo.
* Usuarios. Marcados en gris.
* Roles estándares cargados desde un modelo como recurso. Marcados en Verde.
* Roles asociados a los dispositivos. Marcados en violeta.

Habiendo definido nosotros únicamente los dos conjuntos primeros.

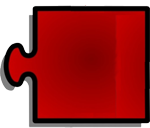
# Gestor de eventos

El gestor de eventos, pretende ser una especie de cadena de bloques conectables, configurados al generar el código que responde a los eventos de una manera predeterminada. Además, se pretende definir una interfaz específica de forma que se puedan generar nuevas piezas y añadirlas a la cadena con el mínimo impacto posible. Esta cadena debe funcionar como un servicio de notificación de forma que no se bloquee ninguna de las piezas.

Para ello, se definen tipos diferentes de servicios dentro de esta cadena: los que producen eventos dentro del sistema (generalmente consumiendo eventos reales) y los que consumen eventos.

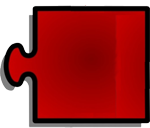


Los generadores de eventos estarían provistos de una interfaz a través de la cual, otras piezas, consumidores de eventos podrían conectarse. Se pueden conectar cuantas piezas se deseen, las cuales serán notificadas cuando se reciba un evento en esta.



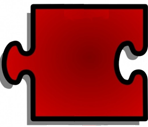
Las otras piezas también deben cumplir una interfaz específica, una para poder ser añadidas a las productoras de eventos. Estas piezas pueden ser conectadas a varios generadores de eventos, de forma que sean activadas en diferentes ocasiones.

Cuando la primera pieza produce un evento (interno al sistema), envía a todas las piezas conectadas un documento con los datos del evento realizado.



Info del evento



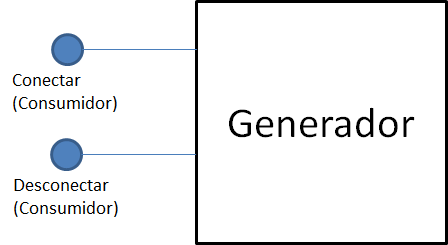
Además, de la combinación de estas dos piezas, aparece una nueva pieza cuya función es transmitir el evento procesándolo primero. Estas piezas de nuevo, reciben un evento interno al sistema y producen otros.

Evento Saliente

Evento Entrante

## Visión general

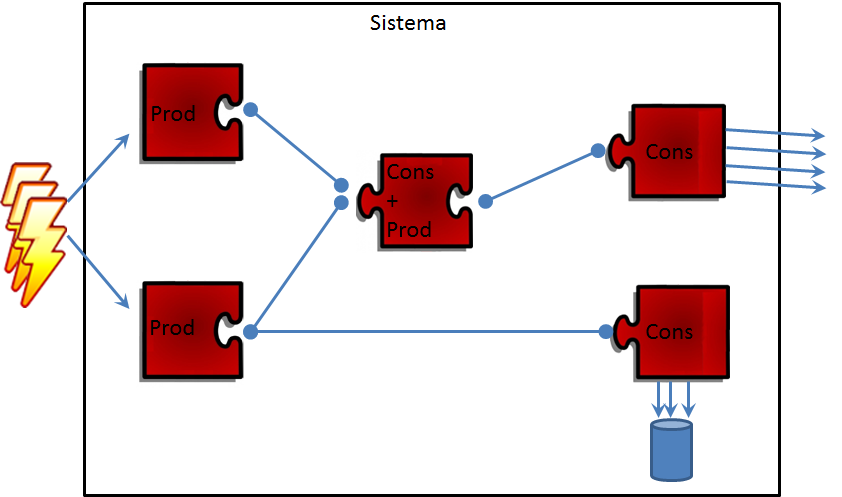
La cadena de eventos se constituye principalmente de dos tipos de servicios, los productores y los consumidores de eventos del sistema. Los productores permiten que se le conecten consumidores, indicando el método que tratara el evento cuando se lance. También se pueden desconectarse en cualquier momento. De esta forma, los productores quedan definidos de la siguiente forma:



Cuando un generador lanza un evento, todos los consumidores conectados serán notificados.

En numerosas ocasiones resulta interesante tener artefactos que sean generadores y consumidores a la vez y cuya función sea simplemente procesar el evento y en determinados casos producir un evento como respuesta a este. De esta forma, al irlos conectado, generaremos una cadena que procesara el evento de la forma que deseemos.

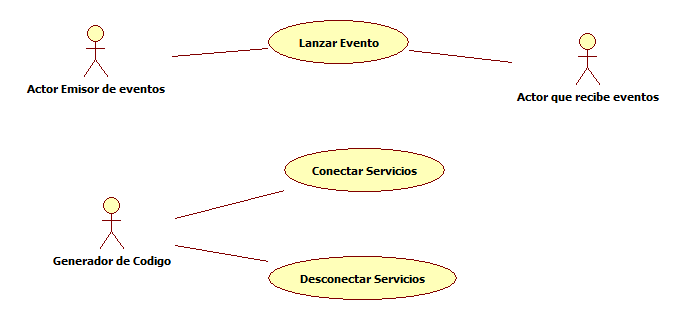
## Ejemplo



Este es un ejemplo en el que eventos del exterior, como detección de movimiento o el envío de un SMS al servidor desencadenan eventos internos en alguno de los productores (también es posible tener productores que se activen de forma programada). Estos enviaran una notificación a los consumidores acoplados. Algunos de estos serán “piezas intermedias” para comprobar por ejemplo que cámara ha lanzado el evento y continuar la cadena si necesario, mientras que los consumidores finales normalmente generaran una respuesta como realizar una grabación, enviar un mail, o simplemente almacenar los datos.

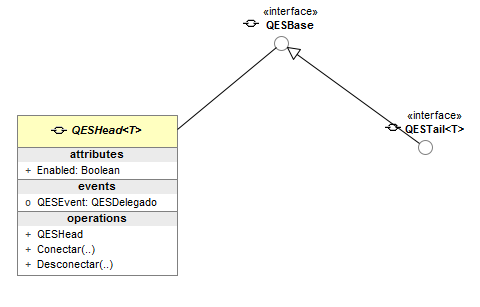
La principal ventaja de este sistema es que en cualquier momento podemos añadir otra pieza sin más complicación que indicando las conexiones que realiza.

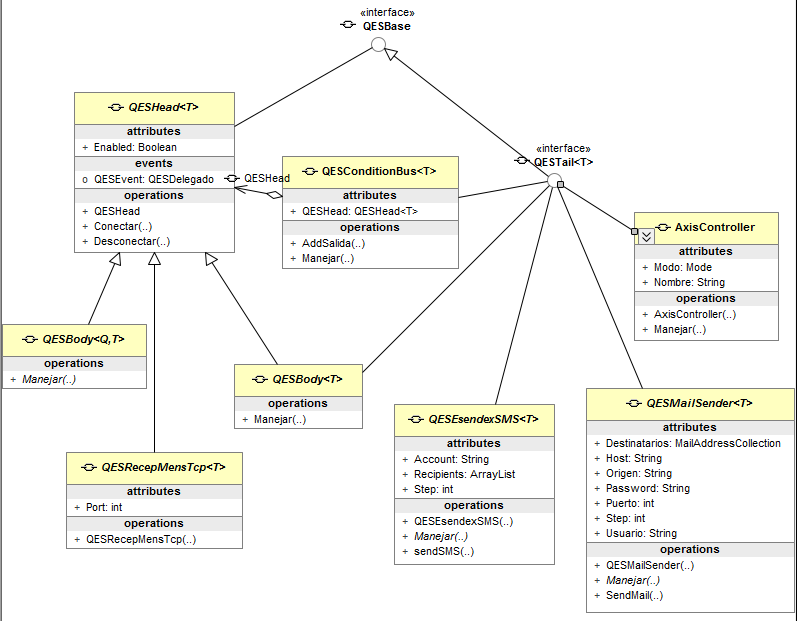
## Casos de uso



## Diagrama de clases

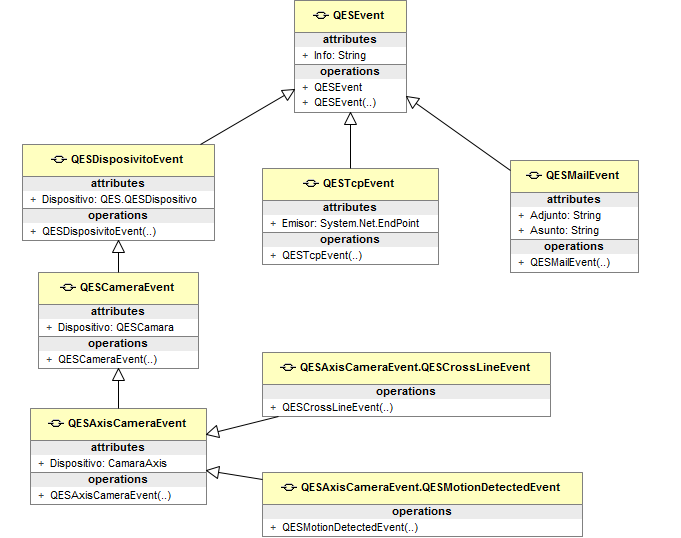
Las clases que implementen servicios que producen eventos dentro del sistema heredan de QESHead, mientras que los consumidores de eventos de QESTail. Ambos implementan QESBase para poder controlar todos los servicios desde una sola interfaz. La interfaz QESTail contiene un método a ser implementado,”void manejar(T evento)”, método que manejara el evento lanzado.



Con esta base, se ha implementado el siguiente modelo de servicios:

## Jerarquía de eventos

Para trabajar con los servicios manejadores de eventos de la cadena mostrada anteriormente, se ha desarrollado una jerarquía de eventos que pretende ser útil en el transporte de información de un punto a otro.



## Servicios Disponibles

|  |  |  |  |
| --- | --- | --- | --- |
| QESBody<T> | | | Abstracta |
|  | **Entrada** | Eventos del tipo T | |
| **Salida** | Eventos del tipo T | |
| **Función** | Pretende ser heredada para tratar/procesar eventos o distribuirlos. | |

|  |  |  |  |
| --- | --- | --- | --- |
| QESTypeBus<T> | | |  |
| T | **Entrada** | Eventos del tipo T | |
| **Salida\*** | Eventos del tipo R(subtipos de T) | |
| **Función** | Un evento entra y toma aquellas salidas que coincidan con su subtipo. De esta forma se pude redireccionar los eventos en función de su subtipo. | |
| \*Hay tantas salidas como tipos se hayan añadido. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| QESConditionBus<T> | | |  |
| C | **Entrada** | Eventos del tipo T | |
| **Salida\*** | Eventos del tipo T | |
| **Función** | Al conectar un receptor se indica una condición a través de una función, cuando se reciba un evento se disparará en todas las salidas que cumpla la función.  A cada salida es posible añadir un consumidor del tipo T. | |
| \*Hay tantas salidas como tipos se hayan añadido. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| QESHead<T> | | | Abstracta |
|  | **Salida** | Eventos del tipo T | |
| **Función** | Clase a heredar para crear productores de eventos | |

|  |  |  |  |
| --- | --- | --- | --- |
| QESRecepMensTcp<T> | | | Abstracta |
|  | **Salida** | Eventos del tipo T | |
| **Función** | Ser heredada para implementar el comportamiento al recibir un mensaje TCP. | |

|  |  |  |  |
| --- | --- | --- | --- |
| QESRecepMensTcp | | |  |
|  | **Salida** | Eventos del tipo QESEvent (Subtipo QESTcpEvent o QESDispositivoEvent Correspondiente) | |
| **Función** | Recoge mensajes Tcp y los transforma en eventos. | |

|  |  |  |  |
| --- | --- | --- | --- |
| QESTimer<T> | | |  |
|  | **Salida** | Eventos del tipo T | |
| **Función** | Se indica una fecha y una acción que produce un evento. Cuando se alcance la fecha, el evento resultante de la acción se dispara | |

|  |  |  |  |
| --- | --- | --- | --- |
| QESAxisRecordingTimer | | |  |
|  | **Salida** | Eventos del tipo QESEvent(Subtipo QESAxisCameraEvent) | |
| **Función** | Al cumplirse una fecha, se envía un evento de grabación, cámara + duración en campo info. | |

|  |  |  |  |
| --- | --- | --- | --- |
| QESTail<T> | | | Interfaz |
|  | **Entrada** | Eventos del tipo T | |
| **Función** | Interfaz a implementar para definir un consumidor de eventos. | |

|  |  |  |  |
| --- | --- | --- | --- |
| AxisController | | |  |
|  | **Entrada** | Eventos del tipo QESEvent | |
| **Función** | Sin importar el contenido, cuando recibe un evento, realiza una grabación con los parámetros predefinidos. | |

|  |  |  |  |
| --- | --- | --- | --- |
| GenericAxisController | | |  |
|  | **Entrada** | Eventos del tipo QESEvent  (Subtipo QESTcpEvent o QESAxisCameraEvent) | |
| **Función** | Realiza una grabación con los datos indicados en el evento. El campo Info del evento puede utilizarse para definir la duración de la grabación. Hay una versión condicional que permite elegir que cámaras se grabará. | |

|  |  |  |  |
| --- | --- | --- | --- |
| QESEsendexSMS<T> | | | Abstracta |
|  | **Entrada** | Eventos del tipo T | |
| **Función** | Clase a ser heredada para definir el como enviar un sms desde la API de Esendex al recibir un evento. | |

|  |  |  |  |
| --- | --- | --- | --- |
| QESEsendexSMS | | |  |
|  | **Entrada** | Eventos del tipo QESEvent | |
| **Función** | Envía un SMS cuyo contenido es el atributo Info del evento recibido. | |

|  |  |  |  |
| --- | --- | --- | --- |
| QESMailSender<T> | | | Abstracta |
|  | **Entrada** | Eventos del tipo T | |
| **Función** | Clase a ser heredada para definir el como enviar un mail al recibir un evento. | |

|  |  |  |  |
| --- | --- | --- | --- |
| QESMailSender | | |  |
|  | **Entrada** | Eventos del tipo QESEvent (Mayor funcionalidad al obtener un evento tipo QESMailEvent) En caso de recibir otro tipo de evento se intentará adaptar la información que este proporciona. | |
| **Función** | Al recibir un evento envía un mail cuyo contenido es el atributo Info del evento. Si el evento es del tipo QESMailEvent se usa también el asunto y adjunto si existe. | |

|  |  |  |  |
| --- | --- | --- | --- |
| Pusher | | |  |
|  | **Entrada** | Eventos del tipo QESEvent  (Subtipo QESTcpEvent o QESAxisCameraEvent) | |
| **Función** | Obteniendo la cámara del evento, envía una notificación a través de Pusher.com para los clientes web. | |

|  |  |  |  |
| --- | --- | --- | --- |
| QESWebSocket | | |  |
|  | **Entrada** | Eventos del tipo QESEvent  (Subtipo QESTcpEvent o QESAxisCameraEvent) | |
| **Función** | Obteniendo la cámara del evento, envía una notificación a través de un WebSocket para los clientes web. | |

## Implementar un nuevo servicio

Antes de crear un nuevo servicio el primer paso sería intentar localizar uno existente y modificarlo al heredar, de estar forma reutilizaríamos la mayoría del código. Otra opción consiste en crear un nuevo servicio simplemente componiendo los existentes. Si ninguna de estas opciones consigue la funcionalidad que se desea, entonces se puede implementar uno nuevo.

### Código

#### Tipo

Antes de implementar un servicio, hay que decidir que tipo de servicio va a ser:

* *Consumidor*: consume eventos del sistema, como un servicio para mandar e-mails.
* *Productor*: produce eventos en el sistema, como un disparador cada X horas o un receptor de mensajes de móvil.
* *Consumidor/productor*: consume y produce eventos del sistema, normalmente para filtrarlos o tratarlos.

Así, si deseamos un consumidor, nuestro servicio tendrá que implementar la interfaz QESTail, en caso de que busquemos un productor heredaremos de la clase QESHead y si queremos con Consumidor/productor de la clase QESBody. De esta forma, los servicios se podrán conectar entre ellos según el rol que deseemos.

#### Parámetros persistentes

Si deseamos que el servicio tenga una serie de parámetros que se guarden y carguen en la base de datos con el resto, deberemos implementar la interfaz IParametrizable, esta interfaz define dos métodos para salvar/restablecer parámetros:

* *String[][] GetDataToStore()*:Devuelve en forma de array pares tipo {<clave>, <valor>}
* *Void RestoreFromData(String[]):* Dado una par {<clave>,<valor>} restablece un parámetro.

Con esto es suficiente, el gestor del servidor se encargará de salvar los parámetros en la base de datos y recuperarlos al iniciar la aplicación.

Es importante añadir el atributo override al método para que sobrescriba el comportamiento anterior. Se recomienda salvar/restaurar únicamente los atributos añadidos y llamar a base.metodo para terminar la carga/volcado.

#### Eventos

Es importante tener en cuenta que un servicio solo se podrá conectar con otros que tengan el mismo tipo que el, es decir, un productor de servicios QESEvent solo se podrá conectar con consumidores del tipo QESEvent, por lo cual se recomienda definir el tipo de eventos del servicio QESEvent, la base de toda la jerarquía de eventos, pudiendo ir en el cualquier clase heredada.

### Interfaz

Si deseamos que el servicio tenga una interfaz especifica, esta se puede crear como un “User Control” o “Control de usuario” en MS Visual Studio, MS Expression Blend o MS Expression Design.

Este tipo de interfaces ofrece una amplia gama de posibilidades, como efectos o videos. Puede consultarse más información sobre WPF online.

Si no se desea especificar una interfaz se asignara automáticamente la interfaz de servicio genérico, la cual solo muestra el nombre.

# Servidor

La aplicación servidor pretende ser el núcleo del sistema, en su interior se encuentra el gestor de eventos indicado anteriormente, es la única aplicación que accede directamente a la base de datos, contiene toda la configuración del sistema y ofrece diversos servicios a otras aplicaciones.

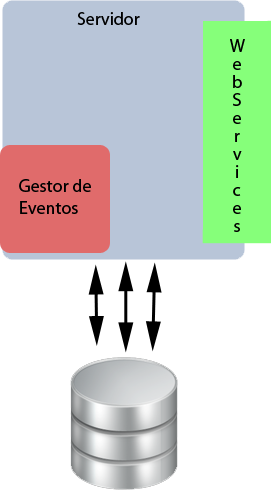
De esta forma las funciones del servidor son:

* Comunicación con la base de datos.
* Gestión de la configuración del sistema.
* Gestión general de cámaras, recintos y dispositivos.
* Control y tratamiento de eventos.
* Servicios para otras aplicaciones.

Para más información sobre la gestión de eventos consulte la sección anterior.

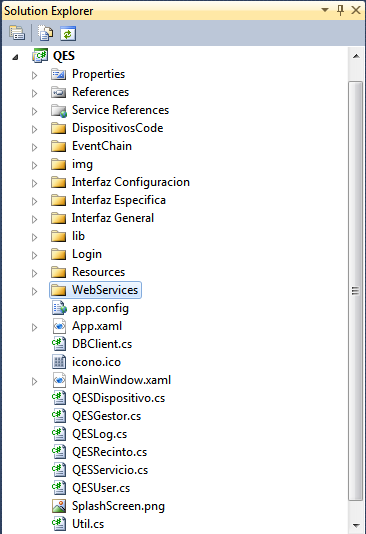
El servidor ha sido construido maximizando la extensibilidad y flexibilidad, de forma que resulte bastante sencillo añadir funcionalidades o configuraciones a este. La aplicación esta construida sobre .Net en C#, utilizando WPF para el diseño de las interfaces y WCF para la exposición de los servicios web.

## Visión General

El servidor será la aplicación principal del sistema, comunicándose con la base de datos, conteniendo la gestión de los eventos y publicando la información necesaria a través de servicios web. La aplicación esta orientada para ser utilizada por el administrador del sistema, ofreciendo diferentes opciones de configuración y visionado de los distintos servicios y dispositivos incluidos. Se presenta también el panel de usuarios para controlar el acceso que estos tienen a los diferentes dispositivos del sistema, al igual que un log donde se almacenan todas las acciones realizadas dentro de esta aplicación, desde el inicio de la aplicación al logeo de un usuario, pasando por el control de eventos. La aplicación utiliza un archivo, config.xml donde se guardan ciertos parámetros de configuración explicados mas adelante.

El servidor utiliza dos aplicaciones externas, IIS y el SGBD, que deberán ser configuradas por el usuario tal y como se indica para que este funcione correctamente.

## Estructura del proyecto

Los diferentes ficheros del proyecto se encuentran organizados por carpetas de la siguiente forma:

* DispositivosCode: Carpeta para los subtipos de QESDispositivo, que representan los diferentes dispositivos que el sistema acepta.
* EventChain: Carpeta para el gestor de eventos, a su vez se encuentra dividida en productores, consumidores y conectores.
* Img: todas las imágenes usadas por la aplicación
* Interfazconfiguracion: Clases para cambiar la configuración del sistema.
* Interfaz específica: Interfaces especificas de un dispositivo/servicio.
* Interfaz general: Interfaces generales del sistema (usuarios, dispositivitos, etc…)
* Login: Clases para el control del login.
* WebServices: Servicios web implementados en el sistema.

Además, hay una serie de archivos:

* App.xaml: starting point de la aplicación
* DBClient: Conexiones a la base de datos.
* MainWindow.xaml: Pantalla principal de la aplicación.
* QESDispotivo: Clase base para todos los dispositivos.
* QESGestor: Gestor general de toda la aplicación.
* QESLog: Control del log.
* QESRecinto: Recinto con cámaras.
* QESServicio: Clase base que contiene un servicio.
* QESUser: Usuarios y roles.
* Util: Extension methods para ampliar la funcionalidad de otras clases y clase para el tratdo del fichero xml.

## Servicios Web

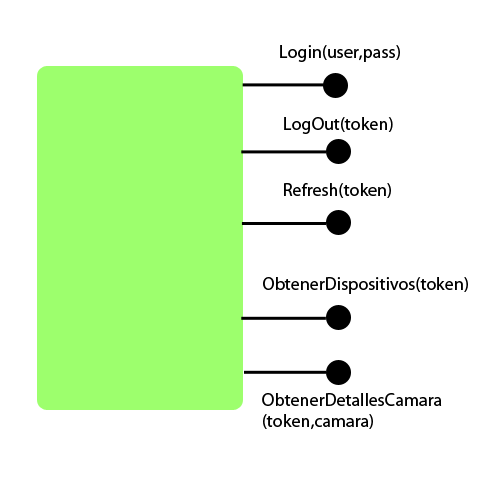
Se han implementado servicios web RESTful, con el objetivo de facilitar la interacción desde las aplicaciones al no tener que enviar el WDSL y poder mandar directamente JSON. Debido a que la información que se envía a través de los servicios web es de tipo sensible (usuarios, contraseñas, direcciones de cámaras,…), los servicios se exponen a través de HTTPS, configurado desde IIS.

Los contratos de los servicios se encuentran en el fichero “Contract.cs” y la implementación de estos en “QESService.cs”.

Al ser servicios web, el acceso a ellos es tan sencillo como acceder a una url para recoger los datos deseados/realizar la acción deseada.

Los servicios web se exponen con ayuda de la biblioteca WCF (Windows Comunication Foundation) en especial con WebServiceHost, debido a que la aplicación desarrollada no es una aplicación web, es una aplicación de escritorio, lo cual complica un poco el despliegue de los servicios web.

Para incrementar la seguridad, el usuario y la contraseña son enviados una única vez, proporcionando el servidor un token de identificación que será válido para la sesión actual. Reduciendo drásticamente de esta forma la probabilidad de que la contraseña sea interceptada, adicionalmente, el token tiene un tiempo de expiración, tras el cual, si el usuario no refresca la sesión, sera invalidado. De esta forma, junto con la seguridad que ofrecen los certificados SSL la conexión es altamente segura.

El servidor ofrece diferentes servicios web, tres relacionados con la sesión del usuario:

* Login: dado un usuario y una contraseña, devuelve un token de identificación y registra al usuario.
* Logout: dado un token, lo desasocia con el sistema.
* Refresh: renueva el token del usuario.

Y dos con los dispositivos:

* ObtenerDispositivos: Devuelve todos los recintos con las cámaras que contiene cada uno.
* ObtenerDetallesCamara: Devuelve todos los datos necesarios de una cámara.

Además de los servicios de administración:

* GetUsuarios: Obtiene una lista de usuarios con sus roles y su estado.
* CreateUser: Registra a un nuevo usuario en el sistema.
* UpdatePassword: Cambia la contraseña al usuario actual.
* GetServicios: Obtiene un listado de los servicios del sistema con su estado actual.
* SetServicio: Activa/desactiva un servicio.
* GetPrivilegios: Obtiene un listado de los roles asignados a un usuario.
* SetPrivilegios: Establece que roles tiene un usuario.
* DesloagearA: Termina la sesión de un usuario determinado.

Algunos servicios devuelven un tipo de dato específico como:

|  |
| --- |
| enum CODE {SUCCESS = 0, SESSIONEXPIRED = -1, UNAUTHORIZED = -2, USERPASSWORDWRONG = -5, NAMEALREADYTAKEN = -7 } |

|  |
| --- |
| public class LoginReturn  {  public int Code { get; set; }  public String Token { get; set; }  public String Info { get; set; }  } |

|  |
| --- |
| public class UsuariosReturn  {  public int Code { get; set; }  public Usuario[] Usuarios { get; set; }  public class Usuario  {  public string Nombre { get; set; }  public Boolean Conectado { get; set; }  public String[] Roles { get; set; }  }  } |

|  |
| --- |
| public class PrivilegiosReturn  {  public int Code { get; set; }  public Privilegio[] Privilegios { get; set; }  public class Privilegio  {  public String Nombre { get; set; }  public Boolean Status { get; set; }  }  } |

|  |
| --- |
| public class DispositivosReturn  {  public class Recinto  {  public class Dispositivo  {  public String Nombre { get; set; }  public int X { get; set; }  public int Y { get; set; }  public String Type { get; set; }  }  public String Nombre { get; set; }  public String Plano { get; set; }  public Dispositivo[] Dispositivos{get;set;}  }  public int Code { set; get; }  public Recinto[] Recintos { get; set; }  } |

|  |
| --- |
| public class CamaraDetailReturn  {  public int Code { get; set; }  public string Nombre { get; set; }  public string IP { get; set; }  public int Puerto { get; set; }  public string User { get; set; }  public string Password { get; set; }  public bool Audio { get; set; }  public bool AudioOut { get; set; }  public bool PTZ { get; set; }  } |

|  |
| --- |
| public class ServiciosReturn  {  public int Code { get; set; }  public Servicio[] Servicios { get; set; }  public class Servicio  {  public string Nombre { get; set; }  public string Status { get; set; }  }  } |

De esta forma, la aplicación web contendría los siguientes servicios de un modo esquemático:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre** | **Método** | **Entrada** | **Salida** | **Descripción** | **Nota** |
| Login | GET | * user: Usuario * password: Contraseña | LoginReturn | Autentica a un usuario contra el sistema. | Se permiten varias sesiones por usuario |
| LogOut | POST | * token: Token de autenticación |  | Cierra la sesión de un usuario. |  |
| Refresh | POST | * token: Token de autenticación |  | Renueva la sesión de un usuario. |  |
| Get Dispositivos | GET | * token: Token de autenticación | Dispositivos Return | Obtiene un listado de los dispositivos. | Solo se obtienen los que usuario tiene privilegios para ver. |
| Get Detalles Camara | GET | * token: Token de autenticación. * Cámara: Camara de la que se solicitan los datos | Camara Detail Return | Obtiene los detalles para poder visualizar el stream de una camara |  |
| GetUsuarios | GET | * token: Token de autenticación. | Usuarios Return | Obtiene un listado de los usuarios junto con su estado. | [ADMIN] |
| Crear Usuario | POST | * token: Token de autenticación * name: Nombre del nuevo usuario * password: Contraseña del usuario | Código de confirmación(CODE) | Registra a un nuevo usuario en el sistema. | Dos usuarios no pueden tener el mismo nombre. [ADMIN] |
| Update Password | POST | * token: Token de autenticación. * Old: Contraseña actual * Password: Nueva contraseña | Código de confirmación (CODE) | Cambia la contraseña del usuario. | El usuario se obtiene a través del token de autenticación. |
| Get Servicios | GET | * Token: Token de autenticación. | Servicios Return | Obtiene un listado de los servicios en el sistema y su estado actual | [ADMIN] |
| Set Servicio | POST | * Token: Token de autenticación. * Servicio: Nombre del servicio. * Status: True = activo False = inactivo | Código de confirmación (CODE) | Cambia el estado de un servicio | Si se activa un servicio activo no ocurre nada, al igual que si se desactiva un inactivo. [ADMIN] |
| Get Privilegios | GET | * Token: Token de autenticación. * User: Usuario a consultar | Privilegios Return | Devuelve una lista con todos los roles y un campo que indica si el usuario lo posee. | [ADMIN] |
| Set Privilegios | POST | * Token: Token de autenticación. * User: Usuario * Priv: Rol * Value: Asignado/ No | Código de confirmación (CODE) | Asigna o quita un rol a un usuario | Si se asigna un rol que tiene subroles, el usuario los posee automáticamente, si se quita un rol pero posee un super rol este no se podrá quitar. [ADMIN] |
| DeslogearA | POST | * Token: Token de autenticación. * Usuario: Usuario a expulsar | Código de confirmación (CODE) | Expira la sesión de un usuario | [ADMIN] |

[ADMIN]: Requiere el rol de administrador.

Para agregar un nuevo servicio web, será suficiente con agregarlo al contrato, e implementarlo en la clase QESService.

## Base de datos

El servidor utiliza una base de datos para almacenar información relativa al sistema y que es producida durante la ejecución o configurable por el usuario.

No es necesario que la base de datos se encuentre en el mismo ordenador que el servidor, permitiendo de esta forma utilizar un centro de datos o almacenamiento en nube para la aplicación.

Las conexiones actualmente implementadas para conectar con la base de datos son a través de:

* Driver PostgreSQL
* Driver ODBC

La primera opción es mas eficiente que la segunda, pero solo es valida para bases de datos PostgreSQL, mientras que el driver ODBC permite utilizar cualquier base de datos, además, crea una nueva capa de abstracción que permite cambiar fácilmente la conexión a la base de datos, incrementando aun más la flexibilidad de la aplicación.

La base de datos incluye las siguientes tablas:

* Log: entradas del log.
* Params: Parámetros de configuración de los servicios configurables por el usuario.
* User: Usuarios del sistema.

### Introducir un nuevo conector

Si se desea introducir un nuevo conector a la aplicación es tan sencillo como implementar la clase DBClient e instanciarla al seleccionar el conector en el gestor. Hay una serie de métodos que se deben implementar, los definidos por la interfaz, para que el conector funcione correctamente.

## Archivo de configuración

La aplicación además utiliza un archivo de configuración, config.xml, el cual contiene ciertos parámetros de configuración que no corresponden a ser almacenados en la base de datos.

El fichero es un fichero de tipo XML que la aplicación gestiona automáticamente por lo que no se recomienda editarlo a mano.

En este se almacena:

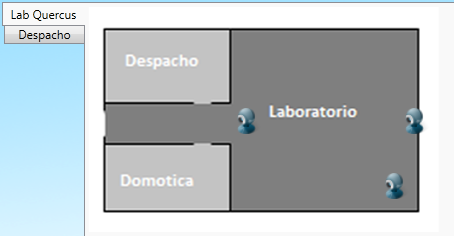
* Configuración de la conexión de la base de datos.
* Tiempo tras el que un usuario será deslogeado de los servicios web.

Se pueden introducir otras configuraciones añadiendo los métodos necesarios a la clase XmlFileManager en el archivo Util.cs

## Interfaz de usuario

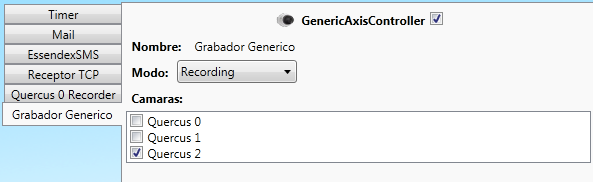
La interfaz de usuario del servidor, construida con WPF en VS se encuentra dividida en siete pestañas.

### Recintos



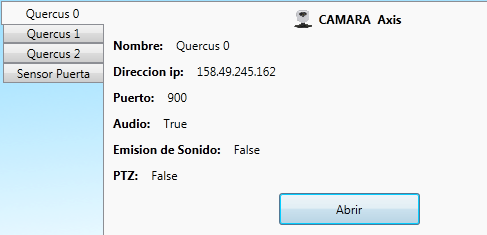
Esta pestaña muestra todos los dispositivos en el plano agrupados por recintos. Si pinchamos en uno de los dispositivos en el mapa, una ventana emergente nos visualizara su contenido (streaming de video para las cámaras)

### Servicios



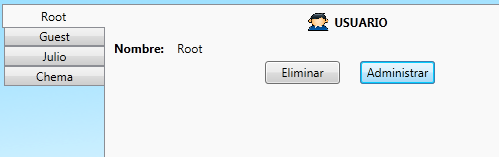
En la pestaña de Servicios a la derecha aparecerá una lista de los servicios activos en el sistema, pudiendo pinchar en ellos individualmente para ver sus detalles y editar su configuración. Cada servicio tiene una interfaz diferente permitiendo editar los detalles propios de cada uno.

### Dispositivos



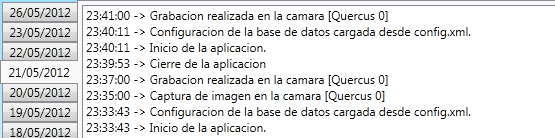
En esta pestaña se listan todos los dispositivos configurados en el sistema, mostrando los detalles de cada uno, así como permitiendo visualizar su contenido.

### Usuarios



Esta pestaña permite visualizar y configurar los usuarios que tiene el sistema así como crear nuevos. Al hacer click en Administrar un usuario podremos cambiar los roles que tiene asociados, teniendo en cuenta que al activar un rol que tiene algún subrol, los subroles se asignaran también.

### Log



En esta pestaña se puede ver el log del sistema agrupado por días. Este log recopila información sobre los usuarios, grabaciones, dispositivos, configuración, etc…

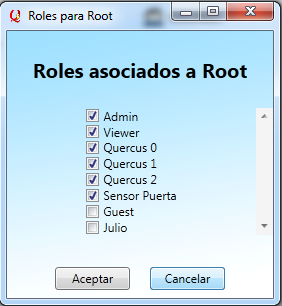
### Configuración



En este panel podemos acceder a la configuración general del sistema. Nos permite crear usuarios, cambiar el intervalo de tiempo tras el que la sesión de un usuario expira, vaciar el log, apagar/encender los servicios web y configurar la conexión a la base de datos. Este último dispone de un pequeño asistente para comprobar que la conexión se puede establecer correctamente.

## Usuarios y Roles

El sistema permite una configuración de usuarios y roles con el objetivo de establecer diferentes niveles de usuarios y permitir que un administrador controle estos. Así, se definen una serie de roles básicos, un administrador, un rol para cada dispositivos y un rol para todos los dispositivos. Con esta base, el administrador puede crear nuevos roles de la conjunción de los existentes.

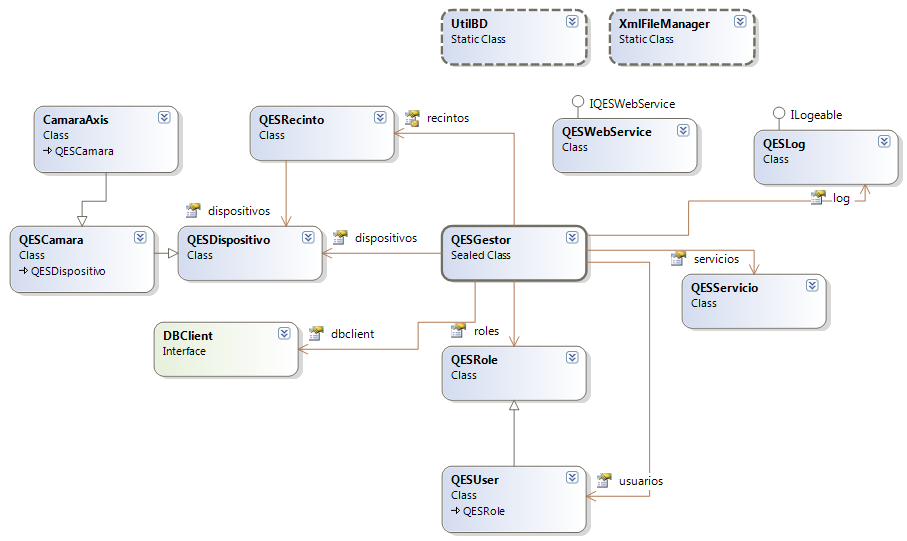
Además, un rol puede tener un conjunto de roles de los que hereda todas sus propiedades (Como administrador que hereda todos o “Viewer” que hereda el de todos los dispositivos). De forma que si un usuario posee el rol Administrador también poseerá el rol de una cámara especifica.

Junto a los roles tenemos los usuarios, que representan a las personas que acceden al sistema. Estos usuarios tendrán una serie de roles que representaran sus capacidades dentro del sistema. Para añadir más flexibilidad, los usuarios se han definido de forma que un usuario puede tener como rol a otro usuario, de forma que hereda todos sus roles y privilegios convirtiendo a los usuarios de esta forma en un rol con contraseña.

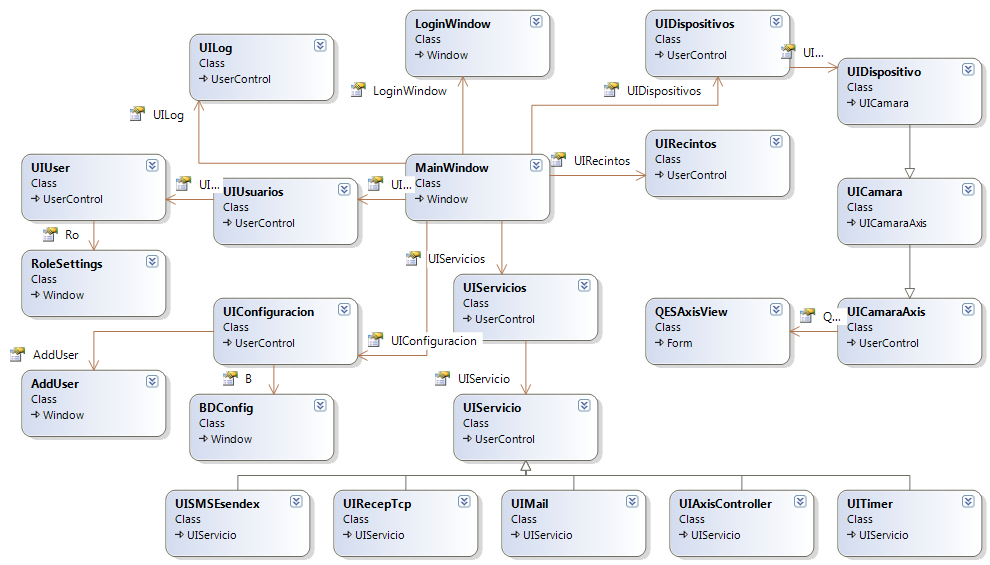
## Diseño

### Diagramas de clases Conceptuales

#### Lógica del sistema:

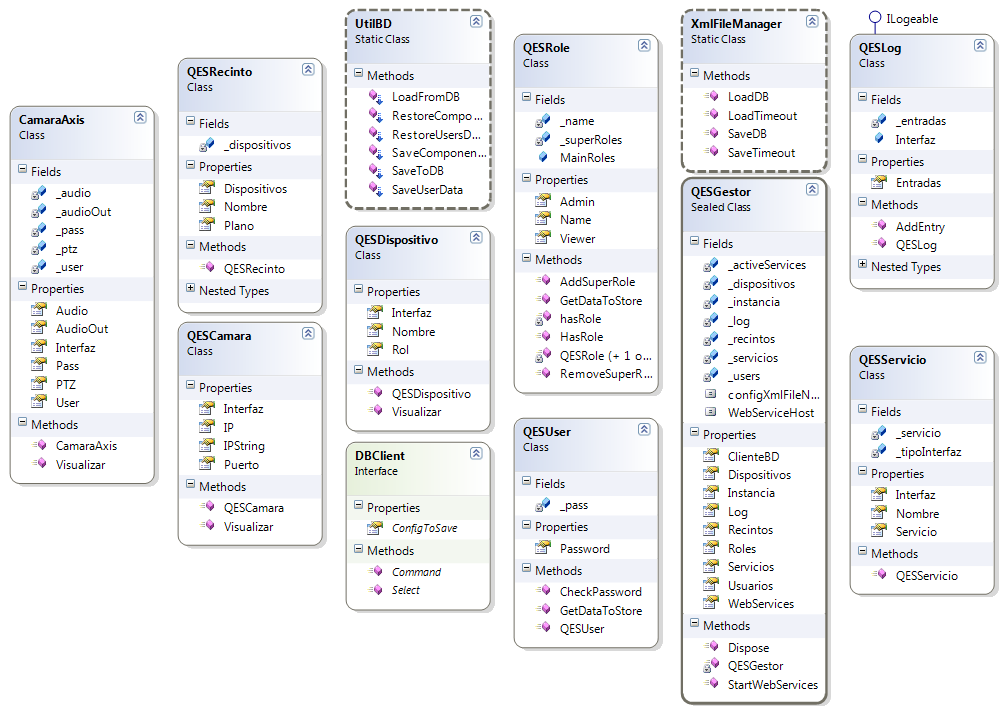


#### Interfaz:

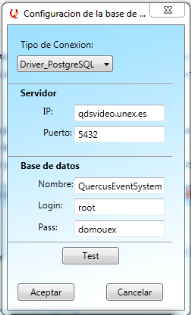


### Diagrama de Clases

Por claridad no se pueden mostrar las relaciones en este diagrama, estas se encuentran en el diagrama de clases conceptuales.



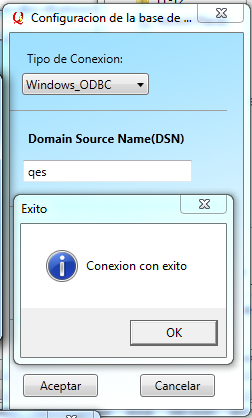
## Configuración de la base de datos

La configuración de la base de datos actualmente admite dos modos:

* Driver PostgreSQL
* Windows ODBC

Si elegimos utilizar el driver de PostgreSQL deberemos indicar los parámetros de configuración y probar la conexión, de forma que el servidor se conectará directamente a la base de datos.

Para utilizar Windows ODBC deberemos configurar primero una entrada en el asistente ODBC (en Windows/sysWow64 para Windows 7) y después configurarlo en la aplicación simplemente indicando el nombre.



## Instalación

La aplicación servidor requiere primero la instalación de:

* .Net Framework 4.0
* Sistema Gestor de Base de Datos
* Internet Information Services
* Axis Media Control

### .Net Framework 4.0

El proyecto utiliza este framework, de forma que es necesario que este instalado para que este se ejecute correctamente. Puede descargarse esta versión en <http://www.microsoft.com/en-us/download/details.aspx?id=17851>

### Sistema Gestor de Base de Datos

El sistema necesita de una base de datos, actualmente se soportan conexiones a través de un driver PostgreSQL y a través de Windows ODBC.

Para configurar el ODBC debemos acceder a la herramienta de Windows de 32 bits (ya que la aplicación esta implementada para 32 bits) ubicada en la carpeta syswow64 en Windows 7.

Podemos descargarlo en el siguiente enlace: <http://www.postgresql.org/download/>

### Internet Information Services

Para poder lanzar los servicios web en modo seguro es necesario tener este servidor web instalado y configurado con un certificado SSL.

Para instalarlo deberemos ir a “activar/desactivar características de Windows”.

### Axis Media Control

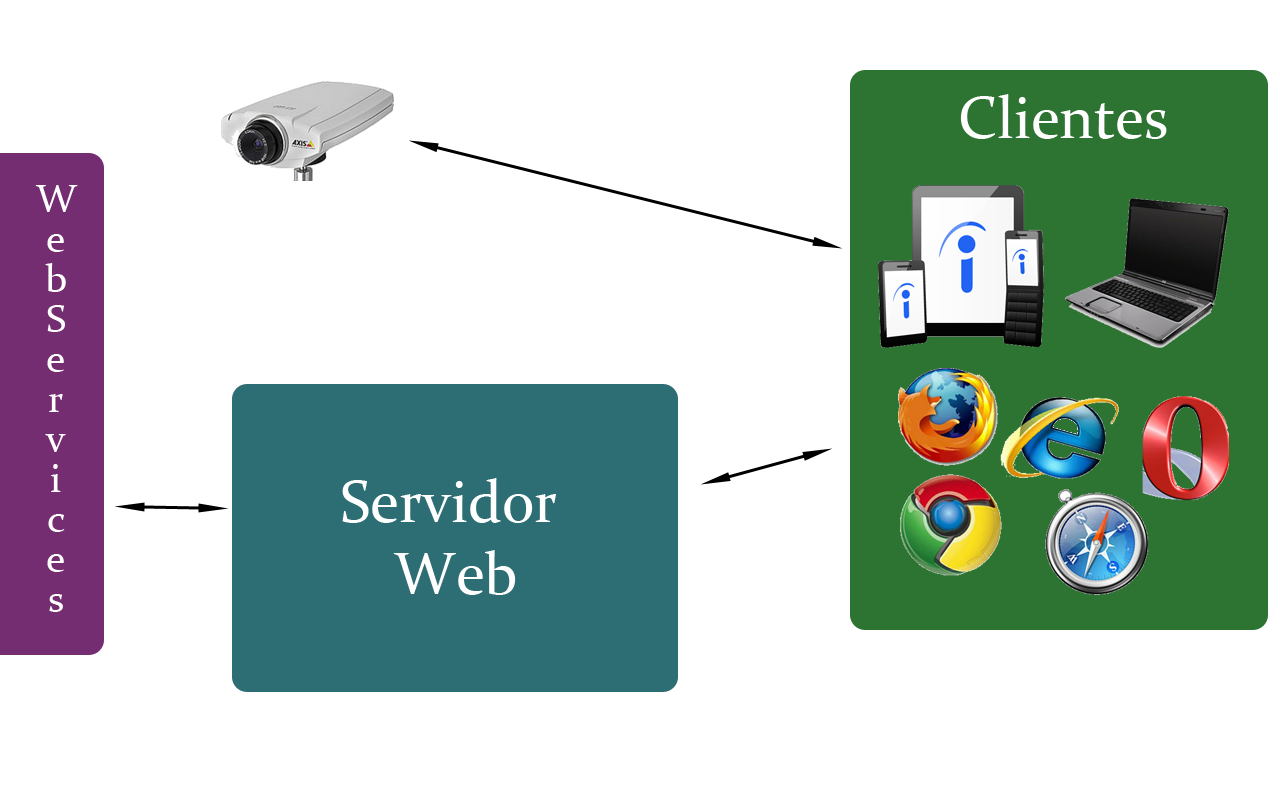
Este ActiveX es necesario para poder trabajar con las cámaras axis, tanto para su visualización como para los servicios de grabación.

Podemos encontrar el instalador en: <http://www.axis.com/techsup/cam_servers/dev/activex.htm>

Con todos los requisitos cumplidos, puede instalar la aplicación a través del asistente.

# Servidor Web

## Visión General



El servidor web, implementado en HTML5 y JavaScript y levantado sobre IIS responde a las peticiones de los clientes, los navegadores web. Este servidor web no contiene datos, solo una capa de presentación y la lógica de ejecución, de esta forma, todos los datos (Usuarios, cámaras, dispositivos, recintos, etc…) se obtiene a través de una comunicación con servicios web RESTfull desplegados en el servidor C#. Es decir, cuando un cliente solicita el acceso al sistema, este es autenticado sobre el servidor, a través de esos servicios web.

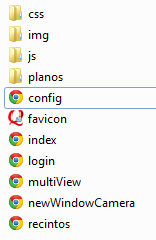
Así, cuando un usuario desea acceder al flujo de video de una cámara, todos los datos son cargados desde el servidor y el cliente, en su navegador, establece una conexión directa con la cámara para obtener el flujo de video. Se optó por esta vía en lugar de pasar el flujo por el servidor para reducir la latencia y la carga en el servidor.

## Seguridad

La seguridad es una parte clave del sistema y por ello, todas las conexiones se establecen de forma segura, sobre HTTPS con un certificado que debe crearse al configurar IIS.

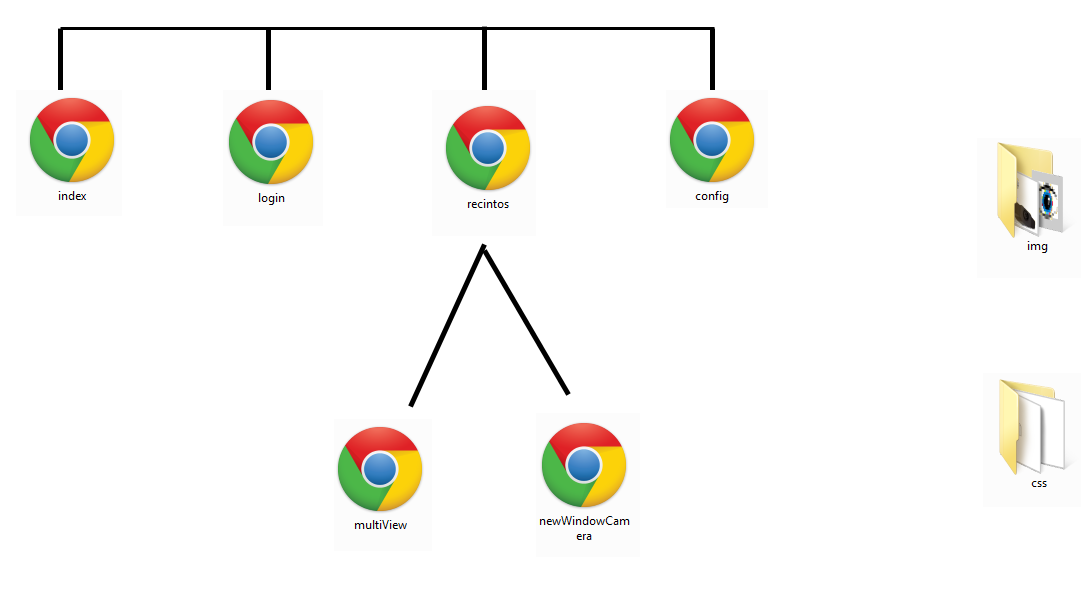
Además, el usuario y la contraseña se envían únicamente una vez, devolviendo el servidor un token de autenticación que es usado durante la ejecución con el objetivo de incrementar la seguridad, reduciendo drásticamente el número de veces que la contraseña viaja por la red.

## Estructura del proyecto

El proyecto se encuentra dividido en carpetas.

* En la carpeta raíz podemos encontrar todo el código HTML, las diferentes páginas del WebSite.
* CSS almacena todas las hojas de estilo (Cascade Style Sheet).
* La carpeta img contiene las imágenes usadas por la web.
* Js almacena todos los scripts externos, importados en la mayoría de las páginas.
* En planos se sitúan las imágenes a mostrar al abrir los recintos.

## Árbol de navegación



Aun con este árbol de navegación es importante tener en cuenta que solo las páginas index y login pueden ser accedidas si el usuario no esta logeado, para el acceso al resto, es necesario que el usuario inicie sesión previamente.

La carpeta img es común para todas las páginas, al igual que css y los scripts.

## Descripción de las páginas

### Index.html

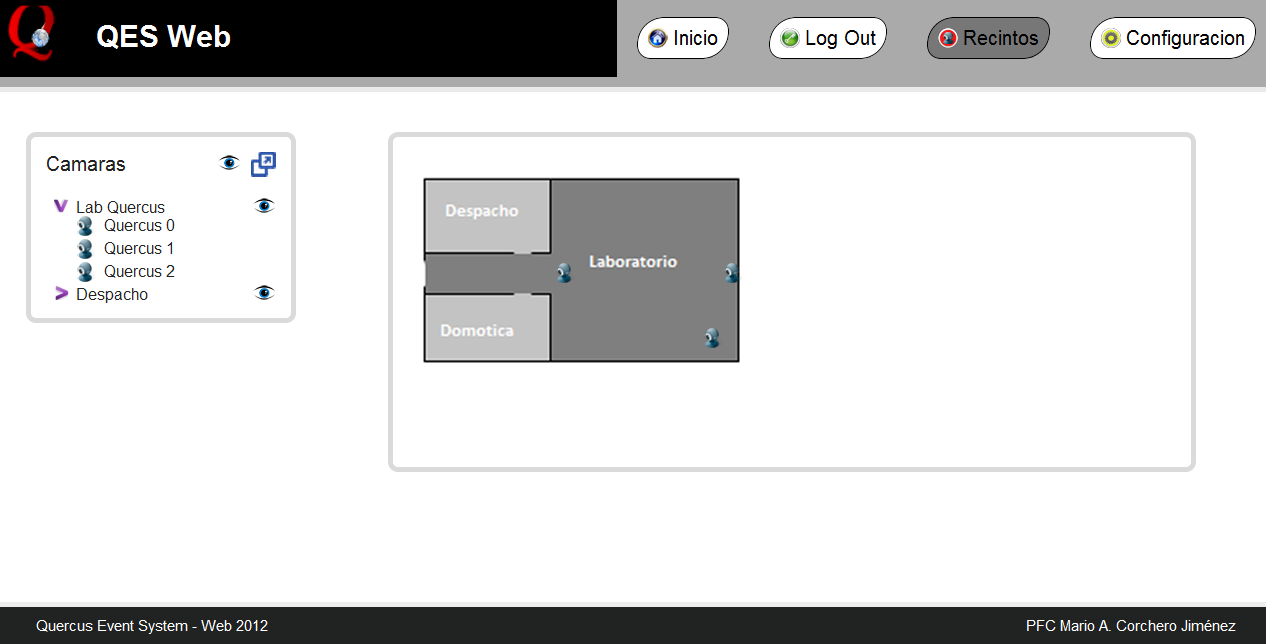


Esta es la land page del website, muestra algunos enlaces de interés, el header común a todas las paginas así como el footer. En el centro encontramos un mensaje de bienvenida.

### Login.html

Esta página permite al usuario autenticarse contra el sistema. Cuando se introduce el usuario y la contraseña, estas se mandan al servidor C# que comprueba que son correctas y devuelve un token de autenticación que caduca tras un determinado tiempo. Este token se refresca siempre que el usuario este conectado. Es totalmente necesario que el servidor este levantado para proceder a la autenticación, en caso contrario se obtendrá un mensaje: “Error al comunicar con el servidor”.

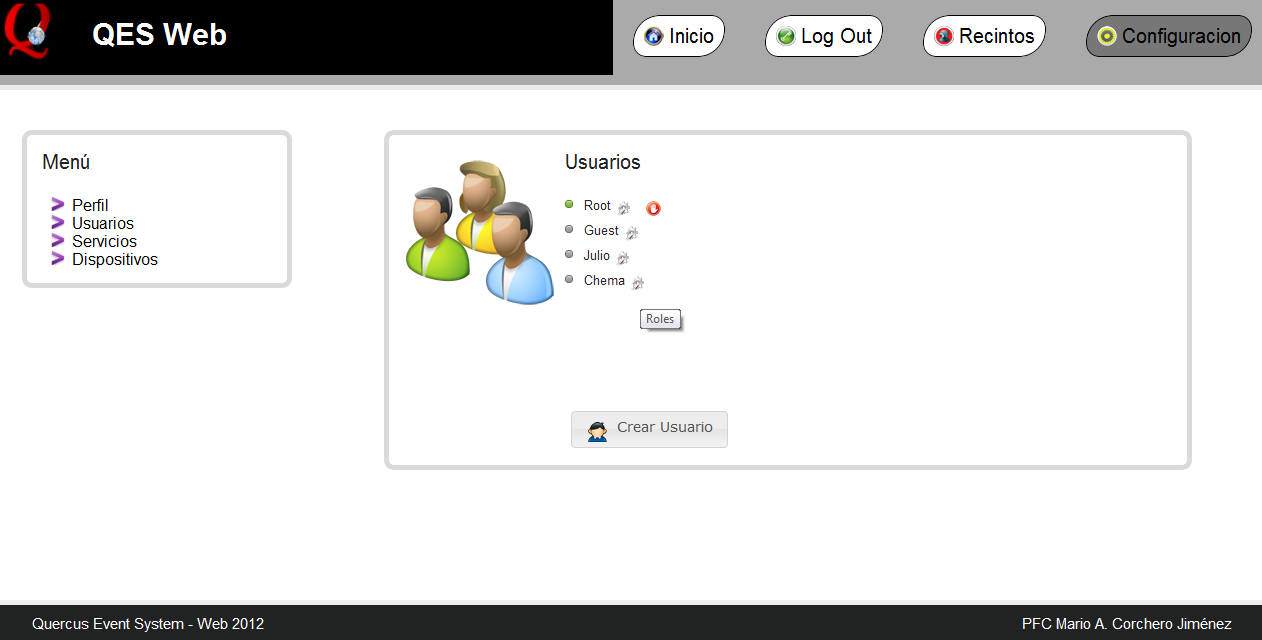
### Recintos.html



Esta es la principal página para visualizar los dispositivos, a la izquierda tenemos un panel que nos permite seleccionar los dispositivos a mostrar y como se muestran. Si hacemos click en el icono  abriremos todos los dispositivos dentro de un recinto o del sistema entero. Arriba a la derecha, tenemos un icono que nos permite cambiar el modo de vista:

* Vista en Ventana: Al hacer click en un recinto se mostrara el mapa de este, pudiendo hacer click sobre la cámara tanto en el mapa del recinto como en el menú desplegable en la izquierda. Al abrir una cámara esta se mostrara en una ventana emergente. Para cerrar la cámara basta con cerrar la ventana, si queremos cambiar el tamaño, podemos redimensionarla a placer, ajustándose el tamaño de la imagen al tamaño de la ventana.
* Vista en Panel: El panel derecho se transforma en la paleta de visualización, en esta se despliegan las cámaras sobre las que hagamos click. Estas pueden cambiar de posición arrastrándolas, redimensionarlas como cualquier ventana (para mantener proporción pulsaremos shift) y cerrarse haciendo doble click sobre ellas.

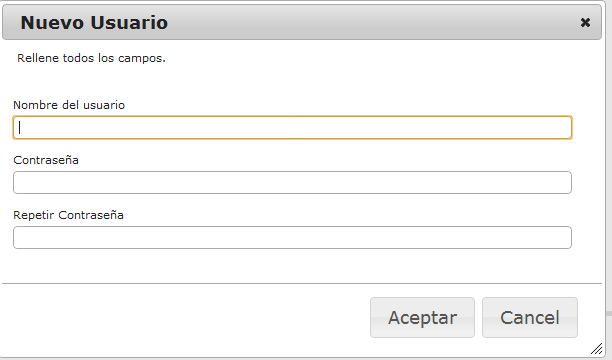
### Config.html



En esta página podemos acceder a la configuración general del sistema. En función de los privilegios que posea el usuario le aparecerán distintos menús en el panel de la izquierda. Perfil esta accesible a todos los usuarios, mientras que el resto únicamente a los administradores.

Los menús accesibles desde esta página son:

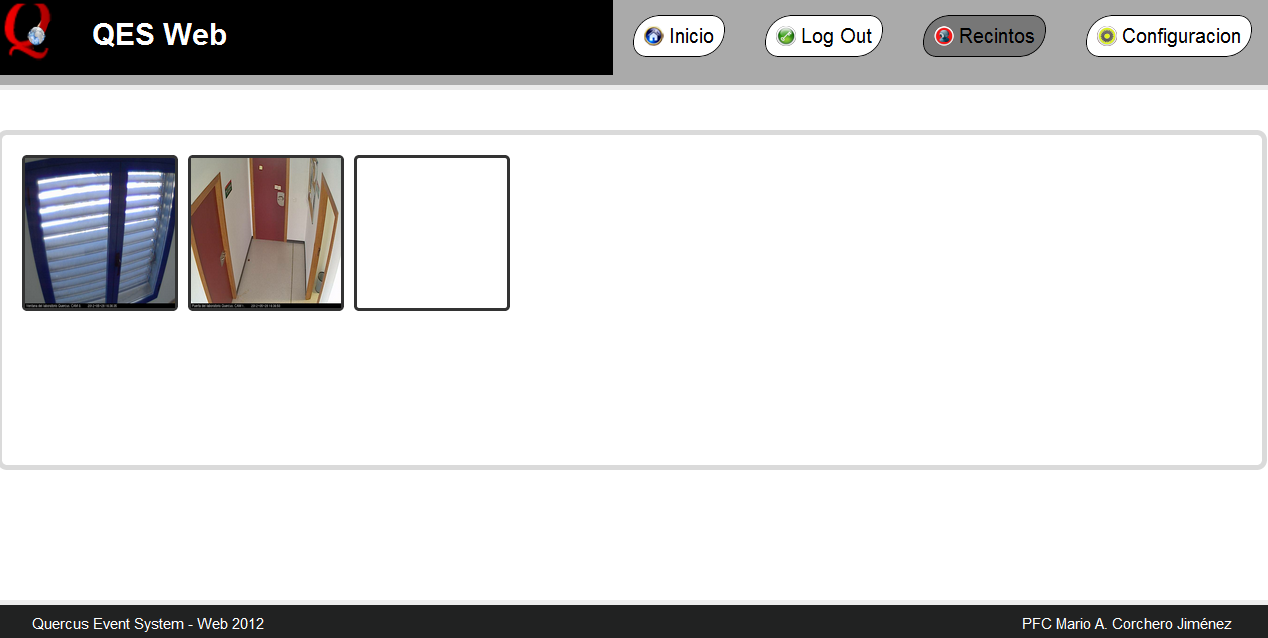
* Perfil: Permite cambiar la contraseña a los usuarios o deslogearse del sistema.
* Usuarios: Muestra todos los usuarios del sistema, con un led verde () para los conectados. A la derecha de los usuarios hay uno o dos iconos. El primero, permite configurar los roles que el usuario tiene asociado, mientras que el segundo expulsa al usuario del sistema (este icono estará disponible solo si el usuario esta actualmente dentro del sistema). Además, hay un botón al final del área de la derecha para crear un usuario. 

Este dialogo comprobara que los datos introducidos son correctos y creara un nuevo usuario en el sistema a través de los servicios web del servidor. Se comprueba que no haya un usuario con el mismo nombre y que la contraseña y el nombre de usuario son adecuados.

* Servicios: Este menú permite visualizar el estado actual de los servicios activados en el sistema y activarlos/desactivarlos pinchando sobre ellos. 
* Dispositivos: En este menú podemos visualizar todos los dispositivos así como los detalles de estos.

### MultiView.html

Esta pagina muestra un grupo de cámaras en una sola ventana, se accede a través de la página recintos al hacer click en uno de los iconos señalados anteriormente.



En esta ventana, para maximizar una de las cámaras basta con hacer click encima de ella para que triplique su tamaño. Además, a la izquierda hay un menú escondido que al pasar el ratón por encima nos permite observar otros recintos.

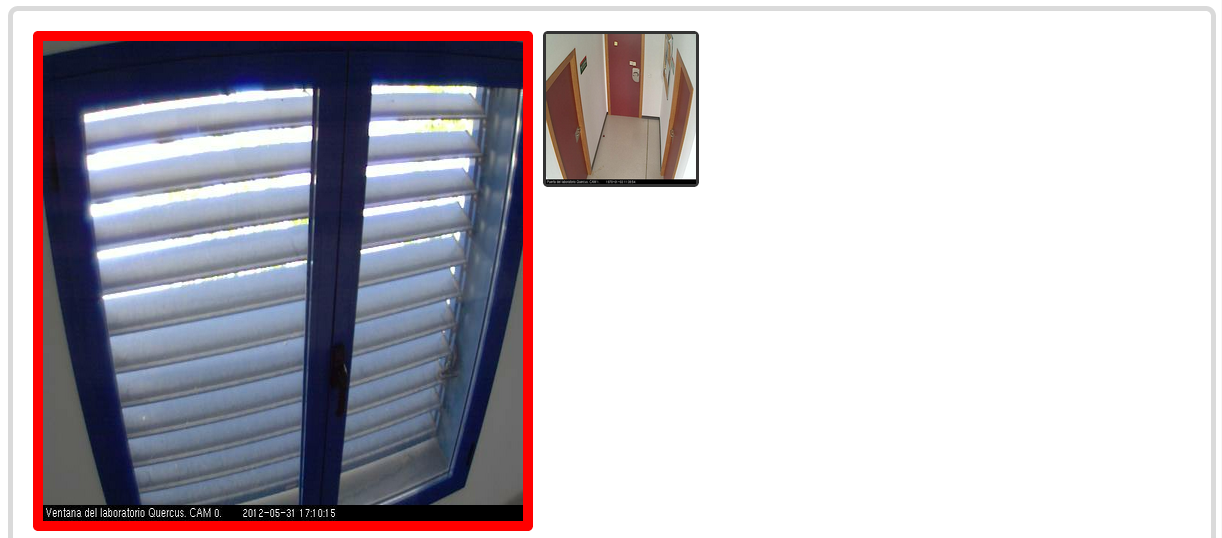
## Script

El WebSite utiliza un script, “qesscript.js” el cual contiene diferentes funciones que son utilizadas para manejar las cookies, refrescar la sesión, mostrar las cámaras y otras funciones auxiliares.

Además, en la segunda línea, “*var server = "https://localhost/WS/";*” la cual define la ubicación de los servicios web a los que se accederá para comunicarse con el servidor.

## Notificaciones desde el servidor

En la ventana de visualización de cámaras, se ha implementado a través de pusher y de WebSockets la posibilidad de recibir eventos desde el servidor, de forma que cuando se produzca un evento sobre un dispositivo en este dispositivo, aumentará de tamaño automáticamente, su borde incrementara su grosor y cambiará de color a la vez que se reproducirá n aviso sonoro para llamar la atención.



La configuración de este servicio se establece en el fichero *qesscript.js*.

## Conexión a las cámaras

La captación del flujo de video se realiza mediante el siguiente proceso:

Una vez que el cliente se autentica contra el servidor y obtiene la información general de los dispositivos mostradas en el mapa, al hacer click en la cámara, se solicitan todos los datos específicos de esta al servidor, y con estos, se establece la comunicación directamente con el dispositivo, sin pasar por el servidor. De esta forma se carga menos la aplicación servidor y se consigue video en tiempo real, puesto que pasar el video a través del servidor podría provocar una sobrecarga de este y podría producir un retraso considerable en el video.

## Soporte

El servidor web pretende ser tan portable como sea posible, por ello no utiliza plugins dependientes de ninguna arquitectura específica. Esta construido en su totalidad en HTML5 y JavaScript, utilizando también la librería JQuery (y JQuery UI). De esta forma, la única compatibilidad que ha tenido que tenerse en cuenta es la de diferentes navegadores.

### Internet Explorer

La versión 9.0 y posteriores de este navegador están soportados, pero no las anteriores debido a problemas de incompatibilidad con HTML5. Así, cuando se entra en el WebSite con una versión anterior el usuario es invitado a actualizar su navegador o utilizar otro para el correcto funcionamiento de la aplicación.

Debido a que este navegador tiene problemas para mostrar el stream de video, pero acepta el uso de ActiveX (lo cual ningún otro navegador soporta), para mostrar las cámaras se utiliza Axis Media Control.

Otro problema con Internet Explorer es el soporte del uso de la herramienta Pusher, la cual no es aceptada por este ya que carga los websockets en otro dominio (el de pusher).

### Firefox y Safari



Ambos navegadores soportan todas las características de la web en su totalidad.

### Google Chrome

Este navegador soporta el sistema en su totalidad salvo en la versión 19, debido a que en esta versión se desactiva la autenticación básica sobre URL, utilizada para autenticarse en las cámaras axis, aun asi, hay un plugin que permite solucionar este comportamiento. El plugin esta disponible en la siguiente url:

“[*http://code.google.com/p/url-embedded-auth/downloads/detail?name=HTTPAuth.crx*](http://code.google.com/p/url-embedded-auth/downloads/detail?name=HTTPAuth.crx)”

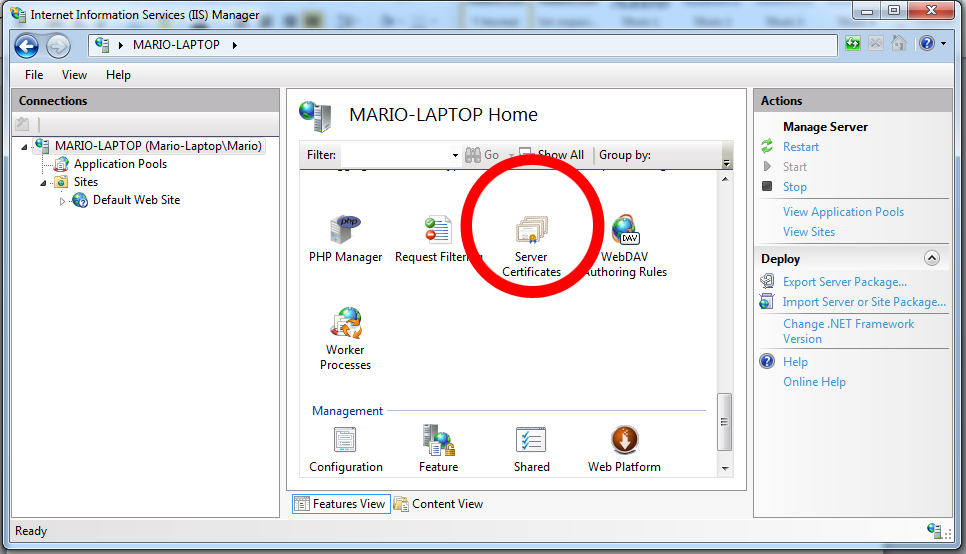
Este cambio a sido duramente criticado en la comunidad open source debido a que parece inadecuado desactivar por completo una herramienta estándar[13] que es actualmente usada por gran multitud de sistema, sin siquiera dejar la posibilidad de activarla manualmente. Por ello, como se puede leer en el link proporcionado a continuación, esta herramienta será incluida en las siguientes versiones.

“[*http:/code.google.com/p/chromium/issues/detail?id=123150*](http://code.google.com/p/chromium/issues/detail?id=123150)”

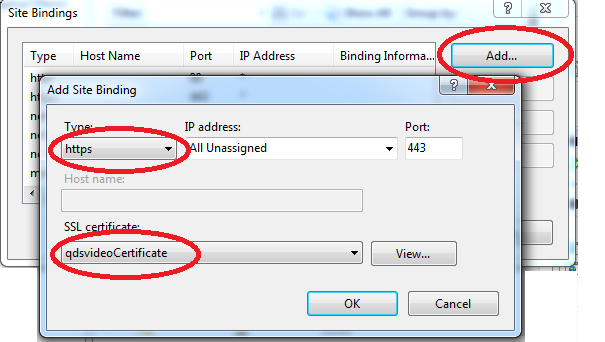
## Instalación

La instalación del servidor web requiere previamente la instalación de IIS y del servidor C# así como la correcta configuración de ambos.

Para instalar IIS debemos ir a el panel de control de Windows y a la opción “Activar/desactivar características de Windows”, y una vez en esta ventana deberemos activar el checkbox de Internet Information Services(Servicios de Información de Internet). Tras tener IIS instalado podremos acceder al administrador de este, donde deberemos de pinchar en Certificados del Servidor para importar un certificado que tengamos ya comprado o crear uno auto firmado (Esto hará que al acceder desde el navegador nos salga una advertencia).



Tras esto, haciendo click derecho en “Defailt Web Site” o “Sitio web por defecto” en bindings o enlaces añadiremos un nuevo enlace, para vincular el certificado con la entrada HTTPS en nuestro servidor.

Es importante que en Tipo seleccionemos https, que comprobemos que el puerto es el 443 y que el certificado es el que importamos o creamos anteriormente en IIS. Tras esto, pincharemos en aceptar en ambos diálogos. Si podemos acceder a “[*https://localhost*](https://localhost)” significara que IIS se ha configurado correctamente.

Una vez instalado y configurado IIS debemos indicar la ruta de nuestra web o mover todos los archivos a la ruta por defecto de IIS “*C:\inetpub\wwwroot”*.

Ahora podremos acceder a la web a través de nuestro dominio con https. Es importante que escribamos el nombre de nuestro dominio y no localhost puesto que sino la conexión con los no podrá establecerse, al intentar acceder desde un dominio (localhost) a otro (midominio.com) para utilizar los web services.

Si deseamos que el servidor redirija de HTTP a HTTPS deberemos de crear un website y asociarle en enlaces el puerto 80 (el sitio con HTTPS debe tenerlo desactivado). Una vez creado el sistio debemos dirigirnos a Redirección HTTP e indicar la dirección https deseada. Es importante que no se indique la misma ruta física para ambos.

Por ultimo es importante abrir el puerto 443 a través del firewall de Windows para que los usuarios puedan acceder desde el exterior

Es importante asegurarse de que el fichero qesscript.js contiene en la segunda línea la dirección correcta a los servicios web aunque esto sea hecho automáticamente a través del DSL.

Además, para personalizar el servicio de eventos, deberemos actualizar las líneas:

|  |
| --- |
| var pusherKey = '9a3fe2f64106d87e0291';  var pusherChannel = 'QESTest';  var soundFile = "beep.wav"; |

Con los datos obtenidos al crear una cuenta en “[*http://www.pusher.com*](http://www.pusher.com)*”,* esta configuración debe estar también reflejada en la parte del servidor. Es importante tener en cuenta la compatibilidad de los archivos de audio al indicar uno nuevo, puesto que no todos los navegadores aceptan por igual los distintos tipos de archivos de audio. La opción mas recomendada es “*Waveform Audio File Format*”(*.wav)* tal y como se muestra en la siguiente tabla[14]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Ogg Vorbis** | **WAV PCM** | **MP3** | **AAC** | **Speex** |
| **Trident** | No | No | 5.0 | 5.0 | No |
| **Gecko** | 1.9.1 | 1.9.1 | No | No | No |
| **WebKit** | Depends | 525 | 525 | 525 | Depends |
| **Presto** | 2.5 | 2.0 | Depends | Depends | No |

Una solución muy utilizada es también incluir varios ficheros con diferentes formatos, pero por comodidad para el usuario, solo se requiere un que sea wav ya que es actualmente el mas compatible.

# Referencias

[1] J. Liberty, *Programming C#*, 3rd ed. Sebastopol, CA, USA: O’Reilly &amp; Associates, Inc., 2003.

[2] A. Jones and A. Freeman, ‘Windows Presentation Foundation’, in *Visual C# 2010 Recipes*, Berkeley, CA: Apress, 2010, pp. 789–904.

[3] J. Rao and X. Su, ‘A Survey of Automated Web Service Composition Methods’, in *Semantic Web Services and Web Process Composition*, vol. 3387, J. Cardoso and A. Sheth, Eds. Springer Berlin / Heidelberg, 2005, pp. 43–54.

[4] D. Crockford, ‘The application/json Media Type for JavaScript Object Notation (JSON)’, Jul-2006. [Online]. Available: http://tools.ietf.org/html/rfc4627. [Accessed: 09-Jun-2012].

[5] M. MacDonald, *HTML5: The Missing Manual*. O’Reilly Media, Inc., 2011.

[6] D. S. McFarland, *JavaScript & JQuery: The Missing Manual*. 2011.

[7] Wikipedia contributors, ‘Ajax (programming)’, *Wikipedia, the free encyclopedia*. Wikimedia Foundation, Inc., 10-Jun-2012.

[8] I. Fette and A. Melnikov, ‘The WebSocket Protocol’, Dec-2011. [Online]. Available: http://grenache.tools.ietf.org/html/rfc6455. [Accessed: 10-Jun-2012].

[9] D. Schiemann, ‘The Long-Polling Technique’, Nov-2007. [Online]. Available: http://cometdaily.com/2007/11/15/the-long-polling-technique/. [Accessed: 10-Jun-2012].

[10] G. Wilkins, ‘Comet is Always Better Than Polling’, Nov-2007. [Online]. Available: http://cometdaily.com/2007/11/06/comet-is-always-better-than-polling/. [Accessed: 10-Jun-2012].

[11] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro, *Emf: Eclipse Modeling Framework*. Pearson Education, 2008.

[12] C. Vicente, D. Alonso, and J. F. Ingles, *Introduccion Practica al Desarrollo de Software Dirigido por Modelos*. 2011.

[13] ‘Basic Auth Standar’. [Online]. Available: http://www.ietf.org/rfc/rfc2617.txt. [Accessed: 28-May-2012].

[14] Wikipedia contributors, ‘HTML5 Audio’, *Wikipedia, the free encyclopedia*. Wikimedia Foundation, Inc., 06-Jun-2012.